

Sommario

Prefazione

1 Introduzione	3
1.1 Scopi delle reti di calcolatori	3
1.1.1 Applicazioni aziendali	3
1.1.2 Applicazioni domestiche	6
1.1.3 Applicazioni mobili	9
1.1.4 Risvolti sociali	12
1.2 Hardware di rete	14
1.2.1 Reti locali	16
1.2.2 Metropolitan Area Network	18
1.2.3 Wide Area Network	19
1.2.4 Reti wireless	21
1.2.5 Reti domestiche	23
1.2.6 Reti tra sistemi	25
1.3 Software di rete	26
1.3.1 Gerarchie dei protocolli	26
1.3.2 Progettazione degli strati	30
1.3.3 Servizi orientati alla connessione e senza connessione	32
1.3.4 Primitive di servizio	34
1.3.5 La relazione tra servizi e protocolli	36

1.4 Modelli di riferimento.....	37	2.3.3 Trasmissione a microonde.....	104
1.4.1 Il modello di riferimento OSI	37	2.3.4 Infrarossi e onde millimetriche	107
1.4.2 Il modello di riferimento TCP/IP.....	41	2.3.5 Trasmissione a onde luminose	107
1.4.3 Confronto tra i modelli di riferimento OSI e TCP/IP	44		
1.4.4 Critica del modello e dei protocolli OSI	46		
1.4.5 Critica del modello di riferimento TCP/IP.....	48		
1.5 Esempi di reti	49	2.4 Comunicazioni satellitari.....	109
1.5.1 Internet.....	50	2.4.1 Satelliti geostazionari	109
1.5.2 Reti orientate alla connessione: X.25, frame relay e ATM.....	59	2.4.2 Satelliti su orbite medie.....	113
1.5.3 Ethernet	65	2.4.3 Satelliti su orbite basse.....	114
1.5.4 LAN senza fili: 802.11	68	2.4.4 Satelliti o fibra?	117
1.6 Standardizzazione delle reti	71	2.5 La rete telefonica pubblica commutata	118
1.6.1 Il who's who del mondo delle telecomunicazioni	71	2.5.1 Struttura del sistema telefonico.....	119
1.6.2 Il who's who del mondo degli standard internazionali	74	2.5.2 Le politiche dei telefoni	122
1.6.3 Il who's who del mondo degli standard Internet	75	2.5.3 I collegamenti locali: modem, ADSL e connessioni wireless	124
1.7 Unità metriche	77	2.5.4 Linee e multiplexing	137
1.8 Organizzazione del libro.....	78	2.5.5 Commutazione	146
1.9 Sommario.....	80	2.6 Il sistema telefonico mobile	152
2 Lo strato fisico	85	2.6.1 Cellulari della prima generazione: voce analogica	153
2.1 Le basi teoriche della comunicazione dati.....	85	2.6.2 Cellulari della seconda generazione: voce digitale	157
2.1.1 Analisi di Fourier.....	86	2.6.3 Cellulari della terza generazione: voce e dati digitali.....	166
2.1.2 Segnali a banda limitata	86		
2.1.3 La velocità massima di un canale	89	2.7 Televisione via cavo	169
2.2 Mezzi di trasmissione guidati	90	2.7.1 Televisione ad antenna collettiva.....	169
2.2.1 Mezzi magnetici	90	2.7.2 Internet via cavo	170
2.2.2 Il doppino	91	2.7.3 Allocazione dello spettro.....	172
2.2.3 Cavo coassiale	92	2.7.4 Cable modem.....	173
2.2.4 Fibra ottica.....	93	2.7.5 ADSL o connessione via cavo?	175
2.3 Trasmissioni Wireless.....	100	2.8 Sommario	177
2.3.1 Lo spettro elettromagnetico.....	100		
2.3.2 Trasmissioni radio	103		
3 Lo strato data link	183		
3.1 Progetto dello strato data link.....	184		
3.1.1 Servizi forniti allo strato network	184		
3.1.2 Suddivisione in frame.....	187		
3.1.3 Controllo degli errori.....	191		
3.1.4 Controllo di flusso.....	192		

3.2 Rilevazione e correzione degli errori.....	192	4.3.3 Il protocollo del sottostrato MAC Ethernet.....	275
3.2.1 Codici per la correzione degli errori.....	193	4.3.4 L'algoritmo di backoff esponenziale binario	278
3.2.2 Codifiche a rilevazione d'errore.....	196	4.3.5 Prestazioni di Ethernet	279
3.3 Protocolli data link elementari.....	200	4.3.6 Ethernet commutata.....	281
3.3.1 Un protocollo simplex senza restrizioni.....	204	4.3.7 Fast Ethernet.....	283
3.3.2 Un protocollo simplex stop-and-wait.....	206	4.3.8 Gigabit Ethernet	286
3.3.3 Un protocollo simplex per canali rumorosi	208	4.3.9 IEEE 802.2: LLC (Logical Link Control)	290
3.4 Protocolli sliding window.....	211	4.3.10 Retrospettiva su Ethernet.....	291
3.4.1 Un protocollo sliding window a 1 bit	214	4.4 LAN wireless.....	292
3.4.2 Un protocollo che usa go back n.....	216	4.4.1 La pila di protocolli 802.11.....	292
3.4.3 Un protocollo che usa la ripetizione selettiva	223	4.4.2 Lo strato fisico di 802.11	293
3.5 Verifica dei protocolli	229	4.4.3 Il protocollo del sottostrato MAC di 802.11	295
3.5.1 Modelli a stati finiti.....	229	4.4.4 La struttura del frame di 802.11.....	299
3.5.2 Modelli a rete di Petri	232	4.4.5 Servizi.....	301
3.6 Esempi di protocolli data link.....	234	4.5 Wireless a banda larga	302
3.6.1 HDLC (High-level Data Link Control).....	234	4.5.1 Confronto tra 802.11 e 802.16	303
3.6.2 Lo strato data link in Internet.....	237	4.5.2 La pila di protocolli 802.16.....	305
3.7 Sommario.....	242	4.5.3 Lo strato fisico di 802.16	306
4 Il sottostrato MAC (<i>Medium Access Control</i>).....	247	4.5.4 Il protocollo del sottostrato MAC di 802.16	307
4.1 Il problema dell'assegnazione del canale	248	4.5.5 La struttura del frame 802.16.....	309
4.1.1 Assegnazione statica del canale in LAN e MAN.....	248	4.6 Bluetooth.....	310
4.1.2 Assegnazione dinamica del canale in LAN e MAN	249	4.6.1 Architettura Bluetooth	311
4.2 Protocolli ad accesso multiplo.....	251	4.6.2 Applicazioni Bluetooth.....	312
4.2.1 ALOHA	251	4.6.3 La pila di protocolli Bluetooth	313
4.2.2 Protocolli ad accesso multiplo con rilevamento della portante	255	4.6.4 Lo strato radio di Bluetooth	314
4.2.3 Protocolli senza collisione.....	259	4.6.5 Lo strato baseband di Bluetooth	315
4.2.4 Protocolli a contesa limitata	261	4.6.6 Lo strato L2CAP di Bluetooth	316
4.2.5 Protocolli WDMA (<i>Wavelength Division Multiple Access</i>).....	265	4.6.7 La struttura del frame di Bluetooth	316
4.2.6 Protocolli LAN wireless.....	267	4.7 Compattezza nello strato data link	317
4.3 Ethernet	271	4.7.1 Bridge tra 802.x e 802.y	319
4.3.1 Cablaggio Ethernet	271	4.7.2 Internetworking locale	322
4.3.2 Codifica Manchester	274	4.7.3 Bridge Spanning Tree	323
		4.7.4 Bridge remoti	325
		4.7.5 Ripetitori, hub, bridge, switch, router e gateway	326
		4.7.6 LAN virtuali	328
		4.8 Sommario	336

5 Lo strato network.....	343
5.1 Problemi dell'architettura dello strato network	343
5.1.1 Compattezza di pacchetto store-and-forward.....	344
5.1.2 Servizi forniti allo strato trasporto	344
5.1.3 Implementazione del servizio senza connessione	345
5.1.4 Implementazione del servizio orientato alla connessione.....	347
5.1.5 Confronto tra sottoreti a circuito virtuale e a datagramma.....	348
5.2 Algoritmi di routing.....	350
5.2.1 Il principio di ottimalità	352
5.2.2 Routing basato sul percorso più breve	353
5.2.3 Flooding	355
5.2.4 Routing basato sul vettore delle distanze.....	357
5.2.5 Routing basato sullo stato dei collegamenti.....	360
5.2.6 Routing gerarchico	366
5.2.7 Routing broadcast.....	368
5.2.8 Routing multicast	370
5.2.9 Routing per host mobili.....	372
5.2.10 Routing nelle reti ad hoc	375
5.2.11 Ricerca del nodo nelle reti peer-to-peer.....	380
5.3 Algoritmi per il controllo della congestione.....	384
5.3.1 Principi generali del controllo della congestione	386
5.3.2 Criteri per prevenire la congestione	388
5.3.3 Controllo della congestione nelle sottoreti a circuito virtuale	389
5.3.4 Controllo della congestione nelle sottoreti a datagrammi	391
5.3.5 Load shedding	394
5.3.6 Controllo del jitter	395
5.4 Qualità del servizio.....	397
5.4.1 Requisiti	397
5.4.2 Tecniche per ottenere una buona qualità di servizio.....	398
5.4.3 Servizi integrati	409
5.4.4 Servizi differenziati	412
5.4.5 Label switching e MPLS.....	415
5.5 Collegamento tra reti.....	418
5.5.1 Differenze tra le reti	419
5.5.2 Connessione tra le reti.....	420
5.5.3 Circuiti virtuali concatenati	422
5.5.4 Collegamento tra reti senza connessione	423
5.5.5 Tunneling.....	425
5.5.6 Routing in una internetwork	426
5.5.7 Frammentazione	427
5.6 Lo strato network in Internet	431
5.6.1 Il protocollo IP	432
5.6.2 Indirizzi IP	436
5.6.3 Protocolli di controllo Internet	449
5.6.4 OSPF – Il protocollo di routing per i gateway interni	454
5.6.5 BGP – Il protocollo di routing per i gateway esterni.....	459
5.6.6 Internet multicasting.....	461
5.6.7 Mobile IP.....	462
5.6.8 IPv6	464
5.7 Sommario.....	473
6 Lo strato trasporto.....	481
6.1 Il servizio di trasporto	481
6.1.1 I servizi offerti agli strati superiori	481
6.1.2 Le primitive del servizio di trasporto.....	483
6.1.3 I socket Berkeley	487
6.1.4 Un esempio di programmazione con socket: un file server Internet	488
6.2 Gli elementi dei protocolli di trasporto	492
6.2.1 L'indirizzamento.....	493
6.2.2 Stabilire la connessione.....	496
6.2.3 Il rilascio della connessione	502
6.2.4 Il controllo di flusso e il buffering	506
6.2.5 Il multiplexing	510
6.2.6 Il ripristino dopo un crash	511
6.3 Un semplice protocollo di trasporto	513
6.3.1 Esempio di primitive di servizio	513
6.3.2 Esempio di entità di trasporto	515
6.3.3 L'esempio come macchina a stati finiti.....	522
6.4 I protocolli di trasporto Internet: UDP	524
6.4.1 Introduzione a UDP.....	525
6.4.2 La chiamata a procedure remote	526
6.4.3 Il protocollo di trasporto in tempo reale	529

6.5 Il protocollo di trasporto Internet: TCP.....	532
6.5.1 Introduzione a TCP	532
6.5.2 Il modello di servizio TCP.....	533
6.5.3 Il protocollo TCP.....	535
6.5.4 L'intestazione del segmento TCP.....	536
6.5.5 Costituzione della connessione TCP	539
6.5.6 Il rilascio della connessione TCP.....	541
6.5.7 Il modello di gestione della connessione TCP.....	541
6.5.8 Il criterio di trasmissione di TCP	543
6.5.9 Il controllo della congestione di TCP	547
6.5.10 La gestione dei timer TCP.....	550
6.5.11 UDP e TCP wireless.....	553
6.5.12 TCP transazionale.....	555
6.6 Problemi di prestazioni	557
6.6.1 I problemi di prestazioni nelle reti di computer	557
6.6.2 La misurazione delle prestazioni della rete.....	560
6.6.3 Progettazione del sistema per aumentare le prestazioni	562
6.6.4 Elaborazione rapida delle TPDU	566
6.6.5 I protocoli per le reti gigabit	569
6.7 Sommario.....	573
7 Lo strato applicazione	579
7.1 DNS: il sistema dei nomi di dominio	579
7.1.1 Lo spazio dei nomi DNS.....	580
7.1.2 I record delle risorse.....	582
7.1.3 I server dei nomi	586
7.2 La posta elettronica	588
7.2.1 L'architettura e i servizi	590
7.2.2 L'agente utente	591
7.2.3 I formati dei messaggi.....	594
7.2.4 Il trasferimento dei messaggi	602
7.2.5 La consegna finale.....	605
7.3 Il World Wide Web	611
7.3.1 Una panoramica dell'architettura.....	612
7.3.2 Documenti Web statici	629

7.3.3 I documenti Web dinamici	643
7.3.4 HTTP (<i>HyperText Transfer Protocol</i>)	651
7.3.5 Il miglioramento delle prestazioni	656
7.3.6 Il Web wireless	662
7.4 Multimedia.....	674
7.4.1 Introduzione all'audio digitale	674
7.4.2 La compressione audio.....	676
7.4.3 L'audio in streaming.....	679
7.4.4 Le radio Internet	683
7.4.5 Voice over IP	685
7.4.7 La compressione video.....	696
7.4.8 Il video on demand.....	704
7.4.9 MBone (<i>Multicast Backbone</i>)	711
7.5 Sommario.....	714
8 Sicurezza delle reti	721
8.1 Crittografia.....	724
8.1.1 Introduzione alla crittografia	725
8.1.2 Cifrari a sostituzione	727
8.1.3 Cifrari a trasposizione	729
8.1.4 Blocchi monouso	730
8.1.5 Due principi crittografici fondamentali	735
8.2 Algoritmi a chiave simmetrica.....	737
8.2.1 DES (<i>Data Encryption Standard</i>)	738
8.2.2 AES (<i>Advanced Encryption Standard</i>)	741
8.2.3 Modalità di cifratura	745
8.2.4 Altri cifrari	750
8.2.5 Criptoanalisi	750
8.3 Algoritmi a chiave pubblica	752
8.3.1 RSA	753
8.3.2 Altri algoritmi a chiave pubblica	755

8.4 Firme digitali	755
8.4.1 Firme a chiave simmetrica	756
8.4.2 Firme a chiave pubblica	757
8.4.3 Message digest	759
8.4.4 L'attacco del compleanno.....	763
8.5 Gestione delle chiavi pubbliche.....	765
8.5.1 Certificati.....	765
8.5.3 Infrastrutture a chiave pubblica.....	768
8.6 Sicurezza delle comunicazioni	772
8.6.1 IPsec	772
8.6.2 Firewall.....	776
8.6.3 VPN, reti private virtuali	779
8.6.4 Sicurezza wireless	780
8.7 Protocolli di autenticazione.....	785
8.7.1 Autenticazione basata su un segreto condiviso	786
8.7.2 Come stabilire una chiave condivisa: lo scambio di chiave di Diffie-Hellman.....	791
8.7.3 Autenticazione usando un centro di distribuzione delle chiavi.....	793
8.7.4 Autenticazione con Kerberos	796
8.7.5 Autenticazione con la crittografia a chiave pubblica	798
8.8 Sicurezza dell'e-mail.....	799
8.8.1 PGP (<i>Pretty Good Privacy</i>).....	799
8.8.2 PEM (<i>Privacy Enhanced Mail</i>).....	803
8.8.3 S/MIME.....	804
8.9 Sicurezza del Web.....	805
8.9.1 Minacce alla sicurezza	805
8.9.2 Sicurezza del naming	806
8.9.3 SSL (<i>Secure Socket Layer</i>).....	813
8.9.4 Sicurezza del codice mobile.....	816
8.10 Aspetti sociali.....	819
8.10.1 Privacy	819
8.10.2 Libertà di parola	822
8.10.3 Copyright.....	826
8.11 Sommario.....	828

9 Elenco di lettura e bibliografia	835
9.1 Suggerimenti per ulteriori letture	835
9.1.1 Introduzione e opere generiche	836
9.1.2 Lo strato fisico.....	838
9.1.3 Lo strato data link	840
9.1.4 Il sottostrato Medium Access Control.....	840
9.1.5 Lo strato network	842
9.1.6 Lo strato trasporto	844
9.1.7 Lo strato applicazione	844
9.1.8 La sicurezza delle reti	846
9.2 Bibliografia in ordine alfabetico.....	848
Indice	869

Prefazione

Questo libro è arrivato alla quarta edizione. Ciascuna edizione riflette una diversa fase dello sviluppo delle reti di calcolatori. Quando fu pubblicata la prima edizione, nel 1980, le reti erano una curiosità accademica. Alla seconda edizione, nel 1988, le reti venivano impiegate dalle università e dalle grandi aziende. All'apparire della terza edizione, nel 1996, le reti, e in particolare Internet, erano diventate ormai una realtà quotidiana per milioni di persone. La novità, alla pubblicazione di questa quarta edizione, è rappresentata dalla rapida crescita delle reti wireless, comprese 802.11, le reti cellulari 2G e 3G, Bluetooth, WAP, i-mode e altre. Di conseguenza, è stato aggiunto molto materiale sulle reti wireless. Un altro argomento di importanza sempre crescente è la sicurezza, per la quale è stato aggiunto un capitolo apposito. Sebbene il Capitolo 1 abbia la medesima funzione introduttiva che già aveva nella terza edizione, il suo contenuto è stato ampiamente rivisto e aggiornato. Per esempio, in questa edizione vengono presentati Internet, Ethernet e LAN wireless, insieme a qualche cenno storico. Il Capitolo 2 è stato non poco riorganizzato. Dopo una breve presentazione dei principi della comunicazione di dati, il capitolo è formato da tre grandi sezioni riguardanti le transmissioni (mezzi con guida, wireless e satellite), seguite da tre ulteriori sezioni che riportano alcuni importanti esempi (la rete telefonica pubblica, il sistema di telefonia mobile e la televisione via cavo). Fra i nuovi argomenti trattati in questo capitolo ci sono ADSL, wireless a banda larga, MAN wireless e accesso a Internet via cavo e via DOCSIS. Il Capitolo 3 è stato sempre dedicato, anche nelle edizioni precedenti, ai protocolli point-to-point che sono rimasti per la maggior parte invariati rispetto alla terza edizione. Al contrario, il sottostrato MAC è stato sottoposto a grandi cambiamenti negli anni recenti, il che ha comportato molte modifiche al Capitolo 4. Il paragrafo riguardante Ethernet è stato aggiornato in modo che comprendesse anche gigabit Ethernet. Ci sono altresì paragrafi completamente nuovi riguardanti le LAN wireless, il wireless broadband, Bluetooth e lo switching dello strato data link, compreso MPLS.

Anche il Capitolo 5 è stato aggiornato, eliminando tutto il materiale su ATM e aggiungendo materiale su Internet. La QOS è ora uno degli argomenti principali. Nel capitolo sono presenti anche le reti wireless, con una trattazione sul routing in reti ad hoc. Altri argomenti comprendono il NAT e le reti peer-to-peer. Il Capitolo 6 riguarda ancora lo strato trasporto, ma con alcune modifiche, fra le quali un esempio di programmazione di socket: vengono presentati un client e un server di una pagina, scritti in C. I due programmi, disponibili sul sito del libro (<http://www.prenhall.com/tanenbaum>), possono essere compilati ed eseguiti; insieme realizzano un server Web o un file server primitivo, adattissimo per svolgere sperimentazioni. Altri argomenti comprendono le chiamate di procedure remote, RTP e transaction/TCP. Il Capitolo 7 riguarda lo strato applicazione ed è maggiormente mirato, appunto, allo strato applicazione del modello OSI. Dopo una breve introduzione al DNS, il resto del capitolo riguarda solo tre argomenti: la posta elettronica, il Web e il multimedia. La trattazione sul funzionamento del Web, per esempio, che comprende una sessantina di pagine, prende in esame una vasta gamma di argomenti tra cui le pagine Web statiche e dinamiche, HTTP e gli script CGI. Il capitolo contiene anche materiale su XML, XSL, XHTML, PHP e altro ancora. Si tratta anche di Web wireless, con particolare riguardo a i-mode e WAP. Il materiale sul multimedia riguarda gli MP3, lo streaming audio e voice over IP.

La sicurezza è diventata un argomento così importante da doverle dedicare un intero capitolo di più di cento pagine, che va dai principi della sicurezza (algoritmi simmetrici e a chiave pubblica, firme digitali e certificati X.509) all'applicazione di questi principi (autenticazione, sicurezza della posta elettronica e del Web). Il capitolo è nello stesso tempo molto ampio (andando dalla crittografia quantizzata alla censura statale) e molto approfondito (per esempio, si analizza in profondità il funzionamento dell'algoritmo SHA-1).

Il Capitolo 9 contiene un elenco di letture consigliate e una bibliografia molto ampia di oltre 350 riferimenti alla letteratura contemporanea. Oltre 200 di queste riguardano riviste e libri scritti dal 2000 in poi.

I libri di informatica sono pieni di acronimi e questo non fa eccezione. Ma ciascuno di essi verrà accuratamente descritto nel corso della trattazione.

- un manuale per la soluzione dei problemi
- file contenenti le figure in diversi formati
- diapositive in PowerPoint per un corso che utilizzi il libro
- un simulatore (scritto in C) per i protocolli di esempio del Capitolo 3
- una pagina Web con collegamenti a molti tutorial, organizzazioni, FAQ, eccetera.

Il manuale delle soluzioni è disponibile, in lingua inglese, direttamente presso Prentice Hall (ma solo per gli insegnanti, non per gli studenti). Tutto il resto del materiale è disponibile all'indirizzo: <http://www.prenhall.com/tanenbaum>. Basta fare clic sulla copertina del libro.

Molte persone mi hanno aiutato durante la scrittura della quarta edizione. Vorrei ringraziare in particolare: Ross Anderson, Elizabeth Belding-Royer, Steve Bellovin, Chatschik Bisdkian, Kees Bot, Scott Bradner, Jennifer Bray, Pat Cain, Ed Felten, Warwick Ford, Kevin Fu, Ron Fulle, Jim Geier, Mario Gerla, Natalie Giroux, Steve Hanna, Jeff Hayes, Amir Herzberg, Philip Homburg, Philipp Hoschka, David Green, Bart Jacobs, Frans Kaashoek, Steve Kent, Roger Kermode, Robert Kinicki, Shay Kutten, Rob Lanphier, Marcus Leech, Tom Maufer, Brent Miller, Shivakant Mishra, Thomas Nadeau, Shlomo Ovadia, Kaveh Pahlavan, Radia Perlman, Guillaume Pierre, Wayne Pleasant, Patrick Powell, Thomas Robertazzi, Medy Sanadidi, Christian Schmutzler, Henning Schulzrinne, Paul Sevinc, Mihail Sichitiu, Bernard Sklar, Ed Skoudis, Bob Strader, George Swallow, George Thiruvathukal, Peter Tomsu, Patrick Verkaik, Dave Vittali, Spyros Voulgaris, Jan-Mark Wams, Ruediger Weis, Bert Wijnen, Joseph Wilkes, Leendert van Doorn e Maarten van Steen.

Un ringraziamento particolare a Trudy Levine per aver dimostrato che le nonne possono fare le revisioni tecniche. Shivakant Mishra ha ideato numerosi problemi di fine capitolo. Andy Dorman ha suggerito ulteriori letture per il Capitolo 9. Jan Looyen ha fornito hardware essenziale in un momento critico. Il Dr. F. de Nies ha svolto un lavoro di taglia e incolla di livello professionale quando ce ne è stato bisogno. Il mio editor presso Prentice Hall, Mary Franz, mi ha dato molto più materiale di lettura di quanto ne ho potuto consumare nei precedenti sette anni ed è stata di grande aiuto in molti altri modi.

Infine, ecco le persone più importanti di tutti: Suzanne, Barbara e Marvin. Suzanne, per il suo amore, la sua pazienza e le sue colazioni al sacco. Barbara e Marvin per essere stati sempre allegri, tranne quando si lamentavano di quanto fossero noiosi i libri universitari, tenendomi con i piedi per terra. Grazie.

ANDREW S. TANENBAUM

Ringraziamenti per l'edizione italiana

Giunto alla sua seconda ristampa italiana, il testo di A. S. Tanenbaum sulle *Reti di calcolatori*, è stato riveduto e corretto.

Nel corso dell'anno scorso e di quest'anno, abbiamo ricevuto segnalazione di inesattezze o refusi da parte di numerosi nostri lettori che hanno contribuito a migliorare la qualità e l'accuratezza della traduzione italiana.

In particolare, l'editore ringrazia **Carlo Piano** per la pazienza e la sollecitudine con cui ha contribuito a migliorare la leggibilità di alcuni capitoli e **Marco Cesati** dell'Università di Roma Tor Vergata per la mole di commenti e correzioni che ci ha inviato e che sono stati inseriti nel corso della trattazione come NdR.

Milano, settembre 2004

1

Introduzione

Ciascuno dei tre secoli trascorsi è stato dominato da una specifica tecnologia. Il diciottesimo secolo ha rappresentato l'era dei grandi sistemi meccanici che hanno accompagnato la Rivoluzione Industriale. Il diciannovesimo secolo è stato l'era del motore a vapore. Nel corso del ventesimo secolo, la tecnologia chiave è stata la raccolta, elaborazione e distribuzione dell'informazione. Alcuni dei principali sviluppi che abbiamo ammirato sono la costruzione di una rete telefonica mondiale, l'invenzione della radio e della televisione, la nascita e la crescita senza precedenti dell'industria del computer e il lancio dei satelliti per telecomunicazioni.

Come risultato del rapido processo tecnologico, queste aree stanno convergendo a grande velocità. Le differenze tra raccolta, trasporto, archiviazione ed elaborazione dell'informazione svaniscono rapidamente. Organizzazioni con centinaia di uffici sparpagliati su una vasta area geografica esigono la capacità di richiamare immediatamente, premendo un solo tasto, lo stato corrente anche del loro più remoto ufficio. Mentre la nostra capacità di raccogliere, elaborare e distribuire l'informazione cresce, la richiesta per elaborazioni maggiormente sofisticate aumenta ancora più velocemente.

Benché l'industria del computer sia ancora giovane a confronto di altre, per esempio quella automobilistica e dei trasporti aerei, i computer hanno fatto progressi spettacolari in poco tempo. Nei primi due decenni della loro esistenza erano altamente centralizzati, di solito ospitati in una sola grande stanza. Spesso questa stanza aveva pareti di vetro, attraverso le quali i visitatori potevano ammirare la grande meraviglia elettronica. Un'azienda

di medie dimensioni o un'università poteva avere uno o due computer, mentre le grandi istituzioni ne avevano al più una dozzina. L'idea che entro vent'anni sarebbero stati prodotti in milioni di pezzi computer di pari potenza, più piccoli di un francobollo, era vera fantascienza.

La convergenza di computer e comunicazioni ha avuto un'influenza profonda sul modo in cui sono strutturati i calcolatori. Il concetto di "centro di calcolo" come una stanza con un grande computer dove gli utenti portano il loro lavoro per l'elaborazione oggi è completamente superata. Il vecchio modello di un solo computer che soddisfa l'intera necessità di calcolo dell'organizzazione è stato sostituito da un altro, in cui il lavoro è svolto da un gran numero di computer distinti ma interconnessi. Questi sistemi sono chiamati **reti di calcolatori**. La progettazione e l'organizzazione di queste reti sono l'oggetto di questo libro. Nel testo useremo il termine "rete di calcolatori" per indicare un insieme di computer autonomi collegati da una singola tecnologia. Due computer si dicono interconnessi quando sono in grado di scambiare informazioni. Il collegamento non è necessariamente realizzato con cavi di rame; si possono usare anche fibre ottiche, microonde, luce infrarossa e satelliti per comunicazioni. Le reti hanno dimensioni, tipologia e forma differente, come vedremo in seguito. Anche se per alcuni può sembrare strano, né Internet né il World Wide Web sono reti di calcolatori. Alla fine di questo libro, il motivo apparirà evidente. La risposta rapida è questa: Internet non è una singola rete ma una rete di reti, e il Web è un sistema distribuito che si appoggia su Internet.

C'è molta confusione nella letteratura tra reti di calcolatori e un **sistema distribuito**. La distinzione fondamentale è che in un sistema distribuito l'insieme di computer indipendenti appare ai propri utenti come un singolo sistema coerente. Di solito ha un solo modello o paradigma che presenta agli utenti. Spesso la realizzazione di questo modello è affidata a uno strato di software posto sopra al sistema operativo, chiamato **middleware**. Un esempio ben noto di sistema distribuito è il **World Wide Web**, dove tutto ha l'aspetto di un documento (pagina Web).

In una rete di calcolatori mancano coerenza, modello e software. Gli utenti vedono le macchine effettive e il sistema non compie tentativi per far apparire e agire le macchine in un nodo coerente. Se le macchine hanno hardware e sistemi operativi differenti, ciò resta completamente visibile agli utenti. Se un utente vuole eseguire un programma su una macchina remota, deve collegarsi a quella macchina e avviarlo da lì.

In effetti un sistema distribuito è un software costruito sopra una rete. Il software dà al sistema un elevato grado di coesione e trasparenza. Di conseguenza, la distinzione tra una rete e un sistema distribuito sta nel software (in particolare nel sistema operativo) piuttosto che nell'hardware.

In ogni caso c'è una sovrapposizione considerevole tra i due argomenti. Per esempio, sia i sistemi distribuiti sia le reti di calcolatori hanno necessità di spostare file. La differenza sta in chi conduce questo movimento, il sistema o l'utente.

Benché questo libro si concentri principalmente sulle reti, molti argomenti sono importanti anche per i sistemi distribuiti. Per maggiori informazioni sui sistemi distribuiti, vedere *Distributed Systems: Principles and Paradigms* Tanenbaum and Van Steen, 2002.

1.1 Scopi delle reti di calcolatori

Prima di iniziare a esaminare nel dettaglio gli aspetti tecnici, vale la pena soffermarsi a considerare i motivi per cui le reti di calcolatori suscitano un così vivo interesse e gli scopi a cui possono essere destinate. Prenderemo in esame gli utilizzi tradizionali in ambito aziendale e personale e in seguito analizzeremo gli sviluppi più recenti, che riguardano gli utenti mobili e le reti domestiche.

1.1.1 Applicazioni aziendali

Molte aziende hanno una quantità considerevole di computer. Per esempio un'azienda può avere computer distinti per governare la produzione, tenere l'inventario dei magazzini e gestire le buste paga. Inizialmente ciascuno di questi computer poteva funzionare separatamente dagli altri, ma a un certo punto la direzione può aver deciso di collegarli tra loro, per raccogliere e correlare informazioni relative all'azienda nel suo complesso.

Generalizzando, il punto focale è in questo caso la **condivisione delle risorse** e l'obiettivo è quello di rendere disponibili a chiunque sulla rete tutti i programmi, le periferiche e soprattutto i dati, indipendentemente dalla posizione fisica dell'utente e della risorsa. Un esempio ovvio e largamente diffuso riguarda la necessità di condividere una stampante tra un gruppo di utenti in un ufficio. Nessuno individualmente ha la necessità effettiva di una stampante personale e una stampante di rete ad alta produttività è spesso più economica, più veloce e più semplice da gestire di un vasto insieme di stampanti personali.

Tuttavia, una risorsa condivisa più importante di quelle fisiche come le stampanti, gli scanner e i masterizzatori è l'informazione. Ogni azienda con dimensioni medie o grandi, e molte piccole industrie, dipendono in modo vitale dalle informazioni computerizzate. La maggioranza delle aziende ha in linea informazioni sulla clientela, inventari della merce, fatture, documenti contabili, moduli fiscali e molto altro ancora. Se tutti i computer si fermassero, una banca non potrebbe sopravvivere più di cinque minuti. Una fabbrica moderna, con una linea di montaggio controllata dai computer, resisterebbe ancora meno. Oggi anche una piccola agenzia viaggi o un ufficio legale di tre persone si affida pesantemente alle reti di calcolatori, per consentire agli impiegati l'accesso immediato a informazioni e documenti necessari all'attività.

Nelle aziende più piccole tutti i computer sono probabilmente in un solo ufficio o edificio, ma in quelle più grandi i computer e gli impiegati possono essere sparsi tra dozzine di uffici e impianti situati in molte nazioni.

Nonostante ciò, un addetto alle vendite a New York può occasionalmente aver bisogno di accedere al database dell'inventario delle merci a Singapore. In altre parole, il semplice fatto che un utente si trova a 15.000 km dai dati non deve impedirgli di usarli come se fossero locali. Questo obiettivo può essere riassunto dicendo che è un tentativo di terminare la "tirannia della geografia".

Nella sua forma più semplice, ci si può immaginare il sistema informatico di un'azienda come formato da uno o più database e da un certo numero di impiegati che hanno bisogno di accedervi remotamente. In questo modello, i dati sono memorizzati in computer ad alte prestazioni chiamati **server**. Frequentemente sono installati in un ufficio centrale e gestiti da un amministratore di sistema. Al contrario, gli impiegati hanno sulla scrivania macchine più semplici chiamate **client**, tramite le quali hanno accesso ai dati remoti, per esempio per incorporarli nei fogli di calcolo che stanno realizzando (a volte indicheremo col nome "client" l'utente umano del computer client, ma sarà chiaro dal contesto se ci riferiamo al computer o al rispettivo utente). Le macchine client e server sono collegate da una rete, come illustrato nella Figura 1.1. Osservare che la rete viene indicata come un semplice ovale, senza dettagli. Si utilizzerà questa forma per indicare una rete in modo astratto. Saranno forniti più dettagli quando sarà necessario.

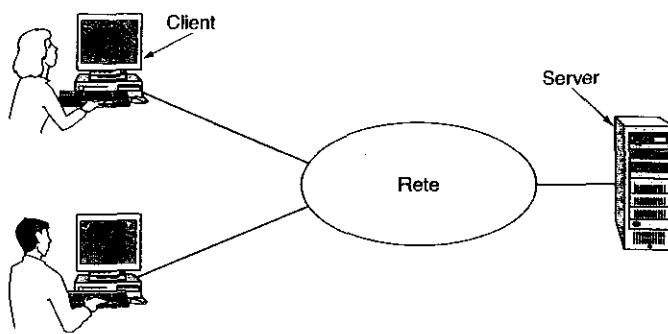


Figura 1.1. Una rete con due client e un server.

Questa configurazione è chiamata **modello client-server**. È ampiamente utilizzato e rappresenta la base di gran parte dell'utilizzo delle reti. È applicabile quando il client e il server sono nello stesso edificio (per esempio appartengono alla stessa azienda), ma anche quando sono molto distanti. Quando una persona accede da casa a una pagina sul Web si utilizza lo stesso modello, dove il server Web remoto svolge il ruolo di server e il personal computer dell'utente è il client. Nella maggior parte delle situazioni, un server può soddisfare un gran numero di client.

Se guardiamo in dettaglio al modello client-server, vediamo che sono coinvolti due processi, uno sulla macchina client e uno sulla macchina server. La comunicazione è rappresentata da un processo client che manda un messaggio attraverso la rete al processo server e resta in attesa di un messaggio di risposta. Quando il processo server riceve la richiesta,

esegue il lavoro o recupera i dati desiderati e manda indietro una risposta. Questi messaggi sono mostrati nella Figura 1.2.

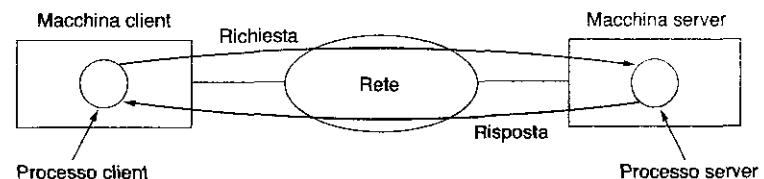


Figura 1.2. Il modello client-server comprende richieste e risposte.

Un secondo motivo per realizzare una rete di calcolatori ha a che fare con le persone, piuttosto che con l'informazione o i computer stessi. Una rete di calcolatori può offrire un potente **mezzo di comunicazione** tra gli impiegati. Praticamente tutte le aziende che hanno due o più computer oggi dispongono di **e-mail** (posta elettronica), che di solito gli impiegati utilizzano per una gran parte delle comunicazioni quotidiane. In effetti, un tipico argomento di discussione attorno alla macchina del caffè riguarda la quantità di e-mail con cui ciascuno ha a che fare, la maggior parte delle quali è inutile, poiché i capi hanno scoperto che con la pressione di un pulsante possono mandare lo stesso messaggio (spesso privo di contenuto) a tutti i subordinati.

Ma l'e-mail non è l'unica forma di comunicazione potenziata resa possibile dalle reti di calcolatori. Con una rete due persone che lavorano a grande distanza tra loro possono facilmente scrivere insieme un rapporto. Quando un utente applica una modifica a un documento online, l'altro può vedere immediatamente il cambiamento, invece di restare alcuni giorni in attesa di una lettera. Questa accelerazione rende semplice la cooperazione tra gruppi di persone distanti tra loro, mentre in precedenza sarebbe stata impossibile. Un'altra forma di comunicazione assistita dal computer è la videoconferenza. Utilizzando questa tecnologia, gli impiegati di sedi dislocate possono incontrarsi in riunione, vedendosi e ascoltandosi a vicenda, magari scrivendo su una lavagna virtuale condivisa. La videoconferenza è un potente strumento per eliminare i costi e il tempo una volta destinato ai viaggi. Spesso si dice che comunicazione e trasporti sono in gara tra loro e chi vincerà renderà l'altro superato.

Un terzo obiettivo per un numero crescente di aziende consiste nel realizzare affari in modo elettronico con altre aziende, specialmente clienti e fornitori. Per esempio, i costruttori di automobili, aerei e computer (e altri ancora) acquistano sottosistemi da molti fornitori e poi assemblano le parti. Utilizzando reti di calcolatori, i costruttori possono emettere elettronicamente gli ordini quando è necessario. Essere in grado di emettere ordini in tempo reale (cioè quando è richiesto) riduce la necessità di scorte di magazzino e migliora l'efficienza.

Un quarto scopo che sta iniziando a diventare importante è la conclusione di affari con i consumatori attraverso Internet. Linee aeree, librerie e vendori di musica hanno scoperto che molti clienti apprezzano la comodità di acquistare da casa. Di conseguenza, molte aziende offrono cataloghi dei loro beni e servizi on-line e prendono ordini on-line. Si ritiene che questo settore chiamato **e-commerce** (commercio elettronico) sia destinato a crescere velocemente in futuro.

1.1.2 Applicazioni domestiche

Nel 1977, Ken Olsen era presidente di Digital Equipment Corporation, che allora era il produttore di computer numero due al mondo (dopo IBM). Quando gli venne chiesto perché Digital non entrava con forza nel mercato dei personal computer, rispose: "Una persona non ha motivo di volere un computer a casa propria". La storia ha dimostrato il contrario, e oggi Digital non esiste più. Perché le persone acquistano computer per uso domestico? Inizialmente per scrivere testi e giocare, ma negli ultimi anni lo scenario è cambiato radicalmente. Probabilmente oggi il motivo determinante per l'acquisto è l'accesso a Internet. Alcuni degli utilizzi più popolari di Internet per gli utenti domestici sono i seguenti:

1. accesso a informazioni remote
2. comunicazione da persona a persona
3. intrattenimento interattivo
4. commercio elettronico.

L'accesso a informazioni remote avviene in diverse forme. Può consistere nel navigare sul Web alla ricerca di informazioni o per puro divertimento. Le informazioni disponibili includono arti, affari, cucina, politica, salute, storia, hobby, intrattenimento, scienza, sport, viaggi e molto altro. Ci si può divertire in troppi modi per elencarli tutti, anche in alcuni che è meglio non menzionare.

Molti quotidiani sono arrivati on-line e possono essere personalizzati. Per esempio a volte è possibile indicare a un quotidiano che si desidera ricevere tutte le informazioni che riguardano politici corrotti, grandi incendi, scandali che coinvolgono celebrità ed epidemie, ma non il calcio. A volte è addirittura possibile ricevere gli articoli desiderati sul proprio hard disk mentre si dorme, oppure stampati sulla propria stampante appena prima della colazione. Il proseguire di questa tendenza causerà disoccupazione massiccia tra i dodicenni che consegnano i giornali al mattino, ma i quotidiani la apprezzano poiché la distribuzione è sempre stato l'anello debole dell'intera catena produttiva.

Il passo successivo dopo i quotidiani (e le riviste periodiche o pubblicazioni scientifiche) è la biblioteca digitale on-line. Molte organizzazioni professionali come ACM (www.acm.org) ed IEEE Computer Society (www.computer.org) hanno già on-line molte pubblicazioni e atti di convegni. Altri gruppi stanno attrezzandosi rapidamente. A seconda del costo, dimensione e peso dei computer portatili, i libri stampati potrebbero diventare superati. Gli scettici dovrebbero considerare l'effetto che la stampa ha avuto sui manoscritti medievali.

Tutte queste applicazioni riguardano interazioni tra una persona e un database remoto pieno di informazioni. La seconda modalità di utilizzo delle reti è la comunicazione da persona a persona, fondamentalmente la risposta del ventunesimo secolo al telefono del diciannovesimo. La posta elettronica viene già usata da milioni di persone in tutto il mondo e il suo utilizzo cresce rapidamente. Già contiene normalmente audio e video, assieme a testo e immagini. Mentre la trasmissione degli odori prevediamo possa richiedere ancora tempo.

Ogni adolescente in gamba è affascinato dall'**instant messaging**. Questo servizio, derivato dal programma UNIX *talk* in uso circa dal 1970, permette a due persone di scambiarsi messaggi in tempo reale. Una versione multiutente di questa idea è la **chat room**, dove un gruppo di persone può scrivere messaggi che vedranno tutti gli altri.

I gruppi di discussione (*newsgroup*) a distribuzione mondiale con dibattiti su qualsiasi argomento immaginabile sono già un fatto scontato per un ristretto numero di persone, e questo fenomeno crescerà fino a comprendere l'intera popolazione. Queste discussioni, in cui una persona inserisce un messaggio e gli altri sottoscrittori del newsgroup possono leggerlo, coprono la gamma dallo scherzoso al serio. A differenza delle chat room, i newsgroup non sono in tempo reale e i messaggi vengono archiviati, così quando qualcuno rientra dalle ferie trova, in paziente attesa di essere letti, tutti i messaggi che sono stati inseriti nel frattempo.

Un altro tipo di comunicazione interpersonale è spesso indicato con il nome **peer-to-peer**, per distinguerlo dal modello client-server (*Parameswaran et al., 2001*). In questa forma di comunicazione, individui che formano gruppi dispersi possono comunicare con altri nel gruppo, come mostrato nella Figura 1.3. Ognuno può, in linea di principio, comunicare con una o più delle altre persone; non c'è una suddivisione rigida tra client e server.

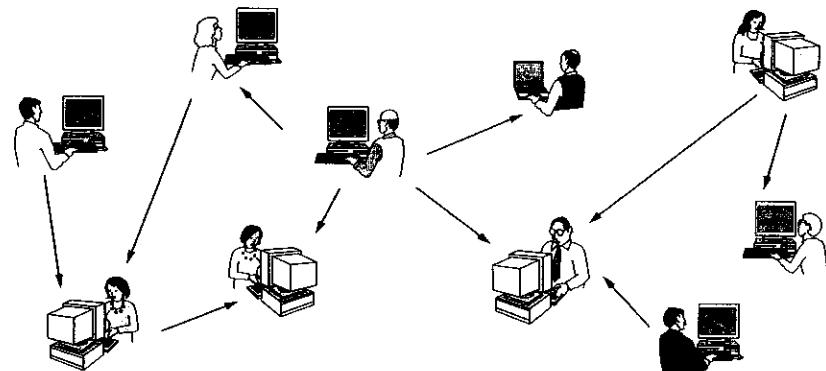


Figura 1.3. In un sistema peer-to-peer non ci sono client e server predefiniti.

La comunicazione peer-to-peer è diventata popolare attorno al 2000 con un servizio chiamato Napster, che al suo momento di massimo sviluppo contava più di 50 milioni di appassionati dediti allo scambio di musica; probabilmente fu la più grande violazione di copyright nella storia dell'industria discografica (*Lam and Tan, 2001; and Macedonia, 2000*). L'idea era molto semplice. I membri inviavano l'elenco della musica che avevano memorizzato sul proprio disco fisso a un database centrale ospitato dal server Napster. Se un membro desiderava una canzone, controllava il database per verificare chi ne aveva una copia, e vi si connetteva direttamente per prelevarla. Non memorizzando alcun brano musicale sulle proprie macchine, Napster sosteneva che non stava infrangendo alcun copyright. Il tribunale non fu d'accordo e chiuse il servizio.

Tuttavia la successiva generazione di sistemi peer-to-peer ha eliminato il database centrale, facendo gestire a ogni utente il proprio database locale, e rendendo disponibile un elenco di utenti vicini che sono membri del sistema. Un nuovo utente può quindi collegarsi a qualsiasi membro esistente per vedere che cosa offre, e ottenere un elenco di altri membri da controllare alla ricerca di altra musica e altri nomi. Questo processo di ricerca può essere ripetuto indefinitamente per costruire un grosso database locale di ciò che si trova sulla rete. Esistono usi legali della comunicazione peer-to-peer. Per esempio, appassionati che condividono musica di pubblico dominio o brani promozionali che nuovi gruppi rilasciano a scopo pubblicitario, famiglie che condividono fotografie, filmini e informazioni genealogiche, e ragazzi che giocano con giochi on-line multiutente. In effetti, una delle applicazioni più popolari di Internet, la posta elettronica, è intrinsecamente peer-to-peer. Per questa forma di comunicazione ci si attende in futuro una consistente crescita.

Il crimine informatico non si esaurisce con la violazione delle leggi sul copyright. Un'altra zona calda è il gioco d'azzardo elettronico. I computer hanno simulato oggetti per decenni. Perché non simulare slot machine, roulette, blackjack e altri attrezzi per scommesse? Semplicemente, perché in molti posti è illegale. Il problema è che l'azzardo è legale in molti altri luoghi (l'Inghilterra, per esempio) e i proprietari di casinò che vi si trovano hanno afferrato il potenziale del gioco d'azzardo tramite Internet. Che cosa succede se il giocatore e il casinò si trovano in paesi diversi, con leggi in conflitto tra loro? Buona domanda.

Altre applicazioni orientate alla comunicazione includono l'uso di Internet per veicolare chiamate telefoniche, videotelefonate e radio su Internet: tre aree in rapida crescita. Un'altra applicazione è l'insegnamento a distanza (*e-learning*), il che significa assistere alla lezione delle otto senza la scomodità di dover prima uscire dal letto. A lungo termine, l'uso delle reti per potenziare la comunicazione interpersonale può rivelarsi più importante di tutti gli altri.

La terza categoria è l'intrattenimento, che è un'industria enorme e in crescita. In questo caso la killer application (quella che trascina tutte le altre) è il video on demand. Tra circa dieci anni sarà possibile scegliere qualsiasi film o programma TV mai realizzato, in qualsiasi paese, e vederlo all'istante sullo schermo. I nuovi film potranno diventare interattivi, e cioè l'utente potrà occasionalmente ricevere richieste sulla direzione preferita della storia (Machbet deve uccidere Duncan o aspettare il momento buono?) con scenari alternativi per ogni direzione. Anche la televisione potrà diventare interattiva, con il pubblico che partecipa ai quiz assieme ai concorrenti e così via.

D'altro canto, forse la killer application non sarà il video on demand. Forse sarà il gioco on-line. Esistono già giochi di simulazione multiutente in tempo reale, come giochi di ruolo in un labirinto virtuale, e simulatori di volo con giocatori di una squadra che tentano di abbattere quelli della squadra avversaria. Se i giochi saranno giocati con occhiali e immagini in movimento tridimensionali, in tempo reale e di qualità fotografica, avremo una specie di realtà virtuale condivisa a livello mondiale.

La nostra quarta categoria è il commercio elettronico nel senso più ampio del termine. L'acquisto da casa è già popolare e permette agli utenti di esaminare i cataloghi on-line di centinaia di aziende. Alcuni di questi cataloghi offriranno presto la possibilità di ottenere immediatamente un filmato di qualsiasi prodotto semplicemente cliccando sul nome cor-

rispondente. Il cliente che non sa come usare un prodotto che ha acquistato elettronicamente potrà consultare il supporto tecnico on-line.

Un'altra area dove l'e-commerce sta già funzionando è l'accesso alle istituzioni finanziarie. Molte persone già pagano le bollette, gestiscono il conto corrente e seguono i propri investimenti on-line. Ciò acquisterà sicuramente più importanza con l'aumento di sicurezza delle reti.

Un'area che praticamente nessuno aveva previsto sono le vendite di seconda mano elettroniche. Le aste on-line di merce usata sono diventate un'industria significativa. A differenza dell'e-commerce tradizionale, che segue il modello client-server, le aste on-line sono più simili a un sistema peer-to-peer, una specie di consumer-to-consumer. Alcune di queste forme di e-commerce hanno ottenuto sigle simpatiche basate sul fatto che in inglese "to" e "2" si pronunciano allo stesso modo. Le più popolari sono elencate nella Figura 1.4.

Sigla	Sigla per esteso	Esempio
B2C	Business-to-consumer	Ordinare libri on-line
B2B	Business-to-business	Il costruttore di automobili ordina pneumatici dal fornitore
G2C	Government-to-consumer	Lo stato raccoglie la dichiarazione dei redditi elettronicamente
C2C	Consumer-to-consumer	Acquistare on-line prodotti usati in un'asta
P2P	Peer-to-peer	Scambio file

Figura 1.4. Alcune forme di e-commerce.

Senza dubbio la gamma di utilizzi delle reti di calcolatori crescerà rapidamente in futuro, e probabilmente in modi che nessuno oggi riesce a immaginare. Dopotutto, quante persone nel 1990 avrebbero predetto che ragazzi impegnati a digitare brevi messaggi di testo, su telefoni cellulari mentre viaggiano sull'autobus, avrebbero rappresentato dieci anni dopo un'immensa fonte di guadagno per le compagnie telefoniche? Eppure gli SMS sono molto redditizi.

Le reti di calcolatori potranno diventare enormemente importanti per le persone che sono svantaggiate geograficamente, dando loro accesso agli stessi servizi di coloro che vivono al centro di una grande città. L'insegnamento a distanza potrà modificare radicalmente l'istruzione; le università potranno diventare nazionali o internazionali. La telemedicina sta iniziando a prendere piede solo ora (per esempio con il controllo remoto dei pazienti), ma potrebbe diventare molto più importante. Ma la killer application potrebbe essere qualcosa di banale, come utilizzare una webcam montata nel frigorifero per vedere se bisogna comprare il latte prima di rientrare a casa dal lavoro.

1.1.3 Applicazioni mobili

I computer portatili, come i notebook e i *Personal Digital Assistant* (PDA) sono uno dei segmenti in maggiore crescita dell'industria dei computer. Molti proprietari di PDA hanno computer da tavolo nei loro uffici e vogliono potervi accedere anche quando sono lontani da casa o in viaggio. Poiché in auto e in aereo è impossibile avere una connessione fissa,

c'è molto interesse per le reti senza filo (*wireless*). In questo paragrafo vedremo brevemente alcuni utilizzi delle reti wireless.

Perché tutti ne vorrebbero una? Una ragione frequente è l'ufficio portatile. Chi è in viaggio spesso vuole usare le proprie apparecchiature elettroniche portatili per mandare e ricevere telefonate, fax e posta elettronica, navigare il Web, accedere a file remoti e collegarsi a macchine lontane. E lo vuole fare da qualsiasi posizione in terra, mare o cielo. Per esempio, oggi alle manifestazioni che riguardano l'informatica spesso gli organizzatori allestiscono una rete wireless nella zona delle conferenze. Chiunque, con un computer portatile e un modem wireless, può semplicemente accendere il computer ed essere connesso a Internet come se il computer fosse collegato a una rete fissa. Allo stesso modo, alcune università hanno installato reti senza fili nel campus, perché gli studenti possano sedere sotto gli alberi e consultare il catalogo della biblioteca o leggere la propria e-mail. Le reti wireless sono molto utili alle flotte di camion, taxi, corrieri espresso e addetti alla manutenzione per rimanere in contatto con la base. Per esempio, in molte città i conducenti dei taxi sono imprenditori indipendenti e non impiegati di una società. In alcune di queste città i taxi hanno un display che può essere visto dal conducente. Quando un cliente chiama, un operatore di centrale inserisce i punti di prelievo e destinazione. Questa informazione viene mostrata sul display del conducente che emette un segnale sonoro. Il primo conducente che preme un pulsante sul display prende la chiamata.

Le reti wireless sono importanti anche per i militari. Se è necessario essere in grado di combattere una guerra in qualsiasi punto della terra con breve preavviso, probabilmente non è una buona idea fare affidamento sulla infrastruttura di rete fissa locale. È meglio portarsi la propria.

Benché le reti senza fili e le applicazioni mobili siano spesso correlate, non sono la stessa cosa, come mostra la Figura 1.5. Qui vediamo una distinzione tra wireless fisso e portatile. Anche i computer portatili sono a volte collegati in modo fisso. Per esempio, se un viaggiatore collega il computer portatile nella presa telefonica di una stanza d'albergo, ottiene mobilità senza usare una rete wireless.

Wireless	Portatili	Applicazioni
No	No	Computer da tavolo negli uffici
No	Si	Computer portatile usato in una stanza d'albergo
Si	No	Reti negli edifici vecchi, non cablati
Si	Si	Ufficio portatile; PDA per l'inventario di magazzino

Figura 1.5. Combinazioni di reti wireless e applicazioni portatili.

D'altro canto, alcuni computer senza fili non sono portatili. Un esempio significativo è un'azienda che possiede un vecchio edificio senza cablaggio di rete, e che vuole collegare i propri computer. Installare una rete locale può richiedere solo l'acquisto di una scatola che contiene qualche apparecchio elettronico: basta aprirla e accenderle. Questa soluzione potrebbe essere molto più economica rispetto ad avere installatori che posano condotti per cablare l'edificio.

Ovviamente esistono anche le vere applicazioni mobili senza fili, che vanno dall'ufficio portatile alle persone che percorrono un negozio con un PDA per fare l'inventario. In molti aeroporti ad alto traffico, gli addetti al ritiro delle auto a noleggio lavorano nei parcheggi con un computer portatile senza fili. Battono il numero di targa dell'auto restituita e il computer portatile, che ha una stampante incorporata, chiama il computer principale, riceve le informazioni sul noleggio e stampa al volo la ricevuta.

Man mano che la tecnologia wireless si diffonde, è probabile l'emergere di molte altre applicazioni. Diamo un rapido sguardo alle possibilità. Parchimetri senza fili offrono vantaggi sia agli utenti finali sia all'amministrazione pubblica. Il parchimetro può accettare carte di credito o di debito con verifica istantanea tramite la connessione senza fili. Quando il tempo scade, può controllare la presenza dell'auto (invianole un segnale) e segnalare l'evento ai vigili. È stato stimato che le amministrazioni pubbliche nei soli Stati Uniti d'America potrebbero raccogliere in questo modo altri 10 miliardi di dollari (Harte et al., 2000). Inoltre un miglior controllo sui parcheggi aiuterebbe l'ambiente, poiché i conducenti che sono certi di essere sanzionati in caso di parcheggio illegale potrebbero scegliere di usare i mezzi pubblici.

Distributori automatici di cibo, bevande e altri generi sono ovunque. Tuttavia il cibo non arriva alle macchine con la magia. Periodicamente qualcuno arriva con un furgone per riempirle. Se il distributore mandasse un rapporto wireless una volta al giorno, segnalando l'inventario in tempo reale, il conducente del furgone saprebbe quali macchine hanno necessità d'intervento e quanto e quale prodotto trasportare. Questa informazione può garantire una pianificazione migliore del percorso. Naturalmente l'informazione potrebbe essere mandata su una normale linea telefonica, ma portare a ogni distributore una linea del telefono fissa per una sola chiamata al giorno sarebbe troppo costoso considerando il costo fisso mensile.

Un'altra area dove la tecnologia wireless può far risparmiare soldi è la lettura dei contatori. Se i contatori di acqua, gas, elettricità e altro installati nelle abitazioni potessero mandare un rapporto usando una rete senza fili, non ci sarebbe bisogno di inviare personale per la lettura. Allo stesso modo, rivelatori di fumo senza fili potrebbero chiamare i vigili del fuoco invece di fare un gran rumore (che ha poca utilità se nessuno è in casa). Man mano che il costo dei dispositivi radio e dell'utilizzo dell'infrastruttura wireless cala, sempre più misure e rapporti saranno affidati alle reti senza filo.

Un'area applicativa completamente diversa per le reti senza fili è l'attesa fusione tra i telefoni cellulari e i PDA per formare piccoli computer wireless. Un primo tentativo sono stati i piccoli PDA wireless che potevano mostrare versioni ridotte di pagine Web sui loro schermi ancora più piccoli. Questo sistema, chiamato **WAP 1.0** (*Wireless Application Protocol*), è fallito soprattutto a causa degli schermi microscopici, banda stretta e servizi scadenti. Ma nuovi dispositivi e servizi saranno migliori con WAP 2.0.

Un'area dove questi dispositivi potrebbero eccellere è chiamata **m-commerce** (*mobile commerce*) (Senn, 2000). La forza propulsiva dietro questo fenomeno è un'amalgama di produttori di PDA wireless e operatori di rete che tentano disperatamente di capire come tagliarsi una fetta della torta dell'm-commerce. Una delle loro speranze è l'uso dei PDA wireless per operazioni di banca e acquisti. Un'idea è quella di usare i PDA wireless come una specie di portafoglio elettronico, che autorizza i pagamenti nei negozi in sostituzione

del denaro contante e delle carte di credito. L'addebito appare sulla bolletta del telefono cellulare. Dal punto di vista del negozio questo schema può eliminare la maggior parte della commissione dovuta alla società che gestisce le carte di credito, che può essere di alcuni punti percentuali. Naturalmente questo progetto può ritorcersi contro i promotori, poiché i clienti del negozio potrebbero usare i loro PDA per controllare i prezzi dei correnti prima dell'acquisto. Ancora peggio, le società telefoniche potrebbero offrire PDA con lettori di codici a barre che permettono all'utente di fare la scansione di un prodotto in negozio e ottenere automaticamente un rapporto dettagliato su dove può essere acquistato alternativamente e a quale prezzo.

Poiché gli operatori di rete sanno dove si trova l'utente, alcuni servizi sono deliberatamente georeferenziati. Per esempio, può essere possibile chiedere dove si trova la libreria più vicina o il ristorante cinese. Le mappe mobili sono un altro candidato, come le previsioni veramente locali ("Quando smetterà di piovere nel mio giardino?"). Senza dubbio appariranno nuove applicazioni quando questi dispositivi saranno più diffusi.

Un'enorme punto a favore dell'm-commerce è che gli utenti mobili sono abituati a pagare per qualsiasi cosa (a differenza degli utenti Internet, che si aspettano che tutto sia gratis). Se un sito Web Internet chiedesse un sovrapprezzo per consentire ai suoi clienti il pagamento tramite carta di credito, gli utenti si lamenterebbero. Se un operatore di telefonia mobile permettesse alla gente di pagare la merce in un negozio usando il telefono e per questo servizio chiedesse un compenso, probabilmente ciò sarebbe percepito come un fatto normale.

Un po' più avanti nel futuro troviamo le personal area network e i computer da indossare (*wearable computer*). IBM ha progettato un orologio da polso che esegue Linux (incluso il sistema a finestre X11) e ha una connettività wireless con Internet per mandare e ricevere e-mail (Narayanaswami et al., 2002). In futuro, la gente potrebbe scambiarsi biglietti da visita semplicemente mostrandosi a vicenda gli orologi. I computer senza fili da indossare potranno dare accesso a locali protetti, nello stesso modo in cui oggi lo fanno le tessere magnetiche, magari in combinazione con un codice da battere o una misura biometrica. Questi orologi potrebbero anche recuperare informazioni relative alla posizione attuale dell'utente (per esempio i ristoranti vicini). Le possibilità sono infinite.

Orologi intelligenti con funzioni radio fanno parte del nostro spazio mentale sin dall'epoca in cui apparvero nei fumetti di Dick Tracy nel 1946. Ma la polvere intelligente? I ricercatori di Berkeley hanno compresso un computer wireless in un cubo di 1 mm di lato (Warneke et al., 2001). Le applicazioni potenziali includono il tracciamento delle scorte di magazzino, delle spedizioni e di piccoli uccelli, roditori e insetti.

1.1.4 Risvolti sociali

L'introduzione in massa delle reti ha fatto nascere nuovi problemi sociali, etici e politici. Menzioniamo brevemente alcuni di essi; uno studio accurato richiederebbe almeno un intero libro. Una funzionalità popolare di molte reti sono i newsgroup o forum di discussione, dove la gente può scambiare messaggi con chi ha interessi simili. Finché gli argomenti sono confinati ad aspetti tecnici o hobby come il giardinaggio, non nascono troppi problemi.

I guai nascono quando i newsgroup riguardano argomenti a cui la gente dà realmente importanza, come la politica, la religione e il sesso. Opinioni espresse su questi gruppi possono essere profondamente offensive per alcuni. Ancor peggio, potrebbero non essere politicamente corrette, e inoltre i messaggi non sono necessariamente limitati al solo testo. Oggi si possono trasmettere facilmente sulle reti le immagini a colori ad alta risoluzione e brevi spezzoni video. Alcune persone hanno una mentalità "vivi e lascia vivere", ma altre ritengono che la divulgazione di certo materiale (per esempio attacchi a particolari nazioni o religioni, pornografia, ecc.) sia semplicemente inaccettabile e vada censurata. Nazioni diverse hanno su questa materia legislazioni dissimili o in conflitto tra loro, per questo motivo infuria il dibattito.

Alcuni hanno citato in giudizio gli operatori di rete, sostenendo che sono responsabili del contenuto di ciò che trasportano, così come lo sono quotidiani e riviste. La risposta inevitabile è che la rete è come una compagnia telefonica o l'ufficio postale, e non si può pretendere che controlli ciò che dicono i suoi utenti. Ancor peggio, se gli operatori di rete dovessero censurare i messaggi, cancellerebbero tutto ciò che implica anche la più piccola possibilità di essere denunciati, e ciò violerebbe il diritto alla libera opinione degli utenti. Probabilmente questo dibattito proseguirà a lungo.

Un'altra area divertente è il confronto tra i diritti degli impiegati e dei datori di lavoro. Molte persone leggono e scrivono e-mail al lavoro. Molti datori di lavoro hanno sostenuto il diritto di leggere e magari censurare i messaggi degli impiegati, inclusi quelli mandati dal computer domestico dopo il lavoro. Non tutti gli impiegati sono d'accordo.

Se i datori di lavoro hanno potere sugli impiegati, questo rapporto si estende a università e studenti? E a scuole superiori e studenti? Nel 1994, la Carnegie-Mellon University ha deciso di troncare il flusso di messaggi in arrivo per diversi newsgroup che riguardano il sesso, perché l'università riteneva che il materiale fosse inappropriato per i minori (cioè gli studenti con meno di 18 anni). Le ricadute di questo evento hanno impiegato anni per quietarsi.

Un altro argomento chiave riguarda i rapporti tra governo e cittadino. L'FBI ha installato un sistema presso molti Internet service provider per spiare tutta la posta elettronica in ingresso e uscita, alla ricerca di parole chiave d'interesse (Blaze and Bellovin, 2000; Sobel, 2001; Zacks, 2001). Il sistema in origine si chiamava **Carnivore**, ma la cattiva fama ha indotto a chiamarlo con il nome più innocente DCS1000. L'obiettivo resta però quello di spiare milioni di persone, nella speranza di trovare informazioni su attività illegali. Sfortunatamente il quarto emendamento della costituzione degli Stati Uniti d'America vieta al governo le inchieste senza un mandato. Stabilire se queste 54 parole scritte nel diciottesimo secolo sono ancora rilevanti nel ventunesimo è un argomento che potrebbe tenere i tribunali impegnati sino al ventiduesimo.

Il governo non ha il monopolio nel minacciare la privacy dei cittadini. Il settore privato fa la sua parte. Per esempio, piccoli file chiamati cookie che i Web browser memorizzano sul computer dell'utente permettono alle aziende di tracciare le sue attività nel cyberspazio, e possono anche abilitare la fuga di informazioni confidenziali come numeri delle carte di credito e codici fiscali (Berghel, 2001). Le reti di calcolatori offrono la possibilità di man-

dare messaggi anonimi. In alcune situazioni questa possibilità può essere desiderabile. Per esempio, permette a studenti, soldati, impiegati e cittadini di segnalare comportamenti illegali da parte di professori, ufficiali, superiori e politici senza il timore di rappresaglie. D'altro canto, negli Stati Uniti e nella maggior parte delle altre democrazie la legge permette esplicitamente a una persona accusata il diritto di confrontarsi con il proprio accusatore e di controbatterlo in tribunale. Le accuse anonime non possono essere usate come prova.

In breve, le reti di calcolatori, come la stampa 500 anni fa, permettono ai comuni cittadini di diffondere le proprie opinioni in modi diversi e a persone diverse da quanto era possibile in precedenza. Questa nuova libertà porta con sé molti problemi sociali, politici e morali irrisolti.

Internet rende possibile trovare le informazioni rapidamente, ma una gran parte è distorta, fuorviante o semplicemente sbagliata. La posta elettronica spazzatura (spam) è diventata una parte dell'esistenza, perché esiste gente che ha raccolto milioni di indirizzi e-mail e li vende su CD-ROM a cosiddetti imprenditori. Messaggi e-mail che contengono contenuto attivo (fondamentalmente programmi e macro eseguiti dalla macchina ricevente) possono contenere virus distruttivi.

Il furto d'identità sta diventando un problema serio, poiché i ladri raccolgono abbastanza informazioni sulla vittima per ottenere carte di credito e altri documenti in suo nome. Infine, la capacità di trasmettere digitalmente musica e video ha aperto la porta a violazioni di copyright di massa che sono difficili da individuare e bloccare.

Molti di questi problemi si potrebbero risolvere se l'industria del computer prendesse seriamente in considerazione l'argomento della sicurezza. Se tutti i messaggi fossero cifrati e autenticati, sarebbe più difficile commettere reati. Questa tecnologia è ben sviluppata e la studieremo in dettaglio nel Capitolo 8. Il problema è che i produttori di hardware e software sanno che inserire le caratteristiche di sicurezza costa, e i loro clienti non chiedono queste funzioni. Inoltre, una buona parte dei problemi è causato da software difettoso, che esiste perché i produttori continuano ad aggiungere sempre nuove caratteristiche ai loro programmi, che inevitabilmente si traducono in altro codice e quindi altri difetti. Una tassa sulle nuove funzioni potrebbe aiutare, ma probabilmente è difficile da accettare. Un rimborso per il software difettoso sarebbe gradito, salvo che manderebbe in bancarotta l'intera industria del software nel primo anno.

1.2 Hardware di rete

È il momento di spostare la nostra attenzione dalle applicazioni e dagli aspetti sociali delle reti (la parte divertente) ai dettagli tecnici che riguardano il progetto (il lavoro). Non c'è una suddivisione accettata in modo universale che risulti applicabile a tutte le reti di computer, ma due dimensioni spiccano per importanza: la tecnologia di trasmissione e la scala. Esamineremo a turno ciascuna di esse.

Parlando in senso lato, ci sono due tipi di tecnologie trasmissive impiegate a largo spettro:

1. collegamenti broadcast
2. collegamenti punto-punto.

Le reti **broadcast** hanno un solo canale di comunicazione che è condiviso fra tutte le macchine della rete. Brevi messaggi, chiamati in alcuni contesti **pacchetti**, sono inviati da ciascuna macchina e ricevuti da tutte le altre. Un campo indirizzo nel pacchetto individua il destinatario. Alla ricezione del pacchetto, una macchina controlla il campo indirizzo. Se il pacchetto è indirizzato alla macchina ricevente viene processato; se è indirizzato a un'altra macchina viene semplicemente ignorato.

Come analogia consideriamo qualcuno che si trova alla fine di un corridoio con molte stanze, e urla "Watson esci, ho bisogno di te qui". Anche se il pacchetto viene ricevuto (ascoltato) da molte persone, solo Watson risponde; gli altri lo ignorano semplicemente. Un'altra analogia è quella degli altoparlanti di un aeroporto che annunciano ai passeggeri del volo 644 di recarsi al gate 12 per l'imbarco immediato.

I sistemi broadcast danno di solito anche la possibilità d'indirizzare un pacchetto a *tutti* i destinatari usando un codice speciale nel campo destinazione. Quando è trasmesso un pacchetto con questo codice, viene ricevuto e processato da tutte le macchine sulla rete. Questo modo operativo è chiamato **broadcasting**. Alcuni sistemi broadcast supportano anche la trasmissione a un sottoinsieme delle macchine, a volte chiamato **multicasting**. Uno schema possibile consiste nel riservare un bit per indicare il multicasting. I rimanenti $n-1$ bit d'indirizzo possono contenere un numero di gruppo. Ogni macchina si può "iscrivere" a uno o all'altro gruppo. Quando viene mandato un pacchetto a un certo gruppo, è consegnato a tutte le macchine che fanno parte di quel gruppo.

Al contrario le reti **punto-punto** consistono di molte connessioni tra singole coppie di macchine. Per andare dalla sorgente alla destinazione, in questo tipo di rete un pacchetto deve visitare una o più macchine intermedie. Spesso sono possibili percorsi multipli con diversa lunghezza, quindi nelle reti punto-punto è importante trovare quelli migliori. Come regola generale (anche se esistono molte eccezioni) le reti più piccole e geograficamente localizzate tendono a usare il broadcasting, mentre le reti con dimensioni maggiori di solito sono punto-punto. La trasmissione punto-punto con un trasmettitore e un ricevitore a volte è chiamata **unicasting**.

Un criterio alternativo per classificare le reti è la loro scala. Nella Figura 1.6 classifichiamo diversi sistemi a processore per la loro dimensione fisica. Alla sommità si trovano le *Personal Area Network*, cioè reti che sono pensate per una sola persona. Per esempio una rete senza fili che collega un computer al suo mouse, tastiera e stampante è una personal area network. Anche un PDA che controlla l'apparecchio acustico dell'utente o il pacemaker è collocato in questa categoria. Dopo la personal area network si trovano reti con raggio d'azione più grande. Si possono dividere in reti locali (*Local Area Network*), reti metropolitane (*Metropolitan Area Network*), e reti ad ampio raggio (*Wide Area Network*). Infine, la connessione di due o più reti è chiamata *internetwork* (o *inter-rete*).

Distanza tra i processori	Processori situati nello stesso/a	Esempio
1 m	Metro quadrato	Personal area network
10 m	Stanza	
100 m	Edificio	Local area network
1 km	Campus	
10 km	Città	Metropolitan area network
100 km	Nazione	
1.000 km	Continente	Wide area network
10.000 km	Pianeta	Internet

Figura 1.6. Classificazione dei processori interconnessi in base alla scala.

Internet è un ben noto esempio di internetwork. La distanza è importante come criterio di classificazione perché per ogni scala si usano tecniche diverse. In questo libro ci occuperemo di reti di tutte le scale. Di seguito diamo una breve introduzione all'hardware di rete.

1.2.1 Reti locali

Le reti locali, solitamente chiamate LAN, sono reti private installate all'interno di un singolo edificio o campus, con dimensione fino a qualche Km. Sono largamente impiegate per collegare personal computer e workstation negli uffici delle aziende e nelle fabbriche, allo scopo di condividere risorse (per esempio stampanti) e scambiare informazioni. Le LAN si distinguono dagli altri tipi di rete per tre caratteristiche: (1) la dimensione, (2) la tecnologia di trasmissione, (3) la topologia.

Le reti LAN hanno dimensioni contenute, il che significa che il tempo di trasmissione più sfavorevole ha un limite, che è noto. Conoscere questo limite permette l'uso di alcune tecniche, che non sarebbero altrimenti applicabili. Semplifica inoltre la gestione della rete. Le LAN possono usare una tecnologia di trasmissione rappresentata da un cavo a cui sono connesse tutte le macchine, come le linee telefoniche duplex usate in passato in alcune zone rurali. Le LAN tradizionali lavorano a velocità comprese tra 10 Mbps e 100 Mbps, hanno bassi ritardi (microsecondi o nanosecondi) e hanno pochissimi errori. Le LAN più recenti operano fino a 10 Gbps. In questo libro ci atterremo alla tradizione, e misureremo le velocità di linea in megabit/sec (1 Mbps è 1.000.000 bit/sec) e gigabit/sec (1 Gbps è 1.000.000.000 bit/sec).

Per le LAN broadcast sono possibili differenti topologie; la Figura 1.7 ne mostra due. In una rete a bus (cioè realizzata con cavo lineare), a ogni istante è master (può trasmettere) al più una macchina. Tutte le altre macchine devono astenersi dal trasmettere. Un meccanismo di arbitraggio è necessario per risolvere i conflitti quando due o più macchine desi-

derano trasmettere simultaneamente. Il meccanismo di arbitraggio può essere centralizzato o distribuito. Per esempio IEEE 802.3, chiamata abitualmente Ethernet, è una rete broadcast a bus con controllo non centralizzato, che normalmente lavora a velocità comprese tra 10 Mbps e 1 Gbps. I computer connessi a una rete Ethernet possono trasmettere quando vogliono; se due o più pacchetti collidono, ogni computer attende per un intervallo di tempo casuale e riprova nuovamente in seguito.

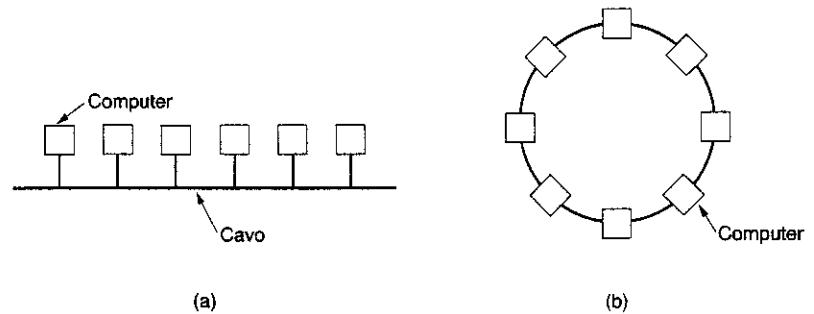


Figura 1.7. Due reti broadcast. (a) Bus. (b) Anello.

Un secondo tipo di sistema broadcast è l'anello. In un anello, ogni bit si propaga in modo autonomo, senza aspettare il resto del pacchetto a cui appartiene. Di solito ogni bit percorre l'intero anello nel tempo che corrisponde alla trasmissione di qualche bit, spesso prima che sia trasmesso l'intero pacchetto. Come per tutti gli altri sistemi broadcast, servono regole per arbitrare l'accesso simultaneo all'anello. Si usano diversi metodi, come l'attribuzione di turni alle macchine. IEEE 802.5 (il sistema token ring IBM) è una LAN ad anello che opera a 4 e 16 Mbps. FDDI è un altro esempio di rete ad anello.

Le reti broadcast si possono ulteriormente dividere in statiche e dinamiche, a seconda del modo in cui è allocato il canale. Una tipica allocazione statica consiste nel suddividere il tempo in intervalli discreti e usare un algoritmo *round-robin*, permettendo a ogni macchina di eseguire il broadcast solo quando è attivo il proprio turno (*time slot*). L'allocazione statica spreca capacità del canale quando una macchina non ha nulla da trasmettere durante il proprio slot, per questo motivo la maggioranza dei sistemi tenta di allocare il canale dinamicamente (cioè su richiesta).

I metodi di allocazione dinamica per un canale condiviso possono essere centralizzati o non centralizzati. Nel metodo di allocazione centralizzato esiste una singola entità, per esempio un'unità di arbitraggio del bus, che stabilisce a chi spetta di volta in volta l'uso del mezzo. Può farlo accettando richieste e prendendo decisioni in base ad algoritmi interni. Nel metodo di allocazione non centralizzato non esiste un'entità centrale, ogni macchina deve decidere in autonomia se trasmettere. Si può pensare che ciò porti sempre al caos, ma non è così. Più avanti vedremo molti algoritmi studiati per portare ordine nel caos potenziale.

1.2.2 Metropolitan Area Network

Una rete metropolitana (*Metropolitan Area Network* o MAN) copre un'intera città. L'esempio più noto di MAN è la rete di televisione via cavo disponibile in molte città degli Stati Uniti. Questo sistema si è sviluppato partendo dagli originari sistemi di antenna comunitari, installati nelle zone con cattiva ricezione della televisione terrestre. In questi sistemi primitivi veniva installata una grande antenna sulla cima di una collina nelle vicinanze e da qui il segnale era trasmesso nelle abitazioni dei clienti.

All'inizio questi sistemi erano progettati localmente e costruiti per l'occasione. In seguito alcune aziende hanno iniziato a entrare nel mercato, ottenendo contratti dall'amministrazione locale per cablare intere città. Il passo successivo fu la realizzazione di programmi televisivi e interi canali appositamente per i clienti della TV via cavo. Spesso questi canali erano altamente specializzati: solo notizie, solo ricette, solo sport, solo giardinaggio e così via. Ma dagli inizi fino ai tardi anni '90 furono concepiti solo per la ricezione della televisione.

Quando Internet ha iniziato ad attirare un pubblico di massa, gli operatori delle TV via cavo hanno cominciato a capire che con alcuni cambiamenti al sistema potevano offrire un servizio Internet bidirezionale in porzioni inutilizzate dello spettro. A quel punto il sistema di TV via cavo ha iniziato a trasformarsi da un mezzo per distribuire il segnale televisivo a una Metropolitan Area Network. In prima approssimazione una MAN assomiglia al sistema mostrato nella Figura 1.8. In questa figura vediamo i segnali televisivi e Internet inseriti nella **stazione di testa** (*head end*) centralizzata, per la successiva distribuzione nelle abitazioni. Torneremo in dettaglio su questo argomento nel Capitolo 2.

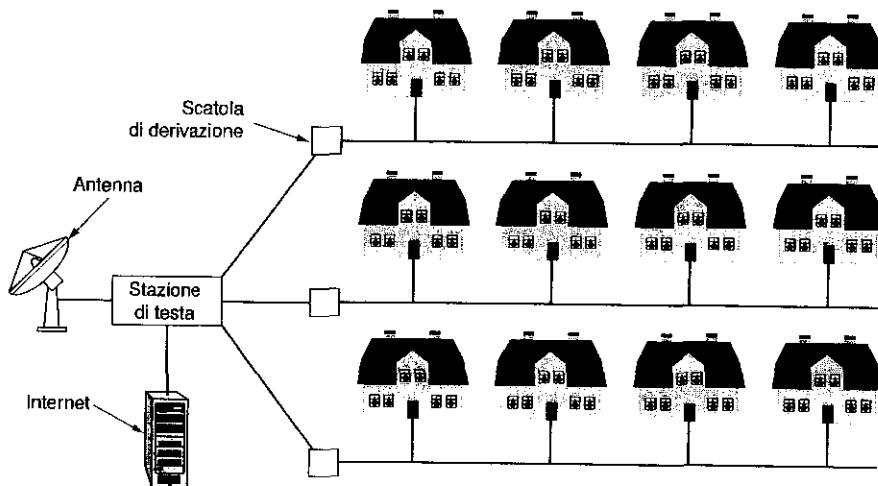


Figura 1.8. Una MAN basata sulla TV via cavo.

La televisione via cavo non è l'unica MAN. Recenti sviluppi nel collegamento wireless a Internet ad alta velocità hanno prodotto un'altra MAN, che è stata standardizzata con la sigla IEEE 802.16. Vedremo questo argomento nel Capitolo 2.

1.2.3 Wide Area Network

Una *Wide Area Network* o WAN copre un'area geograficamente estesa, spesso una nazione o un continente. Racchiude una raccolta di macchine destinate a eseguire programmi utente (applicazioni). Seguendo le consuetudini, queste macchine vengono chiamate **host**. Gli host sono collegati da una **communication subnet**, per brevità chiamata semplicemente **subnet** (sottorete). Gli host sono di proprietà dei clienti (per esempio i personal computer degli utenti), mentre la communication subnet è generalmente posseduta e gestita da una compagnia telefonica o da un Internet service provider. Il compito della subnet è quello di trasportare i messaggi da un host all'altro, come la rete telefonica trasporta le parole da chi parla a chi ascolta. La separazione dell'aspetto di pura comunicazione della rete (la subnet) dagli aspetti applicativi (gli host) semplifica notevolmente il progetto dell'intera rete.

Nella maggior parte delle WAN, la subnet è formata da due componenti: linee di trasmissione ed elementi di commutazione. Le **linee di trasmissione** spostano i bit tra le macchine. Possono essere realizzate con cavo in rame, fibra ottica, o anche collegamenti radio. Gli **elementi di commutazione** sono computer specializzati che collegano tre o più linee di trasmissione. Quando i dati arrivano da una linea ricevente, l'elemento di commutazione deve scegliere una linea di uscita su cui inoltrarlo. In passato questi computer che commutano i dati sono stati chiamati in molti modi; oggi il più diffuso è il termine **router**.

In questo modello, mostrato nella Figura 1.9, ogni host è solitamente collegato a una LAN su cui è presente un router, anche se in alcuni casi un host può essere collegato direttamente a un router. L'insieme di linee di comunicazione e router (ma non gli host) forma la subnet.

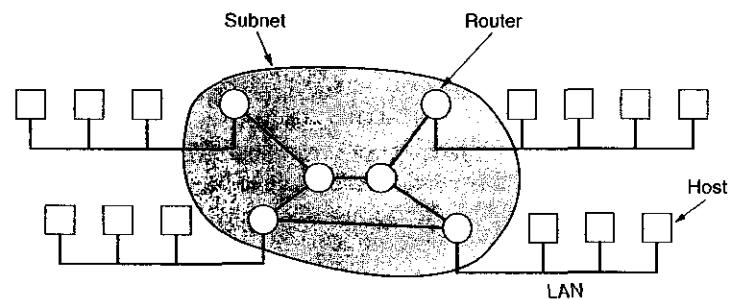


Figura 1.9. Relazione tra gli host delle LAN e la subnet.

È necessario un breve chiarimento sul termine **subnet**. In origine il **solo** significato era quello che indicava l'insieme di router e linee di comunicazione che spostano i dati dall'host sorgente a quello destinatario. Tuttavia alcuni anni dopo ha acquisito un secondo significato in abbinamento all'indirizzamento di rete (che sarà discusso nel Capitolo 5). Sfortunatamente non esiste un'alternativa di largo utilizzo per indicare il suo significato

originario, quindi con qualche esitazione lo useremo con entrambi i significati. Dal contesto sarà sempre chiaro qual è quello corretto.

Nella maggior parte delle WAN la rete contiene molte linee di trasmissione, ciascuna delle quali collega una coppia di router. Se due router che non condividono la stessa linea di trasmissione vogliono comunicare, lo devono fare indirettamente attraverso altri router. Quando un pacchetto viene inviato da un router verso un altro, attraverso router intermedi, il pacchetto viene ricevuto integralmente da ciascun router intermedio, memorizzato finché non si libera la linea di uscita necessaria, e poi inoltrato. Una subnet organizzata secondo questo principio si chiama **store-and-forward** o **packet-switched** (a commutazione di pacchetto). Praticamente tutte le WAN (con l'eccezione di quelle che usano i satelliti) hanno subnet di tipo store-and-forward. Quando i pacchetti sono piccoli e hanno la stessa dimensione, vengono spesso chiamati **celle**.

Il principio di funzionamento di una WAN a commutazione di pacchetto è così importante che vale la pena spendervi qualche parola in più. Di solito, quando un processo di qualche host ha un messaggio da mandare a un processo di qualche altro host, quello che trasmette inizia l'attività dividendo il messaggio in pacchetti, ciascuno dei quali porta un corrispondente numero di sequenza. Questi pacchetti vengono poi inseriti nella rete uno alla volta in rapida successione. I pacchetti sono trasportati individualmente sulla rete e depositati nell'host ricevente, dove sono riassemblati nel messaggio originale e inoltrati al processo ricevente. Il flusso di pacchetti che risulta da un messaggio iniziale è illustrato nella Figura 1.10.

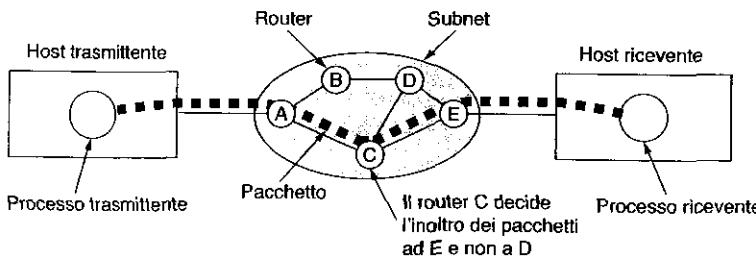


Figura 1.10. Il flusso di pacchetti dal trasmettitore al ricevitore.

In questa figura tutti i pacchetti seguono la regola ACE, invece che ABDE o ACDE. In alcune reti tutti i pacchetti di un dato messaggio devono seguire lo stesso percorso; in altri ogni pacchetto è instradato separatamente. Naturalmente se ACE è il percorso migliore tutti i pacchetti possono esservi inoltrati, anche se ciascuno è instradato separatamente.

La decisione per l'instradamento è presa localmente. Quando un pacchetto arriva al router A, dipende da A decidere se va inoltrato alla linea B o alla linea C. Il modo in cui A prende la decisione è chiamato **algoritmo di routing**. Ne esistono molti, e ne studieremo alcuni in dettaglio nel Capitolo 5.

Non tutte le WAN sono a commutazione di pacchetto. Una seconda possibilità per una WAN è il sistema satellitare. Ogni router ha un'antenna attraverso la quale può trasmettere e ricevere. Tutti i router possono ascoltare l'uscita *dal* satellite, e in alcuni casi anche le trasmissioni dei router paritetici *verso* il satellite. A volte i router sono collegati a una subnet cablata punto-punto, e solo alcuni hanno un'antenna per il satellite. Le reti satellitari sono intrinsecamente di tipo broadcast e sono particolarmente utili quando la proprietà broadcast è importante.

1.2.4 Reti wireless

La comunicazione digitale senza fili (wireless) non è un'idea nuova. Già nel 1901 il fisico Guglielmo Marconi dimostrò la telegrafia senza fili da una nave usando il codice Morse (punti e linee sono binari, dopotutto). I sistemi digitali wireless moderni hanno prestazioni migliori, ma l'idea di base è la stessa.

In prima approssimazione le reti wireless si possono classificare in tre categorie principali:

1. connessioni all'interno di un sistema
2. LAN wireless
3. WAN wireless.

La connessione all'interno di un sistema coinvolge il collegamento delle periferiche di un computer tramite segnali radio a portata ridotta. Quasi tutti i computer hanno monitor, tastiera, mouse e stampante collegate all'unità centrale tramite cavi. Anche se di solito sono marchiati da colori differenti, il numero di principianti che incontrano difficoltà nel collegare correttamente tutti i cavi nei piccoli connettori è così grande che la maggior parte dei produttori di computer offre la possibilità di mandare un tecnico a domicilio per farlo. Di conseguenza, alcuni costruttori si sono accordati per progettare una rete wireless a portata ridotta chiamata **Bluetooth**, per abolire i cavi di collegamento di questi componenti. Bluetooth permette inoltre di collegare al computer macchine fotografiche digitali, cuffie senza fili, scanner e altri dispositivi semplicemente portandoli nella zona di copertura. Niente cavi e niente installazione di driver: solo avvicinare, accendere e usare. Per molti questa semplicità d'uso è un grande vantaggio.

Nella loro forma più semplice, le reti di collegamento interne al sistema usano il paradigma master-slave della Figura 1.11(a). L'unità centrale è normalmente il master, che dialoga con mouse, tastiera e gli altri componenti configurati come slave. Il master dice agli slave che indirizzi usare, quando possono trasmettere e per quanto tempo, che frequenze possono usare e così via. Discuteremo Bluetooth con maggiore dettaglio nel Capitolo 4. Il passo successivo nelle reti wireless sono le LAN. Si tratta di sistemi dove ogni computer ha un modem radio e un'antenna con cui può comunicare con altri sistemi. Spesso c'è un'antenna nel soffitto con cui i computer dialogano, come mostrato nella Figura 1.11(b). Tuttavia, quando i computer sono abbastanza vicini, possono comunicare direttamente tra di loro in una configurazione peer-to-peer. Le LAN wireless stanno diventando sempre più comuni nei piccoli uffici e nelle abitazioni (dove l'impegno per installare Ethernet è considerato eccessivo), negli edifici per uffici più vecchi, nelle sale riunioni e in altri posti.

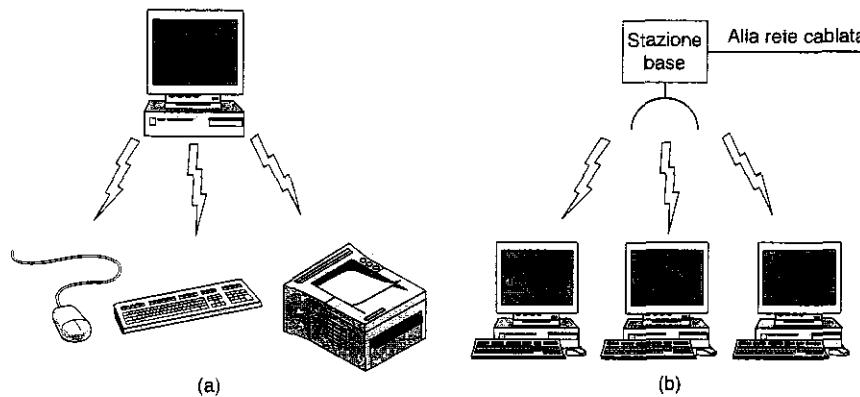


Figura 1.11. (a) Configurazione Bluetooth. (b) LAN wireless.

Esiste uno standard per LAN wireless, chiamato **IEEE 802.11**, che è implementato dalla maggioranza dei sistemi e che sta diventando estremamente diffuso. Lo esamineremo nel Capitolo 4.

Il terzo tipo di reti wireless è usato nei sistemi su area estesa. La rete radio usata dai telefoni cellulari è un esempio di sistema wireless a banda ridotta. Il sistema è già arrivato alla terza generazione. La prima generazione era analogica e solo per la voce. La seconda generazione è digitale e solo per la voce. La terza generazione è digitale e trasporta voce e dati. In un certo senso, le reti wireless cellulari assomigliano a LAN wireless, eccetto che le distanze sono molto superiori e i bit rate molto più bassi. Le LAN wireless possono funzionare a cadenze che si spingono a 50 Mbps su distanze di decine di metri. I sistemi cellulari lavorano sotto a 1 Mbps, ma la distanza tra la stazione base e il computer o telefono si misura in chilometri invece che metri. Si esamineranno a fondo queste reti nel Capitolo 2.

A complemento di queste reti a bassa velocità, sono in sviluppo anche reti wireless a larga banda e copertura estesa. Il focus iniziale è l'accesso a Internet ad alta velocità da abitazioni e uffici, scavalcando il sistema telefonico. Questo servizio è spesso chiamato *local multipoint distribution service*, e lo studieremo più avanti in questo libro. Per esso è stato sviluppato uno standard, chiamato IEEE 802.16, che esamineremo nel Capitolo 4.

Quasi tutte le reti wireless prima o poi si collegano alla rete fissa per consentire l'accesso a file, database e Internet. Questi collegamenti si possono realizzare in molti modi, a seconda delle circostanze. Per esempio, nella Figura 1.12(a) è illustrato un aeroplano con un certo numero di persone che usano modem e telefoni applicati ai sedili per chiamare l'ufficio. Ogni telefonata è indipendente dalle altre. Una soluzione molto più efficiente, tuttavia, è la LAN volante della Figura 1.12(b).

Qui ogni sedile è munito di connettore Ethernet a cui i passeggeri possono collegare i loro computer. Un singolo router installato sull'aereo mantiene un collegamento radio con qualche router a terra, cambiandolo man mano che il velivolo avanza. Questa configurazione è solo una LAN tradizionale, eccetto che la sua connessione con il mondo esterno è un collegamento radio invece di un filo.

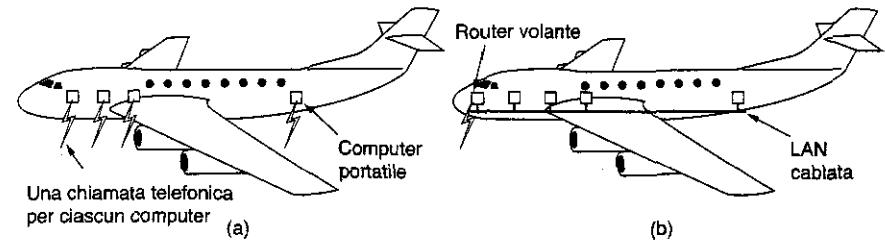


Figura 1.12. (a) Computer portatili individuali. (b) LAN volante.

Molti ritengono che la tecnologia wireless sia il futuro (as.: Bi et al., 2001; Leeper, 2001; Varshey e Vetter, 2000) ma c'è almeno una voce fuori dal coro. Bob Metcalfe, l'inventore di Ethernet, ha scritto "I computer portatili sono come i servizi igienici portatili senza tubazioni. Sono comuni sui veicoli, nei cantieri e ai concerti rock. Il mio consiglio è quello di cablare la propria casa e non abbandonare i cavi" (Metcalfe, 1995). La storia potrebbe accostare questa frase a quella del presidente di IBM T.J. Watson, quando nel 1945 spiegava come mai IBM non intendeva lanciarsi nel mercato dei computer "Nel mondo intero serviranno al massimo quattro o cinque computer, fino all'anno 2000".

1.2.5 Reti domestiche

Le reti locali domestiche sono all'orizzonte. L'idea di base è che in futuro la maggior parte delle abitazioni sarà predisposta per la rete locale; ogni dispositivo domestico sarà capace di comunicare con ogni altro, e tutti quanti saranno accessibili tramite Internet. Questo è uno di quei concetti visionari che nessuno ha previsto (come il telecomando della TV o i telefoni cellulari), ma una volta arrivati nessuno riesce a immaginarsi come poteva vivere senza. Molti dispositivi sono collegabili in rete; alcune delle categorie più ovvie (con esempi) sono le seguenti:

1. computer (PC da tavolo, PC portatili, PDA, periferiche condivise)
2. intrattenimento (TV, DVD, VCR, videocamera, macchine fotografiche digitali, stereo, player MP3)
3. telecomunicazioni (telefoni, telefoni cellulari, citofoni, fax)
4. elettrodomestici (microonde, frigoriferi, orologi, condizionatori, lampade)
5. telemetria (contatori, allarmi antincendio e antifurto, termostati, sistemi di sorveglianza neonati).

Le reti locali domestiche esistono già, in forma limitata. Molte abitazioni hanno già un dispositivo per collegare più computer a una connessione Internet veloce. L'intrattenimento in rete non è ancora arrivato, ma con l'aumentare della musica e dei film che possono essere scaricati da Internet nascerà una domanda per collegarvi stereo e televisioni. Inoltre, la gente vuole condividere i propri filmati con amici e parenti, quindi la connessione deve essere bidirezionale. Gli apparati per telecomunicazioni sono già collegati al mondo esterno, e presto saranno digitali e lavoreranno su Internet. Una tipica abitazione probabilmente ha una dozzina di orologi (per esempio negli elettrodomestici), ciascuno dei quali va regolato due volte all'anno: quando entra in vigore l'ora legale e quando cessa. Se tutti gli orologi fossero su Internet, questa regolazione si potrebbe fare automaticamente. Infine, la sorveglianza remota della casa e del suo contenuto avrà probabilmente un'enorme successo. Presumibilmente molti genitori saranno disposti a spendere un po'di denaro per controllare sullo schermo del proprio PDA, mentre sono fuori a cena, i figli che dormono (anche quando in casa c'è una babysitter). È possibile immaginare una rete separata per ogni area applicativa, ma l'integrazione all'interno di una singola struttura di rete è probabilmente un'idea migliore.

Le reti domestiche hanno alcune caratteristiche fondamentali diverse dagli altri tipi di rete. Prima di tutto, la rete e i dispositivi devono essere facili da installare. Nel corso degli anni l'autore di questo libro ha installato molti componenti hardware e software su svariati computer, con risultati variabili. Una serie di telefonate all'assistenza tecnica del produttore ha tipicamente portato a risposte come (1) Leggere il manuale, (2) Riavviare il computer, (3) Riprovare dopo aver tolto tutto l'hardware e il software eccetto il nostro, (4) Scaricare i driver aggiornati dal nostro sito Web, (5) Riformattare il disco fisso e reinstallare Windows dal CD-ROM.

Dire all'acquirente di un frigorifero Internet di scaricare e installare una nuova versione del sistema operativo del frigorifero difficilmente produrrà clienti soddisfatti. Gli utenti di computer sono abituati a convivere con prodotti che non funzionano; i consumatori di auto, televisioni e frigoriferi sono molto meno tolleranti. Si aspettano prodotti che funzionano al 100% sin dall'inizio.

In secondo luogo, la rete e i dispositivi devono essere "a prova d'idiota". Una volta i condizionatori avevano una manopola con quattro posizioni: OFF, LOW, MEDIUM e HIGH. Oggi hanno un manuale di 30 pagine. Domani, una volta collegati in rete, ci si può aspettare che il capitolo sulla sicurezza occuperà da solo 30 pagine. Ciò sarà al di là delle capacità di comprensione di praticamente qualsiasi utente.

Terzo motivo: il prezzo basso è essenziale per il successo. La gente non è disposta a pagare un sovrapprezzo di 50 euro per un termostato Internet, perché poche persone ritengono importante poter controllare la temperatura di casa dall'ufficio. Per un sovrapprezzo di 5 euro, però, il concetto potrebbe aver successo.

Quattro, l'applicazione più importante sarà probabilmente collegata alla multimedialità, quindi la rete deve avere capacità sufficiente. Non c'è mercato per televisioni collegate a Internet che mostrano film scadenti a risoluzione di 320 x 240 pixel e 10 frame al secondo. Fast Ethernet, il cavallo di battaglia nella maggioranza degli uffici, non è sufficiente

per la multimedialità. Di conseguenza, le reti domestiche per diventare un mercato di massa hanno bisogno di prestazioni più alte delle reti per uffici attuali, e di prezzi molto più bassi.

Quinto, deve essere possibile partire con uno o due dispositivi ed espandere la dimensione della rete con gradualità. Ciò significa che non devono innescarsi guerre tra formati. Dire ai consumatori di comprare periferiche con interfacce IEEE 1394 (Firewire) e, pochi anni dopo, tornare sui propri passi dicendo che USB 2.0 è l'interfaccia del mese può solo rendere i consumatori sospettosi. L'interfaccia di rete deve rimanere stabile per molti anni; il cablaggio (se richiesto) dovrà restare stabile per decenni.

Sesto, sicurezza e affidabilità saranno molto importanti. Perdere qualche file per colpa di un virus arrivato via e-mail è una cosa; un ladro che usa il suo PDA per disattivare l'antifurto e svaligiare la casa è un'altra.

Una domanda interessante è se le reti domestiche saranno cablate o wireless. La maggior parte delle abitazioni ospita già sei reti: elettricità, telefono, TV, acqua, gas, scarichi fognari. Aggiungere un settimo impianto durante la costruzione non è difficile, ma adeguare le abitazioni esistenti è costoso. I costi avvantaggiano le reti wireless, ma la sicurezza delle reti cablate è maggiore. Il problema del wireless è che le onde radio impiegate passano molto facilmente attraverso le barriere. Non tutti sono entusiasti al pensiero di avere vicini che si allacciano alla propria connessione Internet e intercettano le e-mail inviate alla stampante. Nel Capitolo 8 studieremo come si può usare la crittografia per offrire sicurezza, ma nel contesto delle reti domestiche la sicurezza deve essere accessibile anche agli utenti senza esperienza. Si fa prima a dirlo che a farlo, anche con utenti molto sofisticati. In breve, le reti domestiche offrono molte opportunità e sfide. La maggior parte riguarda la necessità di essere facili da gestire, affidabili e sicure specialmente nelle mani di utenti non tecnici, mentre allo stesso tempo devono offrire alte prestazioni a basso costo.

1.2.6 Reti tra sistemi

Nel mondo esistono molte reti, con hardware e software diversi. La gente che si collega a una rete spesso vuole comunicare con persone collegate a una rete differente. La soddisfazione di questo desiderio obbliga all'interconnessione di reti diverse (e spesso incompatibili tra loro), a volte tramite macchine chiamate **gateway** che stabiliscono la connessione e offrono i servizi di conversione necessari, in termini di hardware e di software. Un insieme di reti interconnesse si chiama **internetwork** o **internet**. Questi termini sono usati nel loro significato generico, in contrasto con la Internet mondiale (che è una specifica internet), che indicheremo sempre con l'iniziale maiuscola.

Una forma comune di internet è rappresentata da un gruppo di LAN collegate da una WAN. In effetti, se nella Figura 1.9 sostituiamo la parola "subnet" con "WAN" non ci sarebbe bisogno di cambiare nient'altro per illustrare questo caso. L'unica vera distinzione tecnica tra una subnet e una WAN riguarda la presenza di host. Se il sistema nell'area grigia contiene solo router è una subnet; se contiene router e host, è una WAN. Le vere differenze riguardano la proprietà e l'utilizzo.

Spesso si fa confusione tra subnet, reti e internetwork. Subnet è un termine particolarmente appropriato al contesto delle Wide Area Network, dove si riferisce all'insieme di router e linee di comunicazione possedute dall'operatore di rete. Per analogia, il sistema telefonico è composto da centrali di commutazione connesse una all'altra da linee ad alta velocità, e collegate alle abitazioni e uffici da linee a bassa velocità. Linee e apparati, posseduti e gestiti dalla società telefonica, formano le subnet della rete telefonica. I telefoni (che in questa analogia sono gli host) non fanno parte delle subnet. La combinazione delle subnet e dei rispettivi host è una rete. Nel caso di una LAN, il cavo e gli host formano la rete; non c'è una vera e propria subnet.

Un internetwork viene creato quando si collegano tra loro reti distinte. Nella nostra ottica, collegando una LAN con una WAN o due LAN tra di loro si ottiene un internetwork, ma nell'industria del settore c'è poca concordanza sulla terminologia. Una regola di massima è che se organizzazioni differenti hanno pagato la costruzione di parti diverse della rete, e ciascuna gestisce la propria parte, ci troviamo di fronte a un internetwork e non a una singola rete. Inoltre, se la tecnologia sottostante cambia nelle diverse parti (per esempio in alcune è broadcast e in altre punto-punto), probabilmente abbiamo due reti.

1.3 Software di rete

Le prime reti di computer furono progettate con particolare attenzione all'hardware e il software era concepito come un complemento. Questa strategia non funziona più. Oggi il software di rete è altamente strutturato, e nei paragrafi seguenti si esaminerà qualche dettaglio delle tecniche di strutturazione del software. I metodi descritti rappresentano la parte cruciale dell'intero libro, e saranno rivisti più volte nel seguito.

1.3.1 Gerarchie dei protocolli

Per diminuire la complessità, la maggior parte delle reti è organizzata come pila di strati (*layer*) o livelli, costruiti uno sull'altro. Numero, nome, contenuto e funzione di ogni strato variano da rete a rete. Lo scopo di ogni strato è quello di offrire determinati servizi agli strati di livello superiore, schermendoli dai dettagli sull'implementazione dei servizi. In un certo senso, ogni strato è una specie di macchina virtuale che offre certi servizi allo strato che si trova al di sopra.

Questo concetto familiare è impiegato in tutta l'informatica, dove viene variamente definito come information hiding, tipi dati astratti, encapsulazione dei dati e programmazione orientata agli oggetti. L'idea di base è che una particolare porzione di software (o hardware) offre un servizio ai propri utenti nascondendogli però i dettagli dello stato interno e gli algoritmi utilizzati.

Lo strato *n* di un computer è in comunicazione con lo strato *n* di un altro computer. Le regole e le convenzioni usate in questa comunicazione sono globalmente note come i protocolli dello strato *n*. Fondamentalmente un **protocollo** è un accordo, tra le parti che comu-

nican, sul modo in cui deve procedere la comunicazione. Per analogia, quando una donna viene presentata a un uomo può decidere di porgere la mano. Lui, a sua volta, può decidere di stringere la mano o di baciarla, a seconda (per esempio) del fatto che lei sia un avvocato americano in una riunione d'affari o una principessa europea a un ballo ufficiale. Violare il protocollo rende la comunicazione più difficile, se non del tutto impossibile. Nella Figura 1.13 è illustrata una rete a cinque strati. Le entità che formano gli strati di pari livello sui diversi computer sono chiamati **pari** (*peer*). I pari possono essere processi, dispositivi hardware o anche esseri umani. In altri termini, i pari comunicano usando il protocollo.

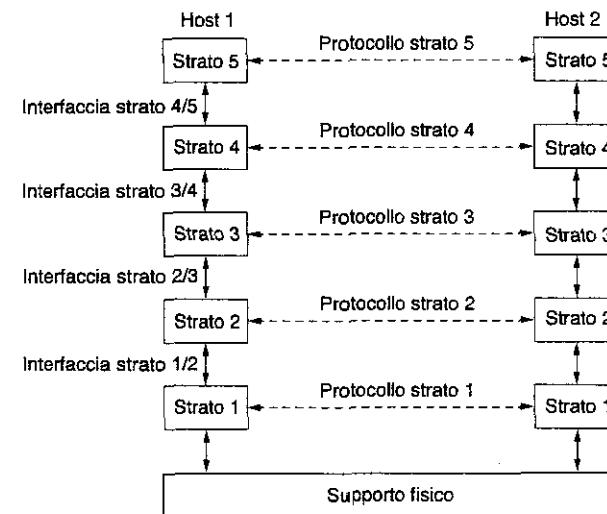


Figura 1.13. Strati, protocolli e interfacce.

In realtà i dati non sono trasferiti direttamente dallo strato *n* di un computer allo strato *n* di un altro. Invece, ogni strato passa dati e informazioni di controllo allo strato immediatamente sottostante, fino a raggiungere quello più basso. Sotto lo strato 1 si trova il **supporto fisico** attraverso cui è possibile la comunicazione vera e propria. Nella Figura 1.13 la comunicazione virtuale è indicata da linee tratteggiate e la comunicazione fisica da linee continue.

Tra ciascuna coppia di strati contigui si trova un'interfaccia. L'interfaccia definisce le operazioni elementari e i servizi che lo strato inferiore rende disponibili a quello soprastante. Quando i progettisti di rete decidono il numero di strati da includere in una rete e la funzione di ciascuno, uno degli aspetti più importanti consiste nel definire interfacce pulite tra gli strati. Per fare ciò è necessario che ogni strato esegua un insieme specifico di funzioni ben definite. Oltre a minimizzare la quantità d'informazioni che devono esse-

re scambiate tra gli strati, le interfacce pulite rendono più facile sostituire l'implementazione di uno strato con un'altra del tutto diversa (per esempio, tutte le linee telefoniche vengono sostituite con canali satellitari), perché la nuova implementazione dovrà semplicemente offrire allo strato immediatamente soprastante esattamente lo stesso insieme di servizi della vecchia. In pratica, è frequente che host distinti usino diverse implementazioni. L'insieme di strati e protocollo si chiama **architettura di rete**. Le specifiche di un'architettura devono contenere abbastanza informazioni per consentire all'implementatore di scrivere il programma o costruire l'hardware di ciascuno strato in modo che possa seguire correttamente il protocollo. Né i dettagli dell'implementazione né le specifiche delle interfacce fanno parte dell'architettura, perché sono nascoste all'interno dei computer e invisibili dall'esterno. Non è neppure necessario che le interfacce di tutti i computer di una rete siano identiche, purché ogni computer possa usare correttamente tutti i protocolli. Un elenco di protocolli usati da uno specifico computer, per semplificare un protocollo per ogni strato [NdR - non è detto che una pila di protocolli contenga esattamente un protocollo per livello], è chiamato **pila di protocolli (protocol stack)**. Gli argomenti principali di questo libro sono le architetture di rete, le pile di protocolli e i protocolli stessi. Un'analogia può aiutare a spiegare l'idea di una comunicazione multistrato. Immaginiamo due filosofi (processi pari dello strato 3), uno dei quali parla urdu e inglese mentre l'altro parla cinese e francese. Poiché non hanno linguaggi in comune, ciascuno assume un interprete (processo pari dello strato 2), ciascuno dei quali a sua volta contatta una segretaria (processo pari dello strato 1). Il filosofo 1 vuole comunicare il proprio apprezzamento verso *oryctolagus cuniculus* al proprio pari. Per farlo, passa un messaggio (in inglese) attraverso l'interfaccia 2/3 al suo interprete, dicendo "mi piacciono i conigli", come mostrato nella Figura 1.14. Gli interpreti si sono accordati su un linguaggio neutro noto a entrambi, l'olandese, in questo modo il messaggio viene convertito in "Ik vind konijnen leuk". La scelta della lingua è il protocollo dello strato 2 e dipende dai processi pari di questo strato.

L'interprete quindi dà il messaggio alla segretaria per la trasmissione, per esempio, via fax (il protocollo dello strato 1). Quando il messaggio arriva, è tradotto in francese e passato attraverso l'interfaccia 2/3 al filosofo 2. Notiamo che ogni protocollo è completamente indipendente dagli altri, fino a quando non vengono cambiate le interfacce. Gli interpreti potrebbero passare liberamente dall'olandese al finlandese (per esempio), a patto che entrambi acconsentano al cambiamento e nessuno dei due cambi l'interfaccia con lo strato 1 o lo strato 3. Allo stesso modo, le segretarie possono passare dal fax alla e-mail o telefono senza disturbare (o informare) gli altri strati. Ogni processo può aggiungere informazioni destinate al proprio pari, che non sono passate allo strato superiore.

Ora esaminiamo un esempio più tecnico: come realizzare la comunicazione tra gli strati più alti della rete a cinque strati della Figura 1.15. Un messaggio M è prodotto da un processo applicativo che lavora nello strato 5 e passato allo strato 4 per la trasmissione. Lo strato 4 mette un **header** davanti al messaggio per identificarlo e passa il risultato allo

Introduzione

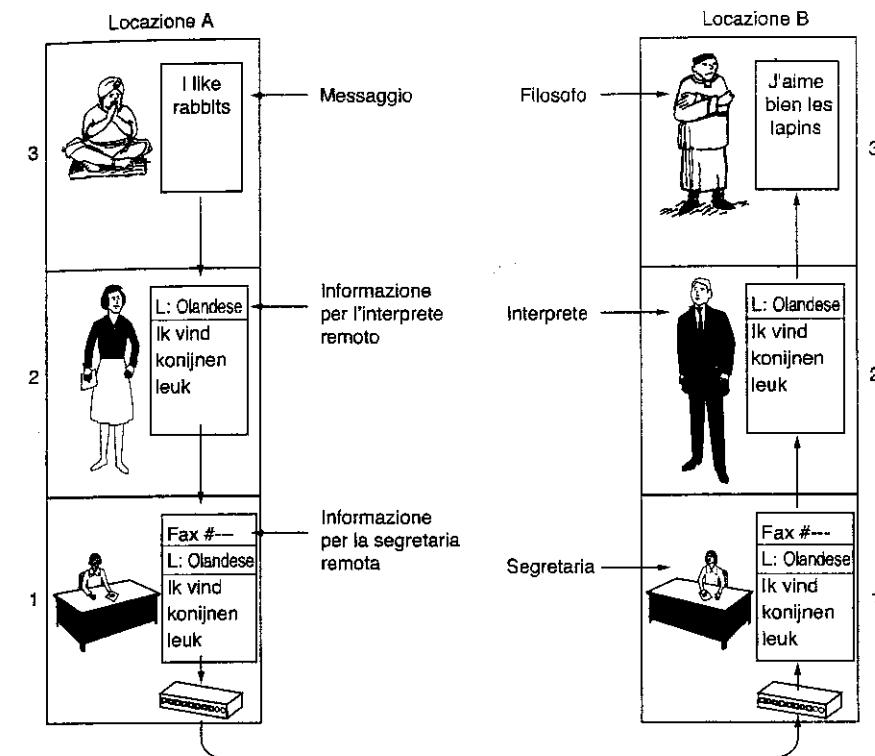


Figura 1.14. L'architettura filosofo-interprete-segretaria.

strato 3. L'header include informazioni di controllo, come i numeri di sequenza, per consentire allo strato 4 del computer destinatario di consegnare il messaggio nell'ordine corretto quando gli strati sottostanti non mantengono la sequenza. In alcuni strati, gli header possono anche contenere dimensioni, tempi e altri campi di controllo.

In molte reti non c'è limite alla dimensione dei messaggi trasmessi nel protocollo dello strato 4, ma quasi sempre esiste un limite imposto dal protocollo dello strato 3. Di conseguenza, lo strato 3 deve spezzare i messaggi in arrivo in unità più piccole chiamate pacchetti, aggiungendo un header dello strato 3 davanti a ogni pacchetto. In questo esempio M è diviso in due parti, M_1 e M_2 .

Lo strato 3 decide la linea di uscita da usare e passa i pacchetti allo strato 2. Lo strato 2 aggiunge a ogni pezzo non solo un header, ma anche un trailer e fornisce l'unità d'informazione risultante allo strato 1 per la trasmissione fisica. Nel computer ricevente il messaggio si muove verso l'alto, passando da strato a strato, e gli header vengono rimossi man mano che avanza. Nessuno degli header per gli strati inferiori a n sono passati allo strato n .

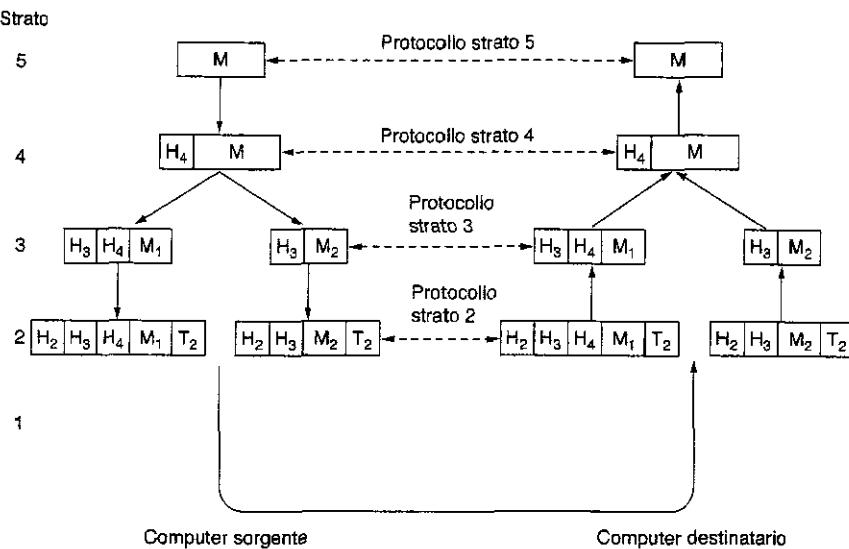


Figura 1.15. Esempio di flusso informativo che supporta la comunicazione virtuale nello strato 5.

La cosa importante da capire nella Figura 1.15 è la relazione tra la comunicazione reale e quella virtuale, e la differenza tra protocolli e interfacce. I processi pari dello strato 4, per esempio, modellano concettualmente la loro comunicazione come se fosse “orizzontale”, usando il protocollo dello strato 4. Ognuno ha probabilmente una procedura chiamata *SendToOtherSide* e *GetFromOtherSide*, anche se queste procedure in realtà comunicano mediante strati inferiori attraverso l’interfaccia 3/4, e non direttamente con la controparte. L’astrazione dei processi pari è fondamentale per l’intero progetto delle reti. Usandola è possibile spezzare in diverse parti più piccole e gestibili il compito impossibile di progettare l’intera rete, poiché basta progettare i singoli strati. Anche se il Paragrafo 1.3 si chiama “software di rete”, vale la pena osservare che gli strati più bassi di una gerarchia di protocolli sono spesso implementati in hardware o firmware. In ogni caso nei protocolli sono coinvolti complessi algoritmi, anche quando sono nascosti (del tutto o in parte) all’interno dell’hardware.

1.3.2 Progettazione degli strati

Alcuni problemi fondamentali da affrontare nel progetto delle reti sono presenti in molti strati. Nel seguito si esaminano brevemente i più importanti.

Ogni strato ha bisogno di un meccanismo per identificare sorgenti e destinazioni. Poiché una rete normalmente ha molti computer, alcuni dei quali hanno più processi, è necessario un mezzo per fare in modo che un processo su un dato computer possa specificare con chi vuole comunicare. Come conseguenza della presenza di destinazioni multiple, è indi-

spensabile qualche forma d'**indirizzamento** per specificare l’esatta destinazione.

Un altro insieme di decisioni di progetto riguarda le regole per il trasferimento dei dati. In alcuni sistemi i dati viaggiano in una sola direzione; in altri i dati possono andare in entrambe le direzioni. Il protocollo deve anche stabilire a quanti canali logici corrisponde la connessione e quali sono le corrispondenti priorità. Molte reti offrono almeno due canali logici per connessione, uno per i dati normali e uno per quelli urgenti.

Il **controllo degli errori** è un argomento importante perché i circuiti fisici di comunicazione non sono perfetti. Si conoscono molti codici a rilevamento e correzione d’errore, ma entrambi gli estremi della connessione devono accordarsi su quello da impiegare. In aggiunta, la destinazione deve avere qualche modo per dire al computer sorgente quali messaggi sono stati ricevuti correttamente e quali no.

Non tutti i canali di comunicazione conservano l’ordine dei messaggi inviati. Per contrastare la possibile perdita della sequenza, il protocollo deve includere esplicite indicazioni per consentire al ricevitore di riassemblare correttamente i pezzi. Una soluzione banale consiste nel numerare i pezzi, ma lascia aperta la questione di cosa fare con i pezzi che arrivano fuori sequenza.

Un problema che si ripresenta a ogni livello è quello d’impedire a una sorgente veloce di sovraccaricare di dati un ricevitore lento. Sono state proposte diverse soluzioni e saranno discusse in seguito. Alcune prevedono qualche genere d’informazione di ritorno, diretta o indiretta, che viaggia dalla destinazione verso la sorgente e la informa sullo stato corrente del ricevitore. Altri limitano il trasmettitore a una cadenza di trasmissione concordata. Questo argomento è chiamato **controllo di flusso**.

Un altro problema che va risolto su più livelli è il fatto che tutti i processi non hanno la possibilità di accettare messaggi arbitrariamente lunghi. Questa proprietà porta ai meccanismi per disassemblare, trasmettere e riassemblare i messaggi. Un problema collaterale riguarda ciò che si deve fare quando i processi insistono a trasmettere dati in unità che sono troppo piccole per una trasmissione efficiente. La soluzione consiste nel raccogliere diversi messaggi brevi, diretti a una destinazione comune, per formare un singolo messaggio più grande che poi sarà separato all’estremità ricevente.

Quando è scomodo o troppo costoso aprire una connessione separata per ogni coppia di processi comunicanti, lo strato sottostante può decidere di usare la stessa connessione per più conversazioni simultanee non correlate. Se **multiplexing** e **demultiplexing** sono realizzati in modo trasparente, ogni strato li può sfruttare. Per esempio il multiplexing è richiesto dallo strato fisico quando tutto il traffico per qualsiasi collegamento deve essere spedito attraverso pochi circuiti fisici.

Quando esistono più percorsi tra sorgente e destinazione è necessario scegliere una strada (route). A volte questa decisione va spezzata tra due o più strati. Per esempio, per mandare dati da Londra a Roma può essere necessaria una decisione di alto livello sul fatto di transitare da Francia o Germania, secondo le leggi sulla privacy corrispondenti. Quindi può servire una decisione di livello più basso per scegliere uno dei circuiti disponibili, sulla base del livello del traffico. Questo argomento è chiamato **routing** o **instradamento**.

1.3.3 Servizi orientati alla connessione e senza connessione

Gli strati possono offrire due tipi diversi di servizio a quelli sovrastanti: orientati alla connessione oppure senza connessione. In questo paragrafo si esamineranno le differenze tra i due tipi. Un **servizio orientato alla connessione** assomiglia al sistema telefonico. Per parlare con qualcuno si deve prendere il telefono, comporre il numero, parlare e poi riagganciare. Allo stesso modo, per usare un servizio di rete orientato alla connessione l'utente deve innanzi tutto stabilire una connessione, usarla e quindi rilasciarla. L'aspetto essenziale di una connessione è che funziona come un tubo: il trasmettitore vi spinge oggetti (bit) a una estremità e il ricevitore li prende dall'altra. Nella maggior parte dei casi l'ordine è conservato, quindi i bit arrivano nella sequenza con cui sono stati trasmessi.

In alcuni casi, quando si stabilisce una connessione il trasmettitore, il ricevitore e la subnet eseguono una **negoziazione** dei parametri da usare, come la massima dimensione di un messaggio, la qualità di servizio richiesta e altri elementi. Tipicamente un lato fa una proposta e il corrispondente la può accettare, rifiutare o proporre una controproposta.

Al contrario, un **servizio senza connessione** si comporta come la posta. Ogni messaggio (lettera) trasporta l'indirizzo completo del destinatario ed è instradato attraverso il sistema postale in modo indipendente dagli altri. Normalmente, quando si mandano due messaggi alla stessa destinazione, il primo inviato è anche il primo ad arrivare; ma è possibile che incontri un ritardo, e quindi arrivi dopo il secondo.

Ogni servizio si può classificare in base alla **qualità del servizio** (*quality of service*). Alcuni servizi sono affidabili, nel senso che non perdono mai dati. Di solito, un servizio affidabile è implementato in modo tale che il ricevitore confermi il ricevimento di ciascun messaggio e così il trasmettitore sappia che è arrivato. Il processo di conferma introduce appesantimenti e ritardi, che spesso valgono la pena ma a volte possono risultare indesiderabili. Una tipica situazione dove è appropriato un servizio orientato alla connessione è il trasferimento dei file. Il proprietario del file vuole essere sicuro che tutti i bit sono arrivati senza errori e nello stesso ordine con cui sono stati inviati. Pochissimi utenti preferirebbero un servizio che di tanto in tanto perde o scambia qualche bit, anche se fosse molto più veloce. Esistono due piccole varianti del servizio affidabile orientato alla connessione, chiamate *message sequence* e *byte stream*. Nella prima variante, i confini dei messaggi vengono preservati. Quando si mandano due messaggi da 1.024 byte, arrivano come due distinti messaggi da 1.024 byte ciascuno, e mai come singolo messaggio da 2.048 byte. Nella seconda variante, la connessione è un semplice flusso di byte, senza divisioni tra i messaggi. Quando 2.048 byte arrivano al ricevitore, non è possibile sapere se erano stati spediti come singolo messaggio da 2.048 byte, due messaggi da 1.024 byte o magari 2.048 messaggi da 1 byte. Se le pagine di un libro vengono spedite attraverso la rete a una macchina tipografica come messaggi separati, può essere importante conservare i confini tra i messaggi. Viceversa, quando un utente si connette a un server remoto è sufficiente un flusso di byte tra il computer dell'utente e il server. I confini tra i messaggi non hanno importanza. Per quanto detto, in alcune applicazioni il ritardo di trasferimento introdotto dalle conferme di ricezione è inaccettabile. Una di queste applicazioni è il traffico vocale digitalizzato.

Per un utente telefonico è meglio ascoltare un po' di rumore di linea di tanto in tanto che percepire un ritardo nell'attesa delle conferme. Allo stesso modo, quando si trasmette una videoconferenza non è un problema avere qualche pixel sbagliato, ma sarebbe irritante vedere l'immagine che procede a scatti quando il flusso si arresta per correggere gli errori. Non tutte le applicazioni hanno bisogno di una connessione. Per esempio, man mano che cresce l'uso della posta elettronica aumenta anche la "spazzatura elettronica". Non è neppure essenziale un'affidabilità di consegna del 100%, specialmente se è più costosa. Tutto ciò che serve è un modo per spedire un singolo messaggio che abbia un'elevata probabilità di arrivare a destinazione, senza garanzie. Un servizio senza connessione non affidabile (cioè privo di conferma) spesso viene chiamato servizio **datagram**, per analogia con il servizio telegrafico (*telegram service*), che non restituisce una conferma a chi trasmette.

In altre situazioni la comodità di non dover stabilire una connessione per spedire un breve messaggio è desiderabile, ma l'affidabilità è essenziale. In questi casi è utilizzabile il servizio **datagram con conferma** (*acknowledged datagram service*). È come spedire una raccomandata con ricevuta di ritorno. Quando la ricevuta torna indietro, il mittente è assolutamente certo che la lettera è stata recapitata al destinatario desiderato e non è stata persa.

Un altro servizio è chiamato **request-reply**. In questo servizio la sorgente trasmette un singolo datagramma che contiene una richiesta; il messaggio ricevuto dal mittente rappresenta la risposta. Per esempio, una ricerca sull'enciclopedia per sapere dove si parla l'Uighur cade in questa categoria. La Figura 1.16 riepiloga i tipi di servizio discussi.

	Servizio	Esempio
Orientato alla connessione	Flusso affidabile di messaggi	Sequenza di pagine
	Flusso affidabile di byte	Login remoto
Senza connessione	Connessione inaffidabile	Voce digitalizzata
	Datagram inaffidabile	Posta elettronica spazzatura
	Datagram con conferma	Posta raccomandata
	Request-reply	Interrogazione di un database

Figura 1.16. Sei diversi tipi di servizio.

A prima vista il concetto di comunicazione inaffidabile può confondere. Dopotutto, chi mai preferirebbe una comunicazione inaffidabile a una affidabile? Prima di tutto, la comunicazione affidabile (nel senso visto, e cioè con conferma) potrebbe non essere disponibile. Per esempio, Ethernet non fornisce una comunicazione affidabile; i pacchetti si possono corrompere di tanto in tanto mentre sono in transito. È compito dei protocolli degli strati superiori prenderci carico del problema. Secondo, i ritardi necessari per fornire un servizio affidabile potrebbero essere inaccettabili, specialmente nelle applicazioni in tempo reale come quelle multimediali. Per questi motivi le comunicazioni affidabili e inaffidabili coesistono.

1.3.4 Primitive di servizio

Un servizio è formalmente specificato da un insieme di **primitive** (operazioni) che i processi utenti hanno a disposizione per accedere al servizio.

Queste primitive istruiscono il servizio a eseguire alcune azioni o riferire quelle prese da entità di pari strato.

Se la pila dei protocolli si trova nel sistema operativo, come spesso accade, le primitive sono generalmente chiamate di sistema.

Queste chiamate causano la commutazione in modalità kernel, che a sua volta fa prendere il controllo del computer al sistema operativo per spedire i pacchetti necessari [NdR - La pila di protocolli è implementata in genere dal sistema operativo, ma non è detto che le primitive dei servizi dei vari livelli siano tutte chiamate di sistema. In particolare solo i livelli a cui un processo utente può accedere direttamente hanno primitive realizzate da chiamate di sistema. Inoltre, ogni servizio utilizzato da un sovrastante livello all'interno del kernel deve poter essere invocato per mezzo di una primitiva che non è una chiamata di sistema, ma per esempio potrebbe essere una semplice funzione del kernel].

L'insieme delle primitive disponibili dipende dalla natura del servizio che viene offerto. Le primitive per un servizio orientato alla connessione sono diverse da quelle di un servizio senza connessione.

Come esempio minimo delle primitive di servizio che possono essere fornite per implementare un flusso di byte affidabile in un ambiente client-server, esaminiamo le primitive elencate nella Figura 1.17.

Primitiva	Significato
LISTEN	Attesa bloccante di una connessione in arrivo
CONNECT	Stabilisce una connessione con un pari in attesa
RECEIVE	Attesa bloccante per un messaggio in arrivo
SEND	Manda un messaggio al pari
DISCONNECT	Termina una connessione

Figura 1.17. Cinque primitive di servizio per implementare un semplice servizio orientato alla connessione.

Queste primitive si possono usare come segue. Prima di tutto, il server esegue LISTEN per indicare che è pronto ad accettare connessioni in arrivo. Un modo comune per implementare LISTEN consiste nel renderlo una chiamata di sistema bloccante. Dopo aver eseguito la primitiva, il processo server resta bloccato fino a quando appare una richiesta di connessione. Successivamente, il processo client esegue CONNECT per stabilire una connessione con il server. La chiamata a CONNECT deve specificare a chi connettersi, quindi può avere un parametro per indicare l'indirizzo del server. A questo punto il sistema operativo di solito manda un pacchetto al pari (*peer*) chiedendogli di connettersi, come è mostrato da (1) nella Figura 1.18. Il processo client resta sospeso fino a quando arriva una risposta.

Quando il pacchetto arriva al server, viene elaborato dal relativo sistema operativo. Appena il sistema vede che il pacchetto chiede una connessione, verifica se esiste un ascoltatore. Se è così fa due cose: sblocca l'ascoltatore e rimanda indietro una conferma (2). L'arrivo di questa conferma sblocca il client. Ora client e server sono entrambi in esecuzione e hanno stabilito la connessione. È importante osservare che la conferma (2) è generata dal codice del protocollo stesso, e non in risposta di una primitiva a livello utente. Se arriva una richiesta di connessione e non ci sono ascoltatori, il risultato è indefinito. In alcuni sistemi il pacchetto può essere messo in coda per un breve periodo, in attesa di un LISTEN.

L'ovvia analogia tra questo protocollo e la realtà è un utente (client) che chiama un addetto commerciale del proprio fornitore. Il commerciale inizia la sua attività stando vicino al telefono per rispondere nel caso in cui squilli. Il cliente fa la chiamata, e quando l'addetto commerciale prende il telefono la comunicazione è stabilita.



Figura 1.18. Pacchetti spediti in una semplice interazione client-server in una rete orientata alla connessione.

Il passo successivo per il server è quello di eseguire RECEIVE per prepararsi ad accettare la prima richiesta. Normalmente il server lo fa immediatamente dopo essere stato sbloccato da LISTEN, prima che la conferma (acknowledgment) ritorni al client. La chiamata a RECEIVE blocca il server.

Ora il client esegue SEND per trasmettere le sue richieste (3), seguito da RECEIVE per ottenere la risposta.

L'arrivo del pacchetto di richiesta al computer server sblocca il processo server, che può elaborare la richiesta. Dopo che ha fatto il lavoro, usa SEND per restituire la risposta al client (4). L'arrivo di questo pacchetto sblocca il client, che ora può esaminare la risposta. Se il client ha richieste aggiuntive, può farle ora. Se ha finito, può usare DISCONNECT per terminare la connessione.

Di solito DISCONNECT iniziale è una chiamata bloccante, che sospende il client e spegne un pacchetto al server indicando che la connessione non è più necessaria (5). Quando il server riceve il pacchetto, emette a sua volta un DISCONNECT dando conferma al client e rilasciando la connessione. Quando il pacchetto del server (6) ritorna alla macchina client, il processo client è rilasciato e la connessione interrotta. In due parole, questo è il modo in cui funziona la comunicazione orientata alla connessione.

Naturalmente la vita non è così semplice. Molte cose possono andare storte.

Le temporizzazioni potrebbero essere sbagliate (per esempio CONNECT viene fatto prima di LISTEN), i pacchetti si possono perdere, e molto altro. In seguito si esamineranno questi problemi con maggior dettaglio, ma per il momento la Figura 1.18 riassume brevemente come può funzionare la comunicazione client-server su una rete orientata alla connessione.

Dato che per completare questo protocollo servono sei pacchetti, ci si può chiedere perché non si usa invece un protocollo senza connessione. La risposta è che in un mondo perfetto si potrebbe fare, nel qual caso basterebbero solo due pacchetti: uno per la richiesta e uno per la risposta. Tuttavia di fronte alla possibilità di messaggi enormi in entrambe le direzioni (per esempio un file di alcuni megabyte), errori di trasmissione e pacchetti persi, la situazione cambia. Se la risposta consiste in centinaia di pacchetti, alcuni dei quali possono andare persi nella trasmissione, il client come può sapere se qualche pezzo manca? Come può sapere se l'ultimo pacchetto che ha ricevuto è veramente l'ultimo inviato? Supponiamo che il client abbia chiesto un secondo file: come si può distinguere il pacchetto 1 del secondo file da un pacchetto 1 del primo file, perso in precedenza, che improvvisamente trova la strada per il client? In breve, nel mondo reale un semplice protocollo request-reply su una rete inaffidabile è spesso inadeguato. Nel Capitolo 3 si studierà in dettaglio un insieme di protocolli che risolve questo e altri problemi. Per il momento, basta dire che a volte è molto utile avere un flusso di byte tra processi che sia affidabile e rispetti la sequenza.

1.3.5 La relazione tra servizi e protocolli

Servizi e protocolli sono concetti distinti, ma vengono spesso confusi. La distinzione è però così importante che è necessario evidenziarla. Un *servizio* è un insieme di primitive (operazioni) che uno strato offre a quello superiore. Il servizio definisce quali operazioni lo strato è in grado di offrire su richiesta dei suoi utenti, ma non dice nulla di come queste operazioni sono implementate. Un servizio è correlato all'interfaccia tra due strati, dove quello inferiore è il provider del servizio mentre quello superiore è l'utente.

Un *protocollo*, invece, è un insieme di regole che controllano il formato e il significato dei pacchetti, o messaggi scambiati tra le entità pari all'interno di uno strato. Le entità usano i protocolli per implementare le loro definizioni dei servizi. Sono libere di cambiare i loro protocolli, a patto di non cambiare il servizio visibile agli utenti. In questo modo, il servizio e il protocollo sono completamente disaccoppiati.

In altri termini, i servizi si riferiscono alle interfacce tra gli strati, come mostrato nella Figura 1.19. I protocolli, invece, riguardano i pacchetti scambiati tra entità di pari strato che risiedono su computer diversi. È importante non confondere i due concetti.

Vale la pena fare un'analogia con i linguaggi di programmazione. Un servizio è come un tipo dati astratto o un oggetto (in un linguaggio orientato agli oggetti). Definisce operazioni che si possono fare sull'oggetto ma non il modo in cui queste operazioni sono implementate. Un protocollo si riferisce all'*implementazione* del servizio, e pertanto non è visibile agli utenti del servizio.

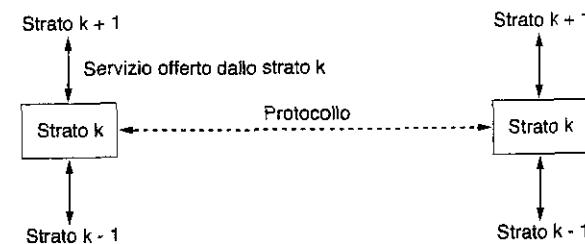


Figura 1.19. La relazione tra servizio e protocollo.

Molti vecchi protocolli non facevano distinzione tra servizio e protocollo. Un tipico strato poteva avere una primitiva di servizio SEND PACKET dove l'utente doveva fornire il puntatore a un pacchetto completamente assemblato. Questa soluzione implicava che tutte le modifiche a un protocollo diventavano immediatamente visibili agli utenti. La maggior parte dei progettisti di rete oggi considera queste tecniche un errore grossolano.

1.4 Modelli di riferimento

Dopo aver discusso degli strati di rete in senso astratto, è il momento di guardare alcuni esempi. Nei prossimi due paragrafi esamineremo due importanti architetture di rete, il modello di riferimento OSI e il modello TCP/IP. Anche se i *protocolli* associati al modello OSI ormai sono in disuso, il *modello* in sé ha valore generale ed è ancora valido, e le caratteristiche discusse per ogni strato sono ancora molto importanti. Il modello TCP/IP ha caratteristiche opposte: il modello in sé è poco utilizzabile, ma i protocolli sono largamente impiegati; per questo motivo entrambi saranno esaminati in dettaglio. A volte s'impone più dai fallimenti che dai successi.

1.4.1 Il modello di riferimento OSI

Il modello OSI (eccetto il mezzo fisico) è mostrato nella Figura 1.20. Questo modello si fonda su una proposta sviluppata dall'*International Standards Organization* (ISO) come primo passo verso la standardizzazione internazionale dei protocolli impiegati nei diversi strati (Day e Zimmermann, 1983), ed è stato revisionato nel 1995 (Day, 1995). Si chiama modello di riferimento ISO OSI (*Open System Interconnection*) perché riguarda la connessione di sistemi aperti, cioè sistemi che sono "aperti" verso la comunicazione con altri; per brevità verrà chiamato in seguito modello OSI.

Il modello OSI ha sette strati. I principi che sono stati applicati per arrivare ai sette strati si possono brevemente riassumere come segue:

1. si deve creare uno strato quando è richiesta un'astrazione diversa
2. ogni strato deve svolgere una funzione ben definita
3. la funzione di ogni strato va scelta con uno sguardo rivolto alla definizione di protocolli internazionali
4. i confini degli strati vanno scelti per minimizzare il flusso d'informazioni attraverso le interfacce
5. il numero di strati deve bastare per evitare la necessità di radunare funzioni distinte nello stesso strato, ma essere abbastanza piccolo da rendere l'architettura attuabile.

In seguito gli strati del modello saranno discussi partendo da quello più basso. Osservare che il modello OSI in sé non è un'architettura di rete, perché non specifica quali sono esattamente i servizi e i protocolli da usare in ciascuno strato; si limita infatti a definire ciò che ogni strato deve compiere. Tuttavia, ISO ha prodotto anche standard per ciascuno strato, benché non facciano parte del modello di riferimento: ognuno è stato pubblicato come standard internazionale distinto.

Lo strato fisico

Lo **strato fisico** si occupa della trasmissione di bit grezzi sul canale di comunicazione. I requisiti di progetto devono assicurare che ogni bit trasmesso con valore 1 sia ricevuto ancora con valore 1, e non con valore 0. Problemi tipici riguardano quanti Volt bisogna usare per rappresentare un 1 e quanti per uno 0, quanti nanosecondi deve durare un bit, se la trasmissione può avvenire simultaneamente in entrambe le direzioni, come si stabilisce la connessione iniziale e come viene abbattuta quando entrambe le parti hanno terminato, quanti contatti deve avere il connettore di rete e che funzione va assegnata a ciascuno. Le specifiche riguardano per lo più interfacce meccaniche o elettriche e temporizzazioni, oltre che il mezzo di trasmissione che si trova sotto allo strato fisico.

Lo strato data link

Il compito principale dello **strato data link** consiste nel cercare di rilevare, per quanto possibile, gli errori di trasmissione così da evitare di trasmettere questi errori riconosciuti al livello superiore. L'obiettivo è raggiunto forzando il trasmettitore a suddividere i dati d'ingresso in **data frame** (tipicamente qualche centinaio o migliaio di byte) che vengono trasmessi sequenzialmente. Se il servizio è affidabile, il ricevitore conferma la corretta ricezione di ciascun frame rimandando indietro un **acknowledgment frame**.

Un altro problema che nasce nello strato data link (e nella maggior parte degli strati superiori) riguarda il modo per evitare che un trasmettitore veloce saturi le possibilità di un ricevitore lento. Spesso occorre un meccanismo per regolare il traffico, che deve informare il trasmettitore sulla quantità di spazio buffer libero nel ricevitore. Spesso il controllo di flusso e quello di errore sono integrati tra loro.

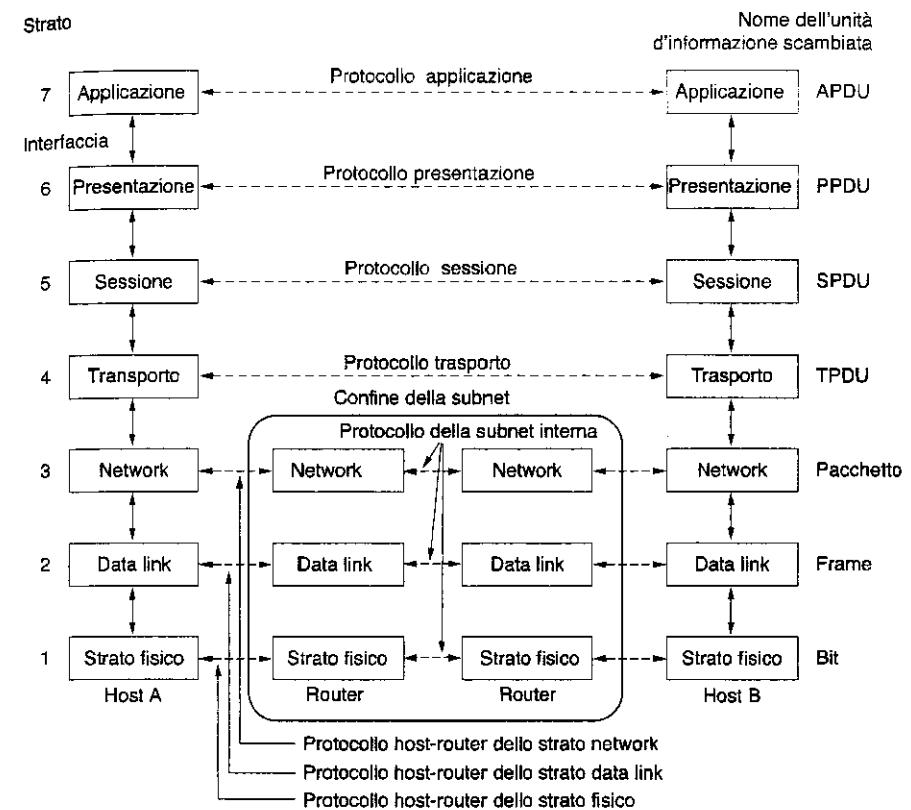


Figura 1.20. Il modello di riferimento OSI.

Le reti broadcast hanno un problema in più nello strato data link: come controllare l'accesso al canale condiviso. Di questo problema si occupa uno speciale sottoinsieme dello strato data link, chiamato **medium access control**.

Lo strato network

Lo **strato network** controlla il funzionamento della subnet. Un problema chiave riguarda la modalità con cui i pacchetti sono inoltrati dalla sorgente alla destinazione. L'inoltro si può basare su tabelle statiche che sono "cablate" dentro la rete e raramente modificate. Si può anche stabilire all'inizio di ogni conversazione che può essere per esempio una sessione terminale (login su un computer remoto). Infine, potrebbe essere altamente dinamico, determinato per ogni pacchetto in modo da riflettere lo stato corrente del carico della rete.

Quando nella subnet sono presenti contemporaneamente troppi pacchetti, vanno a interferire l'un l'altro formando colli di bottiglia. Anche il controllo di queste congestioni spetta allo strato network. Più in generale, la qualità del servizio offerto (ritardo, tempo di transito, jitter, ecc.) è un problema dello strato network.

Quando un pacchetto deve viaggiare da una rete all'altra per arrivare a destinazione possono nascere molti problemi. L'indirizzamento usato dalla seconda rete potrebbe essere diverso da quello della prima, la seconda rete potrebbe rifiutare il pacchetto perché troppo grosso, i protocolli potrebbero essere diversi, e così via. È compito dello strato network risolvere questi problemi per consentire la comunicazione tra reti eterogenee.

Nelle reti broadcast il problema dell'instradamento è semplice, quindi lo strato network è solitamente molto semplice o anche assente.

Lo strato trasporto

La funzione essenziale dello **strato trasporto** è quella di accettare dati dallo strato superiore, dividerli in unità più piccole quando necessario, passarle allo strato network e assicurarsi che tutti i pezzi arrivino correttamente all'altra estremità. Inoltre, tutto ciò va fatto in un modo efficiente che isoli gli strati superiori dalle inevitabili variazioni delle tecnologie hardware.

Lo strato trasporto stabilisce inoltre che tipo di servizio offre allo strato sessione e, in definitiva, agli utenti della rete. Il tipo di connessione trasporto più comune è un canale punto-punto privo di errori che consegna messaggi o byte nello stesso ordine usato per la trasmissione. Altri tipi di servizio trasporto sono lo spostamento di singoli messaggi senza garanzia sull'ordine di consegna, e il broadcast di messaggi a destinazioni multiple. Il tipo di servizio è fissato all'instaurarsi della connessione. Come corollario, è impossibile ottenere un canale privo di errori; ciò che s'intende realmente con questa espressione è un tasso di errore così basso da poter essere trascurato nella pratica.

Lo strato trasporto copre tutto il percorso da sorgente a destinazione; in altri termini, un programma sul computer sorgente instaura una conversazione con un programma corrispondente sul computer destinatario, utilizzando intestazioni dei messaggi e messaggi di controllo. Negli strati inferiori i protocolli riguardano la comunicazione tra ciascun computer e i vicini immediati, e non tra i computer sorgente e destinatario, che possono essere separati da molti router. La Figura 1.20 mostra le differenze tra gli strati da 1 a 3 (che sono concatenati) e gli strati da 4 a 7 (che vanno da bordo a bordo).

Lo strato sessione

Lo strato sessione permette agli utenti su computer diversi di stabilire tra loro una **sessione**. Le sessioni offrono diversi servizi, tra cui: **controllo del dialogo** (tenere traccia di quando è il turno di trasmettere e quando di ricevere), **gestione dei token** (evitare che le

due parti tentino la stessa operazione critica al medesimo istante) e **sincronizzazione** (supervisionare una lunga trasmissione per consentire la sua ripresa dal punto in cui si è interrotta a causa di un crash).

Lo strato presentazione

A differenza degli strati inferiori, che per lo più si occupano di spostare bit, lo **strato presentazione** si occupa della sintassi e della semantica dell'informazione trasmessa. Per consentire la comunicazione tra computer con differenti rappresentazioni dei dati, le strutture dati da scambiare si possono definire in modo astratto, assieme a una codifica standard usata "sul filo". Lo strato presentazione gestisce queste strutture dati astratte e consente lo scambio e la definizione di strutture dati di livello superiore (per esempio transazioni bancarie).

Lo strato applicazione

Lo strato applicazione comprende una varietà di protocolli comunemente richiesti dagli utenti. Un protocollo applicativo largamente usato è **HTTP (HyperText Transfer Protocol)**, che è la base del World Wide Web. Quando un browser richiede una pagina Web, invia al server il nome della pagina desiderata usando HTTP, quindi il server risponde inviandola. Altri protocolli applicativi si usano per trasferimento file, posta elettronica e news.

1.4.2 Il modello di riferimento TCP/IP

Lasciamo da parte il modello di riferimento OSI per analizzare il modello di riferimento del progenitore di tutte le reti di computer geografiche, ARPANET, e il suo successore Internet. Una breve storia di ARPANET sarà fornita più avanti, ma è utile menzionare alcuni dei suoi aspetti chiave. ARPANET fu una rete sperimentale sponsorizzata dal Dod (dipartimento della difesa USA). Connottava centinaia di università e installazioni governative tramite linee telefoniche affittate. Quando furono aggiunte reti satellitari e via radio i protocolli esistenti incontrarono difficoltà nel collegamento, facendo nascere l'esigenza di una nuova architettura di riferimento. Per questo motivo la capacità di collegare tra loro più reti in modo semplice è stata sin dall'inizio uno dei principali obiettivi di progetto. Questa architettura è poi diventata nota con il nome di **modello di riferimento TCP/IP**, dal nome dei suoi due protocolli principali. È stata definita per la prima volta in (Cerf e Kahn, 1974); una visione più tarda è in (Leiner et al., 1985). La filosofia di progetto che sta dietro il modello è discussa in (Clark, 1988).

Il dipartimento della difesa aveva il timore che alcuni dei suoi preziosi host, router e gateway tra reti potessero essere ridotti in briciole improvvisamente, perciò un altro obiettivo principale era che la rete potesse sopravvivere alla perdita dell'hardware di subnet senza interrompere le conversazioni in corso. In altri termini, il dipartimento della difesa voleva

che le connessioni rimanessero intatte per tutto il tempo in cui il computer sorgente e quello destinazione fossero in funzione, anche se alcuni dei computer o delle linee di trasmissione tra essi fossero andati improvvisamente in avaria. Inoltre era necessaria un'architettura flessibile, poiché erano previste applicazioni con richieste divergenti, che spaziavano dal trasferimento di file alla trasmissione della voce in tempo reale.

Lo strato internet

Tutti questi requisiti hanno portato alla scelta di una rete a commutazione di pacchetto basata su uno strato internetwork senza connessione. Viene chiamato **strato internet**, ed è l'architrave su cui poggia l'intera architettura. Il suo scopo è quello di consentire agli host di mandare pacchetti in qualsiasi rete, e farli viaggiare in modo indipendente l'uno dall'altro fino alla destinazione (che magari è su una rete diversa). Potrebbero persino arrivare con un ordine diverso da quello con cui sono stati spediti, e in questo caso (se è richiesta una consegna in sequenza) è compito degli strati superiori riordinarli. Il termine "internet" viene usato in senso generico in questo contesto, anche se questo strato è presente nella Internet.

Si può fare un'analogia con il servizio postale ordinario. Una persona può imbucare una sequenza di lettere con destinatari internazionali e, con un po'di fortuna, la maggior parte sarà consegnata nei paesi di destinazione agli indirizzi corrispondenti. Probabilmente le lettere viaggeranno attraverso uno o più centri di smistamento internazionali, ma ciò è invisibile all'utente; resta nascosto anche il fatto che ogni nazione (cioè ogni rete) ha propri francobolli, dimensioni normalizzate per le buste e regole per la consegna della posta. Lo strato internet definisce un formato ufficiale per i pacchetti e un protocollo chiamato **IP (Internet Protocol)**. Lo scopo dello strato internet è quello di consegnare i pacchetti IP alla destinazione corretta. L'instradamento dei pacchetti è chiaramente il problema più importante, assieme alla necessità di garantire l'assenza di congestioni. Per questi motivi è ragionevole dire che lo strato TCP/IP internet ha funzionalità simili allo strato network OSI. La Figura 1.21 mostra questa corrispondenza.

Lo strato trasporto

Nel modello TCP/IP lo strato superiore a quello internet viene attualmente chiamato **strato trasporto (transport)**. È progettato per consentire la comunicazione tra entità pari degli host sorgente e destinazione, come nello strato trasporto OSI. In questo strato sono stati definiti due protocolli di trasporto end-to-end: il primo, **TCP (Transmission Control Protocol)**, è un protocollo affidabile orientato alla connessione che permette a un flusso di byte emessi da un computer di raggiungere senza errori qualsiasi altro computer sulla Internet. Suddivide il flusso di byte entrante in messaggi discreti e passa ciascun frammento allo strato internet. Nella destinazione, il processo TCP ricevente ricomponne il messaggio ricevuto per formare il flusso di uscita. TCP gestisce anche il controllo di flusso, per garantire che una sorgente veloce non possa sommergere un ricevitore lento con una quantità di messaggi superiore a quelli che sa gestire.

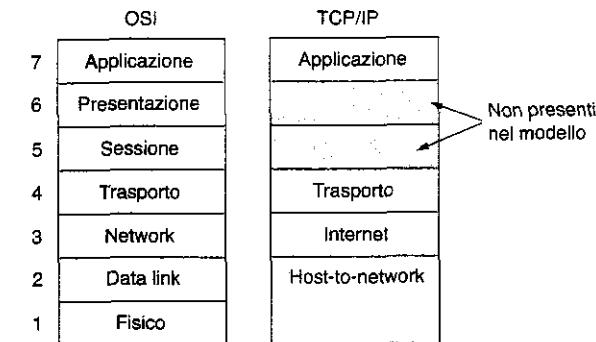


Figura 1.21. Il modello di riferimento TCP/IP.

Il secondo protocollo di questo strato, **UDP (User Datagram Protocol)**, è un protocollo inaffidabile senza connessione per le applicazioni che non vogliono la garanzia di ordinamento e il controllo di flusso di TCP, ma preferiscono gestire queste funzioni in modo autonomo. È inoltre largamente impiegato per le query client-server singole di tipo domanda-risposta, e dalle applicazioni dove la consegna rapida è più importante dell'accuratezza, come la trasmissione di voce e filmati. La relazione tra IP, TCP e UDP è mostrata nella Figura 1.22. Dopo lo sviluppo del modello, IP è stato implementato in molte altre reti.

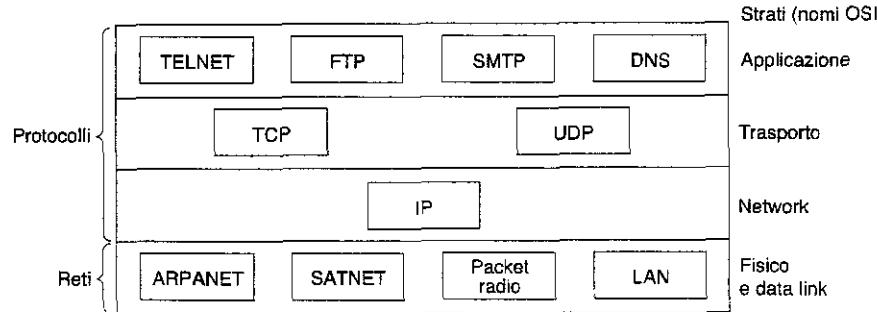


Figura 1.22. Protocolli e reti nel modello iniziale TCP/IP.

Lo strato applicazione

Il modello TCP/IP non ha strati sessione e presentazione, poiché per essi non fu percepito nessun bisogno. L'esperienza del modello OSI ha dimostrato la fondatezza di questa scelta: nella maggior parte dei casi servono a poco.

Sopra lo strato trasporto c'è lo **strato applicazione**, che contiene tutti i protocolli di livello superiore. I primi gestivano un terminale virtuale (TELNET), lo scambio dei file (FTP)

e la posta elettronica (SMTP), come mostrato nella Figura 1.22. Il protocollo di emulazione terminale consente all'utente di un computer di eseguire un login su un computer remoto per lavorare su di esso. Il protocollo di trasferimento dei file è un mezzo efficiente per spostare dati da un computer a un altro: in origine la posta elettronica era solo una specie di trasferimento file, prima che venisse sviluppato un protocollo specializzato (SMTP). Nel corso degli anni a questi si sono aggiunti molti altri protocolli: *Domain Name System* (DNS) che fa corrispondere i nomi degli host ai loro indirizzi di rete; NNTP, il protocollo per spostare sulla rete i messaggi dei gruppi di discussione USENET; HTTP, il protocollo per prelevare pagine sul World Wide Web, e altri ancora.

Lo strato host-to-network

Sotto lo strato internet c'è un grande vuoto. Il modello di riferimento TCP/IP non dice molto su quanto accade in questo territorio, limitandosi a segnalare che l'host deve collegarsi alla rete usando qualche protocollo che gli permetta di spedire pacchetti IP. Questo protocollo non è definito e varia da host a host e da rete a rete. Libri e articoli sul modello TCP/IP lo prendono raramente in esame.

1.4.3 Confronto tra i modelli di riferimento OSI e TCP/IP

I modelli di riferimento OSI e TCP/IP hanno molto in comune. Sono entrambi basati sul concetto di pila (*stack*) di protocolli indipendenti, e la funzione degli strati è grosso modo simile. Per esempio, in entrambi i modelli gli strati di livello pari o superiore a quello trasporto forniscono ai processi che vogliono comunicare un servizio di trasmissione end-to-end e indipendente dalla rete; formano cioè il mezzo di trasporto. Di nuovo, in entrambi i modelli gli strati che stanno sopra a quello di trasporto ne rappresentano gli utenti e sono orientati alle applicazioni.

Nonostante queste similitudini fondamentali, i due modelli di riferimento hanno molte differenze che esamineremo in questo paragrafo. È importante notare che il confronto è fatto tra *modelli di riferimento*, e non tra le corrispondenti *pile di protocolli*. I protocolli veri e propri saranno infatti discussi in seguito. Un intero libro dedicato al confronto tra TCP/IP e OSI è (Piscitello and Chapin, 1993).

Nel modello OSI sono presenti tre concetti essenziali:

1. servizi
2. interfacce
3. protocolli.

Probabilmente il contributo più grande del modello OSI è la sua capacità di esplicitare la distinzione tra questi concetti. Ogni strato offre un servizio a quello che lo sovrasta; la definizione del *servizio* descrive ciò che fa lo strato, e non le modalità d'accesso da parte delle entità sovrastanti o quelle di funzionamento. Definisce la semantica dello strato.

L'*interfaccia* di uno strato spiega le modalità di accesso ai processi sovrastanti. Specifica quali sono i parametri e i risultati, ma non dice nulla delle modalità di funzionamento interno.

Infine, l'essenza fondamentale di uno strato sono i *protocolli pari* (*peer protocols*) che vengono impiegati al suo interno. Lo strato può usare i protocolli che preferisce, se portano ai risultati desiderati (cioè svolgono i servizi offerti), e li può cambiare senza disturbare il software degli strati superiori.

Queste idee si adeguano alla perfezione ai moderni concetti sulla programmazione orientata agli oggetti. Un oggetto, come uno strato, ha un insieme di metodi (operazioni) che sono invocati dai processi esterni all'oggetto. La semantica di questi metodi definisce l'insieme di servizi offerti dall'oggetto, mentre i parametri dei metodi e i risultati formano la sua interfaccia. Il codice interno all'oggetto è il suo protocollo, invisibile e ininfluente per ciò che si trova all'esterno dell'oggetto.

Il modello TCP/IP in origine non faceva una netta distinzione tra servizio, interfaccia e protocollo, anche se molti hanno tentato di calarvi sopra questi concetti per renderlo più simile al modello OSI. Per esempio, gli unici veri servizi offerti dallo strato internet sono SEND IP PACKET e RECEIVE IP PACKET.

La conseguenza è che nel modello OSI i protocolli sono nascosti meglio che nel modello TCP/IP, e si possono sostituire con relativa facilità all'evolvere della tecnologia. La capacità di operare cambiamenti di questa portata è uno degli scopi principali dell'esistenza dei protocolli stratificati.

Il modello di riferimento OSI è stato concepito prima d'inventare i protocolli corrispondenti, e ciò significa che il modello non è orientato verso un insieme specifico di protocolli, cosa che lo rende decisamente generico. L'altra faccia della medaglia è che i progettisti non avevano molta esperienza, e neppure buone idee sulle funzionalità da inserire in ciascuno strato.

Per esempio, lo strato data link in origine copriva solo il caso di reti punto-punto, e quando nacquero le reti broadcast fu necessario incastrare nel modello un nuovo sottostrato. Quando s'iniziarono a costruire vere reti usando il modello OSI e i suoi protocolli si scoprì che non soddisfacevano le specifiche di servizio richieste, quindi fu necessario integrare il modello con sottostrati di convergenza per appianare le differenze. Per colmare la misura, il comitato originariamente supponeva che ogni paese avesse una sola rete, gestita dal governo e basata sui protocolli OSI, quindi non venne dato peso all'internetworking. Per farla breve, le cose non sono andate in questo modo.

Nel caso del TCP/IP si è verificato tutto l'opposto: prima sono arrivati i protocolli, e poi fu realizzato un modello che rappresentava una semplice descrizione dei protocolli esistenti. Non nacque nessun problema per far aderire i protocolli al modello, infatti corrispondevano perfettamente. L'unico problema era che il *modello* non corrispondeva a nessun'altra pila di protocolli e quindi era poco utile per descrivere altre reti non basate su TCP/IP.

Passando dai problemi filosofici a quelli più specifici, una differenza ovvia tra i due modelli è il numero di strati: il modello OSI ne ha sette, e TCP/IP quattro. Entrambi hanno strati (inter)network, trasporto e applicazione, ma gli altri sono diversi.

Un'altra differenza sta nella modalità di comunicazione, orientata oppure no alla connessione. Il modello OSI supporta nello strato network entrambi i tipi di comunicazione, ma nello strato di trasporto (che è quello che conta, perché il servizio di trasporto è visibile agli utenti) supporta solo la comunicazione orientata alla connessione. Il modello TCP/IP ha solo una modalità nello strato network (senza connessione), ma supporta entrambe in quello di trasporto offrendo così agli utenti una vera scelta, di grande importanza per i semplici protocolli richiesta-risposta.

1.4.4 Critica del modello e dei protocolli OSI

Le imperfezioni sono presenti sia nei modelli e protocolli OSI, sia in quelli TCP/IP, e per questo motivo entrambi sono stati bersaglio di critiche accese che esamineremo in questo paragrafo e nel successivo, iniziando da OSI.

Quando fu pubblicata la seconda edizione di questo libro (1989) molti esperti del settore erano dell'opinione che il modello OSI e i corrispondenti protocolli avrebbero dominato la scena mondiale, spazzando via tutto il resto. Perché non è successo? È utile gettare uno sguardo indietro per capire ciò che ha decretato la sconfitta di OSI, e cioè:

1. poca tempestività
2. tecnologia scadente
3. implementazioni carenti
4. incapacità politica.

Poca tempestività

Iniziamo dal primo motivo: la pessima scelta dei tempi. Per il successo di uno standard è d'importanza vitale l'epoca in cui è messo a punto. David Clark del M.I.T. ha una teoria sugli standard che chiama *apocalisse dei due elefanti*, illustrata nella Figura 1.23.

Questa figura mostra l'entità dell'attività che circonda un nuovo argomento. Quando viene scoperto c'è un brulicare di attività di ricerca sotto forma di discussioni, articoli e incontri. Dopo un po' questa attività scema, le grandi aziende scoprono l'argomento e partono gli investimenti da milioni di dollari.

È essenziale che gli standard siano scritti nell'intervallo tra i due "elefanti". Se gli standard sono scritti troppo presto, prima del termine delle ricerche, l'argomento potrebbe essere ancora poco conosciuto e si ottengono standard scadenti. Se vengono scritti troppo tardi, un numero eccessivo di aziende può aver già fatto investimenti sostanziali nelle diverse modalità di realizzare le cose, quindi gli standard sono ignorati.

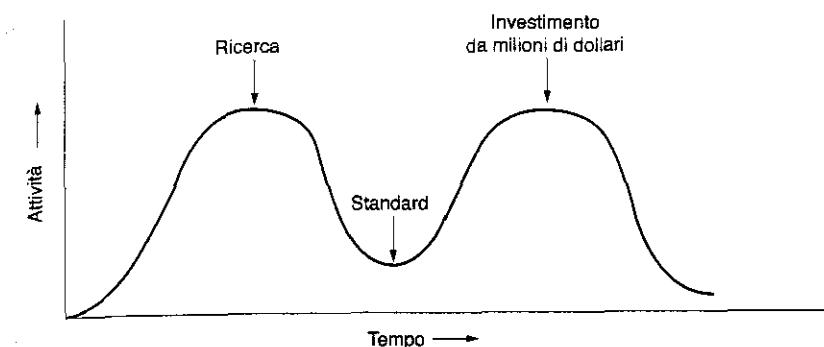


Figura 1.23. L'apocalisse dei due elefanti.

Se l'intervallo tra i due elefanti è molto breve (perché tutti sono ansiosi di partire), chi sviluppa gli standard può rimanere schiacciato.

Ormai è evidente che i protocolli standard OSI sono rimasti schiacciati. Quando sono apparsi i protocolli OSI, le controparti TCP/IP erano già largamente impiegate dalle università che facevano ricerca. Anche se l'onda degli investimenti milionari non era ancora arrivata, il mercato accademico era già abbastanza grande da spingere molti costruttori a offrire cautamente prodotti TCP/IP. Quando OSI è arrivato, non c'è stata disponibilità per supportare una seconda pila di protocolli (salvo quando il costruttore è stato costretto), quindi è mancata l'offerta iniziale. Mentre ogni produttore aspettava la prima mossa dell'altro, nessuno è partito e OSI è svanito.

Tecnologia scadente

Il secondo motivo che ha impedito il successo di OSI è che il modello e i protocolli hanno difetti. La scelta di sette strati fu più politica che tecnica: due strati (sessione e presentazione) sono quasi vuoti, mentre altri due (data link e network) sono sovraccarichi.

Il modello OSI con le corrispondenti definizioni di servizi e protocolli ha una complessità straordinaria: se vengono messe una sull'altra, le copie stampate degli standard formano una pila di carta alta quasi un metro; per giunta sono difficili da implementare e inefficienti. Viene in mente una freddura di Paul Mockapetris citata in (Rose, 1993):

Domanda: Cosa si ottiene incrociando un mafioso con uno standard internazionale?

Risposta: Qualcuno che ti fa un'offerta che non puoi capire.

OSI non ha solo il problema di essere incomprensibile: alcune funzioni come l'indirizzamento, il controllo di flusso e quello di errore riappaiono più e più volte in ogni strato. Saltzer et al. (1984), per esempio, ha segnalato che per essere efficace, il controllo di errore va fatto nello strato più alto, quindi ripeterlo in ogni strato è inutile e inefficiente.

Implementazioni carenti

Vista l'enorme complessità del modello e dei protocolli, non fu una sorpresa che le implementazioni iniziali fossero enormi, scomode e lente. Tutti quelli che ci tentarono rimasero scottati. Non ci volle molto per creare l'associazione mentale tra "OSI" e "scarsa qualità". I prodotti migliorarono col tempo, ma la fama rimase.

Per contrasto, una delle prime implementazioni di TCP/IP faceva parte dello UNIX di Berkeley ed era decisamente buona (e gratuita). Venne rapidamente adottata, sollecitando la formazione di una grande comunità di utenti, che stimolò miglioramenti destinati ad ampliare ancora di più la comunità. La spirale era in ascesa invece che in discesa.

Incapacità politica

In seguito all'implementazione iniziale, molti (soprattutto in ambiente accademico) ritenevano TCP/IP parte integrante di UNIX, e nell'ambiente accademico degli anni '80 UNIX era amato come la mamma e la torta di mele...

OSI, d'altro canto, era largamente ritenuto la creatura dei ministeri delle telecomunicazioni europei, della Comunità Europea, e più tardi del governo USA. L'impressione era in parte vera, ma evocò l'immagine di un gruppo di burocrati che cercavano di cacciare in gola ai poveri ricercatori e programmati uno standard mediocre, ostacolando gli sforzi per ottenere qualcosa che funziona davvero. C'è chi vide in questo sviluppo la stessa immagine degli annunci di metà degli anni '60 dove IBM proclamava che PL/I era il linguaggio del futuro, o quando il dipartimento della difesa USA più tardi correggeva il tiro affermando che invece era Ada.

1.4.5 Critica del modello di riferimento TCP/IP

Anche il modello e i protocolli TCP/IP hanno i loro problemi. Innanzitutto, il modello non distingue in modo chiaro i concetti di servizio, interfaccia e protocollo. Un buon approccio all'ingegneria del software impone di tenere distinte le specifiche dall'implementazione, cosa che OSI segue alla lettera al contrario di TCP/IP. La conseguenza è che il modello TCP/IP serve a poco se bisogna progettare reti basate su tecnologie nuove.

In secondo luogo, il modello TCP/IP è poco generale e inadatto per descrivere pile di protocolli diverse dal TCP/IP: per esempio, è assolutamente impossibile usare il modello TCP/IP per descrivere Bluetooth.

Terzo, lo strato host-to-network non è un vero e proprio strato, inteso con il significato che ha nel contesto dei protocolli stratificati. È piuttosto un'interfaccia tra gli strati network e data link.

La distinzione tra interfaccia e strato è cruciale, e non si dovrebbe trascurare.

Quarto, il modello TCP/IP non fa distinzione (né menziona) gli strati fisico e data link, che sono totalmente diversi tra loro: lo strato fisico ha a che fare con le caratteristiche trasmissive di cavi in rame, fibre ottiche e comunicazioni wireless. Il compito dello strato data link è quello di delimitare inizio e fine dei frame e spostarli da una parte all'altra con il livello di affidabilità richiesto. Un buon modello dovrebbe includerli come strati separati, al contrario di TCP/IP.

Infine, anche se i protocolli IP e TCP furono pensati con cura e implementati bene, molti altri protocolli risolvono problemi ad-hoc, e di solito erano prodotti da dottorandi che facevano tentativi finché non si stancavano. A questo punto le implementazioni del protocollo erano distribuite gratuitamente, diventando largamente utilizzate e molto radicate, quindi difficili da sostituire. Oggi alcune sono un po' imbarazzanti. Un esempio è il protocollo per l'emulazione terminale virtuale, TELNET, progettato per un terminale Teletype meccanico da dieci caratteri al secondo. Non sa nulla di mouse e interfacce utente grafiche, ma 25 anni dopo è ancora ampiamente usato.

Per concludere, nonostante i suoi problemi il *modello OSI* (eccetto gli strati sessione e presentazione) si è dimostrato eccezionalmente utile per discutere le reti di computer; al contrario i *protocolli OSI* non sono diventati popolari. Per TCP/IP è vero l'opposto: il *modello* praticamente non esiste, ma i *protocolli* sono molto usati. Poiché gli informatici amano avere la botte piena e la moglie ubriaca, in questo libro useremo un modello OSI modificato ma ci concentreremo soprattutto su TCP/IP e protocolli correlati, oltre che sui nuovi arrivati 802, SONET e Bluetooth. La cornice di riferimento di questo libro è quindi il modello ibrido della Figura 1.24.

5	Strato applicazione
4	Strato trasporto
3	Strato network
2	Strato data link
1	Strato fisico

Figura 1.24. Il modello di riferimento ibrido usato in questo libro.

1.5 Esempi di reti

Esistono molti tipi di rete, piccole e grandi, note e meno note, con diversi obiettivi, scopi e tecnologie. Nei paragrafi seguenti esamineremo qualche esempio per avere un'idea della varietà di situazioni coperte dall'argomento delle reti di computer.

Inizieremo con Internet, che probabilmente è la rete più nota, esaminando la sua storia, evoluzione e tecnologia. Successivamente esamineremo ATM, che spesso forma il cuore delle grandi reti (telefoniche).

Dal punto di vista tecnico è completamente diversa da Internet, creando un bel contrasto. Successivamente introdurremo Ethernet, lo standard per le reti locali. Infine, esamineremo IEEE 802.11 che è lo standard per LAN wireless.

1.5.1 Internet

Internet non è affatto una rete, ma una vasta raccolta di reti diverse che usano certi protocolli e offrono certi servizi comuni. È un sistema inconsueto, che non ha un progettista e non è controllato da nessuno. Per comprenderlo meglio, partiamo dall'inizio e vediamo come e perché si è sviluppato. Per una splendida storia di Internet raccomando il libro di John Naughton (2000), uno dei pochi testi che non è solo divertente da leggere, ma che ha anche 20 pagine di *ibid.'s* e *op.cit.* per i veri storici; una parte del materiale che segue è tratto da questo libro.

Ovviamente è stata scritta un'infinità di libri su Internet e i suoi protocolli. Per ulteriori informazioni vedere per esempio (Maufer, 1999).

ARPANET

La storia inizia nei tardi anni '50. Nel pieno della guerra fredda, il dipartimento della difesa USA commissiona una rete per comando e controllo che possa sopravvivere a una guerra nucleare. All'epoca tutte le comunicazioni militari usavano la rete telefonica pubblica, considerata vulnerabile: il motivo di questo timore si può capire osservando la Figura 1.25(a). I punti neri rappresentano le centrali di commutazione telefonica, ognuna delle quali era connessa a migliaia di telefoni.

A loro volta queste centrali erano collegate da centrali di commutazione di livello superiore per formare una gerarchia nazionale con poca ridondanza. La vulnerabilità del sistema stava nel fatto che la distruzione di poche centrali di alto livello avrebbe frammentato la rete in molte isole separate.

Attorno al 1960 il dipartimento della difesa assegnò il compito di trovare la soluzione alla RAND Corporation. Uno dei suoi impiegati, Paul Baran, sviluppò il progetto altamente distribuito e resistente ai guasti della Figura 1.25(b). Poiché il percorso tra due centrali di commutazione qualunque era diventato molto maggiore di quello che i segnali analogici potevano percorrere senza distorsioni, Baran propose di usare una tecnologia digitale a commutazione di pacchetto.

Baran scrisse diverse relazioni per il dipartimento della difesa descrivendo nei dettagli le sue idee, che furono apprezzate dagli ufficiali del Pentagono spingendoli a chiedere alla AT&T (che allora gestiva in monopolio la rete telefonica USA) di costruire un prototipo. AT&T rigettò in pieno le idee di Baran.

La più grande e potente corporation del mondo non avrebbe consentito a qualche giovane esaltato di darle lezioni su come costruire una rete telefonica. Dissero che la rete di Baran non si poteva costruire e il caso fu chiuso.

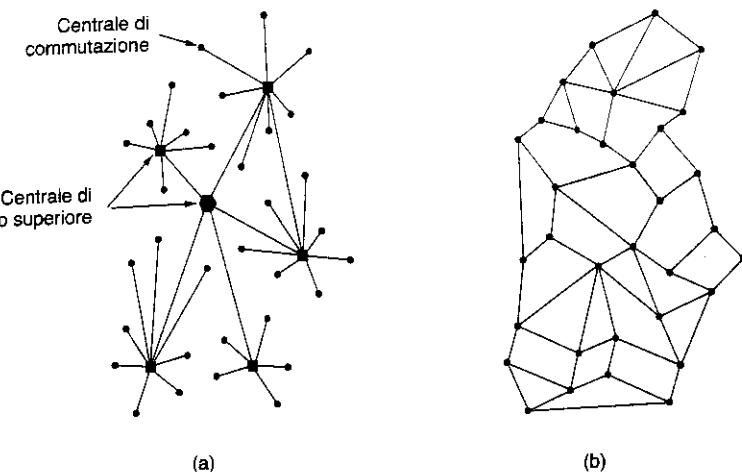


Figura 1.25. (a) Struttura della rete telefonica. (b) Il sistema distribuito a commutazione proposto da Baran.

Passò qualche anno e il dipartimento della difesa continuava a non avere una rete di comando e controllo migliore. Per capire ciò che successe in seguito, dobbiamo tornare all'ottobre 1957 quando l'Unione Sovietica batté gli USA nella corsa allo spazio con il lancio del primo satellite artificiale, Sputnik. Quando il presidente Eisenhower cercò di capire chi aveva "dormito", rimase sconvolto scoprendo come l'esercito, la marina e l'aviazione si contendevano il budget per la ricerca del Pentagono. La sua risposta immediata fu la creazione di una singola organizzazione per la ricerca, **ARPA** (*Advanced Research Projects Agency*). ARPA non aveva scienziati o laboratori; in effetti non aveva altro che un ufficio e un piccolo (per gli standard del Pentagono) budget. Il suo compito era erogare fondi e stipulare contratti con università e aziende che avevano idee promettenti.

ARPA spese i primi anni cercando di capire quale doveva essere la sua missione, ma nel 1967 l'attenzione dell'allora direttore di ARPA (Larry Roberts) si diresse verso le reti, e contattò molti esperti per decidere che cosa fare. Uno di essi, Wesley Clark, suggerì di costruire una subnet a commutazione di pacchetto, attribuendo a ogni host un proprio router come illustrato nella Figura 1.10.

Dopo lo scetticismo iniziale, Roberts sposò l'idea e presentò un articolo generico sull'argomento all'ACM SIGOPS (*Symposium on Operating System Principles*) che si tenne a Gatlinburg in Tennessee nel tardo 1967 (Roberts, 1967). Con sorpresa di Roberts, un altro articolo presentato durante la conferenza descriveva un sistema analogo che non solo era stato progettato, ma persino costruito sotto la direzione di Donald Davies al National Physical Laboratory in Gran Bretagna. Il sistema NPL non era a livello nazionale (connetteva solo alcuni computer nel campus di NPL), ma dimostrava che la tecnologia a com-

mutazione di pacchetto poteva funzionare e inoltre citava il lavoro precedente di Baran. Roberts tornò da Gatlinburg intenzionato a costruire ciò che successivamente divenne noto con il nome **ARPANET**.

La subnet sarebbe stata composta da minicomputer chiamati **IMP** (*Interface Message Processors*) collegati da linee di trasmissione a 56 kbps. Per raggiungere un'alta affidabilità ogni IMP doveva essere collegato ad almeno due altre IMP; la subnet era basata sui datagrammi, così in caso di distruzione di alcune linee e IMP i messaggi sarebbero stati automaticamente instradati su percorsi alternativi.

Ogni nodo della rete doveva consistere in un IMP e un host, posti nella stessa stanza e collegati da un corto cavo. Un host avrebbe potuto mandare messaggi con lunghezza massima di 8.063 bit al suo IMP, che li avrebbe suddivisi in pacchetti di massimo 1.008 bit per inoltrarli indipendentemente verso la destinazione. Ogni pacchetto sarebbe stato integralmente ricevuto prima dell'inoltro, quindi la subnet fu la prima rete a commutazione di pacchetto del tipo store-and-forward.

ARPA preparò una gara d'appalto per costruire la subnet, a cui parteciparono 12 società. Dopo aver valutato tutte le proposte, ARPA scelse BBN, una società di consulenza di Cambridge, Massachusetts, e nel dicembre 1968 le assegnò l'appalto per costruire la subnet e scrivere il relativo software. BBN scelse come IMP dei minicomputer Honeywell DDP-316 appositamente modificati, con memoria a nuclei magnetici di 12 K organizzata in word di 16 bit, senza dischi poiché le parti in movimento erano considerate inaffidabili. Le IMP erano collegate da linee affittate a 56 kbps, che all'epoca erano quanto di meglio la tecnologia poteva proporre.

Il software fu diviso in due parti: subnet e host. Il software della subnet era composto dalla parte lato IMP della connessione tra IMP e host, dal protocollo IMP-IMP e da un protocollo da IMP sorgente a IMP destinazione progettato per migliorare l'affidabilità. Il progetto originale di ARPANET è mostrato nella Figura 1.26.

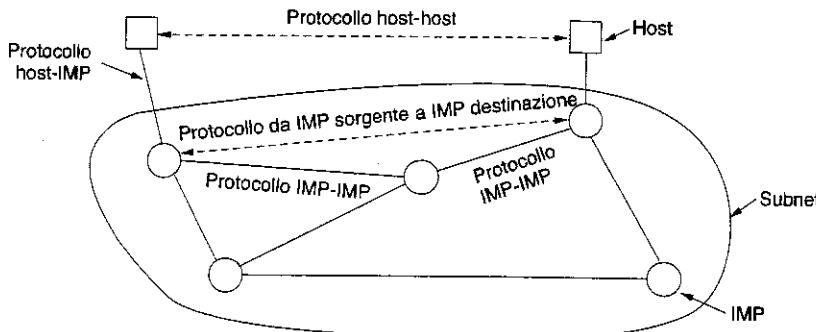


Figura 1.26. Il progetto originale di ARPANET.

Introduzione

Fuori dalla subnet era necessario altro software, in particolare il lato host della connessione host-IMP, il protocollo host-host e il software applicativo. Divenne presto evidente che BBN riteneva concluso il suo compito con l'accettazione del messaggio sul cavo host-IMP sorgente e la sua consegna sul cavo host-IMP destinazione.

Roberts aveva un problema: anche all'host serviva del software. Per affrontarlo, organizzò a Snowbird, Utah, un incontro di ricercatori specializzati in reti (soprattutto dottorandi). Si aspettavano che qualche esperto di reti spiegasse loro i principi generali della rete e del suo software, assegnando a ciascuno il compito di scriverne una parte, e rimasero sbalorditi quando seppero che non c'era nessun esperto né progetto di massima: dovevano cavarsela da soli.

In qualche modo una rete sperimentale entrò in servizio nel dicembre 1969 con quattro nodi: UCLA, UCSB, SRI e università dello Utah, scelti perché ciascuna aveva un gran numero di contratti con ARPA e utilizzava computer host diversi e completamente incompatibili (giusto per divertirsi di più). La rete crebbe rapidamente man mano che altri IMP vennero consegnati e installati; presto coprì tutti gli Stati Uniti. La Figura 1.27 mostra la velocità della crescita di ARPANET nei primi tre anni.

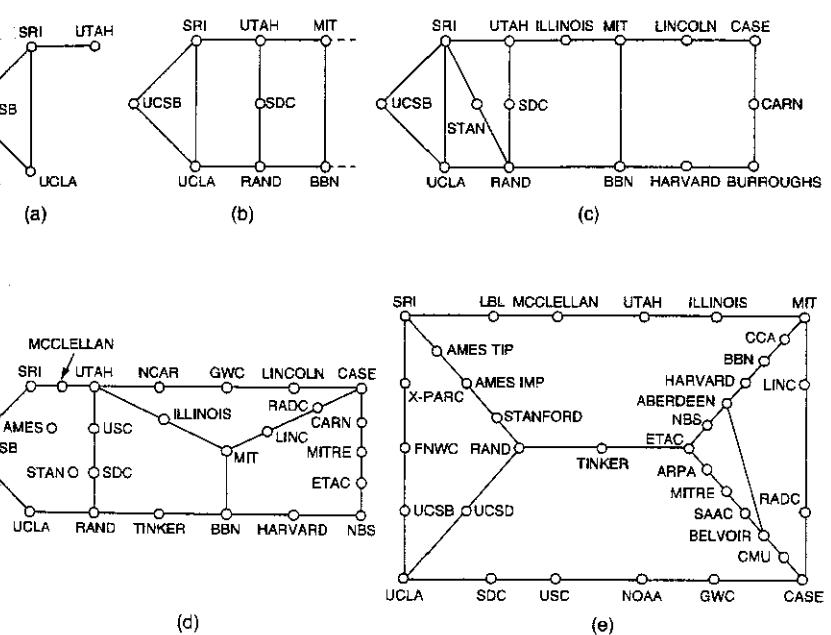


Figura 1.27. Crescita di ARPANET. (a) Dicembre 1969. (b) Luglio 1970. (c) Marzo 1971. (d) Aprile 1972. (e) Settembre 1972.

Oltre ad aiutare la crescita di ARPANET, ARPA finanziò ricerche sull'uso di reti satellitari e reti radiomobili a pacchetto. Durante una celebre dimostrazione, un camion in movimento situato in California usò la rete radiomobile a pacchetto per mandare messaggi a SRI, inoltrati su ARPANET fino alla costa est, e quindi inviati all'University College di Londra tramite una rete satellitare. I ricercatori nel camion poterono usare un computer a Londra mentre si muovevano per la California.

Questo esperimento dimostrò anche che i protocolli ARPANET esistenti non si potevano usare per reti multiple. Questa osservazione stimolò le ricerche sui protocolli, culminate con l'invenzione dei protocolli e del modello TCP/IP (Cerf e Kahn, 1974). TCP/IP fu specificamente progettato per gestire le comunicazioni su internetwork, che stavano crescendo d'importanza man mano che nuove reti erano collegate ad ARPANET.

Per incoraggiare l'adozione di questi nuovi protocolli, ARPA concesse molti contratti a BBN e all'università della California a Berkeley per integrarli nello UNIX di Berkeley. I ricercatori di Berkeley svilupparono una pratica interfaccia di programma verso la rete (socket) e scrissero molte applicazioni, utility e programmi di gestione per rendere più semplice la connessione.

La tempistica era perfetta. Molte università avevano appena acquistato un secondo o terzo computer VAX e una I^{P} per interconnetterli, ma non avevano software di rete. Quando uscì BSD 4.2 con TCP/IP, i socket e una montagna di utility per la rete, l'adozione dell'intero pacchetto fu immediata. Inoltre con TCP/IP diventava semplice collegare una LAN ad ARPANET, e molti lo fecero.

Nel corso degli anni '80 furono connesse ad ARPANET altre reti, soprattutto LAN. Al crescere della complessità la ricerca degli host diventò sempre più costosa, quindi fu creato DNS (*Domain Name System*) per organizzare i computer in domini e abbinare i nomi degli host agli indirizzi IP. Da allora DNS si è evoluto in un database distribuito non specializzato, capace di memorizzare una quantità d'informazioni relative ai nomi; lo studieremo nel Capitolo 7.

NSFNET

Nei tardi anni '70 l'organismo statunitense NSF (*National Science Foundation*) vide l'enorme impatto che ARPANET aveva sulla ricerca universitaria, permettendo a scienziati di tutto il paese di condividere i dati e collaborare a progetti di ricerca. Tuttavia per collegarsi ad ARPANET un'università doveva avere un contratto di ricerca aperto con il ministero della difesa, e molte non lo avevano. NSF reagì a questa situazione progettando un successore di ARPANET aperto a tutti i gruppi di ricerca universitari. Per partire da qualcosa di concreto, NSF decise di costruire una rete backbone per collegare i suoi sei centri (che ospitavano supercomputer) di San Diego, Boulder, Champaign, Pittsburgh, Ithaca e Princeton. A ogni supercomputer fu affiancato un fratellino, rappresentato da un micro-computer LSI-11 chiamato **fuzzball**. I fuzzball erano collegati tra loro con linee affittate a 56 kbps formando una subnet, con la stessa tecnologia hardware usata da ARPANET. La tecnologia software era invece differente: i fuzzball usarono sin dall'inizio TCP/IP, costituendo così la prima WAN TCP/IP.

NSF finanziò anche circa 20 reti regionali che si collegavano al backbone per consentire agli utenti di migliaia di università, centri di ricerca, biblioteche e musei l'accesso ai supercomputer, oltre che permettere la comunicazione reciproca. L'intera rete, composta da backbone e reti regionali, fu chiamata NSFNET. Il collegamento con ARPANET era realizzato da una connessione tra un IMP e un fuzzball situati nella sala macchine dell'università Carnegie Mellon. Il primo backbone NSFNET è mostrato nella Figura 1.28.

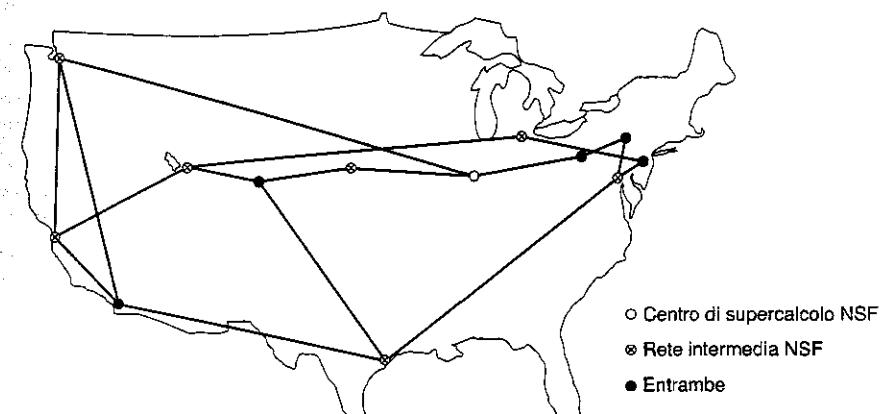


Figura 1.28. Il backbone NSFNET nel 1988.

NSFNET ebbe un successo istantaneo, e risultò sovraccarica sin dal primo giorno. NSF iniziò immediatamente a progettare il suo successore e assegnò l'appalto per realizzarlo al consorzio MERIT situato nel Michigan. Il backbone di seconda generazione fu realizzato affittando da MCI (poi inglobata in WorldCom) canali in fibra ottica da 448 kbps, mentre come router vennero impiegati dei PC-RT IBM. Anche questa rete fu presto sovraccaricata, e nel 1990 il secondo backbone fu aggiornato a 1,5 Mbps.

Con il proseguire della crescita NSF si rese conto che il governo non poteva finanziare le reti per sempre; inoltre avrebbero voluto collegarsi anche organizzazioni commerciali, ma non potevano farlo perché le regole di NSF vietavano l'uso della rete per attività estranee a quelle di NSF. Per questi motivi NSF incoraggiò MERIT, MCI e IBM a formare la società non-profit ANS (*Advanced Networks and Services*) come primo passo verso la commercializzazione.

Nel 1990 ANS prese in carico NSFNET e aggiornò i collegamenti da 1,5 Mbps portandoli a 45 Mbps per formare ANSNET; questa rete restò operativa per cinque anni e venne poi venduta ad America Online. A quel punto molte società offrivano servizi IP commerciali, ed era chiaro che per il governo era arrivato il momento di uscire dal mercato delle reti. Per facilitare la transizione e garantire che ogni rete regionale potesse comunicare con tutte le altre, NSF stipulò contratti con quattro operatori di rete perché ognuno realizzasse

un NAP (*Network Access Point*). Questi operatori erano PacBell (San Francisco), Ameritech (Chicago), MSF (Washington, D.C.) e Sprint (New York City, che a questo scopo era rappresentata da Pennsauken, New Jersey). Ogni operatore di rete che voleva offrire un servizio di backbone alle reti regionali NSF doveva collegarsi a tutti i NAP. Questa soluzione garantisce che un pacchetto originato da una qualsiasi rete regionale può scegliere l'operatore di backbone per transitare dal proprio NAP a quello di destinazione, quindi gli operatori di rete erano costretti a competere tra loro per conquistare i clienti rappresentati dalle reti regionali, confrontandosi per qualità del servizio e prezzo. Come risultato, il concetto di un singolo backbone predefinito fu sostituito da una struttura competitiva guidata da logiche commerciali. Molti accusano il governo federale di essere poco innovativo, ma nel caso del networking furono proprio il dipartimento della difesa e NSF a creare l'infrastruttura che formò la base di Internet, per poi cederla all'industria. Durante gli anni '90 molti altri paesi costruirono reti nazionali per la ricerca, spesso seguendo il modello di ARPANET e NSFNET. In Europa erano operative EuropaNET ed EBONE, che iniziarono con linee da 2 Mbps poi aggiornate a 34 Mbps. Anche l'infrastruttura di rete europea fu ceduta all'industria.

Uso di Internet

Il numero di reti, computer e utenti collegati ad ARPANET crebbe velocemente dopo il primo gennaio 1983, quando TCP/IP divenne l'unico protocollo ufficiale. Quando NSFNET e ARPANET furono interconnesse la crescita divenne esponenziale, si unirono molte reti regionali e furono creati collegamenti internazionali verso reti in Canada, Europa e Pacifico.

Nella metà degli anni '80 alcuni iniziarono a vedere l'insieme di reti come una internet, che più tardi venne considerata l'Internet per eccellenza; tuttavia non ci fu nessuna cerimonia d'inaugurazione con bottiglia di champagne mandata a infrangersi contro un fuzzball da qualche politico.

La colla che tiene insieme Internet è formata dal modello di riferimento TCP/IP e dalla corrispondente pila di protocolli. TCP/IP ha reso possibile un servizio universale e si può paragonare all'adozione del passo standard per le rotaie ferroviarie nel diciannovesimo secolo o all'adozione del protocollo di segnalazione unico da parte di tutte le società telefoniche.

Che cosa significa realmente essere su Internet? La nostra definizione è che un computer è su Internet se esegue la pila di protocolli TCP/IP, ha un indirizzo IP, e può spedire pacchetti IP a tutti gli altri computer su Internet. La semplice capacità di spedire e ricevere posta elettronica non è un requisito sufficiente, perché la e-mail viene convogliata attraverso gateway a molte reti esterne a Internet. Questa definizione è tuttavia offuscata dal fatto che milioni di personal computer possono chiamare un Internet service provider usando il modem, ricevere un indirizzo IP temporaneo, e mandare pacchetti IP ad altri host Internet. Conviene assumere che questi computer sono su Internet solo per il periodo di tempo in cui risultano connessi al router dell'Internet service provider.

Tradizionalmente (dal 1970 fino a circa il 1990) Internet e i suoi predecessori venivano usati per quattro grandi scopi.

1. **E-mail.** La possibilità di comporre, spedire e ricevere posta elettronica esiste sin dai primi giorni di ARPANET ed è estremamente popolare. Molte persone ricevono dozzine di messaggi al giorno e considerano l'e-mail il metodo principale per comunicare con l'esterno, distanziando di gran lunga telefono e posta convenzionale. Oggi esistono programmi di e-mail per qualunque genere di computer.
2. **News.** I gruppi di discussione (*newsgroup*) sono forum specializzati dove utenti con un interesse in comune possono scambiarsi messaggi. Esistono migliaia di newsgroup con argomenti tecnici e non, che affrontano temi come computer, scienza, intrattenimento e politica. Ogni newsgroup ha proprie regole, stile e abitudini: per chi le viola sono guai.
3. **Login remoto.** Usando i programmi telnet, rlogin o ssh gli utenti che si trovano in un qualsiasi punto di Internet possono fare login sui computer per i quali hanno un account.
4. **Trasferimento file.** Con il programma FTP, gli utenti possono copiare file da un computer all'altro attraverso Internet. Sono accessibili in questa modalità tantissimi articoli, database e informazioni.

Fino ai primi anni '90 Internet era per lo più usata da ricercatori accademici, governativi e industriali. Tutto ciò è stato sovvertito da una nuova applicazione, il **WWW (World Wide Web)**, che ha portato sulla rete milioni di nuovi utenti non accademici. Questa applicazione, inventata dal fisico del CERN Tim Berners-Lee, non ha cambiato nessuna delle funzionalità sottostanti ma le ha rese più facili da utilizzare. Assieme al browser Mosaic (scritto da Marc Andreessen al National Center for Supercomputer Applications in Urbana, Illinois) il WWW ha permesso la preparazione di pagine d'informazione contenenti testo, immagini, suoni e video con collegamenti interni ad altre pagine. Facendo clic su un collegamento l'utente è immediatamente trasportato nella pagina a cui il collegamento si riferisce. Per esempio, molte aziende hanno una home page con collegamenti ad altre pagine per informazioni sui prodotti, listini prezzi, informazioni per la vendita, supporto tecnico, comunicazioni con gli impiegati, informazioni per gli investitori, ecc.

In pochissimo tempo sono stati creati molti altri tipi di pagine, per esempio carte geografiche, quotazioni azionarie, cataloghi di biblioteche, programmi radiofonici preregistrati, e anche pagine che referenziano il testo completo di molti libri di cui è scaduto il copyright (Mark Twain, Charles Dickens, ecc). Molte persone hanno anche una propria home page personale.

La maggior parte di questa crescita è stata stimolata nel corso degli anni '90 da aziende chiamate **ISP (Internet Service Provider)**. Si tratta di società che danno agli utenti residenziali la possibilità di chiamare uno dei loro computer e collegarsi a Internet per acce-

dere al WWW, e-mail ed altri servizi Internet. Nei tardi anni '90 queste aziende hanno raccolto decine di milioni di nuovi utenti ogni anno trasformando completamente lo spirito della rete, che da campo riservato ad accademici e militari è diventata un servizio pubblico come la rete telefonica. Oggi il numero di utenti Internet è sconosciuto, ma a livello mondiale si tratta certamente di centinaia di milioni di persone, che molto presto supereranno la soglia del miliardo.

Architettura di Internet

In questo paragrafo faremo un breve riepilogo dell'aspetto attuale di Internet. In seguito alle molte fusioni tra società telefoniche (telco) e ISP, le acque si sono intorbidite e diventa difficile capire chi fa cosa, per questo motivo la descrizione sarà per forza di cose semplificistica. Lo schema di massima è mostrato nella Figura 1.29, che esamineremo pezzo a pezzo.

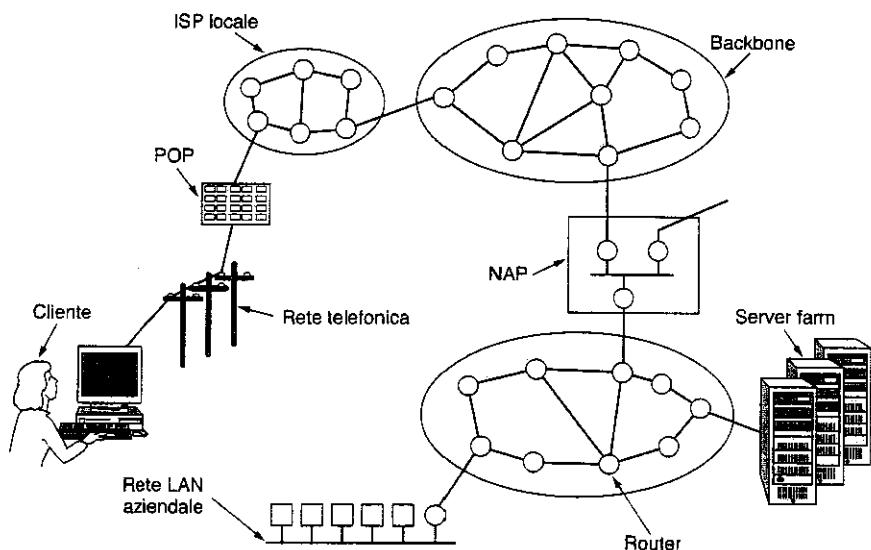


Figura 1.29. Schema di Internet.

Iniziamo dall'abitazione di un cliente; per semplicità supponiamo che i nostri clienti chiamino il proprio ISP attraverso una comune linea telefonica, come è mostrato nella Figura 1.29. Il modem è una scheda del PC che converte i segnali digitali prodotti dal computer in segnali analogici compatibili con le capacità di trasmissione della rete telefonica. Questi segnali sono trasportati fino al POP (*Point Of Presence*) dell'ISP, dove sono estratti dalla rete telefonica e inseriti nella rete regionale dell'ISP. Da questo punto in poi il sistema è completamente digitale e funziona a commutazione di pacchetto. Se l'ISP è la compagnia

telefonica locale, il POP sarà probabilmente installato nella stessa centrale telefonica dove termina il cavo del telefono che arriva al cliente. Se l'ISP non è la compagnia telefonica, il POP potrebbe essere in qualche centrale più a valle.

La rete regionale dell'ISP è composta da router interconnessi collocati nelle città servite dall'ISP. Il pacchetto viene subito consegnato se è destinato a un host direttamente servito dall'ISP, altrimenti è inoltrato sulla rete dell'operatore del backbone.

Sulla cima della "catena alimentare" si trovano i più grandi operatori di backbone a livello mondiale, come AT&T e Sprint, che gestiscono grandi reti backbone internazionali con migliaia di router collegati da fibre ottiche a larga banda. Le grandi aziende e quelle che offrono servizi di hosting gestendo server farm (macchine che servono migliaia di pagine Web ogni secondo) spesso si collegano direttamente al backbone. Gli operatori del backbone incoraggiano la connessione diretta affittando spazio nei cosiddetti **carrier hotel**, che sono in pratica degli armadi per trasmissione dati (*rack*) installati nella stessa sala dove si trovano i router, per garantire un collegamento breve e rapido tra server farm e backbone. Se un pacchetto affidato al backbone è destinato a un ISP o società servita da quel backbone, sarà inoltrato al router più vicino che lo gestirà. Nel mondo però esistono molti backbone con diverse dimensioni, quindi il pacchetto potrebbe dover transitare su un backbone della concorrenza. Per dare ai pacchetti la possibilità di passare da un backbone all'altro tutti i backbone principali sono collegati ai NAP precedentemente discussi. In concreto un NAP è una sala piena di router, almeno uno per ogni backbone. Una LAN installata nella sala collega tra loro tutti i router, quindi si possono inoltrare pacchetti da qualsiasi backbone a qualsiasi altro. Oltre ai punti di collegamento presso i NAP, i backbone più grandi hanno molte connessioni dirette con i propri router, una tecnica chiamata private peering. Uno dei tanti paradossi di Internet è che gli ISP che in pubblico competono tra loro cercando di strapparsi gli utenti, in privato spesso cooperano per fare **private peering** (Metz, 2001).

Qui finisce il nostro breve viaggio attraverso Internet. Nei prossimi capitoli avremo molto da dire sui singoli componenti e i corrispondenti progetti, sugli algoritmi e sui protocolli. Vale inoltre la pena notare che alcune società hanno interconnesso le loro reti interne preesistenti, di solito usando la stessa tecnologia di Internet. Queste intranet sono di solito accessibili solo all'interno dell'azienda, ma si comportano esattamente allo stesso modo di Internet.

1.5.2 Reti orientate alla connessione: X.25, frame relay e ATM

La guerra tra chi preferisce subnet non orientate alla connessione (cioè datagrammi) e chi invece supporta subnet orientate alla connessione è iniziata già con le prime reti. I principali fautori delle subnet non orientate alla connessione provengono dalla comunità ARPA-NET/Internet. Ricordiamo che in origine il dipartimento della difesa aveva finanziato e costruito ARPANET per ottenere una rete capace di continuare a funzionare anche dopo aver ricevuto numerosi attacchi con distruzione di router e linee di trasmissione. Per questo motivo la resistenza ai guasti era in cima alla lista di priorità, al contrario delle moda-

lità di fatturazione ai clienti. Questo approccio ha portato a una struttura senza connessione dove ogni pacchetto è instradato indipendentemente da ogni altro. Come conseguenza, non si verificano danni se alcuni router cadono durante una sessione, a patto che il sistema possa riconfigurarsi dinamicamente in modo tale che i pacchetti successivi trovino una strada qualsiasi verso la destinazione, magari diversa da quella percorsa dal pacchetto precedente.

L'appoggio alla modalità orientata alla connessione viene dal mondo delle società telefoniche. In un sistema telefonico il chiamante deve chiamare il numero del chiamato e attendere l'instaurarsi di una connessione prima di ricevere o inviare dati. Questa preparazione alla connessione iniziale stabilisce un percorso attraverso il sistema telefonico, che viene mantenuto fino alla chiusura della chiamata: tutte le parole o pacchetti seguono la stessa strada. Se cade un router o un collegamento lungo il percorso la chiamata sarà interrotta: questa è esattamente la proprietà che non piaceva al dipartimento della difesa.

Perché alle società telefoniche, invece, questo sistema piace? I motivi sono due:

1. qualità del servizio
2. fatturazione.

Stabilendo una connessione prima della trasmissione, la subnet può riservare risorse come spazio buffer e CPU del router. Quando sono disponibili meno risorse di quelle richieste da un tentativo di chiamata, la chiamata viene rifiutata e il chiamante riceve un qualche segnale di occupato. In questo modo, quando è stabilita una connessione, la qualità del servizio assegnato sarà buona. In una rete non orientata alla connessione, se arrivano nello stesso momento e allo stesso router troppi pacchetti, il router entra in crisi e probabilmente perderà pacchetti. Chi trasmette probabilmente noterà questa situazione e ritrasmetterà i pacchetti, ma la qualità del servizio sarà disomogenea e non adatta all'audio e al video, a meno che la rete abbia un carico molto leggero. È superfluo affermare che per le società telefoniche è molto importante offrire una qualità audio adeguata, da qui la loro preferenza per la tecnologia orientata alla connessione.

Il secondo motivo per cui alle società telefoniche piace il servizio orientato alla connessione è che sono abituati a farsi pagare in base al tempo di connessione. Quando si fa una chiamata interurbana (o anche una chiamata locale, fuori dal nord America) la società si fa pagare per ogni minuto di collegamento. Quando le reti fecero la loro comparsa, le società telefoniche furono naturalmente attratte verso un modello dove è semplice far pagare il tempo di collegamento. Se bisogna attivare una connessione prima di mandare i dati, questo è il momento in cui è possibile iniziare a conteggiare il tempo; quando la connessione manca non è possibile fatturarla.

Ironicamente, la gestione della fatturazione è molto costosa. Se una società telefonica adottasse una formula "flat" mensile con chiamate illimitate, senza tener traccia dei tempi di connessione, probabilmente risparmierebbe un'enorme quantità di denaro nonostante l'aumento delle telefonate stimolato da questa tariffa; la soluzione è però contrastata da motivi politici, legislativi, ecc. È interessante osservare che le tariffe flat esistono in altri settori: per esempio la pay-TV ha un costo fisso mensile, che non dipende dal numero di programmi guardati. Sarebbe stato possibile progettarla già in partenza come pay-per-

view, ma ciò non è stato fatto in parte a causa del costo della fatturazione (e, visto il livello di qualità di tante trasmissioni, si può supporre che i produttori televisivi avrebbero qualche difficoltà per stabilire un prezzo). Anche molti parchi di divertimento chiedono un biglietto giornaliero che vale per un numero illimitato di attrazioni, a differenza delle giostre dove si pagano le singole corse.

Per quanto detto non dovrebbe essere una sorpresa il fatto che tutte le reti progettate dall'industria telefonica abbiano subnet orientate alla connessione, mentre forse può sorprendere il fatto che Internet stia andando verso questa direzione, per fornire una qualità di servizio maggiore per audio e video. Torneremo sull'argomento nel Capitolo 5, ma ora esamineremo alcune reti orientate alla connessione.

X.25 e frame relay

Il primo esempio di rete orientata alla connessione è **X.25**, che fu la prima rete dati pubblica. Venne attivata negli anni '70, quando il servizio telefonico era ovunque un monopolio e ogni società telefonica si aspettava di essere anche l'unico operatore di rete dati nazionale. Per usare X.25 il computer sorgente per prima cosa stabilisce una connessione con il computer remoto (cioè fa una chiamata telefonica). A questa connessione viene dato un numero di collegamento poi usato nel trasferimento dei pacchetti dati, poiché allo stesso istante si possono aprire più connessioni.

I pacchetti dati sono molto semplici, costituiti da un'intestazione di 3 byte seguita dai dati fino a un massimo di 128 byte. L'intestazione è formata da un numero di connessione (*connection number*) a 12 bit, un numero di sequenza del pacchetto (*packet sequence number*), un numero di acknowledge e altri bit. Le reti X.25 restarono in funzione per circa un decennio, con alterni successi.

Negli anni '80 le reti X.25 sono state ampiamente sostituite da un nuovo tipo di rete chiamato **frame relay**, che fondamentalmente è una rete orientata alla connessione senza controllo di flusso o di errore. Poiché è orientata alla connessione, i pacchetti quando vengono consegnati arrivano in sequenza esatta: le caratteristiche di consegna in sequenza, assenza di controllo errori e di flusso rendono frame relay adatta a gestire le reti wide area LAN. L'applicazione principale di frame relay è l'interconnessione di reti LAN tra uffici distinti di un'azienda; in ciò ha trovato un discreto successo ed è ancora utilizzata al giorno d'oggi.

Asynchronous Transfer Mode

Una rete orientata alla connessione con importanza ben maggiore è **ATM (Asynchronous Transfer Mode)**. Il motivo di questo nome un po'strano è dato dal fatto che nella rete telefonica la maggior parte delle trasmissioni è di tipo sincrono (cioè legato a un segnale di clock), a differenza di ATM.

ATM è stata progettata nei primi anni '90 e lanciata con una fortissima azione di propaganda (Ginsburg, 1996; Goralski, 1995; Ibe, 1997; Kim et al., 1994; and Stallings, 2000). ATM avrebbe soddisfatto tutte le esigenze mondiali di networking e telecomunicazione unificando voce, dati, televisione via cavo, telex, telegrafo, piccioni viaggiatori, scatole di

latta unite da spaghetti, tamburi tribali, segnali di fumo (e qualsiasi altra cosa) in un sistema integrato che avrebbe dato tutto a tutti. Non è successo. I problemi erano in gran parte simili a quelli descritti in precedenza per OSI: poca tempestività, tecnologia scadente, implementazioni carenti, incapacità politica. Gran parte della comunità Internet, reduce dalla recente vittoria contro le società telefoniche, interpretò l'annuncio di ATM come una sfida del tipo "Internet contro Telecom - parte 2: la vendetta". In realtà non si trattava di questo, e all'epoca persino i fanatici del servizio datagram sapevano che la qualità del servizio di Internet lasciava molto a desiderare. Per farla breve, ATM ha avuto molto più successo di OSI e oggi è ampiamente usata nella rete telefonica, spesso con il compito di trasportare pacchetti IP. Poiché è usata prevalentemente dai carrier (per il trasporto interno) gli utenti non sono a conoscenza della sua esistenza, ma è viva e vegeta.

Circuiti virtuali ATM

Poiché le reti ATM sono orientate alla connessione, per inviare dati è necessario inviare in precedenza un pacchetto per configurare la connessione. Mentre il pacchetto avanza nella subnet, i router lungo il suo percorso aggiornano le tabelle interne per prendere nota dell'esistenza della connessione e per riservare le risorse di cui ha bisogno. Le connessioni vengono spesso chiamate **circuiti virtuali**, in analogia ai circuiti fisici usati nel sistema telefonico. La maggior parte delle reti ATM supporta anche **circuiti virtuali permanenti**,

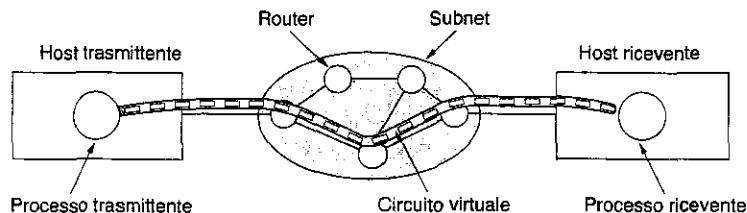


Figura 1.30. Un circuito virtuale.

che sono collegamenti permanenti tra due host (lontani) simili alle linee affittate del mondo telefonico. Ogni connessione temporanea o permanente ha un identificatore di connessione univoco. La Figura 1.30 mostra un circuito virtuale.

Quando una connessione è stabilita, ciascuna parte può iniziare a trasmettere dati. ATM si basa sull'idea di trasmettere l'informazione in piccoli pacchetti con dimensioni fisse detti celle. Le celle sono lunghe 53 byte, di cui 5 formano l'intestazione e 48 il carico utile, come mostrato nella Figura 1.31. Una parte dell'intestazione è l'identificatore di connessione, che serve all'host ricevente, a quello trasmittente e a tutti i router intermedi per discriminare le celle che appartengono a ciascuna connessione. Con questa informazione, ciascun router sa come inoltrare ognuna delle celle in arrivo. L'instradamento delle celle è effettuato in hardware ad alta velocità: il maggior vantaggio garantito dalla scelta di celle con dimensione piccola e costante è proprio la possibilità di costruire facilmente router

hardware. I pacchetti IP (che sono variabili) devono essere inoltrati via software, con un procedimento più lento. Un altro vantaggio di ATM è che si può configurare l'hardware per copiare una cella che arriva all'ingresso su più linee di uscita, proprietà necessaria per gestire un programma televisivo da trasmettere a molti ricevitori. Infine le celle di piccole dimensioni impegnano la linea per breve tempo, e ciò rende più semplice garantire la qualità del servizio.

Tutte le celle seguono lo stesso percorso verso la destinazione; la consegna non è garantita mentre lo è la sequenza. Se la cella 1 viene spedita prima della 2, e per ipotesi entrambe arrivano a destinazione, il destinatario le riceve nello stesso ordine. È impossibile che la cella 2 arrivi prima della 1, però una delle due (o entrambe) potrebbe andare perse: gestire la perdita di celle è compito dei protocolli di livello superiore. Anche se questa garanzia non è perfetta, è meglio di quanto offre Internet, dove i pacchetti non solo possono andare persi, ma anche essere consegnati fuori sequenza.

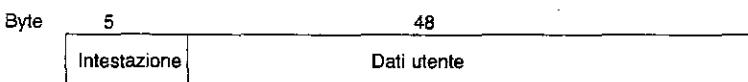


Figura 1.31. Una cella ATM.

Le reti ATM sono organizzate come WAN tradizionali, con linee e switch (router). Le velocità più comuni per le reti ATM sono 155 Mbps e 622 Mbps, ma sono supportate anche velocità superiori. La velocità di 155 Mbps è stata scelta perché vicina a quanto necessario per trasmettere la televisione ad alta definizione; il valore esatto di 155,52 Mbps è stato adottato per compatibilità con il sistema di trasmissione AT&T SONET, che sarà studiato nel Capitolo 2. La velocità di 622 Mbps è stata scelta in quanto permette il trasporto di quattro canali da 155 Mbps.

Il modello di riferimento ATM

ATM ha un proprio modello di riferimento, diverso da quello OSI e da quello di TCP/IP, mostrato nella Figura 1.32. È composto da tre strati (fisico, ATM e ATM adaptation), più quelli che gli utenti vogliono mettervi al di sopra.

Lo **strato fisico** si occupa del mezzo fisico: tensioni, temporizzazioni e altri dettagli. ATM non specifica un insieme di regole predeterminato, ma afferma che le celle ATM devono poter essere spedite su filo o fibra così come sono, oppure impacchettate dentro il carico utile di un altro sistema di trasporto. In altre parole ATM è stato progettato per essere indipendente dal mezzo di trasmissione.

Lo **strato ATM** si occupa di celle e del loro trasporto. Specifica l'organizzazione della cella e definisce il significato dei campi intestazione; si occupa inoltre dell'instaurazione e abbattimento dei circuiti virtuali e include il sistema di controllo delle congestioni.

Poiché la maggior parte delle applicazioni non desiderano lavorare direttamente con le celle (anche se alcune potrebbero), è stato definito uno strato posto sopra a quello ATM che consente agli utenti di spedire pacchetti con dimensioni superiori a una cella.

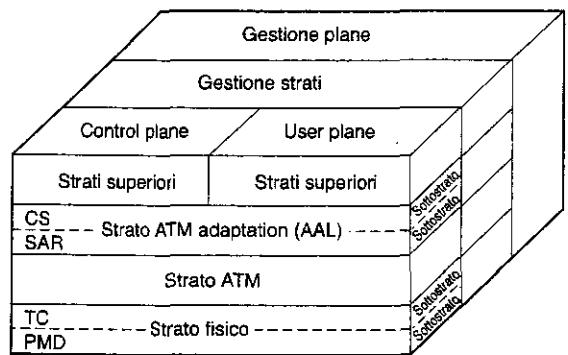


Figura 1.32. Il modello di riferimento ATM.

L'interfaccia ATM suddivide questi pacchetti, trasmette le celle individualmente e infine le ricomponete all'altra estremità. Questo strato prende il nome di **AAL** (*ATM Adaptation Layer*).

A differenza dei precedenti modelli di riferimento bidimensionali, il modello ATM ha una definizione in tre dimensioni visibile nella Figura 1.32. **User plane** si occupa di trasporto dei dati, controllo di flusso, correzione di errore e altre funzioni utente. **Control plane**, invece, s'incarica della gestione della connessione. Le funzioni di gestione degli strati e dei plane controllano l'allocazione delle risorse e il coordinamento tra gli strati.

Gli strati fisico e AAL sono divisi in due sottostrati, dove quello inferiore esegue il lavoro e quello superiore offre l'interfaccia per lo strato superiore. Le funzioni di strati e sottostrati sono mostrate nella Figura 1.33.

Il sottostrato **PMD** (*Physical Medium Dependent*, dipendente dal mezzo fisico) s'interfaccia con il cavo vero e proprio, spostando i bit e gestendo la loro temporizzazione, e cambia col variare del tipo di cavi e trasporto impiegati.

L'altro sottostrato dello strato fisico è **TC** (*Transmission Convergence*, convergenza della trasmissione), che durante la trasmissione delle celle s'incarica di mandarle come stringa di bit al sottostrato PMD: un compito facile. All'altra estremità, il sottostrato TC riceve un flusso di bit dal sottostrato PMD, e il suo lavoro consiste nel trasformarlo in un flusso di celle per lo strato ATM gestendo tutti i problemi legati all'individuazione dei punti d'inizio e termine delle celle all'interno del flusso. Nel modello ATM questa funzione è svolta dallo strato fisico, mentre in quello OSI (e nella maggior parte delle altre reti) è lo strato data link che esegue la funzione di framing, cioè la conversione di un flusso di bit grezzo in una sequenza di frame o celle.

Abbiamo già menzionato il fatto che lo strato ATM gestisce le celle, dalla generazione al trasporto, e qui stanno gli aspetti più interessanti di ATM. È un mix degli strati data link e network del modello OSI, non diviso in sottostrati.

Introduzione

Strato OSI	Strato ATM	Sottostrato ATM	Funzione
3/4	AAL	CS	Offre l'interfaccia standard (convergenza)
		SAR	Segmentazione e ricomposizione
2/3	ATM		Controllo di flusso Generazione ed estrazione dell'intestazione di cella Gestione del circuito e percorso virtuali Multiplexing/demultiplexing cella
2		TC	Disaccoppiamento cadenza celle Generazione e verifica del checksum delle intestazioni Generazione celle Impacchettamento ed estrazione delle celle dall'involucro di contenimento Generazione frame
		Fisico	
1		PMD	Temporizzazioni Accesso alla rete fisica

Figura 1.33. Funzione degli strati e sottostrati ATM.

Lo strato AAL è suddiviso nei sottostrati **SAR** (*Segmentation and Reassembly*, segmentazione e ricomposizione) e **CS** (*Convergence Sublayer*, sottostrato di convergenza). Il sottostrato inferiore divide i pacchetti in celle sul lato di trasmissione e le ricomponete alla destinazione. Il sottostrato superiore permette ai sistemi ATM di offrire tipi di servizio diversi ad applicazioni differenti (per esempio il trasferimento dei file e il video on-demand hanno richieste diverse per la gestione degli errori, le temporizzazioni, ecc.). Poiché ATM è ormai in una fase matura non sarà ulteriormente discusso in questo libro, anche se grazie alla sua vasta base installata resterà probabilmente in servizio ancora qualche anno. Per ulteriori informazioni su ATM vedere (Dobrowski and Grise, 2001; Gadecki and Heckart, 1997).

1.5.3 Ethernet

Internet e ATM sono progettati per le reti che coprono un'ampia estensione geografica, però molte aziende, università e altre organizzazioni hanno un gran numero di computer da collegare tra loro. Questa necessità ha determinato la nascita delle reti locali (LAN, *Local Area Network*), e in questo paragrafo esamineremo qualche aspetto della più popolare: Ethernet.

La sua storia inizia nei primi anni '70 alle Hawaii, che all'epoca ancora non avevano un sistema telefonico funzionante. L'assenza di telefoni rendeva la vita più piacevole per i turisti, che non erano continuamente interrotti da telefonate durante le loro attività, ma

non per il ricercatore Norman Abramson e i suoi colleghi della università delle Hawaii che tentavano di collegare al computer principale di Honolulu gli utenti situati su isole remote. La posa nell'oceano di cavi dedicati era impensabile, così cercarono una soluzione alternativa.

L'unica soluzione trovata era la trasmissione radio a bassa potenza. Il terminale di ciascun utente fu equipaggiato con una piccola radio a due frequenze: upstream (verso il computer centrale) e downstream (dal computer centrale). Quando l'utente voleva collegarsi al computer, trasmetteva nel canale upstream un pacchetto contenente i dati. Se in quel momento nessun altro stava trasmettendo, il pacchetto probabilmente sarebbe arrivato a destinazione e confermato sul canale downstream. In caso di contesa sul canale upstream il terminale avrebbe notato l'assenza di conferma e ritentato la trasmissione. Poiché sul canale downstream era presente una sola entità trasmittente (il computer centrale), su di esso non erano possibili collisioni. Questo sistema, chiamato ALOHANET, funzionava decisamente bene in situazioni di basso traffico ma entrava seriamente in crisi quando il traffico upstream era intenso.

All'incirca nello stesso periodo un dottorando chiamato Bob Metcalfe otteneva la laurea al M.I.T. e si trasferiva ad Harvard per ottenere il Ph.D. Durante i suoi studi conobbe il lavoro di Abramson, e ne fu talmente interessato che dopo il Ph.D decise di trascorrere l'estate alle Hawaii per lavorare con Abramson, prima di iniziare il suo nuovo lavoro al PARC (*Palo Alto Research Center*) di Xerox. Quando arrivò al PARC, vide che i ricercatori avevano progettato e costruito ciò che in seguito prese il nome di personal computer, ma questi computer erano isolati. Utilizzando quello che aveva imparato dal lavoro di Abramson, assieme con il collega David Boggs progettò e costruì la prima rete locale (Metcalfe e Boggs, 1976).

Chiamarono il sistema **Ethernet** citando il *luminiferous ether*, mezzo attraverso cui un tempo gli scienziati immaginavano che si propagasse la radiazione elettromagnetica. Quando nel diciannovesimo secolo il fisico James Clerk Maxwell scoprì che la radiazione elettromagnetica si poteva descrivere con un'equazione d'onda, gli scienziati postularono l'esistenza di un "mezzo etero" che riempiva lo spazio per consentire la propagazione della radiazione. Solo dopo il celebre esperimento di Michelson-Morley del 1887 gli scienziati scoprirono che la radiazione si poteva propagare nel vuoto.

Il mezzo trasmissivo non era il vuoto, ma un cavo coassiale a grande diametro (*thick*) lungo fino a 2,5 km (con ripetitori ogni 500 metri). Al sistema si potevano collegare fino a 256 computer, tramite transceiver applicati al cavo. Un cavo con più computer collegati in parallelo è chiamato cavo **multidrop**. Il sistema funzionava a 2,94 Mbps, e la Figura 1.34 è uno schizzo della sua architettura. Ethernet offriva un'importante miglioria rispetto ad ALOHANET: prima di trasmettere il computer ascolta il cavo per verificare se qualcun altro sta già trasmettendo, e in caso affermativo attende la fine della trasmissione. In questo modo si evitano le interferenze con le trasmissioni in atto, offrendo un'efficienza molto superiore. ALOHANET non funzionava in questo modo poiché il terminale su di un'isola non poteva ascoltare la trasmissione di un altro terminale posto su un'altra isola; questo problema non esiste usando un singolo cavo.

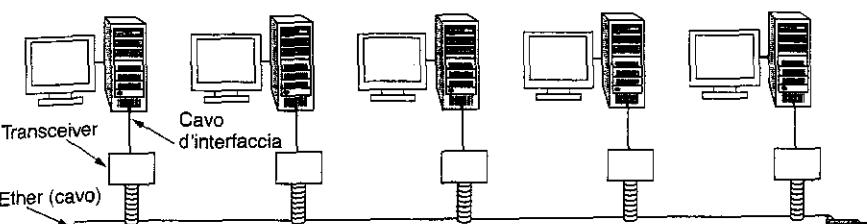


Figura 1.34. Architettura della versione originale di Ethernet.

Rimane un problema: anche se il computer ascolta prima di trasmettere, che succede se due o più computer sono in attesa della trasmissione corrente per trasmettere a loro volta, e quindi iniziano a trasmettere simultaneamente? Si deve fare in modo che ogni computer ascolti anche il proprio segnale trasmesso; se rileva interferenza blocca il cavo per avvertire tutti gli altri trasmettitori, quindi si disconnette e attende per un intervallo di lunghezza casuale prima di ripetere. Se accade una seconda collisione il tempo di attesa casuale raddoppia, e così via per distanziare le trasmissioni in competizione tra loro e dare a una di esse la possibilità di avviarsi prima delle altre.

Ethernet di Xerox ebbe talmente successo che DEC, Intel e Xerox misero a punto nel 1978 uno standard per la versione a 10 Mbps, chiamato **DIX**. Con due piccole modifiche DIX diventò nel 1983 lo standard IEEE 802.3.

Sfortunatamente già altre volte Xerox aveva sviluppato invenzioni vitali (come il personal computer) senza poi commercializzarle, una storia raccontata nel libro *Fumbling the Future* (Smith e Alexander, 1988). Quando Xerox mostrò poco interesse nello sviluppo di Ethernet (a parte un aiuto nello sforzo di standardizzazione), Metcalfe fondò una propria azienda, 3Com, per vendere schede Ethernet da installare nei PC. Ne ha vendute oltre 100 milioni.

Ethernet ha continuato a svilupparsi ed è tuttora in evoluzione; sono già state rilasciate versioni a 100 Mbps, 1.000 Mbps e oltre. Anche il cablaggio è migliorato, e sono state aggiunte altre funzionalità come lo switching. Ethernet sarà discussa in dettaglio nel Capitolo 4.

Per inciso, Ethernet (IEEE 802.3) non è l'unico standard per le LAN; il comitato IEEE ha standardizzato anche token bus (802.4) e token ring (802.5). Il bisogno di tre standard più o meno incompatibili tra loro ha poco a che fare con la tecnologia, ed è prettamente politico. All'epoca della standardizzazione General Motors stava spingendo una LAN con la stessa topologia di Ethernet (un cavo lineare) ma dove i computer trasmettevano a turno passandosi un breve pacchetto chiamato **token**. Per evitare collisioni un computer poteva trasmettere solo quando possedeva il token. General Motors annunciò che questo schema era vitale per costruire automobili e non aveva intenzione di cambiare la sua posizione; nonostante ciò 802.4 è praticamente svanito.

Analogamente, IBM aveva il suo favorito: il sistema proprietario token ring. Il token scorreva lungo l'anello, quindi il computer che lo possedeva poteva trasmettere e poi rimettere il token in circolo. A differenza di 802.4 questo schema (standardizzato come 802.5) è ancora usato in alcuni siti IBM, ma praticamente nessuno lo ha adottato fuori dall'ambito IBM. È in corso lo sviluppo di una versione gigabit (802.5v), ma sembra improbabile che possa mai fare vera concorrenza a Ethernet. In breve c'è stata una guerra tra Ethernet, token bus e token ring; Ethernet l'ha vinta sia perché è stata la prima, sia perché i concorrenti non erano altrettanto validi.

1.5.4 LAN senza fili: 802.11

Con i computer portatili è nato il desiderio di arrivare in ufficio, accendere il proprio notebook e trovarlo magicamente connesso a Internet. Svariati gruppi iniziarono a lavorare sui metodi per ottenere questo risultato, verificando che l'approccio più pratico consiste nell'equipaggiare l'ufficio e il computer portatile con ricevitrasmettitori a bassa potenza per consentire la comunicazione reciproca. Questo lavoro ha velocemente portato molte aziende alla commercializzazione di reti LAN senza fili (*wireless*).

La difficoltà stava nel fatto che queste reti risultavano incompatibili tra loro. La proliferazione degli standard significava che un computer equipaggiato con una scheda radio marca X non poteva funzionare in una stanza attrezzata con la stazione base marca Y. Alla fine l'industria stabilì che sarebbe stata una buona idea creare uno standard per reti LAN senza fili, quindi venne dato il compito di progettare uno standard LAN wireless al comitato IEEE che aveva standardizzato le LAN cablate. Lo standard così realizzato è stato battezzato 802.11. È noto anche con il termine colloquiale WiFi, ma poiché si tratta di uno standard importante (che è giusto rispettare) lo chiameremo con il suo vero nome tecnico 802.11.

Lo standard proposto può funzionare in due modi:

1. in presenza di una stazione base
2. in assenza di una stazione base.

Nel primo caso tutte le comunicazioni passano attraverso la stazione base, che nella terminologia 802.11 prende il nome di **access point**. Nel secondo i computer spediscono i dati direttamente tra loro, e questa modalità viene generalmente chiamata **ad hoc networking**. Un caso tipico si verifica quando due o più persone siedono assieme in una stanza non equipaggiata con una rete wireless LAN, e i computer comunicano tra loro in modo diretto. Le due modalità sono illustrate nella Figura 1.35.

La prima decisione da prendere per creare lo standard fu la più facile: stabilire il nome. Tutti gli altri standard per LAN avevano numeri come 802.1, 802.2, 802.3 fino a 802.10, quindi lo standard per LAN senza fili è stato chiamato 802.11. Le altre decisioni furono meno facili.

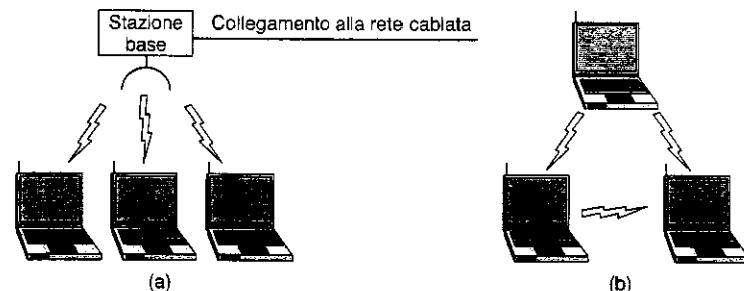


Figura 1.35. (a) Networking senza fili con una stazione base. (b) Networking ad hoc.

In particolare, alcune delle sfide da affrontare riguardavano la scelta di una banda di frequenze che fosse disponibile in ogni parte del mondo (se possibile), la portata finita dei segnali radio, la tutela della privacy degli utenti, l'autonomia limitata delle batterie, la tutela della salute umana (la radiofrequenza è cancerogena?), le implicazioni della mobilità, e infine la creazione di un sistema abbastanza appetibile da essere economicamente fattibile su scala industriale.

Quando iniziò il processo di standardizzazione (metà degli anni '90), Ethernet era già divenuta la forma dominante di rete LAN, quindi il comitato decise di rendere 802.11 compatibile con Ethernet al di sopra dello strato data link. In particolare è possibile mandare un pacchetto IP su una LAN wireless allo stesso modo in cui un computer cablato lo manda su Ethernet. Nonostante ciò, esistono molte differenze con Ethernet negli strati fisico e data link, che lo standard deve prendere in considerazione.

Prima di tutto, un computer su Ethernet ascolta sempre il mezzo trasmisivo prima di trasmettere, e inizia a mandare i dati solo se il mezzo è libero. Con le reti LAN wireless questo metodo non funziona bene: esaminiamo la Figura 1.36 per scoprire come mai. Supponiamo che il computer A stia trasmettendo al computer B, ma la portata del trasmettitore radio A non arriva a raggiungere il computer C. Se C vuole trasmettere a B, può ascoltare il mezzo trasmisivo prima d'iniziare, ma se non ascolta nulla ciò non significa che la sua trasmissione avrà successo. Lo standard 802.11 doveva risolvere questo problema.

Il secondo problema da risolvere è che un segnale radio può essere riflesso da oggetti solidi, quindi può essere ricevuto più volte (su percorsi multipli). L'interferenza che ne deriva si chiama **multipath fading**.

Il terzo problema è che molto software non è concepito per la mobilità. Per esempio, molti software per l'elaborazione dei testi hanno un elenco di stampanti che viene presentato all'utente quando vuole stampare un file. Se il computer su cui gira il programma è spostato in un nuovo ambiente, l'elenco delle stampanti perde validità.

Il quarto problema nasce quando un computer portatile viene allontanato dalla stazione base montata a soffitto che sta usando, e spostato nel raggio di azione di un'altra. È neces-

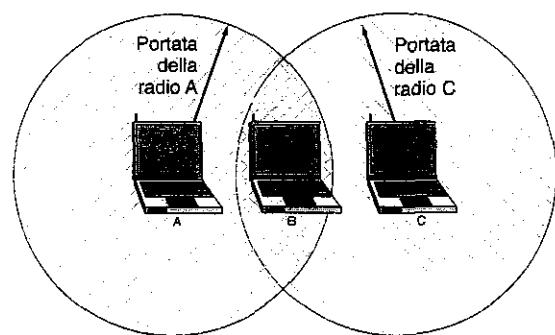


Figura 1.36. La portata di una singola sezione radio potrebbe non coprire l'intero sistema.

saria qualche forma di commutazione: il problema è lo stesso dei telefoni cellulari, ma non si verifica con Ethernet e quindi va risolto. In particolare, la rete proposta è formata da celle multiple dove ciascuna ha la propria stazione base, collegata alle altre tramite Ethernet come mostrato nella Figura 1.37. Dall'esterno il sistema appare come una singola Ethernet. La connessione tra i sistemi 802.11 e il mondo esterno viene chiamata **portale**.

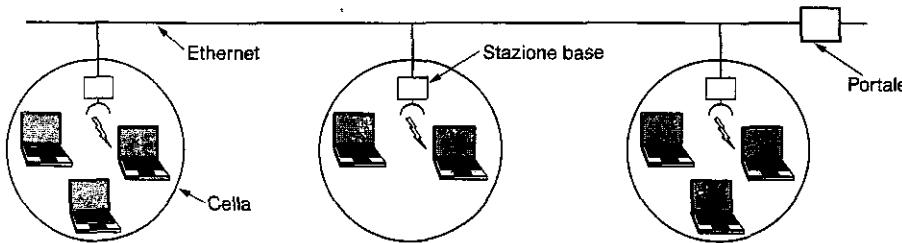


Figura 1.37. Una rete 802.11 multicella.

Nel 1977 il comitato ha presentato uno standard che risolve questi e altri problemi, descrivendo una rete LAN wireless funzionante a 1 Mbps oppure 2 Mbps. Quasi immediatamente la gente si lamentò per la velocità insufficiente, quindi iniziò il lavoro per standard più veloci. Nel comitato si verificò una divisione, che determinò la proposta di due nuovi standard nel 1999. Lo standard 802.11a usa una banda di frequenze più ampia e lavora a velocità fino a 54 Mbps, mentre 802.11b usa la stessa banda di frequenze di 802.11, ma tecniche di modulazione diverse per raggiungere 11 Mbps. Per alcuni ciò ha un'importanza psicologica, poiché 11 Mbps è maggiore della velocità della Ethernet cablata originale. Probabilmente l'802.11 originale a 1 Mbps svanirà velocemente, ma non è ancora chiaro quale dei due nuovi standard prenderà il sopravvento.

Per rendere le cose ancora più complicate di quanto già non siano, il comitato 802.11 ha proposto un'altra variante, 802.11g, che usa le tecniche di modulazione di 802.11a

e la banda di frequenze di 802.11b. Nel Capitolo 4 affronteremo ulteriormente l'argomento 802.11.

È ormai certo che 802.11 genererà una rivoluzione nel mondo dei computer e nell'accesso a Internet. Aeroporti, stazioni ferroviarie, hotel, centri commerciali e università lo stanno velocemente installando. Persino piccoli Internet café installano 802.11 per dare ai clienti la possibilità di navigare il Web mentre sorseggiano il loro cappuccino. È probabile che 802.11 farà per Internet ciò che il computer portatile ha fatto per l'informatica: renderla mobile.

1.6 Standardizzazione delle reti

Esistono molti costruttori e fornitori di reti, ognuno con le proprie idee sul modo di fare le cose. In assenza di coordinamento il caos sarebbe totale, e gli utenti non avrebbero nulla in mano. L'unica strada consiste nel concordare su alcuni standard di rete.

Non solo gli standard permettono la comunicazione tra computer diversi, ma aumentano il mercato per i prodotti che vi aderiscono. Un mercato più ampio sostiene la produzione in serie, assicura economie di scala nella costruzione, giustifica l'implementazione VLSI e garantisce altri benefici che diminuiscono i prezzi dei prodotti. Nei paragrafi che seguono daremo un rapido sguardo al mondo importante ma poco conosciuto della standardizzazione internazionale.

Gli standard ricadono in due categorie: de facto e de jure. Gli standard **de facto** (in latino significa "dalla realtà") sono quelli che si sono stabiliti senza piani formali. Il PC IBM e i suoi successori sono lo standard de facto per i personal computer da casa e ufficio, perché dozzine di produttori hanno scelto di copiare fin nei dettagli le macchine IBM. Allo stesso modo UNIX è lo standard de facto per i sistemi operativi nelle facoltà d'informatica universitarie.

Gli standard **de jure** (in latino significa "per legge") sono al contrario formali, adottati da qualche organismo di standardizzazione autorizzato. Le autorità per la standardizzazione internazionale vengono solitamente divise in due classi: quelle stabilite da trattati tra i governi delle nazioni, e quelle che comprendono organizzazioni volontarie. Nel campo degli standard per le reti di computer esistono diverse organizzazioni di entrambi i tipi, che sono discusse nel seguito.

1.6.1 Il who's who del mondo delle telecomunicazioni

Lo status legale delle aziende telefoniche nel mondo varia notevolmente da nazione a nazione. A un estremo si trovano gli Stati Uniti, che hanno 1.500 aziende telefoniche private e indipendenti. Prima di essere suddivisa nel 1984, AT&T (che era la più grande corporation del mondo) dominava la scena in modo totale e offriva il servizio telefonico all'80% dei telefoni americani su di una metà della sua intera area geopolitica, mentre tutte le altre società messe insieme offrivano servizio ai clienti rimanenti (per lo più rurali). Dopo la divisione AT&T continua a offrire il servizio a lunga distanza, ma in competizio-

ne con altre società. Il servizio telefonico cellulare e quello locale sono offerti dalle sette Regional Bell Operating Companies nate dalla divisione della AT&T, e da molti operatori indipendenti. In seguito alle continue fusioni e cambiamenti l'industria è in una perenne situazione di flusso.

Le società statunitensi che offrono servizi di telecomunicazione al pubblico si chiamano **common carriers**. Offerte e prezzi sono descritte da un documento chiamato **tariffa**, che è approvato dalla Federal Communications Commission nel caso di traffico tra gli stati e internazionale, e dalle commissioni per il servizio pubblico dei singoli stati per il traffico intrastatale.

All'estremo opposto si trovano le nazioni dove il governo ha il monopolio completo di tutte le comunicazioni, incluse posta, telegrafo, telefono e spesso anche radio e televisione. La maggior parte del mondo ricade in questa categoria. In alcuni casi l'authority per le telecomunicazioni è una società a partecipazione statale, mentre in altri è semplicemente un ramo del governo generalmente noto con il nome PTT (amministrazione postale, telegrafica e telefonica). Nel mondo la tendenza è verso liberalizzazione e competizione, allontanandosi dal monopolio statale. La maggioranza delle nazioni europee ha (parzialmente) liberalizzato le proprie PTT, ma altrove il processo è ancora agli inizi.

Con tutti questi diversi fornitori di servizio, è chiaramente necessario offrire compatibilità su scala mondiale per garantire che la gente (e i computer) di una nazione possano chiamare le controparti in un'altra. Questa necessità esiste da molto tempo: nel 1865 i rappresentanti di molti governi europei si riunirono per formare il predecessore dell'attuale ITU (*International Telecommunication Union*). Il suo compito era quello di standardizzare le telecomunicazioni internazionali, che all'epoca significavano la telegrafia. Anche allora fu chiaro che sarebbero nati problemi se metà delle nazioni avesse usato il codice Morse e l'altra metà un altro. Quando il telefono divenne un servizio internazionale, ITU proseguì l'opera di standardizzazione della telefonia, e nel 1947 ITU divenne un'agenzia delle Nazioni Unite.

ITU ha tre settori principali:

1. settore comunicazioni radio (ITU-R)
2. settore per la standardizzazione delle telecomunicazioni (ITU-T)
3. settore sviluppo (ITU-D).

ITU-R si occupa dell'allocazione mondiale delle radiofrequenze ai gruppi d'interesse tra loro in competizione. Ci concentreremo soprattutto su ITU-T, che si occupa di telefonia e sistemi di comunicazione dati. Dal 1956 al 1993, ITU-T è stata conosciuta con la sigla CCITT, un acronimo del suo nome francese: *Comité Consultatif International Télégraphique et Téléphonique*. Il primo marzo 1993 CCITT fu riorganizzata per renderla meno burocratica e ribattezzata per riflettere il suo nuovo ruolo. ITU-T e CCITT hanno

emanato raccomandazioni nell'area delle comunicazioni telefoniche e dati. Spesso ci si imbatte in raccomandazioni CCITT, come CCITT X.25, benché dal 1993 le raccomandazioni portino l'etichetta ITU-T.

I membri di ITU-T ricadono in quattro categorie:

1. governi nazionali
2. membri del settore
3. membri associati
4. enti normativi.

ITU-T ha circa 200 membri governativi, che includono praticamente ogni membro delle Nazioni Unite. Poiché gli Stati Uniti non hanno una PTT, qualcun altro li deve rappresentare in ITU-T. Il compito spetta al Dipartimento di Stato, probabilmente perché ITU-T ha a che fare con le nazioni estere, che sono la specialità del Dipartimento di Stato. Ci sono circa 500 membri di settore, che comprendono società telefoniche (AT&T, Vodafone, WorldCom...), produttori di apparati per telecomunicazioni (Cisco, Nokia, Nortel...), produttori di computer (Compaq, Sun, Toshiba...), produttori di chip (Intel, Motorola, TI...), produttori di contenuti (AOL Time Warner, CBS, Sony...) e altre aziende interessate (per esempio Boeing, Samsung, Xerox). Sono membri di settore anche diverse organizzazioni scientifiche senza scopo di lucro e consorzi industriali (come IFIP e IATA). I membri associati sono organizzazioni più piccole interessate a specifici gruppi di studio. Gli enti normativi sono quelli che controllano il mercato delle telecomunicazioni, come la Federal Communications Commission U.S.A.

Lo scopo di ITU-T è emanare raccomandazioni tecniche su interfacce per telefonia, telegrafia e trasmissione dati, che spesso diventano standard riconosciuti a livello internazionale come V.24 (nota come EIA RS-232 negli Stati Uniti), che definisce la posizione e il significato dei contatti sul connettore usato dalla maggior parte dei terminali asincroni e modem esterni.

Va osservato che dal punto di vista tecnico le raccomandazioni ITU-T sono solo suggerimenti che i governi possono adottare o ignorare a loro scelta (questo perché i governi sono come ragazzi di 13 anni, a cui non piace prendere ordini). In pratica, una nazione che volesse adottare uno standard telefonico diverso da quello usato dal resto del mondo è libera di farlo, ma al prezzo di tagliarsi fuori da chiunque altro. Potrebbe andar bene per la Corea del Nord, ma altrove sarebbe un vero problema.

Il lavoro di ITU-T è portato avanti dai suoi 14 gruppi di studio (*Study Groups*), che possono comprendere anche 400 persone. Gli argomenti che affrontano spaziano dalla tariffazione delle telefonate ai servizi multimediali. Per raggiungere gli obiettivi, i gruppi di studio sono divisi in Working Parties, che a loro volta sono divisi in Expert Teams, divisi infine in gruppi ad-hoc. La burocrazia è fatta così.

Nonostante tutto, ITU-T ottiene realmente dei risultati. Dalla sua formazione ha prodotto quasi 3.000 raccomandazioni, che occupano circa 60.000 pagine di carta, molte delle quali sono ampiamente utilizzate. Per esempio il noto standard V.90 per i modem a 56 kbps è una raccomandazione ITU.

Mentre le telecomunicazioni completano la trasformazione iniziata negli anni '80 che sta portando le entità nazionali verso la globalizzazione, gli standard diventano sempre più importanti, e sempre più organizzazioni vogliono partecipare alla loro formazione. Per ulteriori informazioni su ITU, vedere (Irmer, 1994).

1.6.2 Il who's who del mondo degli standard internazionali

Gli standard internazionali sono prodotti e pubblicati da **ISO** (*International Standard Organization*; per i puristi, il vero nome di ISO è *International Organization for Standardization*), un'organizzazione volontaria fondata nel 1946. I suoi membri sono gli organismi di standardizzazione nazionali degli 89 paesi che vi partecipano. I membri includono ANSI (U.S.A.), BSI (Gran Bretagna), AFNOR (Francia), DIN (Germania) e altri 85.

ISO emette standard su una vastissima gamma di argomenti, dai bulloni al rivestimento dei pali telefonici, senza trascurare le noci di cocco (ISO 2451), le reti da pesca (ISO 1530), l'abbigliamento intimo femminile (ISO 4416) e tanti altri argomenti che a prima vista non si pensa abbiano bisogno di standardizzazione. Sono stati emanati oltre 13.000 standard, inclusi gli standard OSI. ISO ha almeno 200 comitati tecnici (*Technical Committees*), numerati in ordine di creazione, ognuno dei quali affronta un argomento specifico. TC1 si occupa di viti e bulloni, standardizzando il passo e i diametri, mentre TC97 si occupa di computer ed elaborazione delle informazioni. Ogni TC ha sottocomitati (SC) divisi in gruppi di lavoro (WG).

Il lavoro è portato avanti soprattutto nei WG, da più di 100.000 volontari in tutto il mondo. Molti di questi "volontari" ricevono l'incarico di lavorare per ISO dai loro datori di lavoro, i cui prodotti devono essere standardizzati. Altri volontari lavorano per il governo del proprio paese, desideroso che il modo di fare le cose nella propria nazione diventi la norma internazionale. In molti WG sono attivi anche esperti accademici.

ISO e ITU-T cooperano spesso nel campo degli standard per le telecomunicazioni (ISO è membro di ITU-T) per scongiurare il rischio di avere due standard ufficiali mutuamente incompatibili.

Il rappresentante U.S.A. in ISO è **ANSI** (*American National Standards Institute*), che nonostante il nome è un'organizzazione privata senza fini di lucro. I suoi membri sono produttori, aziende telefoniche e altri interessati. Gli standard ANSI vengono spesso adottati da ISO come standard internazionali.

ISO adotta gli standard seguendo una procedura pensata per ottenere il maggior consenso possibile. Il processo inizia quando uno degli enti normativi nazionali sente la necessità di uno standard internazionale in qualche area. Viene quindi formato un gruppo di lavoro per produrre un **CD** (*Committee Draft*). Il CD viene fatto circolare tra tutti i membri, che hanno sei mesi per esporre critiche. Se la maggioranza assoluta lo approva, viene prodotto un documento chiamato **DIS** (*Draft International Standard*), fatto circolare per commenti e votazione. Sulla base dei risultati di questo round, viene preparato, approvato e pubblicato il testo finale dell'**IS** (*International Standard*). Nei settori che suscitano controversie un CD o DIS può richiedere parecchie versioni prima di raggiungere un numero sufficiente di voti, e l'intero processo può impiegare anni.

NIST (*National Institute of Standards and Technology*) fa parte del dipartimento del commercio U.S.A., e in precedenza era chiamato National Bureau of Standards. Emette standard obbligatori per gli acquisti di materiale fatto dal governo U.S.A., con l'eccezione di quelli del dipartimento della difesa (che seguono i loro standard).

Un altro attore importante nel mondo degli standard è **IEEE** (*Institute of Electrical and Electronics Engineers*), la più grande organizzazione professionale del mondo. Oltre alla pubblicazione di montagne di periodici e all'organizzazione di centinaia di conferenze ogni anno, IEEE ha un gruppo di standardizzazione che sviluppa standard nel campo dell'ingegneria elettrica e dei computer. Il comitato IEEE 802 ha standardizzato molti tipi di LAN, e alcuni saranno studiati nel seguito di questo libro. L'attività è svolta da un insieme di gruppi di lavoro, elencati nella Figura 1.38. Il successo dei diversi gruppi di lavoro 802 è stato basso; avere un numero 802.x non è garanzia di successo, anche se l'impatto dei casi fortunati (specialmente 802.3 e 802.11) è stato enorme.

1.6.3 Il who's who del mondo degli standard Internet

La Internet mondiale ha un proprio meccanismo di standardizzazione, molto diverso da ITU-T e ISO. La differenza può essere sbrigativamente riassunta dicendo che la gente che partecipa agli incontri ITU e ISO veste in giacca e cravatta, mentre quella che partecipa agli incontri di standardizzazione Internet porta i jeans (eccetto quando si riunisce a San Diego, dove porta pantaloncini e T-shirt).

Gli incontri ITU-T e ISO sono frequentati da manager e da ufficiali pubblici per i quali la standardizzazione è il lavoro principale. Vedono la standardizzazione come il Bene e gli dedicano la loro esistenza. Al contrario la gente di Internet fondamentalmente preferisce l'anarchia, ma con centinaia di milioni di utenti che svolgono le loro attività sarebbe difficile comunicare. Per questo motivo a volte servono gli standard, purtroppo.

Quando fu creata ARPANET, il dipartimento della difesa creò un comitato informale per supervisionarla. Nel 1983 il comitato fu rinominato **IAB** (*Internet Activities Board*) e gli fu data una missione leggermente più ampia, cioè mantenere orientati nella stessa direzione i ricercatori interessati ad ARPANET/Internet: un'attività che si può paragonare a fare il bagno al gatto. Il significato dell'acronimo IAB venne poi cambiato in *Internet Architecture Board*.

Numero	Argomento
802.1	Aspetti generali e architettura delle LAN
802.2 ↓	Logical link control
802.3 *	Ethernet
802.4 ↓	Token bus (usato brevemente negli impianti produttivi)
802.5	Token ring (la proposta IBM per le LAN)
802.6 ↓	Dual queue dual bus (standard originario per le metropolitan area network)
802.7 ↓	Technical advisory group per le tecnologie a larga banda
802.8 †	Technical advisory group per le tecnologie a fibra ottica
802.9 ↓	LAN isocrone (per applicazioni real-time)
802.10 ↓	Virtual LAN e sicurezza
802.11 *	Wireless LAN
802.12 ↓	Demand Priority (lo standard AnyLAN di HP)
802.13	Numeri sfortunati, nessuno lo ha voluto
802.14 ↓	Cable modem (modem per TV via cavo, standard defunto perché superato da un consorzio industriale)
802.15 *	Personal area network (Bluetooth)
802.16 *	Wireless a banda larga
802.17	Resilient packet ring

Figura 1.38. I gruppi di lavoro 802. I più importanti sono contrassegnati con *; quelli marchiati con ↓ sono in quiescenza. Quelli marchiati con † hanno rinunciato all'incarico e si sono sciolti.

Ciascuno dei circa 10 membri di IAB guidava una task force relativa a qualche argomento importante. IAB si riuniva diverse volte all'anno per discutere dei risultati e dare una retroazione al dipartimento della difesa e a NSF, che all'epoca fornivano la maggior parte dei finanziamenti. Quando era necessario uno standard (per esempio un nuovo algoritmo di routing), i membri di IAB lo modificavano all'ultimo minuto e poi annunciavano la novità, così i dottorandi che formavano il cuore dello sforzo software potevano implementarlo. La comunicazione era affidata a una serie di rapporti tecnici chiamati **RFC** (*Request For Comment*). Gli RFC, numerati cronologicamente, sono disponibili on-line e qualsiasi interessato li può scaricare da www.ietf.org/rfc. Ne esistono più di 3.000, e in questo libro faremo riferimento a molti di essi.

Nel 1989 Internet era diventata talmente grande che questo stile altamente informale non funzionava più. All'epoca molti costruttori offrivano prodotti TCP/IP, e non li volevano modificare solo perché dieci ricercatori avevano avuto un'idea migliore. Nell'estate del 1989 IAB fu nuovamente riorganizzato e i ricercatori si spostarono nell'**IRTF** (*Internet Research Task Force*), che venne resa complementare a IAB assieme a **IETF** (*Internet*

Introduzione

Engineering Task Force). IAB fu ripopolata con persone che non rappresentavano solo la comunità dei ricercatori. All'inizio era un gruppo a "trasmissione ereditaria", dove i membri restavano in carica due anni e designavano il loro successore. Più tardi fu creata **Internet Society**, frequentata da persone interessate a Internet: per questo motivo Internet Society è in un certo senso paragonabile a ACM o IEEE. È guidata da rappresentanti eletti, che eleggono i membri IAB.

Lo scopo di questa divisione fu quello di concentrare IRTF sulla ricerca a lungo termine, mentre IETF avrebbe affrontato i problemi tecnici a breve termine. IETF fu divisa in gruppi di lavoro, ciascuno dei quali ha un problema specifico da risolvere. Inizialmente i capi di ciascun gruppo di lavoro si riunivano a formare il comitato direttivo che guidava gli sforzi tecnici. Gli argomenti dei gruppi di lavoro riguardano nuove applicazioni, informazioni degli utenti, integrazione OSI, instradamento e indirizzamento, sicurezza, gestione della rete e standard. Si crearono così tanti gruppi di lavoro (più di 70) che è stato necessario raggrupparli per aree; il capo di ogni area partecipa al comitato direttivo.

In aggiunta, fu adottato un processo di standardizzazione più formale, modellato secondo ISO. L'idea originale, per diventare un **Proposed Standard**, deve essere completamente spiegata in un RFC, e destare abbastanza interesse nella comunità da essere presa in considerazione. Per passare allo stadio **Draft Standard**, un'implementazione funzionante deve essere stata provata rigorosamente da almeno due siti indipendenti per almeno quattro mesi. Se IAB è convinto che l'idea regge e il software funziona, può dichiarare **Internet Standard** l'RFC. Alcuni Internet Standard sono diventati standard del dipartimento della difesa U.S.A. (MIL-STD), obbligatori per i fornitori del dipartimento. David Clark fece un celebre commento sulla standardizzazione Internet definendola come un "vago consenso su codice che funziona".

1.7 Unità metriche

Questo testo, come tutta l'informatica, adotta le unità di misura metriche invece di quelle inglesi tradizionali. I prefissi metrici principali sono elencati nella Figura 1.39, abbreviati con le loro prime lettere, dove le unità maggiori di 1 hanno iniziali maiuscole (KB, MB, ecc); fa eccezione per motivi storici la sigla kbps che sta per kilobits/sec. Per quanto detto, una linea di comunicazione da 1 Mbps trasmette 10^6 bit/sec e un clock di 100 psec (o 100 ps) commuta ogni 10^{-10} secondi. Poiché milli e micro iniziano entrambi con la lettera "m" si è resa indispensabile una scelta, e normalmente m sta per milli mentre la lettera greca mu (μ) indica micro.

È importante osservare che la misura delle dimensioni di memorie, dischi, file e database segue una convenzione un po'diversa nella pratica industriale. Kilo significa 2^{10} (1.024) invece di 10^3 (1.000), poiché le memorie sono sempre potenze di due. Pertanto una memoria da 1 KB contiene 1.024 byte, e non 1.000. Analogamente una memoria da 1 MB contiene 2^{20} (1.048.576) byte, una da 1 GB ne contiene 2^{30} (1.073.741.824) byte, e un database da 1 TB contiene 2^{40} (1.099.511.627.776) byte. Tuttavia una linea di comunicazione

Esp	Esplicito	Prefisso	Esp	Esplicito	Prefisso
10^{-3}	0,001	milli	10^3	1.000	Kilo
10^{-6}	0,000001	micro	10^6	1.000.000	Mega
10^{-9}	0,00000001	nano	10^9	1.000.000.000	Giga
10^{-12}	0,0000000001	pico	10^{12}	1.000.000.000.000	Tera
10^{-15}	0,000000000001	femto	10^{15}	1.000.000.000.000.000	Peta
10^{-18}	0,00000000000001	atto	10^{18}	1.000.000.000.000.000.000	Exa
10^{-21}	0,0000000000000001	zepto	10^{21}	1.000.000.000.000.000.000.000	Zetta
10^{-24}	0,00000000000000000001	yocto	10^{24}	1.000.000.000.000.000.000.000.000	Yotta

Figura 1.39. I prefissi metrici più comuni.

da 1 kbps trasmette 1.000 bit al secondo, e una LAN da 10 Mbps lavora a 10.000.000 bit/sec poiché queste velocità non sono potenze di due. Sfortunatamente molti mescolano i due metodi, soprattutto per misurare le dimensioni dei dischi. Per evitare ambiguità in questo libro useremo i simboli KB, MB e GB rispettivamente per 2^{10} , 2^{20} e 2^{30} byte, e i simboli kbps, Mbps e Gbps rispettivamente per 10^3 , 10^6 e 10^9 bit/sec.

1.8 Organizzazione del libro

Il libro tratta teoria e pratica delle reti di calcolatori. La maggior parte dei capitoli inizia con una discussione dei principi applicabili, seguita da esempi che li illustrano. Gli esempi sono generalmente tratti dalle reti Internet e wireless, poiché sono entrambe importanti e molto diverse. Dove necessario saranno mostrati altri esempi.

La struttura del libro segue il modello ibrido della Figura 1.24. Con il Capitolo 2 inizia l'esame della gerarchia dei protocolli, iniziando dal basso. Il secondo capitolo offre le informazioni di base nel campo della comunicazione dati e copre i sistemi cablati, wireless e satellitari. Il materiale si occupa dello strato fisico, ma saranno coperti gli aspetti architettonici invece di quelli hardware. Sono discussi molti esempi di strato fisico, come la rete telefonica pubblica, i telefoni cellulari e la rete televisiva via cavo.

Il Capitolo 3 affronta l'argomento dello strato data link e dei suoi protocolli attraverso esempi di complessità crescente; viene svolta anche l'analisi di questi protocolli. Successivamente sono discussi alcuni protocolli di grande importanza nella pratica, tra cui HDLC (usato nelle reti a bassa e media velocità) e PPP (usato in Internet).

Il Capitolo 4 si occupa del sottostrato di accesso al mezzo di trasmissione, che fa parte dello strato data link. Il problema fondamentale che affronta riguarda la determinazione di chi sarà il prossimo a usare la rete, se questa consiste in un singolo canale condiviso (come accade nella maggioranza delle LAN e in alcune reti satellitari). Sono proposti molti esem-

pi tratti dai campi delle reti LAN cablate, LAN wireless (specialmente Ethernet), MAN Wireless, Bluetooth e reti satellitari. Vengono discussi anche bridge e switch a livello data link, che servono per collegare tra loro le LAN.

Il Capitolo 5 descrive lo strato network e in particolare l'instradamento, e analizza molti algoritmi di routing statici e dinamici. Anche se gli algoritmi di routing sono validi, quando la rete è sottoposta a più traffico di quello che è in grado di gestire nasce il problema della congestione: per questo motivo l'argomento sarà affrontato assieme ai metodi per prevenirlo. Garantire una certa qualità di servizio è una soluzione migliore della semplice prevenzione della congestione, perciò sarà discusso anche questo argomento, oltre ai problemi che nascono quando si collegano reti eterogenee per formare una internetwork. Infine è presente una completa analisi dello strato network su Internet.

Il Capitolo 6 riguarda lo strato trasporto. L'enfasi è sui protocolli orientati alla connessione, poiché sono richiesti da molte applicazioni. È quindi discusso in dettaglio un servizio di trasporto dimostrativo con le sue implementazioni; per questo esempio viene fornito anche il codice sorgente, che mostra come si può realizzare. Vengono studiati in dettaglio i due protocolli di trasporto Internet (UDP e TCP) e i loro problemi di prestazioni, inoltre sono esaminate le reti wireless.

Il Capitolo 7 è dedicato allo strato applicazione e i suoi protocolli. Il primo argomento è DNS, la rubrica telefonica di Internet, poi viene la e-mail e la discussione dei suoi protocolli. Ci muoveremo quindi sul Web, con una discussione dettagliata dei contenuti statici e dinamici, di ciò che accade sul lato client e su quello server, di protocolli, prestazioni, applicazioni wireless e altro. Infine esamineremo le applicazioni multimediali di rete tra cui streaming audio, Internet radio e video on-demand.

Il Capitolo 8 riguarda la sicurezza delle reti. Questo argomento ha aspetti che coinvolgono tutti gli strati, quindi è più semplice trattarlo dopo che sono stati dettagliatamente spiegati. Il capitolo inizia con un'introduzione alla crittografia, poi mostra come può essere usata per rendere sicura la comunicazione, la posta elettronica e il Web. Il capitolo termina con la discussione di alcune aree dove la sicurezza interseca privacy, libertà di parola, censura e altri problemi sociali.

Il Capitolo 9 contiene un elenco di letture consigliate, organizzate per capitolo. È offerto per aiutare i lettori che vogliono proseguire il loro studio delle reti. Il capitolo contiene anche una bibliografia alfabetica di tutte le referenze citate in questo testo.

Il sito Web dell'autore presso l'editore Prentice Hall

<http://www.prenhall.com/tanenbaum>

ha una pagina con collegamenti a tutorial, FAQ, aziende, consorzi industriali, organizzazioni professionali, enti normativi, tecnologie, saggi e altro.

1.9 Sommario

Le reti di computer hanno molti usi, per le aziende come per i privati. Per le aziende, le reti di personal computer connessi a server condivisi sono spesso lo strumento per accedere alle informazioni aziendali. Generalmente seguono il modello client-server, dove le postazioni client sulla scrivania degli utenti accedono a potenti server collocati nella sala macchine. Per i privati, le reti danno accesso a una vasta quantità d'informazioni e intrattenimento. Di solito accedono a Internet chiamando un ISP servendosi di un modem, anche se un numero crescente di persone ha una connessione fissa presso la propria abitazione. Un'area in veloce sviluppo è costituita dalle reti senza filo o wireless, con nuove applicazioni come l'accesso mobile alla e-mail e l'm-commerce.

A grandi linee le reti si possono dividere in LAN, MAN, WAN e internetwork, ciascuna con le proprie caratteristiche, tecnologie, velocità e nicchie. Le reti LAN coprono un edificio e lavorano ad alta velocità. Le reti MAN coprono una città, come il sistema televisivo via cavo usato da molti per accedere a Internet. Le WAN coprono una nazione o continente. Le reti LAN e MAN non sono commutate (cioè non hanno router), mentre le WAN sono commutate. Le reti wireless stanno diventando molto popolari, specialmente le LAN senza fili. Le reti si possono collegare per formare internetwork.

Il software di rete è composto da protocolli, che sono le regole seguite dai processi per comunicare. I protocolli sono senza connessione oppure orientati alla connessione. La maggior parte delle reti supporta le gerarchie di protocolli, dove ogni strato offre un servizio allo strato superiore e lo isola dai dettagli dei protocolli usati negli strati inferiori. Le pile di protocolli sono solitamente basate sul modello OSI oppure su quello TCP/IP. Entrambe hanno strati network, trasporto e applicazione, ma differiscono negli altri strati. La maggior parte di questo libro affronta i protocolli e il loro progetto.

Le reti offrono ai loro utenti dei servizi, che possono essere orientati alla connessione oppure senza connessione. In alcune reti è offerto un servizio senza connessione in uno strato e un servizio orientato alla connessione in quello sovrastante.

Internet, ATM, Ethernet e IEEE 802.11 sono reti molto note. Internet si è evoluta da ARPANET, alla quale furono aggiunte altre reti per formare una internetwork. L'attuale Internet non è una rete unica ma una raccolta di migliaia di reti, caratterizzate dall'utilizzo della pila di protocolli TCP/IP. ATM è largamente impiegato all'interno della rete telefonica per il trasporto del traffico dati a lunga distanza. Ethernet è la LAN più usata, e si ritrova nella maggior parte delle aziende di grandi dimensioni e università. Infine, LAN wireless con velocità sorprendentemente alte (fino a 54 Mbps) sono agli inizi di un'adozione su vasta scala.

Per garantire la comunicazione tra più computer è necessaria molta standardizzazione, nel hardware e nel software. Organizzazioni come ITU-T, ISO, IEEE e IAB gestiscono porzioni differenti del processo di standardizzazione.

Problemi

1. Immaginiamo di aver addestrato Bernie, il nostro San Bernardo, a trasportare una scatola con tre nastri da 8mm invece di una bottiglia di brandy (consideriamo un'emergenza il momento in cui si riempie il disco fisso). Ciascuno di questi nastri contiene 7 gigabyte. Il cane può viaggiare al nostro fianco ovunque, alla velocità di 18 km all'ora. Per quale distanza Bernie ha un data rate maggiore di una linea di trasmissione con data rate (overhead escluso) di 150 Mbps?
2. Un'alternativa alla LAN è un sistema a divisione di tempo con un terminale per ciascuno degli utenti. Descrivere due vantaggi di un sistema client-server basato su una LAN.
3. Le prestazioni di un sistema client-server sono influenzate da due caratteristiche della rete: ampiezza di banda (quanti bit/sec può trasportare) e latenza (quanti secondi impiega il primo bit per passare dal client al server). Fare un esempio di rete che ha banda e latenza elevate, e di una che ha banda e latenza basse.
4. Oltre a banda e latenza, quali parametri servono per dare una buona descrizione della qualità di servizio di una rete usata per il traffico vocale digitalizzato?
5. Una parte del ritardo di un sistema store-and-forward a commutazione di pacchetto è dovuta al tempo che occorre per memorizzare e inoltrare un pacchetto attraverso uno switch. Se il tempo di commutazione fosse 10 microsecondi, sarebbe una frazione significativa del tempo di risposta di un sistema client-server dove il client è a New York e il server in California? Assumere che la velocità di propagazione nel rame e nella fibra ottica sia 2/3 della velocità della luce nel vuoto.
6. Un sistema client-server usa una rete satellitare, dove il satellite si trova a un'altezza di 40.000 km. Qual è il più breve ritardo possibile in risposta a una richiesta?
7. In futuro, quando tutti avranno un terminale domestico collegato a una rete di computer, sarà possibile condurre referendum pubblici immediati su importanti leggi in corso di approvazione. In prospettiva sarebbe possibile eliminare le attuali legislature e lasciare che la volontà del popolo si esprima in modo diretto. Gli aspetti positivi di questa democrazia diretta sono evidenti; discutere alcuni degli aspetti negativi.
8. Un insieme di cinque router è collegato a formare una subnet punto-punto. Tra ogni coppia di router i progettisti possono collocare una linea ad alta, media o bassa velocità, oppure nessuna linea. Se il computer impiega 100 ms per generare e ispezionare ciascuna topologia, quanto tempo occorre per ispezionarle tutte?
9. Un gruppo di $2^n - 1$ router è interconnesso in un albero binario centralizzato, con un router per ogni nodo dell'albero. Il router i comunica con il router j mandando un messaggio alla radice dell'albero; la radice quindi manda indietro il messaggio fino a j . Trovare un'espressione approssimata per il numero medio di salti per messaggio con n grande, supponendo che tutte le coppie di router abbiano la stessa probabilità.
10. Uno svantaggio delle subnet broadcast è lo spreco di capacità quando più host tentano di accedere al canale simultaneamente. Come esempio semplicistico, supporre che il tempo sia diviso in slot discreti, e durante ciascuno di essi ciascuno degli n host tenti di usare il canale con probabilità p . Quale frazione degli slot va sprecata a causa delle collisioni?

11. Dare due motivi per usare protocolli organizzati in strati.
12. Il presidente della Vernici Speciali s.p.a. ha l'idea di collaborare con la birreria locale per realizzare una lattina di birra invisibile (come metodo per ridurre l'inquinamento). Il presidente dà l'incarico di esaminare la questione all'ufficio legale, che a sua volta chiede aiuto agli ingegneri. Come risultato, il capo degli ingegneri chiama la sua controparte nell'altra azienda per discutere degli aspetti tecnici del progetto. Gli ingegneri di ciascuna azienda quindi riferiscono i risultati al proprio ufficio legale, e i due fanno una telefonata per discutere degli aspetti legali. Infine i presidenti delle due società discutono gli aspetti finanziari dell'accordo. Questo è un esempio di protocollo a più strati conforme al modello OSI?
13. Qual è la differenza principale tra comunicazione senza connessione e comunicazione orientata alla connessione?
14. Due reti offrono un servizio affidabile orientato alla connessione. Una delle due offre un byte stream affidabile, e l'altra un message stream affidabile. Sono identiche? Se si, indicare perché viene fatta una distinzione tra le due; se no, dare un esempio del perché differiscono.
15. Cosa significa "negoziazione" nel contesto dei protocolli di rete? Dare un esempio.
16. Nella Figura 1.19 è mostrato un servizio. In questa figura sono impliciti altri servizi? In caso affermativo indicare dove, in caso negativo spiegare il perché.
17. In alcune reti lo strato data link tratta gli errori di trasmissione chiedendo una nuova trasmissione dei frame danneggiati. Se la probabilità che un frame venga danneggiato è p , qual è il numero medio di trasmissioni necessarie per mandare un frame? Ipotizzare che gli acknowledge non vadano mai persi.
18. Indicare quale strato OSI si occupa delle seguenti attività:
 - (a) dividere in frame il bit stream trasmesso
 - (b) stabilire il percorso da utilizzare attraverso la subnet.
19. Se l'unità di scambio a livello data link si chiama frame e quella a livello network pacchetto, i frame incapsulano i pacchetti o i pacchetti incapsulano i frame? Spiegare la risposta.
20. Un sistema ha una gerarchia di protocolli a n -strati. Le applicazioni generano messaggi di lunghezza M byte. A ogni strato è aggiunta un'intestazione di h byte. Che frazione della banda della rete è impegnata da intestazioni?
21. Elenicare due punti dove il modello di riferimento OSI e quello TCP/IP sono identici, e due dove differiscono.
22. Qual è la principale differenza tra TCP e UDP?
23. La subnet della Figura 1.25(b) fu progettata per resistere a una guerra nucleare. Quante bombe sono necessarie per partizionare i nodi in due insiemi separati? Supporre che ogni bomba distrugga un nodo e tutti i collegamenti che vi fanno capo.
24. A grandi linee Internet raddoppia le sue dimensioni ogni 18 mesi. Anche se nessuno ne è certo, una stima degli host nel 2001 è di 100 milioni. Usare questi dati per prevedere il numero di host Internet nell'anno 2010. È credibile? Spiegare perché sì o perché no.

25. Quando si trasferisce un file tra due computer sono possibili due politiche di acknowledge. Nella prima il file è diviso in pacchetti, a ciascuno dei quali il ricevente risponde con un'acknowledge. Nella seconda i pacchetti non ricevono un'acknowledge individuale, ma si conferma l'arrivo dell'intero file. Discutere i due approcci.
26. Perché ATM usa celle piccole con dimensione fissa?
27. Quanto era lungo, in metri, un bit nello standard 802.3 originale? Utilizzare una velocità di trasmissione di 10 Mbps e supporre che la velocità di propagazione nel cavo coassiale impiegato sia 2/3 della velocità della luce nel vuoto.
28. Un'immagine ha 1.024 x 768 pixel con 3 byte per pixel. Supporre che l'immagine non sia compressa. Quando tempo impiega per essere trasmessa da un modem su un canale a 56 kbps? E da un cable modem da 1 Mbps? Da una Ethernet a 10 Mbps? Da una Ethernet a 100 Mbps?
29. Ethernet e le reti wireless hanno elementi simili e altri diversi. Una proprietà di Ethernet è che sul mezzo può essere trasmesso solo un frame per volta. 802.11 ha la stessa caratteristica? Discutere la risposta.
30. Le reti wireless sono semplici da installare, e ciò le rende economiche poiché il costo d'installazione di solito supera di gran lunga quello delle apparecchiature. Nonostante ciò hanno alcuni svantaggi: indicarne due.
31. Indicare due vantaggi e due svantaggi dell'esistenza di standard per i protocolli di rete.
32. Quando un sistema ha una parte fissa e una removibile (per esempio il lettore Cd-Rom e il disco Cd-Rom) è importante che sia standardizzato, in modo tale che più aziende possano costruire la parte fissa e quella removibile garantendo il funzionamento. Dare tre esempi fuori dal campo dell'industria del computer dove esiste uno standard internazionale simile, e altrettanti dove non esiste.
33. Elencare le proprie attività quotidiane che implicano l'uso di una rete di computer. Come cambierebbe la propria vita se queste reti fossero improvvisamente speinte?
34. Scoprire quali reti sono usate nella propria scuola o posto di lavoro. Descrivere il tipo di rete, la topologia e i metodi di commutazione.
35. Il programma ping permette l'invio in un luogo specifico di un pacchetto di test e la misura del tempo che occorre per arrivare a destinazione e ritornare. Provare a usare ping per vedere quanto tempo serve a raggiungere diversi luoghi noti, dalla propria postazione. Da questi dati, tracciare il tempo di transito unidirezionale attraverso Internet in funzione della distanza. È meglio usare università poiché la posizione dei loro server è nota con grande precisione. Per esempio *berkeley.edu* è a Berkeley, California, *mit.edu* è a Cambridge, Massachusetts, *vu.nl* è ad Amsterdam, Paesi Bassi, *www.usyd.edu.au* è a Sydney, Australia, e *www.uct.ac.za* è a Città del Capo, Sud Africa.
36. Andare al sito Web di IETF, www.ietf.org, per vedere le attività in corso. Scegliere un progetto di proprio gradimento e scrivere una relazione di mezza pagina sul problema e sulla soluzione proposta.

37. La standardizzazione è molto importante nel mondo del networking. ITU e ISO sono le organizzazioni di standardizzazione principali. Andare sui rispettivi siti Web (www.itu.org e www.iso.org) per informarsi sulle attività di standardizzazione. Scrivere una breve relazione sul genere di cose che hanno standardizzato.
38. Internet è composta da un gran numero di reti, e la loro struttura determina la topologia di Internet. On-line sono disponibili un gran numero d'informazioni sulla topologia di Internet: usare un motore di ricerca per rintracciarle e scrivere una breve relazione che riassume quanto scoperto.

2

Lo strato fisico

Questo capitolo esamina lo strato più basso della gerarchia rappresentata nella Figura 1.24. Lo strato fisico definisce le interfacce meccaniche, elettriche e le temporizzazioni della rete. La prima parte del capitolo descrive la teoria della trasmissione dei dati e spiega in che modo la natura limita ciò che può essere trasmesso e ricevuto attraverso un canale.

I paragrafi successivi studiano tre mezzi di trasmissione: quelli guidati (cavi in rame e fibra ottica), quelli wireless (onde radio terrestri) e quelli basati sui satelliti. Questo materiale fornisce le informazioni di base sulle principali tecnologie di trasmissione utilizzate nelle reti moderne.

La parte finale del capitolo esamina tre esempi pratici di sistemi di comunicazione utilizzati per reti geografiche di computer: il sistema di telefonia su rete fissa, il sistema di telefonia mobile e il sistema di televisione via cavo. Tutti e tre utilizzano dorsali in fibra ottica, ma sono organizzati differentemente e adoperano tecnologie diverse nell'ultimo tratto del collegamento.

2.1 Le basi teoriche della comunicazione dati

Le informazioni possono essere trasmesse via cavo variando alcune proprietà fisiche, per esempio la tensione o la corrente. Rappresentando il valore di questa tensione o corrente attraverso una funzione a un valore, $f(t)$, è possibile modellare il comportamento del segnale e analizzarlo matematicamente. Questa analisi sarà l'argomento dei prossimi paragrafi.

2.1.1 Analisi di Fourier

All'inizio del diciannovesimo secolo, il matematico francese Jean-Baptiste Fourier dimostrò che qualunque funzione periodica sufficientemente regolare $g(t)$ con periodo T può essere ottenuta sommando un numero idealmente infinito di seni e coseni:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft) \quad (2.1)$$

dove $f=1/T$ rappresenta la frequenza fondamentale, a_n e b_n sono rispettivamente le ampiezze seno e coseno della n -esima armonica, e c rappresenta una costante. Questa scomposizione è chiamata anche **serie di Fourier**. La funzione può essere ricostruita a partire dalla sua serie di Fourier: se il periodo T è noto e le ampiezze sono definite, la funzione originale del tempo si ricava eseguendo le somme della equazione 2.1.

Un segnale che ha una durata finita (e questa è una caratteristica comune a tutti segnali) può essere gestito immaginando semplicemente che esso ripeta infinite volte l'intero schema, quindi l'intervallo compreso tra T e $2T$ è identico all'intervallo che va da 0 a T , e così via.

Le ampiezze a_n si possono calcolare per ogni $g(t)$ dato, moltiplicando entrambe le parti dell'equazione (2.1) per $\sin(2\pi kft)$ e poi integrando da 0 a T . Poiché

$$\int_0^T \sin(2\pi kft) \sin(2\pi nft) dt = \begin{cases} 0 & \text{per } k \neq n \\ T/2 & \text{per } k = n \end{cases}$$

alla fine rimane solo un termine della sommatoria: a_n . La sommatoria b_n sparisce completamente.

In modo analogo è possibile derivare b_n moltiplicando l'equazione (2.1) per $\cos(2\pi kft)$ e integrando tra 0 e T . Per trovare c si integrano entrambe le parti dell'equazione così com'è. I risultati ottenuti eseguendo queste operazioni sono elencati di seguito:

$$a_n = \frac{2}{T} \int_0^T g(t) \sin(2\pi nft) dt \quad b_n = \frac{2}{T} \int_0^T g(t) \cos(2\pi nft) dt \quad c = \frac{2}{T} \int_0^T g(t) dt$$

2.1.2 Segnali a banda limitata

Per comprendere meglio in che modo tutto ciò influenza la comunicazione dei dati, consideriamo un esempio specifico: la trasmissione del carattere ASCII "b" codificato in un byte (composto da 8 bit). La sequenza dei bit da trasmettere è 01100010. La prima parte della Figura 2.1(a) mostra la tensione di uscita del computer trasmittente. L'analisi di Fourier di questo segnale produce i coefficienti:

$$a_n = \frac{1}{\pi n} [\cos(\pi n/4) - \cos(3\pi n/4) + \cos(6\pi n/4) - \cos(7\pi n/4)]$$

$$b_n = \frac{1}{\pi n} [\sin(3\pi n/4) - \sin(\pi n/4) + \sin(7\pi n/4) - \sin(6\pi n/4)]$$

$$c = \frac{3}{4}$$

Lo strato fisico

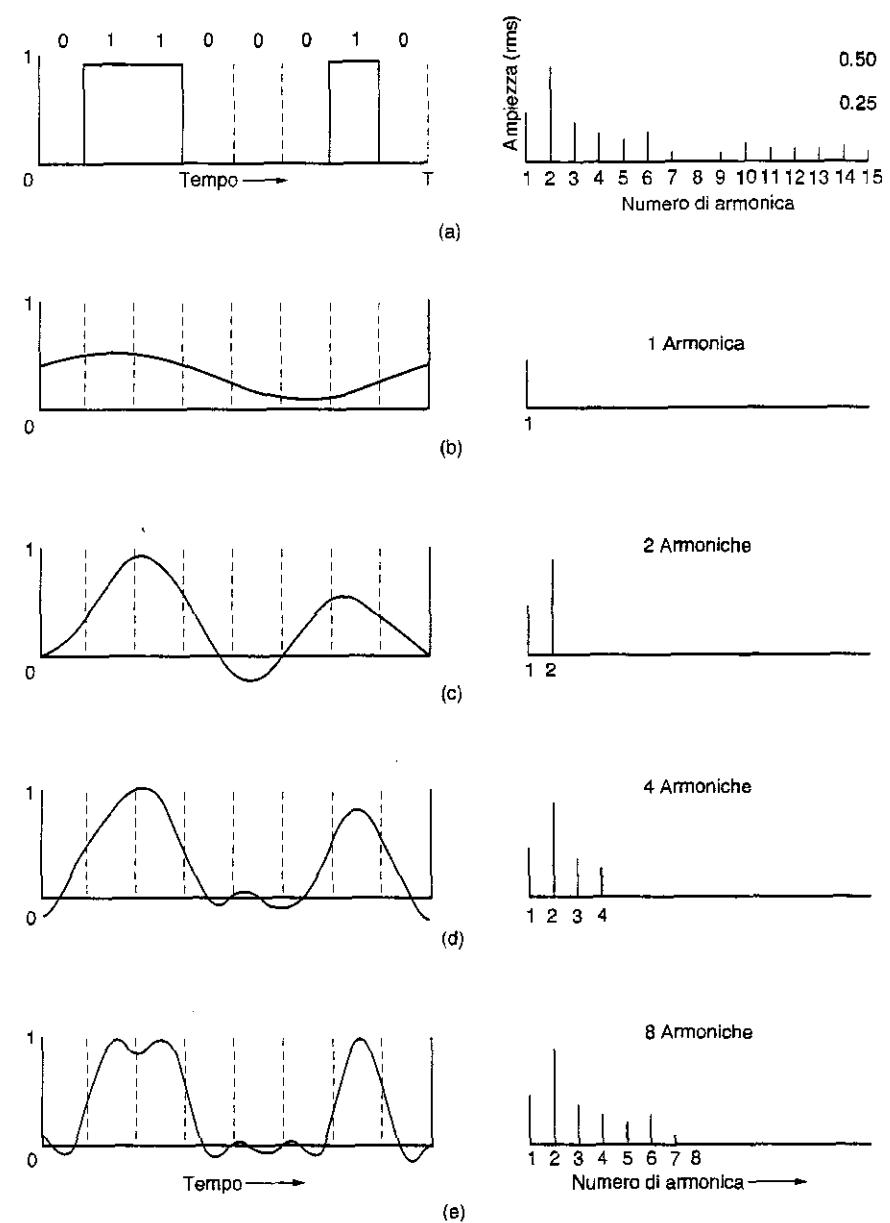


Figura 2.1 (a) Un segnale binario e le ampiezze rms dei coefficienti di Fourier.
(b)-(e) Approssimazioni successive del segnale originale.

Le ampiezze rms (*root mean square*, valore quadratico medio) $\sqrt{a_n^2 + b_n^2}$ relative ai primi termini sono visualizzate nella seconda parte della figura. Questi valori sono interessanti perché i loro quadrati sono proporzionali all'energia trasmessa alla frequenza corrispondente. Nessun mezzo di trasmissione è in grado di trasmettere segnali senza perdere parte dell'energia durante il processo. Se tutti i componenti di Fourier fossero attenuati in modo uniforme, il segnale risultante verrebbe ridotto in ampiezza ma non risulterebbe distorto, quindi avrebbe la stessa forma quadrata rappresentata nella Figura 2.1. Sfortunatamente i mezzi di trasmissione non attenuano i componenti della serie di Fourier in modo uniforme, e ciò genera una distorsione. Di solito le ampiezze sono trasmesse senza modifiche da 0 fino a una certa frequenza f_c (misurata in cicli al secondo o Hertz) e attenuate per tutte le frequenze superiori a questo limite. L'intervallo di frequenze trasmesse senza una forte attenuazione è chiamato **banda passante**, o semplicemente banda. Nella realtà il limite non è molto preciso, perciò spesso viene indicata la banda passante compresa tra 0 e la frequenza dove la potenza è attenuata del 50%.

La banda passante è una proprietà fisica del mezzo di trasmissione e di solito dipende dalla costruzione, dallo spessore e dalla lunghezza del mezzo. In alcuni casi nel circuito s'introduce un filtro per limitare l'ampiezza della banda a disposizione di ogni utente. Per esempio, un cavo telefonico può avere un'ampiezza di banda di 1 MHz per distanze brevi, ma le compagnie telefoniche aggiungono un filtro che riduce a circa 3.100 Hz l'ampiezza di banda per ogni utente: questo valore è adatto alla trasmissione della voce, e migliora l'efficienza dell'intero sistema limitando l'utilizzo della risorsa.

Che aspetto avrebbe il segnale rappresentato nella Figura 2.1(a) se la banda disponibile fosse così scarsa da consentire solo la trasmissione delle frequenze più basse, ossia se la funzione fosse approssimata utilizzando soltanto i primi termini dell'equazione (2.1)? La Figura 2.1(b) mostra il segnale generato su un canale che fa passare solo la prima armonica (quella fondamentale, f). In modo analogo, le Figure 2.1(c), 2.1(d) e 2.1(e) mostrano lo spettro e le funzioni ricostruite per i canali di ampiezza di banda più elevata.

Data una frequenza di bit pari a b bit/sec, il tempo richiesto per inviare 8 bit, un bit alla volta, è $8/b$ sec, perciò la frequenza della prima armonica è $b/8$ Hz. Una linea telefonica ordinaria, detta anche **linea voice-grade**, ha una frequenza limite introdotta artificialmente appena sopra i 3.000 Hz. Questa restrizione indica che il numero della più alta armonica che attraversa il canale di trasmissione è approssimativamente $3.000/(b/8)$ o $24.000/b$ (il limite non è preciso).

Per alcune cadenze dati i numeri si comportano come mostrato nella Figura 2.2. Da questi valori appare evidente che quando si tenta di trasmettere dati a 9.600 bps attraverso una linea telefonica voice-grade, il segnale mostrato nella Figura 2.1(a) si trasforma in qualcosa di simile al segnale della Figura 2.1(c), situazione che rende particolarmente difficile ricevere il flusso di bit originario. Dovrebbe apparire ovvio che i segnali digitali non hanno alcuna speranza di transitare a velocità superiori ai 38,4 kbps, anche quando il sistema di trasmissione è privo di rumore. In altre parole, il limite imposto all'ampiezza di banda limita la velocità dei dati anche su canali perfetti, ma esistono schemi di codifica sofisticati (che fanno uso di diversi livelli di tensione) per ottenere velocità di trasmissione dati superiori. Questo argomento sarà trattato più avanti.

Bps	T (msec)	Prima armonica (Hz)	Numero di armoniche trasmesse
300	26,67	37,5	80
600	13,33	75	40
1.200	6,67	150	20
2.400	3,33	300	10
4.800	1,67	600	5
9.600	0,83	1.200	2
19.200	0,42	2.400	1
38.400	0,21	4.800	0

Figura 2.2. Relazione tra velocità dei dati e armoniche.

2.1.3 La velocità massima di un canale

Nel 1924 Henry Nyquist, ingegnere di AT&T, si rese conto che anche un canale perfetto ha una capacità di trasmissione limitata e ottenne un'equazione che esprime la massima velocità di trasmissione dati di un canale senza rumore ad ampiezza di banda limitata. Nel 1948 Claude Shannon portò avanti il lavoro di Nyquist, estendendolo al caso di un canale soggetto a rumore casuale (termodynamico) (Shannon, 1948). Questo paragrafo riepiloga brevemente i risultati dei loro studi.

Nyquist dimostrò che se si trasmette un segnale arbitrario attraverso un filtro low-pass la cui ampiezza di banda è pari a H , il segnale filtrato può essere ricostruito completamente prendendo solo $2H$ campioni al secondo. Campionare la linea più velocemente di $2H$ volte al secondo non ha senso, perché le componenti delle frequenze superiori sono già state filtrate via. Se il segnale è composto da V livelli discreti, il teorema di Nyquist afferma che:

$$\text{velocità massima} = 2H \log_2 V \text{ bit/sec}$$

Per esempio, un canale a 3 kHz senza rumore non è in grado di trasmettere segnali binari a velocità maggiore di 6.000 bps.

Fino a questo momento sono stati considerati solo i canali senza rumore; se sul canale è presente un rumore casuale la situazione peggiora rapidamente, ma purtroppo il rumore casuale (termico) causato dal movimento delle molecole del sistema è sempre presente. Il livello di rumore termico si misura facendo il rapporto tra la potenza del segnale e la potenza del rumore (detto anche **rapporto segnale-rumore**). Chiamando S la potenza del segnale ed N la potenza del rumore, il rapporto segnale rumore è pari a S/N . Di solito, invece d'indicare direttamente il rapporto si cita la quantità $10 \log_{10} S/N$ che è misurata in **decibel (dB)**.

Poiché il rapporto S/N di 10 è pari a 10 dB, un rapporto uguale a 100 è pari a 20 dB, un rapporto uguale a 1.000 equivale a 30 dB e così via. I produttori di amplificatori stereo spesso caratterizzano la banda passante (intervallo di frequenze) in cui l'apparecchio è lineare indi-

cando i due estremi di frequenza dove l'attenuazione è pari a 3 dB: sono i punti dove il fattore di amplificazione è approssimativamente dimezzato (perché $\log_{10} 3 \approx 0,5$).

Il risultato principale ottenuto da Shannon si può riassumere così: la cadenza dati massima in bit/sec su un canale rumoroso la cui ampiezza di banda è di H Hz e il cui rapporto segnale-rumore è pari a S/N , è dato dal numero

$$\text{massimo numero di bit/sec} = H \log_2 (1 + S/N)$$

Per esempio, un canale che ha una banda passante di 3.000 Hz e un rapporto segnale-rumore di 30 dB (parametri tipici per la parte analogica del sistema telefonico) non può trasmettere più di 30.000 bps, indipendentemente dal numero di livelli di segnale utilizzati o dalla frequenza di campionamento. Il risultato di Shannon deriva dagli argomenti della teoria dell'informazione e si applica a qualunque canale soggetto a rumore termico. Esempi contrari devono essere considerati alla stregua di quelli utilizzati per descrivere le macchine a moto perpetuo. Va anche ricordato che questo valore è solo un limite superiore, che raramente i sistemi reali riescono a raggiungere.

2.2 Mezzi di trasmissione guidati

Lo scopo dello strato fisico è trasportare un flusso grezzo di bit da una macchina a un'altra. È possibile utilizzare diversi tipi di mezzi fisici per realizzare una trasmissione; ognuno è caratterizzato da specifica banda passante, ritardo, costo e facilità d'installazione e manutenzione. I mezzi di trasmissione sono generalmente divisi in mezzi guidati (cavi in rame, fibre ottiche e così via) e mezzi non guidati (onde radio, laser ecc.). I prossimi paragrafi esaminano in dettaglio le loro caratteristiche.

2.2.1 Mezzi magnetici

Uno dei sistemi più comuni adottati per trasferire i dati da un computer a un altro funziona così: si scrivono le informazioni su un nastro magnetico o su un supporto rimovibile (per esempio un DVD), si trasporta fisicamente il nastro o il disco, e infine si utilizza l'apposita unità installata nel computer di destinazione per leggere i dati. Anche se non è sofisticato come la comunicazione satellitare geosincrona, questo metodo è spesso più economico, soprattutto nel caso di applicazioni in cui un'elevata ampiezza di banda o il costo per bit trasmesso rappresentano i fattori chiave.

Un semplice calcolo renderà tutto più chiaro. Un nastro in standard Ultrium può immagazzinare 200 GB di dati; un alloggiamento grande 60 x 60 x 60 cm può contenere 1.000 di questi nastri, per una capacità totale di 200 TB (terabyte) o 1.600 terabit (pari a 1,6 petabit). Una scatola di nastri come questa può essere spedita ovunque negli Stati Uniti in 24 ore con Federal Express o un altro corriere espresso. L'ampiezza di banda effettiva di questa trasmissione è pari a 1.600 terabit/86.400 sec, ossia 19 Gbps. Se la destinazione si trova a un'ora di strada, l'ampiezza di banda supera i 400 Gbps. Nessuna rete di computer è in grado di raggiungere questa velocità.

Per una banca che ha la necessità di trasferire ogni giorno il backup di molti GB di dati su una seconda macchina (soluzione che permette alla banca di funzionare anche in caso di

terremoto o alluvione), probabilmente nessun'altra tecnologia di trasmissione è in grado di raggiungere le stesse prestazioni dei nastri magnetici. Certo, le reti stanno diventando sempre più veloci, ma anche la densità dei nastri sta aumentando.

Un quadro analogo si ottiene analizzando il costo dell'operazione: un singolo nastro costa circa 40 \$ può essere utilizzato almeno dieci volte, perciò il costo di un alloggiamento (200 TB di dati) è di circa 4.000 \$ per utilizzo; se a questa cifra si aggiungono i circa 1.000 \$ (e magari molto meno) per il trasporto si ottiene un costo totale di 5.000 \$. Dividendo questa cifra per il numero di GB di dati trasferiti si ottiene un costo di circa 3 centesimi di \$ per GB. Nessuna rete può battere questa cifra. La morale di tutto questo è:

Mai sottovalutare l'ampiezza di banda di una station wagon piena di nastri lanciata a tutta velocità lungo l'autostrada.

2.2.2 Il doppino

L'ampiezza di banda del nastro magnetico è eccellente, ma il suo ritardo è eccessivo; la durata della trasmissione è espressa in minuti e in ore, non in millisecondi. Molte applicazioni richiedono una connessione on-line: uno dei mezzi di trasmissione più vecchi, ma ancora molto in voga, è il **doppino**.

Il doppino è composto da due conduttori di rame isolati, spessi circa 1 mm, avvolti uno intorno all'altro in una forma elicoidale che ricorda un po' quella della molecola del DNA. L'intreccio è utilizzato perché due cavi paralleli formano un'eccellente antenna; quando invece i cavi sono intrecciati, i campi elettromagnetici generati dai due conduttori si annullano a vicenda, perciò il cavo irradia meno.

L'applicazione più comune del doppino è il sistema telefonico. Quasi tutti i telefoni sono collegati alla centrale telefonica attraverso un doppino. I doppiini possono estendersi per diversi chilometri senza richiedere un'amplificazione del segnale, ma per distanze più lunghe è necessario installare dei ripetitori. Quando molti doppiini procedono in parallelo per lunghe distanze, come accade per i cavi che collegano uno stabile alla centrale telefonica, è necessario fasciare insieme i cavi rivestendoli con una guaina protettiva; se non ci fosse l'intreccio, i doppiini in questi fasci interferirebbero l'uno con l'altro. Nelle zone in cui le linee telefoniche sono aeree, capita di vedere sui pali telefonici fasci di cavi con diametro di alcuni centimetri.

I doppiini si possono usare per trasmettere segnali analogici e digitali. L'ampiezza di banda dipende dal diametro del cavo e dalla distanza percorsa, ma il più delle volte per tratti lunghi pochi chilometri è possibile raggiungere velocità di alcuni Mb al secondo. Per il loro basso costo e il discreto livello di prestazioni, i doppiini sono largamente utilizzati e probabilmente lo saranno anche per gli anni a venire.

Esistono diverse varietà di doppiini, due delle quali molto importanti per le reti di computer. I doppiini di categoria 3 sono composti da due cavi isolati attorcigliati. Di solito quattro coppie di cavi sono raggruppate in una guaina di plastica che protegge e tiene uniti i conduttori. Prima del 1988, la maggior parte degli uffici aveva un sistema di cavi di categoria 3 che partivano dalla centralina principale di ciascun piano e arrivavano a ogni ufficio

cio. Questo schema permetteva di collegare ogni ufficio alla rete telefonica attraverso quattro linee telefoniche analogiche o due connessioni multilinea.

I doppini di categoria 5 sono stati introdotti a partire dal 1988. Assomigliano ai cavi di categoria 3, ma usano più spire per centimetro; questa soluzione riduce l'interferenza e migliora la qualità del segnale trasmesso su lunghe distanze. I cavi di categoria 5 sono perciò più adatti alle comunicazioni ad alta velocità. I cavi di categoria superiore, 6 e 7, possono gestire segnali con banda ampia rispettivamente di 250 MHz e 600 MHz (contro i 16 e i 100 MHz dei cavi di categoria 3 e 5).

Tutti questi tipi di cavi sono chiamati anche UTP (*Unshielded Twisted Pair*, doppini non schermati) per distinguerli dai più grossi e costosi doppini schermati introdotti da IBM all'inizio degli anni Ottanta, e utilizzati prevalentemente nelle installazioni di IBM. La Figura 2.3 mostra alcuni esempi di doppini.



Figura 2.3. (a) UTP Categoria 3. (b) UTP Categoria 5.

2.2.3 Cavo coassiale

Un altro mezzo di trasmissione molto comune è il **cavo coassiale**. Essendo più schermato del doppino, il cavo coassiale può estendersi per distanze più lunghe e consente velocità più elevate. Esistono due tipi di cavi coassiali: il primo, a 50 Ohm, è utilizzato per le trasmissioni digitali; il secondo, a 75 Ohm, è utilizzato per le trasmissioni analogiche, per la televisione e le connessioni Internet via cavo. Questa distinzione dipende più da fattori storici che da motivi tecnici.

Un cavo coassiale è composto da un nucleo conduttore coperto da un rivestimento isolante, a sua volta circondato da un conduttore cilindrico solitamente realizzato con una calza di conduttori sottili, che infine è avvolto da una guaina protettiva di plastica. La Figura 2.4 mostra uno spaccato del cavo coassiale. La costruzione e la schermatura del cavo coassiale forniscono ampiezza di banda ed eccellente immunità al rumore. La banda disponibile dipende dalla qualità, dalla lunghezza del cavo e dal rapporto segnale-rumore del segnale dati; i cavi moderni hanno un'ampiezza di banda vicina a 1 GHz. I cavi coassiali furono ampiamente usati dalle aziende telefoniche per le connessioni sulle lunghe distanze, ma in seguito sono stati sostituiti dalla fibra ottica nelle tratte più lunghe. Il cavo coassiale è ancora molto utilizzato per le reti metropolitane e le televisioni via cavo.

Lo strato fisico

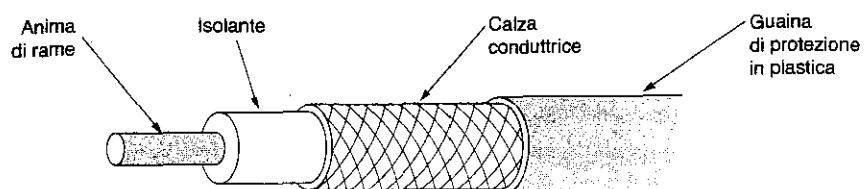


Figura 2.4. Un cavo coassiale.

2.2.4 Fibra ottica

Nell'industria dei computer molti sono orgogliosi del rapido sviluppo della tecnologia informatica. Il primo PC IBM (prodotto nel 1981) aveva una velocità di clock pari a 4,77 MHz. Venti anni dopo, i PC funzionano a 2 GHz, con un fattore di miglioramento di 20 per decennio. Niente male!

Nello stesso periodo le comunicazioni geografiche sono passate dai 56 kbps (ARPANET) a 1 Gbps (le moderne comunicazioni ottiche), con un fattore di miglioramento superiore a 125 per decade; contemporaneamente la frequenza degli errori è passata da 10^{-5} per bit a circa 0.

Mentre le singole CPU stanno raggiungendo i limiti fisici relativi alla velocità della luce e ai problemi di dissipazione termica, con l'*attuale* tecnologia della fibra l'ampiezza di banda raggiungibile va sicuramente oltre i 50.000 Gbps (50 Tbps) e molte persone stanno studiando tecnologie e materiali migliori. Oggi il limite pratico per la trasmissione dei segnali, pari a circa 10 Gbps, deriva dall'incapacità di convertire più velocemente il segnale elettrico in segnale ottico. Esperimenti eseguiti in laboratorio hanno permesso di raggiungere una velocità di 100 Gbps su una singola fibra.

La gara tra la potenza di calcolo e la velocità di comunicazione è stata vinta dalla comunicazione. Tutte le implicazioni di un'ampiezza di banda essenzialmente infinita (sebbene non gratuita) non sono ancora penetrate in una generazione d'ingegneri e scienziati abituati a pensare nei termini dei modesti limiti di Nyquist e Shannon imposti dal cavo di rame. In base alla nuova saggezza formale, tutti i computer sono disperatamente lenti e le reti dovrebbero evitare l'elaborazione anche a costo di sprecare ampiezza di banda. Questo paragrafo esamina la fibra ottica e spiega come funziona la corrispondente tecnologia di comunicazione.

Un sistema di trasmissione ottico è formato da tre componenti fondamentali: la sorgente luminosa, il mezzo di trasmissione e il rilevatore. Per convenzione, un impulso di luce indica il valore 1 e l'assenza di luce indica il valore 0. Il mezzo di trasmissione è una fibra di vetro sottilissima, realizzata in silicio. Quando viene colpito dalla luce, il rilevatore genera un impulso elettrico. Collegando a un'estremità della fibra una sorgente di luce e un rivelatore all'altro, si crea un sistema di trasmissione unidirezionale che accetta un segnale elettrico, lo converte e lo trasmette sotto forma di impulso luminoso; all'altra estremità della fibra converte nuovamente l'output in segnale elettrico.

Questo sistema di trasmissione è utile perché un interessante principio della fisica gli consente di non disperdere la luce: infatti, quando un raggio luminoso passa da un materiale a un altro, per esempio dal silicio fuso all'aria, si rifrange (si curva) sul confine tra i due materiali come mostrato nella Figura 2.5(a). La figura mostra un raggio incidente che colpisce il confine con un'inclinazione di α_1 gradi ed emerge con un'inclinazione di β_1 gradi. L'entità della rifrazione dipende dalle proprietà dei due materiali, e in particolare dal loro indice di rifrazione; per angoli d'incidenza che superano un particolare valore critico, la luce rimane nel silicio senza fuggire nell'aria.

Per questo motivo un raggio di luce con angolo d'incidenza pari o superiore all'angolo critico rimane intrappolato nella fibra, come mostrato nella Figura 2.5(b) e può propagarsi per molti chilometri senza perdite significative.

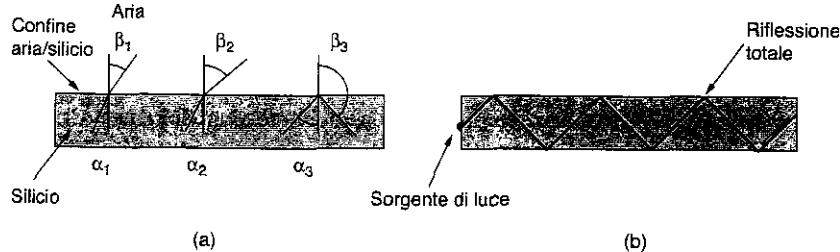


Figura 2.5 (a) Tre esempi di un raggio di luce che dall'interno di una fibra di silicio colpisce il confine aria/silicio con diversi angoli d'incidenza. (b) La luce rimane intrappolata a causa della riflessione totale.

Il disegno rappresentato nella Figura 2.5(b) mostra un solo raggio intrappolato nel mezzo di trasmissione; ma poiché tutti i raggi di luce che colpiscono l'interfaccia con angolo maggiore di quello critico sono riflessi internamente, la fibra può contenere molti raggi che rimbalzano ad angoli diversi. In questo caso si dice che ogni raggio ha una modalità diversa, perciò una fibra dotata di questa proprietà è detta **fibra multimodale**.

Se invece il diametro della fibra viene ridotto fino a poche lunghezze d'onda della luce, la fibra si comporta come una guida d'onda e la luce può propagarsi solo in linea retta, senza rimbalzare; questo tipo di fibra è detta monomodale, è più costosa ed è utilizzata soprattutto sulle lunghe distanze. Le fibre monomodali disponibili possono trasmettere dati a 50 Gbps per 100 Km senza amplificazione. Velocità più elevate sono state raggiunte in laboratorio per brevi distanze.

Trasmissione della luce attraverso la fibra

La fibra ottica è fatta di vetro che, a sua volta, è composto da sabbia, materiale grezzo poco costoso disponibile in quantità illimitate. La fabbricazione del vetro era nota già agli antichi Egizi, ma i loro vetri dovevano essere spessi meno di 1 mm per far passare la luce. Il vetro trasparente adatto alla costruzione di finestre è stato inventato nel Rinascimento. Il vetro uti-

lizzato dalle moderne fibre ottiche è talmente trasparente che, se l'oceano fosse pieno di fibra invece che di acqua, dalla superficie sarebbe possibile osservare il fondale marino così come dal finestrino di un aeroplano si può osservare il terreno in una giornata di sole.

L'attenuazione della luce attraverso il vetro dipende dalla lunghezza d'onda della luce (e da alcune proprietà fisiche del vetro). La Figura 2.6 mostra l'attenuazione tipica del vetro utilizzato nelle fibre; i valori sono espressi in dB per Km lineare di fibra. L'attenuazione espressa in decibel si ricava dalla formula:

$$\text{attenuazione} = 10 \log_{10} \frac{\text{energia trasmessa}}{\text{energia ricevuta}}$$

Per esempio, un fattore di dispersione pari a 2 produce un'attenuazione di $10 \log_{10} 2 = 3$ dB. La figura mostra la parte dello spettro vicina all'infrarosso, che è quella utilizzata nella pratica; la luce visibile ha lunghezze d'onda più corte, che variano dai 0,4 ai 0,7 micron (1 micron = 10^{-6} metri). I puristi del sistema decimale preferiscono indicare queste lunghezze d'onda con valori espressi in nm (400-700 nm).

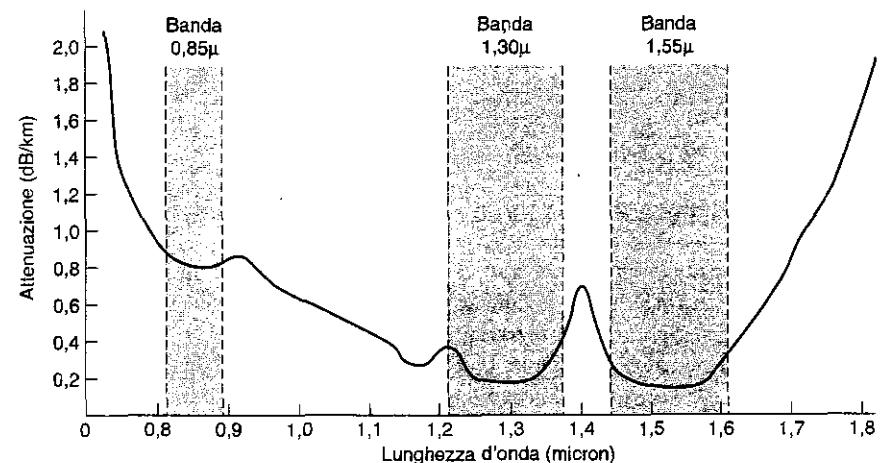


Figura 2.6. Attenuazione della luce attraverso una fibra nella regione dell'infrarosso.

Per la comunicazione ottica si usano tre bande di lunghezze d'onda, centrate rispettivamente a 0,85, 1,30 e 1,55 micron. Le ultime due hanno buone proprietà di attenuazione (meno del 5% di dispersione per Km). La banda 0,85 micron ha un'attenuazione più alta, ma a quella lunghezza d'onda i laser e l'elettronica possono essere costruiti usando lo stesso materiale (arseniuro di gallio). Tutte e tre le bande hanno ampiezze comprese tra i 25.000 e i 30.000 GHz.

Gli impulsi luminosi trasmessi attraverso la fibra si espandono in lunghezza durante la propagazione; il fenomeno si chiama **dispersione cromatica**. Il livello di dispersione cromatica dipende dalla lunghezza d'onda. Per evitare la sovrapposizione degli impulsi diffusi in questo modo è necessario aumentare la distanza tra gli stessi, ma ciò può essere fatto solo riducendo

la velocità di segnale. Per fortuna si è scoperto che creando impulsi di una particolare forma (collegata al reciproco del coseno iperbolico) è possibile annullare quasi tutti gli effetti della dispersione e inviare impulsi per migliaia di chilometri senza che la forma subisca modifiche sensibili. Questi impulsi sono chiamati **solti**. I laboratori di ricerca si stanno impegnando molto per creare applicazioni pratiche basate sui soliti.

Cavi in fibra

I cavi in fibra ottica sono simili ai cavi coassiali, ma non sono avvolti dalla calza conduttrice. La Figura 2.7(a) mostra una singola fibra osservata dal lato. Al centro si trova il nucleo (*core*) di vetro attraverso il quale si propaga la luce; nelle fibre multimodali ha un diametro di 50 micron, circa lo spessore di un cappello umano. Il nucleo delle fibre monomodali ha un diametro che varia dagli 8 ai 10 micron.

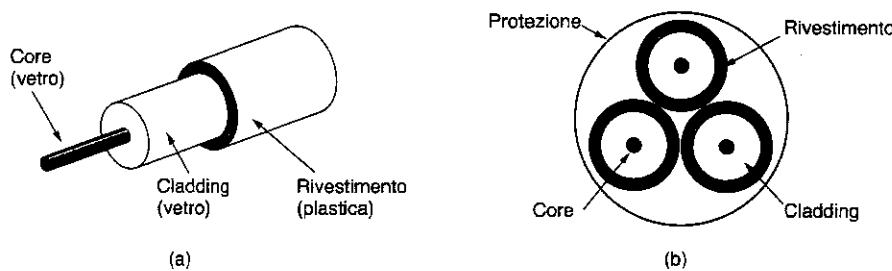


Figura 2.7. (a) Vista laterale di una singola fibra. (b) Vista frontale della guaina con tre fibre.

Il nucleo è circondato da un rivestimento di vetro (*cladding*) che ha un indice di rifrazione più basso; ciò costringe la luce a rimanere nel nucleo. Lo strato successivo è una sottile fodera di plastica che protegge il rivestimento. Le fibre di solito sono raggruppate in fasci, protetti da una guaina più esterna. La Figura 2.7(b) mostra una guaina contenente tre fibre. Le guaine delle fibre terrestri normalmente sono collocate sottoterra a circa un metro dalla superficie, e occasionalmente vengono danneggiate da escavatori o talpe. In prossimità della costa, le guaine delle fibre transoceaniche sono sotterrate in fossati da una spiecie di aratro marino. Nell'acqua profonda queste guaine sono stese sul fondale, e possono quindi essere strappate dai sistemi di pesca a strascico e dai calamari giganti. Le fibre si possono collegare in tre diversi modi.

1. Possono terminare in connettori inseriti in apposite prese; i connettori perdono circa il 10-20% della luce, ma semplificano la riconfigurazione dei sistemi.
2. Possono essere attaccate meccanicamente: l'estremità di un cavo viene appoggiata all'estremità dell'altro; il segmento, dopo essere stato avvolto in una manichetta speciale, viene pinzato. L'allineamento può essere migliorato passando la luce attraverso la giunzione e apportando piccole correzioni che massimizzano il segnale. Personale addestrato può fare una giunzione meccanica in circa 5 minuti; causa una perdita del 10% della luce.

3. Due pezzi di fibra possono essere fusi per formare una connessione solida. Una giuntura di questo tipo è buona quasi quanto una fibra uniforme, ma, anche in questo caso, si genera una piccola attenuazione del segnale.

In tutti e tre i casi, nel punto di giuntura possono presentarsi fenomeni di riflessione, e l'energia riflessa può interferire con il segnale.

Per generare il segnale normalmente si impiegano due tipi di sorgenti luminose: i LED (*Light Emitting Diode*) e i semiconduttori laser. I due dispositivi hanno proprietà differenti (Figura 2.8). La lunghezza d'onda può essere regolata inserendo interferometri Fabry-Perot o Mach-Zehnder tra la sorgente e la fibra. Gli interferometri Fabry-Perot sono semplici cavità risonanti composte da due specchi paralleli; la luce incide perpendicolarmente agli specchi. La lunghezza della cavità esclude le lunghezze d'onda che si propagano un numero intero di volte. Gli interferometri Mach-Zehnder separano la luce in due raggi che viaggiano a distanze leggermente diverse; i raggi sono ricombinati alla fine e sono in fase solo per certe lunghezze d'onda.

Elemento	LED	Laser a semiconduttore
Cadenza dei dati	Bassa	Alta
Tipo di fibra	Multimodale	Multimodale o monomodale
Distanza	Breve	Lunga
Durata	Lunga	Breve
Sensibilità alla temperatura	Scarsa	Elevata
Costo	Basso	Alto

Figura 2.8. Confronto tra le sorgenti luminose realizzate con diodi a semiconduttore e LED.

All'estremità finale della fibra ottica si trova un fotodiodo che genera un impulso elettrico ogni volta che è colpito dalla luce. Il tipico tempo di risposta di un fotodiodo è 1 nsec, perciò la velocità di trasmissione dati è limitata a circa 1 Gbps. Un altro problema è rappresentato dal rumore termico, che permette di rilevare solo gli impulsi luminosi che trasportano energia sufficiente; rendendo gli impulsi sufficientemente potenti, la frequenza di errore può essere resa arbitrariamente piccola.

Reti in fibra ottica

Le fibre ottiche si usano nelle LAN e nei sistemi di trasmissione a lunga distanza, anche se collegarsi a una rete in fibra è più difficile rispetto a una Ethernet. Gli ostacoli possono essere superati realizzando una rete ad anello composta da una serie di connessioni punto a punto (Figura 2.9). L'interfaccia utilizzata da ogni computer passa il flusso di impulsi luminosi verso il collegamento successivo e funge anche da giunzione a T, per dare al computer la capacità di inviare e accettare messaggi. Per realizzare questa struttura si usano due tipi di interfacce. L'interfaccia passiva è composta da due spine fuse nella fibra principale; a un'estremità di una spina si trova un LED o un diodo laser (che trasmette il segnale) e all'estremità dell'altra spina si trova un fotodiodo (che riceve il segnale).

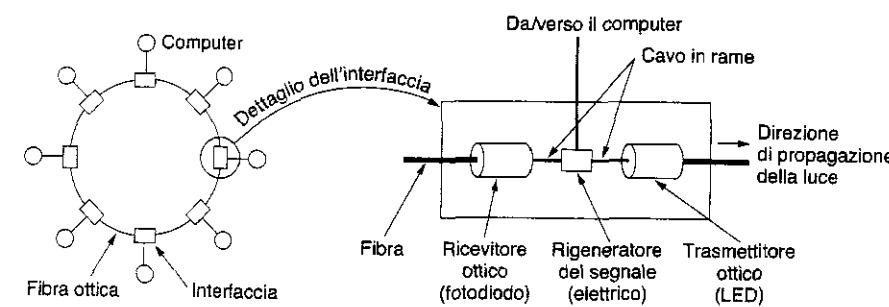


Figura 2.9. Anello in fibra ottica con ripetitori attivi.

La spina stessa è completamente passiva e per questo è estremamente affidabile: infatti un LED o un fotodiodo rotto non interrompono l'anello: semplicemente, isolano un computer dalla connessione.

L'altro tipo di interfaccia, mostrato nella Figura 2.9, è chiamato **ripetitore attivo**. La luce in entrata è convertita in segnale elettrico, rigenerato se necessario alla sua massima potenza, e infine riconvertita e trasmessa sotto forma di segnale luminoso. L'interfaccia collegata al computer è un normale cavo di rame che entra nel rigeneratore di segnale. Oggi sono utilizzati anche ripetitori ottici puri: poiché non richiedono una conversione ottica/elettrica del segnale, questi dispositivi possono operare ad ampiezze di banda estremamente elevate.

Se un ripetitore attivo si guasta, l'anello è interrotto e la rete non funziona più. D'altro canto, poiché il segnale è rigenerato su ogni interfaccia, i collegamenti da computer a computer possono essere lunghi chilometri e la dimensione totale dell'anello non ha virtualmente alcun limite. Le interfacce passive perdono luce a ogni giunzione, perciò il numero di computer e la lunghezza totale dell'anello hanno dei limiti.

Le LAN basate su fibra ottica si possono realizzare anche con altre topologie. La Figura 2.10 mostra una trasmissione realizzata mediante una costruzione a **stella passiva**. In questo schema ogni interfaccia ha una fibra che collega il suo trasmettitore a un cilindro di vetro; le fibre in entrata sono fuse a un capo del cilindro, in modo analogo le fibre utilizzate all'altro capo del cilindro si collegano a ognuno degli altri ricevitori. Ogni volta che un interfaccia emette un impulso luminoso, il segnale è diffuso dentro la stella passiva e illumina tutti i ricevitori, rendendo la trasmissione generalizzata. La stella passiva combina tutti i segnali in entrata e trasmette il segnale risultante su tutte le linee: poiché l'energia in entrata è divisa su tutte le linee in uscita, il numero di nodi nella rete dipende dalla sensibilità dei fotodiodi.

Libre ottiche o cavi in rame?

È interessante confrontare la fibra al cavo in rame. La fibra offre molti vantaggi, primo fra tutti la maggiore ampiezza di banda; già questa caratteristica la rende necessaria per le reti a fascia alta.

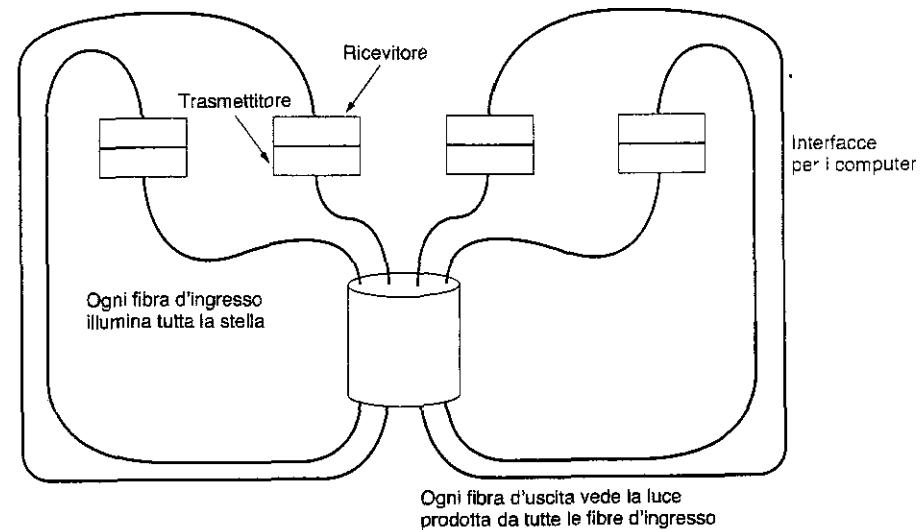


Figura 2.10. Una connessione a stella passiva in una rete a fibre ottiche.

A causa del basso livello di attenuazione, i ripetitori possono essere installati ogni 50 Km di linea, mentre nel caso dei cavi in rame i ripetitori devono essere installati ogni 5 Km. Il risparmio è evidente.

La fibra ha anche il vantaggio di non essere influenzata dalle sorgenti elettriche, dai campi elettromagnetici e dalle interruzioni della linea elettrica. Poiché non è influenzata dai corrosivi chimici presenti nell'aria, la fibra è adatta anche agli ambienti più inospitali. Stranamente, le aziende telefoniche utilizzano la fibra per un altro motivo: perché è sottile e leggera. Molti condotti sono completamente pieni e non c'è spazio per aggiungere altri cavi; però, sostituendo tutti i cavi in rame con cavi in fibra, è possibile svuotare i condotti. Inoltre, poiché il rame ha ancora un elevato prezzo di vendita sul mercato, il materiale recuperato può essere venduto ai raffinatori di rame. La fibra, poi, è molto più leggera del cavo in rame: 1 Km di cavo contenente mille coppie di doppini pesa circa 8.000 Kg, due fibre hanno una capacità superiore e pesano solo 100 Kg; ciò riduce enormemente i costi di cablaggio e manutenzione. Infine, le fibre non perdono la luce ed è piuttosto difficile entrarvi per intercettare i dati; queste proprietà le rendono molto più sicure dei cavi in rame.

D'altro canto, la fibra è una tecnologia meno nota, che richiede conoscenze che non tutti gli ingegneri possiedono, e si può danneggiare se la si piega troppo. Poiché la trasmissione ottica è intrinsecamente unidirezionale, la comunicazione bidirezionale richiede due fibre o due bande di frequenza su una singola fibra. Infine, le interfacce per la fibra ottica costano più di quelle elettriche. Ciò nonostante, la fibra rappresenta sicuramente il futuro di tutte le comunicazioni dati su reti fisse per distanze che superano i pochi metri. Per una discussione di tutti gli aspetti delle fibre ottiche e delle reti che le usano, vedere (Hecht, 2001).

2.3 Trasmissioni Wireless

Questa è l'era dei drogati dell'informazione, persone che hanno la necessità di essere sempre in linea. Per questi utenti mobili doppini, cavo coassiale e fibre ottiche non hanno alcuna utilità: devono consultare i loro dati attraverso computer portatili, palmari e da polso senza dipendere dai sistemi di comunicazione terrestri. Questo tipo di esigenza è soddisfatto solo dalle comunicazioni wireless. I prossimi paragrafi esaminano i concetti generali della comunicazione wireless, che è una tecnologia con applicazioni ben più importanti del fornire connettività agli utenti che vogliono esplorare il Web dalla spiaggia. Alcune persone credono che in futuro esisteranno solo due tipi di comunicazioni: quelle a fibra e quelle wireless. Tutti i computer, i telefoni, i fax e gli altri dispositivi fissi utilizzeranno la fibra, mentre tutti i dispositivi mobili adotteranno le connessioni wireless. In certi casi la trasmissione wireless offre vantaggi anche ai dispositivi fissi. Per esempio, a causa delle caratteristiche del terreno (presenza di montagne, giungle, paludi e così via) risulta difficile far arrivare una fibra a un edificio, può essere più efficiente realizzare una connessione wireless.

È interessante notare che la moderna comunicazione digitale wireless è nata nelle isole Hawaiane: qui gli utenti sono separati da larghi tratti di Oceano Pacifico e il sistema telefonico fisso era inadeguato.

2.3.1 Lo spettro elettromagnetico

Quando si spostano, gli elettroni creano onde elettromagnetiche che si propagano attraverso lo spazio, anche nel vuoto. L'esistenza di queste onde è stata intuita dal fisico inglese James Clerk Maxwell nel 1865; la loro esistenza, comunque, è stata osservata per la prima volta nel 1887 dal fisico tedesco Heinrich Hertz. Il numero di oscillazioni al secondo di un'onda è chiamato **frequenza**, f , ed è misurato in **Hz** (in onore di Heinrich Hertz). La distanza tra due massimi (o minimi) consecutivi è chiamata **lunghezza d'onda** ed è universalmente indicata dalla lettera greca λ (lambda).

Quando un'antenna di dimensioni appropriate è collegata a un circuito elettrico, le onde elettromagnetiche sono trasmesse in modo efficiente e un ricevitore posto a una certa distanza le può captare. Tutte le comunicazioni wireless si basano su questo principio.

Nel vuoto tutte le onde elettromagnetiche viaggiano alla stessa velocità, qualunque sia la loro frequenza. Questa velocità, chiamata **velocità della luce** c , è pari a circa 3×10^8 m/sec, ossia circa 30 cm per nanosecondo. Nei cavi in rame e nelle fibre ottiche la velocità di trasmissione scende a 2/3 di questo valore e diventa leggermente dipendente dalla frequenza. La velocità della luce è il limite ultimo della velocità: nessun oggetto o segnale può muoversi più velocemente.

a relazione fondamentale tra f , λ e c (nel vuoto) è

$$\lambda f = c \quad (2.2)$$

oiché c è una costante, se si conosce f è possibile ricavare λ e viceversa.

Come regola generale, quando λ è espressa in metri e f è espressa in MHz $\lambda f \approx 300$. Per esempio, onde di 100 MHz sono lunghe circa 3 metri, onde di 1.000 MHz sono lunghe

circa 0,3 metri e onde di 0,1 metri hanno una frequenza di 3.000 MHz. La Figura 2.11 mostra lo spettro elettromagnetico.

Le porzioni dello spettro indicate come radio, microonde, infrarosso e luce visibile si possono utilizzare per trasmettere informazioni modulando l'ampiezza, la frequenza o la fase delle onde. La luce ultravioletta, i raggi X e i raggi gamma funzionerebbero anche meglio, per le loro elevate frequenze, ma sono difficili da generare e da modulare, non si propagano bene attraverso i muri e sono dannose per gli esseri viventi. Le bande elencate nella Figura 2.11 sono i nomi ITU ufficiali e si distinguono per le lunghezze d'onda, perciò la banda LF va da 1 Km a 10 Km (circa da 30 kHz a 300 kHz). I termini LF, MF e HF indicano rispettivamente le frequenze basse, medie e alte.

Logicamente, quando vennero assegnati i nomi nessuno si aspettava di superare i 10 MHz, perciò le bande più alte sono state chiamate successivamente bande di frequenza Very, Ultra, Super, Extremely e Tremendously High.

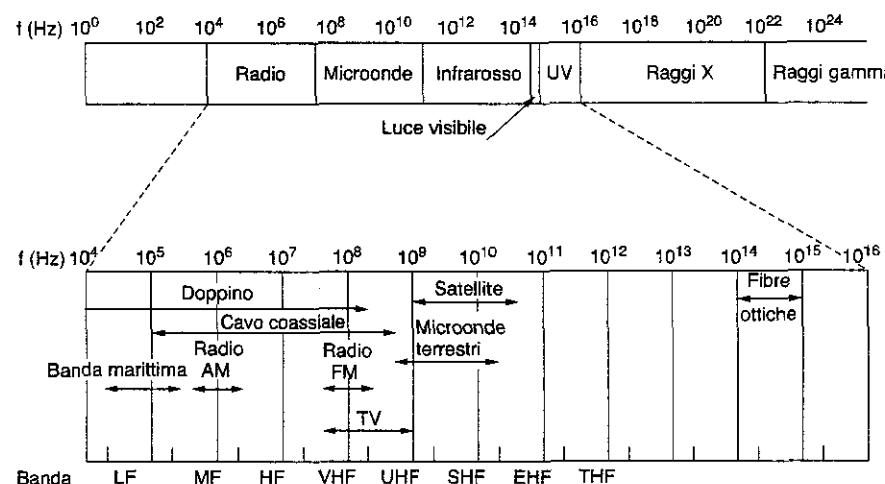


Figura 2.11. Lo spettro elettromagnetico e il suo utilizzo nella comunicazione.

La quantità di informazione che un'onda elettromagnetica può trasportare dipende dalla sua banda. La tecnologia corrente permette di codificare pochi bit per ogni Hertz di bassa frequenza e 8 bit per le alte frequenze, perciò un cavo coassiale con una banda passante di 750 MHz può trasportare diversi Gigabit di dati al secondo. È facile capire, osservando la Figura 2.11, come mai agli utenti di rete piace così tanto la fibra.

Se si risolve l'equazione (2.2) per f e si differenzia rispetto a λ si ottiene

$$\frac{df}{d\lambda} = -\frac{c}{\lambda^2}$$

Se ora si passa alle differenze finite invece dei differenziali e si osservano solo i valori assoluti, si ottiene

$$\Delta f = \frac{c\Delta\lambda}{\lambda^2} \quad (2.3)$$

Perciò, data la larghezza di una banda espressa come intervallo di lunghezze d'onda $\Delta\lambda$, è possibile calcolare la banda di frequenza corrispondente, Δf , e da questa ottenere la massima cadenza dati che la banda può gestire. Maggiore è la banda, più grande è la velocità. Per esempio, si consideri la banda di 1,30 micon mostrata nella Figura 2.6. In questo caso si ha $\lambda = 1,3 \times 10^{-6}$ e $\Delta\lambda = 0,17 \times 10^{-6}$ perciò Δf è circa 30 THz. Supponendo di trasmettere 8 bit/Hz si ottiene una velocità di 240 Tbps.

La maggior parte delle trasmissioni utilizza una banda di frequenze ristretta ($\Delta f / f \ll 1$) per ottenere la migliore ricezione (molti watt/Hz), ma in certi casi si usa una banda larga con due varianti. Nella tecnica a **spettro distribuito a frequenza variabile** (*frequency hopping*), il trasmettitore salta da una frequenza a un'altra centinaia di volte al secondo. Questa variante è utilizzata nelle comunicazioni militari, perché rende difficile rilevare le trasmissioni ed è quasi impossibile da disturbare. Offre anche una buona resistenza al multipath fading, perché il segnale diretto arriva sempre prima al ricevitore. I segnali riflessi seguono un percorso più lungo e arrivano dopo; a quel punto, se ha cambiato frequenza, il ricevitore non accetta più i segnali trasmessi alla frequenza precedente eliminando così l'interferenza tra i segnali diretti e quelli riflessi. Negli ultimi anni questa tecnica ha avuto anche diverse applicazioni commerciali; per esempio, è adottata dallo standard 802.11 e da Bluetooth. Una nota curiosa: questa tecnica è stata inventata da Hedy Lamarr -sex symbol di origine austriaca- la prima donna ad apparire nuda in un film (*Estasi*, film Ceco del 1933); il suo primo marito, un fabbricante di armi, le spiegò quanto fosse facile bloccare i segnali radio utilizzati per controllare i siluri. Quando scoprì che il marito stava vendendo armi a Hitler rimase inorridita; fuggì travestita da cameriera e volò a Hollywood per continuare la sua carriera di attrice. Nel tempo libero inventò il sistema a frequenza variabile con lo scopo di aiutare lo sforzo bellico degli Alleati. Lo schema utilizzava 88 frequenze (che corrisponde al numero di tasti e frequenze del pianoforte).

Per questa invenzione lei e il suo amico, il compositore musicale George Antheil, ricevettero il brevetto U.S. 2.292.387. Purtroppo, i due non riuscirono a convincere la Marina degli Stati Uniti che il sistema poteva avere applicazioni pratiche. La tecnica divenne popolare solo alcuni anni dopo la scadenza della licenza.

L'altra forma di trasmissione, chiamata **spettro distribuito a sequenza diretta** (*direct sequence*), che propaga il segnale in banda larga, sta guadagnando popolarità in ambito commerciale, ed è stato utilizzato per alcuni modelli di telefoni cellulari della seconda generazione. Verrà implementato in maniera consistente sui modelli di terza generazione grazie alle caratteristiche di buona efficienza spettrale, di immunità al rumore e ad altre proprietà. Questa variante è utilizzata anche da alcune LAN wireless.

La trasmissione a spettro distribuito verrà discussa più avanti, per il momento è più semplice supporre che tutte le trasmissioni utilizzino una banda stretta. Il prossimo paragrafo spiega come sono utilizzate le varie parti dello spettro elettromagnetico rappresentato nella Figura 2.11.

2.3.2 Trasmissioni radio

Le onde radio sono semplici da generare, possono viaggiare per lunghe distanze e attraversano facilmente gli edifici; per questi motivi sono largamente utilizzate per la comunicazione, sia all'interno delle costruzioni sia all'esterno. Le onde radio sono omnidirezionali, ossia si espandono dalla sorgente in tutte le direzioni, perciò il trasmettitore e il ricevitore non devono essere fisicamente allineati.

Qualche volta la proprietà omnidirezionale è utile, ma altre invece non lo è. Nel 1970, General Motors decise di dotare tutte le sue nuove Cadillac con un sistema ABS controllato da computer. Quando il guidatore premeva il pedale del freno, il computer modulava l'attivazione dei freni invece di attivarli in un colpo solo. Un bel giorno un poliziotto della stradale dell'Ohio, mentre utilizzava la sua nuova autoradio per chiamare la centrale, si accorse che la Cadillac che viaggiava accanto alla sua auto si comportava come un cavallino imbizzarrito. Quando l'agente fermò l'auto, il conducente affermò di non aver fatto nulla e che la sua auto era improvvisamente impazzita. Alla fine emerse uno strano schema: le Cadillac qualche volta sembravano impazzire, ma solo sulle autostrade dell'Ohio e solo in prossimità delle auto di pattuglia della stradale. Per molto tempo General Motors non riuscì a capire come mai le Cadillac funzionavano bene in tutti gli altri stati e nelle strade secondarie dell'Ohio; solo dopo molte ricerche gli ingegneri capirono che i cavi elettrici della Cadillac si comportavano come ottime antenne per le frequenze del nuovo sistema di comunicazione radio adottato dalla polizia stradale dell'Ohio.

Le proprietà delle onde radio dipendono dalla frequenza. Alle frequenze più basse, le onde radio attraversano bene gli ostacoli ma la potenza diminuisce bruscamente allontanandosi dalla sorgente; nell'aria la diminuzione è pari a circa $1/r^2$. Alle frequenze più alte, le onde radio tendono a viaggiare in linea retta, rimbalzano contro gli ostacoli e sono anche assorbite dalla pioggia.

A tutte le frequenze le onde radio sono soggette a interferenze da parte di motori e altri dispositivi elettrici, e poiché sono in grado di superare lunghe distanze, l'interferenza tra gli utenti è un problema. Per questo motivo tutti i governi limitano le licenze a disposizione dei trasmettitori radio, con un'eccezione descritta più avanti.

Nelle bande VLF, LF e MF, le onde radio seguono il terreno come mostrato nella Figura 2.12(a). Queste onde si possono ricevere per circa 1.000 Km alle frequenze più basse, e a distanze più brevi con frequenze maggiori. Le stazioni radio AM utilizzano la banda MF, ecco perché le stazioni radiofoniche AM di Boston possono essere ascoltate facilmente a New York. Le onde radio in queste bande attraversano facilmente gli edifici, e ciò spiega perché le radio portatili funzionano al chiuso.

Nel caso della comunicazione dati, il problema principale di queste bande è la loro ridotta ampiezza di banda (come si può notare osservando l'equazione 2.3).

Nelle bande HF e VHF le onde terrestri tendono a essere assorbite dal pianeta, ma le onde che raggiungono la ionosfera (uno strato composto da particelle caricate che circonda la Terra a un'altezza compresa tra i 100 e i 500 Km), sono riflesse e tornano verso il pianeta come mostrato nella Figura 2.12(b). In particolari condizioni atmosferiche, i segnali possono rimbalzare diverse volte. I radioamatori utilizzano queste bande per comunicare su lunghe distanze; anche i militari comunicano attraverso le bande HF e VHF.

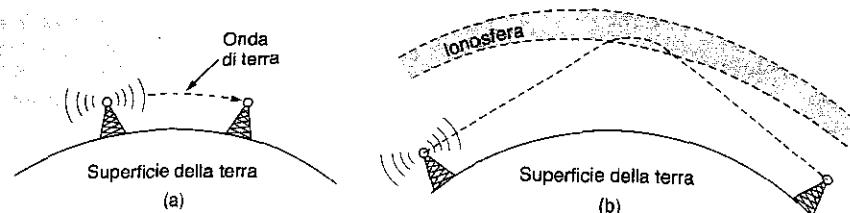


Figura 2.12. (a) Nelle bande VLF, LF ed MF le onde radio seguono la curvatura del pianeta.
(b) Nella banda HF rimbalzano contro la ionosfera.

2.3.3 Trasmissione a microonde

Sopra i 100 MHz le onde viaggiano quasi in linea retta e pertanto si mettono a fuoco facilmente. Concentrando tutta l'energia in un piccolo raggio per mezzo di un'antenna parabolica (come quella delle TV satellitari) si riesce a ottenere un rapporto segnale/rumore molto più alto, ma le antenne trasmittenti e riceventi devono essere accuratamente allineate le une con le altre. Inoltre, questa direzionalità permette a più trasmettitori allineati di comunicare senza interferenze con più ricevitori allineati, purché si osservino alcune regole di distanza minima.

Per decenni, prima dell'avvento delle fibre ottiche, le microonde hanno costituito il cuore dei sistemi di trasmissione telefonica su lunghe distanze. MCI, uno dei primi concorrenti di AT&T, costruì il suo intero sistema con comunicazioni a microonde basate su antenne distanti anche decine di chilometri. Già il nome dell'azienda rifletteva questo sistema di comunicazione: MCI è infatti l'acronimo di Microwave Communications, Inc. MCI è poi passata alla fibra e si è fusa con WorldCom.

Poiché le microonde viaggiano in linea retta, se le antenne sono troppo lontane (si pensi per esempio a un collegamento San Francisco-Amsterdam) entra in gioco la curvatura della Terra; di conseguenza sono necessari dei ripetitori. Più sono alte le antenne, maggiore può essere la loro distanza. La distanza tra ripetitori varia grossolanamente con la radice quadrata dell'altezza dell'antenna. Nel caso di antenne alte 100 metri, i ripetitori possono trovarsi a 80 Km di distanza.

A differenza delle onde radio a bassa frequenza, le microonde non attraversano molto bene gli edifici; inoltre, anche se il raggio può essere messo correttamente a fuoco, c'è ancora un po' di divergenza nello spazio. Alcune onde si possono rifrangere sugli strati più bassi dell'atmosfera e arrivano un po' dopo le onde dirette. Le onde in ritardo possono arrivare fuori fase con le onde dirette e questo può annullare il segnale. L'effetto è chiamato **multipath fading** (dissolvenza multi percorso); dipende dalle condizioni climatiche e dalla frequenza e spesso è un problema serio. Alcuni operatori tengono il 10% dei loro canali inattivi come riserva, da utilizzare quando il multipath fading elimina temporaneamente parte della banda di frequenza.

La domanda di spettro ha portato gli operatori a utilizzare anche le frequenze più alte. Le bande sopra i 10 GHz sono ormai di uso comune, ma a circa 4 GHz è sorto un nuovo problema: l'assorbimento da parte dell'acqua. Queste onde, lunghe solo pochi centimetri, sono assorbite dalla pioggia. L'effetto sarebbe utile se si volesse costruire un enorme forno a microonde esterno per la cottura degli uccelli di passaggio, ma per la comunicazione è un vero problema.

Come nel caso del multipath fading, l'unica soluzione è interrompere i collegamenti sui quali sta piovendo e deviare le comunicazioni.

Riepilogando, la comunicazione a microonde è talmente utilizzata nelle comunicazioni telefoniche su lunghe distanze, nella telefonia cellulare, nella televisione e in altre applicazioni, che si sta manifestando una grave scarsità di spettro. Questo sistema di trasmissione offre diversi vantaggi rispetto alla fibra; il principale è che non richiede alcun diritto di passaggio: acquistando un piccolo appezzamento di terreno ogni 50 Km e costruendo su di esso un'antenna è possibile scavalcare il sistema telefonico e comunicare direttamente. In questo modo MCI è riuscita a diventare in poco tempo un'azienda telefonica per lunghe distanze. Sprint ha seguito una direzione completamente diversa: facendo parte di Southern Pacific Railroad, azienda ferroviaria che già possedeva una gran quantità di diritti di passaggio, ha semplicemente steso le fibre lungo i tracciati esistenti.

Le microonde sono anche relativamente convenienti. L'installazione di due semplici antenne, magari due banali pali controventati, costa meno della posa di 50 Km di fibra in un'area urbana molto congestionata o attraverso una montagna. Magari è anche più economica del noleggio della fibra ottica posseduta da un'azienda telefonica, specialmente se questa azienda non è ancora stata pagata per tutto il rame che ha sradicato durante l'installazione della fibra.

La politica dello spettro elettromagnetico

Per evitare il caos totale sono stati stipulati accordi nazionali e internazionali sui soggetti e le frequenze utilizzabili. Poiché ognuno vuole una maggiore velocità di trasmissione dati, tutti vogliono più spettro. I governi nazionali assegnano lo spettro per le radio AM e FM, le televisioni e i telefoni cellulari, così come per le aziende telefoniche, la polizia, la marina, l'aeronautica, l'esercito, il governo e gli altri utenti in competizione. A livello mondiale, un'agenzia di ITU-T (WARC) tenta di coordinare queste assegnazioni in modo che sia possibile produrre dispositivi in grado di funzionare in paesi diversi. Le nazioni però non sono vincolate alle raccomandazioni di ITU-R, e la FCC (*Federal Communication Commission*), che si occupa dell'assegnazione negli Stati Uniti, ha saltuariamente rifiutato tali indicazioni.

Anche quando una parte dello spettro viene assegnato a un particolare utilizzo, per esempio ai telefoni cellulari, c'è il problema della suddivisione delle frequenze tra i fornitori di servizi. In passato sono stati utilizzati tre algoritmi. Il più vecchio, chiamato anche **concorso di bellezza**, richiedeva che ogni fornitore motivasse il valore della sua proposta per

il pubblico interesse; in seguito gli agenti del governo sceglievano la motivazione che sembrava più interessante. Questo sistema era spesso causa di corruzione e scelte arbitrarie. Inoltre, un agente scrupoloso e onesto che pensava che un'azienda straniera potesse fare un lavoro migliore di quelle nazionali avrebbe dovuto dare un sacco di spiegazioni per la sua scelta.

Queste osservazioni hanno condotto al secondo algoritmo, basato su una lotteria tra le aziende interessate. Il problema di questa idea è che anche aziende non interessate all'utilizzo dello spettro potevano partecipare alla lotteria. Una catena di ristoranti o di negozi di scarpe, dopo aver vinto la gara, poteva rivendere lo spettro a un fornitore di servizi e ottenere un grosso profitto senza alcun rischio. Da queste considerazioni è nato il terzo algoritmo: la vendita all'asta di porzioni dello spettro elettromagnetico. Quando nel 2000 il governo inglese mise all'asta le frequenze necessarie per i sistemi di telefonia mobile di terza generazione, pensava di guadagnare circa 4 miliardi di euro. Al termine dell'asta, il governo era riuscito ad accumulare qualcosa come 40 miliardi di euro, perché le aziende interessate, prese dal timore di perdere il "treno", si fecero travolgere da una frenesia crescente. Questo evento accese molto interesse nei governi vicini e ispirò nuove aste. La cosa ha funzionato, ma ha anche lasciato alcune aziende così piene di debiti da chiudere per bancarotta. Anche nei casi migliori, ci vorranno molti anni per recuperare il costo delle licenze.

Un approccio completamente differente consiste nel non assegnare affatto le frequenze, lasciando che ognuno trasmetta a piacere, ma regolando la potenza utilizzata in modo da limitare la portata delle stazioni per evitare l'interferenza reciproca.

Seguendo questo principio la maggior parte dei governi ha riservato alcune bande di frequenza, chiamate **ISM** (*Industrial, Scientific, Medical*), per l'utilizzo senza licenze. Telecomandi che aprono i garage, telefoni senza fili, giocattoli telecomandati, mouse senza fili e numerosi altri apparecchi domestici utilizzano le bande ISM. Per limitare l'interferenza tra questi dispositivi non coordinati, FCC ordina che tutti gli apparecchi nelle bande ISM utilizzino tecniche di trasmissione a spettro distribuito. Regole simili sono adottate anche da altri paesi.

La posizione delle bande ISM varia leggermente da paese a paese. Negli Stati Uniti, per esempio, gli apparecchi che hanno una potenza minore di 1 Watt possono utilizzare le bande mostrate nella Figura 2.13 senza richiedere la licenza FCC. La banda 900 MHz funziona meglio, ma è piuttosto affollata e non è disponibile a livello mondiale. La banda 2,4 GHz è disponibile nella maggior parte dei paesi, ma subisce interferenze dai forni a microonde e dalle installazioni radar; è usata da Bluetooth e alcune LAN 802.11 wireless. La banda 5,7 GHz è nuova e relativamente libera, perciò gli apparati che la utilizzano sono ancora abbastanza costosi; poiché è la banda usata dallo standard 802.11a, diventerà presto molto popolare.

Lo strato fisico

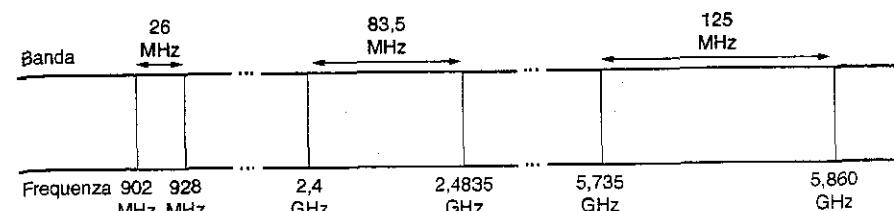


Figura 2.13. Le bande ISM negli Stati Uniti.

2.3.4 Infrarossi e onde millimetriche

Per la comunicazione a corto raggio si usano spesso raggi infrarossi non vincolati e onde millimetriche. I telecomandi dei televisori, dei videoregistratori e degli stereo adottano tutti la comunicazione a infrarossi. Questo sistema è relativamente direzionale, è economico e facile da costruire, ma ha un grande difetto: non riesce ad attraversare ostacoli solidi.

In generale, più ci si allontana dalle onde radio lunghe e ci si avvicina allo spettro della luce visibile, più le onde si comportano come luce e meno come onde radio. D'altra parte, il fatto che le onde infrarosse non superano i muri rappresenta anche un vantaggio; significa che un sistema a infrarossi che si trova in una stanza di un edificio non interferirà con sistemi simili collocati nelle stanze o negli edifici adiacenti: è per questo motivo che non si può controllare con il proprio telecomando il televisore del vicino. Inoltre, i sistemi a infrarossi sono più sicuri di quelli basati sulle onde radio, in quanto sono più difficili da intercettare e non richiedono alcuna licenza governativa (come invece accade per i sistemi a onde radio che non utilizzano le bande ISM). I sistemi a infrarossi sono utilizzati anche per collegare i computer portatili alle stampanti, ma hanno un ruolo secondario nella gara delle telecomunicazioni.

2.3.5 Trasmissione a onde luminose

I segnali ottici non guidati sono utilizzati da secoli. Paul Revere trasmise un segnale ottico binario dalla Old North Church prima di iniziare la sua famosa corsa. In un'applicazione un po' più moderna, laser montati sui tetti permettono di realizzare una connessione LAN tra due edifici. La segnalazione ottica coerente basata sui laser è intrinsecamente unidirezionale, perciò ogni edificio ha bisogno di un laser e di un rilevatore fotoelettrico. Questo schema offre un'ampiezza di banda elevata a un costo ridotto; inoltre è relativamente facile da installare e non richiede licenze FCC, a differenza delle microonde. La potenza del laser, un raggio molto sottile, rappresenta anche la sua debolezza: puntare un raggio laser di 1 mm su un bersaglio della dimensione di una capocchia di spillo distante 500 metri richiede un'abilità incredibile. Di solito nel sistema vengono introdotte lenti per rendere il raggio meno focalizzato.

Uno svantaggio è che i raggi laser non possono attraversare la pioggia e la nebbia fitta, e funzionano bene solo nelle giornate serene, ma i problemi possono presentarsi anche nelle giornate più belle. Per esempio, all'autore una volta è capitato di partecipare a una conferenza tenuta in un grande e moderno hotel in Europa: gli organizzatori avevano preparato una stanza piena di terminali per consentire ai partecipanti di consultare la posta elettronica durante le presentazioni più noiose; poiché la conferenza durava solo 3 giorni, il PTT locale non aveva voluto aggiungere nuove linee telefoniche, perciò gli organizzatori installarono sul tetto dell'edificio un laser puntato sull'edificio del dipartimento di informatica, distante solo qualche chilometro. La connessione, collaudata la sera prima della conferenza, funzionò perfettamente. Alle 9 del mattino successivo, in una giornata piena di sole, il collegamento s'interruppe completamente e così rimase per tutto il giorno. Quella sera gli organizzatori verificarono di nuovo il sistema e, ancora una volta, tutto funzionò perfettamente. Nei due giorni successivi, accadde la stessa cosa.

Solo al termine della conferenza gli organizzatori capirono l'origine del problema: il calore del sole durante il giorno faceva sollevare dal tetto correnti di convezione (Figura 2.14). La turbolenza deviava il raggio facendolo ballare intorno al rilevatore. Effetti di questo tipo rendono le stelle scintillanti (ecco perché gli astronomi collocano i loro telescopi sulle cime delle montagne) e fanno anche luccicare le strade nei giorni più caldi.

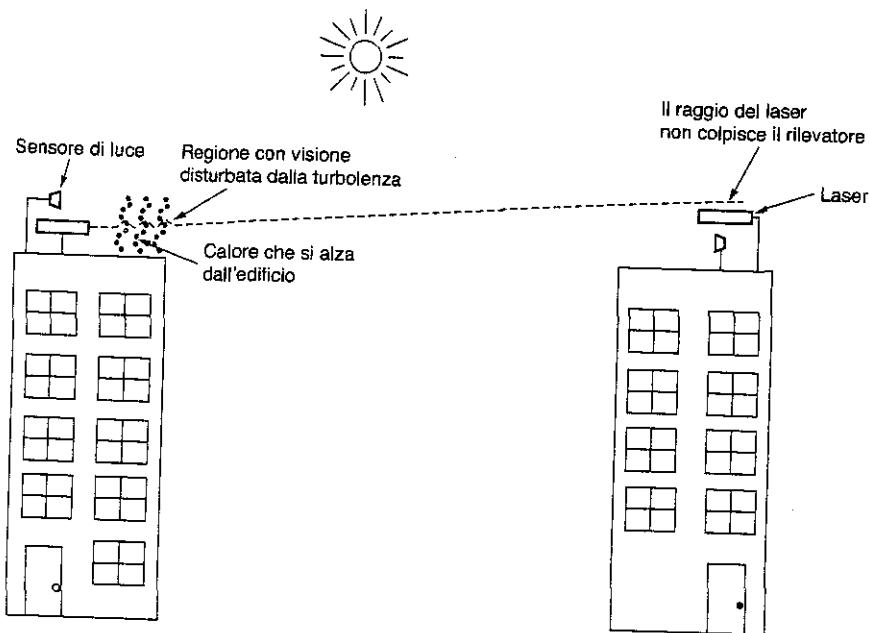


Figura 2.14. Correnti di convezione possono interferire con i sistemi di comunicazione laser. L'immagine rappresenta un sistema bidirezionale con due laser.

2.4 Comunicazioni satellitari

Alla fine degli anni '50 e all'inizio dei '60 gli scienziati tentarono di creare sistemi di comunicazione cercando di far rimbalzare segnali radio su palloni aerostatici meteorologici metallizzati. Sfortunatamente, essendo troppo deboli, i segnali ricevuti non erano di alcune utilità pratica. Successivamente la Marina degli Stati Uniti constatò che nel cielo esisteva una specie di pallone aerostatico permanente, ossia la luna, e costruì un sistema funzionante che permetteva alle navi di comunicare facendo rimbalzare i segnali su di essa.

I progressi successivi nel settore delle comunicazioni astronomiche hanno dovuto attendere il lancio del primo satellite di comunicazioni. La differenza principale tra un satellite artificiale e un satellite reale è che il primo è in grado di amplificare i segnali prima di ritrasmetterli, caratteristica che trasforma una strana curiosità in un potente sistema di comunicazione. I satelliti di comunicazione hanno alcune interessanti proprietà che li rendono allettanti per molte applicazioni. Nella sua forma più semplice, un satellite di comunicazioni può essere immaginato come un grande ripetitore di microonde collocato nel cielo. Questo dispositivo contiene diversi **transponder**, ossia ricetrasmettitori satellitari. Ognuno ascolta una parte dello spettro, amplifica il segnale in ingresso e lo ritrasmette su un'altra frequenza per evitare interferenze con il segnale in arrivo. I raggi puntati verso il basso possono essere larghi e coprire una considerevole frazione della superficie terrestre, oppure stretti e coprire un'area dal diametro di poche centinaia di chilometri. Questa modalità operativa è chiamata **bent pipe**.

In accordo con la legge di Keplero, il periodo orbitale di un satellite varia in base al raggio della sua orbita elevato alla potenza 3/2. Più è alto il satellite, più è lungo il periodo. In prossimità della superficie terrestre il periodo è di circa 90 minuti. Di conseguenza, i satelliti su orbite basse scompaiono dalla vista rapidamente e ne servono molti per fornire una copertura continua. A un'altitudine di circa 35.800 Km il periodo è di 24 ore; a 384.000 Km è di circa un mese, e basta osservare regolarmente la Luna per verificarlo.

Il periodo di un satellite è importante, ma non è l'unica caratteristica che ne determina la collocazione. Un altro problema è rappresentato dalla presenza delle fasce di Van Allen, strati di particelle molto cariche intrappolate dal campo magnetico terrestre; un satellite che le attraversasse verrebbe velocemente distrutto. Questi fattori conducono alle tre zone in cui i satelliti possono essere collocati senza pericolo, che sono illustrate nella Figura 2.15 assieme ad alcune delle loro proprietà. I prossimi paragrafi descrivono brevemente i satelliti che popolano ciascuna zona.

2.4.1 Satelliti geostazionari

Nel 1945, lo scrittore di fantascienza Arthur C. Clarke calcolò che un satellite posto a un'altitudine di 35.800 Km in un'orbita equatoriale circolare sarebbe apparso fermo nel cielo agli osservatori terrestri, perciò non avrebbe avuto bisogno di essere inseguito (Clarke, 1945). L'autore continuò descrivendo un sistema di comunicazione completo basato su questi **satelliti geostazionari**, specificando orbite, pannelli solari, frequenze radio e procedure di lancio.

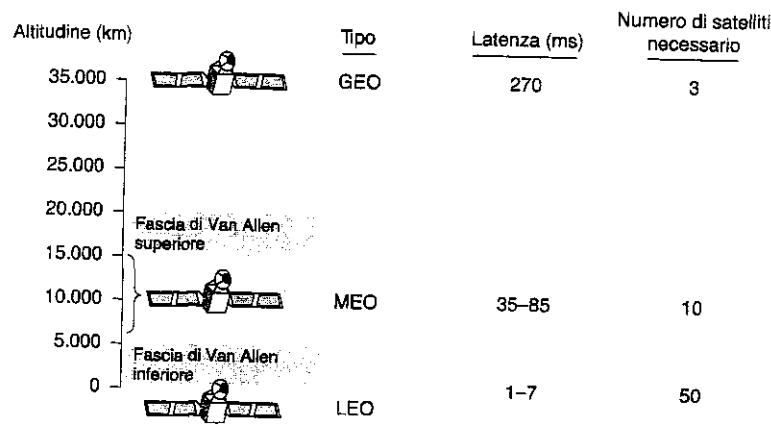


Figura 2.15. Satelliti di comunicazione e alcune delle loro proprietà, incluse le altitudini, il ritardo di andata e ritorno del segnale e il numero di satelliti necessari per fornire una copertura globale.

Purtroppo, concluse che i satelliti non si sarebbero potuti realizzare a causa dell'impossibilità di collocare in orbita amplificatori dall'alto consumo energetico basati su fragili valvole a vuoto, perciò non portò avanti la sua idea, anche se scrisse diverse storie di fantascienza dedicate all'argomento.

L'invenzione del transistor cambiò tutto quanto, e nel luglio 1962 permise di lanciare il primo satellite artificiale di comunicazione, chiamato Telstar. Da quel momento, i satelliti di comunicazione sono diventati affari da molti miliardi di dollari (è l'unico settore dell'industria spaziale che ha prodotto alti profitti). Questi satelliti collocati su orbite alte sono anche chiamati satelliti **GEO** (*Geostationary Earth Orbit*, orbita terrestre geostazionaria).

Con la tecnologia corrente, per evitare interferenze, è meglio non disporre i satelliti geostazionari a intervalli minori di 2 gradi nel piano equatoriale di 360 gradi. Con un intervallo di 2 gradi ci possono essere solo $360/2=180$ satelliti di questo tipo nel cielo, ma ogni transponder può utilizzare frequenze e polarizzazioni multiple per aumentare l'ampiezza di banda disponibile.

Per evitare il caos totale nel cielo, l'allocazione degli slot orbitali è gestita dall'ITU. Questo processo è molto politico e ci sono paesi, appena usciti dall'età della pietra, che chiedono di avere i propri slot orbitali per poterli poi affittare al miglior offerente. Altri paesi sostengono che i diritti di proprietà nazionale non si estendono alla luna, e che nessun paese ha un diritto legale sugli slot orbitali collocati sul suo territorio. Ad alimentare la discordia, le telecomunicazioni commerciali non rappresentano l'unica applicazione pratica; anche le stazioni televisive, i governi e i militari pretendono di aver voce in capitolo.

I moderni satelliti possono raggiungere dimensioni enormi: alcuni pesano 4.000 Kg e consumano diversi kilowatt di energia elettrica prodotta da pannelli solari. Gravità solare, lunare e planetaria tendono ad allontanarli dagli slot e dagli orientamenti assegnati, ma

l'effetto è contrastato dai motori a razzo installati a bordo. La continua attività di messa a punto è chiamata **station keeping** e quando dopo una decina di anni circa il propellente dei motori si esaurisce, il satellite va alla deriva e precipita senza che si possa far nulla, quindi deve essere disattivato.

Alla fine, l'orbita decade e il satellite rientra nell'atmosfera e brucia, o qualche volta precipita al suolo. Gli slot orbitali non sono l'unico pompa della discordia; anche le frequenze creano problemi, poiché le trasmissioni dirette verso il basso interferiscono con i sistemi a microonde esistenti. Di conseguenza, l'ITU ha assegnato alle applicazioni satellitari alcune bande di frequenza, e la Figura 2.16 elenca le principali. La banda C è stata la prima assegnata al traffico commerciale e comprende due intervalli di frequenze assegnate: il più basso per il traffico diretto verso la terra, e il più alto per quello diretto verso il satellite. Per consentire al traffico di viaggiare contemporaneamente in entrambe le direzioni servono due canali, uno per ciascuna direzione. Queste bande sono sovrapposte, perché utilizzate anche dalle aziende telefoniche per i collegamenti a microonde terrestri.

Le bande L ed S sono state aggiunte nel 2000 in base ad accordi internazionali, ma sono strette e affollate.

Banda	Downlink	Uplink	Larghezza della banda	Problemi
L	1,5 GHz	1,6 GHz	15 MHz	Banda stretta; affollamento
S	1,9 GHz	2,2 GHz	70 MHz	Banda stretta; affollamento
C	4,0 GHz	6,0 GHz	500 MHz	Interferenza terrestre
Ku	11 GHz	14 GHz	500 MHz	Pioggia
Ka	20 GHz	30 GHz	3.500 MHz	Pioggia; costo degli apparati

Figura 2.16. Le principali bande usate dai satelliti.

La prossima banda a disposizione delle società di telecomunicazioni commerciali è la banda Ku (*K under*). Questa banda non è ancora congestionata e a queste frequenze i satelliti possono essere collocati a solo un grado angolare di distanza l'uno dall'altro; purtroppo esiste un altro problema: la pioggia. L'acqua assorbe in modo eccellente queste microonde corte. Per fortuna, le tempeste più pesanti di solito sono localizzate, perciò è possibile eludere il problema utilizzando diverse stazioni terrestri molto distanti tra loro invece di usarne solo una, ma al prezzo di antenne aggiuntive, cavi extra e dispositivi elettronici in grado di attivare rapidamente un cambio di stazione.

Le ampiezze di banda sono state assegnate al traffico commerciale anche nella banda Ka (*K above*), ma l'attrezzatura necessaria è ancora costosa. Oltre a queste bande commerciali, esistono molte bande governative e militari. Un satellite moderno ha circa 40 transponder, ognuno con un'ampiezza di banda di 80 MHz. Di solito ogni transponder opera come un bent pipe, ma satelliti recenti hanno capacità di elaborazione interna che permette operazioni più sofisticate. Nei primi satelliti la divisione dei transponder in canali era statica: l'ampiezza di banda era semplicemente divisa in bande di frequenza prefissate.

Oggi ogni raggio di transponder è diviso in slot temporali, passati a diversi utenti. Entrambe le tecniche (multiplexing a divisione di frequenza e multiplexing a divisione di tempo) saranno esaminate meglio più avanti, in questo capitolo.

I primi satelliti geostazionari avevano una singola emissione che illuminava circa 1/3 della superficie terrestre, chiamata **impronta**. L'enorme diminuzione dei costi, delle dimensioni e dei requisiti di alimentazione della microelettronica ha reso possibile una strategia di trasmissione molto più sofisticata. Ogni satellite è dotato di più antenne e transponder; ogni raggio diretto verso il basso può essere concentrato su una piccola area geografica, perciò possono avvenire contemporaneamente più trasmissioni nei due sensi. In genere questi raggi puntiformi ellittici (**spot**) possono avere un diametro di poche centinaia di chilometri.

Un satellite per telecomunicazioni statunitense di solito prevede un'emissione ampia per i 48 stati contigui, più alcune emissioni puntiformi per l'Alaska e le Hawaii.

Un nuovo sviluppo nel settore delle comunicazione satellitari è rappresentato dalle microstazioni a basso costo chiamate anche **VSAT** (*Very Small Aperture Terminal*) (Abramson, 2000). Questi terminali leggeri hanno antenne grandi un metro o anche meno (le antenne GEO sono grandi dieci metri) e possono generare circa 1 watt di potenza. La trasmissione uplink (verso il satellite) di solito arriva a 19,2 kbps, ma la velocità in downlink (verso terra) può anche superare i 512 kbps. Le televisioni satellitari utilizzano questa tecnologia per le loro trasmissioni unidirezionali.

In molti sistemi VSAT le microstazioni non hanno abbastanza energia per comunicare direttamente le une con le altre (via satellite, logicamente), per questo motivo è necessario installare speciali stazioni terrestri, chiamate **hub**, dotate di grosse antenne ad alto guadagno che trasmettono il traffico attraverso le stazioni VSAT (Figura 2.17).

In questa modalità operativa, la stazione trasmittente oppure quella ricevente fa uso di una grande antenna e un potente amplificatore. Il compromesso per avere stazioni terminali utente più economiche è un maggior ritardo di propagazione. VSAT ha un grande potenziale nelle aree rurali; non è apprezzato da tutti ma più della metà della popolazione mondiale vive a un'ora di distanza dal telefono più vicino.

Stendere cavi telefonici per raggiungere migliaia di piccoli villaggi non rientra nel bilancio della maggior parte dei governi del Terzo Mondo, ma installare parabole VSAT di un metro di diametro alimentate da celle a energia solare è spesso fattibile. VSAT fornisce la tecnologia che permetterà di cablare il mondo.

I satelliti di comunicazione hanno molte proprietà radicalmente diverse da quelle dei collegamenti terrestri punto-punto. Prima di tutto, anche se i segnali trasmessi e ricevuti da un satellite viaggiano a velocità della luce (circa 300.000 Km/sec), il lungo viaggio di andata e ritorno introduce un sostanziale ritardo nei satelliti GEO. A seconda della distanza tra l'utente e la stazione terrestre e dell'elevazione del satellite sopra l'orizzonte, il tempo di trasmissione è tra i 250 e i 300 msec; di solito si aggira intorno ai 270 msec (540 msec per un sistema VSAT con hub).

Per confronto, i collegamenti terrestri a microonde hanno un ritardo di propagazione pari a circa 3 μ sec/km, mentre quelli basati su cavi coassiali e fibre ottiche hanno un ritardo

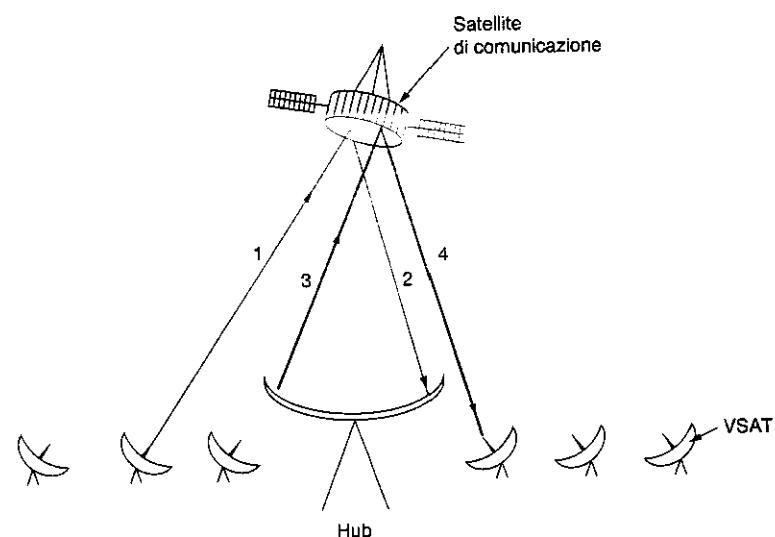


Figura 2.17. VSAT che usano un hub.

che si aggira intorno ai 5 μ sec/km. I cavi sono più lenti poiché i segnali elettromagnetici viaggiano più velocemente in aria che attraverso materiali solidi.

Un'altra proprietà importante dei satelliti è che sono mezzi di trasmissione broadcast per natura: inviare un messaggio a migliaia di stazioni poste nella stessa area del transponder costa quanto inviare un messaggio a una singola stazione. In alcune applicazioni questa proprietà è molto utile. Per esempio, si potrebbe immaginare un satellite che trasmette le pagine Web più popolari alle cache di numerosi computer sparsi in una vasta area. Anche quando la diffusione broadcast può essere simulata con linee punto-punto, la trasmissione satellitare può essere molto più economica. D'altro canto, dal punto di vista della sicurezza e della privacy, i satelliti sono un vero disastro: tutto può essere ascoltato da tutti. Per proteggere le comunicazioni è necessario adottare sistemi di crittografia.

I satelliti hanno anche la proprietà che il costo della trasmissione di un messaggio non dipende dalla distanza traversata. Una chiamata oltre oceano costa quanto una chiamata urbana. I satelliti hanno anche un eccellente tasso di errore e possono essere messi in opera quasi istantaneamente, caratteristica importantissima per le comunicazioni militari.

2.4.2 Satelliti su orbite medie

Ad altitudini molto più basse, comprese tra le due fasce di Van Allen, si trovano i satelliti **MEO** (*Medium Earth Orbit*). Visti dalla Terra, questi satelliti si spostano lentamente lungo la longitudine, impiegando circa 6 ore per compiere un giro intorno al pianeta; di conseguenza devono essere rintracciati mentre si spostano nel cielo. Poiché sono più

bassi dei satelliti GEO, coprono un'area più piccola e possono essere raggiunti usando trasmettitori meno potenti. Al momento non sono utilizzati per le telecomunicazioni. I 24 satelliti **GPS** (*Global Positioning System*) che orbitano a circa 18.000 Km di altezza sono di tipo MEO.

2.4.3 Satelliti su orbite basse

Scendendo di altezza si incontrano i satelliti **LEO** (*Low Earth Orbit*). Poiché si spostano rapidamente, la realizzazione di un sistema completo richiede l'utilizzo di numerosi satelliti di questo tipo. D'altra parte, poiché i satelliti sono molto vicini alla superficie del pianeta, le stazioni terrestri non hanno bisogno di molta energia e il ritardo nelle comunicazioni è di pochi millisecondi. Questo paragrafo esamina tre esempi, due dedicati alla trasmissione della voce e uno dedicato al servizio Internet.

Iridium

In precedenza è stato spiegato che nei primi 30 anni dell'era satellitare i LEO sono stati utilizzati raramente, a causa della loro eccessiva velocità. Nel 1990 Motorola aprì un nuovo orizzonte presentando una domanda all'FCC che chiedeva il permesso di lanciare su orbite basse i 77 satelliti del progetto Iridium (l'elemento numero 77 nella tavola degli elementi è appunto l'iridio). Il piano venne successivamente corretto, in modo da utilizzare solo 66 satelliti.

L'idea era che non appena un satellite spariva dalla vista, un altro sarebbe apparso all'orizzonte. Questa proposta fece esplodere una frenesia crescente tra le altre società di telecomunicazioni; in men che non si dica tutti volevano lanciare la loro catena di satelliti su orbite basse.

Dopo sette anni trascorsi alla ricerca di partner e finanziatori, i satelliti Iridium vennero lanciati nel 1997, e il servizio di comunicazione iniziò nel novembre del 1998. Sfortunatamente la richiesta commerciale dei pesanti e scomodi telefoni satellitari era insistente, perché la rete di telefonia mobile era cresciuta in modo spettacolare dal 1990. Di conseguenza, non riuscendo a essere redditizio, Iridium fu costretto alla bancarotta (agosto 1999) in uno dei più spettacolari fiaschi aziendali della storia. I satelliti e altri beni (per un valore di 5 miliardi di euro) vennero in seguito acquistati da un investitore per 25 milioni di euro in una sorta di liquidazione extraterrestre. Il servizio Iridium venne riavviato nel Marzo del 2001.

Il progetto Iridium fornisce un servizio di telecomunicazione a livello mondiale basato su dispositivi tascabili in grado di comunicare direttamente con i satelliti Iridium. Questo servizio permette di trasmettere voce, dati, fax e informazioni di navigazione ovunque: sulla superficie del pianeta, in mare e in aria. Alcuni clienti sono industrie marittime, aeronautiche e di esplorazione petrolifera, oltre che persone che viaggiano in aree del mondo con infrastrutture di telecomunicazioni carenti (deserti, montagne, giungle e paesi del Terzo Mondo). I satelliti Iridium si trovano a un'altitudine di 750 Km, in orbite polari circolari;

Lo strato fisico

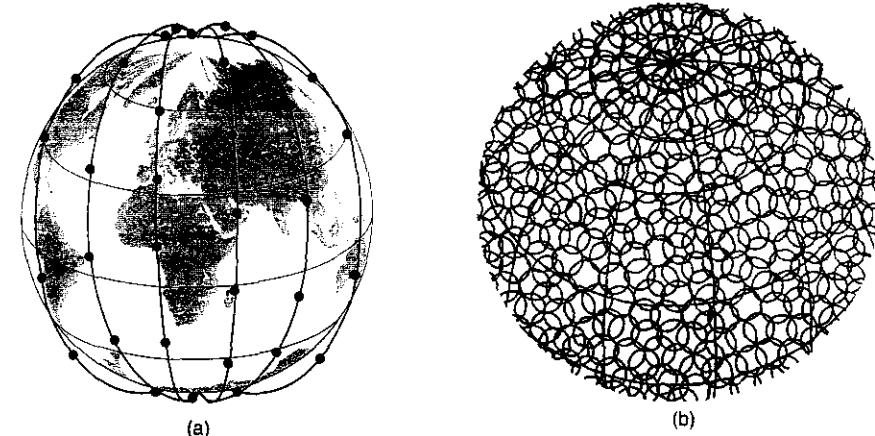


Figura 2.18. (a) I satelliti Iridium formano sei catene intorno alla Terra.
(b) 1.628 celle in movimento coprono il pianeta.

sono sistemati in catene che vanno da nord a sud, con un satellite ogni 32 gradi di latitudine. Come mostra la Figura 2.18(a), con sei catene di satelliti è possibile coprire l'intero pianeta. Questa disposizione ricorda un po' quella di un atomo di disprosio: la Terra funge da nucleo e i satelliti le orbitano intorno come gli elettroni.

Ogni satellite ha un massimo di 48 celle (spot puntiformi), per un totale di 1.628 celle che coprono l'intera superficie terrestre (Figura 2.18(b)). Ogni satellite ha una capacità di 3.840 canali, in totale 253.440. Alcuni di questi canali sono utilizzati per servizi di ricerca e navigazione, altri sono usati per trasmettere dati e voce.

Una proprietà interessante di Iridium è che la comunicazione tra clienti distanti avviene nello spazio, dove ogni satellite trasmette i dati al successivo come mostrato nella Figura 2.19(a). Nel disegno, la persona che chiama si trova al Polo Nord e contatta un satellite posto proprio sopra la sua testa; la chiamata è ritrasmessa attraverso gli altri satelliti fino al destinatario che si trova al Polo Sud.

Globalstar

Globalstar, un progetto alternativo a Iridium, si basa su 48 satelliti LEO e utilizza un diverso schema di commutazione. Iridium ritrasmette le chiamate da satellite a satellite, una tecnica che richiede la presenza di sofisticate attrezzi di commutazione in ogni satellite; Globalstar invece utilizza un modello tradizionale bent pipe. La chiamata che ha origine al Polo Nord nella Figura 2.19(b), ritrasmessa sulla terra, è raccolta da una grande stazione terrestre e dirottata attraverso una rete terrestre alla stazione più vicina al destinatario. Questo schema ha il grande vantaggio di tenere la complessità sulla terra, dove è più facile da gestire.

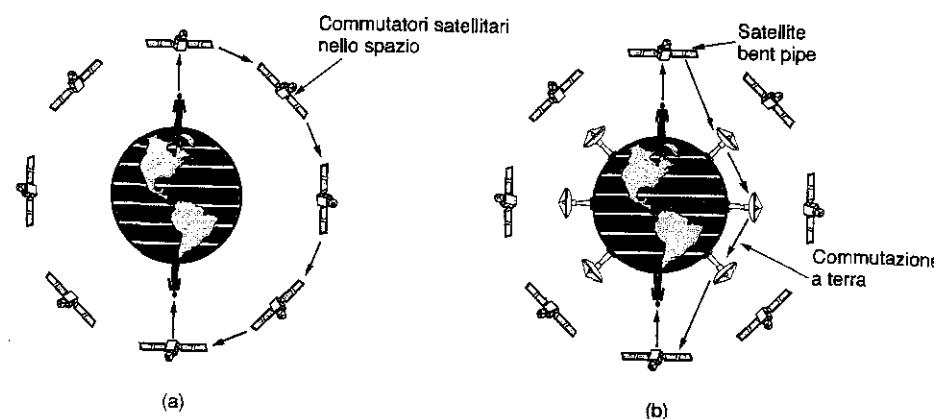


Figura 2.19. (a) Transito della comunicazione nello spazio. (b) Transito della comunicazione a terra.

Inoltre le grandi antenne delle basi terrestri, capaci di trasmettere segnali potenti e di ricevere segnali deboli, consentono di adoperare telefoni di bassa potenza. Dopo tutto, il telefono trasmette solo pochi milliwatt di energia, perciò il segnale che arriva alla stazione terrestre è abbastanza debole anche se è stato amplificato dal satellite.

Teledesic

Iridium è stato progettato per gli utenti telefonici situati in luoghi remoti. L'esempio successivo, ossia **Teledesic**, è stato progettato per gli utenti di Internet affamati di ampiezza di banda sparsi in tutto il mondo. Teledesic è stato concepito nel 1990 da Craig McCaw, pioniere della comunicazione cellulare, e da Bill Gates, fondatore di Microsoft, insoddisfatto della scarsa ampiezza di banda che le compagnie telefoniche offrivano agli utenti di computer. Obiettivo del sistema Teledesic è fornire contemporaneamente a milioni di utenti Internet collegamenti a 100 Mbps in trasmissione e a 720 Mbps in ricezione basati su una piccola antenna fissa di tipo VSAT, completamente indipendente dal sistema telefonico.

Il progetto originale si basava su un sistema composto da 288 satelliti con piccola impronta organizzati in 12 piani posti appena sotto la fascia più bassa di Van Allen, a un'altitudine di 1.350 Km. Successivamente venne trasformato in un sistema composto da 30 satelliti con impronta più grande. La trasmissione avviene nella banda Ka a elevata ampiezza di banda, ancora poco affollata. Il sistema è a commutazione di pacchetto nello spazio: ogni satellite è in grado di instradare i pacchetti attraverso i satelliti più vicini. Quando un utente ha bisogno di una maggior ampiezza di banda per trasmettere i suoi pacchetti, la richiesta viene inviata e gestita in modo dinamico in circa 50 msec. Il sistema dovrebbe entrare in funzione nel 2005, se tutto procede secondo i piani.

2.4.4 Satelliti o fibra?

Il confronto tra la comunicazione satellitare e quella terrestre è istruttivo. Solo 20 anni fa si pensava che il futuro della comunicazione sarebbe stato dominato dai satelliti. Dopo tutto, il sistema telefonico era cambiato pochissimo nei precedenti 100 anni e non sembrava voler cambiare nei successivi 100. Questo movimento glaciale era causato in buona parte dall'insieme di norme che imponeva alle aziende telefoniche di fornire un buon servizio di comunicazione vocale a un prezzo ragionevole, garantendo agli investitori un profitto sicuro. Le persone che avevano dati da trasmettere avevano a disposizione modem a 1.200 bps; questo era più o meno quello che c'era.

L'introduzione della concorrenza, iniziata negli Stati Uniti e in alcuni paesi dell'Europa nel 1984, cambiò tutto radicalmente. Le aziende telefoniche iniziarono a sostituire le loro reti a lunga tratta con fibre ottiche, introdussero servizi a banda larga come ADSL (*Asymmetric Digital Subscriber Line*) e smisero di far pagare prezzi artificiosamente alti per le chiamate a lunga distanza per sovvenzionare il servizio locale.

All'improvviso le connessioni basate su fibre terrestri avevano assunto l'aspetto del vincitore a lungo termine. Ciò nonostante, i satelliti di comunicazione potevano contare su alcune grandi nicchie di mercato che la fibra non era in grado di raggiungere. Per esempio, mentre una singola fibra ha (in linea di principio) un'ampiezza di banda potenziale maggiore di quella di tutti i satelliti lanciati fino a oggi, questa ampiezza di banda non è a disposizione della maggior parte degli utenti. Le fibre installate fino a questo momento sono utilizzate dal sistema telefonico per gestire più chiamate interurbane simultaneamente, non per fornire una grande ampiezza di banda ai singoli utenti. Con i satelliti è realistico che un utente innalzi un'antenna sul tetto di casa e scavalchi completamente il sistema telefonico per ottenere una maggiore ampiezza di banda; Teledesic si basa proprio su questa idea.

Un altro settore che la fibra non può raggiungere è quello della comunicazione mobile. Molte persone oggi vogliono comunicare mentre corrono, guidano, navigano e volano. I collegamenti terrestri basati sulla fibra ottica non sono di alcuna utilità per questi utenti, invece i collegamenti satellitari potenzialmente lo sono. È possibile, comunque, che la combinazione di comunicazione cellulare e fibra svolga un lavoro adeguato per la maggior parte degli utenti.

La terza nicchia è quella delle situazioni in cui la comunicazione broadcast è fondamentale: un messaggio inviato dal satellite può essere ricevuto contemporaneamente da migliaia di stazioni terrestri. Per esempio, il sistema satellitare può essere più economico del sistema broadcast simulato sulla terra, per una società che deve trasmettere un flusso di prezzi relativi a titoli o prodotti a migliaia di operatori.

Il quarto settore è quello della comunicazione in luoghi con terreni inospitali o scarsamente dotati di infrastrutture terrestri. L'Indonesia, per esempio, ha il suo satellite per il traffico telefonico interno: lanciare un satellite è costato meno che stendere migliaia di cavi sottomarini tra le 13.677 isole dell'arcipelago. I satelliti sono una scelta ottima anche per le aree dove è difficile o molto costoso ottenere il diritto di passaggio per stendere la fibra.

Infine, quando è di importanza critica una rapida installazione, come nel caso dei sistemi di comunicazione militari in tempo di guerra, i satelliti vincono facilmente.

In breve, sembra che il sistema di comunicazione principale del futuro sarà quello terrestre basato su fibre ottiche combinato con la rete radio cellulare, ma per alcune applicazioni specializzate i satelliti sono migliori. A tutto ciò va contrapposto l'aspetto economico: sebbene le fibre offrano una maggiore ampiezza di banda, è certamente possibile che la comunicazione terrestre e quella satellitare si facciano concorrenza in modo aggressivo anche sui prezzi. Se gli sviluppi tecnologici ridurranno drasticamente il costo dello spiegamento dei satelliti (per esempio con space shuttle in grado di portare in orbita dozzine di satelliti alla volta) oppure se i satelliti su orbite basse avranno successo la fibra potrebbe perdere il dominio dei mercati.

2.5 La rete telefonica pubblica commutata

Quando devono comunicare tra loro due computer appartenenti alla stessa azienda e situati in prossimità, spesso la cosa più facile da fare è stendere un cavo dall'uno all'altro; le LAN funzionano proprio in questo modo. Se invece le distanze sono grandi, i computer sono molti oppure i cavi devono attraversare strade e altri luoghi pubblici, stendere cavi privati di solito ha un costo proibitivo. Per di più, in quasi ogni nazione del mondo, stendere linee di trasmissione private attraverso (o sotto) proprietà pubbliche è illegale. Di conseguenza i progettisti di reti devono affidarsi ai sistemi di telecomunicazioni esistenti. Questi servizi, specialmente **PSTN** (*Public Switched Telephone Network*, rete telefonica pubblica commutata), sono stati progettati molti anni fa, avendo in mente un obiettivo completamente diverso: trasmettere la voce umana in una forma più o meno comprensibile. L'adattabilità di questi mezzi alla comunicazione tra computer è spesso marginale, comunque la situazione sta rapidamente cambiando con l'introduzione delle fibre ottiche e della tecnologia digitale. In ogni caso il sistema telefonico è talmente intrecciato alle reti geografiche di computer che conviene dedicare un po' di tempo al suo studio.

Per comprendere l'ordine di grandezza del problema è utile confrontare, in modo un po' rozzo ma chiarificatore, le proprietà di una connessione tra computer e computer realizzata attraverso un cavo di rete locale con quelle di un collegamento remoto basato sulla linea telefonica tradizionale.

Un cavo steso tra due computer può trasferire dati a 10^9 bps o più, mentre una linea remota ha una velocità massima di 56 kbps; la differenza è un fattore di quasi 20.000, la stessa che separa un papero che cammina tranquillamente nell'erba da un razzo sparato verso la luna. Se la linea remota venisse sostituita con una connessione ADSL ci sarebbe ancora un fattore di differenza 1.000-2.000.

Il guaio, naturalmente, è che i progettisti di sistemi informatici sono abituati a lavorare con sistemi di computer, perciò quando improvvisamente si confrontano con un altro sistema

le cui prestazioni (secondo il loro punto di vista) sono peggiori di 3 o 4 ordini di grandezza, naturalmente sprecano molto tempo e molte energie tentando di immaginare come utilizzare al meglio il sistema disponibile. I prossimi paragrafi descrivono il sistema telefonico e spiegano come funziona.

2.5.1 Struttura del sistema telefonico

Non appena Alexander Graham Bell brevettò il telefono nel 1876 (solo poche ore prima del suo rivale Elisha Gray), la nuova invenzione ebbe un'enorme richiesta.

Il mercato iniziale fu quello della vendita dei telefoni; gli apparecchi erano venduti a coppie e spettava al cliente tirare un cavo tra i due telefoni. Se voleva comunicare con altri n proprietari di telefoni, un utente doveva tirare cavi separati tra la sua e le altre n abitazioni. In un anno le città furono coperte da cavi che passavano alla rinfusa sopra le abitazioni e gli alberi. Divenne immediatamente chiaro che il modello usato per collegare ogni telefono a ogni altro telefono, rappresentato nella Figura 2.20(a), non poteva funzionare.

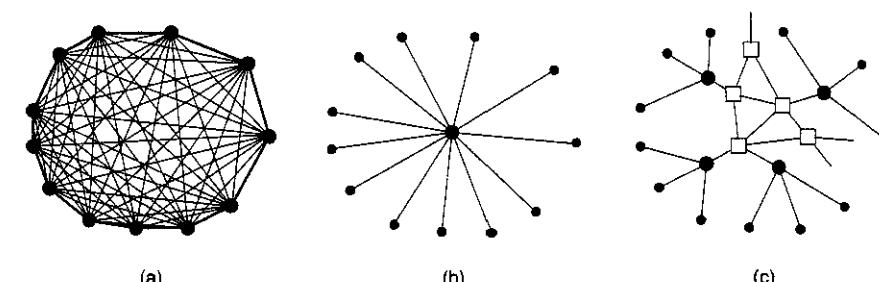


Figura 2.20. (a) Una rete completamente interconnessa. (b) Comutatore centralizzato. (c) Gerarchia a due livelli.

Bell se ne accorse, e costituì la Bell Telephone Company, azienda che aprì il suo primo ufficio di commutazione (in New Haven, nel Connecticut) nel 1878. La società stese cavi fino alle case e agli uffici dei clienti. Per fare una chiamata, un cliente doveva girare la manovella del telefono in modo da far suonare un campanello nell'ufficio dell'azienda telefonica, dove attirava l'attenzione di un operatore; questi manualmente collegava il chiamante al chiamato mediante un cavo a ponte. La Figura 2.20(b) mostra un modello di ufficio a commutazione singola.

Ben presto gli uffici di commutazione di Bell System apparvero un po' ovunque e le persone cominciarono a richiedere telefonate interurbane; allora la società iniziò a collegare tra loro i suoi uffici di commutazione. A questo punto si ripresentò il problema originale: poiché era inimmaginabile collegare ogni ufficio di commutazione a ogni altra centrale

attraverso un cavo, venne inventata una centrale di commutazione di secondo livello. In breve tempo furono necessari diverse centrali di secondo livello, come mostrato nella Figura 2.20(c). Alla fine la gerarchia crebbe fino a raggiungere i cinque livelli. Nel 1890 le tre parti principali del sistema telefonico erano state preparate: gli uffici di commutazione, i cavi tra clienti e centrale (doppini bilanciati e isolati, non più cavi aperti con un ritorno a terra) e le connessioni su lunghe distanze tra le centrali di commutazione. Da allora sono stati apportati miglioramenti in tutti e tre i settori, ma il modello del sistema Bell di base è rimasto sostanzialmente lo stesso per 100 anni. Per una breve storia tecnica del sistema telefonico, vedere (Hawley, 1991).

Prima della divisione di AT&T avvenuta nel 1984, il sistema telefonico era organizzato secondo una gerarchia multilivello ad alta ridondanza. La spiegazione che segue è molto semplificata, tuttavia descrive il meccanismo fondamentale. Da ogni telefono partono due cavi di rame che si collegano direttamente alla centrale più vicina dell'azienda telefonica (chiamata anche **centrale locale**); di solito la centrale si trova a una distanza compresa tra 1 e 10 km, è più vicina nelle città mentre è più lontana nelle aree rurali. Negli Stati Uniti ci sono circa 22.000 centrali locali. Le connessioni bifilari tra ogni telefono e le centrali sono chiamate anche **collegamenti locali**. Se i collegamenti locali di tutto il mondo fossero stesi uno dopo l'altro, coprirebbero mille volte la distanza che separa la Terra dalla Luna.

Un tempo, l'ottanta per cento del valore di AT&T era rappresentato dal rame usato per costruire i collegamenti locali. AT&T era, in effetti, la più grande miniera di rame del mondo. Per fortuna questo fatto non era molto noto nella comunità degli investitori. Se lo fosse stato, alcuni razziatori di società avrebbero potuto acquistare AT&T, interrompere il servizio telefonico in tutti gli Stati Uniti, estrarre tutti i cavi e vendere il rame ai raffinatori per ottenere un rapido guadagno.

Se un abbonato collegato a una certa centrale locale chiama un altro abbonato collegato alla stessa centrale, il meccanismo di commutazione integrato nella centrale crea una connessione elettrica diretta tra i due collegamenti locali; questa connessione rimane attiva per tutta la durata della chiamata.

Se il telefono chiamato è collegato a un'altra centrale locale si attiva una procedura differente: ogni centrale locale ha diverse linee in uscita che conducono a uno o più centri di commutazione attigui chiamati **centrali interurbane**. Queste linee sono chiamate **linee di connessione interurbana**.

Se la centrale locale del chiamante e quella del chiamato hanno una linea di connessione interurbana diretta verso la stessa centrale interurbana (situazione molto comune quando le centrali sono relativamente vicine), è in quest'ultima che si può stabilire la connessione. La Figura 2.20(c) mostra una rete telefonica composta da telefoni (punti più piccoli), centrali locali (punti più grandi) e centrali interurbane (quadrati).

Se il chiamante e il chiamato non hanno una centrale interurbana in comune, il percorso sarà stabilito più in alto nella gerarchia. Centrali principali, locali e regionali costituiscono la rete alla quale si collegano le centrali interurbane; questi elementi comunicano tra

loro attraverso speciali linee a larga banda tra centrali. La varietà di centrali di commutazione e la topologia delle connessioni (per esempio, due centrali interurbane possono essere connesse direttamente o devono passare per una centrale regionale?) varia da paese a paese, secondo la densità telefonica. La Figura 2.21 mostra come potrebbe essere inoltre una connessione a media distanza.

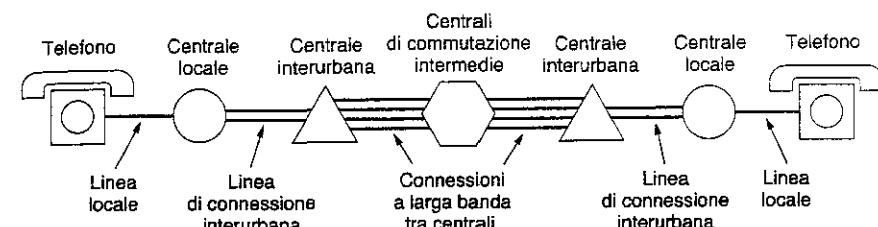


Figura 2.21. Il percorso tipico di una chiamata a media distanza.

Le telecomunicazioni sono realizzate utilizzando diversi tipi di mezzi di trasmissione. Oggi i collegamenti locali sono costruiti con doppini di categoria 3, mentre nei primi anni della telefonia venivano utilizzati cavi non isolati distanziati di 25 cm sui pali del telefono. Le centrali di commutazione utilizzano cavi coassiali, microonde e fibre ottiche. In passato la trasmissione attraverso il sistema telefonico era analogica: il segnale vocale era trasmesso dalla sorgente alla destinazione sotto forma di tensione elettrica. Con l'avvento delle fibre ottiche, dell'elettronica digitale e dei computer, tutte le linee e i commutatori sono diventati digitali; i collegamenti locali rappresentano l'ultimo tassello di tecnologia analogica del sistema. La trasmissione digitale è preferita perché rende superfluo riprodurre in modo accurato una forma d'onda analogica dopo il passaggio attraverso molti amplificatori in una chiamata a lunga distanza. È sufficiente poter distinguere correttamente uno 0 da un 1. Questa proprietà rende la trasmissione digitale più affidabile di quella analogica, inoltre è più economica e facile da gestire.

Riepilogando, il sistema telefonico è costituito da tre componenti principali:

1. i collegamenti locali (doppini analogici che arrivano nelle abitazioni e negli uffici)
2. linee (fibre ottiche digitali che collegano le centrali di commutazione)
3. centrali di commutazione (che spostano le chiamate da una linea all'altra).

Dopo un breve riepilogo della situazione normativa che riguarda la telefonia, il testo esaminerà in dettaglio ognuno di questi componenti. I collegamenti locali permettono l'accesso a tutti, perciò sono cruciali; sfortunatamente sono anche il punto debole del sistema. Per le linee a lunga distanza il problema principale riguarda il modo di raccogliere e trasmettere più chiamate attraverso la stessa fibra. Questo meccanismo è chiamato **multiplex** e il testo descrive tre diverse soluzioni. Infine, l'ultima parte è dedicata ai metodi utilizzati per realizzare la commutazione.

2.5.2 Le politiche dei telefoni

Nei decenni che hanno preceduto il 1984, Bell System ha gestito il servizio locale e quello sulle lunghe distanze attraverso gran parte degli Stati Uniti. Negli anni '70, il governo federale arrivò a credere che questo monopolio fosse illegale e fece di tutto per smantellarlo. Il governo vinse, e il primo gennaio 1984 AT&T venne divisa in AT&T Long Lines, 23 BOC (*Bell Operating Companies*) e pochi altri pezzi. Le 23 BOC vennero raggruppate in sette BOC regionali (RBOC) per renderle economicamente vitali. Nello spazio di una notte, l'intera natura delle telecomunicazioni negli Stati Uniti venne stravolta da un ordinamento del parlamento (non da un atto del Congresso).

Tutti i dettagli di questo disinvestimento sono descritti nel cosiddetto **MFJ** (*Modified Final Judgment*, evidente ossimoro poiché se il giudizio può essere modificato non è certo finale). Questo evento fece aumentare la concorrenza, migliorò il servizio e fece diminuire i prezzi delle telefonate interurbane. Comunque i prezzi dei servizi locali aumentarono: l'eliminazione dei sussidi incrociati derivanti dalle chiamate interurbane, infatti, costrinse il servizio locale ad auto sostenersi. Molti altri paesi hanno introdotto la concorrenza seguendo le stesse linee. Per far capire chi poteva fare cosa, gli Stati Uniti sono stati divisi in 164 LATA (*Local Access and Transport Area*); grosso modo, una LATA è grande quanto l'area coperta da un prefisso telefonico. In ogni LATA c'è un **LEC** (*Local Exchange Carrier*) che ha il monopolio del servizio telefonico tradizionale in quell'area. I LEC più importanti erano i BOC, comunque alcuni LATA contenevano una o più delle 1.500 compagnie telefoniche indipendenti che operavano come LEC. Tutto il traffico tra le LATA era gestito da una diversa azienda, definita **IXC** (*Interexchange Carrier*). Inizialmente AT&T Long Lines era l'unico vero IXC, ma ora WorldCom e Sprint sono diventati concorrenti stabili nel mercato IXC.

Durante la scissione, una delle preoccupazioni principali fu garantire che tutti gli IXC fossero trattati egualmente in termini di qualità della linea, tariffe e numero di cifre assegnate agli abbonati; la Figura 2.22 mostra come è stata soddisfatta questa necessità. L'immagine rappresenta tre esempi di LATA, ognuna con diverse centrali locali. LATA 2 e 3 hanno anche una piccola gerarchia con centrali interurbane.

Ogni IXC che desidera gestire le chiamate che hanno origine in una LATA può costruire all'interno dell'area una centrale di commutazione chiamata **POP** (*Point of Presence*). Il LEC deve collegare ogni IXC a ogni centrale locale, direttamente (come nel caso delle LATA 1 e 3) o indirettamente (come nel caso del LATA 2). Inoltre, le caratteristiche tecniche ed economiche della connessione devono essere identiche per tutti gli IXC; in tal modo, per esempio, un utente in LATA 1 può scegliere quale IXC utilizzare per chiamare gli abbonati in LATA 3.

L'MFJ vieta agli IXC di offrire servizi di telefonia locale e ai LEC di offrire servizi di telefonia tra LATA, ma entrambi possono entrare in altri settori commerciali, per esempio in quello della ristorazione. Quella del 1984 fu una legge abbastanza ambigua.

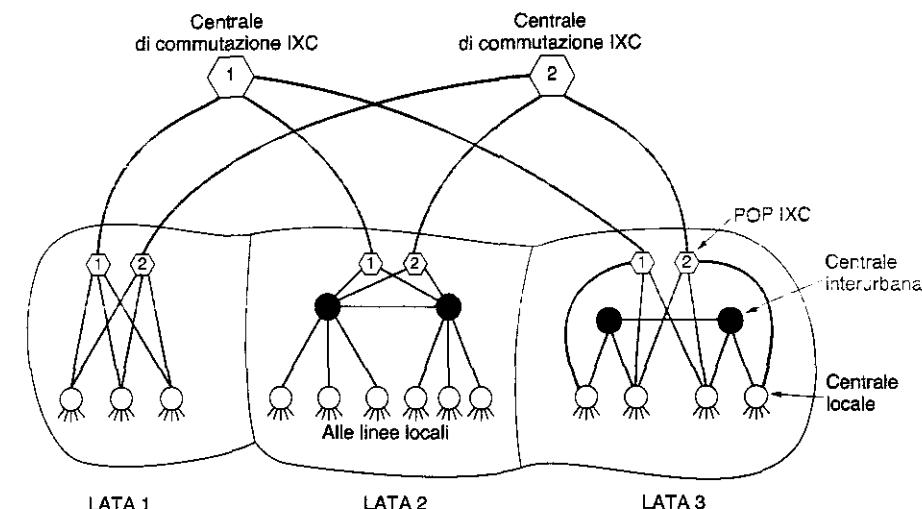


Figura 2.22. Le relazioni tra LATA, LEC e IXC. I cerchi sono le centrali di commutazione LEC. Ogni esagono appartiene all'IXC indicato dal numero al suo interno.

Sfortunatamente la tecnologia ha un modo simpatico di rendere obsoleta la legge. Né la televisione via cavo né i telefoni cellulari furono considerati nell'accordo. Con l'avvento della televisione via cavo bidirezionale e dei telefoni cellulari, LEC e IXC iniziarono ad acquistare o fondersi con operatori di telefonia cellulare e trasmissione via cavo.

Nel 1995 il Congresso capì che non era più possibile mantenere una distinzione tra i diversi tipi di aziende e preparò un documento che consentiva a televisioni via cavo, aziende telefoniche locali, operatori su lunghe distanze e operatori mobili l'ingresso in qualunque settore. L'idea era che ogni società poteva offrire ai suoi clienti un singolo pacchetto integrato comprendente TV via cavo, telefonia e servizi d'informazione, e che società diverse potevano competere sui servizi e sui prezzi. Il documento divenne legge nel febbraio 1996, e come conseguenza alcuni BOC divennero IXC e alcune altre aziende, tra cui televisioni via cavo, iniziarono a offrire servizi di telefonia locale concorrendo con i LEC.

Una proprietà interessante della legge del 1996 è il requisito che impone ai LEC di implementare la portabilità del numero locale; questo significa che un cliente può cambiare il fornitore del servizio telefonico senza cambiare il numero di telefono. Questo provvedimento rimuove quello che per molti rappresentava un enorme ostacolo, rende gli utenti più inclini al cambio di LEC e aumenta la concorrenza. Oggi le telecomunicazioni statunitensi stanno vivendo una radicale ristrutturazione, e molti altri paesi stanno iniziando a comportarsi allo stesso modo. Gli Stati Uniti fanno infatti da modello per molte nazioni: se l'esperimento riesce altri seguono la stessa strada; in caso contrario si tenta qualcosa d'altro.

2.5.3 I collegamenti locali: modem, ADSL e connessioni wireless

È giunto il momento di iniziare l'esame dettagliato del funzionamento del sistema telefonico. La Figura 2.23 mostra le parti principali di questo sistema; il disegno rappresenta i collegamenti locali, le linee e le centrali interurbane e locali che contengono gli apparecchi di commutazione che smistano le chiamate. Una centrale locale ha un massimo di 10.000 collegamenti locali (negli Stati Uniti e in altre grandi nazioni). In effetti, fino a poco tempo fa il prefisso telefonico rappresentava la centrale locale; per esempio il numero (212) 601-xxxx rappresentava una specifica centrale locale contenente 10.000 abbonati numerati da 0000 a 9999. Con l'avvento della concorrenza nel servizio locale il sistema non poté più essere sostenuto, perché più aziende volevano possedere il codice della centrale locale. Il numero stesso dei codici si era sostanzialmente esaurito, per questo vennero introdotti schemi di mappatura più complessi.

Ma veniamo alla parte del sistema con cui gli utenti hanno più familiarità, ossia il collegamento locale bifilare che collega la centrale della società telefonica alle abitazioni e piccoli uffici. Il collegamento locale è chiamato anche ultimo miglio, sebbene la lunghezza superi spesso questa estensione; ha utilizzato la trasmissione analogica per più di 100 anni e probabilmente continuerà a farlo per gli anni a venire, anche a causa degli elevati costi richiesti dalla conversione al digitale. Ciò nonostante, anche in questo bastione della trasmissione analogica avvengono dei cambiamenti. Questo paragrafo esamina il collegamento locale tradizionale e i nuovi sviluppi che stanno avendo luogo, con particolare enfasi alla comunicazione di dati da computer domestici.

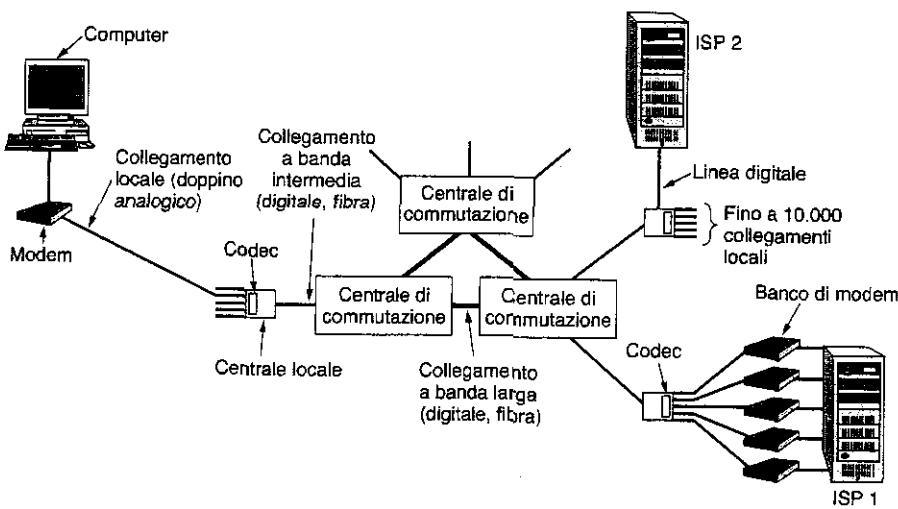


Figura 2.23. L'utilizzo di trasmissione analogica e digitale durante una chiamata da computer a computer. La conversione è gestita dai modem e dai codec.

Per inviare segnali digitali attraverso una linea telefonica, il computer deve convertire i dati in forma analogica; solo in questo modo le informazioni possono esser trasmesse attraverso un collegamento locale. La conversione è eseguita da un apparecchio chiamato modem. Nella centrale della società telefonica i dati sono riconvertiti in formato digitale prima di essere trasmessi attraverso le linee a lunga distanza. Se all'altro capo della linea c'è un computer collegato a un modem, è necessario eseguire una conversione inversa, da digitale ad analogico; in questo modo le informazioni possono attraversare il collegamento locale e raggiungere la destinazione. La Figura 2.23 mostra l'ISP 1 (*Internet Service Provider*) che posiede un banco di modem, ognuno collegato a un diverso collegamento locale.

L'ISP può gestire tante connessioni quanti sono i suoi modem (supponendo che il suo server abbia una capacità di calcolo sufficiente). Questa è stata la configurazione normale fino all'apparizione dei modem a 56 kbps, che (per motivi che verranno presto esaminati) hanno imposto dei cambiamenti. La trasmissione di segnali analogici è ottenuta variando la tensione nel tempo, per rappresentare un flusso d'informazioni. Se il mezzo di trasmissione fosse perfetto, il ricevitore riceverebbe esattamente lo stesso segnale inviato dal trasmettitore. Sfortunatamente, i mezzi di trasmissione non sono perfetti, perciò il segnale ricevuto non è mai identico a quello trasmesso. Nel caso dei dati digitali, questa differenza può generare errori.

I problemi principali delle linee di trasmissione sono tre: attenuazione, distorsione per ritardo e rumore. L'**attenuazione** rappresenta la perdita di energia causata dalla propagazione del segnale verso l'esterno. La perdita è espressa in dB per Km. La quantità di energia persa dipende dalla frequenza. Per vedere gli effetti di questa dipendenza dalla frequenza è utile immaginare il segnale non come una semplice forma d'onda, ma come una serie di componenti di Fourier. Ogni componente subisce un diverso livello di attenuazione, e al ricevitore si ottiene un diverso spettro di Fourier. A peggiorare il tutto, i diversi componenti di Fourier si propagano anche a velocità differenti attraverso il cavo; questa differenza di velocità **distorce** il segnale ricevuto all'altro capo.

Un altro problema è il **rumore**, che rappresenta l'energia indesiderata generata da sorgenti esterne al trasmettitore. Il **rumore termico** è causato dal movimento casuale degli elettroni nel cavo ed è inevitabile. L'interferenza è causata da accoppiamenti induttivi tra due cavi troppo vicini. Qualche volta quando si parla al telefono si riesce a sentire l'eco di un'altra conversazione; questo fenomeno è causato dall'interferenza. Infine, c'è il rumore d'impulso che può essere causato da picchi sulla linea di alimentazione oppure da altre fonti. Nel caso dei dati digitali, il rumore d'impulso può cancellare uno o più bit.

Modem

A causa dei problemi appena descritti, e soprattutto perché l'attenuazione e la velocità di propagazione dipendono dalla frequenza, è meglio che il segnale non abbia un largo intervallo di frequenze. Sfortunatamente, le onde quadre utilizzate nei segnali digitali utilizzano un ampio spettro di frequenza e perciò sono soggette a una forte attenuazione e alla distorsione. Questi effetti rendono adatta la trasmissione in banda base (DC) solo a velocità basse e distanze brevi.

per aggirare i problemi associati alla trasmissione DC (specialmente sulle linee telefoniche) si usa la trasmissione AC: viene introdotto un tono continuo, nell'intervallo compreso tra i 1.000 e i 2.000 Hz, chiamato **portante d'onda sinusoidale**. La sua ampiezza, la frequenza o la fase possono essere modulate per trasmettere informazioni.

Nella **modulazione d'ampiezza** due diverse ampiezze sono utilizzate per rappresentare spettivamente 0 e 1; nella **modulazione di frequenza** (chiamata anche FSK, *Frequency Shift Keying*) si utilizzano due o più toni. Nella forma più semplice di **modulazione di fase**, l'onda portante è spostata sistematicamente di 0 oppure 180 gradi a intervalli uniformi. Uno schema migliore si basa su spostamenti di 45, 135, 225 o 315 gradi, per trasmettere 2 bit d'informazione per intervallo di tempo. Inoltre, per aiutare il ricevitore a conoscere i limiti degli intervalli temporali si può imporre la necessità di un cambiamento di fase alla fine di ogni intervallo.

A Figura 2.24 mostra le tre forme di modulazione. Nella Figura 2.24(b) una delle ampiezze è diversa da zero e una è uguale a zero. Nella Figura 2.24(c) sono utilizzate due frequenze. Nella Figura 2.24(d) si verifica oppure no uno spostamento di fase nel punto di

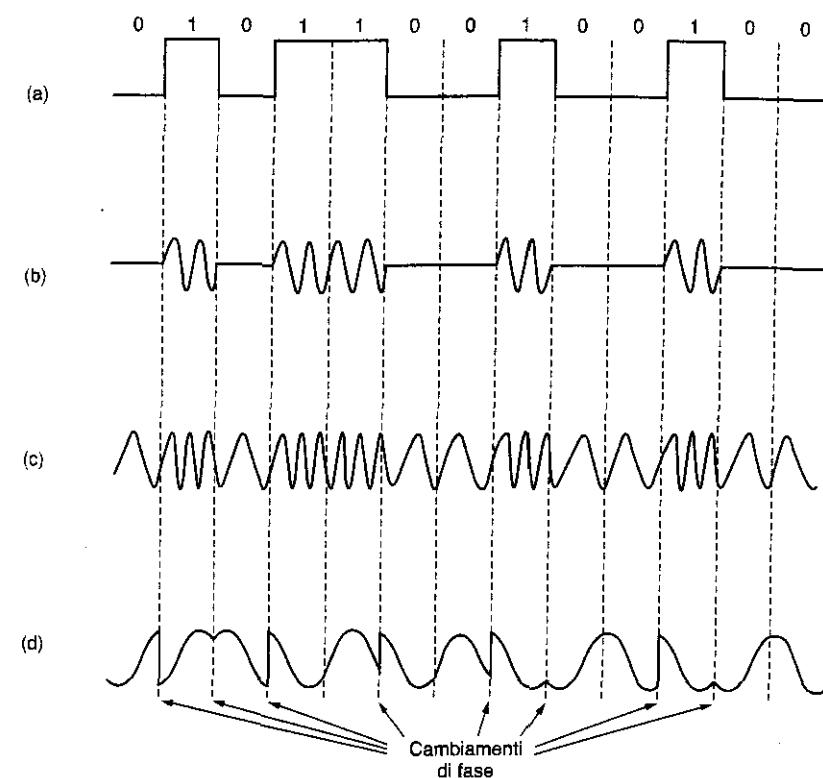


Figura 2.24. (a) Un segnale binario. (b) Modulazione di ampiezza. (c) Modulazione di frequenza. (d) Modulazione di fase.

confine tra due bit successivi. Un apparecchio che accetta un flusso seriale di bit in ingresso e produce una portante modulata attraverso uno (o più) di questi metodi (o viceversa) è chiamato **modem** (acronimo di modulatore-demodulatore). Il modem si trova tra il computer (digitale) e il sistema telefonico (analogico).

Per raggiungere velocità sempre più elevate non è possibile aumentare semplicemente la frequenza di campionamento. Il teorema di Nyquist afferma che anche con una linea a 3.000 Hz perfetta (e la linea telefonica decisamente non lo è), non c'è alcun modo di ottenere un campionamento più veloce di 6.000 Hz. In pratica, la maggior parte dei modem campiona 2.400 volte al secondo, cercando di ottenere il maggior numero di bit per campione.

Il numero di campioni al secondo è misurato in **baud**. Durante ogni baud viene trasmesso un **simbolo**. Perciò una linea a n baud trasmette n simboli al secondo. Per esempio, una linea a 2.400 baud invia un simbolo ogni 416,667 μsec. Se il simbolo rappresenta con 0 Volt uno 0 logico e con 1 Volt un 1 logico, la frequenza di bit è di 2.400 bps. Se invece si utilizzano le tensioni 0, 1, 2 e 3 volt, ogni simbolo è composto da 2 bit, perciò una linea a 2.400 baud può trasmettere 2.400 simboli al secondo a una velocità dati di 4.800 bps. In modo analogo, con quattro possibili cambiamenti di fase ci sono 2 bit per simbolo, perciò di nuovo la frequenza di bit è doppia della frequenza di baud. L'ultima tecnica, la più utilizzata, è chiamata **QPSK** (*Quadrature Phase Shift Keying*).

I concetti di banda passante, baud, simboli e frequenza di bit sono spesso confusi, perciò è meglio riepilogarli. La banda passante di un mezzo di trasmissione è l'intervallo di frequenze che passa attraverso il mezzo con un'attenuazione minima; è una proprietà fisica del mezzo (di solito varia da 0 a una frequenza massima) ed è misurata in Hz. Il baud rate rappresenta il numero di campioni presi ogni secondo. Ogni campione rappresenta un pezzo d'informazione, ossia un simbolo. Baud rate e frequenza dei simboli sono perciò la stessa cosa. La tecnica di modulazione (per esempio QPSK) determina il numero di bit per simbolo. La frequenza di bit è la quantità di informazione inviata attraverso il canale ed è uguale al numero di simboli/sec moltiplicato per il numero di bit/simbolo.

Tutti i modem evoluti utilizzano una combinazione di tecniche di modulazione per trasmettere più bit per baud. Spesso per trasmettere diversi bit/simbolo si combinano insieme più ampiezze e spostamenti di fase. La Figura 2.25(a) mostra i punti a 45, 135, 225 e 315 gradi con ampiezza costante (distanza dall'origine). La fase di un punto è rappresentata dall'angolo che la linea che collega il punto e l'origine forma con l'asse X. La Figura 2.25(a) ha quattro combinazioni valide e può essere utilizzata per trasmettere 2 bit per simbolo. Si tratta della QPSK.

La Figura 2.25(b) mostra un diverso schema di modulazione: sono utilizzate quattro ampiezze e quattro fasi per un totale di 16 diverse combinazioni. Questo schema di modulazione permette di trasmettere 4 bit per simbolo ed è chiamato **QAM-16** (*Quadrature Amplitude Modulation*); qualche volta è utilizzato il termine 16-QAM. QAM-16 permette, per esempio, di trasmettere 9.600 bps attraverso una linea a 2.400 baud.

La Figura 2.25(c) mostra un altro schema di modulazione che coinvolge sia l'ampiezza sia la fase. Permette 64 diverse combinazioni, perciò consente di trasmettere 6 bit per simbolo. Lo schema è chiamato **QAM-64**. Esistono anche QAM di ordine più alto.

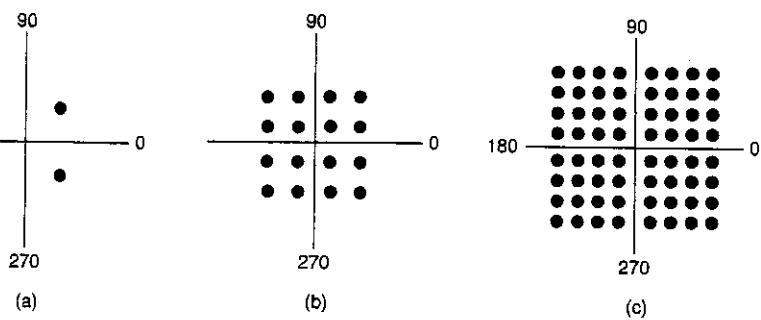


Figura 2.25. (a) QPSK. (b) QAM-16. (c) QAM-64.

diagrammi come quelli nella Figura 2.25, che mostrano le combinazioni valide di ampiezza e fase, sono chiamati **diagrammi costellazione**. Ogni modem ad alta velocità ha il suo schema di costellazione e può comunicare solo con altri modem che adottano lo stesso schema (ma la maggior parte dei modem è in grado di emulare tutti quelli più lenti). In molti punti nello schema di costellazione, anche un piccolo livello di rumore nell'ampiezza o nella fase rilevata può provocare un errore e, potenzialmente, può far perdere molti bit. Per ridurre la possibilità di errore, gli standard adottati dai modem ad alta velocità implementano meccanismi di correzione degli errori basati su bit extra aggiunti a ogni campione. Gli schemi sono chiamati **TCM (Trellis Coded Modulation)**. Così, per esempio, lo standard per modem V.32 utilizza 32 punti costellazione per trasmettere 4 bit dati e 1 bit di parità per simbolo a 2.400 baud, raggiungendo i 9.600 bps con correzione d'errore; la Figura 2.26(a) mostra il suo diagramma costellazione. La decisione di ruotare intorno l'origine di 45 gradi è stata presa per motivi di progettazione: le costellazioni rotate e non ruotate hanno la stessa capacità di dati.

Il passo successivo sopra i 9.600 bps è rappresentato dai 14.400 bps ed è chiamato **V.32 bis**. Questa velocità è ottenuta trasmettendo 6 bit di dati e 1 bit di parità per campione a 400 baud; il suo diagramma di costellazione, composto da 128 punti quando si utilizza QAM-128, è mostrato nella Figura 2.26(b). I modem fax utilizzano questa velocità per trasmettere le pagine digitalizzate. QAM-256 non è utilizzato da alcun modem telefonico standard ma è adoperato sulle reti via cavo.

Il standard per modem telefonici successivo a V.32 bis si chiama **V.34**, e trasmette 28.800 bps a 2.400 baud con 12 bit di dati per simbolo. Lo standard finale di questa serie è il **V.34bis**: utilizza 14 bit di dati per simbolo a 2.400 baud e raggiunge la velocità di 33.600 bps. Per portare la velocità dati effettiva oltre i 33.600 bps, molti modem comprimono i dati prima di trasmetterli. D'altra parte, quasi tutti i modem provano la linea prima di iniziare a trasmettere i dati dell'utente, e se la qualità della connessione non è sufficiente, diminuiscono automaticamente la loro velocità di trasmissione. Questo significa che la velocità effettiva dei modem osservata dagli utenti può essere più bassa, uguale o più alta di quella ufficiale.

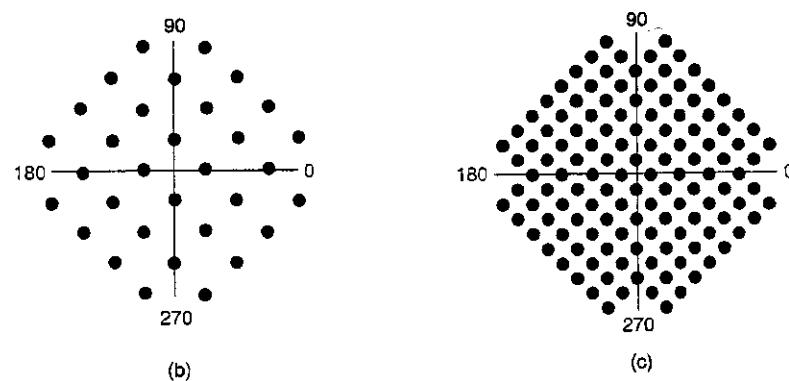


Figura 2.26. (a) V.32 per 9.600 bps. (b) V.32 bis per 14.400 bps.

Tutti i modem moderni permettono di trasmettere contemporaneamente in entrambe le direzioni (usando una frequenza diversa per ogni direzione). Una connessione che permette ai dati di viaggiare contemporaneamente in entrambi i sensi è detta **full duplex**; un esempio di linea full duplex è una strada a due corsie. Una connessione che permette ai dati di scorrere in entrambi i sensi ma solo un senso alla volta è chiamata **half duplex**; un esempio di linea half duplex è il binario ferroviario. Infine, una connessione che permette di trasmettere i dati in una sola direzione è chiamata **simplex**; un esempio di linea simplex è una strada a senso unico. Un altro esempio di connessione simplex è la fibra ottica con un laser a un capo della linea e un rilevatore di luce all'altro capo.

I modem standard si fermano a 33.600 bps, perché il limite di Shannon relativo al sistema telefonico è di circa 35 kbps, perciò il superamento di questa velocità violerebbe le leggi della fisica (in particolare quelle della termodinamica). Come mai il limite è proprio di 35 kbps? Il valore limite dipende dalla lunghezza media dei collegamenti locali e dalla loro qualità, e in particolare 35 kbps è il tetto imposto dalla lunghezza media. La Figura 2.23 mostra una chiamata originata dal computer posto a sinistra; prima di raggiungere l'ISP 1, la chiamata attraversa due collegamenti locali sotto forma di segnale analogico, una volta alla sorgente e un'altra alla destinazione. Ognuna di queste connessioni aggiunge rumore al segnale; se fosse possibile saltare uno di questi collegamenti locali si riuscirebbe a raddoppiare la velocità massima. ISP 2 fa esattamente questo, grazie alla linea tutta digitale proveniente dalla centrale locale più vicina. Il segnale digitale utilizzato sulle linee è inviato direttamente all'ISP 2: ciò elimina i codec, i modem e la trasmissione analogica alla sua estremità. Perciò quando un capo della connessione è completamente digitale, come nel caso della maggior parte degli ISP attuali, la velocità massima può arrivare anche a 70 kbps. Tra due utenti domestici che adoperano modem e linee analogiche, invece, la velocità massima è di 33,6 kbps.

Il motivo per cui è possibile utilizzare modem a 56 kbps ha a che fare con il teorema di Nyquist. Il canale telefonico ha un'ampiezza di circa 4.000 Hz (incluse le bande di guar-

dia). Il numero massimo di campioni indipendenti al secondo è perciò 8.000. Il numero di bit per campione negli Stati Uniti è 8; uno di questi è utilizzato per operazioni di controllo; quindi, ci sono 56.000 bit/sec a disposizione dei dati degli utenti. In Europa gli utenti hanno a disposizione tutti e 8 i bit, perciò i modem potrebbero raggiungere una velocità di 64.000 bit/sec; invece, per adeguarsi agli standard internazionali, anche l'Europa ha adottato la velocità di 56.000 bps.

Questo standard, chiamato **V.90**, fornisce un canale di trasmissione (da utente a ISP) a 33,6 kbps e un canale di ricezione (da ISP a utente) a 56 kbps, perché di solito gli utenti scaricano più di quanto trasmettono (per esempio, la richiesta di una pagina Web occupa solo pochi byte, mentre la pagina vera e propria può essere grande diversi MB). In teoria sarebbe possibile utilizzare un canale di trasmissione più grande di 33,6 kbps ma poiché molti collegamenti locali sono troppo rumorosi anche per i 33,6 kbps, si è deciso di allocare una maggiore ampiezza di banda al canale in ricezione in modo da aumentare le possibilità di funzionare a 56 kbps. Lo standard successivo al V.90 si chiama **V.92**. I modem che adottano questo standard possono raggiungere una velocità di trasmissione di 48 kbps, se la linea è in grado di gestirla; gli apparecchi riescono anche a determinare la velocità supportata dalla linea nella metà del tempo impiegato dai modelli più vecchi (in genere 30 secondi). Infine, i modem V.92 permettono alle chiamate telefoniche in arrivo di interrompere le sessioni di Internet, a patto che la linea abbia un servizio di attesa.

Linee DSL (*Digital Subscriber Line*)

Quando raggiunse finalmente i 56 kbps, l'industria telefonica si congratulò con se stessa. Nel frattempo, però, l'industria televisiva aveva iniziato a offrire velocità che raggiungevano i 10 Mbps sui cavi condivisi e le società di comunicazione satellitare si preparavano a proporre servizi di trasmissione fino a 50 Mbps. Quando l'accesso a Internet divenne una parte importante dell'attività commerciale, le aziende telefoniche (LEC) si resero conto di aver bisogno di un prodotto più competitivo. Per questo iniziarono a offrire nuovi servizi digitali sui collegamenti locali. I servizi con un'ampiezza di banda superiore a quella del servizio telefonico standard si dicono a **banda larga**; il termine, comunque, rappresenta più un concetto di marketing che un concetto tecnico.

Inizialmente ci furono molte offerte sovrapposte, tutte raccolte sotto un unico nome: **xDSL (Digital Subscriber Line)**. Il paragrafo esamina tutti questi servizi, concentrandosi soprattutto su quello che probabilmente diventerà il più popolare, ossia **ADSL (Asymmetric DSL)**. Poiché ADSL è ancora in evoluzione e non tutti gli standard sono stati completati, alcuni dettagli riportati di seguito potrebbero cambiare col tempo, ma il quadro di base dovrebbe rimanere valido.

I modem sono così lenti perché i telefoni sono stati inventati per trasmettere la voce umana e l'intero sistema è stato attentamente ottimizzato per questo scopo; i dati sono sempre stati considerati come figliastri. Nel punto in cui ogni collegamento locale termina nella centrale locale, il cavo attraversa un filtro che attenua tutte le frequenze sotto i 300 Hz e sopra i 3.400 Hz. Il limite non è netto (300 Hz e 3.400 Hz sono i punti a -3 dB) perciò l'ampiezza di banda si considera solitamente 4.000 Hz anche se la distanza tra i punti a -3 dB è di 3.100 Hz. I dati sono perciò limitati a questa banda stretta.

Il trucco che fa funzionare xDSL è che quando un cliente si abbona a questo servizio, la linea in ingresso viene collegata a un diverso tipo di commutatore che, non usando questo filtro, rende disponibile l'intera capacità del collegamento locale. In questo modo il fattore limitante è diventato l'insieme di proprietà fisiche del collegamento locale, non l'ampiezza di banda artificiale 3.100 Hz creata dal filtro.

Purtroppo, la capacità del collegamento locale dipende anche da altri fattori, inclusa la sua lunghezza, il diametro dei cavi e la qualità generale. La Figura 2.27 fornisce un quadro d'insieme dell'ampiezza di banda potenziale come funzione della distanza; la figura presuppone che tutti gli altri fattori siano ottimali (cavi nuovi, fasci non troppo grandi e così via).

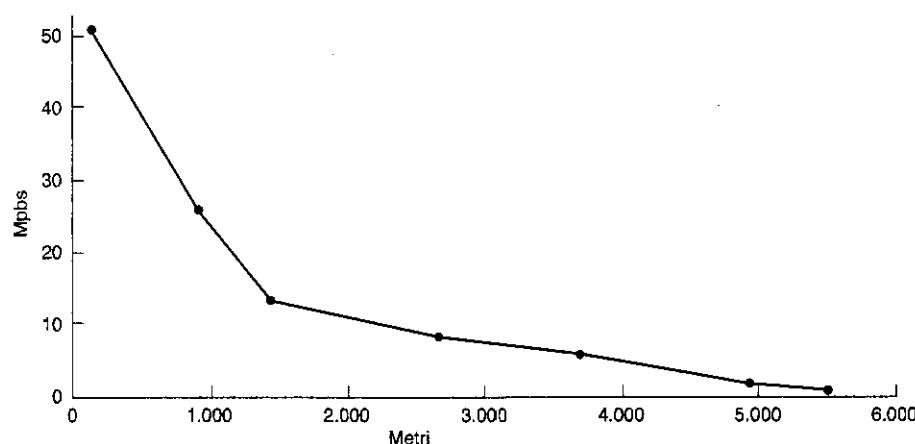


Figura 2.27. Ampiezza di banda e distanza per UTP di categoria 3 usati da DSL.

L'implicazione di questa figura crea un problema per l'azienda telefonica: la portata massima (distanza dalle centrali locali) del servizio dipende dalla velocità che si sceglie di offrire. Questo significa che gli utenti più distanti che tentano di abbonarsi al servizio potrebbero sentirsi dire: "La ringraziamo per l'interesse dimostrato, ma purtroppo la sua abitazione dista 100 metri di troppo dalla più vicina centrale locale". Più è bassa la velocità della connessione, più è ampio il raggio del servizio e maggiore è il numero di clienti raggiunti; ma al diminuire della velocità, diventa meno allentante il servizio e diminuisce il numero di utenti disposti a pagare per ottenerlo. È qui che il mondo degli affari incontra la tecnologia (una soluzione possibile, anche se costosa, potrebbe essere quella di costruire nuove piccole centrali locali). I servizi xDSL sono stati progettati tenendo conto di alcuni obiettivi. Primo, i servizi devono funzionare sui doppini di categoria 3 esistenti adoperati per i collegamenti locali; secondo, non devono influire sugli apparecchi telefonici e sui fax in dotazione degli utenti; terzo, devono essere molto più veloci della connessione a 56 kbps; quarto, dovrebbero sempre essere attivi e avere un costo mensile non legato al tempo di utilizzo.

Il primo servizio ADSL offerto da AT&T divideva lo spettro disponibile sui collegamenti locali (circa 1,1 MHz) in tre bande di frequenza: **POTS** (*Plain Old Telephone Service*), **upstream** (dall'utente alla centrale) e **downstream** (dalla centrale all'utente). La tecnica basata su diverse bande di frequenza è chiamata multiplexing a divisione di frequenza e verrà esaminata in dettaglio più avanti. Le offerte successive proposte da altri fornitori hanno seguito un approccio diverso, che poi si è dimostrato vincente. L'approccio alternativo, chiamato **DMT** (*Discrete MultiTone*), è mostrato nella Figura 2.28. Lo spettro 1,1 MHz disponibile sui collegamenti locali è diviso in 256 canali indipendenti, ognuno ampio 4.312,5 Hz. Il canale 0 è utilizzato per POTS; per evitare che il segnale vocale e quello dati interferiscano tra loro, i canali da 1 a 5 non sono utilizzati; dei canali rimanenti, uno è utilizzato per il controllo della trasmissione e un altro per il controllo della ricezione. Tutti gli altri canali sono a disposizione dei dati degli utenti.

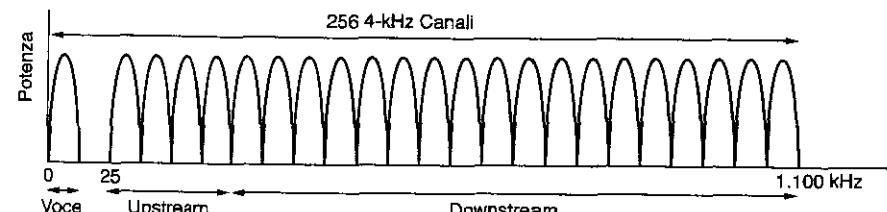


Figura 2.28. ADSL basata su DMT.

In linea di principio, ognuno dei canali rimanenti può essere utilizzato per un flusso di dati full duplex, ma le armoniche, l'interferenza e altri effetti impediscono ai sistemi pratici di avvicinarsi al limite teorico. Il fornitore del servizio decide quanti canali utilizzare per la trasmissione e quanti per la ricezione. Una combinazione 50 e 50 è tecnicamente possibile, ma la maggior parte dei fornitori assegna al canale utilizzato per scaricare i dati l'80-90% dell'ampiezza di banda, poiché la maggior parte degli utenti preleva più di quanto trasmetta. Questa soluzione fa nascere la "A" (Asimmetrica) di ADSL. Di solito 32 canali sono dedicati alla trasmissione e il resto alla ricezione. È possibile aumentare l'ampiezza di banda rendendo bidirezionali alcuni dei canali più alti in trasmissione, comunque questa ottimizzazione richiede l'aggiunta di uno speciale circuito per annullare l'effetto eco. Lo standard ADSL (ANSI T1.413 e ITU G.992.1) supporta velocità che possono arrivare a 8 Mbps in ricezione e 1 Mbps in trasmissione, ma solo pochi fornitori offrono questa velocità. In genere il servizio offre 512 kbps in ricezione e 64 kbps in trasmissione (servizio standard) oppure 1 Mbps in ricezione e 256 in trasmissione (servizio premium). Dentro ogni canale si utilizza uno schema di modulazione simile a quello di V.34, dove la velocità di campionamento è di 4.000 baud invece di 2.400. La qualità della linea in ogni canale è tenuta costantemente sotto controllo e la velocità dati è regolata continuamente, perciò canali differenti possono avere velocità diverse. I dati sono trasmessi con modula-

zione QAM, con un massimo di 15 bit per baud, usando un diagramma di costellazione simile a quello mostrato nella Figura 2.25(b). In teoria con 224 canali in ricezione e 15 bit per baud a 4.000 baud l'ampiezza di banda in ricezione è di 13,44 Mbps, ma in pratica il rapporto segnale rumore non consente di raggiungere questa velocità; è possibile raggiungere una velocità di 8 Mbps su brevi tratti di collegamenti locali di alta qualità.

La Figura 2.29 mostra una configurazione ADSL tipica. In questo schema, un tecnico della società telefonica deve installare un **NID** (*Network Interface Device*) nell'edificio del cliente. Questa piccola scatola di plastica segna la fine della proprietà dell'azienda telefonica e l'inizio della proprietà del cliente. Accanto al NID (e qualche volta al suo interno) si trova uno **splitter**, filtro analogico che divide i dati dalla banda 0-4.000 Hz utilizzata da POTS. Il segnale POTS è inviato all'apparecchio telefonico o al fax esistente, mentre i dati sono instradati verso un modem ADSL. Il modem ADSL è in realtà un elaboratore di segnali digitali, che fa le veci di 250 modem QAM che operano in parallelo a diverse frequenze. Poiché la maggior parte dei modem ADSL è esterna, il computer deve essere collegato all'apparecchio attraverso una connessione veloce; di solito si utilizza una scheda Ethernet oppure una porta USB.

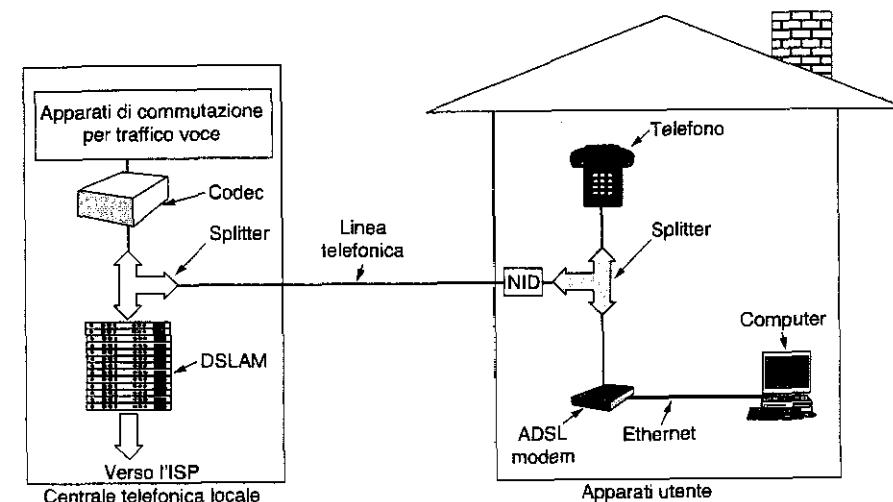


Figura 2.29. Una configurazione ADSL tipica.

All'altra estremità del cavo, nella centrale locale, deve essere installato un analogo splitter. Qui il segnale vocale è filtrato e inviato al normale commutatore per il traffico vocale. Il segnale sopra i 26 kHz, invece, è passato a un nuovo tipo di dispositivo chiamato **DSLAM** (*Digital Subscriber Line Access Multiplexer*) che contiene lo stesso tipo di processore di segnale digitale integrato nel modem ADSL. Dopo essere stato riorganizzato in

un flusso di bit, il segnale digitale è prima organizzato in pacchetti e poi inviato all'ISP. Questa completa separazione tra sistema vocale e ADSL rende relativamente facile per l'azienda telefonica allestire il servizio ADSL: è sufficiente acquistare i dispositivi DSLAM, gli splitter, e collegare gli abbonati ADSL agli splitter. Altri servizi ad alta ampiezza di banda (per esempio ISDN) richiedono un maggior numero di modifiche.

Uno svantaggio della configurazione mostrata nella Figura 2.29 è la presenza del NID e dello splitter nell'edificio del cliente. Poiché questi componenti possono essere installati solo da tecnici dell'azienda telefonica, è necessario organizzare e pagare un intervento sul luogo. Per questo è stato creato uno standard alternativo basato su una configurazione che non usa splitter; ufficiosamente è stato chiamato G.lite, ma il suo codice ITU è G.992.2. La configurazione è identica a quella mostrata nella Figura 2.29, ma non ha lo splitter. La linea telefonica esistente è utilizzata così com'è; l'unica differenza è che si deve inserire un microfiltro tra ogni presa telefonica e il telefono o modem collegato. Il filtro per il telefono è un low-pass che elimina le frequenze sopra i 3.400 Hz; il filtro per il modem ADSL è un high-pass che elimina le frequenze sotto i 26 kHz.

Comunque questo sistema non è affidabile come quello basato sullo splitter, perciò G.lite può essere utilizzato solo fino a 1,5 Mbps (mentre la configurazione ADSL con splitter arriva a 8 Mbps). G.lite richiede ancora la presenza di uno splitter nella centrale locale, ma questa installazione non è costosa come quella nell'abitazione dell'abbonato.

ADSL è semplicemente uno standard di strato fisico; ciò che funziona al di sopra dipende dal gestore. Spesso viene scelto ATM per la sua capacità di gestire la qualità del servizio e per il fatto che le aziende telefoniche utilizzano già questo protocollo nella rete centrale.

Collegamenti locali wireless

Dal 1996 negli Stati Uniti (e qualche anno dopo nelle altre nazioni) le società chiamate **ILEC** (*Incumbent LEC*) che desiderano competere con l'azienda telefonica consolidata (il vecchio monopolista) sono libere di farlo. I candidati più probabili sono le società telefoniche su lunghe distanze (IXC). Un IXC che desidera entrare nel mercato telefonico locale di una città deve: acquistare o affittare un edificio per la sua prima centrale locale in quella città; installare nella centrale i commutatori telefonici e le altre apparecchiature messe in commercio da diversi produttori; stendere una fibra tra la centrale locale e la centrale interurbana più vicina in modo da consentire ai nuovi clienti locali di accedere alla sua rete nazionale; acquisire clienti, in genere proponendo un miglior servizio o prezzi più bassi di quelli attuali.

A questo punto inizia la parte più difficile. Supponiamo che si presentino alcuni clienti; in che modo la nuova azienda telefonica locale, chiamata **CLEC** (*Competitive LEC*), intende collegare i telefoni e i computer dei clienti alla sua nuova e scintillante centrale locale? L'acquisto dei diritti necessari per stendere i propri cavi o fibre è una spesa proibitiva. Molte CLEC hanno scoperto un'alternativa più economica al collegamento locale tradizionale basato su doppini: il collegamento **WLL** (*Wireless Local Loop*).

Da un certo punto di vista, un telefono fisso che utilizza un collegamento locale wireless assomiglia a un telefono cellulare; ci sono però tre differenze cruciali: primo, il cliente del collegamento locale wireless spesso desidera poter accedere a Internet a velocità elevata, almeno pari a quella di ADSL; secondo, il nuovo cliente probabilmente non si preoccupa se un tecnico della CLEC installa sul tetto della sua abitazione una grande antenna direzionale che punta verso la centrale della CLEC; terzo, l'utente non si sposta, e questo elmina tutti i problemi legati alla mobilità e al cambiamento di cella che verrà esaminato in dettaglio più avanti.

Così è nata una nuova industria, quella delle **trasmissioni fisse wireless**: servizio Internet e di telefonia locale gestito da CLEC e basato su collegamenti locali wireless. WLL è diventata una faccenda seria nel 1998, ma le sue origini risalgono al 1969. Quell'anno la FCC assegnò due canali televisivi (ciascuno a 6 MHz) al servizio d'istruzione televisivo a 2,1 GHz. Negli anni successivi furono aggiunti altri 31 canali a 2,5 GHz, per un totale di 198 MHz. Il servizio d'istruzione televisivo non decollò mai e nel 1998 la FCC assegnò quelle frequenze alla trasmissione radio bidirezionale. I collegamenti locali si appropriarono immediatamente delle nuove frequenze.

A queste frequenze le microonde sono lunghe 10...12 cm, possono arrivare a 50 Km di distanza e attraversare la vegetazione e la pioggia leggera. I 198 MHz del nuovo spettro sono stati subito utilizzati per un servizio di collegamento locale wireless chiamato **MMDS** (*Multichannel Multipoint Distribution Service*). MMDS, come LMDS (descritto più avanti) può essere considerato una sorta di MAN (*Metropolitan Area Network*). Il grande vantaggio di questo servizio è che utilizza una tecnologia stabile, inoltre tutta l'attrezzatura è già in commercio. Lo svantaggio è che l'ampiezza di banda totale disponibile è modesta e deve esser condivisa da molti utenti su una vasta area geografica.

La scarsa ampiezza di banda di MMDS porta a valutare come alternativa le onde millimetriche. Alle frequenze di 28...31 GHz negli Stati Uniti e di 40 GHz in Europa non era stata assegnata alcuna frequenza, perché è difficile costruire circuiti integrati di silicio in grado di operare a quelle velocità. I circuiti integrati all'Arsenio di Gallio hanno risolto il problema e hanno aperto le porte alla comunicazione a onde radio sulle bande millimetriche.

FCC rispose alla domanda assegnando 1,3 GHz a un nuovo servizio di collegamento locale wireless chiamato **LMDS** (*Local Multipoint Distribution Service*). Questa assegnazione rappresenta il più grande intervallo singolo di ampiezza di banda mai allocato da FCC. Una porzione simile è stata assegnata in Europa ma a 40 GHz.

La Figura 2.30 mostra come funziona LMDS, raffigurando una torre con diverse antenne, dove ognuna punta in una diversa direzione. Poiché le onde millimetriche sono molto direzionali, ogni antenna definisce un settore indipendente dagli altri. A queste frequenze è possibile raggiungere distanze comprese tra i 2 e i 5 Km, e ciò significa che per coprire un'intera città sono necessarie diverse torri.

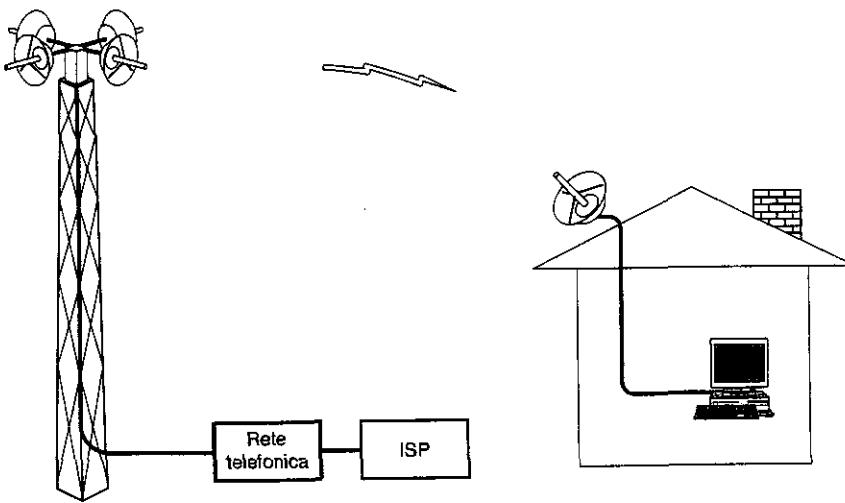


Figura 2.30. Architettura di un sistema LMDS.

Come ADSL, anche LMDS utilizza un'allocazione di ampiezza di banda asimmetrica che favorisce il canale in ricezione. La tecnologia corrente permette di assegnare a ogni settore 36 Gbps in ricezione e 1 Mbps in trasmissione; queste velocità sono condivise tra tutti gli utenti che si trovano nel settore. Se scarica tre pagine da 5 KB al minuto, ogni utente attivo occupa una media di 2.000 bps di spettro; questo significa che è possibile avere un massimo di 18.000 utenti attivi per settore. Comunque, per mantenere il ritardo accettabile, sarebbe meglio non supportare più di 9.000 utenti attivi.

La Figura 2.30 mostra che quattro settori permettono di supportare una popolazione di 360.000 utenti attivi. Supponendo che un terzo dei clienti sia attivo durante i periodi di picco, una singola torre con quattro antenne potrebbe servire 100.000 persone sparse in un raggio di 5 Km dalla torre.

Questi calcoli sono stati fatti da molti potenziali CLEC, e alcune di queste società hanno concluso che con un modesto investimento in torri di trasmissione sarebbe stato possibile entrare nel mercato della telefonia locale e Internet, offrendo agli utenti velocità di trasmissione paragonabili a quelle delle TV via cavo ma a prezzi minori. LMDS ha comunque qualche difetto; per esempio, le onde millimetriche si propagano in linea retta, perciò non ci deve essere alcun ostacolo tra l'antenna posta sul tetto dell'abitazione e la torre. Inoltre, le foglie assorbono bene queste onde perciò la torre deve essere alta abbastanza da evitare la copertura degli alberi. Infine, ciò che appare ben visibile in dicembre potrebbe non essere altrettanto chiaro in luglio quando gli alberi sono pieni di foglie. Anche la pioggia assorbe queste onde, e gli errori che introduce possono essere compensati solo fino a un certo punto applicando codici di correzione o aumentando la potenza del segnale. È quindi molto probabile che il servizio LMDS sarà utilizzato di più nelle zone dal clima secco, per esempio in Arizona, che a Seattle.

I collegamenti locali wireless probabilmente inizieranno a diffondersi solo dopo che saranno stati definiti standard adeguati; gli standard, infatti, incoraggiano la produzione di apparecchiature e danno ai clienti la possibilità di cambiare CLEC senza dover acquistare nuovi dispositivi. Per realizzare questi obiettivi, IEEE ha chiesto a un comitato chiamato 802.16 di redigere lo standard per LMDS, che è stato pubblicato nel mese di aprile del 2002. IEEE definisce lo standard 802.16 MAN wireless.

IEEE 802.16 è stato progettato per la telefonia digitale, l'accesso a Internet, l'interconnessione di due LAN remote, la trasmissione radio televisiva e altre applicazioni (per ulteriori dettagli consultare il Capitolo 4).

2.5.4 Linee e multiplexing

Le economie di scala giocano un ruolo importante nel sistema della telefonia. Installare e gestire una connessione tra centrali di commutazione basata su una linea a larga banda costa quasi quanto una connessione basata su una linea a banda stretta (il costo deriva soprattutto dagli scavi, non dai materiali utilizzati per il cablaggio). Di conseguenza, le aziende telefoniche hanno sviluppato elaborati schemi per convogliare molte conversazioni lungo un singolo collegamento fisico. Questi schemi di multiplexing possono essere divisi in due categorie di base: **FDM** (*Frequency Division Multiplexing*) e **TDM** (*Time Division Multiplexing*).

In FDM, lo spettro di frequenza è diviso in bande di frequenza e ogni utente ha il possesso esclusivo di parte della banda. In TDM, gli utenti si danno il cambio (in una sorta di girotondo), acquisendo periodicamente il possesso di tutta la banda per un tempo brevissimo.

La radiodiffusione AM fornisce un esempio di entrambi i tipi di multiplexing; lo spettro allocato è circa 1 MHz, più o meno da 500 a 1.500 kHz. Frequenze differenti sono assegnate a diversi canali logici (stazioni), ognuna operante in una parte dello spettro, con una separazione tra canali abbastanza grande da prevenire fenomeni di interferenza. Questo sistema è un esempio di multiplexing a divisione di frequenza. In alcuni paesi le singole stazioni hanno due sottocanali logici, uno per la musica e l'altro per la pubblicità; i due sottocanali si alternano nel tempo sulla stessa frequenza: prima si sente un brano musicale, poi uno spot pubblicitario, quindi ancora un po' di musica e così via. Questo meccanismo funziona come un multiplexing a divisione di tempo.

Il testo che segue esamina il multiplexing a divisione di frequenza, spiega in che modo la tecnica FDM si applica alle fibre ottiche (multiplexing a divisione di lunghezza d'onda), quindi descrive TDM e un sistema avanzato adottato per le fibre (SONET).

Multiplexing a divisione di frequenza

La Figura 2.31 mostra la modalità di multiplexing FDM impiegata per tre canali telefonici a qualità vocale. I filtri limitano l'ampiezza di banda utilizzabile a circa 3.100 Hz per canale; quando si uniscono in multiplexing diversi canali, a ciascun canale sono allocati 4.000 Hz in modo da mantenere gli elementi ben separati. Dopo aver aumentato la fre-

quenza di ogni canale vocale di una quantità diversa, si esegue l'unione facendo attenzione a non sovrapporre alcun canale. Anche se i canali sono separati da intervalli (bande di guardia), c'è sempre una leggera sovrapposizione tra canali adiacenti perché i filtri non hanno bordi netti. A causa di questa sovrapposizione, un forte picco di segnale sul bordo di una canale viene avvertito nel canale adiacente sotto forma di rumore non termico.

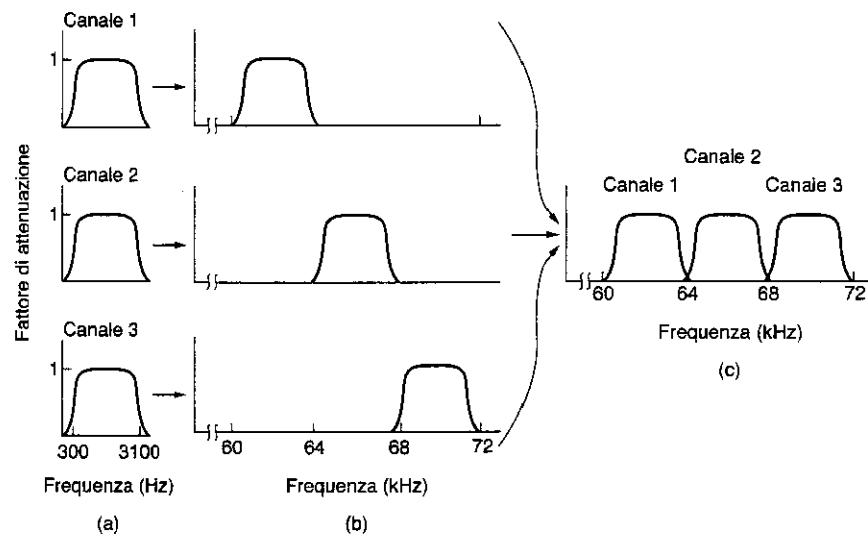


Figura 2.31. Multiplexing a divisione di frequenza. (a) Le ampiezze di banda originali.

(b) Le ampiezze di banda dopo l'aumento di frequenza. (c) Il canale in multiplexing.

Gli schemi FDM utilizzati in tutto il mondo sono parzialmente standardizzati. Uno standard molto comune è quello basato su 12 canali vocali a 4.000 Hz uniti in multiplexing nella banda compresa tra 60 e 108 kHz. Questa unità è chiamata **gruppo**. La banda da 12 kHz a 60 kHz qualche volta è utilizzata per un altro gruppo. Molti operatori offrono ai propri clienti servizi di noleggio linee basati sui gruppi. Cinque gruppi (60 canali vocali) possono essere uniti in multiplexing per creare un **supergruppo**. L'unità successiva si chiama **mastergroup** ed è composta da cinque supergruppi (standard CCITT) o da dieci supergruppi (sistema Bell). Esistono anche altri standard che arrivano fino a 230.000 canali vocali.

Multiplexing a divisione di lunghezza d'onda

Per i canali in fibra ottica si utilizza una variazione del multiplexing a divisione di frequenza chiamata **WDM** (*Wavelength Division Multiplexing*). Il principio di fondo di WDM su fibra è descritto nella Figura 2.32. L'immagine mostra quattro fibre inserite in un combinatore ottico, dove ognuna trasporta energia a diversa lunghezza d'onda. I quattro raggi sono uniti in una singola fibra condivisa prima di essere trasmessi a una destinazione.

zione molto lontana. All'altra estremità del cavo, il raggio è scomposto nuovamente in quattro parti. Ogni fibra in uscita contiene un corto spezzone di core, costruito in modo speciale, che filtra tutto tranne una lunghezza d'onda. I segnali risultanti possono essere instradati verso le loro destinazioni o ricombinati in diversi modi per un ulteriore trasporto in modalità multiplexing.

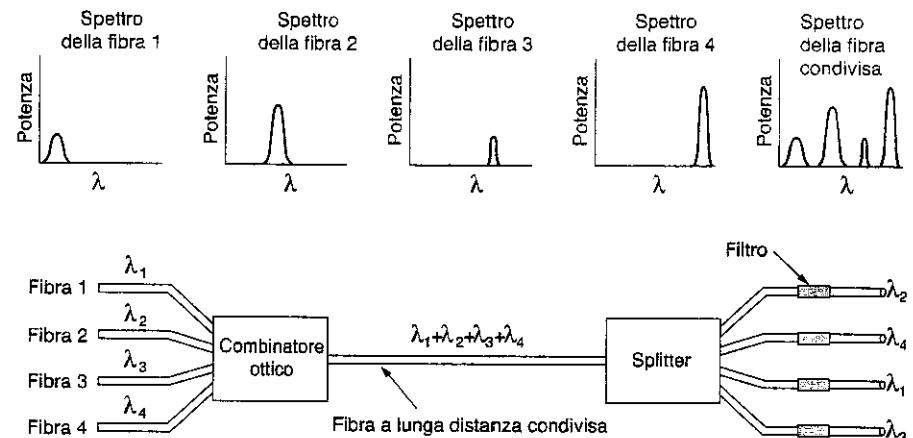


Figura 2.32. Multiplexing a divisione di lunghezza d'onda.

In realtà lo schema non mostra nulla di nuovo, si tratta semplicemente di un multiplexing a divisione di frequenza utilizzato a frequenze molto alte. Si può adoperare il multiplexing sulla fibra a lunga tratta se ogni canale ha il proprio intervallo di frequenza (in questo caso, lunghezza d'onda) e tutti gli intervalli sono distinti. L'unica differenza con la tecnica FDM elettrica è la presenza di un sistema ottico completamente passivo, e perciò altamente affidabile, basato su un reticolo di diffrazione. La tecnologia WDM si è evoluta più velocemente dei computer. WDM è stata inventata all'inizio degli anni '90; i primi sistemi commerciali avevano otto canali da 2,5 Gbps. Nel 1998 il mercato offriva già sistemi a 40 canali da 2,5 Gbps e nel 2001 sono apparsi i primi prodotti basati su 96 canali da 10 Gbps, per un totale di 960 Gbps. Questa ampiezza di banda permette di trasmettere 30 film completi (compresi in MPEG-2) al secondo. Sistemi con 200 canali funzionano già in laboratorio. Quando il numero dei canali è molto elevato e le lunghezze d'onda sono molto vicine tra loro, per esempio a 0,1 nm di distanza, il sistema è definito anche **DWDM** (*Dense WDM*).

WDM è molto popolare anche perché l'energia su una singola fibra è generalmente ampia pochi GHz, poiché è ancora impossibile eseguire conversioni ottico-elettriche in tempi più rapidi. Facendo viaggiare più canali in parallelo su lunghezze d'onda diverse, è possibile aumentare l'ampiezza di banda aggregata in modo lineare rispetto al numero di canali. Poiché l'ampiezza di banda di una singola fibra è di circa 25.000 GHz (Figura 2.6), in teoria c'è spazio per 2.500 canali a 10 GHz anche lavorando ad appena 1 bit/Hz (ed è possibile raggiungere velocità maggiori).

Un altro sviluppo recente riguarda gli amplificatori ottici. In passato ogni 100 Km era necessario suddividere e convertire tutti i canali in un segnale elettrico, amplificare ogni singolo segnale, riconvertirlo in segnale ottico e infine ricombinare tutti i canali. Oggi gli amplificatori ottici possono rigenerare l'intero segnale ogni 1.000 Km, senza eseguire alcuna conversione elettro-ottica. L'esempio della Figura 2.32 mostra un sistema a lunghezza d'onda fissa; i bit ricevuti attraverso la Fibra 1 vengono trasmessi attraverso la Fibra 3, quelli provenienti da Fibra 2 sono trasmessi attraverso Fibra 1 e così via. Si possono comunque costruire anche sistemi WDM che sono commutati. In un dispositivo di questo tipo i filtri in uscita possono essere regolati mediante interferometri Fabry-Perot o Mach-Zehnder. Per maggiori informazioni su WDM e le sue applicazioni alla commutazione di pacchetto Internet vedere (Elmirghani and Mouftah, 2000; Hunter and Andonovic, 2000; Listani et al., 2001).

Multiplexing a divisione di tempo

La tecnologia WDM è meravigliosa, ma il sistema telefonico utilizza ancora un sacco di cavi in rame, perciò è necessario fare un passo indietro. Anche se è utilizzabile su cavi in rame e canali a microonde, FDM richiede collegamenti elettrici analogici e non può essere controllata da computer. Al contrario, TDM può essere gestita completamente da dispositivi elettronici digitali, perciò negli ultimi anni è diventata molto diffusa, ma sfortunatamente può essere utilizzata solo per i dati digitali. Poiché i collegamenti locali producono segnali analogici, è necessario eseguire una conversione analogico-digitale nella centrale locale, dove arrivano i singoli collegamenti locali, prima dell'immissione sulle linee in uscita.

Questo paragrafo spiega in che modo i diversi segnali analogici vocali sono digitalizzati e uniti in una singola linea digitale in uscita. Anche i dati dei computer trasmessi dai modem sono analogici, perciò la descrizione seguente si applica anche a questo tipo di trasmissione. I segnali analogici sono digitalizzati nella centrale locale da un dispositivo chiamato **codec** (*coder-decoder*) che genera una serie di numeri a 8 bit. Il codec elabora 8.000 campioni al secondo (125 μ sec/campione) perché il teorema di Nyquist afferma che tale velocità è sufficiente a catturare tutte le informazioni dall'ampiezza di banda del canale telefonico a 4 kHz. Con una velocità di campionamento inferiore si perderebbero informazioni, mentre a velocità superiori non si guadagnerebbe alcuna informazione aggiuntiva. Questa tecnica è chiamata **PCM** (*Pulse Code Modulation*). PCM costituisce il cuore del sistema telefonico moderno. Di conseguenza, virtualmente tutti gli intervalli di tempo nel sistema telefonico sono multipli di 125 μ sec. Quando la trasmissione digitale assunse i contorni di una tecnologia fattibile, CCITT non riuscì a concordare uno standard internazionale per PCM. Di conseguenza oggi nel mondo sono utilizzati diversi schemi incompatibili tra loro.

Il metodo utilizzato in Nord America e Giappone è la portante T1 descritta nella Figura 2.33 (teoricamente parlando, il formato è chiamato DS1 e la portante è chiamata T1). La portante T1 è composta da 24 canali vocali uniti in multiplexing. Di solito, invece di uti-

lizzare 24 codec separati, i segnali analogici sono campionati in successione e il flusso analogico risultante è passato a un unico codec che genera il flusso digitale in uscita. Ognuno dei 24 canali, a sua volta, aggiunge 8 bit nel flusso in uscita: 7 per i dati e uno per le operazioni di controllo. Ciò significa che in totale si hanno $7 \times 8.000 = 56.000$ bps di dati e $1 \times 8.000 = 8.000$ bps di informazioni di segnalazione per canale.

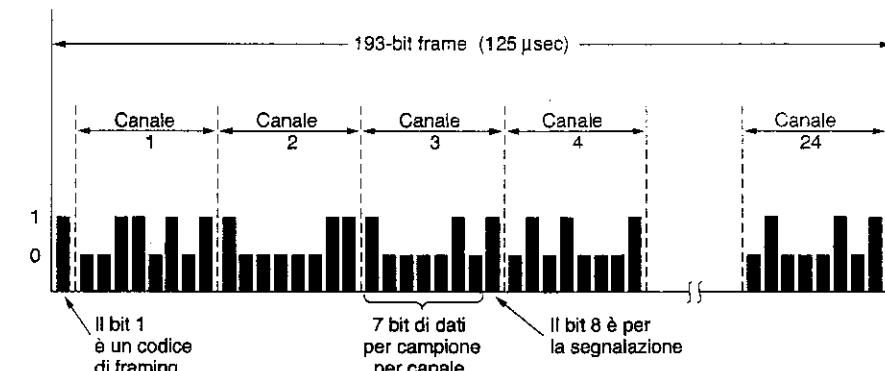


Figura 2.33. La portante T1 (1,544 Mbps).

Un frame è composto da $24 \times 8 = 192$ bit più un bit aggiuntivo utilizzato per la struttura, per un totale di 193 bit ogni 125 μ sec. Questo significa che la velocità dati finale è di circa 1,544 Mbps. Il centonovantatreesimo bit è utilizzato per la sincronizzazione del frame e cambia valore secondo lo schema 0101010101... Normalmente, il ricevitore continua a controllare questo bit per garantire la sincronizzazione; se non è in sincronia, il ricevitore può analizzare questa sequenza per sincronizzarsi, poiché non fa parte di quelle generabili dagli utenti. Gli utenti analogici non possono affatto generare la sequenza di bit perché corrisponde a un'onda sinusoidale a 4.000 Hz che verrebbe filtrata. Al contrario gli utenti digitali potrebbero farlo, ma se il frame perde sincronia la probabilità è minima. Quando un sistema T1 è utilizzato interamente per i dati, solo 23 canali sono assegnati ai dati. Il ventiquattresimo canale è utilizzato per una sequenza speciale di sincronizzazione che consente un recupero più rapido nel caso di perdita del frame.

Quando finalmente raggiunse un accordo, il CCITT si rese conto che 8.000 bps di informazioni di segnale erano troppi, perciò il suo standard 1,544 Mbps adotta 8 bit di dati, e non 7; in pratica, il segnale analogico è quantizzato in 256 livelli discreti, non in 128. Sono fornite due variazioni (incompatibili): nel **sistema di segnalazione a canale comune**, il bit aggiuntivo (inserito alla fine e non all'inizio del frame lungo 193 bit) assume i valori 10101010... nei frame dispari e contiene informazioni di segnale per tutti i canali nei frame pari.

Nel **sistema di segnalazione a canale associato** ogni canale ha il suo sottocanale di segnalazione privato, organizzato in modo da dedicare ai dati cinque frame su sei, e nel

sesto assegnare alla gestione dei segnali uno degli otto bit utente: perciò cinque campioni su sei sono lunghi 8 bit e uno è lungo 7 bit. CCITT ha definito anche portanti PCM a 2,048 Mbps chiamate E1. Questa portante ha 32 campioni di dati a 8 bit impacchettati nel frame elementare di 125 μ sec. Trenta canali sono utilizzati per le informazioni e due per i segnali. Ogni gruppo di quattro frame fornisce 64 bit di segnalazione, metà utilizzati per la segnalazione associata al canale, e metà per la sincronizzazione dei frame oppure riservati ad applicazioni scelte dalle singole nazioni. Al di fuori del Nord America e del Giappone, si utilizza la portante E1 al posto della T1.

Dopo aver digitalizzato il segnale vocale si utilizzano tecniche statistiche per ridurre il numero di bit necessari a ogni canale. Queste tecniche sono adatte non solo per la codifica della voce, ma anche per la digitalizzazione di qualunque altro segnale analogico. Tutti i metodi di compressione si basano sul principio: il segnale cambia in modo relativamente lento rispetto alla frequenza di campionamento, perciò gran parte dell'informazione è ridondante.

Una tecnica, chiamata **differential pulse code modulation**, non trasmette l'ampiezza digitalizzata bensì la differenza tra il valore corrente e quello precedente. Poiché i salti di ± 16 o più su una scala di 128 sono poco probabili, dovrebbero essere sufficienti 5 bit al posto di 7.

Se il segnale ogni tanto compie qualche salto ampio, la logica della codifica può richiedere diversi periodi di campionamento per mettersi in pari. In pratica, l'errore introdotto può essere ignorato.

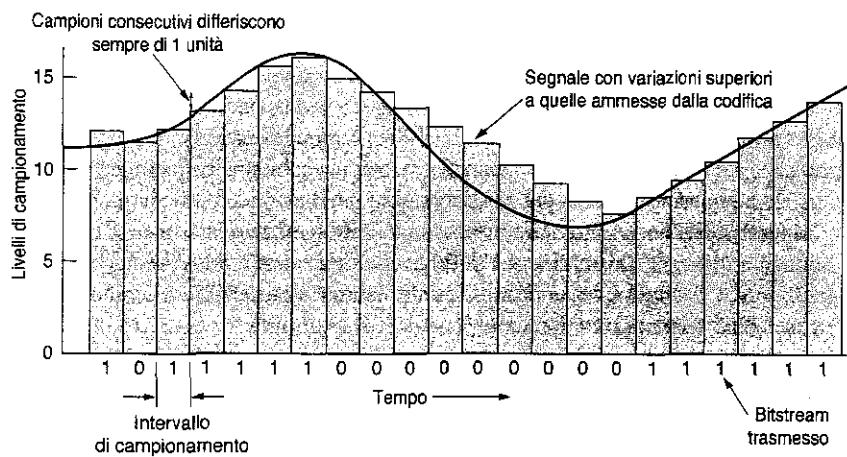


Figura 2.34. Modulazione delta.

Una variante di questo metodo di compressione richiede che ogni valore campionato differisca dal precedente di +1 o -1; sotto queste condizioni, può essere trasmesso un singolo bit che dice se il nuovo campione è maggiore o minore del precedente. La tecnica, chiamata **modulazione delta**, è mostrata nella Figura 2.34. Come nel caso di tutte le tecniche

di compressione che presumono un basso livello di variazione tra campioni consecutivi, la codifica delta può creare problemi se il segnale cambia troppo rapidamente: in questo caso si perdono informazioni.

Per migliorare la tecnologia PCM differenziale si possono estrapolare pochi valori precedenti per predire il valore successivo, e poi codificare la differenza tra il segnale reale e quello previsto. Il trasmettitore e il ricevitore devono ovviamente utilizzare lo stesso algoritmo di previsione. Schemi di questo tipo sono detti **codifiche per ipotesi**. Queste tecniche sono utili perché riducono la dimensione dei numeri da codificare e di conseguenza il numero di bit da trasmettere.

Il multiplexing a divisione di tempo permette di unire diverse portanti T1 in portanti di ordine più elevato. La Figura 2.35 mostra come avviene tutto ciò: a sinistra appaiono quattro canali T1 uniti in un canale T2; la fusione è eseguita bit per bit sui flussi, e non byte per byte tra i 24 canali vocali che compongono un frame T1. Quattro flussi T1 a 1,544 Mbps dovrebbero generare 6,176 Mbps, ma T2 in realtà funziona a 6,312 Mbps. I bit aggiuntivi sono utilizzati per le operazioni di gestione della struttura e quelle di ripristino. T1 e T3 sono molto utilizzate dagli utenti, mentre T2 e T4 sono utilizzate solo all'interno del sistema telefonico e pertanto sono poco note.

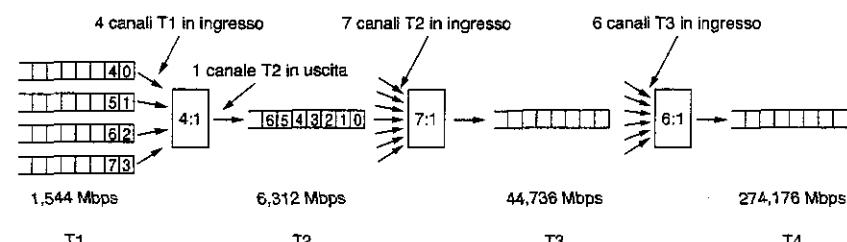


Figura 2.35. Unione multiplexing dei flussi T1 in una portante di ordine maggiore.

Al livello successivo, sette flussi T2 sono uniti bit per bit da formare un flusso T3. Poi sei flussi T3 sono uniti per formare un flusso T4. A ogni passaggio aumenta leggermente il carico di lavoro richiesto per gestire le operazioni di struttura e ripristino, in caso d'interruzione della sincronizzazione tra il trasmettitore e il ricevente.

Così come c'è poca intesa tra gli Stati Uniti e il resto del mondo sulla portante di base, c'è poca intesa anche sul modo in cui la portante deve essere unita in multiplexing per formare portanti di ampiezza di banda maggiore. Lo schema statunitense basato su variazioni di 4, 7 e 6 non ha colpito favorevolmente tutti gli altri paesi, così lo standard CCITT impone un multiplexing di quattro canali per flusso per ogni livello. Anche i dati per la gestione dei frame e le operazioni di ripristino sono diversi. La gerarchia CCITT definisce 32, 128, 512, 2.048 e 8.192 canali che operano a velocità di 2,048, 8,848, 34,304, 139,264 e 565,148 Mbps.

SONET/SDH

Nei primi giorni della fibra, ogni azienda telefonica adottava un sistema TDM ottico proprietario. Dopo lo scioglimento di AT&T avvenuto nel 1984, le aziende telefoniche locali dovevano collegarsi con differenti portanti a lunga distanza, ognuna delle quali adoperava il proprio sistema TDM ottico; divenne ovvia la necessità di definire uno standard. Nel 1985, Bellcore, centro ricerche delle RBOC, iniziò a lavorare su uno standard chiamato **SONET** (*Synchronous Optical NETwork*); successivamente anche CCITT si unì allo sforzo. Alla fine, nel 1989, si ottenne uno standard SONET affiancato da una serie di raccomandazioni CCITT (G.707, G.708, and G.709). Le raccomandazioni CCITT sono chiamate **SDH** (*Synchronous Digital Hierarchy*) ma differiscono da SONET solo in minima parte. Oggi virtualmente tutto il traffico telefonico su lunghe distanze negli Stati Uniti e in quasi tutto il resto del mondo utilizza linee che adoperano SONET sullo strato fisico.

Per maggiori informazioni su SONET, vedere (Bellamy, 2000; Goralski, 2000; e Shepard, 2001).

L'architettura SONET doveva raggiungere quattro obiettivi principali.

1. L'obiettivo più importante è che doveva consentire la comunicazione tra portanti diverse. Per raggiungerlo è stato necessario definire uno standard di segnalazione comune su lunghezza d'onda, temporizzazioni, struttura dei frame e così via.
2. Servivano strumenti per l'unificazione dei sistemi digitali adottati da Stati Uniti, Giappone ed Europa; tutti i sistemi si basavano su canali PCM a 64 kbps, ma ognuno combinava quei canali in modo diverso (e incompatibile).
3. SONET doveva fornire un mezzo per unire in multiplexing diversi canali digitali. Negli anni in cui SONET venne concepita, la portante digitale più veloce utilizzata ampiamente negli Stati Uniti era T3, a 44,736 Mbps; T4 era stata definita ma non era molto utilizzata e nulla era stato definito sopra la velocità T4. Parte della missione di SONET fu portare quella gerarchia a velocità misurabili in Gbps. Era anche necessario trovare un modo standard di unire in multiplexing tanti canali lenti in un unico canale SONET.
4. SONET doveva fornire supporto alle operazioni, all'amministrazione e alla gestione. I precedenti sistemi non erano infatti di grande aiuto.

Una delle prime decisioni fu di rendere SONET un sistema TDM tradizionale, con l'intera ampiezza di banda della fibra dedicata a un canale contenente slot temporali per vari sottocanali. Di per sé, SONET è un sistema sincrono; è controllato da un orologio principale che ha una precisione di circa 1 su 10^9 . I bit sulla linea SONET sono inviati a intervalli precisi, controllati dall'orologio principale. Quando successivamente fu proposto come base di ATM un meccanismo a commutazione di cella che consentiva arrivi irregolari di celle, venne coniata l'etichetta "modalità di trasferimento asincrono" in contrapposizione con la modalità sincrona adottata da SONET. Con SONET, il trasmettitore e il ricevente sono legati a un orologio comune, con ATM invece no.

I frame SONET di base è un blocco di 810 byte emesso ogni 125 μ sec. Poiché SONET è sincrona, i frame sono emessi anche se non ci sono dati utili da trasmettere. Gli 8.000

frame/sec disponibili coincidono perfettamente con la velocità di campionamento dei canali PCM utilizzati in tutti i sistemi telefonici digitali.

I frame SONET a 810 byte si possono immaginare come rettangoli di byte larghi 90 colonne e alti 9 righe. Perciò $8 \times 810 = 6.480$ bit sono trasmessi 8.000 volte al secondo, per una velocità totale di circa 51,84 Mbps. Questo è il canale SONET di base, chiamato anche **STS-1** (*Synchronous Transport Signal-1*). Tutte le linee SONET sono multipli di STS-1.

Le prime tre colonne di ogni frame sono riservate alle informazioni di gestione del sistema (Figura 2.36); le prime tre righe contengono il **codice di controllo della sezione**; le successive sei contengono il **codice di controllo della linea**. Il codice di sezione è generato e controllato all'inizio e alla fine di ogni sezione mentre il codice di linea è generato e controllato all'inizio e alla fine di ogni linea.

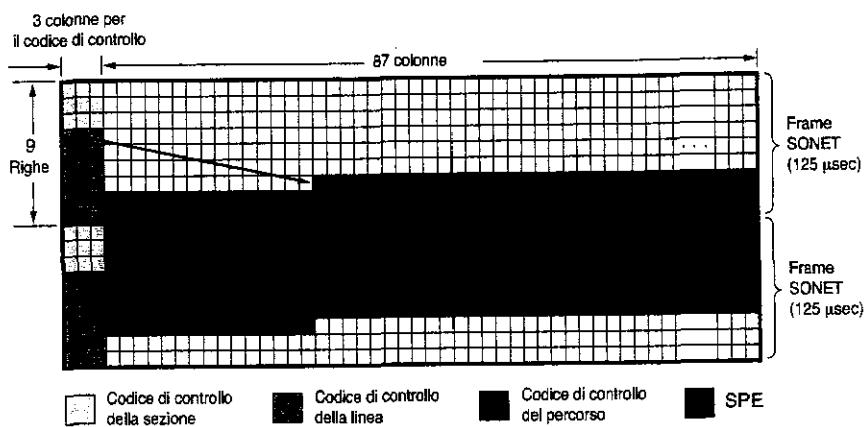


Figura 2.36. Due frame SONET.

Un trasmettitore SONET invia frame a 810 byte addossati, senza creare alcuna interruzione tra frame consecutivi anche quando non ci sono dati utili (in questo caso il dispositivo invia dati fittizi). Poiché i bit formano un flusso continuo, in che modo il ricevitore riesce a individuare il punto iniziale e finale di ogni frame? La risposta è semplice: i primi due byte di ogni frame contengono uno schema fisso e il ricevitore cerca questi elementi per determinare il punto iniziale della struttura. Se trova questo schema nello stesso posto in un gran numero di frame consecutivi, il ricevitore presume di essere in sincronia con il trasmettitore. In teoria un utente potrebbe inserire questo schema in modo regolare nel caricatore utile, ma nella pratica non è possibile eseguire questa operazione a causa dell'unione in multiplexing di più utenti nello stesso frame (e anche per altri motivi).

Le rimanenti 87 colonne contengono $87 \times 9 \times 8 \times 8.000 = 50.112$ Mbps di dati degli utenti. Questi dati, chiamati anche **SPE** (*Synchronous Payload Envelope*), non sempre iniziano alle coordinate riga 1, colonna 4; gli SPE possono iniziare in qualunque punto all'interno del frame. Un puntatore associato al primo byte si trova nella prima riga del codice di controllo della linea. La prima colonna degli SPE rappresenta il codice di controllo del percorso.

e proprietà dei frame SONET rendono il sistema ancora più flessibile: la capacità di inserire gli SPE in qualunque punto all'interno del frame e la possibilità di estendere i dati due frame (Figura 2.36). Per esempio, se riceve un carico utile mentre sta costruendo frame fittizio, la sorgente può inserire i dati nel frame corrente invece di attendere il frame successivo.

Figura 2.37 mostra la gerarchia di multiplexing adottata da SONET; sono state definite velocità da STS-1 a STS-192. La portante ottica corrispondente a STS- n è chiamata OC- n e del tutto identica, a eccezione di un particolare riordinamento di bit utilizzato per la sincronizzazione. I nomi SDH sono diversi e iniziano da OC-3 perché i sistemi basati su ITT non hanno velocità prossime a 51,84 Mbps. La portante OC-9 è presente perché è molto vicina alla velocità di una delle principali linee ad alta velocità utilizzate in Giappone. OC-18 e OC-36 sono utilizzate in Giappone. La velocità dati grezza include anche il codice di controllo; la velocità dati SPE esclude il codice di controllo di sezione e di linea. La velocità dati utente esclude tutto il codice di controllo e tiene conto solo delle colonne di carico utile.

SONET		SDH	Data rate (Mbps)		
Elettrico	Ottico	Ottico	Grazzo	SPE	Utente
STS-1	OC-1		51,84	50,112	49,536
STS-3	OC-3	STM-1	155,52	150,336	148,608
STS-9	OC-9	STM-3	466,56	451,008	445,824
STS-12	OC-12	STM-4	622,08	601,344	594,432
STS-18	OC-18	STM-6	933,12	902,016	891,648
STS-24	OC-24	STM-8	1.244,16	1.202,688	1.188,864
STS-36	OC-36	STM-12	1.866,24	1.804,032	1.783,296
STS-48	OC-48	STM-16	2.488,32	2.405,376	2.377,728
STS-192	OC-192	STM-64	9.953,28	9.621,504	9.510,912

Figura 2.37. Velocità dei multiplex SONET e SDH.

Per curiosità, quando una portante, per esempio OC-3, non è unita in multiplexing e trasporta i dati da una singola sorgente, la lettera *c* (*concatenated*) viene aggiunta alla designazione, così OC-3c indica una portante a 155,52 Mbps composta da tre portanti OC-1 separate ma OC-3c indica un flusso di 155,52 Mbps di dati generati da una singola sorgente. I tre flussi OC-1 dentro il flusso OC-3c si alternano per colonna: prima colonna 1 flusso 1, poi colonna 1 di flusso 2, quindi colonna 1 di flusso 3 seguito da colonna 2 di flusso 1 e così via, fino a riempire 270 colonne per 9 righe.

5.5 Commutazione

Punto di vista dell'ingegnere telefonico medio, il sistema telefonico è diviso in due parti principali: l'impianto esterno (ossia i collegamenti locali e le linee, entrambi collocati fisicamente all'esterno delle centrali di commutazione) e l'impianto interno (composto dai commutatori che si trovano dentro le centrali di commutazione). I paragrafi precedenti hanno descritto l'impianto esterno, ora è giunto il momento di esaminare in dettaglio l'impianto interno.

Attualmente si utilizzano due diverse tecniche di commutazione: la commutazione di circuito e quella di pacchetto. Questo paragrafo presenta entrambe le tecniche ma esamina in dettaglio solo la commutazione di circuito, che è il meccanismo adottato dal sistema telefonico.

Commutazione di circuito

Quando una persona o un computer avvia una telefonata, l'apparecchiatura di commutazione posta dentro il sistema telefonico cerca di creare un percorso fisico completo tra il telefono del chiamante e quello dell'utente chiamato. Questa tecnica, detta **commutazione di circuito**, è rappresentata nella Figura 2.38(a). Ognuno dei sei rettangoli rappresenta una centrale di commutazione dell'operatore. In questo esempio, ogni centrale ha tre linee in ingresso e tre in uscita. Quando una chiamata passa attraverso una centrale di commutazione, viene stabilita (concretamente) una connessione fisica tra la linea di provenienza della chiamata e una delle linee di uscita.

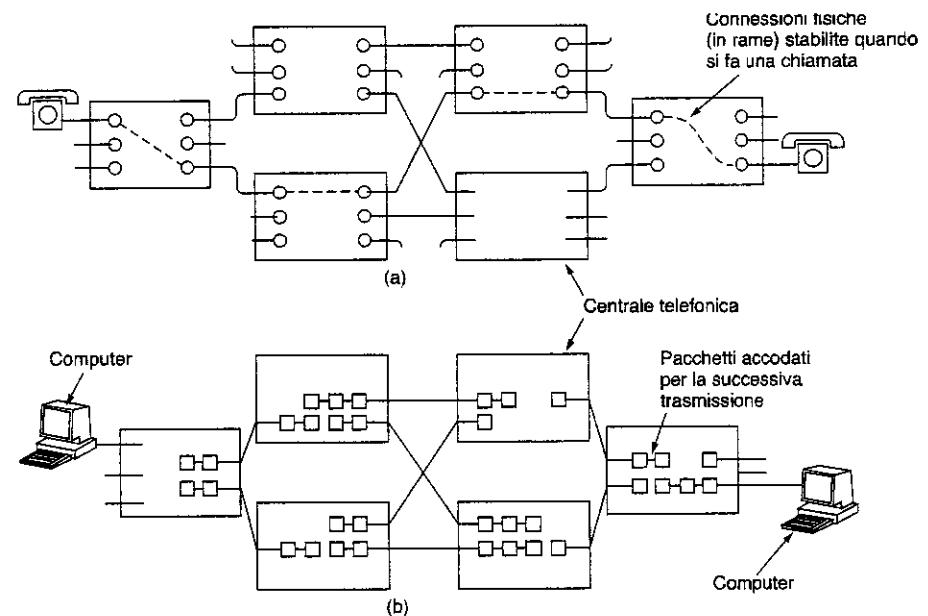


Figura 2.38. (a) Commutazione di circuito. (b) Commutazione di pacchetto.

cordi della telefonia, era l'operatore a creare la connessione inserendo un cavo a delle prese di input e output. In effetti, all'invenzione dei dispositivi di commutazione automatica è associata una storia sorprendente. Questi apparecchi furono inventati annovesimo secolo da un impresario di pompe funebri del Missouri che si chiamava Almon B. Strowger. Poco dopo l'invenzione del telefono, quando moriva qualcuno, i defunti chiamavano l'operatore cittadino e chiedevano di essere collegati con il signor Strowger, in città esistevano imprese di pompe funebri e la moglie dell'altro impresario era l'operatrice telefonica di quella città. Le possibilità per il povero signor Strowger a questo punto erano due: inventare un dispositivo di commutazione automatica per telefoni o andare sul mercato. Egli scelse la prima opzione. Per quasi 100 anni, l'apparecchiatura e commutazione di circuito utilizzata in tutto il mondo è stata chiamata dispositivo Strowger.

Il diagramma mostrato nella Figura 2.39(a) è molto semplificato, certo, perché parte del percorso tra i due telefoni può essere costituito da collegamenti su microonde o fibra ottica i quali viaggiano migliaia di chiamate unite in multiplexing. Ciò nonostante, l'ipotesi è corretta: l'impostazione della chiamata crea tra le due estremità un percorso che perdura fino al termine della chiamata.

Un'altra alternativa alla commutazione di circuito è la commutazione di pacchetto mostrata nella Figura 2.38(b). Con questa tecnologia, singoli pacchetti sono trasmessi all'occorrenza, e ogni percorso dedicato viene impostato in anticipo. Ogni pacchetto trova da solo una via verso la destinazione.

Una proprietà importante della commutazione di circuito è legata alla necessità di configurare un percorso da un punto all'altro *prima* di iniziare a trasmettere i dati. Tra la fine dell'impostazione del numero e l'inizio dello squillo possono tranquillamente trascorrere diversi secondi, di più se la chiamata è su lunghe distanze o internazionale. Durante questo tempo, il sistema telefonico cerca di definire il percorso della connessione come mostrato nella Figura 2.39(a). È bene notare che, prima che la trasmissione dei dati possa iniziare, il segnale della richiesta di chiamata deve propagarsi fino alla destinazione ed essere conosciuto. Per molte applicazioni informatiche (come le verifiche delle carte di credito), tempi di configurazione non sono accettabili.

Per evitare di creare un percorso dedicato tra chiamante e chiamato, una volta completata la connessione l'unico ritardo per i dati è quello dovuto al tempo di propagazione del segnale elettromagnetico, valore pari a circa 5 msec per 1.000 Km. Per lo stesso motivo non corre il rischio di creare una congestione: una volta che la chiamata è stata messa in circuito, non è più possibile ricevere il segnale di occupato. Logicamente, il segnale di occupato può essere ricevuto prima che la connessione sia stabilita, se la commutazione di circuito della linea non sono sufficienti.

Commutazione di messaggio

La Figura 2.39(b) mostra una strategia di commutazione alternativa chiamata **commutazione di messaggio**. Quando si utilizza questa forma di commutazione, nessun percorso

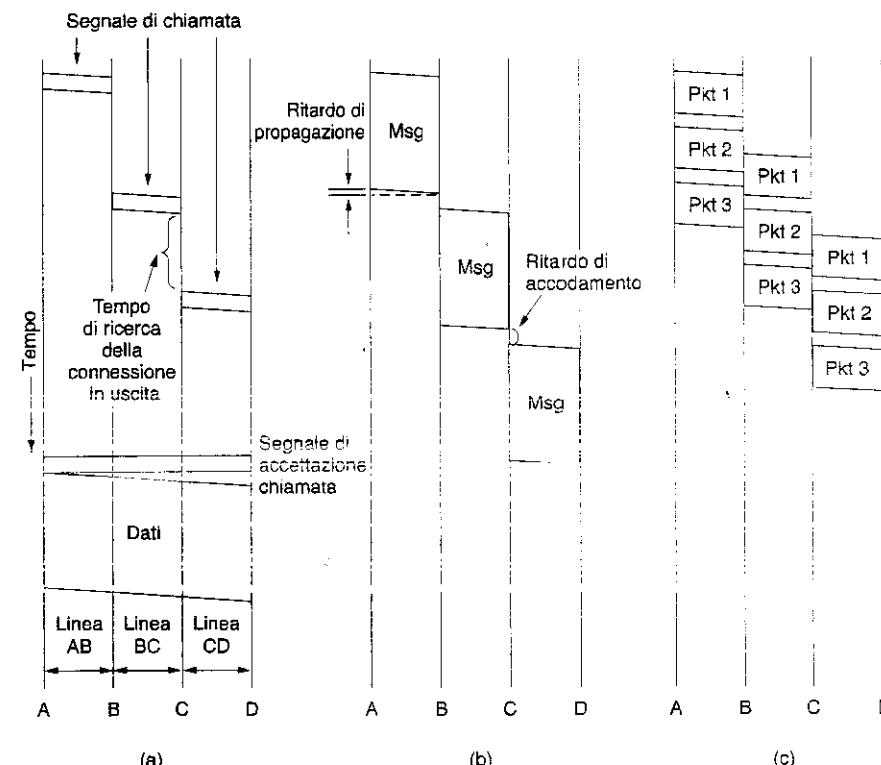


Figura 2.39. Temporizzazione di eventi in (a) commutazione di circuito, (b) commutazione di messaggio e (c) commutazione di pacchetto.

fisico viene stabilito in anticipo tra il trasmettitore e il ricevitore. Quando il trasmettitore ha un blocco di dati da inviare, le informazioni sono passate alla prima centrale di commutazione (per esempio un router), e sono instradate un salto alla volta. Ogni blocco, ricevuto nella sua interezza, è prima esaminato per verificare la presenza di errori e poi ritrasmesso.

Una rete che adotta questa tecnica è definita rete **store and forward** (archivia e inoltra), come spiegato nel Capitolo 1. I primi sistemi di telecomunicazioni elettromeccanici hanno utilizzato la commutazione di messaggio per trasmettere i telegrammi. Il messaggio veniva battuto su un nastro di carta (non in linea) nell'ufficio di trasmissione e poi letto e trasmesso attraverso una linea di comunicazione all'ufficio successivo, dove veniva stampato su carta; a questo punto, un operatore strappava il nastro e lo leggeva usando il dispositivo di lettura associato alla specifica linea in uscita impiegata. Questo tipo di centrale di commutazione era chiamata anche **centrale a nastro strappato**. I nastri di carta sono scomparsi da diverso tempo e la commutazione di messaggio si usa poco.

Commutazione di pacchetto

Con la commutazione di messaggio il blocco dati può avere qualunque dimensione, questo significa che i router (in un sistema moderno) dovrebbero avere dischi capaci di memorizzare temporaneamente i blocchi più lunghi. Inoltre, un singolo blocco di dati potrebbe tenere impegnata una linea da router a router per diversi minuti, rendendo la commutazione di messaggio inutile per il traffico interattivo. Per risolvere questi problemi è stata inventata la **commutazione di pacchetto**. Le reti a commutazione di pacchetto impongono un limite superiore alla dimensione del blocco dati, consentendo ai pacchetti di essere temporaneamente memorizzati nella memoria principale del router invece che su disco. Poiché si assicurano che nessun utente monopolizzi la linea di trasmissione per troppo tempo (miliisecondi), le reti a commutazione di pacchetto riescono a gestire benissimo il traffico interattivo. Un ulteriore vantaggio della commutazione di pacchetto è mostrato nella Figura 2.39(c): il primo pacchetto di un messaggio diviso in più pacchetti può essere inviato prima che il secondo sia completamente arrivato, riducendo il ritardo e migliorando la capacità di trasporto. Per questi motivi, le reti di computer di solito utilizzano la commutazione di pacchetto, ogni tanto la commutazione di circuito, e mai la commutazione di messaggio.

La commutazione di circuito e quella di pacchetto sono molto diverse. Prima di tutto, la commutazione di circuito richiede la configurazione di un circuito da punto a punto prima dell'inizio della comunicazione. La commutazione di pacchetto non richiede alcuna preimpostazione; il primo pacchetto può essere inviato non appena è disponibile. La configurazione della connessione usata nella tecnica a commutazione di circuito permette di riservare l'ampiezza di banda per tutto il percorso che unisce il trasmettitore e il ricevitore. Tutti i pacchetti seguono questo percorso, quindi i dati arrivano già nell'ordine corretto. Nel caso della commutazione di pacchetto non c'è alcun percorso predefinito, perciò diversi pacchetti possono seguire percorsi differenti che dipendono dalle condizioni della rete al momento della trasmissione; inoltre, i dati possono arrivare non in ordine. La commutazione di pacchetto è più resistente agli errori della commutazione di circuito, anzi è stata inventata proprio per questo motivo. Se un commutatore si blocca, tutti i circuiti che lo utilizzano vengono chiusi e nessun dato può essere inviato; nel caso della commutazione di pacchetto, i pacchetti possono essere instradati aggirando i commutatori bloccati.

La preimpostazione di un percorso offre anche la possibilità di riservare in anticipo ampiezza di banda. Se si riserva parte dell'ampiezza di banda, all'arrivo di un pacchetto i dati possono essere inviati immediatamente attraverso l'ampiezza di banda riservata. La commutazione di pacchetto non riserva alcuna ampiezza di banda, perciò ogni pacchetto deve aspettare il proprio turno. Quando si riserva l'ampiezza di banda in anticipo, i pacchetti non possono sovraccaricare la linea (a meno che non si presentino più pacchetti di quelli attesi). D'altra parte, il tentativo di stabilire un circuito potrebbe non andare a buon fine se la linea è sovraccarica; così il sovraccarico si presenta in momenti diversi con la commutazione di circuito (durante l'impostazione iniziale della connessione) e con la commutazione di pacchetto (durante la trasmissione dei pacchetti). Se un circuito è stato riservato per un particolare utente e non c'è traffico da trasmettere, l'ampiezza di banda di quel circuito è sprecata in quanto non può essere utilizzata per gestire altro traffico. La commutazione di pacchetto non sposta l'ampiezza di banda e perciò è più efficiente dal punto di vista generale

del sistema. Comprendere questo compromesso è cruciale per capire la differenza tra la commutazione di circuito e quella di pacchetto. Una soluzione garantisce il servizio ma spreca risorse, l'altra non garantisce il servizio ma non spreca le risorse.

La commutazione di pacchetto utilizza la trasmissione store and forward. Il pacchetto è accumulato nella memoria del router, poi è inviato al router successivo. Con la commutazione di circuito, i bit scorrono attraverso il cavo senza interruzioni. La tecnica store and forward aggiunge ritardo.

Un'altra differenza è che la commutazione di circuito è completamente trasparente: il transmittente e il ricevente possono utilizzare qualunque velocità, formato o metodo di framing; la portante non conosce e non si cura di questi aspetti. Con la commutazione di pacchetto, invece, la portante determina i parametri di base. Una semplice analogia è quella della strada e della rotaia. Sulla strada l'utente determina la dimensione, la velocità e la natura del veicolo; sulla rotaia è la portante che esegue queste operazioni. È questa trasparenza che permette alla voce, ai dati e al fax di coesistere all'interno del sistema telefonico.

L'ultima differenza riguarda l'algoritmo di addebito. Con la commutazione di circuito, storicamente l'addebito dipende da tempo e distanza; per i telefoni cellulari la distanza di solito non gioca un ruolo importante (tranne che per le chiamate internazionali), mentre il tempo gioca solo un ruolo minore.

Con la commutazione di pacchetto, il tempo di connessione non è un problema, ma il volume del traffico qualche volta lo è. Di solito gli ISP addebitano alla maggior parte degli utenti domestici una tariffa netta mensile, che richiede meno lavoro e viene capita meglio dai clienti, ma le portanti dorsali fanno pagare le reti regionali in base al volume del loro traffico. La Figura 2.40 riepiloga le differenze: commutazione di circuito e commutazione di pacchetto hanno un'importanza tale che le riprenderemo presto in considerazione esaminando nel dettaglio le tecnologie che usano.

Caratteristica	Commutazione di circuito	Commutazione di pacchetto
Instaurazione chiamata	Richiesta	Non richiesta
Percorso fisico dedicato	Sì	No
Ogni pacchetto segue la stessa strada	Sì	No
I pacchetti arrivano in ordine	Sì	No
Il guasto dello switch è fatale	Sì	No
Banda disponibile	Fissa	Dinamica
Istante di congestione	All'instaurazione della connessione	A ogni pacchetto
Banda eventualmente sprecata	Sì	No
Trasmissione store and forward	No	Sì
Trasparenza	Sì	No
Tariffa	A minuto	A pacchetto

Figura 2.40. Confronto tra reti a commutazione di circuito e a commutazione di pacchetto.

2.6 Il sistema telefonico mobile

Il sistema telefonico tradizionale, anche aggiornato con fibre multi-gigabit, ancora non riesce a soddisfare un crescente gruppo di utenti: quelli mobili. Le persone oggi vogliono telefonare in aeroplano, in auto, in piscina o mentre corrono nel parco. Entro pochi anni, dagli stessi luoghi vorranno anche inviare messaggi di posta elettronica ed esplorare il Web. Di conseguenza, la telefonia wireless ispira un grandissimo interesse. I paragrafi seguenti esaminano l'argomento in dettaglio.

Esistono due tipi di telefoni wireless: cordless e cellulari (o telefoni mobili). I **cordless** sono dispositivi domestici composti da una base e da un microtelefono; poiché non sono mai utilizzati per la trasmissione di rete, questi apparecchi non verranno studiati. Sono molto più interessanti i sistemi mobili, utilizzati per trasmettere dati e voce attraverso grandi distanze.

Esistono tre generazioni distinte di **telefoni cellulari**, ognuna caratterizzata da una diversa tecnologia:

1. voce analogica
2. voce digitale
3. voce e dati digitali (Internet, posta elettronica e così via).

La maggior parte della discussione si occupa della tecnologia adottata da questi sistemi, comunque è interessante notare l'impatto che possono avere le decisioni politiche e commerciali. Il primo sistema mobile è stato ideato negli Stati Uniti da AT&T e affidato in mandato per l'intero paese dalla FCC. Di conseguenza tutti gli Stati Uniti avevano un singolo sistema (analogico) e un telefono mobile acquistato in California funzionava anche a New York. Al contrario, quando il sistema mobile arrivò in Europa, ogni paese ideò un suo sistema e il risultato fu un fiasco. L'Europa imparò da questo errore: quando apparve la tecnologia digitale, le PTT concordarono un singolo sistema standardizzato (GSM), così qualunque telefono mobile europeo funziona in qualunque paese europeo. Nello stesso periodo il governo degli Stati Uniti, dopo aver deciso che si sarebbe occupato degli standard, lasciò il controllo del digitale al mercato. Questa decisione fece sì che i produttori di apparecchiature producessero diversi tipi di telefoni cellulari. Di conseguenza ora gli Stati Uniti utilizzano due sistemi digitali mobili principali e un sistema minore, incompatibili tra loro.

Nonostante un ritardo iniziale rispetto agli Stati Uniti, oggi l'Europa può vantare un numero di cellulari e un utilizzo maggiore di quello statunitense. L'adozione di un unico sistema non è l'unico motivo di questo successo. Europa e Stati Uniti differiscono anche per un altro aspetto: i numeri di telefono. Negli Stati Uniti, i telefoni cellulari sono mischiati ai telefoni fissi, perciò chi chiama non è in grado di sapere se per esempio il numero (212) 134-5678 appartiene a un telefono fisso (telefonata economica o addirittura gratuita) oppure a un cellulare (telefonata costosa). Per non far innervosire gli utenti, le aziende telefoniche hanno deciso di far pagare ai possessori di cellulari le telefonate in arrivo. Di

conseguenza, molte persone esitano ad acquistare un cellulare per paura di dover pagare bollette salate dovute alle telefonate ricevute.

In Europa i telefoni cellulari usano speciali prefissi facilmente riconoscibili, di conseguenza è applicata la regola classica che addebita il costo della chiamata alla persona che chiama (solo nelle chiamate internazionali il costo viene diviso tra il chiamante e il chiamato).

Un terzo elemento che ha avuto un grande impatto sull'adozione dei telefoni mobili in Europa è l'utilizzo molto esteso delle schede prepagate (in alcune aree il meccanismo ha raggiunto il 75%). Questi apparecchi possono essere acquistati in molti negozi con le stesse formalità dell'acquisto di una radio: il cliente paga e se ne va. L'apparecchio dispone di una scheda da 20 o 50 euro che può essere ricaricata (usando un codice PIN segreto) non appena si esaurisce.

Di conseguenza, praticamente tutti gli adolescenti e molti bambini europei hanno il telefono cellulare (di solito con scheda prepagata) che consente ai genitori di localizzare i figli senza correre il rischio di ricevere bollette salate. Se il telefono mobile è utilizzato solo occasionalmente, il suo utilizzo è essenzialmente gratuito perché non esiste alcun canone fisso e le telefonate ricevute non hanno alcun costo.

2.6.1 Cellulari della prima generazione: voce analogica

Abbiamo detto abbastanza riguardo gli aspetti politici ed economici dei cellulari. Ora è giunto il momento di esaminare la tecnologia, incominciando dal sistema più vecchio. I radiotelefoni mobili furono utilizzati sporadicamente per la comunicazione marittima e militare durante i primi decenni del ventesimo secolo. Nel 1946, a St. Louis venne creato il primo sistema telefonico per auto. Questo sistema utilizzava un singolo trasmettitore di grandi dimensioni collocato in cima a un alto edificio e aveva un unico canale utilizzato sia per trasmettere sia per ricevere. Per parlare, l'utente doveva premere un pulsante che attivava il trasmettitore e disattivava il ricevitore. Questi sistemi, chiamati anche sistemi **premi e parla**, furono installati in diverse città a partire dalla fine degli anni '50. Radio CB, taxi e auto della polizia nei programmi televisivi spesso utilizzano questa tecnologia. Negli anni '60 venne installato **IMTS** (*Improved Mobile Telephone System*); anche questo sistema utilizza un trasmettitore ad alta potenza (200 Watt) collocato in cima a una collina, ma le frequenze questa volta erano due: una per trasmettere e una per ricevere, perciò il pulsante che attivava la comunicazione scomparve. Poiché la comunicazione che dal telefono mobile era diretta verso l'esterno passava attraverso un canale diverso da quello utilizzato per le chiamate in arrivo, gli utenti non potevano ascoltarsi (a differenza del sistema premi e parla utilizzato nei taxi).

IMTS supportava 23 canali distribuiti tra 150 MHz e 450 MHz. A causa del limitato numero di canali, gli utenti spesso dovevano attendere per molto tempo prima di ottenere il segnale di libero. Inoltre, a causa della grande potenza dei trasmettitori, sistemi adiacenti dovevano trovarsi a centinaia di chilometri di distanza per evitare interferenze. Alla fine, la capacità limitata rese il sistema impraticabile.

Sistema telefonico mobile avanzato

Tutto questo cambiò con **AMPS** (*Advanced Mobile Phone System*), sistema inventato da Bell Labs e installato per la prima volta negli Stati Uniti nel 1982. AMPS fu usato anche in Inghilterra, dove venne chiamato TACS, e in Giappone, dove fu chiamato MCS-L1. Sebbene non sia più lo stato dell'arte, è utile esaminarlo in dettaglio perché molte delle sue proprietà fondamentali sono state ereditate direttamente dal suo successore digitale, D-AMPS, per mantenere una compatibilità verso il basso.

In tutti i sistemi telefonici mobili, un'area geografica è divisa in **celle**, ecco perché i dispositivi sono chiamati anche telefoni cellulari. In AMPS, le celle sono grandi 10-20 Km; nei sistemi digitali le celle sono più piccole. Ogni cella utilizza un insieme di frequenze non utilizzate dalle celle vicine. L'idea chiave che dà al sistema cellulare una capacità superiore a quella dei sistemi precedenti è l'utilizzo di celle relativamente piccole e il riutilizzo delle frequenze di trasmissione delle celle vicine (ma non adiacenti). Mentre un sistema IMTS che si estende per 100 Km può avere una chiamata per ogni frequenza, un sistema AMPS può avere nella stessa area 10 celle di 10 Km e gestire da 10 a 15 chiamate per ogni frequenza, in celle distanti.

L'architettura cellulare aumenta pertanto la capacità del sistema di almeno un ordine di grandezza, e anche di più se le celle rimpiccioliscono. Inoltre, tanto più le celle sono piccole, tanto minore è la potenza richiesta e tanto più piccoli ed economici sono i trasmettitori e gli apparecchi portatili. I telefoni cellulari emettono 0,6 Watt e i trasmettitori per auto hanno una potenza di 3 Watt, il massimo consentito da FCC.

Il concetto di riutilizzo della frequenza è rappresentato nella Figura 2.41(a). Le celle di solito hanno una forma approssimativamente circolare, ma sono più semplici da modellare come sagomi. Nella Figura 2.41(a) le celle hanno tutte la stessa dimensione e sono organizzate in gruppi di sette; ogni lettera indica un gruppo di frequenze. È bene notare che per ogni insieme di frequenze c'è un'area cuscinetto grande circa 2 celle che non utilizza quella frequenza; questo sistema consente di separare le frequenze e riduce l'interferenza.

In grosso problema è trovare posizioni elevate dove collocare le antenne di base. Questo problema ha costretto alcune società di telecomunicazioni a stringere alleanze con la Chiesa Cattolica Romana in quanto possiede un altissimo numero di antenne potenziali sparse in tutto il mondo, gestite da un unico ente.

In un'area dove il numero di utenti è cresciuto al punto che il sistema è sovraccaricato, la potenza viene ridotta e le celle sovraccaricate sono divise in celle più piccole chiamate **microcelle** per aumentare il riutilizzo delle frequenze, come nella Figura 2.41(b). A volte le aziende telefoniche, in occasione di importanti eventi sportivi, concerti e così via dove i radunano numerose persone, creano microcelle temporanee utilizzando torri portatili collegate ai satelliti. Il problema del dimensionamento della cella è complesso, e viene trattato in (Hac, 1995).

Al centro di ogni cella si trova una stazione base che comunica con tutti i telefoni che si trovano nella cella. La stazione base è composta da un computer e un trasmettitore/ricevitore collegato a un'antenna. In un piccolo sistema, tutte le stazioni base sono collegate a un singolo dispositivo chiamato **MTSO** (*Mobile Telephone Switching Office*) o **MSC** (*Mobile Switching Center*).

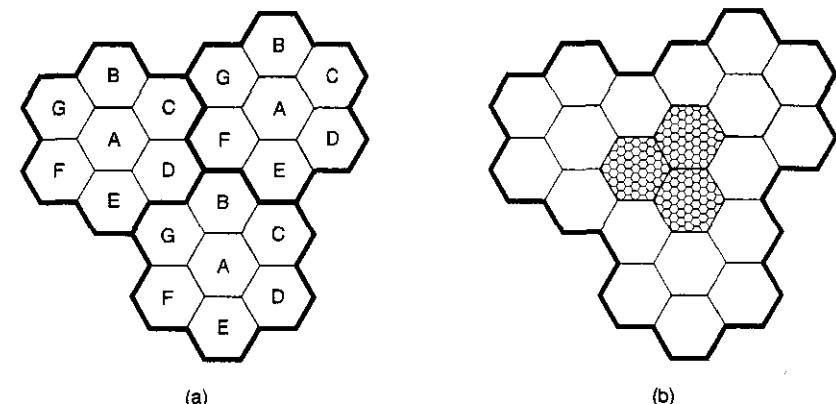


Figura 2.41. (a) Le frequenze non sono utilizzate nelle celle adiacenti.
(b) Per aumentare il numero di utenti si possono utilizzare celle più piccole.

Un sistema più grande può richiedere diversi MTSO, tutti collegati a un MTSO di secondo livello e così via. Gli MTSO sono come le centrali locali del sistema telefonico, e sono collegati ad almeno una centrale del sistema telefonico. Gli MTSO comunicano tra loro, con le stazioni base e con PSTN mediante una rete a commutazione di pacchetto. In ogni istante, ogni telefono cellulare è logicamente in una specifica cella e sotto il controllo della stazione di base di quella cella. Quando un telefono mobile abbandona fisicamente una cella, poiché si accorge che il segnale dell'apparecchio si sta affievolendo, la stazione base di quella cella verifica il livello di potenza del segnale ricevuto dalle stazioni che si trovano nelle celle adiacenti. A questo punto la stazione trasferisce la gestione dell'apparecchio alla cella che riceve il segnale più forte, ossia alla cella in cui ora si trova il telefono.

Il telefono viene informato dell'identità della nuova centrale di controllo e, se durante lo spostamento era in corso una chiamata, l'apparecchio viene forzato a passare su un nuovo canale (perché quello vecchio non è riutilizzato nelle celle adiacenti). Questo processo chiamato **handoff** richiede circa 300 msec. L'assegnazione del canale è eseguita dal MTSO, il centro nevralgico del sistema, mentre le stazioni base sono semplici ripetitori radio. Esistono due tipi di handoff: nel **soft handoff** il telefono è acquisito dalla nuova stazione di base prima di interrompere il segnale precedente; in questo modo non c'è alcuna perdita di continuità ma il telefono deve essere in grado di sintonizzare due frequenze nello stesso momento (quella vecchia e quella nuova). Né i telefoni di prima generazione né quelli di seconda sono in grado di farlo.

Nel caso di **hard handoff**, la vecchia stazione di base rilascia il telefono prima che la nuova lo acquisisca. Se la nuova non è in grado di prendere il controllo del dispositivo (per esempio perché non è disponibile alcuna frequenza), la chiamata viene interrotta bruscamente. Gli utenti tendono a notare questo evento, che però è inevitabile con l'architettura corrente.

nali

istema AMPS utilizza 832 canali full duplex, ognuno composto da una coppia di canali simplex. Esistono 832 canali di trasmissione simplex compresi tra 824 e 849 MHz e 832 canali di ricezione simplex compresi tra 869 e 894 MHz. Ognuno di questi canali simplex ha un intervallo di frequenza di 30 kHz, perciò AMPS utilizza FDM per separare i canali. Nella banda 800 MHz le onde radio sono lunghe circa 40 cm e viaggiano in linea retta; sono assorbite dagli alberi, dalle piante e rimbalzano sul terreno e sugli edifici.

È possibile che un segnale inviato da un telefono mobile raggiunga la stazione di base seguendo un percorso diretto, ma prima di raggiungere la destinazione il segnale potrebbe anche aver rimbalzato sul terreno o su un edificio. Questo può generare un eco e distorcere il segnale (multipath fading). Qualche volta si ascolta una conversazione distante ha rimbalzato diverse volte.

832 canali sono divisi in quattro categorie:

1. controllo (dalla base all'apparecchio), per gestire il sistema
2. paging (dalla base all'apparecchio), per avvisare gli utenti mobili di chiamate in arrivo
3. accesso (bidirezionale), per impostare la chiamata e l'assegnazione del canale
4. dati (bidirezionale), per voce, fax o dati.

Alcuni canali sono riservati al controllo, e sono predeterminati in una PROM presente in ogni telefono. Poiché le stesse frequenze non possono essere riutilizzate nelle celle adiacenti, il numero reale di canali a disposizione della voce è molto più piccolo di 832; in realtà è circa 45.

Esplorazione della chiamata

Un telefono mobile AMPS ha un numero seriale lungo 32 bit e un numero di telefono di 10 cifre registrato nella sua PROM. Il numero di telefono è rappresentato da un prefisso di 10 cifre (10 bit) e dal numero dell'abbonato composto da 7 cifre (24 bit). Quando viene eseguita una chiamata, il telefono esplora l'elenco programmato dei 21 canali di controllo per trovare il canale più potente.

L'apparecchio quindi trasmette in broadcast il numero seriale di 32 bit e il numero di telefono di 34 bit. Come tutte le informazioni di controllo trasmesse in AMPS, questo pacchetto è inviato in forma digitale, più volte e con un codice di correzione degli errori anche se i canali vocali veri e propri sono analogici. Quando riceve la comunicazione, la stazione base aggiorna MTSO che registra la presenza del nuovo cliente; questa MTSO inoltre informa la MTSO principale del cliente, tenendola aggiornata sulla posizione corrente. Infine il normale utilizzo della registrazione del telefono mobile viene eseguita circa una volta ogni 15 minuti.

Per fare una telefonata, l'utente mobile deve accendere l'apparecchio, inserire il numero desiderato e premere il pulsante che trasmette la richiesta. A questo punto il

telefono trasmette il numero da chiamare e la propria identità attraverso il canale di accesso; in caso di collisione, l'operazione viene ripetuta. Quando riceve la richiesta, la stazione base informa l'MTSO. Se il chiamante è un cliente dell'azienda dell'MTSO (o di un suo partner commerciale), l'MTSO cerca un canale libero da assegnare alla chiamata. Se ne trova uno, trasmette il numero del canale attraverso il canale di controllo. Il telefono mobile allora passa automaticamente al canale vocale selezionato e attende che la persona chiamata sollevi il ricevitore.

Le chiamate in arrivo funzionano in modo diverso. Intanto, tutti i telefoni inattivi rimangono in ascolto sul canale di trasferimento per rilevare eventuali messaggi. Quando una chiamata viene trasmessa a un cellulare (da un telefono fisso oppure da un altro apparecchio mobile), l'MTSO principale dell'utente chiamato riceve un pacchetto che ha la funzione di individuare il destinatario. Ora viene trasmesso un altro pacchetto verso la stazione base della cella dove si trova l'utente, che successivamente trasmette in broadcast attraverso il canale di trasferimento un messaggio la cui forma è: "Unità 14, ci sei?". Il telefono chiamato risponde con un "Sì" sul canale di accesso. La base invia quindi un messaggio del tipo: "Unità 14, c'è una chiamata per te sul canale 3". A questo punto, il telefono chiamato passa al canale 3 e inizia a squillare.

2.6.2 Cellulari della seconda generazione: voce digitale

La prima generazione di telefoni cellulari era analogica; la seconda è stata digitale. Così come è mancata una standardizzazione mondiale per la prima generazione, anche durante lo sviluppo della seconda non c'è stata alcuna unificazione. Oggi sono utilizzati quattro sistemi: D-AMPS, GSM, CDMA e PDC. Questo paragrafo esamina i primi tre. PDC è utilizzato solo in Giappone ed è fondamentalmente un D-AMPS modificato per mantenere una compatibilità con il sistema analogico giapponese della prima generazione. Il nome PCS (*Personal Communications Services*) qualche volta è utilizzato nelle riviste di settore per indicare un sistema di seconda generazione. In origine, però, la sigla era associata ai telefoni mobili che utilizzavano la banda 1.900 MHz.

D-AMPS (*Digital Advanced Mobile Phone System*)

La seconda generazione di sistemi AMPS si chiama **D-AMPS** ed è totalmente digitale. È descritta dallo standard internazionale IS-54 e dal suo successore IS-136. D-AMPS è stato progettato attentamente per coesistere con AMPS, perciò i cellulari di prima e seconda generazione possono operare contemporaneamente nella stessa cella. In particolare, D-AMPS utilizza gli stessi canali a 30 kHz di AMPS con le stesse frequenze, perciò un canale può essere analogico mentre quelli adiacenti sono digitali. In base alla combinazione di telefoni presenti in una cella, l'MTSO della cella sceglie i canali analogici e quelli digitali; se la combinazione di apparecchi cambia all'interno della cella, l'MTSO può cambiare in modo dinamico i tipi di canali. Assieme all'introduzione del servizio D-AMPS si è resa disponibile una nuova banda di frequenza per gestire il previsto aumento di carico. I canali in trasmissione spaziano nell'intervallo 1.850-1.910 MHz e i corrispondenti canali in ricezione

cupano l'intervallo 1.930-1.990 MHz, ancora una volta in coppie come per AMPS. In questa banda le onde sono lunghe 16 cm, perciò un'antenna standard di $\frac{1}{4}$ d'onda può essere già solo 4 cm, caratteristica che permette di costruire apparecchi più piccoli. Molti cellulari D-AMPS possono utilizzare entrambe le bande a 850 MHz e 1.900 MHz per ottenere un intervallo più ampio di canali disponibili.

un telefono mobile D-AMPS il segnale vocale raccolto dal microfono viene digitalizzato e compresso usando un modello più sofisticato degli schemi modulazione delta e lifica per ipotesi descritti precedentemente. La compressione tiene conto di proprietà tagliate del sistema vocale umano per portare l'ampiezza di banda dai 56 kbps standard della codifica PCM a 8 kbps o anche meno. La compressione è eseguita nel telefono (non nella stazione base né nella centrale locale), da un circuito chiamato **vocoder** (Bellamy, 2000); questa soluzione è stata adottata per ridurre il numero di bit trasmessi attraverso il collegamento. Nel caso della telefonia fissa, la compressione eseguita nell'apparecchio non offre alcun vantaggio poiché ridurre il traffico sul collegamento locale non aumenta la capacità del sistema. Nel caso della telefonia mobile, invece, la digitalizzazione e la compressione eseguite nell'apparecchio offrono un enorme miglioramento, talmente elevato in D-AMPS che tre utenti possono condividere una singola coppia di frequenze usando la tecnica di multiplexing a divisione di tempo. Ogni coppia di frequenze supporta 25 slot/sec di 40 msec ciascuna. Ogni frame è diviso in sei slot temporali di 6,67 msec, come mostrato nella Figura 2.42(a) per la coppia di frequenze più basse.

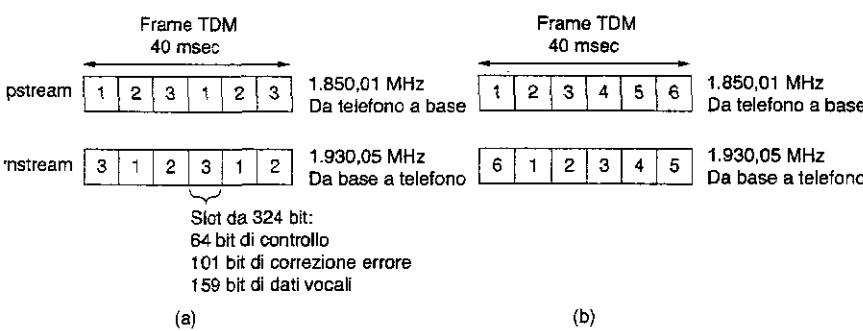


Figura 2.42. (a) Un canale D-AMPS con tre utenti. (b) Un canale D-AMPS con sei utenti.

La figura ogni frame supporta tre utenti che a turno utilizzano i collegamenti in trasmissione e ricezione. Nell'esempio, durante il primo slot rappresentato nella Figura 2(a) l'utente 1 può trasmettere alla stazione base mentre l'utente 3 sta ricevendo informazioni trasmesse dalla stazione base. Ogni slot è lungo 324 bit, 64 dei quali sono utilizzati per tempi di guardia, sincronizzazione e funzioni di controllo; al carico utile perciò angono 260 bit. Di questi bit, 101 sono utilizzati per la correzione degli errori causati da collegamenti rumorosi, perciò alla fine solo 159 bit possono venire assegnati alla voce compressa. Con 50 slot/sec, l'ampiezza di banda disponibile per la voce compressa è appena sotto gli 8 kbps, 1/7 dell'ampiezza di banda dello standard PCM.

Usando algoritmi di compressione migliori si può tenere la voce sotto i 4 kbps, in questo caso sei utenti possono essere stipati nello stesso frame, come mostrato nella Figura 2.42(b). Dal punto di vista dell'operatore far entrare da 3 a 6 utenti D-AMPS nello spettro di un utente AMPS è un'enorme vittoria, e spiega la grande popolarità di PCS. È chiaro che la qualità della voce a 4 kbps non è confrontabile con quella ottenuta a 56 kbps, ma pochi operatori PCS pubblicizzano la qualità del loro suono Hi-Fi. Dovrebbe anche essere evidente che per i dati un canale a 8 kbps non arriva nemmeno alla qualità di un antico modem a 9.600 bps.

La struttura di controllo di D-AMPS è abbastanza complessa. Per riassumere brevemente, gruppi di 16 frame formano un superframe, con alcune informazioni di controllo presenti un numero limitato di volte in ogni superframe. Sono utilizzati sei canali di controllo principali: configurazione di sistema, controllo in tempo reale e non in tempo reale, paging, risposta di accesso e messaggi brevi di testo. Concettualmente il sistema funziona come AMPS: quando viene acceso, il telefono mobile contatta la stazione base per annunciarsi e poi rimane in ascolto sul canale di controllo attendendo le chiamate in arrivo. MTSO, appena acquisito un nuovo dispositivo mobile comunica la sua posizione alla base principale dell'utente, in modo da poter instradare correttamente le chiamate.

Una differenza tra AMPS e D-AMPS riguarda il modo di gestire l'handoff. In AMPS, l'MTSO gestisce l'intera operazione senza il supporto dei dispositivi mobili. In D-AMPS, come si può notare osservando la Figura 2.42, per 1/3 del tempo il cellulare non trasmette né riceve; l'apparecchio utilizza questi intervalli temporali per misurare la qualità della linea. Quando scopre che il segnale sta degradando, il telefono avvisa l'MTSO che può quindi interrompere la connessione; questo permette all'apparecchio mobile di tentare di sintonizzare un segnale più forte trasmesso da un'altra stazione base. Come in AMPS, il cambiamento richiede ancora circa 300 msec. Questa tecnica è chiamata **MAHO** (*Mobile Assisted HandOff*).

Comunicazioni GSM (*Global System for Mobile communications*)

D-AMPS è utilizzato largamente negli Stati Uniti e (in forma modificata) in Giappone. In quasi tutto il resto del mondo, e in scala limitata anche negli Stati Uniti, viene utilizzato un altro sistema chiamato **GSM** (*Global System for Mobile communications*). A prima vista GSM assomiglia a D-AMPS: entrambi sono sistemi cellulari; entrambi adottano il multiplexing a divisione di frequenza, con ogni apparecchio che trasmette su una frequenza e riceve su una frequenza più alta (80 MHz più su per D-AMPS, 55 MHz per GSM); inoltre, in entrambi i sistemi, una singola coppia di frequenze è divisa in slot temporali e condivisa tra più utenti attraverso un meccanismo di multiplexing a divisione di tempo. I canali GSM sono però molto più ampi di quelli AMPS (200 kHz contro 30 kHz) e contengono un numero poco più alto di utenti (8 invece di 3), perciò la velocità dati per utente di GSM è superiore a quella di D-AMPS.

Questo paragrafo descrive brevemente le principali caratteristiche di GSM. Ricordiamo che lo standard completo occupa ben 5.000 pagine stampate, di cui gran parte fa riferimento agli aspetti di progettazione del sistema, specialmente all'architettura dei ricevitori utilizzati per gestire la propagazione multipath del segnale e la sincronizzazione di tra-

smettitori e ricevitori. Nessuno di questi argomenti verrà trattato nel presente paragrafo. Ogni banda di frequenza è ampia 200 kHz, come mostrato nella Figura 2.43. Un sistema GSM ha 124 coppie di canali simplex; ognuno è ampio 200 kHz e supporta otto connessioni separate mediante multiplexing a divisione di tempo. A ogni stazione attiva è assegnato uno slot temporale su una coppia di canali. In teoria supporta 992 canali, ma molti non sono disponibili per evitare conflitti di frequenza con le celle adiacenti. Nella Figura 2.43 gli otto intervalli temporali condivisi (*slot*) appartengono tutti alla stessa connessione, quattro per ogni direzione.

Trasmissione e ricezione non avvengono nello stesso intervallo temporale perché le radio GSM non sono in grado di trasmettere e ricevere contemporaneamente, e passare da un'operazione all'altra richiede tempo. Se la stazione mobile assegnata nell'esempio al secondo slot temporale della coppia 890,4/935,4 MHz desidera trasmettere alla stazione base, deve utilizzare i quattro slot inferiori (in grigio) e quelli successivi, inserendo i dati in ogni slot fino al loro esaurimento.

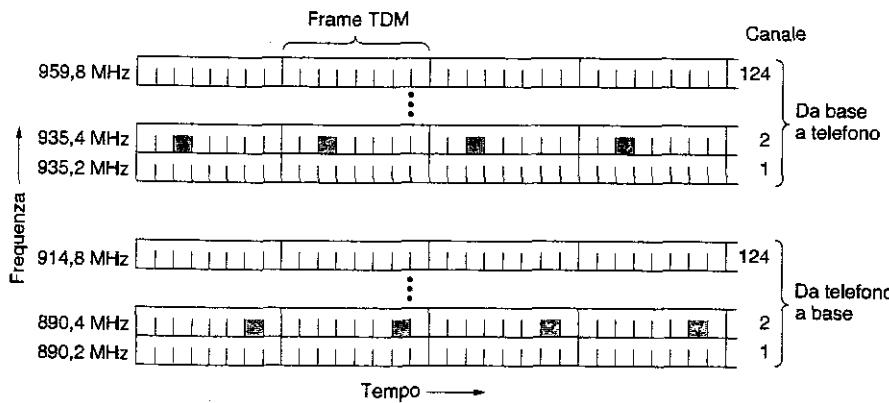


Figura 2.43. GSM usa 124 frequenze di canale, e ogni canale utilizza un sistema TDM a 8 slot.

Gli slot TDM mostrati nella figura 2.43 fanno parte di una complessa gerarchia di frame. Ogni slot TDM ha una struttura specifica e gruppi di slot TDM formano multiframe, anche questi con una struttura specifica. Una versione semplificata di questa gerarchia è mostrata nella Figura 2.44. Come si può notare dall'esempio, ogni slot TDM è composto da un frame con 148 bit di dati che occupa il canale per 577 μ sec (incluso un intervallo di guardia di 30 msec dopo ogni slot). Ogni frame dati inizia e finisce con tre bit a 0 che servono a delimitarlo; la struttura contiene anche due campi *Information* a 57 bit, ognuno dotato di un bit di controllo che indica se il successivo campo *Information* è per la voce oppure per i dati. Tra i campi *Information* si trova un campo *Sync* (sincronizzazione) a 26 bit, utilizzato dal ricevitore per sincronizzarsi con i limiti del frame del trasmettitore. In un frame di dati è trasmesso in 547 μ sec, ma il trasmettitore può spedire un solo frame ogni 4,615 msec poiché sta condividendo il canale con altre sette stazioni. La velocità

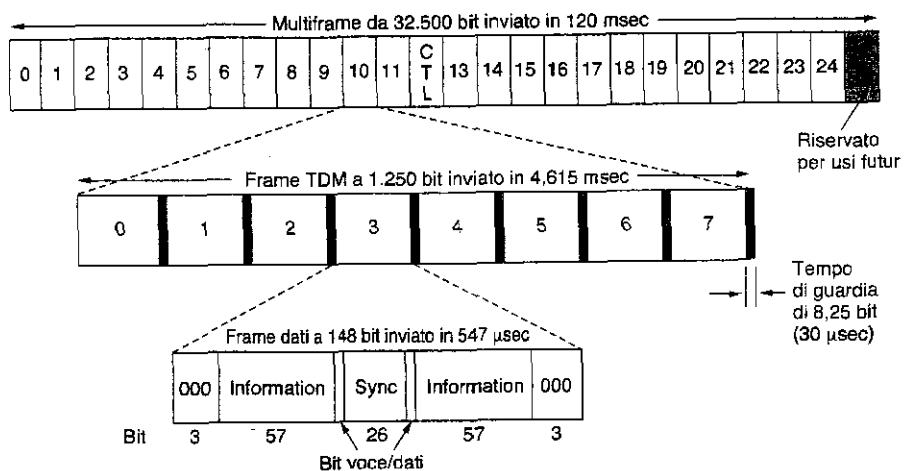


Figura 2.44. Una parte della struttura di frame GSM.

tà approssimativa di ogni canale è di 270.833 bps, condivisa da otto utenti; questo significa che ogni utente può contare su circa 33,854 kbps, più del doppio dei 324 bit 50 volte al secondo (ossia 16,2 kbps) del D-AMPS. Comunque, come nel caso di AMPS, il tempo di elaborazione dati consuma gran parte dell'ampiezza di banda e lascia in definitiva soltanto 24,7 kbps per utente a disposizione del carico utile al lordo della correzione degli errori. Dopo la correzione degli errori, lo spazio a disposizione della voce è di soli 13 kbps. In ogni caso, la qualità della voce è fondamentalmente migliore di quella di D-AMPS, a spese di un utilizzo più alto di banda.

Come si può notare osservando la Figura 2.44, otto frame di dati compongono un frame TDM e 26 frame TDM compongono un multiframe di 120 msec. Esaminando i 26 frame TDM di un multiframe, si nota che lo slot 12 è utilizzato per operazioni di controllo e il 25 è riservato ad applicazioni future, perciò il traffico dell'utente ha a disposizione solo 24 slot.

Oltre al multiframe con 26 slot mostrato nella Figura 2.44 si utilizza anche un multiframe con 51 slot; alcuni sono utilizzati per contenere i canali di controllo necessari a gestire il sistema. Il **canale di controllo broadcast** è un flusso continuo di dati trasmessi dalla stazione base che annuncia identità e stato del canale della stazione base. Tutte le stazioni mobili tengono sotto controllo la potenza del loro segnale per rilevare i cambiamenti di cella.

Il **canale di controllo dedicato** è utilizzato per aggiornare la posizione, registrare il terminale nella rete e configurare la chiamata. Ogni stazione base conserva in un archivio gli identificativi delle stazioni mobili presenti nella cella; le informazioni richieste per gestire questo database sono trasmesse attraverso il canale di controllo dedicato.

In fine, c'è un **canale di controllo comune** che è diviso in tre sottocanali logici. Il primo è il **canale di paging**, utilizzato dalla stazione base per annunciare le chiamate in arrivo; ogni stazione mobile tiene continuamente sotto controllo questo canale per rilevare eventuali chiamate. Il secondo sottocanale si chiama **canale ad accesso casuale** e permette agli utenti di richiedere uno slot sul canale di controllo dedicato. Due richieste che collidono risultano confuse e per questo devono essere ripetute in un secondo momento. La stazione può configurare una chiamata usando uno slot del canale di controllo dedicato; lo slot assegnato è annunciato attraverso il terzo sottocanale, chiamato **canale di assegnazione dell'accesso**.

CDMA (*Code Division Multiple Access*)

D-AMPS e GSM sono sistemi abbastanza convenzionali, che utilizzano FDM e TDM per dividere lo spettro in canali e i canali in slot temporali. C'è un terzo concorrente in gara, CDMA (*Code Division Multiple Access*), che funziona in modo completamente differente. Quando fu proposto per la prima volta, l'industria ebbe più o meno la stessa reazione che Colombo ottenne dalla Regina Isabella quando le propose di voler raggiungere l'India navigando nella direzione sbagliata. Comunque, grazie all'ostinazione di una singola società (Qualcomm), il sistema CDMA è maturato al punto che oggi non soltanto è accettato, ma è anche considerato la migliore soluzione tecnica disponibile e la base per i sistemi mobili di terza generazione. È anche largamente utilizzato nei sistemi mobili di seconda generazione degli Stati Uniti come concorrente diretto di D-AMPS. Per esempio, Sprint CS utilizza CDMA mentre AT&T Wireless usa D-AMPS. CDMA è descritto nello standard internazionale IS-95 e qualche volta è chiamato con quel nome. Viene anche chiamato con il nome del marchio: **cdmaOne**.

CDMA è completamente diverso da AMPS, D-AMPS e GSM. Invece di dividere l'intervallo di frequenze assegnate in poche centinaia di canali a banda stretta, CDMA permette a ogni stazione di trasmettere per tutto il tempo attraverso l'intero spettro di frequenza. Le trasmissioni multiple simultanee sono separate usando la teoria della codifica. CDMA rende meno rigida la premessa secondo la quale i frame entrati in collisione si alterano completamente; CDMA, infatti, presume che segnali sovrapposti si sommino linearmente. Prima di esaminare l'algoritmo è utile considerare la seguente analogia: la sala di attesa di un aeroporto dove molte coppie di persone conversano tra loro. Secondo il modello FDM, tutte le persone sono situate al centro della sala e ogni persona parla solo quando arriva il suo turno; FDM funziona come se le persone fossero divise in gruppi molto istanti e ogni gruppo gestisse la conversazione insieme, ma indipendentemente dalle conversazioni tenute dagli altri gruppi; secondo lo schema CDMA, invece, tutte le persone nella sala parlano contemporaneamente, ogni coppia però comunica in un linguaggio diverso. Per esempio due persone che parlano in francese considerano rumore tutte le voci che non parlano nella loro lingua. La chiave di CDMA è pertanto la capacità di estrarre il segnale desiderato scartando tutto il resto. Di seguito è riportata una descrizione abbastanza semplificata di questo sistema.

In CDMA, ogni tempo bit è suddiviso in m intervalli brevi chiamati **chip**. In genere ci sono 4 o 128 chip per ogni bit, ma per semplicità l'esempio descritto in queste pagine utilizza

solo 8 chip per bit. A ogni stazione è assegnato un codice m -bit univoco chiamato **sequenza di chip**. Per trasmettere un bit 1, una stazione invia la sua sequenza di chip; per trasmettere un bit 0, la stazione invia il complemento a uno della propria sequenza di chip. Non sono ammessi altri schemi, perciò per $m = 8$, se alla stazione A viene assegnata la sequenza di chip 00011011, la stazione invia un bit 1 trasmettendo 00011011 e un bit 0 trasmettendo 11100100.

È possibile incrementare la quantità d'informazioni inviabili passando da b bit/sec a mb chip/sec solo aumentando l'ampiezza di banda di un fattore di m , questo rende CDMA una forma di comunicazione a spettro distribuito (dando per scontato che non ci siano modifiche nelle tecniche di modulazione o codifica). Se abbiamo una banda di 1 MHz a disposizione di 100 stazioni, con FDM ogni stazione avrebbe 10 kHz e potrebbe inviare a 10 kbps (presupponendo 1 bit per Hz); con CDMA, ogni stazione utilizza 1 MHz, perciò la frequenza di chip è pari a 1 megachip al secondo. Con meno di 100 chip per bit, l'ampiezza di banda effettiva per ogni stazione CDMA è superiore a quella di FDM e il problema dell'assegnazione dei canali scompare. Per motivi pedagogici è più utile utilizzare la notazione bipolare: il valore binario 0 è rappresentato da -1 e il valore binario 1 è rappresentato da +1. Le sequenze di chip sono racchiuse tra parentesi, perciò un bit 1 per la stazione A diventa (-1 -1 -1 +1 +1 -1 +1 +1). La Figura 2.45(a) mostra le sequenze di chip binarie assegnate a quattro stazioni; la Figura 2.45(b) mostra le stesse sequenze rappresentate in formato bipolar.

A: 0 0 0 1 1 0 1 1
B: 0 0 1 0 1 1 1 0
C: 0 1 0 1 1 1 0 0
D: 0 1 0 0 0 0 1 0

(a)

A: (-1 -1 -1 +1 +1 -1 +1 +1)
B: (-1 -1 +1 -1 +1 +1 +1 -1)
C: (-1 +1 -1 +1 +1 +1 -1 -1)
D: (-1 +1 -1 -1 -1 +1 -1)

(b)

Sei esempi

--1--	C	$S_1 = (-1 +1 -1 +1 +1 +1 -1 -1)$
-1 1 -	B + C	$S_2 = (-2 \ 0 \ 0 \ 0 +2 +2 \ 0 -2)$
1 0 --	A + B	$S_3 = (\ 0 \ 0 -2 +2 \ 0 -2 \ 0 +2)$
1 0 1 -	A + B + C	$S_4 = (-1 +1 -3 +3 +1 -1 -1 +1)$
1 1 1 1	A + B + C + D	$S_5 = (-4 \ 0 -2 \ 0 +2 \ 0 +2 -2)$
1 1 0 1	A + B + C + D	$S_6 = (-2 -2 \ 0 -2 \ 0 -2 +4 \ 0)$

(c)

$$\begin{aligned}
 S_1 \bullet C &= (1 +1 +1 +1 +1 +1 +1 +1)/8 = 1 \\
 S_2 \bullet C &= (2 +0 +0 +0 +2 +2 +0 +2)/8 = 1 \\
 S_3 \bullet C &= (0 +0 +2 +2 +0 -2 +0 -2)/8 = 0 \\
 S_4 \bullet C &= (1 +1 +3 +3 +1 -1 +1 -1)/8 = 1 \\
 S_5 \bullet C &= (4 +0 +2 +0 +2 +0 -2 +2)/8 = 1 \\
 S_6 \bullet C &= (2 -2 +0 -2 +0 -2 -4 +0)/8 = -1
 \end{aligned}$$

(d)

Figura 2.45. (a) Sequenze di chip binarie delle quattro stazioni. (b) Sequenze di chip bipolar. (c) Sei esempi di trasmissione. (d) Ripristino del segnale della stazione C.

Ogni stazione adotta una sequenza di chip univoca. Si indichi con \mathbf{S} l' m -esimo vettore chip della stazione S ed $\bar{\mathbf{S}}$ il suo contrario. Tutte le sequenze sono a due a due **ortogonali**, questo significa che il prodotto interno normalizzato di qualunque coppia di sequenze di chip distinte \mathbf{S} e \mathbf{T} (rappresentato con $\mathbf{S} \cdot \mathbf{T}$) è uguale a 0. Per generare queste sequenze di frammento ortogonali si utilizza un metodo noto come **codici Walsh**. In termini matematici, l'ortogonalità delle sequenze di frammento può essere espressa come segue:

$$\mathbf{S} \cdot \mathbf{T} = \frac{1}{m} \sum_{i=1}^m S_i T_i = 0 \quad (2.4)$$

In altri termini, il numero di coppie identiche è pari al numero di coppie diverse. Questa proprietà ortogonale si dimostrerà cruciale più avanti. Si noti che se $\mathbf{S} \cdot \mathbf{T} = 0$, allora anche $\mathbf{S} \cdot \bar{\mathbf{T}}$ è uguale a 0. Il prodotto interno normalizzato di qualunque sequenza di chip per se stessa è 1:

$$\mathbf{S} \cdot \mathbf{S} = \frac{1}{m} \sum_{i=1}^m S_i S_i = \frac{1}{m} \sum_{i=1}^m S_i^2 = \frac{1}{m} \sum_{i=1}^m (\pm 1)^2 = 1$$

Questo perché ognuno dei termini m nel prodotto interno è 1, perciò la somma è pari a m . Si noti anche che $\mathbf{S} \cdot \bar{\mathbf{S}} = -1$.

Durante ogni tempo bit, una stazione può trasmettere un 1 inviando la sua sequenza di chip, può trasmettere 0 inviando la negazione della sua sequenza di chip oppure può rimanere in silenzio e non trasmettere alcunché. Per il momento, si supponga che tutte le stazioni siano sincronizzate e quindi tutte le sequenze di chip inizino nello stesso istante.

Quando due o più stazioni trasmettono contemporaneamente, i loro segnali bipolarari si sommano linearmente. Per esempio, se in un periodo di chip tre stazioni trasmettono +1 e una stazione trasmette -1, il risultato è +2. Il tutto può essere immaginato come un aumento di tensione: tre stazioni emettono +1 Volt e una stazione emette -1 Volt, per un totale di 2 Volt.

La Figura 2.45(c) mostra sei esempi con una o più stazioni che trasmettono contemporaneamente. Nel primo esempio, C trasmette un bit 1, perciò il risultato finale è la sequenza di chip di C . Nel secondo esempio, sia B sia C trasmettono i bit 1, perciò il risultato finale è la somma delle sequenze di chip bipolarari:

$$(-1 -1 +1 -1 +1 +1 -1) + (-1 +1 -1 +1 +1 +1 -1) = (-2 0 0 0 +2 +2 0 -2)$$

Nel terzo esempio, la stazione A invia un bit 1 e la stazione B invia un bit 0; le altre rimangono in silenzio. Nel quarto esempio, A e C inviano un bit 1 mentre B invia un bit 0. Nel quinto esempio, tutte e quattro le stazioni trasmettono un bit 1. Infine, nel sesto esempio, A , B e D trasmettono un bit 1 mentre C trasmette un bit 0. Si noti che ognuna delle sei sequenze, da S_1 a S_6 , utilizzate nella Figura 2.45(c) rappresenta solo un tempo bit. Per recuperare il flusso di bit di una singola stazione, il ricevitore deve già conoscere la sequenza di chip della stazione; i dati si ottengono calcolando il prodotto interno normalizzato della sequenza di chip ricevuta (la somma lineare di tutte le stazioni trasmettenti) per la sequenza di chip della stazione il cui flusso di bit si desidera recuperare.

La sequenza di chip ricevuta è \mathbf{S} e il ricevitore sta tentando di ascoltare una stazione la cui sequenza di chip è \mathbf{C} , il prodotto interno normalizzato da calcolare è $\mathbf{S} \cdot \mathbf{C}$.

Per capire come funziona questo meccanismo si supponga che due stazioni, A e C , trasmettano entrambe un bit 1 mentre la stazione B trasmette un bit 0. Il ricevitore vede la somma

$$\mathbf{S} = \mathbf{A} + \bar{\mathbf{B}} + \mathbf{C}$$

e calcola

$$\mathbf{S} \cdot \mathbf{C} = (\mathbf{A} + \bar{\mathbf{B}} + \mathbf{C}) \cdot \mathbf{C} = \mathbf{A} \cdot \mathbf{C} + \bar{\mathbf{B}} \cdot \mathbf{C} + \mathbf{C} \cdot \mathbf{C} = 0 + 0 + 1 = 1$$

I primi due termini spariscono perché tutte le coppie di sequenze di chip sono state scelte per essere ortogonali, come dimostrato dall'equazione (2.4). Ora dovrebbe essere chiaro come mai è necessario imporre alle sequenze di chip questa proprietà.

La situazione può essere descritta anche in un altro modo, immaginando che tutte e tre le sequenze di chip arrivino separatamente invece di essere sommate. Il ricevitore, in questo caso, dovrebbe calcolare separatamente il prodotto interno di ogni sequenza e sommare i risultati. A causa della proprietà ortogonale, il risultato di tutti i prodotti interni, con l'eccezione di $\mathbf{C} \cdot \mathbf{C}$, sarà 0. La somma di questi risultati e l'esecuzione del prodotto interno equivale in effetti all'esecuzione dei prodotti interni seguita dalla somma dei risultati.

Per rendere il processo di decodifica più concreto si considerino i sei esempi della Figura 2.45(c) rappresentati nella Figura 2.45(d). Si supponga che il ricevitore desideri estrarre il bit inviato dalla stazione C da ognuna delle sei somme $S_1 \dots S_6$. Il ricevitore calcola il bit sommando a due a due i prodotti dei vettori \mathbf{S} e \mathbf{C} ricevuti, mostrati nella Figura 2.45(b), prendendo poi 1/8 del risultato (poiché in questo caso $m = 8$). Come è stato dimostrato, ogni volta viene decodificato il bit corretto. È proprio come parlare in francese.

In un sistema CDMA ideale, senza rumore, la capacità (ossia il numero di stazioni) può essere resa arbitrariamente grande, proprio come la capacità di un canale di Nyquist senza rumore può essere resa arbitrariamente grande usando un numero crescente di bit per campione. Nella realtà i limiti fisici riducono notevolmente la capacità. Primo, secondo l'ipotesi iniziale, tutti i chip erano sincronizzati; in realtà una sincronizzazione di questo tipo è impossibile. Ciò che si può fare è sincronizzare il trasmettitore e il ricevitore facendo emettere al trasmettitore una sequenza di chip predefinita la cui lunghezza consenta al ricevitore di allacciarsi. A questo punto, tutte le altre trasmissioni (non sincronizzate) saranno viste come rumore di fondo, ma se non sono troppe l'algoritmo di decodifica di base funziona ancora piuttosto bene. Esistono molte teorie sulla sovrapposizione delle sequenze di chip al livello di rumore (Pickholtz et al., 1982). Come ci si potrebbe aspettare, tanto più è lunga la sequenza di chip, tanto maggiore è la probabilità di rilevarla correttamente in presenza di rumore. Per migliorare l'affidabilità, la sequenza di bit può utilizzare un codice di correzione degli errori. Le sequenze di chip non usano mai i codici di correzione degli errori.

Una premessa implicita nella discussione è che i livelli di potenza di tutte le stazioni percepite dal ricevitore siano identici. CDMA è in genere utilizzato per sistemi wireless con una stazione base fissa e molte stazioni mobili situate a distanze variabili dalla base. I livelli di potenza ricevuti dalla stazione base dipendono dalla distanza dei trasmettitori. Un approccio valido è che ogni stazione mobile trasmetta alla stazione base usando una poten-

a pari all'inverso del segnale ricevuto dalla stazione base. In altre parole, una stazione mobile che riceve dalla stazione base un segnale debole utilizza una potenza maggiore di quella utilizzata da una stazione mobile che riceve un segnale forte. La stazione base può anche dare comandi esplicativi alle stazioni mobili in modo da aumentare o diminuire la loro potenza di trasmissione.

Un'altra ipotesi iniziale era che il ricevitore conoscesse il trasmettitore. In linea di principio, se dispone di una sufficiente capacità di calcolo, il ricevitore può ascoltare tutti i trasmettitori in una volta sola ed eseguire in parallelo l'algoritmo di decodifica per ognuno di essi. In realtà questo è più facile da dire che da fare. CDMA ha anche molti altri fattori, non esaminati in questa breve introduzione, che complicano ulteriormente la situazione; nonostante ciò resta uno schema intelligente che sta per essere introdotto rapidamente nel settore della comunicazione mobile wireless. Normalmente opera in una banda di 1,25 MHz (contro i 30 kHz di D-AMPS e i 200 kHz di GSM), ma supporta in quella banda molti più utenti di qualunque altro sistema. In pratica, l'ampiezza di banda a disposizione di ogni utente è almeno equivalente a quella di GSM e spesso è molto più alta.

Gli ingegneri che desiderano conoscere in modo profondo CDMA possono leggere (Lee and Miller, 1998). Un approccio per l'allargamento della banda alternativo, dove l'allargamento è eseguito nel dominio del tempo invece di quello della frequenza, è descritto in Sari et al., 2000). Queste referenze presuppongono un'elevata competenza nel campo dell'ingegneria delle comunicazioni.

6.3 Cellulari della terza generazione: voce e dati digitali

Qual è il futuro della telefonia mobile? L'industria è mossa da più di un fattore. Innanzitutto, il traffico dati supera già quello della voce sulle reti fisse, e sta crescendo esponenzialmente, mentre il traffico vocale è essenzialmente fisso. Molti esperti credono che presto il traffico dati dominerà quello vocale anche sui dispositivi mobili.

E secondo: telefonia, computer e intrattenimento sono diventati interamente digitali e stanno rapidamente convergendo. Molte persone sbavano per un leggero dispositivo portatile che funzioni come telefono, riproduttore CD e DVD, terminale di posta elettronica, interfaccia Web, macchina da gioco, elaboratore testi e così via, collegato wireless a Internet attraverso una connessione a banda larga. Questo apparecchio e le caratteristiche della sua connessione rappresentano la terza generazione di telefoni cellulari. Per ulteriori informazioni, vedere (Huber et al., 2000; e Sarikaya, 2000).

Nel 1992, l'ITU, tentando di essere un po' più specifica riguardo questo sogno, pubblicò un piano di sviluppo chiamato **IMT-2000**; la sigla IMT è l'acronimo di **International Mobile Telecommunications**. Il numero 2000 doveva indicare tre cose: l'anno in cui il servizio avrebbe dovuto avere inizio; la frequenza (espressa in MHz) utilizzata; l'ampiezza di banda (espressa in kHz) che il servizio avrebbe dovuto avere. Nessuno dei tre obiettivi è stato raggiunto.

Nulla è stato implementato entro l'anno 2000. ITU suggerì a tutti i governi di riservare lo spettro a 2 GHz per permettere ai dispositivi di funzionare in tutti i paesi; solo la Cina ha riservato l'ampiezza di banda richiesta. Infine, ci si è resi conto che 2 Mbps non sono al momento attuabili per gli utenti troppo mobili (poiché è difficile gestire rapidamente i

cambiamenti di cella). Più realistica è una velocità di 2 Mbps per gli utenti fermi al chiuso (paragonabile alla velocità di ADSL), 384 kbps per utenti che camminano e 144 kbps per utenti che si spostano in auto. Ciò nonostante, l'intera area della **3G** (terza generazione, come è stata chiamata) è un gran calderone di attività. La terza generazione sarà forse un po' meno di quello che originariamente si sperava che fosse e arriverà un po' più tardi, ma avrà certamente successo.

I servizi base che la rete IMT-2000 avrebbe dovuto fornire ai suoi utenti sono:

1. trasmissione voce ad alta qualità
2. trasmissione messaggi (al posto di e-mail, fax, SMS, chat e così via)
3. applicazioni multimediali (riproduzione di musica, video, film, televisione e così via)
4. accesso Internet (esplorazione del Web, incluse le pagine con audio e video).

Servizi aggiuntivi potrebbero essere videoconferenza, telepresenza, gioco di gruppo ed e-commerce (usare il telefono cellulare per i pagamenti alla cassa del negozio).

In aggiunta, tutti questi servizi dovrebbero essere disponibili in tutto il mondo (con connessione automatica via satellite quando non è disponibile una rete terrestre), instantaneamente (quindi sempre attivi) e con garanzie di qualità.

Per consentire ai produttori di costruire un singolo apparecchio che potesse essere venduto e utilizzato ovunque nel mondo (come i riproduttori di CD e i computer, non come i cellulari e i televisori), l'ITU concepì per IMT-2000 un'unica tecnologia. L'utilizzo di una sola tecnologia avrebbe anche reso più semplice il lavoro degli operatori di rete e avrebbe incoraggiato un numero maggiore di persone a utilizzare i servizi. Le guerre dei formati, come quella tra Betamax e VHS all'inizio dell'era della videoregistrazione, non aiutano gli affari.

Furono fatte diverse proposte, e dopo lungo esame rimasero solo due possibilità. La prima, **W-CDMA** (*Wideband CDMA*), venne proposta da Ericsson. Questo sistema utilizza una modulazione a spettro distribuito a sequenza diretta del tipo descritto precedentemente; opera su una banda a 5 MHz ed è stato progettato per interagire con le reti GSM, sebbene non sia compatibile con questo standard. Una sua caratteristica dovrebbe permettere agli utenti di lasciare le celle W-CDMA ed entrare nelle celle GSM senza perdere le chiamate. Questo sistema è stato spinto dall'Unione Europea, che lo ha battezzato **UMTS** (*Universal Mobile Telecommunications System*).

L'altro contendente era **CDMA2000**, proposto da Qualcomm. Anche questo sistema si basa su un'architettura a spettro distribuito a sequenza diretta; è un'estensione di IS-95 ed è compatibile con questo standard. Utilizza un'ampiezza di banda di 5 MHz, ma non è stato progettato per interagire con GSM e non è in grado di trasferire le chiamate nelle celle GSM (o D-AMPS). Altre differenze tecniche con W-CDMA includono una diversa frequenza di frammento, un diverso tempo di frame, un differente spettro e una diversa tecnica di sincronizzazione.

Se fossero stati messi nella stessa stanza per progettare un'architettura comune, probabilmente gli ingegneri di Ericsson e Qualcomm avrebbero svolto il loro compito. Dopo tutto, il principio alla base di entrambi i sistemi è lo stesso (CDMA in un canale a 5 MHz), e

nessuno sacrificherebbe la propria vita perché preferisce una specifica frequenza di chip. Il vero problema non è ingegneristico ma politico (come sempre); l'Europa voleva un sistema che interagisse con GSM, gli Stati Uniti volevano un sistema che fosse compatibile con uno dei sistemi già in uso in America (IS-95). Entrambe le parti supportavano le loro aziende locali (Ericsson è un'azienda svedese mentre Qualcomm è californiana). Infine, Ericsson e Qualcomm furono coinvolte in numerose cause legali relative ai rispettivi brevetti CDMA.

Le vertenze legali vennero risolte quando nel marzo 1999 Ericsson accettò di acquistare l'infrastruttura di Qualcomm. Le due aziende si accordarono anche su un singolo standard 3G, ma con diverse opzioni incompatibili che appianano sul piano formale le differenze. A dispetto di queste dispute, i dispositivi e i servizi 3G probabilmente riusciranno a partire nei prossimi anni.

Molto è stato scritto sui sistemi 3G, e la maggior parte delle opere li esalta come la più grande invenzione dopo la ruota. Alcuni riferimenti sono (Collins and Smith, 2001; De Vriendt et al., 2002; Harte et al., 2002; Lu, 2002; e Sarikaya, 2000). Ci sono anche dissidenti, che ritengono sbagliata la direzione presa dall'industria (Garber, 2002; e Goodman, 2000).

In attesa della conclusione della lotta, alcuni operatori si sono mossi cautamente nella direzione di 3G proponendo quello che qualche volta viene definito **2.5G** (in effetti sarebbe stato più corretto definirlo 2.1G). Uno di questi sistemi, chiamato **EDGE** (*Enhanced Data rates for GSM Evolution*), non è altro che GSM con più bit per baud. Il guaio è che più bit per baud comportano un maggior numero di errori per baud, perciò EDGE ha nove diversi schemi dedicati alla modulazione e alla correzione degli errori, che differiscono per la quantità di ampiezza di banda sacrificata alla correzione degli errori introdotti dalla velocità più elevata.

In altro schema 2.5G è **GPRS** (*General Packet Radio Service*), una rete a pacchetti costruita sopra D-AMPS e GSM. GPRS permette alle stazioni mobili di inviare e ricevere pacchetti IP in una cella basata su un sistema vocale. Quando GPRS è operativo, alcuni slot temporali su alcune frequenze sono riservati al traffico dei pacchetti. La stazione base può gestire dinamicamente numero e posizione degli slot secondo il rapporto traffico vocale/traffico dati nella cella.

Il slot disponibili sono divisi in canali logici, utilizzati per applicazioni differenti. La stazione base determina l'associazione tra i canali logici e time slot. Un canale logico è usato per scaricare i pacchetti dalla stazione base nella stazione mobile; ogni pacchetto indica il destinatario. Per inviare un pacchetto IP, una stazione mobile richiede uno o più slot inviando una richiesta alla stazione base. Se la richiesta arriva senza danni, la stazione base comunica all'apparecchio mobile la frequenza e gli slot che dovrà utilizzare per trasmettere il pacchetto. Una volta arrivato alla stazione base, il pacchetto è trasferito su Internet attraverso una connessione via cavo.

Oiché GPRS è una sovrastruttura posta sopra un sistema vocale esistente, rappresenta nella migliore delle ipotesi una soluzione temporanea utile fino all'arrivo di G3. Anche se le reti G3 non sono ancora state completamente realizzate, alcuni ricercatori considerano G3 come un affare fatto e di conseguenza di nessun interesse; queste persone stanno già lavorando ai

sistemi 4G (Berezdivin et al., 2002; Guo and Chaskar, 2002; Huang and Zhuang, 2002; Kellerer et al., 2002; e Misra et al., 2002). Alcune delle funzionalità proposte per i sistemi 4G includono un'ampiezza di banda elevata, l'ubiquità (ossia connessione ovunque), integrazione nativa con le reti su cavi (specialmente con quelle IP), gestione dinamica di risorse e spettro, software radio e alta qualità di servizio per le applicazioni multimediali.

D'altra parte, stanno per essere attivati un po' ovunque talmente tanti punti di accesso LAN wireless 802.11 che alcuni pensano che 3G non soltanto non ha ancora raggiunto il traguardo, ma che addirittura è destinato a fallire. Secondo questa visione, le persone rimarranno collegate passando da un punto di accesso 802.11 a un altro. Dire che l'industria è in uno stato di cambiamento continuo significa sminuire il concetto, per capirlo è sufficiente guardare quello che è successo negli ultimi 5 anni.

2.7 Televisione via cavo

I sistemi telefonici fissi e cellulari esaminati in precedenti paragrafi giocano chiaramente un ruolo importante nella comunicazione via rete del futuro. Esiste però una rete fissa alternativa, che ultimamente sta acquisendo maggiore importanza: la televisione via cavo. Molte persone già telefonano e accedono ai servizi Internet adoperando il cavo, e gli operatori di questo settore stanno lavorando attivamente per aumentare la loro fetta di mercato. I paragrafi seguenti descrivono dettagliatamente il sistema di comunicazione di rete basato sulla televisione via cavo, confrontandolo con i sistemi telefonici appena studiati. Per maggiori informazioni vedere (Laubach et al., 2001; Louis, 2002; Ovadia, 2001; e Smith, 2002).

2.7.1 Televisione ad antenna collettiva

La televisione via cavo fu inventata alla fine degli anni '40 per offrire una migliore ricezione agli utenti che vivevano in aree rurali e montane. Il sistema inizialmente consisteva in una grande antenna collocata sulla cima di una collina che captava il segnale televisivo trasmesso attraverso l'aria, un amplificatore chiamato **headend** utilizzato per rinforzare il segnale e un cavo coassiale che distribuiva il segnale nelle case degli utenti (Figura 2.46).

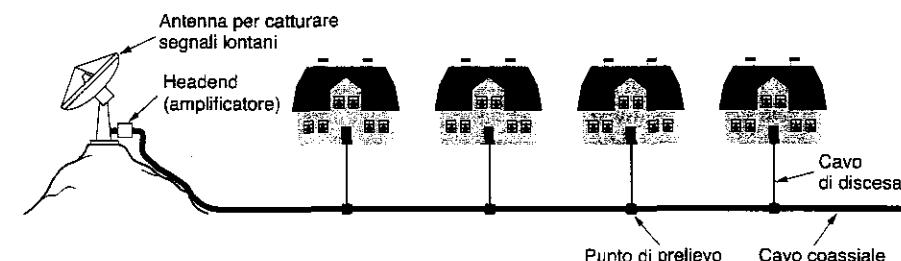


Figura 2.46. Uno dei primi sistemi di televisione via cavo.

Nei primi anni, la televisione via cavo era chiamata **televisione ad antenna collettiva**. Si trattava essenzialmente di un'operazione "fatta in casa": chiunque, con un po' di esperienza elettronica, poteva attivare il servizio nella sua città; gli utenti finali avrebbero contribuito per pagare i costi di attivazione.

Quando il numero di abbonati aumentò, al cavo originale furono collegati nuovi cavi e altri amplificatori vennero aggiunti al sistema. La trasmissione era unidirezionale, dall'headend agli utenti. All'inizio degli anni '70 esistevano migliaia di sistemi indipendenti. Nel 1974, Time Inc. creò un nuovo canale, chiamato Home Box Office, che distribuiva film recenti solo via cavo. Seguirono altri canali trasmessi solo via cavo dedicati alle notizie, allo sport, alla cucina e molti altri temi. Questo sviluppo fece nascere due cambiamenti nell'industria. Primo, grandi società iniziarono ad acquistare i sistemi via cavo esistenti, e a posare nuovi cavi per acquisire nuovi abbonati. Secondo, sorse la necessità di collegare sistemi diversi, spesso situati in città distanti, per poter distribuire i nuovi canali via cavo. Le società iniziarono a stendere cavi tra le loro città per collegarle a un unico sistema, secondo uno schema analogo a quello adottato dall'industria dei telefoni 80 anni prima.

2.7.2 Internet via cavo

Nel corso degli anni, il sistema via cavo crebbe e i cavi tra le varie città furono sostituiti da fibre a elevata ampiezza di banda, proprio come era successo nel sistema telefonico. Un sistema che usa la fibra per tratte più lunghe e i cavi coassiali per le abitazioni degli utenti è chiamato **HFC (Hybrid Fiber Coax)**. I convertitori elettrico ottici che fungono da interfacce tra le parti elettriche e quelle ottiche del sistema sono chiamati **nodi fibra**. Poiché l'ampiezza di banda della fibra è molto più elevata di quella del cavo coassiale, un nodo fibra può alimentare diversi cavi coassiali. La Figura 2.47(a) mostra una parte di un moderno sistema HFC.

Negli ultimi anni molti operatori via cavo hanno deciso di entrare nel settore dell'accesso Internet e spesso anche in quello telefonico, ma le differenze tecniche tra l'impianto via cavo e quello telefonico influenzano ciò che deve essere fatto per raggiungere questi obiettivi. Tanto per cominciare, tutti gli amplificatori unidirezionali usati nel sistema devono essere sostituiti con amplificatori bidirezionali.

Tra il sistema HFC mostrato nella Figura 2.47(a) e il sistema telefonico mostrato nella Figura 2.47(b) esiste un'altra grande differenza, molto più difficile da eliminare. Presso le abitazioni degli utenti un singolo cavo è condiviso tra più case, mentre nel sistema telefonico ogni casa ha il suo collegamento locale privato. Quando è utilizzata per la trasmissione televisiva, questa condivisione non ha importanza: tutti i programmi sono trasmessi via cavo e non fa alcuna differenza se gli spettatori sono 10 oppure 10.000. Se però lo stesso cavo è utilizzato per accedere a Internet, il numero di utenti fa la differenza; se un utente decide di scaricare un file di grandi dimensioni, quella banda viene potenzialmente sottratta a tutti gli altri utenti. Al crescere del numero degli utenti, aumenta la competizione per l'ampiezza di banda. Il sistema telefonico non ha questa proprietà: scaricare un file

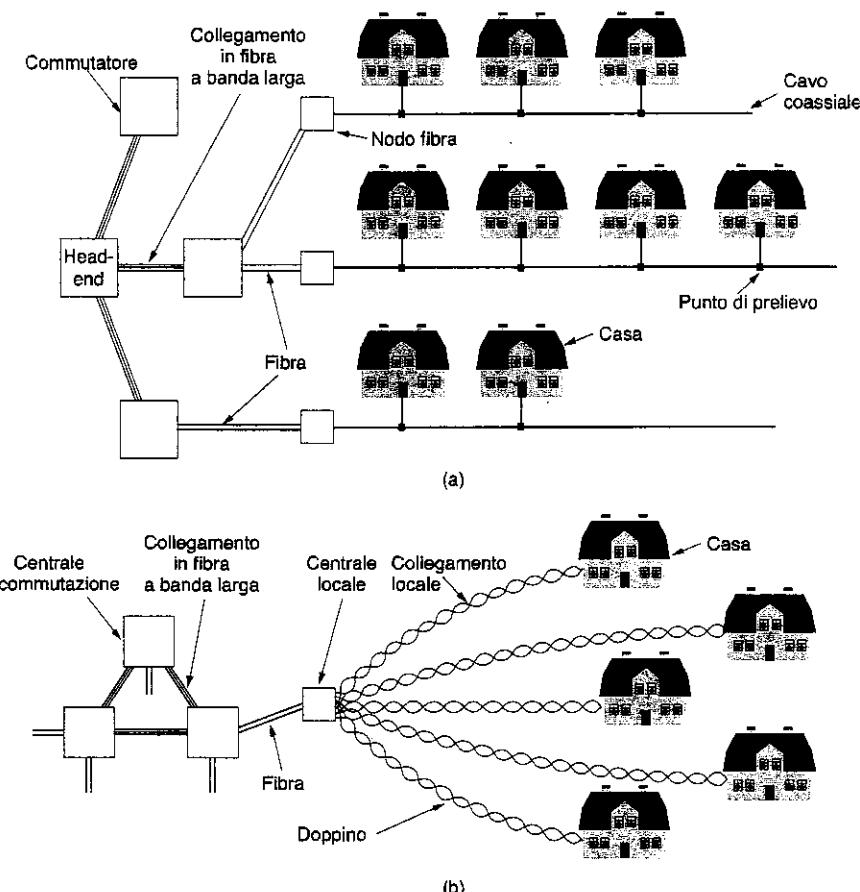


Figura 2.47. (a) Televisione via cavo. (b) Sistema telefonico su rete fissa.

molto grande attraverso una linea ADSL non riduce l'ampiezza di banda del vicino; d'altra parte, l'ampiezza di banda di un cavo coassiale è molto più grande di quella del doppino telefonico.

L'industria della televisione via cavo ha risolto questo problema dividendo i cavi lunghi e collegandoli direttamente ai nodi fibra. L'ampiezza di banda dalla terminazione principale al nodo fibra si può considerare infinita, perciò la quantità di traffico è gestibile se non ci sono troppi abbonati su ogni segmento di cavo. Attualmente ogni cavo collega 500-2.000 abitazioni, ma se il numero degli abbonati al servizio Internet via cavo aumenterà sarà necessario aumentare il numero di ripartizioni e di nodi fibra.

2.7.3 Allocazione dello spettro

Sopprimere tutti i canali televisivi e utilizzare l'infrastruttura basata sui cavi per accedere esclusivamente a Internet avrebbe potuto generare un discreto numero di clienti adirati, perciò le aziende hanno indugiato a farlo. Inoltre, la maggior parte delle città disciplina pesantemente il contenuto trasmesso via cavo, perciò gli operatori non avrebbero potuto farlo neanche se avessero voluto: le società del settore hanno dovuto trovare il modo di far coesistere sullo stesso cavo la televisione e Internet.

I canali televisivi via cavo in Nord America normalmente occupano la banda 54-550 MHz (con l'eccezione dell'intervallo 88-108 MHz dedicato alla radio FM). Questi canali sono ampi 6 MHz, includendo anche le bande di guardia. In Europa il limite inferiore di solito è 65 MHz, e i canali sono ampi 6-8 MHz a causa della più alta risoluzione richiesta dal sistema PAL e SECAM; a parte queste differenze lo schema di allocazione è simile a quello statunitense. La parte bassa della banda non è utilizzata. I cavi moderni possono anche operare bene sopra i 550 MHz, spesso a 750 MHz o più; la soluzione adottata, perciò, fu di introdurre canali upstream nella banda 5-42 MHz (un po' più alta in Europa) e utilizzare le frequenze più alte per la ricezione dei dati. La Figura 2.48 mostra lo spettro finale.

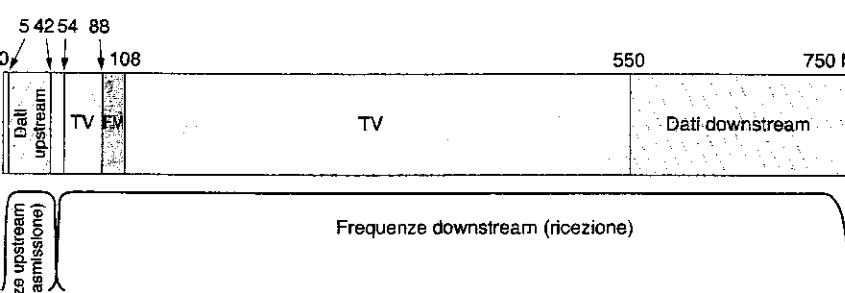


Figura 2.48. Assegnazione delle frequenze in un sistema televisivo via cavo usato per accedere a Internet.

Poiché i segnali televisivi sono tutti in ricezione, è possibile utilizzare amplificatori di trasmissione che funzionano solo nella banda 5-42 MHz e amplificatori di ricezione che funzionano solo sopra i 54 MHz, come mostrato nella figura. Di conseguenza si ottiene una simmetria tra le ampiezze di banda in trasmissione e ricezione perché sopra il segnale televisivo è disponibile più spettro che sotto. D'altra parte, poiché la maggior parte del traffico è diretta verso gli utenti, gli operatori apprezzano questa configurazione. Come è stato spiegato precedentemente, le aziende telefoniche di solito offrono un servizio DSL simmetrico anche se non c'è alcun motivo tecnico per farlo.

I cavi coassiali lunghi non trasmettono i segnali digitali meglio dei collegamenti telefonici locali lunghi, perciò anche in questo caso è richiesta una modulazione analogica. Lo schema tradizionale prende ogni canale di ricezione a 6 o 8 MHz e lo modula con QAM-4; se la qualità del cavo è eccezionalmente buona, con QAM-256. Con un canale a 6 MHz

e QAM-64 si ottengono 36 Mbps, sottraendo il codice di controllo si ottiene un carico utile di circa 27 Mbps. Con QAM-256 il carico utile arriva a 39 Mbps, mentre i valori europei sono 1/3 più grandi.

Per la trasmissione upstream QAM-64 non è sufficiente, a causa dell'eccessivo rumore generato dalle microonde terrestri, dalle radio CB e dalle altre fonti. Si utilizza perciò uno schema più conservativo, QPSK. Questo metodo (mostrato nella Figura 2.25) fornisce 2 bit per baud invece dei 6 o 8 bit per baud che QAM permette di ottenere sui canali di ricezione downstream. L'ampiezza di banda in trasmissione e quella in ricezione è perciò molto più elevata di quella rappresentata nella Figura 2.48. Oltre ad aggiornare gli amplificatori, l'operatore deve aggiornare anche i headend sostituendo l'amplificatore passivo con un sistema di elaborazione digitale intelligente, dotato di un'interfaccia per fibra ad alta ampiezza di banda collegata a un ISP. Spesso anche il nome viene aggiornato, da headend si passa così a **CMTS** (*Cable Modem Termination System*).

2.7.4 Cable modem

Per accedere a Internet è necessario utilizzare un **cable modem** (modem via cavo), dispositivo che ha due interfacce: una verso il computer e l'altra verso la rete via cavo. Nei primi anni dell'era dell'accesso a Internet via cavo ogni operatore adoperava un modem via cavo proprietario che veniva installato da un tecnico dell'azienda, ma presto fu evidente che uno standard aperto avrebbe creato un mercato competitivo e avrebbe fatto diminuire i prezzi dei dispositivi, incoraggiando l'uso del servizio. Inoltre, permettendo ai clienti di acquistare il modem via cavo nei negozi e d'installarlo da soli (come avviene per i modem telefonici V.9x) sarebbe stato possibile eliminare la spesa dell'intervento del tecnico.

I più importanti operatori del settore cooperarono con una società chiamata CableLabs per produrre un modem via cavo standard, per testare i prodotti e renderli conformi. Questo standard, chiamato **DOCSIS** (*Data Over Cable Service Interface Specification*) ha appena iniziato a sostituire i modem proprietari. La versione europea si chiama **EuroDOCSIS**. Non a tutti gli operatori è piaciuta l'idea dello standard, anche perché il noleggio dei modem proprietari rendeva parecchi soldi. Uno standard aperto con decine di produttori che vendono modem via cavo in negozio pone fine a questa pratica redditizia.

L'interfaccia modem/computer è semplice: di solito si usa una porta Ethernet a 10 Mbps, qualche volta si adotta la porta USB. In futuro, l'intero modem potrebbe essere integrato in una piccola scheda installata nel computer, proprio come è accaduto con i modem interni V.9x.

L'altro capo è un po' più complicato. Gran parte dello standard si occupa dell'ingegneria radio, argomento che non rientra tra gli obiettivi del presente volume. L'unica cosa che vale la pena di menzionare in queste pagine è che i modem via cavo, come i modem ADSL, sono sempre attivi. Questi dispositivi creano la connessione quando vengono accesi e mantengono il collegamento fino a quando non vengono spenti, perché gli operatori non addebitano in base al tempo di connessione. Per comprendere meglio il funzionamento dei modem via cavo è utile vedere che cosa accade quando l'utente accende e collega uno di questi apparecchi.

Il modem esamina i canali in ricezione per individuare un pacchetto speciale, che l'headend trasmette periodicamente per fornire ai modem i parametri di sistema necessari ad abilitare la linea. Subito dopo aver trovato questo pacchetto, il modem annuncia la sua presenza attraverso uno dei canali di trasmissione. L'headend risponde assegnando al modem i suoi canali di ricezione e trasmissione. Queste assegnazioni possono essere modificate successivamente, se l'headend ritiene necessario bilanciare il carico.

Il modem determina la sua distanza dalla terminazione principale inviando uno speciale pacchetto e cronometrando il tempo di risposta. Questo processo è chiamato **allineamento**. Determinare la distanza consente al modem di adattare il modo in cui operano i canali in trasmissione e regolare la sincronizzazione. I canali sono divisi in intervalli temporali chiamati **minislot**. Ogni pacchetto trasmesso sui canali upstream deve occupare uno o più minislot consecutivi. L'headend annuncia periodicamente l'inizio di un nuovo ciclo di minislot, ma il via libera non viene ricevuto simultaneamente da tutti i modem, a causa del tempo di propagazione attraverso il cavo. Una volta a conoscenza della distanza dalla terminazione principale, ogni modem può calcolare l'istante in cui inizia il primo minislot. La lunghezza del minislot dipende dalla rete; un carico utile tipico è 8 byte.

Durante l'inizializzazione, l'headend assegna a ogni modem un minislot da utilizzare per richiedere ampiezza di banda in trasmissione. Di solito lo stesso minislot viene assegnato a più modem, creando una contesa. Quando vuole inviare un pacchetto, il computer trasferisce il pacchetto al modem che poi richiede il numero di minislot necessario. Se la richiesta viene accettata, l'headend emette un acknowledge sul canale di ricezione per dire al modem quali minislot sono stati riservati al suo pacchetto. Quindi il pacchetto viene trasmesso, iniziando con il minislot assegnato; utilizzando un campo dell'intestazione si possono richiedere altri pacchetti.

In caso di contesa per il minislot richiesto, non c'è alcun acknowledge e il modem attende semplicemente per un tempo casuale prima di ritentare. Dopo ogni insuccesso consecutivo, il tempo di attesa casuale viene raddoppiato. Per i lettori già un po' esperti di reti, questo algoritmo è semplicemente uno slotted ALOHA con tempo di backoff esponenziale binario. Ethernet non si può usare per le reti TV via cavo poiché le stazioni non possono ascoltare il mezzo di trasmissione. Nel Capitolo 4 si affronterà estesamente l'argomento.

I canali in ricezione sono gestiti in modo diverso. Tanto per cominciare c'è un unico trasmettitore (la stazione headend) perciò non c'è alcuna contesa e i minislot sono inutili, c'è solo un multiplexing statistico a divisione di tempo. Inoltre, il traffico in arrivo di solito è molto più grande di quello in uscita, perciò si utilizza un pacchetto di dimensione fissa grande 204 byte. Parte di questo pacchetto contiene il codice per la correzione degli errori Reed-Solomon e altri codici di controllo; alla fine, il carico utile ha a disposizione 184 byte. Questi numeri sono stati scelti per compatibilità con la televisione digitale che utilizza MPEG-2, perciò i canali televisivi e quelli downstream dedicati ai dati adottano lo stesso formato. La Figura 2.49 mostra le connessioni logiche. Tornando all'inizializzazione, dopo aver completato l'allineamento e aver ottenuto il suo canale di trasmissione, il canale di ricezione e i minislot, il modem è libero di iniziare a inviare pacchetti.

Il primo pacchetto trasmesso è inviato all'ISP per richiedere un indirizzo IP, che viene assegnato dinamicamente usando un protocollo chiamato DHCP. Il pacchetto richiede e

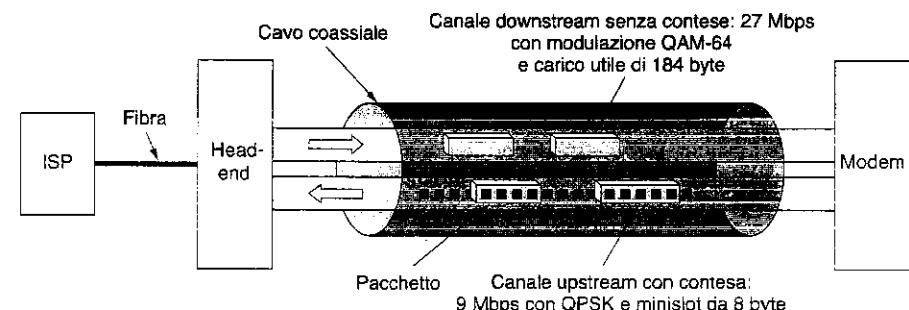


Figura 2.49. Dettagli tipici dei canali di trasmissione e ricezione in Nord America.

ottiene dall'headend anche l'ora del giorno. Il passo successivo coinvolge la sicurezza. Poiché il cavo è un mezzo di trasmissione condiviso, chiunque lo desideri e abbia voglia di farlo può leggere tutto il traffico trasmesso attraverso il cavo. Per impedire agli utenti non autorizzati di intercettare il traffico dei vicini, tutto il traffico è codificato in entrambe le direzioni. Parte della procedura di inizializzazione è dedicata all'impostazione delle chiavi di codifica. A prima vista si potrebbe credere che sia impossibile per due estranei (la terminazione principale e il modem) tentare di stabilire un codice segreto in piena luce del sole con migliaia di persone che osservano. Come verrà spiegato nel Capitolo 8, l'operazione invece è fattibile grazie all'algoritmo di Diffie-Hellman.

Infine, il modem effettua l'accesso fornendo il suo identificatore univoco attraverso il canale protetto. A questo punto l'inizializzazione è terminata e l'utente può collegarsi all'ISP e iniziare a lavorare.

Ci sarebbe molto altro da dire sui modem per le reti via cavo. Alcuni riferimenti sono (Adams and Dulchinos, 2001; Donaldson e Jones, 2001; e Dutta-Roy, 2001).

2.7.5 ADSL o connessione via cavo?

Qual è la soluzione migliore, ADSL o la connessione via cavo? È un po' come chiedere di scegliere il sistema operativo, il linguaggio o la religione migliore. Tutto dipende da ciò che si cerca.

Sia ADSL sia la connessione via cavo utilizzano la fibra ottica nelle dorsali, la differenza è nel tratto finale: l'accesso via cavo utilizza il cavo coassiale, ADSL usa il doppino telefonico. La capacità di trasporto teorica del cavo coassiale è centinaia di volte maggiore di quella del doppino, ma l'intera capacità del cavo non è a disposizione dei dati degli utenti perché una parte dell'ampiezza di banda è dedicata ad applicazioni inutili, come per esempio i programmi televisivi.

In pratica, è difficile generalizzare sulla capacità effettiva. I fornitori di servizi ADSL danno indicazioni precise sull'ampiezza di banda (per esempio, 1 Mbps in ricezione e 256 kbps in trasmissione) e in genere riescono a raggiungere l'80% delle velocità indicate. I fornitori di servizi via cavo non fanno alcuna pubblicità perché la capacità effettiva dipende dal numero di persone attive sul segmento di cavo dell'utente. Qualche volta si otten-

gono velocità migliori di ADSL, altre volte invece no. Quello che dà veramente fastidio, comunque, è l'imprevedibilità del servizio: un minuto di ottimo servizio non garantisce un servizio altrettanto eccellente nel minuto successivo, soprattutto se si è appena collegato alla rete il più grande "divoratore" di ampiezza di banda della città.

Man mano che una rete ADSL attira utenti, il numero crescente di nuovi abbonati influenza molto poco gli utenti esistenti poiché ognuno ha una connessione dedicata. Nel caso del modem via cavo, invece, le prestazioni sono inversamente proporzionali al numero di abbonati. Per limitare questo problema è necessario che l'operatore divida i cavi con maggior traffico e colleghi ogni segmento direttamente al nodo fibra. Purtroppo ciò richiede tempo e denaro, e spesso questa operazione non viene eseguita.

Come digressione, nei paragrafi precedenti è stato studiato un altro sistema basato su un canale condiviso simile al cavo: il sistema telefonico mobile. Anche in questo caso un gruppo di utenti, che potrebbero essere chiamati "compagni di cella", condividono una quantità fissa di ampiezza di banda. Normalmente essa è divisa rigidamente in blocchi fissi da FDM e TDM tra gli utenti attivi, perché il traffico vocale è abbastanza uniforme. Ma nel caso del traffico dati, questa divisione rigida è molto inefficiente perché gli utenti spesso rimangono inattivi, facendo così sprecare l'ampiezza di banda loro riservata; da questo punto di vista l'accesso via cavo assomiglia più al sistema telefonico mobile che a quello fisso.

La disponibilità è un punto sul quale ADSL e il modem via cavo differiscono. Tutti hanno un telefono, ma la distanza dalla centrale locale non consente a tutti gli utenti di ottenere ADSL. D'altra parte, non tutti hanno una connessione via cavo, ma se c'è a disposizione un cavo e se il fornitore offre anche l'accesso a Internet si può richiedere tale servizio. La distanza dal nodo fibra o dalla terminazione principale non è un problema. Vale anche la pena notare che, poiché il cavo è nato come mezzo di distribuzione del segnale televisivo, arriva in pochi locali commerciali.

Essendo un mezzo di trasmissione da punto a punto, ADSL è intrinsecamente più sicuro del modem via cavo. Qualunque utente collegato al cavo televisivo può facilmente leggere tutti i pacchetti trasmessi attraverso il cavo; per questo motivo la maggior parte dei fornitori di servizi via cavo codifica tutto il traffico dati in entrambe le direzioni. Tuttavia, la possibilità che il vicino di casa possa comunque intercettare i messaggi codificati diretti a un altro utente rende la situazione meno sicura rispetto all'ADSL.

Il sistema telefonico è generalmente più affidabile di quello via cavo. Per esempio, dispone di un alimentazione di riserva e continua a funzionare normalmente anche durante le interruzioni di corrente. Con il cavo, se uno degli amplificatori lungo la catena si spegne tutti gli utenti in ricezione sono istantaneamente tagliati fuori dalla connessione. Infine, la maggior parte dei fornitori di servizi ADSL permette di scegliere tra diversi ISP (qualche volta è proprio la legge che impone questo comportamento); lo stesso non sempre accade con gli operatori via cavo.

La conclusione è che ADSL e il cavo sono molto più simili che diversi: offrono servizi comparabili e, al crescere della concorrenza tra i due, probabilmente anche i prezzi si assomiglieranno.

2.8 Sommario

Lo strato fisico è la base di tutte le reti. La natura impone due limiti fondamentali su tutti i canali, che determinano la loro ampiezza di banda: il limite di Nyquist, che si occupa dei canali senza rumore, e il limite di Shannon, che si occupa dei canali rumorosi.

I mezzi di trasmissione possono essere guidati e non guidati. I principali mezzi di trasmissione guidati sono il doppino telefonico, il cavo coassiale e la fibra ottica. I mezzi non guidati includono le onde radio, le microonde, i raggi infrarossi e i laser trasmessi attraverso l'aria. Un sistema di trasmissione in sviluppo è la comunicazione via satellite, in particolare i sistemi LEO.

Un elemento chiave nella maggior parte delle reti geografiche è il sistema telefonico. I suoi componenti principali sono i collegamenti locali, le linee e i commutatori. I collegamenti locali sono analogici, basati su doppini telefonici, per questo la trasmissione di dati digitali richiede l'uso dei modem. ADSL riesce a offrire velocità che arrivano a 50 Mbps dividendo il collegamento locale in molti canali virtuali, e modulando ogni canale separatamente. I collegamenti locali wireless sono un altro nuovo sviluppo che vale la pena di tenere d'occhio, specialmente i sistemi LMDS.

Le linee sono digitali e possono essere unite in multiplexing in diversi modi, tra cui FDM, TDM e WDM. Sono importanti sia la commutazione di circuito sia quella di pacchetto.

Il sistema telefonico fisso non è adatto alle applicazioni mobili; oggi si usano ovunque i telefoni cellulari per trasmettere la voce, e presto saranno adoperati anche per scambiare dati. La prima generazione è stata analogica, dominata da AMPS; la seconda generazione è stata digitale, con D-AMPS, GSM e CDMA; la terza generazione sarà digitale e si baserà su CDMA a banda larga.

Un sistema alternativo di accesso alle reti è il sistema televisivo via cavo, che si è evoluto gradualmente da antenna collettiva a ibrido fibra/coassiale. Potenzialmente offre un'ampiezza di banda molto grande, ma in realtà l'ampiezza di banda disponibile dipende molto dal numero di utenti attivi e dalle loro operazioni.

Problemi

- Calcolare i coefficienti di Fourier per la funzione $f(t) = t$ dove $(0 \leq t \leq 1)$.
- Un canale a 4 kHz senza rumore è campionato ogni 1 msec. Qual è la massima cadenza dei dati?
- I canali televisivi sono larghi 6 MHz. Quanti bit al secondo è possibile inviare se si utilizzano segnali digitali a quattro livelli? Si assuma che il canale sia senza rumore.
- Un segnale binario è trasmesso attraverso un canale a 3 kHz il cui rapporto segnale rumore è di 20 dB; qual è la velocità dati massima che si può raggiungere?
- Quale rapporto segnale rumore consente di inserire una portante T1 su una linea a 50 kHz?
- Qual è la differenza tra una stella passiva e un ripetitore attivo in una rete basata su fibra ottica?
- Quanta ampiezza di banda c'è in una porzione dello spettro ampia 0,1 micron alla lunghezza d'onda di 1 micron?

8. Si desidera inviare attraverso una fibra ottica una sequenza di immagini catturate dallo schermo di un computer. La risoluzione delle immagini è di 640 x 480 pixel, ogni pixel occupa 24 bit, e si vogliono inviare 60 immagini al secondo. Quale ampiezza di banda serve? Quanti micron di spettro sono necessari per ottenere questa banda, alla lunghezza d'onda di 1,30 micron?
9. Il teorema di Nyquist vale anche per la fibra ottica o vale solo per il cavo in rame?
10. La banda mostrata nel riquadro di sinistra della Figura 2.6 è più stretta delle altre. Perché?
11. Le antenne radio spesso funzionano meglio quando il loro diametro è uguale alla lunghezza d'onda. Antenne ragionevoli hanno diametri compresi tra 1 cm a 5 m. Quale intervallo di frequenza è coperto da queste antenne?
12. Il multipath fading è massimo quando i due raggi arrivano fuori fase di 180 gradi. Quanta differenza di percorso rende massimo questo disturbo in un collegamento microonde a 1 GHz lungo 50 Km?
13. Un raggio laser largo 1 mm è puntato su un rilevatore di 1 mm distante 100 m posto sul tetto di un edificio. Quale deviazione angolare (espressa in gradi) deve avere il laser per mancare il bersaglio?
14. I 66 satelliti in orbita bassa del progetto Iridium sono divisi in sei catene che circondano la Terra. All'altitudine corrente, il loro periodo è di 90 minuti. Qual è l'intervallo medio di commutazione da un satellite all'altro per un trasmettitore terrestre?
15. Si consideri un satellite in orbita geostazionaria il cui piano orbitale è inclinato di un angolo f rispetto al piano equatoriale. A un utente stazionario che si trova sulla superficie terrestre a latitudine nord f , questo satellite appare fermo nel cielo? Se la risposta è no, descrivere il suo movimento.
16. Quanti prefissi di centrale locale esistevano prima del 1984, quando ogni centrale era identificata con dal suo codice di tre cifre unito alle prime tre cifre del numero locale? I prefissi iniziavano con un numero compreso tra 2 e 9, avevano 0 o 1 come seconda cifra e potevano terminare con qualunque cifra. Le prime due cifre di un numero locale erano sempre comprese tra 2 e 9 e la terza cifra poteva essere qualunque numero.
17. Utilizzando solo i dati riportati nel testo, qual è il numero massimo di telefoni che il sistema statunitense esistente può supportare senza modificare lo schema di numerazione o aggiungere nuove apparecchiature? Questo numero potrebbe realmente essere raggiunto? Per il risultato di questo problema, ogni computer e ogni fax conta come un telefono. Si supponga che ci sia un unico apparecchio per abbonato.
18. Un semplice sistema telefonico è composto da due centrali locali e una singola centrale interurbana, alla quale si collega ogni centrale locale mediante una linea full duplex a 1 MHz. Il telefono è utilizzato mediamente per fare quattro chiamate durante le 8 ore del giorno lavorativo. La durata media di una chiamata è 6 minuti. Dieci chiamate su cento sono interurbane (e quindi passano attraverso la centrale interurbana). Qual è il numero massimo di telefoni che una centrale locale può supportare? (Si supponga di disporre di 4 kHz per circuito).

19. Un'azienda telefonica regionale ha 10 milioni di abbonati. Ogni telefono si collega a una centrale attraverso un doppino. La lunghezza media di questi doppini è di 10 Km. Qual è il valore del rame utilizzato nei collegamenti locali? Si supponga che la sezione di ogni cavo abbia un diametro di 1 mm, che la densità del rame sia di 9 grammi/cm³ e che il prezzo del rame sia di 3,00 € al Kg.
20. L'oleodotto è un sistema simplex, half duplex, full duplex o di un altro tipo?
21. Il prezzo di un microprocessore veloce è diminuito a tal punto che oggi è possibile inserirne uno in ogni modem. In che modo questo influenza la gestione degli errori di linea del telefono?
22. Un diagramma costellazione di un modem simile a quello mostrato nella Figura 2.25 ha punti collocati alle seguenti coordinate: (1, -1), (1, 1), (-1, 1) e (-1, -1). Quanti bps può raggiungere un modem con questi parametri a 1.200 baud?
23. Un diagramma costellazione di un modem simile a quello mostrato nella Figura 2.25 ha punti collocati alle seguenti coordinate (0, 1) e (0, 2). Il modem utilizza la modulazione di fase o la modulazione di ampiezza?
24. In un diagramma costellazione tutti i punti si trovano su un cerchio centrato sull'origine. Che tipo di modulazione viene usata?
25. Quante frequenze utilizza un modem full duplex QAM-64?
26. Un sistema ADSL che utilizza DMT alloca tre quarti dei canali dati disponibili al collegamento in ricezione. Il sistema utilizza la modulazione QAM-64 su ogni canale. Qual è la capacità del collegamento in ricezione?
27. Nell'esempio LMDS a 4 settori mostrato nella Figura 2.30, ogni settore ha il proprio canale a 36 Mbps. Secondo la teoria delle code, se il canale è carico per il 50% il tempo di accodamento sarà uguale al tempo di ricezione. In base a queste condizioni, in quanto tempo viene scaricata una pagina Web grande 5 KB? Quanto tempo ci vuole per scaricare la stessa pagina attraverso una linea ADSL a 1 Mbps? E usando un modem 56 kbps?
28. Dieci segnali, ognuno dei quali richiede 4.000 Hz, sono uniti in multiplexing su un singolo canale utilizzando la tecnica FDM. Qual è l'ampiezza di banda minima richiesta per il canale in multiplexing? Si supponga che le bande di guardia siano ampie 400 Hz.
29. Perché il tempo di campionamento PCM è stato impostato a 125 µsec?
30. Qual è il valore percentuale utilizzato per il codice di controllo su una portante T1? Ossia, quale percentuale di 1,544 Mbps non è utilizzata per i dati degli utenti?
31. Confrontare la velocità dati massima di un canale a 4 kHz senza rumore che utilizza
 - (a) la codifica analogica (per esempio QPSK) con 2 bit per campione
 - (b) il sistema PCM T1.
32. Se cade e perde traccia della sua posizione, un sistema portante T1 tenta di risincronizzarsi utilizzando il primo bit di ogni frame. Quanti frame devono essere esaminati, in media, per effettuare la sincronizzazione con una probabilità di errore dello 0,001?
33. Qual è la differenza, se c'è, tra la parte del modem che si occupa della demodulazione e la parte di un codec che gestisce la codifica? Dopotutto, entrambi i dispositivi convertono segnali analogici in segnali digitali.

Un segnale è trasmesso digitalmente attraverso un canale a 4 kHz senza rumore con un campione ogni 125 μ sec. Quanti bit al secondo sono trasmessi effettivamente con ognuno di questi metodi di codifica?

- (a) CCITT 2,048 Mbps standard
- (b) DPCM con un valore di segnale relativo di 4 bit
- (c) modulazione delta.

Un'onda sinusoidale pura di ampiezza A è codificata utilizzando la modulazione delta, con x campioni/sec. Un output di +1 corrisponde a una variazione di segnale di $+A/8$, e un segnale in output di -1 corrisponde a una variazione di segnale di $-A/8$. Qual è la frequenza più alta che può essere tracciata senza aggiungere errori?

Gli orologi di SONET hanno una velocità di deriva pari a 1 su 10^9 . Quanto tempo ci vuole perché questa deriva diventi uguale alla larghezza di un bit? Quali implicazioni hanno questi calcoli?

Nella Figura 2.37, la velocità dati di un utente per OC-3 è stata fissata a 148,608 Mbps. Mostrare in che modo questo valore può essere ottenuto dai parametri OC-3 di SONET.

Per fornire velocità dati più basse di STS-1, SONET usa un sistema di tributari virtuali (VT). Un VT è un carico utile che può essere inserito in un frame STS-1 ed essere unito con altri carichi utili parziali in modo da riempire il frame. VT1.5 utilizza 3 colonne, VT2 usa 4 colonne, VT3 utilizza 6 colonne e VT6 utilizza 12 colonne di un frame STS-1. Quale VT può soddisfare

- (a) un servizio DS-1 (1,544 Mbps)?
- (b) un servizio europeo CEPT-1 (2,048 Mbps)?
- (c) un servizio DS-2 (6,312 Mbps)?

Qual è la principale differenza tra la commutazione di messaggio e quella di pacchetto?

Qual è l'ampiezza di banda a disposizione di ogni utente in una connessione OC-12c?

Si considerino tre reti a commutazione di pacchetto, ognuna contenente n nodi. La prima rete ha una topologia a stella con uno switch centrale, la seconda è un anello (bidirezionale) e la terza è totalmente interconnessa, con un cavo che unisce ogni nodo a ogni altro nodo. Qual è il percorso (espresso in salti) di trasmissione medio, qual è il migliore e qual è il peggiore?

Confrontare il ritardo della trasmissione di un messaggio a x bit attraverso un percorso di k salti in una rete a commutazione di circuito e in una rete a commutazione di pacchetto (in una situazione di traffico leggero). Il tempo di setup del circuito è di s secondi, il ritardo di propagazione è di d secondi per salto, la dimensione del pacchetto è p bit e la velocità dati è di b bps. Sotto quali condizioni la rete a commutazione di pacchetto ha un ritardo più basso?

Si supponga di dover trasmettere x bit di dati utente attraverso un percorso a k salti in una rete a commutazione di pacchetto sotto forma di una serie di pacchetti ognuno contenente p dati e h bit di intestazione, con $x >> p + h$. La velocità della linea è di b bps e il ritardo di propagazione è trascurabile. Quale valore di p minimizza il ritardo totale?

In un sistema telefonico mobile tipico con celle esagonali, è vietato riutilizzare una banda di frequenza nelle celle adiacenti. Se sono disponibili 840 frequenze, quante di queste possono essere utilizzate per ogni cella?

Il tracciato reale di celle raramente è regolare come quello mostrato nella Figura 2.41; anche la forma delle celle di solito non è regolare. Spiegare uno dei possibili motivi di questa situazione.

46. Fare una stima approssimativa del numero di microcellule PCS di 100 m di diametro necessarie per coprire l'area della città di San Francisco (120 Km²).
47. Qualche volta, quando un utente mobile attraversa i confini di una cella ed entra in un'altra cella, la chiamata viene interrotta bruscamente anche se tutti i trasmettitori e i ricevitori funzionano perfettamente. Come mai?
48. La qualità della voce trasmessa nel sistema D-AMPS è peggiore della voce trasmessa nel sistema GSM. Questa caratteristica dipende dal fatto che D-AMPS deve essere compatibile con il precedente sistema AMPS mentre GSM invece non lo è, oppure ci sono altri motivi?
49. Calcolare il numero massimo di utenti che D-AMPS può supportare contemporaneamente dentro una singola cella. Eseguire lo stesso calcolo per GSM. Spiegare la differenza.
50. Si supponga che A , B e C stiano trasmettendo contemporaneamente i bit 0 usando un sistema CDMA con le sequenze di chip rappresentate nella Figura 2.45(b). Qual è la sequenza di chip risultante?
51. Nella discussione sull'ortogonalità delle sequenze di chip CDMA è stato asserito che se $S \cdot T = 0$ allora anche $S \cdot \bar{T}$ è uguale a 0. Dimostrare questa proprietà.
52. Si consideri un modo diverso di osservare la proprietà ortogonale delle sequenze di chip CDMA. Ogni bit in una coppia di sequenze può coincidere o non coincidere. Esprimere l'ortogonalità in termini di coincidenza e non coincidenza.
53. Un ricevitore CDMA riceve i seguenti chip: (-1 +1 -3 +1 -1 -3 +1 +1). Supponendo che le sequenze di chip siano quelle definite nella Figura 2.45(b), quale stazione ha trasmesso e quali bit sono stati trasmessi da ogni stazione?
54. La parte termale del sistema telefonico ha una topologia a stella, con i collegamenti locali che uniscono le abitazioni degli utenti alla centrale locale; invece la televisione via cavo è composta da un singolo cavo che si snoda attraverso le abitazioni degli utenti. Si supponga di utilizzare per la TV una fibra a 10 Gbps al posto del cavo in rame; sarebbe possibile simulare il modello telefonico con ogni utente che dispone di una linea privata diretta verso la centrale? In caso affermativo, quante abitazioni dotate di un singolo telefono potrebbero essere agganciate a una singola fibra?
55. Un sistema televisivo via cavo ha più di 100 canali commerciali che alternano programmi e pubblicità. Questo sistema assomiglia più a TDM o a FDM?
56. Un'azienda che fornisce servizi via cavo decide di fornire l'accesso a Internet attraverso il cavo in una zona dove abitano 5.000 famiglie. La società utilizza il cavo coassiale e alloca lo spettro per raggiungere un'ampiezza di banda in ricezione di 100 Mbps per cavo. Per attirare clienti, l'azienda decide di garantire a ogni abitazione un'ampiezza di banda minima di 2 Mbps in ricezione in qualunque momento. Descrivere quello di cui l'azienda ha bisogno per soddisfare questa garanzia.
57. In base all'allocazione dello spettro mostrata nella Figura 2.48 e alle informazioni riportate nel testo, quanti Mbps alloca un sistema via cavo al canale di trasmissione e quanti al canale di ricezione?
58. A quale velocità un utente di una rete via cavo riceve i dati se la rete non è occupata?

La tecnica di unire in multiplexing diversi flussi di dati STS-1, chiamati tributari, gioca un ruolo importante in SONET. Con un multiplexing di 3:1, tre tributari STS-1 in input sono uniti in un flusso di output STS-3. Questo multiplexing è eseguito byte per byte, ossia i primi tre byte in uscita sono rispettivamente il primo byte del tributario 1, il primo byte del tributario 2 e il primo byte del tributario 3; i successivi tre byte sono i secondi byte dei tre tributari e così via. Scrivere un programma che simuli questo multiplexing 3:1. Il programma deve essere composto da cinque processi. Il processo principale crea quattro processi, tre per i tre tributari STS-1 e uno per il dispositivo che esegue il multiplexing. Ogni processo tributario legge un frame STS-1 prelevando una sequenza di 810 byte da un file di input. I processi inviano i loro frame (byte per byte) al processo che esegue il multiplexing. Quest'ultimo riceve i byte e genera in uscita un frame STS-3 (byte per byte) scritto in output standard. Utilizzare le pipe per gestire la comunicazione tra processi.

3

Lo strato data link

In questo capitolo studieremo i principi dell'architettura dello strato numero 2: lo strato *data link*. Discuteremo gli algoritmi per ottenere una comunicazione affidabile ed efficiente fra due macchine adiacenti attraverso lo strato data link. Con il termine adiacenti intendiamo il fatto che le due macchine sono connesse da un canale di comunicazione che agisce concettualmente come un cavo (per esempio un cavo coassiale, una linea telefonica o un canale wireless punto-punto). La proprietà essenziale di un canale per assimilarlo a un cavo è che i bit vengano instradati nell'esatto ordine in cui sono stati inviati.

In prima battuta si può pensare che il problema sia così semplice da non richiedere lo studio di alcun software: banalmente la macchina A mette i bit nel cavo e la macchina B li preleva. Purtroppo i circuiti di comunicazione commettono occasionalmente degli errori. Inoltre la velocità di trasmissione dei circuiti è finita, quindi esiste sempre un ritardo di propagazione fra quando i bit vengono inviati e quando vengono ricevuti. Queste limitazioni hanno implicazioni importanti per l'efficienza del trasferimento dati. I protocolli usati per le comunicazioni, che sono l'argomento del presente capitolo, devono tenere in considerazione tutti questi fattori.

Dopo un'introduzione alle principali caratteristiche strutturali dello strato data link, inizieremo il nostro studio dei protocolli esaminando la natura degli errori, le loro cause e come possono essere rilevati e corretti. Studieremo poi una serie di protocolli di complessità crescente e vedremo come ognuno di questi risolve un numero sempre maggiore di problemi presenti nello strato data link. Infine, concluderemo parlando di modelli per lo studio della correttezza dei protocolli e daremo alcuni esempi di protocollo data link.

1 Progetto dello strato data link

Lo strato data link si prende carico di diverse funzioni fra cui:

1. fornire un ben definito servizio d'interfaccia per lo strato network
2. gestire gli errori di trasmissione
3. regolare il flusso dati in modo che i dispositivi riceventi lenti non vengano sovrappiatti dai trasmettitori veloci.

Per raggiungere questi obiettivi, lo strato data link prende i pacchetti provenienti dallo strato network e li incapsula in frame prima di trasmetterli. Ogni frame contiene un'intestazione (*header*), un campo per contenere il pacchetto e una coda del frame come illustrato nella Figura 3.1. La gestione dei frame costituisce il cuore dell'attività allo strato data link, che studieremo in dettaglio nel prossimo paragrafo.

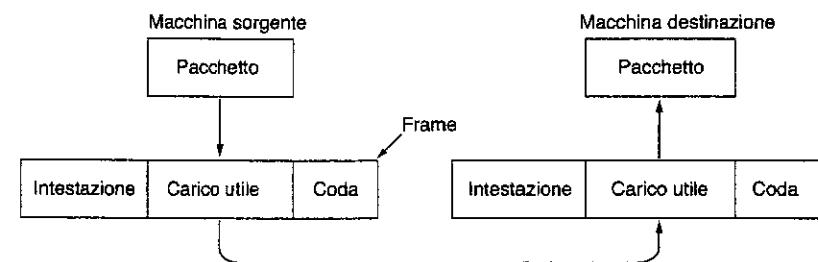


Figura 3.1. Relazione fra pacchetti e frame.

Questo capitolo tratta esplicitamente dello strato data link e dei protocolli a esso associati; nonostante ciò, molti dei principi che studieremo (per esempio la correzione degli errori e il controllo di flusso), si ritrovano anche nei protocolli di trasporto. In effetti, in molte reti questo tipo di funzioni si trova solamente negli strati superiori al data link. I principi base sono sempre gli stessi, indipendentemente da dove si trovano implementati; quindi non è molto importante a che livello li studiamo. Nello strato data link queste funzioni sono spesso implementate nella loro forma più pura, il che ci motiva a studiarle in questo capitolo.

1.1 Servizi forniti allo strato network

La funzione dello strato data link consiste nel fornire servizi allo strato network. Il servizio principale è quello di trasferire dati dallo strato network della macchina sorgente allo strato network della macchina destinazione. Sulla macchina sorgente consideriamo un'entità, che possiamo chiamare processo, localizzata nello strato network e che passa alcuni dati allo strato data link perché vengano trasmessi alla destinazione. Il lavoro dello strato data link è quello di trasmettere i bit alla macchina di destinazione, in modo che possano

essere passati allo strato network della destinazione stessa, come mostrato nella Figura 3.2(a). Nella Figura 3.2(b) è rappresentato il percorso completo della trasmissione dei dati. Risulta comunque più semplice rappresentare la situazione tramite un modello dove i due processi a livello data link comunicano fra di loro direttamente, tramite un protocollo di tipo data link come mostrato nella Figura 3.2(a). Questo tipo di modello verrà usato implicitamente nel resto di questo capitolo.

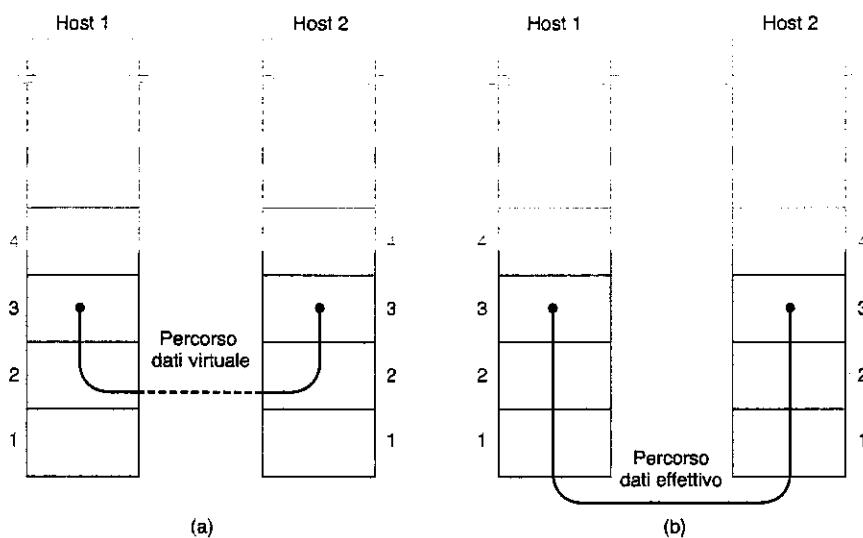


Figura 3.2. (a) Comunicazione virtuale. (b) Comunicazione reale.

Lo strato data link può essere disegnato in modo da offrire una varietà di servizi, che cambiano da sistema a sistema. Tre servizi che vengono comunemente forniti a questo livello sono:

1. servizio unacknowledged senza connessione
2. servizio acknowledged senza connessione
3. servizio acknowledged orientato alla connessione.

Il servizio di tipo unacknowledged senza connessione consiste nell'avere una macchina sorgente che invia dei frame indipendenti alla macchina destinazione, senza che quest'ultima debba rispondere con un acknowledgement (la conferma dell'avvenuta ricezione). Nessun tipo di connessione logica viene stabilita prima della trasmissione né viene rilasciata dopo. Se un frame viene perso a causa del rumore sulla linea, non viene fatto nessun tentativo di rilevare la perdita o di correggerla nello strato data link. L'uso di questa classe di servizio è legittimo quando la frequenza degli errori di trasmissione è molto bassa, così che la correzione può essere fatta dagli strati superiori. È altresì legittimo per

traffico di tipo real-time, come le trasmissioni di voce, per le quali un ritardo è peggio della trasmissione di dati incorretti. La maggior parte delle LAN usano allo strato data link un servizio di tipo unacknowledged senza connessione.

Il passo successivo in termini di affidabilità è dato dai servizi di tipo acknowledged senza connessione. Questo genere di servizio continua a non usare nessun tipo di connessione logica, però ciascun frame è inviato individualmente e ne viene fatto l'acknowledge (conferma della ricezione). In questo modo il mittente riesce a sapere se un frame è arrivato a destinazione in modo corretto oppure no. Se non è arrivato entro uno specifico intervallo di tempo, il frame può essere rispedito. Si tratta di un servizio utile per i canali di trasmissione non affidabili, come per esempio le reti wireless.

Nonostante la pena di enfatizzare il fatto che l'acknowledgement per lo strato data link è da considerarsi semplicemente come un'ottimizzazione, mai come un requisito obbligatorio. Infatti è sempre possibile fare in modo che sia lo strato network a richiedere l'acknowledgement dei pacchetti. Se l'acknowledgement non arriva entro la scadenza di un intervallo temporale, il mittente può semplicemente rispedire l'intero messaggio. Il problema di questa strategia è che i frame tipicamente hanno una lunghezza massima imposta dall'hardware, mentre i pacchetti dello strato network non hanno questo tipo di limite. Per esempio, se il tipico pacchetto viene spezzato in 10 frame e il 20% dei frame è perso, può essere necessario un tempo considerevole per far passare l'intero pacchetto. E invece i frame hanno un acknowledgement con eventuale ritrasmissione a livello individuale, i pacchetti nella loro interezza vengono trasmessi molto più velocemente. Su canali affidabili come le fibre ottiche, la ridondanza nel protocollo data link può rivelarsi inutile; al contrario nei canali di tipo wireless, per via della loro intrinseca inaffidabilità, il costo è ampiamente ripagato.

Ritornando alla descrizione dei servizi per lo strato data link, quello più sofisticato fra i tre menzionati sopra include la possibilità di stabilire una connessione. Ovvvero, con questo tipo di servizio le macchine sorgente e destinazione stabiliscono una connessione prima d'iniziare a trasferire i dati. Ogni frame trasferito attraverso la connessione è numerato, e lo strato data link garantisce che venga effettivamente ricevuto. Risulta anche garantita la ricezione dei frame con l'ordine corretto. Utilizzando dei servizi senza connessione, viceversa, un acknowledgement perso può far sì che il pacchetto venga trasmesso più volte e quindi anche ricevuto ripetutamente. È il contrario di quanto accade per i servizi con connessione, che forniscono ai processi dello strato network un flusso di bit affidabile.

Quando si usa un servizio con connessione, il trasferimento dei dati avviene in tre fasi distinte. Nella prima fase viene stabilita la connessione; il mittente e il ricevente inizializzano variabili e contatori necessari per tenere traccia di quali frame sono stati ricevuti e quali no. Nella seconda fase uno o più frame vengono trasmessi. Nella terza fase la connessione viene rilasciata, liberando le variabili, i buffer e le altre risorse usate per mantenere la connessione.

Consideriamo un tipico esempio: la subnet di una WAN costituita da router connessi a linee telefoniche punto-punto. Quando un frame arriva al router l'hardware controlla se contiene errori (usando delle tecniche che studieremo in questo capitolo), poi passa il frame allo software nello strato data link (che potrebbe essere integrato in un chip nell'inter-

terfaccia di rete). Il software allo strato data link controlla che il frame ricevuto sia quello che si aspettava; se questo è il caso, passa al software di routing il pacchetto contenuto nel campo di carico (*payload*). Il software di routing sceglie l'appropriata via di uscita e passa il pacchetto al software dello strato data link che lo trasmette. Il flusso tra due router è mostrato nella Figura 3.3.

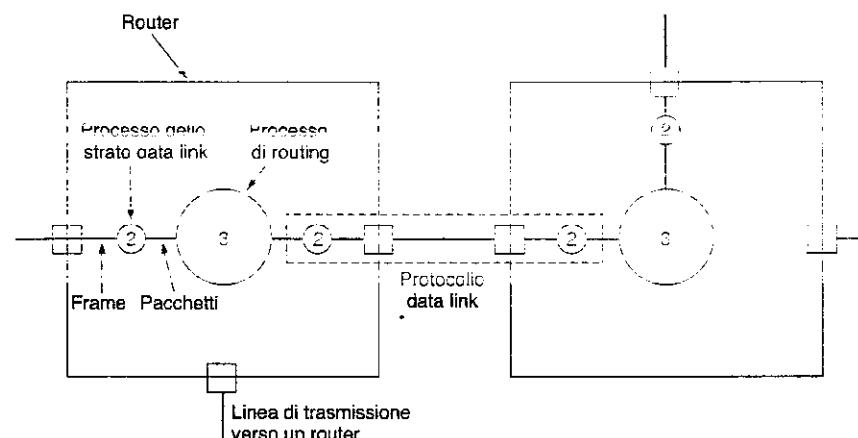


Figura 3.3. Posizione del protocollo data link.

Il software di routing esige che questo lavoro sia fatto bene, il che vuol dire avere connessioni affidabili e che mantengano l'ordine del flusso di dati su ogni linea punto-punto. Il software di routing, al contrario, non gradisce situazioni dove è costretto troppo spesso a trattare pacchetti che hanno perso il corretto instradamento. È compito del protocollo data link (vedi anche il rettangolo a linea punteggiata nella Figura 3.3) far sembrare perfette, o quanto meno più che accettabili, anche linee di comunicazione inaffidabili. Per inciso, anche se abbiamo mostrato delle copie multiple del software per lo strato data link in ogni router, in realtà sarà solo una copia a prendersi in carico tutte le linee, usando però tabelle e strutture dati separate per ciascuna.

3.1.2 Suddivisione in frame

Per servire lo strato network, lo strato data link deve usare a sua volta il servizio che gli è fornito dallo strato fisico, che ha come compito quello di prendere un flusso di bit e cercare di portarli a destinazione. Non esiste la garanzia che tale flusso di bit sia privo di errori. Il numero di bit ricevuti può essere uguale, maggiore o minore del numero dei bit trasmessi, e i loro valori possono essere diversi. Quello di rilevare ed eventualmente correggere gli errori è uno dei compiti dello strato data link.

Il tipico approccio per lo strato data link è quello di suddividere il flusso di bit in una serie discreta di frame e calcolare il checksum per ogni frame (gli algoritmi di checksum saranno

no discussi più avanti in questo capitolo). Quando un frame arriva a destinazione, il checksum viene ricalcolato. Se il checksum calcolato dal destinatario è differente da quello contenuto nel frame, lo strato data link sa che c'è stato un errore e prende i provvedimenti necessari, per esempio scartando il frame corrotto ed eventualmente mandando indietro un messaggio di errore. Suddividere il flusso di bit in frame è più difficile di quanto appare a prima vista. Un metodo per realizzare la suddivisione in frame è quello di inserire intervalli temporali fra i singoli frame, secondo lo stesso principio per cui si usano gli spazi per separare le parole di un testo. Le reti, però, difficilmente riescono a garantire la successione temporale dei segnali; quindi può capitare che gli intervalli temporali vengano modificati per eliminarli o aggiungerne di nuovi. Poiché risulta troppo rischioso contare sulla sequenza temporale per segnare l'inizio e la fine di ciascun frame, sono stati sviluppati altri metodi. In questo capitolo ne vedremo quattro:

1. conteggio dei caratteri
2. flag byte con byte stuffing
3. flag di inizio e fine con bit stuffing
4. violazioni della codifica dello strato fisico.

Il primo metodo di framing usa un campo nell'intestazione per specificare il numero di caratteri nel frame. Quando lo strato data link della destinazione legge tale numero, sa quanti caratteri seguiranno e quindi sa dove si trova la fine del frame. Questa tecnica è mostrata nella Figura 3.4(b) per quattro frame di dimensioni di 5, 5, 8 e 8 caratteri rispettivamente.

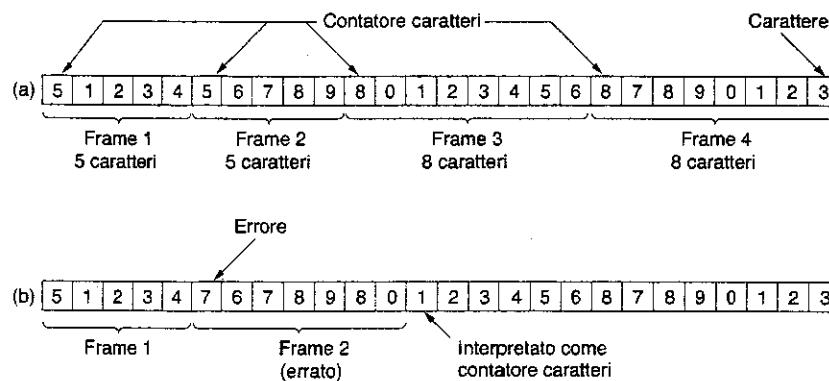


Figura 3.4. Un flusso di caratteri: (a) senza errori, (b) con errori.

Il problema di questo algoritmo è che il conteggio può essere alterato da un errore di trasmissione. Per esempio, se il conteggio dei 5 caratteri nel secondo frame della Figura 3.4(b) diventa un 7, il destinatario andrà fuori sincronia e non sarà in grado di trovare l'inizio del frame successivo. Anche se il calcolo del checksum è scorretto, e perciò il destinatario sa che il frame è effettivamente da scartare, non gli sarà possibile trovare la posi-

zione di partenza del frame successivo. Mandare indietro un frame alla sorgente per chiederne la ritrasmissione non è una soluzione, in quanto il destinatario non riesce a sapere quanti caratteri deve saltare per arrivare all'inizio della nuova trasmissione. Per questo motivo il metodo del conteggio dei caratteri non è quasi più utilizzato.

Il secondo metodo di framing aggira il problema della nuova sincronizzazione, necessaria a seguito di un errore, introducendo un byte speciale all'inizio e al termine di ogni frame. Nel passato i byte di inizio e fine erano differenti, mentre oggi la maggior parte dei protocolli usa lo stesso byte, chiamato **flag byte**, per delimitare sia l'inizio sia la fine dei frame come indicato nella Figura 3.5(a) con FLAG. In questo modo, quando il destinatario perde la sincronizzazione, può semplicemente cercare il flag byte per trovare la fine del frame corrente. Due flag byte consecutivi indicano la fine di un frame e l'inizio del successivo.

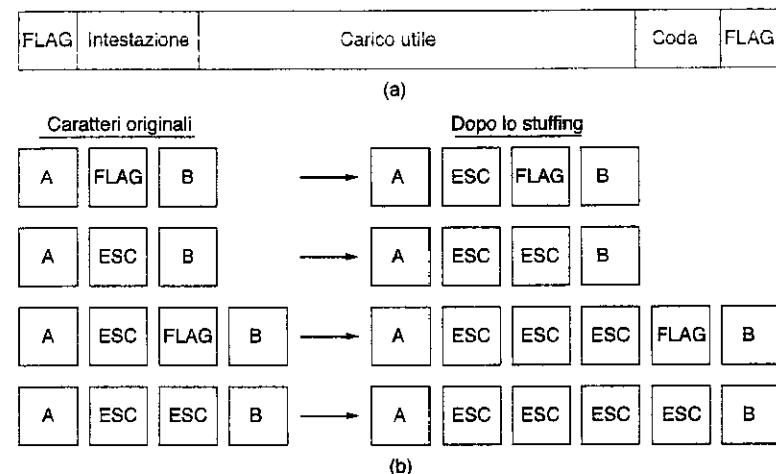


Figura 3.5. (a) Un frame delimitato dai byte di flag.
(b) Quattro esempi di sequenze di byte prima e dopo il byte stuffing.

Questo metodo presenta un problema molto serio quanto vengono trasferiti dati binari, per esempio eseguibili o numeri in virgola mobile. In questo caso può facilmente accadere che il valore corrispondente al flag byte compaia naturalmente dentro ai dati, interferendo così con le operazioni di framing. Un modo per risolvere questo problema consiste nel far sì che la sorgente inserisca un byte di escape (ESC) subito prima di ogni occorrenza "accidentale" del flag byte nei dati. Lo strato data link della destinazione provvederà a rimuovere i byte di escape prima di passare i dati allo strato network. Questa tecnica è chiamata **byte stuffing** o anche **character stuffing**. In questo modo un flag byte usato per il framing può essere differenziato da uno appartenente ai dati, poiché quest'ultimo è preceduto da un byte di escape. Naturalmente la domanda successiva è: cosa succede se un byte di escape si trova dentro ai dati? La risposta, anche in questo caso, è che deve essere preceduto da un byte di escape. In questo modo ogni byte di escape singolo deve essere considerato come parte di una sequenza di escape, mentre ogni valore ripetuto due volte rappresenta un valore nel flus-

o dei dati. Alcuni esempi sono riportati nella Figura 3.5(b). In tutti i casi la sequenza di byte ricavata dopo la conversione dei byte di escape (*destuffing*) è esattamente uguale alla sequenza originale.

Lo schema byte stuffing rappresentato nella Figura 3.5 è una leggera semplificazione di quello usato nel protocollo PPP, che la maggior parte dei computer domestici usano per comunicare con gli Internet service provider (ISP). Discuteremo il protocollo PPP più avanti in questo capitolo.

Questo metodo di framing ha il grave svantaggio di essere legato all'uso di caratteri a 8 bit; infatti non tutte le codifiche dei caratteri usano 8 bit, per esempio UNICODE usa una codifica a 16 bit. Con il tempo e lo svilupparsi delle reti sono emersi gli svantaggi derivanti dall'inserire la lunghezza in caratteri nel meccanismo di framing, e questo ha portato alla necessità di sviluppare una nuova tecnica che consenta di gestire caratteri di lunghezza arbitraria.

Una nuova tecnica permette di creare data frame che contengono sia un numero arbitrario di bit, sia codifiche di carattere con un numero arbitrario di bit. Il funzionamento è il seguente: ogni frame comincia e finisce con un gruppo speciale di bit, 01111110, che in sostanza è un flag byte. Ogni volta che lo strato data link della sorgente incontra cinque 1 consecutivi nei dati inserisce automaticamente un bit con valore 0 nel flusso in uscita. Questa operazione è chiamata **bit stuffing** analogamente al caso del byte stuffing, dove nel flusso dei dati in uscita s'inscrive un byte di escape subito prima di un valore uguale al flag byte.

Quando la destinazione riceve cinque bit consecutivi con valore 1 seguiti da uno 0, automaticamente elimina lo 0. Così come il byte stuffing risultava completamente trasparente allo strato network della sorgente e della destinazione, così accade anche per il bit stuffing. Se i dati da trasferire contengono la sequenza flag 01111110 al loro interno, tale sequenza verrà trasmessa come 011111010, ma arriverà nella memoria della destinazione come 01111110 dopo

(a) 011011111111111111110010
 (b) 0110111110111110111111010010
 Bit sottoposti a stuffing
 (c) 0110111111111111111111110010

Figura 3.6. Bit stuffing. (a) I dati originali. (b) I dati che appaiono sulla linea. (c) I dati che vengono memorizzati nella memoria della destinazione dopo il destuffing.

operazione di destuffing. La Figura 3.6 rappresenta un esempio di destuffing. Con il bit stuffing il confine fra due frame viene riconosciuto in modo inequivocabile tramite l'uso della sequenza flag, visto che questa può essere presente solo al confine fra i frame e mai al loro interno.

L'ultimo metodo di framing si applica solo alle reti in cui la codifica contiene delle redundanze a livello fisico. Per esempio alcune LAN codificano un bit dati usando due bit fisici: normalmente il bit 1 è rappresentato da una coppia alto-basso e lo 0 da una coppia basso-alto. Lo schema implica che ogni bit di dati contenga una transizione,

quindi è facile per la destinazione decodificare la posizione dei singoli bit. Le combinazioni alto-alto e basso-basso non sono usate per i dati, perciò alcuni protocolli le usano per delimitare i frame.

Come nota finale sul framing notiamo che molti protocolli data link usano, per aumentare la ridondanza, l'abbinamento tra la tecnica del conteggio dei caratteri e uno degli altri metodi descritti. Quando il frame arriva a destinazione, il campo con il conteggio dei caratteri viene usato per calcolare la fine del frame. Il frame è accettato come valido solo se l'appropriato delimitatore è presente nella posizione prevista e il checksum è corretto. Altrimenti si procede a leggere il flusso in ingresso fino al successivo delimitatore.

3.1.3 Controllo degli errori

Dopo aver risolto il problema di come delimitare l'inizio e la fine di ogni frame, procediamo con il successivo: come assicurarsi che alla destinazione arrivino tutti i frame, e che arrivino nell'ordine corretto. Supponiamo che la sorgente continui semplicemente a inviare frame, senza controllare se stanno arrivando in modo corretto. Questo potrebbe anche andar bene per un servizio unacknowledged senza connessione, ma quasi certamente non sarebbe accettabile per un servizio con connessione.

Il modo canonico per assicurare l'affidabilità dell'arrivo dei dati consiste nel fornire alla sorgente una qualche forma di reazione su quello che sta succedendo dall'altro capo della linea. Tipicamente il protocollo richiede che la destinazione mandi indietro degli speciali frame di controllo, che contengono un acknowledgement positivo o negativo relativamente ai frame ricevuti. Se la sorgente riceve un acknowledgement positivo riguardo a un certo frame, questo vuol dire che il frame è stato ricevuto correttamente. Al contrario, un feedback negativo vuol dire che qualcosa è andato storto e il frame va ritrasmesso.

Un'altra complicazione nasce quando problemi hardware possono far scomparire totalmente un frame (per esempio a causa di un picco di rumore). In questo caso la destinazione non reagirà in alcun modo, visto che non ha niente su cui produrre una risposta. Dovrebbe essere chiaro che un protocollo in cui la sorgente trasmette un frame e poi aspetta un acknowledgement, positivo o negativo, rimarrà bloccato per sempre nel caso in cui un frame venga perso, per esempio per un malfunzionamento hardware.

Questa possibilità è gestita introducendo timer (allarmi a tempo) nello strato data link. Quando la sorgente trasmette un frame, normalmente fa anche partire un timer. Il timer viene impostato in modo da scattare dopo un intervallo abbastanza lungo da permettere che il frame giunga alla destinazione, vi venga elaborato, e l'acknowledge possa ritornare indietro alla sorgente. Normalmente il frame raggiungerà la destinazione e l'acknowledgement ritornerà indietro prima che il timer scatti, nel qual caso il timer viene ignorato. Se invece il frame o l'acknowledgement vengono persi, il timer scatterà in modo da segnalare alla sorgente un potenziale problema. L'ovvia soluzione consiste nell'inviare di nuovo il frame. Se il frame è nuovamente trasmesso si rischia però che la destinazione lo accetti due o più volte e lo passi quindi ripetutamente allo strato network. Per ovviare al problema è necessario assegnare un numero di sequenza ai frame in uscita: in questo modo la destinazione riesce a distinguere le ritrasmissioni dagli originali.

Il problema della gestione di timer e numeri di sequenza, per assicurarsi che ogni frame

venga (in ultima analisi) passato allo strato network esattamente una e una sola volta, rappresenta una parte importante dei compiti dello strato data link. Per studiare com'è implementata questa funzione, più avanti nel capitolo vedremo una serie di esempi con un livello crescente di sofisticazione.

3.1.4 Controllo di flusso

Un altro importante punto da discutere nel progetto dello strato data link (e anche in strati superiori della gerarchia) riguarda cosa fare quando la sorgente vuole sistematicamente trasmettere i frame più velocemente di quanto la destinazione sia in grado di accettarli. Questa situazione può succedere facilmente se la sorgente è implementata su un computer veloce (oppure con un carico di lavoro basso) mentre la destinazione è implementata su un computer lento (oppure con un forte carico di lavoro). La sorgente continua a pompate i frame ad alta velocità verso la destinazione finché questa non risulta completamente "sommersa". Anche se la trasmissione non presenta errori, si arriverà a un certo punto in cui la destinazione sarà semplicemente incapace di trattare i frame in arrivo, e comincerà a perderne alcuni. Chiaramente bisogna fare qualcosa per prevenire questo tipo di situazioni.

Due sono gli approcci comunemente usati. Nel primo, il **controllo di flusso tramite feedback** (retroazione), la destinazione manda indietro alla sorgente delle informazioni per darle il permesso di mandare altri dati o comunque per informarla dello stato della destinazione. Nel secondo, il **controllo di flusso tramite limitazione della velocità**, il protocollo contiene al suo interno un meccanismo che limita la velocità alla quale la sorgente può trasmettere i dati, senza alcun feedback da parte della destinazione. In questo capitolo studieremo i metodi di controllo del flusso tramite feedback, poiché quelli a limitazione della velocità non sono praticamente mai usati nello strato data link. Vedremo metodi di controllo basati sulla limitazione della velocità nel Capitolo 5.

Esistono diversi metodi di controllo di flusso tramite feedback, ma la maggior parte usa gli stessi principi base. Il protocollo contiene delle regole ben definite su quando una sorgente può trasmettere il frame successivo. Queste regole spesso proibiscono l'invio di pacchetti finché la destinazione non abbia dato il permesso implicito o esplicito. Per esempio, quando viene aperta una connessione la destinazione può dire: "Mandami solamente n frame adesso, e poi non mandare altro finché non ti dico di continuare". Esamineremo i dettagli di questi meccanismi nella restante parte del capitolo.

3.2 Rilevazione e correzione degli errori

Come abbiamo visto nel Capitolo 2, un sistema di telefonia è costituito da tre parti: gli switch, i collegamenti interoffice e i collegamenti locali. I primi due sono quasi totalmente digitali nella maggior parte dei paesi sviluppati. I collegamenti locali sono costituiti da doppini in rame e continueranno a essere così per diversi anni a venire, per l'enorme costo necessario alla loro sostituzione. Mentre gli errori sono rari nella parte digitale, sono ancora comuni nei collegamenti locali. Le comunicazioni wireless stanno inoltre diventando sempre più comuni, e qui la frequenza degli errori è di un ordine di grandezza maggiore rispetto ai rami interoffice in fibra ottica. La conclusione è che gli errori di trasmissione

rimarranno un problema attuale per diversi anni a venire. Dobbiamo imparare a convivere con questo tipo di errori. Per via dei processi fisici che li generano, in alcuni mezzi di trasporto (per esempio le onde radio) gli errori tendono a presentarsi in grappoli piuttosto che singolarmente. Questo tipo di comportamento presenta sia svantaggi sia vantaggi rispetto ad avere degli errori isolati sui singoli bit.

Il vantaggio è che i dati nei computer sono sempre trasmessi in blocchi di bit. Supponiamo di avere un blocco di dimensione 1.000 bit e che la frequenza degli errori sia 0,001 per bit. Se gli errori sono indipendenti fra loro, la maggior parte dei blocchi conterrà degli errori, se invece gli errori appaiono in gruppi (detti errori burst) di 100, solo uno o due blocchi su 100 saranno affetti da errori. Lo svantaggio di avere errori burst è che sono molto più difficili da correggere degli errori isolati.

3.2.1 Codici per la correzione degli errori

Gli architetti delle reti hanno sviluppato due strategie di base per gestire gli errori. Una consiste nell'includere in ciascun blocco dati trasmesso una quantità di informazioni ridondanti, tale da permettere di ricostruire il contenuto del blocco in caso di errore. L'altra strategia consiste nell'introdurre abbastanza ridondanza da permettere alla destinazione di capire che c'è stato un errore, ma non di correggerlo. In questo modo la destinazione potrà richiedere una ritrasmissione del blocco di dati. Si dice che la prima strategia usa una **codifica a correzione d'errore**, la seconda invece una **codifica a rilevazione d'errore**. L'uso delle codifiche a correzione di errore è spesso indicato come **forward error correction** (correzione d'errore in anticipo).

Ognuna di queste tecniche occupa una differente nicchia: su canali affidabili, come per esempio le fibre ottiche, è più economico usare codifiche a rilevazione d'errore e limitarsi a ritrasmettere quei blocchi che eventualmente risultino corrotti. Invece su canali molto rumorosi, come quelli wireless, è più conveniente aggiungere a ogni blocco una ridondanza tale che (in caso di errore) la destinazione sia in grado di ricostruire il blocco, invece di dover richiedere una nuova trasmissione (che potrebbe essere a sua volta in errore).

Per capire come gestire gli errori è necessario dare uno sguardo più da vicino, per comprendere cosa è veramente un errore. Normalmente un frame consiste di m bit di dati (cioè il messaggio) e r bit ridondanti per i controlli. Chiamiamo la lunghezza totale n , dove $n = m + r$. Un'unità di n bit che contiene dati e bit di controllo viene chiamata **codeword** di n bit.

Prendiamo due codeword, per esempio 10001001 e 10110001; com'è possibile determinare quanti bit corrispondenti sono differenti nelle due codeword? In questo caso ci sono 3 bit differenti. Per determinare questo numero basta eseguire l'OR esclusivo delle due codeword e contare il numero di bit a 1 nel risultato, per esempio:

$$\begin{array}{r} 10001001 \\ 10110001 \\ \hline 00111000 \end{array}$$

Il numero di bit corrispondenti diversi nelle due sequenze è detto la **distanza di Hamming** (Hamming, 1950). Il suo significato è che se due codeword sono a distanza di Hamming d una dall'altra, saranno necessari d errori su singoli bit per convertire una sequenza nell'altra.

Nella maggior parte delle applicazioni di trasmissione dati, tutti i 2^m possibili messaggi sono legali, ma per via del modo con cui sono calcolati i bit di controllo, non tutte le possibili 2^n codeword vengono usate. Dato l'algoritmo per calcolare i bit di controllo, è possibile costruire una lista completa delle codeword legali, e da questa lista trovare due codeword con la minima distanza di Hamming. Questa distanza è per definizione la distanza di Hamming dell'intera codifica.

Le proprietà di rilevazione e correzione degli errori di una codifica dipendono dalla sua distanza di Hamming. Per trovare d errori è necessaria una codifica con distanza $d + 1$, in quanto con tale codifica non esiste un modo in cui d errori su singoli bit possano cambiare una codeword valida in un'altra ancora valida. Quando la destinazione vede una codeword non valida riesce a determinare che c'è stato un errore. Analogamente, per correggere d errori è necessaria una codifica con distanza $2d + 1$; in tal modo le codeword legali sono così distanziate fra di loro, che anche con d cambiamenti la codeword originale è sempre più vicina di ogni altra codeword e può quindi essere determinata univocamente.

Un semplice esempio di codifica a rilevazione d'errore si può realizzare aggiungendo un bit di parità ai dati. Il bit di parità viene calcolato in modo che il numero di 1 nella codeword sia sempre pari (o dispari). Per esempio, per trasferire 1011010 con parità pari bisogna aggiungere un bit per ottenere 10110100, mentre con parità dispari sarà 10110101. Una codifica con un singolo bit di parità ha distanza 2, infatti ogni errore su un singolo bit produce una codeword con la parità sbagliata. Questa codifica può essere usata per rilevare errori su singoli bit.

Come esempio semplice di codifica a correzione d'errore consideriamo una codifica con solo quattro codeword valide:

0000000000, 0000011111, 1111100000, 1111111111

Questa codifica ha distanza 5, il che significa che può correggere errori doppi. Se arriva la codeword 0000000111, la destinazione sa che l'originale doveva essere 0000011111. Se invece un triplo errore cambia 0000000000 in 0000000111, allora l'errore non verrà trattato correttamente.

Supponiamo di voler progettare una codifica con m bit di messaggi ed r bit di controllo che permetta di correggere tutti gli errori singoli. Per ognuno dei 2^m messaggi legali esistono n codeword illegali a distanza 1, formate invertendo sistematicamente ognuno degli n bit della codeword legale. Per questo motivo ognuno dei 2^m messaggi legali ha la necessità di avere $n + 1$ combinazioni di bit dedicate. Visto che il numero totale di combinazioni di bit è 2^n , dobbiamo avere $(n + 1) 2^m \leq 2^n$. Sostituendo $n = m + r$, la diseguaglianza diventa $(m + r + 1) 2^m \leq 2^n$. Dato m questa diseguaglianza impone un limite inferiore al numero di bit di controllo necessari per correggere gli errori singoli.

Questo limite teorico può essere effettivamente raggiunto usando un metodo dovuto a Hamming (1950). I bit della codeword vengono numerati consecutivamente a partire da 1 per il primo bit sulla sinistra, 2 per quello immediatamente alla sua destra e così via. I bit che sono una potenza di 2 (1, 2, 4, 8, 16, ecc.) sono bit di controllo. I restanti bit (3, 5, 6, 7, 9, ecc.) sono riempiti con m bit di dati. Ogni bit di controllo forza la parità di alcuni gruppi di bit, incluso se stesso, a essere pari (o dispari). Un bit può essere incluso in diver-

si calcoli di parità. Per vedere quali bit di controllo sono legati a un certo bit di dati, riscriviamo k come somma di potenze di 2. Per esempio $11 = 1 + 2 + 8$ e $29 = 1 + 4 + 8 + 16$. Un bit di dati è controllato solo dai bit di controllo presenti nella sua espansione come somma di potenze di 2 (per esempio 11 è controllato solo dai bit 1, 2 e 8).

Quando una codeword viene ricevuta, la destinazione inizializza il contatore a zero. A quel punto esamina ogni bit di controllo k (dove $k = 1, 2, 4, 8, \dots$) per vedere se la parità è corretta, in caso negativo aggiunge il valore k al contatore. Se il contatore è ancora a zero quando tutti i bit di controllo sono stati esaminati (cioè se non sono stati rilevati errori), la codeword viene accettata come valida. In caso contrario il valore non nullo del contatore indica il numero del bit in errore. Per esempio, se i bit di controllo 1, 2 e 8 sono in errore il bit invertito è 11; infatti questo è l'unico che viene controllato dai bit di controllo 1, 2 e 8. La Figura 3.7 mostra alcuni caratteri ASCII a 7 bit codificati con codeword a 11 bit usando una codifica di Hamming. Ricordiamo che i dati si trovano nelle posizioni 3, 5, 6, 7, 9, 10, 11.

Caratteri	ASCII	Bit di controllo
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
c	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Ordine di trasmissione dei bit

Figura 3.7. Uso di una codifica di Hamming per correggere gli errori burst.

Le codifiche di Hamming riescono a correggere solo gli errori singoli. Con un piccolo trucco si riesce a far sì che le codifiche di Hamming riescano a correggere anche gli errori burst.

Una sequenza di k codeword consecutive è disposta riga per riga in modo da formare una matrice. Normalmente i dati sono trasmessi una codeword alla volta, riga per riga. Per correggere gli errori burst, invece, i dati vengono trasmessi colonna per colonna a partire dalla prima a sinistra. Quando tutti i k bit della prima colonna sono stati inviati si procede con la seconda e così via, come indicato nella Figura 3.7. Dopo che il frame è arrivato a destinazione la matrice viene ricostruita colonna per colonna. Se si verifica un errore burst di lunghezza k , al massimo 1 bit per ogni codeword sarà toccato, ma sappiamo che la codifica di Hamming riesce a correggere 1 errore per codeword, quindi l'intero blocco può essere corretto. Questo metodo usa kr bit di controllo per rendere immuni km bit dati agli errori burst di lunghezza minore o uguale a k .

3.2.2 Codifiche a rilevazione d'errore

Le codifiche a correzione d'errore sono ampiamente usate per le trasmissioni wireless, notoriamente rumorose e piene di errori se le confrontiamo con i cavi in rame o le fibre ottiche. Senza le codifiche a correzione d'errore sarebbe davvero difficile riuscire a trasmettere su canali wireless. Al contrario, per i cavi in rame e le fibre ottiche la frequenza degli errori è molto più bassa. In questo caso una strategia che permetta di rilevare gli errori combinata con la ritrasmissione dei dati difettosi risulta generalmente più efficiente.

Un esempio semplice è costituito da un canale in cui gli errori sono isolati e hanno una frequenza di 10^{-6} per bit. Se abbiamo blocchi con dimensione pari a 1 000 bit serviranno 10 bit di controllo per avere la correzione degli errori. Questo vuol dire che per trasferire un megabit di dati avremo bisogno di 10.000 bit di controllo. Se invece vogliamo solamente riconoscere i blocchi con un singolo bit in errore, basterà avere un bit di parità per blocco. Inoltre, dovrà essere trasmesso un blocco extra (1.001 bit) ogni 1.000 blocchi. Il carico aggiuntivo dovuto alla rilevazione degli errori sommata al metodo di ritrasmissione è solamente di 2.001 bit per ogni megabit di dati, contro i 10.000 bit della codifica di Hamming.

Se aggiungiamo ai blocchi dati un singolo bit di parità, nel caso di un errore burst di lunga estensione la probabilità che l'errore venga rilevato è solamente 0,5, il che è difficilmente accettabile. Questa probabilità può essere aumentata considerevolmente. Per far questo, consideriamo ogni blocco da trasmettere come una matrice rettangolare di n colonne e k righe, come descritto nel precedente paragrafo. Un bit di parità viene calcolato separatamente per ogni colonna e aggiunto come ultima riga della matrice. La matrice è poi trasmessa una riga alla volta. All'arrivo del blocco, la destinazione controlla tutti i bit di parità. Se uno qualunque di questi bit è errato, la destinazione richiede la ritrasmissione del blocco. Altre ritrasmissioni possono essere richieste finché l'intero blocco non sarà ricevuto senza errori di parità.

Questo metodo riesce a rilevare errori burst di lunghezza n , visto che in tale caso solo 1 bit per colonna viene alterato. Un errore burst di lunghezza $n + 1$ invece può passare senza essere rilevato: per esempio se vengono invertiti il primo e l'ultimo bit mentre gli altri rimangono inalterati. Per inciso, un errore burst non implica necessariamente che tutti i bit siano errati, basta solamente che il primo e l'ultimo lo siano. Se il blocco viene profondamente alterato da una lunga serie di errori o da più serie brevi, la probabilità che ognuna delle n colonne abbia casualmente un valore di parità giusto è 0,5, quindi la probabilità che un blocco in errore venga accettato è pari a 2^{-n} .

Lo schema che abbiamo descritto è adeguato ad alcune situazioni, però in pratica si usa comunemente un altro metodo: la **codifica polinomiale**, nota anche come **CRC** (controllo ciclico di ridondanza). Le codifiche polinomiali sono basate sul fatto di trattare le sequenze di bit come dei polinomi a coefficienti che possono assumere solo i valori 0 oppure 1. Un frame di k bit è visto come una lista di coefficienti per un polinomio con k termini che variano da x^{k-1} a x^0 . Tale polinomio è detto di grado $k-1$. Il coefficiente di termine più alto (quello più a sinistra) è il coefficiente per x^{k-1} , il successivo è per x^{k-2} e così via. Per esempio 110001 ha 6 bit e quindi rappresenta un polinomio di 5° grado con coefficienti 1, 1, 0, 0, 0 e 1: $x^5 + x^4 + x^0$.

L'aritmetica dei polinomi si gestisce in modulo 2 secondo le regole della teoria dei campi algebrici. Non ci sono riporti per le addizioni o prestiti per le sottrazioni. Sia l'addizione sia la sottrazione sono identici all'OR esclusivo. Per esempio:

$$\begin{array}{r}
 10011011 \\
 + 11001010 \\
 \hline
 01010001
 \end{array}
 \quad
 \begin{array}{r}
 00110011 \\
 + 11001101 \\
 \hline
 11111110
 \end{array}
 \quad
 \begin{array}{r}
 11110000 \\
 - 10100110 \\
 \hline
 01010110
 \end{array}
 \quad
 \begin{array}{r}
 01010101 \\
 - 10101111 \\
 \hline
 11111010
 \end{array}$$

Le divisioni lunghe vengono eseguite come in binario, salvo che le sottrazioni sono in modulo 2 come spiegato sopra. Si dice che un divisore "sta" nel dividendo se il dividendo ha tanti bit quanti il divisore.

Quando si utilizza una codifica polinomiale, la sorgente e la destinazione devono mettersi d'accordo in anticipo su un **polinomio generatore**, $G(x)$. Il generatore deve avere i bit di ordine più alto e più basso uguali a 1. Per poter calcolare il **checksum** di un frame di m bit che corrisponde al polinomio $M(x)$, il frame deve essere più lungo del polinomio generatore. L'idea è quella di aggiungere un checksum alla fine del frame in modo che il polinomio rappresentato dal frame con checksum sia divisibile per $G(x)$. Quando la destinazione riceve il frame con checksum prova a dividerlo per $G(x)$. Se c'è un resto vuol dire che c'è stato un errore di trasmissione. L'algoritmo per calcolare il checksum è il seguente:

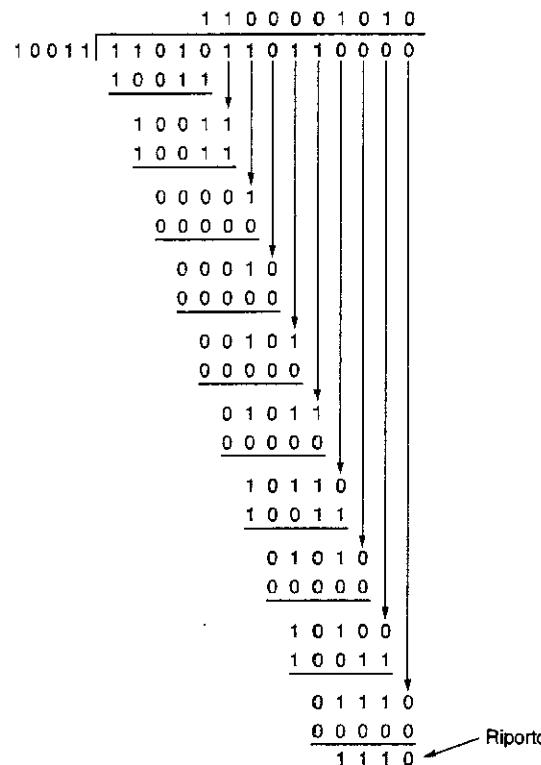
1. posto r il grado di $G(x)$, aggiungere r bit con valore zero dopo la parte di ordine più basso del frame, così che adesso contenga $m + r$ bit e corrisponda al polinomio $x^r M(x)$
2. dividere la sequenza di bit corrispondente a $G(x)$ per la sequenza corrispondente a $x^r M(x)$ usando la divisione modulo 2. [NdR - il dividendo è $x^r M(x)$ mentre il divisore è $G(x)$]
3. sottrarre il resto (che contiene sempre al massimo r bit) dalla sequenza corrispondente a $x^r M(x)$ usando la sottrazione in modulo 2. Il risultato è il frame con checksum pronto per la trasmissione. Chiamiamolo polinomio $T(x)$.

La Figura 3.8 illustra il calcolo per il frame 1101011011 usando il generatore $G(x) = x^4 + x + 1$.

Dovrebbe essere chiaro che $T(x)$ è divisibile (modulo 2) per $G(x)$. In ogni divisione, se si sottra il resto dal dividendo, quello che resta è divisibile per il divisore. Vediamo un esempio: in base 10, se dividiamo 210.278 per 10.941 il resto è 2.399. Sottraendo 2.399 da 210.278, quello che rimane (207.879) è divisibile per 10.941.

Analizziamo la potenza di questo metodo. Quali tipologie di errori potrà rilevare? Immaginiamo che accada un errore in modo che al posto della sequenza di bit $T(x)$ arrivi una sequenza $T(x) + E(x)$. Ogni bit a 1 in $E(x)$ corrisponde a un bit che è stato invertito. Se ci sono k bit a 1 in $E(x)$, vuol dire che ci sono stati k errori su singoli bit. Un singolo burst di errori è caratterizzato da un 1 iniziale, una miscela di 0 e 1 e un 1 finale, mentre tutti gli altri bit sono a 0.

Frame : 1101011011
 Generatore: 10011
 Messaggio dopo l'aggiunta di quattro bit a zero: 11010110110000



Frame trasmesso: 11010110111110

Figura 3.8. Calcolo del checksum a codifica polinomiale.

Dopo aver ricevuto il frame con checksum, la destinazione lo divide per $G(x)$, cioè calcola $[T(x) + E(x)]/G(x)$. Siccome $T(x)/G(x) = 0$ il risultato è pari a $E(x)/G(x)$. [NdR - il testo sarebbe corretto se la notazione X/Y denotasse il resto della divisione di X e Y piuttosto che il prodotto della divisione. Altrimenti la frase deve essere riformulata così: calcola $[T(x) + E(x)]/G(x)$. Siccome $T(x)/G(x)$ ha resto 0, il risultato è pari al resto di $E(x)/G(x)$.] Gli errori che corrispondono a polinomi che contengono $G(x)$ passeranno inosservati, mentre tutti gli altri verranno rilevati.

Se c'è stato un errore su un singolo bit, $E(x) = x^i$, dove i determina quale bit è in errore. Se $G(x)$ contiene due o più termini, non potrà mai dividere $E(x)$, così che tutti gli errori di singoli bit saranno rilevati.

Se abbiamo 2 errori isolati, ognuno su un singolo bit, risulta $E(x) = x^i + x^j$, dove $i > j$, che possiamo anche scrivere come $E(x) = x^i (x^{i-j} + 1)$. Assumendo che $G(x)$ non sia divisibile per x , una condizione sufficiente per poter rilevare tutti gli errori doppi è che $G(x)$ non divida $x^k + 1$ per ogni k fino a un massimo valore di $i - j$ (cioè fino alla massima lunghezza del frame). Polinomi semplici e di basso grado che possono proteggere frame di considerevole lunghezza sono noti in letteratura. Per esempio, $x^{15} + x^{14} + 1$ non divide nessun $x^k + 1$ per ogni valore di k inferiore a 32.768.

Se c'è un numero dispari di bit in errore, $E(x)$ contiene un numero dispari di termini (per esempio $x^5 + x^2 + 1$, ma non $x^2 + 1$). Nessun polinomio con un numero dispari di termini ha $x + 1$ come fattore nel sistema a modulo 2. Perciò, basta usare un $G(x)$ che contiene $x + 1$ come fattore per riuscire a rilevare tutti gli errori che consistono in un numero dispari di bit invertiti.

Per dimostrare che nessun polinomio con un numero dispari di termini può essere divisibile per $x + 1$ si procede per assurdo. Supponiamo che $E(x)$ abbia un numero dispari di termini e sia divisibile per $x + 1$, fattorizziamo $E(x) = (x + 1)Q(x)$. Ora valutiamo $E(1) = (1 + 1)Q(1)$. Siccome $1 + 1 = 0$ (modulo 2), concludiamo che $E(1) = 0$. Se $E(x)$ ha un numero dispari di termini, sostituendo $x = 1$ in ogni termine otteniamo per forza 1 come risultato. Abbiamo raggiunto una contraddizione, e quindi provato per assurdo che nessun polinomio con un numero dispari di termini può essere divisibile per $x + 1$.

In conclusione è importante notare che una codifica polinomiale con r bit di controllo è in grado di rilevare delle sequenze di errori di lunghezza $\leq r$. Un burst di errori di lunghezza k può essere rappresentato da $x^i (x^{k-i} + \dots + 1)$, dove i determina quanto lontano si trova il burst di errori dal termine più a destra della sequenza ricevuta. Se $G(x)$ contiene un termine x^0 , non avrà x^i come fattore, quindi se il grado dell'espressione tra parentesi è inferiore al grado di $G(x)$, il resto non potrà mai essere zero.

Se la lunghezza del burst di errori è $r + 1$, il resto della divisione sarà zero se e solo se l'intero burst è identico a $G(x)$. Per definizione di burst, il primo e ultimo bit devono essere uguali a 1, quindi se c'è l'uguaglianza dipende solo dagli $r - 1$ bit intermedi. Se consideriamo tutte le combinazioni come ugualmente possibili, la probabilità che un frame incorretto venga accettato come valido è $1/2^{r+1}$.

Si può anche dimostrare che per un burst di errori di lunghezza superiore a $r + 1$ bit oppure per una serie di diversi errori più corti, la probabilità che un frame incorretto passi senza essere notato è pari a $1/2^r$, assumendo che tutte le distribuzioni di bit siano ugualmente probabili. Certi polinomi sono diventati degli standard internazionali. Quello usato per l'IEEE 802 è:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

Questo polinomio ha molte proprietà utili, fra cui quella che riesce a rilevare tutti i grappoli di errori di lunghezza uguale o minore a 32 e tutti quelli che toccano un numero dispari di bit. I calcoli necessari per calcolare i checksum possono sembrare complicati. Peterson e Brown (1961) hanno invece dimostrato che basta costruire un semplice circuito a registro di shift per calcolare e controllare in hardware tutti i checksum. In pratica questa implementazione hardware è usata in tutte le LAN e, in alcuni casi, anche per le linee punto-punto.

cenni è stato dato per scontato che i frame su cui calcolare i checksum contenessero con distribuzioni casuali. Tutte le analisi degli algoritmi di checksum sono state fatte queste ipotesi. L'analisi di dati reali, però, ha dimostrato che l'ipotesi è decisamente falsa. Ci sono, infatti, alcune circostanze in cui gli errori non rilevati si presentano con frequenza maggiore di quanto si pensasse in passato (Partridge et al. 1995).

Protocolli data link elementari

Procedendo l'argomento dei protocolli, cominceremo guardandone tre di crescente complessità. Per il lettore interessato, un simulatore di questi protocolli e degli altri che seguono è disponibile via Web (vedi la prefazione). Prima di studiare i protocolli, è utile fare alcune assunzioni che sono implicite nel modello di comunicazione. Per cominciare assumiamo che negli strati fisico, data link e network siano presenti dei processi indipendenti che comunicano passandosi dei messaggi uno con l'altro. In molti casi i processi dello strato fisico e dello strato data link gireranno su un processore all'interno di uno stesso chip per l'I/O, mentre lo strato network girerà sulla CPU principale. Sono tuttavia possibili anche altre implementazioni (per esempio i tre processi possono girare all'interno del chip di I/O, oppure gli strati fisico e data link possono essere implementati come routine chiamate dal processo dello strato network). A ogni modo, il fatto di trattare i tre strati come processi separati rende la discussione concettualmente più chiara e anche a enfatizzare l'indipendenza degli strati.

Un assunto fondamentale è che la macchina A voglia mandare un consistente flusso di dati alla macchina B, usando un servizio affidabile e orientato alla connessione. Sivisivamente considereremo il caso in cui B voglia simultaneamente anche mandare dati a A. Assumiamo inoltre che la sorgente A abbia a disposizione una quantità illimitata di dati pronti da trasmettere e quindi non debba mai aspettare che i dati vengano prodotti, perché lo strato network sia sempre pronto a fornire i dati quando lo strato data link della macchina A li richiede (più avanti considereremo anche dei casi in cui questa ipotesi non vale). Nostro schema le macchine non sono soggette a crash, cioè i protocolli che trattiamo non dicono cura degli errori di comunicazione ma non dei problemi causati dai crash dei computer e dei conseguenti reboot.

Così come per lo strato data link, i bit contenuti nei pacchetti che gli arrivano dallo strato network sono tutti da considerare come appartenenti ai dati e quindi da trasferire allo strato network della destinazione. Il fatto che lo strato network possa considerare alcuni dei bit come appartenenti all'intestazione dei pacchetti o altro, non riguarda assolutamente lo strato data link.

Quando lo strato data link accetta un pacchetto, lo incapsula in un frame aggiungendogli un'intestazione (*header*) e una coda (*trailer*) (vedi anche Figura 3.1). Il frame è quindi trasmesso allo strato data link dell'altra macchina. Assumiamo che esista una libreria di programmi che contenga: *to_physical_layer*, per inviare un frame e *from_physical_layer* per riceverne uno. L'hardware che si occupa della trasmissione calcola e aggiunge in coda

il checksum, così che il software allo strato data link non deve preoccuparsi di questa operazione. Il checksum può essere calcolato, per esempio, con l'algoritmo polinomiale discusso in precedenza.

All'inizio della trasmissione la destinazione non ha niente da fare, deve solamente aspettare che succeda qualcosa. Negli esempi di questo capitolo indicheremo che lo strato data link aspetta che accada qualcosa con la chiamata a *wait_for_event(&event)*. Questa procedura ritorna dalla chiamata solo quando è successo qualcosa (per esempio è arrivato un frame). Al ritorno la variabile *event* è valorizzata in modo da far capire che cosa è successo. L'insieme dei possibili eventi differisce a seconda del protocollo e viene definito per ogni implementazione. In una situazione più realistica lo strato data link non entra in un loop infinito in attesa di un evento, come indicato nel precedente esempio, ma invece aspetterà un interrupt. Al suo arrivo interromperà qualunque cosa stesse facendo e si metterà a gestire il frame in arrivo. Nel seguito ignoreremo quest'ultimo dettaglio, che ha a che vedere con il parallelismo interno delle attività dello strato data link. Per semplicità, supporremo di avere un processo dedicato a tempo pieno all'elaborazione di un solo canale.

Quando un frame arriva alla destinazione, l'hardware calcola il checksum. Se il checksum non è corretto (cioè se c'è stato un errore di trasmissione), lo strato data link viene informato (*event = cksum_err*). Se il frame è arrivato senza problemi lo strato data link è altresì informato (*event = frame_arrival*) e lo può acquisire per esaminarlo usando *from_physical_layer*. Quando lo strato data link della destinazione ha prelevato un frame integro, come prima cosa ispeziona le informazioni di controllo nell'intestazione e, se tutto va bene, passa la parte relativa al pacchetto dei dati allo strato network.

C'è un buon motivo per cui lo strato network non deve mai ricevere nessuna parte dell'intestazione dei frame: tenere completamente separati i protocolli data link e network. Se lo strato network non conosce nessun dettaglio del protocollo data link o del formato dei frame, questi ultimi potranno essere cambiati senza richiedere un cambiamento nel software dello strato network. Avere un'interfaccia rigida fra gli strati network e data link semplifica notevolmente la progettazione del software, in quanto rende possibile lo sviluppo e l'evoluzione indipendente dei diversi protocolli coinvolti.

La Figura 3.9 mostra alcune dichiarazioni (in linguaggio C) comuni a molti protocolli discusssi nel seguito. Sono definite cinque strutture dati: *boolean*, *seq_nr*, *packet*, *frame_kind* e *frame*. Un *boolean* può assumere solo due valori: vero e falso. Un *seq_nr* è un piccolo intero usato per numerare i frame per distinguere uno dall'altro. Questi numeri vanno da 0 fino a *MAX_SEQ*, che è definito in ciascun protocollo che lo usa. Un *packet* è l'unità d'informazione che viene scambiata fra lo strato network e il data link della stessa macchina, oppure fra gli strati network di macchine comunicanti. Nel nostro modello contiene *MAX_PKT* bytes, ma in situazioni più realistiche può essere di lunghezza variabile.

Un *frame* è composto da 4 campi: *kind*, *seq*, *ack* e *info*. I primi tre contengono le informazioni di controllo, l'ultimo contiene i dati da trasferire. I campi con le informazioni di controllo sono anche chiamati **intestazione del frame** (*frame header*).

```

#define MAX_PKT 1024
/* determina la dimensione in byte dei pacchetti */

typedef enum {false, true} boolean;
/* tipo booleano*/
typedef unsigned int seq_nr;
/* numeri di sequenza o ack */
typedef struct {unsigned char data[MAX_PKT];} packet; /* definizione del pacchetto */
typedef enum {data, ack, nak} frame_kind;
/* definizione del frame_kind */

typedef struct {
frame_kind kind;
seq_nr seq;
seq_nr ack;
packet info;
frame;
}

/* Aspetta che accada un evento e ritorna il tipo di evento.*/
void wait_for_event(event_type *event);

/* Prende un pacchetto dallo strato network per trasmetterlo sul canale.*/
void from_network_layer(packet *p);

/* Porta l'informazione dal frame appena arrivato verso lo strato network.*/
void to_network_layer(packet *p);

/* Prende un frame dallo strato fisico e lo copia in r.*/
void from_physical_layer(frame *r);

/* Passa il frame allo strato fisico per la trasmissione.*/
void to_physical_layer(frame *s);

/* Fa partire l'orologio e abilita l'evento di timeout.*/
void start_timer(seq_nr k);

/* Ferma l'orologio e disabilita l'evento di timeout.*/
void stop_timer(seq_nr k);

/* Fa partire il timer ausiliario e abilita l'evento ack_timeout.*/
void start_ack_timer(void);

/* Ferma il timer ausiliario e disabilita l'evento di ack_timeout.*/
void stop_ack_timer(void);

/* Abilita lo strato network a scatenare eventi network_layer_ready.*/
void enable_network_layer(void);

/* Proibisce allo strato network di scatenare eventi network_layer_ready.*/
void disable_network_layer(void);

/* La macro inc viene espansa in linea: incrementa k in modo circolare.*/
#define inc(k) if ((k < MAX3SEQ) k = k + 1; else k = 0

```

Figura 3.9. Alcune definizioni necessarie per i protocolli che seguono.

Queste definizioni si trovano nel file protocol.h.

Il campo *kind* dice se ci sono dati nel frame, poiché alcuni protocolli distinguono i frame che contengono solo informazioni di controllo da quelli che contengono anche i dati. I campi *seq* e *ack* sono usati per i numeri di sequenza e per l'acknowledgement, rispettivamente. Il loro uso verrà descritto con maggior dettaglio più avanti. Il campo *info* di un frame dati contiene un singolo pacchetto e non è usato per i frame di controllo. Un'implementazione più realistica potrebbe usare un campo *info* a lunghezza variabile, omettendolo completamente per i frame di controllo.

A questo punto conviene enfatizzare la relazione che intercorre fra un pacchetto e un frame. Lo strato network costruisce i pacchetti prendendo i messaggi dallo strato trasportato e aggiungendo un'intestazione di tipo network. Questo pacchetto viene passato allo strato data link, che lo include nel campo *info* del frame in uscita. Quando il frame arriva a destinazione, lo strato data link estrae il pacchetto dal frame e lo passa allo strato network. In questo modo lo strato network agisce come se le due macchine si stessero scambiando direttamente i pacchetti.

La Figura 3.9 contiene anche un elenco di procedure. Queste sono funzioni di libreria, i cui dettagli dipendono dall'implementazione. Non ci occuperemo qui dei dettagli interni di queste funzioni. La procedura *wait_for_event* contiene un loop infinito e aspetta che accada qualcosa, come abbiamo detto in precedenza. Le procedure *from_network_layer* e *to_network_layer* sono usate dallo strato data link per trasferire i pacchetti, rispettivamente da e verso lo strato network. Analogamente *from_physical_layer* e *to_physical_layer* trasferiscono i dati da e verso lo strato fisico. In altre parole *from_network_layer* e *to_network_layer* si occupano dell'interfaccia fra gli strati 2 e 3, mentre *from_physical_layer* e *to_physical_layer* hanno a che fare con l'interfaccia fra gli strati 1 e 2.

Nella maggior parte dei protocolli si assume che il canale non sia affidabile e quindi in alcune occasioni perda degli interi frame. Per poter ripristinare questi disastri, lo strato data link della sorgente deve far partire internamente un allarme a tempo (timer) ogni volta che invia un frame. Se non viene ricevuta nessuna risposta entro un tempo determinato, l'allarme scatta e lo strato data link riceve un segnale di interrupt.

Nei nostri protocolli questa funzione è gestita facendo in modo che la procedura *wait_for_event* possa ritornare *event = timeout*. Le procedure *start_timer* e *stop_timer* fanno partire e fermare il timer, rispettivamente. L'evento di timeout si può avere solo se il timer è stato fatto partire. È esplicitamente permesso chiamare *start_timer* mentre un timer sta girando. Tale chiamata semplicemente esegue un reset del timer, così che il successivo timeout avverrà dopo un intero intervallo del timer (a patto che nel frattempo non avvenga un altro reset o il timer sia fermato).

Le procedure *start_ack_timer* e *stop_ack_timer* controllano un timer ausiliario per generare gli acknowledge sotto certe condizioni.

Le procedure *enable_network_layer* e *disable_network_layer* sono usate in protocolli più sofisticati, dove non si assume che lo strato network abbia sempre pacchetti da inviare. Quando lo strato data link abilita lo strato network, vuol dire che lo strato network è autorizzato a mandare interrupt se ha pacchetti da inviare. Indichiamo questo con *event = network_layer_ready*. Quando lo strato network è disabilitato, invece, non può causare even-

ti di questo tipo. Usando con attenzione le funzioni per abilitare e disabilitare lo strato network, lo strato data link può evitare di essere sommerso di pacchetti dati che potrebbero riempire il suo spazio di buffer.

I numeri di sequenza dei frame sono sempre compresi fra 0 e *MAX_SEQ* (estremi inclusi), dove *MAX_SEQ* dipende dal protocollo. Spesso è necessario far avanzare il numero di sequenza con incrementi di 1 e in modo circolare (cioè *MAX_SEQ* è seguito da 0). La macro *inc* esegue tale operazione. È stata definita come una macro per motivi di performance, in quanto si trova sul percorso critico; così facendo viene sostituita in linea risparmiando il costo della chiamata a una funzione. Infatti, come vedremo più avanti, il fattore limitante della performance della rete è spesso dato dal protocollo, quindi definendo operazioni semplici (come l'incremento) tramite macro, non cambia la leggibilità del codice ma aumentano le performance. Inoltre, siccome *MAX_SEQ* ha valori differenti per diversi protocolli, scrivendo l'incremento sotto forma di macro risulta possibile gestire tutti i protocolli nello stesso codice binario senza conflitti. Questa possibilità è utile per il simulatore.

Le dichiarazioni della Figura 3.9 fanno parte di tutti i protocolli che descriveremo in seguito. Sono state estratte a parte in modo da risparmiare spazio e avere un utile riferimento, comunque dobbiamo pensarle come appartenenti a tutte le implementazioni di protocolli. In C questo si realizza inserendo le definizioni in uno speciale file d'intestazione, in questo caso *protocol.h*, e utilizzando il comando *#include* del preprocessore C, che permette di includere il file di intestazione nel file sorgente dei protocolli.

3.3.1 Un protocollo simplex senza restrizioni

Come primo esempio considereremo il protocollo più semplice possibile: i dati sono trasmessi in una sola direzione (simplex), gli strati network della sorgente e della destinazione sono sempre pronti, il tempo per elaborare i dati può essere ignorato ed è disponibile un buffer infinito. Infine, poniamo l'ipotesi più forte: il canale di comunicazione non perde mai nessun frame. Questo è chiaramente un protocollo non realistico, per cui lo chiameremo "utopia", come mostrato nella Figura 3.10.

Il protocollo consiste di due distinte procedure, un mittente e un destinatario. Il mittente gira nello strato data link della macchina sorgente, mentre il destinatario gira nello strato data link della destinazione. Non usiamo, in questo caso, né numeri di sequenza né acknowledgement, così che *MAX_SEQ* non serve. L'unico tipo di evento possibile è *frame_arrival* (cioè l'arrivo di un frame intatto).

Il mittente invia dati dall'interno di un loop infinito costruito con l'istruzione while, semplicemente mandando fuori i dati alla massima velocità che gli è possibile. Nel corpo del loop troviamo tre azioni: andare a prendere il pacchetto dallo strato network (che sarà sempre pronto a fornirlo), costruire un frame in uscita usando la variabile *s*, instradare il frame. Questo protocollo usa solamente il campo *info* del frame, in quanto gli altri campi hanno a che vedere con il controllo degli errori e del flusso, ma non ci sono errori o restrizioni di flusso in questa semplificazione.

/ Il protocollo 1 (utopia) permette la trasmissione dei dati in una sola direzione, dalla sorgente alla destinazione. Si assume che il canale di comunicazione sia privo di errori e che la destinazione sia in grado di elaborare tutto l'input a velocità infinita. Di conseguenza, la sorgente semplicemente invia i dati alla massima velocità possibile da dentro un loop infinito. */*

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
    frame s;
    packet buffer;

    while (true) {
        from_network_layer(&buffer);
        s.info = buffer;
        to_physical_layer(&s);
    }
}

void receiver1(void)
{
    frame r;
    event_type event;

    while (true) {
        wait_for_event(&event);
        from_physical_layer(&r);
        to_network_layer(&r.info);
    }
}
```

/ buffer per i frame in uscita */
/* buffer per i pacchetti in uscita */

/* va a prendere qualcosa da trasmettere */
/* copia il buffer in s per trasmetterlo */
/* trasmette il buffer */
/* Tomorrow, and tomorrow, and tomorrow,
 Creeps in this petty pace from day to day
 To the last syllable of recorded time.
 - Macbeth, V, v */

/* riempito da wait, ma non è implementato */

/* la sola possibilità è frame_arrival */
/* va a prendere il frame in arrivo */
/* passa i dati allo strato network */*

Figura 3.10. Un protocollo simplex senza restrizioni.

Il destinatario ha una struttura ugualmente semplice. Inizialmente aspetta che succeda qualcosa, e l'unica possibilità è l'arrivo di un frame integro.

In seguito il frame arriva e la procedura *wait_for_event* ritorna, con *event* impostato a *frame_arrival* (che comunque è ignorato). La chiamata a *from_physical_layer* rimuove dal buffer hardware il frame appena arrivato e lo mette nella variabile *r*, dove il destinatario lo può prendere. Infine, la porzione di frame che contiene i dati viene passata allo strato network mentre lo strato data link torna ad aspettare un nuovo frame; in pratica sospende la sua attività finché tale frame non arriva.

3.3.2 Un protocollo simplex stop-and-wait

Adesso rimuoviamo l'ipotesi meno realistica usata nel protocollo 1: la possibilità, da parte dello strato network, di elaborare i dati in arrivo a velocità infinita (o, in modo equivalente, la presenza nello strato data link di una quantità infinita di spazio buffer in cui immagazzinare i frame in arrivo prima di elaborarli). Si assume di nuovo che il canale di comunicazione sia senza errori e che il traffico sia di tipo simplex.

Il principale problema che dobbiamo trattare in questo caso è come prevenire che il mittente inondi il ricevente, con dati trasmessi a velocità maggiore di quanto quest'ultimo sia in grado di elaborare. In sostanza, se il ricevente richiede un tempo Dt per eseguire *from_physical_layer* più *to_network_layer*, il mittente deve trasmettere con una frequenza inferiore a un frame per intervallo Dt . Inoltre, se assumiamo che non ci siano dei meccanismi di buffer o accodamento nell'hardware del ricevente, il mittente non deve mai trasmettere un nuovo frame fino all'istante in cui quello precedente viene prelevato da *from_physical_layer*, altrimenti il nuovo frame sarebbe scritto sopra a quello vecchio.

In certe circostanze particolari (per esempio, per trasmissioni sincrone e uno strato data link della destinazione completamente dedicato a elaborare la linea di input) si può far sì che il mittente semplicemente inserisca un ritardo nel protocollo 1 per rallentarlo a sufficienza in modo da non inondare il ricevente. In realtà, però, ogni strato data link dovrà prendersi cura di più linee e quindi l'intervallo di tempo fra quando un frame arriva e quando viene trattato può variare considerevolmente. Se nel disegnare la rete si riesce a calcolare il comportamento del caso peggiore per il ricevente, allora si può programmare il mittente in modo che trasmetta così lentamente che, anche se ogni frame subisce il massimo ritardo, non ci saranno comunque problemi di riempimento dei buffer in arrivo. Il problema di questo approccio è che è troppo conservativo. Infatti porta a un'utilizzazione della banda disponibile ben al di sotto delle possibilità ottimali, fatta eccezione per la situazione dove per il ricevente il caso migliore e peggiore sono praticamente uguali (cioè la variazione dei tempi di risposta dello strato data link è piccola).

Una soluzione più generale a questo dilemma consiste nel far sì che il ricevente mandi un feedback al mittente. Dopo aver passato un pacchetto allo strato network, il ricevente manda indietro al mittente un piccolo frame senza dati (*dummy*), che dà al mittente il permesso di inviare il frame successivo. Dopo aver mandato un frame, il mittente è obbligato dal protocollo ad aspettare finché non riceve il pacchetto dummy (in questo caso l'*acknowledgement*). Usare una retroazione generata dal ricevente per far sapere al mittente quando è autorizzato a trasmettere altri dati è un esempio del controllo di flusso menzionato in precedenza.

Protocolli in cui il mittente manda un frame e poi aspetta l'*acknowledgement* prima di procedere sono chiamati **stop-and-wait** (fermati e aspetta). La Figura 3.11 mostra un esempio di protocollo simplex di tipo stop-and-wait.

Anche se il traffico di questo esempio è simplex (cioè va solamente dalla sorgente alla destinazione) i frame viaggiano in entrambe le direzioni. Conseguentemente, il canale di comunicazione fra i due strati data link deve essere capace di trasmettere informazioni in modo bidirezionale, ma questo protocollo impone una rigida alternanza del flusso: prima il mittente manda un frame, poi il ricevente manda un frame, poi è di nuovo il turno del

/* Il protocollo 2 (stop-and-wait) fornisce anch'esso un flusso unidirezionale di dati dalla sorgente alla destinazione. Si assume ancora che il canale di comunicazione sia privo di errori, come per il protocollo 1. Questa volta, però, la destinazione ha buffer di capacità finita e velocità di elaborazione finita. Perciò il protocollo deve esplicitamente prevenire la possibilità che la destinazione venga sommersa di dati da parte della sorgente, nel caso in cui questa trasmetta a una velocità superiore a quanto la destinazione riesce a elaborare */

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
    frame s;                                /* buffer per i frame in uscita */
    packet buffer;                          /* buffer per i pacchetti in uscita */
    event_type event;                      /* frame_arrival è la sola possibilità */
                                         /* frame_arrival è l'unica possibilità */

    while (true) {
        from_network_layer(&buffer);
        s.info = buffer;
        to_physical_layer(&s);
        wait_for_event(&event);
                                         /* va a prendere qualcosa da inviare */
                                         /* copialo in s per la trasmissione */
                                         /* invia allo strato fisico */
                                         /* non procedere prima di aver ricevuto l'ok */
    }
}

void receiver2(void)
{
    frame r, s;                                /* buffer per i frame */
    event_type event;                          /* frame_arrival è l'unica possibilità */
                                         /* frame_arrival è l'unica possibilità */
                                         /* va a prendere il frame in arrivo */
                                         /* passa i dati allo strato network */
                                         /* manda un frame qualsiasi per svegliare la sorgente */
    while (true) {
        wait_for_event(&event);
        from_physical_layer(&r);
        to_network_layer(&r.info);
        to_physical_layer(&s);
    }
}
```

Figura 3.11. Un protocollo simplex di tipo stop-and-wait.

mittente, e così via. Un canale fisico half-duplex può bastare per questo tipo di comunicazione.

Come nel caso del protocollo 1, il mittente comincia prendendo un pacchetto dallo strato network, lo usa per costruire un frame e lo spedisce. Adesso, a differenza del protocollo 1, il mittente deve aspettare che arrivi un frame di acknowledgement prima di continuare il ciclo e andare a prendere il pacchetto successivo dallo strato network. Lo strato data link della sorgente non deve neanche esaminare il frame in arrivo, in quanto c'è una sola possibilità: il frame è un acknowledgement.

L'unica differenza fra *receiver1* e *receiver2* è che dopo aver mandato il pacchetto allo strato network, *receiver2* manda indietro un frame di acknowledgement al mittente prima di entrare di nuovo nel loop di attesa. Solamente l'arrivo del frame di acknowledgement è importante per il mittente, quindi la destinazione non dovrà preoccuparsi di mettervi informazioni particolari.

3.3.3 Un protocollo simplex per canali rumorosi

Consideriamo adesso una situazione normale, in cui il canale di comunicazione commette degli errori. I frame possono essere danneggiati o addirittura persi completamente. Supponiamo che se un frame è danneggiato durante la trasmissione, l'hardware della destinazione riesce ad accorgersene calcolando il checksum. Nell'eventualità remota che il frame sia danneggiato in modo tale che il checksum risulti comunque corretto, questo protocollo (come tutti gli altri) può fallire, cioè potrebbe trasferire allo strato network un pacchetto erroneo.

A prima vista può sembrare che una variazione del protocollo 2 possa aiutarci: aggiungere un timer. L'idea è che quando la sorgente invia un frame, la destinazione risponde con un acknowledgement solo quando il frame è stato ricevuto correttamente. Se arriva un frame danneggiato, allora la destinazione semplicemente lo scarta. Dopo un po' di tempo il timer della sorgente scattano, e il frame viene inviato di nuovo. L'operazione si può ripetere finché il frame non arriva intatto a destinazione.

Lo schema appena descritto contiene però un errore fatale. Il lettore è invitato a pensarci e a scoprire che cosa può andare storto con questo modo di operare, prima di continuare a leggere.

Per vedere cosa può andare storto, ricordiamo che il compito dei processi allo strato data link è quello di fornire ai processi network la possibilità di comunicare in modo trasparente e senza errori. Lo strato network della macchina A passa una serie di pacchetti al suo strato data link, che deve preoccuparsi che una serie identica di pacchetti arrivi allo strato network della macchina B, tramite il rispettivo strato data link. In particolare, lo strato network di B non ha nessun modo per sapere se un pacchetto sia stato perso o duplicato, perciò lo strato data link deve poter garantire che nessuna combinazione di errori, per quanto improbabile, possa causare la duplicazione di pacchetti.

Consideriamo il seguente scenario.

1. Lo strato network di A manda il pacchetto 1 al suo strato data link. Il pacchetto viene ricevuto correttamente da B e passato allo strato network di B. B manda un frame di acknowledgement ad A.
2. Il frame di acknowledgement viene perso completamente. Non arriva proprio. La vita sarebbe più facile se il canale si perdesse solo dei frame di dati, ma purtroppo non è così: il canale non discrimina i frame fra di loro.
3. Dopo un po' il timer allo strato data link di A va in timeout. Non avendo ricevuto un acknowledgement, A assume erroneamente che il frame dati non sia arrivato integro a destinazione e quindi lo invia di nuovo.

4. Il frame duplicato arriva integro allo strato data link di B e li passa, inavvertitamente, allo strato network. Se A sta trasmettendo un file a B, quella parte del file sarà duplicata, cioè la copia del file ottenuta da B sarà sbagliata e l'errore passerà senza essere notato. In altre parole il protocollo avrà fallito.

Quello che serve, quindi, è un metodo che la destinazione possa usare per distinguere i frame che vede per la prima volta dalle ritrasmissioni. Il metodo ovvio è quello d'inserire un numero di sequenza nell'intestazione del frame quando viene inviato. La destinazione può quindi controllare i numeri di sequenza dei pacchetti man mano che arrivano e quindi di riuscire a sapere se si tratta di un frame nuovo o di un duplicato da scartare.

Visto che è desiderabile avere un'intestazione per i frame, la domanda successiva sarà: qual è il minor numero di bit necessari per il numero di sequenza? La sola possibile ambiguità in questo protocollo è fra il frame m e il suo successore diretto $m + 1$. Se il frame m viene perso o danneggiato, la destinazione non manderà l'acknowledgement e quindi la sorgente continuerà a trasmetterlo. Una volta che il frame è stato ricevuto correttamente, la destinazione invierà l'acknowledgement alla sorgente. È qui che emerge il potenziale problema. A seconda che il frame di acknowledgement arrivi alla sorgente oppure no, questi continuerà inviando il frame $m + 1$ oppure ritrasmettendo il frame m .

L'evento che spinge la sorgente a cominciare a inviare il frame $m + 2$ è l'arrivo dell'acknowledgement del frame $m + 1$. Questo implica che anche m è già stato ricevuto, così come l'acknowledgement relativo a m (altrimenti la sorgente non avrebbe neanche cominciato a inviare $m + 1$). La conseguenza di questo ragionamento è che la possibile ambiguità da risolvere per ogni frame riguarda solamente il frame stesso e il suo immediato predecessore o successore, e non è mai fra il suo predecessore e il suo successore.

Un numero di sequenza con un solo bit è quindi sufficiente. A ogni istante la destinazione si aspetta di ricevere un particolare numero di sequenza. Ogni frame in arrivo con il numero di sequenza sbagliato viene scartato come duplicato; quando invece arriva un frame con il numero di sequenza corretto, viene accettato e passato allo strato network. A questo punto il numero di sequenza atteso è incrementato in aritmetica modulo 2 (cioè lo 0 diventa 1 e viceversa).

Un esempio di questo tipo di protocollo si può trovare nella Figura 3.12. Protocolli in cui la sorgente aspetta un acknowledgement positivo prima di avanzare nella trasmissione sono spesso chiamati **PAR** (*Positive Acknowledgement with Retransmission*, risposta positiva con ritrasmissione) oppure **ARQ** (*Automatic Repeat reQuest*, richiesta automatica di ripetizione). Come nel caso del protocollo 2, anche in questo i dati sono trasmessi in una sola direzione.

Il protocollo 3 differisce dai suoi predecessori per il fatto che sorgente e destinazione hanno delle variabili che devono essere ricordate mentre lo strato data link è in stato di attesa. La sorgente memorizza il numero di sequenza del successivo frame da inviare dentro a *next_frame_to_send*, la destinazione invece memorizza il numero di sequenza che si aspetta di ricevere dentro a *frame_expected*. Ogni protocollo ha una piccola fase d'inizializzazione prima di entrare nel loop infinito.

```

/* Il protocollo 3 (par) permette di gestire il flusso di dati unidirezionale su un canale non affidabile. */

#define MAX_SEQ 1           /* deve essere a 1 per il protocollo 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
    seq_nr next_frame_to_send;
    frame s;
    packet buffer;
    event_type event;

    next_frame_to_send = 0;
    from_network_layer(&buffer);
    while (true) {
        s.info = buffer;
        s.seq = next_frame_to_send;
        to_physical_layer(&s);
        start_timer(s.seq);
        wait_for_event(&event);
        if (event == frame_arrival) {
            from_physical_layer(&s);
            if (s.ack == next_frame_to_send) {
                stop_timer(s.ack);
                from_network_layer(&buffer);
                inc(next_frame_to_send);
            }
        }
    }
}

void receiver3(void)
{
    seq_nr frame_expected;
    frame r, s;
    event_type event;

    frame_expected = 0;
    while (true) {
        wait_for_event(&event);
        if (event == frame_arrival) {
            from_physical_layer(&r);
            if (r.seq == frame_expected) {
                to_network_layer(&r.info);
                inc(frame_expected);
            }
            s.ack = 1 - frame_expected;
            to_physical_layer(&s);
        }
    }
}

```

Figura 3.12. Un protocollo con acknowledgement positivo e retransmission.

Dopo la trasmissione del frame, la sorgente fa partire il timer. Se il timer era già avviato, esegue un reset e lo fa ripartire. L'intervallo di tempo dovrebbe essere scelto in modo da permettere ai frame di arrivare a destinazione ed esservi elaborati (consideriamo il caso peggiore come tempo di elaborazione), infine il frame di acknowledgement deve avere il tempo di tornare indietro. Solo dopo un simile intervallo di tempo è ragionevole assumere che o il frame o il suo acknowledgement siano stati persi ed è quindi necessario inviare un duplicato. Se l'intervallo di timeout è impostato a un valore troppo breve, la sorgente invierà dei frame inutili. Questi frame non incideranno sulla correttezza del protocollo, ma avranno delle ricadute sulle performance. Dopo aver trasmesso un frame e fatto partire il timer, la sorgente aspetta che accada qualcosa. Le possibilità sono solamente tre: l'arrivo di un frame di acknowledgement intatto, l'arrivo di un frame di acknowledgement con problemi, il timeout del timer. Nel primo caso la sorgente prende il successivo pacchetto dallo strato network e lo mette nel buffer, scrivendolo al posto del pacchetto precedente. Se invece arriva un frame di acknowledgement danneggiato oppure non arriva nessun frame, allora, né il buffer, né il numero di sequenza vengono cambiati e viene inviato un duplicato. Quando un frame valido arriva alla destinazione, il suo numero di sequenza viene controllato per vedere se si tratta di un duplicato. Se non è un duplicato, il frame viene accettato e passato allo strato network, quindi è generato un acknowledgement. I frame duplicati o danneggiati non vengono passati allo strato network.

3.4 Protocolli sliding window

Nei protocolli precedenti i frame di dati venivano trasmessi solamente in una direzione. Nella maggior parte delle situazioni reali, risulta invece necessario poter trasmettere in entrambe le direzioni. Un modo per ottenere una trasmissione di dati full-duplex è quello di aver due canali di trasmissione e usare ciascuno di essi per una trasmissione simplex (in direzioni diverse). In questo modo si ottengono due canali fisici separati, ognuno con un canale "di andata" (per i dati) e uno "di ritorno" (per gli acknowledgement). In entrambi i casi la banda del canale di ritorno risulta quasi interamente sprecata. In effetti, l'utente paga per due circuiti, ma usa solo la capacità di uno. Un'idea migliore consiste nell'usare lo stesso circuito per entrambe le direzioni di comunicazione. Dopotutto, già nei protocolli 2 e 3 si usa lo stesso circuito per trasmettere i frame in entrambe le direzioni, inoltre il canale di andata e quello di ritorno hanno la stessa capacità. In questo modello i frame dati che passano da A a B sono mescolati a quelli di acknowledgement da A a B. Guardando al campo *kind* nell'intestazione del frame in arrivo, la destinazione può determinare se si tratta di dati oppure acknowledgement. Dopo aver inframmezzato dati e frame di controllo sullo stesso canale, possiamo aggiungere un'ulteriore miglioria. Quando arriva un frame di dati, la destinazione non invia subito un frame di controllo separato, ma aspetta che lo strato network gli passi il successivo pacchetto. L'acknowledgement viene aggiunto al frame di dati in uscita usando il campo *ack* nell'intestazione del frame. In questo modo l'acknowledgement si procura un passaggio gratis insieme al successivo frame dati trasmesso. La tecnica che consiste nel ritardare gli acknowledgement in uscita in modo da agganciarli al successivo frame di dati è chiamata **piggy-backing** (letteralmente "trasportare in groppa").

Il vantaggio principale del piggybacking rispetto agli acknowledgement separati, sta nel miglior uso della banda disponibile. Il campo *ack* nell'intestazione del frame costa solamente alcuni bit, mentre costruire un frame separato richiederebbe un'intestazione, un acknowledgement e un checksum. Inoltre un minor numero di frame inviati significa anche un minor numero di interrupt di tipo "frame in arrivo", e quindi magari anche un minor numero di buffer nella destinazione, a seconda di come è strutturato il software della destinazione. Nel prossimo protocollo che esamineremo il campo piggyback costerà solo 1 bit nell'intestazione del frame, ma è difficile che possa costare più di qualche bit.

A ogni modo il campo piggyback introduce una complicazione non presente nei protocolli con acknowledgement separato. Quanto a lungo deve aspettare lo strato data link per avere un pacchetto su cui effettuare il piggybacking dell'acknowledgement? Se lo strato data link aspetta più a lungo del periodo di timeout della sorgente, il pacchetto verrà ritrasmesso, rendendo vano l'acknowledgement. Se lo strato data link fosse un oracolo e potesse predire il futuro, saprebbe quanto tempo intercorrerà per avere il prossimo pacchetto dati dallo strato network e quindi potrebbe decidere se è meglio aspettarlo o inviare subito un acknowledgement separato. Ovviamente lo strato data link non può predire il futuro, così è necessario prevedere uno schema specifico per questa situazione, come per esempio aspettare alcuni millisecondi. Se il nuovo pacchetto arriva rapidamente, viene fatto il piggybacking dell'acknowledgement. Altrimenti, se alla fine del periodo di attesa il pacchetto non è arrivato, lo strato data link invia un frame di acknowledgement separato. I successivi tre protocolli sono di tipo bidirezionale e appartengono alla classe chiamata **sliding window** (letteralmente "finestra scorrevole"). Le differenze fra i tre protocolli riguardano l'efficienza, la complessità e i requisiti in termini di buffer, come discuteremo nel seguito. In questi, come in tutti i protocolli sliding window, ogni frame in uscita contiene un numero di sequenza che va da 0 fino a un valore massimo. Il massimo di solito è $2^n - 1$, così che il numero di sequenza entra esattamente in un campo di n bit. I protocolli stop-and-wait sliding window a 1 bit hanno $n = 1$, restringendo i numeri di sequenza a 0 e 1. Esistono delle versioni più sofisticate che usano valori n arbitrari. L'essenza dei protocolli sliding window è che, a ogni istante, la sorgente tiene traccia di un insieme di numeri di sequenza corrispondenti ai frame che è autorizzata a inviare. Si dice che questi frame si trovano nella **finestra di invio** [NdR - si noti però che una finestra di invio vuota non significa che non è possibile inviare alcun frame. Si veda nel testo la definizione formale della finestra di invio e la relativa discussione]. In modo analogo la destinazione tiene traccia della **finestra di ricezione**, che corrisponde all'insieme dei frame che può accettare. Le finestre d'invio e di ricezione non devono necessariamente avere gli stessi limiti inferiori o superiori e neppure la stessa dimensione. In alcuni protocolli tali finestre hanno delle dimensioni fisse, ma in altri possono avere dimensioni variabili nel tempo. Anche se questi protocolli danno più libertà allo strato data link per quanto riguarda l'ordine in cui mandare o ricevere dei frame, sicuramente non abbiamo abbandonato la regola che impone al protocollo di passare i pacchetti allo strato network della destinazione nello stesso ordine in cui erano stati passati allo strato data link della sorgente. Non abbiamo neanche abbandonato l'ipotesi che il canale fisico di comunicazione sia simile a un cavo, cioè che debba trasferire i frame nell'ordine in cui sono stati inviati.

Il numero di sequenza nella finestra d'invio rappresenta i frame che sono già stati trasmessi correttamente oppure che sono in transito, ma non hanno ancora ricevuto l'acknowledgement corrispondente. Quando un nuovo pacchetto arriva dallo strato network, gli viene assegnato il numero di sequenza successivo in ordine crescente e il limite superiore della finestra è incrementato di uno. Quando invece arriva un acknowledgement, s'incrementa di uno il limite inferiore della finestra. In questo modo la finestra continua a mantenere la lista dei frame che ancora necessitano di acknowledgement. La Figura 3.13 mostra un esempio.

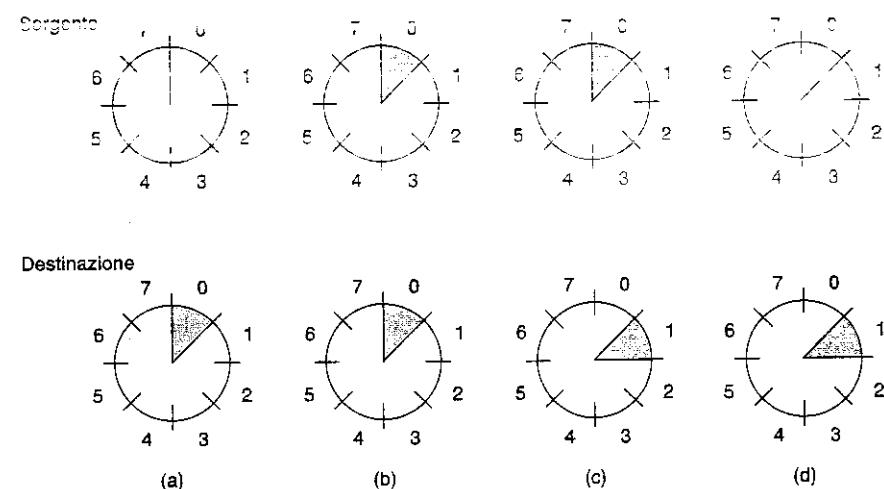


Figura 3.13. Una sliding window con dimensione 1 e numero di sequenza a 3 bit.

- (a) Inizialmente.
- (b) Dopo l'invio del primo frame.
- (c) Dopo la ricezione del primo frame.
- (d) Dopo la ricezione del primo acknowledgement.

Visto che i frame che sono all'interno della finestra d'invio potrebbero andare persi o danneggiati durante la trasmissione, la sorgente deve mantenere tutti questi frame nella sua memoria per l'eventuale ritrasmissione. Quindi, se la massima dimensione della finestra è n , la sorgente ha bisogno di avere n buffer per contenere i frame in attesa di acknowledgement. Se la finestra cresce fino alla sua massima dimensione, lo strato data link è costretto a chiudere forzatamente lo strato network finché non si libera almeno un buffer. La finestra di ricezione tiene traccia dei frame che possono essere accettati dalla destinazione. Ogni frame al di fuori della finestra verrebbe rigettato senza commenti. Quando viene ricevuto un frame con numero di sequenza uguale al limite inferiore della finestra, avvengono le seguenti cose: il frame è passato allo strato network, viene generato un acknowledgement e la finestra è ruotata di uno. Diversamente dalla finestra d'invio, la finestra di ricezione rimane sempre alla sua dimensione iniziale. Notiamo che fissare una dimensione della finestra pari a 1 significa stabilire che lo strato data link accetta i frame solamente in ordine, mentre con finestre più larghe questo non è più vero. Lo strato net-

work riceve sempre i dati nell'ordine corretto, indipendentemente dalla dimensione della finestra nello strato data link.

La Figura 3.13 mostra un esempio di finestra con dimensione massima uguale a 1. Inizialmente non ci sono frame in uscita, quindi i limiti inferiore e superiore della finestra d'invio sono uguali, ma con il progredire del tempo la situazione si evolve come mostrato.

3.4.1 Un protocollo sliding window a 1 bit

Prima di affrontare il caso generale, esaminiamo un protocollo sliding window con una finestra di dimensione massima uguale a 1. Un protocollo simile usa il metodo stop-and-wait, il che significa che prima la sorgente trasmette una frame e poi aspetta di ricevere il corrispondente acknowledgement per poter continuare a trasmettere il successivo frame.

La Figura 3.14 mostra un protocollo di questo tipo. Come per gli altri protocolli, nella prima parte vengono definite alcune variabili. *Next_frame_to_send* dice quale frame la sorgente vuole trasmettere. Analogamente, *frame_expected* indica il frame che la destinazione si aspetta di ricevere. In entrambi casi le sole possibilità sono 0 e 1.

In circostanze normali, uno dei due strati data link parte e trasmette il primo frame. In altre parole, solo uno dei programmi dello strato data link dovrebbe contenere all'esterno del ciclo principale le chiamate alle procedure *to_physical_layer* e *start_timer*. Se invece entrambi gli strati data link partono simultaneamente, allora abbiamo una situazione particolare, come discusso nel seguito. La macchina che parte prende il primo pacchetto dal suo strato network, lo usa per costruire un frame e lo invia. Quando questo (o un altro) frame arriva a destinazione, lo strato data link della destinazione controlla per vedere se è un duplicato (come nel protocollo 3). Se il frame è quello che la destinazione si aspettava, lo passa allo strato network e la finestra della destinazione viene portata avanti.

Il campo di acknowledgement contiene il numero dell'ultimo frame ricevuto senza errori. Se questo numero concorda con il numero di sequenza che la sorgente vuole trasmettere, allora la sorgente sa di aver finito il lavoro con il frame memorizzato nel *buffer* e può prendere il successivo pacchetto dal suo strato network. Se i numeri di sequenza non concordano, allora deve provare a inviare di nuovo lo stesso frame. Quando un frame viene ricevuto, un altro frame è inviato indietro.

Esaminiamo il protocollo 4 per vedere quanto è robusto e capace di affrontare anche scenari patologici. Prendiamo la situazione in cui il computer A tenta di inviare il frame 0 al computer B, mentre B vuole mandare il frame 0 ad A. Supponiamo che A mandi il frame a B, ma l'intervallo di timeout di A è leggermente troppo corto. Di conseguenza, A può generare timeout a raffica mandando una serie di frame identici, tutti con *seq* = 0 e *ack* = 1.

Quando il computer B riceve il primo frame valido lo accetta e imposta la variabile *frame_expected* a 1. Tutti i frame successivi saranno rifiutati perché B ora si aspetta dei frame con numero di sequenza 1 e non 0. Inoltre, visto che i duplicati hanno *ack* = 1 e B continua ad aspettare l'acknowledgement di 0, B non prenderà altri pacchetti dal suo strato network.

Dopo aver rifiutato ogni frame duplicato in arrivo, B manda ad A un frame che contiene *seq* = 0 e *ack* = 0. Infine, uno di questi arriva correttamente ad A e fa sì che A mandi il pac-

/* Il protocollo 4 (sliding window) è bidirezionale. */

```
#define MAX_SEQ 1                                /* deve essere 1 per il protocollo 4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void protocol4 (void)
{
    seq_nr next_frame_to_send;                  /* 0 or 1 only */
    seq_nr frame_expected;                     /* 0 or 1 only */
    frame r, s;                               /* variabili di servizio */
    packet buffer;                           /* pacchetto corrente in invio */

    next_frame_to_send = 0;
    frame_expected = 0;
    from_network_layer(&buffer);
    s.info = buffer;
    s.seq = next_frame_to_send;
    s.ack = 1 - frame_expected;
    to_physical_layer(&s);
    start_timer(s.seq);

    while (true) {
        wait_for_event(&event);
        if (event == frame_arrival) {
            from_physical_layer(&r);
            if (r.seq == frame_expected) {
                to_network_layer(&r.info);
                inc(frame_expected);
            }
            if (r.ack == next_frame_to_send) {
                stop_timer(r.ack);
                from_network_layer(&buffer);
                inc(next_frame_to_send);
            }
        }
        s.info = buffer;
        s.seq = next_frame_to_send;
        s.ack = 1 - frame_expected;
        to_physical_layer(&s);
        start_timer(s.seq);
    }
}
```

/* prossimo frame nel flusso in uscita */
/* il frame atteso come prossimo */
/* prende un pacchetto dallo strato network */
/* prepara per mandare il frame iniziale */
/* inserisce il numero di sequenza nel frame */
/* acknowledgement in piggyback */
/* trasmette il frame */
/* fa partire il timer */

/* frame_arrival, cksum_err, o timeout */
/* un frame è arrivato senza danni */
/* va a prenderlo */
/* gestisce lo stream in ingresso */
/* passa il pacchetto allo strato network */
/* inverte num. seq. atteso come prossimo */

/* gestisce il flusso di frame in uscita */
/* ferma il timer */
/* successivo pacchetto dallo strato network */
/* inverte il numero di sequenza della sorgente */

/* costruisce il frame in uscita */
/* inserisce il numero di sequenza */
/* il num. seq. dell'ultimo frame ricevuto */
/* trasmette un frame */
/* fa partire il timer */

Figura 3.14. Un protocollo sliding window a 1 bit.

chetto successivo. Nessuna combinazione di pacchetti persi o di timeout prematuri può causare la trasmissione di pacchetti duplicati allo strato network, la perdita di pacchetti oppure situazioni di deadlock.

Una situazione particolare si ha quando entrambi i soggetti mandano simultaneamente un pacchetto iniziale. La difficoltà della sincronizzazione è illustrata dalla Figura 3.15: nella parte (a) viene mostrata la normale operatività del protocollo, in (b) si vede invece la situazione particolare. Se B aspetta il primo frame da A prima di inviarne uno dei suoi, la sequenza è come mostrato in (a), e ogni frame viene accettato. Se invece A e B iniziano simultaneamente la comunicazione, i primi frame si incrociano e lo strato data link raggiunge la situazione (b). In (a) l'arrivo di ogni frame porta un nuovo pacchetto per lo strato network. In (b) metà dei frame contengono dei duplicati, anche quando non ci sono errori di trasmissione. Situazioni simili possono accadere come risultato di timeout prematuri, anche se un soggetto inizia a trasmettere nettamente prima dell'altro. In effetti, se accadono molti timeout prematuri i frame possono essere trasmessi anche tre o più volte.

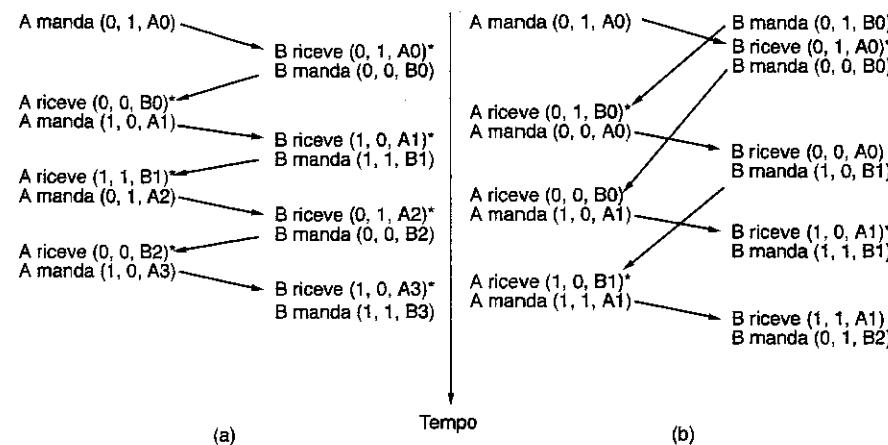


Figura 3.15. Due scenari per il protocollo 4. (a) Caso normale. (b) Caso con problemi. La notazione è (seq, ack, packet number). Un asterisco indica il momento in cui lo strato network accetta un pacchetto.

3.4.2 Un protocollo che usa go back n

Fino ad ora abbiamo tacitamente supposto che il tempo necessario a trasmettere un frame alla destinazione sommato al tempo per l'acknowledgement sia trascurabile. In alcuni casi questa ipotesi è chiaramente falsa. In queste situazioni i tempi lunghi di tracitto dei frame possono avere pesanti implicazioni in termini di efficienza dell'uso della banda. Come esempio consideriamo una canale satellitare a 50 kbps con un ritardo di propagazione totale di 500 millisecondi. Immaginiamo di usare il protocollo 4 per inviare dei frame da 1.000 bit. La sorgente comincia a trasmettere a $t = 0$. Al tempo $t = 20$ millisecondi il frame è stato completamente trasmesso. Il frame viene ricevuto in modo completo dalla destinazione

non prima di $t = 270$ millisecondi e l'acknowledgement ritorna indietro non prima di $t = 520$ millisecondi. Questo nella migliore situazione, in cui non ci sono ritardi da parte della destinazione e il frame di acknowledgement è corto; ciò significa che la sorgente è rimasta bloccata per $500/520$ o 96% del tempo. In altre parole, è stato usato solo il 4% della banda disponibile. Chiaramente la combinazione di tempi di transito lunghi, larga banda e frame corti è disastrosa in termini di efficienza.

Il problema descritto sopra può essere visto come una conseguenza della regola che richiede alla sorgente di aspettare l'acknowledgement della destinazione prima di mandare il frame successivo. Se alleggeriamo questo requisito possiamo ottenere un'efficienza molto migliore. In pratica, la soluzione sta nel permettere alla sorgente di mandare un numero di frame w maggiore di uno, prima di bloccarsi. Con una scelta appropriata di w la sorgente è in grado di trasmettere per un tempo pari al tempo totale di transito senza arrivare a riempire la finestra. Nell'esempio precedente, la sorgente comincia a mandare il frame 0 (come prima) e continua a trasmettere. Quando è arrivata ad aver spedito 26 frame siamo a $t = 520$ e arriva l'acknowledgement del frame 0. Da lì in avanti gli acknowledgement arrivano ogni 20 millisecondi, in questo modo la sorgente ha il permesso di proseguire quando ne ha bisogno. In ogni momento ci sono 25 o 26 frame in attesa di acknowledgement. Detto in altri termini, la dimensione della finestra è 26.

Il bisogno di avere una finestra larga da parte della sorgente si ha quando il prodotto tra banda e ritardo di trasmissione totale è particolarmente grande. Se la banda è larga, anche in presenza di un piccolo ritardo la sorgente è in grado di saturare la finestra velocemente, a meno che questa non sia di grandi dimensioni. Se il ritardo è grande (per esempio su un canale satellitare geostazionario) la sorgente può saturare la finestra anche con una banda moderata. Il prodotto di questi due fattori indica essenzialmente qual è la capacità della pipeline (tubo). La sorgente, per poter operare alla massima efficienza, ha bisogno di poterla utilizzare tutta senza fermarsi.

Questa tecnica va sotto il nome di **pipelining**. Se il canale ha una capacità di b bit/sec, la dimensione del frame è l bit e il tempo totale di propagazione è R sec, allora il tempo necessario per trasmettere un singolo frame è l/b sec. Dopo che l'ultimo bit di un dato frame è stato inviato, c'è un ritardo di $R/2$ prima che il bit arrivi a destinazione e un altro ritardo di $R/2$ per il ritorno dell'acknowledgement, per un totale di R secondi di ritardo. Con lo stop-and-wait, la linea è occupata per l/b e ferma per R , quindi

$$\text{utilizzazione della linea} = l/(l + bR)$$

Se $l < bR$, l'efficienza sarà minore del 50%. Visto che c'è sempre un ritardo dovuto all'acknowledgement che ritorna indietro, il pipelining può essere usato per tenere la linea occupata anche durante questo intervallo. Se però l'intervallo è corto, il guadagno non giustifica la necessità di aggiungere altra complessità al protocollo.

Il pipelining dei frame su un canale di comunicazione non affidabile comporta alcuni seri problemi. Cosa succede quando un frame nel mezzo di una lunga sequenza viene danneggiato o perso? Una gran quantità di frame arriverà alla destinazione prima che la sorgente riesca a sapere che ci sono stati degli errori di trasmissione. Quando un frame danneggiato arriva a destinazione, deve ovviamente essere scartato. Cosa deve fare, però, la destina-

zione con i frame corretti che eventualmente seguono quello danneggiato? Ricordiamo che lo strato data link della destinazione è obbligato a mandare i pacchetti allo strato network nella corretta sequenza. La Figura 3.16 illustra gli effetti del pipelining sul ripristino degli errori, che adesso vedremo in dettaglio.

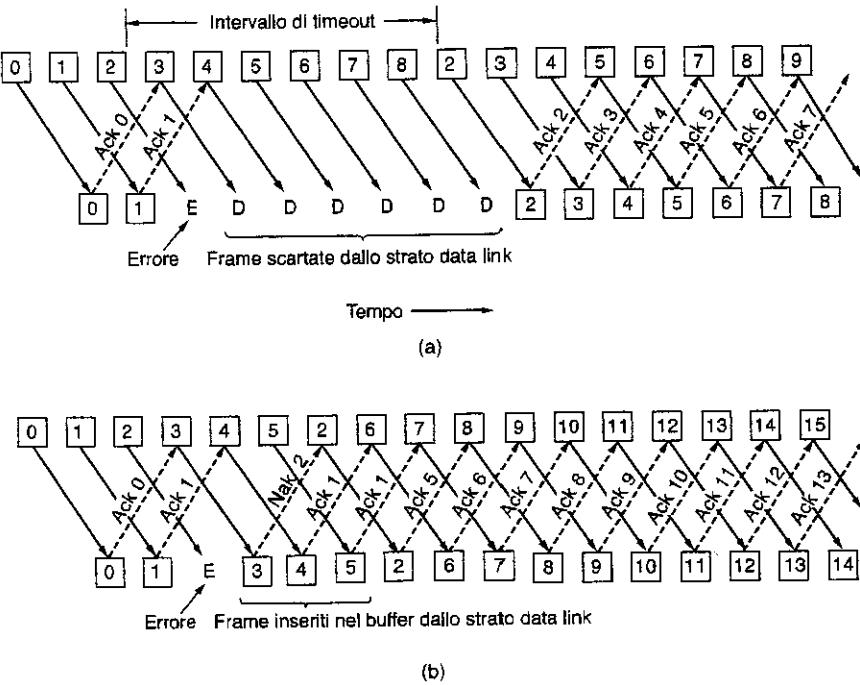


Figura 3.16. Pipelining e ripristino degli errori. Effetto di un errore quando (a) la finestra della destinazione ha dimensione 1 e (b) la finestra della destinazione è grande.

Per il ripristino degli errori in presenza di pipelining sono disponibili due approcci base. Uno è chiamato **go back n** (indietro n), e vuole che la destinazione semplicemente scarti tutti i frame successivi all'errore senza mandare l'acknowledgement per questi frame scartati [NdR - si noti tuttavia che il protocollo è bidirezionale, perciò continuano ad essere inviate le conferme "piggybacking", con il numero di sequenza dell'ultimo frame passato allo strato network]. La strategia corrisponde a una finestra di ricezione di dimensione 1. In altre parole lo strato data link, rilevato l'errore, si rifiuta di accettare qualunque frame eccetto il successivo che deve inviare allo strato network. Se la finestra della sorgente si riempie prima che il timer scatti, la pipeline comincia a svuotarsi. Dopo un po' di tempo la sorgente va in timeout e ritrasmette in ordine tutti i frame che non hanno ricevuto l'acknowledgement, cominciando da quello danneggiato o perso. Questo approccio può far perdere molta banda se la frequenza degli errori è alta.

Nella Figura 3.16(a) vediamo go back n nel caso di una finestra di destinazione larga. I frame 0 e 1 sono ricevuti correttamente con acknowledgement. Il frame 2, invece, è danneggiato o perso. La sorgente, ignara del problema, continua a inviare frame fino allo scatto del timer 2. A quel punto ritorna al frame 2 e ricomincia da lì, inviando di nuovo i frame 2, 3, 4 ecc.

L'altra strategia generale per trattare gli errori in presenza di pipelining è chiamata **ripetizione selettiva**. Con tale metodo, un frame in errore viene scartato, mentre i frame buoni ricevuti successivamente vengono messi in un buffer. Quando la sorgente va in timeout, solo il frame più vecchio senza acknowledgement viene ritrasmesso. Se quel frame arriva correttamente, la destinazione può passare in sequenza allo strato network tutti i frame che ha nel buffer. La ripetizione selettiva è spesso associata al fatto che la destinazione invia un acknowledgement negativo (NAK) quando trova un errore, per esempio quando riceve un errore di checksum o un frame fuori sequenza. I NAK stimolano la ritrasmissione prima che il timer corrispondente scatti, e quindi aumentano le performance.

Nella Figura 3.16(b) i frame 0 e 1 sono di nuovo ricevuti correttamente con acknowledgement, mentre il frame 2 è perso. Quando il frame 3 arriva a destinazione, lo strato data link nota che si è perso un frame, quindi invia un NAK per il frame 2 e mette il 3 nel buffer. Quando anche i frame 4 e 5 arrivano, anche questi vengono messi in buffer dallo strato data link, ma non sono ancora passati allo strato network. Dopo un po' il NAK 2 raggiunge la sorgente, che manda immediatamente una nuova copia del frame 2. Quando la copia arriva a destinazione lo strato data link ha tutti i frame da 2 a 5, e li può quindi passare nell'ordine corretto allo strato network. A questo punto può anche mandare l'acknowledgement di tutti i frame fino a 5 incluso, come mostrato in figura. Se si dovesse perdere il NAK, la sorgente andrà comunque in timeout per il frame 2 e autonomamente prenderà la decisione di ritrasmettere il frame 2 (e solamente quello), ma ciò avviene con un certo ritardo. In effetti, il NAK serve per accelerare la ritrasmissione di uno specifico frame.

La ripetizione selettiva corrisponde ad avere una finestra di ricezione maggiore di 1. Ogni frame nella finestra può essere accettato e messo in buffer fino a che tutti quelli che lo precedono sono stati passati allo strato network.

Questi due approcci alternativi hanno diverse conseguenze in termini di uso di banda e di spazio buffer nello strato data link. Si può usare un approccio oppure l'altro a seconda di quale risorsa è più scarsa. La Figura 3.17 mostra un protocollo pipelining nel quale lo strato data link della destinazione accetta i frame solamente in ordine: i frame che seguono un errore vengono scartati. In questo protocollo, per la prima volta abbiamo eliminato l'ipotesi che lo strato network abbia sempre una riserva infinita di pacchetti da mandare. Quando lo strato network vuole mandare un pacchetto, può farlo scatenando un evento *network_layer_ready*. Lo strato data link deve essere in grado di fermare lo strato network, se necessario, per via della regola di flusso che vuole che non ci siano più di *MAX_SEQ* frame senza acknowledgement. Le procedure *enable_network_layer* e *disable_network_layer* servono a questo scopo.

/* Il protocollo 5 (go back n) gestisce multipli frame in uscita. La sorgente può trasmettere fino a MAX_SEQ frame senza dover aspettare un ack. A differenza dei precedenti protocolli, non si assume più che lo strato network abbia un pacchetto disponibile in ogni momento. Invece, lo strato network scatena un evento network_layer_ready quando c'è un pacchetto da inviare.*/

```
#define MAX_SEQ 7           /* deve essere 2^n - 1 */
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
    /* Ritorna vero se a <= b < c in modo circolare; falso altrimenti */
    if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
        return(true);
    else
        return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[])
{
    /* Costruisce e invia un frame di dati */
    frame s;                      /* variabile di servizio */
    s.info = buffer[frame_nr];     /* inserisce il pacchetto nel frame */
    s.seq = frame_nr;              /* inserisce il numero di sequenza nel frame */
    s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1); /* piggyback ack */
    to_physical_layer(&s);        /* trasmette il frame */
    start_timer(frame_nr);         /* fa partire il timer */
}

void protocol5(void)
{
    seq_nr next_frame_to_send;      /* MAX_SEQ > 1; usato per il flusso in uscita */
    seq_nr ack_expected;           /* frame più vecchio ancora senza acknowledgement */
    seq_nr frame_expected;         /* prossimo frame atteso nel flusso in ingresso */
    frame r;                      /* variabile di servizio */
    packet buffer[MAX_SEQ + 1];    /* buffer per il flusso in uscita */
    seq_nr nbuffered;              /* # buffer di output correntemente in uso */
    seq_nr i;                      /* usato come indice per l'array dei buffer */
    event_type event;

    enable_network_layer();
    ack_expected = 0;
    next_frame_to_send = 0;
    frame_expected = 0;
    nbuffered = 0;
}
```

```
while (true) {
    wait_for_event(&event);                                /* Quattro possibilità: vedi event_type sopra */

    switch(event) {
        case network_layer_ready:                          /* lo strato network ha un pacchetto da inviare */
            /* Accetta, salva e trasmetti un nuovo frame */
            from_network_layer(&buffer[next_frame_to_send]); /* prende un nuovo pacchetto */
            nbuffered = nbuffered + 1;                         /* espande la finestra della sorgente */
            send_data(next_frame_to_send, frame_expected, buffer); /* trasmette il frame */
            inc(next_frame_to_send);                           /* avanza il confine sup. della finestra sorgente */
            break;

        case frame_arrival:                                /* è arrivato un frame di dati o di controllo */
            from_physical_layer(&r);                        /* prende il frame in ingresso dallo strato fisico */

            if (r.seq == frame_expected) {
                /* I frames vengono accettati solo in ordine */
                to_network_layer(&r.info);                  /* passa il pacchetto allo strato network */
                inc(frame_expected);                         /* avanza il confine inf. della finestra destinazione */
            }

        /* Ack n implica n - 1, n - 2, ecc. Controlla che questo sia vero */
        while (between(ack_expected, r.ack, next_frame_to_send)) {
            /* Gestisce il piggybacked ack. */
            nbuffered = nbuffered - 1; /* un frame in meno nei buffer */
            stop_timer(ack_expected); /* il frame è arrivato intatto, ferma il timer */
            inc(ack_expected);       /* riduce la finestra della sorgente */
        }
        break;

        case cksum_err: break;                            /* ignora i frame con errore */

        case timeout:                                     /* problemi, ritrasmetti tutti i frame senza acknowledgement */
            next_frame_to_send = ack_expected; /* comincia a ritrasmettere */
            for (i = t; i <= nbuffered; i++) {
                send_data(next_frame_to_send, frame_expected, buffer); /* riinvia il frame */
                inc(next_frame_to_send); /* prepara per mandare il prossimo frame */
            }
        }

        if (nbuffered < MAX_SEQ)
            enable_network_layer();
        else
            disable_network_layer();
    }
}
```

Figura 3.17. Un protocollo sliding window che usa go back n.

Notiamo che il numero massimo di frame che possono essere in attesa di acknowledgement in un dato istante è pari a MAX_SEQ e non $MAX_SEQ + 1$, anche se ci sono $MAX_SEQ + 1$ numeri di sequenza distinti: 0, 1, 2, ..., MAX_SEQ . Per vedere il perché di questa restrizione consideriamo il seguente scenario con $MAX_SEQ = 7$:

1. la sorgente invia i frame da 0 a 7
2. un acknowledgement per il frame 7 ritorna indietro alla sorgente tramite piggybacking
3. la sorgente manda altri otto frame con numeri di sequenza da 0 a 7
4. un altro acknowledgement per il frame 7 ritorna tramite piggybacking.

A questo punto la domanda è: gli otto frame del secondo gruppo sono arrivati tutti con successo, oppure sono stati persi tutti (contiamo gli scarti che seguono un errore come persi)? In entrambi i casi la destinazione manderebbe il frame 7 come acknowledgement senza che la sorgente abbia modo di discriminare fra le due situazioni. Per questo motivo il massimo numero di frame che possono essere in attesa di acknowledgement deve essere limitato a MAX_SEQ .

Anche se il protocollo 5 non mette in buffer i frame che arrivano dopo un errore, questo non elimina del tutto il problema dell'uso dei buffer. Dato che a un certo punto una sorgente si può trovare nella situazione di dover ritrasmettere tutti i frame che non hanno ricevuto acknowledgement, deve per forza tenere traccia di tutti i frame inviati finché non è sicura che sono stati accettati dalla destinazione. Quando arriva l'acknowledgement per il frame n , anche i frame $n - 1, n - 2, \dots$ hanno automaticamente l'acknowledgement. Questa proprietà è particolarmente importante nel caso in cui alcuni dei frame contenenti i precedenti acknowledgement siano andati persi o si siano corrotti. Quando arriva un qualsiasi acknowledgement, lo strato data link controlla se può rilasciare un buffer. In quest'ultimo caso viene liberato dello spazio nella finestra e quindi lo strato network, se era bloccato, adesso potrà lanciare dei nuovi eventi di tipo *network_layer_ready*.

Per questo protocollo assumiamo che ci sia sempre del traffico all'indietro sul quale effettuare il piggybacking degli acknowledgement. In caso contrario, gli acknowledgement non possono essere trasmessi. Il protocollo 4 non ha bisogno di questa ipotesi poiché manda indietro un frame per ognuno di quelli che riceve, anche nel caso in cui abbia già mandato proprio quel frame. Nel protocollo successivo risolveremo in modo elegante il problema del traffico che va in una sola direzione.

Il protocollo 5 ha logicamente bisogno di più timer, in quanto deve gestire il transito di più frame alla volta e ognuno di questi necessita un timer. Ogni frame può andare in timeout in modo indipendente dagli altri. Tutti questi timer possono essere facilmente simulati via software usando un singolo orologio hardware che lancia degli interrupt periodici. I timeout registrati formano una lista collegata (*linked list*), dove ogni nodo della lista contiene il numero di intervalli (*tick*) dell'orologio che mancano per arrivare al timeout del timer, il frame a cui corrisponde il timer e un puntatore al nodo successivo.

Per illustrare una possibile implementazione dei timer, consideriamo l'esempio della Figura 3.18(a). Assumiamo che l'orologio scandisca intervalli di 100 millisecondi.

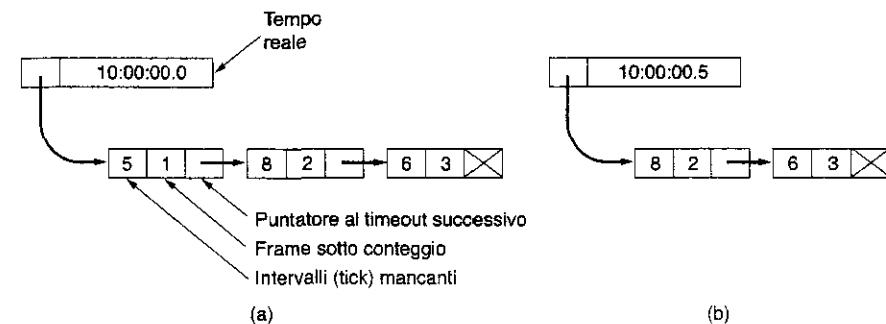


Figura 3.18. Simulazione di timer multipli via software.

Inizialmente l'ora segnata dall'orologio è 10:00:00.0 e ci sono tre timeout in coda: alle 10:00:00.5, alle 10:00:01.3 e alle 10:00:01.9. Ogni volta che l'orologio scandisce un intervallo, l'ora segnata viene incrementata, mentre il contatore dei tick in testa alla lista viene decrementato. Quando il contatore dei tick diventa zero, viene generato un timeout e il nodo è rimosso dalla lista, come si vede nella Figura 3.18(b). Questo modo di implementare il timer non richiede molto lavoro per ogni intervallo dell'orologio, anche se è necessario scandire la lista tutte le volte che vengono chiamate *start_timer* oppure *stop_timer*. Nel protocollo 5, entrambe queste routine sono state implementate con un parametro che indica il frame al quale il timer fa riferimento.

3.4.3 Un protocollo che usa la ripetizione selettiva

Il protocollo 5 lavora bene se gli errori sono rari. Se invece la qualità della linea è scarsa, perde una gran quantità di banda con la ritrasmissione dei frame. Una strategia alternativa per la gestione degli errori consiste nel permettere alla destinazione di accettare, dentro ad appositi buffer, anche i frame che seguono un frame perso o danneggiato. Un tale protocollo non scarta i frame solo perché un frame precedente non è arrivato integro.

In questo protocollo sorgente e destinazione mantengono una finestra di numeri di sequenza accettabili. La finestra della sorgente comincia da 0 e cresce fino a un valore massimo predefinito, MAX_SEQ . La destinazione, invece, ha un buffer di dimensione fissa pari a MAX_SEQ . La destinazione ha un buffer riservato per ogni numero di sequenza all'interno della sua finestra. Associato a ogni buffer c'è un bit (detto *arrived*) che indica quando il buffer è pieno o vuoto. Quando un frame arriva, il suo numero di sequenza è controllato dalla funzione *between* per vedere se si trova all'interno della finestra. In caso affermativo, se non è già stato ricevuto, il frame viene accettato e memorizzato. Quest'azione è eseguita indipendentemente dal fatto che il frame contenga il successivo pacchetto per lo strato network. Ovviamente, il frame non viene passato subito allo strato network, ma è trattenuto dallo strato data link fino al momento in cui tutti i frame con numeri di sequenza inferiori sono stati ricevuti e passati in ordine allo strato network. Un protocollo che usa questo algoritmo è mostrato nella Figura 3.19.

```

/* Il protocollo 6 (selective repeat) accetta dei frame fuori ordine, ma passa i pacchetti allo strato
network in ordine. Ad ogni frame in uscita è associato un timer. Quando il timer va in timeout, solamente il frame coinvolto viene ritrasmesso, non tutti i frame in transito come nel protocollo 5. */

#define MAX_SEQ 7           /* deve essere 2^n - 1 */
#define NR_BUFS ((MAX3SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;
#include "protocol.h"

boolean no_nak = true;      /* nessun nak è stato inviato fino ad ora */
seq_nr oldest_frame = MAX_SEQ + 1;  /* il valore iniziale è solo per il simulatore */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
    /* Come per il protocollo 5, ma più corto e oscuro */
    return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet buffer[])
{
    /* Costruisce e invia un frame di dati, ack, o nak */
    frame s;                  /* variabile di servizio */

    s.kind = fk; /* kind == data, ack, or nak */
    if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
    s.seq = frame_nr;          /* ha senso solo per i frame di dati */
    s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
    if (fk == nak) no_nak = false;  /* un solo nak per frame */
    to_physical_layer(&s);
    if (fk == data) start_timer(frame_nr % NR_BUFS);
    stop_ack_timer();          /* non c'è bisogno di un frame di ack separato */
}

void protocol6(void)
{
    seq_nr ack_expected;        /* limite inferiore della finestra della sorgente */
    seq_nr next_frame_to_send;  /* limite superiore della finestra della sorgente + 1 */
    seq_nr frame_expected;      /* limite inferiore della finestra della destinazione */
    seq_nr too_far;             /* limite sup. della finestra della destinazione + 1 */
                                /* indice del buffer */

    frame r; /* variabile di servizio */
    packet out_buf[NR_BUFS];
    packet in_buf[NR_BUFS];
    boolean arrived[NR_BUFS];
    seq_nr nbuffered;           /* quanti buffer di output sono usati correntemente */
    event_type event;

    enable_network_layer();
    ack_expected = 0;
    next_frame_to_send = 0;
    frame_expected = 0;
    too_far = NR_BUFS;
    nbuffered = 0;
    /* inizialmente non ci sono pacchetti nei buffer */
    for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
    while (true) {

```

```

        wait_for_event(&event);          /* cinque possibilità: vedi event_type sopra */
        switch(event) {
            case network_layer_ready:   /* accetta, salva e trasmetti il nuovo frame */
                nbuffered = nbuffered + 1; /* espande la finestra */
                from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* prende il nuovo pacchetto */
                send_frame(data, next_frame_to_send, frame_expected, out_buf); /* trasmette il frame */
                inc(next_frame_to_send);           /* porta avanti il limite superiore della finestra */
                break;

            case frame_arrival:          /* è arrivato un frame di controllo o di dati */
                from_physical_layer(&r);      /* prende il frame in arrivo dallo strato fisico */
                if (r.kind == data) {
                    /* è arrivato un frame senza danni */
                    if ((r.seq != frame_expected) && no_nak)
                        send_frame(nak, 0, frame_expected, out_buf); else start_ack_timer();
                    if (between(frame_expected, r.seq, too_far) && (arrived[r.seq % NR_BUFS] == false)) {
                        /* 1 frame possono essere accettati in qualunque ordine */
                        arrived[r.seq % NR_BUFS] = true; /* segna il buffer come pieno */
                        in_buf[r.seq % NR_BUFS] = r.info; /* inserisce i dati nel buffer */
                        while (arrived[frame_expected % NR_BUFS]) {
                            /* Passa il frame e porta avanti la finestra */
                            to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                            no_nak = true;
                            arrived[frame_expected % NR_BUFS] = false;
                            inc(frame_expected);           /* avanza la soglia inferiore della finestra della destinazione */
                            inc(too_far);                 /* avanza il limite superiore della finestra della destinazione */
                            start_ack_timer();           /* controlla se serve un ack separato */
                        }
                    }
                }
                if ((r.kind == nak) && between(ack_expected, (r.ack + 1) % (MAX_SEQ + 1), next_frame_to_send))
                    send_frame(data, (r.ack + 1) % (MAX_SEQ + 1), frame_expected, out_buf);

                while (between(ack_expected, r.ack, next_frame_to_send)) {
                    nbuffered = nbuffered - 1;      /* gestisce l'ack con piggyback */
                    stop_timer(ack_expected % NR_BUFS); /* il frame è arrivato intatto */
                    inc(ack_expected);           /* avanza il limite inferiore della finestra della sorgente */
                }
                break;

            case cksum_err:
                if (no_nak) send_frame(nak, 0, frame_expected, out_buf); /* frame danneggiato */
                break;
            case timeout:
                send_frame(data, oldest_frame, frame_expected, out_buf); /* timeout */
                break;
            case ack_timeout:
                send_frame(ack, 0, frame_expected, out_buf); /* ack timer in timeout, manda ack */
        }
        if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
    }
}

```

Figura 3.19. Un protocollo sliding window che usa selective repeat.

La ricezione non sequenziale introduce problemi che non sono presenti in protocolli dove i frame sono accettati solamente in ordine. Per illustrare il problema più facilmente usiamo un esempio. Supponiamo di avere un numero di sequenza a tre bit, così che la sorgente può trasmettere fino a sette frame prima di dover aspettare un acknowledgement. Gli stati iniziali delle finestre della sorgente e della destinazione sono come nella Figura 3.20(a). A questo punto la sorgente trasmette i frame da 0 a 6. La finestra della destinazione permette di ricevere tutti i frame con numero di sequenza fra 0 e 6 incluso. Tutti e sette i frame arrivano in modo corretto, così che la destinazione invia un acknowledgement e porta avanti la sua finestra per permettere la ricezione di 7, 0, 1, 2, 3, 4 o 5, come mostrato nella Figura 3.20(b). Tutti i sette buffer sono segnati come vuoti.

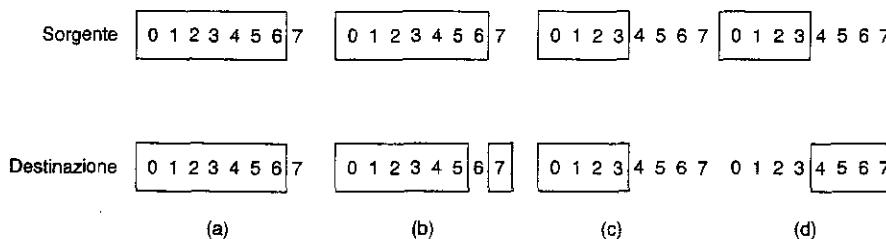


Figura 3.20. (a) Situazione iniziale con una finestra di dimensione sette. (b) Dopo che sette frame sono stati inviati e ricevuti, ma ancora in attesa di acknowledgement. (c) Situazione iniziale con una finestra di dimensione quattro. (d) Dopo che quattro frame sono stati inviati e ricevuti, ma ancora in attesa di acknowledgement.

A questo punto accade un disastro, sotto forma di un fulmine che colpisce un palo della linea telefonica e spazza via tutti gli acknowledgement. La sorgente dopo un po' di tempo va in timeout e ritrasmette il frame 0. Quando questo frame arriva alla destinazione, viene fatto un controllo per vedere se ricade all'interno della finestra della destinazione. Sfortunatamente, nella Figura 3.20(b) il frame 0 è all'interno della nuova finestra, quindi verrà accettato. La destinazione manda (usando il piggybacking) un acknowledgement per il frame 6, visto che i frame da 0 a 6 sono stati ricevuti.

La sorgente è contenta di sapere che tutti i frame trasmessi sono arrivati correttamente, quindi fa scorrere in avanti la sua finestra e manda immediatamente i frame 7, 0, 1, 2, 3, 4, 5. Il frame 7 sarà accettato dalla destinazione e il pacchetto corrispondente verrà passato direttamente allo strato network. Subito dopo, lo strato data link della destinazione controlla per vedere se ha già un frame 0 valido, scopre che ce l'ha e passa il pacchetto allo strato network. In conclusione, lo strato network riceve un pacchetto sbagliato, quindi il protocollo ha fallito.

L'essenza del problema è che, dopo che la destinazione ha portato avanti la sua finestra, il nuovo intervallo di numeri di sequenza validi si è sovrapposto con quello vecchio. Di conseguenza, la serie di frame trasmessa successivamente può essere o un duplicato della precedente (se tutti gli acknowledgement sono stati persi) oppure una nuova serie (se tutti gli acknowledgement sono stati ricevuti). La destinazione non è in grado di distinguere i due casi.

Il metodo per risolvere questo dilemma consiste nell'essere sicuri che, quando la destinazione ha portato avanti la sua finestra, non ci siano sovrapposizioni con la finestra originale. Per fare questo, la massima dimensione della finestra dovrebbe essere uguale alla metà della sequenza di numeri, come è mostrato nella Figura 3.20(c) e nella Figura 3.20(d). Se per esempio usiamo 4 bit per i numeri di sequenza, l'intervallo sarà da 0 a 15. Solo un massimo di otto frame senza acknowledgement dovrebbe essere in transito per ogni dato istante. In quel modo, se la destinazione ha appena accettato i frame da 0 a 7 e portato avanti la finestra in modo da ricevere i frame da 8 a 15, può sicuramente distinguere le ritrasmissioni (numeri da 0 a 7) dalle nuove sequenze (numeri da 8 a 15). In generale, la dimensione della finestra per il protocollo 6 sarà $(MAX_SEQ + 1)/2$. Quindi per una sequenza di tre bit, la dimensione della finestra è quattro.

Un domanda interessante è: quanti buffer deve avere la destinazione? In nessuna circostanza la destinazione accetterà dei frame con numero di sequenza inferiore al valore minimo presente nella finestra o, analogamente, valori maggiori del valore massimo nella finestra. Di conseguenza, il numero di buffer necessari è uguale alla dimensione della finestra, non all'intervallo dei numeri di sequenza. Nell'esempio di sopra, con un numero di sequenza di quattro bit, sono necessari otto buffer identificati da 0 a 7. Quando arriva il frame i , viene messo nel buffer $i \bmod 8$. Notiamo che, anche se i e $i + 8$ "competono" per lo stesso buffer, non verranno mai a trovarsi nella finestra allo stesso tempo, perché questo significherebbe avere una finestra di dimensione 9 o maggiore.

Per lo stesso motivo, il numero di timer necessari è uguale al numero dei buffer, non alla dimensione della sequenza. A tutti gli effetti un timer è associato a ogni buffer. Quando il timer scatta, i contenuti del buffer sono ritrasmessi.

Nel protocollo 5 si assume implicitamente che il canale sia caricato pesantemente. Quando arriva un frame, l'acknowledgement non viene mandato immediatamente, ma viene invece mandato con piggybacking insieme al successivo frame in uscita. Se il traffico di ritorno è leggero, l'acknowledgement può essere trattenuto per un tempo lungo. Se c'è molto traffico in una data direzione e nessun traffico in quella opposta, verranno inviati solo MAX_SEQ pacchetti e poi il protocollo si bloccherà, per questo abbiamo dovuto assumere che ci sia sempre almeno un po' di traffico all'indietro.

Nel protocollo 6 questo problema è risolto. Dopo l'arrivo di una sequenza di frame di dati, viene fatto partire un timer ausiliario usando *start_ack_timer*. Un frame di acknowledgement separato è inviato se non si è verificato nessun traffico di ritorno prima del timeout del timer. Un interrupt causato dal timer ausiliario è chiamato un evento di *ack_timeout*. Con questa modifica, è possibile gestire il flusso di traffico unidirezionale, in quanto l'assenza di frame di dati all'indietro su cui effettuare il piggybacking degli acknowledgement adesso non è più un ostacolo. Esiste un solo timer ausiliario, e quando *start_ack_timer* è invocato mentre il timer sta girando, esso viene semplicemente fatto ripartire per contare un intero intervallo di acknowledgement.

È essenziale che l'intervallo di tempo associato al timer ausiliario sia decisamente più corto di quello usato per il timer dei frame di dati. Questa condizione è necessaria per assicurarsi che i frame ricevuti correttamente riescano anche a trasmettere l'acknowledgement in tempo, e non siano ritrasmessi per via di un timeout del timer ausiliario.

Il protocollo 6 usa una strategia più efficiente del protocollo 5 per gestire gli errori. Quando la destinazione sospetta che sia accaduto un errore, manda indietro alla sorgente un frame di acknowledgement negativo (NAK), necessario per la ritrasmissione del frame specificato nel NAK. Ci sono due casi in cui la destinazione dovrebbe sospettare un errore: l'arrivo di un frame danneggiato oppure l'arrivo di un frame diverso da quello atteso (è possibile che un frame sia stato perso). È buona norma che la destinazione tenga traccia dei frame di cui ha chiesto la ritrasmissione con un NAK, questo per evitare di fare richieste di ritrasmissione multiple per un dato frame. La variabile *no_nak* nel protocollo 6 è impostata a vero se finora nessun NAK è stato inviato per il frame *frame_expected*. Se il NAK viene perso o corrotto non accade nulla di grave, in quanto la sorgente andrà comunque in timeout e ritrasmetterà il frame. Se arriva un frame sbagliato dopo che un NAK è stato inviato e poi perso, allora *no_nak* verrà impostato a vero e il timer ausiliario viene fatto partire. Al raggiungimento del timeout, sarà inviato un ACK per sincronizzare nuovamente lo stato corrente della sorgente e della destinazione.

In alcune situazioni, il tempo necessario perché un frame si propaghi a destinazione, sia elaborato e l'acknowledgement torni indietro è (praticamente) costante. In tali casi la sorgente può impostare il suo timer in modo che abbia un timeout leggermente più lungo dell'intervallo normalmente misurato fra l'invio di un frame e la ricezione del suo acknowledgement. Se invece questo tempo è molto variabile, la sorgente deve scegliere fra un timeout corto (quindi rischiare delle ritrasmissioni inutili) e uno lungo (che comporta un lungo periodo di attesa dopo un errore). Entrambe le scelte sprecano banda. Se il traffico di ritorno è sporadico, il tempo necessario all'acknowledgement sarà irregolare: più corto quando c'è traffico di ritorno e più lungo quando non ce n'è. Un altro possibile problema può essere il tempo di elaborazione variabile alla destinazione. In generale, quando la deviazione standard dell'intervallo di acknowledgement è piccola rispetto all'intervallo stesso, il timer può essere impostato in modo "rigido" e i NAK non sono utili. Altrimenti il timer va impostato in modo "flessibile", per evitare inutili ritrasmissioni, però in questo caso i NAK possono velocizzare considerevolmente la ritrasmissione dei frame danneggiati o persi.

Legato al discorso dei timeout e dei NAK, vi è quello di identificare i frame sui quali è scattato un timeout. Nel protocollo 5, la risposta è *ack_expected*, cioè è sempre l'ultimo frame. Nel protocollo 6 non c'è un modo semplice per determinare quale frame ha ricevuto un timeout. Supponiamo che i frame da 0 a 4 siano stati trasmessi, ovvero che la lista dei frame in transito sia, in ordine cronologico: 0, 1, 2, 3, 4. Immaginiamo la situazione in cui: 0 va in timeout, 5 (un nuovo frame) viene trasmesso, 1 va in timeout, 2 va in timeout e 6 (un altro nuovo frame) viene trasmesso. A questo punto la lista dei frame in transito sarà: 3, 4, 0, 5, 1, 2, 6 (in ordine cronologico). Se tutto il traffico in ingresso (cioè i frame che portano acknowledgement) viene perso per un po' di tempo, i sette frame in transito andranno in timeout in ordine cronologico.

Per non complicare troppo l'esempio non abbiamo mostrato l'amministrazione del timer; supponendo invece che la variabile *oldest_frame* venga impostata al momento del timeout, in modo da indicare quale frame ha ricevuto il timeout stesso.

3.5 Verifica dei protocolli

I protocolli usati nelle situazioni realistiche, così come i programmi che li implementano, sono spesso decisamente complicati. Di conseguenza è stata fatta molta ricerca per trovare tecniche matematiche per definire e verificare i protocolli. Nei paragrafi seguenti vedremo alcuni modelli e tecniche di questo tipo. Anche se li esamineremo nell'ambito dello strato data link, molti sono applicabili anche ad altri strati.

3.5.1 Modelli a stati finiti

Un concetto chiave usato da svariati modelli di protocollo è quello di **macchina a stati finiti**. Seguendo questa tecnica, ogni **macchina protocollo** (cioè la sorgente o la destinazione) si trova sempre in uno specifico stato per ogni istante di tempo. Il suo stato è dato dal valore di tutte le sue variabili, incluso il program counter.

Nella maggior parte dei casi, un gran numero di stati può essere raggruppato per facilitare l'analisi. Per esempio, se consideriamo la destinazione per il protocollo 3, potremmo considerare (fra tutti i suoi possibili stati) due che risultano veramente importanti: attesa del frame 0 e attesa del frame 1. Tutti gli altri stati possono essere pensati come dei transienti, semplici passi intermedi verso uno dei due stati principali. Tipicamente si scelgono gli stati in modo da rappresentare gli istanti in cui la macchina protocollo aspetta il successivo evento [cioè esegue la chiamata alla procedura *wait(event)*]. A questo punto lo stato della macchina protocollo è completamente determinato dallo stato di tutte le sue variabili. Il numero degli stati è 2^n , dove *n* rappresenta il numero di bit necessari a rappresentare tutte le variabili prese insieme.

Lo stato di tutto il sistema è la combinazione di tutti gli stati delle due macchine protocollo e del canale. Lo stato del canale è determinato dal suo contenuto. Usando ancora l'esempio del protocollo 3, il canale ha quattro possibili stati: un frame 0 (o un frame 1) che si muove dalla sorgente alla destinazione, un frame di acknowledgement che torna indietro, un canale vuoto. Se modelliamo la sorgente e la destinazione con due stati ciascuno, il sistema completo ha 16 stati.

È necessaria una precisazione. Il concetto di frame "nel canale" è ovviamente un'astrazione. Quello che vogliamo rappresentare è la situazione in cui il frame è in transito, oppure è stato ricevuto ma ancora non è stato elaborato dalla destinazione. Un frame rimane "nel canale" finché la macchina protocollo non esegue *from_physical_layer* e successivamente lo elabora.

Partendo da ogni stato, ci sono zero o più possibili **transizioni** verso altri stati. Le transizioni avvengono quando accadono degli eventi. Per una macchina protocollo, una transizione può avvenire quando un frame viene inviato, oppure quando arriva, quando scatta un timer o quando viene generato un interrupt, ecc. Per il canale, gli eventi tipici sono l'inserimento di un nuovo frame nel canale o la perdita di un frame dovuta a rumore. Data una descrizione completa delle macchine protocollo e delle caratteristiche del canale, è possibile disegnare un grafo orientato che mostra tutti gli stati come nodi e tutte le transizioni come archi orientati.

Un particolare stato è battezzato con il nome **stato iniziale**; corrisponde alla descrizione del sistema nel momento in cui comincia a lavorare, o comunque a un tempo prossimo a quello e che preferiamo usare come inizio. Dallo stato **iniziale** alcuni (forse tutti) gli stati sono raggiungibili con un'appropriata sequenza di transizioni. Usando delle tecniche ben note nella teoria dei grafi (per esempio la chiusura transitiva del grafo), è possibile determinare quali stati sono raggiungibili e quali no. Questa tecnica è detta **analisi di raggiungibilità** (Lin et al. 1987) ed è utilizzata per determinare la correttezza dei protocolli.

Formalmente, il modello di un protocollo con una macchina a stati finiti può essere considerato come una quadrupla (S, M, I, T) dove:

S è l'insieme degli stati dei processi e del canale

M è l'insieme dei frame che possono essere scambiati su un canale

I è l'insieme degli stati iniziali dei processi

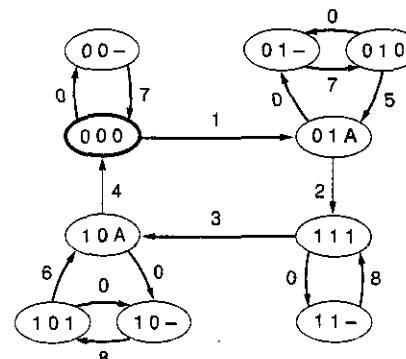
T è l'insieme delle transizioni fra stati.

A tempo zero, tutti i processi sono nei loro stati iniziali. A questo punto cominciano ad accadere degli eventi, per esempio i frame diventano disponibili per la trasmissione, oppure scattano alcuni timer. Ogni evento può far sì che un processo o il canale prendano delle azioni e commutino verso un nuovo stato. Elenmando con precisione tutti i possibili successori di ogni stato, è possibile costruire il grafo di raggiungibilità e analizzare il protocollo.

L'analisi di raggiungibilità può essere usata per identificare una varietà di errori nelle specifiche del protocollo. Per esempio, è possibile avere la situazione in cui un certo frame deve essere elaborato in un dato stato, ma la macchina a stati finiti non dice quale azione bisogna intraprendere. Questo significa che il protocollo è in errore (per incompletezza). Un altro possibile errore si ha quando esiste un insieme di stati dal quale non è possibile uscire, e quindi non si possono più trasmettere frame. In questo caso abbiamo un errore detto deadlock (letteralmente "blocco mortale"). Un errore meno grave consiste nell'avere una specifica di protocollo per trattare un evento in uno stato in cui non potrà mai verificarsi (transizione irrilevante). Con questa tecnica è possibile rilevare anche altri tipi di errore.

Un esempio di modello con una macchina a stati finiti è rappresentato nella Figura 3.21(a). Questo grafo corrisponde al protocollo 3, descritto in precedenza: ogni macchina protocollo ha due stati e il canale ne ha quattro. Esistono un totale di 16 stati, non tutti raggiungibili dallo stato iniziale. Gli stati non raggiungibili non sono mostrati nella figura. Per semplicità ignoriamo gli errori di checksum.

Ogni stato è indicato da tre caratteri: SRC , dove S vale 0 o 1 e corrisponde al frame che la sorgente tenta di inviare, R analogamente vale 0 o 1 e corrisponde al frame che la destinazione si aspetta, mentre C può assumere i valori 0, 1, A , vuoto (-), e corrisponde agli stati del canale. In questo esempio come stato iniziale è stato scelto (000), che corrisponde alla situazione in cui la sorgente ha appena inviato un frame 0, la destinazione si aspetta un frame 0 e il frame 0 è al momento nel canale.



(a)

(b)

Figura 3.21. (a) Diagramma di stato per il protocollo 3. (b) Transizioni.

Nella Figura 3.21 sono mostrati nove tipi di transizioni. La transizione 0 rappresenta la situazione in cui il canale perde il suo contenuto. La transizione 1 rappresenta la situazione in cui il canale porta correttamente il frame 0 alla destinazione, la quale cambia il suo stato per attendere un frame 1 ed emette un acknowledgement. Inoltre nella transizione 1 i dati sono passati allo strato network. Le altre transizioni sono elencate nella Figura 3.21(b). L'arrivo di un frame con un errore di checksum non è stato riportato in quanto non cambia lo stato (nel nostro modello del protocollo 3).

Durante le normali operazioni, le transizioni 1, 2, 3 e 4 vengono ripetute in ordine e ciclicamente. Per ogni ciclo vengono trasmessi due frame, il che riporta la sorgente allo stato iniziale di voler trasmettere un nuovo frame con numero di sequenza uguale a 0. Se il canale perde il frame 0, allora si passa allo stato (00-). Dopo un po' di tempo la sorgente va in timeout (transizione 7) e il sistema ritorna a (000). La perdita di un acknowledgement è più complicata, in quanto richiede due transizioni: 7 e 5 (oppure 8 e 6) per poter riparare il problema.

Una delle proprietà che deve avere un protocollo con numero di sequenza a 1 bit è che, indipendentemente dalla sequenza degli eventi, la destinazione non accetta mai due frame 1 senza che ci sia un frame 0 in mezzo (e viceversa). Dal grafo della Figura 3.21 si vede che questo requisito può essere riformulato in modo più rigoroso come "non deve esistere un percorso che a partire dallo stato iniziale veda due istanze della transizione 1 senza che ci sia una transizione 3 fra di esse, o viceversa". Dalla figura possiamo vedere che il protocollo 3 segue correttamente questa regola.

Un requisito simile vuole che non esista nessun percorso nel quale la sorgente cambia stato due volte (per esempio da 0 a 1 e poi di nuovo a 0) mentre la destinazione rimane costante. Se troviamo un tale percorso, vuol dire che due frame sono stati persi irrimediabilmente senza che la destinazione se ne sia accorta. In questo caso anche la sequenza dei pacchetti trasmessi avrebbe un gap non rilevato.

L'assenza di deadlock è una proprietà importante per un protocollo. Un **deadlock** è una situazione in cui il protocollo non può più proseguire nelle sue attività (cioè mandare pacchetti allo strato network), indipendentemente dalla sequenza degli eventi. Nel modello a grafo, un deadlock è caratterizzato dall'esistenza di un sottoinsieme di stati, raggiungibile dallo stato iniziale e che abbia le due proprietà:

1. non ci sono transizioni che escono dal sottoinsieme
2. non ci sono transizioni nel sottoinsieme che riescano a far proseguire la trasmissione dei frame.

Raggiunta una situazione di deadlock, il protocollo rimane fermo per sempre. Dal grafo del protocollo 3 possiamo facilmente vedere che non ci sono situazioni di deadlock.

3.5.2 Modelli a rete di Petri

La macchina a stati finiti non è la sola tecnica per specificare formalmente un protocollo. In questo paragrafo descriveremo una tecnica completamente diversa: la **rete di Petri** (Danthine, 1980). Una rete di Petri ha quattro elementi costitutivi: posizioni (chiamate anche posti), transizioni, archi e indicatori. La **posizione** rappresenta uno stato nel quale il sistema o una sua parte si possono trovare. La Figura 3.22 mostra una rete di Petri con due posizioni, A e B, entrambe indicate da un cerchietto. Il fatto che il sistema si trovi nello stato A viene indicato con un **indicatore** (punto pieno) nella posizione A. Una **transizione** è indicata da una barretta orizzontale o verticale. Ogni transizione ha zero o più archi di input che arrivano dalle sue posizioni di ingresso (input) e zero o più archi di output che partono dalle posizioni di uscita (output).

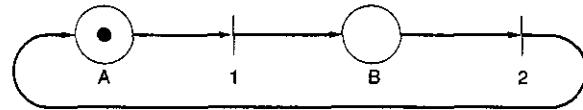


Figura 3.22. Una rete di Petri con due posizioni e due transizioni.

Una transizione è **abilitata** se esiste almeno un indicatore di input in ognuna delle sue posizioni di input. Ogni transizione abilitata può **scattare** a piacimento, per togliere un indicatore da ogni posizione di input e metterlo in ogni posizione di output. Gli indicatori non saranno conservati se il numero di archi di input è differente da quello di output. Ogni transizione abilitata può scattare, e può scegliere di farlo con modalità che non sono deterministiche. Questa proprietà rende le reti di Petri utili come modello dei protocolli. La rete di Petri della Figura 3.22 è deterministica e può essere usata come modello per ogni processo a due fasi (per esempio per il comportamento di un neonato: mangia, dormi, mangia, dormi, ecc). Come per altri strumenti di modellazione, abbiamo soppresso i dettagli inutili.

La Figura 3.23 riporta il modello a rete di Petri della Figura 3.12. A differenza del modello con una macchina a stati finiti, in questo caso non esistono stati composti: gli stati della destinazione, della sorgente e del canale vengono rappresentati separatamente.

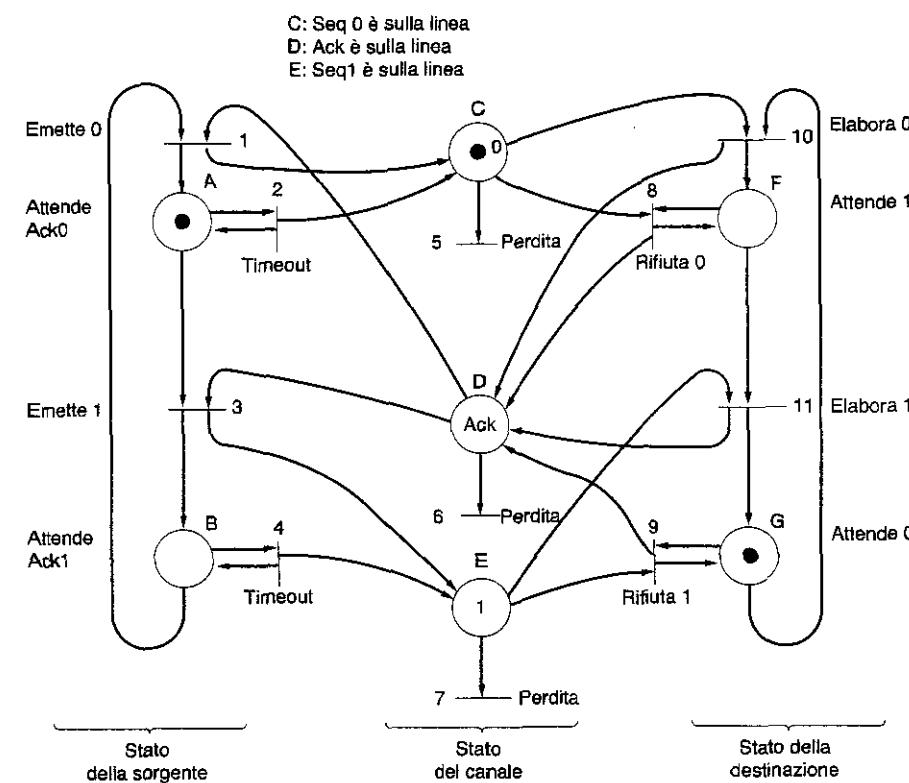


Figura 3.23. Un modello a rete di Petri per il protocollo 3.

Le transizioni 1 e 2 corrispondono alla trasmissione del frame 0, rispettivamente in modo normale e con un timeout. Le transizioni 3 e 4 sono analoghe, ma per il frame 1. Le transizioni 5, 6 e 7 corrispondono alla perdita rispettivamente del frame 0, di un acknowledgement e del frame 1. Le transizioni 8 e 9 rappresentano il caso di un data frame che arriva alla destinazione con il numero di sequenza sbagliato. Le transizioni 10 e 11 rappresentano l'inoltro del frame alla destinazione e il successivo passaggio allo strato network.

Le reti di Petri possono essere usate per identificare gli errori di protocollo in modo analogo a quanto abbiamo visto con le macchine a stati finiti. Per esempio, se una sequenza di eventi include la transizione 10 due volte senza che avvenga la transizione 11, il protocollo è da considerarsi erroneo. Il concetto di deadlock per una rete di Petri è simile al suo analogo per le macchine a stati finiti.

Le reti di Petri possono essere rappresentate, per facilità d'uso, in una forma algebrica che assomiglia a una grammatica. Ogni transizione contribuisce a una regola della grammatica. Ogni regola specifica le posizioni di input e di output delle transizioni. La Figura 3.23 ha 11 transizioni, quindi la grammatica corrispondente avrà 11 regole, che possiamo

merare da 1 a 11 in modo corrispondente alle transizioni. La grammatica per la rete di Petri della Figura 3.23 è:

- 1: BD → AC
- 2: A → A
- 3: AD → BE
- 4: B → B
- 5: C →
- 6: D →
- 7: E →
- 8: CF → DF
- 9: EG → DG
- 10: CG → DF
- 11: EF → DG

Siamo stati in grado di ridurre un protocollo complesso a 11 semplici regole grammaticali, facilmente manipolate da un programma.

Lo stato della rete di Petri è rappresentato da un insieme disordinato di posizioni, dove ogni posizione è presente nell'insieme tante volte quanti sono i suoi indicatori. Ogni regola consente le posizioni presenti nella parte sinistra può scattare; in questo modo rimuove tali posizioni dallo stato corrente, e aggiunge le posizioni di output allo stato corrente. Nella Figura 3.23 vediamo che la configurazione marcata è ACG (cioè A, C e G hanno ognuno un indicatore). Di conseguenza le regole 2, 5 e 10 sono tutte abilitate e possono quindi scattare, portando a un nuovo stato (eventualmente con la stessa configurazione marcata). A contrario, la regola 3 (AD → BE) non può essere applicata perché D non ha un indicatore.

6 Esempi di protocolli data link

Nei paragrafi seguenti esamineremo alcuni protocolli data link molto utilizzati nella pratica. Il primo è HDLC, un classico protocollo orientato ai bit, le cui varianti sono in uso per decenni e per diverse applicazioni. Il secondo, PPP, è il protocollo usato per connettere a Internet con il modem del computer di casa.

5.1 HDLC (High-level Data Link Control)

In questo paragrafo esamineremo un gruppo di protocolli, simili fra di loro, che sono ancora molto usati nonostante siano ormai abbastanza vecchi. Sono tutti derivati da un protocollo nato in principio nell'ambiente IBM dei mainframe: il protocollo **SDLC** (*Synchronous Data Link Control*, protocollo di controllo sincrono del data link). Dopo aver sviluppato il protocollo SDLC, IBM ha sottoposto le sue specifiche ad ANSI e ISO perché le accettassero come standard americani e internazionali, rispettivamente. ANSI ha apportato delle modifiche ed è nato **ADCCP** (*Advanced Data Communication Control Procedure*, procedura avanzata per la comunicazione di dati). Analogamente lo standard ISO è diventato **HDLC** (*High-level Data Link Control*, controllo del data link ad alto livello). CCITT ha adottato e modificato HDLC

per creare il suo **LAP** (*Link Access Procedure*, procedura di accesso al collegamento), che è parte dello standard X.25 per le interfacce di rete, il quale è stato a sua volta modificato in **LAPB** per renderlo più compatibile con una successiva versione di HDLC. Il bello degli standard è che ne sono davvero parecchi fra cui scegliere. Nel caso non ce ne piaccia nessun, si può sempre aspettare la versione dell'anno successivo.

Questi protocolli sono tutti basati sugli stessi principi: sono orientati ai bit e usano il bit stuffing per la trasparenza dei dati. Differiscono per dettagli minori, ma non per questo meno irritanti. La discussione che segue sui protocolli orientati ai bit è da intendersi come un'introduzione generale. Per avere i dettagli specifici di ciascun protocollo si consiglia di consultare la documentazione relativa. Tutti i protocolli orientati ai bit usano una struttura di frame come mostrato nella Figura 3.24. Il campo *Address* è di assoluta importanza per le linee con più terminali, in quanto è usato proprio per indicare il terminale. Nelle linee punto-punto, a volte è usato per distinguere i comandi dalle risposte.

Bit	8	8	8	≥ 0	16	8
	0 1 1 1 1 1 1 0	Address	Control	Data	Checksum	0 1 1 1 1 1 1 0

Figura 3.24. Formato dei frame per i protocolli orientati ai bit.

Il campo *Control* viene usato per numeri di sequenza, per gli acknowledgement e altri scopi che verranno discussi più avanti.

Il campo *Data* può contenere qualunque tipo d'informazione. Può avere lunghezza arbitraria, anche se l'efficienza del checksum diminuisce al crescere della lunghezza del frame, per via della maggior probabilità d'incontrare errori burst multipli.

Il campo *Checksum* è un codice a ridondanza ciclica calcolato con la tecnica che abbiamo esaminato nel Paragrafo 3.2.2.

Il frame è delimitato da una sequenza flag (01111110). Nelle linee punto-punto, quando non ci sono dati da trasmettere, viene continuamente trasmessa la sequenza flag. Il frame di lunghezza minima contiene tre campi per un totale di 32 bit, escludendo i flag ad entrambe le estremità del frame. I frame possono essere di tre tipi: **informazione**, **sovraimmissione** e **senza numero**. I contenuti del campo *Control* per ognuno di questi tre tipi sono riportati nella Figura 3.25. Il protocollo usa una finestra con un numero di sequenza di 3 bit. In ogni istante possiamo avere fino a 7 frame in attesa di acknowledgement. Il campo *Seq* nella Figura 3.25 è il numero di sequenza del frame. Il campo *next* è usato per il piggyback dell'acknowledgement. Tutti i protocolli di questa famiglia aderiscono alla convenzione di usare per il piggyback dell'acknowledgement, non il numero dell'ultimo frame ricevuto, ma invece quello del successivo frame non ancora ricevuto (cioè quello che la destinazione si aspetta). Questa scelta è puramente arbitraria: non importa quale convenzione si usa, l'importante è essere consistenti nel suo uso. Il bit *P/F* (*Poll/Final*) è usato quando un computer (o un concentratore) vuole interrogare un gruppo di terminali. Se usato come *P*, il computer invita il terminale a mandare i dati. Tutti i frame inviati dal terminale, eccetto l'ultimo, hanno il bit *P/F* impostato a *P*, mentre l'ultimo è impostato a *F*.

Bit	1	3	1	3
(a)	0	Seq	P/F	Next
(b)	1	0	Type	P/F
(c)	1	1	Type	P/F
				Modifier

Figura 3.25. Campo di controllo per (a) un frame d'informazioni, (b) un frame di supervisione, (c) un frame senza numero.

In alcuni protocolli il bit *P/F* è usato per forzare l'altra macchina a inviare immediatamente un frame di tipo supervisione, senza quindi aspettare di avere del traffico all'indietro per eseguire il piggybacking. Questo bit è usato, in alcuni rari casi, anche in connessione con i frame senza numero.

I frame di tipo supervisione si distinguono a seconda del valore del campo *Type* (*tipo*). Il tipo 0 è un frame di acknowledgement (ufficialmente chiamato RECEIVE READY, cioè ricevi e stai pronto) usato per indicare il prossimo numero di sequenza atteso. Questo frame è usato quando non c'è traffico all'indietro da usare per il piggybacking.

Il tipo 1 è un frame di tipo acknowledgement negativo (ufficialmente chiamato REJECT, cioè rifiuta). Viene usato per indicare che è stato rilevato un errore di trasmissione. A quel punto il campo *Next* indica il primo frame della sequenza che non è stato ricevuto correttamente (cioè il frame che deve essere ritrasmesso). La sorgente deve ritrasmettere tutti i frame a partire da *Next*. Questo tipo di strategia è simile a quella del nostro protocollo 5.

Il tipo 2 è detto anche RECEIVE NOT READY (ricevi e fermati). Serve per generare l'acknowledgement di tutti i frame fino a, ma senza includere, *Next*. È simile a RECEIVE READY, però in questo caso viene comunicato alla sorgente di smettere di trasmettere. RECEIVE NOT READY viene usato per segnalare dei problemi temporanei con la destinazione, per esempio il riempimento del buffer, e non come alternativa al controllo di flusso con sliding window. Quando la condizione di errore è stata superata, la destinazione manda RECEIVE READY, REJECT o particolari frame di controllo.

Il tipo 3 è SELECTIVE REJECT (rifiuta selettivamente) e richiede la ritrasmissione solo del frame specificato. In questo senso è simile al nostro protocollo 6 e quindi è utile nel caso in cui la finestra della sorgente è minore o uguale alla dimensione allocata per i numeri di sequenza. Quindi se la destinazione desidera mettere in buffer i frame fuori sequenza, per esempio per un loro potenziale uso futuro, può farlo forzando la ritrasmissione degli specifici frame che gli interessano usando SELECTIVE REJECT. I protocolli HDLC e ADCCP implementano questo tipo di frame, mentre per SDLC e LAPB il tipo 3 non è definito, cioè questi ultimi due protocolli non hanno selective reject.

La terza classe di frame è il tipo senza numero. Viene usata a volte per informazioni di controllo, ma può anche trasportare dati, per conto di servizi senza connessione e non affidabili. I vari protocolli orientati ai bit differiscono considerevolmente nell'implementazione di questa classe, al contrario delle altre due che sono sostanzialmente identiche per tutti. Sono dis-

ponibili 5 bit per indicare il tipo di frame, però non tutte le 32 combinazioni sono usate. Tutti i protocolli hanno il comando DISC (*DISConnect*, disconnetti), che permette a una macchina di annunciare il fatto che sta andando fuori linea (per esempio per manutenzione). Hanno anche il comando che permette a una macchina che è appena tornata in linea di annunciare la sua presenza e forzare la sequenza di numeri a zero. Questo comando è chiamato SNRM (*Set Normal Response Mode*, imposta il modo di risposta normale). Sfortunatamente il modo normale di risposta è tutto fuorché normale. È un modo sbilanciato (cioè asimmetrico) in cui un'estremità della linea è master (capo) e l'altra è slave (servo). SNRM risale a un tempo in cui le comunicazioni dati venivano fatte da terminali non intelligenti che colloquiavano con l'elaboratore centrale (host), il che è chiaramente intrinsecamente asimmetrico. Per rendere il protocollo più appropriato al caso di comunicazioni fra pari, HDLC e LAPB introducono un comando aggiuntivo: SABM (*Set Asynchronous Balanced Mode*, imposta il modo asincrono bilanciato). Questo comando esegue un reset della linea e dichiara che entrambi i soggetti della comunicazione sono di pari grado. Vengono introdotti anche i comandi SABME e SNRME, analoghi di SABM e SNRM, con la differenza che i primi abilitano un formato esteso per i frame che usa 7 bit per i numeri di sequenza, al posto dei 3 bit usati dagli altri.

Un terzo comando presente in tutti i protocolli è FRMR (*FrMe Reject*, rifiuta i frame), usato per indicare l'arrivo di un frame con il checksum corretto ma con una semantica impossibile. Esempi di semantiche impossibili sono: un frame di tipo (supervisione) in LAPB, oppure un frame più corto di 32 bit, un frame di controllo illegale, un acknowledgement di un frame che era al di fuori della finestra, ecc. I frame FRMR contengono un campo dati da 24 bit con l'indicazione degli errori nel frame con la semantica impossibile. Quindi questo campo contiene: il campo di controllo del frame errato, i parametri di finestra e una collezione di bit che indica l'errore specifico.

I frame di controllo, così come quelli dati, si possono perdere o danneggiare e quindi devono avere anch'essi un acknowledgement. Per questo scopo viene usato uno speciale frame di controllo, chiamato UA (*Unnumbered acknowledgement*, acknowledgement senza numero). Visto che in ogni istante ci può essere un solo frame di controllo in uscita, non sono possibili ambiguità su quale frame di controllo sta ricevendo l'acknowledgement.

I rimanenti frame di controllo hanno a che vedere con le operazioni di inizializzazione, interrogazione (*polling*), e di rapporto sullo stato. C'è anche un tipo di frame di controllo che può contenere informazioni, UI (*Unnumbered Information*, informazione non numerata). Questi dati non sono passati allo strato network, sono solamente per lo strato data link.

Anche se è molto usato, il protocollo HDLC è ben lontano dall'essere perfetto. Una discussione su una serie di problemi legati all'uso di HDLC si trova in (Fiorini et al. 1994).

3.6.2 Lo strato data link in Internet

Internet consiste in una serie di macchine individuali (host e router) e nell'infrastruttura di comunicazione che le connette fra di loro. All'interno di un singolo edificio si usano spesso delle LAN per l'interconnessione delle macchine, mentre la maggior parte dell'infrastruttura WAN è formata da linee punto-punto dedicate. Nel Capitolo 4 descriveremo le LAN, qui invece vogliamo esaminare i protocolli data link usati nelle linee punto-punto di Internet.

La comunicazione punto-punto è usata principalmente in due situazioni. Nella prima, pensiamo al caso tipico per migliaia di organizzazioni in cui abbiamo una o più LAN, ognuna costituita da diversi host (personal computer, workstation, server, ecc) e router (oppure bridge, che hanno funzioni simili). Spesso i router sono connessi da una LAN detta backbone (letteralmente "spina dorsale"). Tipicamente tutte le connessioni verso il mondo esterno passano attraverso uno o due router che hanno delle linee dedicate verso router remoti. Sono questi router e le linee che li collegano a creare le sottoreti sulle quali è costruito Internet.

Le linee punto-punto giocano un ruolo importante anche in una seconda situazione: la connessione di milioni di individui, che si collegano a Internet da casa attraverso la normale linea telefonica usando un modem. Tipicamente, succede che il PC di casa dell'utente si collega al router di un Internet service provider (ISP), e quindi diventa a tutti gli effetti un host connesso a Internet. Questo modo di operare è praticamente identico ad avere una linea dedicata fra il PC e il router, a parte il fatto che la connessione termina quando l'utente chiude la chiamata. Il caso di un PC di casa che chiama un Internet service provider è illustrato nella Figura 3.26. Nella figura il modem è disegnato esternamente al computer per enfatizzarne il suo ruolo, comunque nella maggior parte dei computer moderni troviamo un modem interno.

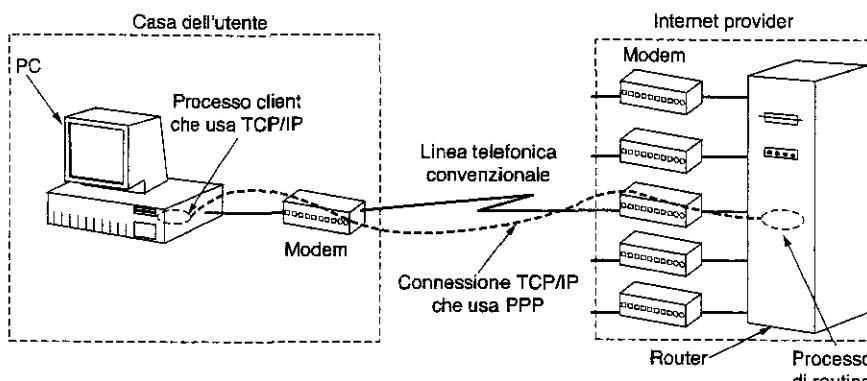


Figura 3.26. Un personal computer domestico collegato a Internet.

Per le linee dedicate router-router come per quelle che usano la linea telefonica per il collegamento host-router, è necessario avere un protocollo data link di tipo punto-punto che gestisca il framing, il controllo degli errori e le altre funzioni descritte in questo capitolo per lo strato data link. Internet usa il protocollo PPP.

PPP – il protocollo punto-punto

Internet ha bisogno di un protocollo punto-punto per una varietà di scopi, fra cui la gestione del traffico da router a router e quello tra gli utenti di casa e gli ISP. Il protocollo usato da Internet si chiama **PPP** (*Point-to-Point Protocol*, protocollo punto a punto), che è defi-

nito nell'[RFC 1661](#) e ulteriormente elaborato in altri RFC (come per esempio [RFC 1662](#) e [1663](#)). Fra le varie funzionalità di PPP troviamo: la rilevazione degli errori, il supporto per più protocolli, la possibilità di negoziare gli IP al momento della connessione e la possibilità di effettuare l'autenticazione. Tre caratteristiche del PPP meritano menzione.

1. Un metodo di framing che permette di delimitare in modo non ambiguo la fine di un frame e l'inizio del successivo. Il formato del frame permette anche di gestire la rilevazione degli errori.
2. Un protocollo di collegamento per gestire la connessione, il test della linea, negoziare le opzioni di collegamento e gestire la disconnessione in modo pulito quando la linea non serve più. Questo protocollo è chiamato **LCP** (*Link Control Protocol*, protocollo di controllo del collegamento). LCP supporta sia circuiti asincroni sia sincroni e gestisce le codifiche orientate ai bit come pure quelle orientate ai byte.
3. Una modalità per negoziare le opzioni relative allo strato network, in modo indipendente dall'implementazione di tale strato che verrà usata per la comunicazione. Il metodo scelto avrà un diverso **NCP** (*Network Control Protocol*, protocollo di controllo del network) per ogni strato network supportato.

Come esempio, consideriamo il tipico scenario di un utente che usa il PC di casa per chiamare un ISP: temporaneamente il PC diventa un host connesso a Internet. Come prima cosa il PC chiama il router del provider tramite il modem. Dopo che il modem del router ha risposto al telefono e stabilito una connessione fisica, il PC manda al router una serie di pacchetti LCP nel campo payload di uno o più frame PPP. Questi pacchetti e le loro risposte servono per stabilire i parametri PPP da usare.

Dopo aver stabilito i parametri PPP, vengono inviati una serie di pacchetti NCP per configurare lo strato network. Tipicamente il PC vuole usare il TCP/IP e quindi ha bisogno di ottenere un indirizzo IP. Siccome gli indirizzi IP sono una risorsa limitata, normalmente un ISP ne ha a disposizione un blocco che poi assegna dinamicamente ai PC man mano che si connettono e solo per la durata della sessione. Se un provider ha a disposizione n indirizzi, allora potrà avere al massimo n macchine connesse simultaneamente, però la sua base di clienti può essere molto maggiore di n . NCP è usato per assegnare gli indirizzi IP.

A questo punto il PC è diventato a tutti gli effetti un host connesso a Internet e quindi può inviare e ricevere pacchetti come se fosse connesso stabilmente alla rete. Quando l'utente ha finito di lavorare, NCP si prende carico di terminare la connessione allo strato network e liberare l'indirizzo IP. A questo punto LCP chiude la connessione data link. Infine il computer comunica al modem di chiudere la linea telefonica e quindi rilasciare la connessione allo strato fisico.

Il formato del frame PPP è stato scelto in modo da assomigliare a quello di HDLC, visto che non c'era ragione di reinventare la ruota. La principale differenza fra PPP e HDLC è che PPP è orientato ai caratteri, mentre HDLC è orientato ai bit. In particolare PPP usa il byte stuffing sulle linee telefoniche, così che tutti i frame sono costituiti da un numero intero di byte. Non è possibile mandare un frame di 30,25 byte con PPP, mentre si può con-

HDLC. I frame PPP possono essere trasferiti, oltre che su linee telefoniche, anche su linee SONET oppure linee orientate ai bit con protocollo HDLC (per esempio per connessioni router-router). Il formato del frame PPP è mostrato nella Figura 3.27. Tutti i frame PPP cominciano con il flag byte standard HDLC (01111110), su cui viene applicato il byte stuffing se compare all'interno dei dati [NdR - stranamente, il PPP (nella modalità sincrona usata sulle linee telefoniche analogiche) utilizza il character stuffing applicato su tutto il frame (ad eccezione ovviamente dei flag byte che marcano l'inizio e la fine del frame)],

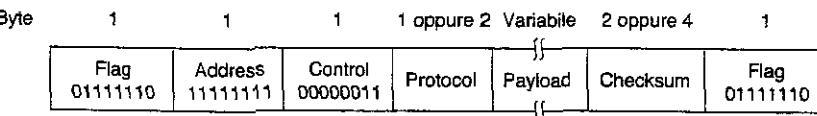


Figura 3.27. Il formato completo dei frame PPP per le operazioni in modalità senza numero.

quindi non solo sul campo dati (Payload). I caratteri escape vengono introdotti in trasmissione dopo aver calcolato il checksum, e vengono rimossi in ricezione prima di ricalcolare il checksum (cfr. RFC1662, paragrafo 4.2, www.faqs.org/rfcs/rfc1662.html). Segue poi il campo *Address*, che viene sempre impostato a 11111111 per indicare che tutte le stazioni devono accettare il frame. In questo modo si evita il problema di dover assegnare indirizzi data link. Il campo *Address* è seguito dal campo *Control*, che ha come valore di default 00000011. Questo valore indica un frame senza numero. In ambienti rumorosi, come le reti wireless, si può usare una trasmissione affidabile con numerazione dei frame. I dettagli sono definiti in RFC 1633, anche se in pratica è usata molto di rado. Visto che *Address* e *Control* sono sempre costanti nella configurazione di default, LCP contiene un meccanismo per far sì che la sorgente e la destinazione possano negoziare l'opzione di omettere completamente questi campi e salvare quindi due byte per frame. Il quarto campo è *Protocol* e serve per comunicare quale tipo di pacchetto è contenuto nel campo *Payload*. Dei codici sono definiti per LCP, NCP, IP, IPX, AppleTalk e anche altri protocolli. I protocolli che cominciano con un bit a 0 sono protocolli di strato network come IP, IPX, OSI, CLNP, XNS. Quelli che cominciano con un bit a 1 sono usati per negoziare altri protocolli. Questi includono LCP e un diverso NCP per ogni strato network supportato. La dimensione di default del campo *Protocol* è 2 byte, ma può essere negoziata fino a 1 byte con LCP. Il campo *Payload* ha lunghezza variabile fino a un valore massimo che può essere negoziato usando LCP durante l'inizializzazione della connessione. Il valore di default è 1.500 byte. Se necessario ci può essere riempimento del campo dopo i dati. Il campo successivo è *Checksum*, normalmente di 2 byte, ma può essere anche negoziato un checksum di 4 byte. Per riassumere, il PPP è un meccanismo di framing multiprotocollo adatto alla trasmissione dei dati via modem, linee seriali HDLC, SONET e altri strati fisici. Supporta la rilevazione degli errori, la negoziazione delle scelte, la compressione dell'intestazione e, optionalmente, la trasmissione affidabile con un formato di frame tipo HDLC. Dopo aver descritto il formato dei frame, vediamo adesso come le linee vengono attivate e disconnesse. Il diagramma (semplificato) della Figura 3.28 mostra le fasi che una linea deve attra-

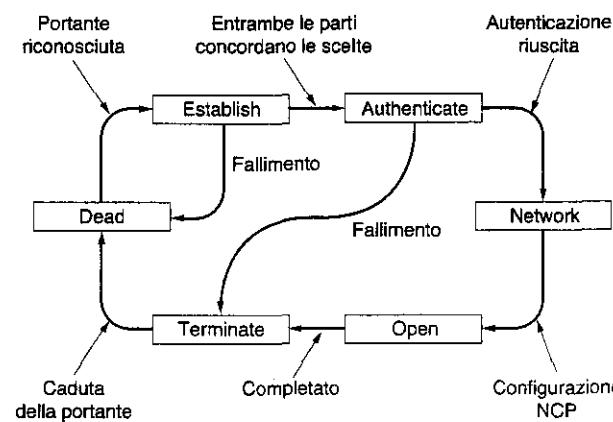


Figura 3.28. Un diagramma di fase semplificato per attivare e scollegare la linea.

versare per essere attivata e poi disconnessa. La sequenza si applica sia alle connessioni via modem sia a quelle router-router. Il protocollo comincia con la linea nello stato *DEAD*, che significa che allo strato fisico non abbiamo nessuna portante e quindi non esiste ancora nessuna connessione. Dopo aver stabilito una connessione fisica la linea si muove allo stato *ESTABLISH*. A quel punto comincia la negoziazione LCP che, se ha successo, porta allo stato *AUTHENTICATE*. Ora sorgente e destinazione possono controllare la loro reciproca identità, se lo desiderano. Quando si entra nella fase *NETWORK*, viene invocato l'appropriato protocollo NCP per configurare lo strato network. Se la configurazione ha successo si raggiunge *OPEN* e può quindi cominciare il trasporto dei dati. Quando il trasporto è finito, la linea si sposta in *TERMINATE* e da lì torna a *DEAD* quando viene rilasciata la portante. Le opzioni del protocollo data link sono negoziate con LCP durante la fase *ESTABLISH*. Il protocollo LCP non si occupa delle opzioni negoziate, ma solo del meccanismo per compiere questa operazione. Fornisce al processo che inizia la comunicazione un modo per poter fare una proposta e alla controparte un modo per poterla accettare o rifiutare, in tutto o in parte. LCP fornisce ai due processi anche un modo per testare la qualità della linea e quindi giudicare se può essere considerata adeguata per stabilire una connessione. Infine, il protocollo LCP consente di disconnettere le linee quando non servono più. RFC 1661 definisce 11 tipi di frame LCP; l'elenco è riportato nella Figura 3.29. I 4 tipi *Configure-* permettono a chi inizia la comunicazione (I) di proporre dei valori optionali alla parte che risponde (R), in modo che questa possa accettarli o rifiutarli. Nel secondo caso, R può fare delle proposte alternative o dichiarare che non intende assolutamente negoziare certe opzioni. Le opzioni negoziate e i valori proposti fanno parte dei frame LCP.

I codici *Terminate-* chiudono una linea quando non serve più. I codici *Code-reject* e *Protocol-reject* indicano che R non ha capito qualche messaggio. Questa situazione può indicare che un errore di trasmissione è passato senza essere rilevato; oppure, più probabilmente, che I e R stanno usando differenti versioni del protocollo LCP. I codici *Echo-*

Nome	Direzione	Descrizione
Configure-request	I → R	Elenca valori e scelte possibili
Configure-ack	I ← R	Tutte le scelte sono accettate
Configure-nak	I ← R	Alcune scelte non sono accettate
Configure-reject	I ← R	Alcune scelte non sono negoziabili
Terminate-request	I → R	Richiesta di chiudere la linea
Terminate-ack	I ← R	OK, linea chiusa
Code-reject	I ← R	Ricevuta una richiesta sconosciuta
Protocol-reject	I ← R	Richiesto un protocollo sconosciuto
Echo-request	I → R	Prego rimandare indietro questo frame
Echo-reply	I ← R	Ecco il frame rimandato indietro
Discard-request	I → R	Scartare questo frame (per test)

Figura 3.29. Le tipologie di frame LCP.

sono usati per i test sulla qualità della linea. Infine, *Discard-request* serve per il debugging. Se ci sono dei problemi di trasmissione, il programmatore può usare questo codice per dei test. Infatti questi codici, quando sono ricevuti, vengono semplicemente scartati senza compiere nessun'altra azione.

Le opzioni che possono essere negoziate includono l'impostazione della massima dimensione del campo payload per i frame di dati, l'abilitazione dell'autenticazione, la scelta del protocollo da usare, l'abilitazione del controllo sulla qualità della linea durante le normali operazioni e la scelta di un eventuale metodo di compressione dell'intestazione dei frame.

Non c'è molto da dire sui protocolli NCP in generale. Ognuno è specifico per un certo protocollo dello strato network e permette quindi di effettuare le configurazioni necessarie per quel protocollo. Per IP, per esempio, la possibilità più importante è quella dell'assegnazione dinamica dell'indirizzo.

3.7 Sommario

Lo strato data link ha il compito di convertire il flusso di bit in ingresso dallo strato fisico in un flusso di frame da passare allo strato network. Si usano diversi metodi per il framing, fra cui il conteggio dei caratteri, il byte stuffing e il bit stuffing. I protocolli data link possono prendersi carico della correzione degli errori e della ritrasmissione dei frame persi o danneggiati. Con il controllo del flusso i protocolli data link prevengono situazioni in cui una sorgente veloce inonda una destinazione lenta. Il metodo sliding window è largamente usato per integrare in modo conveniente il controllo degli errori e del flusso.

I protocolli sliding window sono caratterizzati dalla dimensione delle finestre sorgente e destinazione. Quando sono entrambe uguali a 1, il protocollo è stop-and-wait. Quando la dimensione della finestra della sorgente è maggiore di 1 (per esempio, per evitare che la

sorgente si possa bloccare su un circuito che ha un lungo ritardo di propagazione) la destinazione può essere programmata in due modi: o per scartare tutti i frame eccettuato il successivo in sequenza, o per mettere in un buffer i frame ricevuti fuori sequenza prima del loro uso.

In questo capitolo abbiamo esaminato una serie di protocolli. Il protocollo 1 è pensato per un ambiente privo di errori in cui la destinazione può gestire tutto il flusso che gli viene mandato, indipendentemente dalla velocità di trasmissione. Il protocollo 2 considera ancora un ambiente privo di errori, ma aggiunge il controllo di flusso. Il protocollo 3 gestisce gli errori introducendo i numeri di sequenza e usando un algoritmo stop-and-go. Il protocollo 4 permette la comunicazione bidirezionale e introduce il concetto di piggybacking. Il protocollo 5 usa un protocollo sliding window con metodo go back n. Infine il protocollo 6 usa la ripetizione selettiva e l'acknowledgement negativo.

Esistono diverse tecniche per creare modelli formali dei protocolli, usati per dimostrarne la correttezza (o per trovare eventuali errori di progetto). I modelli con macchine a stati finiti o reti di Petri sono frequentemente impiegati a questo scopo.

Molte reti usano, per lo strato data link, un protocollo orientato ai bit come per esempio SDLC, HDLC, ADCCP, LAPB. Tutti questi protocolli usano dei flag byte per delimitare i frame e il bit stuffing per evitare che un flag byte venga erroneamente inserito nei dati. Un'altra loro caratteristica comune è l'uso del controllo di flusso di tipo sliding window. Il protocollo data link principale per le linee punto-punto di Internet è il PPP.

Problemi

1. Un pacchetto dello strato network viene suddiviso in 10 frame, ognuno dei quali ha l'80% di probabilità di arrivare integro a destinazione. Se non viene eseguito nessun controllo degli errori da parte dello strato data link, quante volte, in media, deve essere ritrasmesso il messaggio perché possa essere infine trasmesso interamente senza errori?
2. Un protocollo data link usa la seguente codifica a caratteri:
A: 01000111; B: 11100011; FLAG: 01111110; ESC: 11100000
Trovare la sequenza di bit trasmessi (in binario) per il frame di quattro caratteri A B ESC FLAG, per ognuno dei seguenti metodi di framing:
(a) conteggio dei caratteri
(b) flag byte con byte stuffing
(c) flag byte di inizio e fine con bit stuffing.
3. Il seguente frammento si trova nel mezzo di un flusso di dati trasmesso usando l'algoritmo di byte stuffing descritto nel testo: A B ESC C ESC FLAG FLAG D. Qual è l'output dopo il byte stuffing?
4. Consideriamo la seguente considerazione riguardo al byte stuffing: avere un flag byte alla fine di un frame e poi subito un altro all'inizio del prossimo frame è uno spreco, si potrebbe infatti risparmiare un byte usando un solo flag byte. È corretto questo ragionamento?
5. Una stringa di bit, 0111101111101111110, deve essere trasmessa al livello data link. Qual è la stringa che viene effettivamente trasmesso dopo il bit stuffing?
6. Quando viene usato il bit stuffing, è possibile che la perdita, inserimento o modifica di un singolo bit possa causare un errore non identificabile dal checksum? Se no, perché? Se sì, come?

La lunghezza del checksum ha un ruolo in questo problema?

7. Esistono delle circostanze in cui un protocollo open-loop (cioè una codifica di Hamming) risulta preferibile ai protocolli di tipo feedback discussi in questo capitolo?
8. Per ottenere un'affidabilità maggiore di quella data dal singolo bit di parità, viene usata una codifica per la rilevazione degli errori che consiste in un bit di parità per tutti i bit dispari e uno per tutti i bit pari. Qual è la distanza di Hamming di questa codifica?
9. Una codifica di Hamming viene usata per trasmettere messaggi da 16 bit. Quanti bit di controllo sono necessari per assicurarsi che la destinazione possa rilevare e correggere gli errori di singoli bit? Qual è la sequenza di bit usata per trasmettere il messaggio 1101001100110101, usando una codifica di Hamming a parità pari?
10. Un byte con valore binario pari a 10101111 deve essere codificato con una parità pari in una codifica di Hamming. Qual è il valore binario dopo la codifica?
11. Un codice di Hamming di 12 bit con valore (in esadecimale) 0xE4F arriva a destinazione. Qual è il valore originale, in esadecimale? Supponiamo che non più di 1 bit sia in errore.
12. Un modo per rilevare gli errori consiste nel trasmettere i dati come blocchi di n righe di k bit per riga e aggiungere i bit di parità a ogni riga e ogni colonna. Il bit in basso a destra controlla la parità della sua riga e della sua colonna. Questo schema riesce a rilevare tutti gli errori di singoli bit? E di due bit? E di tre?
13. Un blocco di bit con n righe e k colonne usa la parità orizzontale e verticale per correggere gli errori. Supponiamo gli errori di trasmissione invertano esattamente 4 bit. Determinare la probabilità che l'errore passi senza essere identificato.
14. Qual è il resto della divisione di $x^7 + x^5 + 1$ per il polinomio generatore $x^3 + 1$?
15. La sequenza di bit 10011101 viene trasmesso usando il metodo standard del CRC descritto nel testo. Il polinomio generatore è $x^3 + 1$. Determinare la sequenza di bit realmente trasmessa dallo strato data link. Supponiamo che il terzo bit da sinistra sia invertito durante la trasmissione. Mostrare il dettaglio di come questo errore viene rilevato dalla destinazione.
16. I protocolli data link mettono il CRC quasi sempre alla fine del frame piuttosto che nell'intestazione. Perché?
17. Un canale trasmette alla velocità di 4 kbps (kilobit per secondo) e ha un ritardo di propagazione di 20 millisecondi. Se si vuole usare un algoritmo stop-and-wait con efficienza di almeno 50%, qual è l'intervallo ammissibile per la dimensione dei frame?
18. Una dorsale T1 lunga 3.000 km viene usata per trasmettere frame di 64 byte usando il protocollo 5. Se la velocità di propagazione è 6 msec/km, quanti bit si devono usare per il numero di sequenza?
19. Nel protocollo 3, è possibile che la sorgente faccia partire il timer mentre sta già girando? Se sì, quando può accadere? Se no, perché è impossibile?
20. Immaginiamo un protocollo di tipo sliding window che utilizzi così tanti bit per i numeri di sequenza da non dover mai riciclare i numeri. Che relazione intercorre fra i quattro bordi della finestra e la sua dimensione, che è un valore costante ed è valido tanto per la sorgente che per

la destinazione?

21. Se la procedura *between* nel protocollo 5 controllasse la condizione $a \leq b \leq c$ al posto di $a \leq b < c$, ci sarebbero degli effetti sulla correttezza del protocollo o sulla sua efficienza? Spiegate la vostra risposta.
22. Nel protocollo 6, quando arriva un frame di dati si effettua il controllo per vedere se il numero di sequenza è differente dal valore atteso e se *no_nak* è vero. Se valgono entrambe le condizioni, viene inviato un NAK. Altrimenti viene fatto partire un timer ausiliario. Supponiamo che quest'ultima operazione venga omessa. Ci sarebbero degli effetti sulla correttezza del protocollo?
23. Supponiamo che il ciclo while con tre istruzioni vicino alla fine del protocollo 6 sia rimosso dal programma. Questo avrebbe degli effetti sulla correttezza del protocollo o solamente sulle performance? Spiegate la vostra risposta.
24. Supponiamo che, nel protocollo 6, dall'istruzione *switch* venga rimosso il case relativo agli errori di checksum. Quale sarebbe l'effetto di questo cambiamento sulle operazioni del protocollo?
25. Nel protocollo 6, il codice della procedura *frame_arrival* ha una sezione usata per i NAK. Questa sezione è invocata se il frame in arrivo è un NAK e vale anche un'altra condizione. Trovare uno scenario in cui la presenza di quest'altra condizione risulta essere essenziale.
26. Immaginiamo di dover scrivere il software dello strato data link per una linea usata da noi per ricevere dati, ma non per inviarne. L'altra parte usa HDLC, con numeri di sequenza a 3 bit e una finestra di dimensione pari a 7 frame. Per aumentare l'efficienza, vogliamo poter mettere in buffer quanti più frame fuori sequenza possibile. Non possiamo però modificare il software lato sorgente. È possibile avere un ricevitore con dimensione della finestra maggiore di 1 e continuare a garantire la correttezza del protocollo? Se sì, qual è la massima dimensione della finestra che può essere usata?
27. Consideriamo l'uso del protocollo 6 su una linea senza errori a 1 Mbps. La massima dimensione dei frame è 1.000 bit. I pacchetti vengono generati alla frequenza di 1 al secondo. L'intervallo di timeout è 10 millisecondi. Se lo speciale timer per gli acknowledgement fosse eliminato, avremmo dei timeout non necessari. Quante volte sarebbe trasmesso in media un messaggio?
28. Nel protocollo 6, $MAX_SEQ = 2^n - 1$. Questa condizione è ovviamente desiderabile per un uso efficiente dei bit dell'header, ma non abbiamo dimostrato che è anche essenziale. Per esempio, il protocollo può funzionare correttamente con $MAX_SEQ = 4$?
29. Frame da 1.000 bit vengono inviati su un canale a 1 Mbps usando un satellite geostazionario con un tempo di propagazione dalla terra di 270 millisecondi. Gli acknowledgement sono sempre aggiunti con piggybacking ai frame di dati. Le intestazioni sono molto corte e vengono usati numeri di sequenza di 3 bit. Qual è la massima utilizzazione del canale raggiungibile per i seguenti protocolli:
 - (a) stop-and-wait
 - (b) protocollo 5
 - (c) protocollo 6.
30. Calcola la frazione di banda che è persa in ridondanza (intestazioni e ritrasmissioni) per il protocollo 6 su un canale satellitare a 50 kbps con molto traffico. I data frame sono composti da 40 bit di header e 3.960 bit di dati. Assumiamo che il tempo di propagazione da terra al satellite sia 270 millisecondi. Non ci sono frame di tipo ACK, mentre i NAK sono lunghi 40 bit. La frequenza degli errori per i frame di dati è 1%, mentre per i NAK è trascurabile. I numeri di

sequenza sono da 8 bit.

31. Consideriamo un canale satellitare da 64 kbps e privo di errori, usato per trasmettere dei frame dati da 512 byte in una direzione, con degli acknowledgement molto corti nella direzione inversa. Qual è la massima velocità di trasmissione per finestre di dimensioni 1, 7, 15 e 127? Il tempo di propagazione dalla terra al satellite è 270 millisecondi.
32. Un cavo di lunghezza 100 km trasmette i dati come una linea T1. La velocità di propagazione nel cavo è 2/3 la velocità della luce nel vuoto. Quanti bit stanno dentro al cavo?
33. Supponiamo di costruire un modello del protocollo 4 con una macchina a stati finiti. Quanti sono gli stati per ogni macchina? Quanti per il canale di comunicazione? Quanti sono gli stati per il sistema completo (le due macchine e il canale)? Ignoriamo gli errori di checksum.
34. Sia data la sequenza degli scatti per la rete di Petri della Figura 3.23, che corrisponde alla sequenza (000), (01A), (01-), (010), (01A) della Figura 3.21. Spiegare a parole che cosa rappresenta la sequenza.
35. Disegnare la rete di Petri corrispondente alle regole di transizione AC → B, B → AC, CD → E ed E → CD. Dalla rete di Petri, ricavare il grafo degli stati finiti raggiungibili dallo stato iniziale ACD. Quale noto concetto viene rappresentato da queste regole di transizione?
36. PPP è basato in gran parte su HDLC, che usa il bit stuffing per evitare che flag byte all'interno dei dati possano causare confusione. Trovare almeno un motivo per cui PPP usa invece il byte stuffing.
37. Qual è la minima quantità d'informazione aggiuntiva necessaria per mandare un pacchetto IP usando PPP? Siamo interessati solo alla parte PPP, quindi escludiamo l'intestazione dell'IP dal calcolo.
38. Lo scopo di questo esercizio di laboratorio è quello di implementare un meccanismo per identificare gli errori usando l'algoritmo standard di CRC descritto nel testo. Vogliamo scrivere due programmi: un generatore e un verificatore. Il generatore legge dallo standard input un messaggio di n bit rappresentato da una stringa di 0 e 1 in una linea di testo ASCII. La seconda linea è il polinomio di k bit, anch'esso in ASCII. Il generatore scrive sullo standard output una linea di testo ASCII con $n + k$ 0s e 1s, che rappresenta il messaggio da trasmettere, e poi stampa anche il polinomio, così come l'aveva letto in input. Il programma verificatore legge l'output del generatore e stampa in output un messaggio che ne indica la correttezza. Infine, scriviamo un programma, alteratore, che inverte un bit nella prima linea a seconda degli argomenti con cui è invocato (il numero di bit si conta a partire da sinistra e partendo da 1), mentre lascia invariato il resto delle due linee. Scrivendo:

```
generatore < file | verificatore
si dovrà vedere un messaggio che indica che tutto è corretto, mentre
generatore < file | alteratore arg | verificatore
si dovrà vedere un messaggio di errore.
```
39. Scrivere un programma per simulare il comportamento di una rete di Petri. Il programma deve leggere le regole di transizione e una lista di stati che corrispondono all'invio e alla ricezione di un nuovo pacchetto da parte dello strato network. Dallo stato iniziale, anch'esso accettato in input, il programma dovrà scegliere a caso delle transizioni e farle scattare. Si dovrà anche verificare se è possibile che una macchina accetti due pacchetti senza che l'altra parte ne abbia

4

Il sottostrato MAC (Medium Access Control)

Come è stato spiegato nel Capitolo 1, esistono due tipi di reti: quelle che utilizzano connessioni punto-punto e quelle che usano i canali broadcast. Questo capitolo si occupa delle reti broadcast e dei loro protocolli.

In qualsiasi rete broadcast il problema chiave è la scelta dell'entità che dovrà acquisire il diritto di utilizzo del canale in caso di competizione. Per chiarire questo punto consideriamo una riunione telefonica in cui sei persone sono munite di altrettanti telefoni, ma cablati in modo che ciascuno ascolta e parla con tutti. Non appena una persona smette di parlare, è molto probabile che altri due (o più) utenti inizieranno a parlare contemporaneamente, generando il caos. Nell'incontro faccia a faccia il caos è evitato grazie a mezzi esterni, per esempio in una riunione le persone possono alzare la mano per chiedere la parola, ma quando è disponibile un solo canale è molto più difficile determinare chi deve essere il prossimo. Sono stati sviluppati molti protocolli per risolvere il problema, che verranno esaminati in questo capitolo. Nella letteratura, i canali broadcast qualche volta sono chiamati anche **canali multiaccesso** o **canali ad accesso casuale**.

I protocolli per assegnare l'uso di un canale multiaccesso appartengono a un sottostrato dello strato collegamento dati, chiamato **MAC** (*Medium Access Control*). Il sottostrato MAC è importante soprattutto nelle LAN, poiché molte si servono per la comunicazione proprio di un canale multiaccesso; al contrario le WAN (con l'eccezione delle reti satellitari) preferiscono le connessioni punto a punto. Poiché i canali multiaccesso sono strettamente collegati alla tecnologia delle LAN, il capitolo esamina le reti locali in generale, includendo alcuni problemi che non riguardano direttamente il sottostrato MAC.

Tecnicamente parlando il sottostrato MAC è la parte inferiore dello strato collegamento dati, perciò dal punto di vista logico sarebbe stato più corretto studiarlo prima di esaminare i protocolli punto-punto descritti nel Capitolo 3. Tuttavia molti trovano più semplice comprendere i protocolli che coinvolgono diverse parti dopo aver studiato quelli che coinvolgono solo due parti, e per questo motivo si è preferito non seguire una presentazione lineare dall'alto verso il basso.

4.1 Il problema dell'assegnazione del canale

Il tema centrale affrontato in questo capitolo è il sistema utilizzato per assegnare un singolo canale broadcast a utenti in competizione tra loro. Alla descrizione generale degli schemi statici e dinamici seguirà l'esame di alcuni algoritmi specifici.

4.1.1 Assegnazione statica del canale in LAN e MAN

Per assegnare un singolo canale, per esempio una linea telefonica, tra più utenti in competizione si usa tradizionalmente il multiplexing a divisione di frequenza (FDM, *Frequency Division Multiplexing*). Se ci sono N utenti, la banda è divisa in N parti uguali (Figura 2.31) e ogni utente ne riceve una. Poiché ciascun utente usa una propria banda di frequenza, non c'è alcuna interferenza tra gli utenti. Se il numero di utenti è piccolo e costante, e ognuno ha un volume di traffico pesante (come nel caso delle centrali di commutazione interurbane), FDM è un meccanismo di assegnazione semplice ed efficiente. Quando però il numero di utenti che trasmettono informazioni è molto elevato e variabile, oppure quando il traffico è irregolare, FDM presenta alcuni problemi. Se lo spettro è diviso in N regioni, ma il numero di utenti impegnato a comunicare è inferiore a N , si spreca parte dello spettro; se invece più di N utenti cercano di comunicare alcuni non potranno farlo (perché non è disponibile la banda necessaria), anche se alcuni utenti che in precedenza hanno ricevuto una banda di frequenza non stanno ricevendo o trasmettendo alcunché.

Anche ipotizzando che il numero di utenti possa in qualche modo essere mantenuto costante a N , dividere il singolo canale disponibile in sottocanali statici è una soluzione intrinsecamente inefficiente. Il problema di fondo è che quando alcuni utenti sono inattivi, la loro banda è semplicemente persa; essi non la utilizzano e a nessun altro è consentito di farlo. Inoltre, nella maggior parte dei sistemi di computer il traffico dati è altamente variabile (rapporti di 1.000 a 1 tra picchi di traffico e traffico medio sono abbastanza comuni), perciò la maggior parte dei canali rimarrebbe quasi sempre inutilizzata.

Le scarse prestazioni del sistema FDM statico si possono facilmente evidenziare considerando un semplice calcolo di teoria delle code. Si consideri un ritardo medio T per un canale la cui capacità è pari a C bps; λ frame/sec rappresenta la frequenza degli arrivi. Ogni frame ha una lunghezza ricavata da una funzione di densità di probabilità esponenziale, che ha una media di $1/\mu$ bit/frame. Con questi parametri la frequenza degli arrivi è

pari a λ frame/sec e la frequenza di servizio è μC frame/sec. La teoria delle code dimostra che per tempi di servizio e arrivo di Poisson,

$$T = \frac{1}{\mu C - \lambda}$$

Per esempio, posto C pari a 100 Mbps, la lunghezza media del frame $1/\mu$ pari a 10.000 bit e la frequenza di arrivo dei frame λ pari a 5.000, allora $T = 200 \mu\text{sec}$. Si noti che ignorando il ritardo di accodamento e chiedendo semplicemente di calcolare il tempo impiegato per inviare un frame di 10.000 bit attraverso una rete a 100 Mbps si otterebbe la risposta (errata) 100 msec. Questo risultato varrebbe solo in caso di assenza di competizione per il canale.

Ora si divida il canale in N sottocanali indipendenti, ognuno di capacità C/N bps. La frequenza di input media su ogni sottocanale ora è pari a λ/N . Ricalcolando T si ottiene:

$$T_{\text{FDM}} = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT \quad (4.1)$$

Il ritardo medio quando si utilizza FDM è N volte peggiore del ritardo che si otterebbe se, magicamente, tutti i frame fossero ordinatamente inseriti in una grande coda centrale.

Lo stesso ragionamento che si applica a FDM vale anche per la tecnica di multiplexing a divisione di tempo (TDM). A ogni utente viene assegnato un *Nesimo* intervallo temporale; se un utente non utilizza il proprio intervallo, questo viene sprecato. Lo stesso accadrebbe se le reti venissero divise fisicamente: utilizzando l'esempio precedente, sostituendo la rete a 100 Mbps con dieci reti a 10 Mbps assegnate staticamente ai singoli utenti il ritardo medio passerebbe da 200 μsec a 2 μsec .

Poiché nessuno dei metodi tradizionali di assegnazione statica del canale funziona bene con il traffico irregolare, inizieremo a esaminare soluzioni dinamiche.

4.1.2 Assegnazione dinamica del canale in LAN e MAN

Prima di studiare il primo dei tanti metodi di assegnazione del canale descritti in questo capitolo, conviene formulare attentamente il problema dell'allocazione. Alla base di tutto il lavoro svolto in questo settore ci sono cinque premesse fondamentali.

1. **Modello della stazione.** Il modello è composto da N stazioni indipendenti (per esempio computer, telefoni o dispositivi individuali), ognuna con un programma o utente che genera frame in trasmissione. Le stazioni qualche volta vengono chiamate **terminali**. La probabilità che un frame venga generato in un intervallo di lunghezza Δt è di $\lambda \Delta t$, dove λ è un valore costante che rappresenta la frequenza di arrivo di nuovi frame. Una volta generato il nuovo frame, la stazione rimane bloccata e non fa nulla fino a che il frame non è stato trasmesso con successo.

2. **Presupposto del canale singolo.** Un solo canale assicura ogni comunicazione. Tutte le stazioni possono trasmettere attraverso questo canale e tutte possono ricevere attraverso di esso. Dal punto di vista hardware tutte le stazioni sono equivalenti, ma il software di protocollo può assegnare loro priorità diverse.
3. **Presupposto della collisione.** Due frame trasmessi contemporaneamente si sovrappongono temporalmente e il segnale finale risulta distorto. Questo evento è chiamato **collisione**. Tutte le stazioni possono rilevare le collisioni. Un frame entrato in collisione con un altro frame deve essere trasmesso un'altra volta. Non ci sono altri errori a parte quelli generati dalle collisioni.
4. (a) **Tempo continuo.** La trasmissione di frame può iniziare in qualunque istante. Non esiste un orologio di riferimento principale che divide il tempo in intervalli discreti.
 (b) **Tempo diviso in intervalli.** Il tempo è diviso in intervalli discreti (detti anche slot). La trasmissione di un frame coincide sempre con l'inizio di un intervallo. Un intervallo può contenere 0, 1 o più frame. Il primo valore rappresenta un intervallo vuoto, il secondo indica una trasmissione che ha avuto successo e il terzo una collisione.
5. (a) **Occupazione del canale verificabile.** Prima di tentare la trasmissione, le stazioni sono in grado di capire se il canale è occupato. Fintanto che il canale rimane occupato, nessuna stazione tenterà di utilizzarlo.
 (b) **Occupazione del canale non verificabile.** Le stazioni non sono in grado di capire se il canale è occupato, si limitano a trasmettere il frame. Solo successivamente possono determinare se la trasmissione ha avuto successo.

Il primo presupposto afferma che le stazioni sono indipendenti e che il lavoro è generato a una frequenza costante; segnala implicitamente che ogni stazione ha un solo programma o utente, perciò nessun nuovo lavoro viene generato mentre la stazione è bloccata. Modelli più sofisticati ammettono stazioni multiprogrammate che possono generare lavoro mentre la stazione è bloccata, ma l'analisi di queste stazioni è molto più complessa.

Il presupposto del canale singolo è il cuore del modello: non è possibile comunicare in nessun altro modo. Le stazioni non possono "alzare le mani" per chiedere il permesso di parlare. Anche il presupposto relativo alla collisione è fondamentale; in alcuni sistemi (in particolare in quelli a spettro distribuito) si ottengono risultati sorprendenti rendendo meno rigida questa ipotesi. Alcune LAN, per esempio le reti token ring, passano uno speciale "gettone" da una stazione all'altra per abilitare la trasmissione, ma i paragrafi successivi considerano solo il modello basato su un singolo canale con competizione e collisioni.

Per quanto riguarda il tempo sono possibili due ipotesi alternative: il tempo può essere continuo (4a) o diviso in intervalli (4b); le due ipotesi sono mutuamente esclusive. Alcuni sistemi

utilizzano il primo metodo, altri utilizzano il secondo, perciò è meglio analizzarli entrambi. In modo analogo, una rete può avere una capacità di rilevamento della portante (5a) oppure non averla (5b). Si noti che la parola "portante" in questo contesto fa genericamente riferimento a un segnale elettrico trasmesso attraverso il cavo. Le LAN generalmente hanno la possibilità di verificare l'occupazione del canale, mentre le reti wireless non sono in grado di svolgere questa funzione in modo efficace in quanto non è detto che ogni stazione si trovi dentro la portata di tutte le altre. Le stazioni su reti via cavo con rilevamento dell'occupazione di canale possono terminare le loro trasmissioni prematuramente, se scoprono che i dati inviati entrano in collisione con altre trasmissioni. Il rilevamento della collisione viene utilizzato raramente sulle reti wireless, per motivi ingegneristici.

4.2 Protocolli ad accesso multiplo

Esistono molti algoritmi per assegnare un canale ad accesso multiplo; i paragrafi successivi esaminano alcuni dei più interessanti e spiegano come sono usati.

4.2.1 ALOHA

Negli anni '70, Norman Abramson e i suoi colleghi dell'Università delle Hawaii idearono un metodo innovativo ed elegante per risolvere il problema dell'assegnazione del canale, che nel corso degli anni è stato sviluppato da molti altri ricercatori (Abramson, 1985). Il progetto di Abramson, chiamato sistema ALOHA, utilizza la trasmissione radio broadcast basata su stazioni terrestri, ma l'idea di fondo può essere applicata a qualunque sistema dove utenti non coordinati competono tra loro per utilizzare un singolo canale condiviso. Questo paragrafo esamina due versioni di ALOHA: quella pura e quella slotted. Nella seconda il tempo è diviso in intervalli discreti dove tutti i frame devono inserirsi. ALOHA puro non richiede una sincronizzazione temporale, slotted ALOHA invece sì.

ALOHA puro

L'idea di fondo del sistema ALOHA è semplice: consentire agli utenti di trasmettere ogni volta che hanno dati da inviare. Naturalmente ci potranno essere collisioni che danneggeranno i frame, ma grazie alla proprietà di feedback della trasmissione broadcast, un trasmettitore potrà sempre scoprire, ascoltando il canale, se il suo frame è andato distrutto. Nel caso di una LAN, il feedback è immediato; con un satellite trascorrono 270 msec prima che il trasmettitore scopra se la trasmissione ha avuto successo. Se per qualche motivo non è possibile ascoltare il canale durante la trasmissione, si deve adottare un sistema di acknowledge. Se il frame è stato distrutto, il trasmettitore rimane in attesa per un intervallo casuale prima di ripetere la trasmissione. Il tempo di attesa deve essere casuale, altrimenti gli stessi frame continueranno a collidere in un ciclo senza fine. I sistemi dove più utenti condividono un canale comune in un modo che può causare conflitto si dicono sistemi a contesa.

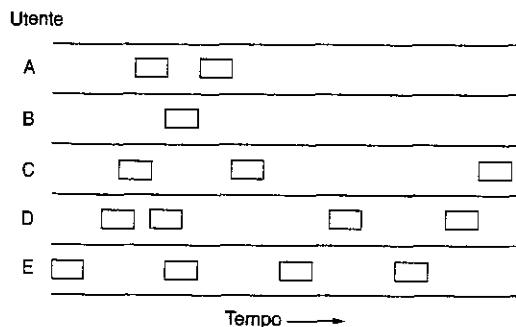


Figura 4.1. Nel sistema ALOHA puro i frame sono trasmessi in momenti totalmente arbitrari.

La Figura 4.1 mostra lo schema di generazione dei frame in un sistema ALOHA. Tutti i frame hanno la stessa lunghezza perché la capacità di trasporto dei sistemi ALOHA è massima quando si utilizza un frame con dimensione uniforme.

Ogni volta che due frame tentano di occupare contemporaneamente il canale si verifica una collisione che danneggia entrambi gli elementi. Basta che il primo bit di un nuovo frame si sovrapponga all'ultimo bit di un frame quasi completato, per considerarli entrambi totalmente distrutti e quindi da ritrasmettere. Il checksum non sa (né deve) distinguere tra perdita totale e successo parziale della trasmissione.

Ma qual è l'efficienza di un canale ALOHA, ossia che frazione dei frame trasmessi sfugge alla collisione in queste circostanze caotiche? Si consideri prima di tutto un gruppo infinito di utenti interattivi seduti davanti ai loro computer (stazioni). Un utente si trova sempre in uno di due stati: sta scrivendo oppure è in attesa. Inizialmente tutti gli utenti si trovano nel primo stato, ossia stanno scrivendo. Quando termina una riga, l'utente smette di scrivere e attende una risposta; a questo punto la stazione trasmette un frame che contiene la riga e controlla il canale per scoprire se la trasmissione ha avuto successo. In caso positivo, l'utente vede apparire la risposta e ricomincia a scrivere; in caso negativo, l'utente continua ad aspettare e il frame è ritrasmesso fino a quando la trasmissione non ha successo.

Si supponga che il tempo di frame indichi la quantità di tempo richiesta per trasmettere un frame di lunghezza standard e fissa (pertanto è pari alla lunghezza del frame divisa per il bit rate). A questo punto si può ammettere per ipotesi che la popolazione infinita di utenti generi nuovi frame in accordo con la distribuzione di Poisson, con media di N frame per tempo di frame. Il presupposto della popolazione infinita garantisce che N non diminuisce quando gli utenti cominciano a essere bloccati. Se $N > 1$, la comunità di utenti sta generando frame a una frequenza più elevata della frequenza che il canale è in grado di gestire, e quasi ogni frame subirà una collisione. Per garantire una ragionevole capacità di trasporto si dovrà assumere $0 < N < 1$.

Oltre ai nuovi frame, le stazioni ritrasmettono anche i frame entrati precedentemente in collisione. Supponiamo di Poisson anche la probabilità di k tentativi di trasmissione per tempo di frame (il vecchio e il nuovo combinati insieme), con media G per tempo di frame.

Chiaramente, $G \geq N$. A basso carico ($N \approx 0$) ci saranno poche collisioni e di conseguenza poche ritrasmissioni, perciò $G \approx N$. A carico elevato ci saranno molte collisioni, perciò $G > N$. Se G è il carico che si presenta, qualunque sia il carico la capacità di trasporto S sarà G volte la probabilità P_0 di una trasmissione che ha successo, ossia $S = GP_0$, dove P_0 è la probabilità che un frame non entri in collisione con un altro frame.

Un frame non entra in collisione quando nessun altro è trasmesso nello stesso intervallo temporale, come mostrato nella Figura 4.2. In quali condizioni il frame evidenziato arriverà a destinazione senza riportare alcun danno? Si indichi con t il tempo richiesto per inviare il frame: la parte iniziale del frame evidenziato entra in collisione con la fine di un qualsiasi altro frame generato da un utente nel tempo compreso tra t_0 e $t_0 + t$.

Il destino del frame evidenziato era segnato già prima che iniziasse la trasmissione del primo bit, ma poiché in un sistema ALOHA puro una stazione non ascolta il canale prima d'iniziare a trasmettere, non c'è modo di sapere che un altro frame è già stato inviato. Analogamente, i frame che inizieranno nel periodo compreso tra $t_0 + t$ e $t_0 + 2t$ incapperranno sicuramente nella fine del frame evidenziato.

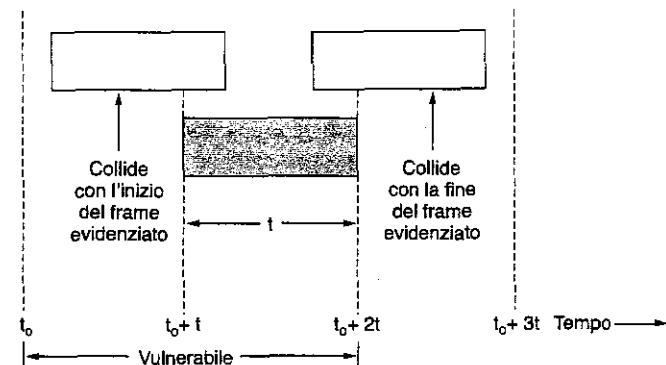


Figura 4.2. Il periodo vulnerabile del frame evidenziato.

La probabilità che k frame siano generati durante un dato tempo di frame è data dalla distribuzione di Poisson:

$$\Pr[k] = \frac{G^k e^{-G}}{k!} \quad (4.2)$$

perciò la probabilità di zero frame è e^{-G} . In un intervallo lungo due tempi frame, il numero medio di frame generati è pari a $2G$. La probabilità che nessun altro traffico inizi durante l'intero periodo vulnerabile è perciò dato da $P_0 = e^{-2G}$. Usando $S = GP_0$ si ottiene

$$S = Ge^{-2G}$$

La relazione tra il traffico che si presenta e la capacità di trasporto è mostrata nella Figura 4.3. La capacità di trasporto massima si ha per $G = 0,5$ con $S = 1/2e$, ed è circa 0,184. In altre parole, al massimo si può sperare di utilizzare il 18% del canale. Questo risultato non

è molto incoraggiante, ma se tutti trasmettono a piacimento è difficile aspettarsi una percentuale di successo del cento per cento.

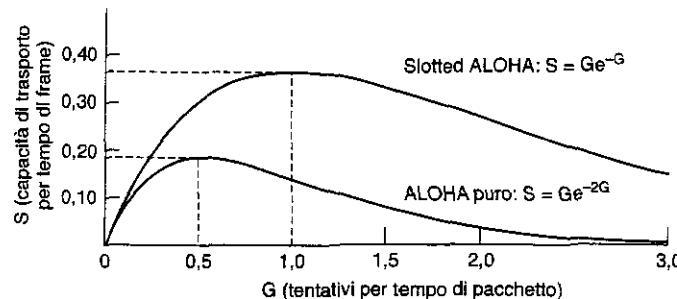


Figura 4.3. La capacità di trasporto in funzione del traffico per un sistema ALOHA.

Slotted ALOHA

Nel 1972, Roberts pubblicò un metodo per raddoppiare la capacità di un sistema ALOHA. Egli propose di dividere il tempo in intervalli discreti, dove ogni intervallo corrisponde a un frame. Questo approccio richiede che gli utenti concordino i limiti degli intervalli, e la necessaria sincronizzazione si può soddisfare con una speciale stazione che emette un segnale all'inizio di ogni intervallo, proprio come un orologio. Nel metodo di Roberts, conosciuto anche come sistema **slotted ALOHA**, c'è una differenza importante rispetto al metodo di Abramson: un computer non può inviare dati ogni volta che viene premuto il tasto di avanzamento riga. Quando l'utente completa una riga il computer deve attendere l'inizio dell'intervallo (slot) successivo, e ciò trasforma il sistema continuo ALOHA puro in un sistema discreto. Poiché ora il periodo vulnerabile è dimezzato, la probabilità che non ci sia altro traffico durante lo stesso intervallo occupato dal frame di prova è pari a e^{-G} , quindi:

$$S = Ge^{-G} \quad (4.3)$$

Come si può notare osservando la Figura 4.3, slotted ALOHA ha il massimo a $G = 1$, con una capacità di trasporto pari a $S = 1/e$ che equivale a circa 0,368, il doppio del valore di ALOHA puro. Se il sistema sta operando a $G = 1$, la probabilità che si presenti un intervallo vuoto è di 0,368 (equazione 4.2). La migliore situazione che si può ottenere con slotted ALOHA prevede il 37% di intervalli vuoti, il 37% di trasmissioni senza errori e il 26% di collisioni. Operando con valori di G più alti si riduce il numero di intervalli vuoti ma aumenta esponenzialmente il numero di collisioni.

Per osservare con quanta rapidità può crescere il numero di collisioni si consideri la trasmissione di un frame di prova: la probabilità che esso eviti la collisione è e^{-G} , valore uguale alla probabilità che tutti gli altri utenti non trasmettano nello stesso intervallo; la

probabilità di una collisione è allora $1 - e^{-G}$; la probabilità che una trasmissione richieda esattamente k tentativi ($k - 1$ collisioni seguite da un successo) sarà:

$$P_k = e^{-G}(1 - e^{-G})^{k-1}$$

Il numero atteso di trasmissioni, E , necessarie per ogni volta che si batte il carattere di fine riga è quindi pari a:

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^G$$

A causa della dipendenza esponenziale di E da G , piccole variazioni nel carico del canale possono ridurre drasticamente le sue prestazioni.

Slotted ALOHA è importante per un motivo che potrebbe non apparire subito ovvio. Concepito negli anni '70, venne utilizzato da alcuni sistemi sperimentali per poi essere quasi del tutto dimenticato. Quando fu inventato il metodo di accesso a Internet via cavo, improvvisamente si presentò il problema di allocare un canale condiviso da più utenti in competizione, e slotted ALOHA venne tirato fuori dal cestino dei rifiuti per salvare la situazione.

È accaduto spesso che protocolli perfettamente validi cadessero in disuso per motivi politici, per poi essere riscoperti anni dopo da persone intelligenti chiamate a risolvere problemi contingenti. Per questo motivo il capitolo, oltre a descrivere i protocolli più comuni, esamina diverse soluzioni eleganti che oggi non sono utilizzate in modo esteso ma che potrebbero diventare molto utili in applicazioni future, se i progettisti delle reti si ricorderanno della loro esistenza.

4.2.2 Protocolli ad accesso multiplo con rilevamento della portante

Con slotted ALOHA si può ottenere un utilizzo massimo del canale di $1/e$. Questo risultato non deve sorprendere, poiché con stazioni che trasmettono a piacimento senza verificare le azioni intraprese dalle altre è naturale avere un numero elevato di collisioni. Fortunatamente nelle reti locali le stazioni sono in grado di rilevare le azioni intraprese dalle altre, e adattare di conseguenza il proprio comportamento. Queste reti possono raggiungere un livello di utilizzo maggiore di $1/e$. Il paragrafo descrive alcuni protocolli che consentono di migliorare le prestazioni.

I protocolli in cui le stazioni rimangono in ascolto di una portante (ossia di una trasmissione) e si comportano di conseguenza sono chiamati **protocolli con rilevamento della portante** (*carrier sense protocol*), e negli anni ne sono stati proposti molti. Kleinrock e Tobagi (1975) hanno analizzato in dettaglio diversi protocolli di questo tipo.

CSMA persistente e non persistente

Il primo protocollo con rilevamento della portante analizzato in questo capitolo si chiama **CSMA (Carrier Sense Multiple Access) 1-persistente**. Quando ha dei dati da trasmettere,

una stazione prima di tutto ascolta il canale per scoprire se qualcun altro in quel momento sta trasmettendo. Se il canale è occupato, la stazione aspetta fino a quando non si libera. Quando si accorge che il canale è libero trasmette un frame; in caso di collisione, la stazione rimane in attesa per un intervallo casuale prima di ritentare la trasmissione. Il protocollo è chiamato 1-persistente perché la stazione trasmette con una probabilità di 1 quando trova il canale libero.

Il ritardo di propagazione ha un effetto importante sulle prestazioni del protocollo: c'è infatti una piccola possibilità che subito dopo l'inizio di una trasmissione da parte di una stazione, un'altra stazione sia pronta a inviare dati e controlli il canale. Se il segnale della prima stazione non ha ancora raggiunto la seconda, quest'ultima potrebbe ritenere il canale libero e iniziare perciò a trasmettere i propri dati, causando una collisione. Più è lungo il ritardo di propagazione, più diventa importante l'effetto e peggiori le prestazioni del protocollo.

Le collisioni continuano a verificarsi anche se il ritardo di propagazione è uguale a zero. Se due stazioni diventassero pronte mentre una terza sta trasmettendo, entrambe attenderebbero educatamente il termine della trasmissione prima di iniziare la loro emissione simultanea; il risultato finale, anche in questo caso, sarebbe una collisione. Se non fossero così impazienti ci sarebbero meno collisioni. Anche così, questo protocollo è migliore del sistema ALOHA puro perché entrambe le stazioni hanno la decenza di desistere dall'interferire con il frame della terza stazione. Intuitivamente, questo approccio conduce a prestazioni migliori di quelle del sistema ALOHA puro, proprio come accade con slotted ALOHA.

Il secondo protocollo con rilevamento della portante si chiama **CSMA non persistente**. In questo protocollo si tenta consapevolmente di essere meno ingordi. Prima di trasmettere, ogni stazione controlla il canale: se nessun altro sta trasmettendo inizia a inviare i dati, ma se il canale è occupato la stazione non esegue un controllo continuo per trasmettere subito il proprio frame; invece attende per un intervallo di tempo casuale prima di ripetere l'algoritmo. Di conseguenza, questo meccanismo permette di utilizzare meglio il canale ma allunga i ritardi.

L'ultimo protocollo si chiama **CSMA p-persistente**; si applica ai canali divisi in intervalli temporali e funziona così: quando è pronta a trasmettere, ogni stazione controlla il canale. Se lo trova libero, trasmette subito con una probabilità p , e rimanda fino all'intervallo successivo con probabilità $q = 1 - p$. Se anche quell'intervallo risulta libero, la stazione trasmette oppure rimanda un'altra volta (anche in questo caso le probabilità sono p e q). Il processo si ripete fino a quando il frame non è stato trasmesso o un'altra stazione non inizia a trasmettere; in questo caso, la stazione sfortunata si comporta come se ci fosse stata una collisione (ossia attende per un intervallo di tempo casuale e poi ricomincia). Se inizialmente rileva che il canale è occupato, la stazione attende fino all'intervallo successivo e poi applica l'algoritmo appena descritto. La Figura 4.4 mostra la capacità di trasporto calcolata rispetto al traffico atteso per tutti e tre i protocolli e per i sistemi ALOHA.

CSMA con rilevamento delle collisioni

I protocolli CSMA persistente e CSMA non persistente sono chiaramente un miglioramento rispetto ad ALOHA, in quanto garantiscono che nessuna stazione trasmette se il

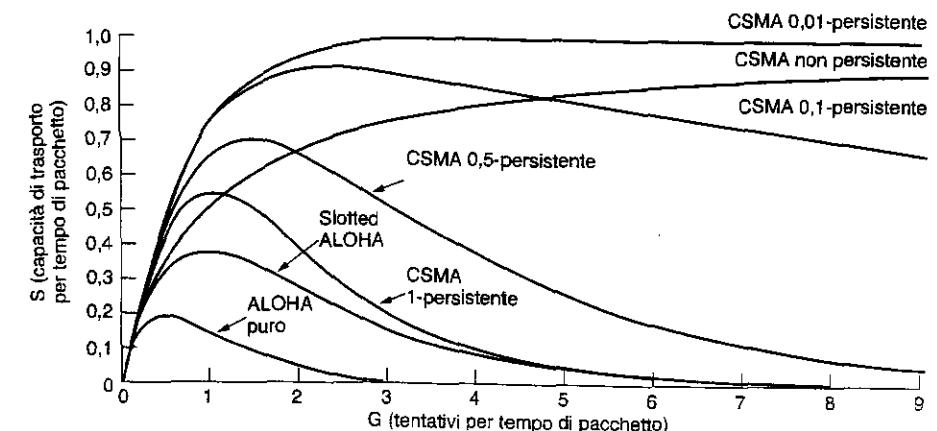


Figura 4.4. Confronto tra gli utilizzi del canale in funzione del carico per protocolli ad accesso casuale.

canale risulta occupato. Un nuovo miglioramento si ottiene consentendo a ogni stazione di annullare la propria trasmissione in caso di collisione. In altre parole, se due stazioni, dopo aver controllato il canale, iniziano a trasmettere contemporaneamente, entrambe rileveranno la collisione quasi immediatamente. Invece di completare la trasmissione dei rispettivi frame, oramai irrimediabilmente danneggiati, le stazioni interrompono bruscamente la trasmissione. La terminazione rapida dei frame danneggiati risparmia tempo e banda. Questo protocollo, chiamato **CSMA/CD** (*CSMA con Collision Detection*), è ampiamente utilizzato nel sottostrato MAC delle LAN e in particolare costituisce la base delle famose LAN Ethernet, perciò vale la pena esaminarlo in dettaglio.

CSMA/CD, come molti altri protocolli per LAN, utilizza il modello concettuale rappresentato nella Figura 4.5. Nel punto contrassegnato dall'etichetta t_0 una stazione ha appena finito di trasmettere il suo frame; ora qualunque altra stazione che ha un frame da inviare può tentare di trasmettere i dati. Se due o più stazioni decidono contemporaneamente di trasmettere si verifica una collisione. Le collisioni possono essere rilevate confrontando con il segnale trasmesso la potenza o la larghezza d'impulso del segnale ricevuto.

Se rileva una collisione, la stazione interrompe la sua trasmissione, attende un periodo di tempo casuale e poi ritenta (ammesso che nessun'altra stazione abbia nel frattempo avviato la trasmissione). Perciò il modello di CSMA/CD è composto da periodi di contesa alternati a periodi di trasmissione, con periodi inattivi che si presentano quando tutte le stazioni sono silenziose (per mancanza di lavoro).

È utile esaminare più in dettaglio l'algoritmo di contesa. Si supponga che due stazioni inizino entrambe a trasmettere esattamente al tempo t_0 . Quanto tempo passerà prima che le stazioni realizzino che è avvenuta una collisione? La risposta è vitale perché permette di determinare la lunghezza del periodo di contesa, e di conseguenza i valori del ritardo e della capacità di trasporto. Il tempo minimo per rilevare una collisione è pari al tempo impiegato dal segnale per propagarsi da una stazione all'altra.

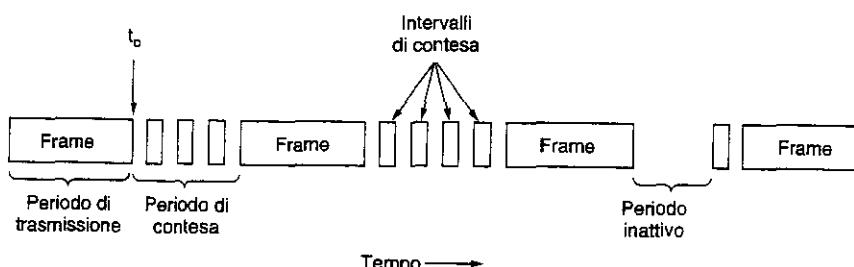


Figura 4.5. CSMA/CD può essere in uno di questi stati: contesa, trasmissione o inattivo.

In base a questo ragionamento si potrebbe pensare che se una stazione, dopo aver avviato la propria trasmissione, non rileva una collisione per un tempo uguale al tempo massimo di propagazione del cavo, può essere certa di aver assunto il controllo del cavo. In altre parole, dopo il tempo indicato la stazione sarà certa che tutte le altre stazioni non interferiranno con la trasmissione in corso. Questa conclusione è sbagliata. Consideriamo lo scenario che descrive la situazione peggiore che può presentarsi: si supponga che τ sia il tempo che il segnale impiega per propagarsi tra le due stazioni più lontane; nell'istante t_0 una stazione inizia a trasmettere e a $\tau - \epsilon$ (un istante prima che il segnale arrivi alla stazione più distante), anche questa inizia a trasmettere. La stazione logicamente rileva la collisione, e interrompe l'operazione quasi istantaneamente, ma il piccolo picco di rumore causato dalla collisione impiega un tempo pari a $2\tau - \epsilon$ per tornare indietro alla stazione originale.

In altre parole, nel caso peggiore una stazione può essere certa di aver assunto il controllo del canale solo se non rileva alcuna collisione per un tempo pari a 2τ . Per questo motivo l'intervallo di contesa viene rappresentato come un sistema slotted ALOHA con intervalli lunghi 2τ . Su un cavo coassiale lungo 1 Km, $\tau \approx 5 \mu\text{sec}$. Per semplicità, si supponga che ogni intervallo contenga solo 1 bit. Dopo aver acquisito il controllo del canale una stazione può trasmettere alla velocità che desidera, senza limitarsi alla velocità di 1 bit ogni 2τ sec.

È importante capire che il rilevamento della collisione è un processo *analogico*. L'hardware della stazione deve controllare il cavo durante la propria trasmissione; se ciò che rileva non è quello che ha trasmesso, la stazione sa che è avvenuta una collisione. L'implicazione è che la codifica del segnale deve permettere il rilevamento delle collisioni (per esempio, una collisione di due segnali da 0 Volt può essere impossibile da rilevare); per questo motivo di solito si utilizza una codifica speciale.

Vale la pena di notare che una stazione trasmittente deve continuamente controllare il canale, ascoltando i picchi di rumore che potrebbero indicare le collisioni; per questo motivo CSMA/CD su singolo canale è intrinsecamente un sistema half duplex. È impossibile per una stazione trasmettere e ricevere frame nello stesso momento, perché durante ogni trasmissione è attiva la logica che cerca di individuare le collisioni.

Per evitare fraintendimenti, è utile notare che nessun protocollo del sottostrato MAC garantisce una consegna affidabile: anche in assenza di collisioni, il ricevitore potrebbe copiare il frame non correttamente per diverse ragioni (per esempio per mancanza di spazio di memorizzazione o per un interrupt perduto).

4.2.3 Protocolli senza collisione

Una volta che la stazione ha acquisito il canale con certezza, con CSMA/CD non ci possono essere collisioni, ma questo problema può ancora presentarsi durante il periodo di contesa. Le collisioni influenzano sfavorevolmente le prestazioni del sistema, specialmente quando il cavo è lungo e i frame sono corti; inoltre CSMA/CD non si può adottare ovunque.

Questo paragrafo esamina alcuni protocolli che risolvono la contesa per il canale senza generare alcuna collisione, nemmeno durante il periodo di contesa. La maggior parte di questi protocolli non è utilizzata nei sistemi più celebri, comunque è utile poter disporre di alcuni protocolli con eccellenti proprietà in un settore in continuo cambiamento.

Nei protocolli descritti di seguito si presuppone che ci siano N stazioni, ognuna associata a un indirizzo univoco che varia da 0 a $N - 1$. Non importa se alcune stazioni possono essere inattive per parte del tempo. Si presuppone anche che il ritardo di propagazione sia trascurabile. La domanda di base è sempre la stessa: "a quale stazione deve essere assegnato il canale dopo una trasmissione che ha avuto successo?". La spiegazione continua a utilizzare il modello rappresentato nella Figura 4.5 basato sugli intervalli di contesa discreti.

Un protocollo a mappa di bit

Nel primo protocollo senza collisione, chiamato **metodo a mappa di bit elementare**, ogni periodo di contesa è composto da esattamente N intervalli. Se la stazione 0 ha un frame da inviare, trasmette un bit 1 durante l'intervallo 0. A nessun'altra stazione è concesso di trasmettere durante questo intervallo. Incaricata di quello che sta facendo la stazione 0, la stazione 1 ha la possibilità di trasmettere un 1 durante l'intervallo 1, ma solo se ha un frame accodato. In generale, la stazione j può annunciare il possesso di un frame da inviare inserendo un bit 1 nello slot j . Una volta trascorsi tutti gli N intervalli, ogni stazione sa quali solo le stazioni che vogliono trasmettere; a questo punto, le stazioni iniziano a trasmettere in ordine numerico (Figura 4.6).

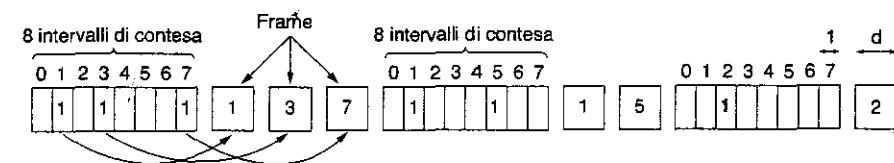


Figura 4.6. Il protocollo a mappa di bit elementare.

Poiché tutti sono d'accordo su chi sarà il prossimo, non ci sarà mai alcuna collisione. Dopo che l'ultima stazione pronta ha trasmesso il proprio frame (evento che può essere monitorato facilmente da tutte le stazioni) ha inizio un altro periodo di contesa di N bit. Se diventa pronta quando il proprio intervallo è ormai scaduto, la stazione deve rimanere in silenzio fino a quando tutte le altre non hanno avuto una possibilità e la mappa di bit non ricomincia daccapo. I protocolli di questo tipo, in cui il desiderio di

trasmettere è comunicato a tutti prima di iniziare la trasmissione vera e propria, si chiamano **protocolli a prenotazione**.

Analizziamo brevemente le prestazioni di questo protocollo. Per comodità, si assume come unità di misura del tempo la durata dell'intervallo di un bit di contesa, con frame dati composti da d unità temporali. Ipotizzando un basso livello di carico, la mappa di bit si ripeterà in continuazione per mancanza di frame di dati.

Si consideri la situazione dal punto di vista di una stazione identificata da un numero basso, per esempio 0 o 1. In media, quando questa stazione sarà pronta a trasmettere l'intervallo corrente si troverà più o meno al centro della mappa di bit, e prima di poter trasmettere la stazione dovrà attendere $N/2$ intervalli (tempo richiesto per completare l'analisi corrente) e poi altri N intervalli (tempo richiesto per eseguire l'analisi successiva). Le aspettative delle stazioni identificate da numeri alti sono più luminose; in generale queste stazioni potranno trasmettere dopo solo mezza analisi ($N/2$ intervalli). Le stazioni con numeri molto grandi raramente devono aspettare l'analisi successiva. Poiché le stazioni con numeri piccoli devono attendere in media $1,5N$ intervalli e le stazioni con numeri più grandi devono attendere in media $0,5N$ intervalli, la media per tutte le stazioni è pari a N intervalli. È facile calcolare l'efficienza del canale a basso carico: i bit usati per il controllo sono N per frame, la quantità di dati è pari a d bit e l'efficienza è uguale a $d/(N + d)$.

Se il carico è elevato, ossia se tutte le stazioni hanno sempre qualcosa da trasmettere, il periodo di contesa di N bit è distribuito proporzionalmente su N frame, quindi un solo bit per frame è usato per il controllo e l'efficienza è $d/(d + 1)$. Il ritardo medio per un frame è uguale al tempo di accodamento dentro la stazione più un tempo aggiuntivo pari a $N(d + 1)/2$ una volta che il frame ha raggiunto la prima posizione nella coda interna.

Conteggio binario

Un problema del protocollo a mappa di bit elementare è che serve 1 bit di controllo per stazione, perciò non si adatta molto bene alle reti composte da migliaia di stazioni. Un risultato migliore si può ottenere usando indirizzi binari. Una stazione che desidera utilizzare il canale deve comunicare a tutti il proprio indirizzo sotto forma di stringa binaria, partendo dal bit di ordine più elevato. Tutti gli indirizzi hanno la stessa lunghezza. I bit che occupano la stessa posizione negli indirizzi di stazioni diverse sono elaborati mediante l'operatore logico booleano OR. Questo tipo di protocollo, chiamato **conteggio binario**, è stato utilizzato in Datakit (Fraser, 1987). Si presuppone implicitamente che i ritardi di trasmissione siano trascurabili, in modo che tutte le stazioni possano vedere nello stesso momento i bit dichiarati.

Per evitare conflitti si deve applicare una regola di arbitraggio: la stazione rinuncia non appena si accorge che è stata sovrascritta da un 1 una posizione di bit di ordine elevato che nel proprio indirizzo vale 0. Per esempio, se le stazioni 0010, 0100, 1001 e 1010 vogliono il canale, durante il primo tempo di bit trasmettono rispettivamente 0, 0, 1 e 1; il risultato ottenuto unendo questi valori binari mediante l'operatore logico OR è 1. Le stazioni 0010 e 0100 notano l'1 e capiscono che una stazione con un numero più grande sta correndo per il canale, perciò si ritirano dal turno corrente. Le stazioni 1001 e 1010 invece vanno avanti.

Il bit successivo è 0 ed entrambe continuano. Il terzo bit è 1, perciò la stazione 1001 si arrende. Il vincitore alla fine sarà 1010, perché è questa la stazione che ha l'indirizzo più alto. Dopo aver vinto la gara la stazione può trasmettere un frame, e al termine della trasmissione ricomincia un nuovo turno. Il protocollo, descritto nella Figura 4.7, ha la proprietà che le stazioni con il numero più alto hanno una priorità maggiore rispetto alle stazioni con il numero più basso; questa caratteristica può essere utile o svantaggiosa in relazione al contesto applicativo.

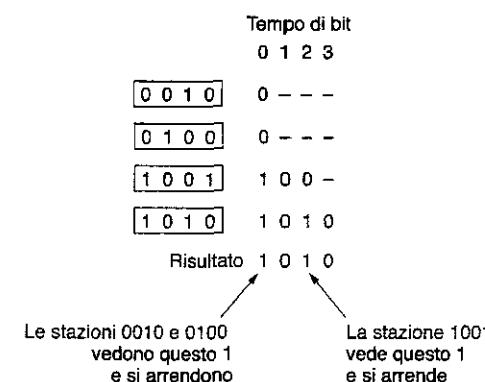


Figura 4.7. Il protocollo a conteggio binario. I trattini indicano i periodi di silenzio.

Con questo metodo l'efficienza del canale è pari a $d/(d + \log_2 N)$, ma se il formato del frame viene scelto con cura in modo che l'indirizzo del mittente costituisca il primo campo del frame, si recuperano anche questi $\log_2 N$ bit e l'efficienza raggiunge il cento per cento. Mok e Ward (1979) hanno descritto una variante del sistema a conteggio binario che utilizza un'interfaccia parallela al posto di quella seriale. Suggeriscono inoltre d'identificare le stazioni con numeri virtuali che variano da 0 fino alla stazione che ha avuto successo; i numeri devono essere permutati circolarmente dopo ogni trasmissione in modo da dare la priorità più elevata alle stazioni che sono rimaste in silenzio più a lungo. Per esempio, se le priorità delle stazioni C, H, D, A, G, B, E e F sono rispettivamente 7, 6, 5, 4, 3, 2, 1 e 0, dopo aver trasmesso con successo un frame, la stazione D viene sposata alla fine della lista; il nuovo ordine di priorità sarà quindi C, H, A, G, B, E, F e D. Perciò C rimane la stazione virtuale numero 7, ma A si sposta verso l'alto di una posizione e D finisce dalla terza all'ultima posizione. La stazione D ora potrà acquisire il canale solo se nessun'altra stazione lo richiede. Il conteggio binario è un esempio di protocollo semplice, elegante ed efficiente in attesa di essere riscoperto. Chissà che un giorno non ottenga gli onori dovuti.

4.2.4 Protocolli a contesa limitata

Fino a questo momento sono state considerate due strategie di base adottate per acquisire il controllo del canale in una rete cablata: il metodo della contesa, come in CSMA, e il metodo senza collisioni. Ogni strategia può essere giudicata determinando due importan-

ti misure di prestazioni, ossia il ritardo a basso carico e l'efficienza del canale a carico elevato. Nelle situazioni di carico leggero, la contesa (per esempio ALOHA puro o quello a intervalli) è preferibile per il suo basso ritardo. Al crescere del carico, però, la contesa diventa sempre più inefficiente perché la quantità di dati necessari all'arbitraggio del canale aumenta. Con i protocolli senza collisioni si verifica esattamente il contrario: a basso carico hanno un ritardo elevato, ma al crescere del carico l'efficienza del canale migliora (invece di peggiorare come accade per i protocolli a contesa).

Sarebbe bello se si potessero combinare le proprietà dei due tipi di protocolli, in modo da creare uno capace di usare il metodo della contesa per ottenere un ritardo limitato a basso carico, e il metodo senza collisioni per raggiungere una buona efficienza di canale nelle situazioni a carico più elevato. Protocolli di questo tipo esistono realmente, si chiamano **protocolli a contesa limitata** e concludono il nostro studio delle reti con controllo di portante.

I protocolli a contesa prima esaminati si basano su un meccanismo simmetrico, cioè ogni stazione tenta di acquisire il canale con una probabilità p che risulta identica per tutte. In maniera abbastanza interessante, le prestazioni generali del sistema qualche volta migliorano usando un protocollo che assegna a ogni stazione una probabilità diversa.

Prima di esaminare i protocolli asimmetrici è utile rivedere rapidamente le prestazioni del caso simmetrico. Si supponga che k stazioni cerchino di accedere contemporaneamente al canale. Ognuna ha una probabilità p di riuscire a trasmettere durante ciascun intervallo. La probabilità che una stazione si appropri del canale durante un dato intervallo è quindi $kp(1-p)^{k-1}$. Per trovare il valore ottimale di p si deve differenziare rispetto a p , impostare il risultato a 0 e risolvere per p . Si scopre così che il miglior valore di p è $1/k$. Sostituendo $p = 1/k$, si ottiene

$$\Pr[\text{successo con } p \text{ ottimale}] = \left[\frac{k-1}{k} \right]^{k-1} \quad (4.4)$$

Questa probabilità è rappresentata graficamente nella Figura 4.8. Se il numero di stazioni è piccolo le possibilità di successo sono buone, ma non appena le stazioni diventano 5 la probabilità diminuisce bruscamente fino a sfiorare il valore asintotico di $1/e$. Osservando la Figura 4.8 diventa evidente che la probabilità che alcune stazioni acquisiscano il controllo del canale può essere aumentata solo diminuendo il livello di competizione. I protocolli a contesa limitata fanno precisamente questo. Prima di tutto dividono le stazioni in gruppi (non necessariamente disgiunti); solo ai membri del gruppo 0 è permesso competere per l'intervallo 0. Se uno di loro vince, il membro acquisisce il controllo del canale e trasmette il frame. Se l'intervallo rimane inutilizzato o se c'è una collisione, i membri del gruppo 1 si contendono l'intervallo 1 e così via. Dividendo le stazioni in gruppi in modo appropriato è possibile ridurre il livello di contesa per ogni intervallo in modo da operare sempre nella parte più a sinistra del grafico mostrato nella Figura 4.8.

Il trucco è capire come assegnare le stazioni agli intervalli. Prima di esaminare il caso generale è utile considerare alcuni casi particolari. A un estremo abbiamo il caso in cui ogni gruppo è composto da un unico membro. Questa ripartizione garantisce l'assenza di collisioni, poiché una sola stazione potrà contendere un dato intervallo. Un protocollo di

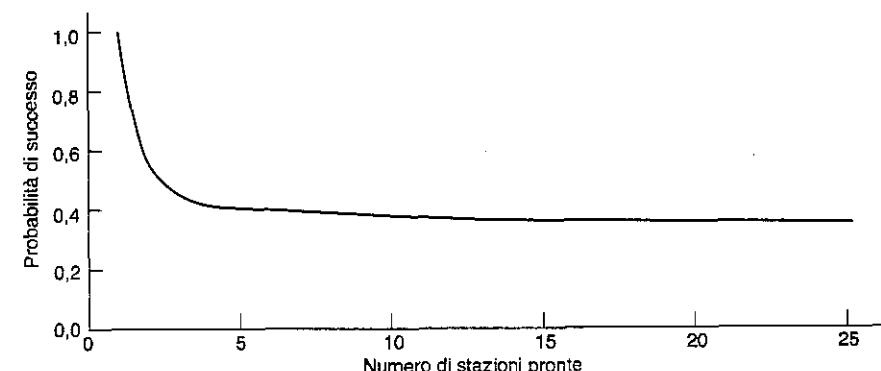


Figura 4.8. Probabilità di acquisizione di un canale a contesa simmetrica.

questo tipo è già stato descritto, e si tratta del protocollo a conteggio binario. Il successivo caso particolare è quello in cui si assegnano due stazioni per gruppo; la probabilità che entrambe tentino di trasmettere durante un intervallo è p^2 ; per piccoli valori di p questa quantità è trascurabile.

Al crescere del numero di stazioni assegnate allo stesso intervallo aumenta la probabilità di una collisione, ma il tempo necessario per analizzare la mappa di bit diminuisce. Il caso limite è un singolo gruppo contenente tutte le stazioni (come nel caso di ALOHA). Serve pertanto un meccanismo che consenta di assegnare dinamicamente le stazioni agli intervalli, in modo da assegnare a ogni intervallo molte stazioni quando il carico è limitato, e poche stazioni (o addirittura una sola) quando il carico è elevato.

Il protocollo Adaptive Tree Walk

Un modo particolarmente semplice per eseguire l'assegnazione si basa sullo stesso algoritmo che l'esercito statunitense utilizzava durante la Seconda Guerra Mondiale per verificare se i soldati erano affetti da lue (Dorfman, 1943). In breve, l'esercito prendeva un campione di sangue da N soldati; una parte di ogni campione era versata in una singola provetta. Questa miscela di campioni era quindi analizzata per verificare la presenza di anticorpi. Se non si trovavano anticorpi, tutti i soldati di quel gruppo di campioni erano dichiarati sani; se invece venivano trovati anticorpi, i medici preparavano due nuovi campioni mischiati, uno contenente un estratto dei campioni dei soldati da 1 a $N/2$ e l'altro contenente i campioni rimanenti. Il procedimento veniva ripetuto in modo ricorsivo fino a quando non venivano individuati i soldati infetti.

Per la versione informatica di questo algoritmo (Capetanakis, 1979) è utile immaginare le stazioni come foglie di una struttura ad albero binaria (Figura 4.9). Nel primo intervallo di contesa che segue una trasmissione senza collisioni, l'intervallo 0, tutte le stazioni possono tentare di acquisire il controllo del canale. Se una ci riesce, bene; altrimenti, in caso di collisione, durante l'intervallo 1 possono competere solo quelle stazioni che si trovano sotto il nodo 2 della struttura ad albero. Se una di queste acquisisce il controllo del can-

le, l'intervallo successivo è riservato alle stazioni che si trovano sotto il nodo 3. Se, d'altra parte, due o più stazioni sotto il nodo 2 vogliono trasmettere, allora durante l'intervallo 1 ci sarà una collisione e di conseguenza durante il nodo 2 il turno passerà al nodo 4.

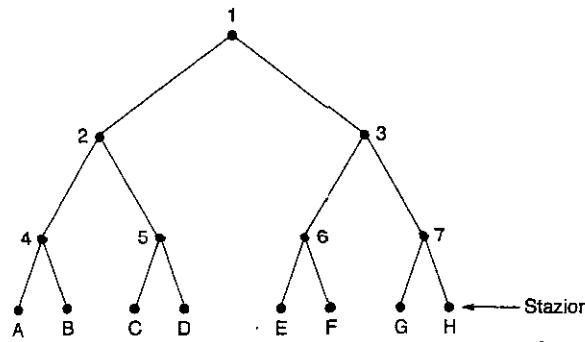


Figura 4.9. La struttura ad albero nel caso di otto stazioni.

In definitiva, in caso di collisione durante l'intervallo 0 l'algoritmo analizza l'intera struttura ad albero partendo dal basso per individuare tutte le stazioni pronte. Ogni intervallo di bit è associato a qualche particolare nodo dell'albero. In caso di collisioni, la ricerca continua in modo ricorsivo con gli elementi figli posti a sinistra e a destra del nodo. Se un intervallo di bit è libero oppure se una sola stazione trasmette durante quel periodo, la ricerca del suo nodo può interrompersi perché tutte le stazioni pronte sono state individuate (se fossero più di una ci sarebbe stata una collisione).

Quando il carico sul sistema è pesante non vale affatto la pena di dedicare l'intervallo 0 al nodo 1; questo approccio, infatti, avrebbe senso solo nell'evento improbabile dove una sola stazione ha un frame da inviare. In modo analogo si potrebbe dimostrare che anche i nodi 2 e 3 dovrebbero essere saltati per lo stesso motivo. Ma allora, in termini più generali, a quale livello della struttura deve avere inizio la ricerca? Chiaramente, più è pesante il carico, più in basso deve iniziare la ricerca. Si supponga che ogni stazione abbia una buona stima del numero di stazioni pronte, q , informazione ottenuta per esempio attraverso un meccanismo di analisi del traffico recente.

Per procedere, si supponga di numerare i livelli della struttura ad albero partendo dalla cima: con il nodo 1 nella Figura 4.9 al livello 0, il nodo 2 e il nodo 3 al livello 1 e così via. Si noti che ogni nodo al livello i ha una frazione 2^{-i} di stazioni poste sotto di esso. Se le q stazioni pronte sono distribuite uniformemente, il numero atteso di stazioni poste sotto un nodo specifico a livello i è uguale a $2^{-i} q$. Si intuisce facilmente che il livello ottimale da dove iniziare la ricerca è quello che ha un numero medio di stazioni contendenti per intervallo pari a 1, ossia il livello dove $2^{-i} q = 1$. Risolvendo l'equazione si ottiene $i = \log_2 q$. Sono stati scoperti numerosi miglioramenti all'algoritmo di base, e vengono discussi in buon dettaglio da Bertsekas e Gallager (1992). Per esempio, si consideri il caso in cui solo le stazioni G e H cercano di trasmettere. Al nodo 1 avviene una collisione, perciò l'algoritmo tenta con il 2 e scopre che è libero. Non ha senso esaminare il nodo 3 perché si può

star certi che si genererà una collisione (poiché due o più stazioni sotto 1 sono pronte e nessuna di queste si trova sotto 2, entrambe devono essere sotto 3). La scansione di 3 può essere saltata per passare a 6. Quando anche questo controllo non dà alcun risultato utile, l'algoritmo salta il nodo 7 e prova G.

4.2.5 Protocolli WDMA (Wavelength Division Multiple Access)

Un diverso approccio per assegnare il canale prevede la sua divisione in sottocanali mediante FDM, TDM o entrambi i sistemi, con allocazione dinamica dei sottocanali in base alle necessità. Schemi di questo tipo sono utilizzati spesso sulle LAN in fibra ottica per permettere a diverse conversazioni di utilizzare contemporaneamente lunghezze d'onda (frequenze) differenti. Il paragrafo esamina uno di questi protocolli (Humblet et al., 1992).

Una LAN completamente ottica si può facilmente costruire usando un accoppiatore a stella passivo (Figura 2.10). In pratica, due fibre collegate a ciascuna stazione sono fuse in un cilindro di vetro: una fibra trasmette l'output verso il cilindro e una trasmette l'input proveniente dal cilindro. L'output luminoso emesso da una stazione illumina il cilindro e può essere rilevato da tutte le altre stazioni. Le stelle passive possono gestire centinaia di stazioni.

Per consentire trasmissioni multiple contemporanee, lo spettro è diviso in canali (bande di lunghezza d'onda), come mostrato nella Figura 2.31. In questo protocollo, chiamato **WDMA** (*Wavelength Division Multiple Access*), a ogni stazione sono assegnati due canali: un canale stretto è utilizzato come canale di controllo per trasmettere segnali alla stazione, e un canale più largo consente alla stazione di trasmettere frame di dati.

Ogni canale è diviso in gruppi di intervalli temporali (slot), come mostrato nella Figura 4.10. Sia m il numero di intervalli presenti nel canale di controllo e $n+1$ il numero di intervalli nel canale dati; n servono per i dati, mentre l'ultimo è utilizzato dalla stazione per

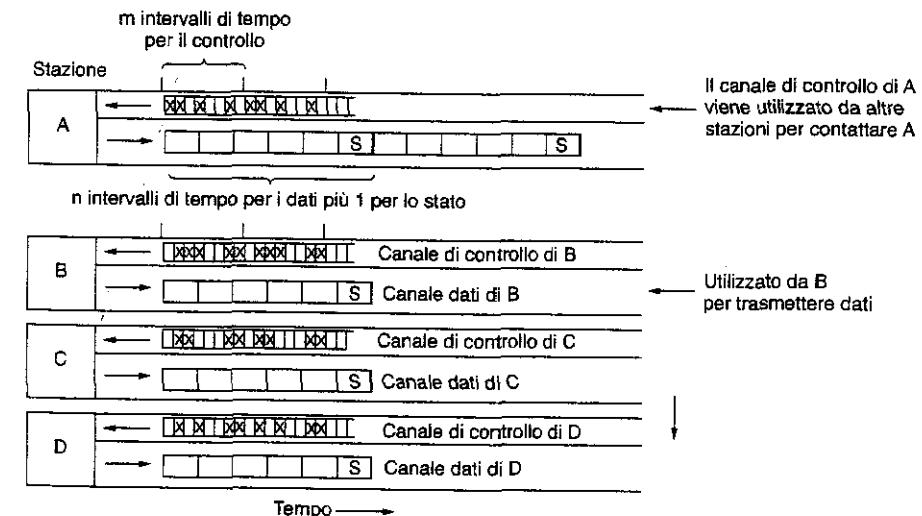


Figura 4.10. WDMA, Wavelength Division Multiple Access.

comunicare il proprio stato (e quindi essenzialmente quali intervalli su entrambi i canali sono liberi). Su entrambi i canali, la sequenza di intervalli si ripete senza fine, con l'intervallo 0 contrassegnato in modo speciale per aiutare i ritardatari a rilevarlo. Tutti i canali sono sincronizzati da un singolo segnale di clock globale.

Il protocollo supporta tre classi di traffico: (1) traffico a velocità dati costante orientato alle connessioni, come quello generato da video non compressi; (2) traffico a velocità dati variabile orientato alle connessioni, come quello generato dal trasferimento dei file; (3) traffico datagram, come quello generato dai pacchetti UDP. Per i due protocolli orientati alle connessioni, l'idea è che la stazione A, se vuole comunicare con B, deve prima inserire un frame CONNECTION REQUEST in un intervallo libero sul canale di controllo di B. Se B accetta la trasmissione, le informazioni possono essere trasmesse attraverso il canale dati di A.

Ogni stazione ha due trasmettitori e due ricevitori:

1. un ricevitore a lunghezza d'onda fissa, utilizzato per ascoltare il proprio canale di controllo
2. un trasmettitore sintonizzabile, utilizzato per trasmettere sul canale di controllo delle altre stazioni
3. un trasmettitore a lunghezza d'onda fissa, utilizzato per trasmettere i frame di dati
4. un ricevitore sintonizzabile, utilizzato per selezionare il trasmettitore dati da ascoltare.

In altre parole, ogni stazione rimane in ascolto sul proprio canale di controllo per rilevare le richieste in arrivo, ma deve sintonizzarsi sulla lunghezza d'onda del trasmettitore per ricevere i dati. La sintonizzazione della lunghezza d'onda viene eseguita mediante un interferometro Fabry-Perot o Mach-Zehnder che filtra tutte le lunghezze d'onda tranne la banda desiderata. Di seguito viene spiegato in che modo la stazione A impone un canale di comunicazione di classe 2 con la stazione B per trasferire un file. Prima di tutto A sintonizza il proprio ricevitore dati sul canale dati di B e attende l'intervallo di stato (*status slot*), che indica quali intervalli di controllo sono correntemente assegnati e quali invece sono liberi. Nella Figura 4.10, per esempio, degli otto intervalli di controllo di B sono liberi solo il numero 0, il numero 4 e il numero 5. Gli altri sono occupati (contrassegnati da una serie di X).

A sceglie uno di questi intervalli di controllo liberi, per esempio il 4, e inserisce il suo messaggio CONNECTION REQUEST. Poiché monitorizza costantemente il proprio canale di controllo, B rileva la richiesta e la accetta assegnando ad A l'intervallo 4. Questa assegnazione è annunciata nell'intervallo di stato del canale dati di B.

Quando rileva questa comunicazione, A sa che dispone di una connessione unidirezionale; ma se avesse chiesto una connessione bidirezionale, B ripete lo stesso algoritmo con A. Può capitare che mentre A tenta di acquisire l'intervallo di controllo 4 di B, la stazione C decida di fare la stessa cosa. In tal caso nessuna delle due stazioni riuscirà a ottenere il controllo ed entrambe rileveranno l'insuccesso monitorando l'intervallo di stato del canale di controllo di B. Ogni stazione allora rimane in attesa per un intervallo di tempo casuale prima di ripetere l'operazione.

A questo punto, ogni parte dispone di un sistema senza conflitto per inviare brevi messaggi

di controllo alle altre stazioni. Per eseguire il trasferimento del file, A invia a B un messaggio di controllo che dice: "Per favore, guarda il prossimo intervallo 3 di trasmissione, al suo interno troverai un frame di dati indirizzato a te". Quando riceve il messaggio di controllo, B sintonizza il suo ricevitore sul canale di output di A in modo da poter leggere il frame di dati. In base al protocollo sullo strato più alto, se B lo desidera può utilizzare lo stesso meccanismo per inviare al mittente un messaggio di riconoscimento.

Si noti che se le stazioni A e C hanno entrambe una connessione con B, e ognuna improvvisamente dice a B di controllare l'intervallo 3, può sorgere un problema: B sceglierà a caso una di queste richieste e l'altra trasmissione andrà perduta.

Per traffico a velocità costante si utilizza una variazione di questo protocollo. Quando chiede una connessione, A trasmette contemporaneamente un messaggio di questo tipo: "Posso inviarti un frame in ogni ricorrenza dell'intervallo 3?". Se B è in grado di accettare la richiesta (perché non ha già impegnato l'intervallo 3), viene stabilita una connessione a banda garantita. In caso contrario, A può tentare nuovamente con un'altra proposta in base agli intervalli di output che ha liberi.

Il traffico di classe 3 (datagram) utilizza ancora un'altra variante. Invece di scrivere un messaggio CONNECTION REQUEST nell'intervallo di controllo appena individuato, la stazione scrive un messaggio DATA FOR YOU IN SLOT 3. Se B è libera durante il successivo intervallo dati 3 la trasmissione ha successo, altrimenti il frame di dati si perde. In questo modo, non è mai richiesta una connessione.

Sono possibili diverse variazioni del protocollo, per esempio invece di assegnare a ogni stazione un proprio canale di controllo si potrebbe utilizzare un canale di controllo condiviso da tutte le stazioni. A ciascuna stazione è assegnato un blocco d'intervalli in ogni gruppo, di fatto unendo in multiplexing più canali virtuali in un solo canale fisico.

È anche possibile cavarsela dotando ogni stazione di un singolo trasmettitore sintonizzabile e di un singolo ricevitore sintonizzabile, dividendo il canale di ogni stazione in m intervalli di controllo seguiti da $n + 1$ intervalli per i dati. Lo svantaggio di questa soluzione è che per catturare un intervallo di controllo le stazioni trasmittenti devono attendere per più tempo e i frame di dati consecutivi sono più lontani perché alcune informazioni di controllo occupano il canale. Sono stati proposti e implementati molti altri protocolli WDMA, che differiscono per i dettagli. Alcuni hanno un solo canale di controllo, altri ne usano diversi; alcuni tengono conto del ritardo di propagazione, altri invece no; alcuni rendono la fase di sintonizzazione una parte esplicita del modello, altri la ignorano. I protocolli differiscono anche per la complessità di elaborazione, la capacità di trasporto e la scalabilità. Quando si utilizza un gran numero di frequenze, il sistema qualche volta è chiamato anche DWDM (*Dense Wavelength Division Multiplexing*). Per maggiori informazioni vedere (Bogineni et al., 1993; Chen, 1994; Goralski, 2001; Kartalopoulos, 1999; e Levine e Akyildiz, 1995).

4.2.6 Protocolli LAN wireless

Al crescere del numero di computer e dispositivi di comunicazione mobili aumenta anche la domanda per collegare tali apparecchi al mondo esterno. Perfino il primo telefono mobile era in grado di collegarsi con altri telefoni. I primi computer portatili non avevano questa capacità, ma in breve i modem integrati sono diventati uno standard per questi dispo-

sitivi. Per comunicare con il resto del mondo, questi computer dovevano essere collegati a una presa telefonica; la necessità di una connessione via cavo con una rete fissa comportava che tutti i computer fossero portatili ma non mobili.

Per realizzare una vera mobilità, i computer hanno bisogno di comunicare con segnali radio (o infrarossi), così gli utenti motivati possono leggere e trasmettere messaggi di posta elettronica mentre fanno escursioni a piedi o navigano in barca. Un sistema di computer portatili in grado di comunicare via radio può essere considerato una LAN wireless (come è stato spiegato nel Paragrafo 1.5.4). Queste LAN hanno proprietà piuttosto diverse da quelle delle LAN convenzionali e richiedono speciali protocolli di sottostrato MAC. Il paragrafo ne esamina solo alcuni; altre informazioni si possono trovare in (Geier, 2002; O'Hara e Petrick, 1999).

Una configurazione comune per una LAN wireless è un ufficio con stazioni base (chiamate anche access point) disposte strategicamente intorno all'edificio e collegate tra loro mediante cavi in rame o in fibra. Se la potenza di trasmissione delle stazioni base e dei computer portatili è regolata in modo da avere una portata di 3 o 4 metri, ogni stanza diventa una singola cella e l'intero edificio può essere visto come un grande sistema cellulare simile a quello telefonico studiato nel Capitolo 2. A differenza dei sistemi telefonici cellulari, ogni cella ha un unico canale che copre l'intera banda disponibile e serve tutte le stazioni nella sua cella. In genere, la sua banda è compresa tra 11 e 54 Mbps.

Per semplicità, la spiegazione seguente presuppone che tutti i trasmettitori radio abbiano una portata fissa. Quando un ricevitore si trova dentro il campo di due trasmettitori attivi, il segnale risultante in generale risulterà confuso e inutilizzabile, in altre parole questa spiegazione non considera i sistemi di tipo CDMA. È importante notare che in alcune LAN wireless non tutte le stazioni sono a portata l'una dell'altra, facendo nascere una serie di complicazioni. Inoltre, per le LAN wireless al chiuso la presenza di pareti tra le stazioni può avere un grosso impatto sulla portata effettiva di ciascuna di esse.

Un approccio banale alle LAN wireless potrebbe essere quello basato sul sistema CSMA: si ascoltano le altre trasmissioni e si trasmette solo se nessun altro sta inviando dati. Peccato che questo protocollo non sia il più adatto, perché ciò che importa è l'interferenza presso il ricevitore, non quella presso il trasmettitore. Per comprendere la natura del problema si consideri la Figura 4.11 che rappresenta quattro stazioni wireless. Per i nostri scopi non importa sapere quali sono le stazioni base e quali sono i computer portatili. La portata del segnale radio è tale che A e B sono uno dentro il campo dell'altro e possono potenzialmente interferire tra loro; inoltre C può interferire sia con B sia con D, ma non con A.

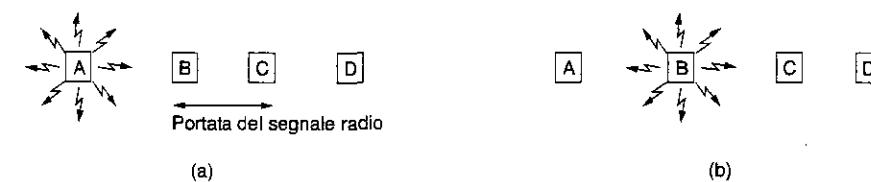


Figura 4.11. Una LAN wireless. (a) A trasmette. (b) B trasmette.

Vediamo che cosa accade se A trasmette a B, come mostrato nella Figura 4.11(a). Se C controlla la presenza di portante sul mezzo di trasmissione non potrà rilevare A, perché si trova al di fuori della sua portata; di conseguenza C pensa erroneamente di poter trasmettere a B. Se inizia a trasmettere, C interferisce con B distruggendo il frame inviato da A. Il problema di una stazione che non è in grado di rilevare i potenziali concorrenti sul mezzo di trasmissione a causa della distanza eccessiva è chiamato **problema della stazione nascosta**.

Ora si consideri la situazione inversa: B trasmette ad A. Se C controlla la presenza di portante sul mezzo di trasmissione, rileva una trasmissione in atto ed erroneamente pensa di non poter inviare dati a D; in realtà la trasmissione causerebbe una cattiva ricezione solo nella zona compresa tra B e C, che non ospita nessuno dei ricevitori designati. La situazione genera quello che è chiamato **problema della stazione esposta**.

Il problema è che, prima di avviare una trasmissione, una stazione avrebbe interesse a sapere se c'è attività intorno al ricevitore; CSMA permette solo di scoprire se c'è attività intorno alla stazione che sta analizzando la portante. Nel caso di un cavo, tutti i segnali si propagano a tutte le stazioni perciò nel sistema può avvenire una sola trasmissione alla volta. Al contrario, in un sistema basato su onde radio a portata ridotta possono aver luogo contemporaneamente trasmissioni multiple, se tutte hanno destinazioni diverse e se queste destinazioni sono oltre la portata l'una dell'altra.

Il problema può essere descritto anche in un altro modo, immaginando un ufficio in cui ogni impiegato ha un computer portatile wireless. Si supponga che Linda desideri inviare un messaggio a Milton. Il computer di Linda controlla l'ambiente locale e, non rilevando alcuna attività, inizia la trasmissione. Purtroppo nell'ufficio di destinazione potrebbero verificarsi collisioni causate da un terzo utente che, nello stesso momento, cerca di trasmettere a Milton da una postazione che non può essere rilevata dal computer di Linda perché è troppo lontana.

MACA e MACAW

Uno dei primi protocolli progettati per le LAN wireless è stato **MACA** (*Multiple Access with Collision Avoidance*) (Karn, 1990). L'idea di base è semplice: il trasmettitore incita il ricevitore a trasmettere un piccolo frame in modo che le stazioni che si trovano nelle vicinanze, rilevando questa trasmissione, evitino di inviare dati durante la trasmissione imminente del frame di dati più grande. MACA è rappresentato nella Figura 4.12.

Si supponga che A invii un frame a B. A invia prima di tutto un frame **RTS** (*Request To Send*) a B, come mostrato nella Figura 4.12(a). Questo piccolo frame (30 byte) contiene la lunghezza del frame di dati che verrà inviato successivamente. B risponde con un frame **CTS** (*Clear to Send*), come mostrato nella Figura 4.12(b). Il frame CTS contiene la lunghezza dei dati copiata dal frame RTS. Non appena riceve il frame CTS, A inizia la trasmissione.

In che modo reagiscono le altre stazioni? Tutte le stazioni che ricevono il frame RTS si trovano chiaramente vicine ad A, e devono rimanere in silenzio per un tempo che consenta al frame CTS di raggiungere A senza causare conflitti. Le stazioni che ricevono il frame CTS sono chiaramente vicine a B, e devono rimanere in silenzio durante la trasmissione dei dati la cui lunghezza può essere determinata esaminando il frame CTS.

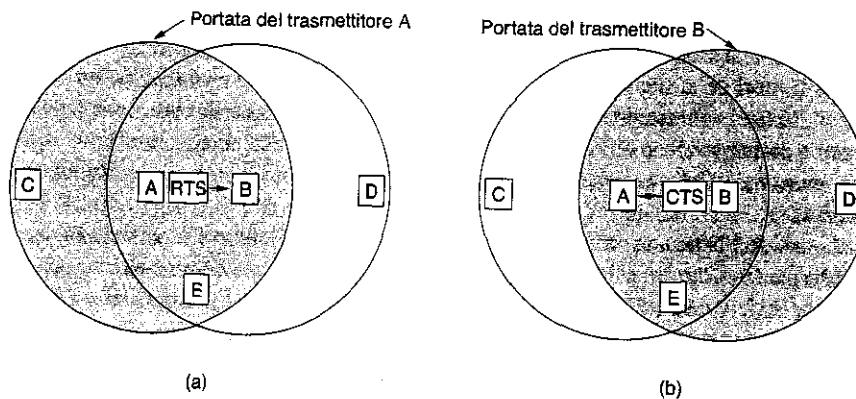


Figura 4.12. Il protocollo MACA. (a) A invia un frame RTS a B. (b) B risponde con un frame CTS.

Nella Figura 4.12, C si trova nel campo di A ma non nel campo di B perciò riceve il frame RTS inviato da A ma non il frame CTS trasmesso da B. Fintanto che non interferisce con il frame CTS, C è libero di trasmettere durante l'invio del frame di dati. Al contrario, D si trova nel raggio d'azione di B ma non in quello di A; D non riceve il frame RTS ma rileva quello CTS. Poiché rileva il frame CTS, D sa che si trova vicino a una stazione che sta per ricevere un frame, perciò rimanda la trasmissione dei propri dati. La stazione E rileva entrambi i messaggi di controllo; come D, deve rimanere in silenzio fino al termine della trasmissione del frame di dati. Nonostante queste precauzioni possono ancora verificarsi collisioni. Per esempio, B e C potrebbero inviare contemporaneamente due frame RTS ad A. Questi frame entrerebbero in collisione e andrebbero perduti.

In caso di collisione, un trasmettitore che non ha avuto successo (ossia un trasmettitore che non riceve un frame CTS nell'intervallo di tempo previsto) aspetta per un intervallo di tempo casuale prima di ripetere l'operazione. L'algoritmo utilizzato è quello di backoff esponenziale binario, che verrà descritto nel paragrafo dedicato a Ethernet.

Analizzando le simulazioni di MACA, Bharghavan et al. (1994) hanno messo a punto un protocollo MACA che ne migliora le prestazioni; la nuova versione è stata chiamata **MACAW** (*MACA per Wireless*). Gli sviluppatori hanno notato che in assenza di acknowledge nello strato data link, i frame perduti non sono trasmessi fino a che lo strato di trasporto non rileva la loro assenza, ossia molto tempo dopo. Per risolvere questo problema è stato introdotto un frame ACK dopo ogni frame di dati trasmesso con successo. Gli sviluppatori hanno anche osservato che CSMA può essere utilizzato per impedire alle stazioni di trasmettere un frame RTS quando un'altra stazione nelle vicinanze sta già inviando quel frame alla stessa destinazione; per questo è stata introdotta la capacità di rilevamento della portante. Inoltre, gli sviluppatori hanno deciso di eseguire l'algoritmo di backoff separatamente per ogni flusso dati (coppia sorgente-destinazione) invece che per ogni stazione. Questa modifica migliora l'imparzialità del protocollo. Infine, è stato aggiunto un meccanismo che consente alle stazioni di cambiare informazioni sulla congestione e un sistema che fa reagire l'algoritmo di backoff meno violentemente ai problemi temporanei, migliorando le prestazioni del sistema.

4.3 Ethernet

Conclusa la discussione generale sui protocolli utilizzati per allocare il canale, è giunto il momento di scoprire in che modo questi principi astratti si applicano ai sistemi reali e in particolare alle LAN. Come è stato spiegato nel Paragrafo 1.5.3, IEEE ha standardizzato diverse reti locali e reti metropolitane sotto il nome di IEEE 802. Alcune sono sopravvissute, molte altre no (Figura 1.38). Alcune persone che credono nella reincarnazione ritengono che Charles Darwin si sia reincarnato come membro della IEEE Standards Association per eliminare i meno adatti. I sopravvissuti più importanti sono 802.3 (Ethernet) e 802.11 (LAN wireless). Su 802.15 (Bluetooth) e 802.16 (MAN wireless) non si possono ancora dare giudizi: rimandiamo il compito alla prossima edizione di questo libro. 802.3 e 802.11 hanno strati fisici diversi e sottostrati MAC differenti, ma convergono sullo stesso sottostrato logical link control (definito in 802.2), perciò hanno la stessa interfaccia verso lo strato di rete.

Lo standard Ethernet è stato presentato nel Paragrafo 1.5.3, perciò non ripeteremo le informazioni riportate nel Capitolo 1. Il testo che segue esamina soprattutto i dettagli tecnici di Ethernet, i protocolli e i recenti sviluppi di Ethernet ad alta velocità (gigabit). Poiché Ethernet e IEEE 802.3 sono identici tranne che per due differenze secondarie che verranno esaminate tra breve, molti utilizzano come sinonimi i termini "Ethernet" e "IEEE 802.3" e quindi faremo altrettanto. Per maggiori informazioni su Ethernet vedere (Breyer e Riley, 1999 ; Seifert, 1998; Spurgeon, 2000).

4.3.1 Cablaggio Ethernet

Poiché il nome Ethernet si riferisce al cavo (definito "etero"), la descrizione inizia proprio da questo componente. Generalmente si utilizzano quattro tipi di cavi (Figura 4.13).

Nome	Cavo	Lunghezza max. del segmento	Nodi/segmento	Vantaggi
10Base5	Coassiale spesso (thick Ethernet)	500 m	100	Cavo originale, ora obsoleto
10Base2	Coassiale sottile (thin Ethernet)	185 m	30	Non occorre un hub
10Base-T	Doppino intrecciato	100 m	1.024	Il sistema più economico
10Base-F	Fibra ottica	2.000 m	1.024	Il migliore fra edifici

Figura 4.13. I tipi più comuni di cavi Ethernet.

Storicamente il cavo più vecchio è il modello **10Base5**, chiamato anche **thick Ethernet**. Questo cavo assomiglia a un tubo da giardino di colore giallo con segni disposti a intervalli di 2,5 metri che indicano la posizione delle spine (lo standard 802.3 in realtà non obbliga a utilizzare un cavo di colore giallo, suggerisce solo di farlo).

Le connessioni sono realizzate generalmente mediante **spine a vampiro**, spingendo con delicatezza uno spillo nel nucleo centrale del cavo coassiale. La notazione 10Base5 indica che opera a 10 Mbps, utilizza un sistema di segnali a banda base e può supportare seg-

menti lunghi fino a 500 metri. Il primo numero rappresenta la velocità espressa in Mbps; la parola "Base" indica la trasmissione su banda base. È esistita anche una variante a banda larga, 10Broad36, che però non è mai arrivata sul mercato. Infine, se il mezzo di trasmissione è coassiale, la lunghezza è indicata approssimandola a unità di 100 m dopo la parola "Base".

Il secondo cavo in ordine di tempo è stato il cavo **10Base2**, o **thin Ethernet**, più facile da piegare rispetto al precedente. Le connessioni sono realizzate usando connettori BNC standard che formano giunzioni a "T". I connettori BNC sono più facili da utilizzare e sono anche più affidabili. Thin Ethernet è molto più economico e semplice da installare, ma ogni segmento può essere lungo al massimo 185 metri e può supportare non più di 30 macchine.

Scoprire interruzioni dei cavi, lunghezze fuori norma, spine difettose o mal collegate può diventare un problema serio usando questo tipo di mezzi di trasmissione. Per questo motivo sono state sviluppate tecniche che aiutano a rintracciare i guasti. Sostanzialmente, un impulso di forma nota viene immesso nel cavo; se l'impulso colpisce un ostacolo o raggiunge la fine del cavo, viene generato un eco che torna indietro. Cronometrando attentamente l'intervallo trascorso tra la trasmissione dell'impulso e la ricezione dell'eco è possibile individuare l'origine dell'eco. Questa tecnica è chiamata **TDR** (*Time Domain Reflectometry*).

I problemi associati alla ricerca dei difetti nel cavo hanno condotto i sistemi verso un differente schema di cablaggio, dove ogni stazione è collegata via cavo a un concentratore centrale chiamato **hub**. Di solito per queste connessioni sono utilizzati i doppini telefonici, poiché la maggior parte degli uffici dispone già di questo tipo di linee di comunicazione interne in ampio soprannumeroso. Questo schema è chiamato **10Base-T**. Gli hub non elaborano il traffico; più avanti sarà descritta una versione più avanzata di hub (chiamata **switch**) che invece elabora i dati.

La Figura 4.14 mostra i tre schemi di cablaggio. Nel caso di 10Base5, un **transceiver** è fissato saldamente intorno al cavo per mantenere stabile il contatto con il nucleo interno. Il transceiver contiene i circuiti elettronici che gestiscono il rilevamento della portante e il rilevamento delle collisioni. Quando rileva una collisione, invia attraverso il cavo uno speciale segnale di non validità per divulgare l'evento a tutti gli altri transceiver.

Con 10Base5, un **cavo del transceiver** o **cavo di discesa** collega il trasmettitore a una scheda di interfaccia installata nel computer. Il cavo del transceiver può essere lungo 50 metri e contiene cinque doppini schermati singolarmente. Un doppino è per i dati in uscita e uno per quelli in entrata, due sono utilizzati per i segnali di controllo in ingresso e in uscita, l'ultimo (non sempre utilizzato) permette al computer di alimentare i circuiti del transceiver. Alcuni transceiver possono essere collegati a più computer (fino a 8), soluzione che consente di ridurre il numero di transceiver installati nella rete.

Il cavo del transceiver termina su una scheda di interfaccia posta dentro il computer. La scheda contiene un chip controller che trasmette e riceve i frame, e si occupa di assemblare i dati nel giusto formato di frame, elaborare i codici di controllo dei frame in uscita e verificare i codici di controllo dei frame in ingresso.

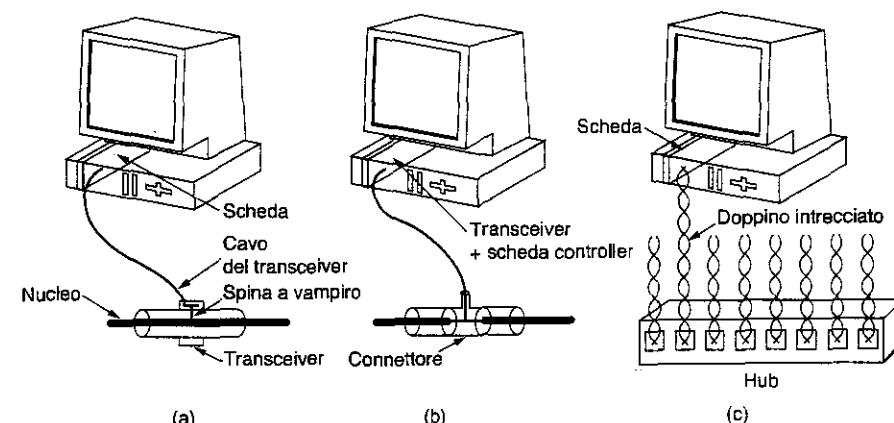


Figura 4.14. Tre tipi di cavi Ethernet. (a) 10Base5. (b) 10Base2. (c) 10Base-T.

Alcuni chip controller gestiscono anche un pool di buffer dedicati ai frame in arrivo, una coda di buffer contenenti dati da trasmettere, i circuiti DMA per il trasferimento dei dati con il computer host e altri aspetti della gestione della rete.

Con 10Base2, la connessione al cavo passa attraverso un connettore BNC a forma di "T". I circuiti elettronici del transceiver si trovano nella scheda di controllo e ogni stazione ha sempre il proprio transceiver.

Con 10Base-T non c'è alcun cavo condiviso: c'è solo l'hub (una scatola piena di circuiti elettronici) al quale ogni stazione si collega attraverso un cavo dedicato (non condiviso). In questa configurazione è semplice aggiungere o rimuovere una stazione, inoltre è facile individuare le interruzioni della linea. Lo svantaggio di 10Base-T è rappresentato dalla lunghezza massima dei cavi che partono dall'hub: 100 metri, che nel caso di cavi di alta qualità (categoria 5) arrivano forse a 200 metri. Ciò nonostante, 10Base-T è diventato rapidamente uno standard dominante grazie all'uso di cablaggi preesistenti e alla facilità di gestione. Una versione più veloce di 10Base-T, chiamata 100Base-T, verrà esaminata nel corso del capitolo.

Il quarto tipo di cavi per Ethernet si chiama **10Base-F** e utilizza le fibre ottiche. È un'alternativa costosa a causa del prezzo dei connettori e dei terminatori, ma offre un'eccellente immunità alle interferenze e consente di collegare edifici distanti o hub molto lontani. I cavi di questo tipo possono essere lunghi anche un chilometro. 10Base-F offre una buona sicurezza poiché intercettare i dati trasmessi attraverso la fibra ottica è più difficile.

La Figura 4.15 mostra i diversi modi di cablare un edificio. La Figura 4.15(a) mostra un singolo cavo che si snoda da una stanza all'altra, ogni stazione si collega al cavo nel punto più vicino; la Figura 4.15(b) mostra una dorsale verticale che si estende dal piano terra al tetto dell'edificio, i cavi orizzontali su ogni piano si collegano alla dorsale attraverso speciali amplificatori (chiamati ripetitori). In alcuni edifici, i cavi orizzontali sono thin Ethernet e la dorsale è thick Ethernet. La topologia più comune è quella ad albero mostrata nella Figura 4.15(c), perché in una rete che offre due percorsi per raggiungere la stazione di destinazione è facile che i due segnali creino interferenza.

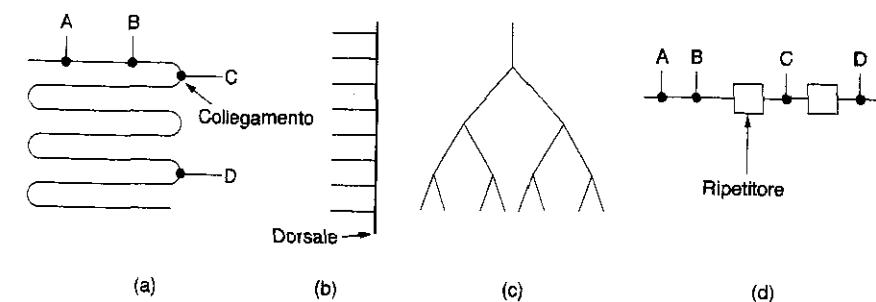


Figura 4.15. Topologie di cavi. (a) Lineare. (b) Dorsale. (c) Ad albero. (d) Segmentata.

Ogni versione di Ethernet ha una lunghezza massima per segmento. Per creare reti più grandi si possono collegare tra loro più cavi facendo uso di **ripetitori**, come mostrato nella Figura 4.15 (d). Un ripetitore è un dispositivo che opera sullo strato fisico: riceve, amplifica (rigenera) e rtrasmette i segnali in entrambe le direzioni. Dal punto di vista del software, una struttura composta da segmenti collegati attraverso ripetitori è identica a una struttura composta da un singolo cavo; l'unica differenza riguarda il ritardo introdotto dai ripetitori. Un sistema può contenere più segmenti e più ripetitori, ma la distanza tra qualsiasi coppia di transceiver non può superare i 2,5 Km e nessun percorso tra due transceiver può attraversare più di quattro ripetitori.

4.3.2 Codifica Manchester

Nessuna delle versioni di Ethernet utilizza la codifica binaria semplice con 0 Volt associati al bit 0 e 5 Volt associati al bit 1, perché questo sistema crea ambiguità. Se una stazione invia la sequenza 0001000, altre stazioni potrebbero interpretare erroneamente questa stringa di bit come 10000000 o 01000000 perché non è possibile distinguere tra il segnale nullo (0 Volt) della linea libera dal segnale nullo (0 Volt) associato al bit 0. Questo problema può essere risolto associando +1 Volt al bit 1 e -1 Volt al bit 0, ma rimane ancora il problema dei ricevitori che possono campionare il segnale a una frequenza estremamente diversa da quella che il trasmettitore ha utilizzato per generarlo. Velocità di clock diverse possono mandare il ricevitore e il trasmettitore fuori sincronia facendo perdere l'informazione del punto in cui iniziano i bit, specialmente dopo una lunga serie di 0 oppure di 1 consecutivi. Ciò che serve è un sistema che consenta ai ricevitori di determinare senza ambiguità il punto iniziale, il punto finale o il punto centrale di ogni bit senza riferimento a impulsi esterni. Per risolvere il problema sono state inventate due tecniche chiamate **codifica Manchester** e **codifica Manchester differenziale**. Con la codifica Manchester, ogni periodo di bit è diviso in due intervalli uguali. L'1 binario è inviato scegliendo un livello di tensione alto durante il primo intervallo e un livello basso durante il secondo. Lo schema contrario è utilizzato per trasmettere uno 0 binario: prima un livello di tensione basso e poi uno alto [NDR - lo standard IEEE 802.3 dice esattamente l'opposto: un 1 binario è codificato con un livello di tensione basso seguito da un livello alto, un 0 binario è codificato da un livello di tensione alto seguito da un livello basso].

(cfr <http://standards.ieee.org/getieee802/download/802.3-2002.pdf> pagina 118).

Con questo schema si ha la certezza che ogni periodo di bit ha una transizione nel punto centrale, caratteristica che aiuta il ricevitore a sincronizzarsi con il trasmettitore. Lo svantaggio della codifica Manchester è che occupa il doppio della banda della codifica binaria elementare, perché gli impulsi sono larghi la metà.

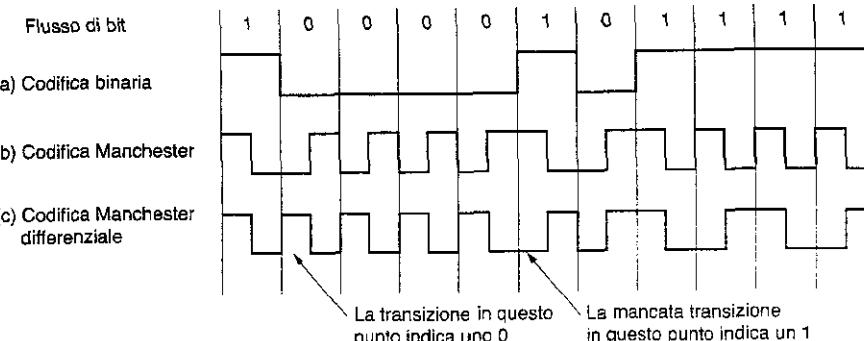


Figura 4.16. (a) Codifica binaria. (b) Codifica Manchester. (c) Codifica Manchester differenziale.

Per esempio, per inviare dati a 10 Mbps il segnale deve cambiare 20 milioni di volte al secondo. La Figura 4.16(b) mostra la codifica Manchester.

La codifica Manchester differenziale, mostrata nella Figura 4.16(c), è una variante della codifica Manchester elementare. Il bit 1 è indicato dall'assenza di una transizione all'inizio dell'intervallo; il bit 0 è indicato dalla presenza di una transizione all'inizio dell'intervallo. In entrambi i casi, c'è una transizione nel punto centrale. Lo schema differenziale richiede dispositivi più complessi ma offre una maggiore immunità ai rumori. Tutti i sistemi Ethernet adottano la codifica Manchester perché è più semplice. Il segnale alto ha una tensione di +0,85 Volt e il segnale basso ha una tensione di -0,85 Volt, con il valore DC pari a 0 Volt. Ethernet non utilizza la codifica Manchester differenziale, altre LAN invece sì (per esempio la 802.5 token ring).

4.3.3 Il protocollo del sottostrato MAC Ethernet

La Figura 4.17(a) mostra la struttura del frame DIX (DEC, Intel, Xerox) originale. Ogni frame comincia con un campo *preamble* (preambolo) di 8 byte, ognuno contenente lo schema di bit 10101010. La codifica Manchester di questo schema produce un'onda quadra di 10 MHz per 6,4 µsec che permette al clock del ricevitore di sincronizzarsi con quello del trasmettitore. Devono rimanere sincronizzati per il resto del frame, usando la codifica Manchester per tenere sotto controllo i limiti dei bit. Il frame contiene due indirizzi: uno rappresenta la destinazione e l'altro la sorgente. Lo standard consente indirizzi da 2 a 6 byte, ma i parametri definiti per la banda base 10 Mbps utilizzano solo indirizzi da 6 byte. Il bit di ordine più elevato nel campo *destination address* (indirizzo di destinazione) è 0 per gli indirizzi ordinari oppure 1 per gli indirizzi di gruppo.

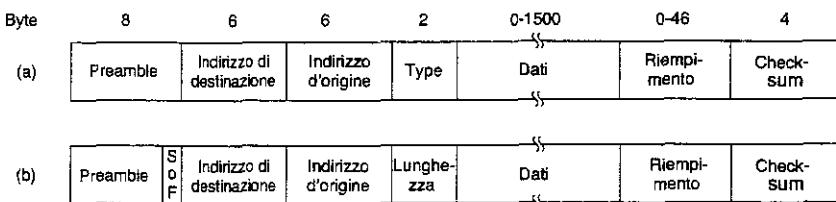


Figura 4.17. Formati di frame. (a) DIX Ethernet. (b) IEEE 802.3.

Gli indirizzi di gruppo permettono a molte stazioni di rimanere in ascolto di un singolo indirizzo. Quando un frame viene inviato a un indirizzo di gruppo, tutte le stazioni di quel gruppo lo ricevono. La trasmissione diretta a un gruppo di stazioni è definita **multicast**. L'indirizzo composto da tutti bit 1 è riservato per la trasmissione **broadcast**. Un frame che contiene tutti 1 nel campo destination address è accettato da tutte le stazioni della rete. La differenza tra multicast e broadcast è abbastanza importante, perciò è utile ricordarla: un frame multicast è inviato a un gruppo selezionato di stazioni su Ethernet; un frame broadcast è inviato a tutte le stazioni su Ethernet. La trasmissione multicast è più selettiva ma richiede una gestione del gruppo. La trasmissione broadcast è più rossa ma non richiede alcuna gestione.

Un'altra caratteristica interessante dell'indirizzamento è legata al bit 46 (adiacente al bit di ordine più elevato), usato per distinguere gli indirizzi locali da quelli globali. Gli indirizzi locali sono assegnati da ogni amministratore di rete e non hanno alcun significato all'esterno della rete locale; gli indirizzi globali, invece, sono assegnati centralmente da IEEE per garantire che nessuna stazione nel mondo utilizzi lo stesso indirizzo globale di un'altra stazione. Con $48 - 2 = 46$ bit disponibili è possibile assegnare circa 7×10^{13} indirizzi globali. In pratica ogni stazione dovrebbe poter comunicare in modo univoco con qualunque altra stazione semplicemente adoperando il corretto numero di 48 bit. Spetta allo strato di rete capire come individuare la destinazione.

Il campo successivo, *type* (tipo), indica al ricevitore che cosa deve fare del frame. Sullo stesso computer si possono usare simultaneamente più protocolli dello strato network, perciò quando arriva un frame Ethernet il kernel deve sapere quale protocollo dovrà gestire il pacchetto. Il campo type indica il processo a cui passare il frame.

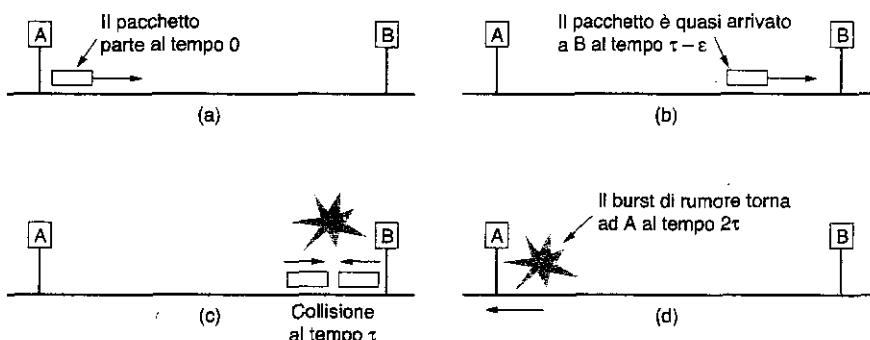
Dopo questo campo arriva il campo *data* (dati), lungo fino a 1.500 byte. Questo limite, scelto abbastanza arbitrariamente al tempo della creazione dello standard DIX, si basava più che altro sul fatto che un transceiver aveva bisogno di una RAM che gli consentisse di conservare l'intero frame, e la RAM nel 1978 era costosa. Un limite più alto avrebbe significato più RAM e quindi apparecchi più costosi.

Il frame non ha solo una lunghezza massima ma anche una lunghezza minima; anche se qualche volta può tornare utile, un campo data lungo 0 byte causa un problema. Quando rileva una collisione il ricestransmettitore tronca il frame corrente, e ciò significa che sul cavo compaiono continuamente bit sparsi e pezzi di frame. Per aiutare a distinguere i frame validi dalla spazzatura, Ethernet richiede che i frame validi siano lunghi almeno 64

byte, dal destination address al checksum inclusi. Se la parte occupata dai dati è lunga meno di 46 byte, il campo *pad* (riempimento) viene utilizzato per riempire il frame. Imporre una lunghezza minima serve anche per un altro (e più importante) motivo: per impedire a una stazione di completare la trasmissione di un frame breve prima che il primo bit abbia raggiunto la fine del cavo, dove potrebbe collidere con un altro frame. Questo problema è rappresentato graficamente nella Figura 4.18. Al tempo 0, la stazione A che si trova a un'estremità della rete invia un frame. Sia τ il tempo di propagazione impiegato dal frame per raggiungere l'altra estremità (per esempio al tempo $\tau - \epsilon$), la stazione più distante B inizia a trasmettere. Quando si accorge di ricevere più potenza di quella emessa, B capisce che è avvenuta una collisione perciò interrompe la sua trasmissione e genera un burst di rumore a 32 bit (cfr. <http://standards.ieee.org/getieee802/download/802.3-2002.pdf> pagina 81) per avvisare tutte le altre stazioni. In altre parole, ostruisce il mezzo di trasmissione per essere sicura che il trasmettitore rilevi la collisione.

All'istante 2τ anche il trasmettitore vede il burst di rumore e interrompe la sua trasmissione; aspetta quindi per un periodo di tempo casuale prima di ritentare.

Se una stazione tenta di trasmettere un frame molto corto, potrebbe succedere che avven-

Figura 4.18. Il rilevamento della collisione può richiedere 2τ .

ga una collisione ma la trasmissione si conclude prima che il burst di rumore torni indietro (ossia prima dell'istante 2τ). In questo caso il trasmettitore concluderà erroneamente che il frame è stato inviato con successo. Per evitare che si presenti una situazione di questo tipo, tutti i frame devono impiegare più di 2τ per arrivare a destinazione; in tal modo la trasmissione è ancora in esecuzione quando il burst di rumore torna al trasmettitore. Per una LAN a 10 Mbps che può essere lunga al massimo 2.500 metri e può contenere non più di 4 ripetitori (secondo le specifiche 802.3), il viaggio di andata e ritorno (considerando anche il tempo di propagazione attraverso i quattro ripetitori) dura circa 50 μsec nel peggiore dei casi. A 10 Mbps, un bit impiega 100 nsec, perciò 500 bit è la dimensione che garantisce che tutto funzioni. Per aggiungere un margine di sicurezza, questo numero è stato arrotondato a 512 bit, ossia 64 byte. I frame con meno di 64 byte sono riempiti automaticamente dal campo pad fino a raggiungere la dimensione minima richiesta.

Al crescere della velocità della rete, la lunghezza minima del frame deve aumentare o, proporzionalmente, deve diminuire la lunghezza massima del cavo. Per una LAN lunga 2.500 che opera a 1 Gbps, la dimensione minima del frame dovrebbe essere 6.400 byte; in alternativa la dimensione minima potrebbe essere 640 byte se la distanza massima tra due stazioni fosse di 250 metri. Spostandosi verso le reti multigigabit, queste restrizioni diventano sempre più dolorose.

L'ultimo campo Ethernet è il *checksum* (codice di controllo), un codice hash dei dati di 32 bit. Se alcuni bit di dati sono interpretati in modo errato (a causa del rumore lungo il cavo), il checksum risulterà certamente sbagliato e l'errore verrà rilevato. Il checksum è un CRC (*Cyclic Redundancy Check*) del tipo descritto nel Capitolo 3; esegue solo il rilevamento degli errori e non si occupa della correzione.

Quando IEEE creò lo standard Ethernet, il comitato apportò due modifiche al formato DIX, come mostrato nella Figura 4.17(b): il campo preamble venne ridotto a 7 byte e l'ultimo byte fu utilizzato come delimitatore di inizio frame (SoF, *Start of Frame*) per mantenere la compatibilità con gli standard 802.4 e 802.5; il campo type divenne il campo *length*.

La nuova struttura non permetteva al ricevitore di capire che cosa si sarebbe dovuto fare con il frame in arrivo; il problema è stato risolto aggiungendo alla parte dei dati una piccola intestazione che fornisce questa informazione. Il formato della parte dati verrà descritto nei paragrafi dedicati al controllo del collegamento logico, più avanti in questo capitolo.

Sfortunatamente, quando venne pubblicato lo standard 802.3, c'era così tanto hardware e software per Ethernet DIX già in uso che pochi produttori e utenti furono entusiasti della conversione del campo type in campo length. Nel 1997 IEEE gettò la spugna e disse che entrambi i formati andavano bene. Per fortuna tutti i campi type in uso prima del 1997 erano più grandi di 1.500, di conseguenza, qualunque numero minore o uguale a 1.500 può essere interpretato come length e qualunque numero maggiore di 1.500 può essere interpretato come type. Ora IEEE può affermare che tutti usano il suo standard, mentre costruttori e sviluppatori possono continuare a fare ciò che hanno sempre fatto senza sentirsi in colpa.

4.3.4 L'algoritmo di backoff esponenziale binario

Vediamo ora in che modo viene gestita l'attesa casuale dopo una collisione. Il modello è quello rappresentato nella Figura 4.5. Dopo una collisione, il tempo è diviso in intervalli discreti la cui lunghezza è uguale al tempo di propagazione di andata e ritorno del caso peggiore sul mezzo di trasmissione (ossia 2τ). Per contenere il percorso più lungo supportato da Ethernet, l'intervallo temporale è stato impostato a intervalli di 512 bit, o di 51,2 μ sec come è stato spiegato precedentemente.

Dopo la prima collisione, ogni stazione aspetta 0 o 1 intervalli temporali prima di ripetere. Se due stazioni collidono e ognuna sceglie lo stesso numero casuale, la collisione si ripeterà. Dopo la seconda collisione, ogni stazione sceglie 0, 1, 2 o 3 a caso e rimane in attesa per quel numero di intervalli temporali. Se avviene una terza collisione (la probabilità che questo accada è pari a 0,25) la volta successiva il numero d'intervalli di attesa è scelto a caso tra 0 e $2^3 - 1$.

In generale, dopo i collisioni, viene scelto un numero casuale compreso tra 0 e $2^i - 1$ e si salta quel numero di intervalli; dopo dieci collisioni il tetto massimo dell'intervallo di scelta rimane bloccato a 1.023 intervalli. Dopo 16 collisioni il chip di controllo getta la spugna e comunica al computer un errore. Ulteriori operazioni di ripristino sono demandate agli strati più alti.

Questo algoritmo, chiamato **backoff esponenziale binario**, è stato scelto perché si adatta dinamicamente al numero di stazioni che tentano di trasmettere. Se l'intervallo casuale per tutte le collisioni fosse 1.023, la possibilità che due stazioni collidano una seconda volta sarebbe trascurabile, ma l'attesa media dopo una collisione durerebbe centinaia di intervalli temporali e il ritardo introdotto sarebbe pesante. D'altro canto, si supponga che ogni stazione aspetti sempre per 0 o 1 intervalli; in questo caso, se 100 stazioni tentassero di trasmettere contemporaneamente, le collisioni si ripeterebbero fino a quando 99 di esse non sceglieressero l'intervallo 1 e la rimanente non scegliesse lo 0. Questo potrebbe richiedere anni.

Poiché l'intervallo di scelta casuale cresce esponenzialmente con il numero di collisioni avvenute, l'algoritmo assicura un basso ritardo quando poche stazioni collidono e garantisce la risoluzione della collisione in un intervallo di tempo ragionevole quando la collisione coinvolge molte stazioni. Troncando il backoff a 1.023 si impedisce al confine di crescere troppo.

Come è stato spiegato precedentemente, CSMA/CD non fornisce alcun meccanismo di riconoscimento. Poiché la semplice assenza di collisioni non garantisce l'arrivo corretto dei bit (i segnali potrebbero essere danneggiati dai picchi di rumore presenti nel cavo), per rendere affidabile la comunicazione è necessario che la destinazione verifichi il codice di controllo e, se il valore è corretto, invii un frame di acknowledge alla sorgente. Normalmente il frame di acknowledge dovrebbe essere un frame come tutti gli altri, e dovrebbe lottare per conquistare il canale proprio come accade con i frame di dati, ma per rendere un più veloce il meccanismo di conferma sarebbe sufficiente apportare una piccola modifica all'algoritmo di contesa (Tokoro and Tamaru, 1977). Non bisogna far altro che riservare alla stazione di destinazione il primo intervallo di contesa successivo alla trasmissione che ha avuto successo. Sfortunatamente, lo standard non lo prevede.

4.3.5 Prestazioni di Ethernet

Questo paragrafo esamina brevemente le prestazioni di Ethernet in condizioni di carico pesante e costante, con k stazioni sempre pronte a trasmettere. Un'analisi rigorosa dell'algoritmo di backoff esponenziale binario risulterebbe molto complessa; è più utile seguire il ragionamento di Metcalfe e Boggs (1976) e supporre una probabilità di ritrasmissione costante in ogni intervallo. Se ogni stazione trasmette durante un intervallo di contesa con probabilità p , allora la probabilità A che qualche stazione acquisisca il controllo del canale in quell'intervallo è

$$A = kp(1-p)^{k-1} \quad (4.5)$$

Il valore di A è massimo quando $p = 1/k$, con $A \rightarrow 1/e$ per $k \rightarrow \infty$. La probabilità che l'in-

intervallo di contesa abbia esattamente j intervalli al suo interno è $A(1 - A)^{j-1}$, perciò il numero medio di intervalli per ogni contesa è dato da

$$\sum_{j=0}^{\infty} jA(1 - A)^{j-1} = \frac{1}{A}$$

Poiché ogni intervallo dura 2τ , l'intervallo di contesa medio w dura $2\tau/A$. Supponendo p ottimale, il numero medio di intervalli di contesa non è mai maggiore di e , perciò w è almeno 2τ e $\approx 5,4\tau$. Se il frame medio impiega P sec per arrivare, quando molte stazioni hanno frame da inviare

$$\text{efficienza del canale} = \frac{P}{P + 2\tau / A} \quad (4.6)$$

L'equazione mostra il rapporto che lega la distanza massima tra coppie di stazioni lungo il cavo alle prestazioni della rete, e fa capire come mai sono state inventate topologie diverse da quella mostrata nella Figura 4.15(a): più il cavo è lungo, più tempo dura l'intervallo di contesa. Questa osservazione spiega come mai lo standard Ethernet specifica una lunghezza massima per i cavi.

È istruttivo formulare l'equazione (4.6) in termini di lunghezza di frame, F , banda di rete, B , lunghezza del cavo, L , e velocità di propagazione del segnale, c , per il caso ottimale di intervalli di contesa per frame. Con $P = F/B$, l'equazione (4.6) diventa

$$\text{efficienza del canale} = \frac{1}{1 + 2BLc/fF} \quad (4.7)$$

Quando il secondo termine nel denominatore è grande, l'efficienza della rete è bassa. Per la precisione, aumentando la banda della rete o la distanza (ossia il prodotto BL) si riduce l'efficienza per una data dimensione di frame. Sfortunatamente, gran parte della ricerca sull'hardware di rete punta proprio ad aumentare questo prodotto. Gli utenti vogliono ampiezze di banda più grandi su lunghe distanze (MAN su fibra ottica, per esempio), questo suggerisce che lo standard Ethernet implementato in questa maniera potrebbe non essere il miglior sistema per queste applicazioni. Più avanti nel capitolo, nella parte dedicata a Ethernet commutata, verranno esaminati altri metodi d'implementazione.

La Figura 4.19 mostra l'efficienza del canale calcolata mediante l'equazione (4.7) in base al numero di stazioni pronte, per $2\tau = 51,2 \mu\text{sec}$ a un data rate di 10 Mbps. Con intervalli temporali ampi 64 byte, non deve sorprendere la scoperta che i frame grandi 64 byte non sono efficienti. D'altro canto, con frame da 1.024 byte e un valore asintotico di e intervalli di 64 byte per intervallo di contesa, il periodo di contesa è lungo 174 byte e l'efficienza è uguale a 0,85. Per determinare il numero medio di stazioni pronte a trasmettere in condizioni di carico pesante si può utilizzare la seguente osservazione: ogni frame occupa il canale per un periodo di contesa e un periodo di trasmissione del frame, per un totale di $P + w$ sec. Il numero di frame al secondo è perciò $1/(P + w)$. Se ogni stazione genera frame a una velo-

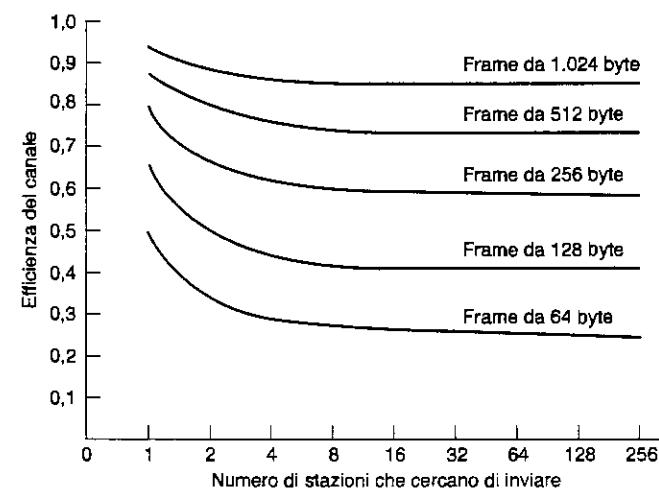


Figura 4.19. Efficienza di Ethernet a 10 Mbps con intervalli temporali (time slot) di 512 bit.

cità media di λ frame al secondo, allora quando il sistema è nello stato k la velocità totale di input di tutte le stazioni non bloccate combinate insieme è di $k\lambda$ frame/sec.

Poiché in condizioni di equilibrio le velocità di input e output devono essere identiche, le due espressioni possono essere espresse in forma di equazione e risolte per k (si noti che w è una funzione di k). Un'analisi più sofisticata è studiata in (Bertsekas e Gallager, 1992). Probabilmente vale la pena di ricordare che è stato condotto un grande lavoro di analisi teorica delle prestazioni di Ethernet (e delle altre reti), dove si fa praticamente sempre l'ipotesi che il traffico sia di tipo Poisson. Quando i ricercatori hanno iniziato ad analizzare i dati reali, si sono accorti che il traffico di rete raramente si comporta secondo la legge di Poisson, ma piuttosto è autosimile (Paxson e Floyd, 1994; e Willinger et al., 1995). Questo significa che la media calcolata su lunghi periodi di tempo non appiana il traffico. Il numero medio di frame in ogni minuto di un'ora ha una varianza pari al numero medio di frame in ogni secondo di un minuto. La conseguenza di questa scoperta è che la maggior parte dei modelli di traffico di rete non si applica al mondo reale e va presa "cum grano salis".

4.3.6 Ethernet commutata

Al crescere del numero di stazioni aggiunte a una Ethernet il traffico aumenta; alla fine la LAN si satura. Per uscirne si può passare a una velocità maggiore, per esempio da 10 Mbps a 100 Mbps, ma con lo sviluppo delle applicazioni multimediali, anche una Ethernet a 100 Mbps oppure a 1 Gbps può saturarsi con il tempo.

Per fortuna si possono gestire gli incrementi di carico anche in un altro modo, attraverso le Ethernet commutate (Figura 4.20). Il cuore di questo sistema è lo **switch** (commutatore) che contiene una scheda backplane ad alta velocità su cui innestare da 4 a 32 schede di

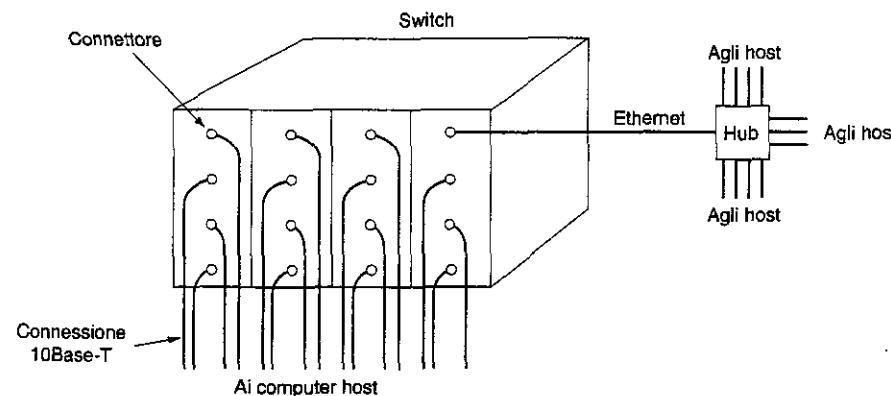


Figura 4.20. Un semplice esempio di Ethernet commutata.

linea, ognuna contenente da uno a otto connettori. Il più delle volte ogni connettore gestisce un doppino 10Base-T diretto verso un singolo computer host.

Quando una stazione vuole trasmettere un frame Ethernet, invia allo switch un frame standard. La scheda dello switch che riceve il frame può controllare se il pacchetto è destinato a una delle altre stazioni collegate alla stessa scheda. In caso positivo, il frame è copiato direttamente sul corrispondente connettore; in caso negativo è trasmesso verso la destinazione attraverso la scheda backplane ad alta velocità, che di solito funziona a molti Gbps utilizzando un protocollo proprietario.

Che cosa accade se due macchine collegate alla stessa scheda di linea trasmettono frame nello stesso momento? Tutto dipende da come è stata costruita la scheda. È possibile che tutte le porte della scheda siano collegate insieme per formare una LAN locale su scheda; le collisioni su questa LAN integrata verranno rilevate e gestite come le collisioni che avvengono su una rete CSMA/CD, con ritrasmissioni controllate attraverso l'algoritmo di backoff esponenziale binario. Con questo tipo di scheda è possibile avere una sola trasmissione per scheda in ogni istante, ma tutte le schede possono trasmettere in parallelo. Secondo questo schema, ogni scheda costituisce il proprio **dominio di collisione** indipendente dagli altri. Con una sola stazione per dominio di collisione, le collisioni sono impossibili e le prestazioni migliorano.

Con l'altro tipo di scheda, ogni porta di input dispone di un buffer di memoria, perciò i frame sono memorizzati al loro arrivo nella RAM integrata sulla scheda. Questa soluzione permette a tutte le porte di input di ricevere (e trasmettere) frame contemporaneamente per garantire operazioni parallele full duplex, risultato che non si può ottenere con CSMA/CD su un singolo canale. Dopo aver ricevuto tutto il frame, la scheda può quindi controllare se il pacchetto è destinato a un'altra porta sulla stessa scheda o a una porta distante: nel primo caso il frame può essere trasmesso direttamente alla destinazione; nel secondo caso il frame deve essere trasmesso alla scheda appropriata attraverso il backpla-

I sottostrati MAC (Medium Access Control)

ne. Con questo schema ogni porta è un dominio di collisione separato, perciò non possono esserci collisioni. La capacità di trasporto totale di solito aumenta di un ordine di grandezza rispetto a 10Base5, che ha un singolo dominio di collisione per l'intero sistema. Poiché il commutatore si aspetta di ricevere frame Ethernet standard su ogni porta, è possibile utilizzare alcune porte come concentratori. Nella Figura 4.20, la porta posta nell'angolo superiore destro non è collegata a una singola stazione ma a un hub con 12 porte. Quando l'hub riceve i frame, questi si contendono il mezzo di trasmissione nel solito modo, incluse le collisioni e il backoff binario. I frame che vincono arrivano al commutatore e sono gestiti come qualunque altro frame in ingresso: sono commutati sulla linea di output corretta attraverso il backplane ad alta velocità. Gli hub sono più economici, ma stanno rapidamente diventando obsoleti per effetto della caduta dei costi degli switch, anche se ancora non si possono considerare estinti.

4.3.7 Fast Ethernet

Al principio la connessione a 10 Mbps sembrava il paradiso, proprio come i modem a 1.200 bps erano apparsi meravigliosi agli utenti dei primi modem acustici a 300 bps. In poco tempo, però, la novità non fece più effetto. Come una sorta di corollario alla Legge di Parkinson (“Il lavoro si espande fino a riempire tutto il tempo disponibile per il suo completamento”), sembrava che anche i dati si espandessero fino a riempire tutta la banda a disposizione della trasmissione. Per aumentare le velocità, differenti gruppi industriali proposero due nuove LAN ottiche basate su topologie ad anello. Una fu chiamata **FDDI** (*Fiber Distributed Data Interface*) e l'altra fu chiamata **Fibre Channel** (fu chiamata “fibre channel” e non “fiber channel” perché il redattore del documento era inglese).

Per arrivare subito al dunque, entrambe furono utilizzate come reti dorsali ma nessuna delle due raggiunge mai il computer desktop. In entrambi i casi la gestione della stazione era troppo complessa, di conseguenza erano complessi i chip e alti i prezzi degli apparecchi. In ogni caso, l'insuccesso delle LAN ottiche aprì un varco a varietà di Ethernet che supportavano velocità superiori ai 10 Mbps. Molte installazioni che avevano bisogno di una banda maggiore avevano numerose LAN a 10 Mbps collegate mediante un dedalo di ripetitori, bridge, router e gateway; gli amministratori di rete qualche volta avevano la sensazione che queste configurazioni fossero tenute insieme con la gomma da masticare e il nastro isolante.

È in questo ambiente che nel 1992 IEEE riunì il comitato 802.3 dando il mandato di creare una LAN più veloce. Una proposta fu di mantenere lo standard 802.3 esattamente com'era, aumentando solo la velocità; un'altra proposta fu di rifare tutto daccapo, per definire nuove funzionalità come il traffico in tempo reale e la voce digitale, mantenendo però il vecchio nome (per motivi di mercato). Dopo la solita baruffa, il comitato decise di mantenere 802.3 così com'era, migliorando però la sua velocità. Quelli che si opposero alla decisione crearono comunque un loro standard LAN indipendente (chiamato 802.12), che fallì miseramente.

Il comitato 802.3 decise di procedere verso la definizione di una Ethernet semplicemente ottimizzata per tre motivi fondamentali:

1. mantenere la compatibilità con le LAN Ethernet esistenti
2. il timore che un nuovo protocollo avrebbe potuto creare problemi imprevisti
3. il desiderio di raggiungere l'obiettivo prima che la tecnologia cambiasse.

Il lavoro venne svolto velocemente e il risultato, **802.3u**, fu approvato ufficialmente da IEEE nel giugno del 1995. Tecnicamente, 802.3u non è un nuovo standard ma un'aggiunta allo standard 802.3 esistente (per enfatizzare la compatibilità verso il basso). Poiché praticamente tutti lo chiamano **fast Ethernet** e non 802.3u, anche in questo capitolo si usa il nome più comune.

L'idea alla base di fast Ethernet è semplice: mantenere tutti i vecchi formati di frame, le interfacce e le regole procedurali, riducendo semplicemente il tempo bit da 100 nsec a 10 nsec. Tecnicamente sarebbe stato possibile copiare 10Base-5 o 10Base-2 e rilevare in tempo le collisioni semplicemente riducendo la lunghezza massima del cavo di un fattore 10, ma i vantaggi dei cavi 10Base-T erano talmente schiaccianti che fast Ethernet si basò interamente sul questa architettura. Perciò tutti i sistemi fast Ethernet utilizzano hub e commutatori; cavi multidrop con connettori a vampiro e connettori BNC non sono consentiti.

Era necessario fare anche altre scelte: la più importante riguardava i tipi di cavi supportati. Un contendente era il doppino di categoria 3; a suo favore c'era il fatto che praticamente ogni ufficio del mondo occidentale disponeva di almeno quattro doppini di categoria 3 (o migliori) collegati a una centralina telefonica distante non più di 100 metri. Alcuni uffici avevano due di questi cavi per posto di lavoro. Utilizzando i doppiini di categoria 3 sarebbe stato possibile collegare i computer desktop con fast Ethernet senza dover ricablarne l'edificio, un enorme vantaggio per molte aziende.

Lo svantaggio principale del doppino di categoria 3 è la sua incapacità di trasportare segnali di 200 megabaud (100 Mbps con codifica Manchester) per 100 metri, la distanza massima computer-hub specificata per 10Base-T (Figura 4.13). Al contrario, i doppiini di categoria 5 potevano arrivare facilmente a 100 metri e le fibre potevano estendersi per una distanza anche superiore. Il compromesso scelto fu di permettere tutte e tre le possibilità, come mostrato nella Figura 4.21, dando alla soluzione basata sui cavi di categoria 3 la capacità di trasporto aggiuntiva necessaria.

Nome	Cavo	Lunghezza max. segmenti	Vantaggi
100Base-T4	Doppino intrecciato	100 m	UTP di categoria 3
100Base-TX	Doppino intrecciato	100 m	Full duplex a 100 Mbps (UTP di categoria 5)
100Base-FX	Fibra ottica	2.000 m	Full duplex a 100 Mbps; distanze elevate

Figura 4.21. I cavi fast Ethernet originali.

Lo schema UTP di categoria 3, chiamato **100Base-T4**, utilizza una velocità di segnale di 25 MHz, solo del 25% più veloce rispetto ai 20 MHz di Ethernet standard. Come è stato spiegato precedentemente, la codifica Manchester della Figura 4.16 richiede due periodi di clock per ognuno dei 10 milioni di bit al secondo. Per raggiungere la banda necessaria 100Base-T4 richiede l'uso di quattro doppiini, ma poiché il cablaggio telefonico standard per decenni ha avuto quattro doppiini per cavo, la maggior parte degli uffici è in grado di gestire questa richiesta. Logicamente ciò significa non poter più disporre del telefono interno, ma questo è un piccolo prezzo da pagare in cambio di una posta elettronica più veloce.

Dei quattro doppiini, uno trasmette sempre all'hub, uno riceve sempre dall'hub e gli altri due sono commutabili nella direzione di trasmissione corrente. Per ottenere la banda necessaria è necessario abbandonare la codifica Manchester; con clock moderni e distanze così piccole non è più necessaria. Inoltre vengono trasmessi segnali ternari, cioè durante un singolo periodo di clock il cavo può contenere il valore 0, 1 o 2. Con tre doppiini dedicati alla trasmissione, adottando il segnale ternario è possibile trasmettere un simbolo qualunque scelto in un alfabeto di 27, quindi si possono inviare 4 bit con un po' di ridondanza. La trasmissione di 4 bit in ognuno dei 25 milioni di cicli di clock al secondo permette di raggiungere i 100 Mbps richiesti; a questi si aggiunge un canale inverso da 33,3 Mbps che utilizza il rimanente doppino. Questo schema, chiamato **8B/6T** (8 mappe bit a 6 trit), probabilmente non è molto elegante ma funziona con i cavi esistenti.

100Base-TX, il modello basato sui cavi di categoria 5, è più semplice perché i cavi possono gestire velocità di clock di 125 MHz. Sono utilizzati solo due doppiini per stazione, uno diretto all'hub e l'altro proveniente dall'hub. Anche in questo caso non è utilizzata la codifica binaria semplice, bensì uno schema chiamato **4B/5B** preso da FDDI e compatibile con esso. Ciascun gruppo di cinque periodi di clock, ognuno contenente uno dei due valori di segnale, permette di ottenere 32 combinazioni. Sedici di queste combinazioni sono utilizzate per trasmettere i quattro gruppi di bit 0000, 0001, 0010...1111; alcune delle rimanenti 16 sono utilizzate per funzioni di controllo, per esempio per indicare i limiti dei frame. Le combinazioni utilizzate sono state scelte attentamente per fornire transizioni sufficienti al mantenimento della sincronizzazione di clock. Il sistema 100Base-TX è full duplex: le stazioni possono trasmettere a 100 Mbps e ricevere contemporaneamente a 100 Mbps. Spesso, 100Base-TX e 100Base-T4 sono chiamate collettivamente **100Base-T**. L'ultima opzione, **100Base-FX**, utilizza due cavi di fibra multimodale, uno per ogni direzione, perciò il sistema è full duplex con 100 Mbps in ogni direzione e la distanza tra una stazione e l'hub può raggiungere i 2 Km.

Per rispondere alla domanda crescente, nel 1997 il comitato 802 aggiunse un nuovo tipo di cavo, 100Base-T2, che consentiva a fast Ethernet di funzionare su due doppiini esistenti di categoria 3. Poiché la gestione dello schema di codifica richiede un sofisticato processore di segnale digitale, questa tecnologia è molto più costosa delle precedenti; a causa della sua complessità, del costo e del fatto che molti edifici hanno già cavi UTP di categoria 5, è una soluzione utilizzata raramente.

Con 100Base-T è possibile utilizzare due tipi di dispositivi d'interconnessione: hub e switch (Figura 4.20). In un hub, tutte le linee in ingresso (o almeno tutte le linee che arri-

vano a una scheda dell'apparato) sono collegate logicamente in modo da costituire un singolo dominio di collisione. Sono applicate tutte le regole standard, inclusa quella dell'algoritmo di backoff esponenziale binario, perciò il sistema funziona proprio come una Ethernet di vecchio tipo; in particolare può trasmettere una sola stazione alla volta. In altre parole, gli hub richiedono la comunicazione half duplex.

In uno switch, ogni frame in arrivo sulla porta è memorizzato in un buffer, e se necessario è passato attraverso un backplane d'interconnessione ad alta velocità dalla porta sorgente alla porta di destinazione. Il backplane non è stato standardizzato, e non è necessario che lo sia poiché è interamente nascosto dentro lo switch. Se l'esperienza del passato si rivelerà di aiuto, i produttori di switch competiteranno tra loro per produrre backplane sempre più veloci e migliorare la capacità di trasporto del sistema. Poiché i cavi 100Base-FX sono troppo lunghi per l'algoritmo di collisione Ethernet normale, devono essere collegati a switch in modo tale che ognuno sia un dominio di collisione verso se stesso. Non sono consentiti hub con 100Base-FX.

Come nota finale, virtualmente tutti gli switch possono gestire combinazioni di stazioni a 10 Mbps e 100 Mbps, per facilitare l'aggiornamento della rete. Quando il numero di stazioni di lavoro a 100 Mbps acquistate aumenta, è sufficiente acquistare un adeguato numero di schede di linea a 100 Mbps e inserirle nello switch. Lo standard stesso permette a ogni coppia di stazioni di negoziare automaticamente la velocità ottimale (10 o 100 Mbps) e la modalità di trasmissione (half o full duplex). La maggior parte dei prodotti fast Ethernet utilizza questa funzionalità per autoconfigurarsi.

4.3.8 Gigabit Ethernet

Lo sviluppo dello standard fast Ethernet era stato da poco concluso quando il comitato 802 iniziò a lavorare su una Ethernet ancora più veloce (1995), subito soprannominata **gigabit Ethernet**. Lo standard fu ratificato da IEEE nel 1998 con il nome di 802.3z; questa identificazione suggerisce che Ethernet gigabit debba concludere il ciclo, a meno che qualcuno non inventi una lettera successiva alla "z". Questo paragrafo descrive alcune delle funzionalità chiave di Ethernet gigabit. Maggiori informazioni possono essere trovate in (Seifert, 1998).

Gli obiettivi del comitato 802.3z erano essenzialmente gli stessi del comitato 802.3u: rendere Ethernet 10 volte più veloce mantenendo la compatibilità con tutti gli standard Ethernet esistenti. In particolare, Ethernet gigabit doveva offrire servizi datagram senza acknowledge di tipo multicast e unicast, adottare lo stesso schema di indirizzamento a 48 bit già in uso e mantenere lo stesso formato di frame, incluse le dimensioni massima e minima. Lo standard finale risponde a tutti questi requisiti.

Tutte le configurazioni Ethernet gigabit sono punto-punto, abbandonando il sistema multidrop dello standard 10 Mbps originale che oggi viene chiamato **Ethernet classico**. Nella più semplice configurazione Ethernet gigabit, mostrata nella Figura 14.22(a), due computer sono collegati direttamente tra loro. Il caso più comune, comunque, è quello dove uno switch o un hub si collegano a più computer e possibilmente ad altri switch o hub, come mostrato nella Figura 14.22(b). In entrambe le configurazioni, ogni singolo cavo Ethernet è collegato a due e due soli dispositivi.

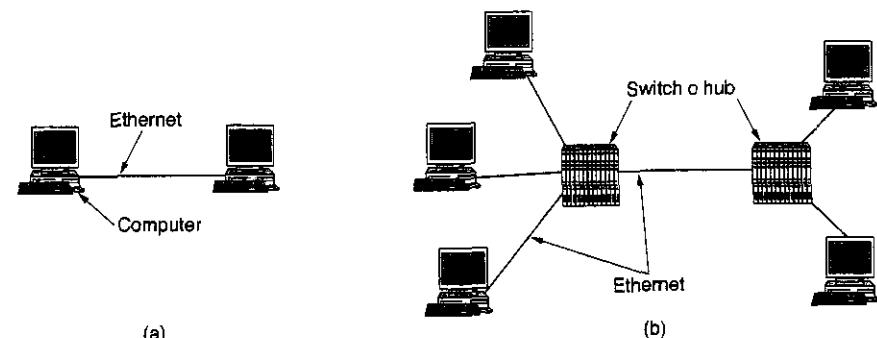


Figura 4.22. (a) Una Ethernet a due stazioni. (b) Una Ethernet a più stazioni.

Gigabit Ethernet supporta due modalità operative: full duplex e half duplex. La modalità normale è quella full duplex, che permette al traffico di viaggiare contemporaneamente in entrambe le direzioni. Questa modalità è utilizzata quando c'è uno switch centrale collegato ai computer (o ad altri switch) del perimetro. In questa configurazione tutte le linee hanno buffer, perciò ogni computer o switch è libero di inviare frame quando vuole. Il trasmettitore non deve esaminare il canale per vedere se qualcun altro lo sta già utilizzando, perché la contesa è impossibile. Sulla linea che collega un computer allo switch, il computer è l'unico che può trasmettere e la trasmissione ha successo anche se lo switch sta contemporaneamente trasmettendo un frame a un altro computer, perché la linea è full duplex. Poiché non si verifica contesa non si utilizza il protocollo CSMA/CD, perciò la lunghezza massima del cavo dipende dalla forza del segnale e non dal tempo che un burst di rumore impiega per propagarsi fino al trasmettitore nel peggior dei casi. Gli switch sono liberi di gestire più velocità, e l'autoconfigurazione è supportata proprio come in fast Ethernet.

L'altra modalità operativa, half duplex, è utilizzata quando i computer sono collegati a un hub e non a uno switch. L'hub non ha un buffer dove memorizzare i frame in arrivo; il dispositivo al suo interno collega elettricamente tutte le linee per simulare il cavo multidrop utilizzato nella Ethernet classica. In questa modalità le collisioni sono ancora possibili, perciò è richiesto il protocollo CSMA/CD standard.

Poiché il frame minimo (di 64 byte) può essere trasmesso 100 volte più velocemente che nella Ethernet classica, la distanza massima risulta 100 volte più corta: 25 metri. Ciò consente di preservare la proprietà essenziale per cui il trasmettitore sta ancora trasmettendo quando il burst di rumore torna indietro, anche nel peggior dei casi. Con un cavo lungo 2.500 metri, la trasmissione di un frame da 64 byte a 1 Gbps di velocità si concluderebbe quando il frame ha percorso appena un centesimo della distanza massima.

Il comitato 802.3z ha considerato inaccettabile un raggio di 25 metri e ha aggiunto allo standard due funzionalità che aumentano la portata. La prima funzionalità, chiamata **carrier extension**, essenzialmente dice all'hardware di aggiungere dei dati di riempimento dopo il frame normale in modo da estendere la dimensione del pacchetto fino a 512 byte. Poiché questo riempimento è aggiunto dall'hardware trasmettitore ed è rimosso dall'hard-

ware che riceve i dati, il software è ignaro della sua esistenza e non è necessario apportare alcuna modifica ai programmi esistenti. Naturalmente, utilizzare 512 byte di banda per trasmettere 46 byte di dati utente (il carico utile di un frame da 64 byte) comporta un'efficienza del 9%.

La seconda funzionalità, chiamata **frame bursting**, permette al trasmittente di inviare una sequenza concatenata di più frame in una singola trasmissione. Se il burst totale è minore di 512 byte, l'hardware aggiunge ancora i dati riempitivi. Se un numero sufficiente di frame è in attesa di trasmissione, questo schema è assai efficiente e preferibile alla carrier extension. Le nuove funzionalità estendono il raggio della rete a 200 metri, distanza che probabilmente è più che sufficiente per la maggior parte degli uffici.

Nella realtà è difficile immaginare che un'azienda, dopo aver implementato Ethernet gigabit per ottenere prestazioni migliori, colleghi i propri computer a un hub per simulare una Ethernet classica con tutte le sue collisioni. Gli switch sono poco più costosi degli hub, il prezzo delle schede Ethernet gigabit, invece, è ancora relativamente alto. È stupido acquistare un hub economico che riduce drasticamente le prestazioni del nuovo sistema, ma poiché la compatibilità verso il basso è considerata sacra nell'industria dei computer, il comitato 802.3z ha dovuto inserire anche questo meccanismo.

Ethernet gigabit supporta cavi in rame e in fibra, come mostrato nella Figura 4.23. Con segnali che viaggiano a 1 Gbps attraverso le fibre, la sorgente luminosa deve essere accesa e spenta in meno di 1 nsec. I LED non sono in grado di operare così velocemente, perciò è necessario utilizzare laser. Sono ammesse due lunghezze d'onda: 0,85 micron (corta) e 1,3 micron (lunga). I laser a 0,85 micron sono più economici ma non funzionano su una fibra monomodale.

Nome	Cavo	Lunghezza massima del segmento	Vantaggi
1000Base-SX	Fibra ottica	550 m	Fibra multimodale (50, 62,5 micron)
1000Base-LX	Fibra ottica	5.000 m	Fibra mono (10 μ) o multimodale (50, 62,5 μ)
1000Base-CX	2 coppie di STP	25 m	Doppino schermato
1000Base-T	4 coppie di UTP	100 m	UTP standard di categoria 5

Figura 4.23. Cavi Ethernet gigabit.

Le fibre possono avere diametro 10, 50 e 62,5 micron. Il primo è per la modalità monomodale e gli ultimi due sono per le fibre multimodali; tuttavia non tutte e sei le combinazioni sono possibili, e la distanza massima dipende dalla combinazione scelta. I numeri riportati nella Figura 4.23 si riferiscono al caso ottimale e, in particolare, si possono raggiungere i 5.000 metri solo con laser di 1,3 micron che operano su fibre a 10 micron monomodali. Questa è la scelta migliore per le dorsali delle università, e per questo motivo ci si aspetta che diventi popolare nonostante sia la scelta più costosa.

L'opzione 1000Base-CX utilizza cavi in rame schermati; il suo problema è che deve competere verso l'alto con la fibra ad alte prestazioni, e verso il basso con gli economici UTP. Probabilmente non sarà molto utilizzata.

L'ultima opzione utilizza il cavo con quattro coppie UTP di categoria 5. Poiché molti di questi cavi sono già installati, probabilmente questa soluzione diventerà la Ethernet gigabit dei poveri.

Gigabit Ethernet utilizza nuove regole di codifica sulle fibre. La codifica Manchester a 1 Gbps richiederebbe un segnale a 2 gigabaud, troppo difficile da realizzare e troppo dispendioso dal punto di vista della banda. È stato perciò scelto un nuovo schema chiamato **8B/10B** basato su fibre channel. Ogni byte da 8 bit è codificato sulla fibra usando 10 bit, da qui il nome 8B/10B. Poiché ci sono 1.024 possibili codeword per ogni byte in input, c'è un po' di margine nella scelta dei codeword consentiti, che sono selezionati seguendo due regole:

1. nessun codeword può avere più di quattro bit identici consecutivi
2. nessun codeword può avere più di sei 0 o sei 1.

Queste scelte sono state fatte per ottenere nel flusso un numero sufficiente di transizioni, che garantiscono che il ricevitore rimanga sincronizzato con il trasmittitore, e per tenere il numero di 0 nella fibra il più vicino possibile a quello degli 1. Inoltre, a molti byte in input sono assegnati due diversi codeword. Quando ha la possibilità di farlo, il codificatore sceglie sempre il codeword che aiuta a bilanciare il numero di 0 e di 1 trasmessi. L'attenzione al bilanciamento di 0 e 1 è necessaria per minimizzare la componente DC del segnale, e quindi permettere il passaggio senza modifiche attraverso i trasformatori. I ricercatori informatici non apprezzano che le proprietà dei trasformatori dettino condizioni ai loro schemi di codifica, ma qualche volta va così.

Ethernet gigabit basata su 1000Base-T utilizza uno schema di codifica diverso, poiché sul cavo in rame è troppo difficile mantenere temporizzazioni dei dati di 1 nsec. Questa soluzione utilizza quattro doppini di categoria 5 per consentire la trasmissione in parallelo di quattro simboli. Ogni simbolo è codificato usando uno dei cinque livelli di tensione disponibili. Questo schema permette a un singolo simbolo di codificare 00, 01, 10, 11 o uno speciale valore di controllo, pertanto sono inviati 2 bit di dati per doppino o 8 bit di dati per ciclo di clock. Il clock funziona a 125 MHz, permettendo operazioni a 1 Gbps. Sono stati consentiti cinque livelli di tensione invece di quattro per poter disporre di combinazioni per la gestione dei frame e le operazioni di controllo.

La velocità di 1 Gbps è decisamente alta. Per esempio, se un ricevitore è occupato in qualche altra operazione anche per 1 msec e non svuota il buffer di input su qualche linea, in quell'intervallo di 1 msec possono accumularsi fino a 1.953 frame. Il buffer si sovraccarica anche quando un computer su una Ethernet gigabit scambia dati con un computer collegato a una Ethernet classica. La conseguenza di queste due osservazioni è che gigabit Ethernet supporta il controllo di flusso (così come fast Ethernet, ma la tecnica è diversa). Il controllo di flusso funziona in questo modo: l'apparato su un'estremità della connessione trasmette un frame di controllo speciale all'altro, dicendo di sospendere la trasmissione per un certo periodo di tempo. I frame di controllo sono normali frame Ethernet contenenti il tipo 0x8808. I primi due byte del campo dati danno il comando; i byte successivi

forniscono i parametri, se ce ne sono. Per il controllo di flusso si utilizzano frame PAUSE, con il parametro che indica la durata della pausa espressa in unità pari al minimo tempo di frame. Nel caos delle Ethernet gigabit l'unità di tempo è 512 nsec, quindi le pause durano fino a 33,6 msec.

Non appena Ethernet gigabit venne standardizzata, il comitato 802 chiese di tornare subito al lavoro. IEEE gli diede l'incarico di studiare uno standard Ethernet 10 Gigabit. Dopo aver cercato inutilmente una nuova lettera da utilizzare dopo la "z", il comitato abbandonò l'approccio e inizio ad adottare i suffissi di due lettere. Tornati al lavoro svilupparono uno standard che fu approvato da IEEE nel 2002 con il nome di 802.3ae. A quando lo standard 100 gigabit Ethernet?

4.3.9 IEEE 802.2: LLC (*Logical Link Control*)

È giunto il momento di tornare indietro e confrontare quanto studiato in questo capitolo con quello che è stato spiegato nel precedente. Il Capitolo 3 ha mostrato in che modo due macchine potrebbero comunicare in modo affidabile attraverso una linea non affidabile, usando svariati protocolli data link. Questi protocolli forniscono funzionalità di controllo d'errore (basato sul riconoscimento) e di controllo di flusso (basato su finestre scorrevoli).

Questo capitolo invece non ha detto nulla sulla comunicazione affidabile. Tutto quello che Ethernet e gli altri protocolli 802 offrono è un servizio datagram best-effort, che qualche volta è già sufficiente. Per esempio, per trasportare i pacchetti IP non è richiesta né attesa alcuna garanzia. Un pacchetto IP può essere inserito in un campo che rappresenta il carico utile di un frame 802 ed essere spedito verso la sua destinazione; se il frame si perde, non importa.

Esistono però sistemi in cui è preferibile adottare un protocollo data link con controllo di errore e di flusso. IEEE ne ha definito uno che può operare sopra Ethernet e gli altri protocolli 802. Inoltre, questo protocollo chiamato **LLC** (*Logical Link Control*) nasconde le differenze tra vari tipi di reti 802 offrendo interfaccia e formato unici verso lo strato network. Il formato, l'interfaccia e il protocollo si basano strettamente sul protocollo HDLC studiato nel Capitolo 3. LLC forma la metà superiore dello strato data link, mentre il sottostrato MAC si trova proprio al di sotto, come mostrato nella Figura 4.24.

In genere LLC è utilizzato in questo modo: lo strato network sulla macchina trasmittente passa un pacchetto a LLC, usando le primitive di accesso di LLC; il sottostrato LLC aggiunge un'intestazione LLC contenente i numeri di sequenza e di acknowledge. La struttura risultante è poi inserita nel campo carico utile di un frame 802, che viene infine trasmesso attraverso la connessione. Il ricevitore esegue il procedimento inverso.

LLC fornisce tre modalità di servizio: datagram inaffidabile, datagram con acknowledge e servizio affidabile orientato alle connessioni. L'intestazione LLC contiene tre campi: access point di destinazione, access point sorgente e un campo di controllo. Gli access point indicano il processo che ha generato il frame e quello di destinazione; queste informazioni sostituiscono il campo *tipo* di DIX. Il campo di controllo contiene i numeri di

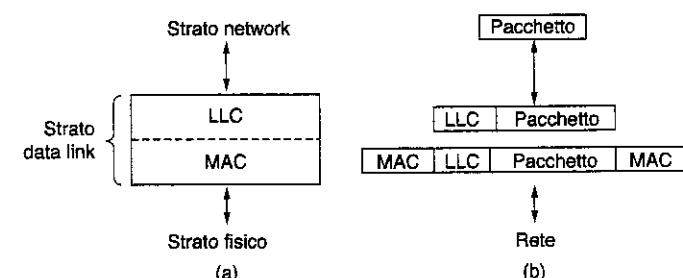


Figura 4.24. (a) Posizione di LLC. (b) Formati di protocollo.

sequenza e di acknowledge, seguendo un approccio simile a quello di HDLC (Figura 3.24) ma non identico. Questi campi sono utilizzati principalmente quando è richiesta una connessione affidabile sul livello data link; in questo caso si utilizzano protocolli simili a quelli descritti nel Capitolo 3. Per Internet sono sufficienti i tentativi di trasmissione best-effort dei pacchetti IP, perciò a livello LLC non è richiesta alcuna operazione di acknowledge.

4.3.10 Retrospettiva su Ethernet

Ethernet è utilizzata da più di 20 anni e non ha alcun serio concorrente in vista, perciò è probabile che continuerà a dominare il mercato per molti anni a venire. Poche architetture di CPU, sistemi operativi o linguaggi di programmazione sono rimasti i re della foresta per tre decenni. Chiaramente, Ethernet ha fatto una scelta vincente. Quale?

Probabilmente questa longevità dipende dalla semplicità e dalla flessibilità. In pratica, "semplice" si traduce in affidabile, economico e facile da mantenere. Dopo aver sostituito i connettori a vampiro con i connettori BNC, i guasti sono diventati estremamente rari. Le persone preferiscono non sostituire le cose che funzionano perfettamente, specialmente quando sanno che nel settore informatico ci sono un sacco di elementi che non funzionano bene, tanto che alcuni "aggiornamenti" spesso dimostrano di essere peggiori dei loro predecessori.

Semplice si traduce anche in economico. Thin Ethernet e i doppini costano relativamente poco. Anche le schede di rete sono economiche. Hub e switch hanno richiesto un investimento più robusto, ma quando hanno fatto la loro comparsa lo standard Ethernet si era già completamente radicato nel settore.

Ethernet è anche facile da mantenere; non richiede l'installazione di alcun software (a parte i driver) e non dipende da tabelle di configurazione difficili da gestire. Inoltre, per aggiungere un nuovo host non bisogna far altro che collegare un cavo.

Un altro dettaglio importante è che Ethernet è in grado di funzionare con TCP/IP, che ormai è diventato un protocollo dominante. IP è un protocollo senza connessione, perciò si adatta perfettamente a Ethernet, a sua volta senza connessione. IP si adatta peggio ad ATM, che è orientato verso le connessioni; questo accoppiamento sbagliato distrugge definitivamente le possibilità di ATM.

Infine, Ethernet è stato in grado di evolversi dove più se ne avvertiva il bisogno: le velocità sono aumentate di diversi ordini di grandezza e sono stati introdotti hub e switch, ma queste modifiche non hanno richiesto alcun cambiamento di software. Un commerciale che vuole vendere una rete a una grande azienda dicendo: "Ecco, ho questa rete fantastica che è perfetta per la tua azienda. Non devi fare altro che buttar via tutto il vecchio hardware e riscrivere tutto il software" ha qualche serio problema. FDDI, Fibre Channel e ATM erano tutte più veloci di Ethernet quando vennero introdotte, ma erano anche incompatibili con Ethernet, erano molto più complesse e più difficili da gestire. Alla fine, quando anche la velocità di Ethernet raggiunse quella delle altre reti, l'ultimo vantaggio rimasto sparì insieme alle reti concorrenti, con l'unica eccezione di ATM ancora utilizzata come struttura di base del sistema telefonico.

4.4 LAN wireless

Sebbene sia largamente utilizzata, Ethernet sta per scontrarsi con un nuovo concorrente. Le LAN wireless sono sempre più popolari, e un numero crescente di uffici, aeroporti e altri luoghi pubblici ne vengono equipaggiati. Le LAN wireless si possono configurare in due modi, come mostrato nella Figura 1.35: con oppure senza stazione base. Lo standard LAN 802.11 ne tiene conto, quindi prevede entrambe le soluzioni.

Nel Paragrafo 1.5.4 sono state fornite alcune informazioni di base sullo standard 802.11, ma ora è arrivato il momento di esaminare questa tecnologia con maggiore attenzione. Nei prossimi paragrafi saranno esaminate la pila di protocolli, le tecniche di trasmissione radio dello strato fisico, il protocollo del sottostrato MAC, la struttura del frame e i servizi. Per maggiori informazioni su 802.11, si consulti (Crow et al., 1997; Geier, 2002; Heegard et al., 2001; Kapp, 2002; O'Hara and Petrick, 1999; e Severance, 1999).

4.4.1 La pila di protocolli 802.11

I protocolli utilizzati da tutte le varianti di 802, inclusa Ethernet, hanno una struttura simile. La Figura 4.25 offre una visione parziale della pila di protocolli 802.11. Lo strato fisico corrisponde più o meno allo strato fisico OSI, ma in tutti i protocolli 802 lo strato data link è diviso in due o più sottostrati.

In 802.11, il sottostrato MAC (*Medium Access Control*) stabilisce il metodo di allocazione del canale, cioè chi deve trasmettere per primo. Al di sopra si trova il sottostrato LLC (*Logical Link Control*), che ha il compito di nascondere le differenze tra le varianti di 802 e renderle indistinguibili allo strato di rete. LLC è stato descritto precedentemente, nel paragrafo dedicato a Ethernet.

Lo standard 802.11 del 1997 specifica tre tecniche di trasmissione supportate nello strato fisico. Il metodo a infrarossi utilizza più o meno la stessa tecnologia dei telecomandi dei televisori; le altre due usano un sistema radio a bassa potenza basato sulle

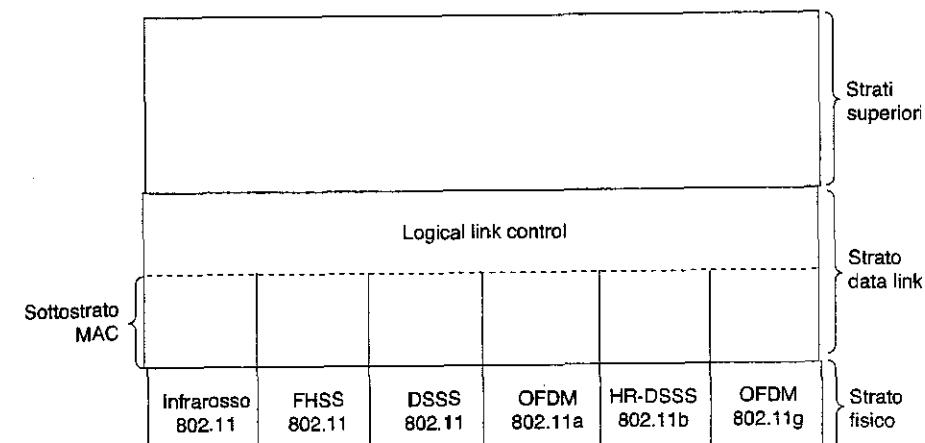


Figura 4.25. Una parte della pila di protocolli 802.11.

tecniche FHSS e DSSS e utilizzano una parte dello spettro che non richiede licenze (la banda ISM 2,4 GHz).

Anche i telecomandi usati per aprire i portoni dei garage usano questa parte dello spettro, perciò il proprio computer portatile può scoprire di essere in competizione con il portone del garage; telefoni cordless e forni a microonde sono altri utenti di questa banda. Entrambe le tecniche di trasmissione operano a 1 o 2 Mbps con potenza abbastanza ridotta, che non crea molti conflitti. Per aumentare la banda disponibile, nel 1999 sono state introdotte due nuove tecnologie chiamate OFDM e HR-DSSS, che operano rispettivamente a 54 e 11 Mbps. Nel 2001 è stata introdotta una seconda modulazione OFDM, ma in una banda di frequenza diversa dalla prima. I prossimi paragrafi descrivono brevemente ciascuna soluzione. Dal punto di vista tecnico appartengono allo strato fisico e la loro descrizione sarebbe dovuta essere nel Capitolo 2, tuttavia la spiegazione è stata inserita in questo capitolo poiché sono strettamente collegate alle LAN in senso generale e al sottostrato MAC di 802.11.

4.4.2 Lo strato fisico di 802.11

Ognuna delle cinque tecniche di trasmissione consente di inviare un frame MAC da una stazione a un'altra, ma le soluzioni differiscono sia per la tecnologia utilizzata sia per le velocità supportate. La descrizione dettagliata di queste tecnologie non rientra tra gli obiettivi del volume, comunque può essere utile spendere due parole e spiegare il significato di alcuni termini-chiave, per fornire ai lettori interessati uno spunto dal quale iniziare una ricerca più approfondita in Internet o da qualche altra parte.

La variante a infrarossi utilizza una trasmissione diffusa (ossia non in linea retta) a 0,85 e 0,95 micron e supporta due velocità: 1 Mbps e 2 Mbps. A 1 Mbps si usa uno schema di codifica dove un gruppo di 4 bit è codificato come una codeword di 16 bit contenente quindici 0 e un singolo 1 usando il cosiddetto **codice Gray**. Questo codice ha la proprietà che un piccolo errore nella sincronizzazione causa la perdita di un solo bit di output. A 2 Mbps, la codifica prende 2 bit e produce una parola in codice di 4 bit contenente un unico 1 (cioè 0001, 0010, 0100 oppure 1000). I segnali a infrarossi non possono passare attraverso i muri, perciò quelle in stanze diverse restano ben isolate le une dalle altre, tuttavia questa tecnica non è molto popolare per colpa della banda scarsa e del fatto che la luce del sole interferisce con i segnali infrarossi.

FHSS (*Frequency Hopping Spread Spectrum*) utilizza 79 canali, larghi 1 MHz ciascuno, che iniziano dal margine inferiore della banda ISM a 2,4 GHz; si usa un generatore di numeri pseudocasuali per produrre la sequenza con cui le frequenze si susseguono. Tutte le stazioni salteranno contemporaneamente nelle stesse frequenze a patto di usare lo stesso seme per generare i numeri casuali e di mantenere la sincronizzazione. La quantità di tempo spesa in ogni frequenza, ossia il **tempo di rotazione**, è un parametro regolabile ma deve essere minore di 400 msec. La casualità di FHSS rappresenta un modo efficace di allocare lo spettro nella banda ISM non regolata, inoltre permette di ottenere un po' di sicurezza poiché un intruso che non conosce la sequenza di salto o il tempo di rotazione non è in grado di intercettare le trasmissioni. Sulle distanze più lunghe, dove il multipath fading può creare problemi, FHSS dimostra di essere abbastanza resistente; è anche relativamente insensibile all'interferenza radio, il che lo rende popolare per i collegamenti tra edifici. Il suo svantaggio principale è la banda ridotta.

Anche il terzo sistema di modulazione, **DSSS** (*Direct Sequence Spread Spectrum*) è limitato a 1 oppure 2 Mbps. Lo schema usato assomiglia al sistema CDMA descritto nel Paragrafo 2.6.2, con alcune importanti differenze. Ogni bit è trasmesso come 11 chip, usando quella che viene chiamata **sequenza Barker**: utilizza la modulazione di fase a 1 Mbaud, per trasmettere 1 bit per baud quando si opera a 1 Mbps e 2 bit per baud quando si opera a 2 Mbps. Per anni, FCC ha richiesto che negli Stati Uniti tutti i dispositivi di comunicazione wireless operanti nelle bande ISM utilizzassero lo spettro distribuito, ma nel maggio 2002 la regola è stata abbandonata in seguito all'emergere di nuove tecnologie.

La prima LAN wireless ad alta velocità, **802.11a**, utilizza **OFDM** (*Orthogonal Frequency Division Multiplexing*) per distribuire fino a 54 Mbps nella banda ISM dei 5 GHz, che è più ampia. Come si può intuire dal nome, questo sistema utilizza similmente ad ADSL molte frequenze (per la precisione 52), 48 delle quali per i dati e 4 per la sincronizzazione. Poiché sono presenti trasmissioni su più frequenze nello stesso momento, la tecnica è considerata una forma di diffusione di spettro, diversa però da CDMA e FHSS. La divisione del segnale in tante bande strette offre vantaggi vitali rispetto all'utilizzo di una singola banda larga, per esempio una migliore resistenza alle interferenze con banda stretta e la possibilità di utilizzare bande non contigue. Per raggiungere velocità fino a 18 Mbps si usa un complesso sistema di codifica basato sulla modulazione di fase, passando a QAM per velocità maggiori. A 54 Mbps, 216 bit di dati sono codificati in simboli da 288 bit.

OFDM è stato scelto anche per la sua compatibilità con il sistema europeo HiperLAN/2 (Doufexi et al., 2002). La tecnica ha una buona efficienza di spettro in termini di bit/Hz e una buona resistenza al multipath fading.

Il passo successivo è **HR-DSSS** (*High Rate Direct Sequence Spread Spectrum*), altra tecnica a diffusione di spettro che raggiunge la velocità di 11 Mbps nella banda 2,4 GHz utilizzando 11 milioni di chip al secondo. È chiamata **802.11b**, ma non è il seguito di 802.11a; anzi, questo standard è stato approvato ed è arrivato sul mercato prima di 802.11a. Le velocità supportate da 802.11b sono 1, 2, 5,5 e 11 Mbps. Le due velocità più basse funzionano a 1 Mbaud, rispettivamente con 1 e 2 bit per baud, usando la modulazione di fase per mantenere la compatibilità con DSSS. Le ultime due velocità operano a 1,375 Mbaud, con 4 e 8 bit per baud, usando i codici **Walsh/Hadamard**. Il data rate è adattabile dinamicamente durante il funzionamento, per ottenere la migliore velocità possibile nelle condizioni di carico e rumore presenti. All'atto pratico la velocità operativa di 802.11b è quasi sempre di 11 Mbps. 802.11b è più lenta di 802.11a ma il campo d'azione è circa sette volte più ampio e in molte situazioni questa è una caratteristica più importante.

Una versione evoluta di 802.11b, chiamata **802.11g**, è stata approvata da IEEE nel novembre 2001 dopo alcune lotte politiche sulla scelta della tecnologia brevettata da utilizzare. Questo standard utilizza il metodo di modulazione OFDM di 802.11a, ma opera nella più ristretta banda ISM dei 2,4 GHz come 802.11b. In teoria può operare fino a 54 Mbps, ma non è ancora chiaro se questa velocità sarà praticamente realizzabile. Ricapitolando, il comitato 802.11 ha prodotto tre diverse LAN wireless veloci: 802.11a, 802.11b e 802.11g (oltre alle tre LAN wireless a bassa velocità). È legittimo chiedersi se è vantaggioso che un comitato impegnato nello sviluppo degli standard adotti un comportamento del genere; forse 3 è il loro numero fortunato.

4.4.3 Il protocollo del sottostrato MAC di 802.11

È il momento di abbandonare la terra dell'ingegneria elettronica per tornare nel paese dell'informatica. Il protocollo del sottostrato MAC 802.11 è leggermente diverso da quello Ethernet, a causa della complessità intrinseca dell'ambiente wireless. Con Ethernet una stazione si limita ad aspettare che il mezzo di trasmissione diventi silenzioso, quindi inizia a trasmettere. Se non riceve in risposta un burst di rumore entro i primi 64 byte, il frame ha quasi certamente raggiunto la destinazione senza problemi. Con il wireless questa situazione non regge.

Tanto per cominciare c'è il problema della stazione nascosta menzionato precedentemente e illustrato nuovamente nella Figura 4.26(a). Poiché non tutte le stazioni sono all'interno del campo radio delle altre, le trasmissioni che avvengono in una parte della cella possono non essere ricevute in un'altra parte della stessa cella. Nella figura d'esempio, la stazione *C* sta trasmettendo alla stazione *B*; dopo aver controllato il canale senza aver captato alcunché, *A* conclude erroneamente che può iniziare a trasmettere a *B*.

C'è poi il problema inverso, detto della stazione esposta, illustrato nella Figura 4.26(b). In questo caso *B*, che vuole trasmettere dei dati a *C*, si mette in ascolto sul canale; quando *B*

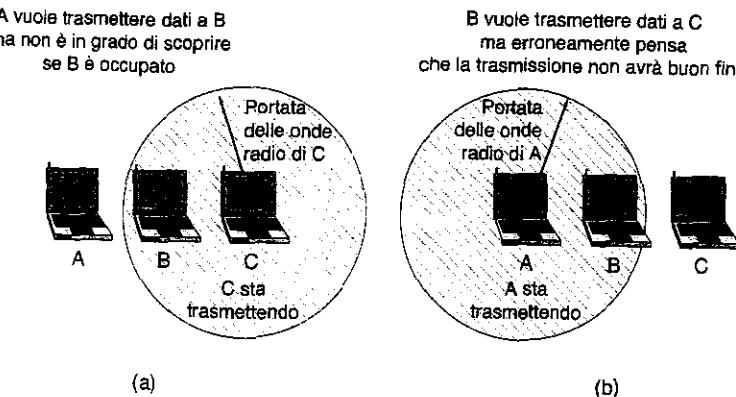


Figura 4.26. (a) Il problema della stazione nascosta. (b) Il problema della stazione esposta.

rileva una trasmissione, conclude erroneamente che non può trasmettere a *C* anche se magari è *A* che sta trasmettendo alla stazione *D* (non rappresentata nel disegno).

Inoltre, la maggior parte delle radio è half duplex: non può trasmettere e contemporaneamente rimanere in ascolto dei burst di rumore su una singola frequenza. A causa di tutti questi problemi, 802.11 non utilizza CSMA/CD come invece fa Ethernet.

Per gestire il problema 802.11 supporta due modalità operative. La prima, chiamata **DCF** (*Distributed Coordination Function*), non utilizza alcun tipo di controllo centrale (in questo senso è simile a Ethernet); l'altra, chiamata **PCF** (*Point Coordination Function*), usa la stazione base per controllare tutta l'attività nella cella. Tutte le implementazioni devono supportare DFC; al contrario PCF è opzionale.

Quando adotta DFC, 802.11 utilizza un protocollo chiamato **CSMA/CA** (*CSMA/Collision Avoidance*). In questo protocollo sono controllati sia il canale fisico sia quello virtuale. CSMA/CD supporta due modalità operative. Nella prima modalità, la stazione controlla se il canale è libero prima di trasmettere i dati; durante la trasmissione la stazione non controlla il canale ma emette l'intero frame, che può comunque rimanere danneggiato a causa delle interferenze sul ricevitore. Se il canale è occupato il trasmettitore rimanda l'operazione fino a quando non diventa libero, e poi inizia a inviare i dati. In caso di collisione, le stazioni coinvolte rimangono in attesa per un tempo casuale usando l'algoritmo di backoff esponenziale binario, quindi ritentano.

L'altra modalità operativa di CSMA/CD si basa su MACAW e utilizza il controllo di presenza del canale virtuale, come mostrato nella Figura 4.27. Nell'esempio mostrato, *A* desidera trasmettere a *B*. *C* è una stazione che si trova nel raggio di azione di *A* (e può darsi anche in quello di *B*, ma questo non importa). *D* è una stazione che si trova nel campo di *A* ma non in quello di *A*.

Il protocollo entra in azione quando *A* decide di voler inviare dati a *B*. Prima invia a *B* un frame RTS che richiede il permesso di trasmettergli un frame. Quando riceve questa

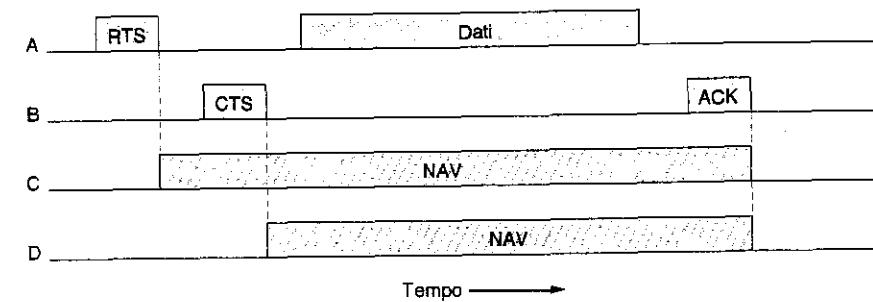


Figura 4.27. Il controllo del canale virtuale mediante CSMA/CA.

richiesta, *B* può decidere di concedere il permesso e, in questo caso, invia in risposta un frame CTS.

Dopo aver ricevuto il frame CTS, *A* invia il suo frame e fa partire un timer ACK. Dopo aver ricevuto il frame di dati, *B* risponde con un frame ACK che conclude lo scambio. Se un timer ACK di *A* scade prima dell'arrivo del frame ACK, l'intero protocollo ricomincia daccapo.

Ora si consideri questo scambio dal punto di vista di *C* e di *D*. *C* si trova nel campo di *A*, perciò riceve il frame RTS. Intuendo che qualcuno sta per iniziare una trasmissione di dati, per il bene di tutti evita di inviare alcunché fino al termine dello scambio. Grazie alle informazioni passate attraverso la richiesta RTS, la stazione può calcolare la durata della sequenza, incluso l'ACK finale, perciò rivendica per sé una specie di canale virtuale indicato nella Figura 4.27 dalla sigla NAV (Network Allocation Vector). *D* non sente il frame RTS ma rileva il frame CTS, di conseguenza anche questa stazione attiva il segnale NAV per sé. Si noti che i segnali NAV non sono trasmessi, sono semplici promemoria interni che inducono a rimanere tranquilli per un certo periodo di tempo.

A differenza delle reti cablate, le reti wireless sono rumorose e inaffidabili, anche per colpa dei fornitori a microonde che utilizzano le stesse bande ISM senza licenze. Di conseguenza, la probabilità che un frame arrivi a destinazione senza problemi diminuisce con la lunghezza del frame. Se p è la probabilità che un bit subisca danni, allora la probabilità che un frame lungo n bit sia ricevuto interamente senza errori è uguale a $(1-p)^n$. Per esempio, per $p = 10^{-4}$ la probabilità di ricevere un frame Ethernet completo (12144 bit) è minore del 30%. Se $p = 10^{-5}$, circa un frame ogni 9 rimarrà danneggiato. Anche se $p = 10^{-6}$, più dell'1% dei frame subirà danni, il che significa quasi una dozzina di frame al secondo, e molti di più se si utilizzano frame con lunghezza minore di quella massima. Riassumendo, un frame troppo lungo ha poche possibilità di raggiungere la destinazione intatto, perciò è probabile che debba essere ritrasmesso.

Per risolvere il problema dei canali rumorosi, 802.11 ammette la frammentazione dei frame in parti più piccole, ognuna dotata del proprio checksum di controllo. I frammenti sono numerati e ricevono l'acknowledgement individualmente con un protocollo stop-and-wait (per esempio, il trasmittente non può inviare il frammento $k + 1$ fino a quando

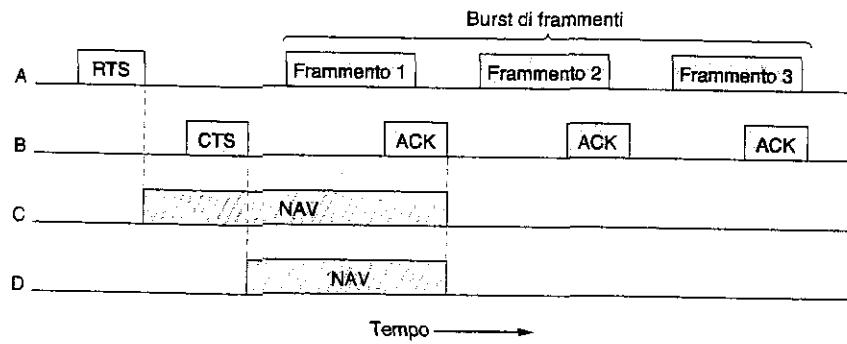


Figura 4.28. Un burst di frammenti.

non ha ricevuto l'acknowledgement per il frammento k). Dopo aver acquisito il canale mediante RTS e CTS si possono inviare più frammenti uno dietro l'altro, come mostrato nella Figura 4.28. La sequenza di frammenti è detta **burst di frammenti**.

La frammentazione aumenta la capacità del canale, perché permette di ritrasmettere solo i frammenti danneggiati invece dell'intero frame. La dimensione di frammento non è fissata dallo standard ma è un parametro di ogni cella, e può essere regolata dalla stazione base. Il meccanismo NAV tiene ferme le altre stazioni solo fino all'acknowledgement successivo, e si usa un altro meccanismo (descritto più avanti) per consentire la trasmissione senza interferenza di un intero burst di frammenti.

Quanto è stato detto si applica alla modalità DCF di 802.11. In questo modalità non esiste alcun controllo centralizzato e le stazioni concorrono per ottenere il controllo del mezzo di trasmissione, proprio come accade in Ethernet. Nell'altra modalità ammessa, PCF, una stazione base sonda le altre stazioni chiedendo se hanno frame da trasmettere. Poiché nella modalità PCF l'ordine di trasmissione è completamente controllato dalla stazione base, non avviene mai alcuna collisione. Lo standard prescrive il meccanismo d'interrogazione ma non la frequenza di tale operazione, né l'ordine o l'eventuale gestione delle priorità di servizio. Il meccanismo principale adottato dalla stazione di base è quello della trasmissione broadcast periodica (10 o 100 volte al secondo) di un **frame di segnalazione**. Questo frame di segnalazione contiene i parametri di sistema: sequenze di salto e tempi di rotazione (per FHSS), sincronizzazione di clock e così via. Inoltre, il frame invita le nuove stazioni a effettuare la registrazione al servizio di interrogazione. Appena una stazione aderisce al servizio d'interrogazione per una specifica cadenza, riceve una certa frazione di banda. Ciò consente di dare garanzie sulla qualità del servizio.

La durata della batteria è sempre un problema con i dispositivi wireless, perciò 802.11 si preoccupa anche della gestione dell'energia. In particolare, la stazione base può obbligare una stazione mobile ad attivare la modalità di sospensione; il dispositivo rimarrà inattivo fino a quando non interviene l'utente o la stazione base. Poiché può sospendere l'attività di un'altra stazione, la stazione base è responsabile dei frame e deve memorizzare in un buffer i dati diretti alle stazioni, se queste sono state disattivate. I frame memorizzati sono successivamente raccolti dalle stazioni al loro risveglio.

PCF e DCF possono coesistere dentro la stessa cella. A prima vista può sembrare impossibile far operare contemporaneamente un controllo centrale e un controllo distribuito, ma 802.11 offre un metodo per farlo: funziona definendo attentamente l'intervallo tra i frame. Dopo la trasmissione di un frame è richiesta una pausa prima di qualsiasi nuova trasmissione; lo standard prevede quattro intervalli, rappresentati nella Figura 4.29, ognuno dei quali ha una funzione specifica.

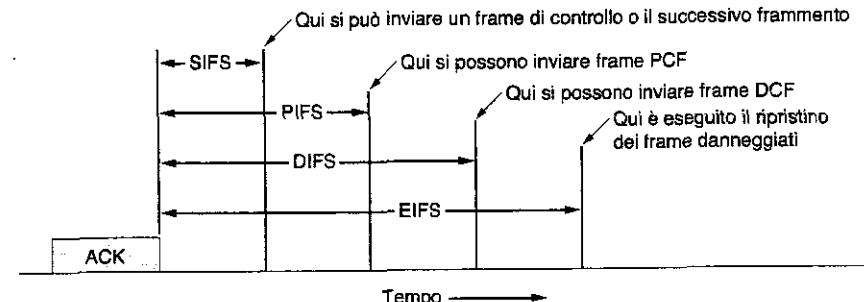


Figura 4.29. Intervalli tra frame in 802.11.

L'intervallo più breve è chiamato **SIFS** (Short InterFrame Spacing) ed è utilizzato per concordare i turni tra le parti coinvolte nella singola conversazione. Senza dover spedire di nuovo un RTS, è necessario tra l'altro attendere che il ricevitore invii un frame CTS per rispondere a un frame RTS, quindi che il ricevitore mandi un ACK per un frammento o un frame completo, e infine che il trasmettitore di un burst di frammenti trasmetta il frammento successivo. C'è sempre un'unica stazione autorizzata a rispondere dopo un intervallo SIFS. Se questa possibilità non viene sfruttata e l'intervallo **PIFS** (PCF InterFrame Spacing) finisce, la stazione base può inviare un frame di segnalazione o di interrogazione. Questo meccanismo permette a una stazione che sta trasmettendo un frame di dati o una serie di frammenti di concludere il frame senza che nessuno si metta in mezzo, ma dà alla stazione base la possibilità di agguantare il canale quando il trasmettitore precedente ha concluso, senza dover competere con utenti avidi. Se la stazione base non ha nulla da dire e trascorre un intervallo **DIFS** (DCF InterFrame Spacing), qualunque stazione può tentare di acquisire il controllo del canale per inviare un nuovo frame. In questo caso si applicano le solite regole della contesa, e in caso di collisioni può essere attivato l'algoritmo di backoff esponenziale binario.

L'ultimo intervallo, **EIFS** (Extended InterFrame Spacing), è utilizzato solo da una stazione che ha appena ricevuto un frame danneggiato o sconosciuto, e serve per annunciarlo. Questo evento ha la priorità più bassa perché il ricevitore potrebbe non avere idea di quello che sta accadendo. Per evitare interferenza con una comunicazione in corso tra altre due stazioni, deve restare in attesa per un tempo considerevole.

4.4.4 La struttura del frame di 802.11

Nelle reti cablate lo standard 802.11 definisce tre diverse classi di frame: dati, controllo e gestione. Ognuna ha un'intestazione composta da campi utilizzati nel sottostrato MAC. Ci

sono anche delle intestazioni usate dallo strato fisico, ma hanno per lo più a che fare con le tecniche di modulazione adottate, perciò non saranno analizzate nel corso di questo paragrafo.

Il formato del frame di dati è mostrato nella Figura 4.30. Il primo campo, che si chiama *frame control*, è composto da 11 campi secondari. Il primo di questi è *protocol version* e permette a due versioni del protocollo di operare contemporaneamente nella stessa cella. Seguono il campo *tipo* (dati, controllo o gestione) e il campo *subtype* (per esempio RTS o CTS). I bit *to DS* e *from DS* indicano se il frame sta andando verso il sistema di distribuzione tra celle (per esempio Ethernet) o se proviene da esso. Il bit *MF* indica se seguiranno più frammenti. Il bit *retry* contrassegna la ritrasmissione di un frame trasmesso precedentemente. Il bit *power management* è utilizzato dalla stazione base per attivare e disattivare la modalità di sospensione del ricevitore. Il bit *more* indica che il trasmettitore ha altri frame per il ricevitore. Il bit *W* specifica che il corpo del frame è stato cifrato usando l'algoritmo **WEP** (*Wired Equivalent Privacy*). Infine, il bit *O* indica al ricevitore che la sequenza di frame con questo bit attivo va elaborata rispettando l'ordine.

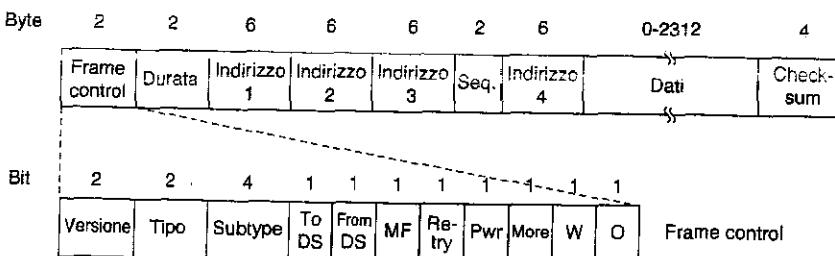


Figura 4.30. Il frame dati 802.11.

Il secondo campo del frame di dati, *durata*, indica per quanto tempo il frame e il suo acknowledgement occuperanno il canale. Questo campo è presente anche nei frame di controllo, ed è grazie a esso che le altre stazioni riescono a gestire il meccanismo NAV. L'intestazione del frame contiene quattro indirizzi, tutti in formato IEEE 802 standard. Naturalmente sono necessari gli indirizzi che identificano la sorgente e la destinazione, ma che dire degli altri due? Poiché i frame possono entrare e uscire dalla cella attraverso una stazione base, gli altri due indirizzi identificano le stazioni base di partenza e di arrivo del traffico tra celle.

I campo *sequenza* permette di numerare i frammenti. Dei 16 bit disponibili, 12 identificano il frame e 4 il frammento; il campo *dati* contiene il carico utile, fino a 2.312 Byte, seguito dal solito *checksum*.

I frame di gestione hanno un formato simile a quello dei frame dati, ma non dispongono di un indirizzo di stazione base perché sono limitati a una singola cella. I frame di controllo sono ancora più corti, in quanto hanno solo uno o due indirizzi e non hanno né il

campo *dati* né il campo *sequenza*. L'informazione chiave, in questo caso, si trova nel campo *subtype* e di solito è RTS, CTS o ACK.

4.4.5 Servizi

Lo standard 802.11 definisce nove servizi che ogni LAN wireless conforme deve fornire. Sono divisi in due categorie: cinque servizi di distribuzione e quattro servizi stazione. I servizi di distribuzione riguardano la gestione dell'appartenenza alla cella e l'interazione con le stazioni poste al di fuori della cella; i servizi stazione si occupano invece dell'attività dentro una singola cella.

I cinque servizi di distribuzione sono forniti dalle stazioni base e si occupano della mobilità delle stazioni quando entrano ed escono dalle celle, perciò gestiscono connessione e disconnessione dalla stazione base; e sono qui di seguito elencati.

- Associazione.** Questo servizio è utilizzato dalle stazioni mobili per effettuare la connessione alle stazioni base. In genere è usato appena una stazione si sposta dentro il campo d'azione di una stazione base. Al suo arrivo la stazione annuncia la propria identità e le funzionalità, che comprendono le velocità supportate, l'eventuale necessità di servizi PCF (per esempio quello di polling) e i requisiti relativi alla gestione dell'alimentazione. La stazione base può accettare o rifiutare la stazione mobile. Se viene accettata, la stazione mobile deve autenticarsi.
- Separazione.** La stazione mobile o quella base possono separarsi interrompendo la relazione. Una stazione mobile dovrebbe utilizzare questo servizio prima di spegnersi o di lasciare la cella; la stazione base può utilizzarlo anche prima di disattivarsi per una manutenzione.
- Riassociazione.** Usando questo servizio, una stazione mobile può cambiare la propria stazione base preferita. Questa funzionalità è utile alle stazioni mobili che si spostano da una cella all'altra. Se utilizzata correttamente, nessun dato andrà perso a causa del passaggio (ma 802.11, come Ethernet, garantisce soltanto un servizio best effort).
- Distribuzione.** Questo servizio stabilisce come saranno instradati i frame verso la stazione base. Se la destinazione è locale alla stazione base i frame possono essere inviati direttamente via radio, altrimenti dovranno essere instradati attraverso la rete cablata.
- Integrazione.** Se un frame deve essere inviato attraverso una rete non 802.11 che adotta un diverso schema di indirizzamento o un diverso formato di frame, questo servizio gestisce la traduzione dal formato 802.11 al formato richiesto dalla rete di destinazione.

Gli altri quattro servizi si occupano delle operazioni interne alle celle e sono utilizzati solo dopo aver completato le associazioni.

1. **Autenticazione.** Poiché la comunicazione wireless può facilmente essere inviata o ricevuta da stazioni non autorizzate, solo le stazioni che si autenticano ricevono l'autorizzazione a trasmettere dati. Dopo aver associato la stazione mobile (ossia dopo averla accettata nella propria cella), la stazione base le invia uno speciale frame di challenge per vedere se la stazione mobile conosce il codice segreto (password) che le è stato assegnato. Per dimostrare di essere a conoscenza del codice segreto, la stazione mobile cifra il frame di challenge e lo rimanda alla stazione base. Se il risultato è corretto, la stazione mobile è registrata definitivamente nella cella. Nello standard iniziale la stazione base non doveva dimostrare la propria identità alla stazione mobile, ma questo difetto dello standard è oggi in fase di correzione.
2. **Invalidamento.** Quando desidera lasciare la rete, una stazione precedentemente autenticata viene invalidata. Dopo l'invalidamento la stazione non può più utilizzare la rete.
3. **Riservatezza.** Le informazioni trasmesse attraverso la LAN wireless devono essere cifrate per garantire la riservatezza, e questo servizio gestisce la codifica e la decodifica dei dati. L'algoritmo di codifica specificato è RC4, inventato da Ronald Rivest del M.I.T.
4. **Trasferimento dati.** In definitiva quello che conta è la trasmissione dati, e naturalmente 802.11 fornisce un modo per trasmettere e ricevere i dati. Poiché 802.11 è modellato su Ethernet e la trasmissione attraverso Ethernet non è affidabile al 100%, anche la trasmissione attraverso 802.11 non è considerata completamente affidabile. Gli strati superiori devono gestire il rilevamento e la correzione degli errori.

Una cella 802.11 ha alcuni parametri che possono essere esaminati e, in alcuni casi, anche regolati. Riguardano la codifica, gli intervalli di timeout, le velocità dati, la frequenza del frame di segnalazione e così via.

Le LAN wireless basate su 802.11 iniziano a essere installate negli edifici per uffici, negli aeroporti, negli alberghi, nei ristoranti e nelle università di tutto il mondo: ci si aspetta una rapida crescita. Per saperne di più sull'installazione di 802.11 su larga scala al CMU consultare (Hills, 2001).

4.5 Wireless a banda larga

Siamo rimasti al chiuso per troppo tempo; è giunto il momento di uscire fuori e vedere se sta accadendo qualcosa di interessante nel settore delle reti. Su questo fronte l'attività è frenetica, e alcuni dei cambiamenti riguardano il cosiddetto "ultimo miglio". Dopo la deregolamentazione del sistema telefonico avvenuta in molti paesi, i concorrenti delle aziende telefoniche consolidate ora hanno la possibilità di offrire servizi di telefonia locale e di accesso a Internet ad alta velocità.

La domanda è sicuramente abbondante, ma il problema è che stendere fibre, cavi coassiali o anche doppini di categoria 5 per milioni di abitazioni e uffici ha un costo proibitivo. Che cosa può fare un concorrente, a questo punto?

La risposta è semplice: offrire servizi wireless a banda larga. Innalzare una grande antenna su una collina all'esterno della città e installare sui tetti dei clienti piccole antenne che puntano verso di essa è molto più semplice ed economico che scavare buche e tirare cavi. Per questo motivo le aziende che competono nel settore delle telecomunicazioni hanno un grande interesse nel fornire un servizio di comunicazione wireless multimegabit in grado di trasmettere voce, Internet, film e così via. Come si può notare osservando la Figura 4.30, LMDS è stato inventato proprio per soddisfare questa esigenza, ma fino a poco tempo fa ogni azienda adottava il proprio sistema. Questa carenza di standard non incoraggiava la produzione su larga scala di hardware e software, di conseguenza i prezzi rimanevano alti e il gradimento basso.

Nell'industria del settore molti si erano accorti che l'elemento chiave mancante era uno standard wireless a banda larga, perciò venne chiesto a IEEE di costituire un comitato per definire lo standard, composto da persone provenienti dalle aziende più importanti e dall'ambiente universitario. Il primo numero disponibile era **802.16**, così lo standard ricevette questo identificativo. Il lavoro iniziato nel mese di luglio 1999 venne concluso con l'approvazione finale nel mese di aprile 2002. Il nome ufficiale dello standard è "Air Interface for Fixed Broadband Wireless Access Systems", però alcuni preferiscono chiamarlo **wireless MAN (Metropolitan Area Network)** o **collegamento locale wireless**; useremo questi termini come sinonimi.

802.16 è stato pesantemente influenzato dal modello OSI, come già altri standard 802 che lo ricalcano nella definizione degli strati, nella terminologia, nelle primitive di servizio e così via. Sfortunatamente, proprio come OSI, è piuttosto complicato. I paragrafi successivi forniscono una breve descrizione di alcune delle parti migliori di 802.16, ma questa trattazione, che non deve essere considerata in alcun modo completa, tralascia parecchi dettagli. Per maggiori informazioni sul wireless a banda larga in generale, consultare (Bolcskei et al., 2001; Webb, 2001). Per informazioni sullo standard 802.16 in particolare, consultare (Eklund et al., 2002).

4.5.1 Confronto tra 802.11 e 802.16

A questo punto il lettore potrebbe chiedersi che senso aveva concepire un nuovo standard. Non si poteva semplicemente utilizzare 802.11? Esistono molte buone ragioni per non utilizzare 802.11, prima di tutto perché 802.11 e 802.16 risolvono problemi diversi. Prima di analizzare gli aspetti tecnologici di 802.16, può essere utile spendere alcune parole sul perché fu necessario creare un nuovo standard.

Gli ambienti in cui operano 802.11 e 802.16 per certi punti di vista si assomigliano, in particolare perché entrambi gli standard sono stati progettati per fornire comunicazioni wireless a banda larga, ma ci sono importanti differenze. Prima di tutto, 802.16 fornisce un servizio agli edifici: e gli edifici non sono mobili, non migrano spesso da cella a cella. Gran parte di 802.11 si occupa della mobilità, e si tratta di un aspetto che non è importante in questa applicazione. Inoltre, gli edifici possono contenere più di un computer, una com-

plicazione che non si presenta quando la stazione finale è un singolo computer portatile. Poiché i proprietari degli edifici generalmente possono investire nella comunicazione più soldi dei proprietari di computer portatili, si possono usare radio migliori. Questa differenza significa che 802.16 può utilizzare la comunicazione full duplex, caratteristica che 802.11 non supporta per tenere bassi i prezzi dei radioricevitori.

Poiché 802.16 si estende su parte di una città le distanze possono arrivare a diversi chilometri, quindi la potenza percepita dalla stazione base può variare enormemente da stazione a stazione. Questa variazione influenza il rapporto segnale/rumore, e di conseguenza impone la necessità di supportare più schemi di modulazione. Inoltre, un sistema di comunicazione aperto su tutta la città rende indispensabili e obbligatori sia la protezione sia la riservatezza.

Per di più ogni cella probabilmente avrà molti più utenti di una tipica cella 802.11 e questi utenti utilizzeranno più banda di quella occupata da un classico utente 802.11. Dopo tutto, è raro che un'azienda inviti 50 impiegati a raccogliersi in una stanza insieme ai loro computer portatili, per vedere se la proiezione contemporanea di 50 diversi film può saturare la rete wireless 802.11. Quindi, poiché è richiesto più spettro di quello che le bande ISM possono offrire, 802.16 è costretto a operare nell'intervallo di frequenze molto più grande compreso tra i 10 e i 66 GHz, l'unica area dello spettro dove ancora resta spazio inutilizzato.

Ma queste onde millimetriche, avendo proprietà fisiche diverse da quelle delle onde più lunghe delle bande ISM, richiedono uno strato fisico completamente differente. Le onde millimetriche, per esempio, vengono assorbite facilmente dall'acqua: specialmente dalla pioggia, e in parte anche da neve, grandine e (con un po' di sfortuna), anche dalla nebbia. Di conseguenza, la gestione degli errori è più importante rispetto all'ambiente interno. Le onde millimetriche possono essere concentrate in raggi direzionali (802.11 è omnidirezionale), perciò le scelte fatte in 802.11 considerando la propagazione multipercorso si possono ridiscutere.

Un altro problema è la qualità del servizio. Benché offra un po' di supporto al traffico in tempo reale (nella modalità PCF), 802.11 in realtà non è stato progettato per la telefonia e l'utilizzo pesante delle applicazioni multimediali. Al contrario, 802.16 deve supportare queste applicazioni in modo completo, poiché è dedicato sia alle aziende sia agli utenti domestici.

In breve, 802.11 è stato progettato per le connessioni Ethernet mobili; 802.16 invece è stato progettato per la comunicazione wireless fissa delle televisioni via cavo. Queste differenze sono talmente grandi che gli standard risultanti hanno veramente poco in comune, in quanto tentano di ottimizzare aspetti differenti della comunicazione.

Può essere utile anche un breve confronto con il sistema telefonico cellulare. Nel caso dei telefoni cellulari si utilizzano stazioni mobili a bassa potenza e banda stretta, ottimizzate per la voce, che comunicano usando microonde di media lunghezza. Nessuno (per il momento) guarda film di due ore in alta risoluzione sul telefono cellulare GSM, e anche UMTS ha poca speranza di cambiare questa situazione. In definitiva, il mondo delle MAN wireless è molto più esigente di quello della telefonia mobile, perciò è necessario un sistema completamente diverso. 802.16 verrà utilizzato in futuro per i dispositivi mobili? La domanda è interessante: anche se non è stato ottimizzato per questi apparecchi, la possibilità esiste. Per il momento lo standard si concentra sulle comunicazioni wireless fisse.

4.5.2 La pila di protocolli 802.16

La Figura 4.31 mostra la pila di protocolli 802.16. La struttura generale è simile a quella delle altre reti 802, ma ha un numero maggiore di sottostrati. Il sottostrato più basso si occupa della trasmissione. Si usa la tradizionale comunicazione radio a banda stretta, con i soliti schemi di modulazione. Sopra lo strato fisico si trova un sottostrato di convergenza che nasconde le differenze tecnologiche allo strato data link. In realtà anche 802.11 ha qualcosa di simile, ma il comitato ha deciso di non formalizzarlo con un nome di tipo OSI.

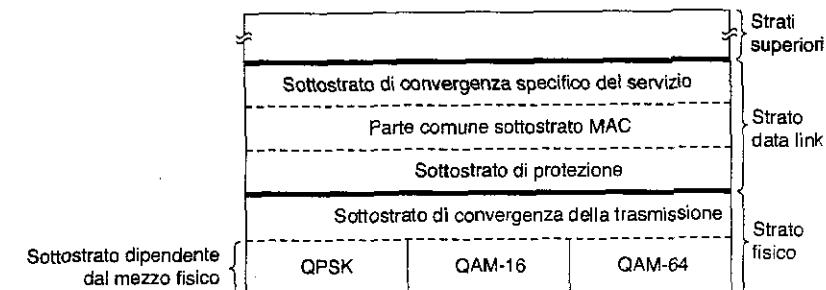


Figura 4.31. La pila di protocolli 802.16.

Nella figura non sono visibili, ma sono già in fase di sviluppo due nuovi protocolli di strato fisico. Lo standard 802.16a supporterà OFDM nell'intervallo di frequenze compreso tra i 2 e gli 11 GHz, e lo standard 802.16b opererà nella banda ISM dei 5 GHz. Questi standard rappresentano un tentativo di avvicinamento a 802.11.

Lo strato data link è composto da tre sottostrati. Quello più basso si occupa della riservatezza e della protezione, aspetti più cruciali nelle reti pubbliche all'aperto che nelle reti private al chiuso; gestisce la cifratura, la decifratura e la gestione delle chiavi.

Ora viene la parte comune del sottostrato MAC: qui si trovano i protocolli essenziali, per esempio quello per la gestione del canale. Nel modello è la stazione base che controlla il sistema; può stabilire i canali di ricezione (dalla base all'abbonato) in modo molto efficiente e gioca un ruolo fondamentale anche nella gestione dei canali di trasmissione (dall'abbonato alla base). Una funzionalità insolita del sottostrato MAC è che, a differenza delle altre reti 802, è stato completamente orientato verso le connessioni per fornire le garanzie di qualità di servizio richieste dalla comunicazione telefonica e multimediale.

Il sottostrato di convergenza specifico del servizio prende il posto del sottostrato di collegamento logico adottato dagli altri protocolli 802. Il suo scopo è interfacciarsi con lo strato di rete. Una complicazione nasce dal fatto che 802.16 è stato progettato per integrarsi in modo nativo sia con i protocolli datagram (per esempio PPP, IP ed Ethernet) sia con ATM. Il problema è che i protocolli a pacchetto non sono orientati verso le connessioni, al contrario di ATM, perciò ogni connessione ATM deve essere mappata su una connessione 802.16, in linea di massima in maniera semplice. Ma a quale connessione 802.16 va associato un pacchetto IP in arrivo? Il problema è gestito da questo sottostrato.

4.5.3 Lo strato fisico di 802.16

Abbiamo detto che il wireless a banda larga ha bisogno di molto spettro, e le uniche zone libere sono quelle comprese tra i 10 e i 66 GHz. Queste onde millimetriche hanno una proprietà interessante, che le microonde più lunghe non hanno: viaggiano in linea retta, quindi si muovono come la luce, non come il suono. Di conseguenza la stazione base può avere più antenne, ognuna che punta verso un settore diverso del terreno circostante (Figura 4.32). Ogni settore ha i suoi utenti ed è relativamente indipendente da quelli adiacenti, cosa che invece non accade nelle stazioni radio cellulari omnidirezionali.

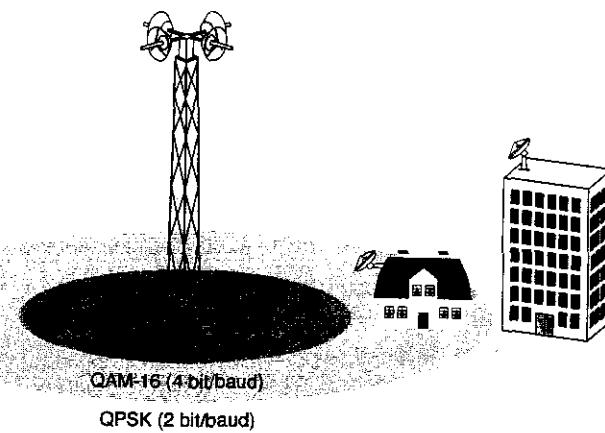


Figura 4.32. L'ambiente di trasmissione di 802.16.

Poiché l'intensità del segnale nella banda millimetrica diminuisce bruscamente con la distanza dalla stazione base, anche il rapporto segnale/rumore diminuisce con la distanza alla stazione base. Per questo motivo 802.16 adotta tre diversi schemi di modulazione, e tengono conto della distanza dell'abbonato dalla stazione base. Con gli abbonati più vicini è utilizzato QAM-64, con 6 bit per baud; con quelli a media distanza si usa QAM-16, con 4 bit per baud; infine, con quelli più lontani è utilizzato QPSK, con 2 bit per baud. Ad esempio, nel caso di uno spettro tipico di 25 MHz, QAM-64 arriva a 150 Mbps, QAM-16 arriva a 100 Mbps e QPSK arriva a 50 Mbps. In altre parole, più l'abbonato è lontano alla stazione base, più lenta risulta la velocità (proprio come accadeva con il sistema DSL descritto nella Figura 2.27). I diagrammi di costellazione delle tre tecniche di modulazione sono mostrati nella Figura 2.25.

Per raggiungere l'obiettivo di produrre un sistema a banda larga, considerati i limiti fisici appena menzionati, i progettisti di 802.16 hanno lavorato duramente per utilizzare in modo efficiente lo spettro disponibile. Una cosa che queste persone non apprezzavano era la modalità di funzionamento di GSM e DAMPS: entrambi i sistemi utilizzano bande di frequenze diverse ma di pari ampiezza per il traffico trasmesso e ricevuto. Nel caso della voce il

traffico è prevalentemente simmetrico, però con Internet di solito i dati scaricati sono più di quelli trasmessi. Di conseguenza, 802.16 permette di assegnare la banda in modo più flessibile. Si utilizzano due schemi: **FDD** (*Frequency Division Duplexing*) e **TDD** (*Time Division Duplexing*). Il secondo è mostrato nella Figura 4.33. La figura mostra la stazione base che trasmette periodicamente i frame. Ogni frame contiene intervalli temporali (*slot*). I primi sono dedicati al traffico in ricezione, segue un periodo di guardia utilizzato dalle stazioni per cambiare direzione, infine arrivano gli intervalli dedicati al traffico in uscita. Il numero degli intervalli temporali dedicati a ogni direzione può essere cambiato dinamicamente, per adattare al traffico la banda in ciascuna direzione.

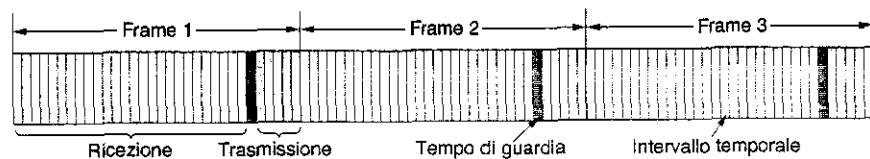


Figura 4.33. Frame e intervalli temporali nella tecnica di duplexing a divisione di tempo.

La stazione base associa il traffico in arrivo agli intervalli temporali, controllando completamente la trasmissione in questa direzione. Il traffico in uscita è più complesso e dipende dalla qualità richiesta del servizio. L'assegnazione degli intervalli sarà trattata durante la descrizione del sottostato MAC.

Un'altra interessante funzionalità dello strato fisico è la sua capacità d'impacchettare diversi frame MAC uno dietro l'altro in una singola trasmissione fisica. Questa funzionalità migliora l'efficienza dello spettro, poiché riduce il numero di preamboli e intestazioni di strato fisico.

È utile evidenziare l'uso dei codici di Hamming nell'operazione di correzione degli errori eseguita nello strato fisico. Quasi tutte le altre reti si affidano semplicemente al checksum per rilevare gli errori e chiedere una ritrasmissione quando i frame ricevuti sono danneggiati. Ma nell'ambiente geografico a banda larga ci si aspettano talmente tanti errori di trasmissione che, oltre alla verifica del checksum gestita dagli strati superiori, si esegue anche una correzione degli errori nello strato fisico. In definitiva la presenza della correzione degli errori fa apparire il canale migliore di quanto sia nella realtà, proprio come il CD-ROM sembra molto affidabile solo perché più della metà dei bit sono utilizzati per correggere gli errori nello strato fisico.

4.5.4 Il protocollo del sottostato MAC di 802.16

Lo strato data link è diviso in tre sottostati (Figura 4.31). Poiché la crittografia sarà studiata solo nel Capitolo 8, è difficile spiegare adesso in che modo funziona il sottostato di protezione. Basti sapere che la cifratura serve a tenere segreti tutti i dati trasmessi. È cifrato solo il carico utile del frame, ma non l'intestazione. Questo significa che un ficcanaso può scoprire chi sta parlando e con chi, ma non può sapere che cosa i due si stanno dicendo.

Per quelli che sanno già qualcosa di crittografia, il paragrafo seguente descrive brevemente il sottostrato di protezione. Chi non sa nulla di crittografia dei dati probabilmente non troverà il paragrafo molto istruttivo, ma potrà rileggerlo dopo aver studiato il Capitolo 8. Quando un abbonato si collega a una stazione base, le due entità eseguono l'autenticazione reciproca con crittografia basata su chiave pubblica RSA e certificati X.509. I carichi utili sono codificati utilizzando il sistema della chiave simmetrica, mediante DES con sistema CBC (*Cipher Block Chaining*) o triplo DES con due chiavi. AES (Rijndael) verrà probabilmente aggiunto presto. Il controllo d'integrità utilizza SHA-1. Non è stato così brutto, vero?

Veniamo ora alla parte comune del sottostrato MAC. I frame MAC occupano un numero intero d'intervalli temporali dello strato fisico. Ogni frame è composto da sottoframe, dove i primi due sono le mappe di ricezione e trasmissione. Queste mappe indicano che cosa c'è in ogni intervallo e quali intervalli sono liberi. La mappa di ricezione contiene anche vari parametri di sistema che servono alle nuove stazioni appena entrate in linea.

Il canale di ricezione è abbastanza semplice; la stazione base decide solo che cosa mettere in ogni sottoframe. Il canale di trasmissione è più complicato, poiché ci sono abbonati non coordinati che competono tra loro per accedere al canale; la sua assegnazione è strettamente legata alle problematiche della qualità di servizio. Il sistema definisce quattro classi di servizio:

1. servizio a bit-rate costante
2. servizio a bit-rate variabile in tempo reale
3. servizio a bit-rate variabile non in tempo reale
4. servizio best effort.

Tutti i servizi in 802.16 sono orientati verso le connessioni, e ogni connessione riceve una delle precedenti classi di servizio durante l'impostazione della connessione. Questa architettura è molto diversa da quella di 802.11 o di Ethernet, che non hanno connessioni nel sottostrato MAC.

I servizi a bit-rate costante sono stati progettati per trasmettere voce non compressa, come succede su un canale T1. Questo servizio ha bisogno d'inviare una quantità di dati predefinita a intervalli di tempo predeterminati, ed è fornito dedicando certi intervalli temporali a ogni connessione di questo tipo. Una volta allocata la banda, gli intervalli temporali sono automaticamente disponibili senza che sia necessario richiederli.

I servizi a bit-rate variabile in tempo reale sono utilizzati per i dati multimediali compressi e altre applicazioni in tempo reale, dove la quantità di banda richiesta può variare in ogni istante. La stazione base interroga l'abbonato a intervalli fissi per sapere quanta banda gli serve in quel momento.

I servizi a bit-rate variabile non in tempo reale sono utilizzati per trasmissioni pesanti che avvengono in tempo reale, per esempio per trasferire file di grandi dimensioni. Per

questo servizio la stazione base interroga spesso l'abbonato, ma non a intervalli imposti rigidamente. Un abbonato a velocità di bit costante può impostare un bit in uno dei suoi frame per richiedere un'interrogazione, e quindi inviare una maggiore quantità di traffico a velocità di bit variabile.

Una stazione che non risponde a un'interrogazione k volte di seguito è inserita dalla stazione base in un gruppo multicast, sopprimendo la corrispondente interrogazione individuale. Quando il gruppo multicast viene interrogato, qualunque stazione che vi appartiene può rispondere concorrendo per il servizio. In tal modo, le stazioni con poco traffico non sprecano interrogazioni preziose.

Il servizio best effort è utilizzato per tutto il resto del traffico. Non viene eseguita alcuna interrogazione e l'abbonato deve concorrere per la banda insieme agli altri abbonati best effort. Le richieste per la banda sono fatte in intervalli temporali contrassegnati come disponibili per la contesa nella mappa di trasmissione. Se una richiesta ha successo ciò sarà annotato nella successiva mappa di ricezione; in caso d'insuccesso dovrà essere tentata in seguito dall'abbonato. Per limitare il numero di collisioni si usa l'algoritmo di backoff esponenziale binario.

Lo standard definisce due forme di assegnazione di banda: per stazione e per connessione. Nel primo caso la stazione dell'abbonato aggrega le necessità di tutti gli utenti presenti nell'edificio e fa una richiesta collettiva. Quando le viene garantita la banda, la stazione la distribuisce con parsimonia ai suoi utenti come ritiene giusto. Nel secondo caso, la stazione base gestisce direttamente ogni connessione.

4.5.5 La struttura del frame 802.16

Tutti i frame MAC iniziano con un'intestazione generica. L'intestazione è seguita da un carico utile e da un checksum (CRC) facoltativo, come mostrato nella Figura 4.34. Il carico utile non serve nei frame di controllo che, per esempio, richiedono intervalli di canale. Anche il checksum (sorprendentemente) è opzionale: questo perché viene fatta una correzione degli errori nello strato fisico, e poiché non viene mai fatto alcun tentativo di ritrasmettere i frame in tempo reale. Se non verrà tentata alcuna ritrasmissione, che senso ha perdere tempo con il checksum?

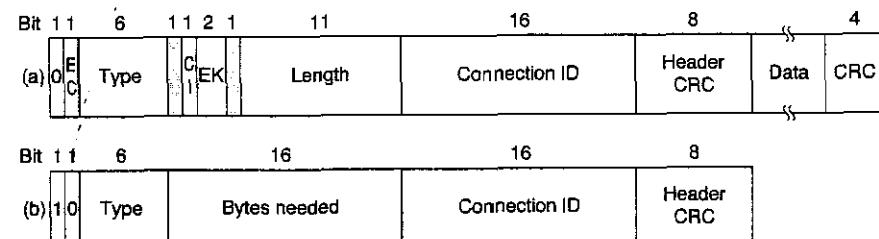


Figura 4.34. (a) Un frame generico. (b) Un frame di richiesta di banda.

segue una rapida descrizione dei campi presenti nell'intestazione del frame rappresentato nella Figura 4.34(a). Il bit *EC* indica se il carico utile è stato cifrato. Il campo *type* identifica il tipo di frame e segnala se sono stati utilizzati meccanismi d'impacchettamento e frammentazione. Il campo *C/I* indica la presenza o l'assenza del codice di controllo finale. Il campo *EK* indica le chiavi di codifica utilizzate. Il campo *length* (lunghezza) riporta la lunghezza totale del frame, inclusa l'intestazione. *Connection ID* definisce la connessione i appartenenza del frame. Infine, il campo *CRC header* è un checksum dedicato solo all'intestazione, basato sul polinomio $x^8 + x^2 + x + 1$.

Figura 4.34(b) mostra un secondo tipo d'intestazione utilizzato dai frame che richiedono banda. Inizia con un bit 1 invece che con un bit 0 ed è simile all'intestazione generica; il secondo e il terzo byte, però, formano un numero di 16 bit che indica la banda chiesta per trasportare il numero specificato di byte. I frame di richiesta di banda non trasportano carico utile.

I sarebbero ancora molto da dire su 802.16, ma questo non è il luogo adatto. Per maggiori informazioni, consultare lo standard.

6 Bluetooth

Nel 1994, l'azienda L.M. Ericsson cominciò a interessarsi alla connessione senza cavo dei suoi telefoni cellulari con altri dispositivi (per esempio i computer palmari). Insieme ad altre quattro aziende (IBM, Intel, Nokia e Toshiba), formò un SIG (*Special Interest Group*, ossia un consorzio) per sviluppare uno standard wireless che avrebbe dovuto permettere il collegamento tra dispositivi di calcolo, di comunicazione e accessori vari mediante un sistema radio wireless a basso costo, a bassa potenza e portata ridotta. Il progetto venne chiamato **Bluetooth**, in onore di Harald Blaatand (Bluetooth) II (940-981), il re vichingo che unificò (cioè sottomise) la Danimarca e la Norvegia, anche in questo caso senza cavi. Ebbene l'idea originale fosse solo quella di liberarsi dei cavi tra dispositivi, il suo ambito ben presto si espanso fino a invadere il settore delle LAN wireless. Questo spostamento rende lo standard più utile, ma crea concorrenza con 802.11. A peggiorare le cose, i due sistemi interferiscono elettricamente l'uno con l'altro. Vale la pena notare che Hewlett-Packard alcuni anni fa ha introdotto un sistema di comunicazione a infrarossi per collegare le periferiche ai computer senza utilizzare cavi, ma il sistema non ha mai riscosso successo.

Prezzante del pericolo, nel mese di luglio 1999 SIG Bluetooth pubblicò le 1.500 pagine specifiche della versione 1.0. Poco dopo, il gruppo di standard IEEE 802.15, che si occupava delle reti personal area network wireless, adottò il documento Bluetooth come punto di partenza e iniziò a lavorare su di esso. Anche se può sembrare strano standardizzare qualcosa che ha già una specifica molto dettagliata e che non ha alcuna implementazione incompatibile da armonizzare, la storia dimostra che la disponibilità di uno standard sotto gestito da un ente neutrale quale IEEE spesso favorisce l'utilizzo di una tecnologia. E se essere un po' più precisi, sarebbe meglio notare che le specifiche Bluetooth descrivono un sistema completo, che parte dallo strato fisico e arriva allo strato applicazione. Il

comitato IEEE 802.15 sta standardizzando solo lo strato fisico e lo strato data link; il resto della pila di protocolli non è contemplato.

Anche se IEEE nel 2002 ha approvato il primo standard PAN (802.15.1), SIG Bluetooth è ancora impegnato attivamente nello sviluppo di migliorie. Le versioni di IEEE e di SIG Bluetooth non sono identiche, ma si spera che presto convergano in un singolo standard.

4.6.1 Architettura Bluetooth

La descrizione del sistema Bluetooth inizia con una rapida panoramica di contenuto e funzionalità. L'unità base di un sistema Bluetooth è la **piconet**, composta da un nodo master e da diversi (ma non più di sette) nodi slave attivi situati entro un raggio di 10 metri. Più piconet possono trovarsi nella stessa stanza (molto grande) e possono anche essere collegate attraverso un nodo ponte come mostrato nella Figura 4.35. Un insieme di piconet interconnesse è chiamato **scatternet**.

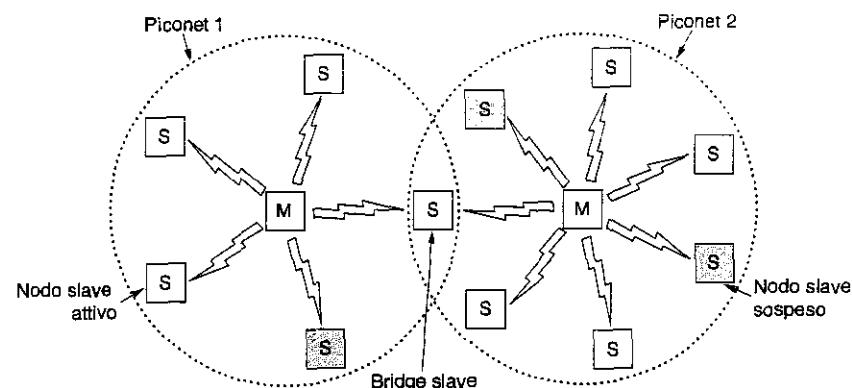


Figura 4.35. Due piconet possono essere collegate per formare una scatternet.

Oltre ai sette nodi slave attivi presenti in una piconet, la rete può contenere fino a 255 nodi sospesi, che sono dispositivi a cui il nodo master ha imposto di attivare uno stato di bassa alimentazione per ridurre il consumo delle batterie. Quando si trova in stato sospeso, un dispositivo può solo rispondere a una richiesta di attivazione o a un segnale trasmesso dal nodo master. Esistono anche altri due stati di alimentazione intermedi, chiamati hold e sniff, che però al momento non ci interessano.

La scelta dei progettisti di un'architettura basata su nodi master e slave facilita l'implementazione di chip Bluetooth completi dal costo inferiore a 5 €. La conseguenza di questa decisione è che i nodi slave sono completamente muti, sostanzialmente fanno solo ciò che il nodo master dice loro di fare. Il cuore della piconet è costituito da un sistema TDM centralizzato: il nodo master controlla il clock e decide quale dispositivo può comunicare in ogni intervallo temporale. Tutta la comunicazione avviene tra il nodo master e un nodo slave; non è ammessa alcuna comunicazione diretta tra nodi slave.

4.6.2 Applicazioni Bluetooth

a maggior parte dei protocolli di rete fornisce i canali tra le entità comunicanti e lascia gli sviluppatori il compito d'immaginare le applicazioni che potranno essere implementate su quei canali. Per esempio, 802.11 non specifica il tipo di attività che gli utenti dovrebbero svolgere usando loro computer portatili (leggere la posta elettronica, esplorare il Web o fare altro); al contrario, la specifica Bluetooth v1.1 nomina 13 applicazioni specifiche da supportare e fornisce a ogni applicazione una diversa pila di protocolli. Fortunatamente questo approccio genera un'enorme complessità, che in questo paragrafo verrà tralasciata. Le 13 applicazioni, chiamate **profili**, sono elencate nella Figura 4.36. Un breve esame di questi profili aiuta a comprendere l'obiettivo che SIG Bluetooth sta cercando di raggiungere.

Nome	Descrizione
Accesso generico	Procedure per la gestione del collegamento
Scoperta del servizio	Protocollo per scoprire i servizi offerti
Porta seriale	Sostituzione del cavo seriale
Scambio di oggetti generico	Definisce la relazione client/server per lo spostamento degli oggetti
Accesso LAN	Protocollo tra un computer portatile e una LAN fissa
Accesso dial-up	Premette a un computer portatile di chiamare attraverso un telefono mobile
Fax	Permette a un apparecchio fax mobile di comunicare con un telefono mobile
Telefonia cordless	Collega un telefono alla sua base
Intercomunicanti	Ricetrasmittenti digitali
Auricolare wireless	Comunicazione vocale senza mani
Invio oggetto	Permette di scambiare semplici oggetti
Trasferimento file	Gestisce il trasferimento di file
Sincronizzazione	Permette a un PDA di sincronizzarsi con un altro computer

Figura 4.36. I profili Bluetooth.

profilo accesso generico non è una vera applicazione, ma piuttosto la base su cui sono struite le applicazioni vere e proprie. Il suo compito principale è fornire un sistema per abilitare e mantenere collegamenti (canali) protetti tra il nodo master e i nodi slave. Anche il profilo scoperta del servizio, utilizzato dai dispositivi per scoprire quali sono i servizi offerti dagli altri apparecchi, è relativamente generico. Tutti i dispositivi Bluetooth devono implementare questi due profili; gli altri sono opzionali.

Il profilo porta seriale è un protocollo di trasporto usato dalla maggior parte degli altri profili. Emula una linea seriale ed è utile soprattutto con le vecchie applicazioni che hanno bisogno di una linea seriale.

Il profilo scambio di oggetti generico definisce una relazione client/server per lo scambio di dati. I client avviano le operazioni, ma un nodo slave può fungere sia da client sia da server. Come il profilo porta seriale, anche questo funge da blocco di costruzione per altri profili. Il gruppo composto dai successivi tre profili è dedicato alla comunicazione di rete. Il pro-

filo accesso LAN permette al dispositivo Bluetooth di collegarsi a una rete fissa; questo profilo è un concorrente diretto di 802.11. Il profilo accesso dial-up costituisce il motivo originale dell'intero progetto; permette a un computer portatile di collegarsi senza utilizzare alcun cavo a un telefono cellulare contenente un modem integrato. Il profilo fax assomiglia al profilo dial-up, ma permette agli apparecchi fax wireless di inviare dati e ricevere fax usando telefoni mobili non collegati via cavo.

I tre profili successivi sono dedicati alla telefonìa. Il profilo telefonìa cordless serve a collegare il microtelefono di un apparecchio cordless alla stazione base. Oggi la maggior parte dei telefoni cordless non può essere utilizzata anche come apparecchio cellulare, ma in futuro i dispositivi cordless e quelli cellulari potrebbero convergere. Il profilo intercomunicanti permette a due telefoni di collegarsi come ricetrasmittenti (walkie-talkie). Infine, il profilo auricolare wireless è quello che permette di collegare l'auricolare alla sua stazione base; questa funzione consente per esempio di telefonare mentre si guida un'automobile.

Gli ultimi tre profili sono dedicati allo scambio di oggetti tra due apparecchi wireless; gli oggetti potrebbero essere biglietti da visita elettronici, immagini o file di dati. In particolare il profilo di sincronizzazione è stato progettato per caricare i dati in un PDA o in un computer portatile quando il dispositivo esce da casa, e per raccogliere dati dal dispositivo al suo ritorno.

Era veramente necessario indicare in modo esplicito e dettagliato tutte queste applicazioni e fornire a ognuna una specifica pila di protocolli? Probabilmente no, ma le parti dello standard sono state concepite da gruppi di lavoro diversi e ogni gruppo, concentrandosi solo su un problema specifico, ha generato un proprio profilo. Questo risultato può essere considerato come una dimostrazione della legge di Conway: nel numero di aprile 1968 della rivista Datamation, Melvin Conway osservò che assegnando n persone alla scrittura di un compilatore si ottiene un compilatore a n passaggi o, più in generale, la struttura del software rispecchia la struttura del gruppo che lo ha prodotto. Probabilmente sarebbe stato possibile farla franca con due stack di protocollo invece che con 13, uno per il trasferimento dei file e l'altro per la gestione del flusso di comunicazione in tempo reale.

4.6.3 La pila di protocolli Bluetooth

Lo standard Bluetooth ha molti protocolli raggruppati non rigidamente in strati. La struttura degli strati non segue né il modello OSI, né il modello TCP/IP, né il modello 802 o qualsiasi altro modello conosciuto. IEEE, comunque, sta modificando Bluetooth per renderlo più simile al modello 802. L'architettura di protocollo di base di Bluetooth, modificata dal comitato 802, è mostrata nella Figura 4.37.

Lo strato posto più in basso è quello della trasmissione radio fisica, che corrisponde piuttosto bene allo strato fisico dei modelli OSI e 802. Questo strato si occupa della trasmissione radio e della modulazione. Molti aspetti dello strato dipendono dall'obiettivo di rendere il sistema economico e quindi interessante per il mercato.

Lo strato baseband è per certi versi analogo al sottostrato MAC, ma include anche elementi dello strato fisico; gestisce il modo in cui il nodo principale controlla gli intervalli temporali e il raggruppamento di questi intervalli in frame.

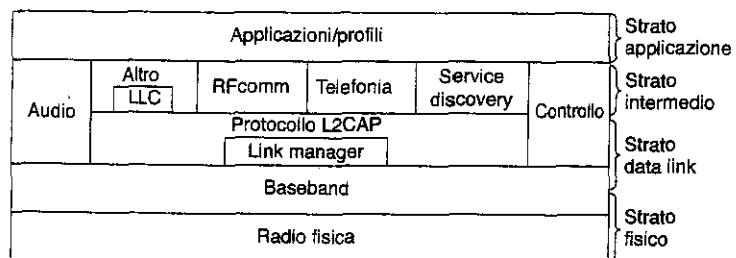


Figura 4.37. La versione 802.15 dell'architettura del protocollo Bluetooth.

in alto si trova uno strato contenente un gruppo di protocolli abbastanza collegati. Il link manager gestisce l'impostazione dei canali logici tra i dispositivi, e si occupa anche della gestione dell'alimentazione, dell'autenticazione e della qualità del servizio. Il protocollo L2CAP (*Logical Link Control Adaptation Protocol*) nasconde agli strati superiori i dettagli relativi alla trasmissione; è simile al sottostato LLC dello standard 802 ma tecnicamente è diverso. Come si può intuire dai nomi, i protocolli Audio e Controllo si occupano rispettivamente dei dati audio e delle operazioni di controllo; le applicazioni possono raggiungerli direttamente senza dover passare attraverso il protocollo L2CAP.

Il strato intermedio successivo contiene un insieme di protocolli. Qui IEEE ha inserito LLC di 802, per mantenere una compatibilità con le altre reti 802. I protocolli RFcomm, telefonía e service discovery sono nativi. RFcomm (*Radio Frequency communication*) è il protocollo che emula la porta seriale standard utilizzata per collegare i PC a diversi dispositivi, per esempio alla tastiera, al mouse e al modem. È stato progettato per facilitare la sua adozione nei dispositivi esistenti. Il protocollo telefonía è un protocollo in tempo reale utilizzato per i tre profili dedicati alla comunicazione vocale; gestisce anche l'impostazione della chiamata e la sua conclusione. Infine, il protocollo service discovery è utilizzato per individuare i servizi dentro la rete.

Il strato superiore contiene le applicazioni e i profili, che utilizzano i protocolli degli strati inferiori per svolgere il loro lavoro. Ogni applicazione ha un proprio sottoinsieme dedicato di protocolli. Dispositivi specifici, per esempio un auricolare senza fili, di solito conoscono solo i protocolli necessari alla specifica funzione e nessun altro.

I paragrafi seguenti esaminano i tre strati più bassi della pila di protocolli Bluetooth, che in questo modo corrispondono al sottostrato MAC e allo strato fisico.

6.4 Lo strato radio di Bluetooth

Lo strato radio sposta i bit dal nodo master al nodo slave e viceversa. È un sistema a bassa potenza che ha una portata di 10 metri e opera nella banda ISM dei 2,4 GHz. La banda è divisa in 79 canali larghi 1 MHz. La modulazione è di tipo FSK (*Frequency Shift Keying*) con 1 bit per Hz, che raggiunge più o meno la velocità di 1 Mbps, ma gran parte dello spettro è consumato dal checksum. Per assegnare i canali in modo imparziale si utilizza la tecnica a spettro distribuito a frequenza variabile con 1.600 cambi al secondo e un tempo di

rotazione di 625 msec. Tutti i nodi della piconet eseguono contemporaneamente il cambio di frequenza seguendo la sequenza scelta dal nodo master.

Poiché 802.11 e Bluetooth operano nella banda ISM a 2,4 GHz sugli stessi 79 canali, possono interferire l'uno con l'altro. Bluetooth esegue i salti di frequenza molto più velocemente di 802.11, perciò è più probabile che un dispositivo Bluetooth danneggi una trasmissione 802.11 che non il contrario. Poiché 802.11 e 802.15 sono entrambi standard IEEE, il comitato sta tentando di risolvere il problema, ma non è facile trovare una soluzione perché entrambi i sistemi utilizzano la banda ISM per lo stesso motivo: non richiedono alcuna licenza.

Lo standard 802.11a utilizza l'altra banda ISM (5 GHz), ma ha una portata molto più bassa di 802.11b (a causa della proprietà fisiche delle onde radio), perciò l'utilizzo di 802.11a non è una soluzione ottimale per tutti i casi. Alcune aziende hanno risolto il problema proibendo del tutto Bluetooth. Secondo una soluzione basata sul mercato, la rete con più potere (politico ed economico, non elettrico) può richiedere alla parte più debole di modificare il suo standard per eliminare l'interferenza. Alcuni pensieri su questo argomento sono contenuti in (Lansford et al., 2001).

4.6.5 Lo strato baseband di Bluetooth

Lo strato baseband di Bluetooth assomiglia molto a un sottostrato MAC, infatti trasforma il flusso di bit grezzi in frame e definisce alcuni formati chiave. Nella sua forma più semplice, il nodo master di ogni piconet definisce una serie d'intervalli temporali di 625 µsec e le trasmissioni del nodo master iniziano negli intervalli pari mentre le trasmissioni dei nodi slave iniziano in quelli dispari. Questo è un meccanismo multiplexing a divisione di tempo tradizionale, con il nodo master che usa la metà degli intervalli e i nodi slave che condividono l'altra metà. I frame possono occupare 1, 3 oppure 5 intervalli.

Le temporizzazioni del sistema frequency hopping, lasciano ai circuiti radio un tempo di assestamento per raggiungere la stabilità di 250 µsec per salto. È possibile ottenere un assestamento più rapido, ma solo a un prezzo più alto. Per un frame che copre un singolo intervallo temporale, dopo l'assestamento rimangono solo 366 dei 625 bit; di questi 126 sono utilizzati per un codice di accesso e l'intestazione, perciò alla fine i dati hanno a disposizione 240 bit. Quando si concatenano cinque intervalli è sufficiente un solo periodo di assestamento che ha durata più breve, perciò dei $5 \times 625 = 3.125$ bit presenti nei cinque intervalli, 2.781 rimangono a disposizione dello strato baseband. La conseguenza è che i frame lunghi sono molto più efficienti dei frame che occupano un singolo intervallo.

Ogni frame è trasmesso attraverso un canale logico, chiamato **link** o **collegamento**, stabilito tra il nodo master e un nodo slave. Esistono due tipi di link. Il primo, chiamato **ACL** (*Asynchronous ConnectionLess*), è utilizzato per i dati a commutazione di pacchetto disponibili a intervalli irregolari; questi dati provengono dallo strato L2CAP della parte trasmettente e sono trasmessi allo strato L2CAP della parte ricevente. Il traffico ACL è trasmesso in modo best effort, cioè non viene offerta alcuna garanzia di consegna: i frame possono perdere e aver bisogno di essere ritrasmessi. Un nodo slave può avere un solo collegamento ACL con il proprio nodo master.

L'altro collegamento si chiama **SCO** (*Synchronous Connection Oriented*) ed è utilizzato per i dati in tempo reale come le connessioni telefoniche. A questo tipo di canale è assegnato un intervallo fisso in ogni direzione. A causa della natura critica dei collegamenti SCO, i frame inviati attraverso questi canali non vengono mai ritrasmessi; piuttosto, si utilizza un meccanismo di correzione degli errori in avanti per migliorare l'affidabilità. Un nodo slave può avere tre collegamenti SCO con il proprio nodo master. Ogni collegamento SCO può trasmettere un canale audio PCM a 64.000 bps.

4.6.6 Lo strato L2CAP di Bluetooth

Lo strato L2CAP svolge tre funzioni principali. Primo, accetta pacchetti grandi fino a 64 KB provenienti dagli strati superiori e li divide in frame per la trasmissione. All'altro capo della linea, i frame sono riassemblati di nuovo in pacchetti. Secondo, gestisce il multiplexing e il demultiplexing da più sorgenti di pacchetti. Dopo aver riassemblato il pacchetto, lo strato L2CAP determina il protocollo di strato superiore che dovrà gestire i dati, per esempio il protocollo RFcomm o quello di telefonia. Terzo, L2CAP gestisce i requisiti di qualità del servizio durante l'impostazione dei collegamenti e durante le normali operazioni. Nella fase d'impostazione viene negoziata anche la dimensione massima del carico utile consentito, per impedire ai dispositivi che generano pacchetti molto grandi di soffocare i dispositivi in grado di gestire solo pacchetti di piccole dimensioni. Questa funzionalità è necessaria perché non tutti i dispositivi sono in grado di gestire il pacchetto di dimensione massima (64 KB).

4.6.7 La struttura del frame di Bluetooth

Esistono diversi formati di frame, e il più importante è mostrato nella Figura 4.38. Inizia con un codice di accesso che di solito identifica il nodo master; questa informazione consente ai nodi slave che si trovano nel raggio d'azione di due nodi master d'identificare il traffico di rispettiva competenza. Segue un'intestazione di 54 bit contenente i classici campi del sottostrato MAC. Poi c'è il campo dati, grande fino a 2.744 bit (per una trasmissione che occupa cinque intervalli). Nel caso di un intervallo singolo, il formato è identico ma il campo dati occupa solo 240 bit.

Per utilizzare uno sguardo all'intestazione. Il campo *indirizzo* identifica il destinatario del

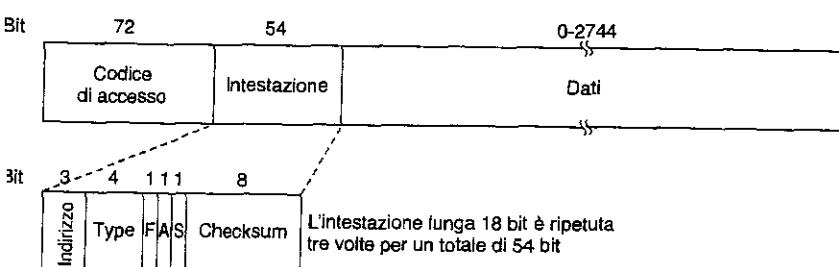


Figura 4.38. Un classico frame dati Bluetooth.

frame tra gli otto dispositivi attivi. Il campo *type* identifica il tipo di frame (ACL, SCO, interrogazione oppure nullo), il tipo di correzione degli errori utilizzato nel campo dati e il numero di intervalli occupati dal frame. Il bit *flow* è attivato da un nodo slave quando non può ricevere altri dati perché ha il buffer pieno. Questa è una forma primitiva di controllo di flusso. Il bit *acknowledgement* è utilizzato per aggiungere al frame un ACK. Il bit *sequence* è utilizzato per numerare i frame in modo da rilevare le ritrasmissioni. Il protocollo adotta un approccio stop-and-wait, perciò è sufficiente 1 bit. Seguono gli 8 bit utilizzati dal *checksum* dell'intestazione. L'intera intestazione lunga 18 bit è ripetuta tre volte per formare l'intestazione di 54 bit mostrata nella Figura 4.38. Dalla parte del ricevente, un semplice circuito esamina tutte e tre le copie di ogni bit; se le tre copie sono identiche il bit è accettato, altrimenti la maggioranza vince. Di conseguenza, per trasmettere 10 bit di intestazione servono 54 bit di capacità di trasmissione. Trasmettere dati in modo affidabile in un ambiente rumoroso usando dispositivi economici, che operano a bassa potenza (2,5 mW) e che sono dotati di una limitata capacità di elaborazione richiede un elevato livello di ridondanza.

Per il campo dati dei frame ACL si usano più formati, mentre i frame SCO sono più semplici: il campo dati occupa sempre 240 bit. Sono definite tre varianti che consentono 80, 160 o 240 bit di carico utile reale, il resto è usato per la correzione degli errori. Nella versione più affidabile (80 bit di carico utile), il contenuto è ripetuto tre volte, proprio come l'intestazione. Poiché può utilizzare solo gli intervalli dispari, il nodo slave ha a disposizione 800 intervalli al secondo, esattamente come il nodo principale. Con 80 bit di carico utile, la capacità del canale che parte dal nodo slave è di 64.000 bps come la capacità del canale che parte dal nodo master, più che sufficiente per un singolo canale vocale PCM full duplex (ecco perché è stata scelta una frequenza di salto di 1.600 hop al secondo). Questi numeri implicano che un canale vocale full duplex, con 64.000 bps in ogni direzione, che utilizza il formato più affidabile satura completamente la piconet nonostante la banda grezza di 1 Mbps. Nel caso della variante meno affidabile (240 bit per intervallo, senza ridondanza a questo livello) sono supportati tre canali vocali full duplex alla volta, ecco perché ogni nodo slave non può avere più di tre collegamenti SCO.

Ci sarebbero molte cose in più da dire su Bluetooth, ma manca lo spazio. Per maggiori informazioni, consultare (Bhagwat, 2001; Bisdikian, 2001; Bray and Sturman, 2002; Haartsen, 2000; Johansson et al., 2001; Miller and Bisdikian, 2001; e Sairam et al., 2002).

4.7 Comutazione nello strato data link

Molte aziende hanno diverse LAN che devono essere interconnesse; possono essere collegate da dispositivi chiamati **bridge**, che operano nello strato data link. I bridge esaminano gli indirizzi dello strato data link per eseguire l'instradamento. Poiché non devono esaminare il carico utile dei frame instradati, possono trasportare pacchetti IPv4 (standard utilizzato oggi in Internet), IPv6 (che verrà utilizzato in futuro su Internet), AppleTalk, ATM, OSI e qualunque altro tipo di pacchetto. Al contrario, i **router** esaminano gli indirizzi nei pacchetti e instradano i dati in base a queste informazioni. Anche se la divisione tra bridge e router sembra netta, più avanti si mostrerà come alcuni moderni sviluppi quali l'avvento di Ethernet commutata hanno confuso le acque. I paragrafi seguenti esaminano i bridge e gli switch, spe-

cialmente quelli utilizzati per collegare LAN 802 diverse. Per un'analisi dettagliata dei bridge, degli switch e degli argomenti collegati, consultare (Perlman, 2000).

Prima d'iniziare a esaminare la tecnologia dei bridge conviene descrivere alcune situazioni comuni che impongono la loro installazione. I motivi che possono spingere una singola azienda a creare più LAN sono sei.

1. Molte facoltà universitarie e reparti aziendali hanno una propria LAN, utilizzata prevalentemente per collegare i personal computer, le stazioni di lavoro e i server interni al dipartimento. Poiché ogni ufficio persegue un obiettivo diverso, uffici differenti scelgono LAN differenti senza tener conto delle scelte adottate dagli altri dipartimenti. Quando si presenta la necessità di interagire, ecco che i bridge diventano indispensabili. In questo esempio, la creazione delle LAN è causata dall'autonomia dei loro proprietari.
2. L'azienda può essere distribuita geograficamente su più edifici molto lontani tra loro. In questo caso può risultare più economico avere una LAN in ogni edificio e collegare poi tutte le reti mediante bridge e collegamenti laser piuttosto che stendere un singolo cavo per tutto il sito.

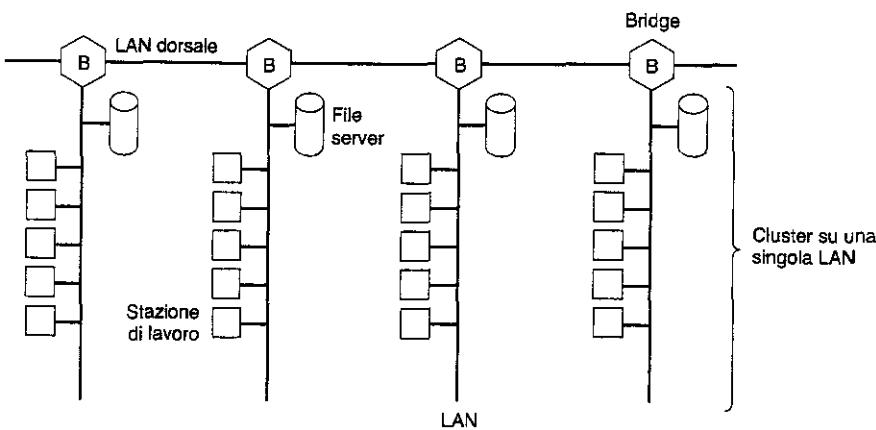


Figura 4.39. LAN collegate da una dorsale per gestire un carico totale superiore della capacità di una singola LAN.

3. Può essere necessario dividere una singola LAN logica in più LAN fisiche per reggere il carico di lavoro. In molte università, per esempio, migliaia di stazioni di lavoro sono a disposizione di studenti e facoltà. I file normalmente sono conservati su computer file server e sono copiati sulle macchine degli utenti su richiesta. L'enorme dimensione di questo sistema non consente di mettere tutte le stazioni di lavoro su una singola LAN, perché l'operazione richiederebbe una banda totale troppo alta. Invece si utilizzano tante LAN collegate mediante bridge (Figura 4.39). Ogni LAN contiene un gruppo di stazioni di lavoro dotate

to di un proprio file server, perciò la maggior parte del traffico circola in una singola LAN e non aumenta il carico della dorsale.

Benché di solito siano rappresentate con cavi multidrop, come nella Figura 4.39 (aspetto classico), oggi le LAN sono implementate soprattutto mediante hub e switch. Dal punto di vista funzionale, però, un lungo cavo multidrop collegato a tante macchine equivale a un hub collegato a più computer. In entrambi i casi, tutte le macchine appartengono allo stesso dominio di collisione e tutte usano il protocollo CSMA/CD per trasmettere i frame. Come verrà spiegato tra poco, le LAN commutate sono diverse.

4. In alcune situazioni una singola LAN potrebbe essere adeguata in termini di carico, ma la distanza fisica tra le macchine più distanti potrebbe essere troppo grande (per esempio potrebbe superare i 2,5 Km nel caso di Ethernet). Anche se fosse facile stendere i cavi, la rete non funzionerebbe a causa dell'eccessivo ritardo tra l'andata e il ritorno. L'unica soluzione è dividere la LAN e installare bridge tra i segmenti. L'utilizzo dei bridge permette di aumentare la distanza fisica totale coperta.
5. C'è la problematica dell'affidabilità. Su una singola LAN, un nodo difettoso che trasmette senza sosta un flusso continuo di segnali difettosi e rumore può paralizzare la rete. I bridge possono essere inseriti in punti critici, come porte antincendio in un edificio, per impedire a un singolo nodo impazzito di paralizzare l'intero sistema. A differenza dei ripetitori, che copiano semplicemente tutto ciò che vedono, un bridge può essere programmato per esercitare un po' di discrezionalità; questo significa che può scegliere che cosa deve passare.
6. I bridge possono essere utilizzati anche per migliorare la sicurezza dell'azienda. La maggior parte delle interfacce LAN supporta una modalità promiscua che consente loro di passare al computer tutti i frame trasmessi attraverso il cavo, non solo quelli indirizzati alla macchina. Spie e ficcanaso adorano questa funzionalità. Inserendo bridge in punti diversi e facendo attenzione a non trasmettere traffico sensibile, un amministratore di sistema può isolare parti della rete in modo che il traffico non esca e cada nelle mani sbagliate.

Idealmente, i bridge dovrebbero essere completamente trasparenti, ossia dovrebbe essere possibile spostare una macchina da un segmento di cavo all'altro senza che sia necessario apportare alcuna modifica all'hardware, al software o alle tabelle di configurazione. Inoltre, le macchine di un segmento dovrebbero poter comunicare con le macchine collegate a qualunque altro segmento senza tener conto dei tipi di LAN in uso sui due segmenti e sui segmenti posti nel mezzo. Questo obiettivo qualche volta è raggiunto, ma non sempre.

4.7.1 Bridge tra 802.x e 802.y

Dopo aver visto perché sono necessari i bridge è giunto il momento di scoprire come funzionano questi apparecchi. La Figura 4.40 mostra il funzionamento di un semplice bridge a due porte. L'host A che si trova su una LAN wireless (802.11) ha un pacchetto da invia-

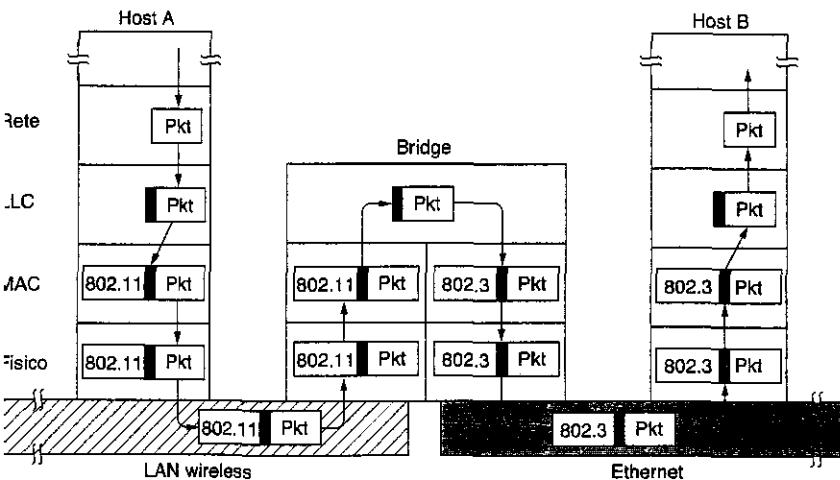


Figura 4.40. Attività svolte da un bridge LAN da 802.11 a 802.3.

a un host fisso *B* che si trova su una Ethernet (802.3) collegata alla LAN wireless. Il pacchetto scende fino al sottostrato LLC e acquisisce un'intestazione LLC (rappresentata nero nella figura d'esempio); quindi passa al sottostrato MAC dove riceve un'intestazione 802.11. L'unità viene trasmessa attraverso lo spazio e arriva alla stazione base, che accorge che i dati devono essere trasferiti all'Ethernet fissa.

Quando arriva al bridge che collega la rete 802.11 alla rete 802.3, il frame parte dallo strato fisico e si sposta verso l'alto. Nel sottostrato MAC del bridge, l'intestazione 802.11 viene strappata via. Il pacchetto nudo (con l'intestazione LLC) viene poi trasmesso nel sottostrato LLC del bridge. In questo esempio il pacchetto è destinato a una LAN 802.3, perciò scende lungo il lato 802.3 del bridge ed entra nella Ethernet. Si noti che un bridge che collega *k* diverse LAN ha *k* diversi sottostrati MAC e *k* differenti strati fisici, uno per ogni tipo.

Ci sembra semplice spostare un frame da una LAN a un'altra. Nulla di più falso. Questo paragrafo mostra alcune delle difficoltà che s'incontrano quando si tenta di costruire un bridge tra differenti LAN (e MAN) 802. Saranno esaminate le reti 802.3, 802.11 e 802.16, ma ce ne sono anche altre, ognuna con specifici problemi.

Per cominciare, ogni LAN utilizza un diverso formato di frame (Figura 4.41). Contrariamente alle differenze tra Ethernet, token bus e token ring (cause dalla storia e l'ego delle grandi aziende), qui le diversità sono per certi versi legittime. Per esempio, campo *durata* di 802.11 è richiesto dal protocollo MACAW e non ha senso in Ethernet. In conseguenza, qualunque operazioni di copia tra LAN diverse richiede una nuova formattazione che consuma capacità di calcolo del processore, esige l'elaborazione di un nuovo checksum e introduce la possibilità di errori non rilevati causati da bit difettosi presenti nella memoria del bridge.

Il secondo problema è che le LAN interconnesse non funzionano necessariamente alla

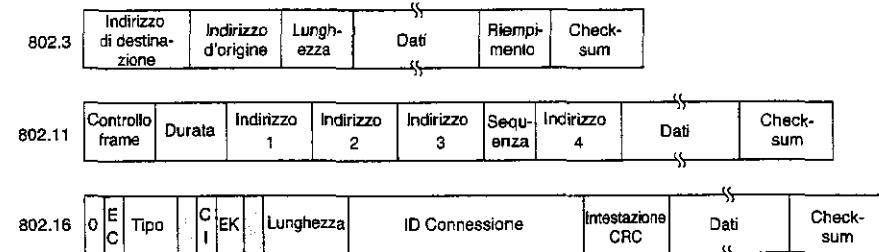


Figura 4.41. I formati di frame IEEE 802. Il disegno non è in scala.

stessa velocità. Quando s'invia una lunga sequenza di frame da una LAN veloce a una LAN più lenta, il bridge non sarà in grado di trasmettere i frame alla stessa velocità con cui li ha ricevuti. Per esempio, se una Ethernet gigabit immette bit alla massima velocità in una LAN 802.11b a 11 Mbps, il bridge deve memorizzare i dati in un buffer sperando di non esaurire la memoria. I bridge che collegano tre o più LAN devono affrontare un problema analogo se diverse LAN tentano di alimentare contemporaneamente la stessa rete di destinazione, anche quando tutte le LAN operano alla stessa velocità.

Un terzo problema, potenzialmente il più serio di tutti, è che LAN 802 diverse adottano differenti lunghezze massime del frame. Un problema ovvio sorge quando un frame lungo deve essere inviato in una LAN che non può accettarlo. In questo strato non è possibile dividere il frame in più parti: tutti i protocolli presuppongono che i frame arrivino oppure no, e non è previsto alcun meccanismo di scomposizione dei frame in unità più piccole. Questo non significa che tali protocolli non possano essere realizzati: sono fattibili e sono stati creati. È solo che nessun protocollo data link fornisce questa funzionalità, perciò i bridge devono tenere le mani lontane dal carico utile del frame. Sostanzialmente, non c'è alcuna soluzione. I frame che non possono essere inviati perché troppo grandi devono essere scartati, per mantenere la trasparenza.

Un altro punto è la sicurezza. 802.11 e 802.16 supportano la crittografia nello strato data link, Ethernet invece no. Questo significa che i servizi di cifratura disponibili sulle reti wireless vanno persi quando il traffico entra in una Ethernet. Peggio ancora, se una stazione wireless utilizza la crittografia nello strato data link, la rete Ethernet non è in grado di decifrare i pacchetti in alcun modo; ma se una stazione wireless non cifra i dati, il suo traffico rimarrà scoperto nel collegamento aereo. In entrambi i casi c'è un problema.

Si può risolvere il problema della sicurezza eseguendo la codifica in un livello più alto, in questo caso però la stazione 802.11 deve sapere se sta comunicando con un'altra stazione di una rete 802.11 (e quindi se è il caso di adottare la codifica sullo strato data link) oppure no (e quindi non utilizzare tale codifica). Costringere la stazione a fare una scelta distrugge la trasparenza.

Un punto finale riguarda la qualità del servizio. 802.11 e 802.16 la forniscono in varie forme, il primo usando la modalità PCF e il secondo usando le connessioni a bit-rate costante. Ethernet non ha alcuna nozione di qualità di servizio, perciò il traffico proveniente dalle altre reti perderà la sua qualità di servizio quando entra in una Ethernet.

4.7.2 Internetworking locale

Il paragrafo precedente ha descritto i problemi che si presentano collegando due diverse LAN IEEE 802 attraverso un singolo bridge. Nelle grandi aziende contenenti molte LAN, però, l'interconnessione solleva una varietà di problematiche anche quando l'unico standard in uso è Ethernet. Idealmente dovrebbe essere possibile ordinare bridge progettati per il standard IEEE, infilare una spina e ottenere un funzionamento perfetto all'istante. In teoria non dovrebbe essere necessario apportare alcuna modifica all'hardware e al software, cambiare alcuna impostazione degli indirizzi, scaricare tabelle di instradamento o parametri e così via: si attaccano i cavi e il problema è risolto. Inoltre, il funzionamento delle LAN esistenti non dovrebbe essere influenzato in alcun modo dai bridge. In altre parole, bridge dovrebbero essere completamente trasparenti (invisibili a tutto l'hardware e a tutto il software). È piuttosto sorprendente scoprire che tutto ciò è realmente possibile. Questo paragrafo spiega come si può realizzare la magia.

Nella sua forma più semplice, un bridge trasparente opera in modalità promiscua, ossia accetta ogni frame trasmesso su ogni LAN a cui è collegato. Come esempio, si consideri la configurazione rappresentata nella Figura 4.42: il bridge B1 è collegato alle LAN 1 e 2, e il bridge B2 è collegato alle LAN 2, 3 e 4. Un frame della LAN 1 in arrivo sul bridge B1 è destinato alla stazione A viene scartato immediatamente perché si trova già sulla LAN giusta, ma un frame della LAN 1 indirizzato a C o F deve essere instradato correttamente.

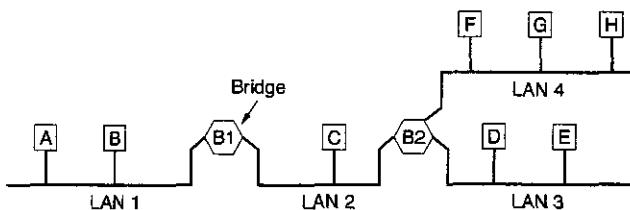


Figura 4.42. Una configurazione con quattro LAN e due bridge.

Quando arriva un frame, il bridge deve decidere se scartare i dati o inoltrarli; nel secondo caso deve stabilire su quale LAN immettere il frame. La decisione è fatta confrontando l'indirizzo di destinazione con le informazioni riportate in una grande tabella (*hash*) memorizzata nel bridge. La tabella può elencare ogni possibile destinazione e dire a quale rete di output (LAN) appartiene. Per esempio, la tabella di B2 elencherà A come appartenente a LAN 2, poiché tutto quello che B2 deve sapere è su quale LAN immettere il frame per A. Il fatto che in realtà verranno successivamente eseguite altre operazioni di output non interessa in alcun modo il bridge.

Appena i bridge sono collegati alla rete per la prima volta, tutte le tabelle di hash sono vuote. Poiché non sanno dove si trovano le varie destinazioni, i bridge utilizzano un algoritmo di allagamento (*flooding*): ogni frame proveniente da una destinazione sconosciuta viene inviato a tutte le LAN connesse al bridge, tranne a quella di input. Col tempo i bridge

imparano dove si trovano le destinazioni. Una volta scoperte, i frame destinati a una destinazione saranno inviati solo alla LAN contenente la stazione di arrivo.

L'algoritmo utilizzato dai bridge trasparenti è chiamato di **apprendimento all'indietro**. Come è stato spiegato precedentemente, i bridge operano in modalità promiscua, perciò vedono ogni frame inviato su qualunque delle loro LAN e l'esame dell'indirizzo sorgente fa scoprire le macchine collegate alle varie LAN. Per esempio, se nella Figura 4.42 il bridge B1 rileva sulla LAN 2 un frame proveniente da C, il dispositivo capisce che la stazione C può essere raggiunta attraverso la LAN 2, e crea una voce nella sua tabella di hash che prende nota del fatto che i frame diretti a C dovrebbero essere trasmessi alla LAN 2. I successivi frame indirizzati a C e provenienti dalla LAN 1 saranno inoltrati, ma i frame per C provenienti da LAN 2 saranno scartati. La topologia può cambiare quando le macchine e i bridge vengono accesi, spenti o spostati. Per gestire le topologie dinamiche, ogni volta che crea una voce della tabella di hash il bridge annota anche il tempo di arrivo del frame. Ogni volta che riceve un frame la cui origine è già stata registrata nella tabella, il bridge aggiorna la voce associata prendendo nota del tempo corrente; di conseguenza, il tempo associato a ogni voce indica quando è stato rilevato per l'ultima volta un frame proveniente da quella macchina. Periodicamente, un processo nel bridge esamina la tabella di hash e cancella le voci che hanno più di qualche minuto. In questo modo un computer scollegato dalla sua LAN, spostato in un altro punto dell'edificio e collegato a un'altra LAN torna operativo in pochi minuti senza necessità d'intervento manuale. A causa di questo algoritmo, se una macchina rimane silenziosa per alcuni minuti tutto il traffico che le viene inviato dovrà essere trasmesso attraverso tutte le reti fino a quando la stazione non invierà un suo frame. La procedura d'instradamento di un frame in arrivo dipende dalla LAN sorgente e dalla LAN di destinazione:

1. se la LAN di destinazione e la LAN sorgente coincidono, il frame è scartato
2. se la LAN di destinazione e la LAN sorgente sono diverse, il frame è inoltrato
3. se la LAN di destinazione è sconosciuta, si utilizza la trasmissione su tutte le LAN [NrR - escludendo ovviamente la LAN dalla quale è arrivato il frame da ritrasmettere].

Ogni volta che arriva un frame il bridge deve applicare questo algoritmo. Chip VLSI con funzioni speciali eseguono la ricerca e aggiornano le voci della tabella in pochi microsecondi.

4.7.3 Bridge spanning tree

Per aumentare l'affidabilità, alcuni siti installano due o più bridge in parallelo tra coppie di LAN, come mostrato nella Figura 4.43. Questa soluzione introduce però nuovi problemi, poiché crea anelli nella topologia. Un semplice esempio di questi problemi è evidente osservando come viene gestito nella Figura 4.43 un frame, F_1 , la cui destinazione è sconosciuta. Ogni bridge, seguendo le normali regole di gestione delle destinazioni sconosciute, trasmette il frame su tutte le altre reti di output, in questo caso solo sulla LAN 2. Poco dopo, il bridge 1 riceve F_2 , un frame con destinazione sconosciuta; questi dati, immessi sulla LAN 1, danno origine al frame F_3 (non rappresentato nell'immagine).

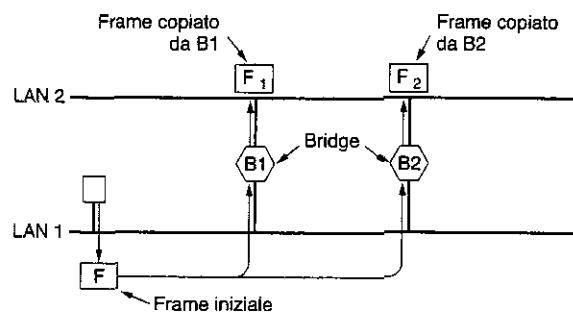


Figura 4.43. Due bridge paralleli trasparenti.

In modo analogo, il bridge 2 copia F_1 sulla LAN 1 generando F_4 , in un ciclo che continua all'infinito. Per risolvere questo problema è necessario che i bridge comunichino tra loro coprano la topologia reale con una struttura spanning tree che raggiunga ogni LAN; in realtà alcune connessioni potenziali tra LAN vengono ignorate, nell'interesse della costruzione di una topologia fittizia priva di anelli. Per esempio, nella Figura 4.44(a) si osservano nove LAN interconnesse da dieci bridge; questa configurazione può essere descritta su un grafico dove ogni LAN è rappresentata da un nodo: un arco collega ogni coppia di LAN connesse da un bridge.

Il grafico può essere ridotto a una struttura ad albero sostituendo gli archi con linee tratteggiate come è stato fatto nella Figura 4.44(b). Nella struttura ad albero un solo percorso collega ogni LAN a ogni altra LAN. Una volta che i bridge hanno concordato lo spanning tree, tutti gli inoltri tra LAN seguono la struttura. Poiché esiste un unico percorso che collega ogni sorgente a ogni destinazione, gli anelli sono impossibili.

Per costruire uno spanning tree, i bridge devono prima di tutto scegliere tra loro quello che avrà funzione da nodo principale della struttura. Per fare questa scelta, ogni bridge trasmette a tutti il proprio numero di serie, installato dal produttore dell'hardware e garantito come univoco in tutto il mondo. Il bridge con il numero di serie più basso diventa il nodo principale. Quindi viene costruita una struttura ad albero, basata sui percorsi più brevi che uniscono il nodo principale a ogni bridge e LAN; questa struttura è lo spanning tree. Se viene meno un bridge o una LAN si elabora una nuova struttura.

Il risultato di questo algoritmo è la definizione di un percorso unico da ogni LAN verso il nodo principale, e di conseguenza verso ogni altra LAN. Sebbene la struttura copra tutte le LAN, non tutti i bridge sono per forza presenti nell'albero (questo per prevenire i cicli). Anche dopo aver stabilito lo spanning tree, durante le normali operazioni l'algoritmo continua a funzionare per rilevare automaticamente le modifiche della topologia e aggiornare la struttura. L'algoritmo distribuito utilizzato per costruire la struttura spanning tree è stato inventato da Radia Perlman ed è descritto dettagliatamente in (Perlman, 2000). Inoltre è stato standardizzato in IEEE 802.1D.

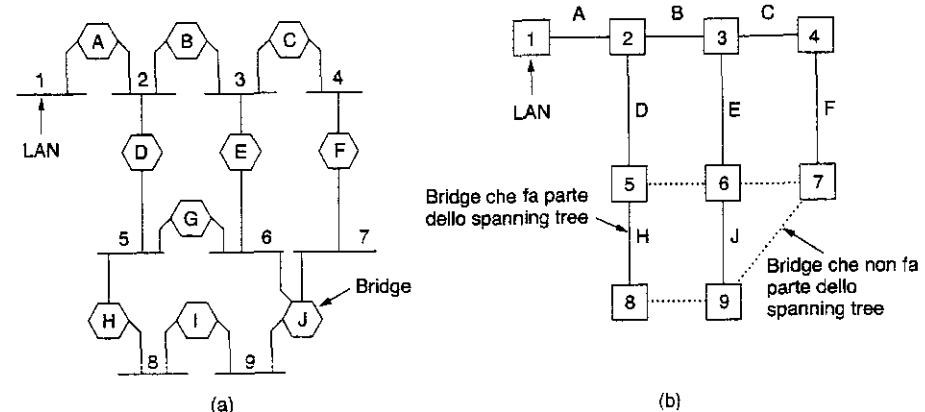


Figura 4.44. (a) LAN collegate. (b) Uno spanning tree che copre le LAN. Le linee tratteggiate non fanno parte dello spanning tree.

4.7.4 Bridge remoti

I bridge sono comunemente utilizzati per collegare due (o più) LAN distanti. Per esempio, un'azienda potrebbe avere sedi in diverse città, ognuna dotata di una propria LAN. Idealmente, tutte le LAN potrebbero essere interconnesse in modo che il sistema funzioni come una singola grande LAN.

Questo obiettivo può essere raggiunto installando un bridge in ogni LAN e collegando i bridge a coppie con una linea punto-punto (per esempio, con linee private prese in affitto da un'azienda telefonica). La Figura 4.45 mostra un sistema semplice composto da tre LAN; i soliti algoritmi d'instradamento si applicano anche in questo caso. Per semplificare il tutto, si considerino le tre linee punto-punto come LAN senza host; un sistema reale composto da sei LAN interconnesse da quattro bridge sarà analizzato in un secondo momento. Nulla di tutto ciò che è stato spiegato fino ad ora impone che la LAN debba avere degli host al suo interno.

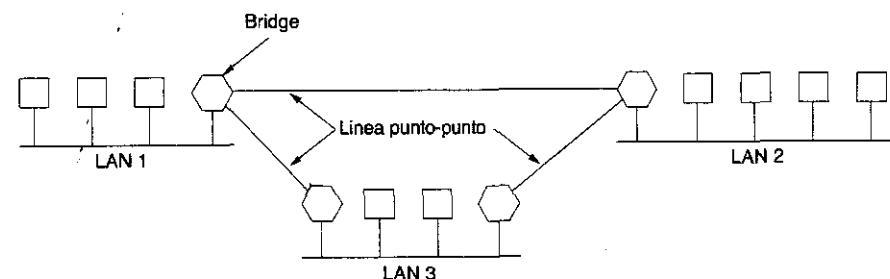


Figura 4.45. Bridge remoti possono essere utilizzati per collegare LAN distanti.

Sulle linee punto-punto si possono usare molti protocolli, per esempio si potrebbero scegliere alcuni protocolli di collegamento dati punto-punto standard (come PPP), inserendo nel carico utile i frame MAC completi. Questa strategia funziona meglio se tutte le LAN sono identiche, e l'unico problema è l'invio dei frame alle giuste LAN. Oppure si potrebbero strappare intestazione e coda MAC nel bridge sorgente, e mettere ciò che rimane nel carico utile del protocollo punto-punto. Una nuova intestazione MAC e una nuova coda potrebbero quindi essere generate nel bridge di destinazione. Questo approccio ha uno svantaggio: il checksum che arriva all'host di destinazione non è quello elaborato dall'host sorgente, perciò non è possibile rilevare gli errori causati da eventuali bit guasti della memoria del bridge.

4.7.5 Ripetitori, hub, bridge, switch, router e gateway

I paragrafi precedenti hanno descritto diverse tecniche che consentono di passare i frame e i pacchetti da un segmento di cavo all'altro. Il libro ha parlato di ripetitori, bridge, switch, hub, router e gateway. Tutti questi dispositivi sono di uso comune, ma differiscono in modo più o meno sottile. Poiché sono così tanti, probabilmente vale la pena esaminarli tutti insieme per vedere quali sono le analogie e le differenze.

Tanto per cominciare, questi dispositivi operano su strati diversi, come mostrato nella Figura 4.46(a). Lo strato è importante perché dispositivi diversi utilizzano differenti porzioni d'informazione per decidere come eseguire la commutazione. In uno scenario tipico, l'utente genera alcuni dati da inviare a una macchina remota; quei dati vengono passati allo strato di trasporto che aggiunge un'intestazione, per esempio un'intestazione TCP, e passa l'unità risultante al sottostante strato network. Questo aggiunge la propria intestazione, in modo da costruire il pacchetto dello strato network, per esempio un pacchetto IP. Nella Figura 4.46(b), il pacchetto IP è evidenziato in grigio. Il pacchetto viene quindi trasferito nello strato data link, che aggiunge la sua intestazione e il suo checksum CRC, e passa il frame risultante allo strato fisico che gestisce la trasmissione, per esempio attraverso una LAN.

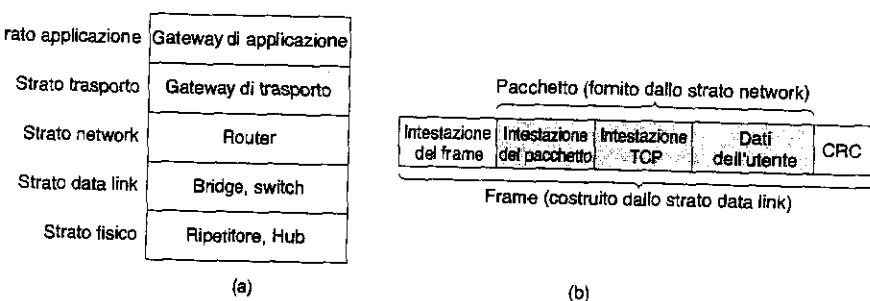


Figura 4.46. (a) Posizione dei dispositivi. (b) Frame, pacchetti e intestazioni.

È utile osservare quali relazioni si formano tra dispositivi di commutazione, pacchetti e frame. I ripetitori si trovano sul fondo, nello strato fisico. Sono dispositivi analogici collegati a due segmenti di cavo. Un segnale che appare su un segmento è amplificato e trasmesso sull'altro. I ripetitori non sanno nulla dei frame, dei pacchetti e delle intestazioni; essi comprendono solo i Volt. Ethernet classica, per esempio, è stata progettata per supportare quattro ripetitori al fine di estendere la lunghezza massima del cavo da 500 metri a 2.500 metri.

Poi ci sono gli hub. Un hub ha diverse linee di input collegate elettricamente. I frame che arrivano su una di queste linee sono trasmessi attraverso tutte le altre. Due frame che arrivano contemporaneamente collidono, proprio come su un cavo coassiale. In altre parole, l'intero hub forma un singolo dominio di collisione. Tutte le linee di input devono operare alla stessa velocità. A differenza dei ripetitori, gli hub (di solito) non amplificano i segnali in ingresso e sono progettati per contenere diverse schede di linea, ognuna con più porte, ma le differenze sono piccole. Come i ripetitori, gli hub non esaminano gli indirizzi 802 e non li utilizzano in alcun modo. La Figura 4.47(a) mostra un hub.

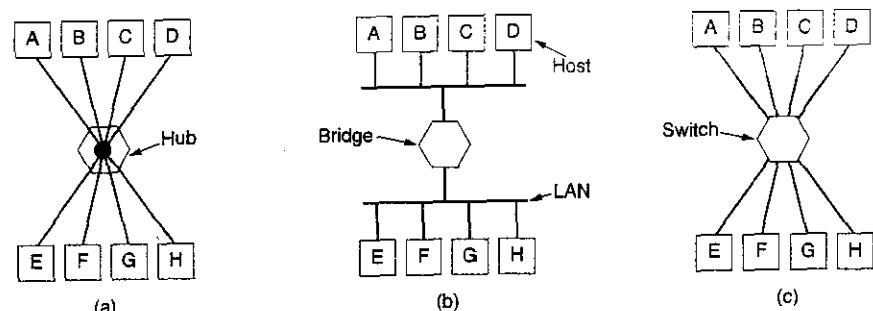


Figura 4.47. (a) Un hub. (b) Un bridge. (c) Uno switch.

É arrivato il turno dello strato data link, dove si trovano i bridge e gli switch. Come è stato spiegato precedentemente, un bridge collega due o più LAN come mostrato nella Figura 4.47(b). Quando arriva un frame, il software che si trova nel bridge estrae l'indirizzo di destinazione dall'intestazione e lo confronta con le voci della sua tabella per scoprire la destinazione dei dati. Nel caso di Ethernet, questo indirizzo è l'indirizzo di destinazione lungo 48 bit mostrato nella Figura 4.17. Come gli hub, i moderni bridge hanno schede di linea, di solito per quattro o otto linee di input di un certo tipo. Una scheda di linea per Ethernet non è in grado di gestire, per esempio, i frame token ring perché non sa dove trovare l'indirizzo di destinazione nell'intestazione del frame. Un bridge però può avere schede di linea per reti e velocità diverse. Con un bridge, ogni linea è un dominio di collisione indipendente, al contrario dell'hub.

Gli switch sono simili ai bridge in quanto entrambi instradano i dati in base agli indirizzi dei frame; molte persone infatti utilizzano i termini in modo intercambiabile. La differenza principale è che lo switch è utilizzato prevalentemente per collegare singoli computer, come

mostrato nella Figura 4.47(c). Di conseguenza, quando l'host A nella Figura 4.47(b) vuole inviare un frame all'host B, il bridge riceve il frame e lo scarta mentre lo switch nella Figura 4.47(c) deve inoltrare attivamente il frame da A a B, perché i dati non possono raggiungere destinazione in nessun altro modo. Poiché ogni porta dello switch di solito conduce a un singolo computer, gli switch devono poter contenere un numero maggiore di schede di linea rispetto ai bridge usati per collegare solo le LAN. Ogni scheda di linea è in grado di memorizzare in un buffer i frame che arrivano attraverso le sue porte. Poiché ogni porta è un dominio di collisione indipendente, gli switch non perdono mai i frame a causa delle collisioni. Comunque, se i frame arrivano più velocemente di quanto possano essere trasmessi, lo switch può esaurire lo spazio del buffer e iniziare a scartare i frame in eccesso.

Per attenuare leggermente questo problema, gli switch moderni iniziano a inoltrare i frame non appena arriva il campo *destinazione* dell'intestazione, prima che arrivi il resto del frame (purché la linea di output sia disponibile, logicamente). Questi switch non usano la commutazione store-and-forward, e qualche volta sono chiamati **switch a commutazione diretta** o **switch cut-through**. Di solito la commutazione diretta è gestita interamente dal hardware, mentre i bridge tradizionali contengono una CPU che gestisce la commutazione store-and-forward via software. Poiché tutti i bridge e gli switch moderni contengono dei speciali circuiti integrati dedicati alla commutazione, la differenza tra uno switch e un bridge è più una questione di mercato che di tecnologia.

Dopo aver esaminato i ripetitori e gli hub, che sono abbastanza simili, e i bridge e gli switch (anche questi molto simili tra loro), è giunto il momento di analizzare i router, che non hanno nulla in comune con i precedenti. Quando un pacchetto raggiunge un router, l'intestazione e la coda del frame sono strappati via e il pacchetto contenuto nel carico utile del frame (evidenziato in grigio nella Figura 4.46) è passato al software d'instradamento, che sfrutta l'intestazione del pacchetto per scegliere la linea di output. Nel caso di un pacchetto IP, l'intestazione del pacchetto contiene un indirizzo a 32 bit (IPv4) o un indirizzo a 128 bit (IPv6), ma non un indirizzo 802 a 48 bit. Il software d'instradamento non sa degli indirizzi del frame e non sa nemmeno se il pacchetto arriva da una LAN o da una rete punto-punto. I router sono descritti in modo dettagliato nel Capitolo 5.

Al livello ancora superiore troviamo i gateway di trasporto. Questi dispositivi collegano i computer che usano protocolli di trasporto orientati alle connessioni differenti. Per esempio, si supponga che un computer che utilizza il protocollo TCP/IP orientato alle connessioni debba comunicare con un computer che usa il protocollo di trasporto ATM, che invece è orientato alle connessioni. Il gateway di trasporto può copiare i pacchetti provenienti da una connessione sull'altra, modificando il formato secondo necessità. Infine, i gateway di applicazione comprendono il formato e il contenuto dei dati e trasformano i messaggi da un formato all'altro. Un gateway di posta elettronica, per esempio, rebbe tradurre i messaggi Internet in messaggi SMS per i telefoni cellulari.

4.6 LAN virtuali

I primi giorni dell'era delle reti locali, grossi cavi gialli si snodavano attraverso i corridoi di molti uffici; ogni computer era collegato a questi cavi, e spesso esistevano molti computer collegati a una dorsale (come nella Figura 4.39) o a un hub centrale. Non ci si curava

di sapere a quale LAN appartenessero i computer: tutte le persone che si trovavano in uffici adiacenti erano collegate alla stessa LAN anche se non facevano parte dello stesso gruppo, secondo una logica basata sulla geografia.

Negli anni '90, l'avvento di 10Base-T e degli hub cambiò tutto. Gli edifici furono cablati un'altra volta (con grandi spese) per sradicare tutti i cavi gialli e installare doppini che collegavano ogni ufficio a una centralina collocata alla fine di ogni corridoio (o in una sala macchine centrale) come mostrato nella Figura 4.48. Se chi gestiva il cablaggio era un visionario, nell'edificio venivano installati doppini di categoria 5; chi invece preferiva risparmiare utilizzava i cavi telefonici esistenti di categoria 3 (sostituiti qualche anno più tardi alla comparsa di fast Ethernet).

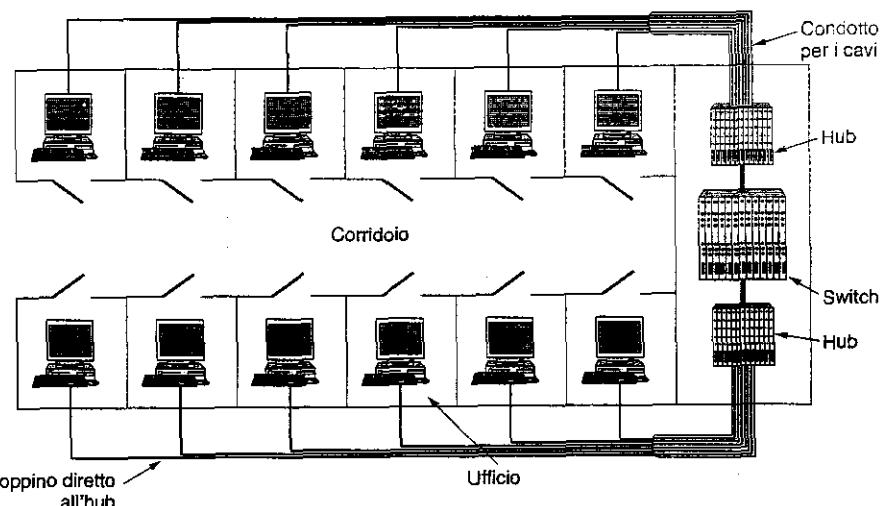


Figura 4.48. Un edificio con cablaggio centralizzato che utilizza hub e switch.

Con Ethernet basata sugli hub (e più tardi sugli switch), spesso era possibile configurare le LAN logicamente invece che fisicamente. Se voleva k LAN, l'azienda poteva acquistare k hub. Scogliendo attentamente il cablaggio dei cavi di permuta collegati a ogni hub, gli utenti di una LAN potevano essere selezionati in un modo che aveva senso dal punto di vista organizzativo, senza badare troppo alla geografia fisica. Certo, se due persone dello stesso reparto lavorano in edifici diversi, è probabile che ognuna sia collegato a un hub differente e di conseguenza a una LAN diversa. Ciò nonostante, la situazione è di gran lunga migliore rispetto a una LAN i cui utenti sono scelti in base alla loro posizione geografica.

Ha importanza sapere chi appartiene a una LAN? Dopo tutto, in qualsiasi azienda tutte le LAN sono interconnesse. La risposta è sì: spesso è importante. Per molte ragioni gli amministratori di rete amano raggruppare gli utenti delle LAN per riflettere la struttura dell'azienda, invece dello schema fisico dell'edificio. La prima è la sicurezza: ogni scheda di rete può essere attivata in modalità promiscua, e copiare tutto il traffico che scorre

attraverso il cavo. Molti reparti, per esempio quelli impegnati nella ricerca, nello studio dei brevetti e nella contabilità, hanno informazioni che non vogliono far uscire all'esterno dei loro uffici. In queste situazioni, ha senso mettere tutte le persone di un reparto su una singola LAN e impedire al traffico di abbandonare la rete. Alla dirigenza non piacerebbe senz'altro rispondere che questa sistemazione sarebbe fattibile solo mettendo tutte le persone di ciascun reparto in uffici adiacenti (e senza intrusi).

Un secondo problema è il carico. Alcune LAN sono utilizzate più intensamente di altre, e talvolta può essere utile separarle. Per esempio, se gli esperti della ricerca eseguono straordinari esperimenti di ogni tipo che qualche volta scappano di mano saturando la loro rete, gli impiegati della contabilità potrebbero non essere entusiasti di regalare un po' della capacità della loro rete per risolvere l'impiccio.

Un terzo problema è la trasmissione broadcast. La maggior parte delle LAN supporta questo tipo di comunicazione, e molti protocolli di strato superiore usano in modo esteso questa funzionalità. Per esempio, quando un utente deve inviare un pacchetto a un dato indirizzo IP x , come fa a sapere l'indirizzo MAC da inserire nel frame? Una risposta dettagliata è riportata nel Capitolo 5, ma riassumendo il concetto si può dire che la macchina dell'utente trasmette in modalità broadcast un frame contenente la domanda "A chi appartiene l'indirizzo x ?" e poi attende una risposta. La trasmissione broadcast è utilizzata in molte altre situazioni. Il numero di trasmissioni broadcast che attraversa ogni computer è direttamente proporzionale al numero di LAN interconnesse.

In problema collegato alla trasmissione broadcast si presenta quando, occasionalmente, una scheda di rete si rompe e inizia a generare un flusso senza fine di frame broadcast. Il risultato di questa **tempesta broadcast** è che (1) l'intera capacità della LAN è occupata da questi frame e che (2) tutte le macchine su tutte le LAN interconnesse restano paralizzate, a causa delle operazioni di elaborazione ed eliminazione dei frame trasmessi in modalità broadcast.

Da prima vista si potrebbe pensare di confinare le tempeste broadcast separando le LAN con bridge o switch, ma poiché l'obiettivo dei bridge è la trasparenza (ossia le macchine devono potersi spostare da una LAN all'altra senza che si noti la differenza), questi dispositivi devono inoltrare i frame broadcast. Dopo aver visto perché le aziende potrebbero voler avere LAN distinte e dedicate, è meglio tornare al problema relativo al disaccoppiamento della topologia logica da quella fisica. Si supponga che un utente all'interno dell'azienda venga spostato da un reparto a un altro senza cambiare ufficio, oppure cambi ufficio senza cambiare reparto. Se si usano gli hub, per spostare l'utente sulla giusta LAN è necessario che l'amministratore di rete raggiunga la centralina, stacchi il cavo di rete della macchina dell'utente dall'hub e lo colleghi al nuovo hub. In molte aziende le modifiche organizzative capitano continuamente, e ciò significa che gli amministratori di sistema passano un sacco di tempo staccando cavi da una porta e inserendoli in un'altra. Inoltre, in alcuni casi, una modifica non può essere apportata perché il doppino che collega la macchina dell'utente è troppo lontano dall'hub (per esempio, si trova nell'edificio sbagliato).

Rispondere agli utenti che chiedevano maggiore flessibilità, i produttori di dispositivi rete hanno studiato un metodo per ricablarle gli edifici interamente via software. Il concepto risultante è stato chiamato **VLAN** (*Virtual LAN*), standardizzato dal comitato 802, e

oggi è utilizzato in molte aziende. Ne vedremo per sommi capi il funzionamento; per ulteriori informazioni sulle VLAN consultare (Breyer and Riley, 1999; e Seifert, 2000).

Le VLAN si basano su switch VLAN compatibili progettati in modo speciale, ma possono avere anche hub lungo il perimetro esterno, come mostrato nella Figura 4.48. Per impostare una rete basata su VLAN, l'amministratore di rete decide quante VLAN creare, quali computer collegare e quale nome assegnare a ogni VLAN. Spesso le VLAN vengono (ufficiosamente) battezzate con nomi di colori, poiché in questo modo è possibile stampare diagrammi a colori che mostrano lo schema fisico delle macchine, con i membri della LAN rossa rappresentati in rosso, quelli della LAN verde rappresentati in verde e così via. In tal modo, con un solo schema è possibile rappresentare sia la disposizione fisica sia la disposizione logica della rete.

Come esempio, si considerino le quattro LAN della Figura 4.49(a): otto macchine appartengono alla VLAN G (grigia) e sette appartengono alla VLAN W (bianca). Le quattro LAN fisiche sono collegate mediante due bridge, $B1$ e $B2$. Usando un cablaggio a doppiini centralizzato ci potrebbero anche essere quattro hub (non mostrati nel disegno), ma dal punto di vista logico un cavo multidrop e un hub sono la stessa cosa. Rappresentando la rete in questo modo la figura appare un po' meno disordinata. Inoltre, il termine *bridge* oggi tende a essere utilizzato perlopiù quando ci sono più macchine su ogni porta, come in questa figura, ma altrimenti bridge e switch sono essenzialmente intercambiabili. La Figura 4.49(b) mostra le stesse VLAN e le stesse macchine collegate tramite switch dove a ogni porta è connesso un solo computer.

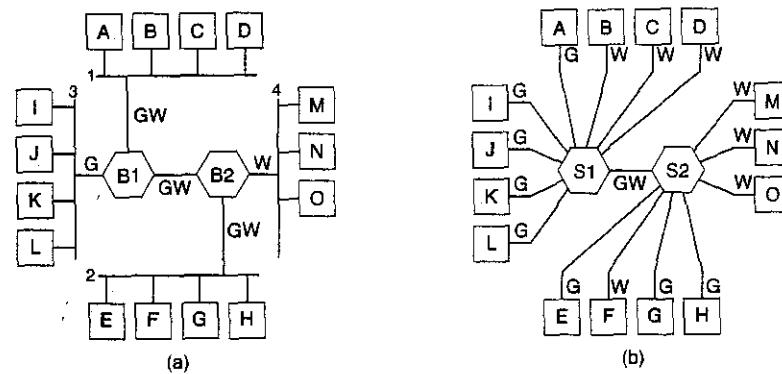


Figura 4.49. (a) Quattro LAN fisiche organizzate in due VLAN, una grigia e l'altra bianca; attraverso due bridge. (b) Le stesse 15 macchine organizzate in due VLAN mediante switch.

Per far funzionare correttamente le VLAN, nei bridge o negli switch devono essere impostate delle tabelle di configurazione. Queste tabelle indicano quali VLAN sono accessibili attraverso le varie porte (linee). Quando arriva un frame, per esempio dalla VLAN grigia, i dati devono essere inoltrati su tutte le porte contrassegnate dalla lettera G. Questo vale per il traffico ordinario (unicast) e per quello multicast o broadcast.

Come si può notare osservando la Figura 4.49(a), una porta può essere etichettata con più colori VLAN. Si supponga che la macchina A stia trasmettendo un frame in modalità broadcast. Il bridge *B1* riceve il frame e scopre che proviene da una macchina della VLAN grigia, perciò inoltra i dati su tutte le porte etichettate con la lettera G (tranne la porta di input). Poiché *B1* ha solo altre due porte ed entrambe sono contrassegnate dalla lettera G, il frame è trasmesso a entrambe.

Con *B2* la storia è diversa: in questo caso il bridge sa che non ci sono macchine grigie sulla LAN 4, perciò non inoltra il frame in quella direzione; i dati arrivano solo alla LAN 2. Se uno degli utenti della LAN 4 cambia reparto e si sposta sulla VLAN grigia, bisogna aggiornare le tabelle dentro *B2* sostituendo la lettera W con le lettere GW. Se la macchina A diventa grigia, allora la porta diretta alla LAN 2 deve passare da GW a G.

Ora si supponga che tutte le macchine di LAN 2 e di LAN 4 diventino grigie. In questo caso, non soltanto le porte di *B2* dirette alle LAN 2 e 4 verrebbero contrassegnate con la lettera G, ma anche la porta di *B1* diretta a *B2* passerebbe da GW a G poiché i frame bianchi che entrano in *B1* dalle LAN 1 e 3 non dovranno più essere inoltrati a *B2*. Nella Figura 4.49(b) accade la stessa situazione, solo che qui tutte le porte che vanno a un dato computer sono etichettate con un solo colore perché ogni porta conduce a una sola VLAN. In cui è stato dato per scontato che i bridge e gli switch possano distinguere in qualche modo il colore dei frame in arrivo. Come fanno a riconoscere le VLAN di destinazione? Sono utilizzati i seguenti tre metodi:

1. a ogni porta è assegnato un colore VLAN
2. a ogni indirizzo MAC è assegnato un colore VLAN
3. a ogni protocollo di strato 3 oppure a ogni indirizzo IP è assegnato un colore VLAN.

Nel primo metodo ogni porta ha un'etichetta che rappresenta un colore VLAN, ma funziona solo se tutti i computer collegati a una porta appartengono alla stessa VLAN. Nella Figura 4.49(a), in *B1* questa proprietà vale per la porta diretta alla LAN 3 ma non per la porta diretta alla LAN 1.

Nel secondo metodo, il bridge o lo switch ha una tabella che elenca l'indirizzo MAC a 48 bit di ogni computer collegato al dispositivo, abbinato alla indicazione della VLAN di appartenenza di quel computer. In queste condizioni è possibile mischiare le VLAN su una LAN fisica, come accade nella LAN 1 della Figura 4.49(a). Quando arriva un frame, per sapere la VLAN di provenienza il bridge o lo switch non deve far altro che estrarre l'indirizzo MAC e cercarlo nella tabella interna.

Terzo metodo permette al bridge o allo switch di esaminare il carico utile del frame, per esempio per classificare tutte le macchine IP come appartenenti a una VLAN e tutte le macchine AppleTalk come appartenenti a un'altra VLAN. Le prime potranno anche essere identificate mediante il loro indirizzo IP. Questa strategia è particolarmente utile quando molte macchine sono computer portatili collegabili a docking station diverse. Poiché ogni docking station ha il proprio indirizzo MAC, individuare la docking station utilizzata aiuta a identificare la VLAN su cui si trova il portatile.

L'unico problema di questo approccio è che viola la più elementare regola della comunicazione di rete: indipendenza dagli strati. Il contenuto del carico utile non riguarda lo strato data link; questo strato non dovrebbe esaminare il carico utile e certamente non dovrebbe prendere decisioni in base al suo contenuto. Una conseguenza dell'utilizzo di questo approccio è che una modifica al protocollo dello strato 3 (per esempio, un aggiornamento da IPv4 a IPv6) può bloccare improvvisamente tutti gli switch. Sfortunatamente sono in commercio switch che funzionano in questo modo.

Naturalmente non c'è nulla di sbagliato nell'instradamento basato sugli indirizzi IP (quasi tutto il Capitolo 5 è dedicato all'instradamento IP), ma mischiare gli strati significa andare in cerca di guai. Un produttore di switch potrebbe prendere alla leggera questo argomento e dire che i suoi switch sono compatibili sia con IPv4 sia con IPv6, perciò qual è il problema? Ma che cosa accadrà quando farà la sua comparsa IPv7? Il produttore probabilmente risponderà: "Basta acquistare nuovi switch."

Lo standard IEEE 802.1Q

Approfondendo la riflessione su questo tema si arriva a stabilire che ciò che importa realmente è la VLAN del frame stesso, non la VLAN della macchina trasmittente. Se ci fosse un modo per identificare la VLAN nell'intestazione del frame, allora non sarebbe più necessario esaminare il carico utile. Per una nuova LAN, come 802.11 o 802.16, sarebbe stato abbastanza facile aggiungere nell'intestazione un nuovo campo VLAN. In effetti, il campo *connection identifier* in 802.16 assomiglia nello spirito all'identificatore di VLAN. Ma che fare con Ethernet, standard dominante nel settore delle LAN, che non ha alcun campo di riserva utilizzabile come identificatore di VLAN?

Il comitato IEEE 802 ha affrontato questo problema nel 1995. Dopo molte discussioni, è accaduto l'impossibile: l'intestazione Ethernet è stata modificata. Il nuovo formato è stato pubblicato nel 1998 nello standard **IEEE 802.1Q**. Il nuovo formato contiene un'etichetta VLAN che verrà descritta fra poco. Com'è prevedibile, cambiare una cosa così ben radicata come l'intestazione Ethernet non è del tutto banale. Alcune domande sorgono spontanee.

1. Dopo la modifica sarà necessario sbarazzarsi di centinaia di milioni di schede di rete esistenti?
2. Se non sarà necessario farlo, chi genererà i nuovi campi?
3. Che cosa accadrà ai frame che hanno già la dimensione massima?

Naturalmente, il comitato 802 era consapevole di questi problemi (anche troppo) e dovete trovare alcune soluzioni.

La chiave di tutto sta nel comprendere che i campi VLAN sono effettivamente utilizzati solo dai bridge e dagli switch, non dalle macchine degli utenti. Così, nella Figura 4.49 non è realmente necessaria la loro presenza sulle linee in uscita collegate alle stazioni finali, a patto che siano presenti sulla linea che collega i bridge o gli switch. Perciò, per utilizzare le VLAN i bridge e gli switch devono essere VLAN compatibili, ma questo era già un requisito necessario. È sufficiente aggiungere un nuovo requisito, quello relativo alla compatibilità con 802.1Q.

per quanto riguarda la domanda relativa alla sostituzione di tutte le vecchie schede di rete la risposta è no. Tenuto conto che il comitato 802.3 non era nemmeno riuscito a convincere le persone a trasformare il campo *tipo* in *lunghezza*, è facile immaginare la ragione che avrebbe generato l'annuncio della necessità di sostituire tutte le schede Ethernet esistenti. Comunque, si spera che le nuove schede Ethernet messe in commercio siano compatibili con lo standard 802.1Q e siano in grado di gestire correttamente i campi VLAN.

così, se il trasmittente non genera i campi VLAN, chi esegue questa operazione? La risposta è che il primo bridge o switch compatibile VLAN che tocca il frame aggiunge i campi all'ultimo dispositivo sul percorso del frame li rimuove. Ma come si determina la VLAN appartenenza del frame? Ebbene, il primo bridge o switch potrebbe assegnare un numero VLAN a una porta, osservare l'indirizzo MAC o addirittura esaminare il carico utile. Non a che le schede Ethernet non saranno tutte compatibili con lo standard 802.1Q, sarà come tornare al punto di partenza. La vera speranza è che tutte le schede Ethernet gigabit saranno compatibili con 802.1Q fin dall'inizio; quando le persone passeranno a Ethernet gigabit, 802.1Q verrà introdotto automaticamente. Infine, per quanto riguarda il problema dei frame più lunghi di 1.518 Byte, 802.1Q ha semplicemente portato il limite a 1.522 byte.

Durante il processo di transizione, molte installazioni avranno alcune macchine di vecchio tipo (legate alla Ethernet classica o fast) non compatibili con VLAN e altre (tipicamente Ethernet gigabit) che invece lo saranno. La Figura 4.50 mostra questa situazione; i simboli evidenziati in grigio sono VLAN compatibili, quelli bianchi non lo sono. Per semplicità tutti gli switch sono considerati VLAN compatibili; in caso contrario il primo switch LAN compatibile può aggiungere le etichette in base agli indirizzi IP o MAC.

In questa figura, le schede Ethernet VLAN compatibili generano direttamente frame etichettati secondo lo standard 802.1Q e la commutazione utilizza queste etichette. Per eseguire la commutazione gli switch devono sapere quali VLAN sono raggiungibili attraverso

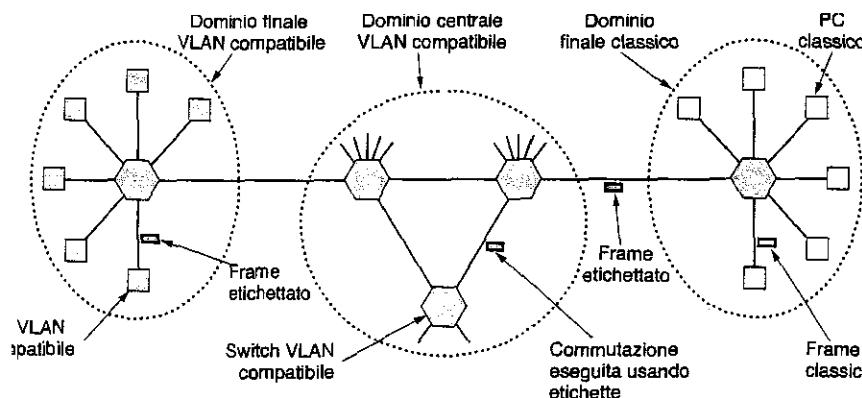


Figura 4.50. Transizione da Ethernet originale a Ethernet compatibile con VLAN. I simboli evidenziati in grigio sono VLAN compatibili. Quelli bianchi non lo sono.

so ogni porta, proprio come prima. Sapere che un frame appartiene alla VLAN grigia non aiuta molto, se non si sa a quali porte sono collegate le macchine della VLAN grigia. Di conseguenza lo switch ha bisogno di una tabella indicizzata in base alle VLAN che indica le porte da utilizzare e se sono oppure no compatibili con VLAN.

Quando un vecchio PC invia un frame a uno switch VLAN compatibile, lo switch costruisce un nuovo frame dotato di etichette basandosi sulla sua conoscenza della VLAN del mittente (usando la porta, l'indirizzo MAC o l'indirizzo IP). Da quel punto in poi, non ha più importanza la compatibilità della macchina trasmittente. In modo analogo, uno switch che deve inviare un frame etichettato a una vecchia macchina deve riportare il frame al suo formato originale prima di trasmettere i dati.

È utile dare uno sguardo al formato di frame 802.1Q mostrato nella Figura 4.51. L'unica modifica è l'aggiunta di un paio di campi lunghi 2 Byte. Il primo è *VLAN protocol ID*; il suo valore è sempre 0x8100. Poiché questo numero è maggiore di 1.500, tutte le schede Ethernet lo interpretano come un tipo e non come una lunghezza. Il modo in cui le vecchie schede di rete gestiscono questo valore è teorico, in quanto tali frame non sono stati progettati per essere inviati alle schede non compatibili.

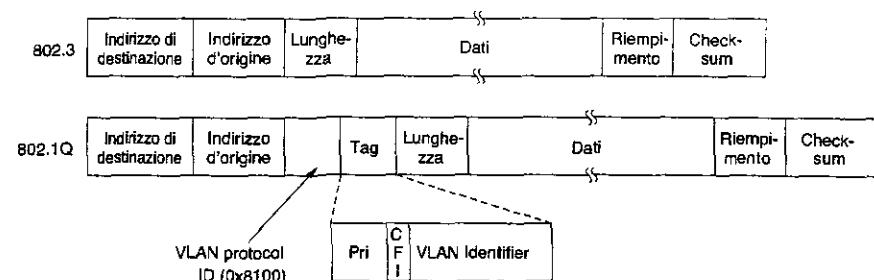


Figura 4.51. I formati di frame di Ethernet 802.3 (originale) e di Ethernet 802.1Q.

Il secondo campo di 2 byte contiene tre sottocampi. Il principale è *VLAN Identifier* che occupa i 12 bit di ordine più basso. Questa è l'informazione più importante che permette di identificare la VLAN di appartenenza del frame. Il campo di 3 bit *priority* non ha nulla a che fare con le VLAN, ma visto che la modifica dell'intestazione Ethernet è un evento raro che ha richiesto tre anni di lavoro e lo sforzo di centinaia di persone, gli sviluppatori hanno pensato bene di aggiungere qualche funzionalità in più. Questo campo permette di distinguere il traffico pesante trasmesso in tempo reale, quello leggero trasmesso in tempo reale e quello insensibile al tempo, e quindi di migliorare la qualità del servizio su una Ethernet. È necessario per la trasmissione della voce attraverso Ethernet (comunque IP ha avuto un campo simile per un quarto di secolo e nessuno lo ha mai utilizzato).

L'ultimo bit si chiama *CFI* (*Canonical Format Indicator*), ma sarebbe stato meglio chiamarlo *CEI* (*Corporate Ego Indicator*). In origine doveva distinguere gli indirizzi MAC più significativi da quelli meno significativi, ma quell'uso è andato perso in altre controversie. La sua presenza ora indica che il carico utile contiene un frame 802.5 liofilizzato che sta cercando di raggiungere un'altra LAN 802.5 passando attraverso una Ethernet. Tutta que-

sta disposizione, naturalmente, non ha nulla a che fare con le VLAN. Ma la politica del comitato che lavora sugli standard funziona come la politica tradizionale: se voti per il mio bit io voterò per il tuo.

Come è stato detto precedentemente, quando un frame etichettato raggiunge uno switch VLAN compatibile, lo switch utilizza l'ID della VLAN come indice per trovare nella tabella la porta da utilizzare per la trasmissione. Ma chi costruisce la tabella? Una costruzione manuale comporterebbe una configurazione diretta dei bridge. La bellezza del bridge trasparente è che si tratta di un dispositivo plug-and-play che non richiede alcuna configurazione manuale, quindi perdere questa proprietà sarebbe terribile. Per fortuna anche i bridge VLAN compatibili sono in grado di autoconfigurarsi in base all'osservazione delle etichette di passaggio. Se un frame etichettato come VLAN 4 entra dalla porta 3, allora apparentemente qualche macchina collegata alla porta 3 si trova sulla VLAN 4. Lo standard 802.1Q spiega come costruire dinamicamente le tabelle, soprattutto referenziando parti appropriate dell'algoritmo di Perlman standardizzato in 802.1D.

Prima di concludere l'argomento dell'instradamento VLAN è utile fare un'ultima osservazione. Molte persone che lavorano nel settore Internet ed Ethernet sostengono in modo fanatico la tecnologia non orientata alla connessione, e si oppongono violentemente a qualsiasi tipo di connessione negli strati data link e network. Tuttavia le VLAN introducono qualcosa che è sorprendentemente simile a una connessione. Per utilizzare correttamente le VLAN, ogni frame trasporta un nuovo identificatore speciale che, utilizzato come indice di una tabella memorizzata nello switch, permette di scoprire la destinazione del frame. Questo è esattamente ciò che accade nelle reti orientate alle connessioni. Nelle reti non orientate alla connessione l'instradamento utilizza l'indirizzo di destinazione, non un identificatore della connessione. L'argomento verrà esaminato in dettaglio nel Capitolo 5.

4.8 Sommario

Alcune reti hanno un solo canale impiegato per tutta la comunicazione. In queste reti il problema chiave è l'assegnazione del canale alle diverse stazioni che competono per poterlo utilizzare. Sono stati escogitati numerosi algoritmi di allocazione, e la Figura 4.52 mostra un riepilogo dei metodi più importanti.

Gli schemi di allocazione più semplici sono FDM e TDM, che sono efficienti quando il numero di stazioni è piccolo e costante e il traffico è continuo. Entrambi sono ampiamente usati sotto queste condizioni, per esempio per dividere la banda sulle linee telefoniche. Quando il numero di stazioni aumenta o è variabile, e i dati sono trasmessi a cadenza irregolare, FDM e TDM non funzionano bene. Il protocollo ALOHA, normale e slotted, è stato proposto come alternativa. Assieme alle sue numerose varianti e derivati è stato ampiamente descritto, analizzato e utilizzato nei sistemi reali.

Quando è possibile analizzare lo stato del canale, le stazioni possono evitare di iniziare una trasmissione mentre un'altra stazione ha già iniziato a trasmettere. Questa tecnica, basata sulla capacità di rilevamento della portante, ha dato vita a diversi protocolli che possono essere utilizzati sia su LAN sia su MAN.

Metodo	Descrizione
FDM	Dedica una banda di frequenza a ogni stazione
WDM	Schema FDM dinamico per la fibra
TDM	Dedica un intervallo temporale a ogni stazione
ALOHA puro	Trasmissione non sincronizzata che può iniziare in qualunque istante
ALOHA slotted	Trasmissione casuale in intervalli di tempo ben definiti
CSMA 1-persistente	Capacità standard di rilevamento della portante ad accesso multiplo
CSMA non persistente	Ritardo casuale quando il canale è occupato
CSMA P-persistente	CSMA, ma con una probabilità p di persistenza
CSMA/CD	CSMA, con interruzione in caso di collisione
Mappa di bit	Pianificazione a turno mediante una mappa di bit
Conteggio binario	Selezione la stazione pronta associata al numero più alto
Tree walk	Contesa ridotta mediante attivazione selettiva
MACA, MACAW	Protocolli LAN wireless
Ethernet	CSMA/CD con backoff esponenziale binario
FHSS	Spettro distribuito a frequenza variabile
DSSS	Spettro distribuito a sequenza diretta
CSMA/CA	Capacità di rilevamento della portante ad accesso multiplo con annullamento delle collisioni

Figura 4.52. Metodi e sistemi di assegnazione del canale per un canale comune.

È nota anche una classe di protocolli che elimina completamente la contesa o almeno la riduce in modo considerevole. Il conteggio binario elimina completamente la contesa. Il protocollo tree walk la riduce dividendo dinamicamente le stazioni in due gruppi disgiunti, uno dei quali ha il permesso di trasmettere e l'altro no; il protocollo tenta di creare la divisione in modo tale che solo una stazione pronta alla trasmissione riceve il permesso di procedere.

Le LAN wireless hanno i loro problemi con le relative soluzioni. Il problema maggiore è causato dalle stazioni nascoste, che impedisce a CSMA di funzionare. Una classe di soluzioni, rappresentata da MACA e MACAW, tenta di stimolare le trasmissioni intorno alla destinazione in modo da far funzionare meglio CSMA. Sono utilizzate anche le tecniche a spettro distribuito a frequenza variabile e sequenza diretta. IEEE 802.11 combina CSMA e MACAW per produrre CSMA/CA.

Ethernet è la forma dominante nelle reti locali. Utilizza CSMA/CD per assegnare il canale. Versioni più vecchie utilizzavano un cavo che si snodava da macchina a macchina, mentre ora sono più comuni i doppini collegati agli hub o agli switch. Le velocità, cresciute da 10 Mbps a 1 Gbps, continuano ad aumentare.

Le LAN wireless stanno diventando comuni, con lo standard 802.11 che domina il campo. Il suo strato fisico supporta cinque modalità di trasmissione, inclusa quella infrarossi, vari schemi di diffusione di spettro e un sistema FDM multicanale. Può operare con una sta-

zione base in ogni cella, ma può anche funzionare senza alcuna stazione base. Il protocollo è una variante di MACAW, con capacità di rilevamento della portante virtuale. Le MAN wireless stanno iniziando ad apparire. Sono sistemi a banda larga che usano le onde radio al posto dell'ultimo miglio delle connessioni telefoniche, e utilizzano le tecniche tradizionali di modulazione a banda stretta. La qualità del servizio è importante, e lo standard 802.16 definisce quattro classi principali (una a bit-rate costante, due a bit-rate variabili e una best effort).

Anche il sistema Bluetooth è wireless, ma è pensato per le connessioni a breve distanza come quelle che consentono al telefono cellulare di collegarsi all'auricolare, o al computer di comunicare con le periferiche senza far uso di cavi. Come nel caso di 802.11, utilizza una tecnica a spettro distribuito a frequenza variabile nella banda ISM. A causa dell'alto livello di rumore di molti ambienti e per la necessità di un'interazione in tempo reale, nei suoi protocolli è stato integrato un elaborato sistema di correzione degli errori. L'esistenza di molte LAN differenti fa nascere l'esigenza di un sistema per collegarle. I bridge e gli switch sono utilizzati a questo scopo. L'algoritmo spanning tree è utilizzato per costruire bridge plug-and-play. Le VLAN rappresentano un nuovo sviluppo nel mondo dell'interconnessione tra LAN; le VLAN separano la topologia logica delle LAN dalla loro topologia fisica. Un nuovo formato di frame Ethernet (802.1Q) è stato creato per facilitare l'introduzione delle VLAN nelle aziende.

Problemi

- Per questo problema si utilizza una formula di questo capitolo, che è necessario indicare. I frame arrivano in modo casuale a un canale a 100 Mbps per la trasmissione. Se al suo arrivo trova il canale occupato, il frame attende il suo turno in coda. La lunghezza del frame ha distribuzione esponenziale con una media di 10.000 bit per frame. Per ognuna delle seguenti frequenze di arrivo dei frame, indicare il ritardo incontrato dal frame medio includendo nel calcolo sia il tempo di accodamento sia quello di trasmissione.
 - 90 frame/sec.
 - 900 frame/sec.
 - 9000 frame/sec.
- Un gruppo di N stazioni condivide un canale ALOHA puro a 56 kbps. Ogni stazione trasmette un frame da 1.000 bit in media una volta ogni 100 sec, anche se il precedente non è stato ancora trasmesso (quindi le stazioni possono memorizzare in un buffer i frame in uscita). Qual è il valore massimo di N ?
- Si consideri il ritardo di ALOHA puro rispetto a slotted ALOHA nel caso di basso carico. Qual è il minore? Motivare la risposta.
- Diecimila stazioni di prenotazione di una linea aerea competono per utilizzare un singolo canale slotted ALOHA. La stazione media genera 18 richieste ogni ora. Un intervallo dura 125 msec. Qual è il carico totale approssimato del canale?

- Un gran numero di utenti ALOHA genera 50 richieste al secondo, comprendendo sia quelle originali sia le ritrasmissioni. Il tempo è diviso in unità di 40 msec.
 - Qual è la probabilità di un successo al primo tentativo?
 - Qual è la probabilità di avere esattamente k collisioni prima di un successo?
 - Qual è il numero atteso di tentativi richiesti?
- Misurazioni di un canale slotted ALOHA con un numero infinito di utenti mostrano che il 10% degli intervalli sono liberi.
 - Qual è il carico del canale, G ?
 - Qual è la capacità di trasporto?
 - Il canale è sovraccaricato oppure sottoutilizzato?
- In un sistema ALOHA slotted a popolazione infinita, il numero medio di intervalli che una stazione deve attendere tra una collisione e la ritrasmissione è 4. Disegnare la curva del ritardo in base alla capacità di trasporto di questo sistema.
- Quanto tempo deve attendere una stazione s , nel peggiore dei casi, prima di iniziare la trasmissione del suo frame attraverso una LAN che utilizza:
 - Il protocollo mappa di bit di base?
 - Il protocollo di Mok e Ward con permuta virtuale dei numeri di stazione?
- Una LAN utilizza la versione di Mok e Ward del conteggio binario. A un certo punto, le dieci stazioni hanno i seguenti numeri virtuali: 8, 2, 4, 5, 1, 7, 3, 6, 9 e 0. Le successive tre stazioni in trasmissione sono rispettivamente 4, 3 e 9. Quali saranno i nuovi numeri virtuali assegnati al termine delle tre trasmissioni?
- Sedici stazioni, numerate da 1 a 16, si contendono l'uso di un canale condiviso mediante il protocollo tree walk adattivo. Se tutte le stazioni che hanno per indirizzo un numero primo diventano di colpo pronte contemporaneamente, quanti intervalli di bit servono per risolvere la contesa?
- Un insieme di 2^n stazioni utilizza il protocollo tree walk adattivo per arbitrare l'accesso a un cavo condiviso. A un certo punto, due di loro diventano pronte. Qual è il numero minimo, massimo e medio di intervalli nell'albero da attraversare se $2^n >> 1$?
- Le LAN wireless studiate utilizzano protocolli di tipo MACA invece che CSMA/CD. In quali condizioni sarebbe possibile utilizzare CSMA/CD?
- Quali proprietà hanno in comune i protocolli di accesso al canale WDMA e GSM? (per GSM vedere il Capitolo 2)
- Sei stazioni, da A a F, comunicano usando il protocollo MACA. È possibile che due trasmissioni avvengano contemporaneamente? Motivare la risposta.
- La facciata di un edificio ha 15 uffici adiacenti per ogni piano. Ogni ufficio contiene una presa a muro per un terminale, perciò le prese formano una griglia rettangolare nel piano verticale con una distanza di 4 m tra ogni coppia di prese, sia orizzontalmente sia verticalmente. Supponendo che sia possibile stendere un cavo dritto tra ogni coppia di prese, orizzontalmente, verticalmente o diagonalmente, quanti metri di cavo servono per collegare tutte le prese usando:
 - Una configurazione a stella con un singolo router nel mezzo?
 - Una LAN 802.3?

6. Qual è il baud rate di Ethernet 10 Mbps standard?
7. Disegnare la codifica Manchester relativa al flusso di bit 0001110101.
8. Disegnare la codifica Manchester differenziale per il flusso di bit indicato nel problema precedente. Si supponga che la linea sia inizialmente nello stato basso.
9. Una LAN CSMA/CD (non 802.3) lunga 1 Km ha una velocità di propagazione pari a 200 m/msec. I ripetitori non sono ammessi in questo sistema. I frame di dati sono lunghi 256 bit, inclusi i 32 bit utilizzati per l'intestazione, checksum e gli altri codici di controllo. Il primo intervallo di bit dopo una trasmissione che ha avuto successo è riservato al ricevitore, che può così catturare il canale e inviare un frame di acknowledgement lungo 32 bit. Qual è il data rate effettivo, escludendo il codice di controllo, supponendo che non ci siano collisioni?
10. Due stazioni CSMA/CD stanno tentando di trasmettere lunghi file (multiframe). Dopo aver trasmesso ogni frame, le stazioni si contendono il canale usando l'algoritmo di backoff esponenziale binario. Qual è la probabilità che la contesa termini in k round? Qual è il numero medio di round per periodo di contesa?
11. Si consideri una rete CSMA/CD a 1 Gbps formata da un cavo lungo 1 Km senza ripetitori. La velocità di segnale nel cavo è di 200.000 km/sec. Qual è la dimensione minima del frame?
12. Un pacchetto IP che sta per essere trasmesso da Ethernet è lungo 60 byte, incluse tutte le sue intestazioni. Se non si usa LLC, è necessario che il frame venga riempito automaticamente con dati cuscinetto? Se sì, quanti byte devono essere aggiunti?
13. I frame Ethernet devono essere lunghi almeno 64 byte per essere certi che il trasmettitore rilevi una collisione all'altro capo del cavo. Fast Ethernet ha la stessa dimensione minima di 64 byte, ma può trasmettere i bit dieci volte più velocemente. Come è possibile mantenere la stessa dimensione minima del frame?
14. Alcuni testi riportano la dimensione media di un frame Ethernet come 1.518 byte invece di 1.600 byte. È un errore? Spiegare la risposta.
15. Le specifiche 1000Base-SX affermano che il clock deve funzionare a 1.250 MHz, anche se Ethernet gigabit è stata progettata per funzionare a 1 Gbps. La maggiore velocità è stata definita per fornire un margine extra di sicurezza? Se no, qual è il motivo?
16. Quanti frame al secondo può gestire una Ethernet gigabit? Fare molta attenzione e considerare tutti i casi rilevanti. Suggerimento: è importante il fatto che si tratti di Ethernet gigabit.
17. Dire il nome di due reti che consentono di impacchettare i frame uno in fila all'altro. Perché questa funzionalità è importante?
18. Nella Figura 4.27 sono mostrate quattro stazioni, A, B, C e D. Quale delle ultime due stazioni ritenevi più vicina ad A, e perché?
19. Si supponga che una LAN 802.11b a 11 Mbps stia trasmettendo frame da 64 byte uno dopo l'altro attraverso un canale radio con una frequenza di errore pari a 10^{-7} . Quanti frame al secondo verranno danneggiati in media?
20. Una rete 802.16 ha un'ampiezza di canale di 20 MHz. Quanti bit/sec possono essere inviati a una stazione utente?



31. IEEE 802.16 supporta quattro classi di servizio. Quale classe di servizio è più adatta alla trasmissione di video non compresso?
32. Indicare due motivi per cui le reti potrebbero utilizzare un codice di correzione degli errori invece di rilevare gli errori e ritrasmettere.
33. La Figura 4.35 mostra che un dispositivo Bluetooth può trovarsi contemporaneamente in due piconet. C'è un motivo che impedisce a un dispositivo di essere contemporaneamente il nodo master di entrambe le piconet?
34. La Figura 4.25 mostra diversi protocolli di strato fisico. Quale di questi è il più vicino al protocollo di strato fisico Bluetooth? Qual è la più grande differenza tra i due?
35. Bluetooth supporta due tipi di link tra nodo master e nodo slave. Quali sono e in quali situazioni sono utilizzati?
36. I frame di segnalazione nella variante della tecnica a spettro distribuito a frequenza variabile adottata da 802.11 contengono il tempo di rotazione. Anche gli analoghi frame di segnalazione di Bluetooth contengono il tempo di rotazione? Spiegare la risposta.
37. Si considerino le LAN interconnesse mostrate nella Figura 4.44. Si supponga che gli host *a* e *b* si trovino sulla LAN 1, che *c* sia sulla LAN 2 e che *d* si trovi sulla LAN 8. Inizialmente le tabelle di hash in tutti i bridge sono vuote ed è utilizzata la struttura spanning tree mostrata nella Figura 4.44(b). Mostrare in che modo cambiano le tabelle di hash dei diversi bridge dopo ognuno di questi eventi:
 - (a) *a* trasmette a *d*;
 - (b) *c* trasmette a *a*;
 - (c) *d* trasmette a *c*;
 - (d) *d* si sposta sulla LAN 6;
 - (e) *d* trasmette ad *a*.
38. Quando si utilizza una struttura spanning tree per inoltrare i frame in una LAN estesa può accadere che alcuni bridge non partecipino all'inoltro dei dati. Identificare nella Figura 4.44 tre bridge di questo tipo. C'è qualche motivo per cui è necessario tenere questi tre bridge anche quando non sono utilizzati per l'inoltro?
39. Si supponga che uno switch abbia schede di linea per quattro linee di input. Accade spesso che un frame in arrivo su una delle linee debba uscire su un'altra linea che si trova sulla stessa scheda. Quali scelte ha fatto il progettista dello switch per affrontare questa situazione?
40. Uno switch progettato per funzionare con fast Ethernet ha un backplane che può funzionare a 10 Gbps. Quanti frame al secondo può gestire nel peggiore dei casi?
41. Si consideri la rete mostrata nella Figura 4.49(a). Se la macchina *J* diventasse improvvisamente bianca, sarebbe necessario apportare qualche modifica alle etichette? Se sì, quale?
42. Descrivere brevemente la differenza tra gli switch store-and-forward e cut-through.
43. Gli switch store-and-forward hanno un vantaggio rispetto ai dispositivi a cut-through relativamente ai frame danneggiati, qual è?

Per far funzionare le VLAN, servono tabelle di configurazione negli switch e nei bridge. Che cosa accade se le VLAN mostrate nella Figura 4.49(a) utilizzano hub al posto di cavi multidrop? Anche gli hub hanno bisogno di tabelle di configurazione? Perché?

Nella Figura 4.50 lo switch nel vecchio dominio mostrato sulla destra è uno switch VLAN compatibile. Sarebbe possibile utilizzare switch classici in questo caso? Se sì, come funzionerebbe il sistema? Se no, perché no?

Scrivere un programma che simuli il comportamento del protocollo CSMA/CD attraverso Ethernet quando ci sono N stazioni pronte a trasmettere mentre un frame è già in trasmissione. Il programma dovrebbe indicare il momento in cui ogni stazione inizia a trasmettere con successo il proprio frame. Si supponga che il clock avanza ad ogni intervallo di tempo (51,2 microsecondi) e che il rilevamento di una collisione e la trasmissione di una sequenza di disturbo richiedano un intervallo di tempo. Tutti i frame hanno una lunghezza pari alla lunghezza massima consentita.

5

Lo strato network

Lo strato network si occupa del trasporto dei pacchetti lungo tutto il cammino percorso dall'origine alla destinazione finale, che per essere raggiunta può richiedere molti salti attraverso i router intermedi lungo il tragitto. Questa funzione è chiaramente distinta da quella dello strato data link, che ha il più modesto obiettivo di spostare semplicemente i frame da un capo all'altro del cavo. Di conseguenza lo strato network è lo strato più basso a occuparsi della trasmissione da punto a punto.

Per raggiungere i suoi obiettivi, lo strato network deve conoscere la topologia della sottorete di comunicazione (ossia l'insieme di tutti i router) e scegliere i percorsi appropriati attraverso di essa. Nel compiere la scelta, deve anche evitare di sovraccaricare alcune linee di comunicazione lasciando altre completamente libere. Deve infine occuparsi dei nuovi problemi che nascono quando la sorgente e la destinazione si trovano in reti diverse. Questo capitolo analizza tutti questi argomenti, usando come riferimento Internet e il suo protocollo di strato network, IP; la trattazione si occuperà anche delle reti wireless.

5.1 Problemi dell'architettura dello strato network

I prossimi paragrafi introducono alcuni problemi che vanno affrontati dai progettisti dello strato network, tra cui il servizio fornito allo strato trasporto e la struttura interna della sottorete.

1.1 Comutazione di pacchetto store-and-forward

Prima d'iniziare a spiegare i dettagli dello strato network è bene ricordare il contesto in cui operano i protocolli dello strato network, rappresentato nella Figura 5.1. I componenti principali del sistema sono gli apparecchi dell'operatore di telecomunicazioni (i router collegati alle linee di trasmissione) mostrati dentro l'ellisse ombreggiata, e i dispositivi degli utenti (disegnati all'esterno dell'ellisse). L'host *H1* è collegato direttamente a uno dei router dell'operatore *A* attraverso una linea affittata. Al contrario, *H2* si trova su una LAN che ha un router *F* posseduto e gestito dal cliente; anche a questo router fa capo una linea affittata diretta alle apparecchiature dell'operatore di telecomunicazioni. *F* è disegnato all'esterno dell'ellisse perché non appartiene all'operatore, ma in termini di costruzione, software e protocolli, probabilmente non è diverso dai router dell'operatore telefonico. Si può discutere sul fatto che appartenga alla sottorete, ma per quanto riguarda gli scopi di questo capitolo i router nei locali dei clienti sono considerati come parte della sottorete perché utilizzano gli stessi algoritmi dei router dell'operatore (e il tema principale di questo capitolo riguarda proprio gli algoritmi).

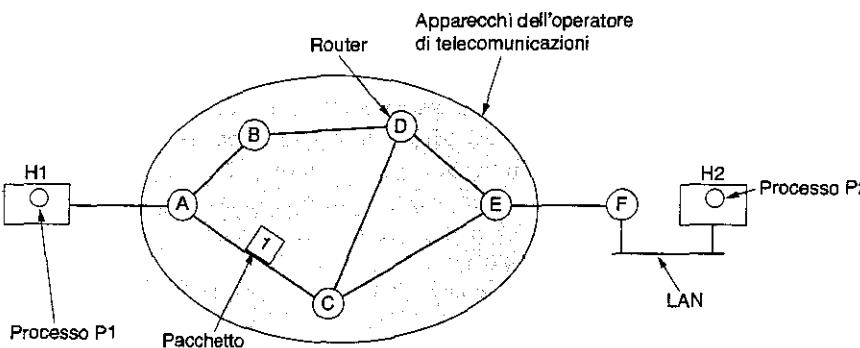


Figura 5.1. L'ambiente dei protocolli dello strato network.

Li apparecchi sono utilizzati in questo modo: un host con un pacchetto da trasmettere invia i dati al router più vicino attraverso la sua stessa LAN, oppure attraverso un collegamento punto-punto con l'operatore di telecomunicazioni. Qui il pacchetto viene memorizzato fino a quando non è interamente arrivato, per verificare il checksum; quindi viene oltrato al router successivo che si trova lungo il percorso, fino a quando non raggiunge host di destinazione dove viene consegnato. Questo meccanismo, come è stato spiegato nei capitoli precedenti, si chiama commutazione di pacchetto store-and-forward.

1.2 Servizi forniti allo strato trasporto

Lo strato network fornisce servizi allo strato trasporto attraverso l'interfaccia strato network/strato trasporto. È importante capire che tipo di servizi fornisce lo strato network allo strato trasporto. I servizi dello strato network sono stati progettati tenendo in mente quegli obiettivi:

1. i servizi non dovrebbero essere legati alla tecnologia del router
2. allo strato trasporto dovrebbero essere nascosti dettagli quali il numero, il tipo e la topologia dei router
3. gli indirizzi di rete disponibili allo strato trasporto dovrebbero utilizzare uno schema di numerazione uniforme, anche attraverso le LAN e le WAN.

Dati questi obiettivi, i progettisti dello strato network hanno piena libertà di definire le specifiche dettagliate dei servizi da offrire allo strato di trasporto, che spesso degenera in una furiosa battaglia tra due fazioni in lotta. Il nocciolo della questione è se lo strato network dovrebbe fornire un servizio orientato alle connessioni oppure un servizio senza connessione.

Una fazione (rappresentata dalla comunità Internet) afferma che il lavoro dei router è spostare i pacchetti da un punto all'altro e niente di più. Nella loro visione (basata su 30 anni di esperienza reale con una rete di computer reale e operativa) la sottorete è intrinsecamente inaffidabile, indipendentemente da come è stata progettata, quindi gli host dovrebbero accettare questo fatto ed eseguire da soli il controllo degli errori (ossia, rilevamento e correzione degli errori) e il controllo del flusso.

Questo punto di vista conduce rapidamente alla conclusione che il servizio di rete non dovrebbe essere orientato alle connessioni, con primitive come SEND PACKET e RECEIVE PACKET e poco più. In particolare, non si dovrebbe eseguire alcuna operazione di ordinamento dei pacchetti e di controllo di flusso: ci sono già gli host che si occupano di questo aspetto, e di solito fare due volte la stessa operazione non aiuta un granché. Inoltre, ogni pacchetto deve contenere l'intero indirizzo di destinazione, perché ogni pacchetto che viene trasmesso è trasportato in modo indipendente dai suoi eventuali predecessori.

L'altra fazione (rappresentata dalle aziende telefoniche) afferma che la sottorete dovrebbe fornire un servizio affidabile orientato alle connessioni. Sostengono che l'esperienza garantita da 100 anni di successi del sistema telefonico mondiale offre una guida eccellente. In questa visione la qualità del servizio è il fattore dominante, ma se nella sottorete non sono presenti connessioni è molto difficile da ottenere, specialmente per il traffico in tempo reale (per esempio audio e video).

Queste due fazioni sono perfettamente rappresentate da Internet e da ATM. Internet offre un servizio di strato network non orientato alle connessioni, al contrario delle reti ATM. Comunque è interessante notare che Internet si sta evolvendo per effetto della crescente importanza del garantire il servizio; in particolare, sta cominciando ad acquisire proprietà normalmente associate al servizio orientato alle connessioni (come verrà spiegato più avanti). Abbiamo già avuto una vaga idea di questa evoluzione durante lo studio delle VLAN nel Capitolo 4.

5.1.3 Implementazione del servizio senza connessione

Dopo aver indicato le due classi di servizio che lo strato network può fornire ai suoi utenti, è giunto il momento di vedere in che modo funziona questo strato al suo interno. Sono possibili due diverse organizzazioni, che dipendono dal tipo di servizio offerto. Se il ser-

vizio è senza connessione, i pacchetti sono inoltrati nella sottorete individualmente e instradati indipendentemente l'uno dall'altro; non occorre configurazione anticipata. In questo contesto i pacchetti sono chiamati spesso **datagramma** o **datagram** (è evidente l'analogia con i telegrammi) e la sottorete è chiamata **sottorete a datagrammi**. Se il servizio è orientato alle connessioni, prima di inviare i pacchetti si deve stabilire un percorso che collega il router sorgente al router destinazione. Questa connessione è chiamata **CV** (circuito virtuale), in analogia con i circuiti fisici configurati dal sistema telefonico, e la sottorete è chiamata **sottorete a circuito virtuale**. Questo paragrafo esamina le sottoreti a datagrammi, mentre il successivo si occuperà di quelle a circuito virtuale.

Vediamo come funziona una sottorete a datagrammi. Si supponga che il processo *P1* nella Figura 5.2 abbia un lungo messaggio per *P2*. Il processo sorgente porge il messaggio allo strato di trasporto dando istruzioni di inoltrarlo al processo *P2* che si trova sull'host *H2*. Il codice dello strato di trasporto in esecuzione in *H1*, tipicamente dentro il sistema operativo, aggiunge un'intestazione di trasporto all'inizio del messaggio e passa il risultato allo strato network, rappresentato probabilmente da un'altra procedura del sistema operativo. Si supponga che il messaggio sia lungo quattro volte la dimensione massima del pacchetto; in questo caso, lo strato network deve suddividere il messaggio in quattro pacchetti, 1, 2, 3 e 4, e inviare un pacchetto dopo l'altro al router *A* usando un protocollo punto-punto, per esempio PPP. A questo punto l'operatore di telecomunicazioni assume il controllo dell'operazione. Ogni router ha una tabella interna che indica dove devono essere inviati i

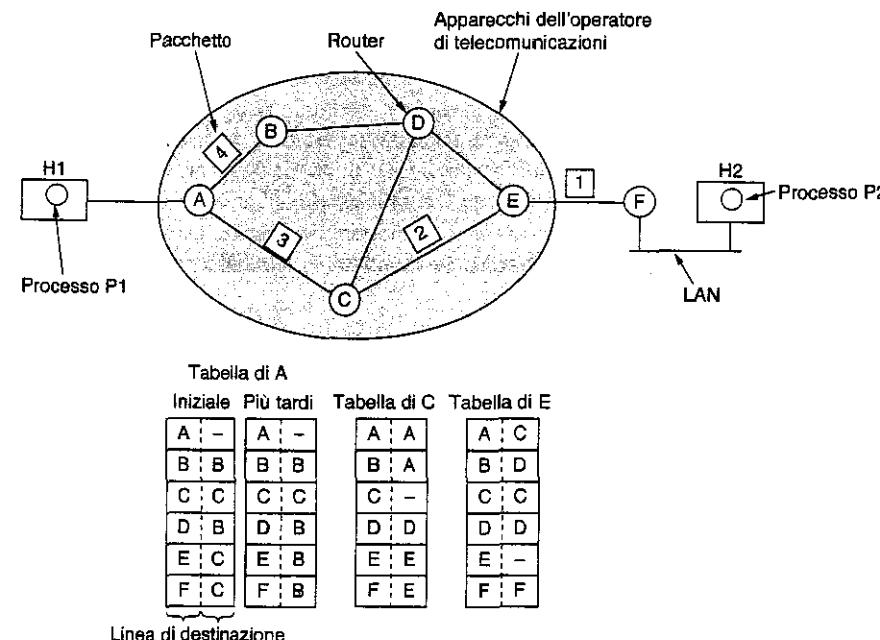


Figura 5.2. Routing con una sottorete a datagrammi.

pacchetti diretti a ogni possibile destinazione. Ogni voce della tabella è costituita da una coppia di valori, che rappresentano una destinazione e la linea di trasmissione da utilizzare per raggiungere tale destinazione. Possono essere utilizzate solo linee collegate direttamente. Per esempio, nella Figura 5.2, *A* ha solo due linee in uscita, *B* e *C*, perciò ogni pacchetto in arrivo deve essere inviato a uno di questi due router, anche se la destinazione finale si trova su qualche altro router. La tabella di routing iniziale di *A* è mostrata nella figura sotto la voce "Iniziale".

Non appena raggiungono *A*, i pacchetti 1, 2 e 3 sono archiviati temporaneamente (per verificare il checksum), e poi ogni pacchetto viene inoltrato verso *C* secondo la tabella di *A*. Il pacchetto 1 viene quindi trasmesso a *E* e poi a *F*. Quando arriva a *F*, il pacchetto è encapsulato in un frame dello strato data link e inviato ad *H2* attraverso la LAN. I pacchetti 2 e 3 seguono lo stesso percorso.

Al pacchetto 4, invece, accade qualcosa di diverso. Quando raggiunge *A* viene inviato al router *B*, anche se la destinazione finale è ancora *F*. Per qualche motivo, *A* ha deciso di trasmettere il pacchetto 4 attraverso un percorso diverso da quello seguito dai primi 3 pacchetti. Forse, dopo aver rilevato un ingorgo di traffico da qualche parte lungo il percorso *ACE*, ha aggiornato la sua tabella di routing, come indicato dai valori riportati sotto l'etichetta "Più tardi". L'algoritmo che gestisce le tabelle e prende le decisioni d'instradamento è chiamato **algoritmo di routing**. Gli algoritmi di routing sono uno degli argomenti principali trattati in questo capitolo.

5.1.4 Implementazione del servizio orientato alla connessione

Nel caso di un servizio orientato alle connessioni ci serve una sottorete a circuito virtuale; vediamo come funziona. L'idea alla base dei circuiti virtuali è quella di evitare di dover scegliere una nuova strada per ogni pacchetto inviato, come nella Figura 5.2. Invece, quando è stabilita una connessione, il percorso dal computer sorgente a quello di destinazione viene scelto durante l'impostazione della connessione, e archiviato nelle tabelle dei router. Quel percorso è utilizzato per tutto il traffico che scorre attraverso la connessione, con funzionamento identico a quello del sistema telefonico. Quando la connessione viene rilasciata, anche il circuito virtuale viene terminato. Con un servizio orientato alle connessioni, ogni pacchetto contiene un identificatore che indica il circuito virtuale di appartenenza.

Come esempio si consideri la situazione mostrata nella Figura 5.3. In questo caso, l'host *H1* ha stabilito la connessione 1 con l'host *H2*. La connessione è memorizzata nella prima voce delle tabelle di routing. La prima linea della tabella di *A* dice che, se da *H1* arriva un pacchetto contenente l'identificatore di connessione numero 1, allora i dati devono essere inviati al router *C* assegnando ancora l'identificatore 1. In modo analogo, la prima voce della tabella di *C* instrada il pacchetto a *E*, ancora con l'identificatore di connessione numero 1. Che cosa accade se anche *H3* cerca di stabilire una connessione con *H2*? L'host sceglie l'identificatore di connessione numero 1 (perché sta iniziando la connessione e questa è la sua unica connessione) e chiede alla sottorete di stabilire un circuito virtuale. Questo conduce alla seconda riga della tabella: in questo caso si verifica un conflitto, perché sebbene *A* possa facilmente distinguere i pacchetti della connessione 1 provenienti da *H1* dai pac-

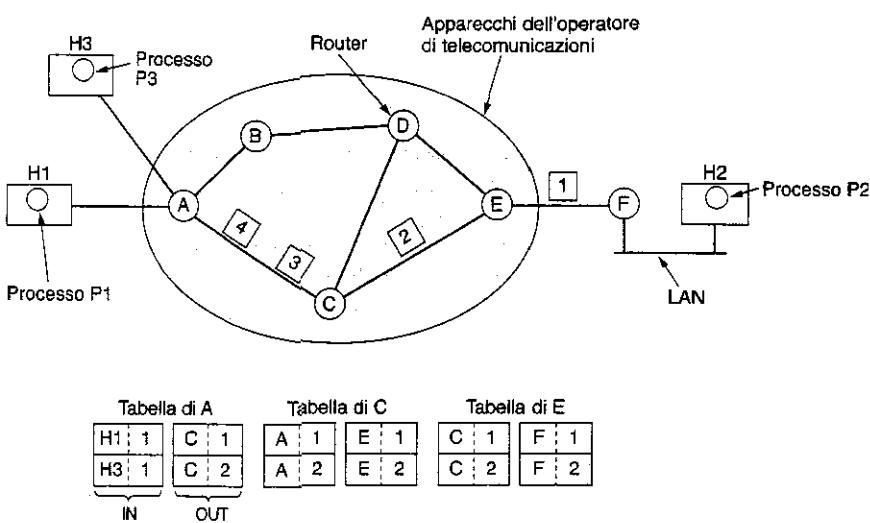


Figura 5.3. Routing con una sottorete a circuito virtuale.

menti della connessione 1 provenienti da H_3 , C non è in grado di farlo. Per questo motivo, A assegna un diverso identificatore di connessione al traffico in uscita relativo alla seconda connessione.

I router è stata data la capacità di sostituire gli identificatori di connessione nei pacchetti in uscita proprio per evitare conflitti di questo tipo. In alcuni contesti, questa capacità è chiamata commutazione di etichetta.

5.1.5 Confronto tra sottoreti a circuito virtuale e a datagramma

I circuiti virtuali e datagrammi hanno i loro sostenitori e detrattori; tenteremo ora di riappareggiare gli argomenti di entrambi. I concetti principali sono elencati nella Figura 5.4, anche se i puristi, probabilmente, potrebbero trovare un controsenso per ciascun elemento indicato.

Il l'interno della sottorete esistono diversi compromessi tra circuiti virtuali e datagrammi. Un compromesso va fatto tra spazio di memoria del router e banda. I circuiti virtuali permettono ai pacchetti di utilizzare i numeri di circuito al posto degli indirizzi di destinazione completi. Se i pacchetti tendono a essere corti, un indirizzo di destinazione completo in ogni pacchetto può aumentare in modo significativo l'overhead e di conseguenza preoccupare la banda. Il prezzo da pagare per l'utilizzo interno di circuiti virtuali è rappresentato dallo spazio delle tabelle dei router. Al cambiare del costo relativo dei circuiti di comunicazione e della memoria dei router può essere più economico l'uno o l'altro.

Un altro compromesso è tra il tempo per la configurazione del circuito e il tempo di analisi dell'indirizzo. L'utilizzo di circuiti virtuali richiede una fase di configurazione che impiega tempo e consuma risorse, ma dopo è facile capire che cosa si deve fare con un pac-

Problema	Sottorete a datagrammi	Sottorete a circuito virtuale
Impostazione circuito	Non è necessaria	È richiesta
Indirizzamento	Ogni pacchetto contiene l'indirizzo completo di destinazione e quello di origine	Ogni pacchetto contiene un numero CV corto
Informazioni sullo stato	I router non conservano informazioni sullo stato delle connessioni	Ogni CV richiede spazio nella tabella del router per la connessione
Routing	Ogni pacchetto è instradato indipendentemente	Il percorso è scelto durante l'impostazione del CV. Tutti i pacchetti seguono lo stesso percorso
Effetto dei guasti nel router	Nessuno, tranne per i pacchetti perduti durante il guasto	Tutti i CV che passano attraverso il router guasto sono terminati
Qualità del servizio	Difficile	Facile se è possibile assegnare in anticipo abbastanza risorse a ogni CV
Controllo della congestione	Difficile	Facile se è possibile assegnare in anticipo abbastanza risorse a ogni CV

Figura 5.4. Confronto tra sottoreti a datagrammi e a circuito virtuale.

chetto dati in una sottorete a circuito virtuale: il router usa semplicemente il numero di circuito come indice nella tabella per trovare la destinazione dei dati. In una sottorete a datagrammi, per individuare la voce relativa alla destinazione serve una procedura di ricerca più complicata.

Un altro problema è la quantità di spazio di tabella richiesto nella memoria del router. Una sottorete a datagrammi ha bisogno di una voce per ogni possibile destinazione, mentre una sottorete a circuito virtuale si accontenta di una voce per ogni circuito virtuale. In realtà questo vantaggio è un po' illusorio, poiché anche i pacchetti utilizzati per impostare la connessione devono essere instradati usando l'indirizzo di destinazione, proprio come avviene con i datagrammi.

I circuiti virtuali sono in vantaggio quando si tratta di dare garanzie sulla qualità del servizio ed evitare le congestioni all'interno della sottorete, perché le risorse (per esempio i buffer, la banda e i cicli di CPU) possono essere riservate in anticipo, al momento di stabilire la connessione. Una volta che i pacchetti iniziano ad arrivare, la banda necessaria e la capacità del router sono già pronte. Nella sottorete a datagrammi è più difficile evitare la congestione del traffico.

Per i sistemi che elaborano transazioni (come quelli utilizzati dai negozi per verificare gli acquisti con carta di credito), l'overhead richiesto per impostare e rilasciare un circuito virtuale può facilmente superare quello di utilizzo vero e proprio del circuito. Se si pensa che la maggior parte del traffico sarà di questo tipo, ha poco senso utilizzare i circuiti virtuali all'interno della sottorete. D'altra parte, in queste situazioni possono dimostrarsi molto utili i circuiti virtuali permanenti, che sono impostati manualmente e durano mesi o anni.

I circuiti virtuali hanno anche un problema di vulnerabilità. Se un router va in crash e perde tutti i dati nella sua memoria, tutti i circuiti virtuali che passano attraverso l'appa-

recchio devono essere terminati, anche se questo si riattiva un solo secondo dopo. Al contrario, se un router per datagrammi si blocca soffrono solo gli utenti che in quel momento avevano pacchetti accodati nel router, e forse nemmeno tutti (dipende dagli acknowledgement già trasmessi). La perdita della linea di comunicazione è fatale per i circuiti virtuali che la utilizzano, ma può essere facilmente compensata quando si usano datagrammi, che inoltre permettono ai router di bilanciare il traffico attraverso la sottorete, poiché i percorsi possono essere cambiati in corsa, durante una lunga sequenza di trasmissione.

5.2 Algoritmi di routing

La funzione principale dello strato network è quella d'instradare i pacchetti dal computer sorgente al computer di destinazione. Nella maggior parte delle sottoreti i pacchetti compiono salti multipli per raggiungere la meta; l'unica eccezione degna di nota riguarda le reti broadcast, ma anche in questo caso se sorgente e destinazione non si trovano sulla stessa rete il routing è un problema. Gli algoritmi che scelgono i percorsi, e le strutture dati che usano, sono una parte fondamentale dell'architettura dello strato network.

L'algoritmo di routing è quella parte del software dello strato network che si preoccupa di scegliere lungo quale linea di uscita vanno instradati i pacchetti in arrivo. Se la sottorete utilizza internamente i datagrammi questa decisione va ripetuta per ogni pacchetto dati in arrivo, poiché il percorso migliore può cambiare nel tempo. Se la sottorete utilizza internamente i circuiti virtuali, le decisioni che riguardano il routing vengono prese solo quando viene impostato un nuovo circuito virtuale. Da quel momento in poi, i pacchetti con i dati seguono semplicemente la rotta prestabilita. Il secondo caso è qualche volta chiamato **routing di sessione**, perché un percorso rimane valido per tutta la sessione utente (per esempio, durante l'accesso a un terminale o durante il trasferimento di un file).

Qualche volta è utile distinguere tra instradamento o routing (che sceglie la strada da seguire) e inoltro (che indica l'operazione eseguita all'arrivo di un pacchetto). È come se i router contenesse due processi al suo interno. Uno dei due gestisce ogni pacchetto che arriva, cercando nella tabella di routing la linea di trasmissione più adatta; questo processo è chiamato **inoltro**. L'altro processo si occupa di riempire e aggiornare le tabelle di inoltro, ed è qui che entrano in gioco gli algoritmi di routing.

Ci sono proprietà dell'algoritmo di routing sono desiderabili indipendentemente dal fatto che i percorsi siano scelti indipendentemente per ogni pacchetto o stabiliti durante l'impostazione di una nuova connessione: precisione, semplicità, robustezza, stabilità, imparzialità e ottimizzazione.

Precisione e semplicità non richiedono commenti, ma la necessità di robustezza può essere meno ovvia a prima vista. Ci si aspetta che ogni nuova grande rete continui a funzionare per anni, senza che si verifichino guasti capaci di coinvolgere l'intero sistema. Durante quel periodo ci saranno guasti hardware e software di tutti i tipi; host, router e rete si guarteranno ripetutamente e la topologia subirà molte modifiche. L'algoritmo di routing dovrebbe essere in grado di far fronte ai cambiamenti di topologia e di traffico, senza che sia necessario interrompere tutto il lavoro in tutti gli host e riavviare la rete ogni volta che un router si blocca.

Anche la stabilità è un obiettivo importante per l'algoritmo di routing. Esistono algoritmi di routing che non convergono mai all'equilibrio, anche se eseguiti per molto tempo. Un algoritmo stabile raggiunge l'equilibrio e lì rimane. Imparzialità e ottimizzazione possono sembrare qualità ovvie (sicuramente nessuna persona ragionevole cercherebbe di opporsi a queste caratteristiche), ma spesso sono obiettivi contraddittori. Come semplice esempio di questo conflitto si osservi la Figura 5.5. Si supponga che ci sia abbastanza traffico tra A e A', tra B e B' e tra C e C' da saturare i collegamenti orizzontali. Per massimizzare il flusso totale, il traffico tra X e X' dovrebbe essere interrotto del tutto. Sfortunatamente, X e X' potrebbero non vederla nello stesso modo. Evidentemente, sono necessari alcuni compromessi tra efficienza globale e imparzialità verso le singole connessioni.

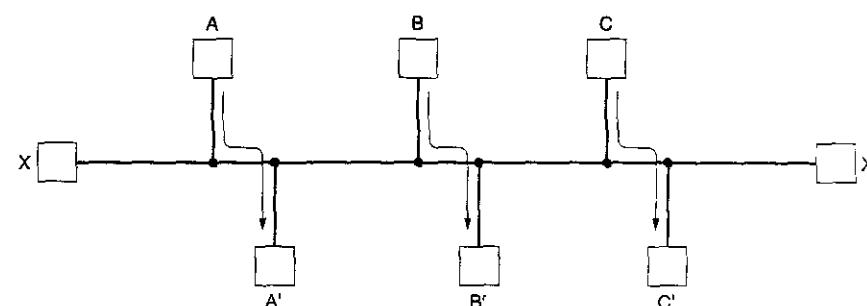


Figura 5.5. Conflitto tra imparzialità e ottimizzazione.

Ancor prima di poter tentare di trovare compromessi tra imparzialità e ottimizzazione, è necessario scegliere che cosa ottimizzare. Un ovvio candidato è la riduzione al minimo possibile del ritardo medio dei pacchetti, così come la massimizzazione della capacità di carico totale della rete. Anche questi due obiettivi sono però in conflitto, poiché far lavorare un sistema ad accodamento vicino alla sua capacità massima crea un lungo tempo di attesa. Come compromesso, molte reti tentano di ridurre il numero dei salti che il pacchetto deve compiere, perché in questo modo si migliora il tempo di attesa, si riduce la banda consumata, e di conseguenza si migliora anche la capacità di carico.

Gli algoritmi di routing si possono raggruppare in due classi principali: adattivi e non adattivi. Gli **algoritmi non adattivi** non basano le loro decisioni su misure o stime del traffico e della topologia corrente. Il percorso utilizzato per collegare I a J (per tutti gli I e J) è invece calcolato in anticipo, in modalità fuori linea, ed è scaricato nei router all'avvio della rete. Questa procedura si chiama anche **routing statico**.

Al contrario gli **algoritmi adattivi** cambiano le loro decisioni secondo le modifiche apportate alla topologia, e di solito anche al traffico. Gli algoritmi adattivi si distinguono per la fonte da cui traggono le loro informazioni (ossia localmente, da router adiacenti o da tutti i router), per il momento in cui modificano i percorsi (per esempio ogni DT secondi, quando il carico cambia, o quando la topologia cambia) e per il tipo

i metrica utilizzata nell'operazione di ottimizzazione (per esempio la distanza, il numero di salti o il tempo di transito stimato). I paragrafi seguenti descrivono diversi algoritmi di routing statici e dinamici.

5.2.1 Il principio di ottimalità

Prima di esaminare algoritmi specifici, può essere utile notare che è possibile fare un'ipotesi generale riguardo ai percorsi ottimali indipendentemente dalla topologia di rete o dal traffico. Questo assioma, conosciuto come **principio di ottimalità**, afferma che se il router *J* si trova sul percorso ottimale che collega *I* a *K*, allora anche il percorso ottimale da *J* a *K* segue la stessa rotta. Per dimostrarlo è sufficiente chiamare r_1 la parte del percorso che unisce *I* a *J* e r_2 la parte restante del tracciato. Se tra *J* e *K* esistesse un percorso migliore di r_2 , esso potrebbe essere concatenato con r_1 per migliorare la strada da *I* a *K*, ma questo contraddicebbe l'ipotesi iniziale che r_1, r_2 è ottimale.

Come diretta conseguenza del principio di ottimalità, si vede che la serie di percorsi ottimali che collegano tutte le sorgenti a una data destinazione formano una struttura ad albero dove il nodo principale è costituito dalla destinazione. In questo albero, chiamato **sink tree** (Figura 5.6), la distanza è calcolata in base al numero di salti. Si noti che un sink tree non è necessariamente unico; possono esistere altre strutture con le stesse lunghezze di percorso. L'obiettivo di tutti gli algoritmi di routing è scoprire e utilizzare i sink tree per tutti i router.

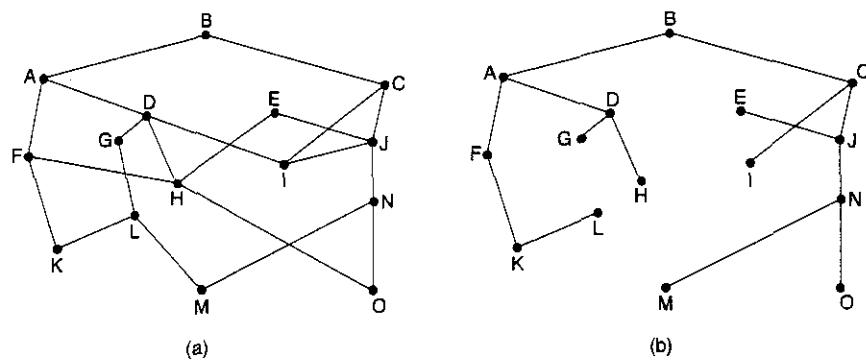


Figura 5.6. (a) Una sottorete. (b) Un sink tree per il router B.

In sink tree è una struttura ad albero e pertanto non contiene alcun ciclo, quindi ogni pacchetto verrà inoltrato in un numero finito e limitato di salti. In realtà la vita non è così semplice. I collegamenti e i router possono disattivarsi e attivarsi durante il funzionamento della rete, perciò router diversi possono avere idee differenti sulla topologia corrente. Inoltre bisogna ancora decidere come ciascun router deve acquisire le informazioni su cui basa il calcolo del sink tree: devono essere recuperate individualmente dai singoli router.

oppure in qualche altro modo? Tra poco vedremo la risposta, ma resta assodato che il principio di ottimalità e la struttura sink tree forniscono un modo per valutare le prestazioni degli algoritmi di routing.

5.2.2 Routing basato sul percorso più breve

L'analisi dei possibili algoritmi di routing ha inizio da una tecnica che è largamente utilizzata in molte forme, perché è semplice e facile da comprendere. L'idea è quella di costruire un grafo della sottorete dove ogni nodo rappresenta un router e ogni arco rappresenta una linea di comunicazione (spesso chiamata collegamento). Per scegliere un percorso tra una coppia di router, l'algoritmo cerca semplicemente la strada più corta (cammino minimo) che collega i due nodi del grafo.

Il concetto di **cammino minimo** o **percorso più breve** merita una spiegazione. La lunghezza del percorso può essere misurata con il numero di salti; usando questa metrica i percorsi ABC e ABE nella Figura 5.7 sono uguali. Un'altra metrica è basata sulla distanza geografica espressa in chilometri; in questo caso il percorso ABC è chiaramente più lungo di ABE (supponendo che la figura sia disegnata in scala).

Si possono usare molte altre metriche oltre alla distanza fisica e al numero di salti. Per esempio, ogni arco potrebbe essere etichettato con i valori che rappresentano i valori medi di accodamento e ritardo della trasmissione, calcolati mediante test eseguiti ogni ora su alcuni pacchetti di prova standard. Con questo tipo di classificazione, il percorso più breve è rappresentato dal percorso più veloce e non da quello con il numero minore di archi o di chilometri.

Nel caso generale, le etichette sugli archi possono essere calcolate come una funzione della distanza, della banda, del traffico medio, del costo della comunicazione, della lunghezza media della coda, del ritardo e di altri fattori. Modificando la funzione che attribuisce i pesi, l'algoritmo calcolerebbe il percorso "più breve" misurato secondo qualunque numero di criteri o loro combinazione.

Si conoscono diversi algoritmi per elaborare il percorso più breve tra due nodi di un grafo. Quello descritto è stato ideato da Dijkstra (1959, SPF, *Shortest Path First*). Ogni nodo è associato a un'etichetta (racchiusa tra parentesi) che riporta la sua distanza dal nodo d'origine lungo il miglior percorso conosciuto. Inizialmente nessun percorso è noto, perciò tutti i nodi sono etichettati con il simbolo che rappresenta l'infinito. Mano a mano che l'algoritmo procede e si trovano i percorsi, le etichette cambiano rispecchiando i percorsi migliori. Un'etichetta può essere provvisoria o permanente. Inizialmente tutte le etichette sono provvisorie, e quando si scopre che rappresenta il percorso più breve possibile dall'origine a quel nodo viene resa permanente smettendo di modificarla.

Per illustrare come funziona l'algoritmo basato sulle etichette, si osservi il grafo pesato non orientato rappresentato nella Figura 5.7(a), dove i pesi rappresentano, per esempio, una distanza. Si desidera trovare il percorso più breve che collega A a D. Si comincia contrassegnando il nodo A come permanente; per farlo è sufficiente tracciare un cerchietto pieno. Poi si esaminano, uno dopo l'altro, tutti i nodi adiacenti ad A (*working node*), rietichettando ogni nodo con la distanza da A. Ogni volta che si rietichetta un nodo, si prende nota anche del

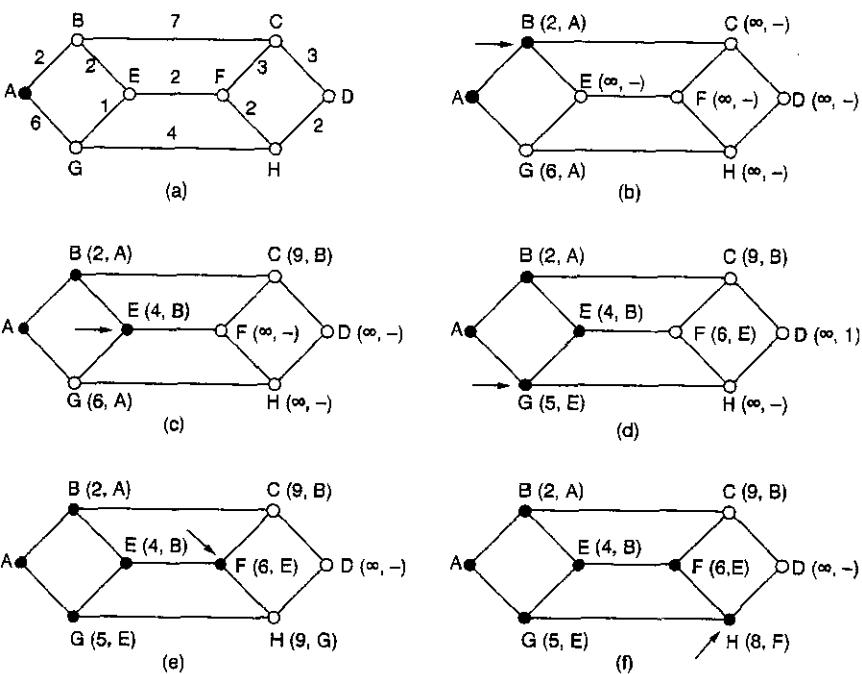


Figura 5.7. I primi cinque passaggi utilizzati nell'elaborazione del percorso più breve che collega A a D. Le frecce indicano il working node.

come del nodo dal quale è partita l'indagine, in modo da poter successivamente ricostruire percorso finale. Dopo aver esaminato ogni nodo adiacente ad A si esaminano tutti i nodi etichettati dell'intero grafico, e si rende permanente quello con l'etichetta più piccola come mostrato nella Figura 5.7(b). Questo diventa il nuovo working node.

ra si comincia da B e si esaminano tutti i nodi che gli sono adiacenti. Se l'etichetta su B sommata alla distanza da B al nodo considerato è minore dell'etichetta di quel nodo, allora è stato trovato un percorso più breve e il nodo è rietichettato.

Dopo aver esaminato tutti i nodi adiacenti al working node e aver modificato dove possibile le etichette provvisorie, si cerca nell'intero grafo il nodo etichettato provvisoriamente tenente il valore più piccolo. Questo nodo è reso permanente e diventa il working node per il passaggio successivo. La Figura 5.7 mostra i primi cinque passaggi dell'algoritmo.

Per capire perché l'algoritmo funziona, si osservi la Figura 5.7(c). In questo istante il nodo E è appena stato reso permanente. Si supponga che ci sia un percorso più breve di ABE, per esempio AXE. Ci sono due possibilità: o il nodo Z è già stato reso permanente oppure non lo è stato. Nel primo caso, E è stato già esaminato (nel passaggio successivo al passaggio in cui E è stato reso permanente), perciò il percorso AXE non è sfuggito all'attenzione e non può essere un percorso più breve.

Ora si consideri il caso in cui Z ha ancora un'etichetta temporanea. O l'etichetta di Z è maggiore o uguale a quella di E, nel qual caso AXE non può essere un percorso più corto di ABE, oppure è minore di quella di E, nel qual caso Z e non E diventerà permanente per primo, permettendo a E di essere esaminato da Z.

L'algoritmo è riportato nella Figura 5.8. Le variabili globali n e $dist$ descrivono il grafo e sono inizializzate prima della chiamata indirizzata a `shortest_path`. L'unica differenza tra il programma e l'algoritmo appena descritto è che nella Figura 5.8 il percorso più breve è calcolato iniziando dal nodo terminale t e non dal nodo d'origine s . Poiché il percorso più corto che va da t a s in un grafo non orientato è identico al percorso più corto che va da s a t , non ha importanza l'estremità dalla quale si parte (a meno che non ci siano diversi percorsi più brevi, nel qual caso invertendo la ricerca se ne potrebbe scoprire uno diverso). La ricerca viene eseguita al contrario perché ogni nodo è etichettato con il suo predecessore e non con il suo successore. Quando il percorso finale viene copiato nella variabile di output `path`, il percorso è perciò invertito. Invertendo la ricerca, i due effetti si annullano e la risposta è prodotta nell'ordine corretto.

5.2.3 Flooding

Un altro algoritmo statico è quello di **flooding**, in cui ogni pacchetto in arrivo è inviato a tutte le linee tranne quella da cui proviene. Logicamente questo meccanismo genera un elevato numero di pacchetti duplicati; anzi teoricamente il numero è infinito, a meno che non si prendano delle misure per attenuare il processo. Per esempio, si potrebbe utilizzare un contatore di salti inserito nell'intestazione di ogni pacchetto, e decrementare il suo valore a ogni salto in modo da scartare automaticamente il pacchetto quando il contatore raggiunge lo zero. Idealmente il valore iniziale da assegnare al contatore dei salti dovrebbe corrispondere alla lunghezza del percorso dall'origine alla destinazione. Se il trasmettente non conosce la lunghezza del percorso, può assegnare al contatore il valore che rappresenta il caso peggiore, vale a dire il diametro dell'intera sottorete.

Una tecnica di moderazione alternativa tiene traccia dei pacchetti trasmessi nel flood, in modo da evitare una seconda ritrasmissione. Per ottenere questo risultato, il router d'origine deve inserire in ogni pacchetto che riceve dai suoi host un numero di sequenza. Ogni router ha quindi bisogno di una lista per ciascuno dei router sorgenti, che indichi i numeri di sequenza già trasmessi da quella origine. Se appare nella lista, il pacchetto in arrivo non viene trasmesso.

Per impedire una crescita illimitata ogni lista dovrebbe essere integrata con un contatore k , in modo da indicare che sono stati visti tutti i numeri di sequenza fino a k . Quando arriva un pacchetto è facile controllare se è un duplicato; in caso affermativo viene scartato. Inoltre non è necessaria tutta la lista al di sotto di k , poiché in pratica k la riassume totalmente.

Una variante un po' più pratica è chiamata **flooding selettivo**. In questo algoritmo, i router non trasmettono ogni pacchetto in arrivo attraverso tutte le linee ma solo attraverso quelle che vanno approssimativamente nella direzione giusta. Di solito non ha molto senso inviare lungo una linea diretta a est un pacchetto indirizzato a ovest, a meno che la topologia non sia particolarmente strana e il router non sia sicuro di questo fatto.

```

define MAX_NODES 1024           /* Numero massimo di nodi */
define INFINITY 1000000000      /* Un numero più grande di qualunque percorso massimo */
int n, dist[MAX_NODES][MAX_NODES]; /* dist[i][j] rappresenta la distanza da i a j */

void shortest3path(int s, int t, int path[])
{
    struct state {
        int predecessor;          /* Il percorso elaborato */
        int length;                /* nodo precedente */
        enum {permanent, tentative} label; /* distanza dall'origine a questo nodo */
        state[MAX_NODES];
    } state;
    state[s].label = permanent;
    state[s].length = 0;
    state[t].label = permanent;
    state[t].length = INFINITY;
    state[t].predecessor = -1;
    state[t].label = tentative;
    state[t].length = 0;
    int k, min;
    struct state *p;
    p = &state[0];
    for (i = 0; i < n; i++)
        if (dist[k][i] != 0 && state[i].label == tentative) {
            if (state[k].length + dist[k][i] < state[i].length) {
                state[i].predecessor = k;
                state[i].length = state[k].length + dist[k][i];
            }
        }
    /* Trova il nodo etichettato provvisoriamente con l'etichetta più piccola. */
    k = 0; min = INFINITY;
    for (i = 0; i < n; i++)
        if (state[i].label == tentative && state[i].length < min) {
            min = state[i].length;
            k = i;
        }
    state[k].label = permanent;
    while (k != s)
        path[i++] = k;
        k = state[k].predecessor;
    path[i] = -1;
}

Copia il percorso nella matrice di output.
= 0; k = s;
for (path[i++] = k; k = state[k].predecessor; ) while (k >= 0);

```

Figura 5.8. Algoritmo di Dijkstra per calcolare il cammino minimo in un grafo.

Nella maggior parte delle applicazioni l'algoritmo di flooding non è molto funzionale, ma comunque ha alcuni utilizzi pratici. Per esempio nelle applicazioni militari, dove un gran numero di router può saltare in aria in un istante, la grande robustezza dell'algoritmo di flooding è molto desiderabile. Nelle applicazioni database distribuite qualche volta è necessario aggiornare simultaneamente tutti i database; in questo caso può essere utile adottare l'algoritmo di flooding. Nelle reti wireless, tutti i messaggi trasmessi da una stazione possono essere ricevuti da tutte le altre stazioni che si trovano nel campo della sorgente radio; si tratta in effetti di una tecnica di flooding, e alcuni algoritmi utilizzano proprio questa proprietà. Infine, l'algoritmo di flooding può essere utilizzato anche come metrica di confronto per altri algoritmi di routing; il flooding sceglie sempre il percorso più breve perché sceglie ogni possibile percorso in parallelo, di conseguenza nessun altro algoritmo può produrre un ritardo più breve (se si ignora il tempo di elaborazione dati generato dal processo stesso di flooding).

5.2.4 Routing basato sul vettore delle distanze

Le moderne reti di computer generalmente utilizzano algoritmi di routing dinamici al posto di quelli statici descritti nei paragrafi precedenti, poiché gli algoritmi statici non tengono conto del carico istantaneo della rete. I due algoritmi dinamici più popolari sono il routing basato sul vettore delle distanze e il routing basato sullo stato dei collegamenti; questo paragrafo esamina il primo algoritmo, il seguente studierà il secondo.

Gli algoritmi di routing basati sul **vettore delle distanze** (*distance vector routing*) operano facendo in modo che ogni router conservi una tabella (ossia un vettore) che definisce la migliore distanza conosciuta per ogni destinazione e la linea che conduce a tale destinazione. Queste tabelle sono aggiornate scambiando informazioni con i router vicini.

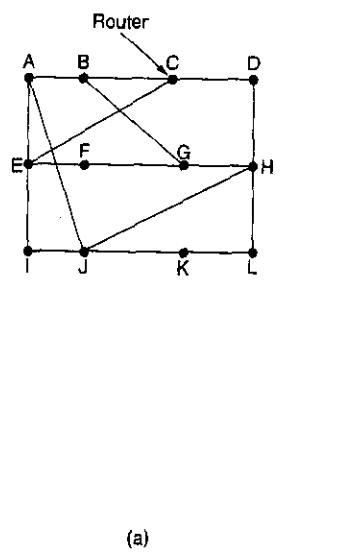
Il routing basato sul vettore delle distanze qualche volta è chiamato anche con altri nomi; i più comuni sono algoritmo di routing **Bellman-Ford** distribuito o algoritmo **Ford-Fulkerson**, dai nomi dei ricercatori che lo hanno sviluppato (Bellman, 1957; Ford e Fulkerson, 1962). È stato l'algoritmo di routing originale di ARPANET ed è stato utilizzato anche in Internet con il nome di RIP.

Nel routing basato sul vettore delle distanze ciascun router conserva una tabella di routing indicizzata da ogni router della sottorete, dove è memorizzata una voce per ogni router. Questa voce è composta da due parti: la linea di trasmissione preferita da utilizzare per quella destinazione, e una stima del tempo o della distanza associata a quella destinazione. La metrica utilizzata potrebbe essere il numero di salti, il ritardo espresso in millisecondi, il numero totale di pacchetti accodati lungo il percorso o qualcosa di simile.

Si presume che il router conosca la "distanza" che lo separa da ognuno dei suoi vicini. Se è misurata in salti, la distanza è pari a un solo salto. Se la metrica è la lunghezza della coda, il router esamina ogni coda. Se la metrica è il ritardo, il router può misurare direttamente questo valore mediante speciali pacchetti ECHO che il ricevitore rimanda indietro il più velocemente possibile dopo aver registrato al loro interno la data e l'ora di trasmissione. Come esempio, si supponga che la distanza sia misurata con il ritardo e che il router conosca il ritardo di ogni vicino. Una volta ogni T msec, ogni router invia ai suoi vicini una

lista delle proprie stime di ritardo relative a ciascuna destinazione. Il router riceve inoltre una lista simile da ognuno dei suoi vicini. S'immagini che una di queste tabelle sia appena arrivata dal vicino X , e che X_i rappresenti la stima del tempo impiegato da X per raggiungere il router i . Se sa che il ritardo per arrivare a X è pari a m msec, allora il router sa anche che può raggiungere il router i passando attraverso X in $X_i + m$ msec. Eseguendo questo calcolo per ognuno dei vicini un router può determinare la stima migliore, e utilizzarla nella sua nuova tabella di routing assieme all'indicazione della linea corrispondente. Si noti che la vecchia tabella di routing non è utilizzata nel calcolo.

Questo processo di aggiornamento è illustrato nella Figura 5.9. La parte (a) mostra una sottorete; le prime quattro colonne della parte (b) mostrano i vettori di ritardo ricevuti dai vicini del router J . A vanta un ritardo di 12 msec verso B , un ritardo di 25 msec verso C , un ritardo di 40 msec verso D e così via. Si supponga che J abbia misurato o stimato il suo ritardo verso i suoi vicini A, I, H e K rispettivamente pari a 8, 10, 12 e 6 msec.



(a)

	To	A	I	H	K	→ Linea
	Router	Ritardo	Ritardo	Ritardo	Ritardo	
A	A	0	24	20	21	8 A
B	B	12	36	31	28	20 A
C	C	25	18	19	36	28 I
D	D	40	27	8	24	20 H
E	E	14	7	30	22	17 I
F	F	23	20	19	40	30 I
G	G	18	31	6	31	18 H
H	H	17	20	0	19	12 H
I	I	21	0	14	22	10 I
J	J	9	11	7	10	0
K	K	24	22	22	0	6 K
L	L	29	33	9	9	15 K
	JA	Ritardo	Ritardo	Ritardo	Ritardo	Nuova tabella di routing di J
		is	is	is	is	
		8	10	12	6	

Vettori ricevuti dai quattro vicini di J

(b)

Figura 5.9. (a) Una sottorete. (b) Input da A, I, H, K e la nuova tabella di routing di J .

vediamo come J calcola il suo nuovo percorso per il router G . Poiché sa di poter raggiungere A in 8 msec, e poiché A dichiara di poter arrivare a G in 18 msec, J sa che trasmettendo i suoi pacchetti verso A potrà raggiungere G in 26 msec. In modo analogo, la distanza da G calcolata passando attraverso I, H e K è rispettivamente di 41 (31 + 10), 18 (6 + 2) e 37 (31 + 6) msec. Il valore migliore è 18, perciò J crea nella sua tabella di routing una voce associata a G e vi registra il ritardo 18 msec e la linea di trasmissione H . Lo stesso calcolo è eseguito per tutte le altre destinazioni; la nuova tabella di routing è mostrata nell'ultima colonna della figura.

Il problema del conto all'infinito

In teoria il routing basato sul vettore delle distanze funziona, ma ha un serio difetto pratico: sebbene converga verso la risposta corretta, può raggiungere l'obiettivo molto lentamente. In particolare, reagisce rapidamente alle buone notizie ma troppo lentamente a quelle cattive. Per esempio, si supponga che il percorso migliore da un router a una destinazione X sia un valore molto grande. Se uno degli scambi successivi con il vicino A improvvisamente indica un ritardo breve verso X , il router inizia a utilizzare la linea che punta ad A per inoltrare il traffico verso X . Le buone notizie sono elaborate in un solo scambio di vettori.

Per vedere con quale velocità si propaga la buona notizia, si consideri la sottorete (lineare) composta da cinque nodi mostrata nella Figura 5.10; in questo caso la distanza è rappresentata dal numero di salti. Si supponga che A inizialmente non sia attivo e che tutti i router siano a conoscenza di questo. In altre parole, tutti i router hanno registrato un ritardo infinito verso A .

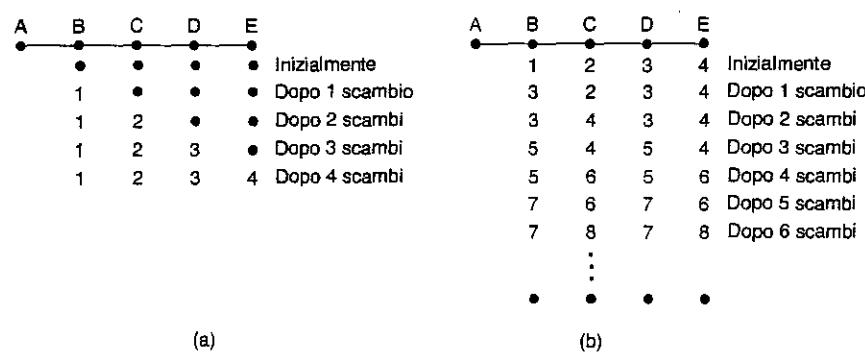


Figura 5.10. Il problema del conteggio infinito.

Quando A entra in azione, gli altri router si accorgono della sua presenza grazie allo scambio di vettori. Per semplicità, si supponga che da qualche parte ci sia un gigantesco gong, colpito periodicamente per iniziare uno scambio di vettori contemporaneo su tutti i router. Al primo scambio, B scopre che il suo vicino di sinistra dista 0 da A ; B crea quindi una voce nella sua tabella di routing registrando che A si trova a un salto di distanza a sinistra. Tutti gli altri router credono ancora che A non sia attivo. Le voci della tabella di routing relative ad A sono riportate nella seconda riga mostrata nella Figura 5.10 (a). Allo scambio successivo, C scopre che B ha un percorso di lunghezza 1 diretto ad A , perciò aggiornerà la sua tabella di routing per indicare un percorso di lunghezza 2 ma D ed E non sentono ancora le buone notizie. Logicamente, la buona notizia si diffonde alla velocità di un salto per ciascuno scambio. In una sottorete il cui percorso più lungo richiede N salti, entro N scambi tutti vengono a conoscenza dei nuovi router e delle nuove linee.

Ora si consideri la situazione mostrata nella Figura 5.10(b), in cui tutte le linee e i router sono inizialmente attivi. La distanza tra A e i router B, C, D ed E è rispettivamente di 1, 2, 3 e 4. Improvvistamente A si spegne, oppure la linea tra A e B si interrompe (ciò rappresenta la stessa cosa, dal punto di vista di B).

imo scambio di pacchetti *B* non riceve nulla da *A*. Fortunatamente *C* dice: "Non ti preoccupare, io ho un percorso che dista 2 da *A*". *B* non può sapere che il percorso di *C* passa da *B*. Per quanto ne sa *B*, *C* potrebbe avere dieci linee, tutte con percorsi separati diretti ad *A* (lunghezza 2). Di conseguenza, *B* pensa di poter raggiungere *A* attraverso *C* con un percorso (lunghezza 3). *D* ed *E* non aggiornano le loro voci relative ad *A* durante il primo scambio. Il secondo scambio, *C* nota che tutti i suoi vicini affermano di essere a distanza 3 da *A*; e quindi a caso uno dei vicini e imposta a 4 la sua nuova distanza da *A*, come mostrato nella terza riga della Figura 5.10(b). Gli scambi seguenti producono i risultati mostrati nelle altre righe della Figura 5.10(b). Dalla figura dovrebbe essere chiaro perché le cattive notizie viaggiano lentamente: nessun router ha mai un valore che supera di oltre una unità minima di tutti i vicini. Lentamente tutti i router trovano la loro strada per *A*, ma il numero di scambi richiesti dipende dal valore numerico utilizzato per rappresentare l'infinito. Per questo motivo è meglio impostare come infinito il valore corrispondente al percorso più lungo più uno. Se la metrica è basata sul ritardo, non esiste un valore superiore ben definito, perciò per impedire i problemi causati dai ritardi più lunghi è meglio impostare un valore alto. Questo problema è stato chiamato **problema del conto finito** (convergenza lenta). Sono stati fatti diversi tentativi per risolverlo, ma in genere nessuno di questi espedienti funziona bene. Il nocciolo del problema è che quando *X* e *Y* che ha un percorso che punta da qualche parte, *Y* non ha modo di sapere se lui fa parte di quel percorso.

Routing basato sullo stato dei collegamenti

Routing basato sul vettore delle distanze è stato utilizzato in ARPANET fino al 1979, quando fu sostituito dal routing basato sullo stato dei collegamenti. Due problemi principali causarono il suo abbandono. Primo: poiché si basava sulla lunghezza della coda, la misura del ritardo non teneva conto della banda della linea, nella scelta dei percorsi. Inoltre tutte le linee erano a 56 kbps, perciò la banda non era un problema, ma quando alcune linee vennero portate a 230 kbps e altre raggiunsero i 1,544 Mbps, non consentì la banda divenne un difetto serio. Naturalmente sarebbe stato possibile cambiare la misura del ritardo in modo da tenere conto anche della banda, ma c'è un altro problema: il ritmo spesso impiegava troppo tempo a raggiungere la convergenza (problema dell'infinito). Per questo motivo venne sostituito da un algoritmo completamente diverso, che oggi è chiamato **routing basato sullo stato dei collegamenti** (*link state routing*) ed è largamente usato in diverse varianti.

Alla base di questo algoritmo è semplice e può essere riassunta in cinque punti. Ogni router deve:

1. scoprire i propri vicini e i relativi indirizzi di rete
2. misurare il ritardo o il costo di ogni vicino
3. costruire un pacchetto che contiene tutte le informazioni raccolte
4. inviare questo pacchetto a tutti gli altri router
5. elaborare il percorso più breve verso tutti gli altri router.

In pratica, si misurano sperimentalmente la topologia completa e tutti i ritardi per poi distribuirli a ogni router; successivamente viene eseguito l'algoritmo di Dijkstra per trovare il percorso più breve associato a ogni altro router. Il prossimo paragrafo esamina in dettaglio ognuno dei cinque passaggi.

Scoperta dei vicini

Quando viene acceso, il router prima di tutto cerca di scoprire chi sono i suoi vicini. Il dispositivo raggiunge il suo obiettivo inviando uno speciale pacchetto HELLO su ogni linea punto-punto; il router all'altro capo della linea deve rispondere fornendo la propria identità. Questi nomi devono essere globalmente unici, perché quando in seguito un router lontano rileva che tre router sono collegati a *F*, è essenziale che possa determinare se tutti e tre intendono lo stesso *F*.

Quando due o più router sono collegati a una LAN, la situazione è leggermente più complicata. La Figura 5.11(a) mostra una LAN a cui sono direttamente collegati tre router *A*, *C* e *F*, e ognuno di questi è collegato a uno o più router aggiuntivi come indicato.

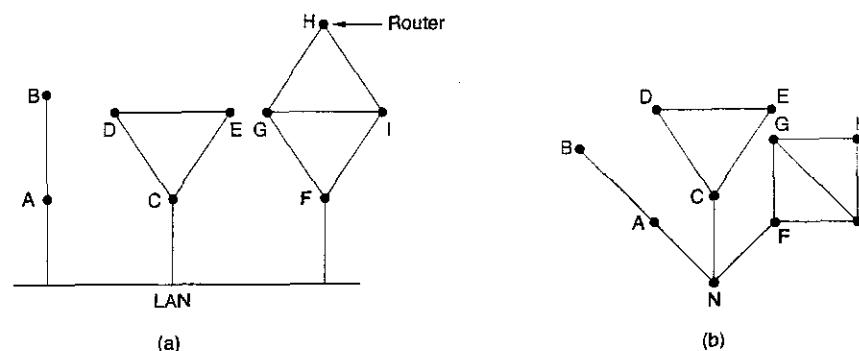


Figura 5.11. (a) Nove router e una LAN. (b) Un modello grafico di (a).

Un possibile modello della LAN si ottiene considerandola nell'insieme come un solo nodo, come è stato fatto nella Figura 5.11(b). In questa figura è stato introdotto un nuovo nodo artificiale, *N*, al quale si collegano *A*, *C* ed *F*. Il fatto che sia possibile andare da *A* a *C* sulla LAN è rappresentato dal percorso *ANC*.

Misurazione del costo della linea

L'algoritmo di routing basato sullo stato dei collegamenti richiede che ogni router conosca il ritardo di ognuno dei suoi vicini, o almeno ne abbia una stima ragionevole. Il modo più diretto per determinare questo ritardo consiste nell'inviare attraverso la linea uno speciale pacchetto ECHO, al quale l'altra parte deve rispondere immediatamente. Misurando il tempo di andata e ritorno e dividendo il risultato per due, il router trasmittente può ottenere una stima ragionevole del ritardo. Per ottenere risultati ancora più precisi si può eseguire un ping-pong bidirezionale.

ire il test diverse volte calcolando poi la media dei risultati. Naturalmente questo sistema presuppone implicitamente che i ritardi siano simmetrici, il che non sempre è vero. Ecco domandarsi se durante la misurazione del ritardo non sarebbe meglio tenere conto del carico. Per considerare anche il carico, il cronometro che misura il tempo di andata e ritorno deve essere avviato nel momento in cui il pacchetto ECHO viene inserito nella coda; per ignorarlo, il cronometro dovrebbe essere avviato quando il pacchetto ECHO raggiunge il fronte della coda.

Entrambi i metodi sono discutibili. Includere nelle misurazioni i ritardi indotti dal traffico significa che quando dovrà scegliere tra due linee che hanno la stessa banda, una delle quali è caricata pesantemente tutto il tempo e l'altra no, il router considererà più breve il percorso che attraversa la linea non appesantita dal traffico. Questa scelta migliorerà le prestazioni.

Fortunatamente, c'è anche un aspetto negativo che consiglia d'includere l'effetto del carico nel calcolo del ritardo. Si consideri la sottorete mostrata nella Figura 5.12, divisa in due parti (est e ovest) collegate da due linee, *CF* e *EI*.

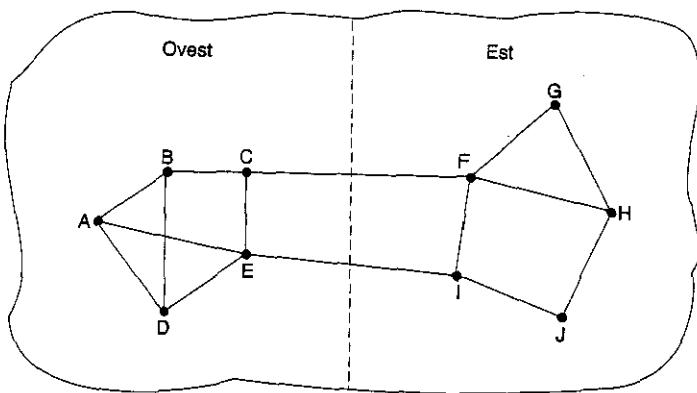


Figura 5.12. Una sottorete in cui le parti est e ovest sono collegate da due linee.

Supponga che la maggior parte del traffico tra est e ovest utilizzi la linea *CF*, pertanto questa linea è pesantemente caricata e genera lunghi ritardi. L'inclusione del ritardo di trasferimento nel calcolo del percorso più breve rende *EI* molto più attraente. Dopo l'installazione delle nuove tabelle di routing, la maggior parte del traffico Est - Ovest seguirà il percorso *EI*, sovraccaricando questa linea. Di conseguenza, nel successivo aggiornamento *CF* sembrerà essere il percorso più breve: il risultato è che le tabelle di routing inizieranno a oscillare paurosamente, causando intransigimenti irregolari e molti problemi potenziali. Questo problema non si presenta ignorando il carico e considerando solo la banda. L'alternativa il carico può essere distribuito su entrambe le linee, ma questa soluzione non raggiunge al massimo il percorso migliore. Ciò nonostante, per evitare oscillazioni nella scelta del percorso migliore può essere saggio distribuire il carico su più linee, attribuendone quantità ben definite a ciascuna.

Costruzione dei pacchetti che contengono lo stato dei collegamenti

Dopo aver raccolto le informazioni necessarie per lo scambio, ogni router deve costruire un pacchetto contenente tutti i dati. Il pacchetto inizia con l'identità del trasmittente, seguita da un numero di sequenza, dall'età (descritta più avanti) e da una lista dei vicini. Per ogni vicino è riportato il ritardo misurato. La Figura 5.13(a) mostra un esempio di sottorete; i ritardi sono rappresentati da etichette associate alle linee. I pacchetti che contengono lo stato dei collegamenti per tutti e sei i router sono mostrati nella Figura 5.13(b).

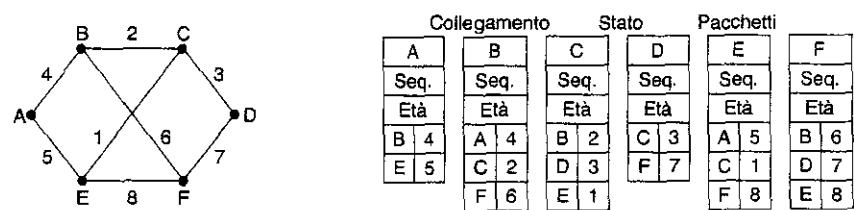


Figura 5.13. (a) Una sottorete. (b) I pacchetti che contengono lo stato dei collegamenti per questa sottorete.

È facile costruire questi pacchetti, e la parte difficile è determinare quando costruirli. I pacchetti che contengono le informazioni sullo stato dei collegamenti potrebbero essere costruiti periodicamente, ossia a intervalli regolari, oppure potrebbero essere creati quando accade un evento significativo: quando si interrompe una linea, o un vicino si spegne, torna di nuovo attivo o cambia le sue proprietà.

Distribuzione dei pacchetti che contengono lo stato dei collegamenti

La parte più delicata dell'algoritmo è quella relativa alla distribuzione affidabile dei pacchetti contenenti i dati che descrivono lo stato dei collegamenti. Durante la loro distribuzione e l'installazione i router che ricevono i primi pacchetti cambieranno i loro percorsi. Di conseguenza, router diversi potrebbero usare versioni differenti della topologia col rischio di creare inconsistenze, cicli, computer irraggiungibili e così via.

Questo paragrafo descrive l'algoritmo di distribuzione di base; più avanti saranno analizzati alcuni miglioramenti. L'idea fondamentale è di utilizzare il flooding per distribuire i pacchetti contenenti le informazioni sullo stato dei collegamenti. Per tenere sotto controllo il flusso di dati in ogni pacchetto è inserito un numero di sequenza, incrementato per ogni nuovo pacchetto inviato. I router tengono traccia di tutte le coppie (router sorgente, sequenza) rilevate. Quando arriva un nuovo pacchetto contenente informazioni sullo stato della connessione, il router confronta i dati con quelli già visti. Se è nuovo, il pacchetto è inoltrato attraverso tutte le linee tranne quella di ricezione; se è un duplicato, i dati sono scartati. Se arriva un pacchetto con un numero di sequenza inferiore al numero più alto visto fino a quel momento, i dati sono rifiutati in quanto obsoleti, poiché il router ha informazioni più recenti. Questo algoritmo ha alcuni difetti, che possono comunque essere gestiti. Prima di tutto, numeri di sequenza ripetitivi potrebbero generare il caos. Per risolvere questo problema si utilizzano numeri di

uenza lunghi 32 bit; trasmettendo un pacchetto di dati sullo stato dei collegamenti ogni secondo, i numeri di sequenza si esaurirebbero non prima di 137 anni, perciò questa possibilità può essere ignorata. Secondo, quando un router si blocca perde traccia dei suoi numeri di sequenza. Se il conteggio ricomincia da 0, il pacchetto successivo viene rifiutato perché ritenuto un duplicato.

zo, se un numero di sequenza arriva danneggiato e il router legge 65.540 al posto del numero (è un errore di un solo bit), i pacchetti contrassegnati dai numeri che vanno da 5 a 65.540 sono considerati obsoleti poiché il router crede che il numero di sequenza sia 65.540.

risolvere tutti questi problemi è necessario includere l'età di ogni pacchetto subito dopo il numero di sequenza, e decrementare il suo valore una volta al secondo. Quando l'età raggiunge lo zero, le informazioni provenienti da quel router vengono scartate. Normalmente un nuovo pacchetto arriva (per esempio) ogni 10 secondi, perciò le informazioni del router sono scritte quando un router è spento (o sei pacchetti consecutivi vengono perduti, evento più raro). Il campo *età* è anche decrementato da ogni router durante il processo di flooding locale, per garantire che nessun pacchetto possa andare perduto e durare per un periodo indebolito di tempo (un pacchetto la cui età è zero viene scartato).

uni miglioramenti apportati a questo algoritmo rendono il processo ancora più robusto. Un pacchetto contenente i dati sullo stato dei collegamenti raggiunge un router per il routing, non viene accodato immediatamente per la trasmissione. I dati vengono prima inseriti in un'area di mantenimento e qui rimangono per un breve tempo. Se prima della trasmissione arriva dalla stessa sorgente un altro pacchetto contenente dati sullo stato dei collegamenti, il router confronta i numeri di sequenza; se i valori sono uguali il duplicato viene scartato, mentre viene eliminato il più vecchio. Per prevenire errori sulle linee da router a router, tutti i pacchetti contenenti informazioni relative allo stato dei collegamenti ricevono acknowledgement. Quando una linea smette di essere impegnata, l'area di mantenimento viene esaminata casualità round robin per selezionare un pacchetto o un acknowledgement da inviare.

struttura dati utilizzata dal router *B* per la sottorete mostrata nella Figura 5.13(a) è riportata nella Figura 5.14. Ogni riga corrisponde a un pacchetto contenente informazioni sullo stato dei collegamenti, arrivato di recente ma non ancora elaborato. La tabella contiene l'origine del pacchetto, il suo numero di sequenza, l'età e i dati; inoltre esistono flag per l'invio e la ricezione dell'acknowledgement per ognuna delle tre linee di *B* (rispettivamente per *A*, *C* e *F*). La presenza del flag d'invio indica che il pacchetto deve essere trasmesso su quella linea; il flag ack indica se è atteso un acknowledgement.

a Figura 5.14, il pacchetto contenente dati sullo stato dei collegamenti proveniente da *E* deve essere inviato a *C* e a *F* mentre ad *A* va inviato un acknowledgement, come indicato dai bit di flag. In modo analogo, il pacchetto proveniente da *D* deve essere inoltrato ad *A* e a *C* e l'acknowledgement inviato a *F*. La situazione cambia con il terzo pacchetto, quello proveniente da *E*. I dati arrivano due volte, ma attraverso *EAB* e l'altra attraverso *EFB*. Di conseguenza, il pacchetto deve essere trasmesso solo a *C* ma l'acknowledgement va inviato sia ad *A* sia a *F*, come indicato dai bit. Si tratta di un duplicato mentre l'originale è ancora nel buffer, i bit devono essere modificati.

Origine	Seq.	Età	Flag di invio			ACK			Dati
			A	C	F	A	C	F	
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Figura 5.14. Il buffer di pacchetti per il router *B* della Figura 5.13.

cati. Per esempio, se una copia dello stato di *C* arriva da *F* prima dell'inoltro della quarta voce della tabella, i sei bit devono diventare 100011 per indicare che a *F* va inviato l'acknowledgement del pacchetto, ma non il pacchetto vero e proprio.

Elaborazione dei nuovi percorsi

Dopo aver accumulato una serie completa di pacchetti sullo stato della connessione, il router può costruire l'intero grafo della sottorete poiché è rappresentato ogni collegamento. In realtà ogni collegamento è rappresentato due volte, una per ogni direzione. Si può fare la media dei due valori, oppure ognuno può essere utilizzato separatamente.

A questo punto si può eseguire localmente l'algoritmo di Dijkstra per costruire il percorso più breve verso tutte le possibili destinazioni. I risultati di questo algoritmo possono essere installati nelle tabelle di routing, prima di riprendere le normali operazioni.

Per una sottorete con n nodi, ognuno dei quali ha k vicini, la memoria richiesta per conservare i dati in input è proporzionale a kn . Per sottoreti grandi questo può diventare un problema, così come il tempo di elaborazione, ma in molte situazioni pratiche il routing basato sullo stato dei collegamenti funziona bene.

Purtroppo con questo algoritmo (ma anche con altri), i problemi causati dall'hardware e dal software possono provocare disastri. Per esempio, se un router afferma di avere una linea che in realtà non ha, o dimentica di indicare una linea che invece esiste, il grafo della sottorete risulterà scorretto. Se un router non riesce a inoltrare i pacchetti o li danneggia durante l'inoltro, s'inergeranno guai. Infine, se esaurisce la memoria o esegue un calcolo di routing errato c'è da aspettarsi il peggio. Quando la sottorete inizia a essere composta da decine o centinaia di migliaia di nodi, non è più trascurabile la probabilità che saltuariamente un router si guasti. Il trucco sta nel tentare di prepararsi a limitare i danni prima che accada l'inevitabile. Perlman (1988) parla di questi problemi e descrive in dettaglio le soluzioni.

Il routing basato sullo stato dei collegamenti è molto utilizzato nelle reti reali, perciò è utile spendere qualche parola esemplificando alcuni protocolli che lo utilizzano. Il protocollo OSPF, ampiamente utilizzato in Internet, utilizza l'algoritmo basato sullo stato dei collegamenti ed è descritto nel Paragrafo 5.6.4.

altro protocollo basato sullo stato dei collegamenti è **IS-IS** (*Intermediate System-Intermediate System*), progettato inizialmente per DECnet e successivamente adottato da ISO il suo CLNP (il protocollo dello strato network senza connessione). Da allora è stato modi-to per gestire anche altri protocolli, tra i quali IP. IS-IS è usato in alcune dorsali Internet (usa la vecchia dorsale NSFNET) e in alcuni sistemi cellulari digitali quali CDPD. Novell Ware utilizza una variante minore di IS-IS (NLSP) per il routing dei pacchetti IPX. Tuttavia, IS-IS distribuisce una pianta della topologia dei router, da cui è possibile calcolare i percorsi più brevi. Ogni router annuncia, nelle sue informazioni sullo stato collegamenti, gli indirizzi dello strato network che è in grado di raggiungere direttamente. Questi indirizzi possono essere IP, IPX, AppleTalk o di qualunque altro tipo; IS-IS anche supportare più protocolli di strato network nello stesso momento.

Le innovazioni progettate per IS-IS sono state adottate da OSPF (OSPF è stato pro-ato diversi anni dopo IS-IS). Tra queste spiccano un metodo (che si stabilizza automaticamente) per il flooding degli aggiornamenti sullo stato della connessione, il concetto di rou-redefinito su una LAN e il metodo per calcolare e supportare la divisione dei percorsi elettriche molteplici. Di conseguenza, c'è poca differenza tra IS-IS e OSPF; la più impor-ta è che IS-IS è codificato in modo tale da rendere semplice e naturale il trasporto simulo di informazioni relative a più protocolli di strato network, funzionalità non supportata OSPF. Questo vantaggio è prezioso soprattutto in grandi ambienti multiprotocollo.

6 Routing gerarchico

Dimensione delle tabelle di routing cresce proporzionalmente alla dimensione della rete. La crescita della tabella non soltanto consuma la memoria del router, ma aumenta il tempo che la CPU impiega ad analizzare i dati e la banda necessaria per inviare le informazioni sullo stato. La rete può crescere al punto che non è più possibile per i router avere voce per ogni altro router; in questo caso il routing dovrà essere fatto gerarchicamente in una rete telefonica.

ndo si utilizza il routing gerarchico, i router sono divisi in **regioni**: ogni router cono-uti i dettagli relativi al routing dei pacchetti diretti a destinazioni nella stessa regio-na non sa nulla della struttura interna delle altre regioni. Quando si interconnettono diverse è naturale considerare ogni rete come una regione separata; in questo modo, ieri di una rete non sono costretti a conoscere la struttura topologica delle altre reti. Reti enormi può essere insufficiente una struttura gerarchica a due livelli; può essere ssario raggruppare le regioni in cluster, i cluster in zone, le zone in gruppi e così via, a esaurire tutti i nomi delle aggregazioni. Come esempio di gerarchia multilivello, si consideri il modo in cui un pacchetto potrebbe essere inoltrato da Berkeley, California, a ndi, città del Kenya. Il router di Berkeley conosce i dettagli della topologia della California, ma invia al router di Los Angeles tutto il traffico diretto all'esterno dello stato. Il router di Los Angeles è in grado di instradare il traffico diretto agli altri router statunitensi, ma invia al router di New York i dati diretti all'estero. Il router di New York è programato per dirigere tutto il traffico al router del paese di destinazione responsabile della parte del traffico estero, che in questo caso potrebbe trovarsi a Nairobi. Infine, il pacchetto da Nairobi troverà la strada per raggiungere il Kenya e quindi Malindi.

La Figura 5.15 fornisce un esempio quantitativo di routing in una gerarchia a due livelli contenente cinque regioni. La tabella di routing completa del router 1A contiene 17 voci, come mostrato nella Figura 5.15(b). Quando il routing è fatto gerarchicamente, come nella Figura 5.15(c), ci sono voci per tutti i router locali (come prima) ma tutte le altre regioni sono condensate in un singolo router, perciò tutto il traffico per la regione 2 viene tra-smesso attraverso la linea 1B 2A, ma il resto del traffico remoto passa per la linea 1C 3B. Il routing gerarchico ha ridotto il numero di voci nella tabella da 17 a 7. Lo spazio rispar-miato nello spazio delle tabelle aumenta al crescere del rapporto tra numero di regioni e numero di router per regione.

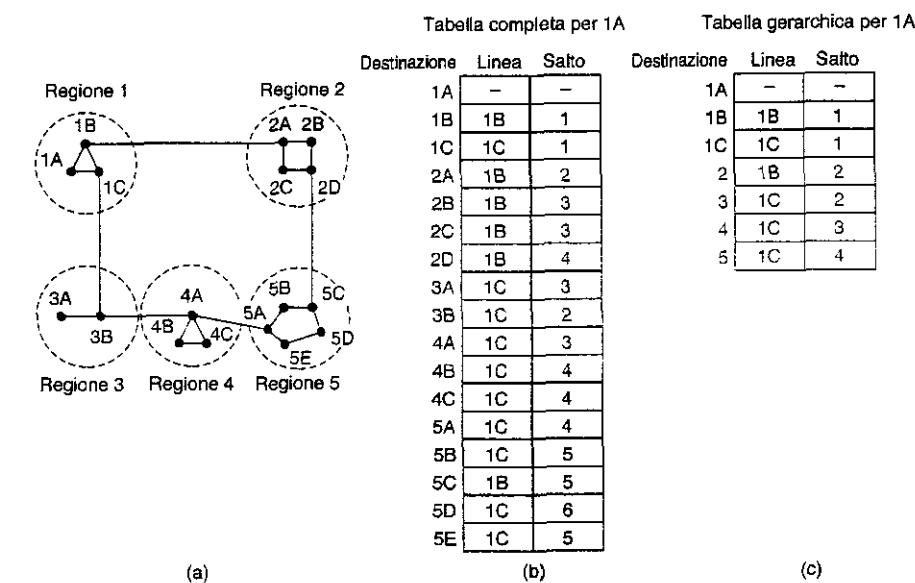


Figura 5.15. Routing gerarchico.

Sfortunatamente, questi risparmi di spazio hanno un costo: la penalità da pagare è rappre-sentata dalla crescita della lunghezza dei percorsi. Per esempio, il percorso migliore che collega 1A a 5C passa attraverso la regione 2 ma, con il routing gerarchico, tutto il traffico diretto alla regione 5 viene inviato alla regione 3 perché quella è la strada migliore per la maggior parte delle destinazioni nella regione 5.

Quando una singola rete diventa molto grande è lecito domandarsi quanti livelli dovrebbe avere la gerarchia. Per esempio, si consideri una sottorete composta da 720 router. Se non ci fosse alcuna gerarchia, ogni router dovrebbe gestire una tabella composta da 720 voci. Se la sottorete fosse divisa in 24 regioni ognuna contenente 30 router, ogni router dovrebbe gestire una tabella composta da 30 voci locali più 23 voci remote per un totale di 53 voci. Scegliendo una gerarchia a tre livelli, con otto cluster ognuno contenente 9 regioni

i 10 router, le tabelle dei router conterrebbero 10 voci per i router locali, 8 voci per i router delle altre regioni nello stesso cluster e 7 voci per cluster distanti, per un totale di 25 voci. Kamoun e Kleinrock (1979) hanno scoperto che il numero ottimale di livelli per una sottorete di N router è uguale a $\ln N$, per un totale di $e \ln N$ voci per router. I due hanno anche dimostrato che l'aumento della lunghezza del percorso medio reale causato dal routing gerarchico è sufficientemente piccolo da renderlo il più delle volte accettabile.

2.7 Routing broadcast

In alcune applicazioni gli host hanno bisogno d'inviare messaggi a molti o a tutti gli altri host. Per esempio, un servizio di distribuzione di rapporti meteorologici, di aggiornamenti nel mercato azionario o programmi radio dal vivo potrebbero funzionare meglio trasmettendo in modalità broadcast a tutti i computer, lasciando che quelli interessati leggano i dati. La trasmissione contemporanea di un pacchetto a tutte le destinazioni è chiamata **trasmissione broadcast**. Sono stati proposti diversi metodi per ottenere questo risultato.

Un metodo di trasmissione broadcast che non richiede l'implementazione di alcuna funzionalità speciale nella sottorete è quello in cui la sorgente invia un pacchetto distinto a ogni destinazione. Questo metodo non si limita a sprecare banda, ma obbliga la sorgente a possedere una lista completa di tutte le destinazioni. All'atto pratico questa può essere unica possibilità, ma tra tutti i metodi è il meno desiderabile.

Un altro ovvio candidato è il meccanismo di flooding. Sebbene non si adatti bene alla comunicazione ordinaria punto-punto, nel caso della trasmissione broadcast potrebbe essere preso in seria considerazione, specialmente quando nessun altro metodo è applicabile. Il problema della tecnica broadcast basata sul flooding è lo stesso dell'algoritmo di flooding punto-punto: genera troppi pacchetti e consuma troppa banda.

Un terzo algoritmo è quello del **multidestination routing** (routing a più destinazioni). Se si adotta questo metodo, ogni pacchetto contiene una lista delle destinazioni, o una mappa bit che indica le destinazioni desiderate. Quando riceve un pacchetto, il router controlla tutte le destinazioni per determinare l'insieme di linee di trasmissione richieste (una linea di trasmissione è necessaria se rappresenta il percorso migliore verso almeno una delle destinazioni). Il router genera una nuova copia del pacchetto per ogni linea di output attiva e include in ogni pacchetto solo quelle destinazioni che si trovano su quella linea; in altri termini l'insieme delle destinazioni viene diviso tra le linee di trasmissione. Dopo un numero sufficiente di salti, ogni pacchetto conterrà una sola destinazione e potrà essere trattato come un pacchetto normale. È come se il routing a più destinazioni utilizzasse pacchetti indirizzati separatamente, ma quando diversi pacchetti devono seguire lo stesso percorso, uno solo paga il prezzo completo mentre gli altri viaggiano gratis.

Quarto algoritmo di trasmissione broadcast fa esplicito uso del sink tree del router che ha iniziato la trasmissione (o di qualunque altro spanning tree utile). Uno **spanning tree** è un insieme che comprende tutti i router ma che non contiene cicli. Se ogni router sa quali sue linee appartengono allo spanning tree, allora può copiare un pacchetto broadcast in uscita su tutte le linee dello spanning tree esclusa quella d'ingresso. Questo metodo usa in modo eccellente la banda e genera in assoluto il minimo numero di pacchetti necessari per

svolgere il lavoro. L'unico problema sta nel fatto che, per rendere applicabile il metodo, ogni router deve conoscere uno spanning tree. Qualche volta questa informazione è disponibile (per esempio nel caso del routing basato sullo stato dei collegamenti), altre volte invece non lo è (per esempio nel caso del routing basato sul vettore delle distanze).

L'ultimo algoritmo usato per trasmettere in modalità broadcast tenta di approssimare il comportamento dell'algoritmo precedente anche quando i router non sanno nulla degli spanning tree. L'idea, chiamata **reverse path forwarding**, è straordinariamente semplice una volta che è stata capita. Quando riceve un pacchetto broadcast, il router verifica se il pacchetto è giunto attraverso la linea che normalmente è utilizzata per inviare i pacchetti *alla* sorgente della trasmissione broadcast. In caso affermativo, c'è una forte probabilità che il pacchetto broadcast stesso abbia seguito il percorso migliore dal router, e che perciò sia la prima copia arrivata al router. In questo caso, il router inoltra le copie del pacchetto attraverso tutte le linee esclusa quella di input. Se al contrario il pacchetto broadcast è giunto attraverso una linea diversa da quella che viene preferita per raggiungere la sorgente, il pacchetto è scartato in quanto è probabile che si tratti di un duplido.

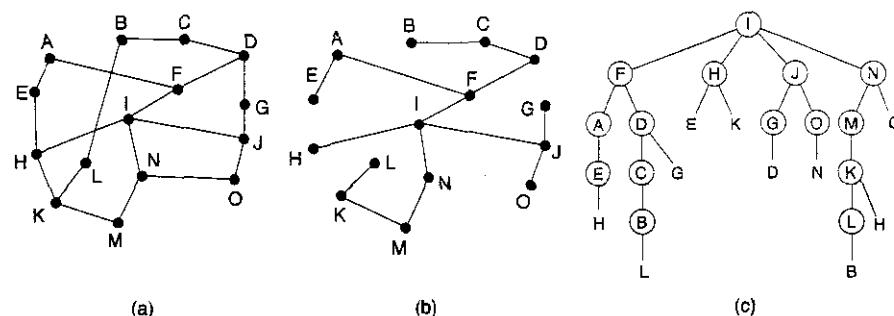


Figura 5.16. Inoltro a percorso inverso. (a) Una sottorete. (b) Un sink tree. (c) L'albero realizzato mediante l'inoltro a percorso inverso.

La Figura 5.16 mostra un esempio di reverse path forwarding. La parte (a) mostra una sottorete, la parte (b) mostra un sink tree per il router *I* della sottorete e la parte (c) mostra in che modo funziona l'algoritmo del percorso inverso. Al primo salto, *I* invia pacchetti a *F*, *H*, *J* e a *N*, come indicato nella seconda riga della struttura ad albero. Ognuno di questi pacchetti giunge attraverso il percorso preferito associato a *I* (supponendo che il percorso preferito cada lungo il sink tree) ed è perciò indicato da un cerchietto intorno alla lettera. Al secondo salto vengono generati otto pacchetti: ogni router che ha ricevuto un pacchetto al primo salto ne genera due. Tutti e otto arrivano a router non visitati precedentemente, e cinque di questi giungono attraverso la linea preferita. Dei sei pacchetti generati al terzo salto, solo tre giungono attraverso il percorso preferito (*C*, *E* e *K*); gli altri sono duplicati. Dopo cinque salti e 24 pacchetti la trasmissione broadcast termina, mentre il sink tree impiega esattamente quattro salti e 14 pacchetti.

Il vantaggio principale del reverse path forwarding è che si tratta di un sistema ragionevolmente efficiente e facile da implementare. Non richiede che i router conoscano gli

spanning tree né che si debba elaborare una lista di destinazione o una mappa di bit per ogni pacchetto broadcast, come invece accade nel routing a più destinazioni. Inoltre, non è necessario alcun meccanismo speciale d'interruzione del processo, come richiesto dal flooding (che adotta un contatore di salto in ogni pacchetto e una conoscenza a priori del diametro della sottorete, oppure un elenco di pacchetti già visti per ogni sorgente).

5.2.8 Routing multicast

Alcune applicazioni richiedono che processi molto separati funzionino insieme a gruppi, per esempio come nel caso di un gruppo di processi che implementa un sistema database distribuito. In queste situazioni, spesso è necessario che un processo invii un messaggio a tutti gli altri membri del gruppo. Se il gruppo è piccolo si può semplicemente inviare un messaggio punto-punto a tutti gli altri membri, ma se il gruppo è grande questa strategia diventa costosa. Qualche volta si può utilizzare la trasmissione broadcast, ma l'utilizzo del broadcast per informare 1.000 macchine in una rete composta da milioni di nodi è inefficiente, perché la maggior parte dei ricevitori non è interessata a quel messaggio (o peggio ancora, potrebbe essere molto interessata al contenuto del messaggio ma non dovrebbe ricevere quei dati). Così è necessario trovare il modo di inviare messaggi a gruppi ben definiti che, pur essendo numericamente grandi, sono piccoli se confrontati con l'intera rete. La trasmissione di un messaggio a un gruppo di questo tipo è definita **multicast** e il suo algoritmo di routing è chiamato **routing multicast**. Questo paragrafo descrive un modo di eseguire il routing multicast. Per maggiori informazioni, consultare (Chu et al., 2000; Costa et al. 2001; Kasera et al., 2000; Madruga and Garcia-Luna-Aceves, 2001; Zhang and Ryu, 2001).

La trasmissione multicast richiede la gestione dei gruppi: occorre un sistema che consente di creare e distruggere i gruppi e che permetta ai processi di entrare e uscire dai gruppi. Il modo in cui queste operazioni sono eseguite non riguarda l'algoritmo di routing; ciò che importa è che quando si unisce a un gruppo, il processo informi il suo host di questo fatto. È importante che i router sappiano quali sono gli host appartenenti a ogni gruppo, quindi li host devono comunicare ai loro router i cambi di gruppo oppure i router devono interrogare periodicamente i loro host. In entrambi i casi, i router scoprono i gruppi di appartenenza dei loro host. I router comunicano con i loro vicini, perciò le informazioni si propagano attraverso la sottorete.

Per ottenere una trasmissione multicast, ogni router elabora uno spanning tree che copre tutti i altri router. Per esempio, nella Figura 5.17(a) appaiono due gruppi, 1 e 2. Alcuni router sono collegati agli host che appartengono a uno o a entrambi i gruppi, come mostrato nella figura. La Figura 5.17(b) rappresenta uno spanning tree relativo al router posto più a sinistra. Quando un processo invia un pacchetto multicast a un gruppo, il primo router esamina il suo spanning tree e lo accorcia, rimuovendo tutte le linee che non conducono agli host che non sono membri di quel gruppo. Nell'esempio, la Figura 5.17(c) mostra lo spanning tree accorciato associato al gruppo 1. In modo analogo, la Figura 5.17(d) mostra lo spanning tree accorciato associato al gruppo 2. I pacchetti multicast sono inoltrati solo attraverso lo spanning tree appropriato.

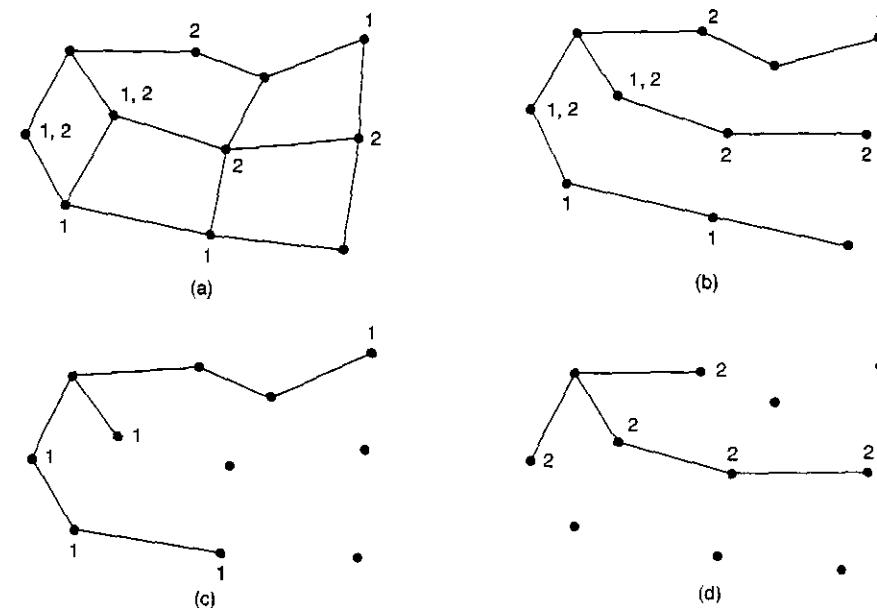


Figura 5.17. (a) Una rete. (b) Uno spanning tree per il router più a sinistra. (c) Un albero multicast per il gruppo 1. (d) Un albero multicast per il gruppo 2.

È possibile ridurre lo spanning tree in molti modi. Quello più semplice può essere utilizzato se si adotta il routing basato sullo stato dei collegamenti e se ogni router conosce la topologia completa, quindi sa quali host appartengono a ciascun gruppo. In queste circostanze lo spanning tree si può accorciare partendo dalla fine di ogni percorso, procedendo verso il nodo principale (la radice) e rimuovendo tutti i router che non appartengono al gruppo in questione.

Quando si usa il routing basato sul vettore delle distanze si deve seguire una strategia di riduzione diversa. L'algoritmo di base è quello del reverse path forwarding, ma ogni volta che un router che non ha host interessati a un particolare gruppo (e non ha connessioni dirette ad altri router) riceve un messaggio multicast diretto a quel gruppo, risponde con un messaggio PRUNE che avvisa il trasmittente di non inviare più alcun messaggio multicast per quel gruppo. Può rispondere con un messaggio PRUNE anche un router che riceve messaggi di questo tipo su tutte le sue linee, e non ha host appartenenti a quel gruppo. In questo modo, la sottorete viene ridotta in modo ricorsivo.

Uno svantaggio potenziale di questo algoritmo è che si adatta male alle grandi reti. Si supponga che la rete abbia n gruppi, ognuno composto da un numero medio di m membri. Per ogni gruppo si devono memorizzare m spanning tree accorciati, per un totale di mn strutture ad albero. Quando esistono molti gruppi di grosse dimensioni, l'archiviazione di tutte le strutture ad albero può richiedere molta memoria.

Un'architettura alternativa utilizza strutture chiamate **core-based tree** (Ballardie et al., 1993). In questo caso, viene elaborato un singolo spanning tree per gruppo, con il nodo principale (la radice, ovvero il nucleo della struttura) vicino alla metà del gruppo. Per inviare un messaggio multicast, un host trasmette i dati al nodo principale, che poi esegue il multicast lungo lo spanning tree. Sebbene questa struttura ad albero non sia ottimale per tutte le risorse, la riduzione del fabbisogno per ciascun gruppo (in termini di consumo di memoria) da m a una sola struttura ad albero rappresenta un grande risparmio.

5.2.9 Routing per host mobili

Oggi milioni di persone utilizzano i computer portatili; generalmente questi utenti desiderano poter leggere la posta elettronica e accedere ai loro filesystem standard da qualunque punto del mondo. Gli host mobili introducono un nuovo problema: per instradare un pacchetto a un host mobile, la rete deve prima di tutto individuare il computer di destinazione. L'incorporamento di host mobili nelle reti è un tema molto recente, ma questo paragrafo esamina alcune delle problematiche e delle possibili soluzioni.

Il modello che i progettisti delle reti usano per rappresentare il mondo è mostrato nella Figura 5.18. L'immagine mostra una WAN composta da router e da host. MAN, LAN e celle wireless del tipo descritto nel Capitolo 2 sono tutte collegate alla WAN.

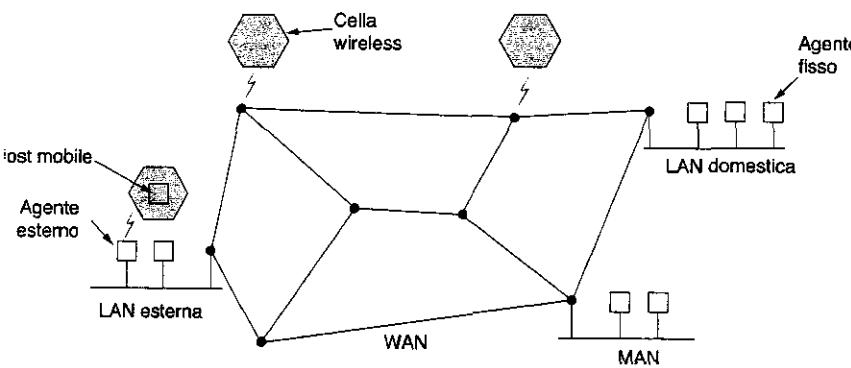


Figura 5.18. Una WAN a cui sono collegate celle LAN, MAN e wireless.

Gli host che non si spostano mai sono detti fissi, e si collegano alla rete attraverso cavi a rame e fibre ottiche. Si possono poi distinguere altri due tipi di host: gli host migratori sono fondamentalmente host fissi che di tanto in tanto si spostano da un sito fisso a un altro, ma usano la rete solo quando sono collegati fisicamente a essa; gli host roaming eseguono elaborazioni mentre si muovono e vogliono mantenere le loro connessioni durante gli spostamenti. In questo capitolo il termine di **host mobili** verrà utilizzato per indicare entrambe le categorie, ossia tutti gli host che pur essendo lontani dalla loro casa vogliono rimanere collegati. Si presuppone che tutti gli host abbiano una **ubicazione domestica** (LAN di appartenenza) che non cambia mai. Gli host hanno anche un indi-

rizzo domestico permanente che può essere utilizzato per determinare la loro locazione domestica, proprio come il numero di telefono 1-212-5551212 indica gli Stati Uniti (prefisso del paese 1) e Manhattan (prefisso 212). Il routing nei sistemi con host mobili ha come obiettivo l'invio dei pacchetti a host mobili usando i loro indirizzi domestici, consentendo ai pacchetti di raggiungere gli host mobili in modo efficiente ovunque si trovino i computer di destinazione. La parte difficile sta ovviamente nel trovarli.

Nel modello rappresentato nella Figura 5.18, il mondo è diviso (geograficamente) in piccole unità, chiamate aree; in genere un'area è una LAN o una cella wireless. Ogni area dispone di uno o più **agenti esterni**, processi utilizzati per tenere traccia di tutti gli host mobili che visitano l'area. Inoltre, ogni area ha un **agente fisso**, che tiene traccia degli host che hanno casa nell'area ma che in quel momento si trovano in un'altra area.

Quando un nuovo host entra nell'area collegandosi a essa (per esempio connettendosi alla LAN oppure semplicemente vagabondando nella cella), il suo computer deve registrarsi presso l'agente esterno. La procedura di registrazione funziona in genere nel modo qui di seguito elencato.

1. Ogni agente esterno trasmette periodicamente in modalità broadcast un pacchetto che annuncia la sua esistenza e il suo indirizzo. Un host mobile appena arrivato può aspettare uno di questi messaggi, ma se nessuno di essi arriva abbastanza velocemente, l'host mobile può trasmettere un pacchetto broadcast che contiene il seguente messaggio: "C'è qualche agente esterno nei paraggi?".
2. L'host mobile si registra presso l'agente esterno, indicando il proprio indirizzo di casa, l'indirizzo data link corrente e alcune informazioni di protezione.
3. L'agente esterno contatta l'agente fisso dell'host mobile dicendo: "Uno dei tuoi host si trova qui." Il messaggio che l'agente esterno invia all'agente fisso contiene l'indirizzo di rete dell'agente esterno e include anche le informazioni di protezione necessarie per convincere l'agente fisso che l'host mobile si trova realmente nell'area indicata.
4. L'agente fisso esamina le informazioni di protezione, che includono un contrassegno temporale per dimostrare che i dati sono stati generati pochi secondi prima. Se è tutto corretto, l'agente fisso dice all'agente esterno di procedere.
5. Quando riceve il messaggio di conferma dall'agente fisso, l'agente esterno crea una voce nelle sue tabelle e informa l'host mobile che la registrazione è stata completata.

Idealmente, l'host mobile deve annunciare l'abbandono dell'area in modo da consentire l'annullamento della registrazione, ma molti utenti spengono brutalmente i loro computer quando hanno finito di lavorare.

Quando è trasmesso a un host mobile, il pacchetto è instradato verso la LAN domestica dell'host perché è questo che l'indirizzo dice di fare, come illustrato nella fase 1 della Figura 5.19. In questo caso il trasmittente, situato a nord ovest della città di Seattle, desidera inviare un pacchetto a un host che normalmente si trova negli Stati Uniti, nella città

di New York. I pacchetti trasmessi verso la LAN dell'host mobile sono intercettati dall'agente fisso di New York; l'agente fisso cerca la nuova posizione (temporanea) dell'host mobile e trova l'indirizzo dell'agente esterno che sta gestendo l'host mobile, per esempio a Los Angeles.

L'agente fisso a questo punto fa due cose. Primo, incapsula il pacchetto nel carico utile di un pacchetto più esterno e invia quest'ultimo all'agente esterno (fase 2 nella Figura 5.19); questo meccanismo, chiamato tunneling, verrà descritto in dettaglio più avanti. Dopo aver ricevuto il pacchetto incapsulato, l'agente esterno rimuove il pacchetto originale dal carico utile e lo trasmette all'host mobile come frame data link.

Come seconda azione, l'agente fisso dice al trasmittente di inviare da quel momento in poi all'agente esterno tutti i pacchetti diretti all'host mobile, incapsulando i dati nel carico utile di nuovi pacchetti (fase 3). I pacchetti successivi possono quindi essere instradati direttamente all'host attraverso l'agente esterno (fase 4), scavalcando completamente la locazione domestica.

I vari schemi che sono stati proposti differiscono in molti aspetti. Primo, bisogna definire

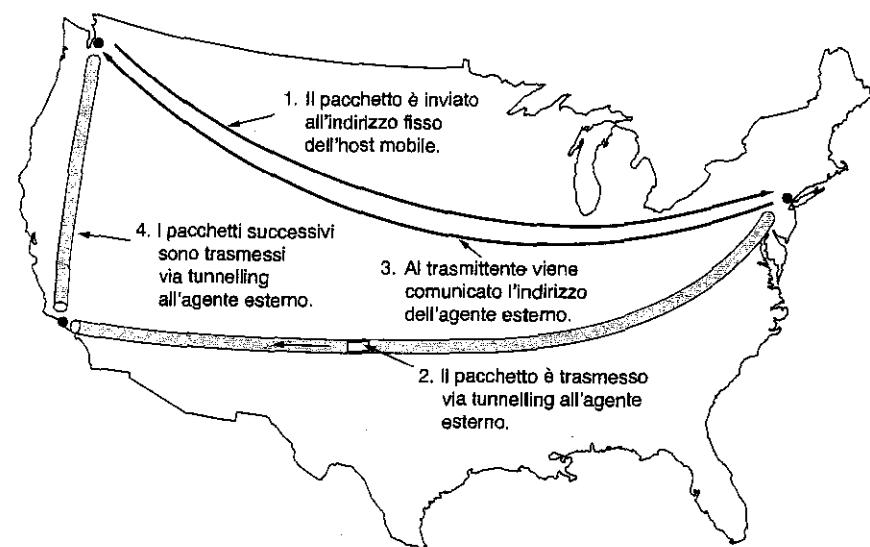


Figura 5.19. Routing di pacchetto per host mobili.

il funzionamento di questo protocollo va attuato dai router e quanto dagli host e, nel secondo caso, quale strato degli host. Secondo, in qualche schema i router lungo la strada registrano i indirizzi mappati, così da poter intercettare e reindirizzare il traffico anche prima che giunga la locazione domestica. Terzo, in alcuni schemi a ogni ospite è assegnato un indirizzo temporaneo unico; in altri l'indirizzo temporaneo si riferisce a un agente che gestisce il traffico per tutti gli ospiti. Quarto, gli schemi differiscono per il modo in cui deviano verso una destinazione i pac-

chetti indirizzati a un'altra destinazione. Per esempio, alcuni cambiano semplicemente l'indirizzo di destinazione e ritrasmettono il pacchetto modificato. In alternativa l'intero pacchetto, indirizzo domestico e tutto il resto, può essere incapsulato dentro il carico utile di un altro pacchetto trasmesso all'indirizzo temporaneo. Infine, gli schemi si distinguono anche per gli aspetti inerenti la protezione. In generale, quando riceve un messaggio del tipo "A partire da adesso, per favore, trasmetti a me tutta la posta elettronica indirizzata a Stefania", l'host o il router potrebbe voler conoscere l'identità dell'altra parte impegnata nella trasmissione e appurare se è una buona idea soddisfare la richiesta. Diversi protocolli per host mobili sono descritti e confrontati in (Hac and Guo, 2000; Perkins, 1998a; Snoeren and Balakrishnan, 2000; Solomon, 1998; e Wang and Chen, 2001).

5.2.10 Routing nelle reti ad hoc

Il paragrafo precedente ha spiegato come funziona il routing quando gli host sono mobili ma i router sono fissi. Un caso ancora più estremo è quello in cui i router stessi sono mobili. Tra le possibilità ci sono:

1. veicoli militari che si trovano sul campo di battaglia e che non hanno a disposizione un'infrastruttura esistente
2. una flotta di navi nell'oceano
3. squadre di soccorso che operano su un territorio dove un terremoto ha distrutto l'infrastruttura
4. un raduno di persone con computer portatili in un'area senza copertura fissa

In tutti questi casi (e anche in altri) ogni nodo è composto da un router e da un host, di solito collocati sullo stesso computer. Le reti costituite da nodi collocati l'uno vicino all'altro sono chiamate **reti ad hoc** o **MANET** (*Mobile Ad hoc NETworks*), e sono brevemente esaminate in questo paragrafo; per ulteriori dettagli consultare (Perkins, 2001).

Ciò che rende le reti ad hoc diverse dalle reti cablate è che tutte le regole abituali relative alle topologie fisse, ai vicini fissi e noti, alla relazione fissa tra indirizzo IP e posizione e così via, vengono improvvisamente buttate dalla finestra. I router possono andare e venire o apparire in nuovi luoghi in un battito di ciglia. Con una rete cablata, se un router ha un percorso valido diretto verso una particolare destinazione, il percorso continua a essere valido indefinitamente (a meno di un guasto nel sistema). Con una rete ad hoc, la topologia può cambiare in qualunque momento, perciò la desiderabilità e anche la validità dei percorsi può cambiare spontaneamente senza alcun preavviso. Inutile dirlo, queste circostanze rendono il routing nelle reti ad hoc molto diverso dal routing nelle controparti fisse. Sono stati proposti diversi algoritmi di routing per le reti ad hoc. Uno dei più interessanti è l'algoritmo di routing **AODV** (*Ad hoc On-demand Distance Vector*), (Perkins and Royer, 1999). È un lontano parente dell'algoritmo basato sul vettore delle distanze di Bellman-Ford, adattato per funzionare in un ambiente mobile; l'algoritmo tiene conto della banda limitata e della scarsa durata delle batterie dei dispositivi che operano all'interno di que-

sto ambiente. Un'altra caratteristica insolita è che è un algoritmo a richiesta, ossia determina un percorso diretto a una destinazione solo quando qualcuno tenta di inviare pacchetti verso quella destinazione. Ecco come funziona.

Route discovery

In qualunque istante, una rete ad hoc può essere descritta da un grafo dei nodi (router + host). Due nodi sono collegati (ossia hanno un arco che li unisce) se possono comunicare direttamente usando la loro sezione radio. Poiché uno dei due può avere un trasmettitore più potente dell'altro, è possibile che *A* sia collegato a *B* ma che *B* non sia collegato ad *A*, ma per semplicità sarà fatta l'ipotesi che tutte le connessioni siano simmetriche. È bene notare poi che il semplice fatto di trovarsi l'uno nel campo di ricezione dell'altro non implica automaticamente un collegamento tra due nodi. Ci possono essere edifici, colline o altri ostacoli che bloccano la comunicazione.

Per descrivere questo algoritmo si consideri la rete ad hoc mostrata nella Figura 5.20, dove un processo del nodo *A* cerca di inviare un pacchetto al nodo *I*. L'algoritmo AODV mantiene in ogni nodo una tabella, indicizzata per destinazione, che fornisce informazioni su quella destinazione; tra esse c'è l'indicazione dei vicini a cui trasmettere i pacchetti per raggiungerla. Si supponga che *A* cerchi nella sua tabella e non trovi una voce per *I*, pertanto *A* deve scoprire un percorso diretto a *I*. La proprietà di scoprire i percorsi solo quando si ha la necessità di raggiungerli rende l'algoritmo "a richiesta".

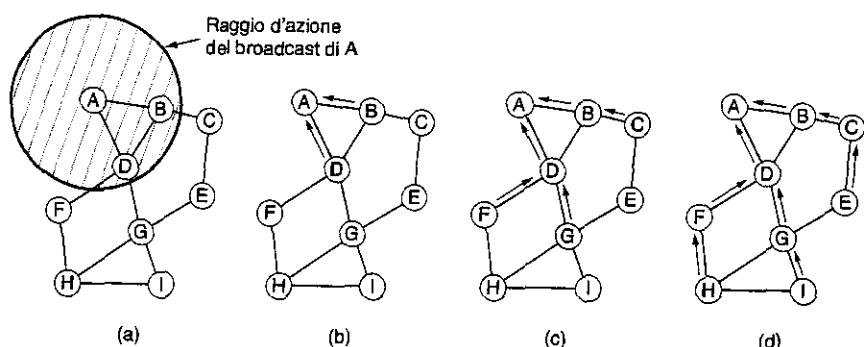


Figura 5.20. (a) Raggio d'azione del broadcast di *A*. (b) Dopo che *B* e *D* hanno ricevuto il broadcast di *A*. (c) Dopo che *C*, *F* e *G* hanno ricevuto il broadcast di *A*. (d) Dopo che *E*, *H* e *I* hanno ricevuto il broadcast di *A*. I nodi ombreggiati rappresentano i nuovi destinatari. Le frecce mostrano i percorsi inversi possibili.

Per individuare *I*, *A* costruisce uno speciale pacchetto ROUTE REQUEST e lo trasmette in modalità broadcast. Il pacchetto raggiunge *B* e *D*, come mostrato nella Figura 5.20(a). *B* e *D* nel grafo appaiono collegati ad *A* perché possono ricevere le comunicazioni provenienti da *A*. *F*, per esempio, non è unito ad *A* da alcun arco perché non è in grado di ricevere il segnale radio di *A*. Il formato del pacchetto ROUTE REQUEST è mostrato nella Figura 5.21. Contiene gli indirizzi d'origine e di destinazione, in genere indirizzi IP, che

identificano chi sta cercando chi. Contiene anche *request ID*, che è un contatore locale gestito separatamente da ogni nodo e incrementato ogni volta che si trasmette in modalità broadcast un pacchetto ROUTE REQUEST. Insieme, i campi *source address* (indirizzo sorgente) e *request ID* identificano in modo univoco il pacchetto ROUTE REQUEST e consentono ai nodi di scartare i duplicati ricevuti. Oltre al contatore *request ID*, ogni nodo gestisce anche un secondo contatore di sequenza incrementato ogni volta che viene trasmesso un pacchetto ROUTE REQUEST (o una risposta associata a un ROUTE REQUEST di qualcun altro). Funziona un po' come un orologio, ed è utilizzato per distinguere i nuovi percorsi da quelli vecchi. Il quarto campo mostrato nella Figura 5.21 rappresenta il contatore della sequenza di *A*; il quinto campo è il valore più recente del numero di sequenza di *I* che *A* ha ricevuto (0 indica che non ne ha ricevuto alcuno). L'utilizzo di questi campi verrà chiarito fra poco. Il campo finale, *hop counter*, tiene traccia del numero di salti compiuti dal pacchetto. Il valore iniziale è 0.

Source address	Request ID	Destination address	Numero sequenza d'origine	Numero sequenza di destinazione	Hop counter
----------------	------------	---------------------	---------------------------	---------------------------------	-------------

Figura 5.21. Formato di un pacchetto ROUTE REQUEST.

Quando raggiunge un nodo (*B* e *D*, in questo caso), il pacchetto ROUTE REQUEST è elaborato come descritto di seguito.

1. Viene cercata la coppia di valori (*source address*, *request ID*) nella tabella della cronologia locale per vedere se questa richiesta è già stata ricevuta ed elaborata. In caso affermativo si tratta di un duplicato, che può essere scartato interrompendo l'elaborazione. Se non è un duplicato, la coppia di valori viene inserita nella tabella in modo da poter rifiutare futuri duplicati, e l'elaborazione continua.
2. Il ricevitore cerca la destinazione nella propria tabella di routing. Se individua un percorso recente diretto alla destinazione, trasmette all'origine un pacchetto ROUTE REPLY che spiega come raggiungere la destinazione (sostanzialmente, il messaggio è "Usa me"). Recent significa che il *numero sequenza di destinazione* archiviato nella tabella di routing è maggiore o uguale al *numero sequenza di destinazione* contenuto nel pacchetto ROUTE REQUEST. Se il valore è minore, il percorso memorizzato è più vecchio del percorso precedente conosciuto dall'origine, e in questo caso si passa alla fase 3.
3. Poiché non conosce un percorso recente diretto alla destinazione, il ricevitore incrementa il campo *hop counter* e trasmette nuovamente in modalità broadcast il pacchetto ROUTE REQUEST. Il ricevitore estrae anche i dati dal pacchetto e li conserva in una nuova voce nella sua tabella di routing inverso. Queste informazioni saranno utilizzate per costruire il percorso inverso, che consentirà successivamente alla risposta di tornare all'origine. Le frecce mostrate nella Figura 5.20 sono utilizzate per costruire il percorso inverso. Viene anche avviato un timer per la voce di percorso inverso appena creata; non appena scade, la voce è cancellata.

Né *B* né *D* sanno dove si trova *I*, perciò entrambi creano una voce di percorso inverso che punta verso *A*, come mostrato dalle frecce nella Figura 5.20, e trasmettono in modalità broadcast il pacchetto assegnando il valore 1 al campo *hop counter*. Il pacchetto trasmesso da *B* raggiunge *C* e *D*. *C* crea nella sua tabella di percorso inverso una voce associata al pacchetto prima di ritrasmettere i dati in modalità broadcast. Al contrario, *D* rifiuta il pacchetto in quanto duplicato. In modo analogo, la trasmissione broadcast di *D* viene rifiutata da *B*, ma il pacchetto di *D* è accettato da *F* e da *G* e archiviato, come mostrato nella Figura 5.20 (c). Quando la trasmissione broadcast arriva a *E*, *H* e *I*, finalmente il pacchetto ROUTE REQUEST raggiunge una destinazione che sa dove si trova *I*, ossia *I* stesso, come illustrato nella Figura 5.20 (d). Si noti che sebbene la trasmissione broadcast sia stata descritta in tre fasi discrete, le trasmissioni dai diversi nodi non sono coordinate in alcun modo.

In risposta alla richiesta appena giunta, *I* costruisce un pacchetto ROUTE REPLY simile a quello mostrato nella Figura 5.22. *Source address* (indirizzo d'origine), *destination address* (indirizzo di destinazione) e *hop counter* (contatore di salti) sono copiati dalla richiesta, il *destination sequence #* (numero sequenza di destinazione) è invece preso dal relativo contatore in memoria. Al campo *hop counter* è assegnato il valore 0. Il campo *lifetime* (vita utile) controlla la durata della validità del percorso. Questo pacchetto è trasmesso in modalità unicast al nodo di origine del pacchetto ROUTE REQUEST, in questo caso *G*. Quindi, seguendo il percorso inverso, raggiunge prima *D* e infine *A*. In ogni nodo, il *hop counter* è decrementato in modo che il nodo possa determinare la distanza della destinazione (*I*).

Source address	Destination address	Destination sequence #	Hop counter	Lifetime
----------------	---------------------	------------------------	-------------	----------

Figura 5.22. Formato di un pacchetto ROUTE REPLY.

Ogni nodo intermedio sulla via di ritorno esamina il pacchetto. I dati sono inseriti nella tabella di routing locale come percorso diretto a *I* se è soddisfatta almeno una delle seguenti condizioni:

1. non è noto alcun percorso diretto a *I*
2. il numero di sequenza associato a *I* nel pacchetto ROUTE REPLY è più grande del valore registrato nella tabella di routing
3. i numeri di sequenza sono uguali ma il nuovo percorso è più corto.

In questo modo, tutti i nodi del percorso inverso scoprono automaticamente il percorso verso *I*, grazie all'azione di ricerca intrapresa inizialmente da *A*. I nodi che hanno ricevuto il pacchetto REQUEST ROUTE originale ma che non si trovano sul percorso inverso (in questo esempio, *B*, *C*, *E*, *F* e *H*), scartano la voce nella tabella di routing inverso allo scadere del timer associato.

In una rete di grandi dimensioni l'algoritmo genera molte trasmissione broadcast, anche per destinazioni che sono vicine. Il numero di pacchetti broadcast può essere ridotto nel seguente modo. Il trasmittente assegna al campo *time to live* (durata) del pacchetto IP il valore che rappresenta il diametro atteso della rete; questo valore viene decrementato a ogni salto. Non appena il contatore raggiunge lo 0, il pacchetto invece di essere trasmesso in modalità broadcast viene scartato. Il processo di scoperta è quindi modificato: per individuare la destinazione, il trasmittente invia un pacchetto ROUTE REQUEST broadcast contenente il campo *time to live* impostato a 1. Se non arriva nessuna risposta in un tempo ragionevole, il trasmittente invia un altro pacchetto, questa volta con il campo *time to live* impostato a 2. I tentativi successivi usano i valori 3, 4, 5 e così via. In tal modo, la ricerca viene eseguita prima localmente e poi in un cerchio sempre più ampio.

Aggiornamenti del percorso

Poiché i nodi si possono spostare o spegnere, la topologia può cambiare spontaneamente. Per esempio, se nella Figura 5.20 *G* si spegne, *A* non si accorge che il percorso che sta utilizzando per raggiungere *I* non è più valido. L'algoritmo deve essere in grado di gestire anche queste situazioni. Periodicamente, ogni nodo trascorre in modalità broadcast un messaggio *Hello*; ognuno dei vicini deve rispondere a tale messaggio. Se un vicino non trascorre la risposta, il trasmittente sa che quel dispositivo si è spostato oltre il suo raggio d'azione e non è più collegato. In modo analogo, se tenta di inviare un pacchetto a un vicino che non risponde, il trasmittente scopre che il destinatario non è più disponibile. Questa informazione è utilizzata per eliminare i percorsi che non sono più validi. Per ogni possibile destinazione, ogni nodo *N* tiene traccia dei vicini che gli hanno inviato un pacchetto per quella destinazione durante gli ultimi *DT* secondi. Questi sono considerati i *vicini attivi* di *N* per quella destinazione. Per far questo, *N* usa una tabella di routing indicizzata per destinazione; la tabella riporta il nodo in uscita da utilizzare per raggiungere la destinazione, il numero di salti richiesti per raggiungere la destinazione, il numero sequenza di destinazione più recente e l'elenco dei vicini attivi per quella destinazione. La Figura 5.23(a) mostra una tabella di routing relativa al nodo *D* per la topologia usata come esempio.

Destina- zione	Salto successivo	Distanza	Vicini attivi	Altri campi
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)

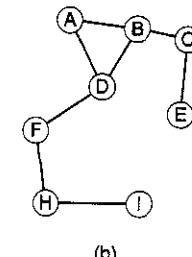


Figura 5.23. (a) Tabella di routing di *D* prima del blocco di *G*. (b) Il grafo dopo il blocco di *G*.

quando uno dei vicini di N diventa irraggiungibile, il nodo controlla la propria tabella di routing per vedere quali destinazioni sono associate ai percorsi che usano il vicino appena scomparso. Per ognuno di questi percorsi, i vicini attivi sono informati che il loro percorso attraverso N non è più valido e che deve essere eliminato dalle tabelle di routing. Ogni vicino attivo passa quindi l'informazione ai suoi vicini attivi e così via, in modo cattivo, fino a quando tutti i percorsi associati al nodo scomparso non sono eliminati dalle tabelle di routing.

Un esempio di gestione di un percorso si consideri il caso precedente, immaginando che improvvisamente G si spenga. La topologia modificata è mostrata nella Figura 5.23 (b). Quando scopre che G non c'è più, D controlla la propria tabella di routing e vede che G era utilizzato per i percorsi diretti verso E , G e I . L'unione dei vicini attivi associati a queste destinazioni è l'insieme $\{A, B\}$. In altre parole, A e B dipendono da G per alcuni dei loro percorsi, perciò devono essere informati che questo percorso non sono più validi. D tranneva loro la notizia inviando un pacchetto che costringe i due nodi ad aggiornare le rispettive tabelle di routing. D elimina anche le voci relative a E , G e I dalla sua tabella di routing. Dalla descrizione forse non è ovvio, ma una differenza critica tra AODV e l'algoritmo Bellman-Ford è che i nodi non trasmettono periodicamente in modalità broadcast le loro tabelle di routing. Questa differenza risparmia banda e riduce il consumo delle batterie. AODV è anche in grado di eseguire il routing broadcast e multicast. Per ulteriori dettagli, consultare (Perkins and Royer, 2001). Il routing ad hoc è un'area di ricerca in fermento, e molti testi sono stati pubblicati su questo argomento. Alcuni dei più interessanti sono Chen et al., 2002; Hu and Johnson, 2001; Li et al., 2001; Raju and Garcia-Luna-Aceves, 2001; Ramanathan and Redi, 2002; Royer and Toh, 1999; Spohn and Garcia-Luna-Aceves, 2001; Tseng et al., 2001; e Zadeh et al., 2002).

5.11 Ricerca del nodo nelle reti peer-to-peer

Le reti peer-to-peer sono un fenomeno relativamente nuovo in cui moltissime persone, di solito con collegamenti cablati permanenti a Internet, sono in contatto per condividere risorse. La prima applicazione estesa della tecnologia peer-to-peer ha provocato un reato massiccio: 50 milioni di utenti Napster hanno scambiato canzoni protette dal diritto d'autore senza il permesso dei proprietari del copyright, fino a quando Napster non è stato dissolto per ordine del tribunale dopo molte controversie. Ciò nonostante, la tecnologia peer-to-peer ha molti utilizzi interessanti e legali; ha anche un problema simile a quello del routing, sebbene diverso da quelli descritti precedentemente, e vale la pena esaminarli più da vicino.

Che rende interessanti i sistemi peer-to-peer è il fatto di essere totalmente distribuiti. I nodi sono simmetrici e non c'è alcun controllo centrale né alcuna gerarchia. In un sistema peer-to-peer, ogni utente ha alcune informazioni che possono interessare gli altri. Queste informazioni possono essere costituite da programmi gratuiti, da musica di pubblico dominio, da fotografie e così via. Se sono moltissimi, gli utenti non si ricorderanno e non sapranno dove trovare ciò che stanno cercando. Si potrebbe utilizzare un grande archivio centrale, ma questa soluzione non è fattibile per diversi motivi (per esempio, nessuno potrebbe voler gestire o conservare tale archivio). Perciò, il problema si

riduce a come un utente può trovare un nodo che contiene ciò che sta cercando in assenza di un database centralizzato o di un indice centralizzato.

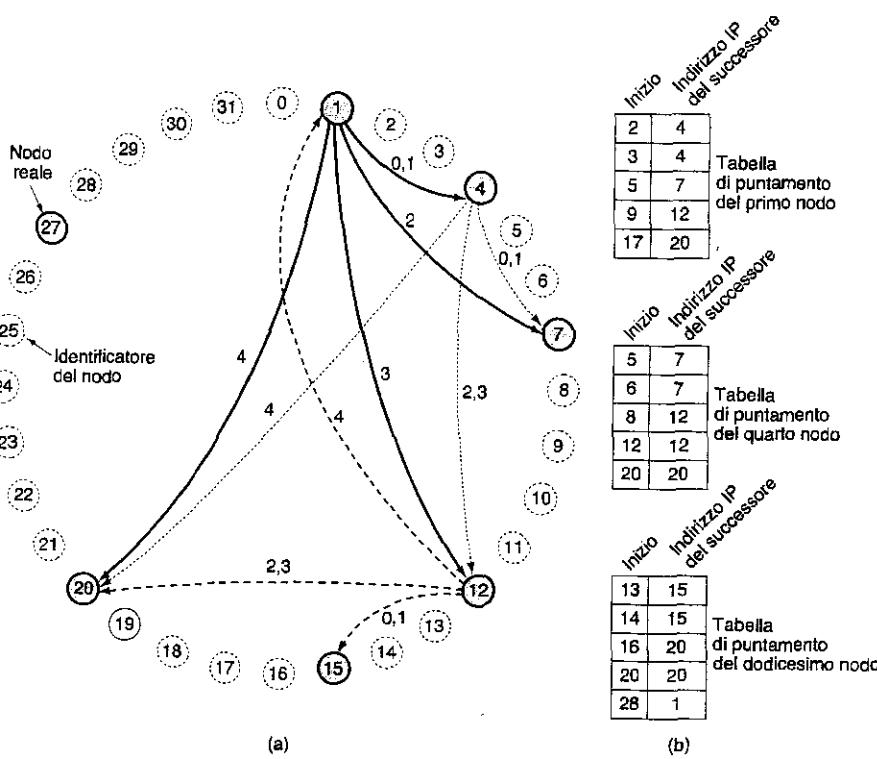
Si supponga che ogni utente abbia uno o più oggetti digitali (canzoni, immagini, programmi, file e così via), che altri utenti potrebbero voler leggere. Ogni elemento è identificato da un nome rappresentato da una stringa di caratteri ASCII. Un utente potenziale, che conosce solo la stringa ASCII, desidera scoprire se ci sono persone che dispongono di copie dell'oggetto e, in caso affermativo, quali sono gli indirizzi IP di tali persone.

Come esempio, si consideri un database genealogico distribuito. Ogni genealogista rende disponibili in linea alcuni record relativi ai suoi antenati e parenti, possibilmente con foto, audio o anche video della persona. Diverse persone possono avere lo stesso nonno, perciò un antenato può avere record sparsi in più nodi. Il nome del record è il nome della persona espresso in qualche forma tradizionale. A un certo punto, un genealogista scopre in un archivio le volontà del suo bisnonno; questi aveva deciso di lasciare in eredità a suo nipote un orologio d'oro da taschino. Il genealogista ora conosce il nome del nipote e desidera scoprire se qualche altro genealogista ha un record su di lui. In che modo, senza disporre di un archivio centrale, è possibile scoprire se qualcun altro ha qualche record?

Per risolvere questo problema sono stati proposti diversi algoritmi; qui si esamina l'algoritmo Chord (Dabek et al., 2001a; e Stoica et al., 2001) con una spiegazione semplificata del suo funzionamento. Il sistema Chord è composta da n utenti partecipanti, ciascuno dei quali può aver archiviato qualche record ed è disposto a conservare porzioni dell'indice per divulgarle agli altri utenti. Ogni nodo utente ha un indirizzo IP, che può essere fuso con un numero di m bit mediante la funzione di hash di nome $hash$. Chord utilizza SHA-1, una funzione hash utilizzata in crittografia che verrà spiegata nel Capitolo 8. Per adesso è sufficiente considerarla una funzione che riceve come parametro una stringa di byte di lunghezza variabile e restituisce un numero casuale a 160 bit. In questo modo è possibile convertire un indirizzo IP in un numero di 160 bit chiamato **identificatore del nodo**. Concettualmente, tutti i 2^{160} identificatori di nodi sono organizzati in ordine crescente in un grande cerchio. Alcuni di loro corrispondono ai nodi partecipanti, ma la maggior parte invece no. La Figura 5.24(a) mostra il cerchio di identificatori di nodi per $m = 5$ (per il momento si ignorino gli archi disegnati dentro il cerchio). In questo esempio, i nodi con gli identificatori 1, 4, 7, 12, 15, 20 e 27 corrispondono a nodi reali e per questo sono stati colorati; gli altri non esistono.

Sia $successor(k)$ la funzione che identifica il primo nodo reale che segue k procedendo in senso orario. Per esempio, $successor(6) = 7$, $successor(8) = 12$ e $successor(22) = 27$.

Anche i nomi dei record (nomi di canzoni, nomi di antenati e così via) sono modificati mediante la funzione $hash$ (ossia SHA-1) per generare un numero di 160 bit chiamato **chiave**. Quindi, per convertire $nome$ (il nome ASCII del record) nella sua chiave si utilizza $key = hash(nome)$. Questo calcolo è solo una chiamata di procedura locale indirizzata a $hash$. Se una persona che conserva record genealogici relativi a $nome$ vuole mettere i suoi dati a disposizione di tutti, deve prima costruire un tuple composto da $(nome, mio-indirizzo-IP)$ e poi chiedere a $successor(hash(nome))$ di archiviare il tuple. Se esistono più record (su nodi diversi) associati a questo nome, i loro tuple verranno archiviati sullo stesso nodo. In questo modo, l'indice è distribuito in modo casuale attraverso i nodi.



per migliorare la tolleranza ai guasti, p diverse funzioni di hash potrebbero essere utilizzate per conservare ogni record in p nodi, ma non prenderemo in esame questo caso. Un utente successivamente decide di cercare *nome*, utilizza la funzione di hash per ottenere la *chiave* e poi utilizza *successor(chiave)* per trovare l'indirizzo IP del nodo che conserva i tuple del suo indice. La prima operazione è semplice; la seconda no. Per rendere possibile la ricerca dell'indirizzo IP del nodo corrispondente a una particolare chiave, ogni nodo deve mantenere alcune strutture di dati amministrativi. Una di queste è l'indirizzo IP del suo nodo successore sul cerchio degli identificatori dei nodi. Per esempio, nella Figura 24 il successore del nodo 4 è 7 e il successore del nodo 7 è 12.

La ricerca ora può procedere come segue. Il nodo richiedente invia un pacchetto al suo successore; il pacchetto, che contiene il suo indirizzo IP e la chiave cercata, si propaga lungo il cerchio fino a quando individua il successore dell'identificatore del nodo cercato. Quel-

nodo verifica se dispone di informazioni che corrispondono alla chiave di ricerca e, in caso positivo, restituisce i dati direttamente al nodo richiedente del quale ha l'indirizzo IP. Come prima ottimizzazione, ogni nodo potrebbe conservare l'indirizzo IP del proprio predecessore e del proprio successore, in modo che le interrogazioni possano essere trasmesse sia in senso orario sia in senso antiorario, in base al percorso considerato più corto. Per esempio, il nodo 7 nella Figura 5.24 potrebbe procedere in senso orario per trovare l'identificatore del nodo 10, ma in senso antiorario per raggiungere l'identificatore del nodo 3. Anche con due possibili direzioni, la ricerca lineare di tutti i nodi è molto inefficiente in un grande sistema peer-to-peer poiché il numero medio di nodi richiesti per ogni ricerca è $n/2$. Per velocizzare di molto l'operazione, ogni nodo mantiene anche ciò che Chord chiama **finger table** (tabella di puntamento). La tabella di puntamento ha m voci, indicizzate da 0 a $m - 1$, ognuna delle quali punta a un nodo effettivo diverso. Ogni voce contiene due campi: *start* e l'indirizzo IP di *successor(start)*, come mostrato nella Figura 5.24(b). I valori dei campi per la voce i nel nodo k sono:

$$\text{start} = k + 2^i \text{ (modulo } 2^m\text{)}$$

Indirizzo IP di *successor(start [i])*

Perciò ogni nodo conserva gli indirizzi IP di un numero relativamente piccolo di nodi, e la maggior parte di questi è abbastanza vicina in termini di identificatore di nodo. Utilizzando la tabella di puntamento, la ricerca di *chiave* al nodo k procede come segue. Se *chiave* è compresa tra *start* e *successor(k)*, allora il nodo contenente le informazioni relative a *chiave* è *successor(k)* e la ricerca si conclude. Altrimenti, si esamina la tabella di puntamento per scoprire qual è la voce il cui campo *start* è il più vicino predecessore di *chiave*. Quindi viene trasmessa una richiesta direttamente all'indirizzo IP relativo a quella voce della tabella di puntamento, per sollecitare la continuazione della ricerca. Poiché è più vicino a *chiave* ma ancora al di sotto di esso, ci sono buone possibilità che si possa ottenere una risposta con un numero basso di interrogazioni aggiuntive. In effetti, poiché ogni ricerca dimezza la distanza rimanente dall'obiettivo, si può dimostrare che il numero medio di ricerche è $\log_2 n$.

Come primo esempio, si consideri la ricerca di *chiave* = 3 al nodo 1. Poiché il nodo 1 sa che 3 si trova tra lui e il suo successore, 4, il nodo desiderato è 4 e la ricerca termina restituendo l'indirizzo IP del nodo 4.

Come secondo esempio, si consideri la ricerca di *chiave* = 14 al nodo 1. Poiché il nodo 14 non si trova tra 1 e 4, viene consultata la tabella di puntamento. Il predecessore più vicino a 14 è 9, perciò la richiesta è inoltrata all'indirizzo IP della voce relativa a 9, ossia, all'indirizzo IP del nodo 12. Il nodo 12 si accorge che 14 si trova tra lui e il suo successore (15), perciò restituisce l'indirizzo IP del nodo 15.

Come terzo esempio, si consideri la ricerca di *chiave* = 16 al nodo 1. Anche in questo caso l'interrogazione è inviata al nodo 12 ma questa volta 12 non sa rispondere da solo. Cerca il nodo più vicino che precede 16 e trova 14, questo conduce all'indirizzo IP del nodo 15. Una richiesta viene trasmessa al nodo 15, che si accorge che 16 si trova tra lui e il suo suc-

essore (20), perciò restituisce al chiamante l'indirizzo IP di 20 che torna indietro fino al nodo 1.

Poiché i nodi si uniscono e si staccano dal sistema continuamente, Chord ha bisogno di un nodo per gestire queste operazioni. Si supponga che quando inizia a funzionare, il sistema sia così piccolo da consentire a tutti i nodi di scambiarsi le informazioni direttamente per costruire il primo cerchio e le tabelle di puntamento. Dopo l'avvio iniziale è necessaria una procedura automatica, che svolge le funzioni descritte di seguito. Se desidera unirsi al sistema, un nuovo nodo r deve contattare un nodo esistente chiedendogli di cercare l'indirizzo IP di $\text{successor}(r)$. Poi il nuovo nodo chiede a $\text{successor}(r)$ di indicare il suo predecessore. Infine il nuovo nodo chiede al suo predecessore e al suo successore di inserire r tra i due nodi del cerchio appena individuati. Per esempio, se volesse unirsi al sistema, il nodo 24 della Figura 5.24 dovrebbe chiedere a un nodo esistente di cercare $\text{successor}(24)$ che è 27. Poi dovrebbe chiedere a 27 di indicare il suo predecessore (20). Dopo aver comunicato a entrambi la sua esistenza, 20 utilizzerà 24 come suo successore e 27 utilizzerà 24 come suo predecessore. Infine, il nodo 27 consegna le chiavi dell'intervallo 1-24, che ora appartengono a 24. A questo punto 24 è completamente inserito.

Ora molte tabelle di puntamento contengono errori. Per correggerli, ogni nodo esegue un processo in background che periodicamente ricalcola ogni puntamento chiamando successor . Quando una di queste interrogazioni individua un nuovo nodo, viene aggiornata la base del puntamento corrispondente.

Quando abbandona il sistema in modo regolare, il nodo consegna le proprie chiavi al suo predecessore e informa il suo predecessore che sta uscendo dal sistema, in modo che il predecessore possa collegarsi al successore del nodo uscente. Quando un nodo si blocca improvvisamente, sorge un problema perché il suo predecessore non ha più un successore valido. Per attenuarlo, ogni nodo tiene traccia non solo del suo diretto successore, ma anche dei suoi precedenti 5 successori diretti, in modo da poter rientrare nel cerchio entro 5 tentativi consecutivi falliti. Chord è stato utilizzato per costruire un file system distribuito (Dabek et al., 2001b) e altre applicazioni; la ricerca inoltre va avanti. Un diverso sistema peer-to-peer, Pastry, e le sue applicazioni sono descritte in (Rowstron and Druschel, 2001a; e Rowstron and Druschel, 2001b). Un terzo sistema, peer-to-peer, e2net, è esaminato in (Clarke et al., 2002). Un quarto sistema di questo tipo è descritto (Ratnasamy et al., 2001).

3 Algoritmi per il controllo della congestione

Ioando troppi pacchetti sono presenti in una porzione della sottorete, le prestazioni degradano. Questa situazione è chiamata **congestione** e la Figura 5.25 descrive i sintomi. Quando il numero di pacchetti immesso dagli host nella sottorete è minore della capacità di trasporto della sottorete, tutti i pacchetti vengono inoltrati (tranne quei pochi afflitti dagli errori di trasmissione) e il loro numero è proporzionale al numero trasmesso. Se invece il traffico aumenta in modo eccessivo, i router non riescono a far fronte alla situazione e cominciano a perdere pacchetti; ciò tende a far peggiorare le cose. Con un traffico elevatissimo, le prestazioni crollano completamente e quasi nessun pacchetto viene inoltrato.

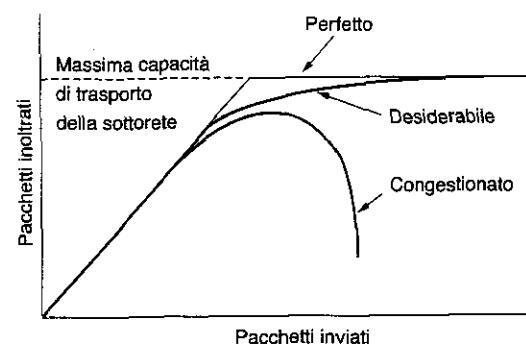


Figura 5.25. Quando viene immesso troppo traffico, si crea una congestione e le prestazioni degradano improvvisamente.

La congestione può essere causata da più fattori. Se improvvisamente diversi flussi di pacchetti cominciano ad arrivare attraverso tre o quattro linee di input e tutti i pacchetti necessitano della stessa linea di output, inizia a formarsi una coda. Se la memoria non è sufficiente, non sarà possibile conservare tutti i dati, perciò alcuni pacchetti andranno persi. Aggiungere nuova memoria può aiutare fino a un certo punto, ma Nagle (1987) ha scoperto che se i router hanno una memoria infinita la congestione peggiora invece di migliorare, poiché quando raggiungeranno il fronte della coda, i pacchetti saranno già scaduti (ripetutamente) e quindi saranno già stati trasmessi dei duplicati. Tutti questi pacchetti saranno doverosamente inoltrati al router successivo, aumentando il carico lungo il percorso diretto alla destinazione.

Anche i processori lenti possono causare congestioni. Se le CPU dei router eseguono lentamente le operazioni richieste (accodamento dei buffer, aggiornamento delle tabelle e così via), le code possono formarsi anche quando la capacità della linea è maggiore del necessario; allo stesso modo le congestioni possono formarsi per colpa delle linee a banda stretta. Aggiornare le linee senza cambiare i processori (e viceversa) può essere d'aiuto, ma spesso non fa altro che spostare il collo di bottiglia; aggiornando solo una parte del sistema di solito si ottiene il risultato di spostare il collo di bottiglia da qualche altra parte. Il più delle volte il vero problema è un cattivo adattamento tra le parti del sistema, che perdura fino a quando non si raggiunge un equilibrio tra tutti i componenti.

Vale la pena di richiamare l'attenzione sulla differenza tra controllo della congestione e controllo di flusso, in quanto la relazione è sottile. Controllare la congestione significa garantire che la sottorete sia in grado di trasportare il traffico immesso: è un problema globale, che coinvolge il comportamento di tutti gli host, tutti i router, l'elaborazione di tipo store-and-forward eseguita dai router, e tutti gli altri fattori che tendono a diminuire la capacità di carico della sottorete.

Il controllo del flusso, invece, ha a che fare con il traffico punto-punto tra un dato trasmettitore e un dato ricevitore; il suo compito è evitare che una sorgente veloce trasmetta continuamente una quantità di dati maggiore di quella che il ricevitore è in grado di assorbire. Il controllo di flusso coinvolge spesso una retroazione inviata dal ricevitore al trasmettitore, che indica al trasmettitore come vanno le cose all'altro capo della linea.

Per comprendere la differenza tra i due concetti, si consideri una rete a fibra ottica con una capacità di 1.000 gigabit/sec attraverso la quale un supercomputer tenta di trasferire un file a un personal computer a una velocità di 1 Gbps. Sebbene non ci sia congestione (la rete in sé non ha problemi), è necessario utilizzare il controllo di flusso per costringere il supercomputer a interrompere spesso la trasmissione in modo da dare al PC la possibilità di respirare.

All'altro estremo, si consideri una rete di tipo store-and-forward composta da linee a 1 Mbps e da 1.000 grandi computer, metà dei quali tenta di trasferire file a 100 kbps all'altra metà. In questo caso il problema non è causato da trasmettitori veloci che opprimono ricevitori lenti, ma dal traffico totale immesso che supera quello che la rete è in grado di gestire.

Il motivo per cui spesso il controllo della congestione viene confuso con il controllo di flusso è che alcuni algoritmi di controllo della congestione inviano alle sorgenti messaggi che dicono di rallentare la trasmissione, quando la rete ha problemi. Per questo motivo un host può ricevere un messaggio "rallenta" sia perché il ricevitore non è in grado di ricevere il carico a quella velocità, sia perché la rete non è in grado di gestirlo. Questo punto verrà discusso più avanti.

Lo studio del controllo della congestione inizierà descrivendo un modello generale che consente di gestire il problema, per poi esaminare gli approcci per prevenire la congestione, e infine descrivere algoritmi dinamici per affrontare il problema quando si presenta.

5.3.1 Principi generali del controllo della congestione

Molti problemi nei sistemi complessi, come le reti di computer, si possono considerare dal punto di vista della teoria di controllo. Questo approccio porta a dividere tutte le soluzioni in due gruppi: cicli aperti e cicli chiusi. Le soluzioni a ciclo aperto tentano di risolvere il problema mediante un buon progetto che renda in primo luogo improbabile la manifestazione del problema stesso. Una volta che il sistema è stato attivato, non viene eseguita alcuna correzione in corsa.

Gli strumenti che eseguono il controllo a ciclo aperto possono decidere quando deve essere accettato nuovo traffico, quando e quali pacchetti devono essere scartati, e prendere decisioni sulla pianificazione in vari punti della rete. Tutti questi approcci hanno in comune il fatto che le decisioni sono prese senza tener conto dello stato corrente della rete. Al contrario, le soluzioni a ciclo chiuso si basano sul concetto della retroazione. Questo approccio, quando è applicato al controllo della congestione, è composto da tre parti:

1. controllo del sistema per rilevare quando e dove si presenta la congestione
2. passaggio di queste informazioni ai punti dove si possono eseguire le azioni di correzione
3. regolazione del funzionamento del sistema per correggere il problema.

Per tenere sotto controllo la sottorete si possono usare diverse metriche. Le principali sono percentuale di pacchetti scartati a causa di un esaurimento di buffer, la lunghezza media della coda, il numero di pacchetti scaduti e ritrasmessi, il ritardo medio dei pacchetti e la

deviazione standard del ritardo di pacchetto. In tutti i casi, valori elevati indicano una congestione crescente.

Nella seconda fase del ciclo di retroazione le informazioni sulla congestione sono trasferite dal punto dove sono state rilevate a quello dove è possibile fare qualcosa per risolvere il problema. La cosa più ovvia è che il router che ha rilevato la congestione invii un pacchetto alla sorgente o alle sorgenti del traffico, annunciando il problema. Naturalmente questi pacchetti aggiuntivi aumentano il carico proprio nel momento in cui sarebbe più opportuno diminuirlo, vale a dire quando la sottorete è congestionata.

Esistono però altre possibilità. Per esempio, un bit o un campo di ogni pacchetto può essere riservato ai router e utilizzato quando la congestione supera un determinato valore di guardia. Ogni volta che rileva lo stato di congestione, il router riempie il campo di tutti i pacchetti in uscita, in modo da avvisare i suoi vicini.

Un altro approccio prevede che host e router spediscano periodicamente pacchetti di segnalazione per chiedere in modo esplicito informazioni sulla congestione; queste informazioni si possono usare per instradare il traffico in modo da aggirare le aree che hanno problemi. Alcune stazioni radio fanno volare sulla città elicotteri che segnalano in tempo reale le strade più congestionate; gli ascoltatori sintonizzati su quelle stazioni possono così dirottare i loro "pacchetti" (in questo caso, le auto) in modo da evitare i punti caldi.

In tutti gli schemi di retroazione si spera che la conoscenza della congestione costringa gli host a prendere provvedimenti per ridurla. Per far funzionare correttamente lo schema è necessario regolare con attenzione la scala temporale: se ogni volta che due pacchetti arrivano in fila il router grida STOP, e ogni volta che è inattivo per 20 msec grida GO, il sistema oscillatorà in modo sfrenato senza mai convergere. D'altra parte, se aspetta 30 minuti prima di dire qualunque cosa, il meccanismo di controllo della congestione reagirà troppo lentamente e non sarà di alcuna utilità pratica. Per funzionare bene è necessario fare qualche tipo di media; ma non è semplice definire i tempi giusti.

Si conoscono molti algoritmi di controllo della congestione, e per organizzarli in modo sensato, Yang and Reddy (1995) hanno sviluppato una tassonomia. I due studiosi hanno diviso prima di tutto gli algoritmi in soluzioni a ciclo aperto e soluzioni a ciclo chiuso, come è stato spiegato precedentemente; poi hanno diviso gli algoritmi a ciclo aperto in soluzioni che agiscono sulla sorgente e soluzioni che lavorano sulla destinazione. Anche gli algoritmi a ciclo chiuso sono stati divisi in due categorie: soluzioni a retroazione esplicita e soluzioni a retroazione implicita. Negli algoritmi a retroazione esplicita i pacchetti per avvisare la sorgente vengono spediti indietro dal punto della congestione. Negli algoritmi a retroazione implicita la sorgente deduce l'esistenza della congestione effettuando osservazioni locali, basate per esempio sul tempo di arrivo dei pacchetti di acknowledgement.

La presenza di congestione indica che il carico è (temporaneamente) più grande di quello che può essere gestito dalle risorse (in una parte del sistema). Vengono alla mente due soluzioni: aumentare le risorse o diminuire il carico. Per esempio, la sottorete può iniziare a utilizzare linee telefoniche dial-up per aumentare temporaneamente la banda tra certi punti; nei sistemi satellitari, un aumento della potenza della trasmissione spesso fa aumentare la banda. Anche dividere il traffico tra più percorsi invece di utilizzare sempre il per-

corso migliore può aumentare efficacemente la banda. Infine, router di riserva utilizzati normalmente solo come dispositivi di emergenza (per rendere il sistema resistente ai guasti) possono essere messi in linea per aumentare la capacità in caso di congestioni gravi. Purtroppo qualche volta non è possibile aumentare la capacità, oppure essa è già stata portata al limite. In questo caso la congestione può essere vinta in un solo modo: diminuendo il carico. Il carico può essere ridotto in diversi modi, per esempio negando servizi ad alcuni utenti, degradando il servizio ad alcuni o a tutti gli utenti, e facendo in modo che gli utenti programmino le richieste in un modo più prevedibile.

Alcuni di questi metodi, descritti più avanti, possono essere applicati meglio ai circuiti virtuali. Per le sottoreti che usano internamente i circuiti virtuali, questi metodi possono essere implementati sullo strato network. Per le sottoreti a datagramma a volte si possono implementare egualmente sulle connessioni dello strato trasporto. Questo capitolo si concentra sul loro uso nello strato network, mentre il successivo spiega che cosa si può fare sullo strato di trasporto per gestire la congestione.

5.3.2 Criteri per prevenire la congestione

Lo studio sui metodi di controllo della congestione inizia dai sistemi a ciclo aperto, che sono stati progettati in primo luogo per ridurre la possibilità di una congestione, non per risolvere il problema quando si presenta. Tentano di raggiungere il loro obiettivo usando opportuni criteri su più livelli. La Figura 5.26 mostra diversi criteri che possono influenzare le congestioni sugli strati data link, network e trasporto. (Jain, 1990).

Inizieremo dallo strato data link per procedere verso l'alto. Il criterio di ritrasmissione riguarda la velocità con la quale un trasmettitore gestisce la scadenza dei pacchetti e che cosa ritrasmette dopo un timeout. Un trasmettitore nervoso che ritrasmette troppo velocemente tutti i

Strato	Criteri
Trasporto	<ul style="list-style-type: none"> • Criterio di ritrasmissione • Criterio di caching fuori sequenza • Criterio di acknowledgement • Criterio di controllo di flusso • Determinazione dei timeout
Network	<ul style="list-style-type: none"> • Scelta fra circuiti virtuali e datagrammi nella sottorete • Criterio di accodamento e di servizio dei pacchetti • Criterio di eliminazione dei pacchetti • Algoritmo di routing • Gestione della vita utile del pacchetto
Data link	<ul style="list-style-type: none"> • Criterio di ritrasmissione • Criterio di caching fuori sequenza • Criterio di acknowledgement • Criterio di controllo di flusso

Figura 5.26. Criteri che influenzano la congestione.

pacchetti scaduti usando il meccanismo "torna indietro n volte" immetterà nel sistema un carico più pesante di quello introdotto da un trasmettitore che utilizza una tranquilla ripetizione selettiva. Strettamente collegato a questo è il criterio di utilizzo dei buffer. Se i ricevitori scartano sistematicamente tutti i pacchetti fuori sequenza, questi pacchetti verranno successivamente ritrasmessi, creando altro carico. In rapporto al controllo della congestione, la ripetizione selettiva è chiaramente più utile del tornare indietro di n volte.

Anche il criterio di generazione degli acknowledgement influenza le congestioni. Se ogni pacchetto riceve immediatamente acknowledgement, i pacchetti di acknowledgement generano traffico aggiuntivo, ma se al contrario gli acknowledgement vengono trasportati dal traffico inverso si possono avere ritrasmissioni e scadenze aggiuntive. Uno schema di controllo di flusso rigido (per esempio, una finestra piccola) riduce la velocità di trasmissione dei dati e di conseguenza aiuta a combattere la congestione.

Sullo strato network, la scelta tra l'utilizzo dei circuiti virtuali e dei datagrammi influenza la congestione perché molti algoritmi di controllo della congestione funzionano solo con le sottoreti a circuito virtuale. Il criterio di servizio e accodamento dei pacchetti riguarda il numero di code presenti sui router: una coda per ogni linea di input, una coda per ogni linea di output o entrambe. Inoltre, si riferisce all'ordine in cui i pacchetti sono elaborati (ossia round robin o in base alla priorità). Il criterio di scarto è la regola che stabilisce quale pacchetto scartare quando non c'è più spazio. Un buon criterio può aiutare ad alleviare la congestione, mentre uno cattivo può peggiorarla.

Un buon algoritmo di routing può aiutare a evitare la congestione spargagliando il traffico attraverso tutte le linee, mentre un cattivo algoritmo può immettere troppo traffico sulle linee già congestionate. Infine, la gestione del tempo di vita dei pacchetti definisce quanto devono vivere i pacchetti prima di essere scartati. Se la vita è troppo lunga, i pacchetti persi possono intasare tutto per molto tempo, ma se è troppo breve i pacchetti possono scadere prima di raggiungere le loro destinazioni, causando ritrasmissioni.

Lo strato di trasporto ha gli stessi problemi dello strato data link, inoltre è più difficile determinare l'intervallo di scadenza perché il tempo di transito attraverso la rete è meno prevedibile del tempo di transito attraverso un cavo che collega due router. Se l'intervallo di scadenza è troppo breve verranno trasmessi inutilmente pacchetti aggiuntivi; se è troppo lungo, la congestione verrà ridotta ma il tempo di risposta ne soffrirà ogni volta che un pacchetto si perde.

5.3.3 Controllo della congestione nelle sottoreti a circuito virtuale

I metodi per il controllo della congestione descritti precedentemente sono sostanzialmente basati sui cicli aperti: tentano di prevenire la congestione prima che accada, non cercano di mitigare i suoi effetti. Questo paragrafo studia alcuni approcci per controllare in modo dinamico la congestione nelle sottoreti a circuito virtuale, mentre nelle prossime due si esaminano le tecniche utilizzabili in ogni sottorete.

Una tecnica largamente utilizzata per evitare il peggioramento di una congestione già iniziata si chiama **controllo di ammissione**. L'idea è semplice: una volta che la congestione è stata segnalata, nessun circuito virtuale viene più impostato fino a quando il problema non scompare. I tentativi di impostare nuove connessioni sullo strato di trasporto falliscono

no, infatti lasciare che nuove persone si colleghino peggiorerebbe la situazione. Benché sia rozzo, questo approccio è semplice e facile da implementare. I commutatori del sistema telefonico, quando sono sovraccaricati, praticano il controllo di ammissione non dando il segnale di libero.

Un approccio alternativo consente la creazione di nuovi circuiti virtuali, ma li instrada attentamente per aggirare le aree congestionate. Per esempio, si consideri la sottorete mostrata nella Figura 5.27(a) in cui due router sono congestionati.

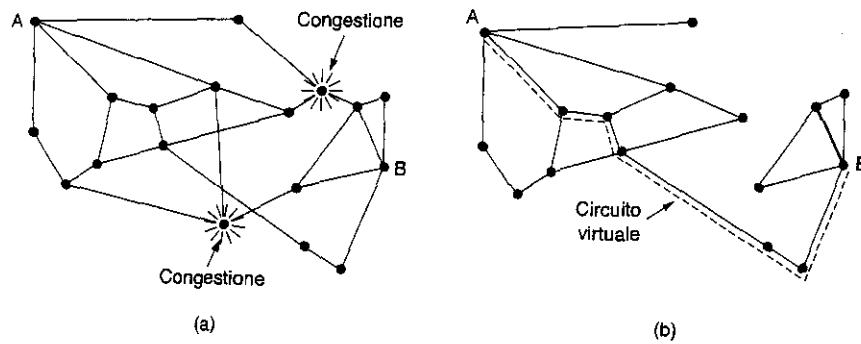


Figura 5.27. (a) Una sottorete congestionata. (b) Una sottorete ridisegnata, che elimina la congestione. È mostrato anche un circuito virtuale che collega A a B.

Si supponga che un host collegato al router A voglia impostare una connessione con un host collegato al router B. Normalmente, questa connessione passerebbe attraverso uno dei due router congestionati. Per evitare questa situazione, si può ridisegnare la sottorete come mostrato nella Figura 5.27(b), omettendo i router congestionati e tutte le loro linee. Le linee tratteggiate rappresentano un possibile percorso per il circuito virtuale che evita i router congestionati.

Un'altra strategia che lavora sui circuiti virtuali consiste nel negoziare un accordo tra host e la sottorete durante l'impostazione del circuito virtuale. Questo accordo normalmente specifica il volume e la forma del traffico, la qualità del servizio richiesta e altri parametri. Per tener fede all'accordo, di solito la sottorete riserva le risorse lungo il percorso durante l'impostazione del circuito. Queste risorse possono comprendere dello spazio nelle tabelle e nei buffer dei router e della banda sulle linee. In questo modo è poco probabile che la congestione si presenti sui nuovi circuiti virtuali, perché tutte le risorse necessarie sono garantite e disponibili.

Questo tipo di prenotazione può essere eseguita sempre, come procedura operativa standard, oppure solo quando la sottorete è congestionata. L'utilizzo costante ha uno svantaggio: tende a sprecare risorse. Se sei circuiti virtuali che potrebbero utilizzare 1 Mbps passano tutti attraverso la stessa linea a 6 Mbps, la linea deve essere considerata piena anche se raramente accade che tutti e sei i circuiti virtuali trasmettano nello stesso momento. Di conseguenza, il prezzo del controllo della congestione è una banda non del tutto utilizzata (sprecata).

5.3.4 Controllo della congestione nelle sottoreti a datagrammi

È giunto il momento di esaminare alcuni approcci utilizzabili nelle sottoreti a datagrammi (e anche in quelle a circuiti virtuali). Ogni router può facilmente tenere sotto controllo l'utilizzo delle sue linee di output e di altre risorse. Per esempio, il dispositivo potrebbe associare a ogni linea una variabile reale u il cui valore, compreso tra 0,0 e 1,0, riflette l'utilizzo recente di quella linea. Per mantenere una buona stima di u , periodicamente si potrebbe prendere un campione f (che vale zero oppure uno) dell'utilizzo istantaneo della linea, aggiornando u secondo questa formula:

$$u_{\text{nuovo}} = au_{\text{vecchio}} + (1 - a)f$$

La costante a determina la velocità con cui il router dimentica la cronologia recente. Ogni volta che u si sposta sopra la soglia di guardia, la linea di output entra in stato di "allarme". Il router a questo punto controlla ogni pacchetto in arrivo per vedere se la linea di output associata a quei dati è in stato di allarme e, in caso affermativo, esegue un'azione correttiva. L'azione intrapresa può essere una di quelle descritte di seguito.

Il bit di allarme

La vecchia architettura DECNET segnalava lo stato di allarme impostando uno speciale bit nell'intestazione del pacchetto, proprio come fa frame relay. Quando il pacchetto arriva alla sua destinazione, l'entità di trasporto copia il bit nel successivo pacchetto di acknowledgement trasmesso alla sorgente, che quindi riduce il traffico.

Finché rimane in stato di allarme, il router continua a impostare il bit di allarme e la sorgente continua a ricevere acknowledgement contenenti questo valore. La sorgente tiene sotto controllo la frazione di acknowledgement contenenti il bit impostato e regola la propria velocità di trasmissione di conseguenza. Finché continuano ad arrivare i bit di allarme la sorgente seguita a diminuire la sua velocità di trasmissione, e non appena la loro frequenza si riduce al minimo, la sorgente aumenta la propria velocità di trasmissione. Si noti che poiché ogni router lungo il percorso potrebbe impostare i bit di allarme, il traffico aumenta solo quando non è rimasto alcun router in difficoltà.

Choke packet

Il precedente algoritmo di controllo della congestione è abbastanza astuto: utilizza un mezzo indiretto per dire alla sorgente di rallentare. Perché non farglielo sapere direttamente? In questo approccio il router invia all'host sorgente un **choke packet** dandogli la destinazione trovata nel pacchetto. Il pacchetto originale viene etichettato (un bit dell'intestazione è impostato a 1) in modo da impedire la generazione di altri choke packet lungo il percorso, e poi è inoltrato nel solito modo.

Quando riceve il choke packet, l'host sorgente deve ridurre dell'X per cento il traffico inviato alla destinazione specificata. Poiché altri pacchetti che puntano alla stessa destinazione probabilmente sono già in viaggio e genereranno nuovi choke packet, per un intervallo di tempo prefissato l'host dovrebbe ignorare i choke packet che si riferiscono a quella destinazione. Trascorso questo periodo, l'host rimane in ascolto di ulteriori choke packet per un altro inter-

allo: se ne arriva uno, allora la linea è ancora congestionata perciò l'host riduce il flusso un altro po' e inizia a ignorare di nuovo i choke packet; se non arriva più alcun choke packet durante il periodo di ascolto, l'host può aumentare nuovamente il flusso. L'effetto retroattivo implicito in questo protocollo può aiutare a impedire la congestione, rallentando il flusso dei dati solo quando si presenta un problema.

I host possono ridurre il traffico regolando i loro parametri, per esempio la dimensione della loro finestra. In genere, il primo choke packet riduce la cadenza dei dati allo 0,50 della precedente velocità di trasmissione; il successivo riduce la velocità di un altro 0,25 così via. Per evitare una veloce ricomparsa della congestione, la velocità è ripristinata ottanto incrementi più piccoli. Sono state proposte diverse varianti di questo algoritmo di controllo della congestione. In una di queste, i router possono mantenere diverse soglie di guardia. In base alla soglia che viene oltrepassata, il choke packet può contenere un avviso, un allarme o un vero e proprio ultimatum.

In un'altra variante, come segnale di attivazione si utilizza la lunghezza della coda o l'utilizzo del buffer al posto dell'utilizzazione della linea. Naturalmente, con queste metriche si può utilizzare la stessa ponderazione esponenziale applicata a n .

Choke packet hop-by-hop

Alte velocità o su lunghe distanze, l'invio dei choke packet agli host sorgente non funziona bene perché la reazione è lenta. Si consideri, per esempio, un host a San Francisco (router A nella Figura 5.28) che trasmette il traffico a un host che si trova a New York (router D nella Figura 5.28) a 155 Mbps. Se l'host di New York comincia a esaurire il buffer, un choke packet che rallenterà la trasmissione impiegherà circa 30 msec per raggiungere San Francisco. La propagazione del choke packet è mostrata nella seconda, nella terza e nella quarta fase della Figura 5.28(a). In quei 30 msec potrebbero essere trasmessi altri 4,6 megabit di dati. Anche se l'host di San Francisco interrompesse immediatamente la trasmissione, i 4,6 megabit nel canale continuerebbero ad affluire e ciò richiederebbe comunque una gestione. Solo nel settimo diagramma nella Figura 5.28(a) il router di New York terebbe un rallentamento del flusso.

In un approccio alternativo, il choke packet ha effetto su tutti i salti (*hop*) attraversati, come illustrato nella sequenza descritta nella Figura 5.28(b). In questo caso, non appena il choke packet raggiunge F, il router F riduce il flusso diretto a D. Quando si agisce in questo modo, non serve dedicare più buffer al flusso poiché la sorgente continua a inviare i dati a piena velocità; comunque, D riceve un sollievo immediato, proprio come accade quando si assume uno dei quei rimedi contro il mal di testa pubblicizzati in televisione. Nella fase successiva, il choke packet raggiunge E e il router riduce il flusso dati diretto verso F. Questa azione aumenta l'utilizzo dei buffer di E, ma dà un sollievo immediato a F. Infine, il choke packet raggiunge A e il flusso rallenta realmente.

Il schema hop-by-hop ha l'effetto di alleviare rapidamente la congestione nel punto dove si trova, al prezzo di un incremento dell'utilizzo dei buffer di trasmissione nella parte del corso precedente. In questo modo la congestione può essere stroncata sul nascere senza

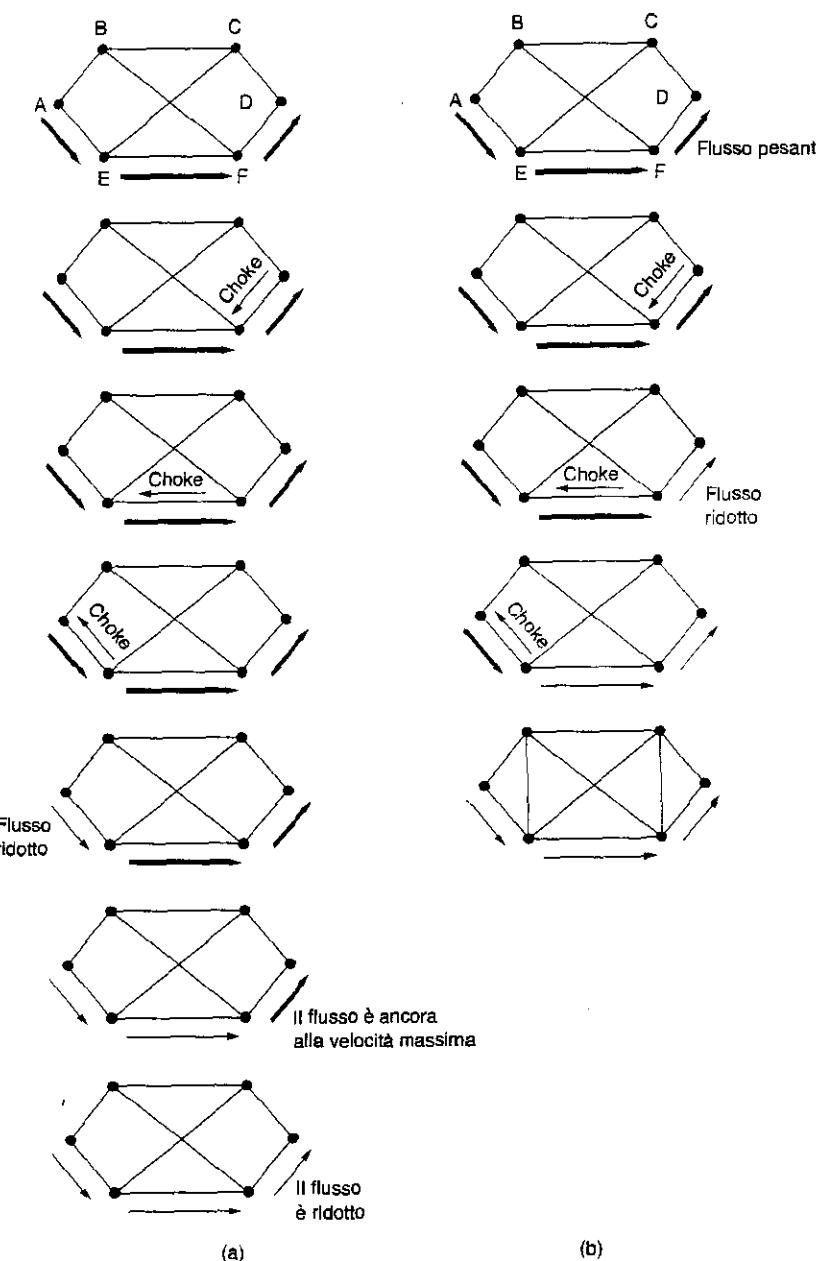


Figura 5.28. (a) Un choke packet che influenza solo la sorgente.
(b) Un choke packet che influenza tutti gli hop attraversati.

perdere alcun pacchetto. L'idea è descritta in dettaglio, insieme ai risultati ottenuti attraverso alcune simulazioni, in (Mishra and Kanakia, 1992).

5.3.5 Load shedding

Quando nessuno dei metodi descritti precedentemente riesce a eliminare la congestione, i router possono tirar fuori "l'artiglieria pesante": gettare via il carico. La tecnica **load shedding** è un modo estroso per indicare l'azione di eliminazione intrapresa dai router inondati da troppi pacchetti. Il termine deriva dal mondo della generazione dell'energia elettrica e si riferisce alla pratica di oscurare intenzionalmente certe aree per evitare il collasso dell'intera rete durante i giorni più caldi d'estate, quando il consumo di elettricità supera enormemente la disponibilità.

Un router sommerso da pacchetti può banalmente scartare qualche pacchetto a caso, ma di solito si può fare decisamente di meglio. La scelta del pacchetto da scartare può dipendere dalle applicazioni in esecuzione. Per il trasferimento dei file, un pacchetto vecchio è più prezioso di un pacchetto nuovo, perché eliminare il pacchetto 6 e tenere i pacchetti 7, 8, 9 e 10 provoca una lacuna che può costringere il ricevitore a invocare la ritrasmissione dei pacchetti compresi tra 6 e 10 (se il ricevitore abitualmente scarta i pacchetti fuori sequenza). In un file lungo 12 pacchetti, l'eliminazione del pacchetto 6 richiede la ritrasmissione dei pacchetti da 7 a 12, mentre l'eliminazione del numero 10 può richiedere solo la ritrasmissione degli ultimi 3 pacchetti (dal 10 al 12). Al contrario, nel caso dei dati multimediali un pacchetto nuovo è più importante di un pacchetto vecchio. Il criterio precedente (il vecchio è migliore del nuovo) è spesso chiamato **wine** (vino) mentre il secondo criterio (il nuovo è migliore del vecchio) è spesso chiamato **milk** (latte).

Un ulteriore passo in avanti richiede la cooperazione dei trasmettitori. Per molte applicazioni, alcuni pacchetti sono più importanti di altri. Per esempio, certi algoritmi usati per comprimere i documenti video trasmettono periodicamente un intero fotogramma e poi inviano i fotogrammi successivi come differenza dall'ultimo fotogramma completo. In questo caso, è preferibile eliminare un pacchetto che fa parte di una differenza piuttosto che scartare un pacchetto che contiene un fotogramma completo. Per vedere un altro esempio, si consideri un documento contenente testo ASCII e immagini: la perdita di una linea di pixel di un'immagine è meno grave della perdita di una riga di testo.

Per implementare un criterio di eliminazione intelligente, le applicazioni devono contrassegnare i loro pacchetti in classi di priorità in modo da indicare la loro importanza. Se lo fanno, i router possono scartare i pacchetti partendo da quelli di classe più bassa. Naturalmente, senza un adeguato incentivo tutti utilizzeranno il contrassegno MOLTO IMPORTANTE — NON CANCELLARE MAI.

L'incentivo potrebbe essere economico, per esempio i pacchetti a bassa priorità potrebbero essere meno costosi di quelli trasmessi ad alta priorità. In alternativa, si potrebbe consentire ai trasmettenti di inviare pacchetti ad alta priorità solo in condizioni di basso carico; all'aumentare del carico, i pacchetti potrebbero essere scartati, in modo da incoraggiare gli utenti a interrompere la loro trasmissione.

In'altra possibilità sta nel consentire agli host di superare i limiti specificati nell'accordo egoziato durante l'impostazione del circuito virtuale (per esempio, utilizzo di una banda

maggiore di quella consentita), a condizione che essi accettino di trasmettere tutto il traffico in eccesso come traffico a bassa priorità. Questa strategia, in effetti, è molto intelligente: rende più efficiente l'utilizzo delle risorse poiché permette agli host di adoperarle quando nessun altro è interessato, senza stabilire un diritto di utilizzo quando la situazione si fa difficile.

RED (*Random Early Detection*)

È risaputo che risulta più semplice gestire la congestione appena viene rilevata piuttosto che cercare di porvi rimedio dopo averle dato il tempo di bloccare tutto. Questa osservazione conduce all'idea di scartare i pacchetti prima che tutto lo spazio del buffer sia completamente esaurito. Un celebre algoritmo usato per mettere in pratica questo schema è chiamato **RED** (*Random Early Detection*) (Floyd and Jacobson, 1993).

In alcuni protocolli di trasporto (incluso TCP), la perdita di pacchetti causa il rallentamento della sorgente. Il ragionamento dietro questa logica è che TCP è stato progettato per le reti cablate e tali reti sono sempre disponibili, perciò la perdita dei pacchetti è causata più che altro dall'esaurimento dei buffer e non dagli errori di trasmissione. Questo fatto può essere sfruttato per aiutare a ridurre la congestione.

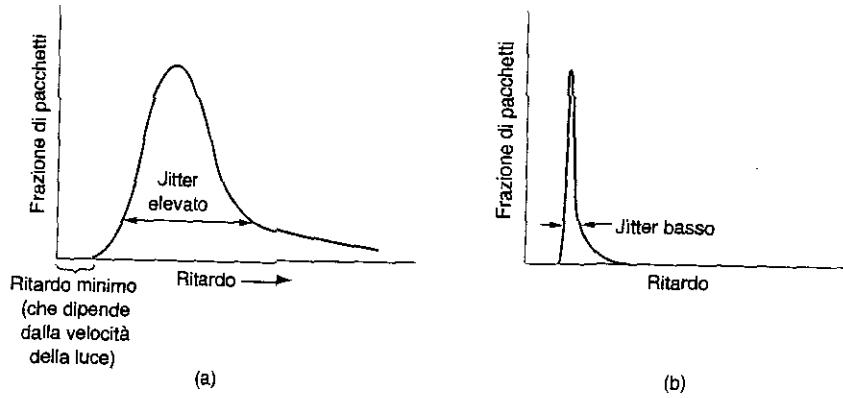
In pratica, facendo in modo che i router scartino i pacchetti prima che la situazione diventi senza speranza (da qui l'uso del termine "early" nel nome) è possibile bloccare il problema sul nascere, prima che sia troppo tardi. Per stabilire quando è il momento giusto per iniziare a scartare i pacchetti, i router mantengono una media mobile delle lunghezze delle code. Quando la lunghezza media della coda su una linea supera una soglia di guardia, la linea è considerata congestionata e viene intrapresa l'azione di correzione. Poiché il router probabilmente non è in grado di dire qual è la sorgente che causa la maggior parte del pasticcio, scegliere un pacchetto a caso dalla coda che ha attivato l'azione è probabilmente il massimo che può fare.

In che modo il router dovrebbe avvisare la sorgente del problema? Il dispositivo potrebbe per esempio trasmettere un choke packet come quello descritto precedentemente. Questo approccio, però, aumenta il carico sulla rete già congestionata. Una strategia diversa consiste banalmente nello scartare il pacchetto selezionato senza rendere nota l'operazione. La sorgente alla fine noterà l'assenza del pacchetto di acknowledgement e prenderà un'iniziativa. Poiché sa che i pacchetti persi sono causati generalmente da congestioni ed eliminazioni, la sorgente risponderà rallentando il flusso invece di ritentare in modo più accanito. Questa forma implicita di feedback funziona solo quando le sorgenti rispondono ai pacchetti perduti rallentando la loro velocità di trasmissione. Nelle reti wireless, dove la maggior parte delle perdite è causata dal rumore presente nel collegamento aereo, questo approccio non può essere utilizzato.

5.3.6 Controllo del jitter

Per applicazioni come la trasmissione di flussi audio e video, non importa se i pacchetti sono trasmessi in 20 o 30 msec, l'essenziale è che il tempo di transito sia costante. La variazione (ossia la deviazione standard) nel tempo di arrivo del pacchetto è chiamata **jitter**. Un jitter

elevato, in cui per esempio alcuni pacchetti arrivano dopo 20 msec e altri dopo 30 msec, genererà un suono o un filmato di qualità variabile. Il jitter è mostrato nella Figura 5.29. Al contrario, un accordo che stabilisce che il 99 per cento dei pacchetti sarà trasmesso con un ritardo compreso tra i 24,5 e i 25,5 msec potrebbe essere accettabile. Naturalmente, l'intervallo scelto deve essere fattibile; deve tener conto del tempo di transito a velocità della luce e del ritardo minimo causato dai router, lasciando magari un piccolo margine per altri inevitabili ritardi.



Il jitter può essere delimitato calcolando il tempo di transito atteso per ogni salto lungo il percorso. Quando un pacchetto raggiunge un router, il router controlla di quanto il pacchetto è in anticipo o in ritardo rispetto alla sua programmazione. Questa informazione è memorizzata nel pacchetto ed è aggiornata a ogni salto. Se è in anticipo, il pacchetto è trattato quanto basta per rientrare nella pianificazione; se il pacchetto è in ritardo, il router tenta di trasmetterlo il prima possibile.

In effetti, l'algoritmo che determina quale dei diversi pacchetti che competono per la linea di trasmissione deve uscire per primo può sempre scegliere il pacchetto più in ritardo rispetto alla sua programmazione. In questo modo, i pacchetti che sono in anticipo vengono rallentati e i pacchetti in ritardo vengono accelerati; in entrambi i casi, il livello di jitter diminuisce.

In alcune applicazioni, per esempio in quelle basate sui video a richiesta, il jitter può essere eliminato memorizzando i dati in un buffer del computer ricevente; i dati delle immagini sono quindi recuperati dal buffer e non dalla rete in tempo reale. In altre applicazioni, specialmente quelle che richiedono interazione in tempo reale tra le persone, come le applicazioni di telefonia e videoconferenza via Internet, il ritardo inerente alla memorizzazione nel buffer non è accettabile.

Il controllo della congestione è un'area di ricerca molto attiva. Lo stato dell'arte è riassunto in (Gevros et al., 2001).

5.4 Qualità del servizio

Le tecniche descritte nei precedenti paragrafi sono state progettate per ridurre la congestione e migliorare le prestazioni della rete, ma con lo sviluppo delle reti multimediali spesso queste misure ad hoc non sono sufficienti: bisogna fare sforzi significativi per garantire la qualità del servizio attraverso architetture di protocollo e di rete. I paragrafi seguenti continuano lo studio sulle prestazioni della rete, concentrando però l'attenzione sui modi che consentono di fornire la qualità di servizio richiesta dalle applicazioni. Avvisiamo però che molte di queste idee sono ancora in fase di sviluppo e quindi in continuo cambiamento.

5.4.1 Requisiti

Un flusso di pacchetti diretto da una sorgente a una destinazione è chiamato semplicemente **flusso**. In una rete orientata alle connessioni tutti i pacchetti che appartengono a un flusso seguono lo stesso percorso; in una rete senza connessioni i pacchetti possono seguire percorsi differenti. Le esigenze di ogni flusso possono essere caratterizzate da quattro parametri primari: affidabilità, ritardo, jitter e banda. Insieme, questi parametri determinano la **QoS (Quality of Service)**, ossia la qualità del servizio richiesta dal flusso. La Figura 5.30 elenca diverse applicazioni note e il rigore dei corrispondenti requisiti.

Applicazione	Affidabilità	Ritardo	Jitter	Banda
Posta elettronica	Alta	Bassa	Bassa	Bassa
Trasferimento file	Alta	Bassa	Bassa	Media
Accesso al Web	Alta	Media	Bassa	Media
Login remoto	Alta	Media	Media	Bassa
Audio a richiesta	Bassa	Bassa	Alta	Media
Video a richiesta	Bassa	Bassa	Alta	Alta
Telefonia	Bassa	Alta	Alta	Bassa
Videoconferenza	Bassa	Alta	Alta	Alta

Figura 5.30. Rigidità dei requisiti relativi alla qualità del servizio.

Le prime quattro applicazioni hanno requisiti rigidi sull'affidabilità; nessun bit può essere trasmesso in modo scorretto. Questo obiettivo di solito è raggiunto creando il checksum di ogni pacchetto e verificandolo alla destinazione. Il pacchetto in transito che arriva danneggiato non riceve acknowledgement, di conseguenza alla fine sarà ritrasmesso. Questa strategia dà un'alta affidabilità. Le ultime quattro applicazioni (audio e video) possono tollerare errori, perciò nessun checksum è elaborato o verificato.

Le applicazioni di trasferimento file, inclusa la posta elettronica e il video, non sono sensibili al ritardo. Se tutti i pacchetti subiscono un ritardo uniforme di pochi secondi non

accade nulla di male. Applicazioni interattive, quali l'esplorazione del Web e l'accesso remoto, sono sensibili al ritardo. Applicazioni in tempo reale, quali la telefonia e la videoconferenza, hanno requisiti severi nei confronti del ritardo. Se tutte le parole di una chiamata telefonica sono ritardate esattamente di 2 secondi, gli utenti considereranno la connessione inaccettabile. D'altra parte, la riproduzione di file audio e video scaricati da un server non richiede un basso ritardo.

Le prime tre applicazioni non sono sensibili ai pacchetti che arrivano a intervalli irregolari. Il login remoto è un po' più sensibile a questo problema, poiché se la connessione soffre di un elevato jitter i caratteri sullo schermo potrebbero essere aggiornati a scatti. Il video e soprattutto l'audio sono estremamente sensibili allo jitter. Se un utente sta guardando un video trasmesso attraverso la rete e i fotogrammi hanno un ritardo di esattamente 2 secondi, non accade nulla di male; ma se il tempo di trasmissione varia in modo casuale tra 1 e 2 secondi, il risultato sarà terribile. Nel caso dell'audio, anche uno jitter di pochi millisecondi risulta chiaramente udibile. Infine, le applicazioni differiscono per le loro esigenze di banda: la posta elettronica e l'accesso remoto non richiedono molta banda, il video in tutte le sue forme invece ne richiede molta.

Le reti ATM classificano i flussi in quattro ampie categorie a seconda delle QoS richieste:

1. velocità costante (per esempio, telefonia)
2. velocità variabile in tempo reale (per esempio videoconferenza compressa)
3. velocità variabile non in tempo reale (per esempio la visione di un film trasmesso via Internet)
4. velocità disponibile (per esempio, il trasferimento di file).

Queste categorie sono utili anche per altre funzioni e altre reti. La velocità costante tenta di simulare un cavo fornendo una banda uniforme e un ritardo uniforme. La velocità variabile è usata in presenza di video compresso; alcuni fotogrammi si comprimono meglio di altri, perciò la trasmissione di un fotogramma contenente tanti dettagli può richiedere molti bit, mentre la trasmissione della ripresa di un muro bianco può essere compressa molto bene. La velocità disponibile è per le applicazioni, quali la posta elettronica, che non sono sensibili né al ritardo né allo jitter.

5.4.2 Tecniche per ottenere una buona qualità di servizio

Dopo aver dato uno sguardo ai requisiti di QoS è giunto il momento di scoprire come si possono raggiungere questi obiettivi. Innanzitutto, non esiste una bacchetta magica. Nessuna tecnica è in grado di fornire QoS efficiente e sicura in modo ottimo; sono state invece sviluppate diverse tecniche, e soluzioni pratiche che spesso ne combinano più di una. Questo paragrafo descrive alcune delle tecniche che i progettisti di sistemi usano per ottenere QoS.

Sovradimensionamento

Una soluzione semplice consiste nel fornire così tanta capacità del router, spazio di buffer e banda da far transitare facilmente i pacchetti. Il problema di questa soluzione è che è costosa.

sa. Quando i progettisti avranno un'idea del significato di "sufficienza" questa tecnica potrà anche diventare pratica. Entro certi limiti il sistema telefonico è sovradimensionato; è raro, sollevando la cornetta, non sentire il segnale di libero istantaneamente. C'è semplicemente una capacità talmente elevata che la domanda può sempre essere soddisfatta.

Utilizzo dei buffer

I flussi possono essere memorizzati dal dispositivo ricevente in un buffer prima di essere consegnati. L'archiviazione nei buffer non influenza né l'affidabilità né la banda, ma aumenta il ritardo ed elimina lo jitter. Nel caso dell'audio e del video a richiesta lo jitter è il problema principale, perciò la tecnica è veramente molto utile.

La Figura 5.29 ha mostrato la differenza tra un jitter elevato e un jitter basso. La Figura 5.31 mostra un flusso di pacchetti trasmesso con molto jitter. Il pacchetto 1 è inviato dal server a $t = 0$ e arriva al client al tempo $t = 1$. Il pacchetto 2 accumula un ritardo maggiore e arriva dopo 2 secondi. Non appena raggiungono il computer client, i pacchetti sono memorizzati in un buffer.

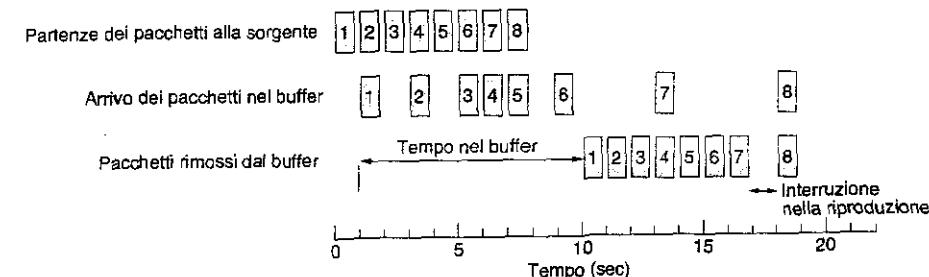


Figura 5.31. Aumento della fluidità del flusso di output mediante archiviazione dei pacchetti nei buffer.

Al tempo $t = 10$ sec la riproduzione ha inizio. In quell'istante, i pacchetti da 1 a 6 sono stati memorizzati, perciò possono essere rimossi dal buffer a intervalli uniformi garantendo una riproduzione fluida. Sfortunatamente, il pacchetto 8 ha accumulato molto ritardo e non è disponibile quando arriva il momento di riprodurlo, perciò la riproduzione deve interrompersi fino al suo arrivo; questo crea un'interruzione seccante nella musica o nel film. Il problema può essere alleviato ritardando un po' di più il tempo di inizio, sebbene questa scelta richieda un buffer più grande. I siti Web commerciali che contengono flussi audio e video utilizzano riproduttori che sono in grado di memorizzare nei buffer circa 10 secondi di dati prima di avviare la riproduzione.

Traffic shaping

Negli esempi descritti precedentemente la sorgente trasmette i pacchetti lasciando una spaziatura uniforme tra l'uno e l'altro; in altri casi invece i pacchetti possono essere emessi in modo irregolare, causando congestioni nella rete. Un output non uniforme è frequente se

il server gestisce molti flussi nello stesso momento e supporta altre azioni, come il riavvolgimento rapido in avanti o indietro, l'autenticazione dell'utente e così via. Inoltre, l'approccio utilizzato precedentemente (memorizzazione nel buffer) non è sempre possibile, come per esempio nel caso della videoconferenza. Se si potesse fare qualcosa per rendere uniforme la trasmissione del server (e degli host in generale) la qualità del servizio risulterebbe migliore. Questo paragrafo esamina una tecnica, chiamata **traffic shaping**, che fluidifica il traffico sul lato server, invece che su lato client.

Il traffic shaping regola la *velocità* media della trasmissione dei dati. Al contrario, i protocolli a finestra scorrevole studiati precedentemente limitano la quantità di dati in transito in ogni momento, e non la velocità di trasmissione. Quando una connessione viene impostata, l'utente e la sottorete (ossia il cliente e l'operatore di telecomunicazioni) si mettono d'accordo su un particolare modello di traffico (ossia su una forma) per quel circuito. Qualche volta questa operazione è chiamata **service level agreement**. Fintanto che il cliente si attiene alla sua parte dell'accordo e invia solo pacchetti secondo il contratto concordato, l'operatore di telecomunicazioni promette di consegnare tutti i dati in modo tempestivo. Il traffic shaping riduce la congestione e aiuta così l'operatore di telecomunicazioni a tener fede alle sue promesse. Questi accordi non sono molto importanti nel trasferimento dei file, ma sono di grande rilievo per i dati in tempo reale come le connessioni audio e video, che hanno rigorosi requisiti di qualità del servizio.

In realtà, con il traffic shaping, il cliente dice all'operatore di telecomunicazioni: "Il mio modello di trasmissione avrà questo aspetto; sei in grado di gestirlo?". Se accetta, l'operatore di telecomunicazioni deve avere il modo di verificare se il cliente sta seguendo l'accordo e deve sapere che cosa fare in caso negativo. La supervisione del flusso di traffico è chiamata **traffic policing**. È più facile concordare una forma di traffico e provvedere alla sua successiva supervisione nelle sottoreti a circuito virtuale che nelle sottoreti a datagrammi. Comunque, anche nelle sottoreti a datagrammi queste idee sono applicabili alle connessioni sullo strato di trasporto.

L'algoritmo leaky bucket

S'immagini un secchio con un piccolo buco sul fondo, come quello disegnato nella Figura 5.32(a). Qualunque sia la velocità d'ingresso dell'acqua nel secchio, l'efflusso avrà una velocità costante p quando ci sarà acqua nel secchio e avrà una velocità pari a 0 quando il secchio sarà vuoto. Inoltre, una volta che il secchio è pieno, l'acqua aggiuntiva versata nel contenitore si riverserà all'esterno e andrà perduta (ossia non apparirà nel flusso di output sotto il buco). La stessa idea può essere applicata anche ai pacchetti, come mostrato nella Figura 5.32 (b). Concettualmente, ogni host è collegato alla rete da un'interfaccia che contiene un leaky bucket (secchio che perde), ossia una coda interna finita. Un pacchetto è scartato quando entra nella coda già piena. In altre parole, se uno o più processi all'interno dell'host tentano di inviare un pacchetto quando il numero massimo è già accodato, il nuovo pacchetto viene scartato senza tante ceremonie. Questa configurazione può essere costruita nell'interfaccia hardware o simulata dal sistema operativo dell'host. È stata proposta per la prima volta da Turner (1986) ed è stata chiamata **algoritmo leaky bucket**. In effetti, non è altro che un sistema di accodamento a singolo server con tempo di servizio costante.

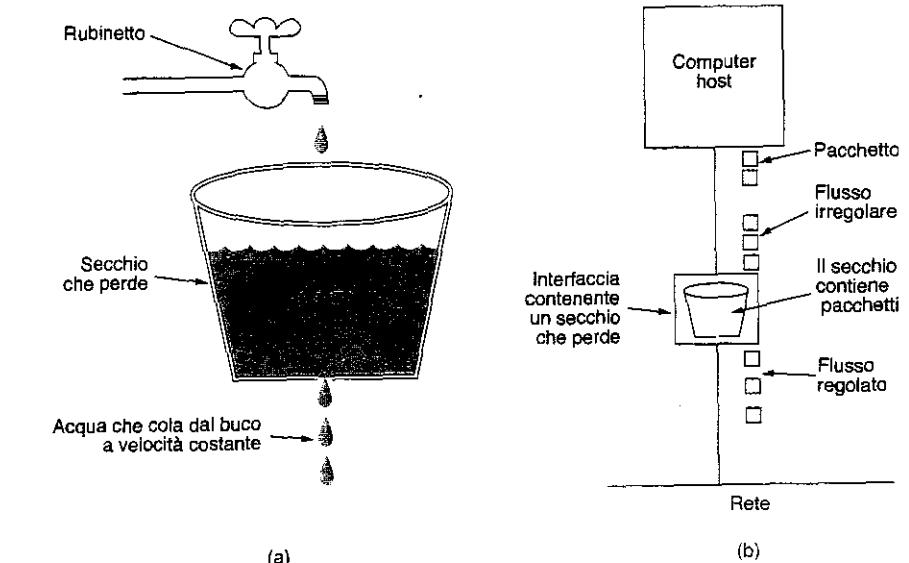


Figura 5.32. (a) Un secchio che perde acqua. (b) Un secchio che perde pacchetti.

L'host è autorizzato a inserire nella rete un solo pacchetto per ogni ciclo di clock; anche questo comportamento può essere imposto dalla scheda di rete o dal sistema operativo. Il meccanismo trasforma un flusso irregolare di pacchetti provenienti dai processi dell'utente dell'host in un flusso regolare di pacchetti immesso nella rete, appianando i picchi e riducendo enormemente le possibilità di congestioni.

Quando i pacchetti hanno tutti la stessa dimensione (come nelle celle ATM), questo algoritmo può essere utilizzato così come è stato descritto. Quando invece si usano pacchetti a dimensione variabile, è meglio consentire un numero fisso di byte per ogni ciclo di clock, invece che limitarsi a un unico pacchetto. Così, se la regola ammette 1.024 byte per ogni ciclo di clock, allora in un ciclo può essere introdotto un singolo pacchetto grande 1.024 byte, due pacchetti da 512 byte, quattro da 256 byte e così via. Se il conteggio dei byte residui è troppo basso, il pacchetto successivo deve attendere il prossimo ciclo.

È facile implementare l'algoritmo originale leaky bucket. Leaky bucket è composto da una coda finita. Al suo arrivo, se c'è spazio il pacchetto viene aggiunto alla coda, altrimenti viene scartato. A ogni ciclo di clock viene trasmesso un pacchetto (a meno che la coda non sia vuota).

Il leaky bucket a conteggio di byte è implementato quasi nello stesso modo. Per ogni ciclo un contatore è inizializzato a n . Se il primo pacchetto della coda ha meno byte del valore corrente del contatore, allora viene trasmesso e il contatore è decrementato di quel numero di byte. Pacchetti aggiuntivi possono essere trasmessi fintanto che il valore del contatore è abbastanza alto. Quando il contatore scende sotto la lunghezza del pacchetto successivo nella coda, la trasmissione s'interrompe fino al ciclo di clock successivo, quando il conteggio dei byte residui è azzerato e il flusso può riprendere.

Come esempio di leaky bucket, si supponga che un computer possa produrre dati a 25 milioni di byte/sec (200 Mbps) e che la rete funzioni alla stessa velocità, mentre i router possono accettarla solo per brevi intervalli (in genere fino a quando i buffer non si riempiono completamente). Per lunghi intervalli, essi funzionano meglio a velocità che non superano i 2 milioni di byte/sec. Ora, si supponga che i dati arrivino a raffiche di 1 milione di byte, lunghe 40 msec, che si ripetono ogni secondo. Per ridurre la velocità media a 2 MB/sec si potrebbe utilizzare un leaky bucket con $\rho = 2$ MB/sec e una capacità C di 1 MB. Questo significa che le raffiche più piccole di 1 MB possono essere gestite senza perdere dati, venendo disperse su 500 msec qualunque sia la velocità di arrivo.

La Figura 5.33(a) mostra i dati che entrano nel leaky bucket a 25 MB/sec per 40 msec. La Figura 5.33(b) mostra l'output trasmesso a velocità uniforme di 2 MB/sec per 500 msec.

L'algoritmo token bucket

L'algoritmo leaky bucket impone un rigido modello di output a velocità media, che non segue la variabilità del traffico. Per molte applicazioni è meglio permettere all'output di accelerare un po' quando arrivano grandi raffiche di dati, perciò è necessario un algoritmo più flessibile, preferibilmente che non perda mai dati. Un algoritmo di questo tipo è l'**algoritmo token bucket**. In questo algoritmo, il leaky bucket contiene token, generati da un clock che ha velocità di un token ogni DT sec. La Figura 5.34(a) mostra un secchio contenente tre token, con cinque pacchetti in attesa di essere trasmessi. Perché possa essere trasmesso, un pacchetto deve catturare e distruggere un token. Nella Figura 5.34 (b) si nota che tre dei cinque pacchetti sono passati, ma gli altri due sono bloccati in attesa che vengano generati altri token.

La modellatura del traffico dell'algoritmo token bucket è diversa da quella dell'algoritmo leaky bucket. L'algoritmo leaky bucket non permette agli host inattivi di risparmiare permessi in modo da inviare successivamente grandi raffiche di dati; l'algoritmo token bucket consente di risparmiare fino a riempire la dimensione massima del secchio, n . Questa proprietà permette di trasmettere raffiche composte da un massimo di n pacchetti, perciò può generare flussi di output meno regolari e dare una risposta più veloce a picchi improvvisi in ingresso.

Un'altra differenza tra di due algoritmi è che quello token bucket getta via i token (ossia capacità di trasmissione) quando il secchio si riempie completamente, ma non scarta mai i pacchetti. Al contrario, l'algoritmo leaky bucket scarta i pacchetti quando il secchio è pieno. Esiste una piccola variante, in cui ogni token rappresenta il diritto di inviare non un pacchetto ma k byte. Un pacchetto può essere trasmesso solo se è disponibile un numero di token sufficiente a coprire la sua lunghezza espressa in byte. Token frazionari sono conservati per un uso futuro.

Gli algoritmi leaky bucket e token bucket si possono anche usare per regolarizzare il traffico tra i router, oltre che per regolare l'output dell'host come nei nostri esempi. Comunque una chiara differenza è che un token bucket che regola un host può costringere l'host a interrompere la trasmissione quando le regole dicono di smettere; invece istruire un router a interrompere la trasmissione mentre il suo input continua ad arrivare può far perdere dati.

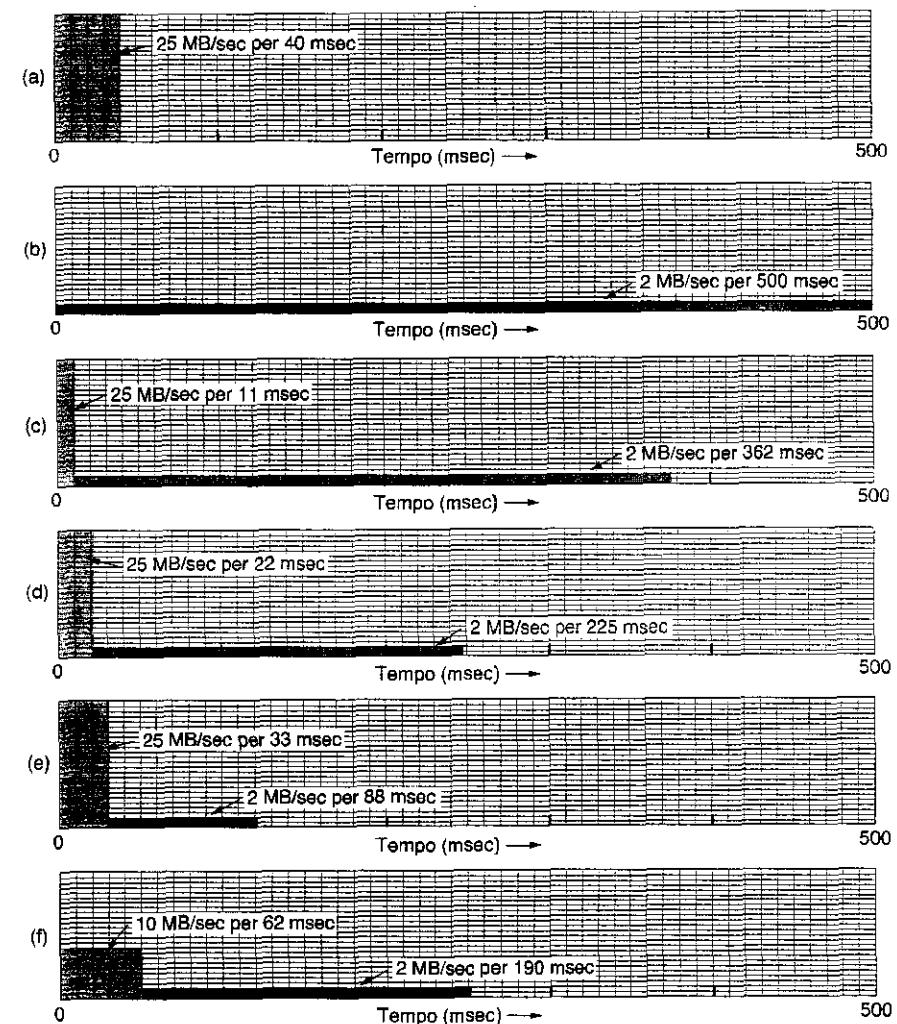


Figura 5.33. (a) Input del leaky bucket. (b) Output del leaky bucket. Output da un token bucket con capacità di (c) 250 KB, (d) 500 KB e (e) 750 KB. (f) Output da un token bucket da 500 KB che alimenta un leaky bucket a 10 MB/sec.

L'implementazione dell'algoritmo elementare del token bucket richiede solo una variabile che conti i token. Il contatore è incrementato di uno ogni DT ed è decrementato di uno ogni volta che viene trasmesso un pacchetto. Quando il contatore raggiunge lo zero, nessun pacchetto può essere trasmesso.

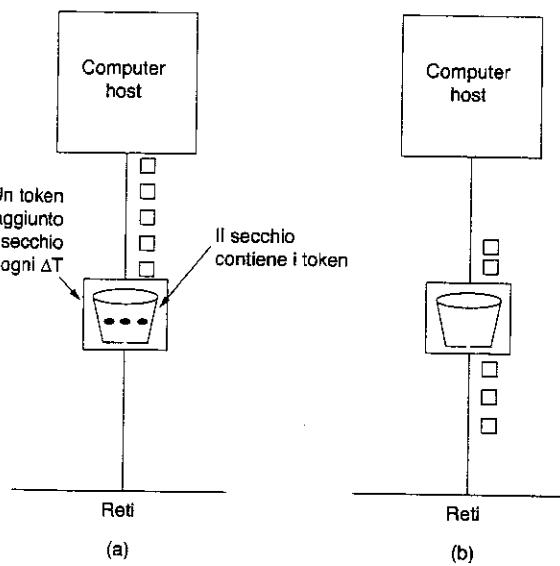


Figura 5.34. L'algoritmo token bucket. (a) Prima. (b) Dopo.

Nella variante che conta i byte, il contatore è incrementato di k byte ogni ΔT ed è decrementato della lunghezza di ogni pacchetto trasmesso.

Essenzialmente il token bucket permette raffiche di dati, ma solo se non superano una lunghezza massima prefissata. Si osservi l'esempio illustrato nella Figura 5.33(c), dove vediamo un token bucket che ha una capacità di 250 KB. I token arrivano a una velocità che consente di trasmettere l'output a 2 MB/sec. Supponendo che il token bucket sia pieno quando arriva una raffica di 1 MB di dati, il secchio può svuotarsi a 25 MB/sec per circa 11 msec; poi deve tornare a 2 MB/sec fino a che non è stata spedita l'intera raffica in input. Non è molto facile calcolare la lunghezza della raffica a velocità massima. Non è sufficiente dividere 1 MB per 25 MB/sec, perché arrivano altri token mentre la raffica viene trasmessa. Sia S la lunghezza della raffica, C la capacità del token bucket, p Byte/sec la velocità di arrivo dei token e M byte/sec la velocità di output massima; allora una raffica di output contiene un massimo di $C + pS$ byte. Si sa anche che il numero di byte in una raffica a velocità massima di lunghezza S secondi è MS . Quindi

$$C + pS = MS$$

Si può risolvere l'equazione per ottenere $S = C/(M - p)$. Per i parametri di $C = 250$ KB, $M = 25$ MB/sec e $p = 2$ MB/sec si ottiene un tempo di raffica di circa 11 msec. La Figura 5.33(d) e la Figura 5.33(e) mostrano due secchi a token della capacità di 500 KB e 750 KB rispettivamente.

Un problema potenziale dell'algoritmo token bucket è che consente di trasmettere grandi raffiche di dati, anche se l'intervallo massimo della raffica può essere regolato selezio-

nando attentamente p e M . Spesso si desidera ridurre la velocità di picco, ma senza tornare al basso valore del leaky bucket originale.

Si può ottenere un traffico più uniforme inserendo un leaky bucket dopo il token bucket. La velocità del leaky bucket dovrebbe essere più alta di quella p del token bucket, ma più bassa della velocità massima della rete. La Figura 5.33(f) mostra l'output di un token bucket da 500 KB seguita da un leaky bucket da 10 MB/sec.

La supervisione di tutti questi schemi può risultare un po' complessa. Essenzialmente, la rete deve simulare l'algoritmo e assicurarsi che non vengano trasmessi più pacchetti o byte di quelli consentiti. Ciò nonostante, questi strumenti permettono di dare al traffico di rete forme più maneggevoli che aiutano a soddisfare i requisiti di qualità del servizio.

Prenotazione delle risorse

Poter regolare la forma del traffico offerto è un buon punto di partenza per garantire la qualità del servizio. Comunque, affinché questa informazione possa essere utilizzata efficacemente, è implicitamente necessario che tutti i pacchetti di un flusso seguano lo stesso percorso; sparagliare i pacchetti su router scelti a caso rende difficile garantire qualunque cosa. Di conseguenza, è necessario impostare qualcosa che assomigli a un circuito virtuale tra sorgente e destinazione, e tutti i pacchetti che appartengono al flusso devono seguire questo percorso.

Una volta definito il percorso del flusso, diventa possibile prenotare lungo tutto il tracciato le risorse che garantiscono la disponibilità della capacità necessaria. Potenzialmente si possono prenotare tre diversi tipi di risorse:

1. banda
2. spazio del buffer
3. cicli di CPU.

La banda è la risorsa più ovvia. Se un flusso richiede 1 Mbps e la linea di trasmissione ha una capacità di 2 Mbps, tentare di spingere tre flussi attraverso quella linea non è possibile: prenotare la banda significa non sovraccaricare alcuna linea di output.

Una seconda risorsa che spesso scarreggia è lo spazio dei buffer. Quando arrivano, i pacchetti di solito sono depositati dall'hardware nella scheda di rete stessa. Il software del router poi copia i dati in un buffer in RAM, che accoda per la trasmissione sulla linea di output scelta. Se non è disponibile alcun buffer, il pacchetto deve essere scaricato poiché non c'è un posto dove tenerlo. Per una buona qualità di servizio, alcuni buffer possono essere dedicati a un flusso specifico, in modo che quel flusso non debba competere con altri: ci sarà sempre un buffer disponibile quando ne avrà bisogno, fino a un certo massimo.

Infine, anche i cicli di CPU sono una risorsa limitata. L'elaborazione di un pacchetto consuma tempo di CPU del router, perciò un apparecchio può elaborare solo un certo nume-

ro di pacchetti al secondo. Accertandosi che la CPU non sia sovraccaricata si può garantire un'elaborazione tempestiva di ogni pacchetto. A prima vista si potrebbe credere che se ci vuole, per esempio, 1 msec per elaborare un pacchetto, un router può elaborare 1 milione di pacchetti al secondo. Questa osservazione non è vera, perché ci saranno sempre periodi di inattività causati da fluttuazioni statistiche nel carico. Se la CPU ha bisogno di ogni singolo ciclo per svolgere il suo lavoro, anche la perdita di pochi cicli dovuta a inattività occasionali crea un arretrato che non potrà mai essere recuperato.

Comunque, anche con un carico leggermente sotto la capacità teorica, le code possono incrementarsi e i ritardi aumentare. Si consideri una situazione in cui i pacchetti arrivano a caso con una velocità di arrivo media di Δ pacchetti/sec. Anche il tempo di CPU richiesto da ogni pacchetto è casuale, con una capacità di elaborazione media di m pacchetti/sec. Supponendo che la distribuzione di servizio e quella di arrivo siano distribuzioni di Poisson, usando la teoria delle code si può dimostrare che il ritardo medio T sperimentato da un pacchetto è

$$T = \frac{1}{\mu} \times \frac{1}{1 - \lambda / \mu} = \frac{1}{\mu} \times \frac{1}{1 - \rho}$$

Dove $\rho = \Delta/\mu$ rappresenta l'utilizzo della CPU. Il primo fattore, $1/\mu$, rappresenta il tempo di servizio in assenza di competizione. Il secondo fattore indica il rallentamento causato dalla competizione con altri flussi. Per esempio, se $\lambda = 950.000$ pacchetti/sec e $\mu = .000.000$ pacchetti/sec, allora $\rho = 0,95$ e il ritardo medio sperimentato da ogni pacchetto sarà di 20 μ sec invece di 1 μ sec. Questo tempo comprende sia il tempo di accodamento sia il tempo di servizio, come si può osservare quando il carico è molto basso ($\lambda/\mu = 0$). Per esempio, se lungo il percorso del flusso ci fossero 30 router, il solo ritardo di accodamento causerebbe un ritardo di 600 μ sec.

Controllo di ammissione

I è arrivati al punto in cui il traffico in arrivo associato a un flusso è ben modellato e può idealmente seguire un singolo percorso in cui la capacità può essere prenotata in anticipo dai router lungo il percorso. Quando un flusso di questo tipo è offerto a un router, il dispositivo deve decidere, in base alla propria capacità e al numero di impegni già presi con altri flussi, se accettare il nuovo flusso oppure no.

La decisione di accettare o rifiutare un flusso non dipende solo dall'esito del confronto tra banda, buffer e cicli richiesti dal flusso e la capacità in eccesso del router in queste tre dimensioni; è un po' più complicato. Prima di tutto, sebbene alcune applicazioni possano conoscere i loro requisiti di banda, solo alcune sanno valutare i requisiti relativi ai buffer ai cicli di CPU, perciò è necessario avere minimo un sistema di descrizione dei flussi inversi. In secondo luogo, alcune applicazioni sono molto più tolleranti di altre nei confronti di limiti che occasionalmente non vengono mantenuti. Infine, alcune applicazioni possono essere disposte a contrattare i parametri del flusso, altre invece no. Per esempio, un riproduttore di film che normalmente visualizza 30 fotogrammi al secondo può accettare di scendere a 25 fotogrammi al secondo se la banda disponibile non è in grado di sup-

portare la velocità di 30 fotogrammi al secondo; in modo analogo è possibile regolare il numero di pixel per fotogramma, la banda audio e altre proprietà.

Poiché nella negoziazione del flusso possono essere coinvolte molte parti (il trasmittente, il ricevitore e tutti i router lungo il percorso che collega i due punti), i flussi devono essere descritti accuratamente in termini di parametri specifici che possono essere negoziati. Un insieme di parametri di questo tipo è chiamato **specifiche di flusso**. In genere, il trasmittente (per esempio il server che distribuisce i video) produce una specifica di flusso per proporre i parametri da utilizzare. Quando la specifica si propaga lungo il percorso, ogni router la esamina e modifica i parametri in base alle necessità. Le modifiche possono solo ridurre il flusso, non possono aumentarlo (ossia, si può diminuire la velocità dati ma non aumentarla). Quando arriva all'altro capo, i parametri possono essere fissati.

Come esempio di quello che ci può essere in una specifica di flusso si consideri la Figura 5.35, che si riferisce a un caso basato su RFC 2210 e 2211. Ci sono cinque parametri; il primo, *token bucket rate*, rappresenta il numero di byte al secondo immessi nel secchio. Questa è la massima velocità di trasmissione sostenuta della sorgente, mediata su un lungo intervallo.

Significato	Unità
Token bucket rate	Byte/sec
Token bucket size	Byte
Peak data rate	Byte/sec
Dimensione minima del pacchetto	Byte
Dimensione massima del pacchetto	Byte

Figura 5.35. Un esempio di specifica di flusso.

Il secondo parametro rappresenta la dimensione del secchio espressa in byte. Per esempio, se *token bucket rate* è di 1 Mbps e *token bucket size* è di 500 KB, il secchio può riempirsi in continuazione per 4 sec prima di esaurirsi (in assenza di trasmissioni). I token inviati successivamente saranno perduti. Il terzo parametro, *peak data rate*, è la velocità di trasmissione massima tollerata, anche per brevi intervalli di tempo. Il trasmittente non deve mai superare questa velocità.

Gli ultimi due parametri specificano la dimensione minima e massima del pacchetto, incluse le intestazioni degli strati network e di trasporto (per esempio, TCP e IP). La dimensione minima è importante perché l'elaborazione di ogni pacchetto richiede un tempo prefissato, qualunque sia la lunghezza. Un router può essere pronto a gestire 10.000 pacchetti/sec grandi 1 KB, ma non 100.000 pacchetti/sec grandi 50 byte ognuno, anche se questo valore rappresenta una velocità dati più bassa. La dimensione massima del pacchetto è importante a causa delle limitazioni interne della rete che non possono essere superate. Per esempio, se parte del percorso attraversa una Ethernet, la dimensione di ogni pacchetto non potrà superare i 1.500 byte anche se il resto della rete supporta dimensioni maggiori.

Una domanda interessante riguarda il modo in cui un router trasforma una specifica di flusso in una serie di prenotazioni di risorse specifiche. Questa associazione è specifica dell'implementazione e non è standardizzata. Si supponga che un router possa elaborare 100.000 pacchetti/sec. Se al dispositivo viene presentato un flusso di 1 MB/sec con dimensione minima e massima del pacchetto di 512 byte, il router calcola di poter ricevere 2.048 pacchetti/sec da quel flusso. In questo caso, deve riservare a quel flusso il 2% della sua CPU, e preferibilmente un po'di più in modo da evitare lunghi ritardi di accodamento. Se un criterio impone al router di non allocare più del 50% della CPU (il che implica un fattore due di ritardo) e il dispositivo ha già assegnato il 49% della risorsa, allora il flusso deve essere rifiutato. Calcoli simili sono necessari per le altre risorse.

Più la specifica di flusso è stretta, più è utile ai router. Se una specifica di flusso dichiara che è necessaria una velocità del token bucket di 5 MB/sec ma i pacchetti hanno dimensioni che variano dai 50 byte ai 1.500 byte, allora la velocità di pacchetto sarà compresa tra 3.500 pacchetti/sec e 105.000 pacchetti/sec. Il router può lasciarsi prendere dal panico a causa del secondo valore e rifiutare il flusso, mentre con una dimensione minima di 1.000 byte il flusso di 5 MB/sec verrebbe accettato.

Routing proporzionale

La maggior parte degli algoritmi di routing tenta di trovare il percorso migliore per ogni destinazione, per inviare attraverso di esso tutto il traffico. Un approccio diverso, che è stato proposto per offrire una qualità di servizio più elevata, divide su più percorsi il traffico diretto a ogni destinazione. Poiché i router generalmente non hanno una visione completa del traffico su tutta la rete, il traffico deve essere diviso su più percorsi utilizzando solamente informazioni disponibili localmente. Un metodo semplice consiste nel dividere il traffico in parti uguali, o proporzionali alla capacità dei collegamenti diretti verso l'esterno. Algoritmi più sofisticati sono descritti in (Nelakuditi and Zhang, 2002).

Pianificazione dei pacchetti

Se un router sta gestendo più flussi, c'è il pericolo che un flusso accappari troppa capacità lasciando senza risorse tutti gli altri. Se i pacchetti sono elaborati nell'ordine con cui arrivano, host aggressivi sono liberi di catturare la maggior parte della capacità dei router attraversati dai pacchetti, riducendo la qualità di servizio degli altri host. Per impedire che questo accada, sono stati sviluppati diversi algoritmi di pianificazione dei pacchetti (Bhatti e Crowcroft, 2000).

Uno dei primi è stato l'algoritmo di **accodamento equo** (*fair queuing*, Nagle, 1987). L'essenza di questo algoritmo sta nell'attribuire ai router code separate per ogni linea di trasmissione, una per ogni flusso. Quando una linea si libera, il router esamina le code in nodo round robin prendendo il primo pacchetto della coda successiva. In questo modo, su n host che concorrono per una data linea di trasmissione, ogni host riesce a trasmettere uno dei suoi pacchetti ogni n pacchetti. L'invio di un numero maggiore di pacchetti non migliora questa frazione.

L'algoritmo è meglio di niente, ma ha un problema: offre più banda agli host che usano pacchetti grandi. Demers et al. (1990) ha suggerito un miglioramento in cui il round robin

è fatto in modo da simulare una sequenza byte per byte, invece che pacchetto per pacchetto. In pratica, questa variante esamina ripetutamente le code un byte alla volta, fino a quando non trova il ciclo di clock finale di ogni pacchetto. I pacchetti sono quindi ordinati in base al loro completamento e vengono trasmessi esattamente in questo ordine.

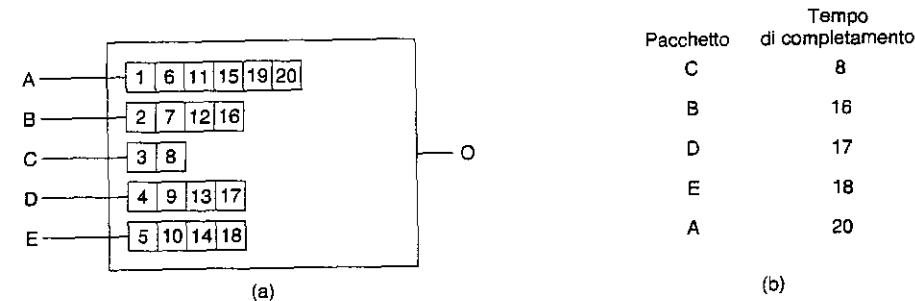


Figura 5.36. (a) Un router con cinque pacchetti accodati sulla linea O. (b) Tempi di completamento dei cinque pacchetti.

La Figura 5.36(a) mostra pacchetti di lunghezza compresa tra 2 e 6 byte. All'istante (virtuale) di clock 1 viene trasmesso il primo byte sulla linea A, poi viene il turno del primo byte del pacchetto che si trova sulla linea B e così via. Il primo pacchetto completato è C, dopo otto cicli di clock; l'ordine di fine trasmissione dei pacchetti è riportato nella Figura 5.36(b). In assenza di nuovi arrivi, i pacchetti saranno inviati nell'ordine indicato, da C ad A. Un problema di questo algoritmo è che dà a tutti gli host la stessa priorità. In molte situazioni è utile dare ai server video più banda di quella offerta ai server di file, in modo da offrire ai primi due o più byte per ciclo di clock. L'algoritmo modificato, largamente utilizzato, è stato chiamato **accodamento equo pesato**. Qualche volta il peso è uguale al numero di flussi trasmessi da un computer, perciò ogni processo riceve la stessa banda. Un'implementazione efficiente dell'algoritmo è descritta in (Shreedhar and Varghese, 1995). Sempre più spesso, l'inoltro dei pacchetti attraverso un router o uno switch è eseguito nell'hardware. (Elhanany et al., 2001).

5.4.3 Servizi integrati

Tra il 1995 e il 1997 IETF dedicò molte energie allo sviluppo di un'architettura per la trasmissione dei flussi di dati multimediali. Il risultato di questo lavoro fu raccolto in più di venti RFC, a partire da RFC 2205-2210. Il nome generico di questa opera è **algoritmi basati sui flussi o servizi integrati**. Il lavoro era rivolto alle applicazioni unicast e multicast. Un esempio di applicazione unicast è rappresentato dal singolo utente che scarica un video clip da un sito di notizie; un esempio di applicazione multicast è rappresentato da un gruppo di stazioni televisive digitali che trasmettono i loro programmi, sotto forma di flussi di pacchetti IP, a molti ricevitori sparsi in più locazioni. Questo paragrafo si concentra soprattutto sulla trasmissione multicast, poiché quella unicast può essere considerata un caso speciale della multicast.

In molte applicazioni multicast i gruppi possono cambiare dinamicamente l'appartenenza; per esempio, ci possono esserci persone che, dopo essersi unite a una videoconferenza, prese dalla noia decidono di guardare un dramma sentimentale a puntate o il canale delle bocce. In queste condizioni l'approccio che fa riservare in anticipo la banda ai trasmettitori non funziona bene, poiché sarebbe necessario che ogni trasmettitore tenesse traccia di tutte gli ingressi e le uscite del pubblico. In un sistema progettato per trasmettere televisione a milioni di abbonati ciò non funzionerebbe.

RSVP (Resource reSerVation Protocol)

Il protocollo IETF principale per l'architettura di servizi integrati è **RSVP**, descritto in RFC 2205 e altri documenti. Questo protocollo si occupa delle prenotazioni; altri protocolli sono utilizzati per trasmettere i dati. RSVP consente a più trasmettitori di trasmettere a diversi gruppi di ricevitori, permette a singoli ricevitori di cambiare canale liberamente, e ottimizza l'uso della banda eliminando contemporaneamente le congestioni.

Nella sua forma più semplice, il protocollo utilizza il routing multicast basato su strutture di tipo spanning tree (descritte precedentemente). Ogni gruppo riceve un indirizzo di gruppo. Per trasmettere a un gruppo, un trasmettitore inserisce l'indirizzo del gruppo nei suoi pacchetti; in questo modo l'algoritmo di routing multicast standard (che non fa parte di RSVP) costruisce uno spanning tree che copre tutti i membri del gruppo. L'unica differenza rispetto alla normale trasmissione multicast è rappresentata da qualche informazione in più, trasmessa periodicamente in modalità multicast al gruppo per dire ai router lungo la struttura ad albero di conservare alcune strutture dati nella loro memoria.

Come esempio si consideri la rete mostrata nella Figura 5.37(a). Gli host 1 e 2 sono trasmettitori multicast, e gli host 3, 4 e 5 sono ricevitori multicast. In questo esempio i trasmettitori e i ricevitori sono disgiunti, ma in generale i due gruppi possono sovrapporsi. Le Figure 5.37(b) e 5.37(c) mostrano le strutture ad albero relative agli host 1 e 2.

Per avere una migliore ricezione ed eliminare le congestioni, tutti i ricevitori di un gruppo possono inviare al trasmettitore un messaggio di prenotazione attraverso la struttura ad albero. Il messaggio è propagato usando l'algoritmo di reverse path forwarding descritto precedentemente. Ad ogni salto il router prende nota della prenotazione, e riserva la banda necessaria; se non è disponibile una banda sufficiente, il dispositivo risponde comunicando il fallimento. Quando il messaggio ritorna alla sorgente, la banda è stata prenotata su tutto il percorso che collega il trasmettitore al ricevitore che ha inviato la richiesta di prenotazione lungo lo spanning tree.

Un esempio di prenotazione di questo tipo è mostrato nella Figura 5.38(a). L'host 3 ha richiesto un canale all'host 1; una volta attivato i pacchetti possono scorrere da 1 a 3 senza creare congestioni. Si supponga che in seguito l'host 3 prenoti un canale di collegamento verso un altro trasmettitore, per esempio l'host 2, per dare all'utente la possibilità di guardare contemporaneamente due programmi televisivi. In questo caso viene riservato un secondo percorso, come mostrato nella Figura 5.38(b). Si noti che dall'host 3 al router E occorrono due canali separati perché stanno per essere trasmessi due flussi indipendenti.

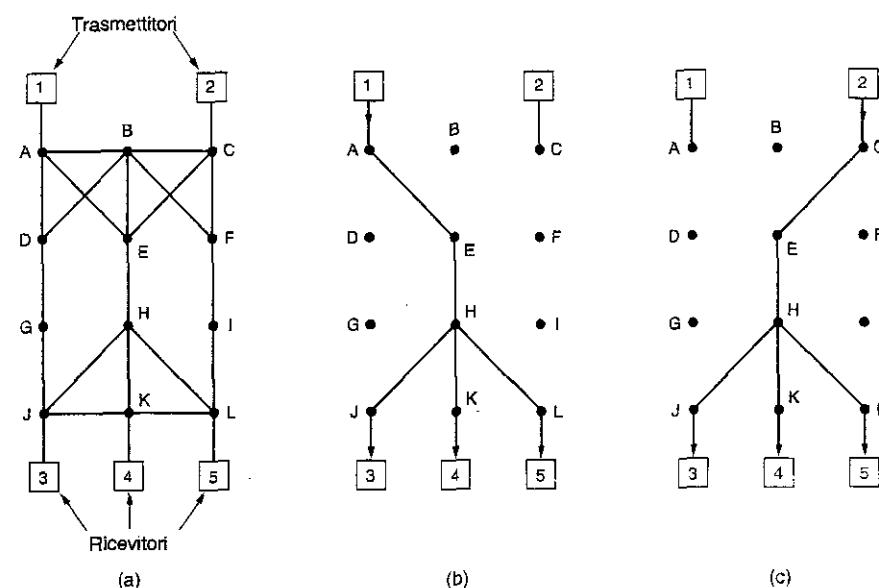


Figura 5.37. (a) Una rete. (b) Lo spanning tree multicast dell'host 1.
(c) Lo spanning tree multicast dell'host 2.

Infine, nella Figura 5.38(c), l'host 5 decide di guardare il programma trasmesso dall'host 1 e fa un'altra prenotazione. Prima di tutto, la banda dedicata è riservata fino al router H, ma questo router si accorge di essere già alimentato dall'host 1, perciò se la banda necessaria è già stata riservata non è necessario riservarne altra. Si noti che gli host 3 e 5 potrebbero aver chiesto bande diverse (per esempio, l'host 3 potrebbe utilizzare un televisore in bianco e nero e quindi non aver bisogno delle informazioni relative al colore), perciò la capacità riservata deve essere abbastanza larga da soddisfare il ricevitore più esigente. Quando fa una prenotazione, il ricevitore può (facoltativamente) specificare una o più sorgenti dalle quali desidera ricevere i dati; può anche specificare se queste scelte sono fisse per la durata della prenotazione o se preferisce aver la possibilità di modificare le sorgenti in un secondo momento. I router utilizzano queste informazioni per ottimizzare la pianificazione della banda. In particolare, due ricevitori sono configurati per condividere un percorso solo se entrambi accettano di non cambiare le risorse in un secondo tempo. Il motivo di questa strategia nel caso completamente dinamico è che la banda riservata è disaccoppiata dalla scelta della sorgente. Una volta che ha riservato la banda, il ricevitore può passare a un'altra sorgente e mantenere la parte del percorso esistente che è valida per la nuova sorgente. Se per esempio l'host 2 sta trasmettendo diversi flussi video, l'host 3 può passare dall'uno all'altro senza cambiare la sua prenotazione: ai router non interessa sapere quale programma sta guardando il ricevitore.

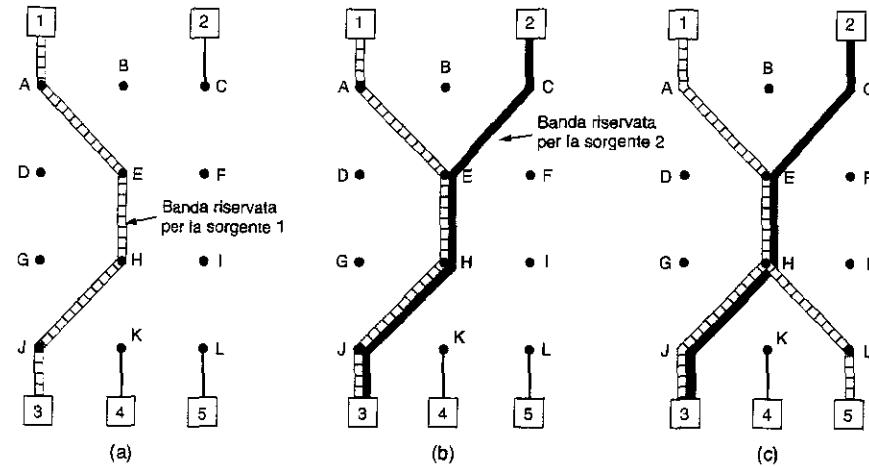


Figura 5.38. (a) L'host 3 richiede un canale all'host 1. (b) Successivamente l'host 3 richiede un secondo canale all'host 2. (c) L'host 5 richiede un canale all'host 1.

5.4.4 Servizi differenziati

Gli algoritmi basati sui flussi hanno la capacità di offrire una buona qualità di servizio a uno o più flussi perché riservano tutte le risorse necessarie lungo il percorso. Comunque, hanno anche un lato negativo: richiedono la preventiva impostazione di ogni flusso, approccio che non si adatta bene a sistemi dove circolano migliaia o milioni di flussi. Mantengono inoltre nei router uno stato interno per ogni flusso, cosa che li rende vulnerabili ai guasti dei router. Infine, le modifiche da apportare al codice dei router sono considerevoli e coinvolgono complessi scambi da router a router per l'impostazione dei flussi. È per questo motivo che oggi esistono poche implementazioni di RSVP.

IETF ha pertanto concepito anche un approccio più semplice mirato alla qualità di servizio, che può essere implementato per lo più localmente in ogni router, senza richiedere l'impostazione anticipata e senza coinvolgere l'intero percorso. Questo approccio è chiamato qualità di servizio **basata sulle classi** (in contrapposizione al sistema basato sui flussi). IETF ha standardizzato un'architettura per questa soluzione, chiamata **servizi differenziati** (*differentiated services*), descritta in RFC 2474, 2475 e in numerosi altri documenti. L'architettura è descritta in questo paragrafo.

I servizi differenziati (DS) possono essere offerti da un insieme di router che costituiscono un dominio amministrativo (per esempio, un ISP o una compagnia telefonica). L'amministrazione definisce una serie di classi di servizio e le corrispondenti regole di inoltro. Se un cliente firma un contratto per DS, i pacchetti del cliente che entrano nel dominio possono contenere un campo *type of service* (tipo di servizio); alcune classi avranno diritto a un servizio migliore di altre (servizio premium). Il traffico appartenente a una data classe può essere obbligato a seguire una forma specifica, per esempio a una basata sul leaky bucket con una velocità di scarico specificata. Un operatore abile negli

affari potrebbe far pagare per ogni pacchetto premium aggiuntivo trasportato, oppure potrebbe consentire l'invio di massimo N pacchetti premium al mese con un contributo mensile aggiuntivo. È bene notare che questo schema, a differenza di quello adottato dai servizi integrati, non richiede alcuna impostazione anticipata, nessuna prenotazione di risorse e nessuna pesante negoziazione punto-punto per ogni flusso; ciò rende DS relativamente facile da implementare.

Il servizio basato sulle classi è utilizzato anche in altre industrie. Per esempio, le aziende che si occupano della consegna di pacchi spesso offrono un servizio notturno, un servizio di recapito in due giorni e un servizio di consegna in tre giorni; le linee aeree offrono un servizio di prima classe, di classe economica e di classe bestiame; treni su lunghe distanze spesso hanno più classi di servizio; anche la metropolitana di Parigi ha due classi di servizio. Per i pacchetti, le classi possono differire tra l'altro per il ritardo, lo jitter e la probabilità di essere scartate in caso di congestione.

Per rendere ancora più chiara la differenza tra la qualità di servizio basata sui flussi e quella basata sulle classi, si consideri l'esempio seguente: la telefonia via Internet. Con uno schema basato sui flussi, ogni chiamata telefonica riceve le sue risorse e garanzie; con uno schema basato sulla classe, tutto l'insieme delle chiamate telefoniche riceve le risorse prenotate per la classe della telefonata. Queste risorse non possono essere sottratte dai pacchetti che appartengono alla classe associata al trasferimento dei file (o ad altre classi), ma nessuna singola chiamata telefonica riceve risorse private riservate solo a lei.

Inoltro accelerato

La scelta delle classi di servizio spetta a ogni operatore, ma poiché i pacchetti sono spesso inoltrati tra sottoreti gestite da operatori diversi, IETF sta cercando di definire classi di servizio indipendenti dalle reti. La classe più semplice, e la prima esaminata in questo paragrafo, si chiama **expedited forwarding** (*inoltro accelerato*) ed è descritta nel documento RFC 3246.

L'idea alla base dell'inoltro accelerato è molto semplice. Sono disponibili due classi di servizio: una regolare e una accelerata. La maggior parte del traffico dovrà essere considerata regolare, mentre una piccola frazione di pacchetti potrà essere accelerata. I pacchetti accelerati dovrebbero poter attraversare la sottorete come se non fosse presente nessun altro pacchetto. Una rappresentazione simbolica di questo sistema "a due condotte" è visibile nella Figura 5.39. Si noti che c'è ancora una sola linea fisica. Le due condotte logiche mostrate nella figura rappresentano un modo di riservare la banda, non una seconda linea fisica.

Questa strategia può essere implementata, per esempio, programmando i router in modo che abbiano due code di output per ogni linea in uscita, una per i pacchetti accelerati e l'altra per i pacchetti regolari. Al suo arrivo, il pacchetto è inserito nella coda appropriata. La pianificazione dei pacchetti dovrebbe utilizzare un sistema simile all'accodamento equo pesato. Per esempio, se si pensa che il 10% del traffico sarà accelerato e il 90% sarà regolare, il 20% della banda potrebbe essere dedicato al traffico accelerato e il resto al traffico regolare. In questo modo, il traffico accelerato riceverà una banda pari al doppio di quella richiesta, per ridurre il ritardo della trasmissione. Questa distribuzione può essere otte-

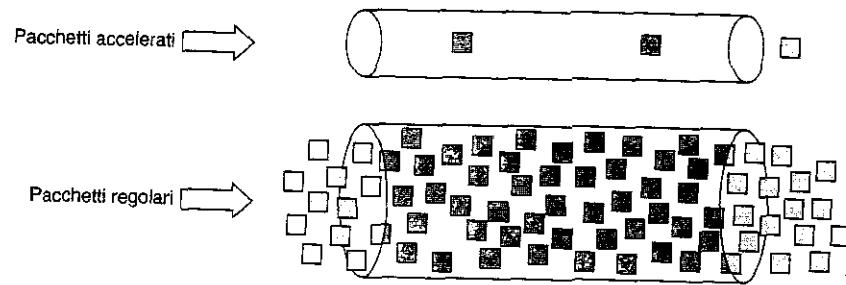


Figura 5.39. Pacchetti accelerati incontrano una rete senza traffico.

nutra trasmettendo un pacchetto accelerato ogni quattro pacchetti regolari (supponendo che per entrambe le classi la distribuzione della dimensione sia simile). In tal modo, si spera che i pacchetti accelerati trovino la sottorete scarica anche quando, in realtà, c'è un carico pesante.

Inoltro sicuro

Assured forwarding (inoltro sicuro) è il nome di uno schema di gestione delle classi di servizio un po' più elaborato, descritto in RFC 2597. Il documento specifica quattro classi di priorità, dove ogni classe ha a disposizione le proprie risorse. Inoltre, definisce tre probabilità di eliminazione per i pacchetti che stanno subendo gli effetti di una congestione: bassa, media e alta. Nel loro complesso questi due fattori definiscono 12 classi di servizio. La Figura 5.40 mostra un modo in cui i pacchetti potrebbero essere elaborati nello schema di inoltro sicuro. Durante la prima fase, i pacchetti sono classificati in una delle quattro classi di priorità. Questa operazione potrebbe essere fatta sull'host trasmittente (come mostrato nella figura) oppure nel router di ingresso (cioè il primo). Se la classificazione è eseguita nell'host trasmittente si ha il vantaggio di rendere disponibili più informazioni relative ai flussi di appartenenza dei pacchetti.

Durante la seconda fase i pacchetti sono marcati in base alla classe di appartenenza. Per applicare il contrassegno è necessario utilizzare un campo di intestazione. Per fortuna è disponibile

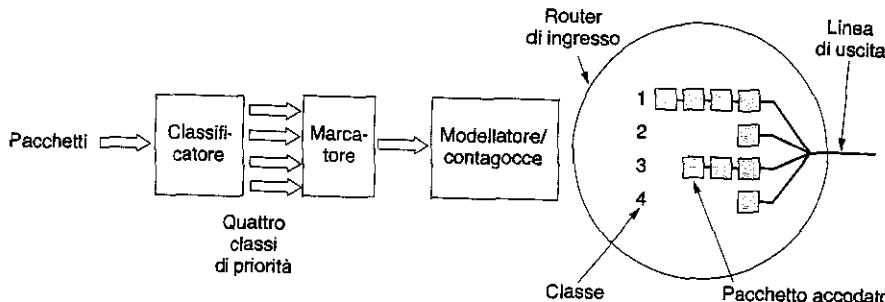


Figura 5.40. Una possibile implementazione del flusso di dati per un invio sicuro.

le un campo *tipo di servizio* lungo 8 bit nell'intestazione IP, come verrà spiegato più avanti. Il documento RFC 2597 specifica che solo sei di questi bit devono essere utilizzati per la classe di servizio; questo consente di avere spazio per classi di servizio storiche e per quelle future. Durante la terza fase, i pacchetti passano attraverso un filtro modellatore/contagocce che può ritardare o abbandonare alcuni pacchetti per dare una forma accettabile ai quattro flussi, per esempio mediante un leaky bucket o un token bucket. Se ci sono troppi pacchetti, alcuni possono essere scartati in base alla categoria di eliminazione. È possibile implementare anche schemi più elaborati, che coinvolgono misurazioni e retroazione. In questo esempio, le tre fasi sono eseguite sull'host trasmittente, perciò il flusso di output finale alimenta un router di ingresso. È utile notare che queste fasi possono essere eseguite da speciali programmi di rete o anche dal sistema operativo, per non modificare le applicazioni esistenti.

5.4.5 Label switching e MPLS

Mentre IETF stava lavorando sui servizi integrati e sui servizi differenziati, diversi produttori di router stavano studiando metodi di inoltro migliori. Questo lavoro puntava ad aggiungere davanti a ogni pacchetto un'etichetta che consentisse ai router di instradare i dati in base all'etichetta, invece che in base all'indirizzo di destinazione. Usando l'etichetta come un indice associato a una tabella interna, per determinare la linea di output corretta è sufficiente eseguire una ricerca in una tabella. Attraverso questa tecnica, il routing può essere fatto molto velocemente e qualunque risorsa necessaria può essere riservata lungo il percorso.

Naturalmente, etichettare i flussi in questo modo sfiora pericolosamente i circuiti virtuali. Anche X.25, ATM, frame relay e tutte le altre reti con una sottorete basata su circuiti virtuali applicano un'etichetta (ossia l'identificatore del circuito virtuale) in ogni pacchetto, la cercano in una tabella e instradano i dati in base alla voce individuata nella tabella. Nonostante il fatto che molte persone della comunità Internet provino una forte antipatia per le reti basate sulle connessioni, sembra che l'idea continui a ritornare, questa volta per fornire un routing veloce e la qualità del servizio. Comunque, ci sono differenze fondamentali tra il modo in cui Internet gestisce la costruzione del percorso e il modo in cui funzionano le reti orientate alle connessioni, perciò la tecnica è senza alcun dubbio diversa dalla commutazione di circuito tradizionale.

Questa "nuova" idea di commutazione è nota con diversi nomi (proprietari), incluso **label switching** (commutazione di etichetta) e **tag switching**. Alla fine, IETF ha standardizzato l'idea con il nome di **MPLS** (*MultiProtocol Label Switching*) ed è questo il nome utilizzato nel corso del presente paragrafo. Lo standard è descritto nel documento RFC 3031 e in molti altri RFC.

Per inciso, alcune persone distinguono tra *routing* e *commutazione*. Il *routing* è il processo di ricerca dell'indirizzo di destinazione eseguito in una tabella, e stabilisce la destinazione dei dati; al contrario, la commutazione usa come indice della tabella di routing un'etichetta presa dal pacchetto. Queste definizioni, però, non sono per niente universali.

Il primo problema è dove inserire l'etichetta. Poiché i pacchetti IP non sono stati progettati per i circuiti virtuali, nell'intestazione IP non ci sono campi a disposizione della numerazione dei circuiti virtuali. Per questo motivo è stato necessario aggiungere una nuova

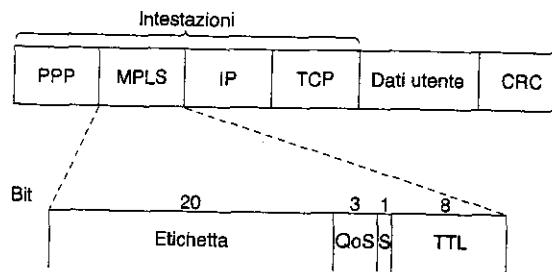


Figura 5.41. Trasmissione di un segmento TCP usando IP, MPLS e PPP.

intestazione MPLS di fronte all'intestazione IP. Su una linea tra due router che usa PPP come protocollo di costruzione dei frame, il formato del frame è quello indicato nella Figura 5.41, che mostra le intestazioni PPP, MPLS, IP e TCP. In un certo senso, MPLS può essere considerato come lo strato 2,5.

L'intestazione MPLS generica ha quattro campi, il più importante dei quali è il campo *label* (etichetta), che contiene l'indice. Il campo *QoS* indica la classe di servizio. Il campo *S* si riferisce all'accatastamento di più etichette in reti gerarchiche (caso descritto più avanti); se il suo valore è 0, il pacchetto è scartato. Questa funzionalità impedisce la formazione di cicli infiniti nei casi di instabilità di routing.

Poiché le intestazioni MPLS non fanno parte del pacchetto dello strato network o del frame dello strato data link, MPLS è in larga misura indipendente da entrambi gli strati. Tra le altre cose, questa proprietà significa che è possibile costruire switch MPLS in grado di inoltrare sia pacchetti IP sia celle ATM, a seconda di ciò che si presenta. Da questa funzionalità deriva il termine "multiprotocollo" che appare nel nome di MPLS.

Quando un pacchetto (o una cella) potenziata da MPLS raggiunge un router compatibile, l'etichetta è utilizzata come indice in una tabella per determinare la linea di trasmissione e la nuova etichetta da utilizzare. Questo scambio di etichette è utilizzato in tutti le sottoreti a circuito virtuale, perché le etichette hanno un significato logico soltanto locale. Due router possono inviare per la trasmissione sulla stessa linea di uscita dei pacchetti senza rapporto tra loro, che però usano la stessa etichetta. Per essere distinguibili all'altro capo della linea, le etichette devono essere rimappate a ogni salto; il meccanismo è già stato visto in azione nella Figura 5.3, e MPLS utilizza la stessa tecnica.

Una differenza rispetto ai circuiti virtuali tradizionali è rappresentata dal livello di aggregazione. È senz'altro possibile che ogni flusso abbia il proprio gruppo di etichette per tutta la sottorete, ma è più comune che i router raggruppino più flussi che terminano in un particolare router o LAN e utilizzino una singola etichetta collettiva. I flussi che sono raggruppati insieme sotto una singola etichetta sono detti appartenenti alla stessa FEC (*Forwarding Equivalence Class*). Questa classe copre non solo la destinazione dei pacchetti, ma anche la loro classe di servizio (nel senso dei servizi differenziati) perché tutti i pacchetti di un flusso sono trattati nello stesso modo durante l'inoltro.

Con il routing basato sui circuiti virtuali non è possibile raggruppare sotto lo stesso identificatore di circuito virtuale più percorsi distinti con punti finali diversi, perché non esiste un

modo per distinguere l'uno dall'altro una volta arrivati alla destinazione finale. Con MPLS, i pacchetti contengono ancora il loro indirizzo di destinazione finale in aggiunta all'etichetta, perciò alla fine del percorso etichettato è possibile rimuovere l'intestazione di etichetta e continuare l'inoltro nel solito modo, usando l'indirizzo di destinazione dello strato network. Una grande differenza tra MPLS e le architetture a circuito virtuale convenzionali è rappresentata dal modo in cui è costruita la tabella di inoltro. Nelle reti a circuito virtuale tradizionali, quando un utente vuole stabilire una connessione viene immesso nella rete un pacchetto d'impostazione per creare il percorso e aggiungere le voci nelle tabelle di inoltro. MPLS non funziona in questo modo, perché non c'è alcuna fase d'impostazione per ogni connessione (in quanto la sua implementazione richiederebbe la modifica di gran parte del software esistente su Internet).

Le voci delle tabelle d'inoltro vengono create in altri due modi. Nell'approccio **data-driven** (basato sui dati), il primo router attraversato dal pacchetto contatta il router a valle lungo il percorso del pacchetto, e gli chiede di generare un'etichetta per il flusso. Questo metodo è applicato in modo ricorsivo. A tutti gli effetti, si tratta di un meccanismo di creazione di circuito virtuale a richiesta.

I protocolli che gestiscono questa propagazione fanno molta attenzione a evitare i cicli. Spesso usano una tecnica chiamata tecnica dei **fili colorati**. La propagazione all'indietro di una FEC può essere paragonata al traino nella sottorete di un filo colorato in modo univoco. Se un router scorge un colore che ha già, sa che c'è un ciclo ed esegue un'azione correttiva. L'approccio basato sui dati è utilizzato principalmente sulle reti in cui il trasporto sottostante è ATM (per esempio, nel sistema telefonico).

L'altro modo, adottato sulle reti non basate su ATM, è l'approccio **control-driven** (guidato dal controllo). Questa soluzione ha molte varianti, e una di queste funziona come descritto di seguito. Quando viene avviato, il router cerca di capire quali sono i percorsi per i quali esso rappresenta la destinazione finale (per esempio, determina gli host presenti sulla sua LAN); quindi crea una o più FEC per queste destinazioni, assegna un'etichetta a ognuna di esse e le passa ai suoi vicini. Questi, in cambio, inseriscono le etichette nelle loro tabelle di inoltro e inviano nuove etichette ai loro vicini, fino a quando tutti i router non hanno acquisito il percorso. Anche le risorse possono essere riservate durante la costruzione del percorso, per garantire una qualità di servizio appropriata.

MPLS può operare su più livelli contemporaneamente. Al livello più alto la rete di ciascun operatore di telecomunicazioni è considerata come una specie di metarouter, e il percorso dalla sorgente alla destinazione attraversa i metarouter secondo necessità: questo percorso può utilizzare MPLS. MPLS può anche essere utilizzato dentro la rete di ogni operatore di telecomunicazioni, generando un secondo livello di etichette; nella realtà un pacchetto può contenere un'intera pila di etichette. Il bit *S* mostrato nella Figura 5.41 permette a un router che rimuove un'etichetta di sapere se dietro ci sono altre etichette. Il suo valore è 1 per l'etichetta posta più in basso e 0 per tutte le altre etichette. In pratica, questo meccanismo è utilizzato prevalentemente per implementare reti a circuito virtuale e tunnel ricorsivi.

Sebbene l'idea alla base di MPLS sia semplice, i dettagli sono estremamente complicati a causa di molte variazioni e ottimizzazioni. Per maggiori informazioni si può consultare (Davie and Rekhter, 2000; Lin et al., 2002; Pepelnjak and Guichard, 2001; e Wang, 2001).

5.5 Collegamento tra reti

Fino a questo momento è stata implicitamente presupposta l'esistenza di una singola rete omogenea in cui ciascuna macchina utilizza lo stesso protocollo in ogni strato. Sfortunatamente questa premessa è esageratamente ottimistica. Esistono molte reti diverse, tra cui LAN, MAN e WAN; inoltre esistono numerosi protocolli utilizzati comunemente in ogni strato. I paragrafi seguenti esaminano con attenzione i problemi che sorgono quando si collegano tra loro due o più reti per formare una **internet**.

In molti si domandano se l'attuale abbondanza di tipi di rete è una condizione temporanea che sparirà non appena tutti si renderanno conto di quanto è meravigliosa la rete [scrivere qui il nome della propria rete preferita] o se è un'inevitabile ma permanente caratteristica del mondo che rimarrà così per sempre. L'esistenza di reti diverse comporta inevitabilmente l'esistenza di protocolli diversi.

È probabile che una varietà di reti (e di protocolli) differenti ci sarà sempre, per i seguenti motivi. Prima di tutto, la base installata delle diverse reti è grande. Quasi tutti i personal computer usano TCP/IP. Molte grandi aziende hanno mainframe che adottano SNA di IBM. Un numero considerevole di aziende telefoniche adoperano reti ATM. Alcune LAN di personal computer usano ancora NCP/IPX di Novell o AppleTalk. Infine, il wireless è un'area in evoluzione che usa una grande varietà di protocolli. Questa tendenza continuerà per anni a causa dei problemi di compatibilità, delle nuove tecnologie e per il fatto che non tutti i produttori pensano che sia vantaggioso per i loro interessi aiutare i clienti a migrare al sistema di un altro produttore.

In secondo luogo, a mano a mano che i computer e le reti diventeranno sempre più economiche, le decisioni saranno prese sempre più in basso nelle organizzazioni. In molte aziende, gli investimenti che superano il milione di euro devono essere approvati dal vertice aziendale, quelli sopra i 100.000 euro devono essere approvati dai quadri intermedi, ma quelli sotto i 100.000 euro possono essere presi dai capi dipartimento e non richiedono alcuna approvazione superiore. Questo può facilmente portare all'installazione di stazioni di lavoro UNIX che usano TCP/IP nel reparto di ingegneria e all'installazione di computer Macintosh basati su AppleTalk nel reparto commerciale.

Terzo, reti differenti (per esempio ATM e wireless) adottano tecnologie radicalmente diverse, perciò non deve sorprendere il fatto che con lo sviluppo di nuovi componenti hardware verrà creato nuovo software adatto a quelle apparecchiature. Per esempio, la tipica casa di oggi assomiglia molto all'ufficio di dieci anni fa: è piena di computer che non parlano l'uno con l'altro. In futuro potrebbe diventare un fatto normale avere il telefono, il televisore e le altre apparecchiature domestiche collegate in rete tra loro e controllabili da remoto. Questa nuova tecnologia farà sicuramente nascere nuove reti e nuovi protocolli.

Per vedere un esempio di come si possono collegare tra loro reti diverse, si consideri lo schema mostrato nella Figura 5.42. Il disegno mostra una rete aziendale composta da tante sedi collegate tra loro mediante una rete geografica ATM. In uno di questi luoghi, una dor-

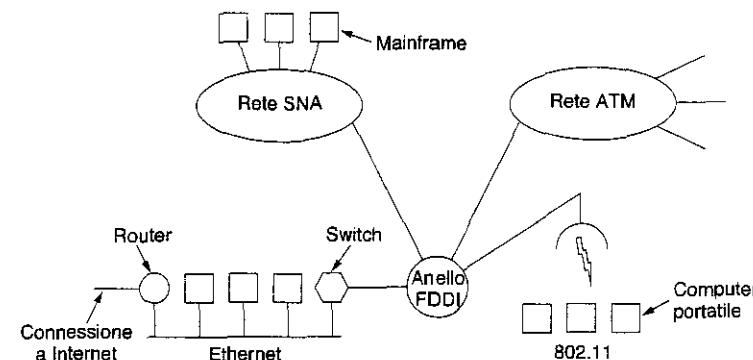


Figura 5.42. Un insieme di reti connesse.

sale ottica FDDI è utilizzata per collegare una LAN Ethernet, una rete wireless 802.11 e la rete mainframe SNA del centro di calcolo dell'azienda.

L'obiettivo della interconnessione tra tutte queste reti è consentire agli utenti di ogni rete di comunicare con gli utenti di tutte le altre, e anche permettere agli utenti di ciascuna di accedere ai dati delle altre. Per realizzare questo obiettivo è necessario l'invio di pacchetti da una rete all'altra; ma poiché le reti hanno spesso caratteristiche diverse, riuscire a trasportare i pacchetti da una all'altra non è sempre facile, come vedremo.

5.5.1 Differenze tra le reti

Le reti possono differire in molti modi. Alcune di queste differenze, come le diverse tecniche di modulazione o i formati dei frame, si trovano nello strato fisico e in quello di collegamento dati, e non saranno affrontate in questo capitolo. Nella Figura 5.43 vediamo invece alcune delle differenze che possono presentarsi nello strato network: sono proprio queste che rendono la comunicazione che avviene tra reti diverse più difficile di quella all'interno di una singola rete.

Quando i pacchetti inviati da una sorgente su una rete devono attraversare una o più reti straniere prima di raggiungere la rete di destinazione (che a sua volta può essere diversa dalla rete di origine), possono presentarsi molti problemi sulle interfacce che collegano le diverse reti. Prima di tutto, se i pacchetti provenienti da una rete orientata alle connessioni devono attraversare una rete senza connessione, può essere necessario riordinare i dati, cosa che il trasmettitore non si aspetta e che il ricevente non è preparato ad affrontare. Spesso sono richieste conversioni di protocollo, che possono risultare difficili se la funzionalità richiesta non può essere espressa. È necessario anche convertire gli indirizzi, operazione che può aver bisogno di qualche tipo di infrastruttura per gestire gli elenchi. Il passaggio di pacchetti multicast attraverso una rete che non supporta la trasmissione multicast richiede la creazione di pacchetti separati per ogni destinazione. La dimensione massima dei pacchetti cambia da un tipo di rete all'altro, e ciò può rap-

Elemento	Alcune possibilità
Servizio offerto	Orientato alle connessioni oppure no
Protocolli	IP, IPX, SNA, ATM, MPLS, AppleTalk e così via
Indirizzamento	Piatto (802) o gerarchico (IP)
Trasmissione multicast	Presente o assente (vale anche per la modalità broadcast)
Dimensione del pacchetto	Ogni rete supporta una dimensione massima particolare
Qualità del servizio	Presente o assente; molti tipi diversi
Gestione degli errori	Inoltro affidabile, ordinato o non ordinato
Controllo di flusso	Sliding window, controllo della velocità, altro o nessuno
Controllo della congestione	Secchio bucato, secchio a token, RED, pacchetti di interruzione e così via
Sicurezza	Regole di riservatezza, codifica dei dati e così via
Parametri	Scadenze diverse, specifiche di flusso e così via
Conteggio dei costi	In base al tempo di connessione, per pacchetto, per byte, nessuna

Figura 5.43. Alcune delle caratteristiche che rendono le reti diverse tra loro.

presentare una grossa seccatura. In che modo si può passare un pacchetto grande 8.000 byte attraverso una rete che supporta pacchetti grandi al massimo 1.500 byte?

Anche le diverse qualità di servizio rappresentano un problema, se un pacchetto che deve essere inoltrato in tempo reale passa attraverso una rete che non offre alcuna garanzia per la trasmissione in questa modalità.

Il controllo degli errori, del flusso e delle congestioni spesso cambiano in base alla rete. Quando sorgente e destinazione si aspettano una trasmissione in sequenza e senza errori di tutti i pacchetti, ma la rete intermedia scarta i pacchetti ogni volta che scorge una congestione all'orizzonte, molte applicazioni possono smettere di funzionare. Inoltre, si possono presentare diversi inconvenienti quando non ci si aspetta che i pacchetti possano andare a zonzo senza meta per qualche tempo, prima di emergere improvvisamente ed essere consegnati. Altri potenziali problemi nascono dai diversi meccanismi di protezione, dalle impostazioni dei parametri, dalle regole per il conteggio dei costi e persino dalle leggi sulla privacy.

5.5.2 Connessione tra le reti

Le reti possono essere collegate tra loro attraverso dispositivi di tipo diverso, come è stato spiegato nel Capitolo 4, perciò è opportuno ripetere i concetti principali. Nello strato fisico le reti possono essere connesse mediante ripetitori o hub, cioè apparecchi che spostano i bit da una rete a un'altra rete identica, senza modifiche. Si tratta per lo più di dispositivi analogici che non sanno nulla dei protocolli digitali, ma si limitano a rigenerare il segnale trasmesso.

Nello strato data link sovrastante operano bridge e switch. Questi apparecchi possono accettare frame, esaminare gli indirizzi MAC e inoltrare i frame su reti diverse eseguendo nel contempo una limitata conversione del protocollo, per esempio da Ethernet a FDDI oppure 802.11.

Nello strato network, due reti possono essere collegate mediante router. Se i loro strati di rete sono differenti, il router può essere capace di tradurre i formati dei pacchetti, sebbene la traduzione del pacchetto oggi sia sempre più rara. Un router che può gestire più protocolli è chiamato **router multiprotocollo**.

Nello strato trasporto si trovano i gateway, che possono interfacciare due connessioni di trasporto. Per esempio, un gateway di trasporto potrebbe consentire ai pacchetti di scorrere tra una rete TCP e una rete SNA, che usa un diverso protocollo di trasporto, essenzialmente incollando una connessione TCP a una connessione SNA.

Infine, nello strato di applicazione, i gateway applicativi traducono la semantica dei messaggi. Come esempio, i gateway tra la posta elettronica di Internet (RFC 822) e la posta elettronica X.400 devono analizzare i messaggi di posta elettronica e modificare alcuni campi di intestazione.

Questo capitolo si concentra soprattutto sulla connessione tra reti eseguita sullo strato network. La Figura 5.44 evidenzia le differenze tra questa situazione e la commutazione eseguita nello strato data link. Nella Figura 5.44(a), il computer sorgente *S* vuole inviare un pacchetto al computer di destinazione *D*; i due si trovano su Ethernet diverse, collegate da uno switch. *S* incapsula il pacchetto in un frame e lo invia per la sua strada. Il frame raggiunge lo switch, che determina la destinazione dei dati (LAN 2) esaminando l'indirizzo MAC. Alla fine, il dispositivo rimuove semplicemente il frame dalla LAN 1 e lo deposita nella LAN 2.

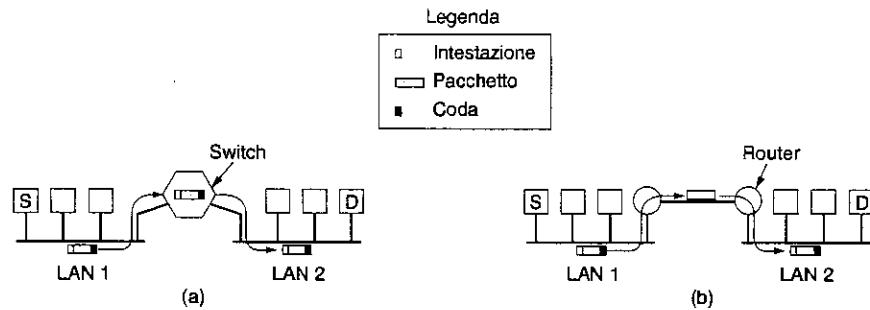


Figura 5.44. (a) Due Ethernet collegate da uno switch. (b) Due Ethernet collegate tramite router.

Ora si consideri la stessa situazione con le due Ethernet collegate mediante una coppia di router, invece che da uno switch. I router sono collegati tra loro da una linea punto-punto, per esempio una linea affittata lunga migliaia di chilometri. Il frame è raccolto dal primo router e il pacchetto è rimosso dal suo campo dati. Il router esamina l'indirizzo nel pacchetto (per esempio un indirizzo IP) e lo cerca nella sua tabella di routing. In base all'indirizzo il dispositivo decide di inviare il pacchetto al router remoto, magari dopo averlo incapsulato in un diverso tipo di frame, secondo il protocollo di linea. Arrivato al secondo router, il pacchetto è inserito nel campo dati di un frame Ethernet e depositato nella LAN 2.

Una differenza sostanziale tra il caso basato sullo switch e il caso basato sul router è questa: con uno switch (o un bridge), si trasporta l'intero frame, sulla base del suo indirizzo

MAC; con un router, il pacchetto è estratto dal frame e l'indirizzo nel pacchetto è utilizzato per decidere dove inviarlo. Gli switch non devono comprendere il protocollo dello strato network utilizzato per commutare i pacchetti, i router invece sì.

5.5.3 Circuiti virtuali concatenati

Sono possibili due generi di connessione: il primo usa una concatenazione (orientata alle connessioni) di sottoreti a circuito virtuale, l'altro si basa sui datagrammi. Prima di esaminare i dettagli, un avvertimento: in passato la maggior parte delle reti (pubbliche) era orientata alle connessioni (frame relay, SNA, 802.16 e ATM ancora lo sono); successivamente, con il rapido sviluppo di Internet, i datagrammi sono diventati la soluzione più utilizzata. Sarebbe un errore pensare che i datagrammi verranno utilizzati per sempre: in questo campo l'unica cosa eterna è il cambiamento. L'importanza crescente delle reti multimediali rende probabile un ritorno delle soluzioni orientate alle connessioni, in una forma o in un'altra, poiché con le connessioni è più facile garantire la qualità del servizio; per questo motivo il paragrafo dedica un po' di spazio anche alle reti orientate alle connessioni.

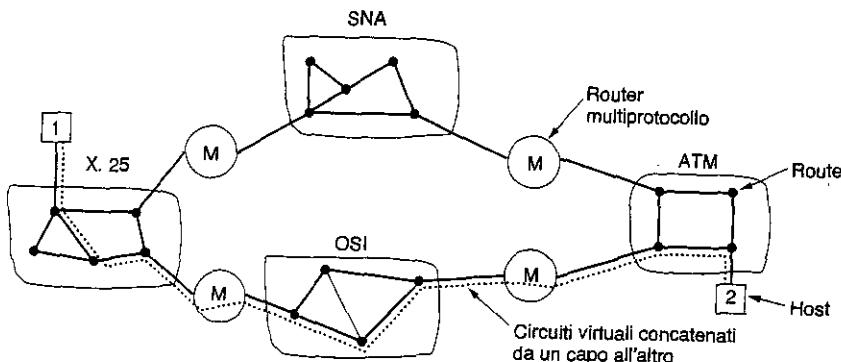


Figura 5.45. Collegamento tra reti basato su circuiti virtuali concatenati.

Nel modello a circuiti virtuali concatenati mostrato nella Figura 5.45, viene impostata una connessione all'host di una rete lontana, seguendo le consuete modalità. La sottrete si accorge che la destinazione è remota e costruisce un circuito virtuale diretto al router più vicino alla rete di destinazione; quindi crea un circuito virtuale da quel router a un gateway esterno (router multiprotocollo). Questo gateway registra l'esistenza del circuito virtuale nelle sue tabelle e procede costruendo un altro circuito virtuale diretto a un router nella sottrete successiva. Questo processo continua fino a quando non viene raggiunto l'host di destinazione.

Una volta che i pacchetti iniziano a scorrere lungo il percorso, ogni gateway inoltra i pacchetti in arrivo, convertendo i formati dei pacchetti e i numeri di circuito virtuale in base alle necessità. Chiaramente, tutti i pacchetti di dati devono attraversare la stessa sequenza di gateway. Di conseguenza, i pacchetti di un flusso non sono mai riordinati dalla rete.

La caratteristica essenziale di questo approccio è che una sequenza di circuiti virtuali è impostata dalla sorgente, attraverso uno o più gateway, fino alla destinazione. Ogni gateway gestisce tabelle che indicano i circuiti virtuali che passano attraverso il dispositivo, le destinazioni degli instradamenti e i numeri dei nuovi circuiti virtuali.

Questo schema funziona meglio quando tutte le reti hanno all'incirca le stesse proprietà. Se, per esempio, tutte le reti garantiscono una trasmissione affidabile dei pacchetti dello strato network, sarà affidabile anche il flusso dalla sorgente alla destinazione (fatta eccezione per eventuali errori e interruzioni lungo il percorso). In modo analogo, se nessuna delle reti garantisce una trasmissione affidabile, allora anche la concatenazione dei circuiti virtuali non è affidabile. D'altro canto, se la macchina sorgente si trova su una rete che garantisce la trasmissione affidabile, ma una delle reti intermedie può perdere i pacchetti, la concatenazione pesantemente modifica la natura del servizio.

I circuiti virtuali concatenati sono comuni anche nello strato trasporto. In particolare, è possibile costruire un canale di bit usando (per esempio) SNA che termina in un gateway, e una connessione TCP che va da tale gateway al successivo. Operando in questo modo si può realizzare un circuito virtuale punto-punto che attraversa reti e protocolli diversi.

5.5.4 Collegamento tra reti senza connessione

Il modello alternativo di connessione tra reti è basato sui datagrammi, ed è illustrato nella Figura 5.46. In questo modello l'unico servizio che lo strato network offre allo strato trasporto è la capacità di introdurre datagrammi nella sottrete, sperando per il meglio. Non c'è alcuna nozione di circuito virtuale nello strato network, e tanto meno di una concatenazione di circuiti. Questo modello non richiede che tutti i pacchetti che appartengono a una connessione debbano attraversare la stessa sequenza di gateway. Nella Figura 5.46, i datagrammi provenienti dall'host 1 e diretti all'host 2 seguono percorsi diversi attraverso la internetwork. Una decisione di routing è presa separatamente per ogni pacchetto, magari in base al traffico al momento della trasmissione del pacchetto. Questa strategia può utilizzare più percorsi e perciò può ottenere una banda maggiore rispetto al modello basato sui circuiti virtuali concatenati, ma d'altra parte non c'è alcuna garanzia che i pacchetti raggiungano la destinazione in ordine. Addirittura, potrebbero non arrivare del tutto.

Il modello mostrato nella Figura 5.46 non è semplice come sembra. Tanto per cominciare, se ogni rete usa un protocollo di strato network diverso, un pacchetto proveniente da una rete non può passare in un'altra rete. Si potrebbe credere che i router multiprotocollo tentino veramente di tradurre un formato nell'altro, ma a meno che i due formati non siano molto simili e non utilizzino gli stessi campi di informazioni, queste conversioni saranno sempre incomplete e spesso destinate al fallimento. Per questo motivo raramente si tenta di eseguire una conversione.

Un secondo problema, più serio, riguarda l'indirizzamento. Si immagini un caso semplice: un host di Internet tenta di inviare un pacchetto IP a un host che si trova su una rete SNA limitrofa. Gli indirizzi IP e SNA sono diversi. Sarebbe necessaria una mappatura tra indirizzi IP e SNA in entrambe le direzioni.

Per di più cambia il concetto di ciò che è indirizzabile. In IP, gli host (in realtà le schede

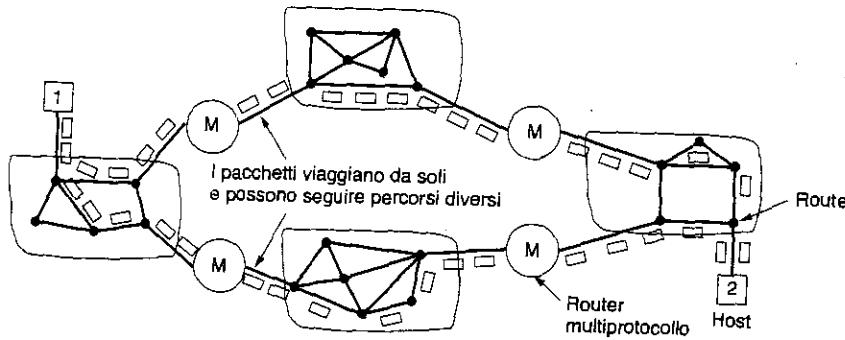


Figura 5.46. Una internet ad assenza di connessione.

di rete) hanno indirizzi; in SNA possono avere indirizzi anche entità diverse dagli host (ossia dispositivi di rete). Nella migliore delle ipotesi, qualcuno dovrebbe gestire un database che tenga traccia di ogni possibile associazione, ma questa soluzione rappresenterebbe una costante fonte di problemi.

Un'altra idea è quella di progettare un pacchetto "internet" universale facendo in modo che tutti i router lo riconoscano. Questo è, in effetti, l'approccio di IP: un pacchetto progettato per essere trasportato attraverso molte reti. Naturalmente, può accadere che IPv4 (il protocollo attuale di Internet) faccia sparire dal mercato tutti gli altri formati, che IPv6 (il futuro protocollo di Internet) non si diffonda e che nulla di nuovo venga mai più inventato, ma la storia suggerisce altrimenti. Far accettare a tutti un singolo formato è difficile, se le aziende pensano che avere un formato proprietario garantisce un vantaggio commerciale.

È giunto il momento di ricapitolare brevemente i due modi in cui può essere gestita la comunicazione tra reti. Il modello a circuiti virtuali concatenati offre essenzialmente gli stessi vantaggi offerti dai circuiti virtuali usati in una singola sottorete: permette di riservare in anticipo i buffer, garantisce l'ordine dei pacchetti, permette di utilizzare intestazioni corte ed evita i problemi causati dai pacchetti duplicati trasmessi in ritardo.

Ma questa soluzione ha anche gli stessi svantaggi: ogni connessione aperta occupa spazio nelle tabelle dei router, non è possibile utilizzare percorsi alternativi per evitare le zone congestionate, e il sistema è vulnerabile ai guasti dei router che si trovano sul percorso. Un altro svantaggio è rappresentato dalla difficoltà, se non addirittura dall'impossibilità, di implementare questo metodo se una delle reti coinvolte è una rete a datagrammi inaffidabile.

Le proprietà dell'approccio basato sui datagrammi sono più o meno le stesse di quelle delle sottoreti a datagrammi: maggiore probabilità di congestioni ma anche più strumenti per aggirarle, robustezza nonostante i guasti dei router e necessità di intestazioni più lunghe. In una Internet si possono adottare diversi algoritmi di routing adattivi, proprio come nelle singole reti a datagramma.

Un grande vantaggio dell'approccio basato sui datagrammi è che può essere utilizzato su

sottoreti che non usano al loro interno circuiti virtuali. Molte LAN, reti mobili (per esempio flotte aeree e navali) e anche alcune WAN rientrano in questa categoria. Quando una Internet include una di queste sottoreti, se si adotta una strategia di comunicazione tra reti basata sui circuiti virtuali si presentano seri problemi.

5.5.5 Tunneling

Gestire il caso generale di una connessione tra due reti diverse è estremamente difficile. Comunque esiste un caso particolare, molto frequente, che fortunatamente è gestibile: un host sorgente e un host di destinazione che si trovano sullo stesso tipo di rete, ma che sono separati da una rete differente. Come esempio, si consideri una banca internazionale: una filiale a Parigi usa una rete Ethernet basata su TCP/IP, una sede a Londra usa una rete Ethernet basata su TCP/IP, e le due reti sono separate da una rete geografica non IP (per esempio, ATM) come mostrato nella Figura 5.47.

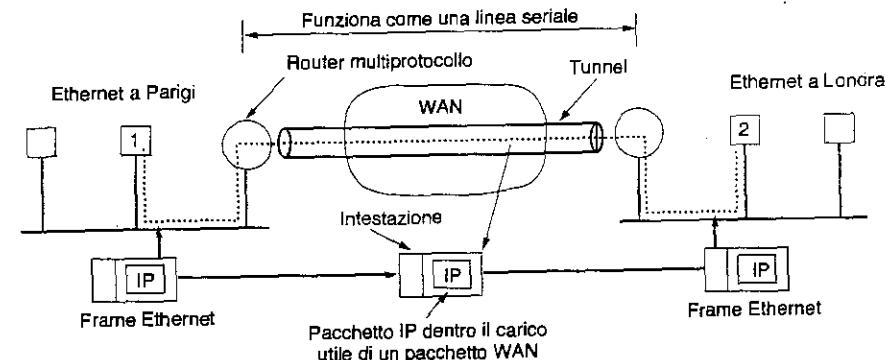


Figura 5.47. Tunneling di un pacchetto da Parigi a Londra.

Per risolvere questo problema si adotta una tecnica chiamata **tunneling**. Per inviare un pacchetto IP all'host 2, l'host 1 costruisce il pacchetto contenente l'indirizzo IP dell'host 2, inserisce il tutto in un frame Ethernet indirizzato al router multiprotocollo di Parigi e immette i dati nella Ethernet. Quando riceve il frame, il router multiprotocollo rimuove il pacchetto IP, lo inserisce nel carico utile di un pacchetto dello strato network della WAN e indirizza quest'ultimo all'indirizzo WAN del router multiprotocollo di Londra. Quando arriva a destinazione, il router di Londra rimuove il pacchetto IP e lo trasmette all'host 2 in un frame Ethernet.

La WAN può essere vista come un grande tunnel che si estende da un router multiprotocollo all'altro. Il pacchetto IP viaggia da un'estremità del tunnel all'altra, rannicchiato nella sua bella scatolina. Né lui né gli host sulle due Ethernet si devono preoccupare delle operazioni eseguite nella WAN. Solo il router multiprotocollo deve comprendere IP e i pacchetti WAN; in pratica l'intero percorso compreso tra i due router funziona come una linea seriale.

Un'analogia può aiutare a comprendere meglio la funzione del tunneling. Si consideri una persona che guida un'auto da Parigi a Londra. In Francia l'auto si sposta spinta dal proprio motore ma, quando raggiunge il canale della Manica, il veicolo è caricato su un treno ad alta velocità che lo trasporta in Inghilterra. L'auto, a tutti gli effetti, è trasportata come se fosse un carico (Figura 5.48). All'altra estremità del tunnel, il treno libera il veicolo che può così spostarsi a piacimento sulle strade inglesi. Il tunneling dei pacchetti attraverso una rete di tipo diverso funziona nello stesso modo.

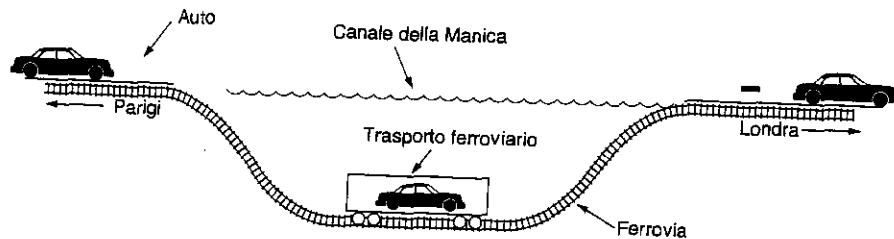


Figura 5.48. Passaggio sotterraneo di un'auto dalla Francia all'Inghilterra.

5.5.6 Routing in una internetwork

Il routing attraverso una internetwork è simile al routing eseguito in una singola sottorete, ma con alcune complicazioni in più. Si consideri, per esempio, la connessione tra reti mostrata nella Figura 5.49(a) in cui cinque reti sono collegate mediante sei router (eventualmente multiprotocollo). Fare un modello grafico di questa situazione è complicato dal fatto che ogni router può accedere direttamente (ossia può inviare i pacchetti) a ogni altro router collegato a qualunque rete al quale è connesso. Per esempio, B nella Figura 5.49(a) può accedere direttamente ad A e a C attraverso la rete 2 e a D attraverso la rete 3. Il grafo mostrato nella Figura 5.49(b) rappresenta queste connessioni.

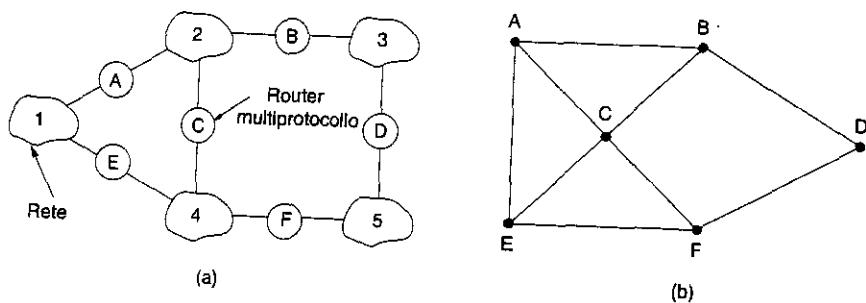


Figura 5.49. (a) Una internetwork. (b) Grafo di una internetwork.

Dopo aver costruito il grafo, si possono applicare all'insieme di router multiprotocollo gli algoritmi di routing già noti, per esempio quelli basati sullo stato dei collegamenti o sul vettore delle distanze. Si ottiene perciò un algoritmo di routing a due livelli: dentro ogni rete viene utilizzato un **protocollo di gateway interno** (*interior gateway protocol*), ma tra le reti è adottato un **protocollo di gateway esterno** (*exterior gateway protocol*); il termine gateway era storicamente usato per indicare i router. In realtà, poiché ogni rete è indipendente, ciascuna può usare algoritmi differenti. Poiché nella internetwork ogni rete è indipendente da tutte le altre, spesso le reti sono chiamate **Autonomous System** (AS).

Un classico pacchetto internet nasce nella propria LAN, dove viene indirizzato al router multiprotocollo locale (nell'intestazione dello strato MAC). Raggiunto il router, il codice dello strato network decide, usando le sue tabelle di routing, a quale router multiprotocollo deve essere inoltrato il pacchetto. Se quel router può essere raggiunto usando i protocolli di rete nativo del pacchetto, il pacchetto è inoltrato direttamente; altrimenti i dati vengono trasmessi via tunneling, ossia sono incapsulati nel protocollo richiesto dalla rete intermedia. Questo processo si ripete fino a quando il pacchetto non raggiunge la rete di destinazione.

Una delle differenze tra il routing nelle internetwork e quello interno alle reti è che il primo può richiedere l'attraversamento di confini internazionali. Improvvisamente entrano in gioco molte leggi, per esempio le severe leggi svedesi sulla privacy che impediscono di esportare dati personali che riguardano i cittadini svedesi. Un altro esempio è costituito dalla legge canadese che impedisce ai dati che hanno origine e fine in Canada di uscire dal paese; in base a questa legge, il traffico generato a Windsor, Ontario, e diretto a Vancouver non può essere instradato attraverso la vicina città di Detroit, anche se quel percorso è il più veloce e il meno costoso.

Un'altra differenza tra il routing interno e quello esterno riguarda il costo. Dentro una singola rete normalmente si applica un solo algoritmo di addebito, mentre reti diverse possono essere gestite da enti diversi e un percorso può essere meno costoso di un altro. In modo analogo, anche la qualità del servizio offerto dalle diverse reti può cambiare e da questo può dipendere la scelta di un percorso piuttosto che un altro.

5.5.7 Frammentazione

Ogni rete impone una dimensione massima ai suoi pacchetti. Questi limiti possono dipendere:

1. dall'hardware (come nel caso del frame Ethernet)
2. dal sistema operativo (per esempio, tutti i buffer sono grandi 512 byte)
3. dai protocolli (numero di bit nel campo che definisce la lunghezza del pacchetto)
4. dalla conformità con qualche standard internazionale
5. dal desiderio di ridurre di un certo livello le ritrasmissioni causate dagli errori
6. dal desiderio di impedire a un pacchetto di occupare il canale per troppo tempo.

Il risultato di tutti questi fattori è che i progettisti di rete non sono liberi di scegliere la dimensione massima di pacchetto che preferiscono. I carichi utili massimi variano da 48 byte (delle celle ATM) a 65.515 byte (dei pacchetti IP), anche se la dimensione del carico utile negli strati più alti spesso è maggiore.

Un problema ovvio si presenta quando un pacchetto grande tenta di viaggiare attraverso una rete che supporta pacchetti di piccole dimensioni. Una soluzione è evitare l'insorgere del problema a monte; in altre parole, la connessione tra reti potrebbe utilizzare un algoritmo di routing che evita di trasmettere il pacchetto attraverso reti che non sono in grado di gestirlo. Comunque, questa non è una vera soluzione. Che cosa accade, infatti, se il pacchetto originale è troppo grande per essere gestito dalla rete di destinazione? L'algoritmo di routing non può scavalcare la destinazione.

Fondamentalmente, il problema può essere risolto consentendo ai gateway di suddividere i pacchetti in **frammenti**, inviando poi ogni frammento come pacchetto internet separato. Tuttavia, come sanno tutti i genitori di figli piccoli, è molto più facile convertire un oggetto grande in piccoli frammenti che eseguire l'operazione inversa (i fisici hanno anche dato un nome a questo effetto, lo hanno chiamato seconda legge della termodinamica). Anche le reti a commutazione di pacchetto hanno problemi a rimettere insieme i frammenti.

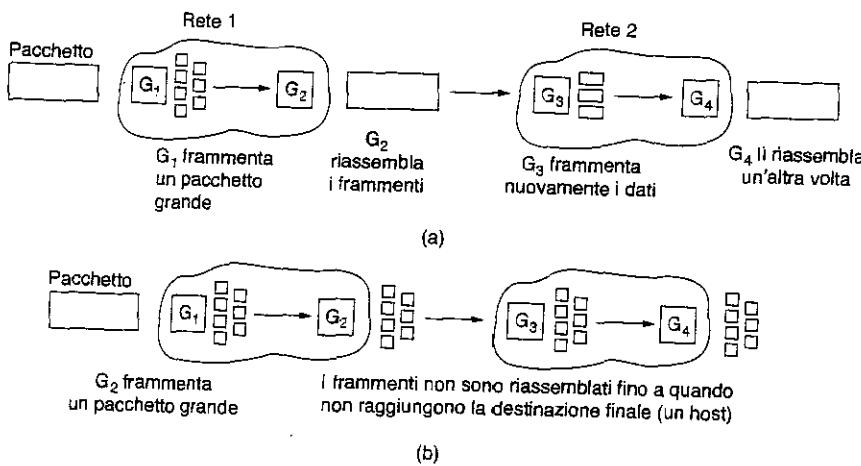


Figura 5.50. (a) Frammentazione trasparente. (b) Frammentazione non trasparente.

Si adottano due opposte strategie per ricostruire il pacchetto originale partendo dai frammenti. In base alla prima strategia, la frammentazione che è eseguita da una rete "a pacchetti piccoli" viene resa trasparente a ogni altra rete situata più a valle lungo il percorso che il pacchetto deve attraversare per raggiungere la destinazione. Questa scelta è mostrata nella Figura 5.50(a). In questo approccio, la rete a pacchetti piccoli ha gateway (o negozi, router specializzati) che si interfacciano con altre reti. Quando riceve un pacchetto troppo grande, il gateway suddivide i dati in frammenti. Ogni frammento è indirizzato allo stesso gateway di uscita, dove i pezzi sono riassemblati. In questo modo, il passaggio

attraverso la rete a pacchetti piccoli diventa trasparente. Le reti successive non si accorgono nemmeno che è avvenuta la frammentazione. Le reti ATM, per esempio, usano speciali dispositivi hardware che frammentano in modo trasparente i pacchetti in celle e poi riassegnano le celle in pacchetti. Nel mondo ATM, la frammentazione è chiamata segmentazione; il concetto è lo stesso ma alcuni dettagli cambiano.

La frammentazione trasparente è semplice ma ha alcuni problemi. Tanto per cominciare, il gateway di uscita deve sapere quando ha ricevuto tutti i pezzi, perciò è necessario fornire un campo contatore o un campo che indichi la fine del pacchetto. Poi, tutti i pacchetti devono uscire attraverso lo stesso gateway. Poiché tutti i frammenti devono seguire lo stesso percorso e quindi non possono viaggiare parallelamente su strade diverse, le prestazioni possono diminuire. Un ultimo problema è rappresentato dal tempo di elaborazione richiesto per suddividere e ricostruire ripetutamente un pacchetto grande che attraversa una serie di reti a pacchetti piccoli (ATM impone la frammentazione trasparente).

L'altra strategia di frammentazione si astiene dal ricombinare i frammenti a ogni gateway intermedio. Una volta che il pacchetto è stato suddiviso, ogni frammento è trattato come se fosse un pacchetto originale. Tutti i frammenti passano attraverso il gateway di uscita (o i gateway), come mostrato nella Figura 5.50(b), e la ricostruzione viene eseguita solo dall'host di destinazione. IP funziona in questo modo.

La frammentazione non trasparente ha alcuni problemi. Per esempio, richiede che *ogni* host sia in grado di ricostruire i pacchetti. Un altro problema si presenta quando si spezza un pacchetto molto grande: in questo caso, il tempo di elaborazione totale aumenta perché ogni frammento deve avere un'intestazione. Mentre con il primo metodo questo tempo di elaborazione si azzerà non appena si esce dalla rete a pacchetti piccoli, nel secondo metodo l'overhead rimane per tutta la durata del viaggio. Tuttavia un vantaggio della frammentazione non trasparente è che permette di utilizzare più gateway di uscita e quindi consente di ottenere prestazioni migliori. Naturalmente se si utilizza il modello a circuiti virtuali concatenati, questo vantaggio non è di alcuna utilità.

Quando il pacchetto viene diviso in frammenti, questi vanno numerati in modo che alla fine si possa ricostruire il flusso di dati originale. Per esempio, i frammenti possono essere numerati usando una struttura ad albero. Se si scomponete il pacchetto 0, ai frammenti sono assegnati i nomi 0.0, 0.1, 0.2 e così via. Quando uno di questi frammenti viene a sua volta scomposto in più parti, i nuovi frammenti vengono chiamati 0.0.0, 0.0.1, 0.0.2, ..., 0.1.0, 0.1.1, 0.1.2 e così via. Se nell'intestazione sono stati riservati campi sufficienti al caso peggiore (e se non vengono generati duplicati), questo schema assicura la possibilità di riassemblare correttamente tutti i frammenti una volta giunti a destinazione, qualunque sia il loro ordine di arrivo.

Tuttavia, se anche una sola delle reti perde o scarta pacchetti, sono necessarie ritrasmissioni end-to-end che sfortunatamente influenzano il sistema di numerazione. Si supponga che un pacchetto da 1.024 bit sia inizialmente diviso in quattro frammenti di eguale dimensione: 0.0, 0.1, 0.2 e 0.3. Il frammento 0.1 si perde, ma gli altri raggiungono la destinazione, perciò allo scadere del tempo la sorgente ritrasmette nuovamente il pacchetto originale. Questa volta però, a causa della legge di Murphy, il percorso scelto attraversa una rete che ha il limite di 512 bit, perciò vengono creati due frammenti. Quando il nuovo

frammento 0.1 raggiunge la destinazione, il ricevente pensa di disporre di tutti e quattro i pezzi e ricostruisce il pacchetto in modo errato.

Un sistema di numerazione completamente diverso (e migliore) per il protocollo di comunicazione della internetwork definisce una dimensione di frammento elementare così piccola da permettere al frammento di passare attraverso tutte le reti. Quando un pacchetto è diviso in frammenti, tutti i pezzi hanno la dimensione del frammento elementare tranne l'ultimo, che può essere più corto. Un pacchetto internet può contenere diversi frammenti, per motivi di efficienza. L'intestazione internet deve indicare il numero di pacchetto originale e il numero del (primo) frammento elementare contenuto nel pacchetto. Come al solito, ci deve essere anche un bit che indica che l'ultimo frammento elementare contenuto nel pacchetto internet è l'ultimo del pacchetto originale.

Questo approccio richiede due campi di sequenza nell'intestazione internet: il numero di pacchetto originale e il numero del frammento. C'è chiaramente un compromesso tra la dimensione del frammento elementare e il numero di bit del numero di frammento. Poiché la dimensione del frammento elementare è accettata da ogni rete, la successiva frammentazione di un pacchetto internet contenente diversi frammenti non causa alcun problema. Il limite massimo in questo caso è rappresentato da un frammento elementare grande come un singolo bit o byte, con il numero di frammento che rappresenta l'offset del bit o del byte dentro il pacchetto originale, come mostrato nella Figura 5.51.

Numero del primo frammento elementare in questo pacchetto

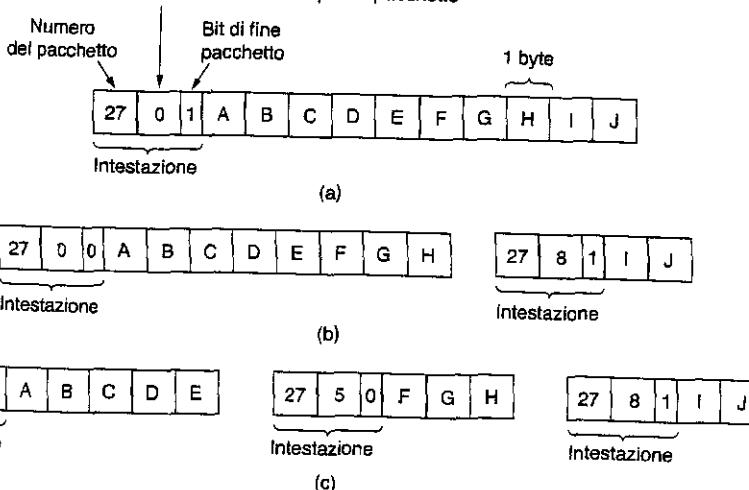


Figura 5.51. Frammentazione quando la dimensione dei dati elementari è di 1 byte.

(a) Pacchetto originale contenente 10 byte di dati. (b) Frammenti dopo il passaggio attraverso una rete che supporta una dimensione massima di pacchetto di 8 byte di carico utile, più l'intestazione. (c) Frammenti dopo il passaggio attraverso un gateway di dimensione 5.

Alcuni protocolli Internet sviluppano ulteriormente questo metodo e considerano l'intera trasmissione su un circuito virtuale come un pacchetto gigantesco, così che ogni frammento contiene il numero di byte assoluto del primo byte dentro il frammento.

5.6 Lo strato network in Internet

Prima di studiare in dettaglio lo strato network di Internet vale la pena esaminare brevemente i principi che in passato hanno guidato lo sviluppo della sua architettura e che hanno decretato il suo successo odierno: troppo spesso, di questi tempi, la gente sembra dimenticare gli esordi. Questi principi sono elencati e descritti in RFC 1958, che progettisti di protocolli dovrebbero leggere obbligatoriamente, con tanto di esame finale. Questo RFC fa largo uso delle idee riportate in (Clark, 1988; e Saltzer et al., 1984). L'elenco seguente riepiloga i dieci principi fondamentali, ordinati dal più importante al meno importante.

- 1. Assicurarsi che funzioni.** Non ultimare il progetto o lo standard fino a quando diversi prototipi non hanno comunicato con successo gli uni con gli altri. Troppo spesso i progettisti scrivono uno standard di 1.000 pagine, lo fanno approvare e poi, dopo aver scoperto che non funziona a causa di difetti profondi, ne scrivono la versione 1.1. Questo non è il modo corretto di procedere.
- 2. Mantenerlo semplice.** Nel dubbio, si utilizzi la soluzione più semplice. William di Occam ha formulato questo principio (detto rasoio di Occam) nel quattordicesimo secolo. Tradotto in termini moderni: lotta contro le funzionalità. Le funzionalità non strettamente necessarie vanno eliminate, specialmente se lo stesso effetto può essere ottenuto combinando altre caratteristiche.
- 3. Fare scelte chiare.** Se la stessa cosa può essere fatta in modi diversi, sceglierne uno. Fare la stessa cosa in due o più modi significa cercare guai. Gli standard spesso hanno più opzioni o modalità o parametri perché membri influenti insistono nel dire che il loro è il sistema migliore. I progettisti dovrebbero resistere accanitamente a questa ridondanza. Basta dire di no.
- 4. Sfruttare la modularità.** Questo principio conduce direttamente all'idea di avere pile di protocolli con ciascuno degli strati indipendente da tutti gli altri. In tal modo, se le circostanze richiedono la modifica di un modulo o di uno strato, l'intervento apportato a un elemento non modificherà gli altri elementi.
- 5. Aspettarsi l'eterogeneità.** In qualsiasi rete di grandi dimensioni si usano diversi tipi di hardware, di attrezzature di trasmissione e di applicazioni. Per gestire tutti questi elementi, l'architettura di rete deve essere semplice, generale e flessibile.
- 6. Evitare opzioni e parametri statici.** Se i parametri sono inevitabili (come nel caso della dimensione massima del pacchetto), invece di definire scelte fisse è meglio fare in modo che il trasmettitore e il ricevente concordino un valore.

7. **Mirare a un buon progetto; non è necessario che sia anche perfetto.** Spesso i progettisti hanno un buon progetto che però non riesce a gestire alcuni misteriosi casi speciali. Piuttosto che mettere da parte il progetto, i progettisti dovrebbero procedere con la loro idea e passare il fardello delle modifiche a quelli che hanno i requisiti strani.
8. **Essere rigorosi nella trasmissione e tolleranti nella ricezione.** In altre parole, inviare solo pacchetti che si attengono rigorosamente agli standard, ma aspettarsi che i pacchetti in arrivo non siano completamente compatibili, tentando in ogni caso di gestirli.
9. **Pensare alla scalabilità.** Se il sistema deve gestire milioni di host e miliardi di utenti in modo efficace, non è tollerabile alcun tipo di database centralizzato e il carico deve essere il più possibile distribuito tra tutte le risorse disponibili.
10. **Considerare le prestazioni e i costi.** Se una rete ha scarse prestazioni o costi oltraggiosi, nessuno la utilizzerà.

È giunto il momento di abbandonare i principi generali e iniziare a esaminare i dettagli dello strato network di Internet. Nello strato network, la rete di Internet può essere vista come un insieme di sottoreti o di **Autonomous Systems (AS)** interconnessi. Non esiste alcuna vera struttura, esistono solo molte dorsali principali formate da linee a banda larga e router veloci. Alle dorsali si collegano le reti regionali (di livello intermedio), a cui si collegano le LAN di molte università, aziende e fornitori di servizi Internet. La Figura 5.52 mostra uno schema di questa organizzazione quasi gerarchica.

La colla che tiene unito Internet è il protocollo dello strato network, **IP (Internet Protocol)**. A differenza della maggior parte dei protocolli di strato network più vecchi, IP è stato progettato fin dall'inizio pensando alla comunicazione tra reti. Un bel modo di immaginare lo strato network è questo: il suo compito è fornire un mezzo di tipo best effort (perciò non garantito) per trasportare i datagrammi inviati da una sorgente a una destinazione, senza tener conto della posizione delle macchine (che possono trovarsi su reti diverse) e della eventuale presenza di reti intermedie (poste tra le macchine).

La comunicazione in Internet funziona in questo modo. Lo strato trasporto prende i flussi di dati e li divide in datagrammi. In teoria ogni datagramma può essere grande 64 KB, ma all'atto pratico la dimensione dei datagrammi di solito non supera i 1.500 byte (quindi coincide con quella del frame Ethernet). Ogni datagramma è trasmesso attraverso Internet, e magari viene frammentato in unità più piccole. Quando alla fine tutti i pezzi raggiungono la macchina di destinazione, lo strato network ricostruisce il datagramma originale, che sarà passato allo strato trasporto che lo inserisce nel flusso di input del processo ricevente. La Figura 5.52 mostra un pacchetto generato dall'host 1 che deve attraversare sei reti per raggiungere l'host 2. In realtà, spesso il numero di reti attraversate è maggiore di sei.

5.6.1 Il protocollo IP

Il punto giusto per iniziare lo studio dello strato network in Internet è l'esame del formato dei datagrammi IP. Un datagramma IP è costituito da una parte di intestazione e

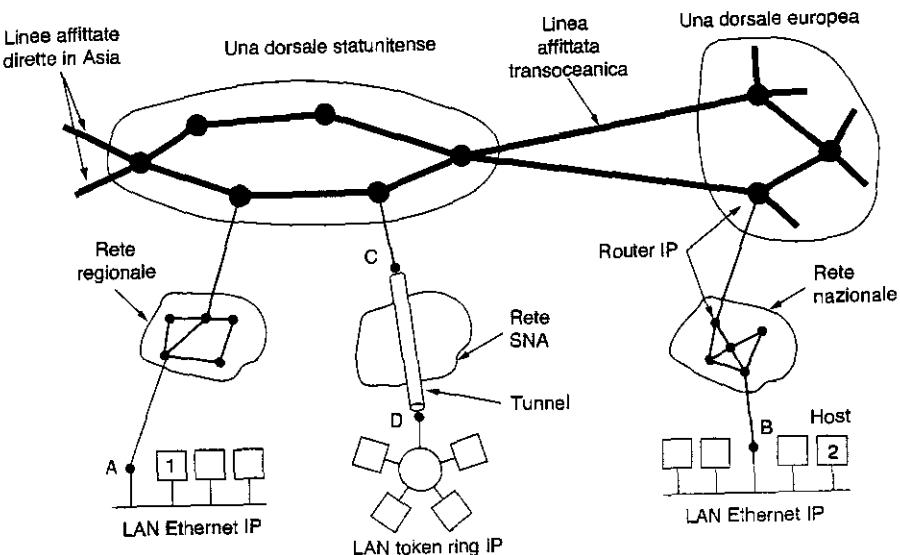


Figura 5.52. Internet è un insieme di reti interconnesse.

da una parte di testo. L'intestazione ha una parte fissa di 20 byte e una parte opzionale di lunghezza variabile; la Figura 5.53 mostra il suo formato. È trasmessa in ordine big endian: da sinistra a destra, a partire dal bit di ordine più elevato del campo *version* (SPARC è big endian, Pentium è little endian.) Sulle macchine little endian, è necessario eseguire una conversione software sia in ricezione sia in trasmissione. Il campo *version* tiene traccia della versione del protocollo usato per il datagramma. Grazie alla informazione sulla versione inserita in ogni datagramma, le transizioni tra versioni possono avvenire lentamente nel corso degli anni, con alcune macchine che continuano a utilizzare la vecchia versione e altre che usano la nuova. Al momento è in corso una transizione da IPv4 a IPv6, che ha avuto inizio anni fa e il cui completamento richiederà ancora parecchio tempo (Durand, 2001; Wiljakkala, 2002; e Waddington and Chang, 2002). Alcune persone pensano addirittura che la transizione non finirà mai (Weiser, 2001). A proposito di numerazione, IPv5 è stato un protocollo sperimentale streaming in tempo reale, mai utilizzato nella realtà.

Poiché la lunghezza dell'intestazione non è costante, l'intestazione contiene un campo *IHL* che indica la lunghezza dell'intestazione espressa in parole di 32 bit: il valore minimo, 5, si applica quando non sono presenti altre opzioni; il valore massimo di questo campo a 4 bit è 15, quindi la dimensione massima dell'intestazione è 60 byte e il campo *options* (opzioni) non può occupare più di 40 byte. Alcune opzioni, come quella che registra il percorso seguito dal pacchetto, hanno bisogno di più di 40 byte, e quindi non possono usare questo campo. Il campo *type of service* (tipo di servizio) è uno dei pochi campi che nel corso degli anni ha (leggermente) cambiato il suo significato. Era (ed è ancora) usato per distinguere diverse classi di servizio; sono ammesse svariate combinazioni di affidabilità e velocità. Per la trasmis-

L'elenco completo può essere scaricato dalla pagina Web www.iana.org/assignments/ip-parameters. L'opzione *security* (sicurezza) indica il livello di segretezza delle informazioni. In teoria, un router militare potrebbe utilizzare questo campo per impedire il routing dei dati attraverso certi paesi considerati ostili; in realtà tutti i router lo ignorano, perciò la sua unica funzione pratica è quella di aiutare le spie a trovare più facilmente le informazioni più interessanti.

Significato	Descrizione
Sicurezza	Specifica il livello di segretezza del datagramma
Instradamento strettamente definito dall'origine	Definisce il percorso completo da seguire
Instradamento lasciamente definito dall'origine	Elenca i router che non devono essere mancati
Registra il percorso	Fa sì che ogni router aggiunga il proprio indirizzo IP
Contrassegno temporale	Fa sì che ogni router aggiunga indirizzo e ora

Figura 5.54. Alcune opzioni di IP.

L'opzione *strict source routing* definisce il percorso completo dalla sorgente alla destinazione, attraverso una sequenza di indirizzi IP. Il datagramma deve seguire quel percorso preciso. Per lo più è utilizzato dagli amministratori di sistema, per inviare pacchetti di emergenza in caso di danneggiamento dei router o per eseguire misure di sincronizzazione.

L'opzione *loose source routing* costringe il pacchetto ad attraversare un elenco di router specificati, nell'ordine indicato; il pacchetto può comunque attraversare anche altri router durante il suo viaggio. Normalmente, per forzare un particolare percorso, questa opzione indica solo pochi router. Per esempio, per costringere un pacchetto proveniente da Londra e diretto a Sydney a viaggiare in direzione ovest invece che est, questa opzione potrebbe specificare i router di New York, Los Angeles e Honolulu. Questa opzione è utilizzata soprattutto quando gli attori politici ed economici impongono di attraversare o di evitare particolari paesi.

L'opzione *record route* (registra il percorso) costringe ogni router lungo la strada ad aggiungere il proprio indirizzo IP al campo delle opzioni. Questo permette agli amministratori di sistema di individuare gli errori negli algoritmi di routing ("Come mai i pacchetti provenienti da Houston e diretti a Dallas passano prima per Tokyo?"). Ai tempi di ARPANET, nessun pacchetto superava più di nove router, perciò 40 byte di opzioni erano più che sufficienti. Come è stato spiegato precedentemente, oggi questo spazio è inadeguato.

Infine, l'opzione *timestamp* (contrassegno temporale) assomiglia all'opzione *record route* e costringe ogni router a registrare sia i 32 bit che rappresentano l'indirizzo IP sia i 32 bit che indicano la data e l'ora di elaborazione. Anche questa opzione è utilizzata per correggere gli algoritmi di routing.

5.2 Indirizzi IP

Un host e router di Internet ha un indirizzo IP che codifica il suo indirizzo di rete e il suo numero di host. La combinazione è unica: in teoria, non possono esistere su Internet due macchine con lo stesso indirizzo IP.

Tutti gli indirizzi IP sono lunghi 32 bit e sono utilizzati nei campi *indirizzo sorgente* e *indirizzo di destinazione* dei pacchetti IP. È importante notare che un indirizzo IP non si riferisce veramente a un host ma a una scheda di rete, perciò quando un host ha due schede di rete, deve avere due indirizzi IP. Comunque, la maggior parte degli host è collegata a un'unica rete e perciò ha un solo indirizzo IP.

Per molti decenni, gli indirizzi IP sono stati divisi nelle cinque categorie elencate nella Figura 5.55. Questa assegnazione è stata chiamata **indirizzamento per classi**; il sistema oggi non è più utilizzato, anche se nella letteratura informatica si fa ancora riferimento a esso. La soluzione adottata al posto delle classi di indirizzi è descritta più avanti.

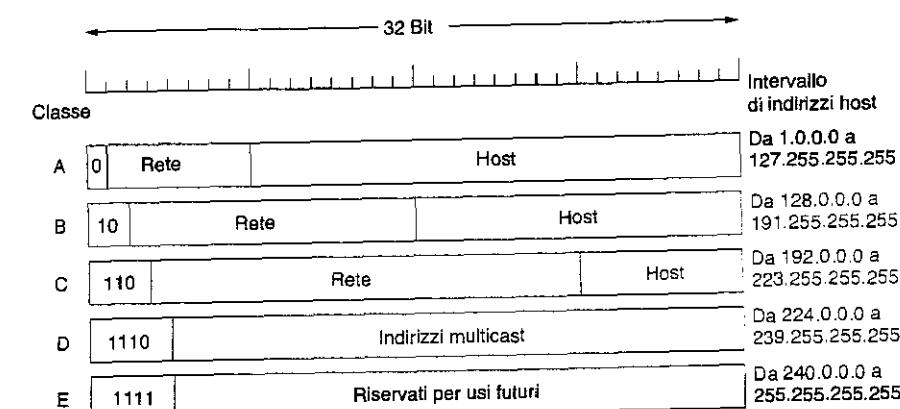


Figura 5.55. Formati degli indirizzi IP.

I formati di classe A, B, C e D supportano complessivamente un numero massimo di 128 reti con 16 milioni di host ciascuna, 16.384 reti da 64.000 host e 2 milioni di reti (LAN) da 256 host (alcune di queste sono riservate a utilizzi particolari). È supportato anche il multicast, cioè l'invio dei datagrammi verso più host. Gli indirizzi che iniziano con 1111 sono riservati per applicazioni future. Oggi sono collegate a Internet più di 500.000 reti, e il numero cresce di anno in anno. Per evitare conflitti, i numeri di rete sono gestiti da un ente no profit chiamato ICANN (*Internet Corporation for Assigned Names and Numbers*). ICANN ha affidato la gestione di alcune parti dello spazio di indirizzamento ad autorità regionali, che distribuiscono con parsimonia gli indirizzi IP agli ISP e alle altre aziende. Gli indirizzi di rete, rappresentati da numeri a 32 bit, sono di solito scritti in **notazione decimale a punti**. In questo formato, ognuno dei 4 byte è rappresentato in forma decimale con un numero che varia tra 0 e 255. Per esempio, l'indirizzo esadecimale a 32 bit C0290614 è scritto come 192.41.6.20. Il più basso indirizzo IP è 0.0.0.0, il più alto è 255.255.255.255.

I valori 0 e -1 (tutti 1) hanno un significato speciale, come mostrato nella Figura 5.56. Lo 0 indica la rete corrente (oppure l'host corrente), -1 è utilizzato come indirizzo broadcast e rappresenta tutti gli host sulla rete indicata.

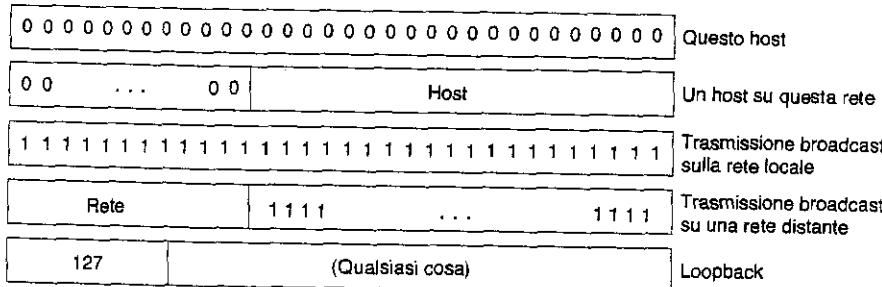


Figura 5.56. Indirizzi IP speciali.

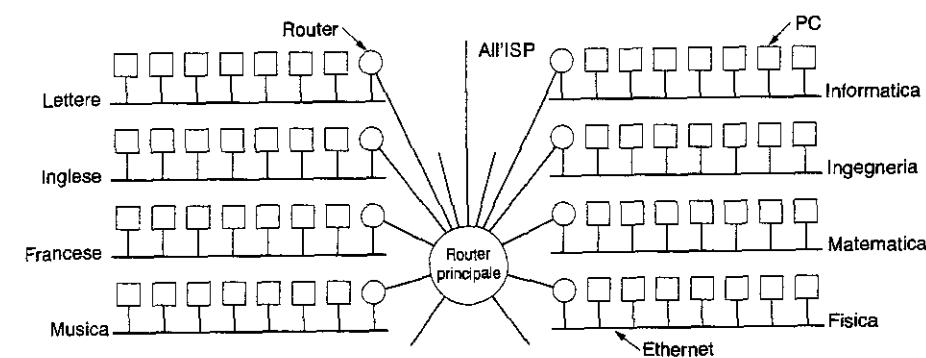
L'indirizzo IP 0.0.0.0 è utilizzato dagli host al momento del boot. Gli indirizzi IP che hanno lo 0 come numero di rete si riferiscono alla rete corrente. Grazie a questi indirizzi, i computer possono utilizzare la rete locale senza conoscerne il numero (devono però conoscere la sua classe, per sapere quanti zeri includere). L'indirizzo composto da tutti 1 permette la trasmissione broadcast sulla rete locale, in genere una LAN. Gli indirizzi con un numero di rete opportuno e tutti 1 nel campo host permettono ai computer l'invio di pacchetti broadcast a LAN distanti collegate a Internet (però molti amministratori di rete disattivano questa funzionalità). Infine, tutti gli indirizzi espressi nella forma 127.xx.yz sono riservati per le prove di loopback. I pacchetti diretti a quell'indirizzo non sono immessi nel cavo, ma elaborati localmente e trattati come pacchetti in arrivo. Questo permette di inviare pacchetti alla rete locale senza che il trasmettitore conosca il suo numero.

Sottoreti

Come si è visto, tutti gli host di una rete devono avere lo stesso numero di rete. Questa proprietà dell'indirizzamento IP può causare problemi quando le reti crescono. Per esempio, si consideri un'università che comincia con una rete di classe B usata dalla facoltà di Ingegneria Informatica per i computer della sua Ethernet. Un anno dopo, anche la facoltà di Ingegneria Elettronica vuole accedere a Internet, perciò per collegare la prima Ethernet all'edificio della seconda facoltà viene acquistato un ripetitore. Con l'andare del tempo, molte altre facoltà acquistano computer e il limite di quattro ripetitori per Ethernet viene raggiunto. È necessario adottare una soluzione diversa.

Ottenerne un secondo indirizzo di rete è difficile, poiché gli indirizzi di rete sono scarsi e l'università ha già abbastanza indirizzi per più di 60.000 host. Il problema dipende dalla regola che lega un singolo indirizzo di classe A, B o C a una sola rete, invece che a un gruppo di LAN. Con l'andare del tempo il numero di organizzazioni imbattutesi in questa situazione è cresciuto, e per risolvere il problema è stato necessario apportare una piccola modifica al sistema di indirizzamento.

La soluzione consiste nel dividere internamente la rete in più parti, facendo in modo che il mondo esterno veda ancora una singola rete. Una tipica rete universitaria attuale potrebbe assomigliare a quella rappresentata nella Figura 5.57, con un router principale collegato a un



ISP o a una rete regionale, e numerose Ethernet sparse in diverse facoltà dell'università. Ogni Ethernet ha il proprio router collegato al router principale (per esempio attraverso una LAN dorsale, ma in questo caso la natura della connessione tra router non è importante).

Nella letteratura dedicata a Internet, le parti della rete (in questo caso le Ethernet) sono chiamate **sottoreti**. Come è stato spiegato nel Capitolo 1, questa consuetudine è in conflitto con il termine "sottorete" usato per indicare l'insieme composto da tutti i router e dalle linee di comunicazione di una rete. In genere il contesto dovrebbe rendere più chiaro il significato della parola. In questo paragrafo e nel successivo, sarà utilizzata soltanto la nuova definizione.

Quando un pacchetto raggiunge il router principale, come viene identificata la sottorete (Ethernet) di destinazione di quei dati? Per esempio, si potrebbe memorizzare nel router principale una tabella contenente 65.536 voci che indicano il router da utilizzare per ogni host dell'università; questa idea funzionerebbe, ma richiederebbe l'implementazione nel router principale di una tabella molto grande e un'impegnativa gestione manuale degli host aggiuntivi, spostati o guasti.

Per questo motivo è stato inventato uno schema diverso. In breve, invece di avere un singolo indirizzo di classe B con 14 bit per il numero di rete e 16 bit per il numero host, alcuni bit sono tolti al numero host per creare un numero di sottorete. Per esempio, se avesse 35 facoltà, l'università potrebbe utilizzare un numero di sottorete di 6 bit e un numero host di 10 bit; questa combinazione permetterebbe 64 Ethernet, ognuna con una massima di 1.022 host (0 e -1 non sono disponibili, come è stato spiegato precedentemente). Questa suddivisione potrebbe essere modificata in un secondo momento in caso di necessità.

Per implementare le sottoreti, il router principale ha bisogno di una **maschera di sottorete** (*subnet mask*) che indichi il punto di demarcazione tra numero di rete (inclusivo di quello della sottorete) e quello di host, come mostrato nella Figura 5.58. Anche le maschere di sottorete sono scritte in notazione decimale a punti, oppure da una barra seguita dal numero di bit della parte che rappresenta la rete più la sottorete. Nel caso della Figura 5.58 la maschera di sottorete può essere scritta come 255.255.252.0, oppure dalla notazione alternativa /22 che indica che la maschera di sottorete è lunga 22 bit.

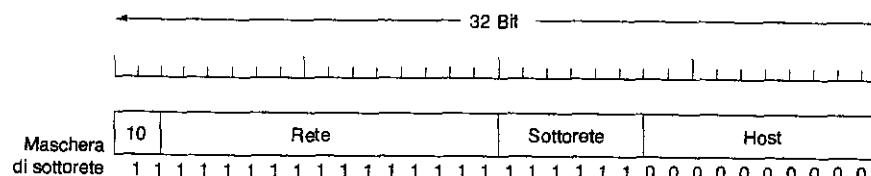


Figura 5.58. Una rete di classe B divisa in 64 sottoreti.

All'esterno della rete la divisione in sottoreti non è visibile, perciò l'assegnazione di una nuova sottorete non richiede l'approvazione di ICANN o la modifica di database esterni. In questo esempio, la prima sottorete potrebbe utilizzare gli indirizzi IP che iniziano dal valore 130.50.4.1; la seconda sottorete potrebbe iniziare da 130.50.8.1; la terza sottorete potrebbe iniziare da 130.50.12.1 e così via. Per vedere come mai le sottoreti sono contate per multipli di 4, si noti che gli indirizzi binari corrispondenti sono:

Subnet 1: 10000010 00110010 00000100 00000001
 Subnet 2: 10000010 00110010 00001010 00000001
 Subnet 3: 10000010 00110010 00001110 00000001

La barra verticale (|) indica il confine tra il numero di sottorete e il numero di host. Alla sua sinistra c'è il numero di sottorete di 6 bit, alla sua destra si trova il numero di host a 10 bit. Per capire come funzionano le sottoreti è necessario spiegare in che modo i pacchetti IP sono elaborati dai router. Ogni router ha una tabella che elenca numeri di indirizzi IP del tipo (rete, 0) e numeri di indirizzi IP del tipo (questa rete, host). Il primo tipo indica come raggiungere reti distanti; il secondo tipo indica come raggiungere host locali. A ogni tabella è associata la scheda di rete da utilizzare per raggiungere la destinazione, insieme ad altre informazioni. Quando riceve un pacchetto IP, il router cerca il suo indirizzo di destinazione nella tabella di routing. Se è indirizzato a una rete distante, il pacchetto viene inoltrato al router successivo sull'interfaccia indicata dalla tabella; se la destinazione è un host locale (sulla LAN del router), il pacchetto viene inviato direttamente alla destinazione. Se la rete non viene trovata nella tabella, il pacchetto viene inoltrato a un router predefinito contenente tabelle più estese. In base a questo algoritmo, ogni router deve tenere traccia solo delle altre reti e degli host locali, invece che di tutte le coppie (rete, host); ciò riduce di molto le dimensioni della tabella di routing.

Quando si introduce la tecnica della divisione in sottoreti le tabelle di routing cambiano, con l'aggiunta di voci espresse nella forma (questa rete, sottorete, 0) e (questa rete, questa sottorete, host). Perciò un router su una sottorete k sa come raggiungere tutte le altre sottoreti e sa anche come raggiungere tutti gli host della sottorete k . Il dispositivo non deve conoscere i dettagli relativi agli host che si trovano sulle altre sottoreti. In realtà la modifica è minima: è stato sufficiente consentire a ogni router di eseguire una operazione AND booleana con la maschera di sottorete, per eliminare il numero host e cercare l'indirizzo risultante nelle proprie tabelle (dopo aver determinato la classe di rete di appartenenza). Nell'esempio, un pacchetto indirizzato a 130.50.15.6 che arriva al router principale è sot-

toposto a un'operazione di AND con la maschera di sottorete 255.255.252.0/22; il risultato è 130.50.12.0. Il router principale cerca questo indirizzo nelle tabelle di routing per identificare la linea di output da utilizzare per raggiungere il router della sottorete 3. La divisione in sottoreti riduce lo spazio della tabella del router, in quanto crea una gerarchia a tre livelli associata alla rete, alla sottorete e agli host.

CIDR (*Classless InterDomain Routing*)

IP è utilizzato intensamente da decenni e ha funzionato estremamente bene, come dimostra la crescita esponenziale di Internet. Sfortunatamente, IP sta diventando rapidamente una vittima della sua stessa popolarità: sta esaurendo gli indirizzi. Questo disastro che si profila all'orizzonte ha suscitato moltissime discussioni e polemiche nella comunità di Internet, impegnata a trovare una soluzione. Questo paragrafo descrive alcuni problemi e diverse soluzioni proposte.

Nel lontano 1987 alcuni visionari predissero che un giorno Internet sarebbe stata composta da 100.000 reti. La maggior parte degli esperti prese alla leggera questa eventualità, ritenendola lontana decenni se non addirittura impossibile. La centomillesima rete si è collegata a Internet nel 1996. Il problema, come è stato detto precedentemente, è che Internet sta rapidamente esaurendo gli indirizzi IP. Inizialmente esistevano più di due miliardi di indirizzi, ma il sistema di suddivisione basato sulle classi (Figura 5.55) ha sprecato milioni di indirizzi. In particolare, il vero cattivo della situazione è rappresentato dalla rete di classe B. Per la maggior parte delle società, una rete di classe A con 16 milioni di indirizzi è troppo grande, e una rete di classe C con 256 indirizzi è troppo piccola. Una rete di classe B (con 65.536 indirizzi) invece è perfetta.

In realtà un indirizzo di classe B è ancora di gran lunga troppo grande per la maggior parte delle aziende. Le ricerche hanno dimostrato che più della metà di tutte le reti di classe B hanno meno di 50 host. Una rete di classe C sarebbe stata sufficiente, ma senza dubbio ogni azienda che ha chiesto un indirizzo di classe B pensava che un giorno gli 8 bit del campo host non sarebbero più stati sufficienti. In retrospettiva, sarebbe stato meglio assegnare al numero di host delle reti di classe C 10 bit invece di 8, in modo da supportare 1.022 host per rete. Con una scelta del genere, la maggior parte delle aziende avrebbe richiesto una rete di classe C e gli indirizzi di questo tipo sarebbero stati mezzo milione (mentre le reti di classe B sono solo 16.384).

È difficile criticare i progettisti di Internet per non aver fornito un numero maggiore di indirizzi di classe B (più piccoli). Quando venne presa la decisione di creare le tre classi, Internet era una rete di ricerca che collegava le principali università degli Stati Uniti (più un piccolo numero di aziende e di siti militari impegnati nelle ricerche della comunicazione di rete). Nessuno pensava che Internet sarebbe diventato un sistema di comunicazione di massa in grado di competere con la rete telefonica. Allora qualcuno senza dubbio pensò: "Gli Stati Uniti hanno circa 2.000 università; anche se si collegassero tutte a Internet e anche se le università di altri paesi si unissero alla rete, non sarà mai possibile raggiungere il valore limite di 16.000, poiché in tutto il mondo non ci sono così tante università. Inoltre, usando come numero host un valore intero di byte si accelera l'elaborazione dei pacchetti."

Comunque, se la divisione avesse assegnato 20 bit al numero di rete di classe B, sarebbe emer-

un altro problema: l'esplosione delle tabelle di routing. Dal punto di vista dei router, lo spazio degli indirizzi IP costituisce una gerarchia a due livelli rappresentata dai numeri di rete e numeri di host. I router non devono sapere tutto degli host, devono però conoscere tutte le reti. Se si utilizzassero cinquecentomila reti di classe C, ogni router di Internet avrebbe bisogno di una tabella contenente mezzo milione di voci, una per ogni rete, indicante la linea da utilizzare per raggiungere quella rete (insieme ad altre informazioni).

L'archiviazione fisica di tabelle da mezzo milione di voci è probabilmente un'operazione fatale, sebbene costosa per router più importanti, che conservano le tabelle nella RAM statica installata nelle schede di I/O. Un problema più serio è la crescita della complessità dei vari algoritmi che si occupano della gestione delle tabelle, più rapida di quella lineare. Peggiora, gran parte del software e del firmware utilizzato dai router esistenti è stata progettata quando Internet era composta da 1.000 reti collegate e 100.000 reti sembravano lontane anni. Le scelte di progetto prese allora sono spesso lontane dall'essere ottimali oggi.

Tre vari algoritmi di routing richiedono che ogni router trasmetta periodicamente le proprie tabelle (come accade con i protocolli basati sul vettore delle distanze). Più sono grandi le tabelle, più è probabile che alcune parti si perdano lungo la strada, generando dati incomplete all'altro capo della linea e probabilmente instabilità nel routing. Il problema della tabella di routing potrebbe essere risolto creando una gerarchia più ramificata; per esempio, si poteva immaginare di inserire in ogni indirizzo IP dei campi per indicare il paese, lo stato o la provincia, la città, la rete e l'host. In quel caso, un router avrebbe dovuto conoscere solamente le reti di ogni paese, gli stati o le province nel proprio paese, le città nella sua provincia o stato, e le reti nella sua città. Sfortunatamente, questa soluzione richiederebbe un numero maggiore dei 32 bit usati dagli indirizzi IP, e li utilizzerebbe in modo inefficiente. L'Europa avrebbe lo stesso numero di bit degli Stati Uniti.

Sumiendo, alcune soluzioni risolvono un problema ma ne creano uno nuovo. La soluzione implementata per dare a Internet un po' di respiro si chiama **CIDR** (*Classless Domain Routing*). L'idea di fondo di CIDR, descritta nel documento RFC 1519, è di assegnare gli indirizzi IP rimanenti in blocchi di dimensioni variabili, senza tener conto delle classi. Per esempio, a un sito che ha bisogno di 2.000 indirizzi viene assegnato un blocco di 2.048 indirizzi (con un allineamento a 2.048 byte).

Abbandonando delle classi rende l'inoltro più complicato. Nel vecchio sistema basato su classi, l'inoltro funzionava in questo modo: quando un pacchetto raggiungeva un router, una copia dell'indirizzo IP era spostata a destra di 28 bit in modo da ottenere il numero di classe. Poi una diramazione a sedici percorsi ordinava i pacchetti in A, B, C, D e E (se supportata), con otto casi per la classe A, quattro casi per la classe B, due per la classe C e uno per ogni D ed E. Il codice di ogni classe poi mascherava il numero di rete a 8, 16 o 24 bit e lo allineava a destra in una parola di 32 bit. Il numero di host era quindi cercato nella tabella A, B o C, di solito per indice nelle reti di classe A e per hash nelle reti di classe C. Una volta trovata la voce, si cercava la linea di output e si inoltrava il pacchetto.

CIDR questo semplice algoritmo non funziona più. Adesso ogni voce della tabella di routing è integrata da una maschera di 32 bit, perciò c'è una singola tabella di routing per tutte le reti costituita da una matrice di valori nella forma (indirizzo IP, maschera di sottorete, linea di output). Quando arriva un pacchetto, il router estrae prima di tutto il suo indirizzo IP di destinazione; poi (concettualmente), esamina la tabella di routing, voce per voce, mascherando l'indirizzo di destinazione e confrontandolo con le voci della tabella. È possibile che più voci (con diverse lunghezze di maschera di sottorete) coincidano con il valore cercato, in questo caso è utilizzata la maschera più lunga. Così, se trova una corrispondenza con una maschera /20 e con una maschera /24, il router sceglie la voce /24. Sono stati sviluppati algoritmi complessi che accelerano il processo di ricerca dell'indirizzo (Ruiz-Sánchez et al., 2001). I router commerciali utilizzano chip VLSI personalizzati, che incorporano questi algoritmi nell'hardware.

Per aiutare a comprendere il funzionamento dell'algoritmo di inoltro, si consideri una situazione di esempio in cui milioni di indirizzi sono disponibili a partire da 194.24.0.0. Si supponga che all'Università di Cambridge, che ha bisogno di 2.048 indirizzi, siano assegnati gli indirizzi compresi tra 194.24.0.0 e 194.24.7.255, insieme alla maschera 255.255.248.0. Successivamente l'Università di Oxford chiede 4.096 indirizzi; poiché un

Università	Primo Indirizzo	Ultimo indirizzo	Totale indirizzi	Scritti come
Cambridge	194.24.0.0	194.24.7.255	2.048	194.24.0.0/21
Edimburgo	194.24.8.0	194.24.11.255	1.024	194.24.8.0/22
(disponibili)	194.24.12.0	194.24.15.255	1.024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4.096	194.24.16.0/20

Figura 5.59. Una serie di assegnazioni di indirizzi IP.

blocco di 4.096 indirizzi deve trovarsi allineato a 4.096 byte, non è possibile assegnare al richiedente indirizzi che partono da 194.24.8.0. L'università riceve perciò gli indirizzi compresi tra 194.24.16.0 e 194.24.31.255 insieme alla maschera di sottorete 255.255.240.0. A questo punto, all'Università di Edimburgo che chiede 1024 indirizzi sono assegnati gli indirizzi compresi tra 194.24.8.0 e 194.24.11.255 insieme alla maschera 255.255.252.0. Queste assegnazioni sono riepilogate nella Figura 5.59.

Le tabelle di routing in tutto il mondo sono aggiornate con le tre voci appena assegnate. Ogni voce contiene un indirizzo base e una maschera di sottorete. Le voci, espresse in formato binario, sono:

Indirizzo	Maschera
C: 11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
E: 11000010 00011000 00010000 00000000	11111111 11111111 11111100 00000000
O: 11000010 00011000 00010000 00000000	11111111 11111111 11110000 00000000

Ora si consideri che cosa accade quando arriva un pacchetto indirizzato a 194.24.17.4, che in formato binario è rappresentato dalla seguente stringa di 32 bit

11000010 00011000 00010001 00000100

L'indirizzo viene prima di tutto sottoposto a un'operazione logica AND con la maschera di Cambridge, il risultato è

11000010 00011000 00010000 00000000

Questo valore non corrisponde all'indirizzo di base di Cambridge, perciò l'indirizzo originale viene sottoposto a un'operazione logica AND con la maschera di Edimburgo. Il risultato è

11000010 00011000 00010000 00000000

Questo valore non corrisponde all'indirizzo di base di Edimburgo, così si tenta con Oxford. Il risultato questa volta è

11000010 00011000 00010000 00000000

L'ultimo valore coincide con l'indirizzo di Oxford. Se più in giù nella tabella non si trovano altre corrispondenze, il router utilizza la voce relativa a Oxford e il pacchetto è inviato lungo la linea associata.

Ora, si considerino le tre università dal punto di vista di un router che si trova a Omaha, Nebraska, che dispone solo di quattro linee di output dirette a Minneapolis, New York, Dallas e Denver. Quando il software del router riceve le tre nuove voci, nota che può combinare tutti e tre gli elementi in una singola **voce aggregata** 194.24.0.0/19. La rappresentazione binaria dell'indirizzo base e della maschera di sottorete è

11000010 0000000 00000000 00000000 11111111 11111111 11100000 00000000

Questa voce invia a New York tutti i pacchetti destinati a ognuna delle tre università. Aggregando queste tre voci, il router di Omaha ha diminuito di due voci le dimensioni della sua tabella.

Se New York ha una sola linea diretta a Londra per tutto il traffico inglese, anche questo router può utilizzare una voce aggregata, ma se il dispositivo ha linee separate per Londra ed Edimburgo deve avere voci separate. L'aggregazione è usata intensamente in Internet per ridurre la dimensione delle tabelle dei router.

Come nota finale su questo esempio, la voce aggregata nel router di Omaha invia a New York anche i pacchetti trasmessi a indirizzi non assegnati. Finché gli indirizzi rimangono veramente non assegnati, tutto ciò non ha importanza; ma se gli indirizzi successivamente sono assegnati a una società californiana, allora è necessario gestire una nuova voce, 194.24.12.0/22.

NAT (*Network Address Translation*)

Gli indirizzi IP sono scarsi. Un ISP potrebbe avere un indirizzo /16 (un tempo chiamato di classe B) che consente di gestire 65.534 numeri host. Se il numero dei clienti supera questo valore, sorge un problema. Con i clienti domestici che usano connessioni remote è possibile evitarlo assegnando gli indirizzi IP ai computer in modo dinamico al momento della connessione, e recuperando gli indirizzi al termine della sessione. In questo modo, un singolo indirizzo /16 permette di gestire fino a 65.534 utenti attivi, soluzione abbastanza buona per un ISP con diverse centinaia di migliaia di clienti. Al termine della sessione, l'indirizzo IP è assegnato a un altro chiamante. Questa strategia funziona bene con gli ISP che hanno un numero modesto di utenti domestici, ma non è adatta agli ISP che servono principalmente clienti commerciali.

Il problema è che i clienti commerciali prevedono di essere continuamente in linea durante le ore di lavoro. I piccoli uffici (come le agenzie di viaggio con tre impiegati) e le grandi aziende hanno più computer collegati via LAN. Alcuni computer sono i PC degli impiegati, altri possono essere dei server Web; in generale, sulla LAN c'è un router collegato all'ISP da una linea affittata che fornisce connessione continua. Questa disposizione significa che ogni computer deve mantenere il proprio indirizzo IP disponibile per tutto il giorno. In pratica, il numero totale di computer posseduti dai clienti commerciali non può superare il numero di indirizzi IP a disposizione dell'ISP; se possiede un indirizzo /16, questo limita il numero totale di computer a 65.534. Un ISP con decine di migliaia di clienti commerciali supera rapidamente questo limite.

A peggiorare il tutto, è in continua crescita il numero degli utenti domestici che sottoscrivono abbonamenti ADSL o Internet via cavo. Due caratteristiche frequenti di questi servizi sono (1) che l'utente riceve un indirizzo IP permanente e (2) che il costo dell'abbonamento non dipende dalla durata della connessione (l'abbonamento è di tipo flat, cioè l'utente paga un addebito fisso mensile). Per questi motivi molti utenti ADSL o via cavo rimangono collegati in modo permanente, creando una situazione che peggiora la scarsità di indirizzi IP. Il meccanismo dell'assegnazione provvisoria degli indirizzi IP, adottato con gli utenti remoti collegati via telefono, non è di alcuna utilità in questo caso: a un dato istante il numero di indirizzi IP necessari agli utenti connessi può superare di molto il numero d'indirizzi IP posseduti dall'ISP.

E tanto per rendere le cose ancora più complicate, molti utenti ADSL e via cavo hanno a casa due o più computer, spesso uno per ogni membro della famiglia, e tutti vogliono essere sempre collegati usando il singolo indirizzo IP che l'ISP ha assegnato loro. Questo risultato si può ottenere collegando tutti i PC in una LAN e installando un router. Dal punto di vista degli ISP, la famiglia è simile a un piccolo ufficio contenente una manciata di computer: benvenuti alla ditta Rossi.

Il problema dell'esaurimento degli indirizzi IP non è un problema teorico che potrebbe presentarsi in un lontano futuro: sta accadendo proprio qui e ora. La soluzione a lungo termine è che tutta Internet passi a IPv6, protocollo che adotta indirizzi a 128 bit. Questa transizione sta procedendo lentamente, e ci vorranno anni prima che si completi. Di conseguenza, alcune persone si sono rese conto che era necessario trovare una soluzione rapida attuabile in tempi brevi. La soluzione adottata si chiama **NAT** (*Network Address Translation*), è descritta nel documento RFC 3022 ed esaminata brevemente in questo paragrafo. Per informazioni più dettagliate, consultare (Dutcher, 2001).

L'idea di base di NAT è assegnare a ogni azienda un singolo indirizzo IP (o, al massimo, un piccolo numero di indirizzi) per il traffico di Internet. Dentro l'azienda, ogni computer riceve un indirizzo IP unico utilizzato per instradare il traffico interno alla rete locale, ma quando un pacchetto lascia l'azienda e si dirige verso l'ISP, viene eseguita una traduzione di indirizzo. Per rendere fattibile questo schema, sono stati dichiarati privati tre intervalli di indirizzi IP, che le aziende possono utilizzare internamente come preferiscono. L'unica regola è che nessun pacchetto contenente questi indirizzi può apparire su Internet. I tre intervalli riservati sono:

10.0.0.0 – 10.255.255.255/8 (16.777.216 host)
 172.16.0.0 – 172.31.255.255/12 (1.048.576 host)
 192.168.0.0 – 192.168.255.255/16 (65.536 host)

Il primo intervallo permette di gestire 16.777.216 host (sono esclusi, come sempre, i valori 0 e -1) ed è la scelta comune in molte aziende, anche in quelle che non hanno bisogno di così tanti indirizzi.

Il funzionamento di NAT è mostrato nella Figura 5.60. Dentro i locali dell'azienda, ogni macchina ha un unico indirizzo espresso nella forma 10.x.y.z, ma quando un pacchetto lascia l'azienda, passa attraverso un dispositivo NAT che converte l'indirizzo IP interno (10.0.0.1 nella figura) nel vero indirizzo IP assegnato all'azienda (198.60.42.12 in questo esempio). Il dispositivo NAT è spesso abbinato a un firewall, inserito all'interno di un singolo apparecchio che protegge la rete locale controllando attentamente tutti i dati che entrano e che escono dalla LAN (i firewall sono esaminati nel Capitolo 8). Si può anche integrare il dispositivo NAT nel router aziendale.

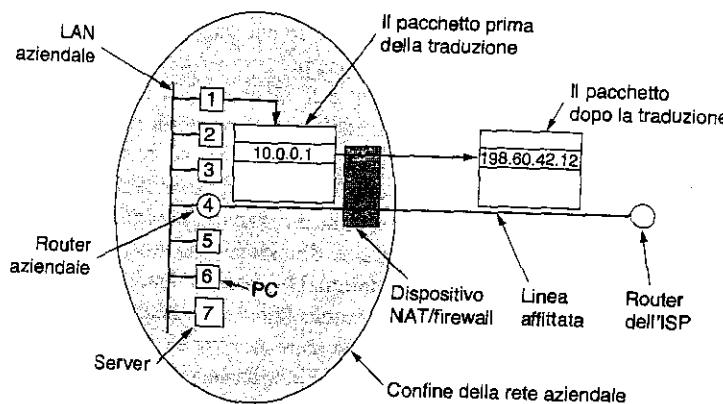


Figura 5.60. Collocazione e funzionamento di un dispositivo NAT.

Fino a questo punto è stato trascurato un piccolo dettaglio: quando ritorna la risposta (per esempio da un server Web), i dati sono naturalmente indirizzati a 198.60.42.12; in che modo, allora, il dispositivo NAT sceglie l'indirizzo di destinazione interno? È qui che si presenta il problema di NAT. Se ci fosse un campo di riserva nell'intestazione IP lo si potrebbe utilizzare per tener traccia del trasmittente reale, ma è rimasto inutilizzato solo 1 bit [NdR - non si considerano i bit inutilizzati all'interno del campo *type of service*].

In linea di principio si sarebbe potuta creare una nuova opzione che conservasse il vero indirizzo sorgente; la gestione di questo meccanismo, però, avrebbe richiesto la modifica del codice IP di tutti i computer di Internet, quindi non sarebbe stata rapida. Ecco che cosa è successo realmente: i progettisti di NAT hanno osservato che la maggior parte dei pacchetti IP trasporta carichi utili TCP o UDP. Come è verrà spiegato nel capitolo 6 nei paragrafi dedicati a TCP e UDP, entrambi i protocolli hanno intestazioni che

contengono una porta sorgente e una porta di destinazione. La descrizione che segue, relativa alle porte TCP, vale in modo analogo per le porte UDP. Le porte sono numeri interi lunghi 16 bit che indicano dove inizia e finisce la connessione TCP, e offrono il campo che permette di far funzionare NAT.

Quando un processo desidera stabilire una connessione TCP con un processo remoto, si lega a una porta TCP inutilizzata presente nella sua macchina; questa porta è chiamata **porta sorgente** e indica al codice TCP dove devono essere inviati i pacchetti in arrivo appartenenti alla connessione. Il processo fornisce anche una **porta di destinazione** che indica chi deve ricevere i pacchetti sulla parte remota. Le porte da 0 a 1023 sono riservate a servizi noti; per esempio, la porta 80 è utilizzata dai server Web, e ciò consente ai client remoti di individuare il servizio. Ogni messaggio TCP in uscita contiene sia la porta sorgente sia la porta di destinazione, e insieme aiutano a identificare il processo che utilizza la connessione.

Un'analogia può chiarire il funzionamento delle porte. S'immagini un'azienda che dispone di un unico numero di telefono. Quando qualcuno chiama il numero principale, risponde un operatore che chiede il numero di interno con cui la persona desidera parlare e poi inoltra la chiamata all'interno indicato. Il numero principale equivale all'indirizzo IP assegnato all'azienda, e i numeri degli interni rappresentano le porte. Le porte sono altri 16 bit di indirizzamento che identificano il processo al quale è destinato il pacchetto in arrivo. L'utilizzo del campo *source port* (porta sorgente) permette di risolvere il problema della mappatura. Ogni volta che un pacchetto diretto verso l'esterno raggiunge il dispositivo NAT, l'indirizzo sorgente 10.x.y.z è sostituito dall'indirizzo IP dell'azienda. Inoltre, il campo *source port* è sostituito da un indice che punta alla tabella di traduzione da 65.536 voci del dispositivo NAT. Questa voce di tabella contiene l'indirizzo IP originale e la porta sorgente originale. Infine, i codici di controllo delle intestazioni IP e TCP sono ricalcolati e inseriti nel pacchetto.

È necessario sostituire il campo *source port* perché, per esempio, le connessioni dalle macchine 10.0.0.1 e 10.0.0.2 possono entrambe utilizzare la porta 5000; perciò la porta sorgente non basta a identificare, da sola, il processo trasmittente.

Quando un pacchetto trasmesso dall'ISP raggiunge il dispositivo NAT, la *source port* nell'intestazione TCP viene estratta e utilizzata come indice nella tabella di mappatura del dispositivo NAT. Dalla voce individuata, il dispositivo estrae l'indirizzo IP interno e la *source port* TCP originale, quindi inserisce entrambe le informazioni nel pacchetto; poi ricalcola e inserisce nel pacchetto sia il checksum IP sia quello TCP. Alla fine il pacchetto viene passato al router aziendale per il normale inoltro basato sull'indirizzo 10.x.y.z. NAT può essere utilizzato anche per alleviare la carenza di indirizzi IP causata dagli utenti ADSL e via cavo. Quando assegna a ogni utente un indirizzo, l'ISP utilizza gli indirizzi 10.x.y.z; quando escono dall'ISP ed entrano nella Internet principale, i pacchetti provenienti dai computer degli utenti passano attraverso un dispositivo NAT che li traduce nel vero indirizzo Internet dell'ISP. Al ritorno, i pacchetti sono sottoposti alla mappatura inversa. Da questo punto di vista, l'ISP e i suoi utenti domestici ADSL/via cavo appaiono a tutto il resto di Internet come una grande azienda.

Sebbene questo schema in un certo senso risolva il problema, molti nella comunità IP lo considerano un vero e proprio abominio. In breve, ecco alcune delle obiezioni. Primo, NAT viola il modello gerarchico di IP, che afferma che ogni indirizzo IP identifica in modo univoco a livello mondiale una singola macchina. L'intera struttura del software di Internet è costruita su questa verità. Con NAT, migliaia di macchine possono utilizzare l'indirizzo 10.0.0.1 (e lo fanno realmente).

Secondo, NAT trasforma Internet da una rete ad assenza di connessione in un tipo di rete orientata alle connessioni, perché il dispositivo NAT deve conservare le informazioni (la mappatura) relativa a ogni connessione che lo attraversa. Il mantenimento dello stato della connessione è una proprietà delle reti orientate alle connessioni, non delle reti ad assenza di connessione. Se il dispositivo NAT si blocca e la sua tabella di mappatura si perde, tutte le sue connessioni TCP vanno distrutte. In assenza di NAT, i guasti dei router non hanno effetto su TCP, perché il processo di trasmissione dopo alcuni secondi va in time-out e ritrasmette tutti i pacchetti che non hanno ricevuto acknowledgement. Con NAT, Internet diventa vulnerabile come una rete a commutazione di circuito.

Terzo, NAT viola la più importante regola della stratificazione dei protocolli: lo strato k non deve essere costretto a fare alcuna ipotesi su ciò che lo strato $k+1$ ha inserito nel campo che rappresenta il carico utile. Questo principio fondamentale serve a rendere gli strati indipendenti. Se in futuro TCP venisse aggiornato a TCP-2, con uno schema di intestazione diverso (per esempio, porte a 32 bit), NAT non funzionerebbe più. L'idea centrale del concetto di protocollo stratificato è quella di assicurare che la modifica apportata a uno strato non renda necessario modificare gli altri, e NAT distrugge questa indipendenza.

Quarto, i processi su Internet non sono obbligati a utilizzare TCP e UDP. Se un utente sulla macchina A decidesse di utilizzare qualche nuovo protocollo di trasporto per parlare con un utente seduto davanti alla macchina B (per esempio, attraverso un'applicazione multimediale), l'introduzione di un dispositivo NAT non farebbe funzionare l'applicazione perché il dispositivo NAT non sarebbe in grado di individuare in modo corretto il campo *source port* TCP. Quinto, alcune applicazioni inseriscono gli indirizzi IP nel corpo del testo; il ricevente estrae questi indirizzi e li utilizza. Poiché NAT non sa nulla di questi indirizzi, non è in grado di sostituirli, perciò qualunque tentativo di utilizzarli fallirebbe. **FTP (File Transfer Protocol)**, il protocollo di trasferimento file standard, funziona in questo modo e può fallire in presenza di NAT, a meno che non vengano prese speciali precauzioni. In modo analogo, anche il protocollo telefonico di Internet H.323 (descritto nel Capitolo 7) ha la stessa proprietà e può non funzionare in presenza di NAT. Si potrebbe correggere NAT per far funzionare H.323, ma essere obbligati ad aggiornare il codice del dispositivo NAT ogni volta che nasce una nuova applicazione non è una buona idea. Sesto, poiché il campo *source port* di TCP è grande 16 bit, ad un indirizzo IP si possono associare al massimo 65.536 macchine. In realtà il numero è un po' più piccolo, perché le prime 4.096 porte sono riservate per usi speciali. Comunque, quando sono disponibili diversi indirizzi IP, ognuno non può gestire più di 61.440 macchine.

Questi e altri problemi legati a NAT sono discussi nel documento RFC 2993. In generale chi si oppone a NAT afferma che risolvere il problema dell'esaurimento degli indirizzi IP usando un espediente temporaneo di scarsa qualità riduce il desiderio di implementare la soluzione reale, ossia IPv6, e questo è male.

5.6.3 Protocolli di controllo Internet

Oltre a IP, utilizzato per il trasferimento dei dati, Internet ha diversi protocolli di controllo utilizzati nello strato network, tra cui ICMP, ARP, RARP, BOOTP e DHCP, che vengono esaminati in questo paragrafo.

ICMP (*Internet Control Message Protocol*)

Il funzionamento di Internet è monitorato attentamente dai router. Quando avviene qualcosa di imprevisto, l'evento è comunicato da **ICMP** (*Internet Control Message Protocol*), che viene utilizzato anche per testare Internet. Sono stati definiti circa una dozzina di tipi di messaggi ICMP, e i più importanti sono elencati nella Figura 5.61. Ogni tipo di messaggio ICMP è incapsulato in un pacchetto IP.

Tipo di messaggio	Descrizione
Destinazione irraggiungibile	Il pacchetto potrebbe non essere inoltrato
Tempo superato	Il campo Time to live ha raggiunto il valore 0
Problema di parametri	Campo dell'intestazione non valido
Spegnimento della sorgente	Pacchetto di interruzione
Reindirizzamento	Insegna a un router la geografia
Eco	Chiede a una macchina se è "viva"
Risposta a eco	Sì, sono "vivo"
Richiesta di contrassegno temporale	Simile alla richiesta di eco, ma con contrassegno temporale
Risposta alla richiesta di contrassegno temporale	Simile alla risposta eco, ma con contrassegno temporale

Figura 5.61. I principali tipi di messaggi ICMP.

Il messaggio DESTINATION UNREACHABLE è utilizzato quando una sottorete o un router non sono in grado di individuare la destinazione, o quando un pacchetto con il bit *DF* non può essere inoltrato perché lungo il percorso si trova una rete a pacchetti piccoli. Il messaggio TIME EXCEEDED è inviato quando un pacchetto è scartato perché il suo contatore ha raggiunto il valore zero. Questo evento indica che i pacchetti si trovano in un ciclo, che c'è un'enorme congestione o che i valori assegnati ai timer sono troppo bassi. Il messaggio PARAMETER PROBLEM indica che è stato rilevato un valore illegale in un campo dell'intestazione. Questo problema può dipendere da un errore nel software IP dell'host trasmittente, o nel software di un router attraversato.

Il messaggio SOURCE QUENCH era utilizzato in passato per rallentare gli host che trasmettevano troppi pacchetti; quando riceveva questo messaggio, l'host doveva rallentare. Oggi è usato raramente perché in caso di congestione questi pacchetti tendono ad alimentare il fuoco. Il controllo della congestione in Internet è ora eseguito in larga misura nello strato trasporto (i dettagli sono descritti nel Capitolo 6).

Il messaggio REDIRECT è utilizzato quando un router si accorge che un pacchetto sembra essere stato instradato in modo errato. È usato dal router per comunicare all'host trasmittente il probabile errore.

I messaggi ECHO e ECHO REPLY sono utilizzati per scoprire se una certa destinazione è raggiungibile e attiva. Non appena riceve il messaggio ECHO, la destinazione deve rispondere con un messaggio ECHO REPLY. I messaggi TIMESTAMP REQUEST e TIMESTAMP REPLY funzionano in modo simile, ma registrano nella risposta l'ora di arrivo del messaggio e l'ora di invio della risposta. Queste informazioni sono utilizzate per misurare le prestazioni della rete.

Oltre a questi messaggi, ne sono stati definiti anche altri. La lista completa si può scaricare dal sito www.iana.org/assignments/icmp-parameters.

ARP (Address Resolution Protocol)

Anche se ogni macchina di Internet ha uno o più indirizzi IP, in realtà questi non possono essere utilizzati per inviare pacchetti perché l'hardware che opera sullo strato data link non comprende gli indirizzi Internet. Oggi la maggior parte degli host nelle aziende e nelle università è collegata a una LAN attraverso una scheda di rete che conosce solo gli indirizzi LAN. Per esempio, qualsiasi scheda Ethernet incorpora un indirizzo Ethernet a 48 bit. Ogni produttore di schede Ethernet richiede a un'autorità centrale un blocco di indirizzi, per garantire che nessuna coppia di schede utilizzi lo stesso indirizzo (così facendo, nessuna coppia di schede può creare conflitti dentro la LAN). Le schede inviano e ricevono frame basati sugli indirizzi Ethernet a 48 bit, e non sanno nulla degli indirizzi IP a 32 bit. La domanda che sorge spontanea è: in che modo gli indirizzi IP vengono associati agli indirizzi dello strato data link (per esempio agli indirizzi Ethernet)? Per spiegare come funziona il meccanismo è utile esaminare l'esempio rappresentato nella Figura 5.62, che mostra una piccola università contenente diverse reti di classe C (ora chiamate /24). Ci sono due Ethernet, una nella facoltà di Scienze dell'informazione associata all'indirizzo IP 192.31.65.0, e una nella facoltà di Ingegneria elettronica associata all'indirizzo IP 192.31.63.0. Queste sono collegate a un anello principale (per esempio FDDI) che ha l'indirizzo IP 192.31.60.0. I computer sulle Ethernet hanno indirizzi Ethernet unici, indicati da *E1*, *E2*, ...*E6*, mentre quelli sull'anello FDDI hanno indirizzi FDDI indicati dalle etichette *F1*, *F2* e *F3*.

Vediamo prima di tutto in che modo l'utente dell'host 1 invia un pacchetto all'utente dell'host 2. Si supponga che il trasmittente conosca il nome del ricevente desiderato, nome che potrebbe essere per esempio *mary@eagle.cs.uni.edu*. La prima mossa consiste nel cercare l'indirizzo IP dell'host 2 noto come *eagle.cs.uni.edu*. Questa ricerca è eseguita dal DNS (Domain Name System); DNS è esaminato nel Capitolo 7, per il momento è sufficiente supporre che il sistema restituiscia l'indirizzo IP dell'host 2 (192.31.65.5). Il software dello strato superiore sull'host 1 ora costruisce un pacchetto che riporta nel campo *destination address* l'indirizzo IP 192.31.65.5, e lo passa al software IP che si occupa di trasmetterlo. Il software IP esamina l'indirizzo e scopre che la destinazione si trova sulla sua stessa rete, ma ha bisogno di trovare in qualche modo l'indirizzo Ethernet di destinazione.

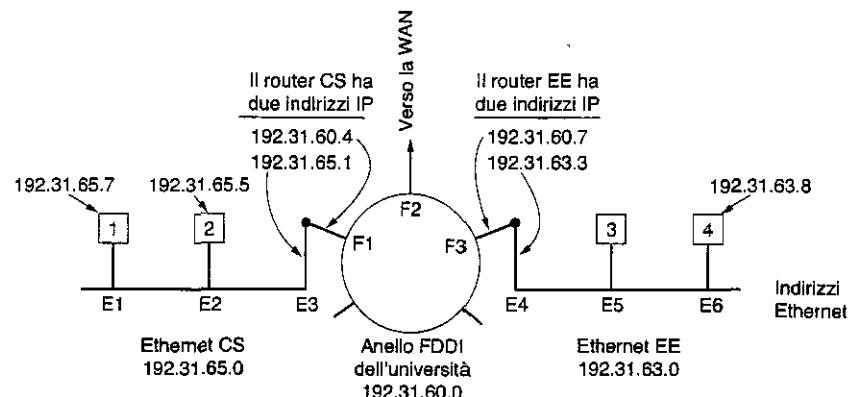


Figura 5.62. Tre reti /24 interconnesse: due Ethernet e un anello FDDI.

Il problema potrebbe essere risolto mediante un file di configurazione, collocato da qualche parte nel sistema, che associa gli indirizzi IP agli indirizzi Ethernet. Questa soluzione è certamente possibile, ma in un'azienda con migliaia di macchine l'aggiornamento di questi file sarebbe soggetto a errori e richiederebbe molto tempo.

Esiste una soluzione migliore. L'host 1 trasmette attraverso la Ethernet un pacchetto broadcast che chiede: chi è il proprietario dell'indirizzo IP 192.31.65.5? La trasmissione broadcast arriva a tutte le macchine della Ethernet 192.31.65.0, e ognuna controlla il proprio indirizzo IP. Solo l'host 2 risponde inviando il proprio indirizzo Ethernet (*E2*). In questo modo, l'host 1 scopre che l'indirizzo IP 192.31.65.5 è assegnato all'host che ha l'indirizzo Ethernet *E2*. Il protocollo utilizzato per fare questa domanda e ottenere una risposta si chiama ARP (Address Resolution Protocol). Quasi tutte le macchine di Internet lo utilizzano. ARP è definito in RFC 826.

Utilizzare ARP al posto dei file di configurazione rende tutto più semplice. Il gestore del sistema non deve fare altro che assegnare a ogni macchina un indirizzo IP e decidere le maschere di sottorete. ARP si occupa di tutto il resto.

A questo punto, il software IP sull'host 1 costruisce un frame Ethernet indirizzato a *E2*, inserisce il pacchetto IP (indirizzato a 192.31.65.5) nel campo carico utile e scarica il tutto sulla Ethernet. La scheda Ethernet dell'host 2 rileva il frame, si accorge di essere il destinatario della comunicazione, preleva i dati e genera un interrupt. Il driver Ethernet estrae il pacchetto IP dal carico utile e lo passa al software IP, che verifica la correttezza dell'indirizzo ed elabora i dati.

Varie ottimizzazioni permettono ad ARP di funzionare in modo più efficiente. Tanto per cominciare, una volta eseguita l'operazione ARP, il computer memorizza in cache il risultato, caso mai dovesse essere necessario ricontattare lo stesso computer. La volta successiva, trovando l'informazione desiderata nella cache, il computer non dovrà generare alcun messaggio broadcast. In molti casi l'host 2 potrebbe aver bisogno di inviare una richiesta per determinare grazie ad ARP l'indirizzo Ethernet del trasmittente; per evitare questa trasmissione ARP broadcast, è sufficiente che l'host 1 includa la propria associazione IP-

Ethernet nel pacchetto ARP. Quando la trasmissione ARP raggiunge l'host 2, la coppia di valori del tipo (192.31.65.7, E1) viene inserita nella cache ARP dell'host 2 in previsione di un utilizzo futuro. In realtà, tutti i computer sulla Ethernet possono inserire queste associazioni nelle proprie cache ARP.

Un'ulteriore miglioramento si ottiene facendo in modo che ogni computer trasmetta in broadcast la propria associazione durante l'accensione. Questa trasmissione broadcast è generalmente eseguita sotto forma di una ricerca ARP mirata al proprio indirizzo IP. L'operazione, che non genera alcuna risposta diretta, ottiene l'effetto di inserire una voce nella cache ARP di tutti gli altri computer. Se arriva una risposta (inattesa), significa che due computer hanno ricevuto lo stesso indirizzo IP, perciò l'ultimo a collegarsi dovrebbe informare l'amministratore di sistema e non avviarsi.

Per consentire la modifica delle associazioni, dovute per esempio a schede Ethernet guaste sostituite con nuove schede (con nuovi indirizzi Ethernet), le voci nella cache ARP dovrebbero scadere dopo pochi minuti.

Si osservi nuovamente la Figura 5.62, e si supponga che questa volta l'host 1 voglia inviare un pacchetto all'host 4 (192.31.63.8). ARP non funziona perché l'host 4 non riceve la trasmissione broadcast (i router non inoltrano i pacchetti broadcast a livello Ethernet). Il problema può essere risolto in due modi. Primo, il router CS potrebbe essere configurato per rispondere alle richieste ARP della rete 192.31.63.0 (e possibilmente delle altre reti locali); se si adotta questa soluzione, l'host 1 crea nella sua cache ARP una voce contenente la coppia di valori (192.31.63.8, E3) e trasmette al router CS tutto il traffico diretto all'host 4. Questa soluzione è chiamata **ARP proxy**. Il secondo modo consiste nel fare in modo che l'host 1 si accorga immediatamente che la destinazione si trova su una rete remota; in questo caso tutto il traffico viene trasmesso a un indirizzo Ethernet predefinito (E3 nella figura) in grado di gestire il traffico remoto. Questa soluzione non richiede che il router CS sappia quali sono le reti remote che sta servendo.

In entrambi i casi, ciò che accade è che l'host 1 impacchetta il pacchetto IP nel campo carico utile di un frame Ethernet indirizzato a E3. Quando riceve il frame Ethernet, il router CS rimuove il pacchetto IP dal carico utile e cerca l'indirizzo IP nelle sue tabelle di routing, scoprendo che i pacchetti per la rete 192.31.63.0 devono essere inviati al router 192.31.60.7. Se non conosce già l'indirizzo FDDI di 192.31.60.7, il router trasmette in modalità broadcast un pacchetto ARP nell'anello e scopre che il suo indirizzo è F3. Quindi inserisce il pacchetto nel campo carico utile di un frame FDDI indirizzato a F3 e immette i dati nell'anello.

Sul router EE, il driver FDDI rimuove il pacchetto dal carico utile e lo passa al software IP, che scopre di dover inviare il pacchetto all'indirizzo 192.31.63.8. Se questo indirizzo non si trova nella sua cache ARP, il dispositivo trasmette in modalità broadcast una richiesta ARP attraverso la Ethernet EE e scopre che l'indirizzo di destinazione è E6, perciò costruisce un frame Ethernet indirizzato a E6, inserisce il pacchetto nel campo carico utile e invia i dati attraverso la Ethernet. Quando il frame Ethernet raggiunge l'host 4, il pacchetto viene estratto dal frame e passato al software IP per l'elaborazione finale.

La trasmissione dall'host 1 verso una rete distante attraverso una WAN funziona essenzialmente nello stesso modo, tranne che questa volta le tabelle del router CS gli dicono di utilizzare il router WAN il cui indirizzo FDDI è F2.

RARP, BOOTP e DHCP

ARP risolve il problema della scoperta dell'indirizzo Ethernet che corrisponde a un dato indirizzo IP. Qualche volta però deve essere risolto il problema inverso: dato un indirizzo Ethernet, qual è il corrispondente indirizzo IP? In particolare, questo problema si presenta all'avvio delle stazioni di lavoro senza dischi interni. Questi computer normalmente ricevono da un file server remoto un'immagine binaria del sistema operativo, perciò come possono scoprire il proprio indirizzo IP? La prima soluzione sviluppata si chiama **RARP** (*Reverse Address Resolution Protocol*), definita in RFC 903. Questo protocollo permette a una stazione di lavoro appena avviata di inviare in modalità broadcast il proprio indirizzo Ethernet insieme al seguente messaggio: "Il mio indirizzo Ethernet a 48 bit è 14.04.05.18.01.25; per caso qualcuno là fuori conosce il mio indirizzo IP?". Il server RARP riceve la richiesta, cerca l'indirizzo Ethernet nei suoi file di configurazione e trasmette una risposta contenente l'indirizzo IP corrispondente. È meglio utilizzare RARP piuttosto che incorporare un indirizzo IP nell'immagine della memoria, perché diventa possibile utilizzare la stessa immagine del sistema operativo per tutti i computer senza dischi. Se l'indirizzo IP fosse incorporato nell'immagine, ogni stazione di lavoro avrebbe bisogno di un'immagine personalizzata. Lo svantaggio di RARP è che utilizza un indirizzo di destinazione composto da tutti 1 (trasmissione broadcast limitata) per raggiungere il server RARP; queste trasmissioni broadcast non sono inoltrate dai router, perciò è necessario installare un server RARP su ogni rete. Per aggirare il problema è stato inventato un protocollo alternativo di bootstrap, chiamato **BOOTP**. A differenza di RARP, BOOTP utilizza messaggi UDP, che sono inoltrati attraverso i router [NdR - in pratica le richieste di BOOTP (e DHCP) vengono effettuate per mezzo di datagrammi UDP con indirizzo di destinazione 255.255.255.255, che si traducono in frame broadcast per la rete fisica locale, esattamente come in RARP]. BOOTP (come DHCP) consente però l'esistenza di *relay agent* (agente di collegamento, uno in ciascuna rete fisica locale), con il compito di intercettare le richieste e rispedirle al server BOOTP (o DHCP) con un normale datagramma UDP unicast (infatti gli agenti di collegamento conoscono l'indirizzo IP del server, e quindi può risiedere in una rete remota). Inoltre fornisce alle stazioni di lavoro senza dischi interni alcune informazioni aggiuntive, tra cui l'indirizzo IP del file server che contiene l'immagine della memoria, l'indirizzo IP del router predefinito e la maschera di sottorete da utilizzare. BOOTP è descritto nei documenti RFC 951, 1048 e 1084. Un problema serio con BOOTP è che richiede la configurazione manuale delle tabelle che associano gli indirizzi IP agli indirizzi Ethernet. Quando si aggiunge alla LAN un nuovo host, non è possibile utilizzare BOOTP fino a quando l'amministratore non assegna alla macchina un indirizzo IP e non inserisce manualmente l'associazione del tipo (indirizzo Ethernet, indirizzo IP) nelle tabelle di configurazione di BOOTP. Per eliminare questo passaggio suscettibile di errori, BOOTP è stato esteso; al protocollo è stato dato un nuovo nome: **DHCP** (*Dynamic Host Configuration Protocol*). DHCP permette l'assegnazione manuale oppure automatica degli indirizzi IP, ed è descritto nei documenti RFC 2131 e 2132. Nella maggior parte dei sistemi, ha ampiamente sostituito RARP e BOOTP. Come RARP e BOOTP, DHCP si basa sull'idea di un server speciale che assegna gli indirizzi IP agli host che ne richiedono uno.

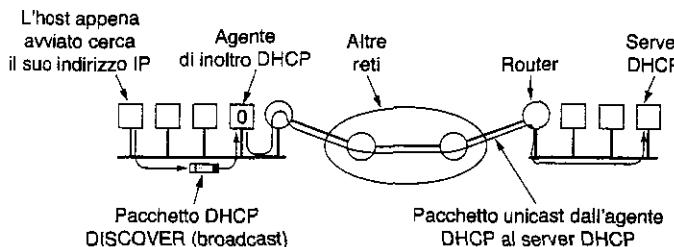


Figura 5.63. Come funziona DHCP.

Questo server non deve trovarsi necessariamente sulla stessa LAN dell'host richiedente. Poiché il server DHCP potrebbe non essere raggiunto dalle trasmissioni broadcast, è necessario installare in ogni LAN un **agente di inoltro DHCP (DHCP relay agent)**, come mostrato nella Figura 5.63. Per trovare il suo indirizzo IP, una macchina appena accesa invia in modalità broadcast un pacchetto DHCP DISCOVER. L'agente di inoltro DHCP presente su quella LAN intercetta tutte le trasmissioni DHCP; quando individua un pacchetto DHCP DISCOVER, l'agente trasmette il pacchetto in modalità unicast al server DHCP, che può anche trovarsi su una rete distante. L'agente di inoltro ha bisogno di una sola informazione: l'indirizzo IP del server DHCP. Un problema che si presenta quando si assegnano automaticamente indirizzi IP estratti da un pool comune riguarda la durata dell'allocazione. Se un host abbandona la rete e non restituisce il proprio indirizzo IP al server DHCP, quell'indirizzo andrà definitivamente perso, perciò dopo un po' di tempo molti indirizzi IP potrebbero non essere più disponibili. Per impedire che questo accada, l'assegnazione dell'indirizzo IP può essere fissata per un periodo di tempo, mediante una tecnica chiamata **leasing**. Poco prima della scadenza del leasing, l'host deve chiedere al server DHCP il rinnovo dell'indirizzo. Se non riesce a fare la richiesta o la richiesta viene rifiutata, l'host non può più utilizzare l'indirizzo IP che gli era stato assegnato precedentemente.

5.6.4 OSPF - Il protocollo di routing per i gateway interni

Dopo aver concluso lo studio dei protocolli di controllo di Internet è giunto il momento di passare all'argomento successivo: il routing in Internet. Come è stato detto precedentemente, Internet è una rete composta da un gran numero di AS (*Autonomous System*). Ogni AS è gestito da un'organizzazione diversa, che può adottare un proprio algoritmo di routing. Per esempio, le reti interne delle aziende X, Y e Z di solito sono viste come tre AS se sono tutte e tre collegate a Internet, e possono utilizzare internamente algoritmi di routing diversi. Tuttavia, anche nel caso del routing interno l'esistenza degli standard semplifica l'implementazione al confine tra gli AS e permette il riutilizzo del codice. Questo paragrafo esamina il routing in un AS, mentre il paragrafo successivo è dedicato al routing tra AS. L'algoritmo di routing adottato in un AS è detto anche **interior gateway protocol** (protocollo di routing per gateway interni); l'algoritmo di routing tra AS è chiamato invece **exterior gateway protocol** (protocollo di routing per gateway esterni). Il protocollo di routing per gateway interni di Internet originario era un protocollo basato sul vettore delle

distanze (RIP), fondato sull'algoritmo Bellman-Ford ereditato da ARPANET. Funzionava bene nei sistemi piccoli, meno bene negli AS più grandi; soffriva del problema del conto all'infinito e aveva generalmente una convergenza lenta, perciò nel maggio 1979 venne sostituito da un protocollo basato sullo stato dei collegamenti. Nel 1988, IETF (Internet Engineering Task Force) iniziò a lavorare sul successore, chiamato **OSPF (Open Shortest Path First)**: il nuovo protocollo è diventato uno standard nel 1990. OSPF, oggi supportato dalla maggior parte dei produttori di router, è diventato il principale protocollo di routing per gateway interni. Questo paragrafo spiega brevemente come funziona. Per maggiori informazioni consultare RFC 2328. Data la lunga esperienza con gli altri protocolli di routing, il gruppo che stava progettando il nuovo protocollo aveva un lungo elenco di requisiti che voleva soddisfare. Prima di tutto l'algoritmo doveva essere pubblicato nella letteratura senza vincoli di brevetto (cioè aperta), da qui la lettera "O" (open, aperta) dell'acronimo OSPF. Una soluzione proprietaria posseduta da un'azienda non sarebbe stata accettabile. Secondo, il nuovo protocollo doveva supportare diverse metriche di distanza, per esempio la distanza fisica, il ritardo e così via. Terzo, doveva essere un algoritmo dinamico, in grado di modificarsi rapidamente e automaticamente in base alla topologia. Quarto (e nuovo per OSPF), doveva supportare il routing basato sul tipo di servizio. Il nuovo protocollo doveva essere in grado di instradare il traffico in tempo reale in un modo e il traffico di altro tipo in un altro modo. Il protocollo IP aveva un campo *type of service*, ma nessun protocollo di routing esistente lo utilizzava. Questo campo fu incluso in OSPF, ma poiché continuava a essere inutilizzato, alla fine venne rimosso. Quinto (e collegato a quanto appena detto), il nuovo protocollo doveva eseguire il bilanciamento del carico, ossia suddividere il carico su più linee. La maggior parte dei protocolli precedenti inviava tutti i pacchetti lungo il percorso migliore; il percorso che viene dopo il migliore non era mai utilizzato. In molti casi, suddividendo il carico su più linee si ottengono prestazioni più alte. Sesto, era necessario supportare i sistemi gerarchici. Nel 1988, Internet era diventata così grande che nessun router poteva conoscerne l'intera topologia. Il nuovo protocollo di routing doveva essere progettato in modo che nessun router avrebbe dovuto farlo. Settimo, era necessario implementare un po' di sicurezza per impedire agli utenti più curiosi di imbrogliare i router inviando loro false informazioni di routing. Infine, era necessario provvedere alla gestione dei router collegati a Internet tramite tunnel. I protocolli precedenti non gestivano bene questa situazione.

OSPF supporta tre tipi di connessioni e di reti:

1. linee punto-punto tra due router
2. reti multiaccesso con trasmissione broadcast (la maggior parte delle LAN)
3. reti multiaccesso senza trasmissione broadcast (la maggior parte delle WAN a commutazione di pacchetto).

Una rete **multiaccesso** è una rete che può contenere più router, ognuno in grado di comunicare direttamente con tutti gli altri. Tutte le LAN e le WAN hanno questa proprietà. La Figura 5.64(a) mostra un AS contenente tutti e tre i tipi di rete. Si noti che gli host generalmente non giocano un ruolo importante in OSPF.

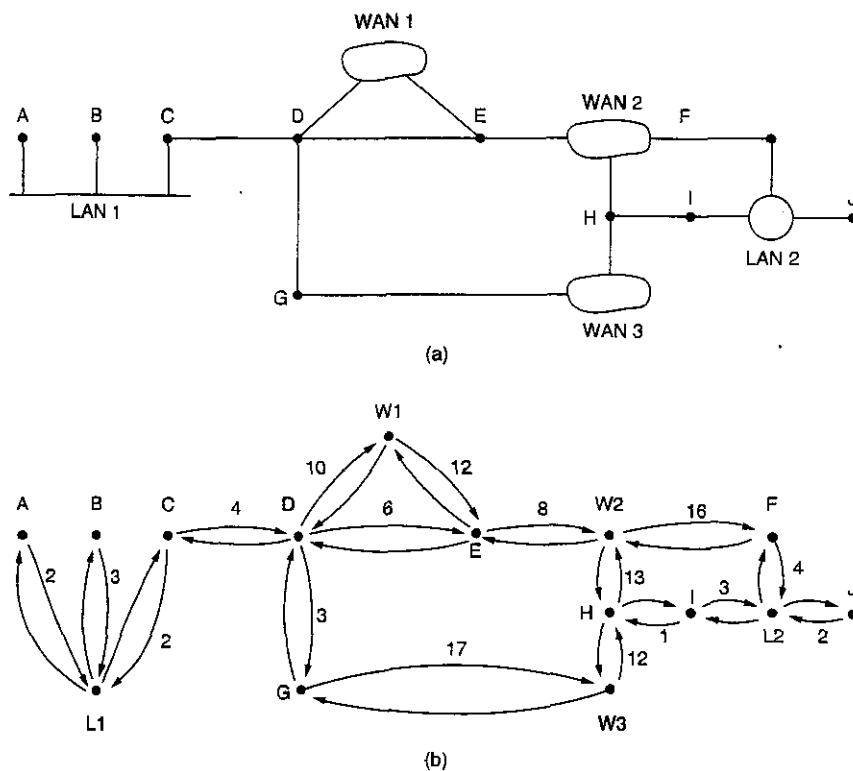


Figura 5.64. (a) Un autonomous system. (b) Una rappresentazione grafica di (a).

OSPF opera riassumendo l'insieme di reti reali, router e linee in un grafo orientato in cui a ogni arco è assegnato un costo (distanza, ritardo e così via), quindi calcola il percorso più breve in base al peso degli archi. Una connessione seriale tra due router è rappresentata da una coppia di archi, uno per ogni direzione; i loro pesi possono essere diversi. Una rete multiaccesso è rappresentata da un nodo per la rete stessa più un nodo per ogni router. Gli archi dal nodo di rete ai router hanno peso 0 e sono omessi dal grafico. La Figura 5.64(b) mostra la rappresentazione grafica della rete rappresentata nella Figura 5.64(a). I pesi sono simmetrici, a meno che non sia indicato il contrario. Ciò che OSPF fondamentalmente fa è rappresentare la rete reale come un grafo simile a questo, e poi elaborare il percorso più breve da ogni router a ogni altro router. Molti AS di Internet sono a loro volta grandi e difficili da gestire. OSPF permette di dividere tali AS in più aree, dove ogni area è una rete o un insieme di reti contigue. Le aree non si sovrappongono e non hanno bisogno di essere complete, cioè alcuni router possono non appartenere ad alcuna area. Un'area è una generalizzazione di una sottorete. Dall'esterno, la topologia e i dettagli dell'area non sono visibili. Ogni AS ha un'area dorsale chiamata area 0. Tutte le aree sono collegate alla dorsale, per esempio con tunnel, perciò è possibile passare da un'area a qualunque altra area dello stesso AS passando attraverso la dorsale.

Un tunnel è rappresentato nel grafico come un arco che ha un costo. Ogni router collegato a due o più aree fa parte della dorsale. Come nel caso delle altre aree, la topologia della dorsale non è visibile all'esterno della stessa.

Dentro un'area, ogni router ha lo stesso database degli stati dei collegamenti ed esegue lo stesso algoritmo di individuazione del percorso più breve; il suo compito principale è calcolare il percorso più breve da sé stesso a ogni altro router nell'area, incluso il router che è collegato alla dorsale (ci deve essere almeno un router di questo tipo). Un router che collega due aree ha bisogno dei database di entrambe le aree, e deve eseguire separatamente un diverso algoritmo di ricerca del percorso più breve per ogni area.

Durante il funzionamento normale, possono essere necessari tre tipi di percorsi: interni all'area, tra aree e tra AS. I percorsi interni all'area sono i più semplici, poiché il router sorgente conosce già il percorso più breve diretto al router di destinazione. Il routing tra aree segue sempre tre fasi: dalla sorgente alla dorsale; attraverso la dorsale fino all'area di destinazione; infine alla destinazione. Questo algoritmo crea su OSPF una configurazione a stella, con la dorsale che funge da hub e le altre aree che fungono da raggi. I pacchetti sono istradati dalla sorgente alla destinazione così come sono, perciò non sono incapsulati né trasmessi attraverso tunnel, a meno che l'area di destinazione non si colleghi alla dorsale attraverso un tunnel. La Figura 5.65 mostra parte di Internet con AS e aree.

OSPF distingue quattro classi di router:

1. i router interni (*internal router*) sono quelli che si trovano completamente dentro un'area
2. i router di confine (*border router*) collegano due o più aree
3. i router di dorsale (*backbone router*) sono sulla dorsale
4. i router sul confine dell'AS (*boundary router*) comunicano con i router che si trovano in altri AS.

Queste classi possono sovrapporsi. Per esempio, tutti i router di confine fanno automaticamente parte della dorsale; inoltre, un router che si trova sulla dorsale ma che non fa parte di nessun'altra area è anche un router interno. La Figura 5.65 mostra esempi di tutte e quattro le classi di router.

All'avvio, il router trasmette messaggi HELLO attraverso tutte le sue linee punto-punto e comunica questi messaggi in modalità multicast sulle LAN, ai gruppi composti da tutti gli altri router. Sulle WAN, per individuare chi deve essere contattato, il router ha bisogno di alcune informazioni di configurazione. In base alle risposte, ogni router scopre chi sono i suoi vicini. I router sulla stessa LAN sono tutti vicini.

OSPF funziona scambiando informazioni tra router adiacenti, il che non equivale a dire tra router vicini. In particolare, è inefficiente che ogni router su una LAN comunichi con ogni altro router sulla LAN. Per evitare questa situazione, un router è nominato **router designato**. Questo dispositivo, che è considerato **adiacente** a tutti gli altri router della sua LAN, scambia informazioni con gli altri apparecchi. I router confinanti che non sono adiacenti non scambiano informazioni gli uni con gli altri. Un router di backup viene costantemente aggiornato per facilitare la transizione in caso di blocco del router designato principale.

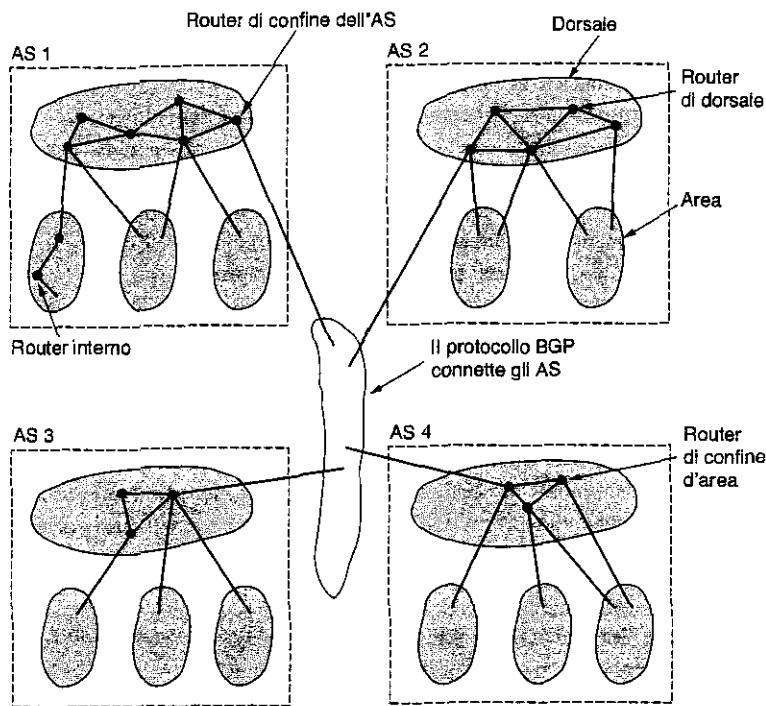


Figura 5.65. La relazione tra SA differenti, dorsali e aree in OSPF.

Durante le normali operazioni, ogni router invia periodicamente in flooding messaggi LINK STATE UPDATE a ognuno dei suoi router adiacenti. Questo messaggio comunica lo stato del dispositivo e i costi utilizzati nel database della topologia.

La ricezione di acknowledgement rende questi messaggi affidabili. Ogni messaggio ha un numero di sequenza, perciò un router può scoprire se un LINK STATE UPDATE appena arrivato è più vecchio o più nuovo di quello già posseduto. I router inviano questi messaggi anche quando una linea si attiva o si disattiva, e quando il suo costo cambia.

I messaggi DATABASE DESCRIPTION contengono i numeri di sequenza di tutte le voci relative allo stato dei collegamenti possedute in quel momento dal trasmittente. Confrontando i propri valori con quelli inviati dal router trasmittente, il ricevente può scoprire chi possiede i valori più recenti. Questi messaggi sono utilizzati quando una linea viene attivata.

Ognuno dei partner può richiedere all'altro alcune informazioni sullo stato dei collegamenti, usando i messaggi LINK STATE REQUEST. Il risultato di questo algoritmo è che ogni coppia di router adiacenti si controlla per vedere chi ha i dati più recenti, e in questo modo le nuove informazioni si propagano attraverso l'area.

Tipo di messaggio	Descrizione
Presentazione	Utilizzato per scoprire chi sono i vicini
Aggiornamento stato del collegamento	Comunica ai vicini i costi del trasmittente
Conferma stato del collegamento	Conferma l'aggiornamento dello stato del collegamento
Descrizione del database	Annuncia gli aggiornamenti posseduti dal trasmittente
Richiesta stato del collegamento	Richiede informazioni al partner

Figura 5.66. I cinque tipi di messaggi OSPF.

Tutti questi messaggi sono inviati come pacchetti IP grezzi. La Figura 5.66 riepiloga i cinque tipi di messaggi.

Finalmente possiamo rimettere insieme tutti i pezzi. Il flooding permette a ogni router di comunicare a tutti gli altri router della sua area chi sono i vicini e i costi utilizzati. Queste informazioni consentono a ogni router di costruire il grafo dell'area e di calcolare il percorso più corto. L'area dorsale fa lo stesso. Inoltre, i router della dorsale accettano informazioni dai router di confine d'area in modo da poter calcolare il percorso migliore diretto da ogni router di dorsale a ogni altro router. Queste informazioni si propagano indietro verso i router di confine d'area, che le annunciano dentro le loro aree. Usando questi dati, un router che sta per inviare un pacchetto da un'area a un'altra può selezionare il router di uscita migliore verso la dorsale.

5.6.5 BGP – Il protocollo di routing per i gateway esterni

Dentro un singolo AS, il protocollo di routing raccomandato è OSPF (sebbene non sia certamente l'unico utilizzato). Tra AS si utilizza un protocollo diverso chiamato **BGP** (*Border Gateway Protocol*). Tra AS è necessario un protocollo diverso perché gli obiettivi sono differenti. Un protocollo per gateway interni non deve far altro che spostare i pacchetti nel modo più efficiente possibile dalla sorgente alla destinazione; non deve preoccuparsi della politica. I router che gestiscono il protocollo di routing per gateway esterni devono considerare anche la politica (Metz, 2001). Per esempio, un AS aziendale potrebbe voler inviare pacchetti a qualunque sito di Internet e ricevere pacchetti da qualunque altro sito di Internet; ma potrebbe anche non essere disposto a trasportare pacchetti generati in un AS straniero e diretti a un altro AS straniero, anche quando il suo AS si trova sul percorso più corto tra i due AS stranieri ("Il problema è loro, non nostro"). D'altro canto, potrebbe essere pronto a trasportare il traffico di passaggio per i suoi vicini o anche per altri AS specifici che hanno pagato per questo servizio. Le aziende telefoniche, per esempio, potrebbero essere felici di fungere da portanti per i loro clienti ma non per i clienti di altre aziende. Il protocollo di gateway esterno in generale, e BGP in particolare, è stato progettato per consentire il rispetto di molti tipi di criteri di routing nel traffico tra AS.

Criteri tipici coinvolgono considerazioni politiche, di sicurezza o economiche. Ecco alcuni esempi di vincoli applicati al routing:

1. nessun traffico di passaggio attraverso certi AS
2. mai mettere l'Iraq su un percorso che inizia dal Pentagono
3. non utilizzare gli Stati Uniti per andare dalla British Columbia all'Ontario
4. passare per l'Albania solo se non esistono alternative per la destinazione
5. il traffico generato o in arrivo a IBM non dovrebbe passare attraverso Microsoft.

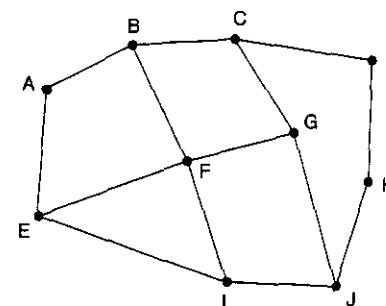
In genere i criteri sono configurati manualmente in ogni router BGP (o inseriti usando qualche tipo di script), e non fanno parte del protocollo stesso.

Dal punto di vista del router BGP, il mondo è composto da AS e da linee che li collegano. Due AS sono considerati collegati se esiste una linea che unisce un router di confine del primo AS con un router di confine del secondo AS. Dato lo speciale interesse di BGP verso il traffico di passaggio, le reti sono raggruppate in tre categorie. La prima categoria è costituita dalle **stub networks** (reti terminali), che hanno una sola connessione al grafo BGP; queste reti non possono essere utilizzate per il traffico di passaggio perché non c'è nessuna rete dall'altra parte. La seconda categoria è composta dalle **multiconnected networks** (reti multicollegate); queste potrebbero essere utilizzate per il traffico di passaggio, se non rifiutano di farlo. Infine, ci sono le **transit networks** (reti di transito) come le dorsali, che sono pronte a gestire pacchetti di terze parti, possibilmente con alcune restrizioni e di solito a pagamento.

Copie di router BGP comunicano le une con le altre stabilendo connessioni TCP; questo tipo di meccanismo rende la comunicazione affidabile e nasconde tutti i dettagli della rete attraversata.

BGP è fondamentalmente un protocollo basato sul vettore delle distanze, abbastanza diverso però dalla maggior parte degli altri protocolli, come RIP. Invece di conservare solo il costo di ogni destinazione, ogni router BGP tiene traccia del percorso utilizzato. Analogamente, invece di comunicare periodicamente a ogni vicino il costo stimato associato a ogni possibile destinazione, ogni router BGP comunica l'esatto percorso in uso. Come esempio, si considerino i router BGP mostrati nella Figura 5.67(a). In particolare, si consideri la tabella di routing di *F* e si supponga che questo apparecchio utilizzi il percorso *FGCD* per raggiungere *D*. Quando i vicini passano a *F* le informazioni di routing, forniscono i loro percorsi completi come mostrato nella Figura 5.67(b) (per semplicità, la figura mostra solo la destinazione *D*).

Dopo aver ricevuto dai vicini tutti i percorsi che stanno usando, *F* esamina le informazioni per identificare il percorso migliore. Rapidamente scarta i percorsi da *I* e *E*, poiché passano attraverso lo stesso *F*; la scelta è tra *B* e *G*. Ogni router BGP contiene un modulo che esamina i percorsi diretti a una data destinazione e assegna loro un punteggio, restituendo un numero che rappresenta la "distanza" associata a quella destinazione per ogni percorso. Ogni percor-



Le informazioni relative a *D* che *F* riceve dai suoi vicini

- Da *B*: "Io uso BCD"
- Da *G*: "Io uso GCD"
- Da *I*: "Io uso IFGCD"
- Da *E*: "Io uso EFGCD"

Figura 5.67. (a) Un insieme di router BGP. (b) Informazioni inviate a *F*.

so che viola un criterio di vincolo riceve automaticamente un punteggio infinito. Il router quindi adotta il percorso che ha la distanza minore. La funzione che valuta il punteggio dei percorsi non fa parte del protocollo BGP, e può essere scelta dagli amministratori del sistema. BGP risolve facilmente il problema del conto all'infinito che affligge gli altri algoritmi di routing basati sul vettore delle distanze. Per esempio, si supponga che *G* si rompa o che la linea *FG* si interrompa. *F* allora riceve i percorsi dai vicini superstizi; questi percorsi sono *BCD*, *IFGCD* e *EFGCD*. *F* si accorge immediatamente che gli ultimi due percorsi sono privi di senso, poiché passano attraverso *F*, perciò sceglie *FBCD* come suo nuovo percorso. Altri algoritmi basati sul vettore delle distanze spesso fanno la scelta sbagliata perché non sono in grado di capire quali vicini hanno percorsi indipendenti associati alla destinazione e quali invece no. La definizione di BGP si trova nei documenti RFC da 1771 a 1774.

5.6.6 Internet multicasting

La normale comunicazione IP avviene tra un trasmittente e un ricevente, ma per alcune applicazioni è utile che un processo possa comunicare contemporaneamente con un gran numero di ricevitori. Alcuni esempi sono l'aggiornamento di database distribuiti e replicati, la trasmissione dei valori delle azioni a più agenti di cambio e la gestione di audio-conferenze digitali.

IP supporta la trasmissione multicast, usando gli indirizzi di classe D. Ogni indirizzo di classe D identifica un gruppo di host e 28 bit sono utilizzati per identificare i gruppi, perciò possono esistere nello stesso momento più di 250 milioni di gruppi. Quando un processo invia un pacchetto a un indirizzo di classe D, viene fatto un tentativo best effort per trasmettere i dati a tutti i membri del gruppo di destinazione, ma non viene data alcuna garanzia. Alcuni membri possono non ricevere il pacchetto.

Sono supportati due tipi di indirizzi di gruppo: permanenti e temporanei. Un gruppo permanente è sempre presente, non deve essere impostato e possiede un indirizzo di gruppo permanente. Alcuni esempi di indirizzi di gruppo permanente sono:

- 224.0.0.1 tutti gli apparati su una LAN
- 224.0.0.2 tutti i router su una LAN
- 224.0.0.5 tutti i router OSPF su una LAN
- 224.0.0.6 tutti i router OSPF designati su una LAN.

I gruppi temporanei devono essere creati prima di poter essere utilizzati. Un processo può chiedere al suo host di unirsi a un gruppo specifico oppure può chiedergli di abbandonarlo. Quando l'ultimo processo su un host lascia un gruppo, quel gruppo non è più presente sull'host. Ogni host tiene traccia dei gruppi di appartenenza correnti dei propri processi. La trasmissione multicast è implementata da speciali router multicast, che eventualmente possono essere abbinati ai router standard. Circa una volta al minuto, ogni router multicast genera una trasmissione hardware multicast (sullo strato data link) diretta agli host sulla sua LAN (indirizzo 224.0.0.1) che chiede alle macchine di indicare i gruppi di appartenenza dei loro processi. Ogni host risponde comunicando tutti gli indirizzi di classe D che gli interessano. Questi pacchetti di interrogazione e risposta utilizzano un protocollo chiamato **IGMP** (*Internet Group Management Protocol*), che assomiglia vagamente a ICMP. Ha solo due tipi di pacchetti: interrogazione e risposta, ognuno con un formato semplice prefissato, che contiene alcune informazioni di controllo nella prima word del campo carico utile e un indirizzo di classe D nella seconda word. Il protocollo è descritto nel documento RFC 1112.

Il routing multicast è eseguito usando le strutture spanning tree. Ogni router multicast scambia informazioni con i suoi vicini usando un protocollo basato sul vettore delle distanze; ciò consente a ogni router di costruire per ogni gruppo uno spanning tree che copre tutti i membri del gruppo. Si usano svariati meccanismi di ottimizzazione per sfondare la struttura ad albero ed eliminare i router e le reti che non sono interessati a particolari gruppi. Il protocollo usa in modo pesante i tunnel per evitare di infastidire i nodi che non fanno parte dello spanning tree.

5.6.7 Mobile IP

Molti utenti di Internet usano computer portatili e vogliono rimanere collegati quando visitano un sito Internet remoto, anche mentre si spostano verso quel sito. Sfortunatamente, il sistema di indirizzamento IP rende il lavoro lontano da casa più semplice da spiegare che da realizzare. Questo paragrafo esamina il problema e la soluzione. Una descrizione più dettagliata è riportata in (Perkins, 1998a).

Il vero nemico, in questo caso, è proprio lo schema di indirizzamento. Ogni indirizzo IP contiene un numero di rete e un numero host. Per esempio, si consideri la macchina associata all'indirizzo IP 160.80.40.20/16. Il numero di rete è rappresentato da 160.80 (8272 in formato decimale); 40.20 rappresenta la parte host (10260 in formato decimale). Tutti i router del mondo hanno tabelle di routing che indicano quale linea utilizzare per raggiungere la rete 160.80. Ogni volta che riceve un pacchetto destinato a un indirizzo IP espresso nella forma 160.80.xxx.yyy, il router invia i dati attraverso quella linea.

Se improvvisamente la macchina con quell'indirizzo si spostasse in un sito lontano, i pacchetti diretti a essa continuerebbero a essere instradati verso la LAN (o router) domestici. Il proprietario in questo caso non riceverebbe più la posta elettronica, e così via. Dare alla macchina un nuovo indirizzo IP corrispondente alla sua nuova locazione non è fattibile, perché

troppe persone, programmi e database dovrebbero essere informati del cambiamento. In alternativa si potrebbe fare in modo che i router utilizzino per l'instradamento degli indirizzi IP completi, invece della sola indicazione della rete; questa strategia, però, richiederebbe la costruzione di tabelle con milioni di voci in ogni router, di certo un costo astronomico per Internet.

Quando gli utenti hanno iniziato a chiedere di poter collegare i loro computer portatili a Internet da qualunque luogo, IETF ha chiesto a un gruppo di lavoro di trovare una soluzione. Il gruppo di lavoro ha rapidamente formulato diversi obiettivi considerati desiderabili in ogni situazione. I principali sono elencati di seguito:

1. ogni host mobile deve poter utilizzare il proprio indirizzo IP domestico ovunque si trovi
2. non è consentito apportare modifiche al software degli host fissi
3. non è consentito apportare modifiche al software dei router e alle loro tabelle
4. alla maggior parte dei pacchetti diretti agli host mobili non dovrebbe essere consentito di deviare dal percorso
5. quando si trova a casa, l'host mobile non dovrebbe subire alcun overhead.

La soluzione scelta è quella descritta nel Paragrafo 5.2.8. In breve, ogni sito che vuole consentire ai suoi utenti di girovagare deve creare un agente fisso (*home agent*), e ogni sito che desidera supportare visitatori deve creare un agente esterno (*foreign agent*). Quando si presenta in un sito straniero, l'host mobile deve contattare l'agente esterno locale e registrarsi; l'agente esterno quindi contatta l'agente fisso dell'utente dandogli un **indirizzo temporaneo**, normalmente rappresentato dal proprio indirizzo IP.

Quando un pacchetto raggiunge la LAN domestica dell'utente, i dati raggiungono qualche router collegato alla LAN. Il router tenta di individuare l'host nel solito modo, trasmettendo in modalità broadcast un pacchetto ARP che chiede, per esempio: "Qual è l'indirizzo Ethernet di 160.80.40.20?". L'agente fisso risponde alla domanda dando il proprio indirizzo Ethernet. Il router invia all'agente fisso i pacchetti diretti a 160.80.40.20; l'agente, in cambio, inoltra i dati attraverso un tunnel all'indirizzo temporaneo, incapsulandoli nel campo carico utile di un pacchetto IP indirizzato all'agente esterno. L'agente esterno estrae i dati e li passa all'indirizzo dello strato data link dell'host mobile. L'agente fisso comunica l'indirizzo temporaneo al trasmittente, così da permettergli di inviare i pacchetti successivi direttamente all'agente esterno tramite un tunnel. Questa soluzione soddisfa tutti i requisiti elencati in precedenza.

Vale la pena menzionare un piccolo dettaglio. Quando l'host mobile si sposta, il router probabilmente ha il suo indirizzo Ethernet (presto non più valido) memorizzato nella sua cache. La sostituzione di quell'indirizzo Ethernet con quello dell'agente fisso è eseguita attraverso una tecnica chiamata **ARP gratuito**. Questo è un messaggio speciale (non sollecitato) diretto al router che costringe il dispositivo a sostituire una voce specifica della cache, in questo caso quella dell'host mobile che sta per lasciare la rete. Quando l'host mobile in un secondo momento ritorna nella rete, la stessa tecnica è utilizzata per aggiornare di nuovo la cache del router.

Nell'architettura niente impedisce all'host mobile di essere il proprio agente esterno, ma questo approccio funziona solo se l'host mobile (nella sua veste di agente esterno) è connesso logicamente a Internet presso il suo sito corrente. Inoltre, l'host mobile deve essere in grado di acquisire un indirizzo IP temporaneo da utilizzare, e tale indirizzo IP deve appartenere alla LAN alla quale è correntemente collegato.

La soluzione che IETF ha adottato per gli host mobili risolve diversi altri problemi non menzionati in precedenza. Per esempio, come sono individuati gli agenti? La soluzione è semplice: ogni agente periodicamente trasmette in modalità broadcast il proprio indirizzo e il tipo di servizio che è disposto a offrire (per esempio domestico, straniero o entrambi). Al suo arrivo un host mobile può semplicemente ascoltare questi messaggi broadcast, chiamati **avvisi** (*advertisement*), oppure può trasmettere un pacchetto broadcast che annuncia il suo arrivo e sperare che l'agente esterno locale risponda al messaggio.

IETF ha dovuto risolvere anche un altro problema, quello relativo agli host mobili maleducati che se ne vanno senza salutare. La soluzione, in questo caso, è stata di rendere la registrazione valida solo per un intervallo di tempo prefissato. Se non è aggiornata periodicamente scade e l'agente esterno può cancellare le sue tabelle.

Ancora un altro problema riguarda la sicurezza. Quando l'agente fisso riceve un messaggio che gli chiede di inoltrare tutti i pacchetti di Roberta a un altro indirizzo IP, dovrebbe assecondare la richiesta solo quando è assolutamente certo che la richiesta viene da Roberta, e nessun altro sta tentando di impersonare quell'utente. Per gestire queste situazioni si utilizzano protocolli di autenticazione critografica (descritti nel Capitolo 8).

Un punto finale affrontato dal gruppo di lavoro di IETF riguarda i livelli di mobilità. Si immagini un aeroplano con una Ethernet a bordo usata per i computer di navigazione e i dispositivi avionici. Su questa Ethernet c'è un router standard che comunica con la rete Internet cablata terrestre, attraverso un collegamento radio. Un bel giorno, un dirigente del marketing più scaltri degli altri decide di far installare connettori Ethernet in tutti i sedili dei passeggeri per consentire a chi viaggia di collegare alla Ethernet il proprio computer portatile.

La situazione crea due livelli di mobilità: quello dei computer dell'aereo, che sono stazionari rispetto alla Ethernet, e i computer dei passeggeri, che sono mobili rispetto alla rete locale. Inoltre, il router a bordo è mobile rispetto ai router terrestri. La mobilità rispetto a un sistema che a sua volta è mobile può essere gestita usando tunnel ricorsivi.

5.6.8 IPv6

Sebbene CIDR e NAT possano far guadagnare ancora qualche anno, tutti si rendono conto che l'attuale forma di IP (IPv4) ha i giorni contati. Oltre ai problemi tecnici, si profila sullo sfondo un'altra questione. Nei suoi primi anni di vita, Internet era utilizzata dalle università, dall'industria high-tech e dal Governo degli Stati Uniti (specialmente dal dipartimento della difesa). Con l'esplosione dell'interesse verso Internet iniziata a metà degli anni '90, la rete iniziò a essere utilizzata da altri gruppi di persone, che in particolare hanno requisiti differenti. Tanto per dirne una, molta gente con portatili wireless utilizza Internet per rimanere in contatto con la propria sede centrale. In secondo luogo, con l'imminente convergenza tra computer, comunicazione e industria dell'intrattenimento, fra non molto ogni telefono e televisore del mondo potrebbe trasformarsi in un nodo di Internet, produ-

cendo un miliardo di macchine in grado di riprodurre audio e video a richiesta. Dati questi fatti, divenne chiaro che IP doveva evolversi e diventare più flessibile.

Vedendo questi problemi all'orizzonte, IETF nel 1990 iniziò a lavorare a una nuova versione di IP che non avrebbe mai esaurito i suoi indirizzi, che avrebbe risolto una varietà di altri problemi e che sarebbe stata ancora più flessibile ed efficiente. I suoi obiettivi principali erano:

1. supportare miliardi di host, anche con un'allocazione inefficiente dello spazio di indirizzi
2. ridurre la dimensione delle tabelle di routing
3. semplificare il protocollo, per consentire ai router di elaborare più rapidamente i pacchetti
4. offrire una sicurezza migliore (autenticazione e riservatezza) di quella fornita dall'attuale IP
5. fare più attenzione al tipo di servizio, in particolare ai dati in tempo reale
6. aiutare la trasmissione multicast permettendo di specificare gli ambiti
7. dare all'host la possibilità di spostarsi senza cambiare indirizzo
8. permettere al protocollo di evolversi in futuro
9. consentire ai protocolli vecchi e a quelli nuovi di coesistere per diversi anni.

Per sviluppare un protocollo in grado di soddisfare tutti questi requisiti, IETF fece un appello per proposte e discussioni nel documento RFC 1550. Vennero ricevute ventuno risposte, non tutte complete. Nel dicembre 1992, sette proposte serie furono messe sul tavolo. Furono disposte in ordine, da quella che apportava il minor numero di correzioni a IP a quella che sostituiva completamente il vecchio IP con un protocollo completamente differente. Una proposta fu di eseguire TCP su CLNP, che con i suoi indirizzi a 160 bit avrebbe fornito spazio sufficiente in eterno e avrebbe unificato i due principali protocolli dello strato network. Molte persone, però, pensavano che questo sarebbe stato come ammettere che qualcosa nel mondo OSI era stata fatta in modo davvero corretto, affermazione considerata politicamente scorretta nei circoli di Internet. CLNP era modellato strettamente su IP, perciò i due non erano molto diversi. In effetti, il protocollo che alla fine è stato scelto differisce da IP molto più di quanto faccia CLNP. Un altro attacco contro CLNP prese di mira il suo scarso supporto dei tipi di servizio, caratteristica richiesta per trasmettere in modo efficiente dati multimediali.

Tre delle migliori proposte furono pubblicate in *IEEE Network* (Deering, 1993; Francis, 1993; e Katz and Ford, 1993). Dopo molte discussioni, revisioni e manovre per guadagnare una posizione, venne selezionata una versione che combinava le proposte fatte da Deering e Francis, oggi chiamata **SIPP** (*Simple Internet Protocol Plus*) e indicata come **IPv6** [NdR - in realtà SIPP era il nome di un protocollo (frutto di fusioni di altre proposte) che era stato proposto per IPng; quando è stato scelto come soluzione per la nuova versione del protocollo IP è stato adottato con il nome di IPv6, ma sono stati anche effettuati alcuni cambiamenti alle specifiche di SIPP (es., gli indirizzi sono passati da 64 a 128 bit)].

IPv6 soddisfa i requisiti abbastanza bene. Mantiene le buone caratteristiche di IP, abbandona o migliora quelle cattive e ne aggiunge di nuove dove occorre. In generale, IPv6 non è compatibile con IPv4, ma è compatibile con gli altri protocolli Internet ausiliari, inclusi TCP, UDP, ICMP, IGMP, OSPF, BGP e DNS, qualche volta dopo piccole modifiche (solitamente necessarie per gestire indirizzi più lunghi). Le caratteristiche principali di IPv6 sono descritte in questo paragrafo; maggiori informazioni sono contenute nei documenti RFC da 2460 a 2466. Prima di tutto, IPv6 ha indirizzi più lunghi di IPv4. Essendo lunghi 16 byte, soddisfano il problema fondamentale che IPv6 deve risolvere: fornire una scorta di indirizzi Internet praticamente illimitata. Fra poco si parlerà in dettaglio di questi indirizzi. Il secondo grande miglioramento di IPv6 è rappresentato dalla semplificazione dell'intestazione. Contiene solo sette campi (contro i 13 di IPv4).

Questa modifica permette ai router di elaborare i pacchetti più velocemente e di migliorare in tal modo la capacità di trasporto e il ritardo. Fra poco si parlerà anche di intestazioni. Il terzo grande miglioramento è rappresentato dal migliore supporto per le opzioni. Questa modifica era essenziale con la nuova intestazione, perché campi che prima erano necessari ora sono diventati opzionali. Inoltre, il modo in cui le opzioni sono rappresentate è diverso e aiuta i router a tralasciare le opzioni non progettate per loro.

Questa caratteristica accelera l'elaborazione dei pacchetti. Un quarto settore in cui IPv6 fa un grande passo in avanti è quello della sicurezza. C'era la forte sensazione che qualcosa doveva essere fatto per migliorare la sicurezza. Autenticazione e riservatezza sono le chiavi del nuovo IP. Poiché queste caratteristiche sono state introdotte successivamente anche in IPv4, le differenze nell'area della sicurezza non sono più così grandi. Infine, è stata data più attenzione alla qualità del servizio. Diversi timidi tentativi erano stati fatti in passato, ma con la crescita delle applicazioni multimediali su Internet, il senso di urgenza è più grande.

L'intestazione principale di IPv6

La Figura 5.68 mostra l'intestazione IPv6. Il campo *version* contiene sempre il valore 6 associato a IPv6 (4 è il valore associato a IPv4). Durante il periodo di transizione da IPv4, che probabilmente si protrarrà per una decina di anni, i router saranno in grado di esaminare questo campo e identificare il tipo di pacchetto in transito. È interessante notare che questo test fa sprecare alcune istruzioni nel percorso critico, perciò molte implementazioni probabilmente tenteranno di non eseguirlo, usando un campo nell'intestazione data link che distingue i pacchetti IPv4 dai pacchetti IPv6; in tal modo, i pacchetti possono essere passati direttamente al gestore di strato network corretto.

Rendere lo strato data link consapevole del tipo di pacchetto di rete viola però completamente la regola essenziale che afferma che ogni strato non dovrebbe essere a conoscenza del significato dei bit ricevuti dallo strato sovrastante. I dibattiti tra i sostenitori del "Fallo bene" e del "Fallo velocemente" saranno senza dubbio lunghi e vigorosi. Il campo *traffic class* (classe del traffico) è utilizzato per distinguere i pacchetti con diversi requisiti di distribuzione in tempo reale. Un campo progettato per questo scopo esiste in IP fin dall'inizio, ma è stato implementato solo sporadicamente dai router. Sono in corso esperimenti per determinare il miglior utilizzo per la distribuzione dei dati multimediali.

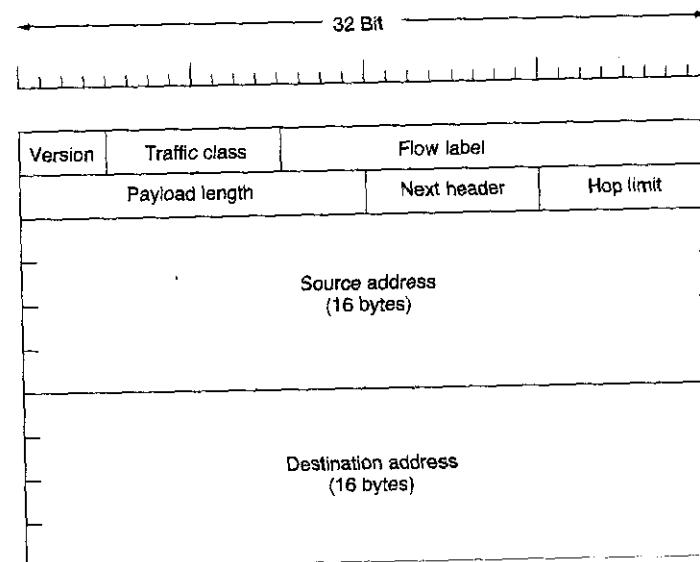


Figura 5.68. L'intestazione fissa IPv6 obbligatoria.

Anche il campo *flow label* (etichetta di flusso) è ancora sperimentale, ma sarà utilizzato per consentire a una sorgente e a una destinazione di impostare una pseudoconnessione con particolari proprietà e requisiti. Per esempio, un flusso di pacchetti generato da un processo su un particolare host sorgente e diretto a un particolare processo su un particolare host di destinazione potrebbe avere rigorosi requisiti di ritardo e quindi aver bisogno di banda riservata. Il flusso può essere impostato in anticipo e gli si può assegnare un identificatore. Quando appare un pacchetto con *flow label* diversa da zero, tutti i router cercano l'etichetta nelle loro tabelle interne per scoprire il tipo di trattamento speciale richiesto dal pacchetto. In realtà, i flussi sono un tentativo di combinare la flessibilità della rete a datagrammi e le garanzie di una sottorete a circuito virtuale.

Ogni flusso è rappresentato dall'indirizzo sorgente, dall'indirizzo di destinazione e dal numero di flusso, perciò tra una data coppia di indirizzi IP possono essere attivi molti flussi. Inoltre, in questo modo, anche se due flussi provenienti da host diversi ma con la stessa etichetta di flusso passano attraverso lo stesso router, il router sarà in grado di distinguergli usando gli indirizzi di origine e di destinazione. Poiché ci si aspetta che le etichette di flusso siano scelte a caso e non siano assegnate in modo sequenziale partendo da 1, i router devono eseguire su di esse un'operazione di hash.

Il campo *payload lenght* (lunghezza del carico utile) indica il numero di byte che seguono l'intestazione di 40 byte mostrata nella Figura 5.68. È stato abbandonato il nome del campo *total lenght* adottato da IPv4 perché il significato è leggermente cambiato: i 40 byte di intestazione non sono più conteggiati nel calcolo della lunghezza.

Il campo *next header* (intestazione successiva) fa trapelare un segreto. Il motivo per cui è stato possibile semplificare l'intestazione è che ci possono essere altre intestazioni estese

(opzionali). Questo campo indica quale delle sei (per il momento) intestazioni estese, se presente, segue l'intestazione corrente. Se questa intestazione è l'ultima intestazione IP, il campo *next header* indica il gestore del protocollo di trasporto (TCP o UDP) al quale va passato il pacchetto.

Il campo *hop limit* è utilizzato per impedire ai pacchetti di vivere per sempre. In pratica, funziona come il campo *time to live* di IPv4, vale a dire che è decrementato a ogni salto del pacchetto. In teoria, in IPv4 il valore rappresentava una durata espressa in secondi, ma nessun router utilizzava l'etichetta in quel modo, perciò il nome è stato modificato per riflettere il modo in cui è realmente utilizzato.

Gli ultimi campi rappresentano l'indirizzo sorgente (*source address*) e l'indirizzo di destinazione (*destination address*). La proposta originale di Deering (SIP) utilizzava indirizzi a 8 byte, ma durante il processo di revisione in molti pensavano che IPv6 avrebbe esaurito gli indirizzi entro pochi decenni usando 8 byte, mentre con indirizzi a 16 byte non si sarebbero mai esauriti. Altri sostenevano che 16 byte erano troppi, mentre altri ancora avrebbero preferito utilizzare indirizzi a 20 byte compatibili con il protocollo di datagramma OSI. Un'altra fazione proponeva indirizzi di lunghezza variabile. Dopo un lungo dibattito vennero scelti gli indirizzi a 16 byte di lunghezza fissa, il miglior compromesso tra tutte le possibilità.

Per scrivere gli indirizzi a 16 byte è stata escogitata una nuova notazione. Gli indirizzi sono scritti come otto gruppi di quattro cifre esadecimale separate da due punti:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Poiché molti indirizzi possono contenere numerosi zero, sono state autorizzate tre ottimizzazioni. Primo, è possibile omettere gli zero iniziali di un gruppo, perciò 0123 può essere scritto come 123. Secondo, uno o più gruppi contenenti 16 bit a zero possono essere sostituiti da due punti. Perciò, l'indirizzo precedente diventa:

8000::123:4567:89AB:CDEF

Infine, gli indirizzi IPv4 possono essere scritti in notazione decimale a punti dopo una copia di due punti:

::192.31.20.46

Forse non è necessario essere così esplicativi, ma esistono moltissimi indirizzi a 16 byte. Per la precisione, ci sono 2^{128} indirizzi, ossia circa 3×10^{38} . Se l'intero pianeta, terraferma e acqua, fosse coperto di computer, IPv6 permetterebbe di utilizzare 7×10^{23} indirizzi IP per metro quadrato. Gli studenti di chimica noteranno che questo numero è più grande del numero di Avogadro. Anche se nessuno aveva intenzione di dare un indirizzo IP a ogni molecola della superficie terrestre, non siamo troppo lontani dal farlo.

In pratica, lo spazio di indirizzamento non sarà utilizzato efficientemente, proprio come non lo è lo spazio di indirizzamento dei numeri telefonici (il prefisso di Manhattan, 212, è quasi esaurito, ma quello del Wyoming, 307, è quasi tutto vuoto). Nel documento RFC 3194, Durand e Huitema hanno calcolato che, usando come guida l'allocazione dei numeri telefonici, anche nello scenario più pessimistico ci saranno ancora più di 1.000 indirizzi.

zi IP per ogni metro quadrato di superficie terrestre (terraferma e acqua). In uno scenario probabile, ci saranno trilioni di indirizzi per metro quadrato. In breve, è molto improbabile che nell'immediato futuro gli indirizzi si esauriscano.

È istruttivo confrontare l'intestazione IPv4 (Figura 5.53) con l'intestazione IPv6 (Figura 5.68) per vedere che cosa è stato eliminato in IPv6. Il campo *IHL* è stato tolto perché l'intestazione IPv6 ha una lunghezza fissa. Il campo *protocol* è stato tolto perché il campo *next header* indica che cosa c'è dopo l'ultima intestazione IP (per esempio un segmento UDP o TCP).

Tutti i campi che riguardano la frammentazione sono stati rimossi, perché IPv6 gestisce la frammentazione in modo diverso. Tanto per cominciare, tutti gli host IPv6 compatibili devono determinare dinamicamente la dimensione del datagramma da utilizzare. Questa regola già rende la frammentazione meno probabile; poi la dimensione minima è stata portata da 576 a 1.280, per consentire 1.024 byte di dati e molte intestazioni. Inoltre, quando un host invia un pacchetto IPv6 che è troppo grande, il router che non è in grado di inoltrarlo trasmette indietro un messaggio di errore invece di frammentarlo. Questo messaggio dice all'host di suddividere tutti i pacchetti futuri diretti a quella destinazione. Fare in modo che l'host invii pacchetti della dimensione corretta è in definitiva più efficiente che costringere i router a frammentare i pacchetti al volo.

Infine, il campo *checksum* è stato eliminato perché l'elaborazione di questa informazione riduce enormemente le prestazioni. Con le reti affidabili utilizzate oggi, e visto che lo strato data link e gli strati di trasporto hanno i loro codici di controllo, il valore di un altro checksum non varrebbe il prezzo delle prestazioni richieste per estrarlo. La rimozione di tutte queste funzionalità ha permesso di ottenere un protocollo di strato network leggero e non invadente. Perciò questa architettura ha raggiunto l'obiettivo di IPv6: un protocollo veloce e flessibile, con una miriade di indirizzi.

Intestazione estesa

Alcuni dei campi mancanti di IPv4 sono ancora saltuariamente necessari, perciò IPv6 ha introdotto il concetto di **intestazione estesa** (opzionale), chiamata anche **extension header**. Queste intestazioni possono essere fornite per offrire informazioni aggiuntive, codificate in modo efficiente. Al momento sono definiti sei tipi di intestazioni estese, elencati nella Figura 5.69. Ogni estensione è opzionale, ma quando ne è presente più di una devono apparire direttamente dopo l'intestazione fissa e preferibilmente nell'ordine elencato. Alcune intestazioni hanno un formato fisso, altre contengono un numero variabile di campi di lunghezza variabile. Per queste, ogni elemento è codificato come una combinazione di tre elementi: *tipo*, *lunghezza*, *valore*. Il *tipo* è un campo lungo 1 byte che indica l'opzione; i valori di *tipo* sono stati scelti in modo tale che i primi due bit istruiscono i router che non sanno come elaborare l'opzione su ciò che devono fare. Le possibilità sono: non considerare l'opzione, scartare il pacchetto, scartare il pacchetto e inviare un pacchetto ICMP di risposta, scartare i dati e non inviare pacchetti ICMP per gli indirizzi multicast (in modo da impedire ai pacchetti multicast corrotti di generare milioni di comunicazioni ICMP).

Intestazione estesa	Descrizione
Opzioni hop per hop	Informazioni varie per i router
Opzioni di destinazione	Informazioni aggiuntive relative alla destinazione
Routing	Lista di router da visitare
Frammentazione	Gestione dei frammenti di datagramma
Autenticazione	Verifica dell'identità del trasmittente
Carico utile cifrato	Informazioni relative al contenuto cifrato

Figura 5.69. Le intestazioni estese IPv6.

Anche la *lunghezza* occupa 1 byte; questo campo indica l'estensione del valore (da 0 a 255 byte). Il *valore* è una qualsiasi informazione necessaria, lunga fino a 255 byte.

L'intestazione *hop-by-hop* è utilizzata per le informazioni che tutti i router lungo il percorso devono esaminare. Fino a questo momento è stata definita una sola opzione: supporto dei datagrammi che superano i 64 KB. La Figura 5.70 mostra il formato di questa intestazione. Quando è utilizzata, il campo *payload length* nell'intestazione fissa è impostato a zero.

Intestazione successiva	0	194	4
Jumbo payload length			

Figura 5.70. L'intestazione estesa hop-by-hop per grandi datagrammi (jumbogrammi).

Come nel caso di tutte le intestazioni estese, anche questa inizia con un byte che indica il tipo di intestazione che segue. Questo byte è seguito da un altro byte che indica la lunghezza, espressa in byte, dell'intestazione *hop-by-hop*, esclusi i primi 8 byte che sono obbligatori. Tutte le intestazioni iniziano in questo modo.

I successivi due byte indicano che questa opzione definisce la dimensione del datagramma (codice 194) e che la dimensione è un numero a 4 byte. Gli ultimi 4 byte rappresentano la dimensione del datagramma. Dimensioni minori di 65.536 byte non sono consentite e, se utilizzate, costringono il primo router a scartare il pacchetto e a inviare indietro un messaggio di errore ICMP. I datagrammi che usano questa estensione di intestazione sono chiamati **jumbogrammi**. L'uso dei jumbogrammi è importante per applicazioni di supercomputer che devono trasferire gigabyte di dati in modo efficiente attraverso Internet.

L'intestazione *destination options* (opzioni di destinazione) è utilizzata dai campi che devono essere interpretati dall'host di destinazione. Nella versione originale di IPv6, le uniche opzioni definite sono opzioni nulle usate per riempire questa intestazione con multipli di 8 byte, perciò inizialmente non sarà utilizzata. È stata inclusa per essere certi che il nuovo software di routing e il software host possano gestirla, in caso qualcuno pensi di utilizzare un giorno un'opzione di destinazione.

L'intestazione di routing elenca uno o più router che devono essere visitati lungo la strada

che porta alla destinazione. Assomiglia molto al *loose source routing* di IPv4, in quanto tutti gli indirizzi elencati devono essere visitati nell'ordine indicato, ma in mezzo possono essere visitati anche altri router non elencati. Il formato dell'intestazione di routing è mostrato nella Figura 5.71.

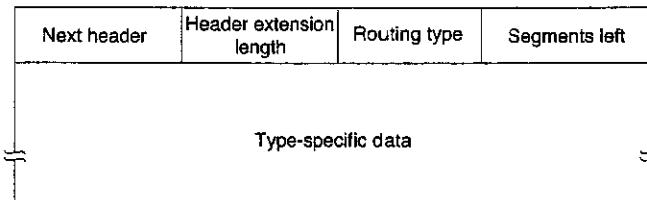


Figura 5.71. L'intestazione estesa per il routing.

I primi 4 byte dell'intestazione estesa di routing contengono quattro valori interi lunghi 1 byte. I campi *next header* (intestazione successiva) e *header extension lenght* (lunghezza estensione dell'intestazione) sono stati descritti precedentemente. Il campo *routing type* (tipo di routing) indica il formato del resto dell'intestazione. Il tipo 0 indica che una parola riservata a 32 bit segue la prima parola, seguita da qualche indirizzo IPv6. Altri tipi potranno essere inventati in futuro a seconda delle necessità. Infine, il campo *segment left* (segmenti rimasti) tiene traccia del numero di indirizzi nell'elenco che non sono ancora stati visitati, ed è decrementato ogni volta che ne viene visitato uno. Quando il valore raggiunge lo 0, il pacchetto è lasciato libero e può seguire il percorso che desidera. Di solito, a questo punto è talmente vicino alla destinazione che il percorso migliore appare ovvio. L'intestazione del frammento si occupa delle informazioni di frammentazione proprio come fa IPv4. L'intestazione contiene l'identificatore di datagramma, la posizione del frammento [NdR - è del tutto analogo alla frammentazione IPv4, con due sole eccezioni: solo gli host mittenti possono frammentare un pacchetto (non router), e inoltre ogni pacchetto ha una porzione non frammentabile costituita dalla testata principale e da tutti gli extention header che devono essere interpretati dai router. La posizione del frammento si intende relativa alla parte frammentabile del pacchetto] all'interno del datagramma originale e un bit che indica se il frammento corrente è seguito da altri. In IPv6, a differenza di IPv4, solo l'host sorgente può frammentare un pacchetto; i router lungo il percorso non possono farlo. Sebbene questo approccio rappresenti una rottura con la filosofia del passato, semplifica il lavoro dei router e rende più rapido il processo di routing. Come è stato spiegato precedentemente, se riceve un pacchetto troppo grande, il router scarta i dati e invia alla sorgente un pacchetto di risposta ICMP. Tale informazione permette all'host sorgente di spezzare il pacchetto in pezzi più piccoli, usando questa intestazione e tentando di nuovo la trasmissione.

L'intestazione di autenticazione fornisce un meccanismo che garantisce al ricevente di un pacchetto l'autenticità del mittente. Il carico utile cifrato permette di proteggere con la crittografia il contenuto del pacchetto in modo da renderlo leggibile solo ai destinatari desiderati. Queste intestazioni utilizzano allo scopo le tecniche crittografiche.

Controversie

Il processo di sviluppo di tipo aperto e le opinioni difese con vigore da molte delle persone coinvolte hanno reso naturale la nascita di controversie su molte delle scelte adottate per IPv6. Questo paragrafo riassume brevemente alcuni di queste discussioni; per maggiori dettagli, consultare i documenti RFC. È già stata menzionata la disputa legata alla lunghezza dell'indirizzo. Il risultato è stato un compromesso: indirizzo di lunghezza fissa a 16 byte. Un'altra lotta si è sviluppata intorno alla lunghezza del campo *hop limit*. Una fazione pensava che limitare il numero massimo di salti a 255 (scelta implicita se si utilizza un campo lungo 8 bit) sarebbe stato un madornale errore; dopo tutto, i percorsi di 32 salti oggi sono comuni, e tra dieci anni potranno essere comuni percorsi molto più lunghi. Queste persone sostenevano che l'utilizzo di un'enorme dimensione per l'indirizzo era una buona idea, mentre l'adozione di un conteggio di salti minuscolo era una sciocchezza. Secondo loro, il più grande peccato che un informatico può commettere è assegnare troppo pochi bit da qualche parte. La risposta fu che era possibile trovare argomenti che giustificassero l'aumento di ogni campo, cosa che avrebbe gonfiato enormemente l'intestazione. Inoltre, la funzione del campo *hop limit* è impedire ai pacchetti di girovagare per troppo tempo, e 65536 salti è un tempo eccessivo. Infine, al crescere di Internet aumenterà il numero dei collegamenti a lunga distanza, il che permetterà di raggiungere qualunque paese da qualunque altro in non più di una mezza dozzina di salti. Se la sorgente o la destinazione impiegano più di 125 salti per raggiungere i rispettivi gateway internazionali, c'è qualcosa di sbagliato nelle dorsali nazionali. I promotori degli 8 bit hanno quindi vinto lo scontro. Un'altra patata bollente era la dimensione massima del pacchetto. La comunità dei supercomputer voleva pacchetti che superassero i 64 KB. Quando si utilizza un supercomputer per trasferire dati la faccenda diventa seria, ed è meglio non interrompere l'operazione ogni 64 KB. Il ragionamento contro i pacchetti di grandi dimensioni è semplice: se un pacchetto da 1 MB raggiunge una linea T1 a 1,5 Mbps, quel pacchetto occupa la linea per 5 secondi, producendo un ritardo molto evidente agli utenti interattivi che condividono la linea; è stato quindi raggiunto un compromesso: i pacchetti normali non possono superare i 64 KB, ma l'intestazione estesa hop-by-hop può essere utilizzata per consentire il trasferimento di jumbogrammi. Un terzo tema scottante è stato la rimozione del checksum di IPv4. Alcune persone paragonavano l'operazione alla rimozione dei freni da un'automobile. Agendo in questo modo si ottiene un'auto più leggera e più veloce, ma in caso di eventi inaspettati si va incontro a problemi. L'argomento contro il checksum era che ogni applicazione che si preoccupa realmente dell'integrità dei suoi dati deve avere comunque un checksum sullo strato trasporto, perciò averne un altro in IP (oltre a quello presente nello strato data link) è eccessivo. Per di più, l'esperienza mostrava che l'elaborazione del checksum IP rappresenta una delle spese maggiori in IPv4. Vinse la fazione che si opponeva al checksum, e IPv6 non ha alcun checksum. Gli host mobili rappresentavano un altro punto di contesa. Se un computer portatile vola da una parte all'altra del mondo, può continuare a operare alla destinazione con lo stesso indirizzo IPv6 o deve utilizzare uno schema basato sugli agenti domestici e stranieri? Gli host mobili introducono anche asimmetrie nel sistema di routing. Un piccolo computer mobile potrebbe facilmente ascoltare il forte segnale emesso da un grande router stazionario, ma il router stazionario potrebbe non ricevere il debole segnale emesso dall'host mobile; di conseguenza, alcune persone volevano realizzare in IPv6 un supporto esplicito per gli host mobili.

Il tentativo fallì quando non fu possibile trovare consenso per alcuna proposta specifica. Probabilmente la battaglia più grande fu quella relativa alla sicurezza. Tutti concordavano che fosse essenziale, il problema era definire il dove e il come. Prima il dove: chi proponeva di inserirla nello strato network faceva notare che in questo modo la sicurezza sarebbe diventata un servizio standard, che tutte le applicazioni avrebbero potuto utilizzare senza fare alcuna pianificazione in anticipo. Chi si opponeva a questa scelta affermava che le applicazioni veramente sicure lo sono, solo quando si applica la crittografia da estremo a estremo, con l'applicazione sorgente che codifica i dati e l'applicazione di destinazione che li decodifica. Qualunque altro meccanismo avrebbe lasciato l'utente in balia di implementazioni dello strato network potenzialmente piene di errori, sulle quali non avrebbe potuto avere alcun controllo. Si ribatté a queste argomentazioni affermando che queste applicazioni possono semplicemente astenersi dall'utilizzare le funzionalità di protezione di IP e fare il lavoro da sole. La risposta a questa affermazione fu che coloro che non si fidano della rete non vogliono pagare il prezzo di lente e ingombranti implementazioni IP che hanno questa funzionalità, anche se disattivata.

Un altro aspetto relativo al dove inserire la sicurezza fa riferimento al fatto che molti i paesi (ma non tutti) hanno leggi di esportazione molto severe sulla crittografia. Alcuni paesi, specialmente la Francia e l'Iraq, ne limitano addirittura il utilizzo domestico. Di conseguenza, qualunque implementazione IP che utilizza sistemi crittografici abbastanza forti da essere molto validi non potrebbe essere esportata dagli Stati Uniti (e da molti altri paesi) verso i clienti di tutto il mondo. Il mantenimento di due categorie di software, una per uso domestico e l'altra per l'esportazione, è contrastato dalla maggior parte dei produttori di computer. Un punto sul quale non c'è stata alcuna controversia è quello dei tempi di transizione: nessuno, infatti, si aspetta che il passaggio da Internet IPv4 a Internet IPv6 possa avvenire da un giorno all'altro. Al contrario, saranno convertite a IPv6 singole "isole", messe inizialmente in comunicazione via tunnel. Quando le isole IPv6 incominceranno a crescere, si fonderanno tra loro e alla fine Internet sarà totalmente convertita. Dato il massiccio investimento di router IPv4 schierati al momento, il processo di conversione probabilmente richiederà un decennio. Per questo motivo, è stato fatto il massimo per assicurare che questa transizione sia il più indolore possibile. Per maggiori informazioni su IPv6, consultare (Loshin, 1999).

5.7 Sommario

Lo strato network fornisce servizi allo strato trasporto. Può basarsi su circuiti virtuali o datagrammi; in entrambi i casi, il suo compito principale è instradare i pacchetti dalla sorgente alla destinazione. Nelle sottoreti a circuito virtuale la decisione di routing è presa durante l'impostazione del circuito virtuale, nelle sottoreti a datagrammi la decisione è presa per ogni pacchetto. Nelle reti di computer si usano molti algoritmi di routing. Gli algoritmi statici includono il routing basato sul percorso più corto e il flooding; gli algoritmi dinamici includono il routing basato sul vettore delle distanze e quello basato sullo stato dei collegamenti. La maggior parte delle reti utilizza uno di questi algoritmi. Altri importanti argomenti che

riguardano il routing sono il routing gerarchico, il routing per gli host mobili, il routing broadcast, quello multicast e il routing nelle reti peer to peer.

Le sottoreti possono facilmente congestionarsi, aumentando il ritardo e diminuendo la capacità di trasporto dei pacchetti. I progettisti di reti tentano di evitare le congestioni con un'attenta progettazione; le tecniche disponibili includono il criterio di ritrasmissione, la memorizzazione in cache, il controllo di flusso e così via. Quando si crea una congestione è necessario affrontare il problema, inviando choke packet alla sorgente, gettando il carico o applicando altri metodi.

Il passo successivo alla gestione della congestione è tentare di ottenere realmente la qualità di servizio promessa. I metodi che possono essere utilizzati a questo scopo includono il buffering sul client, la modellazione del traffico, la prenotazione delle risorse e il controllo di accesso. Le strategie progettate per ottenere una buona qualità di servizio includono i servizi integrati (tra cui RSVP), i servizi differenziati e MPLS.

Le reti possono differire per molti aspetti, perciò quando vengono interconnesse molte reti possono nascere problemi. Qualche volta i problemi si possono ridurre inviando attraverso un tunnel i pacchetti che devono transitare per la rete aliena, ma l'approccio non funziona se la rete sorgente e quella di destinazione sono diverse. Quando reti differenti adottano dimensioni massime di pacchetto diseguali si può utilizzare la frammentazione.

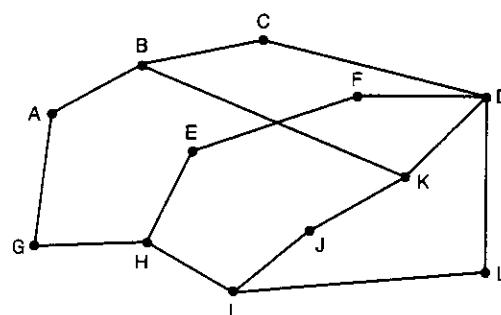
Internet supporta una gran varietà di protocolli collegati allo strato network. Tra questi, il protocollo di trasporto dati IP, ma anche i protocolli di controllo ICMP, ARP e RARP, e i protocolli di routing OSPF e BGP. Internet sta esaurendo rapidamente gli indirizzi IP, perciò è stata sviluppata una nuova versione di IP chiamata IPv6.

Problemi

1. Fornire due esempi di applicazioni di computer che si avvantaggiano di un servizio orientato alle connessioni. Fornire poi due esempi dove è preferibile il servizio ad assenza di connessione.
2. In quali casi un servizio orientato alle connessioni deve (o almeno dovrebbe) distribuire i pacchetti senza rispettare l'ordine di trasmissione?
3. Le sottoreti a datagrammi instradano ogni pacchetto come unità separata, indipendente dalle altre. Le sottoreti a circuito virtuale non hanno l'obbligo di seguire lo stesso approccio, poiché ogni pacchetto di dati segue un percorso predeterminato. Questa osservazione significa che le sottoreti a circuito virtuale non hanno bisogno della capacità di instradare pacchetti isolati da una sorgente arbitraria a una destinazione arbitraria? Spiegare la risposta.
4. Fornire tre esempi di parametri di protocollo che potrebbero essere negoziati durante l'impostazione di una connessione.
5. Si consideri il seguente problema di progettazione che riguarda l'implementazione di un servizio a circuito virtuale. Se all'interno della sottorete si usano i circuiti virtuali, ogni pacchetto di dati deve avere un'intestazione di 3 byte e ogni router deve assegnare 8 byte di memoria all'identificazione del circuito. Se internamente si usano i datagrammi, sono richieste intestazioni di 15 byte ma non è necessario assegnare alcuno spazio nella tabella del router. La capacità di trasmissione costa 1 centesimo per 10^6 byte, per ogni salto. Memoria molto veloce per il router può essere acquistata al prezzo di 1 centesimo al byte ed il suo valore commerciale scende a zero in due anni, considerando una settimana lavorativa di 40 ore. Statisticamente, la sessione

media dura 1.000 sec; durante quell'intervallo sono trasmessi 200 pacchetti. Il pacchetto medio richiede 4 salti. Quale implementazione è la meno costosa, e di quanto?

6. Supponendo che tutti i router e gli host funzionino correttamente e che tutto il software al loro interno non contenga errori, c'è qualche possibilità, seppur piccola, che un pacchetto venga trasmesso alla destinazione sbagliata?
7. Si consideri la rete mostrata nella Figura 5.7, ignorando i pesi sulle linee e supponendo che utilizzi la trasmissione di flusso come algoritmo di routing. Se un pacchetto inviato da A a D ha un contatore massimo di salti uguale a 3, elencare tutti i percorsi che seguirà. Indicare anche quanta banda consuma, espressa in termini di numero di salti.
8. Fornire una semplice euristica per trovare due percorsi attraverso una rete, tra una sorgente e una destinazione assegnate, che possano resistere alla perdita di una qualsiasi linea di comunicazione (supporre che esistano due percorsi di questo tipo). I router sono considerati abbastanza affidabili, perciò non è necessario preoccuparsi dei possibili guasti.
9. Si consideri la sottorete mostrata nella Figura 5.13(a). È utilizzato il routing basato sul vettore delle distanze e nel router C sono stati inseriti i vettori elencati di seguito. Da B: (5, 0, 8, 12, 6, 2); da D: (16, 12, 6, 0, 9, 10); da E: (7, 6, 3, 9, 0, 4). I ritardi misurati associati a B, D ed E, sono rispettivamente 6, 3 e 5. Quale sarà la nuova tabella di routing di C? Indicare la linea di trasmissione da utilizzare e il ritardo previsto.
10. Se i ritardi sono registrati sotto forma di numeri a 8 bit in una rete composta da 50 router, e i vettori dei ritardi sono scambiati due volte al secondo, quanta banda per linea (full duplex) è occupata dall'algoritmo di routing distribuito? Si supponga che ogni router abbia tre linee dirette agli altri router.
11. Nella Figura 5.14, l'OR logico delle due serie di bit ACF produce 111 in ogni riga. Il risultato è solo un caso o vale per tutte le sottoreti in tutte le circostanze?
12. Per il routing gerarchico con 4.800 router, quali dimensioni di cluster e regione dovrebbero essere scelte per ridurre al minimo la dimensione della tabella di routing in una gerarchia a tre livelli? Un buon punto di partenza è l'ipotesi che una soluzione con k cluster di k regioni di k router è quasi ottimale, quindi k è circa la radice cubica di 4.800 (circa 16). Controllare per tentativi le combinazioni quando tutti e tre i parametri sono vicini a 16.
13. Nel testo si afferma che quando un host mobile non è a casa, i pacchetti inviati alla sua LAN domestica sono intercettati dal suo agente fisso presente sulla LAN. Per una rete IP su una LAN 802.3, in che modo l'agente fisso compie questa intercettazione?
14. Osservando la sottorete mostrata nella Figura 5.6, quanti pacchetti sono generati da una trasmissione broadcast di B, nel caso di:
 - (a) un reverse path forwarding
 - (b) una struttura sink tree.
15. Si consideri la rete mostrata nella Figura 5.16(a). Si immagini di aggiungere una nuova linea tra F e G senza cambiare la struttura sink tree mostrata nella Figura 5.16(b). Quali modifiche devono essere apportate alla Figura 5.16(c)?
16. Calcolare uno spanning tree multicast per il router C nella sottorete seguente, per un gruppo con membri ai router A, B, C, D, E, F, I e K.



17. Nella Figura 5.20, il nodo *H* o il nodo *I* trasmettono mai in broadcast nella ricerca mostrata (che parte da *A*)?
18. Si supponga che il nodo *B* nella Figura 5.20 si sia appena riavviato e che non abbia informazioni di routing nelle sue tabelle. Improvvisamente ha bisogno di un percorso per raggiungere *H*. Invia pacchetti broadcast con *TTL* impostati a 1, 2, 3 e così via. Dopo quanti round trova un percorso?
19. Nella versione più semplice dell'algoritmo di Chord per la ricerca peer to peer, le ricerche non usano la tabella di puntamento, piuttosto sono lineari intorno al cerchio in entrambe le direzioni. Un nodo può predire in modo preciso la migliore direzione da seguire durante la ricerca? Spiegare la risposta.
20. Si consideri il cerchio di Chord mostrato nella Figura 5.24. Si supponga che il nodo 10 improvvisamente entri in linea. Questo evento influenza la tabella di puntamento del primo nodo, e se sì, in che modo?
21. Come meccanismo di controllo della congestione in una sottorete che usa internamente i circuiti virtuali, un router potrebbe astenersi dall'inviare l'acknowledgement in seguito all'arrivo di un pacchetto fino a quando (1) non sa che la sua ultima trasmissione attraverso il circuito virtuale è stata ricevuta con successo e (2) non ha un buffer vuoto. Per semplicità, si supponga che i router utilizzino un protocollo stop-and-wait e che ogni circuito virtuale abbia un buffer dedicato per ciascuna direzione del traffico. Se ci vogliono *T* sec per trasmettere un pacchetto (di dati o di acknowledgement) e se ci sono *n* router sul percorso, con quale velocità i pacchetti sono consegnati all'host di destinazione? Si supponga che gli errori di trasmissione siano rari e che la connessione host-router sia infinitamente veloce.
22. Una sottorete a datagrammi permette ai router di scartare pacchetti ogni volta che serve. La probabilità che un router scarti un pacchetto è *p*. Si consideri il caso di un host sorgente collegato al router sorgente, che è collegato al router di destinazione, che a sua volta è collegato all'host di destinazione. Se entrambi i router scartano un pacchetto, l'host sorgente alla fine va in timeout e rientra. Contando la linea host-router e quella router-router come un salto ciascuna, qual è il numero medio di
 (c) Salti compiuti da un pacchetto per ogni trasmissione?
 (d) Trasmissioni di un pacchetto?
 (e) Salti richiesti per ogni pacchetto ricevuto?
23. Descrivere due differenze principali tra il metodo del bit di allarme e il metodo RED.

24. Fornire un motivo per cui l'algoritmo leaky bucket dovrebbe consentire un solo pacchetto per ciclo di clock, qualunque sia la dimensione del pacchetto.
25. La variante a conteggio di byte dell'algoritmo leaky bucket è utilizzata in un particolare sistema. La regola è che in ogni istante possono essere inviati un pacchetto da 1.024 byte, o due pacchetti da 512 byte, e così via. Fornire una limitazione seria di questo sistema che non è stata menzionata nel testo.
26. Una rete ATM utilizza uno schema basato sul token bucket per modellare il traffico. Un nuovo token è inserito nel secchio ogni $5 \mu\text{sec}$. Ogni token vale per una cella, che contiene 48 byte di dati. Qual è la velocità trasferimento dati massima sostenibile?
27. Un computer su una rete a 6 Mbps è regolato da un token bucket. Il token bucket è riempito a una velocità di 1 Mbps e inizialmente contiene 8 Mb. Per quanto tempo il computer può trasmettere a 6 Mbps?
28. Si immagini una specifica di flusso che ha una dimensione massima di pacchetto pari a 1.000 byte, una velocità di riempimento del token bucket di 10 milioni di byte al secondo, un dimensione di token bucket pari a 1 milione di byte e una velocità di trasmissione massima uguale a 50 milioni di byte al secondo. Quanto può durare un burst alla velocità massima?
29. La rete mostrata nella Figura 5.37 utilizza RSVP con alberi multicast per gli host 1 e 2, come indicato. Si supponga che l'host 3 richieda un canale di banda 2 MB/sec per un flusso proveniente dall'host 1 e un altro canale di banda 1 MB/sec per un flusso proveniente dall'host 2. Contemporaneamente, l'host 4 richiede un canale di banda 2 MB/sec per un flusso proveniente dall'host 1 e l'host 5 richiede un canale di banda 1 MB/sec per un flusso proveniente dall'host 2. Qual è la banda totale riservata a queste richieste dai router *A*, *B*, *C*, *E*, *H*, *J*, *K* e *L*?
30. La CPU di un router può elaborare 2 milioni di pacchetti/sec. Il carico offerto al processore è pari a 1,5 milioni di pacchetti/sec. Se un percorso dalla sorgente alla destinazione contiene 10 router, quanto tempo dura l'accodamento e l'elaborazione eseguita dalle CPU?
31. Si consideri l'utente di servizi differenziati con inoltro accelerato. C'è una garanzia che i pacchetti accelerati subiscano un ritardo più piccolo dei pacchetti regolari? Perché?
32. La frammentazione è necessaria nelle internet a circuito virtuale concatenate o solo nei sistemi basati sui datagrammi?
33. L'utilizzo di tunnel attraverso una sottorete a circuito virtuale concatenata è semplice: il router multiprotocollo a un capo imposta semplicemente un circuito virtuale diretto all'altro capo e passa i pacchetti attraverso di esso. I tunnel possono essere utilizzati anche nelle sottoreti basate sui datagrammi? Se sì, in che modo?
34. Si supponga che l'host *A* sia collegato a un router *R* 1, che *R* 1 sia collegato a un altro router *R* 2 e che *R* 2 sia collegato all'host *B*. Si supponga che un messaggio TCP che contiene 900 byte di dati e 20 byte di intestazione TCP sia passato al codice IP dell'host *A* per la trasmissione a *B*. Mostrare i campi *total length*, *identification*, *DF*, *MF* e *fragment offset* dell'intestazione IP in ogni pacchetto trasmesso attraverso i tre collegamenti. Si supponga che il collegamento *A-R* possa supportare una dimensione massima di frame pari a 1.024 byte inclusi 14 byte di intestazione del frame, che il collegamento *R*-*R* 2 possa supportare una dimensione massima di frame pari a 512 byte, inclusi 8 byte dell'intestazione del frame, e che il collegamento *R*-*B* possa supportare una dimensione massima di frame di 512 byte inclusi 12 byte di intestazione.

35. Un router sta emettendo un getto di pacchetti IP la cui lunghezza totale (dati più intestazione) è di 1.024 byte. Supponendo che i pacchetti durino 10 sec, qual è la velocità massima della linea che il router può utilizzare senza correre il rischio di ripetere il numero di ID dei datagrammi IP?
36. Un datagramma IP che utilizza l'opzione *Strict source routing* deve essere frammentato. L'opzione va copiata in ogni frammento oppure è sufficiente inserirla solo nel primo frammento? Motivare la risposta.
37. Si supponga che invece di utilizzare 16 bit per la parte di rete di un indirizzo di classe B originale, fossero stati utilizzati 20 bit. Quante reti di classe B ci sarebbero state in questo caso?
38. Convertire in notazione decimale a punti l'indirizzo IP la cui rappresentazione esadecimale è C22F1582.
39. Una rete su Internet ha la maschera di sottorete 255.255.240.0. Quanti host può gestire al massimo?
40. Un gran numero di indirizzi IP consecutivi sono disponibili a partire da 198.16.0.0. Si supponga che quattro aziende, A, B, C e D, richiedano rispettivamente 4000, 2000, 4000 e 8000 indirizzi. Per ognuna di queste, dare il primo e l'ultimo indirizzo IP assegnato e la maschera nella notazione w.x.y.z/s.
41. Un router ha appena ricevuto i seguenti nuovi indirizzi IP: 57.6.96.0/21, 57.6.104.0/21, 57.6.112.0/21 e 57.6.120.0/21. Se usano tutti la stessa linea di trasmissione in uscita, possono essere aggregati? Se sì, in che modo? Se no, perché?
42. L'insieme di indirizzi IP compreso tra 29.18.0.0 e 29.18.128.255 è stato aggregato in 29.18.0.0/17. Comunque, c'è un intervallo di 1.024 indirizzi non assegnati che va da 29.18.60.0 a 29.18.63.255 che improvvisamente viene assegnato a un host che usa una diversa linea di trasmissione. È necessario dividere l'indirizzo aggregato nei suoi blocchi costituenti, aggiungere il nuovo blocco alla tabella e poi vedere se è possibile eseguire qualche aggregazione? Se no, che cosa si può fare?
43. Un router ha nella sua tabella di routing le seguenti voci (CIDR):
- | Indirizzo/maschera | Hop successivo |
|--------------------|----------------|
| 135.46.56.0/22 | Interfaccia 0 |
| 135.46.60.0/22 | Interfaccia 1 |
| 192.53.40.0/23 | Router 1 |
| predefinito | Router 2 |
- Per ognuno dei seguenti indirizzi IP, che cosa fa il router se arriva un pacchetto con quell'indirizzo?
- (f) 135.46.63.10
 - (g) 135.46.57.14
 - (h) 135.46.52.2
 - (i) 192.53.40.7
 - (j) 192.53.56.7
44. Molte aziende adottano il criterio di avere due (o più) router che collegano la società a Internet in modo da fornire ridondanza nel caso che uno si guasti. Questo criterio è ancora possibile con NAT? Spiegare la risposta.

45. Dopo aver spiegato a un amico come funziona il protocollo ARP, questa persona chiede: "Ho capito. ARP fornisce un servizio allo strato network perciò fa parte dello strato data link." Che cosa bisogna rispondergli?
46. ARP e RARP associano gli indirizzi da uno spazio all'altro. Da questo punto di vista, sono simili, ma le loro implementazioni sono fondamentalmente diverse. Quali sono le principali differenze?
47. Descrivere un sistema per riassemblare i frammenti IP giunti a destinazione.
48. La maggior parte degli algoritmi di ricostruzione dei datagrammi IP ha un timer che serve ad evitare che i frammenti perduti tengano impegnati in eterno i buffer di ricostruzione. Si supponga che un datagramma sia frammentato in quattro frammenti. I primi tre arrivano, l'ultimo invece è in ritardo. Alla fine, il timer scade e i tre frammenti vengono scartati dalla memoria del ricevente. Un po' più tardi, sopraggiunge l'ultimo frammento. Che cosa si deve fare in questo caso?
49. Sia in IP sia in ATM, il checksum copre solo l'intestazione e non i dati. Come mai è stato scelto questo approccio?
50. Una persona che vive a Boston viaggia fino a Minneapolis portandosi dietro il computer portatile. La LAN di destinazione a Minneapolis è una LAN IP wireless, perciò l'utente non deve collegare alcun cavo. È ancora necessario passare attraverso l'intero processo basato sull'agente fisso e sull'agente esterno per far arrivare correttamente la posta elettronica e il resto del traffico?
51. IPv6 utilizza indirizzi a 16 byte. Se un blocco di 1 milione di indirizzi venisse assegnato ogni picosecondo, quanto tempo durerebbero gli indirizzi?
52. Il campo *protocol* utilizzato nell'intestazione IPv4 non è presente nell'intestazione fissa IPv6. Come mai?
53. Quando è stato introdotto il protocollo IPv6, è stato necessario apportare modifiche al protocollo ARP? Se sì, le modifiche sono state concettuali o tecniche?
54. Scrivere un programma che simuli il routing eseguito tramite flooding. Ogni pacchetto dovrebbe contenere un contatore decrementato a ogni salto. Quando il contatore raggiunge lo zero, il pacchetto è scartato. Il tempo è discreto, ogni linea gestisce un pacchetto per intervallo di tempo. Creare tre versioni del programma: tutte le linee inondate, tutte le linee tranne quella di input, solo le k linee migliori (scelte in modo statico). Confrontare il routing basato sul flooding con il routing deterministico ($k = 1$) in termini di ritardo e banda usata.
55. Scrivere un programma che simuli una rete di computer a tempo discreto. Il primo pacchetto su ogni coda di router compie un salto ogni intervallo di tempo. Ogni router ha un numero finito di buffer. Se un pacchetto arriva e non c'è spazio per accoglierlo i dati sono scartati e non ritrasmessi, esiste però un protocollo end-to-end, completo di scadenze e pacchetti di acknowledgement, che in questi casi rigenera il pacchetto dal router sorgente. Disegnare la capacità di trasporto della rete come funzione dell'intervallo di scadenza end-to-end, parametrizzato in base alla frequenza di errore.

56. Scrivere una funzione che esegue l'inoltro in un router IP. La procedura ha un parametro, l'indirizzo IP, e accede a una tabella globale composta da una matrice di triplete. Ogni tripletta contiene tre valori interi: un indirizzo IP, una maschera di sottorete e la linea di trasmissione da utilizzare. La funzione cerca l'indirizzo IP nella tabella usando CIDR e restituisce la linea da utilizzare come suo valore.

57. Utilizzare i programmi *traceroute* (UNIX) o *tracert* (Windows) per tracciare il percorso dal proprio computer a varie università che si trovano in altri continenti. Creare la lista dei collegamenti transoceanici scoperti. Ecco alcuni siti da provare:

www.berkeley.edu (California)
www.mit.edu (Massachusetts)
www.vu.nl (Amsterdam)
www.ucl.ac.uk (London)
www.usyd.edu.au (Sydney)
www.u-tokyo.ac.jp (Tokyo)
www.uct.ac.za (Cape Town)

6

Lo strato trasporto

Lo strato trasporto non è uno strato qualsiasi: è il cuore dell'intera gerarchia di protocolli. Il suo compito è fornire il trasporto dei dati, affidabile ed efficiente in termini di costi, dal computer di origine a quello di destinazione, indipendentemente dalla rete o dalle reti fisiche effettivamente utilizzate. Senza lo strato trasporto, l'intero concetto di protocolli a strati avrebbe poco senso. In questo capitolo studieremo lo strato trasporto in dettaglio, analizzando i suoi servizi, la struttura, i protocolli e le prestazioni.

6.1 Il servizio di trasporto

Nei paragrafi seguenti forniremo un'introduzione al servizio di trasporto e osserveremo quale tipo di servizio viene fornito allo strato applicazione. Per rendere più concrete le problematiche del servizio di trasporto, esamineremo due gruppi di primitive dello strato. Per prima cosa sarà illustrato un insieme semplice (ma ipotetico) per mostrare le idee di base, dopodiché sarà presentata l'interfaccia utilizzata comunemente su Internet.

6.1.1 I servizi offerti agli strati superiori

L'obiettivo finale dello strato trasporto è fornire un servizio efficace, affidabile ed efficiente in termini di costi ai suoi utenti, che normalmente sono processi nello strato applicazione. Per raggiungere questo obiettivo, lo strato trasporto utilizza i servizi for-

niti dallo strato network. L'hardware e/o il software all'interno dello strato trasporto che svolge il lavoro è chiamato **entità di trasporto**. L'entità di trasporto può essere situata nel kernel del sistema operativo, in un processo utente separato, in un pacchetto di librerie associate alle applicazioni di rete, oppure (in linea di principio) in una scheda di interfaccia di rete. La relazione (logica) tra gli strati network, trasporto e applicazione è mostrata nella Figura 6.1.

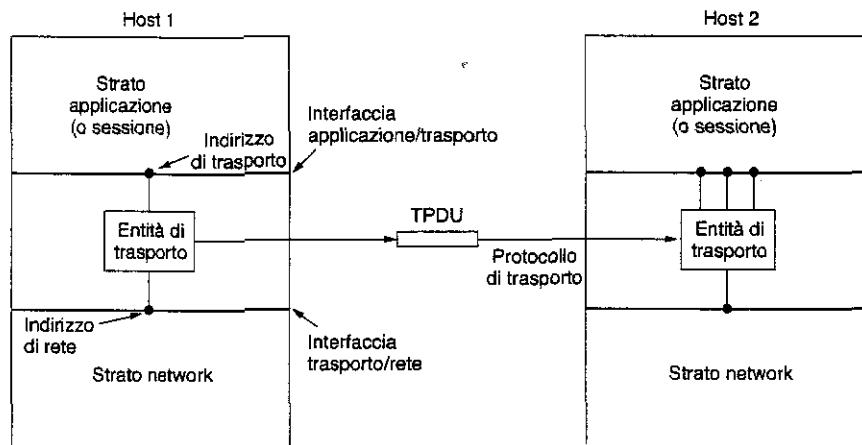


Figura 6.1. Gli strati network, trasporto e applicazione.

Proprio come esistono due tipi di servizi di rete, orientati oppure no alla connessione, esistono anche due tipi di servizi di trasporto. Il servizio di trasporto orientato alla connessione è per molti aspetti simile al servizio di rete orientato alla connessione. In entrambi i casi, le connessioni seguono tre fasi: costituzione, trasferimento dei dati e rilascio. Anche l'indirizzamento e il controllo del flusso sono simili in entrambi gli strati. Inoltre, il servizio di trasporto senza connessione è molto simile al servizio di rete senza connessione. La domanda ovvia è quindi la seguente: se il servizio dello strato trasporto è così simile al servizio dello strato network, perché esistono due strati distinti? Perché un solo strato non sarebbe adeguato? La risposta è sottile ma cruciale, e richiede di fare riferimento alla Figura 1.9. Il codice di trasporto scorre interamente sulle macchine dell'utente, mentre lo strato network è eseguito per la maggior parte nei router, che sono gestiti dall'operatore di telecomunicazioni (per lo meno nelle reti estese). Che cosa accade se lo strato network offre un servizio inadeguato? Magari se perde frequentemente i pacchetti? Che cosa accade se i router subiscono un crash di tanto in tanto?

I problemi esistono, questa è la verità. Gli utenti non hanno un reale controllo sullo strato network, pertanto non possono risolvere il problema di un servizio scadente utilizzando router più perfezionati o attivando una migliore gestione degli errori nello strato data link. L'unica possibilità è inserire sopra lo strato network un altro strato che migliori la qualità

del servizio. In una sottorete orientata alla connessione, se un'entità di trasporto viene informata a metà di una lunga trasmissione che la sua connessione di rete è stata terminata improvvisamente, senza un'indicazione relativa a cosa è accaduto ai dati attualmente in transito, l'entità può impostare una nuova connessione di rete verso l'entità di trasporto remota. Utilizzando questa nuova connessione di rete, può inviare un'interrogazione al suo pari (*peer*) chiedendo quali dati sono arrivati e quali no, riprendendo poi la trasmissione dove era stata interrotta.

In sostanza, l'esistenza dello strato trasporto consente al servizio di trasporto di essere più affidabile del sottostante servizio di rete. I pacchetti persi e i dati danneggiati possono essere rilevati e compensati dallo strato trasporto. Inoltre, le primitive del servizio di trasporto possono essere implementate come chiamate a procedure di libreria, per renderle indipendenti dalle primitive del servizio di rete. Le chiamate del servizio di rete possono variare considerevolmente da rete a rete: per esempio, un servizio LAN senza connessione può essere piuttosto diverso da un servizio WAN orientato alla connessione. Nascondendo il servizio di rete dietro un insieme di primitive del servizio di trasporto, la modifica del servizio di rete richiede solo la sostituzione di un insieme di procedure di libreria con un altro insieme capace di svolgere le stesse operazioni affidandosi a un differente servizio sottostante.

Grazie allo strato trasporto, i programmati di applicazioni possono scrivere codice usando un insieme standard di primitive e ottenere programmi funzionanti su molte reti diverse, senza doversi preoccupare della gestione delle interfacce delle sottoreti e delle trasmissioni inaffidabili. Se tutte le reti reali fossero prive di difetti, e avessero le stesse immutabili primitive di servizio, lo strato trasporto potrebbe non essere necessario, ma nel mondo reale svolge la funzione chiave di isolare gli strati superiori dalla tecnologia, dalla struttura e dalle imperfezioni della sottorete.

Per questo motivo, molte persone hanno tradizionalmente creato una distinzione tra i primi quattro strati e quelli superiori. I quattro strati inferiori possono essere visti come i **fornitori del servizio di trasporto**, mentre gli strati superiori sono gli **utenti del servizio di trasporto**. Questa distinzione tra fornitore e utente ha un impatto considerevole sulla struttura degli strati e pone lo strato trasporto in una posizione chiave, dal momento che forma il confine principale tra il fornitore e l'utente di un servizio di trasmissione dati affidabile.

6.1.2 Le primitive del servizio di trasporto

Per consentire agli utenti di accedere al servizio di trasporto, lo strato trasporto deve fornire alcune funzioni ai programmi applicativi, vale a dire un'interfaccia per il servizio di trasporto. Ogni servizio di trasporto ha la propria interfaccia. In questo paragrafo esamineremo per prima cosa un semplice servizio di trasporto (ipotetico) e la sua interfaccia per scoprire lo stretto indispensabile, mentre nel paragrafo successivo studieremo un esempio reale.

Il servizio di trasporto è simile al servizio network, ma vi sono alcune importanti differenze. La principale sta nel fatto che il servizio network è destinato a modellare quello offerto dalle reti reali, nel bene e nel male. Le reti reali possono perdere pacchetti, pertan-

to il servizio di rete è generalmente inaffidabile. Al contrario, il servizio di trasporto (orientato alla connessione) è affidabile. Naturalmente, le reti reali non sono prive di errori, ma questo è proprio lo scopo dello strato trasporto: fornire un servizio affidabile a una rete inaffidabile.

Come esempio, si possono considerare due processi connessi da pipe UNIX, che presuppongono una connessione perfetta. Non vogliono sapere nulla su acknowledgement, pacchetti persi, congestione o questioni simili; ciò che vogliono è una connessione affidabile al 100%. Il processo *A* inserisce i dati a un'estremità della pipe mentre il processo *B* li preleva all'altro capo. Il servizio di trasporto orientato alla connessione non è altro che questo: nasconde le imperfezioni del servizio di rete in modo che i processi utente possano presumere l'esistenza di un flusso di bit privo di errori.

Tra parentesi, lo strato trasporto può anche fornire un servizio (datagram) inaffidabile, ma c'è relativamente poco da dire sull'argomento, pertanto in questo capitolo ci concentreremo sul servizio di trasporto orientato alla connessione. Ciò nonostante esistono alcune applicazioni, per esempio per l'elaborazione client/server e lo streaming di elementi multimediali, che sfruttano il trasporto ad assenza di connessione, pertanto ne parleremo brevemente in seguito.

Una seconda differenza tra il servizio network e il servizio di trasporto riguarda i destinatari. Il servizio network è utilizzato solo dalle entità di trasporto. Pochi utenti scrivono le loro entità di trasporto, e quindi pochi utenti o programmi hanno visibilità del nudo servizio network. Al contrario, molti programmi (e programmatore) vedono le primitive di trasporto. Di conseguenza, il servizio di trasporto deve essere conveniente e facile da utilizzare.

Per farsi un'idea dell'aspetto di un servizio di trasporto è possibile considerare le cinque primitive elencate nella Figura 6.2. Questa interfaccia di trasporto è realmente essenziale, ma offre il senso di ciò che un'interfaccia di trasporto orientata alla connessione deve fare: consentire ai programmi applicativi di stabilire, utilizzare e rilasciare le connessioni; questo è sufficiente per molte applicazioni.

Primitiva	Pacchetto inviato	Significato
LISTEN	(nessuno)	Si blocca fino a quando un processo cerca di connettersi
CONNECT	CONNECTION REQ.	Tenta in modo attivo di stabilire una connessione
SEND	DATA	Invia informazioni
RECEIVE	(nessuno)	Si blocca fino all'arrivo di un pacchetto DATA
DISCONNECT	DISCONNECTION REQ.	Questo lato desidera rilasciare la connessione

Figura 6.2. Le primitive per un semplice servizio di trasporto.

Per vedere come possono essere utilizzate queste primitive, consideriamo un'applicazione con un server e diversi client remoti. Per iniziare, il server esegue una primitiva LISTEN, chiamando generalmente una procedura di libreria, che esegue una chiamata di sistema per bloccare il server fino all'accensione di un client. Quando un client desidera comunicare con il server, esegue una primitiva CONNECT. L'entità di trasporto esegue questa primitiva

tiva bloccando il chiamante e inviando un pacchetto al server. Incapsulato nel carico utile di questo pacchetto, troviamo un messaggio dello strato trasporto per l'entità di trasporto del server.

Ora è necessaria una breve nota sulla terminologia. In mancanza di un termine migliore, utilizzeremo in modo riluttante lo sgraziato acronimo TPDU (*Transport Protocol Data Unit*, unità dati del protocollo di trasporto) per i messaggi inviati da entità di trasporto verso entità di trasporto. Di conseguenza, le TPDU (scambiate dallo strato trasporto) sono contenute in pacchetti (scambiati dallo strato network). A loro volta, i pacchetti sono contenuti in frame (scambiati dallo strato data link). Quando arriva un frame, lo strato data link elabora l'intestazione del frame e passa il contenuto del campo *payload* (carico utile) del frame all'entità di rete. L'entità di rete elabora l'intestazione del pacchetto e passa il contenuto del carico utile del pacchetto all'entità di trasporto. Questa nidificazione è illustrata nella Figura 6.3.

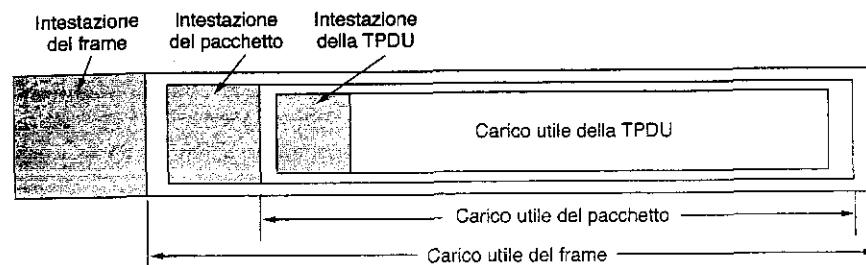


Figura 6.3. La nidificazione di TPDU, pacchetti e frame.

Tornando all'esempio client/server, la chiamata CONNECT del client provoca l'invio al server di una TPDU CONNECTION REQUEST. Quando arriva, l'entità di trasporto controlla se il server è bloccato su una chiamata LISTEN (vale a dire se è interessato alla gestione delle richieste). Sblocca poi il server e invia una TPDU CONNECTION ACCEPTED al client. Quando questa TPDU arriva, il client viene sbloccato e la connessione è stabilita.

Ora i dati possono essere scambiati con le primitive SEND e RECEIVE. Nella forma più semplice, ogni lato può inviare una primitiva RECEIVE (di blocco) per attendere l'invio del SEND da parte del corrispondente. Quando arriva la TPDU il ricevitore viene sbloccato, può quindi elaborarla e inviare una risposta. Finché entrambi i lati possono tenere traccia dei turni di invio, lo schema funziona bene.

Occorre notare che nello strato trasporto persino un semplice scambio di dati unidirezionale è più complicato rispetto allo strato network. Ogni pacchetto dati inviato viene (possibilmente) confermato con un acknowledgement. Anche i pacchetti che portano le TPDU di controllo sono confermati in modo implicito o esplicito. Questi acknowledgement sono gestiti dalle entità di trasporto utilizzando il protocollo dello strato network, e non sono visibili agli utenti del trasporto. Allo stesso modo le entità di trasporto dovranno preoccuparsi di

timer e ritrasmissioni, ma nessuno dei meccanismi è visibile agli utenti del trasporto. Per gli utenti del trasporto, una connessione è un canale affidabile per i bit: un utente inserisce i bit ed essi appaiono magicamente all'altra estremità. Questa capacità di nascondere la complessità è il motivo per cui i protocolli a strati sono uno strumento così potente.

Quando una connessione non è più necessaria, deve essere rilasciata per liberare spazio nella tabella all'interno delle due entità di trasporto. La disconnessione presenta due varianti: asimmetrica e simmetrica. Nella variante asimmetrica, ogni utente del trasporto può emettere una primitiva DISCONNECT, che provoca l'invio di una TPDU DISCONNECT all'entità di trasporto remota. Dopo l'arrivo, la connessione viene rilasciata.

Nella variante simmetrica, ogni direzione viene chiusa separatamente, indipendentemente dall'altra. Quando un lato emette una primitiva DISCONNECT, significa che non ha altri dati da inviare ma è ancora in grado di accettare dati dal suo partner. In questo modello, una connessione viene rilasciata quando entrambi i lati hanno inviato DISCONNECT. Un diagramma di stato per la costituzione della connessione e il rilascio di queste semplici primitive è mostrato nella Figura 6.4. Ogni transizione è innescata da qualche evento, cioè una primitiva eseguita dall'utente locale del trasporto o un pacchetto in ingresso. Per semplicità, presumiamo che ogni TPDU riceva separatamente l'acknowledgement. Supponiamo anche di impiegare un modello di disconnessione simmetrica, con inizio da parte del client. Occorre notare che questo modello è piuttosto semplice. Osserveremo modelli più realistici in seguito.

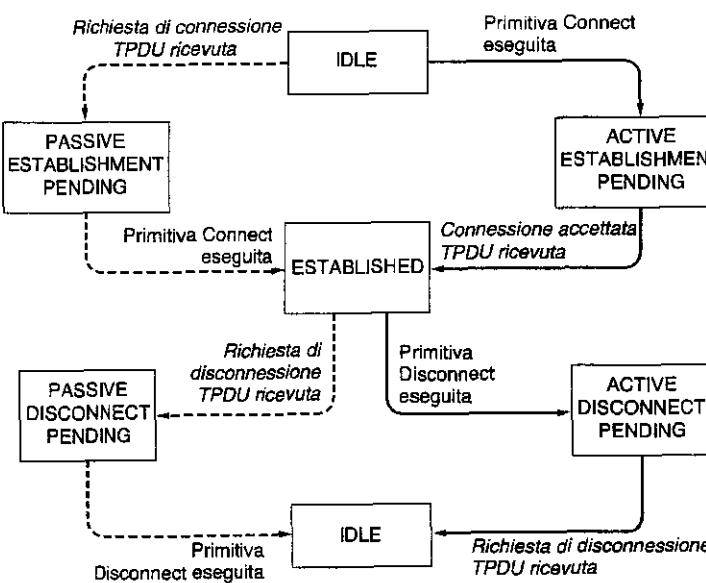


Figura 6.4. Un diagramma di stato per un semplice schema di gestione della connessione. Le transizioni in corsivo sono provocate dall'arrivo dei pacchetti. Le linee continue mostrano la sequenza di stato del client; le linee tratteggiate mostrano la sequenza di stato del server.

6.1.3 I socket Berkeley

Analizziamo velocemente un altro insieme di primitive di trasporto, le primitive socket utilizzate per TCP in Berkeley UNIX. Queste primitive sono ampiamente utilizzate per la programmazione Internet e sono elencate nella Figura 6.5. Seguono all'incirca il modello del nostro primo esempio, ma offrono maggiori caratteristiche e flessibilità. Per questo motivo non esamineremo le TPDU corrispondenti: l'argomento sarà affrontato con lo studio del TCP, più avanti nel capitolo.

Primitiva	Significato
SOCKET	Crea un nuovo punto finale di comunicazione
BIND	Associa un indirizzo locale a un socket
LISTEN	Annuncia la capacità di accettare connessioni; dà la dimensione della coda
ACCEPT	Blocca il chiamante fino all'arrivo di un tentativo di connessione
CONNECT	Tenta in modo attivo di stabilire una connessione
SEND	Invia alcuni dati sulla connessione
RECEIVE	Riceve alcuni dati sulla connessione
CLOSE	Rilascia la connessione

Figura 6.5. Le primitive socket per TCP.

Le prime quattro primitive dell'elenco sono eseguite nello stesso ordine dai server. La primitiva SOCKET crea un nuovo punto finale e alloca il corrispondente spazio nella tabella all'interno dell'entità di trasporto. I parametri della chiamata specificano il formato d'indirizzamento da utilizzare, il tipo di servizio desiderato (per esempio flusso di byte affidabile) e il protocollo. Una chiamata SOCKET eseguita con successo restituisce un normale descrittore di file da utilizzare nelle chiamate successive, in modo simile a una chiamata OPEN. I socket appena creati non hanno indirizzi di rete, che vengono assegnati utilizzando la primitiva BIND. I client remoti possono connettersi al server dopo che ha associato un indirizzo a un socket. Il motivo per cui non si utilizza la chiamata SOCKET per creare direttamente un indirizzo è che alcuni processi si prendono cura dei loro indirizzi (per esempio, se hanno utilizzato lo stesso indirizzo per anni e tutti lo conoscono), mentre altri non se ne occupano. A questo punto viene la chiamata LISTEN, che alloca spazio per accodare le chiamate in ingresso nel caso in cui diversi client provino a connettersi contemporaneamente. Al contrario di LISTEN nel primo esempio, nel modello socket LISTEN non è una chiamata bloccante. Per bloccarsi in attesa di una connessione in ingresso il server esegue una primitiva ACCEPT. Quando arriva una TPDU che richiede una connessione, l'entità di trasporto crea un nuovo socket con le stesse proprietà di quello originale, e restituisce un corrispondente descrittore di file. Il server può quindi generare un processo o un thread per gestire la connessione sul nuovo socket, e tornare ad attendere la connessione successiva sul socket originale. ACCEPT restituisce un normale descrittore di file, che può essere utilizzato per leggere e scrivere in modo analogo a un file.

Osserviamo ora il lato client. Anche qui è necessario creare un socket utilizzando la primitiva `SOCKET`, ma `BIND` non è richiesto perché l'indirizzo utilizzato non è importante per il server. La primitiva `CONNECT` blocca il chiamante e avvia il processo di connessione. Una volta completato (vale a dire quando la TPDU appropriata viene ricevuta dal server), il processo del client è sbloccato e viene stabilita la connessione. Entrambi i lati ora possono utilizzare `SEND` e `RECV` per trasmettere e ricevere dati sulla connessione full-duplex. Se non sono richieste le opzioni speciali di `SEND` e `RECV` si possono utilizzare anche le chiamate di sistema standard di `UNIX READ` e `WRITE`.

Con i socket il rilascio della connessione è simmetrico: la connessione viene rilasciata quando entrambi i lati hanno eseguito una primitiva `CLOSE`.

6.1.4 Un esempio di programmazione con socket: un file server Internet

Come esempio di utilizzo delle chiamate socket, esamineremo il codice di client e server della Figura 6.6, dove abbiamo un file server Internet molto primitivo e un client di esempio che lo utilizza. Questo codice ha severe limitazioni (discusse in seguito), ma in linea di principio il codice del server può essere compilato ed eseguito su qualsiasi sistema UNIX connesso a Internet. Il codice del client può quindi essere compilato ed eseguito su qualsiasi altra macchina UNIX connessa a Internet, ovunque nel mondo. Può essere eseguito con i parametri appropriati per prelevare qualsiasi file a cui il server ha accesso sul proprio computer. Il file viene scritto sullo standard output, che naturalmente può essere reindirizzato su file o pipe. Osserviamo per prima cosa il codice del server. Inizia con l'inclusione di alcune intestazioni standard, di cui le ultime tre contengono le principali definizioni e strutture dati correlate a Internet. Subito dopo è presente la definizione di `SERVER_PORT` come 12345: questo numero è stato scelto in modo arbitrario, qualsiasi numero compreso tra 1.024 e 65.535 funzionerà alla perfezione purché non utilizzato da altri processi. Naturalmente, il client e il server devono utilizzare la stessa porta. Se il server diventa un successo mondiale (poco probabile, considerando quanto è rudimentale), gli sarà assegnata una porta permanente inferiore a 1.024 e apparirà su www.iana.org.

Le due righe successive nel codice del server definiscono due costanti necessarie: la prima determina la dimensione del blocco utilizzato per il trasferimento di file, la seconda il numero di connessioni che possono essere tenute in sospeso prima di scartare all'arrivo le successive.

Il codice del server comincia dopo le dichiarazioni delle variabili locali, con l'inizializzazione di una struttura dati che conterrà l'indirizzo IP del server. Questa struttura dati sarà presto associata al socket del server. La chiamata a `memset` imposta la struttura dati con tutti i valori a zero, quindi le tre assegnazioni seguenti riempiono altrettanti campi; l'ultima contiene la porta del server. Le funzioni `htonl` e `htons` hanno a che fare con la conversione dei valori in un formato standard, in modo che il codice sia eseguito correttamente sia sulle macchine big-endian (per esempio SPARC) sia sulle macchine little-endian (per esempio Pentium). La loro esatta semanticà non è rilevante in questo caso.

Il server crea quindi un socket e controlla gli errori (indicati da `s < 0`). Nella versione di produzione del codice il messaggio di errore potrebbe essere più esplicativo. La chiamata a `setsockopt` è necessaria per consentire il riutilizzo della porta, in modo che il server possa

essere eseguito indefinitamente, rispondendo a ogni richiesta. Arrivati a questo punto l'indirizzo IP viene associato al socket, e si esegue un controllo per vedere se la chiamata a `bind` ha avuto successo. Il passaggio finale dell'inizializzazione è la chiamata a `listen` per annunciare la disponibilità del server ad accettare chiamate in ingresso, e comunicare al sistema di conservarne una quantità massima pari al valore `QUEUE_SIZE` nel caso arrivino nuove richieste mentre il server sta ancora elaborando quella attuale. Se la coda è piena e arrivano altre richieste, verranno tranquillamente scartate.

Adesso il server entra nel suo ciclo principale, che non abbandonerà mai: l'unico modo per fermarlo è interromperlo dall'esterno. La chiamata ad `accept` blocca il server fino a quando qualche client tenta di stabilire una connessione con esso. Se la chiamata `accept` ha successo, restituisce un descrittore di file che può essere utilizzato per la lettura e la scrittura, in modo analogo all'utilizzo dei descrittori di file per la lettura e la scrittura da pipe. Tuttavia, a differenza delle pipe (che sono unidirezionali), i socket sono bidirezionali ed è possibile utilizzare `sa` (*socket address*, indirizzo del socket) sia per leggere dalla connessione, sia per scrivervi.

Dopo avere stabilito la connessione, il server legge da questa il nome del file. Se il nome non è ancora disponibile, il server si blocca in attesa. Dopo avere ottenuto il nome file, il server lo apre ed entra in un ciclo che, alternativamente, legge blocchi dal file e li scrive sul socket fino a copiare l'intero file. Al termine il server chiude il file e la connessione, attendendo la comparsa della richiesta di connessione successiva. Il ciclo è ripetuto all'infinito.

Osserviamo ora il codice del client. Per comprenderne il funzionamento, è necessario capire come viene invocato. Supponendo di chiamarlo *client*, una chiamata tipica è la seguente:

```
client flits.cs.vu.nl /usr/tom/nomofile >f
```

Questa chiamata funziona solo se il server è già in esecuzione su `flits.cs.vu.nl`, il file `/usr/tom/nomofile` esiste, e il server possiede l'accesso in lettura per esso. Se la chiamata ha successo, il file viene trasferito su Internet e scritto su `f`, dopo di che il programma client esce. Dal momento che il server prosegue dopo un trasferimento, il client può essere avviato di nuovo per ottenere altri file.

Il codice del client inizia con alcune inclusioni e dichiarazioni. L'esecuzione inizia controllando se la funzione è stata chiamata con il numero corretto di argomenti (`argc = 3` significa il nome del programma più due argomenti). Occorre notare che `argv [1]` contiene il nome del server (per esempio `flits.cs.vu.nl`) ed è convertito in un indirizzo IP da `gethostbyname`. Questa funzione utilizza DNS per cercare il nome. Il DNS verrà affrontato nel Capitolo 7.

Successivamente viene creato e inizializzato un socket. Il client tenta poi di stabilire una connessione TCP al server utilizzando `connect`. Se il server è in esecuzione sulla macchina invocata, è collegato a `SERVER_PORT`, ed è idle o possiede spazio nella coda `listen`, verrà infine stabilita la connessione. Utilizzando la connessione, il client invia il nome del file scrivendo sul socket. Il numero di byte inviati è superiore al numero di byte del nome, perché deve essere inviato anche il byte 0 che termina il nome in modo che il server individui la fine del nome.

```

/*
 * Programma client che può richiedere un file al programma
 * server, listato nella pagina seguente. Il server risponde inviando l'intero file.
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345           /* arbitrario, ma client e server devono essere d'accordo */
#define BUF_SIZE 4096              /* dimensione del blocco di trasferimento */

int main(int argc, char **argv)
{
    int c, s, bytes;
    char buf[BUF_SIZE];           /* buffer per il file in arrivo */
    struct hostent *h;             /* informazioni sul server */
    struct sockaddr_in channel;   /* contiene l'indirizzo IP */

    if (argc != 3) fatal("Usage: client server-name file-name");
    h = gethostbyname(argv[1]);     /* cerca l'indirizzo IP dell'host */
    if (!h) fatal("gethostbyname failed");

    s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (s < 0) fatal("socket");
    memset(&channel, 0, sizeof(channel));
    channel.sin_family= AF_INET;
    memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);
    channel.sin_port= htons(SERVER_PORT);

    c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
    if (c < 0) fatal("connect failed");

    /* La connessione è stabilita. Invia il nome file includendo byte a 0 alla fine. */
    write(s, argv[2], strlen(argv[2])+1);

    /* Prende il file e lo scrive nello standard output. */
    while (1) {
        bytes = read(s, buf, BUF_SIZE);          /* legge dal socket */
        if (bytes <= 0) exit(0);                  /* controlla la fine del file */
        write(1, buf, bytes);                    /* scrive nello standard output */
    }
}

fatal(char *string)
{
    printf("%s\n", string);
    exit(1);
}

```

Lo strato trasporto

Figura 6.6. Il codice del client che utilizza i socket, seguito dal codice del server.

Ora il client entra in un ciclo, dove legge il file blocco per blocco dal socket e lo copia sullo standard output. Al termine, esce semplicemente.

La procedura *fatal* stampa un messaggio di errore ed esce. Il server necessita della stessa procedura, ma è stata omessa a causa della mancanza di spazio sulla pagina. Dal momento che il client e il server sono compilati separatamente e normalmente sono eseguiti su computer diversi, non possono condividere il codice di *fatal*.

Questi due programmi (insieme ad altro materiale correlato al libro) possono essere prelevati dal sito Web del libro:

<http://www.prenhall.com/tanenbaum>

facendo clic sul collegamento Companion Web Site accanto alla foto della copertina. Si possono scaricare e compilare su qualsiasi sistema UNIX (per esempio Solaris, BSD, Linux) con:

```
cc -o client client.c -lsocket -lnsl
cc -o server server.c -lsocket -lnsl
```

Il server viene avviato digitando:

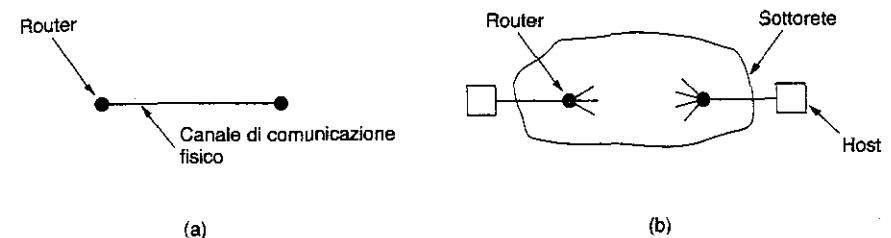
```
server
```

Il client richiede due argomenti, come discusso in precedenza. Sul sito Web è disponibile anche una versione Windows.

Per la precisione, come server questo non è il massimo. Il suo controllo degli errori è scarso e la segnalazione degli errori è mediocre. Chiaramente non ha mai sentito parlare della sicurezza e l'utilizzo di semplici chiamate di sistema UNIX non è la soluzione migliore per ottenere indipendenza dalla piattaforma. Fa anche supposizioni tecnicamente illegali: per esempio, presume che il nome file possa essere contenuto nel buffer e trasmesso in modo atomico. Dal momento che gestisce tutte le richieste in modo rigorosamente sequenziale (perché dispone di un singolo thread), le prestazioni sono scadenti. Nonostante queste carenze, è un file server Internet completo e funzionante. Negli esercizi il lettore è invitato a perfezionarlo. Per ulteriori informazioni sulla programmazione con i socket, vedere Stevens, 1997).

6.2 Gli elementi dei protocolli di trasporto

servizio di trasporto è implementato da un **protocollo di trasporto** utilizzato tra due entità di trasporto. Per certi aspetti, i protocolli di trasporto ricordano i protocolli di collegamento dati studiati in dettaglio nel Capitolo 3. Entrambi hanno a che fare, tra le altre cose, con il controllo degli errori, l'ordinamento in sequenza e il controllo di flusso. Tuttavia, tra i due esistono differenze significative, dovute alle diversità tra gli ambienti in cui operano i protocolli, come mostrato nella Figura 6.7. Nello strato data link due router comunicano direttamente tramite un canale fisico, mentre nello strato trasporto il canale fisico è sostituito dall'intera sottorete. Questa differenza porta con sé molte importanti applicazioni per i protocolli, come vedremo in questo capitolo.



Da una parte, nello strato data link non è necessario che un router specifichi con quale router desidera comunicare: ogni linea in uscita specifica in modo univoco un particolare router [Ndr - qui si descrive una configurazione costituita di router collegati tra loro da linea punto a punto come se questa fosse l'unica esistente. Ovviamente le funzionalità dello strato datalink sono necessarie e implementate per ogni calcolatore inserito in una rete. Anche restringendo il discorso ai soli router, è possibile che questi siano inseriti in reti broadcast (si considerino ad esempio le reti multi accesso in OSPF). In breve potremmo più correttamente affermare che nello strato datalink è -talvolta- indirizzare esplicitamente la destinazione (ma spesso è necessaria) mentre nello strato transport è sempre necessario]. Nello strato trasporto è invece richiesto l'indirizzamento esplicito delle destinazioni. Sotto un altro aspetto, il processo d'impostazione di una connessione sul filo della Figura 6.7(a) è semplice: l'altra estremità è sempre presente (a meno che subisca un crash, nel cui caso non è disponibile), e non c'è molto da fare. Nello strato trasporto la costituzione della connessione iniziale è più complicata, come vedremo.

Un'altra fastidiosissima differenza tra lo strato data link e lo strato trasporto è la potenziale esistenza di capacità di memorizzazione nella sottorete. Quando un router invia un frame, questo potrebbe arrivare o essere perso, ma non può rimbalzare avanti e indietro, nascondersi in un remoto angolo del mondo e riemergere improvvisamente trenta secondi dopo, in un momento inopportuno. Se la sottorete utilizza datagrammi e instradamento adattativo, c'è la possibilità che un pacchetto venga memorizzato per qualche secondo e consegnato in seguito.

La capacità della sottorete di memorizzare i pacchetti può avere conseguenze disastrose e richiedere l'utilizzo di protocolli speciali.

L'ultima differenza tra gli strati data link e di trasporto è di tipo quantitativo. Il buffering e il controllo di flusso sono necessari in entrambi gli strati, ma la presenza di un enorme numero di connessioni (variabile dinamicamente) nello strato trasporto può richiedere un approccio diverso da quello utilizzato nello strato data link.

Nel Capitolo 3 alcuni protocolli allocano un numero fisso di buffer per ogni linea, in modo che quando arriva un frame vi sia sempre un buffer disponibile. Nello strato trasporto, il gran numero di connessioni da gestire rende meno attraente l'idea di dedicare molti buffer a ogni linea. Nei paragrafi seguenti esamineremo queste ed altre importanti questioni.

6.2.1 L'indirizzamento

Quando un processo applicativo (per esempio un utente) vuole creare una connessione verso un processo applicativo remoto, deve specificare a quale intende connettersi. Il trasporto non orientato alla connessione presenta lo stesso problema: a chi bisogna inviare ogni messaggio? Il metodo normalmente utilizzato consiste nel definire indirizzi di trasporto su cui i processi possono restare in ascolto delle richieste di connessione. Su Internet questi punti finali sono chiamati **porte** (*port*), mentre nelle reti ATM sono chiamati **AAL-SAP**. Noi useremo il termine generico **TSAP** (*Transport Service Access Point*, punto di accesso al servizio di trasporto). I corrispondenti punti finali nello strato network

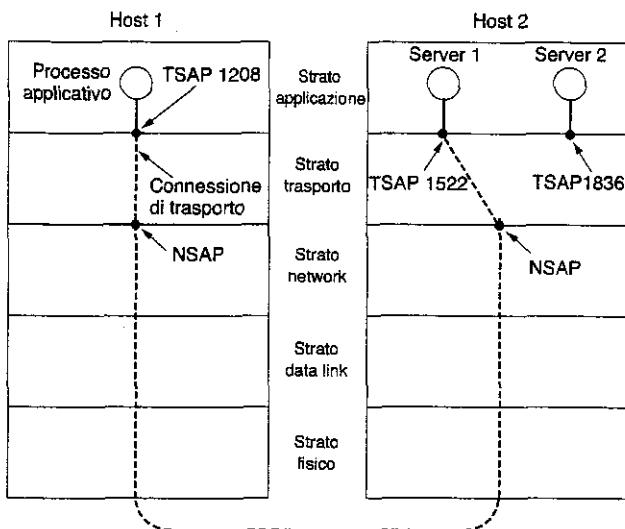


Figura 6.8. TSAP, NSAP e connessioni di trasporto.

(vale a dire gli indirizzi dello strato network) sono quindi chiamati **NSAP**. Gli indirizzi IP sono esempi di NSAP. La Figura 6.8 illustra la relazione tra NSAP, TSAP e la connessione di trasporto. I processi applicativi, client e server, possono collegarsi a un TSAP per stabilire una connessione con un TSAP remoto. Queste connessioni scorrono attraverso gli NSAP su ogni host. Il motivo per cui servono i TSAP è che in alcune reti ogni computer possiede un singolo NSAP, pertanto è necessario distinguere in qualche modo i diversi punti finali di trasporto che condividono tale NSAP.

Un possibile scenario per una connessione di trasporto è il seguente.

1. Un processo del server "ora esatta" sull'host 2 si collega a TSAP 1522 per attendere una chiamata in ingresso. Il modo in cui un processo si collega a TSAP è esterno al modello di rete, e dipende interamente dal sistema operativo locale. Potrebbe essere utilizzata una chiamata LISTEN, per esempio.

2. Un processo applicativo sull'host 1 desidera conoscere l'ora del giorno, pertanto emette una richiesta CONNECT che specifica TSAP 1208 come origine e TSAP 1522 come destinazione. Questa azione provoca la costituzione di una connessione di trasporto tra il processo applicativo sull'host 1 e il server 1 sull'host 2.
3. Il processo applicativo invia una richiesta per l'ora.
4. Il processo del server "ora esatta" risponde con l'ora corrente.
5. La connessione di trasporto viene rilasciata.

Notiamo che potrebbero esistere altri server sull'host 2 collegati ad altri TSAP, in attesa di connessioni in ingresso in arrivo sullo stesso NSAP.

Lo scenario appena visto è corretto, tranne per il fatto che abbiamo tacito un piccolo problema nascosto: come fa il processo utente sull'host 1 a sapere che il server per l'ora esatta è collegato a TSAP 1522? Una possibilità è che il server per l'ora esatta si sia collegato a TSAP 1522 da anni, e gradualmente gli utenti della rete lo hanno imparato. In questo modello i servizi dispongono di indirizzi TSAP stabili elencati in file salvati in posizioni ben note, come il file */etc/services* nei sistemi UNIX, che elenca quali server sono permanentemente collegati alle porte. Anche se gli indirizzi TSAP stabili funzionano bene per un numero ridotto di servizi chiave che non cambiano mai (per esempio un server Web), i processi utente spesso vogliono comunicare con altri processi utente che esistono solo per un breve periodo di tempo, e non possiedono un indirizzo TSAP conosciuto in anticipo. Inoltre, se esistono molti processi del server di cui alcuni utilizzati raramente, è dispendioso mantenerli tutti attivi e in ascolto su un indirizzo IP stabile per tutto il giorno. In pratica, è necessario uno schema migliore. Uno schema di questo tipo è mostrato nella Figura 6.9 in forma semplificata ed è noto come **protocollo di connessione iniziale**. Anziché avere ogni server in ascolto su uno TSAP noto, ogni macchina che desidera offrire servizi agli utenti remoti dispone di uno speciale **process server** che agisce da proxy per i server utilizzati più raramente. Ascolta un insieme di porte contemporaneamente, attendendo una richiesta di connessione. I potenziali utenti di un servizio iniziano con una richiesta CONNECT, specificando l'indirizzo TSAP del servizio desiderato. Se nessun server è in attesa la connessione avviene con il process server, come mostrato nella Figura 6.9(a). Dopo avere ottenuto la richiesta in ingresso, il process server crea il server richiesto, consentendogli di ereditare la connessione esistente con l'utente. Il nuovo server svolge quindi il lavoro, mentre il process server torna ad ascoltare nuove richieste come mostrato nella Figura 6.9(b). Il protocollo di connessione iniziale funziona bene per i server che possono essere creati in caso di necessità, ma si verificano spesso situazioni dove i servizi esistono indipendentemente dal process server. Un file server, per esempio, deve essere eseguito su hardware speciale (una macchina con un disco) e non può essere creato al volo quando qualcuno desidera comunicare con esso.

Per gestire questa situazione, viene spesso utilizzato uno schema alternativo. In questo modello esiste un processo speciale chiamato **server dei nomi** (*name server*) o a volte **directory server**. Per trovare l'indirizzo TSAP corrispondente a un dato nome di servizio,

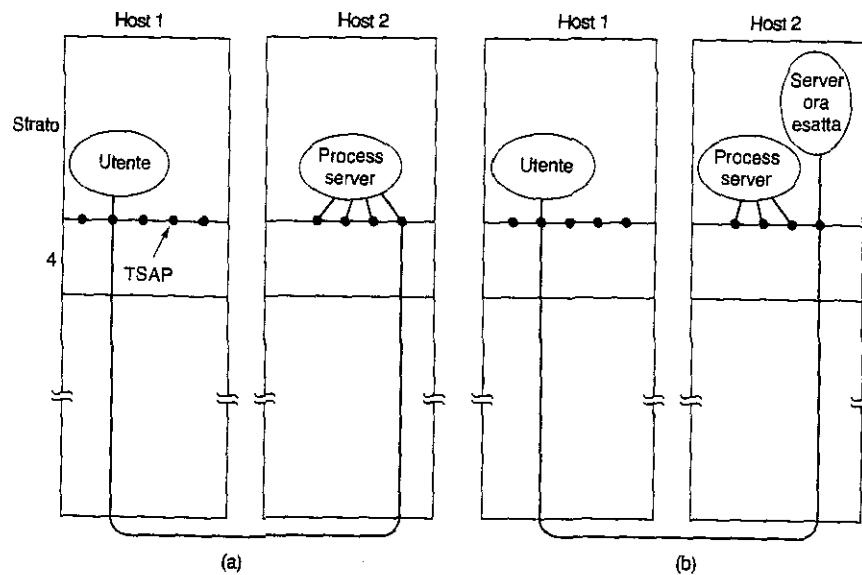


Figura 6.9. Modalità di connessione tra un processo utente nell'host 1 e un server ora esatta nell'host 2.

come "ora esatta", l'utente apre una connessione con il server dei nomi (che ascolta uno TSAP noto). L'utente invia poi un messaggio che specifica il nome del servizio, mentre il server dei nomi risponde con l'indirizzo TSAP. L'utente rilascia infine la connessione con il server dei nomi e ne stabilisce una nuova con il servizio desiderato.

In questo modello un nuovo servizio appena creato si deve registrare sul server dei nomi, presentando il nome del servizio (generalmente una stringa ASCII) e uno TSAP. Il server dei nomi registra queste informazioni nel suo database interno, per poter rispondere alle successive interrogazioni.

La funzione del server dei nomi è analoga a quella di un operatore addetto alla ricerca di numeri telefonici: fornisce un'associazione di nomi e numeri. Come in un sistema telefonico, è essenziale conoscere l'indirizzo dello TSAP utilizzato dal server dei nomi (o dal process server nel protocollo di connessione iniziale). Se non si conosce il numero dell'operatore per le informazioni, non è possibile contattarlo per chiedergli di eseguire una ricerca. Se qualcuno ritiene che il numero da comporre per ottenere le informazioni sia ovvio, provi a chiamarlo da un'altra nazione.

6.2.2 Stabilire la connessione

Sembra facile stabilire una connessione, ma in realtà l'operazione è sorprendentemente complessa. A prima vista, sembra sufficiente che un'entità di trasporto invii una TPDU CONNECTION REQUEST alla destinazione e attenda una risposta CONNECTION

ACCEPTED. Il problema si verifica se la rete può perdere, memorizzare e duplicare i pacchetti. Questo comportamento provoca serie complicazioni.

Si immagini una sottorete così congestionata che gli acknowledgement non riescono quasi mai a tornare indietro in tempo, ogni pacchetto subisce un timeout e viene ritrasmesso due o tre volte. Si supponga che la sottorete usi i datagrammi al suo interno e che ogni pacchetto segua una via diversa. Alcuni pacchetti potrebbero restare bloccati in una congestione del traffico all'interno della sottorete e richiedere molto tempo per arrivare, vale a dire che sono memorizzati nella sottorete e sbucano fuori in seguito.

Il peggior incubo possibile è il seguente. Un utente stabilisce una connessione con una banca, invia messaggi che comunicano alla banca di trasferire un'elevata quantità di denaro al conto di una persona non completamente affidabile e quindi rilascia la connessione. Sfortunatamente, ogni pacchetto in questo scenario viene duplicato e memorizzato nella sottorete. Dopo il rilascio della connessione, tutti i pacchetti escono dalla sottorete e arrivano ordinatamente alla destinazione, chiedendo alla banca di stabilire una nuova connessione, trasferire (di nuovo) il denaro e rilasciare la connessione. La banca non ha modo di capire che si tratta di duplicati: presume che sia una seconda transazione indipendente e trasferisce di nuovo il denaro. Nella parte rimanente di questo paragrafo studieremo il problema dei duplicati ritardati, dedicando una particolare attenzione agli algoritmi per stabilire connessioni in modo affidabile, in modo che incubi come quello presentato sopra non possano accadere.

Il nodo del problema è l'esistenza di duplicati ritardati. Può essere affrontato in vari modi, nessuno dei quali è realmente soddisfacente. Un modo prevede l'utilizzo di indirizzi di trasporto monouso. In questo approccio, ogni volta che diventa necessario un indirizzo di trasporto, ne viene generato uno nuovo. Quando una connessione viene rilasciata, l'indirizzo è scartato e non sarà mai più utilizzato. Questa strategia non consente di realizzare il modello di process server della Figura 6.9.

Un'altra possibilità richiede di assegnare a ogni connessione un identificatore (cioè un numero sequenziale incrementato a ogni connessione stabilita), scelto dalla parte che inizia la connessione e inserito in ogni TPDU, compresa quella che richiede la connessione. Dopo il rilascio di una connessione, ogni entità di trasporto potrebbe aggiornare una tabella che elenca le connessioni obsolete sotto forma di coppie (entità di trasporto, identificatore di connessione). Ogni volta che giunge una richiesta di connessione, è possibile confrontarla con la tabella per vedere se apparteneva a una connessione rilasciata in precedenza.

Sfortunatamente, questo schema presenta un difetto di base: richiede che ogni entità di trasporto mantenga indefinitamente una certa quantità di informazioni storiche. Se una macchina subisce un crash e perde la sua memoria, non saprà più quali identificatori di connessione sono già stati utilizzati.

Occorre quindi seguire un'altra direzione. Anziché consentire ai pacchetti di "vivere" all'infinito all'interno della sottorete, si deve escogitare un meccanismo per distruggere i pacchetti obsoleti che sono ancora in circolo. Se è possibile garantire che nessun pacchetto viva più a lungo di un tempo ben definito, il problema diventa qualcosa di più gestibile.

La durata del pacchetto può essere limitata a un valore massimo prefissato utilizzando una (o più) delle seguenti tecniche:

1. progettazione di sottoreti limitate
2. inserimento di un contatore di salti in ogni pacchetto
3. applicazione di un contrassegno temporale (*timestamp*) a ogni pacchetto.

Il primo metodo combina due elementi: una qualsiasi soluzione che impedisce ai pacchetti di essere ripetuti ciclicamente, e un modo per contenere il ritardo dovuto alla congestione sul percorso più lungo possibile (ora noto). Il secondo metodo consiste nell'inizializzare il conteggio dei salti a un valore appropriato, e decrementarlo ogni volta che il pacchetto viene inoltrato. Il protocollo di rete scarta tutti i pacchetti dove il contatore di salti è arrivato a zero. Il terzo metodo richiede che ogni pacchetto porti con sé l'orario in cui è stato creato; i router scarteranno qualsiasi pacchetto più vecchio di un tempo stabilito. Ciò richiede che gli orologi dei router siano sincronizzati, che è un'operazione per nulla semplice, a meno di ottenere la sincronizzazione esternamente alla rete (per esempio utilizzando GPS o una stazione radio che trasmette periodicamente l'ora esatta).

In pratica, occorre garantire non solo che un pacchetto sia "defunto", ma anche che tutti i suoi acknowledgement siano stati rimossi; per questo motivo introdurremo T , un piccolo multiplo della reale vita massima del pacchetto. Il multiplo dipende dal protocollo e ha il semplice effetto di rendere T più lungo. Se si attende un tempo T dopo l'invio di un pacchetto, è possibile essere certi che tutte le sue tracce sono scomparse e che né il pacchetto né i suoi acknowledgement appariranno improvvisamente a complicare le cose.

Contenendo le durate dei pacchetti si può escogitare un metodo infallibile per stabilire le connessioni in sicurezza. Il metodo che viene descritto è dovuto a Tomlinson (1975), e risolve il problema ma introduce alcune peculiarità: è stato quindi ulteriormente rifinito da Sunshine e Dalal (1978). Alcune sue varianti sono ampiamente utilizzate, anche in TCP. Per aggirare il problema di una macchina che perde tutta la memoria quando subisce un crash, Tomlinson ha proposto di equipaggiare ogni host con un orologio giornaliero. Gli orologi in host diversi non devono necessariamente essere sincronizzati; ciascun orologio è costituito da un contatore binario incrementato a intervalli regolari. Inoltre, il numero di bit del contatore deve essere uguale o maggiore al numero di bit dei numeri di sequenza. Infine, la cosa più importante è che l'orologio deve continuare a funzionare anche se l'host si blocca e riavvia.

L'idea di base è assicurare che non siano mai in circolazione contemporaneamente due TPDU numerate in modo identico. Quando viene impostata una connessione, i k bit di ordine basso dell'orologio sono utilizzati come numero di sequenza iniziale (composto da k bit). Di conseguenza, a differenza dei protocolli nel Capitolo 3, ogni connessione inizia a numerare le sue TPDU con un numero di sequenza differente. Lo spazio della sequenza dovrebbe essere abbastanza grande da garantire che, nel momento in cui i numeri di sequenza riprendono dall'inizio, le vecchie TPDU con lo stesso numero di sequenza siano già state rimosse. Questa relazione lineare tra il tempo e i numeri di sequenza iniziali è mostrata nella Figura 6.10.

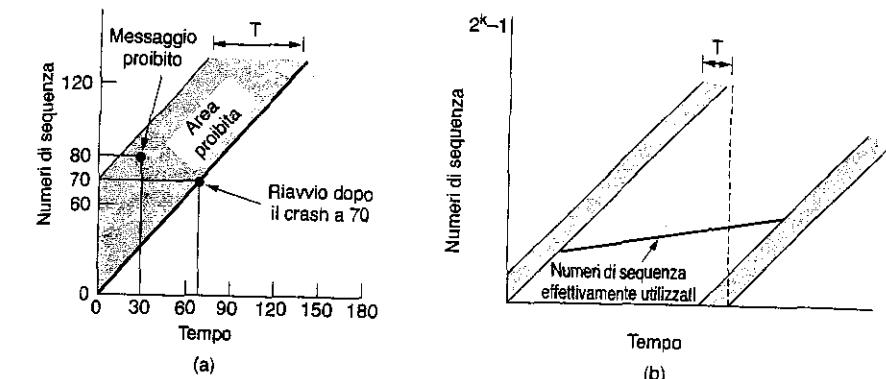


Figura 6.10. (a) Le TPDU non possono entrare nell'area proibita.
(b) Il problema della risincronizzazione.

Quando le entità di trasporto si sono accordate sul numero di sequenza iniziale, è possibile utilizzare qualsiasi protocollo sliding window per il controllo di flusso dei dati. In realtà, la curva del numero di sequenza iniziale (mostrata dalla linea spessa) non è lineare ma a scalini, dal momento che l'orologio procede con passaggi discreti. Per semplicità questo dettaglio verrà ignorato.

Si verifica un problema quando l'host subisce un crash. Al riavvio, la sua entità di trasporto non sa dove si trovava nello spazio della sequenza. Una soluzione prevede di obbligare le entità di trasporto a restare inattive per T secondi dopo un ripristino, per consentire la rimozione di tutte le vecchie TPDU; ma in una internetwork complessa T può essere grande, per cui questa strategia è poco attraente.

Per evitare la necessità di T secondi di tempo morto dopo un crash, è necessario introdurre una nuova limitazione sull'utilizzo dei numeri di sequenza. È possibile convincersi della necessità di questa limitazione con un esempio. Supponiamo che T (la vita massima del pacchetto) sia 60 secondi e che il clock batta ogni secondo. Come mostrato dalla linea spessa nella Figura 6.10(a), il numero di sequenza iniziale per una connessione aperta all'ora x sarà x . Immaginiamo che a $t = 30$ secondi sia assegnato il numero di sequenza 80 a una TPDU dati ordinaria, inviata sulla connessione 5 aperta in precedenza. Chiamiamo X questa TPDU. Immediatamente dopo avere inviato la TPDU X , l'host subisce un crash e viene rapidamente riavviato. A $t = 60$, inizia a riaprire le connessioni da 0 a 4. A $t = 70$ riapre la connessione 5, utilizzando il numero di sequenza iniziale 70 come richiesto. Nei successivi 15 secondi invia le TPDU dati da 70 a 80, di conseguenza, a $t = 85$ una nuova TPDU con numero di sequenza 80 e connessione 5 viene inviata nella sottorete. Sfortunatamente, la TPDU X esiste ancora. Se dovesse arrivare al ricevitore prima della nuova TPDU 80, la TPDU X sarà accettata e la TPDU 80 valida sarà rifiutata come duplicata.

Per impedire questo genere di problemi, bisogna vietare l'utilizzo (vale a dire l'assegnazione a nuove TPDU) dei numeri di sequenza per un tempo T prima del loro potenziale utilizzo come numeri di sequenza iniziali. Le combinazioni illegali di tempo e numero di

sequenza sono mostrate come **area proibita** nella Figura 6.10(a). Prima di inviare una TPDU su una qualsiasi connessione, l'entità di trasporto deve leggere il clock e controllare se non si trova nell'area proibita. Il protocollo può mettersi nei guai in due modi distinti. Se un host invia molti dati troppo velocemente su una connessione appena aperta, il numero di sequenza attuale potrebbe salire più rapidamente rispetto al numero di sequenza iniziale, sul diagramma temporale. Ciò significa che la velocità massima dei dati su qualsiasi connessione non deve superare una TPDU per battito dell'orologio, e significa anche che l'entità di trasporto, prima di aprire una nuova connessione dopo il riavvio in seguito al crash, deve attendere fino al battito dell'orologio per evitare di usare due volte lo stesso numero. Entrambi i requisiti invitano a scegliere un intervallo breve (pochi microsecondi o meno) tra un battito e l'altro. Sfortunatamente, l'ingresso nell'area proibita dal lato inferiore (a causa di un invio troppo rapido) non è l'unico modo per incorrere in problemi. Nella Figura 6.10(b) è possibile notare che, per qualsiasi cadenza dei dati inferiore a quella dell'orologio, la curva dei numeri di sequenza effettivi entra nell'area proibita da sinistra. Maggiore è la pendenza della curva dei numeri di sequenza effettivi, più lungo sarà ritardato questo evento. Come affermato in precedenza, appena prima di inviare ogni TPDU l'entità di trasporto deve controllare se sta per entrare nell'area proibita; in tal caso, deve ritardare la TPDU per T secondi o risincronizzare i numeri di sequenza. Il metodo basato sull'orologio risolve per le TPDU dati il problema del duplicato in ritardo; tuttavia, affinché questo metodo sia utile bisogna prima stabilire una connessione. Dal momento che anche le TPDU di controllo possono essere ritardate, esiste un problema potenziale per accordare le due parti sul numero di sequenza iniziale. Supponiamo, per esempio, che le connessioni vengano stabilite mediante l'host 1 che invia a un peer remoto (l'host 2) una TPDU CONNECTION REQUEST, contenente il numero di sequenza iniziale proposto e un numero di porta di destinazione. Il ricevitore host 2 riconosce la richiesta restituendo una TPDU CONNECTION ACCEPTED. Se la TPDU CONNECTION REQUEST viene persa, ma nell'host 2 appare improvvisamente una CONNECTION REQUEST duplicata e ritardata, la connessione viene stabilita in modo non corretto.

Per risolvere questo problema, Tomlinson (1975) ha introdotto l'**handshake a tre vie**. Questo protocollo non richiede a entrambe le parti d'iniziare la trasmissione partendo dallo stesso numero di sequenza, pertanto può essere utilizzato con metodi di sincronizzazione diversi dal metodo dell'*orologio globale* [NdR - l'handshake a tre vie non richiede sincronizzazione sui numeri di sequenza iniziali, e può essere utilizzato con qualunque metodo di assegnazione del numero di sequenza iniziale (dunque anche con il *metodo dell'orologio locale* appena descritto)]. Si noti che il metodo dell'*orologio globale* non è stato discusso esplicitamente, ma evidentemente richiede che tutti i calcolatori siano sincronizzati temporalmente, ed è dunque collegato al *metodo del timestamp* per limitare la vita massima di un pacchetto]. La normale procedura per stabilire la connessione quando l'host 1 viene inizializzato è mostrata nella Figura 6.11. L'host 1 sceglie un numero di sequenza, x , e invia una TPDU CONNECTION REQUEST contenente x all'host 2. L'host 2 risponde con una TPDU ACK che riconosce x e annuncia il suo numero di sequenza iniziale, y . Per finire, l'host 1 riconosce la scelta del numero di sequenza iniziale dell'host 2 nella prima TPDU dati inviata.

Ora vediamo come funziona l'handshake a tre vie in presenza di TPDU di controllo duplicate e ritardate. Nella Figura 6.11(b) la prima TPDU è una CONNECTION REQUEST

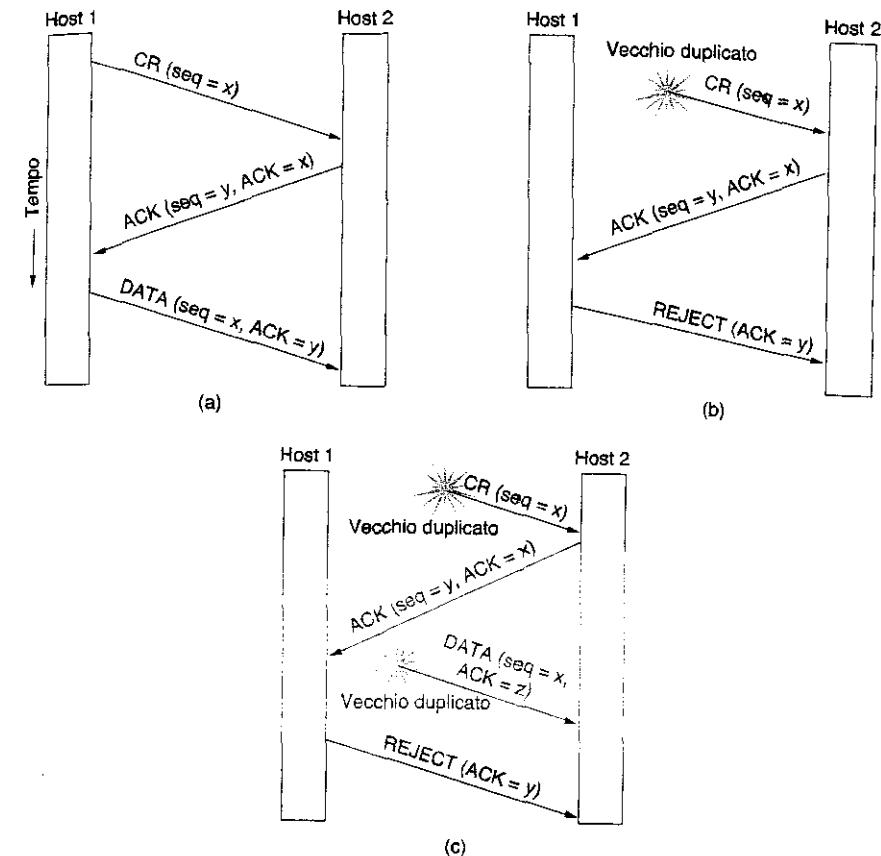


Figura 6.11. Tre scenari relativi alla costituzione di una connessione con l'handshake a tre vie. CR indica CONNECTION REQUEST.
 (a) Operazioni normali.
 (b) Una vecchia CONNECTION REQUEST duplicata appare dal nulla.
 (c) La presenza di CONNECTION REQUEST e ACK duplicati.

duplicata e ritardata che si riferisce a una vecchia connessione. Questa TPDU arriva all'host 2 senza che l'host 1 lo sappia. L'host 2 reagisce inviando all'host 1 una TPDU ACK, chiedendo in definitiva una conferma del fatto che l'host 1 sta tentando di impostare una nuova connessione. Quando l'host 1 rifiuta il tentativo di stabilire una connessione dell'host 2, l'host 2 comprende di essere stato ingannato da un duplicato in ritardo, e abbandona la connessione: grazie a questo metodo un duplicato in ritardo non fa danni. Il caso peggiore, mostrato nella Figura 6.11(c), si verifica quando nella sottorete circolano sia una CONNECTION REQUEST ritardata sia un ACK. Come nell'esempio precedente, l'host 2 riceve una CONNECTION REQUEST ritardata e risponde ad essa.

A questo punto è fondamentale comprendere che l'host 2 ha proposto l'utilizzo di y come numero di sequenza iniziale per il traffico dall'host 2 all'host 1, sapendo bene che nessuna TPDU contenente il numero di sequenza y o l'acknowledgement a y è ancora in circolazione. Quando la seconda TPDU ritardata giunge all'host 2, il fatto che sia stato riconosciuto z al posto di y comunica all'host 2 che anche questo è un vecchio duplicato. È importante osservare che non esiste una combinazione di vecchie TPDU che può provocare il fallimento del protocollo o l'instaurarsi accidentale di una connessione indesiderata.

6.2.3 Il rilascio della connessione

Rilasciare una connessione è più facile che stabilirla, però le trappole sono più di quante se ne possano immaginare. Come affermato in precedenza, esistono due modi per terminare una connessione: il rilascio asimmetrico e il rilascio simmetrico. Il rilascio asimmetrico è il metodo utilizzato dal sistema telefonico: quando una delle due parti riattacca la connessione viene interrotta. Il rilascio simmetrico tratta la connessione come se fosse composta da due connessioni unidirezionali separate, e richiede il rilascio separato di ognuna di esse.

Il rilascio asimmetrico è improvviso e può provocare una perdita di dati. Si consideri lo scenario della Figura 6.12. Dopo avere stabilito la connessione, l'host 1 invia una TPDU che arriva correttamente nell'host 2, e successivamente invia un'altra TPDU. Sfortunatamente, l'host 2 emette una TPDU DISCONNECT prima dell'arrivo della seconda TPDU. Il risultato è che la connessione è rilasciata e i dati vanno persi.

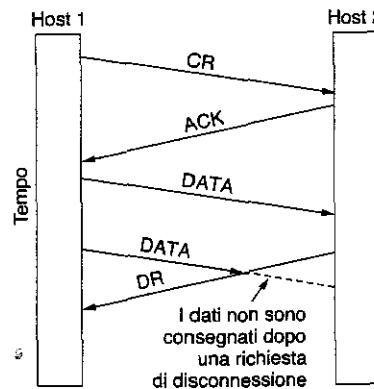


Figura 6.12. Una disconnessione improvvisa con perdita di dati.

Chiaramente, è necessario un protocollo di rilascio più sofisticato per evitare la perdita dei dati. Un metodo prevede l'utilizzo del rilascio simmetrico, in cui ogni direzione è rilasciata indipendentemente dall'altra. Un host può continuare a ricevere dati anche dopo avere inviato una TPDU DISCONNECT.

Il rilascio simmetrico svolge il suo compito quando ogni processo presenta una quantità fissa di dati da inviare e sa chiaramente quando li ha inviati; ma in certe situazioni deter-

minare che tutto il lavoro è stato svolto e che la connessione dovrebbe essere terminata non è un'operazione così ovvia. Qualcuno potrebbe immaginare un protocollo in cui l'host 1 afferma: "Ho finito. Hai finito anche tu?". Se l'host 2 risponde: "Ho finito anch'io. Ciao", la connessione può essere rilasciata in sicurezza.

Sfortunatamente, questo protocollo non funziona sempre. Esiste un famoso problema che illustra la questione: è chiamato **problema dei due eserciti**. Supponiamo che l'esercito bianco sia accampato in una valle, come mostrato nella Figura 6.13. Sulle colline a entrambi i fianchi si trovano gli eserciti blu. L'esercito bianco è più grande di ciascuno degli eserciti blu, ma insieme gli eserciti blu sono più grandi dell'esercito bianco. Se ogni esercito blu lancia l'attacco da solo sarà sconfitto, ma se i due eserciti blu attaccano contemporaneamente saranno vittoriosi.

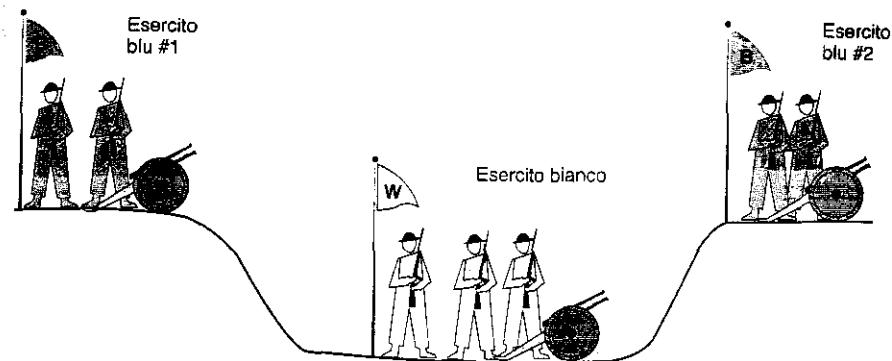


Figura 6.13. Il problema dei due eserciti.

Gli eserciti blu vogliono sincronizzare i loro attacchi. Tuttavia, il loro unico mezzo di comunicazione è l'invio di messaggeri a piedi nella valle, dove potrebbero essere catturati perdendo il messaggio (significa che devono utilizzare un canale di comunicazione inaffidabile). La domanda è: esiste un protocollo che consente agli eserciti blu di vincere?

Supponiamo che il comandante del primo esercito blu invii questo messaggio: "Propongo di attaccare all'alba del 29 marzo. Cosa ne pensate?". Ora supponiamo che il messaggio arrivi: il comandante del secondo esercito blu è d'accordo e la sua risposta viene consegnata al primo esercito blu. L'attacco avverrà? Probabilmente no, perché il comandante del secondo esercito non sa se la sua risposta è giunta a destinazione. Se non fosse arrivata il primo esercito blu non attaccherebbe, perché sarebbe insensato sostenere la battaglia.

Proviamo a migliorare il protocollo con un handshake a tre vie: l'autore della proposta originale deve dare l'acknowledgement alla risposta. Supponendo che non vengano persi messaggi, il secondo esercito blu riceverà l'acknowledgement, ma il comandante del primo esercito blu esiterà ancora. Dopo tutto, non sa se il proprio acknowledgement è giunto a destinazione: se non lo fosse, il secondo esercito blu non attaccherebbe.

Potremmo provare a creare un protocollo di handshake a quattro vie, ma non aiuterebbe comunque.

In effetti, si può provare per assurdo che non esiste alcun protocollo funzionante. Supponiamo che esista un protocollo di questo tipo: l'ultimo messaggio del protocollo potrebbe essere o meno fondamentale. Se non lo è, occorre rimuoverlo (insieme a tutti gli altri messaggi non fondamentali) fino a ottenere un protocollo in cui ogni messaggio è fondamentale. Che cosa accade se il messaggio finale non giunge a destinazione? Abbiamo già detto che è essenziale, per cui se viene perso l'attacco non può avvenire. Dal momento che il mittente del messaggio finale non può essere certo del suo arrivo, non rischierà l'attacco. Peggio ancora, l'altro esercito blu è a conoscenza di questo fatto e non lancerà l'attacco.

Per capire che il problema dei due eserciti è rilevante per il rilascio delle connessioni basta sostituire il verbo "disconnettere" ad "attaccare". Se entrambe le parti esigono la certezza che anche l'altra è pronta a terminare la connessione, la disconnessione non avverrà mai. In pratica si è disposti ad accettare più rischi quando c'è da rilasciare le connessioni piuttosto che quando si deve attaccare l'esercito bianco, pertanto la situazione non è poi senza speranze. La Figura 6.14 illustra quattro scenari di rilascio utilizzando un handshake a tre vie; anche se questo protocollo non è infallibile, di solito è adeguato.

Nella Figura 6.14(a) è mostrato il caso normale in cui uno degli utenti invia una TPDU DR (DISCONNECTION REQUEST) per iniziare il rilascio di una connessione. Quando arriva, il destinatario restituisce una TPDU DR e avvia un timer (servirà nel caso la sua DR vada persa). All'arrivo di questa DR, il mittente originale invia una TPDU ACK e rilascia la connessione. Per finire, quando arriva la TPDU ACK, il ricevitore rilascia la connessione. Rilasciare una connessione significa che l'entità di trasporto rimuove le informazioni sulla connessione dalla propria tabella di connessioni aperte e lo segnala al proprietario della connessione (l'utente del trasporto). Questa azione è diversa da un utente del trasporto che emette una primitiva DISCONNECT.

Se la TPDU ACK finale viene persa, come mostrato nella Figura 6.14(b), la situazione viene risolta dal timer. Alla scadenza del timer la connessione viene comunque rilasciata. Consideriamo ora il caso della perdita della seconda DR. L'utente che inizia la disconnessione non riceve la risposta prevista, si verifica un timeout e tutto ricomincia; la sequenza degli eventi è visibile nella Figura 6.14(c), facendo l'ipotesi che la seconda volta nessuna TPDU venga persa e che tutte le TPDU siano consegnate correttamente e in tempo.

L'ultimo scenario, mostrato nella Figura 6.14(d), equivale a quello della Figura 6.14(c), tranne per il fatto che ora si presume che i tentativi ripetuti di ritrasmettere la DR falliscono a causa delle TPDU perse. Dopo N tentativi il mittente rilascia la connessione, e nel frattempo si verifica un timeout nel ricevitore che a sua volta esce.

Questo protocollo è generalmente sufficiente, ma in teoria può fallire se vengono perse la DR iniziale e le N ritrasmissioni. Il mittente abbandonerà i tentativi e rilascerà la connessione, mentre l'altro lato non saprà nulla di tutti i tentativi di disconnessione e sarà pienamente attivo. Questa situazione provoca una connessione aperta a metà.

Avgremmo potuto evitare il problema non consentendo al mittente di abbandonare i tentativi dopo N volte, ma obbligandolo a proseguire fino al ricevimento di una risposta.

Lo strato trasporto

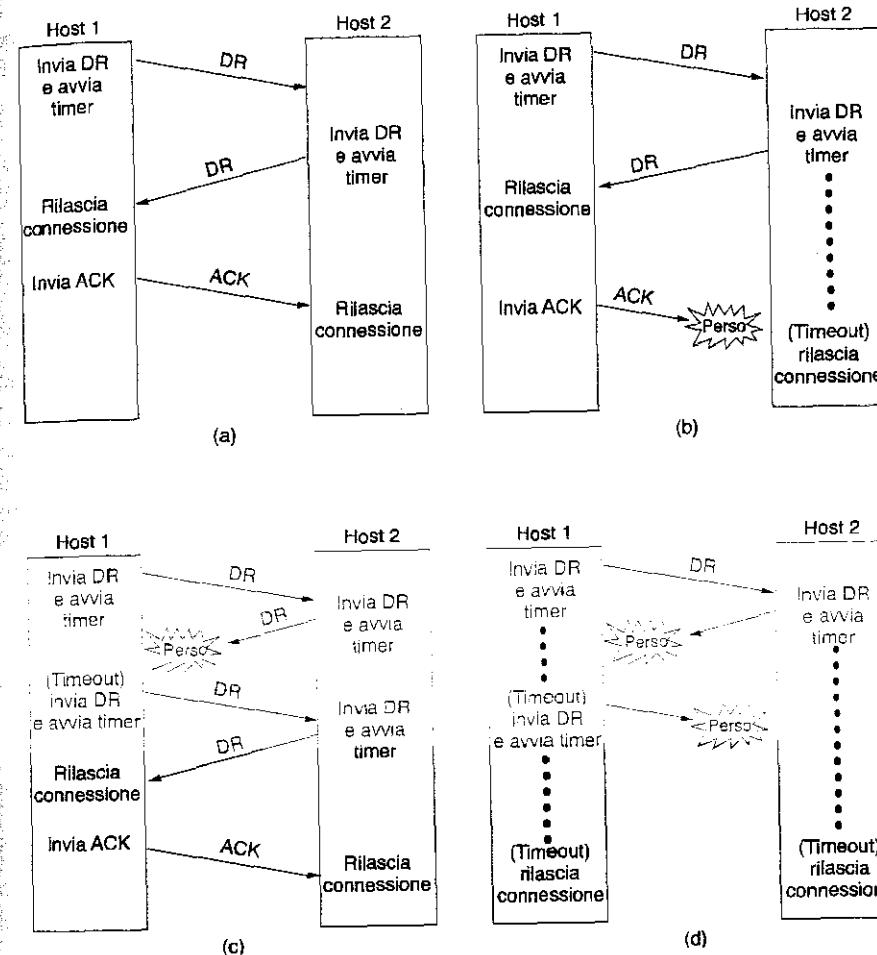


Figura 6.14. Quattro scenari per il rilascio di una connessione.
 (a) Caso normale di handshake a tre vie. (b) ACK finale perduto.
 (c) Risposta persa. (d) Risposta persa e DR successive perdeute.

Tuttavia, se si consente alla controparte di terminare per timeout, il mittente procederà all'infinito perché non riceverà mai risposta. Se non consentiamo il timeout del lato ricevente, il protocollo rimane sospeso come nella Figura 6.14(d).

Un modo per interrompere tutte le connessioni aperte a metà può essere la definizione di una regola: se non arrivano TPDU per un certo numero di secondi, la connessione viene automaticamente terminata. In questo modo, se un lato si disconnette, l'altro lato rileverà la mancanza di attività ed eseguirà a sua volta la disconnessione. Naturalmente, se viene introdotta questa regola, è necessario che ogni entità di trasporto abbia un timer che venga fermato e

ricaricato a ogni invio di TPDU. Se il timer scade, viene trasmessa una TPDU fittizia solo per impedire la disconnessione dell'altro lato. D'altra parte, se viene utilizzata la regola di disconnessione automatica e si perdono troppe TPDU fittizie consecutive su una connessione altrimenti inattiva, prima un lato e poi l'altro si disconnetteranno automaticamente. Non ci dilungheremo ulteriormente su questo punto, ma per ora dovrebbe essere chiaro che il rilascio di una connessione senza perdita di dati non è così semplice come potrebbe sembrare.

6.2.4 Il controllo di flusso e il buffering

Dopo avere esaminato in dettaglio la costituzione e il rilascio della connessione, vediamo come vengono gestite le connessioni durante il loro utilizzo. Una delle questioni chiave è già apparsa in precedenza: il controllo di flusso. Sotto certi aspetti il problema del controllo di flusso nello strato trasporto equivale a quello nello strato data link, ma per altri è differente. La somiglianza di fondo sta nel fatto che in entrambi gli strati su ogni connessione è necessaria una sliding window o un altro schema per impedire che un trasmettitore rapido sommerga un ricevitore lento. La differenza principale sta nel fatto che un router presenta di norma poche linee, mentre un host può avere numerose connessioni. Questa differenza rende poco pratica l'implementazione nello strato trasporto della strategia di buffering data link.

Nei protocolli data link del Capitolo 3 i frame erano sottoposti a buffering sia nel router di invio sia in quello di ricezione. Nel protocollo 6, per esempio, sia il mittente sia il ricevitore devono dedicare $MAX_SEQ + 1$ buffer a ogni linea, metà per l'input e metà per l'output. Per un host con un massimo di 64 connessioni e un numero di sequenza a 4 bit, per esempio, questo protocollo richiederebbe 1.024 buffer.

Nello strato data link la parte che trasmette deve inserire in un buffer i frame in uscita, perché potrebbe essere necessario ritrasmetterli. Se la sottrete offre un servizio datagram, per la stessa ragione anche l'entità di trasporto che trasmette deve utilizzare il buffer. Se il ricevitore sa che il mittente inserisce nel buffer tutte le TPDU fino a quando non riceve l'acknowledgement, potrebbe decidere se dedicare o no buffer specifici alle singole connessioni. Il ricevitore potrebbe, per esempio, mantenere un singolo pool di buffer condiviso da tutte le connessioni. All'arrivo di una TPDU, viene svolto un tentativo di acquisire dinamicamente un nuovo buffer. Se vi sono buffer disponibili, la TPDU viene accettata, altrimenti viene scartata. Dal momento che il mittente è preparato a ritrasmettere le TPDU perse dalla sottrete, non si provocano danni se il ricevitore scarta alcune TPDU (a parte lo spreco di risorse). Il mittente continua a provare fino a quando riceve un acknowledgement.

Riepilogando, se il servizio di rete è inaffidabile, il mittente deve inserire nel buffer tutte le TPDU inviate, come nello strato data link, ma con un servizio di rete affidabile diventano possibili altri compromessi. In particolare, se il mittente sa che il ricevitore ha sempre uno spazio nei buffer, non deve mantenere copie delle TPDU inviate. Tuttavia, se il ricevitore non può garantire l'accettazione di ogni TPDU in ingresso, il mittente dovrà comunque utilizzare il buffer. Nell'ultimo caso il mittente non può fidarsi dell'acknowledgement dello strato network, perché indica solo l'arrivo della TPDU e non la sua accettazione. Torneremo in seguito su questo importante punto.

Anche se il ricevitore accetta di fare buffering, resta da risolvere il problema della dimensione del buffer. Se la maggior parte delle TPDU ha una dimensione simile, è naturale organizzare i buffer sotto forma di pool di buffer con dimensione identica e una TPDU per buffer, come mostrato nella Figura 6.15(a). Tuttavia, se esiste una variazione notevole nella dimensione delle TPDU (da alcuni caratteri digitati su un terminale fino a migliaia di caratteri provenienti da un trasferimento file), un pool di buffer con dimensioni fisse potrebbe provocare problemi. Se la dimensione del buffer è scelta in base alla TPDU con la massima dimensione possibile, si verifica uno spreco di spazio all'arrivo di una TPDU breve. Se la dimensione del buffer è inferiore alla dimensione massima delle TPDU, si devono impiegare buffer multipli per le TPDU lunghe, incrementando la complessità. Un altro approccio al problema della dimensione del buffer è l'utilizzo di buffer con dimensioni variabili, come mostrato nella Figura 6.15(b). Il vantaggio è un migliore utilizzo della memoria, al prezzo di una gestione più complicata dei buffer. Una terza possibilità prevede di dedicare un singolo buffer circolare di grandi dimensioni per ogni connessione, come mostrato nella Figura 6.15(c). Questo sistema fa un valido utilizzo della memoria quando tutte le connessioni sono caricate in modo pesante, ma è scadente se alcune connessioni hanno poco carico.

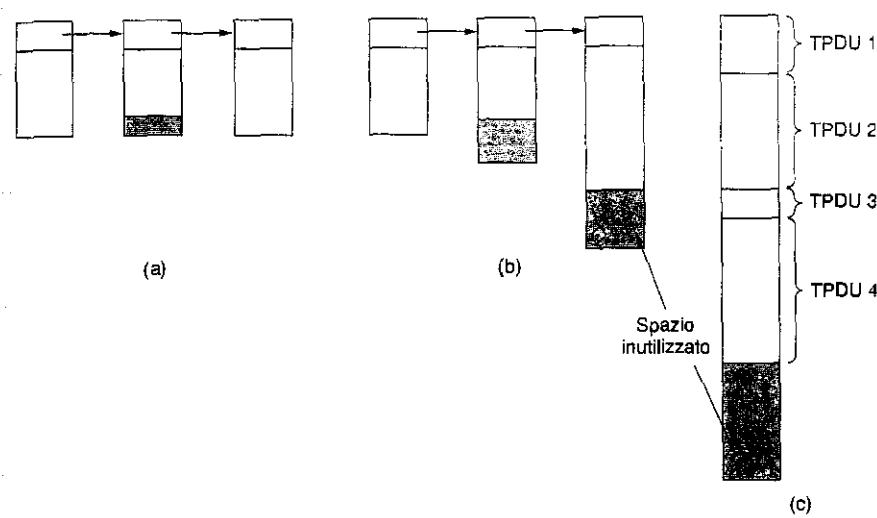


Figura 6.15. (a) Buffer concatenati a dimensione fissa. (b) Buffer concatenati a dimensione variabile. (c) Un grosso buffer circolare per ogni connessione.

Il miglior compromesso tra il buffering all'origine e quello alla destinazione dipende dal tipo di traffico sulla connessione. Per un traffico irregolare e con larghezza di banda ridotta, come quello prodotto da un terminale interattivo, è meglio non dedicare i buffer, ma acquisirli dinamicamente a entrambe le estremità. Dal momento che il mittente non può essere certo che il ricevitore sia in grado di acquisire un buffer, deve mantenere una copia della TPDU fino all'acknowledgement. Viceversa, per il trasferimento di file e il traffico

con larghezza di banda elevata è preferibile che il ricevitore dedichi un'intera window di buffer, per consentire il flusso dei dati alla massima velocità. Di conseguenza, per un traffico irregolare a larghezza di banda ridotta è preferibile eseguire il buffering nel mittente, mentre per un traffico uniforme a larghezza di banda elevata è preferibile eseguire il buffering nel ricevitore.

Con l'aprirsi e il chiudersi delle connessioni e il variare dello schema del traffico il mittente e il ricevitore devono regolare dinamicamente le allocazioni dei buffer. Di conseguenza, il protocollo di trasporto dovrebbe consentire a un host mittente di richiedere spazio nel buffer all'altra estremità. I buffer possono essere allocati per connessione, oppure collettivamente per tutte le connessioni in esecuzione tra i due host. In alternativa, il ricevitore che conosce la situazione dei suoi buffer (ma non il traffico offerto) potrebbe comunicare al mittente "Ti ho riservato X buffer". Se il numero di connessioni aperte aumenta può essere necessario ridurre un'allocazione, pertanto il protocollo deve prevedere questa possibilità.

Un modo generico per gestire l'allocazione dinamica dei buffer consiste nel separare il buffering dagli acknowledgement, a differenza dei protocolli sliding window del Capitolo 3. La gestione dinamica dei buffer si traduce in definitiva nell'uso di una window di dimensione variabile. Inizialmente il mittente richiede un certo numero di buffer, basato sulla previsione delle proprie esigenze, e il ricevitore garantisce il numero che può permettersi. Ogni volta che il mittente trasmette una TPDU decrementa la sua allocazione, fermandosi quando raggiunge il valore zero. Il ricevitore può quindi aggiungere separatamente gli acknowledgement e le allocazioni del buffer sul traffico che scorre nella direzione opposta.

La Figura 6.16 mostra un esempio di funzionamento della gestione dinamica delle window in una sottorete a datagrammi con numeri di sequenza a 4 bit. Supponiamo che le informazioni di allocazione del buffer viaggino in TPDU separate, come mostrato, e non siano appoggiate sul traffico che scorre in direzione opposta. Inizialmente A vuole otto buffer, ma gliene vengono garantiti solo quattro. Successivamente invia tre TPDU, di cui la terza viene persa. La TPDU 6 fornisce l'acknowledgement a tutte le TPDU precedenti (a partire da quella che ha numero di sequenza 1), consentendo ad A di rilasciare tali buffer; inoltre dà il permesso ad A per inviare altre tre TPDU dopo la 1 (vale a dire le TPDU 2, 3 e 4). Tuttavia A ha già inviato il numero 2, pertanto pensa che potrebbe inviare le TPDU 3 e 4 e procede in questo modo. A questo punto si blocca, in attesa di un'ulteriore allocazione dei buffer. Durante il blocco possono comunque avvenire ritrasmissioni indotte dal timeout (riga 9), dal momento che utilizzano i buffer già allocati. Nella riga 10, B conferma la ricezione di tutte le TPDU fino alla 4, ma impedisce ad A di continuare. Tale situazione è impossibile con i protocolli fixed window del Capitolo 3. La successiva TPDU da B ad A alloca un altro buffer e consente ad A di proseguire.

Nelle reti a datagrammi, se le TPDU di controllo vengono perse possono verificarsi problemi con gli schemi di allocazione dei buffer di questo tipo. Si osservi la riga 16. Ora B ha allocato altri buffer ad A, ma la TPDU di allocazione è andata persa. Dal momento che le TPDU di controllo non sono sottoposte a sequenza o timeout, A è in stallo. Per impedi-

A	Messaggio	B	Commenti
1	→ <request 8 buffers>	→	A desidera 8 buffer
2	→ <ack = 1, buf = 4>	→	B conferma l'arrivo solo dei messaggi 0-3
3	→ <seq = 0, data = m0>	→	A dispone di 3 buffer
4	→ <seq = 1, data = m1>	→	Adesso A dispone di 2 buffer
5	→ <seq = 2, data = m2>	...	Messaggio perso, ma A pensa che ne sia rimasto 1
6	→ <ack = 1, buf = 3>	→	B conferma l'arrivo di 0 e 1, e permette 2-4
7	→ <seq = 3, data = m3>	→	A dispone di 1 buffer
8	→ <seq = 4, data = m4>	→	A dispone di 0 buffer e deve fermarsi
9	→ <seq = 2, data = m2>	→	Timeout e ritrasmissione di A
10	→ <ack = 4, buf = 0>	→	Tutti gli acknowledgement sono arrivati, ma A è ancora bloccato
11	→ <ack = 4, buf = 1>	→	A ora può inviare 5
12	→ <ack = 4, buf = 2>	→	B ha trovato un nuovo buffer altrove
13	→ <seq = 5, data = m5>	→	A dispone di 1 buffer
14	→ <seq = 6, data = m6>	→	A è di nuovo bloccato
15	→ <ack = 6, buf = 0>	→	A è ancora bloccato
16	... <ack = 6, buf = 4>	→	Possibile stallo

Figura 6.16. Allocazione dinamica del buffer. Le frecce mostrano la direzione della trasmissione. I puntini di sospensione (...) indicano una TPDU persa.

re questa situazione, ogni host deve periodicamente inviare TPDU di controllo per fornire l'acknowledgement e lo stato del buffer di ogni connessione. In questo modo, prima o poi lo stallo verrà superato.

Finora abbiamo tacitamente presunto che l'unico limite imposto sulla velocità dei dati del mittente fosse la quantità di spazio del buffer disponibile nel ricevitore. Dal momento che il prezzo della memoria continua a diminuire, è possibile equipaggiare gli host con una quantità di memoria sufficiente a rendere la mancanza di buffer un problema raro.

Quando lo spazio per i buffer non limita più il flusso massimo, nasce un altro collo di bottiglia: la capacità di trasporto della sottorete. Se router adiacenti possono scambiare al massimo k pacchetti al secondo ed esistono k percorsi disgiunti tra una coppia di host, non vi è modo per tali host di scambiare più di kx TPDU al secondo, indipendentemente dalla quantità di buffer disponibile a ogni estremità. Se il mittente esegue l'invio troppo rapidamente (più di kx TPDU al secondo), la sottorete si congestionata perché non è in grado di consegnare le TPDU con la stessa velocità con cui le riceve.

È quindi necessario un meccanismo per regolare il flusso basato sulla capacità di trasporto della sottorete invece che sulla capacità dei buffer del ricevitore. Chiaramente questo meccanismo va applicato al mittente, per impedire la circolazione simultanea di troppe TPDU che non hanno ancora ricevuto acknowledgement. Belsnes (1975) ha proposto l'utilizzo di uno schema di controllo del flusso sliding window, in cui il mittente regola dinamicamente la dimensione della window per farla corrispondere alla capacità di trasporto della rete. Se la rete può gestire c TPDU al secondo e la durata del ciclo (comprensiva di trasmissione, propagazione, accodamento, elaborazione nel ricevitore e restituzione dell'acknowledgement) è r , la window del mittente dovrebbe corrispondere almeno a cr . Con

una window di questa dimensione, il mittente opera normalmente con il canale pieno. Un piccolo calo delle prestazioni di rete potrebbe provocarne il blocco.

Per regolare periodicamente la dimensione della window, il mittente potrebbe monitorare entrambi i parametri e poi calcolare la dimensione desiderata per la window. La capacità di trasporto può essere determinata con il semplice conteggio del numero di TPDU che hanno ricevuto acknowledgement in un dato periodo di tempo, e dividendo tale valore per il periodo esaminato. Durante la misura il mittente dovrebbe eseguire l'invio alla massima velocità; ciò assicura che il fattore limitante per la velocità di acknowledgement è la capacità di trasporto della rete, e non un valore ridotto della velocità di input. Il tempo richiesto per dare l'acknowledgement a una TPDU trasmessa è misurabile con precisione e se ne può calcolare una media mobile. Dal momento che la capacità di rete disponibile per un dato flusso varia nel tempo, la dimensione della window dovrebbe essere regolata frequentemente per tenere traccia dei cambiamenti nella capacità di trasporto. Come vedremo in seguito, Internet utilizza uno schema simile.

6.2.5 Il multiplexing

Il multiplexing di più conversazioni su connessioni, circuiti virtuali e collegamenti fisici svolge un ruolo importante in diversi strati dell'architettura di rete. Nello strato trasporto l'esigenza del multiplexing può sorgere in molti modi. Per esempio, se un host ha a disposizione un solo indirizzo di rete, tutte le connessioni di trasporto del computer devono utilizzare tale indirizzo. Quando arriva una TPDU, è necessario stabilire in qualche modo a quale processo attribuirla. Questa situazione, chiamata **upward multiplexing**, è mostrata nella Figura 6.17(a). In questa figura, quattro connessioni di trasporto distinte utilizzano la stessa connessione di rete (per esempio l'indirizzo IP) con l'host remoto.

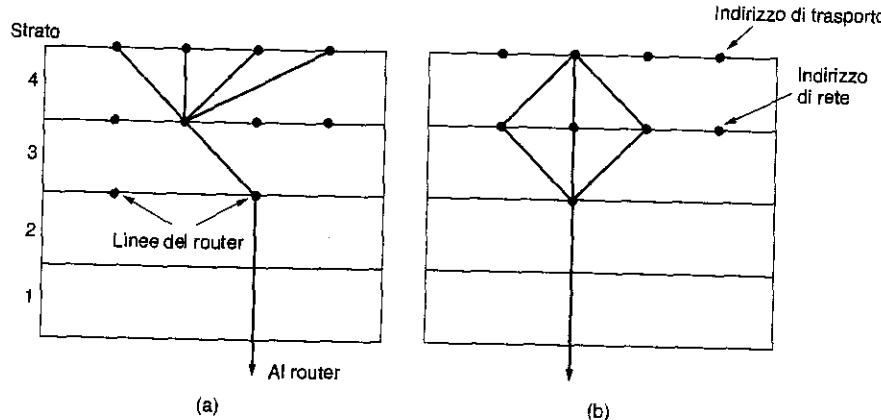


Figura 6.17. (a) Upward multiplexing. (b) Downward multiplexing.

Il multiplexing può essere utile nello strato trasporto anche per un altro motivo. Si supponga, per esempio, che una sottorete usi internamente dei circuiti virtuali e imponga una velocità massima dei dati su ognuno. Se un utente necessita di una larghezza di banda superiore a quella che un circuito virtuale può fornire, una soluzione consiste nell'aprire più connessioni di rete e distribuire il traffico tra esse su base round-robin, come indicato nella Figura 6.17. Questo metodo di operare è definito **downward multiplexing**. Con k connessioni di rete aperte, la larghezza di banda effettiva è aumentata di un fattore k . Un popolare esempio di downward multiplexing è costituito dagli utenti residenziali con linea ISDN. Questa linea fornisce due connessioni separate di 64 Kbps ognuna. Utilizzandole entrambe per chiamare un provider Internet e ripartendo il traffico su entrambe le linee è possibile ottenere una larghezza di banda effettiva di 128 Kbps.

6.2.6 Il ripristino dopo un crash

Se host e router sono soggetti a crash, il ripristino può diventare un problema. Quando l'entità di trasporto è interamente all'interno degli host, il ripristino dopo un crash di rete o del router è banale. Se lo strato network fornisce un servizio datagram, le entità di trasporto prevedono la perdita di TPDU e sanno come gestirla. Se lo strato network fornisce un servizio orientato alla connessione, la perdita di un circuito virtuale è gestita stabilendone uno nuovo e quindi sondando l'entità di trasporto remota per chiederle quali TPDU ha ricevuto e quali non sono ancora giunte a destinazione; queste ultime possono essere ritrasmesse.

Un problema più complesso riguarda il ripristino dopo un crash dell'host. In particolare si potrebbe esigere che i client siano in grado di continuare a lavorare quando i server subiscono un crash e vengono rapidamente riavviati. Per illustrare la difficoltà, supponiamo che un host, il client, stia inviando un lungo file a un altro host, il file server, utilizzando un semplice protocollo stop-and-wait. Lo strato trasporto sul server si limita a passare le TPDU in arrivo all'utente del trasporto, una per una. Durante la trasmissione, il server subisce un crash. Dopo il riavvio le sue tabelle vengono reinizializzate, pertanto il server non sa più precisamente a che punto era arrivato.

In un tentativo di recuperare il suo stato precedente, il server potrebbe inviare una TPDU broadcast a tutti gli altri host, annunciando di aver subito un crash e richiedendo che i client lo informino dello stato delle connessioni aperte. Ogni client può trovarsi in uno dei due stati: *S1*, con una TPDU in circolazione, o *S0*, senza TPDU in circolazione. In base solo a queste informazioni di stato, il client deve decidere se ritrasmettere la TPDU più recente.

A prima vista il risultato appare ovvio: il client dovrebbe ritrasmettere se e solo se quando viene informato del crash ha circolazione una TPDU che non ha ancora ricevuto acknowledgement (stato *S1*). Tuttavia, un'analisi più attenta scopre alcune difficoltà di questo approccio. Si consideri, per esempio, la situazione in cui l'entità di trasporto del server invia prima l'acknowledgement e in seguito, dopo l'invio del riconoscimento, scrive nel processo applicativo. La scrittura di una TPDU nel flusso di output e l'invio di un

acknowledgement sono due eventi distinti che non possono essere svolti contemporaneamente. Se si verifica un crash dopo l'invio dell'acknowledgement ma prima della scrittura, il client riceverà l'acknowledgement e pertanto si troverà nello stato *S0* quando giunge l'annuncio di ripristino dal crash. Il client non eseguirà di nuovo la trasmissione, pensando (non correttamente) che la TPDU è arrivata. Questa decisione del client porta a una TPDU mancante.

A questo punto i lettori potrebbero pensare: "Il problema può essere risolto facilmente. È sufficiente riprogrammare l'entità di trasporto per svolgere prima la scrittura e poi inviare l'acknowledgement". Riproviamo. Si supponga che la scrittura sia stata eseguita ma che il crash avvenga prima dell'invio dell'acknowledgement. Il client sarà nello stato *S1* ed eseguirà di nuovo la trasmissione, portando a una TPDU duplicata non rilevata nel flusso di output verso il processo applicativo del server.

Indipendentemente dalla programmazione del client e del server, esistono sempre situazioni in cui i protocolli falliscono il corretto ripristino. Il server può essere programmato in due modi: prima il riconoscimento o prima la scrittura. Il client può essere programmato in quattro modi: per ritrasmettere sempre l'ultima TPDU, per non ritrasmettere mai l'ultima TPDU, per ritrasmetterla solo nello stato *S0* o per ritrasmetterla solo nello stato *S1*. Si ottengono così otto combinazioni: come potremo vedere, per ogni combinazione esiste un insieme di eventi che provocano il fallimento del protocollo.

Per il server sono possibili tre eventi: invio di un acknowledgement (A), scrittura nel processo di output (W) e crash (C). I tre eventi possono avvenire in sei ordini diversi: *AC(W)*, *AWC*, *C(AW)*, *C(WA)*, *WAC* e *WC(A)* (le parentesi sono usate per indicare che né A né W possono seguire C). La Figura 6.18 mostra le otto combinazioni della strategia per client e server e le sequenze di eventi valide per ognuno. Occorre notare che per ogni strategia esiste una sequenza di eventi che provoca il fallimento del protocollo. Per esempio, se il client esegue sempre la ritrasmissione, l'evento *AWC* genera un duplicato non rilevato, anche se gli altri due eventi funzionano correttamente.

Elaborando ulteriormente il protocollo non si ottengono risultati. Anche se il client e il server scambiano diverse TPDU prima che il server tenti di eseguire la scrittura, in modo che il client sappia esattamente cosa sta per accadere, il client non ha modo di sapere se un crash è avvenuto appena prima o subito dopo la scrittura. La conclusione è inevitabile: se postuliamo la mancata simultaneità degli eventi, il crash e il ripristino dell'host non possono essere resi trasparenti agli strati superiori.

In termini più generici, questo risultato può essere esposto come segue: un ripristino dal crash di uno strato *N* può essere svolto solo dallo strato *N + 1*, e solo se il livello superiore dispone di informazioni di stato sufficienti. Come affermato in precedenza, lo strato trasporto può ripristinare gli errori nello strato network, purché ciascun estremo di una connessione tenga traccia del punto in cui è arrivato.

Questo problema ci porta ad affrontare la questione del vero significato di un acknowledgement end-to-end. In linea di principio, il protocollo di trasporto è di tipo end-to-end, e non di tipo concatenato come gli strati inferiori. Ora si consideri il caso di un utente che inserisce richieste di transazioni su un database remoto. Si supponga che l'entità di tra-

			Strategia utilizzata dall'host che riceve					
			Prima ACK, poi scrittura			Prima scrittura, poi ACK		
Strategia utilizzata dall'host che invia			AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Ritrasmetti sempre	OK	DUP	OK			OK	DUP	DUP
Non ritrasmettere mai	LOST	OK	LOST			LOST	OK	OK
Ritrasmette in S0	OK	DUP	LOST			LOST	DUP	OK
Ritrasmette in S1	LOST	OK	OK			OK	OK	DUP

OK = il protocollo funziona correttamente
DUP = il protocollo genera un messaggio duplicato
LOST = il protocollo perde un messaggio

Figura 6.18. Le diverse combinazioni di strategie per client e server.

sporto remota sia programmata per passare prima le TPDU allo strato superiore e poi inviare l'acknowledgment. Anche in questo caso, la ricezione di un acknowledgement sulla macchina dell'utente non significa necessariamente che l'host remoto è rimasto attivo per il tempo sufficiente ad aggiornare il database. Un vero acknowledgement end-to-end, la cui ricezione indica che il lavoro è stato svolto e la cui mancanza indica il contrario, è praticamente impossibile da raggiungere. Questo punto è discusso in maggiore dettaglio da Saltzer et al. (1984).

6.3 Un semplice protocollo di trasporto

Per rendere più concrete le idee discusse finora, in questo paragrafo studieremo in dettaglio uno strato trasporto di esempio. Le primitive di servizio astratte che saranno utilizzate corrispondono alle primitive orientate alla connessione della Figura 6.2. La scelta di queste primitive orientate alla connessione rende l'esempio simile al popolare protocollo TCP, ma più semplice.

6.3.1 Esempio di primitive di servizio

Il primo problema riguarda l'espressione concreta di queste primitive di trasporto. CONNECT è facile: disponiamo di una procedura di libreria *connect* che può essere chiamata con i parametri necessari per stabilire una connessione. I parametri sono TSAP locale e remoto. Durante la chiamata, il chiamante viene bloccato (o sospeso) mentre l'entità di trasporto cerca di impostare la connessione. Se la connessione ha successo, il chiamante viene sbloccato e può iniziare a trasmettere dati.

Quando un processo desidera accettare chiamate in ingresso, esegue la chiamata a *listen* specificando il particolare TSAP su cui eseguire l'ascolto. Il processo resta bloccato fino a quando qualche processo remoto tenta di stabilire una connessione al suo TSAP.

Osserviamo che questo modello è altamente asimmetrico. Un lato è passivo ed esegue un *listen* attendendo che accada qualcosa. L'altro lato è attivo e inizia la connessione. Una domanda interessante riguarda le operazioni da svolgere se il lato attivo inizia per primo. Una strategia prevede di fare fallire il tentativo di connessione se non vi sono processi in ascolto nello TSAP remoto. Un'altra strategia richiede di bloccare chi inizia (magari in eterno) fino alla comparsa di un ascoltatore.

Un compromesso, utilizzato nel nostro esempio, richiede di mantenere la richiesta di connessione all'estremità ricevente per un certo intervallo di tempo. Se un processo su tale host esegue la chiamata a *listen* prima della scadenza del timer viene stabilita la connessione; in caso contrario, la connessione viene rifiutata, il chiamante viene sbloccato e viene restituito un errore.

Per rilasciare una connessione si utilizzerà una procedura *disconnect*. Quando entrambi i lati saranno disconnessi verrà rilasciata la connessione; in altre parole, si utilizza un modello di disconnessione simmetrico.

La trasmissione dei dati presenta lo stesso problema della costituzione della connessione: l'invio è attivo ma la ricezione è passiva. Utilizzeremo per la trasmissione dei dati la stessa soluzione scelta per la costituzione della connessione: una chiamata attiva *send* che trasmette i dati e una chiamata passiva *receive* che crea un blocco fino all'arrivo di una TPDU. La definizione concreta del servizio consiste quindi di cinque primitive: CONNECT, LISTEN, DISCONNECT, SEND e RECEIVE. Ogni primitiva corrisponde esattamente a una procedura di libreria che esegue la primitiva. I parametri per le primitive di servizio e le procedure di libreria sono i seguenti:

```
connum = LISTEN(local)
connum = CONNECT(local, remote)
status = SEND(connum, buffer, bytes)
status = RECEIVE(connum, buffer, bytes)
status = DISCONNECT(connum)
```

La primitiva LISTEN annuncia il desiderio del chiamante di accettare le richieste di connessione dirette al TSAP indicato. L'utente della primitiva è bloccato fino a quando viene svolto un tentativo di connessione ad esso. Non esiste un timeout.

La primitiva CONNECT accetta due parametri, un TSAP locale (per esempio l'indirizzo di trasporto) *local* e un TSAP remoto *remote*, e prova a stabilire una connessione di trasporto tra i due. Se riesce, restituisce in *connum* un numero non negativo utilizzato per identificare la connessione nelle chiamate successive. Se fallisce, il motivo del fallimento è riportato in *connum* come numero negativo. In questo modello semplificato ogni TSAP può partecipare a una sola connessione di trasporto, pertanto una possibile causa di fallimento è che uno degli indirizzi di trasporto sia già in uso. Altre ragioni sono l'inattività dell'host remoto oppure un indirizzo locale o remoto non valido.

La primitiva SEND trasmette il contenuto del buffer come un messaggio sulla connessione di trasporto indicata, se necessario diviso in più unità. Gli errori possibili, restituiti in *status*, sono la mancata connessione, l'indirizzo del buffer non valido o un conteggio negativo.

La primitiva RECEIVE indica il desiderio del chiamante di accettare dati. La dimensione del messaggio in arrivo è inserita in *bytes*. Se il processo remoto ha rilasciato la connessione o l'indirizzo del buffer non è valido (per esempio esterno al programma dell'utente), *status* viene caricato con un codice di errore che indica la natura del problema.

La primitiva DISCONNECT termina una connessione di trasporto indicata dal parametro *connum*. Gli errori possibili riguardano l'appartenenza di *connum* a un altro processo o il fatto che *connum* non sia un identificatore di connessione valido. Il codice di errore viene restituito in *status*; 0 indica la riuscita.

6.3.2 Esempio di entità di trasporto

Prima di studiare il codice dell'entità di trasporto, si deve comprendere che questo esempio assomiglia a quelli presentati precedentemente nel Capitolo 3: più che una proposta seria, ha uno scopo pedagogico. Molti dettagli tecnici (come un controllo degli errori estensivo) che sarebbero necessari in un sistema di produzione sono stati omessi per semplicità.

Lo strato trasporto utilizza le primitive del servizio network per inviare e ricevere TPDU. Per questo esempio bisogna scegliere quali primitive del servizio network utilizzare. Una scelta poteva essere il servizio datagram non affidabile, ma per mantenere la semplicità si è deciso di scartarlo. Con un servizio datagram inaffidabile il codice di trasporto sarebbe stato lungo e complesso, perché avrebbe dovuto gestire pacchetti persi e in ritardo. Inoltre, la maggior parte di queste tecniche è già stata discussa a lungo nel Capitolo 3.

Abbiamo invece scelto di utilizzare un servizio di rete affidabile orientato alla connessione. In questo modo è possibile concentrarci sulle questioni di trasporto che non avvengono nei livelli inferiori, come la costituzione della connessione, il rilascio della connessione e la gestione dei crediti. Ciò potrebbe assomigliare a un semplice servizio di trasporto costruito su una rete ATM.

In generale, l'entità di trasporto potrebbe essere parte del sistema operativo dell'host, oppure potrebbe essere un package di routine di libreria all'interno dello spazio d'indirizzamento dell'utente. Per semplicità, questo esempio è stato programmato come se fosse un package di libreria, ma le modifiche necessarie per renderlo parte del sistema operativo sono minime (riguardano principalmente l'accesso ai buffer dell'utente).

Vale la pena notare, però, che in questo esempio l'entità di trasporto non è realmente un'entità separata, ma fa parte del processo utente. In particolare, quando l'utente esegue una primitiva che provoca un blocco come LISTEN, viene bloccata anche l'intera entità di trasporto. Anche se questa struttura è perfetta per un host con un solo processo relativo a un singolo utente, su un host con più utenti sarebbe più naturale avere un'entità di trasporto strutturata come processo separato, distinto da tutti i processi utente.

L'interfaccia per lo strato network richiede le procedure *to_net* e *from_net* (non mostrate). Ognuna ha sei parametri: per prima cosa c'è l'identificatore della connessione, che viene associato a ogni circuito virtuale di rete; seguono i bit *Q* e *M* che, quando sono impostati a 1, indicano rispettivamente un messaggio di controllo e la presenza di altri dati di que-

sto messaggio nel pacchetto successivo. Di seguito abbiamo il tipo di pacchetto, scelto dall'insieme dei sei tipi elencati nella Figura 6.19. Per finire, è presente un puntatore ai dati veri e propri e un intero che comunica il numero di byte di dati.

Pacchetto di rete	Significato
CALL REQUEST	Inviato per stabilire una connessione
CALL ACCEPTED	Risposta a CALL REQUEST
CLEAR REQUEST	Inviato per rilasciare una connessione
CLEAR CONFIRMATION	Risposta a CLEAR REQUEST
DATA	Utilizzato per trasportare i dati
CREDIT	Pacchetto di controllo per gestire la window

Figura 6.19. I pacchetti dello strato network utilizzati nell'esempio.

Nelle chiamate a *to_net*, l'entità di trasporto compila tutti i parametri che devono essere letti dallo strato network; nelle chiamate a *from_net*, lo strato network scomponete un pacchetto in ingresso per passarlo all'entità di trasporto. Passando le informazioni come parametri della procedura, anziché fornire i pacchetti veri e propri in ingresso o in uscita, lo strato trasporto viene protetto dai dettagli del protocollo dello strato network. Se l'entità di trasporto dovesse tentare di inviare un pacchetto quando la sliding window del circuito virtuale sottostante è piena, sarebbe sospesa all'interno di *to_net* fino a quando si libera spazio nella window. Questo meccanismo è interamente trasparente rispetto all'entità di trasporto, ed è controllato dallo strato network utilizzando comandi analoghi a *enable_transport_layer* e *disable_transport_layer*, utilizzati nei protocolli del Capitolo 3. La gestione della window per lo strato di pacchetti è svolta sempre dallo strato network. Oltre a questo meccanismo di sospensione trasparente, l'entità di trasporto esegue anche chiamate a procedure esplicite *sleep* e *wakeup* (non mostrate). La procedura *sleep* viene chiamata quando l'entità di trasporto è bloccata in modo logico, in attesa di un evento esterno (generalmente l'arrivo di un pacchetto). Dopo la chiamata a *sleep*, l'entità di trasporto (e il processo utente, naturalmente) interrompe la sua esecuzione. Il codice dell'entità di trasporto è mostrato nella Figura 6.20. Ogni connessione si trova sempre in uno dei sette stati, come segue:

1. IDLE (inattiva): la connessione non è ancora stata stabilita
2. WAITING (in attesa): CONNECT è stato eseguito e CALL REQUEST inviato
3. QUEUED (accodata): CALL REQUEST è arrivato; nessun LISTEN
4. ESTABLISHED (stabilita): la connessione è stata stabilita
5. SENDING (invio): l'utente attende l'autorizzazione per inviare un pacchetto
6. RECEIVING (ricezione): è stato eseguito RECEIVE
7. DISCONNECTING (disconnessione): è stato eseguito DISCONNECT localmente.

Le transizioni tra gli stati avvengono in corrispondenza dei seguenti eventi: viene eseguita una primitiva, arriva un pacchetto o scade il timer.

Le procedure mostrate nella Figura 6.20 sono di due tipi. La maggior parte può essere chiamata direttamente dai programmi utente. *Packet_arrival* e *clock* sono comunque diverse. Vengono spontaneamente innestate da eventi esterni, rispettivamente l'arrivo di un pacchetto o il battito del clock, e in pratica si tratta di routine di interrupt. Supporremo che non vengano mai invocate durante l'esecuzione di una procedura dell'entità di trasporto. Possono essere chiamate solo quando il processo utente è inattivo o eseguito all'esterno dell'entità di trasporto. Questa proprietà è fondamentale per il corretto funzionamento del codice.

L'esistenza del bit *Q* (*Qualifier*, qualificatore) nell'intestazione del pacchetto consente di evitare la trasmissione di un'intestazione del protocollo di trasporto. I messaggi con dati ordinari sono inviati come pacchetti di dati con *Q* = 0. I messaggi di controllo del protocollo di trasporto, di cui ne esiste una sola istanza nell'esempio (CREDIT), sono inviati come pacchetti di dati con *Q* = 1. Questi messaggi di controllo sono rilevati ed elaborati dall'entità di trasporto ricevente.

La struttura dati principale utilizzata dall'entità di trasporto è l'array *conn*, che contiene un record per ogni potenziale connessione. Il record mantiene lo stato della connessione, compresi gli indirizzi di trasporto a entrambe le estremità, il numero di messaggi inviati e ricevuti sulla connessione, lo stato corrente, il puntatore al buffer dell'utente, il numero di byte del messaggio corrente inviati o ricevuti finora, un bit che indica se l'utente remoto ha emesso un DISCONNECT, un timer e un contatore di autorizzazioni utilizzato per consentire l'invio dei messaggi. Non tutti questi campi vengono utilizzati nel nostro semplice esempio, ma un'entità di trasporto completa li richiederebbe tutti (e forse anche di più). Si presume che ogni voce di *conn* sia inizializzato allo stato *IDLE*.

Quando l'utente esegue la chiamata a CONNECT, lo strato network riceve l'ordine di inviare un pacchetto CALL REQUEST alla macchina remota, poi l'utente viene forzato in attesa. Quando il pacchetto CALL REQUEST giunge all'altro lato, l'entità di trasporto viene interrotta per eseguire *packet_arrival* e controllare se l'utente locale sta eseguendo l'ascolto sull'indirizzo specificato. In questo caso, viene restituito un pacchetto CALL ACCEPTED e l'utente remoto viene risvegliato; in caso contrario CALL REQUEST viene accodato per *TIMEOUT* battiti del clock. Se in questo periodo viene eseguito un LISTEN la connessione è stabilita; in caso contrario si verifica un timeout e la connessione viene rifiutata con un pacchetto CLEAR REQUEST affinché non si blocchi per sempre.

Anche se è stata eliminata l'intestazione del protocollo di trasporto, è pur sempre necessario un modo per tenere traccia di quale pacchetto appartiene a quale connessione di trasporto, dal momento che possono esistere contemporaneamente più connessioni. L'approccio più semplice prevede di utilizzare il numero del circuito virtuale dello strato network come numero della connessione di trasporto. Inoltre, il numero del circuito virtuale può essere utilizzato anche come indice nell'array *conn*. Quando un pacchetto giunge sul circuito virtuale *k* dello strato network, appartiene alla connessione di trasporto *k*, il cui stato è contenuto nel record *conn[k]*. Per le connessioni iniziate da un host, il numero di connessione è scelto dall'entità di trasporto originaria.

```

#define MAX_CONN 32           /* numero massimo di connessioni simultanee */
#define MAX_MSG_SIZE 8192     /* massima dimensione messaggio in byte */
#define MAX_PKT_SIZE 512      /* massima dimensione pacchetto in byte */
#define TIMEOUT 20
#define CRED 1
#define OK 0

#define ERR_FULL -1
#define ERR_REJECT -2
#define ERR_CLOSED -3
#define LOW_ERR -3

typedef int transport_address;
typedef enum {CALL_REQ,CALL_ACC,CLEAR_REQ,CLEAR_CONF,DATA_PKT,CREDIT} pkt_type;
typedef enum {IDLE,WAITING,QUEUED,ESTABLISHED,SENDING,RECEIVING,DISCONN} cstate;

/* Variabili globali. */
transport_address listen_address;      /* indirizzo locale su cui viene eseguito l'ascolto */
int listen_conn;                      /* identificatore di connessione per listen */
unsigned char data[MAX_PKT_SIZE];    /* memoria di appoggio per i dati dei pacchetti */

struct conn {
    transport_address local_address, remote_address;
    cstate state;                      /* stato di questa connessione */
    unsigned char *user_buf_addr;       /* puntatore al buffer di ricezione */
    int byte_count;                   /* conteggio invii/ricezioni */
    int clr_req_received;            /* impostato alla ricezione del pacchetto CLEAR_REQ */
    int timer;                        /* utilizzato per il timeout dei pacchetti CALL_REQ */
    int credits;                      /* numero di messaggi che possono essere inviati */
} conn[MAX_CONN + 1];                 /* lo slot 0 non è utilizzato */

void sleep(void);                     /* prototypes */
void wakeup(void);
void to_net(int cid, int q, int m, pkt_type pt, unsigned char *p, int bytes);
void from_net(int *cid, int *q, int *m, pkt_type *pt, unsigned char *p, int *bytes);

int listen(transport_address t)
{ /* L'utente vuole ascoltare una connessione. Controlla se CALL_REQ è già arrivato. */
    int i, found = 0;

    for (i = 1; i <= MAX_CONN; i++) /* cerca CALL_REQ nella tabella */
        if (conn[i].state == QUEUED && conn[i].local_address == t) {
            found = i;
            break;
        }

    if (found == 0) {
        /* Nessun CALL_REQ in attesa. Torna inattivo fino all'arrivo o al timeout. */
        listen_address = t; sleep(); i = listen_conn;
    }
}

conn[i].state = ESTABLISHED;          /* connessione stabilita */
conn[i].timer = 0;                   /* il timer non è utilizzato */

```

```

listen_conn = 0;                      /* si presume che 0 sia un indirizzo non valido */
to_net(i, 0, 0, CALL_ACC, data, 0);   /* comunica alla rete di accettare la connessione */
return(i);                            /* restituisce l'identificatore di connessione */
}

int connect(transport_address l, transport_address r)
{ /* L'utente vuole connettersi a un processo remoto; invia il pacchetto CALL_REQ. */
    int i;
    struct conn *cptr;

    data[0] = r; data[1] = l;           /* il pacchetto CALL_REQ li richiede */
    i = MAX_CONN;                    /* cerca nella tabella, al contrario */
    while (conn[i].state != IDLE && i > 1) i = i - 1;
    if (conn[i].state == IDLE) {
        /* Crea una voce della tabella relativa all'invio di CALL_REQ. */
        cptr = &conn[i];
        cptr->local_address = l; cptr->remote_address = r;
        cptr->state = WAITING; cptr->clr_req_received = 0;
        cptr->credits = 0; cptr->timer = 0;
        to_net(i, 0, 0, CALL_REQ, data, 2);
        sleep();                      /* attende CALL_ACC o CLEAR_REQ */
        if (cptr->state == ESTABLISHED) return(i);
        if (cptr->clr_req_received) {
            /* L'altro lato ha rifiutato la chiamata. */
            cptr->state = IDLE;      /* torna allo stato IDLE */
            to_net(i, 0, 0, CLEAR_CONF, data, 0);
            return(ERR_REJECT);
        }
    } else return(ERR_FULL);         /* rifiuta CONNECT: non c'è spazio nella tabella */
}

int send(int cid, unsigned char bufptr[], int bytes)
{ /* L'utente vuole inviare un messaggio. */
    int i, count, m;
    struct conn *cptr = &conn[cid];

    /* Entra nello stato SENDING. */
    cptr->state = SENDING;
    cptr->byte_count = 0;             /* # byte inviati finora di questo messaggio */
    if (cptr->clr_req_received == 0 && cptr->credits == 0) sleep();
    if (cptr->clr_req_received == 0) {
        /* Credito disponibile; divide il messaggio in pacchetti se necessario. */
        do {
            if (bytes - cptr->byte_count > MAX_PKT_SIZE) /* messaggio di più pacchetti */
                count = MAX_PKT_SIZE; m = 1; /* seguiranno altri pacchetti */
            else {                                /* messaggio a pacchetto singolo */
                count = bytes - cptr->byte_count; m = 0; /* ultimo pacchetto di questo messaggio */
            }
            for (i = 0; i < count; i++) data[i] = bufptr[cptr->byte_count + i];
            to_net(cid, 0, m, DATA_PKT, data, count); /* invia 1 pacchetto */
            cptr->byte_count = cptr->byte_count + count; /* incrementa i byte inviati finora */
        } while (cptr->byte_count < bytes); /* esegue un ciclo fino a inviare l'intero messaggio */
}

```

```

cptr->credits--;
cptr->state = ESTABLISHED;
return(OK);
} else {
    cptr->state = ESTABLISHED;
    return(ERR_CLOSED); /* invio non riuscito: il peer vuole disconnettersi */
}

int receive(int cid, unsigned char bufptr[], int *bytes)
/* L'utente è pronto per ricevere un messaggio. */
struct conn *cptr = &conn[cid];

if (cptr->clr_req_received == 0) {
    /* Connessione ancora attiva; cerca di ricevere. */
    cptr->state = RECEIVING;
    cptr->user_buf_addr = bufptr;
    cptr->byte_count = 0;
    data[0] = CRED;
    data[1] = 1;
    to_net(cid, 1, 0, CREDIT, data, 2); /* invia il credito */
    sleep(); /* blocco in attesa dei dati */
    *bytes = cptr->byte_count;
}
cptr->state = ESTABLISHED;
return(cptr->clr_req_received ? ERR_CLOSED : OK);
}

int disconnect(int cid)
/* L'utente vuole rilasciare una connessione. */
struct conn *cptr = &conn[cid];

if (cptr->clr_req_received) { /* l'altro lato inizia la terminazione */
    cptr->state = IDLE;
    /* la connessione viene rilasciata */
    to_net(cid, 0, 0, CLEAR_CONF, data, 0);
} else { /* la terminazione iniziata da noi */
    cptr->state = DISCONN;
    /* non viene rilasciata fino a quando l'altro lato è d'accordo */
    to_net(cid, 0, 0, CLEAR_REQ, data, 0);
}
return(OK);

void packet_arrival(void)
/* È arrivato un pacchetto, che viene elaborato. */
int cid; /* la connessione su cui è arrivato il pacchetto */
int count, i, q, m;
pkt_type ptype; /* CALL_REQ,CALL_ACC,CLEAR_REQ,CLEAR_CONF,DATA_PKT,CREDIT */
unsigned char data[MAX_PKT_SIZE]; /* porzione dati del pacchetto in arrivo */
struct conn *cptr;

from_net(&cid, &q, &m, &ptype, data, &count);
cptr = &conn[cid];
switch (ptype) {
}

```

```

case CALL_REQ: /* l'utente remoto vuole stabilire la connessione */
    cptr->local_address = data[0]; cptr->remote_address = data[1];
    if (cptr->local_address == listen_address) {
        listen_conn = cid; cptr->state = ESTABLISHED; wakeup();
    } else {
        cptr->state = QUEUED; cptr->timer = TIMEOUT;
    }
    cptr->clr_req_received = 0; cptr->credits = 0;
    break;

case CALL_ACC: /* l'utente remoto ha accettato la nostra CALL_REQ */
    cptr->state = ESTABLISHED;
    wakeup();
    break;

case CLEAR_REQ: /* l'utente remoto vuole disconnettersi o rifiutare la chiamata */
    cptr->clr_req_received = 1;
    if (cptr->state == DISCONN) cptr->state = IDLE; /* cancella la collisione */
    if (cptr->state == WAITING || cptr->state == RECEIVING || cptr->state == SENDING) wakeup();
    break;

case CLEAR_CONF: /* l'utente remoto è d'accordo sulla disconnessione */
    cptr->state = IDLE;
    break;

case CREDIT: /* l'utente remoto è in attesa dei dati */
    cptr->credits += data[1];
    if (cptr->state == SENDING) wakeup();
    break;

case DATA_PKT: /* l'utente remoto ha inviato dati */
    for (i = 0; i < count; i++) cptr->user_buf_addr[cptr->byte_count + i] = data[i];
    cptr->byte_count += count;
    if (m == 0) wakeup();
}
}

void clock(void)
/* Controlla il timeout delle richieste di connessione accodate. */
int i;
struct conn *cptr;
for (i = 1; i <= MAX_CONN; i++) {
    cptr = &conn[i];
    if (cptr->timer > 0) { /* il timer è in esecuzione */
        cptr->timer--;
        if (cptr->timer == 0) { /* il timer è scaduto */
            cptr->state = IDLE;
            to_net(i, 0, 0, CLEAR_REQ, data, 0);
        }
    }
}

```

Figura 6.20. Un'entità di trasporto di esempio.

Per le chiamate in ingresso, è lo strato network a eompiere la scelta selezionando qualsiasi

numero di circuito virtuale inutilizzato.

Per evitare di dover fornire e gestire i buffer all'interno dell'entità di trasporto, si utilizza un meccanismo di controllo del flusso diverso dalla normale sliding window.

Quando un utente esegue la chiamata a RECEIVE, uno speciale **messaggio di credito** è inviato all'entità di trasporto sulla macchina che trasmette ed è registrato nell'array *conn*. Quando viene chiamato SEND, l'entità di trasporto controlla se è giunto un credito sulla connessione specificata. In questo caso, il messaggio viene inviato (con più pacchetti, se necessario) e il credito decrementato; in caso contrario l'entità di trasporto diventa inattiva fino all'arrivo di un credito. Questo meccanismo garantisce che non vengano inviati messaggi se l'altra parte non ha già chiamato un RECEIVE. Come risultato, all'arrivo di un messaggio è garantita la disponibilità di un buffer in cui inserire il messaggio. Lo schema può facilmente essere generalizzato per consentire ai ricevitori di fornire più buffer e richiedere più messaggi.

È opportuno tenere a mente le semplificazioni della Figura 6.20. Un'entità di trasporto realistica controllerà la validità dei parametri forniti dall'utente, gestirà il ripristino dei crash dello strato network, si occuperà delle collisioni tra le chiamate e supporterà un servizio di trasporto più generico che include caratteristiche come interrupt, datagram e versioni non bloccanti delle primitive SEND e RECEIVE.

6.3.3 L'esempio come macchina a stati finiti

La scrittura di un'entità di trasporto è difficile, specialmente per i protocolli più realistici. Per ridurre le possibilità di commettere errori, spesso è utile rappresentare lo stato del protocollo come macchina a stati finiti.

Abbiamo già visto che il nostro protocollo di esempio presenta sette stati per connessione. È anche possibile isolare 12 eventi che possono spostare una connessione da uno stato all'altro. Cinque di questi eventi sono primitive di servizio, altri sei sono l'arrivo di sei tipi di pacchetto ammessi, l'ultimo è la scadenza del timer. La Figura 6.21 mostra le azioni del protocollo principale sotto forma di matrice. Le colonne sono gli stati e le righe sono i 12 eventi.

Ogni voce della matrice (vale a dire la macchina a stati finiti) della Figura 6.21 contiene fino a tre campi: un predicato, un'azione e un nuovo stato. Il predicato indica a quale condizione viene svolta l'azione. Per esempio, nella voce in alto a sinistra, se viene eseguito LISTEN e non c'è più spazio nella tabella (predicato *P1*), LISTEN fallisce e lo stato non cambia. D'altra parte, se per l'indirizzo di trasporto che si sta ascoltando è già arrivato un pacchetto CALL REQUEST (predicato *P2*), la connessione viene stabilita immediatamente. Un'altra possibilità è che *P2* sia falso, vale a dire che non siano giunte CALL REQUEST: in questo caso la connessione resta nello stato IDLE, in attesa di un pacchetto CALL REQUEST.

Vale la pena evidenziare che la scelta degli stati da utilizzare nella matrice non è interamente dipendente dal protocollo stesso. In questo esempio non esiste uno stato LISTENING, che potrebbe essere ragionevole dopo un LISTEN. LISTENING non esiste perché uno stato è associato a una voce del record di connessione, ma LISTEN non crea alcun record di connessione. Perché no? Perché abbiamo deciso di utilizzare i numeri del cir-

		Idle	Wait (attesa)	Queued (accodato)	Established (stabilito)	Sending (trasmissione)	Receiving (ricezione)	Disconnecting (disconnessione)
LISTEN		<i>P1: ~Idle</i> <i>P2: A1/Estab</i> <i>P3: A2/Idle</i>		-Estab				
CONNECT								
DISCONNECT					<i>P4: A5/Idle</i> <i>P4: A6/Disc</i>			
SEND					<i>P5: A7/Estab</i> <i>P5: A8/Send</i>			
RECEIVE						<i>A9/Receiving</i>		
Call_req		<i>P3: A1/Estab</i> <i>P3: A4/Queud</i>						
Call_acc			-Estab					
Clear_req			-idle		<i>A10/Estab</i>	<i>A10/Estab</i>	-idle	
Clear_conf								-idle
DataPkt							<i>A12/Estab</i>	
Credit					<i>A11/Estab</i>	<i>A7/Estab</i>		
Timeout			-idle					

Predicati	Azioni
<i>P1: tabella delle connessioni piena</i>	<i>A1: invio di Call_acc</i>
<i>P2: Call_req in sospeso</i>	<i>A2: attesa di Call_req</i>
<i>P3: LISTEN in sospeso</i>	<i>A3: invio di Call_req</i>
<i>P4: Clear_req in sospeso</i>	<i>A4: avvio del timer</i>
<i>P5: credito disponibile</i>	<i>A5: invio di Clear_conf</i>
	<i>A6: invio di Clear_req</i>
	<i>A7: invio del messaggio</i>
	<i>A8: attesa del credito</i>
	<i>A9: invio del credito</i>
	<i>A10: impostazione del flag Clr_req_received</i>
	<i>A11: registrazione del credito</i>
	<i>A12: accettazione del messaggio</i>

Figura 6.21. Il protocollo di esempio come macchina a stati finiti. Ogni voce presenta un predicato facoltativo, un'azione facoltativa e il nuovo stato. La tilde (~) indica che non viene svolta alcuna azione principale. Una linea sopra un predicato indica la negazione del predicato. Le caselle vuote corrispondono a eventi non validi o impossibili.

cuito virtuale dello strato network come identificatori di connessione e per LISTEN il numero di circuito virtuale è scelto dallo strato network all'arrivo del pacchetto CALL REQUEST.

Le azioni da *A1* ad *A12* sono quelle principali, come l'invio di pacchetti e l'avvio dei timer. Non sono invece elencate tutte le azioni minori, come l'inizializzazione dei campi di un record di connessione. Se un'azione coinvolge il risveglio di un processo inattivo, conta-

no anche le azioni successive al risveglio. Per esempio, se arriva un pacchetto CALL REQUEST e un processo è inattivo in attesa del pacchetto, la trasmissione del pacchetto CALL ACCEPT successiva al risveglio conta come parte dell'azione per CALL REQUEST. Dopo l'esecuzione di ogni azione, la connessione potrebbe passare a un nuovo stato, come mostrato nella Figura 6.21.

Il vantaggio di rappresentare il protocollo come una matrice ha tre risvolti. Innanzitutto, questa forma facilita al programmatore il controllo sistematico di ogni combinazione di stato ed evento, per vedere se è necessaria un'azione. Nelle implementazioni di produzione, alcune combinazioni sono utilizzate per la gestione degli errori. Nella Figura 6.21 non si fa distinzione tra le situazioni impossibili e quelle illegali. Per esempio, se una connessione si trova nello stato *waiting*, l'evento DISCONNECT è impossibile perché l'utente è bloccato e non può eseguire alcuna primitiva. D'altra parte, in uno stato *sending*, i pacchetti di dati non sono previsti perché non è stato emesso alcun credito. L'arrivo di un pacchetto di dati è un errore del protocollo.

Il secondo vantaggio della rappresentazione del protocollo sotto forma di matrice coinvolge la sua implementazione. È possibile immaginare un array bidimensionale in cui l'elemento $a[i][j]$ è un puntatore o un indice per la procedura che gestisce l'evento i all'interno dello stato j . Una possibile implementazione prevede di scrivere l'entità di trasporto come un breve ciclo, con l'attesa dell'evento all'inizio del ciclo. Al verificarsi di un evento s'individua la connessione di pertinenza e ne viene estratto lo stato. Conoscendo l'evento e lo stato, l'entità di trasporto indica l'array a ed esegue la chiamata alla procedura appropriata. Questo approccio offre una struttura più regolare e sistematica rispetto all'entità di trasporto vista come esempio.

Il terzo vantaggio dell'approccio con la macchina a stati finiti riguarda la descrizione del protocollo. In alcuni documenti standard, i protocolli sono forniti come macchine di stato finito del tipo mostrato nella Figura 6.21. Passare da questo tipo di descrizione a un'entità di trasporto funzionante è più facile se l'entità di trasporto è guidata da una macchina a stati finiti basata su quella contenuta nello standard.

Lo svantaggio principale dell'approccio con la macchina a stati finiti è che può essere più difficile da comprendere rispetto all'esempio di programmazione diretta utilizzato inizialmente. Tuttavia questo problema può essere parzialmente risolto disegnando la macchina a stati finiti graficamente, come mostrato nella Figura 6.22.

6.4 I protocolli di trasporto Internet: UDP

Internet possiede due protocolli principali nello strato trasporto, un protocollo senza connessione e uno orientato alla connessione. Nei paragrafi seguenti li studieremo entrambi. Il protocollo senza connessione è UDP; il protocollo orientato alla connessione è TCP. Dal momento che UDP equivale fondamentalmente a IP con l'aggiunta di una breve intestazione, inizieremo con questo e ne studieremo anche due applicazioni.

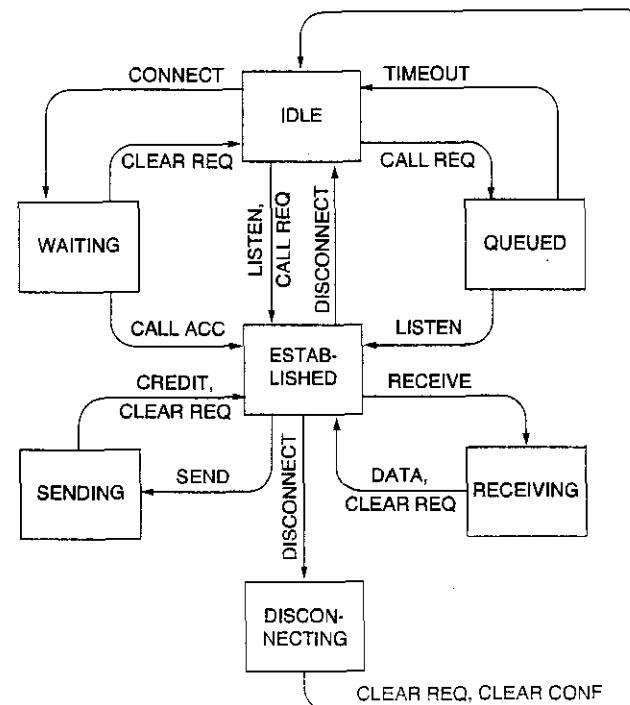


Figura 6.22. Il protocollo di esempio in forma grafica. Le transizioni che lasciano immutato lo stato della connessione sono state omesse per semplicità.

6.4.1 Introduzione a UDP

La suite di protocolli Internet supporta un protocollo di trasporto senza connessione, vale a dire **UDP** (*User Datagram Protocol*, protocollo per datagrammi utente), descritto in RFC 768. UDP offre alle applicazioni un modo per inviare datagrammi IP incapsulati senza dover stabilire una connessione.

UDP trasmette **segmenti** costituiti da un'intestazione di 8 byte seguita dal carico utile. L'intestazione è mostrata nella Figura 6.23. Le due porte servono per identificare i punti finali all'interno dei computer di origine e destinazione. Quando arriva un pacchetto UDP, il suo carico utile è consegnato al processo associato alla porta di destinazione. Questa associazione avviene con l'utilizzo della primitiva BIND o qualcosa di simile, come abbiamo osservato nella Figura 6.6 per TCP (il processo di associazione è lo stesso per UDP). In effetti, il valore principale dell'utilizzo di UDP rispetto a IP è l'aggiunta delle porte di origine e destinazione. Senza i campi per le porte, lo strato trasporto non saprebbe cosa farsene del pacchetto. Grazie a tali campi, invece, consegna i segmenti correttamente.

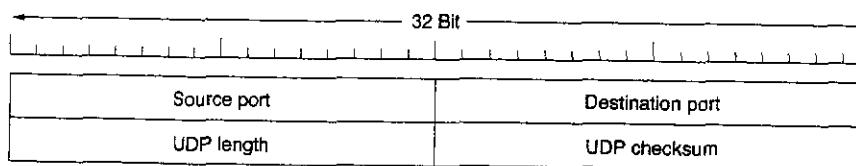


Figura 6.23. L'intestazione di UDP.

La porta di origine serve principalmente quando si deve inviare una risposta all'origine. Copiando il campo *source port* (porta origine) dal segmento in ingresso nel campo *destination port* (porta destinazione) del segmento in uscita, il processo che invia la risposta può specificare il processo sulla macchina sorgente che deve riceverla.

Il campo *UDP length* (lunghezza UDP) include l'intestazione di 8 byte e i dati. *UDP checksum* è facoltativo e contiene 0 se non è elaborato (un vero 0 elaborato è memorizzato con tutti 1). Disattivarlo è insensato, a meno che la qualità dei dati non sia importante (per esempio nel caso di voce digitalizzata).

Vale probabilmente la pena di menzionare esplicitamente alcune delle cose che UDP *non* può fare. Il protocollo non si occupa del controllo di flusso, del controllo degli errori o della ritrasmissione dopo la ricezione di un segmento errato. Questi compiti sono lasciati ai processi utente. UDP si occupa invece di fornire un'interfaccia al protocollo IP, con la caratteristica aggiunta del demultiplexing di più processi utilizzando le porte. Questo è tutto. Per le applicazioni che necessitano di un controllo preciso sul flusso dei pacchetti, sul controllo degli errori o sulla temporizzazione, UDP fornisce soltanto ciò che ha ordinato il dottore.

Un'area in cui UDP è particolarmente utile riguarda le situazioni client/server. Spesso il client invia una breve richiesta al server e si aspetta una breve risposta. Se la richiesta o la risposta vengono perse, il client può andare in timeout e riprovare. Non solo il codice è semplice, ma sono richiesti meno messaggi (uno in ogni direzione) rispetto a un protocollo che necessita dell'impostazione iniziale.

Un'applicazione che utilizza UDP in questo modo è DNS (*Domain Name System*, sistema dei nomi di dominio), che studieremo nel Capitolo 7. In breve, un programma che deve cercare l'indirizzo IP corrispondente a un host, per esempio www.cs.berkeley.edu, può inviare un pacchetto UDP contenente il nome dell'host a un server DNS. Il server risponde con un pacchetto UDP contenente l'indirizzo IP dell'host. Non è necessaria una preparazione della connessione e nemmeno un successivo rilascio. Sono sufficienti due messaggi trasmessi sulla rete.

6.4.2 La chiamata a procedure remote

In un certo senso, l'invio di un messaggio a un host remoto e la ricezione di una risposta corrispondono all'esecuzione di una chiamata a una funzione in un linguaggio di programmazione. In entrambi i casi s'inizia con uno o più parametri, ottenendo alla fine un risultato. Questa osservazione ha portato al tentativo di disporre le interazioni richiesta/risposta sulle reti per modellarle come chiamate a procedure. Tale disposizione facilita la programmazione e la gestione delle applicazioni di rete. Per esempio, s'imma-

gini una procedura denominata *get_IP_address (host_name)* che funziona tramite l'invio di un pacchetto UDP a un server DNS e l'attesa di una risposta, il timeout e un nuovo tentativo se qualcosa non è avvenuto abbastanza rapidamente. In questo modo, tutti i dettagli della rete possono essere nascosti al programmatore.

Un lavoro fondamentale in questa area è stato svolto da Birrell e Nelson (1984). In breve, Birrell e Nelson hanno suggerito di consentire ai programmi di chiamare le procedure situate su host remoti. Quando un processo sulla macchina 1 chiama una procedura sulla macchina 2, il processo chiamante su 1 viene sospeso e l'esecuzione della procedura chiamata avviene su 2. Le informazioni possono essere trasportate dal chiamante al chiamato con i parametri e possono tornare indietro nel risultato della procedura. Il passaggio dei messaggi non è visibile al programmatore. Questa tecnica è nota come **RPC** (*Remote Procedure Call*, chiamata a procedure remote) ed è diventata la base per molte applicazioni di rete. Tradizionalmente, la procedura chiamante è nota come **client** mentre la procedura chiamata è nota come **server**: anche noi utilizzeremo questi nomi.

L'idea alla base di RPC è eseguire una chiamata a procedure remote il più possibile simile a una locale. Nella forma più semplice, per chiamare una procedura remota il programma client deve essere associato a una piccola procedura di libreria, chiamata **stub del client**, che rappresenta la procedura del server nello spazio di indirizzamento del client. Allo stesso modo, il server è associato a una procedura chiamata **stub del server**. Queste procedure nascondono il fatto che la chiamata a procedura dal client al server non sia locale.

I passaggi effettivi per creare una RPC sono mostrati nella Figura 6.24. Il passaggio 1 rappresenta il client che chiama lo stub del client. Questa è una chiamata a procedura locale, con i parametri inseriti nello stack con il metodo normale. Il passaggio 2 rappresenta lo stub del client che inserisce i parametri in un messaggio ed effettua una chiamata di sistema per inviare il messaggio. L'inserimento dei parametri nel pacchetto è definito **marshaling**. Il passaggio 3 mostra il kernel che invia il messaggio dalla macchina client alla macchina server. Il passaggio 4 rappresenta il kernel che passa il pacchetto in ingresso allo stub del server. Per finire, il passaggio 5 mostra lo stub del server che chiama la procedura del server con i parametri presi dal marshaling. La risposta segue lo stesso percorso nell'altra direzione.

L'elemento da osservare è che la procedura del client, scritta dall'utente, esegue una chiamata a procedura normale (quindi locale) allo stub del client, che possiede lo stesso nome della procedura del server. Dal momento che la procedura e lo stub del client si trovano nello stesso spazio di indirizzamento, i parametri vengono passati nel modo solito. Allo stesso modo, la procedura del server viene chiamata da una procedura nel suo spazio di indirizzamento con i parametri previsti. Per la procedura del server niente è insolito. In questo modo, al posto dell'I/O svolto sui socket, la comunicazione di rete viene svolta mimando una normale chiamata a procedura.

Nonostante l'eleganza concettuale di RPC, esistono pericoli in agguato. Uno dei principali riguarda l'utilizzo dei parametri puntatore. Normalmente, il passaggio di un puntatore a una procedura non è un problema. La procedura chiamata può utilizzare il puntatore in modo simile al chiamante, perché entrambe le procedure risiedono nello stesso spazio di indirizzamento virtuale. Con RPC, il passaggio dei puntatori è impossibile perché il client e il server si trovano in spazi di indirizzamento diversi.

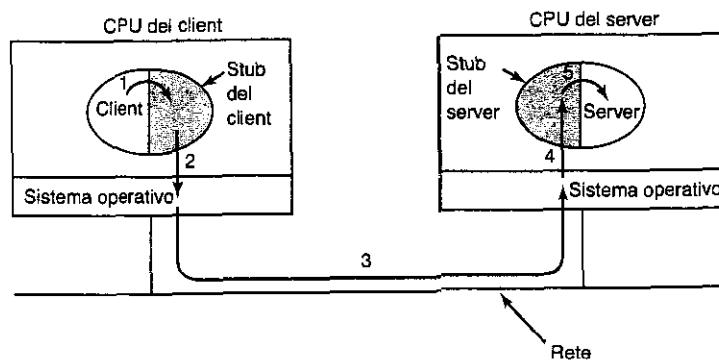


Figura 6.24. I passaggi per effettuare una chiamata a procedure remote.

Gli stub sono ombreggiati.

In alcuni casi, è possibile sfruttare qualche trucco per consentire il passaggio dei puntatori. Supponiamo che il primo parametro sia un puntatore a un intero, k . Lo stub del client può eseguire il marshaling di k e inviarlo al server. Lo stub del server crea poi un puntatore a k e lo passa alla procedura del server, come previsto. Quando la procedura del server restituisce il controllo allo stub del server, quest'ultimo invia k al client, dove il nuovo k viene copiato sopra il vecchio nel caso il server lo abbia modificato. In effetti, la chiamata standard per riferimento è stata sostituita da un'operazione di copia/ripristino. Sfortunatamente, questo trucco non funziona sempre, per esempio se il puntatore fa riferimento a un grafo o a un'altra struttura dati complessa. Per questo motivo, occorre porre alcune limitazioni sui parametri delle procedure chiamate in remoto.

Un secondo problema riguarda il fatto che, nei linguaggi con tipizzazione debole come C, è perfettamente legale scrivere una procedura che elabora il prodotto interno di due vettori (array), senza specificare la grandezza di ognuno. Entrambi potrebbero essere terminati da un valore speciale noto solo alle procedure chiamante e chiamata. In queste circostanze, è praticamente impossibile pretendere che lo stub del client esegua il marshaling dei parametri: non c'è infatti modo di determinarne la loro dimensione.

Un terzo problema sta nel fatto che non è sempre possibile dedurre i tipi dei parametri, nemmeno da una specifica formale o dal codice stesso. Un esempio riguarda *printf*, che può avere qualsiasi numero di parametri (almeno uno): inoltre, i parametri possono essere un mix arbitrario di numeri decimali, interi brevi o lunghi, caratteri, stringhe, numeri in virgola mobile di varie lunghezze o altri tipi. Sarebbe praticamente impossibile chiamare *printf* come procedura remota, vista la permissività di C. Tuttavia, decidere che RPC si può usare solo se non si programma in C (o C++) non sarebbe una scelta molto popolare.

Un quarto problema riguarda l'utilizzo delle variabili globali. Normalmente, le procedure chiamante e chiamata possono comunicare tramite le variabili globali, oltre che mediante i parametri. Se la procedura chiamata viene spostata su una macchina remota, il codice fallisce perché le variabili globali non sono più condivise.

Questi problemi non vogliono suggerire che RPC sia inservibile. In effetti è molto diffuso, ma sono necessarie alcune restrizioni per farlo funzionare bene nelle situazioni reali. Ovviamente RPC non è obbligato a utilizzare i pacchetti UDP, ma RPC e UDP rappresentano una valida unione e UDP viene spesso utilizzato per RPC. Tuttavia, quando i parametri o i risultati possono essere più grandi del pacchetto UDP massimo o quando l'operazione richiesta non è equipotente (vale a dire che non può essere ripetuta in sicurezza, come per l'incremento di un contatore), potrebbe essere necessario impostare una connessione TCP per la richiesta, anziché utilizzare UDP.

6.4.3 Il protocollo di trasporto in tempo reale

RPC client/server è una delle aree dove UDP è molto diffuso, e un'altra riguarda le applicazioni multimediali in tempo reale. In particolare, man mano che le radio Internet, la telefonia Internet, la musica su richiesta, le videoconferenze, il video su richiesta e altre applicazioni multimediali diventavano più comuni, la gente scopriva che ogni applicazione reinventava più o meno lo stesso protocollo di trasporto in tempo reale. Gradualmente si è capito che l'utilizzo di un protocollo di trasporto generico in tempo reale per più applicazioni sarebbe stato una buona idea. Nacque così RTP (*Real-time Transport Protocol*, protocollo di trasporto in tempo reale). È descritto nella RFC 1889 e ormai è ampiamente utilizzato. La posizione di RTP nello stack dei protocolli è in un certo senso strana. È stato deciso di inserire RTP nello spazio dell'utente e di eseguirlo (normalmente) sopra UDP. Il suo funzionamento è descritto di seguito. Un'applicazione multimediale è composta di più flussi audio, video, di testo e magari di altro tipo. Questi flussi vengono passati alla libreria RTP, che si trova nello spazio dell'utente insieme all'applicazione: la libreria esegue il multiplexing dei flussi e li codifica in pacchetti RTP, che vengono poi inseriti in un socket. All'altra estremità del socket (nel kernel del sistema operativo), vengono generati i pacchetti UDP e incorporati in pacchetti IP. Se il computer si trova su una Ethernet, i pacchetti IP vengono poi inseriti in frame Ethernet per la trasmissione. La pila di protocolli per questa situazione è mostrata nella Figura 6.25(a); la nidificazione dei pacchetti nella Figura 6.25(b).

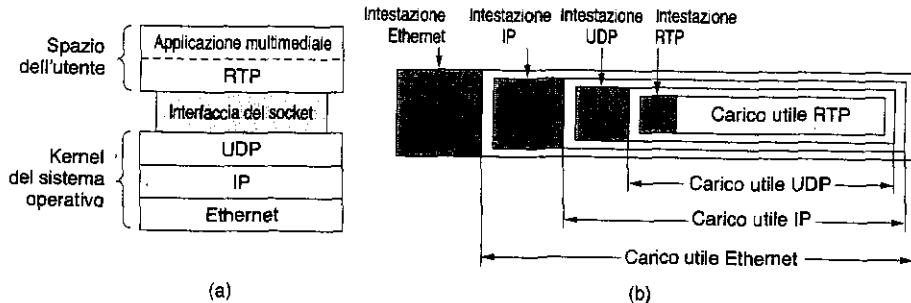


Figura 6.25. (a) La posizione di RTP nella pila dei protocolli. (b) La nidificazione dei pacchetti.

Come conseguenza di questa struttura, è difficile affermare in quale strato si trova RTP. Dal momento che viene eseguito nello spazio dell'utente ed è collegato al programma applicativo, certamente somiglia a un protocollo dello strato applicazione. D'altra parte, si tratta di un protocollo generico indipendente dall'applicazione che fornisce funzionalità di trasporto, pertanto somiglia anche a un protocollo di trasporto. Probabilmente la descrizione migliore è che si tratta di un protocollo di trasporto implementato nello strato applicazione.

La funzione base di RTP è eseguire il multiplexing di flussi di dati in tempo reale in un singolo flusso di pacchetti UDP. Il flusso UDP può essere inviato a una singola destinazione (unicasting) o a più destinazioni (multicasting). Dal momento che RTP utilizza il normale UDP, i suoi pacchetti non sono trattati in modo speciale dai router, a meno che siano attivate alcune delle normali funzionalità QoS (*Quality of Service*, qualità del servizio) IP. In particolare, non vi sono garanzie speciali sulla consegna, sui ritardi temporali e così via.

A ogni pacchetto inviato in un flusso RTP è assegnato un numero incrementato di 1 rispetto al suo predecessore; la numerazione serve alla destinazione per scoprire se mancano alcuni pacchetti. In mancanza di un pacchetto, l'azione migliore che la destinazione può compiere è approssimare il valore mancante per interpolazione. La ritrasmissione non è un'opzione pratica, perché il pacchetto ritrasmesso arriverebbe probabilmente troppo tardi per essere utile. Come conseguenza, RTP non possiede il controllo di flusso, il controllo degli errori, gli acknowledgement o un meccanismo per richiedere la ritrasmissione.

Ogni carico utile di RTP può contenere più campioni, codificati nel modo desiderato dall'applicazione. Per consentire l'interworking, RTP definisce diversi profili (per esempio un singolo flusso audio) e può consentire l'utilizzo di più formati di codifica per ogni profilo. Per esempio, un singolo flusso audio potrebbe essere codificato in campioni PCM di 8 bit a 8 kHz, oppure con codifica delta, predittiva, GSM, MP3 e così via. RTP offre un campo intestazione in cui l'origine può specificare la codifica, ma non è altrimenti coinvolto nell'esecuzione della codifica.

Un'altra caratteristica necessaria a molte applicazioni in tempo reale è il contrassegno temporale. L'idea sta nel consentire all'origine di associare un contrassegno temporale o timestamp al primo campione di ogni pacchetto. I timestamp sono relativi all'inizio del flusso, per cui sono significative solo le differenze tra timestamp, e i valori assoluti non hanno significato. Questo meccanismo consente alla destinazione di svolgere una piccola quantità di buffering e di riprodurre ogni campione nell'esatto istante richiesto, indipendentemente dall'istante di arrivo del pacchetto contenente il campione. Il timestamp riduce gli effetti del jitter (cioè l'incostanza dei ritardi di trasmissione) e consente la sincronizzazione di più flussi con altri flussi. Per esempio, un programma televisivo digitale potrebbe avere un flusso video e due flussi audio. I due flussi potrebbero trasportare l'audio stereofonico del programma in onda a un dato momento, oppure le colonne sonore in lingua originale e doppiata di un film, per dare allo spettatore la possibilità di scelta. Ogni flusso proviene da un dispositivo fisico diverso, ma se possiede un timestamp generato dallo stesso contatore centralizzato può essere riprodotto in modo sincrono anche quando i flussi vengono trasmessi in modo errato.

L'intestazione RTP è illustrata nella Figura 6.26. Consiste di tre parole di 32 bit e di alcune estensioni facoltative. La prima parola contiene il campo *version*, che attualmente cor-

risponde a 2. Speriamo che questa versione sia molto vicina alla versione definitiva, visto che resta un solo numero di versione a disposizione (anche se in futuro si potrebbe decidere che il numero 3 vuol dire che il numero di versione è in una parola di estensione).

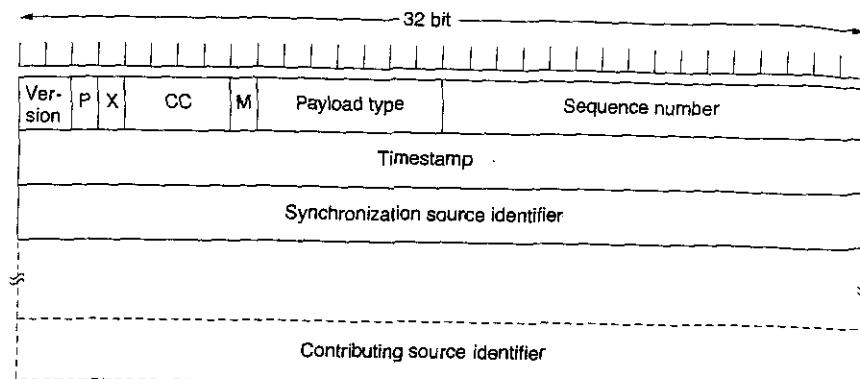


Figura 6.26. L'intestazione RTP.

Il bit *P* indica che il pacchetto è stato riempito fino a ottenere un multiplo di 4 byte. L'ultimo byte del riempimento indica quanti byte sono stati aggiunti. Il bit *X* indica che è presente un'intestazione estesa. Il formato e il significato dell'intestazione estesa non sono definiti; l'unica cosa regola è che la prima parola dell'estensione indica la lunghezza. Si tratta di una via di fuga per qualsiasi requisito imprevisto.

Il campo *CC* indica il numero di origini presenti, da 0 a 15 (vedere sotto). Il bit *M* è un bit di contrassegno specifico dell'applicazione. Può essere utilizzato per contrassegnare l'inizio di un'inquadratura video, l'inizio di una parola in un canale audio o qualcosa' altro che l'applicazione è in grado di comprendere. Il campo *payload type* (tipo di carico utile) indica quale algoritmo di codifica è stato utilizzato (per esempio audio a 8 bit non compresso, MP3 e così via). Dal momento che ogni pacchetto contiene questo campo, la codifica può cambiare durante la trasmissione. *Sequence number* (numero di sequenza) è un contatore incrementato a ogni invio di un pacchetto RTP; serve a rilevare i pacchetti persi.

Il timestamp è prodotto da chi genera il flusso, per indicare quando è stato creato il primo campione nel pacchetto. Questo valore può aiutare a ridurre il jitter nel ricevitore, disaccoppiando l'istante di riproduzione dal quello di arrivo del pacchetto. Il campo *synchronization source identifier* specifica a quale flusso appartiene il pacchetto, e supporta il metodo utilizzato per il multiplexing e il demultiplexing di più flussi dati in un singolo flusso di pacchetti UDP. Per finire, i campi facoltativi *contributing source identifier* sono utilizzati in presenza di mixer nello studio. In tal caso, il mixer è l'origine della sincronizzazione e in questo campo si trova l'elenco dei flussi mixati.

RTP si accompagna a un protocollo di pari livello chiamato **RTCP** (*Real-time Transport Control Protocol*, protocollo di controllo del trasporto in tempo reale). Gestisce il feedback, la sincronizzazione e l'interfaccia utente, ma non trasporta alcun dato. La prima funzione può essere utilizzata per fornire alla sorgente del flusso una retroazione (*feedback*)

su ritardi, jitter, larghezza di banda, congestione e altre proprietà di rete. Queste informazioni possono servire al processo di codifica per aumentare la velocità dei dati (e fornire una qualità superiore) quando la rete funziona bene e per ridurre la velocità dei dati quando vi sono problemi nella rete. Fornendo un feedback continuo, gli algoritmi di codifica si possono adattare costantemente per fornire la migliore qualità permessa dalle circostanze. Per esempio, se la larghezza di banda aumenta o diminuisce durante la trasmissione la codifica può passare da MP3 a PCM 8 bit, fino alla codifica delta (secondo necessità). Il campo *payload type* viene utilizzato per comunicare alla destinazione l'algoritmo di codifica utilizzato per il pacchetto corrente, in modo che sia possibile variarlo su richiesta. RTCP gestisce anche la sincronizzazione tra flussi. Il problema è che flussi diversi possono utilizzare clock differenti, con grana e velocità di deriva diverse. RTCP può essere utilizzato per mantenere la sincronia.

Per finire, RTCP fornisce un metodo per denominare le sorgenti (per esempio in testo ASCII). Queste informazioni possono essere visualizzate sullo schermo del ricevitore per indicare chi sta parlando al momento.

Ulteriori informazioni su RTP sono disponibili in (Perkins, 2002).

6.5 Il protocollo di trasporto Internet: TCP

UDP è un protocollo semplice con alcuni utilizzi di nicchia, come le interazioni client/server e il multimediale, ma per la maggior parte delle applicazioni Internet è necessaria una consegna affidabile in sequenza che UDP non può fornire, pertanto è richiesto un altro protocollo. È chiamato TCP ed è il motore di Internet, che ora studieremo in dettaglio.

6.5.1 Introduzione a TCP

TCP (*Transmission Control Protocol*, protocollo di controllo della trasmissione) è stato progettato appositamente per fornire un flusso di byte affidabile end-to-end su una internetwork inaffidabile. Una internetwork differisce da una singola rete perché le diverse parti possono avere topologie, larghezze di banda, ritardi, dimensioni di pacchetti e altri parametri completamente differenti tra loro. TCP è stato progettato per adattarsi dinamicamente alle proprietà della internetwork e per continuare a offrire solide prestazioni in presenza di molti tipi di errore.

TCP è stato formalmente definito in RFC 793. Con il passare del tempo sono stati scoperti errori e incongruenze, e sono cambiati requisiti in molte aree. I necessari chiarimenti e la soluzione di alcuni problemi si trovano in RFC 1122, mentre le estensioni sono dettagliate in RFC 1323.

Ogni computer che supporta TCP dispone di un'entità di trasporto TCP, che può essere una procedura di libreria, un processo utente o una parte del kernel, ma in tutti i casi gestisce i flussi TCP e le interfacce verso lo strato IP. Un'entità TCP accetta dai processi locali i flussi dati dell'utente; li suddivide in pezzi di dimensione non superiore a 64 KB (nelle applicazioni pratiche sono quasi sempre 1.460 byte di dati, in modo che stiano in un singolo frame Ethernet con le intestazioni IP e TCP); infine invia ogni pezzo in un data-

gramma (*datagram*) IP autonomo. I datagrammi contenenti i dati TCP che arrivano al computer vengono consegnati all'entità TCP che ricostruisce i flussi di byte originali. Per semplicità, a volte si utilizzerà il termine generico TCP per fare riferimento all'entità di trasporto TCP (un componente software) oppure al protocollo TCP (un insieme di regole); dal contesto dovrebbe essere chiaro che cosa si intende. Per esempio, nella frase "l'utente consegna i dati a TCP", si fa chiaramente riferimento all'entità di trasporto TCP.

Lo strato IP non garantisce la consegna senza errori dei datagrammi, pertanto è TCP che deve eseguire il timeout e la ritrasmissione secondo necessità. I datagrammi potrebbero anche arrivare nell'ordine sbagliato; è sempre compito di TCP riassemblare i messaggi nella giusta sequenza. Riassumendo, TCP deve fornire l'affidabilità che la maggior parte degli utenti desidera e che IP non fornisce.

6.5.2 Il modello di servizio TCP

Il servizio TCP è ottenuto con la creazione di punti finali da parte di mittente e ricevente, che vengono chiamati socket come già visto nel paragrafo 6.1.3. Ogni socket possiede un indirizzo (il numero di socket) composto dall'indirizzo IP dell'host e da un numero di 16 bit locale all'host, chiamato porta. Una porta è il nome TCP per un TSAP. Per ottenere il servizio TCP, si deve stabilire esplicitamente una connessione tra un socket sulla macchina di invio e un socket sulla macchina ricevente. Le chiamate ai socket sono elencate nella Figura 6.5.

Un socket supporta più connessioni contemporaneamente. In altre parole, due o più connessioni possono terminare nello stesso socket. Le connessioni sono individuate dagli identificatori di socket a entrambe le estremità, vale a dire dalla coppia (*socket1, socket2*). Non vengono utilizzati i numeri di circuito virtuale o altri identificatori.

I numeri di porta minori di 1.024 identificano le **well-known ports** (porte ben note) e sono riservati ai servizi standard. Per esempio, qualsiasi processo che desidera stabilire una connessione a un host per trasferire un file con FTP può connettersi alla porta 21 dell'host di destinazione per contattare il suo daemon FTP. L'elenco delle porte ben note è fornito all'indirizzo www.iana.org; ne sono state assegnate più di 300. Alcune tra le più conosciute sono elencate nella Figura 6.27.

In fase di avvio dell'host sarebbe certamente possibile associare il daemon FTP alla porta 21, il daemon Telnet alla porta 23 e così via; tuttavia, in questo modo si occupa la memoria con daemon inattivi per la maggior parte del tempo. In genere si preferisce invece avviare un singolo daemon, che in UNIX si chiama **inetd** (*Internet daemon*), associato a più porte per attendere la prima connessione in ingresso. Quando questa si verifica, inetd genera un nuovo processo e vi esegue il daemon appropriato, consentendogli di gestire la richiesta. In questo modo i daemon diversi da inetd sono attivi solo quando hanno lavoro da compiere. Inetd apprende quali porte deve utilizzare da un file di configurazione; di conseguenza, l'amministratore di sistema può configurare il computer perché disponga di daemon permanenti sulle porte più usate (per esempio la porta 80) mentre inetd si occupa delle altre.

Porta	Protocollo	Utilizzo
21	FTP	Trasferimento di file
23	Telnet	Login remoto
25	SMTP	Posta elettronica
69	TFTP	Trivial file transfer protocol
79	Finger	Ricerca di informazioni su un utente
80	HTTP	World Wide Web
110	POP3	Accesso remoto alla posta elettronica
119	NNTP	News di USENET

Figura 6.27. Alcune porte assegnate.

Tutte le connessioni TCP sono di tipo full-duplex punto-punto. Full-duplex indica che il traffico può procedere in entrambe le direzioni contemporaneamente. Punto-punto significa che ogni connessione ha esattamente due punti finali. TCP non supporta il multicasting o il broadcasting.

Una connessione TCP è un flusso di byte, non un flusso di messaggi. I confini dei messaggi non vengono conservati da un'estremità all'altra della connessione. Per esempio, se il processo mittente esegue quattro scritture di 512 byte su un flusso TCP, questi dati possono essere consegnati al processo ricevente come quattro blocchi di 512 byte, due blocchi di 1.024 byte, un solo blocco di 2.048 byte (Figura 6.28) o in altri modi ancora. Il ricevente non può sapere l'unità elementare di scrittura dei dati.

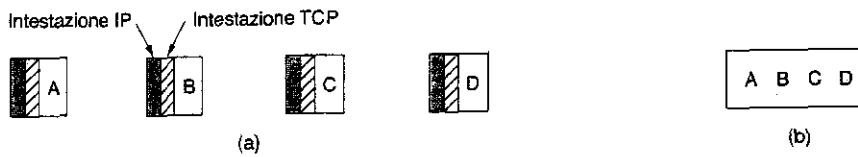


Figura 6.28. (a) Quattro segmenti di 512 byte inviati come datagrammi IP separati.
 (b) I 2.048 byte di dati consegnati all'applicazione in risposta a una singola chiamata READ.

Anche i file in UNIX possiedono questa proprietà. Chi legge un file non può capire se è stato scritto un blocco alla volta, un byte alla volta o tutto insieme. Come con un file UNIX, il software TCP non ha idea del significato dei byte e non è interessato a trovarlo: un byte è solo un byte.

Quando un'applicazione passa i dati a TCP, questo a sua discrezione può inviarli immediatamente o inserirli in un buffer, per raccoglierne una quantità maggiore da inviare tutta assieme. A volte può succedere che l'applicazione desideri realmente un invio immediato dei dati. Per esempio, si supponga che un utente abbia eseguito l'accesso a una macchina remota. Dopo il completamento della riga di comando e la pressione del tasto Invio, è essenziale che la riga sia trasmessa immediatamente alla macchina remota, e non conser-

vata nel buffer fino all'inserimento della riga successiva. Per forzare l'uscita dei dati le applicazioni possono utilizzare il flag PUSH, che comunica a TCP di non ritardare la trasmissione.

Alcune vecchie applicazioni usavano il flag PUSH come una specie d'indicatore dei confini del messaggio. Il trucco molte volte funziona, ma non sempre, perché non tutte le implementazioni di TCP passano il flag PUSH all'applicazione sul lato ricevente. Inoltre, se vengono ricevuti altri flag PUSH prima della trasmissione del primo (per esempio perché la linea di output è occupata), TCP è libero di raccogliere tutti i dati PUSH in un singolo datagramma IP, senza separazione tra le parti.

Un'ultima caratteristica del servizio TCP che vale la pena di menzionare sono i **dati urgenti**. Quando un utente interattivo preme i tasti CANC o CTRL+C per interrompere un'elaborazione remota che ha già avuto inizio, l'applicazione trasmittente inserisce alcune informazioni di controllo nel flusso dei dati e le consegna a TCP con il flag URGENT. Questo evento obbliga TCP a interrompere l'accumulazione dei dati e trasmettere immediatamente tutto ciò che ha a disposizione per quella connessione.

Quando i dati urgenti vengono ricevuti dalla destinazione, l'applicazione ricevente viene fermata (per esempio con una signal di UNIX) in modo che possa interrompere ciò che stava facendo e leggere il flusso dei dati per trovare quelli urgenti. La fine dei dati urgenti è contrassegnata, in modo che l'applicazione sappia dove terminano, mentre l'inizio dei dati urgenti non è contrassegnato ed è compito dell'applicazione calcolarlo. Questo schema fornisce un meccanismo di segnalazione basilare lasciando tutto il resto all'applicazione.

6.5.3 Il protocollo TCP

In questo paragrafo viene fornita una panoramica generale del protocollo TCP, e nel successivo sarà analizzato l'intestazione del protocollo campo per campo.

Una funzionalità vitale di TCP, che guida la struttura del protocollo, consiste nel fatto che ogni byte in una connessione TCP ha un proprio numero di sequenza a 32 bit. Agli esordi di Internet le linee tra i router erano principalmente linee affittate a 56 kbps, pertanto un host che inviava ininterrottamente alla velocità massima impiegava circa una settimana per utilizzare tutti i numeri di sequenza. Alle moderne velocità di rete, i numeri di sequenza possono essere consumati a velocità preoccupanti, come vedremo in seguito. Per gli acknowledgement e per il meccanismo window si usano numeri di sequenza a 32 bit distinti, come si vedrà in seguito.

Le entità TCP di invio e ricezione scambiano i dati sotto forma di segmenti. Un **segmento TCP** consiste di un'intestazione fissa di 20 byte (più una parte facoltativa) seguita da zero o più byte di dati. Il software TCP decide la dimensione dei segmenti, e può accumulare in un segmento i dati provenienti da più scritture oppure dividere i dati di una scrittura in più segmenti. I limiti sulla dimensione del segmento sono due.

Ogni segmento, compresa l'intestazione TCP, deve essere contenuto nel carico utile di 65.515 byte di IP.

Ogni rete ha una **MTU** (*Maximum Transfer Unit*, unità di trasferimento massima) e ogni

segmento deve essere contenuto nella MTU. In pratica, la MTU è generalmente lunga 1.500 byte (la dimensione del carico utile di Ethernet) e ciò definisce il limite superiore per la dimensione del segmento.

Il protocollo di base utilizzato dalle entità TCP è il protocollo sliding window. Quando un mittente trasmette un segmento avvia anche un timer. Quando il segmento arriva a destinazione, l'entità TCP ricevente invia un segmento (con i dati, se esistono, oppure senza) contrassegnato da un numero di acknowledgement uguale al numero di sequenza successivo che prevede di ricevere. Se il timer del mittente scade prima della ricezione dell'acknowledgement, il mittente ritrasmette il segmento.

Anche se questo protocollo sembra semplice, esistono pro e contro (magari poco evidenti) che saranno illustrati in seguito. I segmenti possono arrivare in ordine sbagliato; in questo caso i byte 3.072-4.095 potrebbero arrivare ma non essere confermati perché i byte 2.048-3.071 non si sono ancora visti. Durante il transito i segmenti possono anche essere ritardati così tanto che il mittente va in timeout ed è costretto a ripetere la trasmissione. Le ritrasmissioni possono includere intervalli di byte diversi rispetto alla trasmissione originale, e ciò costringe a un'attenta gestione per tenere traccia di quali byte sono stati ricevuti correttamente a ogni istante, ma l'operazione può essere svolta perché ogni byte nel flusso ha un offset univoco.

TCP deve essere preparato a gestire questi problemi e risolverli in modo efficiente. Una quantità considerevole di sforzi è stata spesa per l'ottimizzazione delle prestazioni dei flussi TCP, anche in presenza di problemi di rete. Nel seguito saranno discussi diversi algoritmi utilizzati da molte implementazioni di TCP.

6.5.4 L'intestazione del segmento TCP

La Figura 6.29 mostra la struttura di un segmento TCP. Ogni segmento inizia con un'intestazione di 20 byte con formato fisso; l'intestazione fissa può essere seguita da alcune opzioni dell'intestazione. Dopo le opzioni, seguono fino a $65.535 - 20 - 20 = 65.495$ byte di dati, dove i primi 20 fanno riferimento all'intestazione IP e i secondi all'intestazione TCP. Sono ammessi segmenti senza dati, ampiamente usati per acknowledgement e messaggi di controllo.

Analizziamo l'intestazione TCP campo per campo. I campi *source port* (porta origine) e *destination port* (porta destinazione) identificano gli estremi locali della connessione. Le porte ben note sono definite all'indirizzo www.iana.org, ma ogni host può allocare altre se desidera. Una porta e l'indirizzo IP del suo host formano un punto finale univoco di 48 bit. La coppia di punti finali origine e destinazione identifica la connessione.

I campi *sequence number* (numero di sequenza) e *acknowledgement number* (numero di acknowledgement) eseguono le loro solite funzioni. Occorre notare che il secondo specifica il successivo byte previsto, non l'ultimo byte ricevuto correttamente. Entrambi sono lunghi 32 bit, perché ogni byte di dati è numerato in un flusso TCP.

Il campo *TCP header length* (lunghezza intestazione TCP) indica quante parole di 32 bit sono contenute nell'intestazione TCP. L'informazione è necessaria perché il campo *options* (opzioni) ha una lunghezza variabile, come l'intestazione. Tecnicamente questo

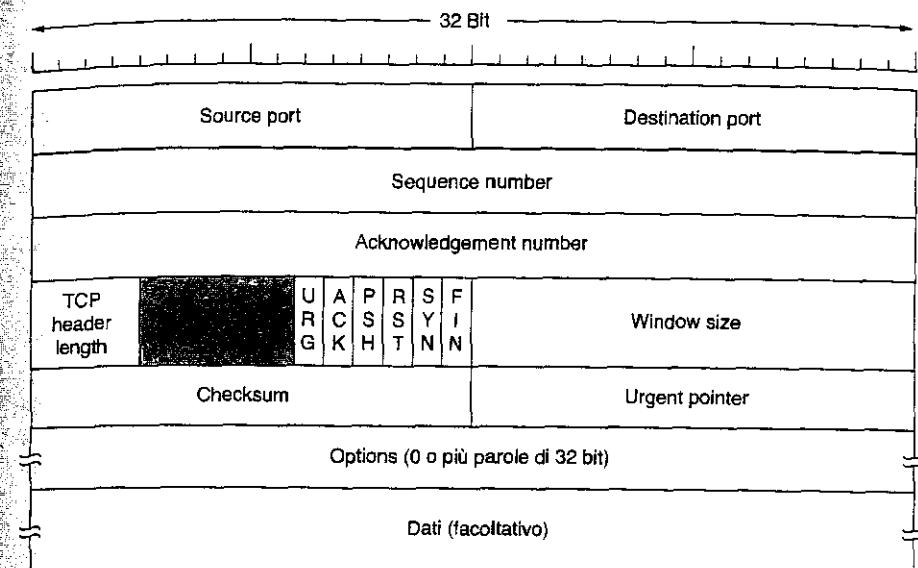


Figura 6.29. L'intestazione TCP.

campo indica l'inizio dei dati effettivi all'interno del segmento, misurato in parole di 32 bit; ma questo numero rappresenta praticamente la lunghezza dell'intestazione espressa in word, che è poi la stessa cosa. Segue un campo di 6 bit inutilizzato. Il fatto che questo campo sia sopravvissuto intatto per un quarto di secolo testimonia quanto TCP sia ben studiato. Un protocollo studiato male lo avrebbe già sfruttato per correggere i bug nella struttura originale. Seguono sei flag di 1 bit. *URG* è impostato quando si usa *urgent pointer* (puntatore urgente), che indica l'offset in byte (partendo dal numero di sequenza corrente) in cui si trovano i dati urgenti [NdR - In realtà indica l'offset in byte dell'ultimo dato urgente; la Posizione iniziale del primo dato urgente non è indicata, e l'applicazione ricevente deve determinarla da sola, senza aiuto da parte del protocollo TCP (cfr pag. 535, ed anche W.R. Stevens, *TCP/IP Illustrated*, vol. 1, pp. 292-293)]. Questa funzionalità è utilizzata al posto dei messaggi di interrupt e, come affermato in precedenza, è un metodo rudimentale per consentire al mittente di inviare segnali al ricevente senza coinvolgere TCP nel motivo dell'interrupt. Il bit *ACK* è impostato a 1 per indicare che *acknowledgement number* (numero di acknowledgement) è valido. Se *ACK* è 0, il segmento non contiene un acknowledgement, pertanto il campo *acknowledgement number* viene ignorato. Il bit *PSH* segnala la presenza di dati PUSH. Al ricevente viene gentilmente chiesto di consegnare i dati all'applicazione all'arrivo, e di non archiviarli nel buffer per poi trasmettere un buffer completo (come potrebbe fare per migliorare l'efficienza). Il bit *RST* viene utilizzato per reimpostare una connessione che è diventata incongruente a causa di un crash dell'host o per altre ragioni. È anche utilizzato per rifiutare un segmento non valido o un tentativo di aprire una connessione. In generale, se si riceve un segmento con il bit *RST* attivato, esiste un problema.

Il bit *SYN* viene utilizzato per stabilire le connessioni. La richiesta di connessione presenta *SYN* = 1 e *ACK* = 0 per indicare che il campo acknowledgement non è utilizzato. La replica della connessione porta un acknowledgement, pertanto possiede *SYN* = 1 e *ACK* = 1. In pratica, il bit *SYN* è utilizzato per segnalare CONNECTION REQUEST e CONNECTION ACCEPTED, mentre il bit *ACK* distingue tra le due possibilità.

Il bit *FIN* viene utilizzato per rilasciare una connessione. Specifica che il mittente non ha altri dati da trasmettere. Tuttavia, dopo avere chiuso una connessione, il processo di chiusura potrebbe continuare a ricevere dati all'infinito. I segmenti *SYN* e *FIN* possiedono numeri di sequenza, garantendo così l'elaborazione nell'ordine corretto.

Il controllo di flusso in TCP è gestito con una sliding window a dimensione variabile. Il campo *window size* (dimensione finestra) indica quanti byte possono essere inviati a partire da quello che ha ricevuto acknowledgement. Un campo *windows size* con valore 0 è ammesso, e afferma che i byte fino ad *acknowledgement number* - 1 compreso sono stati ricevuti, ma che il ricevente ha attualmente bisogno di riposo e non desidera altri dati per il momento. Il ricevente può in seguito dare l'autorizzazione all'invio trasmettendo un segmento con lo stesso *acknowledgement number* e un campo *window size* diverso da zero. Nei protocolli del Capitolo 3, gli acknowledgement dei frame ricevuti e le autorizzazioni per inviare nuovi frame sono uniti assieme; questa è una conseguenza della dimensione fissa della finestra per ogni protocollo. In TCP, gli acknowledgement e le autorizzazioni per inviare dati aggiuntivi sono completamente separati. In effetti, un ricevente può affermare: "Ho ricevuto i byte fino a k compreso ma non ne voglio altri per ora". Questa divisione (in effetti una finestra a dimensione variabile) offre una flessibilità aggiuntiva, che studieremo in dettaglio in seguito.

Per una maggiore flessibilità viene anche fornito un campo *checksum*, che esegue la somma di controllo dell'intestazione, dei dati e della pseudointestazione concettuale mostrata nella Figura 6.30. Quando si esegue questo calcolo, il campo *checksum* di TCP viene impostato a zero e il campo dati viene riempito con un altro byte zero se la sua lunghezza è un numero dispari. L'algoritmo di checksum somma semplicemente i complementi a uno delle parole di 16 bit e quindi calcola il complemento a uno della somma. Come conseguenza, quando il ricevente esegue il calcolo sull'intero segmento (compreso il campo *checksum*) il risultato dovrebbe essere zero.

La pseudointestazione contiene gli indirizzi IP a 32 bit delle macchine di origine e destinazione, il numero di protocollo per TCP (6) e il conteggio dei byte per il segmento TCP (compresa l'intestazione). Includendo la pseudointestazione nel calcolo della somma di controllo TCP è possibile rilevare i pacchetti consegnati in modo errato, ma si viola la gerarchia dei protocolli, perché gli indirizzi IP appartengono allo strato IP, e non allo strato TCP. UDP utilizza la stessa pseudointestazione per il proprio checksum.

Il campo *options* (opzioni) è un modo per aggiungere caratteristiche extra non disponibili nella normale intestazione. L'opzione più importante è quella che permette a ogni host di specificare il carico utile massimo che TCP potrà accettare. L'utilizzo di segmenti grandi è più efficiente dell'utilizzo di segmenti piccoli, perché l'intestazione di 20 byte può essere ammortizzata su più dati, ma gli host più piccoli non sono in grado di gestire i segmenti grandi.

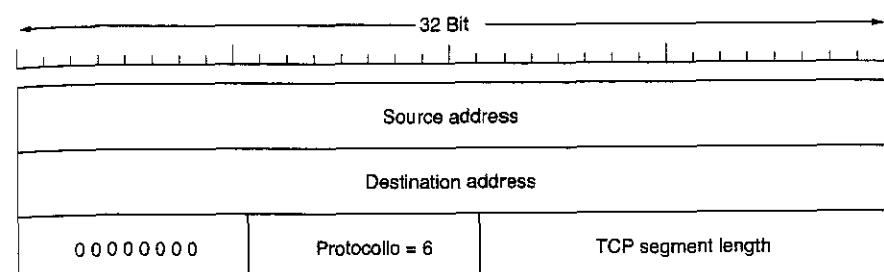


Figura 6.30. La pseudointestazione inclusa nella somma di controllo di TCP.

Durante l'impostazione della connessione, ciascuna parte ha la possibilità di annunciare il massimo carico utile che è capace di gestire, e confrontarlo con quello del partner. Se un host non utilizza questa opzione, per impostazione predefinita il carico utile è pari a 536 byte. Tutti gli host Internet devono accettare segmenti TCP di $536 + 20 = 556$ byte. La dimensione massima del segmento nelle due direzioni può non essere la stessa.

Per le linee con banda elevata e/o ritardi elevati, una finestra di 64 KB è spesso un problema. Su una linea T3 (44,736 Mbps), bastano 12 millisecondi per spedire un'intera finestra di 64 KB. Se il ritardo della propagazione round-trip è 50 millisecondi (tipico di una fibra intercontinentale), il mittente sarà inattivo per 3/4 del tempo in attesa di acknowledgement; su una connessione satellitare la situazione è ancora peggiore. Una dimensione più elevata della finestra consentirebbe al mittente di continuare a pompare dati, ma utilizzando il campo *window size* (dimensione finestra) a 16 bit non c'è modo di esprimere dimensioni maggiori. In RFC 1323 è stata proposta un'opzione *window scale* (scala finestra), che consente al mittente e al ricevente di negoziare un fattore di scala. Questo numero consente a entrambi i lati di far scorrere il campo *window size* di 14 bit verso sinistra, permettendo una finestra di 2^{30} byte. La maggior parte delle implementazioni TCP attuali supporta questa opzione.

Un'altra opzione proposta da RFC 1106 e ora ampiamente implementata è l'utilizzo della ripetizione selettiva al posto di un protocollo go back n, che cioè torna indietro n volte. Se il ricevente ottiene un segmento errato e quindi un numero elevato di segmenti validi, il normale protocollo TCP va in timeout e ritrasmette tutti i segmenti privi di acknowledgement, inclusi quelli ricevuti correttamente (questo è un protocollo di tipo "torna indietro n volte"). RFC 1106 introduce i NAK per consentire al ricevente di chiedere uno o più segmenti specifici. Dopo averli ottenuti, può inviare un acknowledgement per tutti i dati nel buffer, riducendo così la quantità di dati ritrasmessi.

6.5.5 Costituzione della connessione TCP

Le connessioni in TCP vengono stabilite mediante l'handshake a tre vie trattato nel Paragrafo 6.2.2. Per stabilire una connessione, un lato (per esempio il server) attende in modo passivo una connessione in ingresso eseguendo le primitive LISTEN e ACCEPT, indicando un'origine specifica oppure nessuna in particolare.

L'altro lato (diciamo il client) esegue una primitiva CONNECT, specificando l'indirizzo IP e la porta a cui vuole connettersi, la dimensione massima del segmento TCP che potrà accettare e facoltativamente alcuni dati utente (per esempio una password). La primitiva CONNECT invia un segmento TCP con il bit SYN a 1 e il bit ACK a 0, poi attende una risposta. Quando questo segmento arriva a destinazione, l'entità TCP controlla se esiste un processo che ha eseguito un LISTEN sulla porta indicata nel campo *destination port*, e in caso negativo invia una risposta con il bit RST a 1 per rifiutare la connessione.

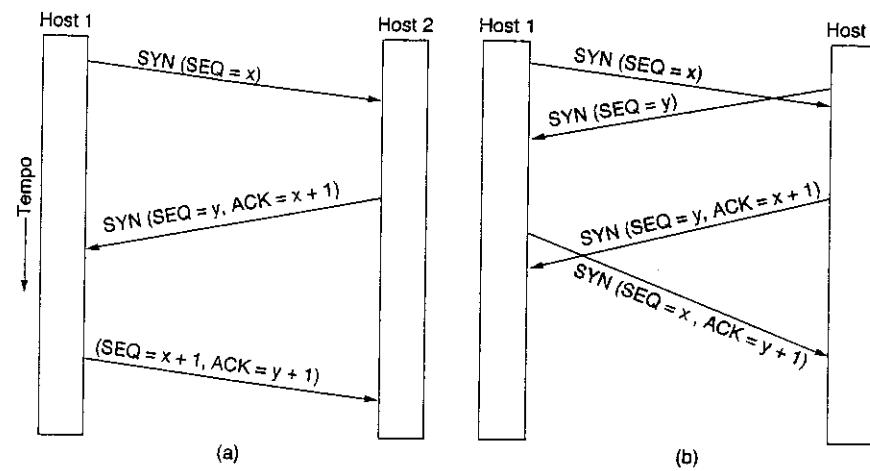


Figura 6.31. (a) La costituzione della connessione TCP nel caso normale.

(b) Una collisione tra le chiamate.

Se un processo è in ascolto sulla porta riceve il segmento TCP in ingresso, quindi può accettare o rifiutare la connessione. Se accetta, viene restituito un segmento di acknowledgement. La sequenza dei segmenti TCP inviata nel caso normale è mostrata nella Figura 6.31(a). Si nota che un segmento SYN consuma un byte dello spazio della sequenza, così viene riconosciuto in modo non ambiguo.

Nel caso che due host tentino contemporaneamente di stabilire una connessione tra gli stessi due socket, la sequenza di eventi è illustrata nella Figura 6.31(b). Il risultato di questi eventi è la costituzione di una sola connessione, e non due, perché le connessioni sono identificate dai loro punti finali. Se la prima attivazione genera una connessione identificata da (x, y) e la seconda svolge la stessa operazione, viene creata una sola voce della tabella per (x, y) .

Il numero iniziale della sequenza di una connessione non è 0 per i motivi discussi in precedenza. Si usa uno schema basato su orologio, con un battito ogni 4 μ sec. Per maggiore sicurezza, quando un host subisce un crash potrebbe non essere riavviato per un tempo pari a quello di vita massimo del pacchetto, in modo da garantire che non vi siano in circolazione su Internet pacchetti riferiti a connessioni precedenti.

6.5.6 Il rilascio della connessione TCP

Anche se le connessioni TCP sono di tipo full-duplex, per comprendere come vengono rilasciate è meglio immaginarle come una coppia di connessioni simplex, dove ogni connessione simplex è rilasciata in modo indipendente dalla sua gemella. Per rilasciare una connessione, entrambe le parti possono inviare un segmento TCP con il bit FIN impostato, per indicare che non hanno più dati da trasmettere. Quando FIN riceve l'acknowledgement la direzione viene chiusa per i nuovi dati; tuttavia, i dati possono continuare a fluire indefinitamente nell'altra direzione. Quando entrambe le direzioni saranno state chiuse la connessione sarà rilasciata. Normalmente, sono necessari quattro segmenti TCP per rilasciare una connessione, un FIN e un ACK per ogni direzione; tuttavia, è possibile inserire il primo ACK e il secondo FIN nello stesso segmento, riducendo il totale a tre.

Proprio come nelle conversazioni telefoniche dove entrambe le persone si salutano e riagganciano contemporaneamente, entrambe le estremità di una connessione TCP possono inviare segmenti FIN contemporaneamente, che ricevono acknowledgement nel modo abituale e terminano la connessione. In effetti, non c'è una vera e propria differenza tra due host che rilasciano in sequenza o contemporaneamente.

Per evitare il problema dei due esercizi si usano i timer. Se una risposta a FIN non arriva entro la durata massima di due pacchetti, il mittente di FIN rilascia la connessione. L'altro lato noterà che nessuno sembra ascoltarlo più e andrà in timeout. Anche se questa soluzione non è perfetta, non resta che accettarla poiché la soluzione perfetta è teoricamente impossibile. Nella pratica raramente nascono problemi.

6.5.7 Il modello di gestione della connessione TCP

I passaggi richiesti per stabilire e rilasciare le connessioni possono essere rappresentati in una macchina a stati finiti con gli 11 stati elencati nella Figura 6.32. In ogni stato sono ammessi alcuni eventi; quando si verifica un evento ammesso è possibile svolgere alcune azioni, mentre se si verificano altri eventi è restituito un errore.

Ogni connessione inizia nello stato *CLOSED*. Lo stato viene lasciato quando la connessione esegue un'apertura passiva (LISTEN) o un'apertura attiva (CONNECT). Se l'altro lato esegue l'operazione opposta, è stabilita una connessione e lo stato diventa *ESTABLISHED*. Il rilascio della connessione può essere iniziato da una qualunque delle parti. Una volta completato, lo stato ritorna a *CLOSED*.

La macchina a stati finiti vera e propria è mostrata nella Figura 6.33. Il caso tipico di un client connesso in modo attivo a un server passivo è mostrato con linee spesse (continue per il client, tratteggiate per il server). Le linee sottili rappresentano sequenze di eventi insolite. Ogni linea nella Figura 6.33 è contrassegnata da una coppia *evento/azione*. L'evento può essere sia una chiamata di sistema iniziata dall'utente (CONNECT, LISTEN, SEND o CLOSE), l'arrivo di un segmento (SYN, FIN, ACK o RST), oppure un timeout pari al doppio della massima vita utile del pacchetto. L'azione è l'invio di un segmento di controllo (SYN, FIN, o RST) oppure nulla (indicato con -). I commenti sono mostrati tra parentesi.

Stato	Descrizione
CLOSED	Nessuna connessione è attiva o in sospeso
LISTEN	Il server è in attesa di una chiamata in ingresso
SYN RCV	È arrivata una richiesta di connessione; attesa di ACK
SYN SENT	L'applicazione ha iniziato ad aprire una connessione
ESTABLISHED	Il normale stato di trasferimento dei dati
FIN WAIT 1	L'applicazione afferma di avere terminato
FIN WAIT 2	L'altro lato accetta il rilascio
TIMED WAIT	Attende la scadenza di tutti i pacchetti
CLOSING	Entrambi i lati hanno cercato di chiudere contemporaneamente
CLOSE WAIT	L'altro lato ha iniziato il rilascio
LAST ACK	Attende la scadenza di tutti i pacchetti

Figura 6.32. Gli stati utilizzati nella macchina a stati finiti per la gestione della connessione TCP.

Potrebbe essere più facile comprendere il diagramma seguendo prima il percorso di un client (la linea continua spessa) e poi il percorso di un server (la linea tratteggiata spessa). Quando un programma applicativo sulla macchina client emette una richiesta CONNECT, l'entità TCP locale crea un record per la connessione, la contrassegna nello stato SYN SENT e invia un segmento SYN. Si noti che molte connessioni potrebbero essere aperte contemporaneamente da più applicazioni, pertanto lo stato è relativo alla connessione e registrato nel record della connessione. Quando arriva SYN+ACK, TCP invia l'ACK finale dell'handshake a tre vie e passa nello stato ESTABLISHED. Ora è possibile inviare e ricevere dati.

Quando un'applicazione ha finito, esegue una primitiva CLOSE che provoca l'invio da parte dell'entità TCP locale di un segmento FIN e l'attesa dell'ACK corrispondente (il quadro tratteggiato contrassegna la chiusura attiva). Quando arriva l'ACK, viene eseguita una transizione allo stato FIN WAIT 2 e una direzione della connessione si chiude. Quando anche l'altro lato esegue la chiusura, viene ricevuto un FIN che richiede l'invio di un acknowledge. Ora entrambi i lati sono chiusi, ma TCP attende un tempo uguale alla durata massima del pacchetto per garantire che tutti i pacchetti della connessione siano caduti (nel caso in cui l'acknowledgement fosse stato perso). Alla scadenza del timer, TCP elimina il record della connessione.

Ora esaminiamo la gestione della connessione dal punto di vista del server. Il server esegue un LISTEN e attende che qualcuno effettui l'attivazione. Quando arriva SYN, provoca come risposta un acknowledge e il server passa allo stato SYN RCV. Quando il SYN del server riceve un acknowledge l'handshake a tre vie è completo e il server passa allo stato ESTABLISHED. Ora può avvenire il trasferimento dei dati.

Quando il client ha terminato esegue un CLOSE, che provoca l'arrivo di FIN al server (il quadro tratteggiato che indica la chiusura passiva). Il server riceve un signal. Quando a sua volta esegue CLOSE, FIN viene mandato al client. All'arrivo dell'acknowledgement dal client, il server rilascia la connessione ed elimina il record della connessione.

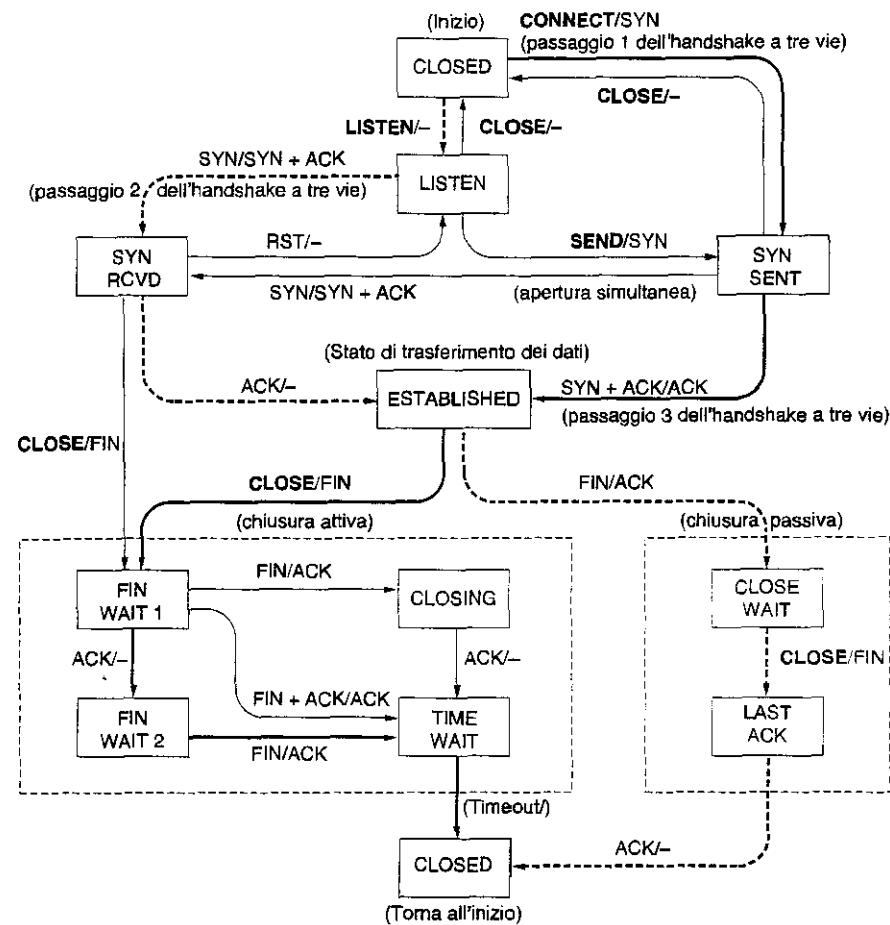


Figura 6.33. La macchina a stati finiti per la gestione della connessione TCP.

La linea continua spessa è il percorso normale per un client; la linea tratteggiata spessa è il percorso normale per un server. Le linee sottili rappresentano eventi insoliti. Ogni transizione è etichettata con l'evento che la provoca e l'azione risultante, separati da una barra.

6.5.8 Il criterio di trasmissione di TCP

Come affermato in precedenza, la gestione della finestra in TCP non è associata direttamente agli acknowledgement [NdR - la posizione della finestra di trasmissione è certamente legata agli acknowledgement, per la natura stessa di ogni protocollo sliding window]. La differenza con gli altri protocolli sliding window consiste nel fatto che la dimensione massima della finestra di trasmissione è variabile e determinata dal ricevente, mentre nella maggior parte degli altri protocolli basati su sliding window la dimensione massima è costante. La confusione della spiegazione sta nel fatto che l'autore descrive i protocolli sliding window del livel-

lo Data Link come costituiti da una finestra di trasmissione di dimensione variabile. Purtroppo però nella maggior parte delle descrizioni del protocollo TCP si definisce una finestra di trasmissione di dimensione variabile ma con dimensione che non dipende dalle conferme: la

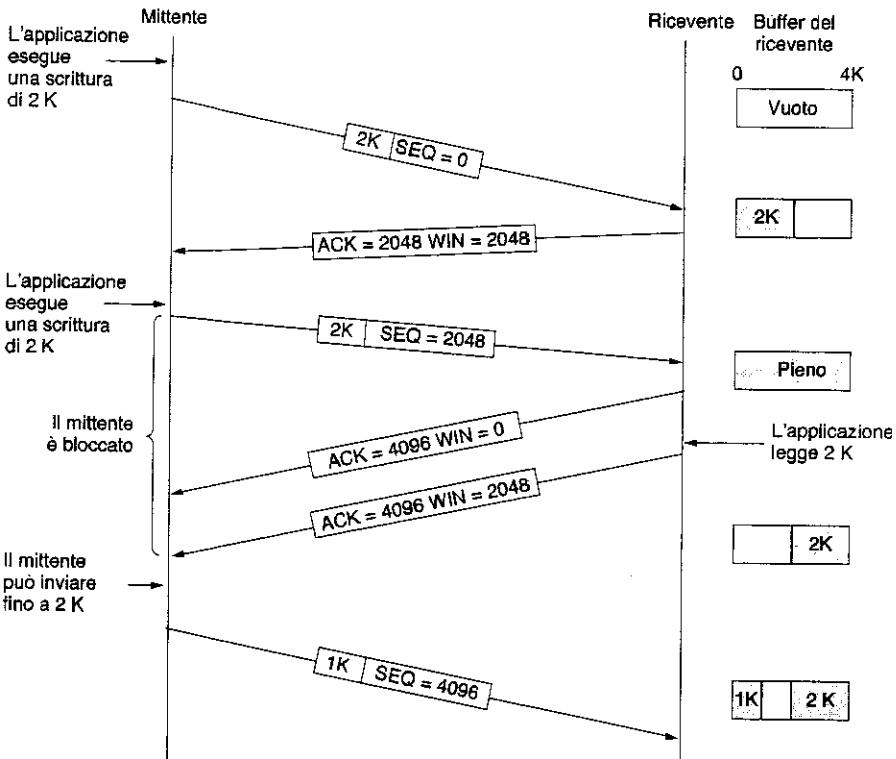


Figura 6.34. La gestione della finestra (window) in TCP.

finestra è internamente suddivisa in due parti, l'una contenente i numeri spediti e non confermati, l'altra contenente i numeri spedibili ma non ancora spediti. L'autore sembra mischiare le due definizioni di finestra di trasmissione (ovviamente le definizioni sono equivalenti, è solo un problema di terminologia), come nella maggior parte dei protocolli di collegamento dati. Per esempio, si supponga che il ricevente abbia un buffer di 4.096 byte, come mostrato nella Figura 6.34. Se il mittente trasmette un segmento di 2.048 byte che viene correttamente ricevuto, il ricevente darà un acknowledgement al segmento. Tuttavia, dal momento che ora possiede solo 2.048 byte di spazio nel buffer (fino a quando l'applicazione rimuove alcuni dati dal buffer), "pubblicizza" una finestra di 2.048 byte partendo dal successivo byte previsto. Dopo il mittente trasmette altri 2.048 byte, che ricevono acknowledgement; la finestra pubblicizzata è quindi 0. Il mittente deve fermarsi fino a quando il processo applicativo all'host ricevente ha rimosso alcuni dati dal buffer; in quel momento TCP potrà mostrare una

finestra più grande. Quando la finestra è 0, il mittente non può di norma inviare segmenti, con due eccezioni. Prima: è possibile inviare dati urgenti, per esempio per consentire all'utente di interrompere il processo in esecuzione sulla macchina remota. Seconda: il mittente può inviare un segmento di 1 byte, per fare in modo che il ricevente annunci di nuovo il successivo byte previsto e la dimensione della finestra. Lo standard TCP fornisce esplicitamente questa possibilità per scongiurare un blocco in caso di perdita di un annuncio della finestra. I mittenti non sono obbligati a trasmettere i dati appena li ricevono dall'applicazione, e i riceventi non hanno l'obbligo di inviare gli acknowledgement il prima possibile. Nell'esempio della Figura 6.34, TCP sapeva di avere a disposizione una finestra di 4 KB all'arrivo dei primi 2 KB di dati, quindi si sarebbe comportato in modo legittimo se avesse inserito dati nel buffer fino all'arrivo di altri 2 KB, in modo da trasmettere un segmento con carico utile di 4 KB. Questa libertà può essere sfruttata per migliorare le prestazioni. Si consideri una connessione telnet a un editor interattivo, che reagisce a ogni pressione di tasto. Nel caso peggiore, quando un carattere arriva all'entità TCP di invio, TCP crea un segmento TCP di 21 byte, che passa a IP affinché lo invii come datagramma IP di 41 byte. Sul lato ricevente, TCP invia immediatamente un acknowledgement di 40 byte (20 byte di intestazione TCP e 20 byte di intestazione IP). In seguito, quando l'editor ha letto il byte, TCP invia un aggiornamento della dimensione della finestra, aumentandola di 1 byte. Anche questo pacchetto è di 40 byte. Per finire, quando l'editor ha elaborato il carattere, restituisce il carattere come pacchetto di 41 byte. In tutto, vengono utilizzati 162 byte di banda e quattro segmenti per ogni carattere digitato. Se la banda è ridotta, questo metodo potrebbe non essere adatto. Un approccio per ottimizzare questa situazione, utilizzato da molte implementazioni di TCP, prevede di ritardare gli acknowledgement e gli aggiornamenti della finestra di 500 msec, nella speranza di acquisire alcuni dati da far viaggiare gratis. Supponendo che l'editor esegua l'echo entro i 500 msec, all'utente remoto bisogna spedire un solo pacchetto di 41 byte, dimezzando il numero di pacchetti e l'utilizzo della banda. Questa regola riduce il carico che il ricevente applica alla rete, ma il mittente opera ancora in modo inefficiente, inviando pacchetti di 41 byte contenenti 1 byte di dati. Un modo per ridurre questo spreco è chiamato **algoritmo di Nagle** (Nagle, 1984). Ciò che Nagle suggerisce è semplice: quando i dati arrivano al mittente un byte alla volta, è sufficiente inviare il primo byte e inserire il resto nel buffer fino a che arriva l'acknowledgement del byte circolante. A questo punto si possono inviare tutti i caratteri nel buffer usando un solo segmento TCP, e iniziare a inserire di nuovo nel buffer i successivi fino a quando arriva il prossimo acknowledge. Se l'utente digita rapidamente e la rete è lenta, in ogni segmento è possibile inserire un buon numero di caratteri, riducendo notevolmente la banda utilizzata. L'algoritmo consente inoltre l'invio di nuovo pacchetto se sono arrivati abbastanza dati da riempire metà della finestra o un segmento con dimensione massima. L'algoritmo di Nagle è ampiamente utilizzato dalle implementazioni TCP, ma a volte è preferibile disattivarlo. In particolare, quando un'applicazione X Window è in esecuzione su Internet, i movimenti del mouse devono essere inviati al computer remoto (il sistema X Window è il sistema a finestre utilizzato nella maggior parte dei sistemi UNIX). Raccogliendoli per inviarli a gruppi si potrebbe osservare un moto irregolare del puntatore del mouse, che infastidisce l'utente. Un altro problema che può degradare le prestazioni di TCP è la **silly window syndrome** (Clark, 1982). Questo problema si verifica quando i dati vengono passati all'entità TCP di

invio in blocchi grandi, ma l'applicazione interattiva sul lato ricevente legge i dati 1 byte alla volta. Per comprendere il problema, si osservi la Figura 6.35. Inizialmente, il buffer TCP sul lato ricevente è pieno e il mittente ne è a conoscenza (ha una finestra con dimensione 0). L'applicazione interattiva legge quindi un carattere dal flusso TCP. Questa azione rende felice il TCP di ricezione, che invia un aggiornamento della finestra al mittente comunicandogli che può inviare 1 byte. Il mittente obbedisce e invia 1 byte. Ora il buffer è pieno, pertanto il ricevente riconosce il segmento di 1 byte ma imposta la finestra a 0. Questo comportamento può andare avanti per sempre.

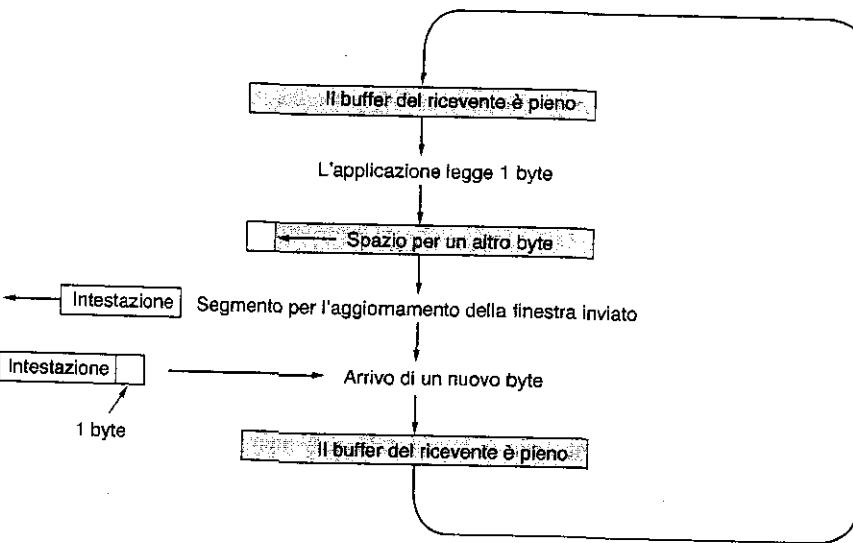


Figura 6.35. Silly window syndrome.

La soluzione di Clark sta nell'impedire che il ricevente invii un aggiornamento della finestra per 1 byte; invece è obbligato ad attendere la disponibilità di una certa quantità di spazio. Nello specifico, il ricevente può inviare un aggiornamento della finestra solo quando è in grado di estinguere la dimensione massima del pacchetto (concordata quando la connessione è stata instaurata), o quando il suo buffer è vuoto per metà. Inoltre, il mittente può essere di aiuto se non invia segmenti di piccole dimensioni. Dovrebbe provare ad attendere fino a quando ha accumulato spazio sufficiente nella finestra per inviare un segmento completo, o almeno un segmento contenente dati per la metà della dimensione del buffer di ricezione (che deve stimare allo schema degli aggiornamenti della finestra ricevuti in passato). L'algoritmo di Nagle e la soluzione di Clark al problema della silly window syndrome sono complementari. Nagle prova a risolvere il problema provocato dal fatto che l'applicazione di invio consegna i dati a TCP un byte alla volta. Clark prova a risolvere il problema di un'applicazione ricevente che legge i dati a TCP un byte alla volta. Entrambe le soluzioni sono valide e possono cooperare. Lo scopo è che il mittente non invii segmenti piccoli e che il ricevente non li richieda.

Il TCP di ricezione può andare oltre nel miglioramento delle prestazioni. Proprio come il TCP di invio, può eseguire il buffering dei dati per tenere bloccata una richiesta READ dell'applicazione fino a quando può fornirle un blocco di dati abbastanza grande. In questo modo si riduce il numero di chiamate a TCP, e quindi il sovraccarico. Naturalmente aumenta anche il tempo di risposta, ma per applicazioni non interattive come il trasferimento di file l'efficienza potrebbe essere più importante del tempo di risposta alle singole richieste.

Un altro problema del ricevente è cosa fare con i segmenti fuori ordine: possono essere conservati o scartati, a discrezione del ricevente, ma gli acknowledgement possono essere inviati solo quando sono stati ricevuti tutti i dati fino all'ultimo byte. Se il ricevente ottiene i segmenti 0, 1, 2, 4, 5, 6 e 7, può mandare acknowledgement relativi al più all'ultimo byte del segmento 2. Al timeout del mittente, la trasmissione riprende dal segmento 3. Se il ricevente ha inserito nel buffer i segmenti da 4 a 7, dopo la ricezione del segmento 3 può mandare acknowledgement per tutti i byte fino alla fine del segmento 7.

6.5.9 Il controllo della congestione di TCP

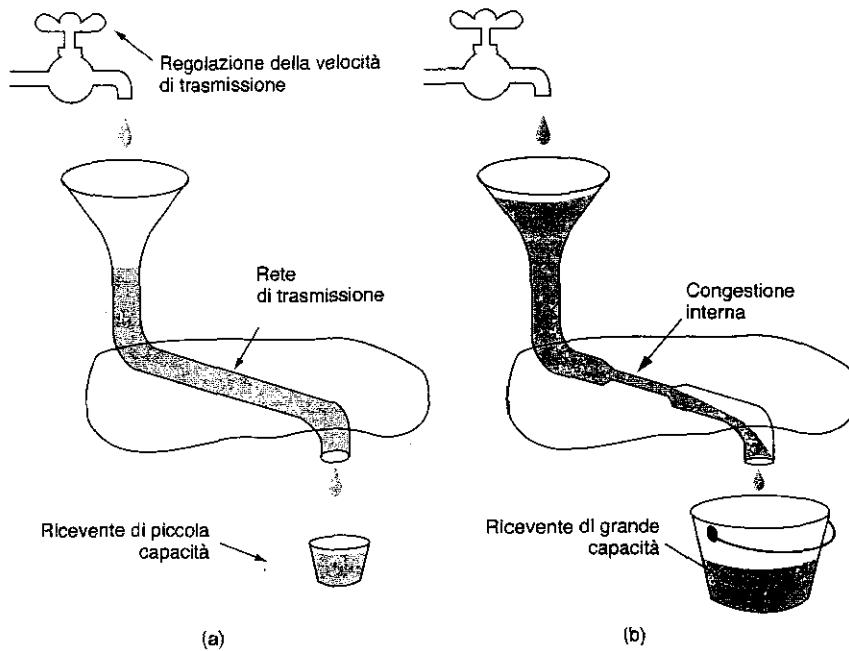
Quando il carico applicato a una rete è superiore a quello che può gestire, si verifica una congestione. Internet non è un'eccezione. In questo paragrafo parleremo degli algoritmi che sono stati sviluppati nell'ultimo quarto di secolo per gestire le congestioni. Anche se lo strato network tenta di gestire le congestioni, la maggior parte del lavoro è svolta da TCP, perché la vera soluzione alle congestioni è la diminuzione della velocità dei dati. In teoria, la congestione può essere affrontata prendendo in prestito un principio dalla fisica: la legge della conservazione dei pacchetti. L'idea è d'impedire l'inserimento di un nuovo pacchetto nella rete fino a quando un pacchetto vecchio non la lascia (vale a dire che viene consegnato). TCP tenta di raggiungere l'obiettivo manipolando dinamicamente la dimensione della finestra.

Il primo passo per la gestione delle congestioni è il rilevamento. Un tempo era difficile rilevare le congestioni. Un timeout provocato da un pacchetto perso avrebbe potuto essere causato da (1) un disturbo su una linea di trasmissione o (2) un pacchetto scartato da un router congestionato. Scoprire la differenza era difficile.

Oggi la perdita di pacchetti dovuta a errori di trasmissione è relativamente rara, perché le linee più lunghe sono realizzate in fibra ottica (le reti wireless sono un'altra storia). Di conseguenza, la maggior parte dei timeout di trasmissione su Internet è dovuta alla congestione. Tutti gli algoritmi TCP Internet presumono che i timeout siano provocati dalla congestione, e controllano i timeout per riscontrare i problemi (con lo stesso metodo utilizzato dai minatori nei confronti dei canarini).

Prima di spiegare come TCP reagisce alla congestione, dobbiamo prima di tutto descrivere che cosa fa per cercare d'impedire il verificarsi delle congestioni. Quando viene stabilita una connessione si deve scegliere una dimensione adatta per la finestra. Il ricevente può specificare una finestra in base alla dimensione del suo buffer; se il mittente si attiene a questa dimensione, non si verificheranno problemi dovuti al traboccare del buffer all'estremità ricevente, ma potrebbero ancora verificarsi a causa della congestione interna alla rete.

Nella Figura 6.36 il problema è illustrato con un'analogia idraulica. Nella Figura 6.36(a) una condutture di grande diametro porta a un ricevente di piccola capacità. Finché il mittente non invia più acqua di quella che può essere contenuta nel catino, non ci saranno perdite. Nella Figura 6.36(b), il fattore limitante non è la capacità del catino ma la capacità di trasporto interna della rete. Se troppa acqua giunge in modo troppo rapido, parte di essa verrà persa (in questo caso per traboccamento del tubo).



La soluzione Internet consiste nel comprendere l'esistenza di due distinti problemi potenziali (capacità della rete e capacità del ricevente) e nel gestirli separatamente. A questo scopo ogni mittente mantiene due finestre: quella che il ricevente ha garantito e una seconda, la **finestra di congestione**. Ognuna riflette il numero di byte che il mittente può trasmettere. Il numero di byte che possono essere inviati corrisponde alla più piccola delle due finestre. Di conseguenza, la finestra effettiva corrisponde al valore minimo tra ciò che il mittente pensa sia giusto e ciò che il ricevente pensa sia corretto. Se il ricevente afferma "Invia 8 KB" ma il mittente sa che un pacchetto con dimensione superiore a 4 KB congestionerebbe la rete, viene inviato un pacchetto di 4 KB. D'altra parte, se il ricevente afferma "Invia 8 KB" e il mittente sa che l'invio di un pacchetto di 32 KB non provocherebbe problemi, vengono inviati comunque gli 8 KB richiesti.

Quando viene stabilita una connessione, il mittente inizializza la finestra di congestione alla dimensione del segmento massimo usato sulla connessione, poi invia un segmento massimo.

Se questo segmento riceve acknowledgement prima della scadenza del timer, aggiunge al valore della finestra di congestione un numero pari alla dimensione del segmento, in modo che la sua dimensione sia il doppio della dimensione massima del segmento, poi invia due segmenti. Quando ognuno di questi segmenti riceve acknowledgement, la dimensione della finestra di congestione viene aumentata di un valore pari alla dimensione massima del segmento. Quando la finestra di congestione raggiunge la dimensione di n segmenti, per i quali l'acknowledgement è sempre arrivato in tempo, la dimensione della finestra di congestione viene aumentata del numero di byte corrispondenti a n segmenti. In pratica ogni gruppo che riceve acknowledgement raddoppia la dimensione della finestra di congestione.

La finestra di congestione continua a crescere in modo esponenziale fino a un timeout o al raggiungimento della finestra del ricevente. L'idea è semplice: se un aumento della dimensione da 1.024 a 2.048 e poi 4.096 è corretto, però un aumento da 4.096 a 8.192 byte provoca un timeout, la finestra di congestione dovrebbe essere impostata a 4.096 per evitare la congestione. Finché la finestra di congestione rimane a 4.096, non verranno spediti gruppi più lunghi di tale valore, indipendentemente dallo spazio disponibile per la finestra garantita dal ricevente. Questo algoritmo è chiamato **avvio lento**, ma non è per nulla lento (Jacobson, 1988): è esponenziale. Tutte le implementazioni TCP devono supportarlo.

Osserviamo ora l'algoritmo di controllo della congestione Internet. Utilizza un terzo parametro, una **soglia** inizialmente pari a 64 KB, oltre alle finestre di congestione e del ricevente. Al verificarsi di un timeout, la soglia viene impostata alla metà della finestra di congestione corrente, e la finestra di congestione viene reimpostata alla dimensione massima di un segmento. L'avvio lento viene quindi utilizzato per determinare che cosa può gestire la rete, ma la crescita esponenziale termina al raggiungimento della soglia. Da quel punto in poi, le trasmissioni avvenute con successo aumentano la finestra di congestione in modo lineare (cioè di un segmento di lunghezza massima per ogni blocco, anziché di un blocco intero). In definitiva l'algoritmo suppone che sia probabilmente accettabile tagliare a metà la finestra di congestione, e poi lavora gradualmente partendo da questo valore.

Si osservi la Figura 6.37 per un'illustrazione del funzionamento dell'algoritmo di congestione. Qui la dimensione massima del segmento è 1.024 byte. Inizialmente la finestra di congestione era di 64 KB; si è però verificato un timeout, pertanto la soglia è stata impostata a 32 KB e la finestra di congestione a 1 KB per la trasmissione 0. La finestra di congestione cresce pertanto in modo esponenziale fino a raggiungere la soglia (32 KB). A partire da qui, inizia a crescere in modo lineare. La trasmissione 13 è sfortunata (si poteva immaginare) perché si verifica un timeout. La soglia è impostata a metà della finestra corrente (40 KB, pertanto la metà è 20 KB) e riparte l'avvio lento. Gli acknowledgement della trasmissione 14 cominciano ad arrivare, e ciascuno dei primi quattro raddoppia la finestra di congestione; dopo, la crescita comincia a divenire di nuovo lineare.

Se non avvengono altri timeout, la finestra di congestione continuerà a crescere fino a raggiungere la dimensione della finestra del ricevente. A quel punto smetterà di crescere, e resterà costante finché non vi saranno più timeout e la finestra del ricevente non cambierà dimensione. Tra parentesi, se arriva un pacchetto SOURCE QUENCH ICMP passato a TCP, l'evento viene trattato in modo simile a un timeout. Un approccio alternativo (e più recente) è descritto nella RFC 3168.

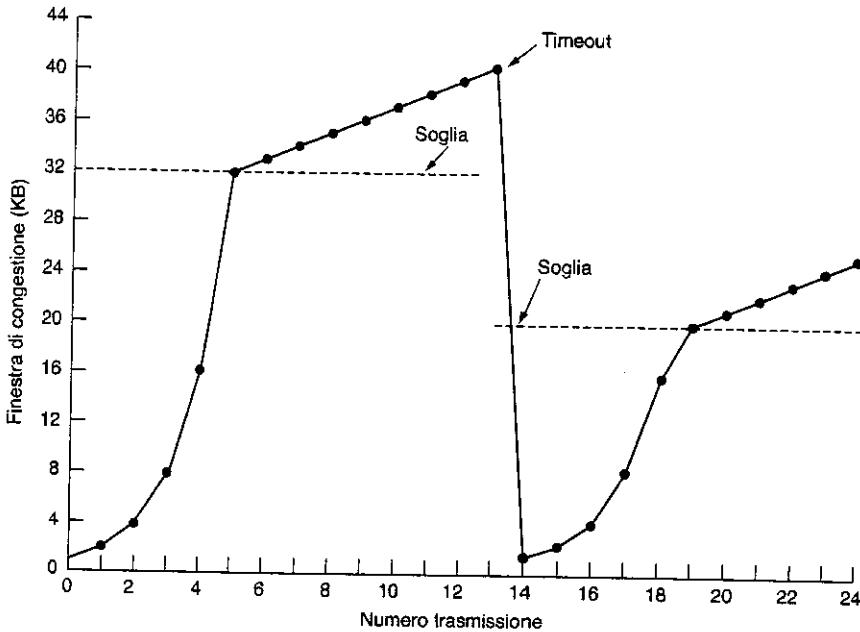


Figura 6.37. Un esempio dell'algoritmo di congestione Internet.

5.10 La gestione dei timer TCP

CTCP utilizza più timer (almeno concettualmente) per svolgere il proprio lavoro. Il più importante è il **timer di ritrasmissione**. Quando viene inviato un segmento, si avvia un timer di ritrasmissione. Se il segmento riceve acknowledgement prima della scadenza il timer viene fermato; se invece il timer scade prima dell'arrivo dell'acknowledgement, il segmento viene ritrasmesso (e il timer riavviato). La questione che si pone è: quando dovrebbe essere lungo l'intervallo di timeout?

Questo problema è molto più difficile nello strato trasporto Internet rispetto ai protocolli gerarchici dello strato data link visti nel Capitolo 3, perché in quel caso la stima del ritardo è altamente prevedibile (vale a dire che ha una varianza bassa), pertanto il timer può essere impostato per scadere appena dopo il tempo stimato di ricezione dell'acknowledgement, come mostrato nella Figura 6.38(a). Dal momento che nello strato data link gli acknowledgement vengono raramente ritardati (a causa della mancanza di congestione), l'assenza di un acknowledgement nel momento previsto indica generalmente che il frame o acknowledgement sono stati persi.

TCP deve affrontare un ambiente radicalmente diverso. La funzione densità di probabilità attiva al tempo richiesto da un acknowledgement TCP per tornare indietro è più simile quella della Figura 6.38(b) che a quella della Figura 6.38(a). Determinare il tempo di round-trip è complicato, e anche quando è noto non è semplice decidere l'intervallo di

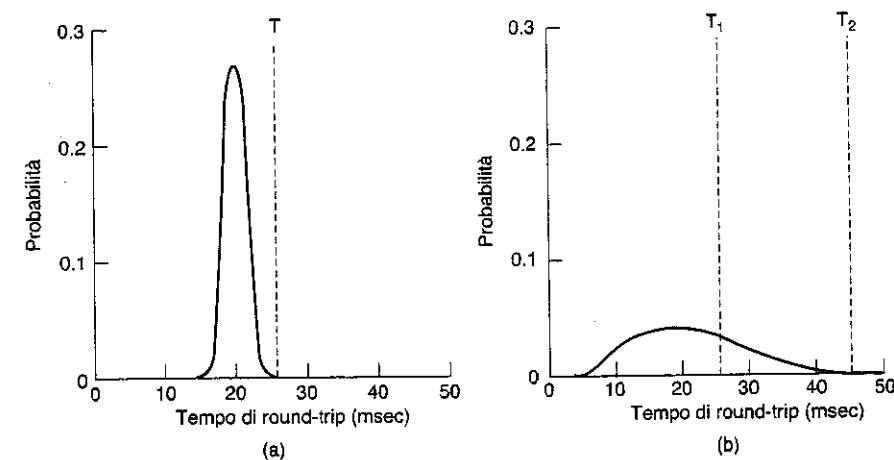


Figura 6.38. (a) Densità di probabilità dei tempi di arrivo degli acknowledgement nello strato data link. (b) Densità di probabilità dei tempi di arrivo degli acknowledgement per TCP.

timeout. Se il timeout impostato è troppo breve (T_1 nella Figura 6.38(b)), si verificheranno ritrasmissioni non necessarie, che congestionano Internet con pacchetti inutili. Se è troppo lungo (per esempio T_2), le prestazioni ne risentiranno a causa del lungo ritardo nella trasmissione ogni volta che un pacchetto viene perso. Inoltre, la media e la varianza della distribuzione di arrivo degli acknowledgement possono cambiare rapidamente in pochi secondi, a causa di una congestione o della sua risoluzione.

La soluzione è utilizzare un algoritmo altamente dinamico che regola costantemente l'intervallo di timeout, in base a continue misurazioni delle prestazioni di rete. L'algoritmo utilizzato generalmente da TCP è dovuto a Jacobson (1988) e funziona come segue. Per ogni connessione, TCP mantiene una variabile, RTT , che rappresenta la migliore stima attuale del tempo di round-trip per la destinazione in questione. Quando viene inviato un segmento si avvia un timer, che serve a due scopi: per sapere quanto tempo richiede l'acknowledgement e per innescare un'eventuale ritrasmissione. Se l'acknowledgement torna indietro prima della scadenza del timer, TCP misura il tempo richiesto, che chiameremo M . RTT viene così aggiornato secondo la formula

$$RTT = \alpha RTT + (1 - \alpha)M$$

dove α è un fattore di perequazione che determina il peso dato al vecchio valore. Generalmente α corrisponde a 7/8.

Anche con un valore di RTT valido, la scelta di un timeout di ritrasmissione adatto è una questione complessa. Di norma, TCP utilizza βRTT , ma la parte complessa sta nello scegliere β . Nelle implementazioni iniziali β era sempre 2, ma l'esperienza ha insegnato che un valore costante era inflessibile perché non rispondeva al cambiamento della varianza.

nel 1988, Jacobson ha proposto di rendere β proporzionale alla deviazione standard della dimensione di densità di probabilità relativa al tempo di arrivo dell'acknowledgement, in modo che una varianza grande producesse un valore β elevato e viceversa. In particolare, suggerito l'utilizzo della *deviazione media* come stima approssimata della *deviazione standard*. Il suo algoritmo richiede di tenere traccia di un'altra variabile perequata, vale a dire la deviazione D . Al ricevimento di un acknowledgement è elaborata la differenza tra i valori previsto e osservato, $|RTT - M|$. Un valore perequato di questo risultato è inserito in D con la formula

$$D = \alpha D + (1 - \alpha) |RTT - M|$$

ove α può essere o meno lo stesso valore utilizzato per perequare RTT . Anche se D non è uguale esattamente alla deviazione standard, è un'approssimazione sufficiente; Jacobson ha mostrato come può essere elaborato utilizzando solo addizioni intere, sottrazioni e scoramenti. La maggior parte delle implementazioni TCP ora utilizza questo algoritmo e posta l'intervallo di timeout a

$$\text{Timeout} = RTT + 4 \times D$$

La scelta del fattore 4 è arbitraria, ma presenta due vantaggi. Innanzitutto, la moltiplicazione per 4 può essere eseguita con un solo scorrimento. In secondo luogo, riduce i timeout e le ritrasmissioni inutili perché meno dell'1% dei pacchetti giunge in un tempo superiore a quattro volte la deviazione standard. In realtà, Jacobson inizialmente proponeva di utilizzare 2, ma studi successivi hanno mostrato che 4 genera prestazioni migliori. Il problema che si verifica nella stima dinamica di RTT è la scelta delle azioni da compiere quando un segmento subisce un timeout e viene inviato nuovamente. Quando giunge l'acknowledgement, non è chiaro se fa riferimento alla prima trasmissione o alla successiva. Un errore di valutazione potrebbe seriamente contaminare la stima di RTT . Phil Karn ha scoperto questo problema nel modo peggiore. È un radioamatore interessato alla trasmissione di pacchetti TCP/IP mediante ham radio, un mezzo notoriamente inaffidabile (se va bene, solo la metà dei pacchetti arriva a destinazione). Ha quindi fatto una semplice proposta: evitare di aggiornare RTT per ogni segmento che è stato ritrasmesso. Altrimenti, il timeout viene raddoppiato a ogni errore fino a quando i segmenti giungono a destinazione al primo tentativo. Questa correzione è chiamata **algoritmo di Karn**. Molte implementazioni di TCP lo utilizzano.

Un timer di ritrasmissione non è l'unico utilizzato da TCP. Un altro è il **timer di persistenza**. È progettato per impedire la situazione di stallo (*deadlock*) seguente: il ricevente invia un acknowledgement con una dimensione della finestra pari a 0, chiedendo al mittente di aspettare. In seguito, il mittente aggiorna la finestra, ma il pacchetto con l'aggiornamento viene perso. Ora sia il mittente sia il ricevente attendono che l'altro faccia qualcosa. Quando il timer di persistenza scade, il mittente trasmette una sonda al ricevente. La risposta alla sonda fornisce la dimensione della finestra. Se è ancora zero il timer di persistenza viene reimpostato e il ciclo si ripete; se non è zero è possibile inviare i dati. Un altro timer utilizzato da alcune implementazioni è il **timer keepalive**. Quando una connessione

è stata inattiva per lungo tempo, il timer keepalive scade in modo che una delle parti controlli se l'altra è ancora presente. Se non ottiene risposta la connessione viene terminata. Questa funzionalità è controversa, perché aggiunge carico di lavoro e può terminare una connessione attiva a causa di una partizione transitoria della rete.

L'ultimo timer utilizzato da ogni connessione TCP è quello utilizzato nello stato *TIMED WAIT* durante la chiusura. Il conteggio prosegue per un tempo pari al doppio del tempo di vita del pacchetto, per garantire che alla chiusura di una connessione tutti i pacchetti creati siano stati rimossi.

6.5.11 UDP e TCP wireless

In teoria, i protocolli di trasporto dovrebbero essere indipendenti dalla tecnologia dello strato network sottostante. In particolare, TCP non dovrebbe preoccuparsi del fatto che IP sia in esecuzione su fibra ottica oppure onde radio. In pratica però l'informazione è importante, perché la maggior parte delle implementazioni TCP è stata attentamente ottimizzata sulla base di supposizioni che sono vere per le reti cablate ma non per le reti wireless. Ignorando le proprietà della trasmissione wireless si ottiene un'implementazione TCP logicamente corretta ma con prestazioni scadenti.

Il problema principale è l'algoritmo di controllo della congestione. Quasi tutte le implementazioni TCP odierebbero presumono che i timeout siano provocati dalla congestione, non dai pacchetti persi. Di conseguenza, alla scadenza di un timer, TCP rallenta ed esegue l'invio in modo meno vigoroso (per esempio con l'algoritmo di avvio lento di Jacobson). L'idea alla base di questo approccio è ridurre il carico di rete alleviando così la congestione.

Sfortunatamente i collegamenti per la trasmissione wireless sono altamente inaffidabili: perdono pacchetti in continuazione. L'approccio corretto alla gestione dei pacchetti persi è un nuovo invio, il più rapidamente possibile; il rallentamento non può che peggiorare le cose. Se il 20% dei pacchetti viene perso, quando il mittente trasmette 100 pacchetti al secondo la velocità effettiva è di 80 pacchetti al secondo. Se il mittente rallenta a 50 pacchetti al secondo, la velocità scende a 40 pacchetti al secondo.

Riassumendo, quando viene perso un pacchetto in una rete cablata il mittente dovrebbe rallentare, quando viene perso un pacchetto in una rete wireless il mittente dovrebbe rientrare con determinazione. Se il mittente non conosce la rete è difficile prendere la decisione giusta.

Spesso il percorso dal mittente al ricevente è eterogeneo. I primi 1.000 km potrebbero essere su una rete cablata, mentre l'ultimo chilometro potrebbe essere wireless. In questo caso prendere la decisione corretta su un timeout è ancora più difficile, perché dipende dal punto in cui si è verificato il problema. Una soluzione proposta da Bakne e Badrinath (1995), **TCP indiretto**, consiste nel dividere la connessione TCP in due connessioni separate, come mostrato nella Figura 6.39. La prima connessione va dal mittente alla stazione base, la seconda va dalla stazione base al ricevente. La stazione base copia semplicemente i pacchetti tra le connessioni, in entrambe le direzioni.

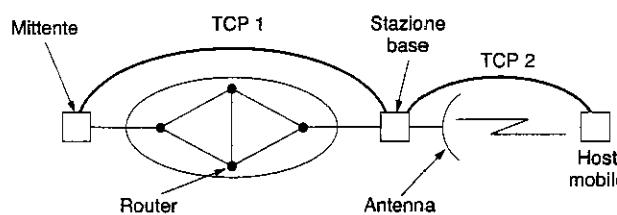


Figura 6.39. La divisione di una connessione TCP in due connessioni.

vantaggio di questo schema è che entrambe le connessioni ora sono omogenee. I timeout della prima connessione possono rallentare il mittente, mentre i timeout sulla seconda possono accelerarla; anche altri parametri possono essere regolati separatamente per le due connessioni. Lo svantaggio di questo schema è che viola le semantiche di TCP. Dal momento che ogni porzione della connessione è una connessione TCP completa, la stazione base manda gli acknowledgement a ogni segmento TCP nel solito modo, quindi la ricezione di un acknowledgement da parte del mittente non indica che il ricevente ha ottenuto il segmento, ma solo che la stazione base lo ha ricevuto.

Una soluzione diversa, dovuta a Balakrishnan et al. (1995), non interrompe le semantiche TCP, ma funziona apportando diverse piccole modifiche al codice dello strato network della stazione base. Una delle modifiche è l'aggiunta di un agente di snooping, che osserva e archivia nella cache i segmenti TCP in uscita dall'host mobile e gli acknowledgement provenienti da esso. Quando l'agente di snooping vede un segmento TCP in uscita dall'host mobile ma non rileva un acknowledgement prima della scadenza del suo timer (relativamente breve), ritrasmette il segmento senza comunicare all'origine l'operazione. Esegue la trasmissione anche quando rileva acknowledgement duplicati provenienti dall'host mobile che indicano che l'host mobile si è perso qualcosa. Gli acknowledgement duplicati sono smarriti immediatamente, per evitare che l'origine li interpreti come una congestione. Un svantaggio di questa trasparenza è che se il collegamento wireless è molto soggetto a perdite, l'origine potrebbe provocare un timeout in attesa di un acknowledgement e invocare l'algoritmo di controllo della congestione. Con TCP indiretto, l'algoritmo di controllo della congestione non sarà mai avviato, a meno che vi sia realmente una congestione sulla parte cablata della rete.

Il documento di Balakrishnan e altri offre anche una soluzione al problema dei segmenti persi originati dall'host mobile. Quando la stazione base osserva una lacuna nei numeri di sequenza, genera una richiesta di ripetizione selettiva dei byte mancanti utilizzando un'opzione TCP.

Utilizzando queste soluzioni, il collegamento wireless diventa più affidabile in entrambe le direzioni, senza che l'origine sia a conoscenza del fatto e senza cambiare le semantiche TCP.

Anche se UDP non risente degli stessi problemi di TCP, la comunicazione wireless introduce alcune difficoltà. Il problema principale è che i programmi utilizzano UDP pensando che sia altamente affidabile. Non sanno quali garanzie sono fornite, ma si aspettano che

sia quasi perfetto. In un ambiente wireless, UDP è tutto tranne che perfetto. Per i programmi che possono recuperare i messaggi UDP persi a costo considerevole, il passaggio improvviso da un ambiente dove i messaggi teoricamente possono essere persi (ma raramente questo avviene) a uno in cui vengono costantemente perduti potrebbe provocare prestazioni davvero scadenti.

La comunicazione wireless influisce anche su altre aree. Per esempio, come fa un host mobile a individuare una stampante locale a cui connettersi, anziché utilizzare la sua stampante di casa? Un problema correlato riguarda come ottenere la pagina WWW per la cella locale, anche se il suo nome non è noto. Inoltre, i designer di pagine WWW tendono a presumere la disponibilità di un'elevata banda. L'inserimento di un grande logo in ogni pagina diventa controproducente se richiede 10 secondi per la trasmissione su un collegamento wireless lento ogni volta che si fa riferimento alla pagina, irritando l'utente.

Quando le reti wireless diventeranno più comuni, i problemi dell'esecuzione di TCP su esse diverranno più importanti. Altri lavori sull'argomento sono riportati in (Barakat et al., 2000; Ghani e Dixit, 1999; Huston, 2001; Xylogenos et al., 2001).

6.5.12 TCP transazionale

Nella parte iniziale del capitolo si è osservato come la chiamata a procedure remote rappresenti un metodo per implementare sistemi client/server. Se la richiesta e la risposta sono abbastanza piccole per essere contenute in pacchetti singoli e l'operazione è equipotente, è possibile utilizzare semplicemente UDP. Tuttavia, se le condizioni non vengono soddisfatte, l'utilizzo di UDP è meno attraente. Per esempio, se la risposta può essere piuttosto grande, gli elementi devono essere disposti in sequenza; occorre inoltre ideare un meccanismo per ritrasmettere i pezzi perduti. In pratica l'applicazione deve reinventare TCP.

Chiaramente l'operazione non è delle più attraenti, ma nemmeno l'utilizzo di TCP è piacevole. Il problema riguarda l'efficienza. La normale sequenza dei pacchetti per eseguire una RPC su TCP è mostrata nella Figura 6.40(a). Nel caso migliore sono richiesti nove pacchetti.

I nove pacchetti sono i seguenti.

1. Il client invia un pacchetto *SYN* per stabilire una connessione.
2. Il server invia un pacchetto *ACK* per dare conferma di ricezione del pacchetto *SYN*.
3. Il client completa l'handshake a tre vie.
4. Il client invia la richiesta vera e propria.
5. Il client invia un pacchetto *FIN* per indicare che ha terminato l'invio.
6. Il server dà l'acknowledgement per richiesta e *FIN*.
7. Il server invia la risposta al client.
8. Il server invia un pacchetto *FIN* per indicare che ha terminato.
9. Il client dà l'acknowledgement per il pacchetto *FIN* del server.

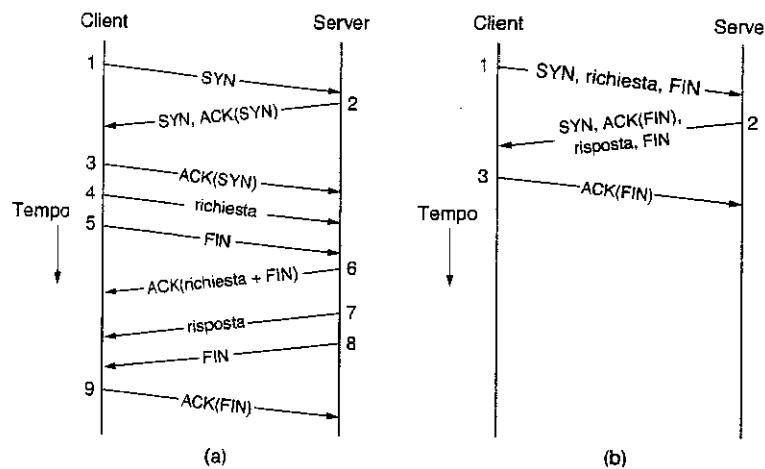


Figura 6.40 (a) Una RPC utilizzando TCP normale. (b) Una RPC utilizzando T/TCP.

erviamo che questo è il caso migliore; nel caso peggiore, richiesta e *FIN* del client vengono acknowledgement separati, così come risposta e *FIN* del server.

si può quindi chiedere se esiste la possibilità di combinare l'efficienza della RPC su TCP (solo due messaggi) ma con l'affidabilità di TCP. La risposta è: ci siamo quasi. È infatti possibile utilizzare una variante sperimentale di TCP chiamata **T/TCP** (*Transactional TCP*, TCP transazionale), descritta in RFC 1379 e 1644.

La logica centrale prevede di modificare leggermente la sequenza standard di impostazione di una connessione, per consentire il trasferimento dati durante l'impostazione. Il protocollo T/TCP è illustrato nella Figura 6.40(b). Il primo pacchetto del client contiene il bit *SYN*, la richiesta stessa e *FIN*. In pratica afferma: "Voglio stabilire una connessione; questi sono i dati, poi ho terminato".

Quando il server riceve la richiesta, preleva o elabora la risposta e decide come rispondere. Se la risposta può essere contenuta in un pacchetto, viene generata la risposta della Figura 6.40(b), che afferma: "Riconosco il tuo *FIN*; questa è la risposta, poi ho terminato". Il client dà quindi l'acknowledgement per il *FIN* del server e il protocollo termina in due messaggi.

Tuttavia, se il risultato è più grande di un pacchetto, il server ha la possibilità di non attivare il bit *FIN* e inviare più pacchetti prima di chiudere la sua direzione.

È probabilmente la pena ricordare che T/TCP non è l'unico miglioramento proposto per TCP. Un'altra proposta è **SCTP** (*Stream Control Transmission Protocol*). Le sue funzionalità comprendono la conservazione dei confini del messaggio, diverse modalità di consegna (per esempio la consegna non ordinata), il multihoming (destinazioni di backup) e acknowledgement selettivi (Stewart e Metz, 2001). Tuttavia, quando qualcuno propone di modificare qualcosa che ha funzionato così bene per lungo tempo, si sviluppa sempre una battaglia tra gli utenti che richiedono maggiori funzionalità e quelli che ritengono che se ha funzionato finora, non vale la pena di cambiarlo.

6.6 Problemi di prestazioni

I problemi che interessano le prestazioni sono molto importanti nelle reti di computer. Quando centinaia o migliaia di computer sono interconnessi, si verificano spesso interazioni complesse con conseguenze impreviste. Spesso questa complessità porta a prestazioni ridotte, e nessuno sa perché. Nei paragrafi seguenti verranno esaminate molte questioni che riguardano le prestazioni di rete, per vedere quali tipi di problemi esistono e che cosa è possibile fare per risolverli.

Sfortunatamente, la comprensione delle prestazioni di rete è più un'arte che una scienza. La scarsa teoria di base non ha quasi alcun valore nella pratica. Il meglio che si possa fare è fornire alcune regole generali acquisite con l'esperienza, e presentare esempi presi dal mondo reale. Questa trattazione è stata intenzionalmente ritardata fino a dopo lo studio dello strato trasporto in TCP, in modo da poter utilizzare TCP come esempio in vari casi. Lo strato trasporto non è l'unico in cui sorgono problemi di prestazioni; ne abbiamo visti alcuni nello strato network del capitolo precedente. Ciò nonostante, lo strato network tende a essere più interessato al controllo dell'instradamento e delle congestioni. Gli argomenti generali che riguardano l'intero sistema sono invece molto correlati al trasporto, pertanto questo capitolo rappresenta un punto appropriato per esaminarli.

Nei prossimi cinque paragrafi osserveremo altrettanti aspetti delle prestazioni di rete:

1. problemi di prestazioni
2. misurazione delle prestazioni della rete
3. progettazione del sistema per prestazioni migliori
4. elaborazione rapida delle TPDU
5. protocolli per future reti ad alte prestazioni.

Tra parentesi occorre un nome generico per le unità scambiate dalle entità di trasporto. Il termine TCP (segmento) provoca confusione e non viene mai utilizzato in questo contesto, fuori dal mondo di TCP. I termini ATM (CS-PDU, SAR-PDU e CPCS-PDU) sono specifici per ATM. "Pacchetto" fa chiaramente riferimento allo strato network, mentre "messaggio" appartiene allo strato applicazione. In mancanza di un termine standard, le unità scambiate dalle entità di trasporto saranno di nuovo chiamate TPDU. Il termine "pacchetto" sarà utilizzato come nome collettivo, per esempio nella frase "La CPU deve essere abbastanza veloce per elaborare i pacchetti in ingresso in tempo reale". Con questo intendiamo fare riferimento sia al pacchetto dello strato network sia alla TPDU encapsulata in esso.

6.6.1 I problemi di prestazioni nelle reti di computer

Alcuni problemi di prestazioni, come la congestione, sono provocati da sovraccarichi temporanei delle risorse. Se improvvisamente a un router giunge una quantità di traffico superiore a quella che può gestire, si verifica una congestione che influenza sulle prestazioni. La congestione è stata studiata in dettaglio nel capitolo precedente.

Le prestazioni degradano anche in presenza di uno squilibrio strutturale di risorse. Per

Esempio, se una linea di comunicazione gigabit termina in un PC low-end, la povera CPU non sarà in grado di elaborare i pacchetti in ingresso abbastanza velocemente e ne perderà qualcuno. Questi pacchetti saranno eventualmente ritrasmessi, provocando ritardi, perdita di banda e prestazioni ridotte in generale.

sovraffatti si possono innescare anche in modo sincrono. Per esempio, se una TPDU contiene un parametro errato (supponiamo sia la porta cui è destinata), in molti casi il ricevente restituira diligentemente una notifica di errore. Ora si pensi a cosa potrebbe accadere se una TPDU errata fosse trasmessa a 10.000 macchine: ognuna potrebbe restituire un messaggio di errore. Il **broadcast storm** (tempesta di broadcast) risultante potrebbe paralizzare la rete. UDP ha sofferto per questo problema fino a quando il protocollo è stato modificato in modo che gli host si astenessero dal rispondere agli errori nelle TPDU di DP inviate agli indirizzi di broadcast.

In secondo esempio di sovraccarico sincrono si può verificare dopo una mancanza di energia elettrica. Al ritorno della corrente, tutti i computer consultano contemporaneamente le loro boot ROM per iniziare il riavvio. Una tipica sequenza di riavvio potrebbe richiedere innanzitutto la visita a qualche server (DHCP) per conoscere l'identità del computer, e poi a qualche altro server per ottenere una copia del sistema operativo. Se centinaia di macchine svolgono questa operazione contemporaneamente, il server sarà bloccato dal carico di lavoro.

Anche in assenza di sovraccarichi sincroni e in presenza di risorse sufficienti, si potrebbero tenere prestazioni scadenti a causa della mancanza di ottimizzazioni del sistema. Per esempio, se un computer ha una CPU potente e molta memoria, ma non è stata allocata memoria sufficiente per lo spazio del buffer, si verificheranno alcuni overrun e le TPDU verranno perse. Nello stesso modo, se l'algoritmo di pianificazione non fornisce una priorità abbastanza elevata per l'elaborazione delle TPDU in arrivo, alcune potrebbero andare perse.

Un altro problema di ottimizzazione riguarda l'impostazione corretta dei timeout. Quando viene inviata una TPDU, viene normalmente avviato un timer per impedire la perdita delle PDU. Se il timeout è troppo breve si verificheranno ritrasmissioni inutili che intaseranno i canali. Se i timeout sono troppo lunghi, ci saranno invece ritardi inutili dopo la perdita di una PDU. Altri parametri regolabili comprendono il tempo di attesa prima di inviare un acknowledgement separato e il numero di ritrasmissioni dopo le quali abbandonare i tentativi.

Le reti gigabit portano con sé nuovi problemi di prestazioni. Si consideri, per esempio, l'invio di un burst di dati di 64 KB da San Diego a Boston con lo scopo di riempire il buffer di 64 KB del ricevente. Si supponga che il collegamento sia a 1 Gbps e che il ritardo relativo alla velocità della luce nella fibra in una direzione sia 20 msec. Inizialmente, a $t = 0$, il canale è vuoto, come illustrato nella Figura 6.41(a). Appena 500 μ sec dopo, nella Figura 6.41(b), tutte le TPDU sono uscite nella fibra. La TPDU di testa ora si trova nelle vicinanze di Brawley, nel sud della California. Tuttavia, il trasmettitore deve fermarsi fino a quando ottiene un aggiornamento della finestra.

Dopo 20 msec, la TPDU di testa raggiunge Boston, come mostrato nella Figura 6.41(c), e viene emesso il corrispondente acknowledgement. Per finire, 40 msec dopo l'inizio, il primo

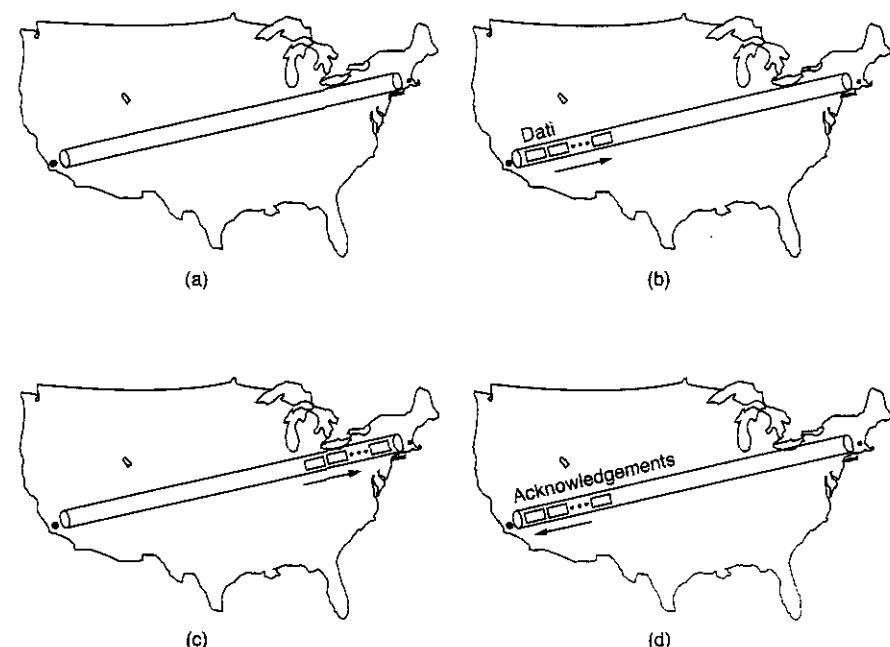


Figura 6.41. Lo stato della trasmissione di un megabit da San Diego a Boston. (a) A $t = 0$. (b) Dopo 500 μ sec. (c) Dopo 20 msec. (d) Dopo 40 msec.

acknowledgement torna al mittente e il secondo blocco può essere trasmesso. Dal momento che la linea di trasmissione è stata utilizzata per 0,5 msec su 40, l'efficienza è pari a 1,25%. Questa situazione è tipica dei vecchi protocolli in esecuzione su linee gigabit.

Un valore da ricordare durante l'analisi delle prestazioni di rete è il **prodotto banda-ritardo**, che si ottiene moltiplicando la banda (in bit al secondo) per il tempo di ritardo round-trip (in secondi). Il prodotto è la capacità del canale dal mittente al ricevente e viceversa (in bit).

Per l'esempio nella Figura 6.41, il prodotto banda-ritardo è 40 milioni di bit. In altre parole, il mittente dovrebbe trasmettere un blocco di 40 milioni di bit per essere in grado di continuare a trasmettere a piena velocità fino al ritorno del primo acknowledgement. Sono quindi necessari molti bit per riempire il canale (in entrambe le direzioni). Ecco perché un blocco di mezzo milione di bit raggiunge un'efficienza pari ad appena 1,25%: viene utilizzato solo l'1,25% della capacità del canale.

La conclusione che possiamo trarre è la seguente: per avere prestazioni elevate, la finestra del ricevente deve essere grande almeno quanto il prodotto banda-ritardo, e preferibilmente un po' di più, in modo che il ricevente possa non rispondere immediatamente. Per una linea gigabit intercontinentale, sono richiesti almeno 5 megabyte.

l'efficienza è talmente ridotta per l'invio di un megabit, è possibile immaginare quali orni può raggiungere per una breve richiesta di centinaia di byte. A meno che sia possibile trovare altri usi per la linea mentre il primo client attende la sua risposta, una linea abit non è migliore di una linea megabit, ma solamente più costosa.

Un altro problema relativo alle prestazioni è il jitter, che riguarda le applicazioni in cui il tempo è fondamentale. Garantire un tempo di trasmissione medio ridotto non è sufficiente: anche la deviazione standard deve essere piccola. Per raggiungere un tempo di trasmissione medio ridotto con una deviazione standard piccola servono notevoli sforzi ingegneristici.

.2 La misurazione delle prestazioni della rete

Quando le prestazioni di una rete sono scadenti, gli utenti si lamentano con gli amministratori chiedendo miglioramenti. Per migliorare le prestazioni, gli operatori devono prima determinare esattamente che cosa sta accadendo, e per farlo devono eseguire alcune misurazioni. Questo paragrafo esamina la misura delle prestazioni delle reti. La discussione uente è basata sul lavoro di Mogul (1993).

Ciclo di base utilizzato per migliorare le prestazioni di rete prevede i seguenti passaggi:

1. misurare le prestazioni e i parametri di rete rilevanti
2. cercare di capire che cosa sta accadendo
3. cambiare un parametro.

Questi passaggi vengono ripetuti fino a quando le prestazioni sono sufficienti o fino a quando diventa chiaro che non si possono fare altri perfezionamenti.

Misurazioni si possono eseguire in molti modi e in molte ubicazioni (sia fisicamente, nella pila di protocolli). La misurazione più elementare consiste nell'avviare un timer all'inizio di qualche attività, e utilizzarlo per vedere quanto tempo impiega. Per esempio, il tempo richiesto da una TPDU per il riconoscimento è una misurazione fondamentale. Altre misurazioni vengono eseguite con i contatori, che registrano la frequenza con cui si verifica un evento (per esempio il numero di TPDU perse). Per finire, spesso gli operatori interessati a conoscere alcune quantità, per esempio il numero di byte elaborati in un certo intervallo di tempo.

Misurazione dei parametri e delle prestazioni di rete presenta molti tranelli. Di seguito sono elencati alcuni. Qualsiasi tentativo sistematico di misurare le prestazioni di rete dovrebbe cercare attentamente di evitarli.

Assicurarsi che la dimensione del campione sia sufficientemente elevata

In bisogna misurare il tempo di invio di una sola TPDU, ma ripetere la misurazione per un milione di volte e calcolare la media. Un campione grande ridurrà l'incertezza nella media e nella deviazione standard misurate. Questa incertezza può essere calcolata utilizzando formule statistiche standard.

Assicurarsi che i campioni siano rappresentativi

Teoricamente, l'intera sequenza di un milione di misurazioni dovrebbe essere ripetuta diverse volte al giorno e alla settimana per vedere l'effetto di diversi carichi di sistema sulla quantità misurata. La misurazione della congestione, per esempio, è di poco conto se viene fatta in un momento in cui non ci sono congestioni. A volte i risultati possono sembrare poco intuitivi inizialmente, per esempio perché vi sono pesanti congestioni alle 10, alle 11, alle 13 e alle 14, ma nessuna a mezzogiorno (perché gli utenti sono fuori a pranzo).

Fare attenzione quando si usa un orologio grossolano

Gli orologi dei computer lavorano incrementando un contatore a intervalli regolari. Per esempio, un timer in millisecondi aggiunge 1 a un contatore ogni 1 msec. Utilizzare tale timer per misurare un evento che avviene in meno di 1 msec è possibile, ma l'operazione richiede maggiore cura (naturalmente ci sono computer che hanno orologi più accurati). Per misurare il tempo di invio di una TPDU, per esempio, l'orologio di sistema (supponiamo in millisecondi) dovrebbe eseguire la lettura all'ingresso nel codice dello strato trasporto e all'uscita da quest'ultimo. Se il tempo di invio reale della TPDU è 300 μ sec, la differenza tra le due letture sarà 0 o 1 (entrambe errate). Tuttavia, se la misurazione viene ripetuta un milione di volte e il totale di tutte le misurazioni sommato e diviso per un milione, il tempo medio sarà accurato fino a 1 μ sec.

Assicurarsi che non accada nulla di imprevisto durante i test

La realizzazione di misurazioni su un sistema universitario nel giorno in cui vengono svolti i principali progetti di laboratorio può produrre risultati diversi da quelli ottenibili il giorno dopo. allo stesso modo, se qualche ricercatore ha deciso di tenere una videoconferenza sulla rete durante i test, i risultati potrebbero essere tendenziosi. È preferibile eseguire i test su un sistema inattivo e creare autonomamente l'intero carico di lavoro. Anche questo approccio presenta però dei tranelli. Si potrebbe pensare che nessuno usi la rete alle 3 del mattino, ma quello potrebbe essere il momento preciso in cui i programmi di backup automatico iniziano a copiare tutti i dischi su nastri. Inoltre, il traffico sulle proprie pagine World Wide Web potrebbe essere intenso perché proviene da zone con fusi orari lontani.

Le cache possono disturbare le misurazioni

Il modo più ovvio per misurare i tempi di trasferimento dei file è aprire un file grande, leggerlo interamente, chiuderlo e annotare il tempo necessario. Occorre poi ripetere la misurazione molte altre volte per ottenere una buona media. Il problema sta nel fatto che il sistema potrebbe archiviare il file nella cache, pertanto solo la prima misurazione riguarderebbe effettivamente il traffico di rete: il resto sarebbero soltanto letture dalla cache locale. I risultati di queste misurazioni sono essenzialmente privi di valore (a meno che si vogliano misurare le prestazioni della cache).

esso è possibile aggirare l'archiviazione nella cache con un semplice traboccamento. Per esempio, se la cache è di 10 MB, il ciclo di test potrebbe aprire, leggere e chiudere due file di 10 MB a ogni passaggio, nel tentativo di forzare a 0 l'hit rate della cache. Ancora una volta è opportuno essere prudenti, se non si conosce alla perfezione l'algoritmo di caching.

Buffering può avere un effetto simile. Un popolare programma per la misura delle prestazioni di TCP/IP segnalava notoriamente che UDP poteva raggiungere prestazioni sostanzialmente maggiori di quelle ammesse dalla linea fisica. Come poteva succedere? La chiamata a UDP restituiva normalmente il controllo non appena il messaggio veniva catturato dal kernel e aggiunto alla coda di trasmissione. Se esiste spazio sufficiente per i buffer, la temporizzazione di 1.000 chiamate UDP non indica la spedizione di tutti i dati. La maggior parte potrebbe trovarsi ancora nel kernel, ma il programma che misura le prestazioni pensa che siano stati tutti trasmessi.

Imprendere che cosa si sta misurando

Se si misura il tempo di lettura di un file remoto, le misurazioni dipendono dalla rete, sistemi operativi sul client e sul server, dalle schede di interfaccia hardware utilizzate, driver e da altri fattori. Se le misurazioni sono svolte con attenzione, sarà possibile scoprire il tempo di trasferimento dei file per la configurazione utilizzata. Se l'obiettivo è ottimizzare questa particolare configurazione, le misurazioni sono perfette. Tuttavia, per eseguire misurazioni simili su tre sistemi diversi al fine di scegliere quale scheda di interfaccia di rete acquistare, i risultati potrebbero essere completamente sbagliati per il fatto che uno dei driver di rete è veramente scadente e utilizza solo il 10% delle stazioni della scheda.

Prendere attenzione all'estrapolazione dei risultati

Supponga di eseguire alcune misurazioni con carichi di rete simulati da 0 (inattivo) a 0,4% della capacità, come mostrato dai punti dei dati e dalla linea continua nella Figura 2. Potrebbe essere allentante estrapolarli linearmente, come mostrato dalla linea tratteggiata. Tuttavia, molti risultati di accodamento implicano un fattore $1/(1 - \rho)$, dove ρ è il carico; pertanto i valori reali potrebbero somigliare più ai valori mostrati con i trattini lunghi, che salgono in modo più rapido del grafico lineare.

3 Progettazione del sistema per aumentare le prestazioni

Misurazioni possono spesso migliorare notevolmente le prestazioni, ma non possono sostituire una valida progettazione iniziale. Una rete progettata in modo scadente può essere migliorata solo relativamente. In effetti, dovrebbe essere riprogettata da zero. Questo paragrafo, presenteremo alcune regole basate sull'esperienza con diverse reti. Queste regole sono correlate alla progettazione del sistema, non solo della rete, dato che il software e il sistema operativo sono spesso più importanti dei router e

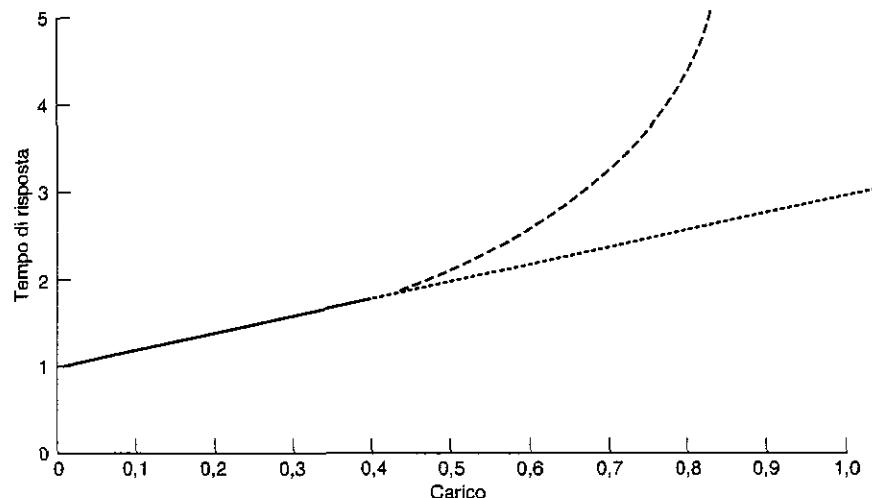


Figura 6.42. La risposta come funzione del carico.

delle schede di interfaccia. La maggior parte delle idee sono da anni ben note ai progettisti di rete, e sono state trasmesse oralmente di generazione in generazione. Sono state elencate in modo esplicito da Mogul (1993); la descrizione seguente segue sostanzialmente la sua. Un'altra fonte rilevante è (Metcalfe, 1993).

Regola 1: la velocità della CPU è più importante della velocità della rete

Una lunga esperienza ha dimostrato che, in quasi tutte le reti, l'overhead (sovraffaccia) del sistema operativo e dei protocolli domina sul tempo effettivo speso nel cavo. Per esempio, in teoria il tempo RPC minimo su una Ethernet è 102 µsec, corrispondenti a una richiesta minima (64 byte) seguita da una risposta minima (64 byte). In pratica, a causa dell'inefficienza del software, ottenere un valore simile per una RPC è una conquista sensazionale. Allo stesso modo, i problemi più grandi quando si lavora a 1 Gbps riguardano il modo per ottenere il trasporto rapido dei bit dal buffer di uscita dell'utente verso la fibra, e per elaborare i dati alla stessa velocità con cui arrivano alla CPU. Nelle applicazioni pratiche, raddoppiando la velocità della CPU spesso è possibile raddoppiare la velocità di trasferimento. Al contrario, raddoppiando la capacità di rete spesso non si ottengono effetti perché il collo di bottiglia si trova generalmente negli host.

Regola 2: ridurre il numero di pacchetti per ridurre l'overhead del software

L'elaborazione di una TPDU provoca overhead per ogni TPDU (per esempio l'elaborazione delle intestazioni) e una certa quantità di elaborazione per byte (per esempio il calcolo del checksum). Quando viene inviato un milione di byte, l'overhead per byte è lo stesso

indipendentemente dalla dimensione della TPDU. Tuttavia, usando TPDU di 128 byte l'overhead per TPDU è 32 volte maggiore rispetto a quello che si misura usando TPDU a 1 KB. Questo overhead cresce velocemente.

Inoltre all'overhead della TPDU, occorre considerare un overhead dei livelli inferiori. Ogni pacchetto in arrivo provoca un interrupt. Su un moderno processore pipelined, ogni interrupt spezza la pipeline della CPU, interferisce con la cache, richiede una modifica al contesto di gestione della memoria e forza il salvataggio di un buon numero di registri della CPU. Un fattore di riduzione n nel numero di TPDU inviate riduce quindi l'overhead dei pacchetti e gli interrupt di un fattore n .

Questa osservazione incoraggia a raccogliere una buona quantità di dati prima della trasmissione, in modo da ridurre gli interrupt all'altro capo. L'algoritmo di Nagle e la soluzione di Clark al problema della silly window syndrome sono tentativi di svolgere questa operazione.

Regola 3: minimizzare i cambi di contesto

I cambi di contesto (per esempio dalla modalità kernel alla modalità utente) sono inizialmente presenti. Presentano le stesse caratteristiche negative degli interrupt, la peggiore delle quali è una lunga serie di cache miss iniziali. I cambi di contesto possono essere ridotti se la procedura di libreria che invia dati esegue un buffering interno fino ad accumularne una buona quantità. Allo stesso modo, sul lato ricevente, le piccole TPDU in ingresso dovrebbero essere raccolte e passate all'utente in blocco anziché singolarmente, in modo da ridurre i cambi di contesto.

Il migliore dei casi, un pacchetto in arrivo provoca un cambio di contesto dall'utente corrente al kernel, e un passaggio al processo ricevente per fornirgli i dati appena arrivati. Sfortunatamente, con molti sistemi operativi avvengono altri cambi di contesto. Per esempio, se il gestore della rete viene eseguito come processo speciale nello spazio dell'utente, l'arrivo di un pacchetto può provocare un cambio di contesto dall'utente corrente al kernel, poi un altro dal kernel al gestore di rete, seguito da un altro di ritorno al kernel e infine da un passaggio dal kernel al processo ricevente. Questa sequenza è mostrata nella figura 6.43. Tutti questi cambi di contesto eseguiti provocano sprechi del tempo della CPU e hanno un effetto devastante sulle prestazioni della rete.

Regola 4: minimizzare le copie

È una cosa peggiore dei cambi di contesto numerosi, e sono le copie multiple. Non è insolito che un pacchetto in ingresso venga copiato tre o quattro volte prima che la TPDU sia racchiusa e consegnata. Dopo che un pacchetto è stato ricevuto dall'interfaccia di rete in uno speciale buffer hardware sulla scheda, viene di norma copiato in un buffer del kernel. Da qui viene copiato nel buffer dello strato network, quindi in un buffer dello strato di trasporto e infine nel processo applicativo ricevente.

Un sistema operativo intelligente copia una word per volta, ma non è insolito aver bisogno di circa cinque istruzioni per word (caricamento, archiviazione, incremento di un

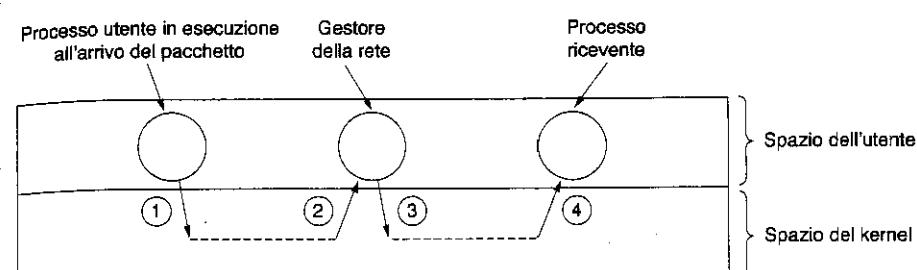


Figura 6.43. Quattro cambi di contesto per gestire un pacchetto con un gestore di rete nello spazio utente.

registro indice, verifica della fine dei dati, salto condizionale). Per creare tre copie di ogni pacchetto, con cinque istruzioni per ogni word di 32 bit copiata, servono 15/4 ovvero circa quattro istruzioni per ogni byte copiato. Su una CPU da 500 MIPS un'istruzione richiede 2 nsec, pertanto ogni byte necessita di 8 nsec di tempo di elaborazione o circa 1 nsec per bit, generando una velocità massima di 1 Gbps. Andando a considerare l'overhead per l'elaborazione delle intestazioni, la gestione degli interrupt e i cambi di contesto, si potrebbero raggiungere al massimo 500 Mbps, senza considerare l'elaborazione effettiva dei dati. Chiaramente, la gestione di una Ethernet 10 Gbps a piena velocità è fuori discussione.

In effetti, probabilmente non è possibile gestire nemmeno una linea a 500 Mbps a piena velocità. Nel calcolo precedente, abbiamo supposto che un computer da 500 MIPS può eseguire 500 milioni di istruzioni al secondo. In realtà, i computer possono lavorare a queste velocità solo se non fanno riferimento alla memoria. Le operazioni sulla memoria sono circa 10 volte più lente delle istruzioni registro/registro (vale a dire 20 nsec per istruzione). Se il 20% delle istruzioni fa riferimento alla memoria (vale a dire che si tratta di cache miss), che è un valore tipico lavorando su pacchetti in ingresso, il tempo medio di esecuzione delle istruzioni è 5,6 nsec (0,8 x 2 + 0,2 x 20). Con quattro istruzioni per byte sono necessari 22,4 nsec per byte, o 2,8 nsec per bit, ottenendo circa 357 Mbps, e considerando un overhead del 50% si ottengono 178 Mbps. Occorre notare che in questo caso l'assistenza dell'hardware non sarebbe di aiuto: il problema sta nelle troppe copie eseguite dal sistema operativo.

Regola 5: è possibile comprare più banda ma non diminuire il ritardo

Le tre regole successive hanno a che fare con la comunicazione, anziché con l'elaborazione del protocollo. La prima regola afferma che, se si desidera maggiore banda, è sufficiente acquistarla. Inserendo una seconda fibra accanto alla prima si raddoppia la banda pur senza ridurre il ritardo. La riduzione del ritardo richiede il miglioramento del software che gestisce il protocollo, del sistema operativo o dell'interfaccia di rete. Anche se vengono apportati tutti questi miglioramenti, il ritardo non si riduce se il collo di bottiglia è il tempo di trasmissione.

Regola 6: evitare la congestione è meglio che correggere il problema

Il vecchio detto "prevenire è meglio che curare" vale certamente anche per la congestione di rete. Quando una rete è congestionata i pacchetti vengono persi, la banda sprecata, s'introducono ritardi inutili e così via. Il recupero dalle congestioni richiede tempo e pazienza: è quindi preferibile evitarle del tutto. Evitare le congestioni è come vaccinarsi: fa un po' male all'inizio, ma impedisce che qualcosa faccia ancora più male in futuro.

Regola 7: evitare i timeout

I timer sono necessari nelle reti, ma dovrebbero essere utilizzati raramente; anche i timeout dovrebbero essere minimizzati. Alla scadenza di un timer viene generalmente ripetuta qualche azione. Se è realmente necessario ripetere l'azione è giusto farlo, ma ripeterla senza necessità è uno spreco di risorse.

La strada maestra per evitare lavoro extra consiste nell'assicurarsi di regolare i timer in modo leggermente conservativo. Un timer che per scadere impiega troppo tempo aggiunge un ritardo extra alla connessione nell'evento (improbabile) che una TPDU vada persa.

Un timer che scade quando non dovrebbe utilizza il tempo della CPU, spreca banda e impone un carico extra su decine di router senza valide ragioni.

5.6.4 Elaborazione rapida delle TPDU

La morale della storia precedente è che l'ostacolo principale alle reti veloci è il software che gestisce il protocollo, e in questo paragrafo si esaminano alcuni metodi per accelerarlo. Per ulteriori informazioni, consultare (Clark et al., 1989; Chase et al., 2001).

L'overhead di elaborazione delle TPDU ha due componenti, entrambi da ridurre: l'overhead per TPDU e l'overhead per byte. La chiave di volta per ottenere un'elaborazione rapida della TPDU sta nell'isolare il caso normale (trasferimento di dati unidirezionale) ed elaborarlo in modo speciale. Anche se è necessaria una sequenza di TPDU speciali per entrare nello stato *ESTABLISHED*, una volta entrati nello stato l'elaborazione della TPDU resta ovvia fino a quando un lato inizia a chiudere la connessione.

Iniziamo esaminando il lato di invio nello stato *ESTABLISHED* quando vi sono dati da trasmettere. Per chiarezza, qui dobbiamo presumere che l'entità di trasporto si trovi nel kernel, anche se le stesse idee valgono per un processo dello spazio dell'utente o per una libreria all'interno del processo di invio. Nella Figura 6.44, il processo di invio fa una trap nel kernel per eseguire SEND. La prima operazione svolta dall'entità di trasporto è valutare se questo è il caso normale: lo stato è *ESTABLISHED*, nessun lato sta tentando di chiudere la connessione, è stata inviata una TPDU completa regolare (non fuori banda) e il ricevente è disponibile spazio finestra sufficiente. Se tutte le condizioni sono soddisfatte, non servono altri test e l'entità di trasporto può imboccare il percorso rapido per l'invio. Generalmente, questo percorso viene seguito la maggior parte delle volte. Formalmente le intestazioni delle TPDU dati consecutive sono quasi le stesse. Per trarre

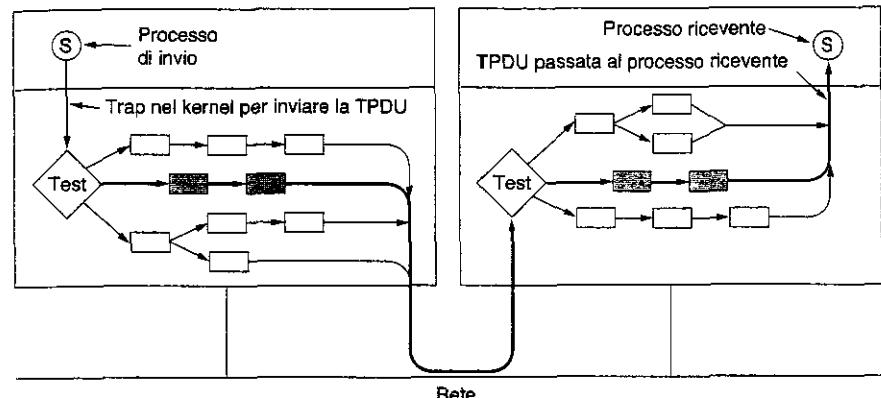


Figura 6.44. Il percorso rapido dal mittente al ricevente è mostrato con una linea spessa. I passaggi che richiedono elaborazione sono ombreggiati nel tracciato.

vantaggio da questo fatto, un'intestazione prototipo viene memorizzata all'interno dell'entità di trasporto. All'inizio del percorso rapido, viene copiata il più velocemente possibile in un buffer di memoria, e i campi che cambiano da una TPDU all'altra TPDU vengono sovrascritti nel buffer. Spesso, questi campi vengono facilmente derivati dalle variabili di stato, come il numero di sequenza successivo. Allo strato network viene quindi passato un puntatore all'intera intestazione della TPDU, insieme a un puntatore ai dati dell'utente. Qui si può adottare la stessa strategia (non mostrata nella Figura 6.44). Per finire, lo strato network consegna il pacchetto risultante allo strato data link per la trasmissione.

Come esempio del funzionamento pratico di questo principio, consideriamo TCP/IP. La Figura 6.45(a) mostra l'intestazione TCP. I campi identici nelle TPDU consecutive del flusso unidirezionale sono ombreggiati. Tutto ciò che l'entità di trasporto in uscita deve fare è copiare le cinque word dell'intestazione prototipo nel buffer di output, inserire il numero di sequenza successivo (copiandolo da una word in memoria), calcolare il checksum e incrementare il numero di sequenza in memoria. Può quindi consegnare l'intestazione e i dati a una speciale procedura IP per l'invio di una TPDU standard con dimensione massima. IP copia poi la sua intestazione prototipo di cinque parole (Figura 6.45(b)) nel buffer, compila il campo *identificazione* e calcola il proprio checksum. Ora il pacchetto è pronto per la trasmissione.

Ora osserviamo l'elaborazione del percorso rapido sul lato ricevente della Figura 6.44. Il primo passaggio è l'individuazione del record di connessione per la TPDU in ingresso. Per TCP, il record della connessione può essere memorizzato in una tabella hash la cui chiave è rappresentata da alcune semplici funzioni dei due indirizzi IP e delle relative porte. Una volta individuato il record della connessione, gli indirizzi e le porte devono essere confrontati per verificare di aver individuato il giusto record.

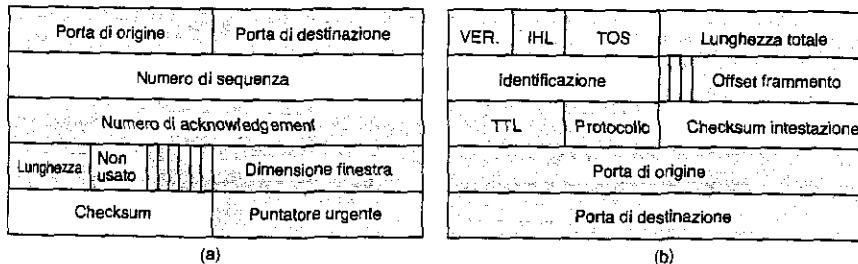


Figura 6.45. (a) L'intestazione di TCP. (b) L'intestazione di IP. In entrambi i casi, i campi ombreggiati vengono presi dal prototipo senza modifiche.

Un'ottimizzazione che spesso accelera ancor più la ricerca del record di connessione consiste nel mantenere un puntatore all'ultimo record utilizzato, provandolo per primo. Clark et al. (1989) hanno provato il metodo e hanno osservato una frequenza superiore al 90%. Altre euristiche di ricerca sono descritte in (McKenney e Dove, 1992).

La TPDU viene poi controllata per stabilire se è normale: lo stato è *ESTABLISHED*, entrambi i lati non stanno provando a chiudere la connessione, la TPDU è completa, non sono impostati flag speciali e il numero di sequenza è quello previsto. Questi test richiedono solo una manciata di istruzioni. Se tutte le condizioni vengono soddisfatte, si esegue la chiamata a una speciale procedura TCP per il percorso rapido.

Il percorso rapido aggiorna il record della connessione e copia i dati verso l'utente. Durante la copia elabora anche il checksum, eliminando un passaggio extra sui dati. Se il checksum è esatto, il record della connessione viene aggiornato e viene restituito un acknowledgement. Lo schema generale che prevede di controllare prima se l'intestazione è quella prevista, e poi di utilizzare una procedura speciale per la gestione di questo caso è chiamato **header prediction** (predizione dell'intestazione). Molte implementazioni di TCP lo utilizzano. Quando questa ottimizzazione e tutte le altre discusse nel capitolo vengono utilizzate insieme, è possibile eseguire TCP al 90% della velocità di una copia locale da memoria a memoria, nell'ipotesi che la rete sia abbastanza veloce.

Altre due aree dove sono possibili guadagni in termini di prestazioni sono la gestione dei buffer e quella dei timer. Per la gestione del buffer si devono evitare copie non necessarie, come affermato in precedenza. La gestione dei timer è importante, perché quasi tutti i timer non scadono: sono programmati per proteggere dalle perdite delle TPDU, ma la maggior parte delle TPDU arriva correttamente insieme ai relativi acknowledgement. Di conseguenza, è importante ottimizzare la gestione dei timer per il caso in cui la scadenza è rara.

Uno schema popolare consiste nell'usare una linked list di eventi dei timer ordinati per tempo di scadenza. La voce di testa contiene un contatore che specifica il numero di battiti mancanti alla scadenza; ogni voce successiva contiene un contatore che specifica il numero di battiti che mancano alla scadenza oltre a quelli della voce precedente. Pertanto, se i timer scadono a 3, 10 e 12 battiti, i tre contatori presenteranno i valori 3, 7 e 2.

A ogni battito dell'orologio, il contatore nella voce di testa viene decrementato. Quando raggiunge zero, il suo evento viene elaborato e la successiva voce nell'elenco diventa quella di testa. Il suo contatore non deve essere modificato. In questo schema, l'inserimento e l'eliminazione di timer sono operazioni costose: i tempi di esecuzione sono proporzionali alla lunghezza della lista.

Un approccio più efficiente può essere utilizzato se l'intervallo massimo del timer ha un valore limitato e conosciuto in anticipo. È possibile utilizzare un array chiamato **timing wheel**, come mostrato nella Figura 6.46. Ogni slot corrisponde a un battito dell'orologio; è mostrata la situazione al tempo $T = 4$. I timer sono pianificati per scadere ai battiti 3, 10 e 12. Se improvvisamente viene impostato un nuovo timer che deve scadere entro 7 battiti, si crea una voce nello slot 11. Allo stesso modo, se il timer impostato per $T + 10$ deve essere annullato, occorre eseguire la ricerca nella lista che inizia con lo slot 14 e rimuovere la voce richiesta. È possibile notare che l'array della Figura 6.46 non può accomodare i timer oltre $T + 15$.

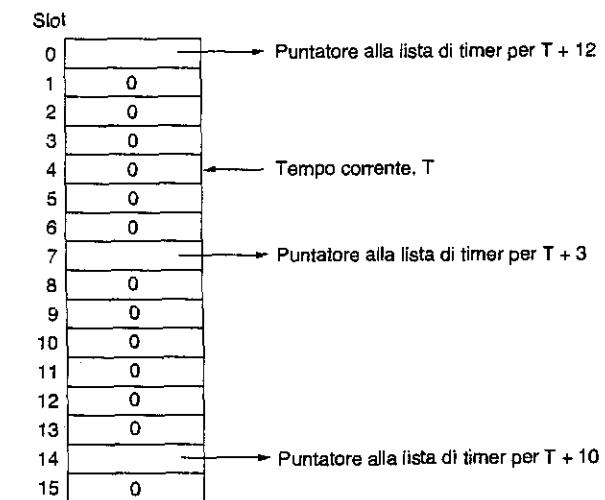


Figura 6.46. Una timing wheel.

A ogni battito dell'orologio, il puntatore del tempo corrente avanza circolarmente di uno slot: Se la voce indicata è diversa da zero, tutti i suoi timer vengono elaborati. Molte variazioni all'idea di base sono discusse in (Varghese e Lauck, 1987).

6.6.5 I protocolli per le reti gigabit

All'inizio degli anni '90 hanno iniziato ad apparire le reti gigabit. All'inizio la gente ha cercato di utilizzarle con i vecchi protocolli, ma sorse presto dei problemi. In questo paragrafo discuteremo alcuni di questi problemi e la direzione che i nuovi protocolli stanno prendendo per risolverli, mentre all'orizzonte già compaiono reti ancora più veloci.

primo problema è che molti protocolli utilizzano numeri di sequenza a 32 bit. Agli esordi di Internet le linee tra i router erano per lo più linee affittate a 56 kbps, pertanto un host che inviava ininterrottamente alla velocità massima impiegava circa una settimana per utilizzare tutti i numeri di sequenza. Per chi progettò TCP, 2^{32} era un'approssimazione decente di infinito, perché il pericolo di avere in circolazione pacchetti trasmessi una settimana prima era davvero ridotto. Con Ethernet a 10 Mbps il tempo di ciclo divenne 57 minuti, più breve ma ancora gestibile. Una Ethernet a 1 Gbps che riversa dati su Internet ha un tempo di ciclo di circa 34 secondi, molto inferiore al minimo tempo di vita dei pacchetti su Internet (120 secondi). Detto questo, 2^{32} non è una valida approssimazione di infinito, perché un mittente può eseguire un intero ciclo nello spazio dei numeri di sequenza quando sono ancora presenti vecchi pacchetti. RFC 1323 offre una via di fuga.

Il problema è che molti progettisti di protocolli hanno semplicemente presunto, senza controllare, che il tempo necessario a utilizzare l'intero spazio della sequenza avrebbe ampiamente superato la vita massima dei pacchetti; di conseguenza, non c'era bisogno di preoccuparsi del problema dei vecchi duplicati rimasti in circolazione al termine di un ciclo sui numeri di sequenza. A una velocità gigabit questa supposizione infondata non è più valida.

Un secondo problema è che le velocità di comunicazione sono migliorate più in fretta delle velocità di elaborazione. Negli anni '70, ARPANET funzionava a 56 kbps e utilizzava computer che lavoravano a circa 1 MIPS. I pacchetti erano di 1.008 bit, pertanto ARPANET era in grado di consegnare circa 56 pacchetti al secondo. Con quasi 18 msec disponibili per pacchetto, un host poteva permettersi di spendere 18.000 istruzioni per elaborare un pacchetto. Naturalmente, questo metodo avrebbe occupato l'intera CPU; tuttavia, era possibile dedicare 9.000 istruzioni per pacchetto e avere ancora metà del tempo macchina libero per svolgere altri lavori.

Proviamo a confrontare questi numeri con computer a 1.000 MIPS che scambiano pacchetti di 1.500 byte su una linea gigabit. I pacchetti possono fluire a una velocità di 80.000 al secondo, pertanto l'elaborazione di ciascuno va completata in 6,25 μ sec per poter assegnare metà della CPU alle applicazioni. In 6,25 μ sec un computer a 1.000 MIPS può eseguire 6.250 istruzioni, solo 1/3 di quelle disponibili per gli host ARPANET. Inoltre, le moderne istruzioni RISC svolgono meno operazioni per istruzione rispetto alle vecchie istruzioni CISC, pertanto il problema è ancora peggiore di quanto possa apparire. La conclusione è la seguente: c'è sempre meno tempo per l'elaborazione dei protocolli, che perciò devono diventare più semplici.

Il terzo problema è che un protocollo del tipo "torna indietro di n" ha prestazioni scattanti se viene eseguito su linee con un prodotto banda-ritardo elevato. Si consideri, per esempio, una linea di 4.000 Km che opera a 1 Gbps. Il tempo di trasmissione round-trip è 1 msec: in questo intervallo un mittente può trasmettere 5 MB. Se viene rilevato un errore, trascorreranno 40 msec prima che il mittente sia avvisato. Se viene utilizzato un protocollo torna indietro di n, il mittente dovrà ritrasmettere non solo il pacchetto errato, ma anche i 5 MB di pacchetti seguenti; è chiaramente un enorme spreco di risorse.

Il quarto problema è che le linee gigabit sono fondamentalmente diverse dalle linee a giga-bit, perché le linee gigabit sono limitate dal ritardo anziché dalla banda.

Nella Figura 6.47 è mostrato il tempo richiesto per trasferire un file di 1 megabit lungo una linea di 4.000 Km a differenti velocità di trasmissione. A una velocità di 1 Mbps, il tempo di trasmissione è dominato dalla velocità d'invio dei bit. Per 1 Gbps, il ritardo di round-trip pari a 40 msec domina sul tempo di 1 msec necessario per trasferire i bit sulla fibra. Ulteriori aumenti della banda non fornirebbero praticamente alcun aumento di prestazioni.

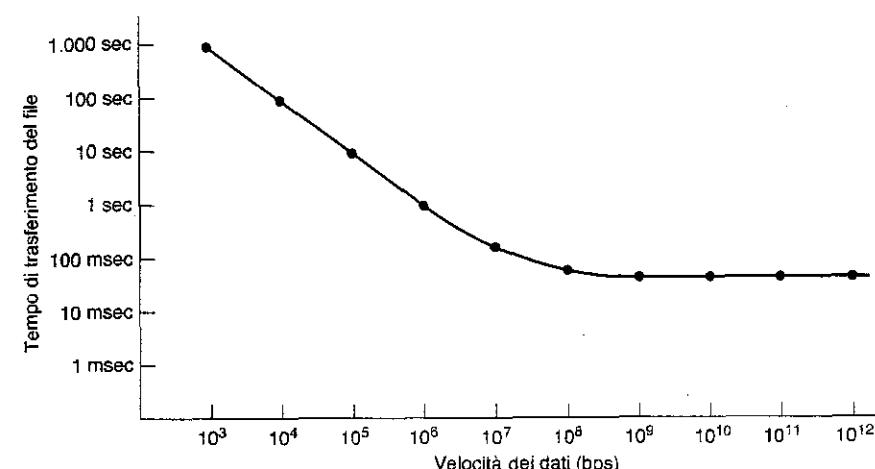


Figura 6.47. Il tempo per trasferire e dare l'acknowledgement a un file di 1 megabit su una linea di 4.000 Km.

La Figura 6.47 rivela spiacimenti ricadute sui protocolli di rete: afferma che i protocolli stop-and-wait, come RPC, hanno un limite superiore di prestazioni intrinseco. Questo limite è dettato dalla velocità della luce. Nessun progresso tecnologico nell'ottica potrà risolvere il problema (anche se nuove leggi della fisica potrebbero essere utili). Un quinto problema che vale la pena menzionare non riguarda come gli altri una tecnologia o un protocollo, ma il risultato di nuove applicazioni. In parole povere, per molte applicazioni gigabit (come quelle multimediali) la varianza nei tempi di arrivo del pacchetto è importante quanto il ritardo medio stesso. Una velocità di consegna lenta ma uniforme è spesso preferibile a una velocità rapida ma incostante.

Passiamo ora dai problemi ai modi per risolverli. Innanzitutto si forniranno alcune considerazioni generali, poi saranno esaminati i meccanismi del protocollo, la struttura dei pacchetti e il software dei protocolli.

Il principio di base che tutti i progettisti di reti gigabit dovrebbero imparare a memoria è il seguente:

Progettare per la velocità, non per ottimizzare la banda.

I vecchi protocolli spesso sono stati progettati per ridurre il numero di bit sul cavo, e in molti casi utilizzano piccoli campi radunati insieme per formare byte e word.

Oggi la banda disponibile è molto elevata e il problema è l'elaborazione dei protocolli, che di conseguenza vanno progettati per minimizzarla. I disegnatori di IPv6 hanno chiaramente compreso questo principio.

Un modo allettante per migliorare la velocità consiste nel costruire veloci interfacce di rete hardware. La difficoltà di questa strategia sta nel fatto che, a meno che il protocollo sia davvero semplice, l'hardware è rappresentato da una scheda con una seconda CPU e un suo programma. Per assicurare che il coprocessore di rete sia più economico della CPU principale, viene di norma utilizzato un chip più lento. La conseguenza di questa progettazione è che gran parte del tempo della CPU principale (veloce) è trascorso nell'inattività, in attesa che la seconda CPU (lenta) svolga il lavoro importante. È ridicolo supporre che la CPU principale abbia altro lavoro da svolgere nell'attesa. Inoltre, quando due CPU generiche comunicano, si possono verificare situazioni di conflitto, pertanto si devono elaborare protocolli per sincronizzare correttamente i due processori. Solitamente l'approccio migliore prevede di creare protocolli semplici e di lasciare che la CPU principale svolga il lavoro.

Vediamo ora il problema della retroazione nei protocolli ad alta velocità. A causa del ritardo di andata e ritorno relativamente lungo, è preferibile evitare la retroazione: il ricevente impiega troppo tempo per la segnalazione al mittente. Un esempio di retroazione è il controllo della velocità di trasmissione utilizzando un protocollo sliding window. Per evitare lunghi ritardi dovuti al ricevente che invia gli aggiornamenti della finestra al mittente, è preferibile utilizzare un protocollo basato sulla velocità. In questo protocollo il mittente può inviare tutto ciò che vuole, purché non lo faccia più velocemente della frequenza consigliata in anticipo tra mittente e ricevente.

In secondo esempio di retroazione è rappresentato dall'algoritmo di avvio lento di Jacobson. Questo algoritmo crea alcune sonde per vedere quanto può reggere la rete. Nelle reti ad alta velocità, la produzione di una mezza dozzina di piccole sonde per conoscere le risposte della rete spreca enormi quantità di banda. Uno schema più efficiente prevede che mittente, il ricevente e la rete riservino le risorse necessarie al momento della impostazione della connessione. La riserva anticipata delle risorse offre il vantaggio aggiuntivo di facilitare la riduzione del jitter. In sostanza, il passaggio verso le alte velocità spinge inesorabilmente il progetto verso un funzionamento orientato alla connessione, o a qualcosa di molto simile. Naturalmente, se in futuro la banda diventerà così abbondante che a nessuno importerà di sprecarla, le regole di progettazione saranno molto diverse.

a struttura dei pacchetti ha una grande importanza nelle reti gigabit. L'intestazione ovrebbe contenere il minor numero di campi possibile, per ridurre il tempo di elaborazione; questi campi dovrebbero essere abbastanza grandi per svolgere il lavoro e allineati alle word per facilitarne l'elaborazione. In questo contesto, "abbastanza grande" indica che non si verificano più problemi come il riutilizzo dei numeri di sequenza quando sono ancora in circolazione vecchi pacchetti, l'incapacità del ricevente di comunicare lo spazio della finestra perché il relativo campo è troppo piccolo e così via.

checksum dell'intestazione e dei dati dovrebbe essere calcolato separatamente, per due motivi. Il primo è che conviene dare la possibilità di calcolare il checksum dell'intestazione ma non dei dati. Il secondo è per consentire la verifica dell'esattezza dell'intesta-

zione prima di copiare i dati nello spazio utente. È vantaggioso calcolare il checksum dei dati nel momento in cui sono copiati nello spazio utente, ma se l'intestazione non è corretta la copia potrebbe andare al processo sbagliato. Per evitare una copia non corretta e consentire il calcolo del checksum durante la copia, è essenziale che i due checksum siano separati.

La dimensione massima dei dati dovrebbe essere elevata, per consentire operazioni efficienti anche in presenza di forti ritardi. Inoltre, più grande è il blocco dei dati, più piccola è la porzione della banda totale dedicata alle intestazioni. 1.500 byte è un valore troppo piccolo.

Un'altra caratteristica importante è la capacità d'inviare una quantità normale di dati insieme alla richiesta di connessione. In questo modo, è possibile risparmiare il tempo di un round-trip.

Per finire, spendiamo qualche parola sul software del protocollo. È importante concentrarsi sull'evento normale. Molti vecchi protocolli tendono a enfatizzare le operazioni da svolgere se qualcosa va storto (per esempio se viene perso un pacchetto). Per eseguire più velocemente il protocollo, il progettista dovrebbe mirare a ridurre il tempo di elaborazione quando va tutto bene. La riduzione del tempo di elaborazione in presenza di un errore è secondaria.

Una seconda questione relativa al software è la riduzione del tempo di copia. Come affermato in precedenza, la copia dei dati è spesso la fonte principale dell'overhead. Teoricamente l'hardware dovrebbe scaricare ogni pacchetto in ingresso direttamente nella memoria come un blocco contiguo di dati. Il software dovrebbe poi copiare questo pacchetto nel buffer dell'utente con una singola operazione; a seconda del funzionamento della cache, potrebbe persino essere più conveniente evitare un ciclo di copia. In altri termini, per copiare 1.024 parole, il modo più veloce potrebbe essere l'esecuzione una dentro l'altra di 1.024 istruzioni MOVE (o 1.024 coppie caricamento/archiviazione). La routine di copia è così importante che dovrebbe essere scritta a mano in linguaggio assembly, se non è possibile trovare un modo per obbligare il compilatore a produrre il codice ottimale in modo preciso.

6.7 Sommario

Lo strato trasporto è la chiave per capire i protocolli a strati. Fornisce molti servizi, il più importante dei quali è un flusso di byte affidabile, end-to-end e orientato alla connessione dal mittente al ricevente. È possibile accedervi con primitive di servizio che consentono la costituzione, l'utilizzo e il rilascio delle connessioni. Una nota interfaccia per lo strato trasporto è quella fornita dai socket Berkeley.

I protocolli di trasporto devono essere in grado di gestire la connessione sulle reti inaffidabili. La costituzione della connessione è complicata dall'esistenza di pacchetti duplicati e ritardati che possono ricomparire in momenti inopportuni. Per gestirli, servono handshake a tre vie per stabilire le connessioni. Rilasciare una connessione è più facile che stabilirla, ma comunque l'operazione non è semplice a causa del problema dei due eserciti.

Anche quando lo strato network è completamente affidabile, lo strato trasporto ha molto

avoro da svolgere. Deve gestire tutte le primitive di servizio, gestire le connessioni e imer, allocare e utilizzare i crediti.

Internet possiede due protocolli di trasporto principali: UDP e TCP. UDP è un protocollo senza connessione che fondamentalmente un involucro per i pacchetti IP, con la funzionalità aggiuntiva del multiplexing e del demultiplexing di più processi su un singolo indirizzo IP. UDP può essere utilizzato per le interazioni client/server, per esempio utilizzando RPC. Può anche essere utilizzato per costruire protocolli in tempo reale come RTP.

Il protocollo di trasporto Internet principale è TCP. Fornisce un flusso di byte bidirezionale e affidabile. Utilizza un'intestazione di 20 byte su tutti i segmenti. I segmenti possono essere rammentati dai router all'interno di Internet, pertanto gli host devono essere in grado di eseguire il riassemblaggio. Molto lavoro è stato dedicato all'ottimizzazione delle prestazioni di TCP, utilizzando gli algoritmi di Nagle, Clark, Jacobson, Karn e altri. I collegamenti wireless giungono diverse complicazioni a TCP. TCP transazionale è un'estensione di TCP che estisce le interazioni client/server con un numero ridotto di pacchetti.

Le prestazioni della rete sono generalmente dominate dagli overhead del protocollo e di elaborazione della TPDU: questa situazione può peggiorare alle alte velocità. I protocolli ovrebbero essere progettati per ridurre il numero di TPDU, quello dei cambiamenti di contesto e delle copie delle TPDU. Nelle reti gigabit si devono usare protocolli semplici.

Problemi

Nelle primitive di trasporto dell'esempio della Figura 6.2, LISTEN è una chiamata bloccante. È strettamente necessaria? Se no, spiegare come si potrebbe utilizzare una primitiva non bloccante. Quale vantaggio si avrebbe rispetto allo schema descritto nel testo?

Nel modello della Figura 6.4 si presume che i pacchetti possano essere persi dallo strato network e devono quindi ricevere individualmente l'acknowledgement. Supponiamo che lo strato network sia affidabile al 100% e non perda mai pacchetti. Quali eventuali modifiche sono necessarie alla Figura 6.4?

In entrambe le parti della Figura 6.6 esiste un commento che indica che il valore di SERVER_PORT deve essere lo stesso sul client e sul server. Perché è così importante?

Supponete che lo schema per la generazione dei numeri di sequenza iniziali guidato dall'orologio sia utilizzato con un contatore di orologio a 15 bit. Il battito dell'orologio avviene ogni 100 msec e la durata massima del pacchetto è 60 secondi. Con che frequenza è necessaria la risincronizzazione

a) nel caso peggiore?

b) quando i dati consumano 240 numeri di sequenza al minuto?

Perché la durata massima del pacchetto, T , deve essere abbastanza grande da assicurare che non solo il pacchetto ma anche i suoi acknowledgement siano svaniti?

Si immagini che per impostare le connessioni venga utilizzato un handshake a due vie anziché a tre. In altre parole, il terzo messaggio non è richiesto. Potrebbero verificarsi delle situazioni di stallo? Presentare un esempio o dimostrare che non ne esistono.

Si immagini un problema generalizzato di n eserciti, in cui l'accordo di due degli eserciti blu è sufficiente per la vittoria. Esiste un protocollo che consente ai blu di vincere?

8. Si consideri il problema del ripristino dai crash dell'host (Figura 6.18). Se l'intervallo tra la scrittura e l'invio di un acknowledgement (o viceversa) è reso relativamente piccolo, quali sono le due strategie mittente/ricevente migliori per ridurre le probabilità di fallimento del protocollo?
9. Sono possibili situazioni di stallo con l'entità di trasporto descritta nel testo (Figura 6.20)?
10. Per curiosità, l'implementatore dell'entità di trasporto della Figura 6.20 ha deciso di inserire alcuni contatori nella procedura *sleep* per raccogliere statistiche sull'array *conn*. Tra queste vi sono il numero di connessioni in ognuno dei sette possibili stati n_i ($i = 1, \dots, 7$). Dopo avere scritto un programma FORTRAN per analizzare i dati, l'implementatore scopre che la relazione $\sum n_i = MAX_CONN$ sembra essere sempre vera. Esistono altre costanti che coinvolgono solo queste sette variabili?
11. Che cosa accade quando l'utente dell'entità di trasporto della Figura 6.20 invia un messaggio di lunghezza zero? Spiegare il significato della propria risposta.
12. Esaminare ogni evento che si può verificare nell'entità di trasporto della Figura 6.20 per indicare se è ammesso quando l'utente è inattivo nello stato *sending*.
13. Discutere vantaggi e svantaggi dei crediti rispetto ai protocolli sliding window.
14. Perché esiste UDP? Non sarebbe stato sufficiente consentire ai processi utente di inviare pacchetti IP grezzi?
15. Si consideri un semplice protocollo a livello di applicazione costruito sopra UDP, che consente a un client di recuperare un file da un server remoto con un indirizzo ben noto. Il client invia prima una richiesta con il nome del file; il server risponde con una sequenza di pacchetti contenenti parti diverse del file richiesto. Per assicurare l'affidabilità e la consegna sequenziale, il client e il server utilizzano un protocollo stop-and-wait. Ignorando le ovvie questioni relative alle prestazioni, questo protocollo ha un problema? Pensare attentamente alle possibilità di crash dei processi.
16. Un client invia una richiesta di 128 byte a un server situato a 100 Km di distanza su una fibra ottica a 1 gigabit. Qual è l'efficienza della linea durante la chiamata a procedure remote?
17. Si consideri nuovamente la situazione del problema precedente. Calcolare il tempo minimo di risposta per la linea a 1 Gbps e per una linea a 1 Mbps. Quale conclusione è possibile trarre?
18. UDP e TCP utilizzano i numeri di porta per identificare l'entità di destinazione durante la consegna di un messaggio. Esporre due ragioni per cui questi protocolli hanno inventato un nuovo ID astratto (numeri di porta) anziché utilizzare gli ID di processo, che esistevano già nel momento in cui sono stati progettati questi protocolli.
19. Qual è la dimensione totale minima della MTU TCP, comprendendo l'overhead di TCP e IP ma non quello dello strato data link?
20. La frammentazione e il riassemblaggio dei datagrammi sono gestiti da IP e invisibili per TCP. Questo significa che TCP non si deve preoccupare dell'arrivo dei dati in ordine errato?
21. RTP è utilizzato per trasmettere audio di qualità CD, composto da una coppia di campioni di 16 bit 44100 volte al secondo, un campione per ogni canale stereo. Quanti pacchetti al secondo deve trasmettere RTP?

22. Sarebbe possibile inserire il codice RTP nel kernel del sistema operativo, insieme al codice UDP? Spiegare la risposta.
23. Un processo sull'host 1 è stato assegnato alla porta p , mentre un processo sull'host 2 è stato assegnato alla porta q . È possibile stabilire due o più connessioni TCP tra queste due porte contemporaneamente?
24. Nella Figura 6.29 abbiamo visto che, oltre al campo *Acknowledgement* di 32 bit, esiste un bit *ACK* nella quarta word. Aggiunge realmente qualcosa? Perché (o perché no)?
25. Il carico utile massimo di un segmento TCP è 65.495 byte. Perché è stato scelto questo numero strano?
26. Descrivere due modi per entrare nello stato *SYN RCVD* della Figura 6.33.
27. Indicare un potenziale svantaggio dell'utilizzo dell'algoritmo di Nagle su una rete molto congestionata.
28. Si consideri l'effetto dell'avvio lento su una linea con un tempo di round-trip pari a 10 msec e nessuna congestione. La finestra di ricezione è di 24 KB e la dimensione massima del segmento è 2 KB. Quanto tempo è necessario prima che sia possibile inviare la prima finestra completa?
29. Si supponga che la finestra di congestione TCP sia impostata a 18 KB e che si verifichi un timeout. Quanto sarà grande la finestra se le quattro trasmissioni successive hanno successo? Presumere che la dimensione massima del segmento sia 1 KB.
30. Se il tempo di round-trip TCP, *RTT*, è attualmente 30 msec e i seguenti acknowledgement vengono ricevuti rispettivamente in 26, 32 e 24 msec, qual è il nuovo *RTT* stimato utilizzando l'algoritmo di Jacobson? Utilizzare $\alpha = 0,9$.
31. Una macchina TCP sta inviando finestre complete di 65.535 byte su un canale a 1 Gbps con un ritardo unidirezionale di 10 msec. Qual è la massima velocità dei dati raggiungibile? Qual è l'efficienza della linea?
32. Qual è la velocità massima della linea a cui un host può inviare carichi utili TCP di 1.500 byte con una durata massima del pacchetto di 120 secondi, senza dover riutilizzare i numeri di sequenza? Tenere conto dell'overhead TCP, IP ed Ethernet. Presumere che i frame Ethernet possano essere inviati continuamente.
33. In una rete con una dimensione massima della TPDU di 128 byte, una durata massima della TPDU di 30 secondi e un numero di sequenza di 8 bit, qual è la velocità massima dei dati per connessione?
34. Si supponga di misurare il tempo necessario per ricevere una TPDU. Quando avviene un interrupt, viene letto l'orologio di sistema in millisecondi; dopo l'elaborazione della TPDU, l'orologio viene letto nuovamente. Misurate 0 msec per 270.000 volte e 1 msec per 730.000 volte. Quanto tempo è necessario per ricevere una TPDU?
35. Una CPU esegue istruzioni alla velocità di 1.000 MIPS. I dati possono essere copiati 64 bit alla volta, e la copia di ogni word richiede 10 istruzioni. Se un pacchetto in ingresso deve essere copiato quattro volte, questo sistema può gestire una linea a 1 Gbps? Per semplicità, presumere che tutte le istruzioni, anche quelle che leggono o scrivono nella memoria, siano eseguite alla velocità piena di 1.000 MIPS.

36. Per aggirare il problema del riutilizzo dei numeri di sequenza quando esistono ancora vecchi pacchetti, si potrebbero impiegare numeri di sequenza di 64 bit. Tuttavia, teoricamente una fibra ottica può lavorare a 75 Tbps. Quale durata massima del pacchetto è necessaria per assicurare che le future reti a 75 Tbps non incontrino il suddetto problema anche con i numeri di sequenza a 64 bit? Presumere che ogni byte possieda il proprio numero di sequenza, come in TCP.
37. Indicare un vantaggio di RPC su UDP rispetto a TCP transazionale. Indicare un vantaggio di T/TCP rispetto a RPC.
38. Nella Figura 6.40(a) abbiamo visto che sono necessari 9 pacchetti per completare RPC. Vi sono circostanze in cui sono necessari esattamente 10 pacchetti?
39. Nel paragrafo 6.6.5, abbiamo calcolato che una linea gigabit scarica 80.000 pacchetti al secondo sull'host, lasciandogli solo 6.250 istruzioni per l'elaborazione se metà del tempo della CPU è riservato alle applicazioni. Questo calcolo presumeva un pacchetto di 1.500 byte. Rieseguire il calcolo per un pacchetto con la dimensione per ARPANET (128 byte). In entrambi i casi, presumere che questa dimensione del pacchetto includa tutti gli overhead.
40. Per una rete a 1 Gbps operante su 4.000 Km il fattore limitante è il ritardo, non la banda. Si consideri una MAN con l'origine e la destinazione distanti di 20 Km. A quale velocità dei dati il ritardo di round-trip dovuto alla velocità della luce equivale al ritardo di trasmissione per un pacchetto di 1 KB?
41. Calcolare il prodotto banda-ritardo per le seguenti reti: (1) T1 (1,5 Mbps), (2) Ethernet (10 Mbps), (3) T3 (45 Mbps), (4) STS-3 (155 Mbps). Presumere un RTT pari a 100 msec. Ricordare che l'intestazione TCP riserva 16 bit alla dimensione della finestra. Quali sono le implicazioni di questo fatto alla luce dei calcoli?
42. Qual è il prodotto banda-ritardo per un canale di 50 Mbps su un satellite geostazionario? Se i pacchetti sono tutti di 1.500 byte (compreso l'overhead), quanto deve essere grande la finestra nei pacchetti?
43. Il file server della Figura 6.6 non è perfetto, pertanto si possono apportare alcuni miglioramenti. Eseguire le seguenti modifiche:
fornire al client un terzo argomento che specifica un intervallo di byte.
aggiungere un flag -W al client che consente la scrittura del file sul server.
44. Modificare il programma della Figura 6.20 per eseguire il ripristino dagli errori. Aggiungere un nuovo tipo di pacchetto, *reset*, che può arrivare dopo che una connessione è stata aperta da entrambi i lati, ma non è stata chiusa da nessuno. Questo evento, che si verifica contemporaneamente a entrambe le estremità della connessione, indica che ogni pacchetto in transito è stato consegnato o distrutto, ma in ogni caso non si trova più nella sottorete.
45. Scrivere un programma che simuli la gestione del buffer in un'entità di trasporto, utilizzando una sliding window per il controllo di flusso anziché il sistema dei crediti della Figura 6.20. Consentire ai processi dello strato superiore di aprire le connessioni, inviare i dati e chiudere le connessioni. Per semplicità, tutti i dati viaggiano dalla macchina A alla macchina B, e non viceversa. Sperimentare diverse strategie di allocazione del buffer in B, per esempio dedicando i buffer a specifiche connessioni o utilizzando un pool di buffer comune, e misurare la velocità totale raggiunta da ogni strategia.

6. Progettare e implementare un sistema di messaggistica (chat) che permette a più gruppi di utenti di conversare. Il coordinatore della chat risiede a un indirizzo di rete ben noto, utilizza UDP per la comunicazione con i client, imposta i server per ogni sessione di conversazione e mantiene una directory delle sessioni. Esiste un server di conversazione per ciascuna sessione, e utilizza TCP per la comunicazione con i client. Un client consente agli utenti di creare, unirsi e abbandonare una sessione di conversazione. Progettare e implementare il codice per il coordinatore, il server e il client.

7

Lo strato applicazione

Terminati i preliminari, si può passare allo strato dove si trovano tutte le applicazioni. Gli strati sotto lo strato applicazione forniscono il trasporto affidabile, ma non svolgono alcun lavoro per gli utenti. In questo capitolo studieremo alcune applicazioni concrete delle reti. Tuttavia, anche nello strato applicazione c'è l'esigenza dei protocolli di supporto che permettono alle applicazioni di funzionare. Per questo motivo studieremo uno di questi protocolli prima di iniziare l'esame delle applicazioni vere e proprie: DNS, che gestisce i nomi all'interno di Internet. Successivamente esamineremo tre applicazioni reali: posta elettronica, World Wide Web e multimedia.

7.1 DNS: il sistema dei nomi di dominio

Anche se i programmi possono teoricamente fare riferimento a host, caselle di posta e altre risorse tramite i loro indirizzi di rete (per esempio IP), le persone ricordano con difficoltà questi indirizzi. Inoltre, inviare un messaggio di posta elettronica a *tana@128.111.24.41* significa che se l'ISP o l'organizzazione di Tana sposta il server di posta su una macchina con un indirizzo IP diverso, anche l'indirizzo di posta elettronica deve cambiare. Per questi motivi sono stati introdotti nomi ASCII per separare i nomi dei computer dai rispettivi indirizzi, e in questo modo l'indirizzo di Tana potrebbe essere simile a *tana@art.ucsb.edu*. Ciò nonostante, la rete in sé sa interpretare solo indirizzi numerici, pertanto sono richiesti meccanismi per convertire le stringhe ASCII in indirizzi di rete. Nei paragrafi successivi studieremo come viene realizzata su Internet questa associazione.

tempo di ARPANET esisteva solamente un file, *host.txt*, che elencava tutti gli host e i loro indirizzi IP. Ogni notte, tutti gli host lo prelevavano dal sito in cui era mantenuto. Per una rete composta da poche centinaia di grandi computer che lavoravano in timesharing questo approccio funzionava abbastanza bene.

Tuttavia, quando alla rete furono connesse centinaia di minicomputer e PC, tutti capirono che questo metodo non poteva funzionare per sempre. Da una parte, la dimensione del file avrebbe diventato troppo elevata. Inoltre c'era il rischio (decisamente più grave) di continui conflitti tra i nomi degli host, se questi non fossero stati gestiti centralmente; ma ciò è pensabile in una vasta rete internazionale, a causa del carico e della latenza. Per risolvere questi problemi fu inventato DNS (*Domain Name System*, sistema dei nomi di dominio).

L'essenza di DNS è l'invenzione di uno schema di denominazione gerarchico basato su dominio, e di un sistema di database distribuito per l'implementazione di questo schema di denominazione. È principalmente utilizzato per associare nomi di host e destinazioni di posta elettronica agli indirizzi IP, ma può essere utilizzato anche per altri scopi. DNS è definito in RFC 1034 e 1035.

In breve, DNS viene utilizzato come segue. Per associare un nome a un indirizzo IP, un programma applicativo chiama una procedura di libreria chiamata **risolutore**, passando il nome come parametro. Abbiamo visto un esempio di risolutore, *gethostbyname*, nella Figura 6.6. Il risolutore invia un pacchetto UDP a un server DNS locale, che quindi cerca il nome e restituisce l'indirizzo IP al risolutore, che a sua volta lo restituisce al chiamante. Armato dell'indirizzo IP, il programma può quindi stabilire una connessione TCP con destinazione oppure inviarle pacchetti UDP.

7.1 Lo spazio dei nomi DNS

La gestione di un insieme di nomi grande e in continuo cambiamento non è un problema poco. Nel sistema postale, la gestione dei nomi viene svolta imponendo che sulle lettere siano specificate (in modo implicito o esplicito) la nazione, lo stato o la provincia, la città e la via del destinatario. Utilizzando questo tipo di indirizzo gerarchico non è possibile fare confusione tra Marvin Anderson abitante in Main St. a White Plains, N.Y., e Marvin Anderson residente in Main St. ad Austin, Texas. DNS funziona allo stesso modo. Concettualmente, Internet è divisa in oltre 200 **domini** di primo livello, dove ogni dominio copre molti host. Ogni dominio è partitionato in sottodomini, che sono a loro volta sottosussidiari e così via. Tutti questi domini possono essere rappresentati con una struttura ad albero, come mostrato nella Figura 7.1. Le foglie rappresentano i domini che non hanno sottodomini (ma naturalmente contengono computer). Un dominio foglia può contenere un solo host, o può rappresentare una società e contenere migliaia di host.

I domini di primo livello sono di due tipi: generici e per nazioni. I domini generici originali erano *com* (commerciale), *edu* (istituzioni educative), *gov* (governo federale degli

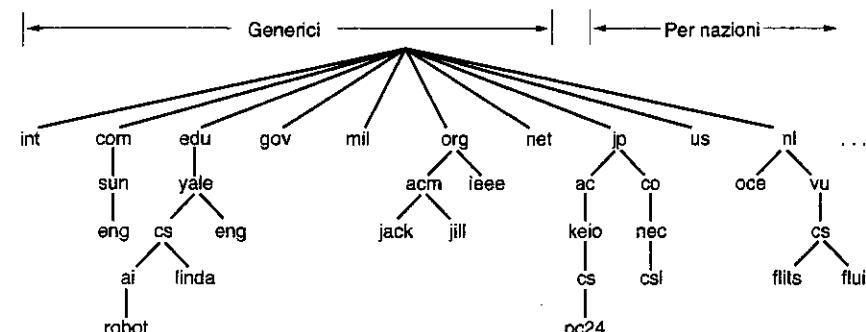


Figura 7.1. Una porzione dello spazio dei nomi di dominio Internet.

Stati Uniti), *int* (organizzazioni internazionali selezionate), *mil* (forze armate statunitensi), *net* (provider di rete) e *org* (organizzazioni senza scopo di lucro). I domini per nazioni comprendono una voce per ogni nazione, come definito in ISO 3166.

Nel novembre del 2000, ICANN ha approvato quattro nuovi domini di primo livello generici, vale a dire *biz* (business), *info* (informazioni), *name* (nomi delle persone) e *pro* (professioni, come medici e avvocati). Inoltre, su richiesta di alcune industrie sono stati introdotti tre domini di primo livello più specializzati. Si tratta di *aero* (industria aerospaziale), *coop* (cooperative) e *museum* (musei). Altri domini di primo livello saranno aggiunti in futuro.

Internet sta diventando sempre più commerciale, ma anche più controversa. Si prenda *pro*, per esempio. È pensato per i professionisti certificati. Ma chi sono i professionisti? E certificati da chi? Medici e avvocati sono chiaramente professionisti. Ma cosa dire di fotografi freelance, insegnanti di pianoforte, maghi, idraulici, partucchieri, disinfestatori, tatuatori, mercenari e prostitute? Queste sono occupazioni professionali e quindi idonee per i domini *pro*? E se così fosse, chi certifica i singoli praticanti?

In generale, ottenere un dominio di secondo livello come *nomesocietà.com* è più facile. È necessario solamente contattare un *registrar* (autorità di registrazione dei domini) per il dominio di primo livello corrispondente (*com* in questo caso), per controllare se il nome desiderato è disponibile e se altri non possiedono il marchio. Se non vi sono problemi, il richiedente paga una tariffa annuale e ottiene il nome. Per ora, quasi ogni parola inglese è stata utilizzata per il dominio *com*. Si possono provare articoli, animali, piante, parti del corpo: quasi tutto è già stato utilizzato.

Ogni dominio è denominato dal percorso compreso tra esso e la radice (non denominata). I componenti sono separati da punti (a volte pronunciati "dot"), di conseguenza il reparto ingegneristico di Sun Microsystems potrebbe utilizzare *eng.sun.com.*, piuttosto che un nome in stile UNIX come */com/sun/eng*. Occorre notare che questa denominazione gerarchica garantisce che *eng.sun.com.* non è in conflitto con un potenziale utilizzo di *eng* in

eng.yale.edu., che potrebbe essere utilizzato dal dipartimento di lingua inglese di Yale. I nomi di dominio possono essere assoluti o relativi. Un nome di dominio assoluto termina sempre con un punto (per esempio *eng.sun.com.*), al contrario di un nome relativo. I nomi relativi devono essere interpretati in un contesto per determinarne univocamente il reale significato. In entrambi i casi, il dominio fa riferimento a un nodo specifico nella struttura e a tutti i nodi sottostanti.

I nomi di dominio sono *case insensitive*, pertanto *edu*, *Edu* ed *EDU* indicano la stessa cosa. I componenti del nome possono essere lunghi fino a 63 caratteri, mentre il percorso completo non deve superare 255 caratteri.

In teoria i domini possono essere inseriti nella struttura in due modi diversi. Per esempio, *cs.yale.edu* poteva essere elencato sotto il dominio per nazioni *us* come *.cs.yale.ct.us*. In pratica, però, la maggior parte delle organizzazioni negli Stati Uniti utilizza un nome generico, mentre la maggior parte di quelle esterne agli Stati Uniti sceglie il dominio della sua nazione. Non esistono regole per la registrazione in due domini di primo livello, ma ben poche società lo fanno (a eccezione delle multinazionali, per esempio *sony.com* e *sony.nl*).

Ogni dominio controlla come allocare i domini sottostanti. Per esempio, il Giappone possiede i domini *ac.jp* e *co.jp* che rispecchiano *edu* e *com*. L'Olanda non fa questa distinzione e pone tutte le organizzazioni direttamente sotto *nl*. Di conseguenza, i tre nomi seguenti rappresentano dipartimenti di informatica delle rispettive università:

1. *cs.yale.edu* (Yale University, Stati Uniti)
2. *cs.vu.nl* (Vrije Universiteit, Olanda)
3. *cs.keio.ac.jp* (Keio University, Giappone).

Per creare un nuovo dominio è necessaria l'autorizzazione del dominio in cui sarà incluso. Per esempio, se a Yale viene avviato un gruppo VLSI che desidera essere conosciuto come *vlsi.cs.yale.edu*, deve richiedere l'autorizzazione a chi gestisce *cs.yale.edu*. Allo stesso modo, se viene concesso lo statuto a una nuova università, per esempio University of Northern South Dakota, questa deve chiedere al gestore del dominio *edu* di assegnarle *nsd.edu*. In questo modo, i conflitti tra i nomi vengono evitati e ogni dominio può tenere traccia di tutti i suoi sottodomini. Una volta creato e registrato un nuovo dominio, questo può creare sottodomini, come *cs.unsd.edu*, senza bisogno di ottenere l'autorizzazione delle parti della struttura che si trovano al di sopra.

a denominazione segue i confini dell'organizzazione, non le reti fisiche. Per esempio, se reparti di informatica e ingegneria elettronica sono situati nello stesso edificio e condividono la stessa LAN, possono comunque utilizzare domini distinti. Allo stesso modo, se il dipartimento di informatica viene diviso in Babbage Hall e Turing Hall, gli host in entrambi gli edifici apparterranno di norma allo stesso dominio.

1.2 I record delle risorse

Ogni dominio, che sia rappresentato da un singolo host o sia un dominio di primo livello, può essere associato un insieme di **resource records** (record delle risorse).

Per un singolo host, il record delle risorse più comune è solo l'indirizzo IP, ma possono esistere molti altri tipi di record delle risorse. Quando un risolutore fornisce un nome di dominio a DNS, ciò che ottiene sono i record delle risorse associati a tale nome. Di conseguenza, la funzione principale di DNS è associare i nomi di dominio ai record delle risorse.

Un record delle risorse è una quintupla. Anche se sono codificati in binario per efficienza, nella maggior parte delle esposizioni i record delle risorse sono presentati come testo ASCII, una riga per ciascun record delle risorse. Il formato che useremo è il seguente:

Domain_name Time_to_live Class Type Value

Domain_name indica il dominio a cui si riferisce il record. Normalmente esistono molti record per ogni dominio, e ogni copia del database contiene informazioni su più domini. Questo campo è pertanto la prima chiave di ricerca utilizzata per soddisfare le interrogazioni. L'ordine dei record nel database non è significativo.

Il campo *time_to_live* offre un indicatore della stabilità del record. Alle informazioni altamente stabili viene assegnato un valore elevato, come 86.400 (il numero di secondi in un giorno). Alle informazioni di breve durata viene invece assegnato un valore piccolo, come 60 (1 minuto). Torneremo su questo punto in seguito quando parleremo del caching.

Il terzo campo di ogni record delle risorse è *class*. Per le informazioni Internet è sempre *IN*; per altre informazioni non Internet possono essere utilizzati altri codici, che in pratica però sono utilizzati raramente.

Il campo *type* specifica il tipo di record. I tipi più importanti sono elencati nella Figura 7.2.

Tipo	Nome	Significato	Valore
SOA	Start of Authority	Fonction dell'autorità	I parametri per questa zona
A	IP address of a host	Indirizzo IP di un host	Intero a 32 bit
MX	Mail exchange	Scambio di posta	La priorità e il nome con cui il dominio desidera accettare la posta elettronica
NS	Name Server	Server dei nomi	Il nome del server per questo dominio
CNAME	Canonical name	Nome canonico	Il nome di dominio
PTR	Pointer	Puntatore	L'alias per un indirizzo IP
HINFO	Host description	Descrizione dell'host	La CPU e l'OS in ASCII
TXT	Text	Testo	Testo ASCII non interpretato

Figura 7.2. I principali tipi di record delle risorse DNS per IPv4.

Un record SOA fornisce il nome della fonte primaria di informazioni sulla zona del server dei nomi (descritta in seguito), l'indirizzo di posta elettronica del suo amministratore, un numero di serie univoco e diversi flag e timeout.

Il tipo di record più importante è il record *A (Address)*. Contiene l'indirizzo IP a 32 bit

di un qualche host. Ogni host Internet deve avere almeno un indirizzo IP affinché le altre macchine possano comunicare con esso. Alcuni host presentano due o più connessioni di rete, e in questo caso possiedono un record delle risorse di tipo *A* per ciascuna connessione di rete (e quindi per indirizzo IP). DNS può essere configurato per eseguire un ciclo tra essi, restituendo il primo record alla prima richiesta, il secondo record alla seconda richiesta e così via.

Un altro tipo di record importante è il record *MX*. Specifica il nome dell'host configurato per accettare la posta elettronica per il dominio specificato. È utilizzato perché non tutte le macchine sono capaci di accettare la posta elettronica. Se qualcuno desidera inviare messaggi a *bill@microsoft.com*, per esempio, l'host di invio deve trovare un server di posta in *microsoft.com* che desidera accettare la posta elettronica. Il record *MX* può fornire queste informazioni.

Il record *NS* specifica i server dei nomi. Per esempio, ogni database DNS possiede normalmente un record *NS* per ogni dominio di primo livello, quindi la posta elettronica può essere inviata anche a parti distanti della struttura di denominazione. Torneremo in seguito su questo punto.

I record *CNAME* consentono la creazione di alias. Per esempio, una persona che ha acquistato familiarità con i nomi Internet in generale e desidera inviare un messaggio a qualcuno il cui nome di login è *paul* nel dipartimento di informatica del MIT, potrebbe eseguire un tentativo con *paul@cs.mit.edu*. In realtà, questo indirizzo non funzionerà perché il dominio del dipartimento di informatica di MIT è *lcs.mit.edu*. Tuttavia, come servizio per le persone che non lo sanno, MIT potrebbe creare una voce *CNAME* che diriga persone e programmi sulla strada corretta. Una voce come la seguente potrebbe occuparsi della questione:

```
cs.mit.edu 86400 IN CNAME lcs.mit.edu
```

Come *CNAME*, *PTR* punta verso un altro nome. Tuttavia, a differenza di *CNAME* che è soltanto una definizione macro, *PTR* è un tipo di dati DNS regolare la cui interpretazione dipende dal contesto in cui viene trovata. In pratica, è quasi sempre utilizzato per associare un nome a un indirizzo IP per consentire le ricerche che forniscono gli indirizzi IP e restituiscono i nomi delle macchine corrispondenti. Sono chiamate **reverse lookups** (ricerche inverse).

I record *HINFO* consentono alle persone di scoprire a quale tipo di macchina e sistema operativo corrisponde un dominio. Per finire, i record *TXT* consentono ai domini di identificare sé stessi in modo arbitrario. Entrambi i tipi di record sono disponibili per convenienza dell'utente. Non sono obbligatori, pertanto i programmi non possono contare sulla loro presenza (e probabilmente non li sanno neppure usare).

Per finire è disponibile il campo *value*, che può essere un numero, un nome di dominio o una stringa ASCII. La semantica dipende dal tipo di record. Una breve descrizione dei campi *value* per ognuno dei tipi principali di record è data nella Figura 7.2.

Per un esempio del tipo di informazioni presenti nel database DNS di un dominio, osservare la Figura 7.3. Questa figura mostra parte di un database semi-ipotetico per il dominio *cs.vu.nl* mostrato nella Figura 7.1. Il database contiene sette tipi di record delle risorse.

; Dati autorevoli per cs.vu.nl				
cs.vu.nl.	86400	IN	SOA	star boss (9527,7200,7200,241920,86400)
cs.vu.nl.	86400	IN	TXT	"Divisie Wiskunde en Informatica."
cs.vu.nl.	86400	IN	TXT	"Vrije Universiteit Amsterdam."
cs.vu.nl.	86400	IN	MX	1 zephyr.cs.vu.nl.
cs.vu.nl.	86400	IN	MX	2 top.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	HINFO	Sun Unix
flits.cs.vu.nl.	86400	IN	A	130.37.16.112
flits.cs.vu.nl.	86400	IN	A	192.31.231.165
flits.cs.vu.nl.	86400	IN	MX	1 flits.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	MX	2 zephyr.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	MX	3 top.cs.vu.nl.
www.cs.vu.nl.	86400	IN	CNAME	star.cs.vu.nl
ftp.cs.vu.nl.	86400	IN	CNAME	zephyr.cs.vu.nl
rowboat		IN	A	130.37.56.201
		IN	MX	1 rowboat
		IN	MX	2 zephyr
		IN	HINFO	Sun Unix
little-sister		IN	A	130.37.62.23
		IN	HINFO	Mac MacOS
laserjet		IN	A	192.31.231.216
		IN	HINFO	"HP Laserjet IISi" Proprietary

Figura 7.3. Una porzione di un possibile database DNS per *cs.vu.nl*.

La prima riga non commentata della Figura 7.3 offre alcune informazioni di base sul dominio, che non ci riguarderanno in futuro. Le due righe successive offrono informazioni di testo sull'ubicazione del dominio. Dopo di che appaiono due voci che comunicano il primo e il secondo luogo dove tentare di consegnare la posta elettronica inviata a *persona@cs.vu.nl*. Per prima cosa è opportuno provare *zephyr* (una macchina specifica); in caso di fallimento, si dovrebbe provare *top*. Dopo la riga vuota, aggiunta a fini di leggibilità, appaiono le righe che indicano che *flits* è una workstation Sun che esegue Unix, e comunicano entrambi i suoi indirizzi IP. Sono quindi offerte tre scelte per la gestione della posta elettronica inviata a *flits.cs.vu.nl*. La prima scelta è naturalmente *flits* stesso, ma se non fosse disponibile si possono provare *zephyr* e *top*. Successivamente appare un alias, *www.cs.vu.nl*, che consente di utilizzare questo indirizzo senza designare una macchina specifica. La creazione di questo alias consente a *cs.vu.nl* di cambiare il suo server World Wide Web senza invalidare gli indirizzi utilizzati dalle persone per raggiungerlo. Un argomento simile è fornito per *ftp.cs.vu.nl*. Le quattro righe successive contengono una voce tipica per una workstation, in questo caso *rowboat.cs.vu.nl*. Le informazioni fornite contengono l'indirizzo IP, i luoghi predefiniti per la posta (primario e secondario) e alcune informazioni sulla macchina. Appare quindi una voce per un sistema non UNIX che non è in grado di ricevere la posta autonomamente [NdR - la semplice assenza di un record di tipo MX associato al nome di un host non significa neces-

sariamente che l'host non è in grado di ricevere posta autonomamente un messaggio indirizzato a persona@little-sister.cs.vu.nl potrebbe quindi essere ricevuto correttamente dall'host], seguita da una voce per una stampante laser connessa a Internet. Quello che non è mostrato (e non si trova in questo file) sono gli indirizzi IP utilizzati per cercare i domini di primo livello. Sono necessari per cercare host distanti, ma dal momento che non sono parte del dominio cs.vu.nl, non sono contenuti in questo file. Sono forniti dai root server, i cui indirizzi IP sono presenti in un file di configurazione del sistema e sono caricati nella cache DNS all'avvio del server DNS. Esistono una decina di root server sparsi per il mondo, ognuno dei quali conosce gli indirizzi IP di tutti i server di dominio di primo livello. Di conseguenza, se una macchina conosce l'indirizzo IP di almeno un root server, può cercare qualsiasi nome DNS.

7.1.3 I server dei nomi

In teoria, un singolo server dei nomi potrebbe contenere l'intero database DNS e rispondere a tutte le interrogazioni. In pratica, questo server sarebbe troppo sovraccaricato per essere utile. Inoltre, se dovesse incontrare un problema, l'intera Internet sarebbe bloccata. Per evitare i problemi associati alla disponibilità di una singola fonte di informazioni, lo spazio dei nomi DNS viene diviso in **zone** non sovrapposte. Un modo per dividere lo spazio dei nomi della Figura 7.1 è mostrato nella Figura 7.4. Ogni zona contiene alcune parti della struttura, e contiene anche i server dei nomi con le informazioni su tale zona. Di norma, una zona avrà un server dei nomi primario, che ottiene le sue informazioni da un file su disco, e uno o più server dei nomi secondari, che ottengono le loro informazioni dal

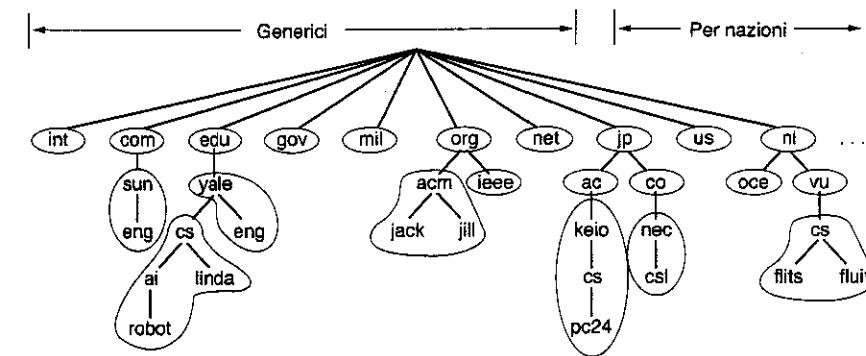


Figura 7.4. Una parte dello spazio dei nomi DNS che mostra la divisione in zone.

server dei nomi primario. Per migliorare l'affidabilità, alcuni server per una zona possono essere situati all'esterno della zona.

La posizione dei confini di zona all'interno di una zona è una scelta lasciata all'amministratore. La decisione viene presa in gran parte sulla base del numero di server dei nomi desiderati e della loro posizione. Per esempio, nella Figura 7.4, Yale ha un server per yale.edu che gestisce eng.yale.edu ma non cs.yale.edu, che è una zona separata con i suoi server dei nomi. Tale decisione può essere presa quando un dipartimento come

"Inglese" non desidera implementare il suo server dei nomi, mentre un dipartimento come "Informatica" preferisce farlo. Di conseguenza, cs.yale.edu è una zona separata ma eng.yale.edu non lo è.

Quando un risolutore ha un'interrogazione su un nome di dominio, passa l'interrogazione a uno dei server dei nomi locali. Se il dominio è all'interno della giurisdizione del server dei nomi (come ai.cs.yale.edu che ricade in cs.yale.edu), restituisce i record autorevoli delle risorse. Un **record autorevole** è un record fornito dall'autorità che gestisce il record, ed è pertanto sempre corretto. I record autorevoli si contrappongono ai record archiviati nella cache, che potrebbero non essere aggiornati.

Se invece il dominio è remoto e non sono disponibili localmente informazioni su di esso, il server dei nomi invia un messaggio d'interrogazione al server dei nomi di primo livello per il dominio richiesto. Per chiarire il processo, si consideri l'esempio nella Figura 7.5. Un risolutore su flits.cs.vu.nl desidera conoscere l'indirizzo IP dell'host linda.cs.yale.edu. Nel passaggio 1, invia un'interrogazione al server dei nomi locale, cs.vu.nl. Questa interrogazione contiene il nome di dominio, il tipo (A) e la classe (IN).

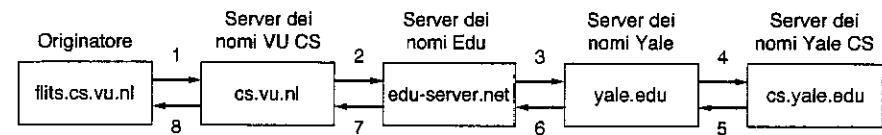


Figura 7.5. La ricerca di un nome remoto in otto passaggi, da parte di un risolutore.

Si supponga che il server dei nomi locale non abbia mai fatto un'interrogazione su questo dominio e di conseguenza non conosca nulla di esso. Potrebbe chiedere ad altri server dei nomi vicini, ma se nessuno di essi possiede informazioni deve inviare un pacchetto UDP al server per edu indicato nel suo database (Figura 7.5), edu-server.net. È improbabile che questo server conosca l'indirizzo di linda.cs.yale.edu, e probabilmente non conosce nemmeno cs.yale.edu, ma deve conoscere tutti i suoi figli, pertanto inoltra la richiesta al server dei nomi per yale.edu (passaggio 3).

A sua volta, questo inoltra la richiesta a cs.yale.edu (passaggio 4), che deve disporre dei record autorevoli delle risorse. Visto che ogni richiesta proviene dal client per un server, il record delle risorse richiesto percorre la sua strada tornando indietro nei passaggi da 5 a 8.

Quando i record vengono consegnati al server dei nomi cs.vu.nl sono inseriti in una cache locale, nel caso fossero necessari in seguito. Tuttavia, queste informazioni non sono autorevoli, perché le modifiche apportate a cs.yale.edu non saranno propagate in tutte le cache del mondo conosciute. Per questo motivo, le voci nella cache non dovrebbero durare troppo a lungo. Questo è il motivo per cui il campo *time_to_live* è stato incluso in ogni record delle risorse. Comunica ai server dei nomi remoti il tempo per cui archiviare i record nella cache. Se una macchina ha mantenuto lo stesso indirizzo IP per

anni, potrebbe essere sicuro archiviare l'informazione per 1 giorno. Per informazioni di breve durata, è più sicuro eliminare i record dopo pochi secondi o dopo un minuto.

Vale la pena ricordare che il metodo di interrogazione descritto è conosciuto come **interrogazione ricorsiva**, perché ogni server che non possiede le informazioni richieste deve trovarle altrove e poi restituirlle. È anche possibile utilizzare una forma alternativa. In questa forma, quando un'interrogazione non può essere soddisfatta localmente, l'interrogazione fallisce ma viene restituito il nome del successivo server da provare a usare. Alcuni server non implementano le interrogazioni ricorsive, e restituiscono sempre il nome del server successivo. Bisogna anche evidenziare che quando un client DNS non riesce a ottenere una risposta prima della scadenza del suo timer, la volta successiva proverà di norma un altro server. Suppone infatti che il server non sia attivo, e non che il problema sia dovuto alla perdita della richiesta o della risposta.

Anche se DNS è molto importante per il corretto funzionamento di Internet, il suo unico compito è associare nomi simbolici per le macchine ai loro indirizzi IP. Non aiuta a individuare persone, risorse, servizi e oggetti in generale. Per queste operazioni è stato definito un altro servizio directory, chiamato **LDAP** (*Lightweight Directory Access Protocol*). È una versione semplificata del servizio directory OSI X.500 ed è descritta in RFC 2251. Organizza le informazioni come una struttura ad albero e supporta la ricerca per diversi elementi. Può essere considerato come una sorta di rubrica telefonica o "pagine gialle". Non sarà presentato in questo libro, ma per ulteriori informazioni è possibile consultare (Weltman e Dahbura, 2000).

7.2 La posta elettronica

La posta elettronica, o **e-mail**, come è nota a molti suoi fan, è disponibile da circa due decenni. Prima del 1990 era utilizzata principalmente nelle università. Negli anni '90 è divenuta nota al grande pubblico, ed è cresciuta in modo esponenziale: basti pensare che il numero di e-mail attualmente spedite ogni giorno è di molto superiore al numero di lettere tradizionali (chiamate anche **snail mail**).

La posta elettronica, come molte altre forme di comunicazione, presenta propri stili e convenzioni. In particolare, è molto informale e ha una bassa soglia di utilizzo. Le persone che non si sono mai sognate di telefonare o scrivere una lettera a un VIP non esitano un secondo a spedire un messaggio di posta elettronica scritto in modo approssimativo.

I messaggi di posta elettronica contengono numerose "parole in codice", come CMQ (comunque) in italiano oppure ROTFL (Rolling On The Floor Laughing, mi sto rotolando sul pavimento dalle risate) in inglese. Molte persone utilizzano nei loro messaggi anche piccoli simboli ASCII chiamati **smiley** o **emoticon**. Alcuni dei più interessanti sono elencati nella Figura 7.6. Per la maggior parte, occorre ruotare il libro di 90° in senso orario per renderli più chiari. Un piccolo libro che offre oltre 650 smiley è (Sanderson e Dougherty, 1993).

I primi sistemi di posta elettronica erano semplicemente protocolli di trasferimento file, con la convenzione che la prima riga di ogni messaggio (o file) conteneva l'indirizzo del destinatario. Con il trascorrere del tempo le limitazioni di questo approccio sono diventate più evidenti.

Smiley	Significato	Smiley	Significato	Smiley	Significato
:)	Sono felice	=):-)	Abramo Lincoln	:+)	Nasone
:-(Sono triste/arrabbiato	=):-)	Zio Sam	:))	Doppio mento
:	Sono apatico	*<:-)	Babbo Natale	:(-)	Baffi
:)	Sto strizzando l'occhio	<:-)	Asino	#:-)	Capelli arruffati
:(O)	Sto gridando	(:-	Australiano	8-)	Occhiali
:(*")	Sto vomitando	:)X	Uomo con papillon	C:-)	Cervellone

Figura 7.6. Alcuni smiley. Non saranno presenti nell'esame finale. :-)

Alcune tra le più gravi sono qui elencate.

1. L'invio di un messaggio a un gruppo di persone era scomodo. I manager spesso hanno bisogno di questa funzione per inviare promemoria a tutti i loro subordinati.
2. I messaggi non avevano una struttura interna, complicando l'elaborazione da parte del computer. Per esempio, se un messaggio inoltrato era incluso nel corpo di un altro messaggio, l'estrazione della parte inoltrata dal messaggio ricevuto era difficile.
3. Il mittente non poteva sapere se un messaggio era arrivato o no.
4. Se qualcuno pianificava di allontanarsi dall'azienda per diverse settimane e voleva che tutta la posta in arrivo fosse gestita dalla sua segretaria, non era facile organizzarsi.
5. L'interfaccia utente era integrata in modo scadente con il sistema di trasmissione, e richiedeva agli utenti di modificare il file, uscire dall'editor e aprire il programma di trasferimento file.
6. Non era possibile creare e inviare messaggi contenenti un insieme di testo, disegni, fax e voce.

Con l'esperienza sono stati proposti sistemi di posta elettronica più elaborati. Nel 1982, le proposte per la posta elettronica di ARPANET furono pubblicate in RFC 821 (protocollo di trasmissione) e 822 (formato dei messaggi). Alcune revisioni minori (RFC 2821 e RFC 2822) sono divenute standard Internet, ma tutti fanno ancora riferimento alla posta Internet con RFC 822.

Nel 1984, CCITT ha abbozzato la sua raccomandazione X.400. Dopo due decenni di competizione, i sistemi di posta elettronica basati su RFC 822 sono ampiamente utilizzati, mentre quelli basati su X.400 sono scomparsi. Il fatto che un sistema messo insieme da un gruppo di studenti di informatica abbia battuto uno standard internazionale ufficiale, fortemente sostenuto da tutte le PTT nel mondo, da molti governi e da una parte sostanziale dell'industria dei computer, ricorda un po' la storia biblica di Davide e Golia.

Il motivo del successo di RFC 822 non sta tanto nella sua validità, ma nel fatto che X.400 era progettato in modo così scadente e complesso che nessuno poteva implementarlo correttamente. La scelta era tra un sistema di posta elettronica semplice (ma funzionante) basato su RFC 822 e un sistema di posta meraviglioso (ma non funzionante) basato su X.400, quindi molte organizzazioni scelsero il primo. È stata una bella lezione! Di conseguenza, la nostra discussione sulla posta elettronica si concentrerà sul sistema di posta elettronica Internet.

7.2.1 L'architettura e i servizi

In questa sezione forniremo una panoramica di ciò che possono fare i sistemi di posta elettronica, e del modo in cui sono organizzati. Normalmente sono composti da due sottosistemi: gli **agenti utente**, che consentono alle persone di leggere e inviare la posta elettronica, e gli **agenti di trasferimento dei messaggi**, che spostano i messaggi dall'origine alla destinazione. Gli agenti utente sono programmi locali che forniscono un metodo basato su comandi, menu o interfaccia grafica per interagire con il sistema di posta elettronica. Gli agenti di trasferimento dei messaggi sono generalmente **daemon** di sistema, vale a dire processi eseguiti in background. Il loro compito è spostare i messaggi di posta elettronica nel sistema.

Generalmente, i sistemi di posta elettronica supportano cinque funzioni di base. Vediamo quali sono.

La **composizione** è il processo di creazione di messaggi e risposte. Anche se è possibile utilizzare un editor di testo qualsiasi per il corpo del messaggio, il sistema stesso può fornire assistenza per l'indirizzamento e i numerosi campi di intestazione associati a ogni messaggio. Per esempio, quando si risponde a un messaggio, il sistema di posta elettronica può estrarre l'indirizzo del mittente dal messaggio in arrivo e inserirlo automaticamente nella risposta.

Il **trasferimento** indica lo spostamento dei messaggi dal mittente al destinatario. Per lo più consiste nello stabilire una connessione con la destinazione o a qualche computer intermedio, copiare il messaggio in output e rilasciare la connessione. Il sistema di posta elettronica dovrebbe svolgere automaticamente l'operazione, senza disturbare l'utente.

Il **reporting** ha a che fare con la comunicazione al mittente di ciò che è avvenuto al messaggio. È stato consegnato? È stato rifiutato? È stato perso? Esistono numerose applicazioni in cui la conferma della consegna è importante e può avere un significato legale ("Beh, Vostro Onore, il mio sistema di posta elettronica non è molto affidabile, pertanto immagino che il mandato di comparizione elettronico sia andato perduto").

La **visualizzazione** dei messaggi in arrivo è necessaria affinché le persone possano leggere la loro posta elettronica. A volte è richiesta una conversione, oppure deve essere aperto un visualizzatore speciale, per esempio se il messaggio è un file PostScript o una voce digitalizzata. A volte vengono tentate semplici operazioni di conversione e formattazione. La **collocazione** è il passaggio finale e riguarda l'operazione svolta dal destinatario con il messaggio dopo la ricezione. Egli potrebbe cancellarlo prima di leggerlo, eliminarlo dopo averlo letto, salvarlo e così via. Dovrebbe inoltre essere possibile recuperare e rileggere i messaggi salvati, inoltrarli o elaborarli in altri modi.

Oltre a questi servizi di base, alcuni sistemi di posta elettronica (specialmente quelli interni alle aziende) forniscono numerose funzionalità avanzate; di seguito ne sono citate alcune. Quando le persone si spostano o si allontanano per qualche periodo, potrebbero desiderare che la loro posta venga inoltrata: il sistema dovrebbe quindi essere in grado di farlo automaticamente.

La maggior parte dei sistemi consente agli utenti di creare **mailbox** (caselle di posta) per archiviare la posta in arrivo. Sono necessari comandi per creare ed eliminare le caselle di posta, analizzarne il contenuto, inserire ed eliminare messaggi e così via.

I manager delle aziende spesso inviano un messaggio a tutti i loro subordinati, clienti o fornitori. Da qui è nata l'idea delle **mailing list**, vale a dire elenchi di indirizzi di posta elettronica. Quando un messaggio viene inviato alla mailing list, copie identiche vengono consegnate a tutti i membri dell'elenco.

Altre funzionalità avanzate sono le copie per conoscenza, le copie per conoscenza nascosta, i messaggi di posta elettronica ad alta priorità, la posta segreta (o meglio crittografata), destinatari alternativi se il primo non è disponibile, la capacità da parte delle segretarie di leggere e rispondere alla posta del capo.

Ora la posta elettronica è ampiamente utilizzata nell'industria per la comunicazione tra società. Consente a dipendenti lontani di cooperare su progetti complessi, anche se vivono in zone con fusi orari diversi. Eliminando molti dei pregiudizi associati allo stato sociale, all'età e al sesso, i dibattiti per posta elettronica tendono a concentrarsi sulle idee, non sullo stato aziendale. Con la posta elettronica, una brillante idea di uno studente può avere maggiore impatto di un'idea stupida del vicepresidente.

Un'idea importante nei sistemi di posta elettronica è la distinzione tra l'**invólucro** e il suo contenuto. L'invólucro incapsula il messaggio e contiene tutte le informazioni necessarie per il trasporto del messaggio, come l'indirizzo di destinazione, la priorità e il livello di protezione, che sono distinte dal messaggio stesso. Gli agenti di trasporto dei messaggi utilizzano l'invólucro per il routing, come gli uffici postali tradizionali utilizzano le buste. Il messaggio all'interno dell'invólucro consiste di due parti: l'**intestazione** e il **corpo**. L'intestazione contiene le informazioni di controllo per gli agenti utente. Il corpo è dedicato interamente al destinatario umano. Invóluchi e messaggi sono illustrati nella Figura 7.7.

7.2.2 L'agente utente

I sistemi di posta elettronica presentano due parti fondamentali, come abbiamo visto: gli agenti utente e gli agenti di trasferimento dei messaggi; questa sezione presenta gli agenti utente. Un agente utente è di norma un programma che accetta una varietà di comandi per comporre, ricevere e rispondere ai messaggi, nonché per organizzare le caselle di posta. Alcuni agenti utente presentano un'interessante interfaccia grafica con menu e icone che richiede l'utilizzo di un mouse, mentre altri prevedono comandi a caratteri imparititi da tastiera. A livello di funzioni, comunque, si equivalgono.

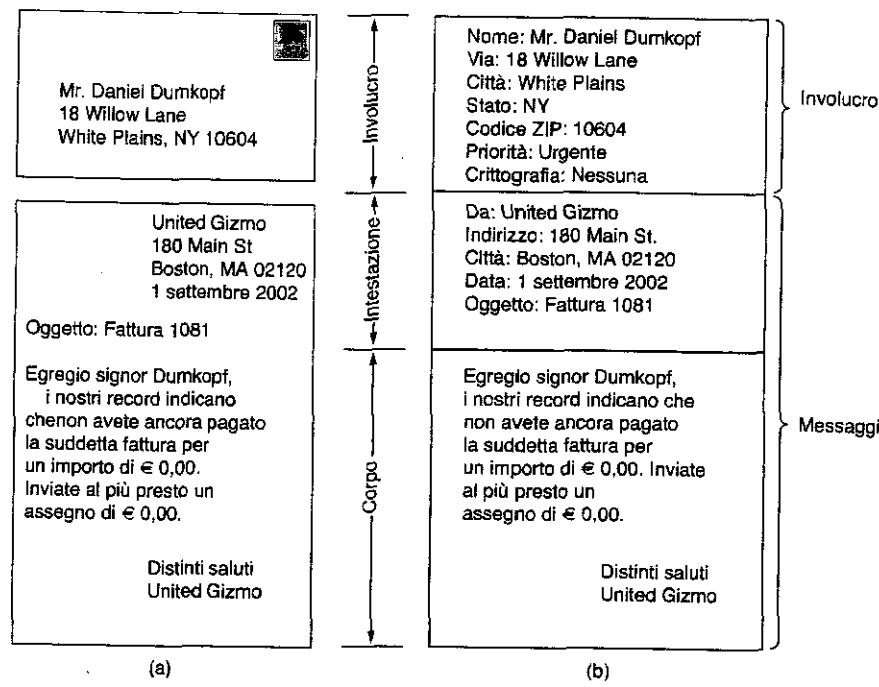


Figura 7.7. Involucri e messaggi. (a) Posta tradizionale. (b) Posta elettronica.

Invio di posta elettronica

Per inviare un messaggio di posta elettronica, un utente deve fornire il messaggio, l'indirizzo di destinazione e magari qualche altro parametro. Il messaggio può essere prodotto con un editor di testo autonomo, un programma di elaborazione testi o possibilmente un editor di testo specializzato incorporato nell'agente utente. L'indirizzo di destinazione deve essere in un formato che l'agente utente può gestire. Molti agenti utente si aspettano gli indirizzi nella forma *utente@indirizzo-dns*. Visto che abbiamo già studiato DNS non ripeteremo qui il suo funzionamento.

Tuttavia, vale la pena notare che esistono altre forme di indirizzamento. In particolare, gli indirizzi X.400 appaiono radicalmente diversi dagli indirizzi DNS. Sono composti di coppie *attributo = valore* separate da barre, come in questo esempio:

/C=US/ST=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/

Questo indirizzo specifica una nazione, uno stato, una località, un indirizzo personale e un nome comune (Ken Smith). Sono consentiti molti altri attributi, pertanto potete inviare la posta elettronica a qualcuno di cui non conoscete l'esatto indirizzo di posta elettronica, purché conoscate altri attributi (per esempio la società e il titolo professionale). Anche se i nomi X.400 sono considerevolmente meno pratici dei nomi DNS, la maggior parte dei sistemi di posta elettronica dispone di alias (o nickname) che permettono all'utente di

Lo strato applicazione

inserire o selezionare il nome di una persona per ottenere il corretto indirizzo di posta elettronica. Di conseguenza, anche con gli indirizzi X.400, solitamente non è necessario digitare queste stringhe.

La maggior parte dei sistemi di posta elettronica supporta le mailing list, pertanto un utente può inviare lo stesso messaggio a un elenco di persone con un singolo comando. Se la mailing list viene gestita localmente, l'agente utente può inviare un messaggio separato a ogni destinatario previsto; se invece la lista è gestita in remoto, i messaggi saranno espansi in tale posizione. Per esempio, se un gruppo di amanti del bird watching possiede una mailing list chiamata *birders* installata su *meadowlark.arizona.edu*, qualsiasi messaggio inviato a *birders@meadowlark.arizona.edu* sarà instradato all'università dell'Arizona e qui espanso in singoli messaggi per tutti i membri della mailing list, ovunque si trovino nel mondo. Gli utenti di questa mailing list non possono capire che si tratta di una mailing list. Potrebbe anche essere la casella di posta personale del professor Gabriel O. Birders.

Lettura della posta elettronica

Generalmente, quando un agente utente viene avviato, cerca nella casella di posta dell'utente i messaggi in arrivo prima di visualizzare altro sullo schermo. Potrebbe quindi annunciare il numero di messaggi nella casella di posta o visualizzare un riepilogo di una riga per ognuno, attendendo un comando.

Per un esempio del funzionamento di un agente utente, diamo un'occhiata a un tipico scenario di posta. Dopo aver avviato l'agente utente, l'utente chiede un riepilogo dei suoi messaggi. Sullo schermo potrebbe apparire qualcosa di simile alla Figura 7.8, dove ogni riga fa riferimento a un messaggio. In questo esempio, la casella di posta contiene otto messaggi.

#	Flag	Byte	Mittente	Oggetto
1	K	1030	asw	Modifiche a MINIX
2	KA	6348	trudy	Non tutte le Trudy sono cattive
3	K F	4519	Amy N. Wong	Richiesta di informazioni
4		1236	bal	Bioinformatica
5		104110	kaashoek	Materiale su peer-to-peer
6		1223	Frank	R: Una fantastica proposta
7		3110	guido	Il documento è stato accettato
8		1204	dmr	R: La visita dei miei studenti

Figura 7.8. Esempio di visualizzazione del contenuto di una casella di posta.

Ogni riga della visualizzazione contiene diversi campi estratti dall'involucro o dall'intestazione del messaggio corrispondente. In un semplice sistema di posta elettronica la scelta dei campi visualizzati è fissa, mentre nei sistemi più sofisticati l'utente può specificare quali campi visualizzare fornendo un **profilo utente**, vale a dire un file che descrive il formato di visualizzazione. In questo esempio di base il primo campo è il numero del mes-

saggio. Il secondo campo, *flag*, può contenere una *K*, che indica che il messaggio non è nuovo ma è stato letto in precedenza e conservato nella casella di posta, una *A*, che indica che al messaggio è stata inviata una risposta, e/o una *F*, che indica che il messaggio è stato inoltrato a qualcun altro. Sono consentiti anche altri flag.

Il terzo campo comunica la lunghezza del messaggio, mentre il quarto specifica chi ha inviato il messaggio. Dal momento che questo campo è stato semplicemente estratto dal messaggio, potrebbe contenere nomi di campo, nomi completi, iniziali, nomi di login o qualsiasi cosa che il mittente ha scelto di inserire qui. Infine, il campo *oggetto* offre un breve riepilogo del contenuto del messaggio. Le persone che dimenticano di includere un campo *oggetto* sconoscono spesso che le risposte ai loro messaggi tendono a non avere la priorità più alta.

Dopo avere visualizzato le intestazioni, l'utente può scegliere quale azione eseguire, per esempio visualizzare un messaggio, eliminare un messaggio e così via. I vecchi sistemi erano basati su testo e generalmente usavano comandi di un carattere per eseguire queste attività, come T, A, D e F (rispettivamente per comporre, rispondere, eliminare e inoltrare un messaggio). Un argomento specificava il messaggio in questione. I sistemi più recenti utilizzano le interfacce grafiche: di solito l'utente seleziona un messaggio con il mouse e poi fa clic su un'icona per digitare, rispondere, eliminarlo o inoltrarlo.

La posta elettronica ha compiuto molti passi avanti dai giorni in cui rappresentava solo un trasferimento di file. Agenti utente sofisticati permettono la gestione di un elevato volume di messaggi. Per le persone che ricevono e inviano migliaia di messaggi all'anno, questi strumenti sono inestimabili.

7.2.3 I formati dei messaggi

Passiamo ora dall'interfaccia utente al formato dei messaggi di posta elettronica stessi. Per prima cosa osserveremo i messaggi di posta ASCII che utilizzano RFC 822, e poi studieremo le estensioni multimediali a RFC 822.

RFC 822

I messaggi sono costituiti da un primitivo involucro (descritto in RFC 821), alcuni campi di intestazione, una riga vuota e il corpo del messaggio. Ogni campo di intestazione consiste logicamente di una singola riga di testo ASCII contenente il nome del campo, un carattere di due punti e, per la maggior parte dei campi, un valore. RFC 822 è stata progettata decentri fa e non distingue chiaramente i campi dell'involucro dai campi dell'intestazione. Anche se è stata rivista in RFC 2822, riscriverla completamente non è stato possibile a causa del suo utilizzo diffuso. Di norma, l'agente utente costruisce un messaggio e lo passa all'agente di trasferimento dei messaggi, che utilizza i campi di intestazione per costruire l'involucro effettivo, che è una sorta di ibrido "vecchio stile" tra messaggio e involucro.

I campi di intestazione principali correlati al trasporto del messaggio sono elencati nella Figura 7.9. Il campo *to:* indica l'indirizzo DNS del destinatario principale; è consentito indicare più destinatari. Il campo *cc:* indica gli indirizzi dei destinatari secondari. In termini di consegna non vi è una distinzione tra destinatari principali secondari: si tratta solo di una differenza psicologica che può essere importante per le persone coinvolte, ma non

per il sistema di posta elettronica. Il termine *cc:* significa copia per conoscenza; molte persone lo intendono come "copia carbone", una definizione datata ma sempre valida. Il campo *bcc:* (copia per conoscenza nascosta) è simile al campo *cc:*, tranne per il fatto che la riga viene eliminata da tutte le copie inviate ai destinatari principali e secondari. Questa funzionalità consente alle persone di inviare copie a terze parti senza che i destinatari principali e secondari lo sappiano.

Intestazione	Significato
To:	Gli indirizzi di posta elettronica dei destinatari primari
Cc:	Gli indirizzi di posta elettronica dei destinatari secondari
Bcc:	Gli indirizzi di posta elettronica per le copie per conoscenza nascosta
From:	La persona che ha creato il messaggio
Sender:	L'indirizzo di posta elettronica del mittente vero e proprio
Received:	La riga aggiunta da ogni agente di trasferimento lungo il percorso
Return-path:	Può essere utilizzato per identificare un percorso di ritorno al mittente

Figura 7.9. I campi di intestazione di RFC 822 correlati al trasporto del messaggio.

I due campi successivi, *from:* e *sender:*, indicano rispettivamente chi ha scritto e inviato il messaggio. Non devono necessariamente avere lo stesso contenuto. Per esempio, un dirigente commerciale può scrivere un messaggio, ma potrebbe essere la sua segretaria a trasmetterlo. In questo caso, il dirigente sarebbe elencato nel campo *from:* e la sua segretaria nel campo *sender:*. Il campo *from:* è obbligatorio, mentre il campo *sender:* può essere omesso se equivale al campo *from:*. Questi campi sono necessari nel caso in cui il messaggio non sia consegnabile e debba essere restituito al mittente.

Una riga contenente *received:* viene aggiunta da ogni agente di trasferimento dei messaggi lungo il percorso. La riga contiene l'identità dell'agente, la data e l'ora di ricezione del messaggio e altre informazioni che possono essere utilizzate per trovare i bug nel sistema di routing.

Il campo *return-path:* viene aggiunto dall'agente di trasferimento dei messaggi finale e ha lo scopo di spiegare come ritornare al mittente. In teoria, questa informazione può essere raccolta da tutte le intestazioni *received:* (tranne per il nome della casella di posta del mittente), ma viene raramente compilata in questo modo e di norma contiene solo l'indirizzo del mittente.

Oltre ai campi della Figura 7.9, i messaggi di RFC 822 possono contenere diversi campi d'intestazione utilizzati dagli agenti utente o dai destinatari umani. I più comuni sono elencati nella Figura 7.10. La maggior parte si spiega da sola, pertanto non saranno presentati tutti in dettaglio.

Il campo *reply-to:* viene a volte utilizzato quando né la persona che compone il messaggio né la persona che lo invia desiderano vedere la risposta. Per esempio, il direttore del marketing scrive un messaggio di posta elettronica parlando ai clienti di un nuovo prodotto. Il messaggio viene inviato da una segretaria, ma il campo *reply-to:* indica il capo del

Intestazione	Significato
Date:	La data e l'ora di invio del messaggio
Reply-to:	L'indirizzo di posta elettronica a cui inviare le risposte
Message-id:	Un numero univoco per fare riferimento a questo messaggio in seguito
In-reply-to:	L'ID del messaggio a cui si riferisce questa risposta
References:	Altri ID di messaggio rilevanti
Keywords:	Parole chiave scelte dall'utente
Subject:	Un breve riassunto del messaggio da visualizzare su una riga

Figura 7.10. Alcuni campi utilizzati nell'intestazione dei messaggi RFC 822.

reparto vendite, che può rispondere alle domande e accettare ordini. Questo campo è utile anche quando il mittente dispone di due account di posta elettronica e desidera che tutte le risposte giungano in uno.

Il documento RFC 822 afferma in modo esplicito che gli utenti possono inventare nuove intestazioni per utilizzo privato, purché tali intestazioni inizino con la stringa *X-*. È garantito che nessuna intestazione futura userà nomi che iniziano con *X-*, per evitare conflitti con intestazioni private e ufficiali. A volte gli universitari sapientoni creano campi come *X-Frutto-del-giorno:* o *X-Malattia-della-settimana:*, che sono legali ma non illuminanti. L'intestazione è seguita dal corpo del messaggio. Gli utenti possono inserire tutto quello che desiderano. Alcune persone terminano i loro messaggi con firme elaborate, compresi semplici disegni ASCII, citazioni di personaggi più o meno famosi, citazioni politiche e postille di ogni tipo (per esempio, "la società XYZ non è responsabile delle mie opinioni; in effetti, non è nemmeno in grado di comprenderle").

MIME (*Multipurpose Internet Mail Extensions*)

Agli esordi di ARPANET, la posta elettronica consisteva esclusivamente di messaggi di testo scritti in inglese ed espressi in ASCII. Per questo ambiente, RFC 822 svolgeva alla perfezione il lavoro: specificava le intestazioni e lasciava il contenuto agli utenti. Oggi sulla Internet mondiale questo approccio non è più adeguato. I problemi che possono nascere comprendono l'invio e la ricezione di:

1. messaggi scritti in lingue con accenti (per esempio italiano e francese)
2. messaggi in alfabeti non latini (per esempio ebraico e russo)
3. messaggi scritti in lingue con alfabeti ideografici (per esempio cinese e giapponese)
4. messaggi che non contengono testo (per esempio audio o immagini).

Lo strato applicazione

Una soluzione è stata proposta in RFC 1341 e aggiornata nelle RFC da 2045 a 2049. Questa soluzione, chiamata **MIME** (*Multipurpose Internet Mail Extensions*), oggi è ampiamente utilizzata e quindi sarà descritta di seguito; per ulteriori informazioni si possono consultare le RFC.

L'idea di base di MIME è continuare a utilizzare il formato RFC 822, aggiungendo una struttura al corpo del messaggio e definendo le regole di codifica per i messaggi non ASCII. Visto che non si discosta da RFC 822, i messaggi MIME possono essere inviati utilizzando i programmi e i protocolli di posta esistenti. Tutto ciò che deve essere cambiato sono i programmi di invio e ricezione, che è un compito lasciato agli utenti.

MIME definisce cinque nuove intestazioni dei messaggi, come mostrato nella Figura 7.11. La prima indica semplicemente all'agente utente di ricevere il messaggio in questione come un messaggio MIME e specifica la versione utilizzata. Qualsiasi messaggio che non contiene l'intestazione *MIME-version:* è un messaggio di testo normale in lingua inglese e viene elaborato come tale.

Intestazione	Significato
MIME-version:	Identifica la versione di MIME
Content-description:	Una stringa leggibile che comunica che cosa contiene il messaggio
Content-id:	Un identificatore univoco
Content-transfer-encoding:	Indica come il corpo del messaggio è stato preparato per la trasmissione
Content-type:	Il tipo e il formato del contenuto

Figura 7.11. Le intestazioni di RFC 822 aggiunte da MIME.

L'intestazione *content-description:* è una stringa ASCII che comunica il contenuto del messaggio. Questa intestazione è necessaria per consentire al destinatario di decidere se vale la pena di decodificare e leggere il messaggio. Se la stringa afferma "Foto del criceto di Barbara" e la persona che riceve il messaggio non è un grande estimatore di criceti, il messaggio sarà probabilmente scartato anziché decodificato in una fotografia a colori ad alta risoluzione.

L'intestazione *content-id:* identifica il contenuto. Usa lo stesso formato dell'intestazione standard *message-id:*.

L'intestazione *content-transfer-encoding:* spiega da cosa è costituito l'involucro per la trasmissione su una rete che può accettare solo oggetti contenenti caratteri costituiti da lettere, numeri e segni di punteggiatura. Sono previsti cinque schemi (più un carattere per indicarne di nuovi), e il più semplice è il testo ASCII. I caratteri ASCII usano 7 bit e possono essere trasportati direttamente dal protocollo di posta elettronica se le righe non superano la lunghezza di 1.000 caratteri.

Lo schema successivo in ordine di semplicità è identico salvo che utilizza caratteri di 8 bit (vale a dire tutti i valori da 0 a 255). Questo schema di codifica viola il protocollo di posta elettronica Internet originale, ma è utilizzato da alcune parti di Internet che implementano estensioni del protocollo originale. Anche se la dichiarazione della codifica non rende legale la tecnica, l'indicazione esplicita può per lo meno spiegare molte cose se qualcosa dovesse andare storto. I messaggi che utilizzano la codifica a 8 bit devono comunque aderire alla lunghezza massima standard delle righe.

Ancora peggiori sono i messaggi che utilizzano la codifica binaria. Si tratta di file binari arbitrari che non solo utilizzano tutti gli 8 bit, ma non rispettano nemmeno il limite di 1.000 caratteri per riga. I programmi eseguibili appartengono a questa categoria. Non vi sono garanzie che il messaggio in binario arrivi correttamente, ma alcune persone provano comunque.

Il modo corretto per codificare i messaggi binari prevede l'utilizzo della **codifica base64**, a volte chiamata **armatura ASCII**. In questo schema, gruppi di 24 bit sono suddivisi in quattro unità di 6 bit; ogni unità viene inviata come carattere ASCII legale. La codifica è "A" per 0, "B" per 1 e così via, seguita dalle 26 lettere minuscole, dalle dieci cifre e infine da + e / (rispettivamente 62 e 63). Le sequenze == e = indicano che l'ultimo gruppo contiene solo 8 o 16 bit (rispettivamente). I ritorni carrello e gli avanzamenti riga sono ignorati, pertanto possono essere inseriti a piacere per rispettare la lunghezza delle righe. Con questo schema si può inviare in sicurezza un testo binario arbitrario.

Per i messaggi quasi interamente ASCII ma contenenti alcuni caratteri non ASCII, la codifica base64 è inefficiente. Al suo posto viene utilizzata una codifica nota come **codifica quoted-printable**. Si tratta di ASCII a 7 bit, con tutti i caratteri superiori a 127 codificati come segni uguali seguiti dal valore del carattere espresso da due cifre esadecimale.

Riepilogando, i dati binari dovrebbero essere inviati con la codifica base64 o quoted-printable. Quando vi sono valide ragioni per non utilizzare questi schemi, è possibile specificare una codifica definita dall'utente nell'intestazione *content-transfer-encoding*:

L'ultima intestazione mostrata nella Figura 7.11 è quella più interessante, che specifica la natura del corpo del messaggio. RFC 2045 definisce sette tipi, per ognuno dei quali sono disponibili uno o più sottotipi. Il tipo e il sottotipo sono separati da una barra, come in

Content-type: video/mpeg

Il sottotipo deve essere fornito esplicitamente nell'intestazione; non sono forniti valori predefiniti. L'elenco iniziale di tipi e sottotipi specificato in RFC 2045 è mostrato nella Figura 7.12. Da allora ne sono stati aggiunti altri; voci aggiuntive vengono inserite ogni volta che se ne presenta la necessità.

Viene ora brevemente analizzato l'elenco dei tipi. Il tipo *text* è per il testo ASCII normale. La combinazione *text/plain* è per i messaggi ordinari che possono essere visualizzati come vengono ricevuti, senza codifica o ulteriori elaborazioni. Questa opzione consente ai messaggi ordinari di essere trasportati in MIME con poche intestazioni extra.

Il sottotipo *text/enriched* permette l'inclusione nel testo di un semplice linguaggio di markup. Questo linguaggio offre un modo indipendente dal sistema per esprimere grassetti, corsivi, dimensioni in punti più o meno elevate, rientri, giustificazioni, apici e pedici, non

Tipo	Sottotipo	Descrizione
Text	Plain	Testo non formattato
	Enriched	Testo con semplici comandi di formattazione
Image	Gif	Immagine statica nel formato GIF
	Jpeg	Immagine statica nel formato JPEG
Audio	Basic	Suono udibile
Video	Mpeg	Filmato nel formato MPEG
Application	Octet-stream	Una sequenza di byte non interpretata
	Postscript	Un documento stampabile in PostScript
Message	Rfc822	Un messaggio MIME RFC 822
	Partial	Un messaggio diviso per la trasmissione
	External-body	Il messaggio vero e proprio deve essere prelevato dalla rete
Multipart	Mixed	Parti indipendenti nell'ordine specificato
	Alternative	Lo stesso messaggio in formati diversi
	Parallel	Le parti devono essere visualizzate contemporaneamente
	Digest	Ogni parte è un messaggio RFC 822 completo

Figura 7.12. I tipi e i sottotipi MIME definiti in RFC 2045.

ché un semplice layout di pagina. Il linguaggio di markup è basato su SGML (Standard Generalized Markup Language), utilizzato anche come base per HTML. Per esempio, il messaggio

È giunto il **<bold>momento</bold>**, disse il **<italic>tricheco</italic>**...

sarebbe visualizzato come

È giunto il **momento**, disse il **tricheco**...

Sta al sistema ricevente stabilire la rappresentazione appropriata. Se sono disponibili, può utilizzare grassetti e corsivi; altrimenti può ricorrere al colore, al lampeggiamento, alla sottolineatura o ad altro per aggiungere l'enfasi. Sistemi diversi possono eseguire scelte differenti.

Quando il Web è divenuto popolare, è stato aggiunto un nuovo sottotipo *text/html* (RFC 2584) per consentire alle pagine Web di essere inviate come posta elettronica RFC 822. Un sottotipo per eXtensible Markup Language, *text/xml*, è definito in RFC 3023. HTML e XML verranno studiati in seguito in questo capitolo.

Il tipo MIME successivo è *image*, utilizzato per trasmettere immagini statiche. Oggi sono ampiamente utilizzati molti formati per archiviare e trasmettere le immagini, con o senza compressione. Due di questi, GIF e JPEG, sono incorporati in quasi tutti i browser, ma ne esistono altri che sono stati aggiunti all'elenco originale.

I tipi *audio* e *video* sono dedicati rispettivamente al suono e alle immagini in movimento. Occorre notare che *video* include solo le informazioni visive, non la colonna sonora. Se diventa necessario trasmettere un filmato con audio, le porzioni audio e video devono esse-

re trasmesse separatamente, a seconda del sistema di codifica utilizzato. Il primo formato video definito è stato quello inventato da Moving Picture Experts Group (MPEG), ma in seguito ne sono stati aggiunti altri. Oltre ad *audio/basic*, in RFC 3003 è stato aggiunto un nuovo tipo audio, *audio/mpeg*, per consentire alle persone di inviare per posta elettronica i file audio MP3.

Il tipo *application* è un “ripostiglio” per i formati che richiedono un’elaborazione esterna non compresi nelle altre tipologie. Un *octet-stream* è semplicemente una sequenza di byte non interpretati. Dopo la ricezione di tale flusso, un agente utente deve probabilmente visualizzarlo suggerendo all’utente che deve essere copiato in un file e richiedendo un nome file. L’elaborazione successiva è lasciata all’utente.

L’altro sottotipo definito è *postscript*, che fa riferimento al linguaggio PostScript definito da Adobe Systems e ampiamente utilizzato per la descrizione di pagine stampate. Molte stampanti dispongono di interpreti PostScript incorporati. Anche se un agente utente può chiamare un interprete PostScript esterno per visualizzare i file PostScript in ingresso, l’operazione non è priva di pericoli. PostScript è un linguaggio di programmazione completo. Con abbondante tempo a disposizione, una persona masochista potrebbe scrivere un compilatore C o un sistema di gestione dei database in PostScript. La visualizzazione di un messaggio PostScript in ingresso viene svolta eseguendo il programma PostScript che contiene. Oltre a visualizzare il testo, il programma può leggere, modificare o eliminare i file dell’utente, nonché produrre altri spiacevoli effetti collaterali.

Il tipo *message* consente a un messaggio di essere incapsulato in un altro. Questo schema è utile per l’inoltro della posta elettronica. Quando un messaggio RFC 822 completo viene incapsulato in un messaggio più esterno, deve essere utilizzato il sottotipo *rfc822*.

Il sottotipo *partial* permette di suddividere in più parti un messaggio incapsulato e di inviarle separatamente (per esempio se il messaggio incapsulato è troppo lungo). I parametri permettono di riassemblare tutte le parti nell’ordine corretto una volta giunte a destinazione.

Per finire, il sottotipo *external-body* può essere utilizzato per messaggi molto lunghi (come i filmati). Anziché includere il file MPEG nel messaggio, viene fornito un indirizzo FTP cui l’agente utente del ricevitore può fare riferimento nel momento opportuno. Questa funzione è particolarmente utile quando si invia un filmato a una mailing list, nel caso in cui solo poche persone desiderano realmente vederlo (si pensi alla posta elettronica indesiderata contenente video pubblicitari).

L’ultimo tipo è *multipart*, che consente di inserire più parti in un messaggio delimitando chiaramente l’inizio e la fine di ciascuna. Il sottotipo *mixed* consente a ogni parte di essere diversa, senza strutture aggiuntive imposte. Molti programmi di posta elettronica consentono all’utente di fornire uno o più allegati a un messaggio di testo. Questi allegati sono inviati utilizzando il tipo *multipart*.

In contrasto con *multipart*, il sottotipo *alternative* consente allo stesso messaggio di essere incluso più volte ma espresso con due o più supporti diversi. Per esempio, un messag-

gio potrebbe essere inviato in ASCII normale, testo formattato e PostScript. Un agente utente correttamente progettato che riceve tale messaggio lo visualizza probabilmente in PostScript, in subordine proverebbe il testo formattato, e se nessuna di queste soluzioni è disponibile verrebbe visualizzato il testo ASCII normale. Le parti dovrebbero essere ordinate dalla più semplice alla più complessa per aiutare i destinatari con agenti utente precedenti a MIME a dare un senso a un messaggio (anche un agente precedente a MIME può leggere il testo ASCII normale).

Il sottotipo *alternative* può essere utilizzato anche per più lingue. In questo contesto, la stele di Rosetta può essere vista come un vecchio messaggio *multipart/alternative*.

Un esempio multimediale è mostrato nella Figura 7.13. Trasmette gli auguri di compleanno utilizzando sia il testo sia una canzone. Se il ricevitore dispone delle funzionalità audio, l’agente utente analizzerà il file audio, *birthday.snd*, e lo riprodurrà; in caso contrario viene visualizzato sullo schermo il testo del brano, senza audio. Le parti sono delimitate da due trattini seguiti da una stringa (generata dal software) specificata nel parametro *boundary*.

```
From: elinor@abcd.com
To: carolyn@xyz.com
MIME-version: 1.0
Message-id: <0704760941.AA00747@abcd.com>
Content-type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: La terra orbita intorno al sole un numero intero di volte
```

Questo è il preambolo. L’agente utente lo ignora. Buona giornata.

```
--qwertyuiopasdfghjklzxcvbnm
Content-type: text/enriched
```

```
Tanti auguri a te
Tanti auguri a te
Tanti auguri, cara <b>Carolyn</b>
Tanti auguri a te
```

```
--qwertyuiopasdfghjklzxcvbnm
Content-type: message/external-body;
access-type="anon-ftp";
site="bicycle.abcd.com";
directory="pub";
name="birthday.snd"
```

```
Content-type: audio/basic
Content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

Figura 7.13. Un messaggio multipart contenente come alternative audio e testo formattato.

Occorre notare che l'intestazione *content-type* appare in tre posizioni all'interno di questo esempio. Nella parte superiore, indica che il messaggio è diviso in più parti; all'interno di ogni parte, restituisce il tipo e il sottotipo di tale parte. Per finire, all'interno del corpo della seconda parte, è richiesta per comunicare all'agente utente quale tipo di file esterno deve analizzare. Le intestazioni non fanno distinzione tra maiuscole e minuscole. *Content-transfer-encoding* è richiesto anche per un corpo esterno che non è codificato come ASCII a 7 bit.

Tornando ai sottotipi per i messaggi in più parti, esistono due possibilità. Il sottotipo *parallel* è utilizzato quando tutte le parti devono essere "viste" contemporaneamente; un esempio sono i filmati con un canale audio e un canale video. I filmati sono più efficaci se questi due flussi sono riprodotti in parallelo, anziché consecutivamente.

Per finire, il sottotipo *digest* viene utilizzato quando molti messaggi sono uniti in un messaggio composito. Per esempio, alcuni gruppi di discussione su Internet raccolgono messaggi dagli iscritti e poi li inviano al gruppo come singolo messaggio *multipart/digest*.

7.2.4 Il trasferimento dei messaggi

Il sistema di trasferimento dei messaggi si occupa della trasmissione dei messaggi dal mittente al destinatario. Il modo più semplice per svolgere questa operazione è stabilire una connessione di trasporto dalla macchina di origine a quella di destinazione per poi trasferire semplicemente il messaggio. Dopo avere esaminato come viene normalmente svolta questa operazione, analizzeremo alcuni situazioni in cui non funziona e illustreremo che cosa è possibile fare.

SMTP (*Simple Mail Transfer Protocol*)

All'interno di Internet, la posta elettronica viene consegnata costituendo una connessione tra la macchina di origine e la porta 25 della macchina di destinazione. In ascolto su questa porta esiste un daemon di posta elettronica che utilizza **SMTP** (*Simple Mail Transfer Protocol*). Questo daemon accetta le connessioni in ingresso e copia nelle caselle di posta appropriate i messaggi ricevuti da queste connessioni. Se un messaggio non può essere consegnato, al mittente viene restituito un rapporto di errore contenente la prima parte del messaggio non consegnabile.

SMTP è un semplice protocollo ASCII. Dopo avere stabilito la connessione TCP alla porta 25, la macchina di invio (che opera come client) attende la comunicazione da parte della macchina ricevente (che opera come server). Il server inizia inviando una riga di testo che comunica la sua identità e la possibilità di inviare la posta. Se questo non avviene, il client rilascia la connessione e riprova in seguito.

Se il server è pronto ad accettare la posta elettronica, il client annuncia da chi proviene il messaggio e a chi è destinato. Se tale destinatario esiste nella destinazione, il server comunica al client di inviare il messaggio. Dopo di che, il client invia il messaggio e il server fornisce l'acknowledgement. Non sono necessari checksum, perché TCP offre un flusso di byte affidabile. Se vi sono più messaggi di posta elettronica, ora vengono inviati tutti. Una volta scambiata tutta la posta elettronica in entrambe le direzioni, la connessione viene

rilasciata. Una semplice finestra di dialogo per inviare il messaggio della Figura 7.13, comprensiva dei codici numerici utilizzati da SMTP, è mostrata nella Figura 7.14. Le linee inviate dal client sono contrassegnate con *C:*. Quelli inviati dal server sono contrassegnati con *S:*.

Alcuni commenti sulla Figura 7.14 potrebbero essere utili. Il primo comando dal client è *HELO*. Delle varie combinazioni di quattro caratteri che rappresentano abbreviazioni di *HELLO* (ciao), questa presenta numerosi vantaggi sulle rivali. Il motivo per cui tutti i comandi devono essere di quattro caratteri si è perso nel tempo. Nella Figura 7.14, il messaggio è inviato a un solo destinatario, per cui viene utilizzato un solo comando *RCPT*. Questo comando permette l'invio di un singolo messaggio a più destinatari, dove per ognuno viene ricevuto un acknowledgement o un rifiuto. Anche se alcuni destinatari vengono rifiutati (perché non esistono nella destinazione), il messaggio può essere inviato agli altri.

Per finire, anche se la sintassi dei comandi di quattro caratteri dal client è specificata in modo rigido, la sintassi delle risposte lo è di meno: in realtà contano solo i codici numerici, e ogni implementazione può inserire la stringa che desidera dopo di essi.

Per comprendere meglio come funzionano SMTP e altri protocolli descritti in questo capitolo si può fare una prova. Per prima cosa occorre individuare una macchina connessa a Internet. Su un sistema UNIX, digitare nella shell

```
telnet mail.isp.com 25
```

sostituendo alla scritta *mail.isp.com* dell'esempio il nome DNS del server di posta dell'ISP (internet service provider) utilizzato. Su un sistema Windows, selezionate Start, Esegui e digitate il comando nella finestra di dialogo. Questo comando stabilirà una connessione telnet (cioè TCP) alla porta 25 del computer indicato, cioè alla sua porta SMTP (vedere la Figura 6.27 per alcune porte comuni). Probabilmente otterrete una risposta come la seguente:

```
Connessione a mail.isp.com...
Connesso a mail.isp.com
Il carattere di escape è ']'.
220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200
```

Le prime tre righe provengono da telnet, che comunica che cosa sta facendo. L'ultima riga proviene dal server SMTP sulla macchina remota, che annuncia il suo desiderio di comunicare e di accettare la posta elettronica. Per scoprire i comandi accettati è possibile digitare

```
HELP
```

Da questo punto in poi, è possibile creare una sequenza di comandi come quella nella Figura 7.14, iniziando con il comando *HELO* del client.

Vale la pena notare che l'utilizzo di righe di testo ASCII per i comandi non è un caso. Molti protocolli Internet lavorano in questo modo. L'utilizzo del testo ASCII facilita la verifica e il debug dei protocolli. Possono essere provati inviando manualmente i comandi, come visto in precedenza; i blocchi di messaggi sono facili da leggere.

```

S: 220 xyz.com SMTP service ready
C: HELO abcd.com
    S: 250 xyz.com says hello to abcd.com
C: MAIL FROM: <elinor@abcd.com>
    S: 250 sender ok
C: RCPT TO: <carolyn@xyz.com>
    S: 250 recipient ok
C: DATA
    S: 354 Send mail; end with "." on a line by itself
C: from: elinor@abcd.com
C: to: carolyn@xyz.com
C: MIME-version: 1.0
C: message-id: <0704760941.AA00747@abcd.com>
C: content-type: multipart/alternative; boundary=qwertyuopasdfghjklzxcvbnm
C: subject: La terra orbita intorno al sole un numero intero di volte
C:
C: Questo è il preambolo. L'agente utente lo ignora. Buona giornata.
C:
C: --qwertyuopasdfghjklzxcvbnm
C: content-type: text/enriched
C:
C: Tanti auguri a te
C: Tanti auguri a te
C: Tanti auguri, cara <bold>Carolyn</bold>
C: Tanti auguri a te
C:
C: --qwertyuopasdfghjklzxcvbnm
C: content-type: message/external-body;
C: access-type="anon-ftp";
C: site="bicycle.abcd.com";
C: directory="pub";
C: name="birthday.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuopasdfghjklzxcvbnm
C:
    S: 250 message accepted
C: QUIT
    S: 221 xyz.com closing connection

```

Figura 7.14. Il trasferimento di un messaggio da elinor@abcd.com a carolyn@xyz.com.

Anche se il protocollo SMTP è completamente ben definito, possono comunque sorgere dei problemi. Uno è legato alla lunghezza del messaggio: alcune vecchie implementazioni non possono gestire i messaggi che superano 64 KB. Un altro problema riguarda i timeout. Se il client e il server sperimentano diversi timeout, uno dei due potrebbe abban-

donare i tentativi mentre l'altro è ancora occupato, terminando inaspettatamente la connessione. Per finire, in casi rari, si potrebbero innescare tempeste di mail infinite. Per esempio, se l'host 1 gestisce la mailing list A e l'host 2 gestisce la mailing list B, e ogni lista contiene una voce relativa all'altra, un messaggio inviato a una delle liste potrebbe generare una quantità illimitata di traffico di posta elettronica, fino a quando qualcuno va a controllare che cosa sta succedendo.

Per risolvere questi problemi, in RFC 2821 è stato definito **ESMTP** (*Extended SMTP*). I client che desiderano utilizzarlo devono inviare un messaggio iniziale ***EHLO*** al posto di ***HELO***. Se viene rifiutato, il server è un normale server SMTP e il client deve procedere nel modo solito. Se ***EHLO*** viene accettato, sono consentiti i nuovi comandi e parametri.

7.2.5 La consegna finale

Finora abbiamo supposto che tutti gli utenti lavorino su computer in grado di inviare e ricevere la posta elettronica. Come abbiamo visto, la posta elettronica viene consegnata quando il mittente stabilisce una connessione TCP al destinatario e quindi spedisce la posta tramite tale connessione. Questo modello ha funzionato bene per decenni, quando tutti gli host ARPANET (e in seguito Internet) erano concretamente sempre online per accettare le connessioni TCP.

Tuttavia, con l'avvento delle persone che accedono a Internet chiamando l'ISP tramite un modem, il sistema è diventato inadeguato. Il problema è il seguente: che cosa accade quando Elinor desidera inviare un messaggio di posta elettronica a Carolyn, ma Carolyn al momento non è online? Elinor non può stabilire una connessione TCP a Carolyn, e quindi non può eseguire il protocollo SMTP.

Una soluzione consiste nell'avere un agente di trasferimento dei messaggi su una macchina dell'ISP, che accetta la posta elettronica per i suoi clienti e la archivia nelle caselle di posta sulla macchina dell'ISP. Dal momento che questo agente può essere sempre online, la posta elettronica può essere inviata in qualsiasi momento.

POP3

Sfortunatamente, questa soluzione crea un altro problema: come può l'utente ricevere la posta elettronica dall'agente di trasferimento dei messaggi dell'ISP? La soluzione a questo problema è creare un altro protocollo, che consente agli agenti di trasferimento dell'utente (sui PC client) di contattare l'agente di trasferimento dei messaggi (sulla macchina dell'ISP) per effettuare la copia della posta elettronica dall'ISP all'utente. Uno di questi protocolli è **POP3** (*Post Office Protocol version 3*), descritto in RFC 1939.

La situazione in cui il mittente e il ricevitore dispongono di una connessione permanente a Internet è mostrata nella Figura 7.15(a). La situazione in cui il mittente è attualmente online, al contrario del destinatario, è illustrata nella Figura 7.15(b).

POP3 viene avviato quando l'utente apre il lettore della posta. Il lettore della posta chiama l'ISP (a meno che sia già disponibile una connessione) e stabilisce una connessione

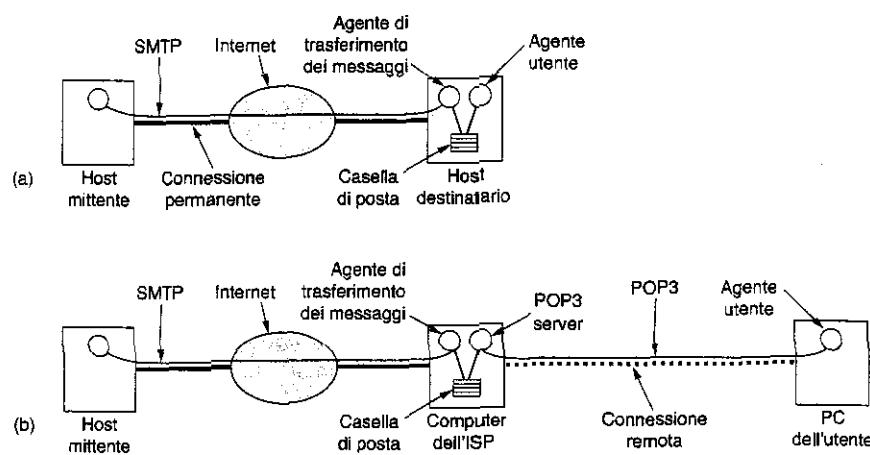


Figura 7.15. (a) Invio e lettura della posta quando il destinatario ha una connessione permanente a Internet e l'agente utente viene eseguito sulla stessa macchina dell'agente di trasferimento dei messaggi. (b) Lettura della posta elettronica quando il destinatario utilizza una connessione telefonica a un ISP.

TCP con l'agente di trasferimento dei messaggi alla porta 110. Una volta stabilita la connessione, il protocollo POP3 attraversa sequenzialmente tre stati:

1. autorizzazione
2. transazioni
3. aggiornamento.

Lo stato di autorizzazione si occupa del login dell'utente; lo stato delle transazioni consente all'utente di raccogliere la posta e cancellarla dalla casella dell'ISP. Lo stato di aggiornamento provoca l'effettiva eliminazione dei messaggi.

Questo comportamento può essere osservato digitando qualcosa di simile a

```
telnet mail.isp.com 110
```

dove *mail.isp.com* rappresenta il nome DNS del server di posta dell'ISP. Telnet stabilisce una connessione alla porta 110, su cui è in ascolto il server POP3. Dopo avere accettato la connessione TCP, il server invia un messaggio ASCII che annuncia la sua presenza. Solitamente, inizia con +OK seguito da un commento. Uno scenario di esempio è mostrato nella Figura 7.16; inizia dall'istante successivo alla costituzione della connessione TCP. Come prima, le righe contrassegnate da C: provengono dal client (utente), mentre quelle indicate con S: provengono dal server (agente di trasferimento dei messaggi sulla macchina dell'ISP).

```

S: +OK POP3 server ready
C: USER carolyn
S: +OK
C: PASS vegetables
S: +OK login successful
C: LIST
S: 1 2505
S: 2 14302
S: 3 8122
S: .
C: RETR 1
S: (invia il messaggio 1)
C: DELE 1
C: RETR 2
S: (invia il messaggio 2)
C: DELE 2
C: RETR 3
S: (invia il messaggio 3)
C: DELE 3
C: QUIT
S: +OK POP3 server disconnecting

```

Figura 7.16. Utilizzo di POP3 per il prelievo di tre messaggi.

Durante lo stato di autorizzazione, il client invia il suo nome utente e la sua password. Dopo il login, il client può inviare il comando *LIST*, che chiede al server di elencare il contenuto della casella di posta, un messaggio per riga, specificando la lunghezza di ogni messaggio. L'elenco è terminato da un punto.

Il client può quindi recuperare i messaggi utilizzando il comando *RETR* e contrassegnarli per l'eliminazione con *DELE*. Una volta recuperati tutti i messaggi (e possibilmente dopo averli contrassegnati per l'eliminazione), il client impedisce il comando *QUIT* per terminare lo stato delle transazioni ed entrare nello stato di aggiornamento. Quando il server ha eliminato tutti i messaggi, invia una risposta e interrompe la connessione TCP.

Anche se il protocollo POP3 supporta la capacità di scaricare un messaggio o gruppo di messaggi specifico lasciando gli altri sul server, la maggior parte dei programmi di posta elettronica scarica tutto e svuota la casella di posta. Questo comportamento, in pratica, garantisce che il disco rigido dell'utente contiene l'unica copia di ogni messaggio. In caso di crash, tutti i messaggi di posta elettronica possono andare persi per sempre.

Riepiloghiamo ora il funzionamento della posta elettronica per i clienti degli ISP. Elinor crea un messaggio per Carolyn utilizzando un programma di posta elettronica (agente utente) e fa clic su un'icona per inviarlo. Il programma di posta elettronica spedisce il messaggio all'host di Elinor utilizzando l'agente di trasferimento dei messaggi. L'agente di trasferimento dei messaggi vede che è diretto a *carolyn@xyz.com*, pertanto utilizza DNS per individuare il record *MX* per *xyz.com* (dove *xyz.com* è l'ISP di Carolyn). Questa inter-

rogazione restituisce il nome DNS del server di posta di *xyz.com*. L'agente di trasferimento dei messaggi cerca ora l'indirizzo IP di questo computer utilizzando di nuovo DNS, per esempio con *gethostbyname*. Stabilisce poi una connessione TCP al server SMTP sulla porta 25 di questo computer. Utilizzando una sequenza di comandi SMTP analoga a quella della Figura 7.14, trasferisce il messaggio alla casella di posta di Carolyn e interrompe la connessione TCP.

Dopo qualche tempo, Carolyn accende il suo PC, si connette all'ISP e avvia il suo programma di posta elettronica. Il programma di posta elettronica stabilisce una connessione TCP al server POP3, sulla porta 110 del computer con il server di posta dell'ISP. Il nome DNS o l'indirizzo IP del server vengono di norma configurati durante l'installazione del programma di posta elettronica o l'iscrizione ai servizi dell'ISP. Una volta stabilita la connessione TCP, il programma di posta elettronica di Carolyn esegue il protocollo POP3 per scaricare il contenuto della sua casella di posta sul disco rigido, utilizzando comandi simili a quelli della Figura 7.16. Una volta trasferita tutta la posta elettronica, la connessione TCP viene rilasciata. In effetti, anche la connessione all'ISP può essere terminata, perché tutta la posta elettronica si trova sul disco rigido di Carolyn. Naturalmente, per inviare una risposta sarà di nuovo necessaria la connessione all'ISP; per questo motivo di solito non viene interrotta appena dopo il recupero della posta.

IMAP

Per un utente con un account di posta elettronica presso un ISP, che accede sempre dallo stesso PC, POP3 è perfetto ed è ampiamente utilizzato per la sua semplicità e solidità. Tuttavia, è una verità dell'industria informatica che, non appena qualcosa funziona bene, qualcuno inizia a richiedere altre funzionalità (ottenendo altri bug). Questo è accaduto anche per la posta elettronica. Per esempio, molte persone che disponevano di un singolo account di posta elettronica al lavoro o a scuola desideravano accedervi dal lavoro, dal PC della loro abitazione, dai loro portatili quando erano in viaggio di affari e dai cyber-café in vacanza. Anche se POP3 lo consente, di norma scarica tutti i messaggi archiviati ad ogni contatto: il risultato è che la posta dell'utente viene rapidamente dispersa su più macchine, più o meno casualmente (alcune non appartengono nemmeno all'utente!).

Questo svantaggio ha portato alla nascita di un protocollo di consegna alternativo, **IMAP** (*Internet Message Access Protocol*), definito in RFC 2060. A differenza di POP3, che fondamentalmente presume che l'utente cancelli la casella di posta a ogni contatto, IMAP presume che tutti i messaggi devono rimanere sul server all'infinito, in più cartelle di posta. IMAP fornisce sofisticati meccanismi per leggere in tutto o in parte i messaggi, funzione utile durante l'utilizzo di un modem lento per leggere la parte di testo di un messaggio *multipart* con grandi allegati audio e video. Visto che si ipotizza che i messaggi non saranno trasferiti al computer dell'utente per un'archiviazione permanente, IMAP fornisce meccanismi per creare, distruggere e manipolare più cartelle di posta sul server. In questo modo, un utente può mantenere una cartella di posta per ogni corrispondente e spostare i messaggi dalla posta in arrivo dopo la lettura.

IMAP ha molte funzionalità, come la capacità d'indicizzare la posta non per numeri di arrivo, come nella Figura 7.8, ma utilizzando gli attributi (per esempio, mostra il primo

messaggio di Roberto). A differenza di POP3, IMAP può anche accettare la posta in uscita per l'invio alle destinazioni, oltre che consegnare la posta in arrivo.

L'impostazione generale del protocollo IMAP è simile a quella di POP3, come mostrato nella Figura 7.16, tranne per il fatto che esistono decine di comandi. Il server IMAP è in ascolto sulla porta 143. Un confronto tra POP3 e IMAP è presentato nella Figura 7.17. Si dovrebbe notare, comunque, che non tutti gli ISP e non tutti i programmi di posta elettronica supportano entrambi i protocolli. Di conseguenza, quando si sceglie un programma di posta elettronica, è importante scoprire quali protocolli supporta e assicurarsi che l'ISP ne supporti almeno uno.

Caratteristica	POP3	IMAP
Definizione del protocollo	RFC 1939	RFC 2060
Porta TCP utilizzata	110	143
Luogo di archiviazione della posta	PC dell'utente	Server
Lettura della posta	Offline	Online
Tempo di connessione richiesto	Ridotto	Elevato
Utilizzo delle risorse del server	Minimo	Estensivo
Più cartelle di posta	No	Sì
Chi esegue il backup delle caselle di posta	Utente	ISP
Valido per utenti in movimento	No	Sì
Controllo dell'utente sul download	Ridotto	Elevato
Download parziale dei messaggi	No	Sì
Le quote del disco sono un problema	No	Potrebbero esserlo nel tempo
Semplice da implementare	Sì	No
Supporto diffuso	Sì	In crescita

Figura 7.17. Un confronto di POP3 e IMAP.

Le funzionalità di consegna

Indipendentemente dall'utilizzo di POP3 o IMAP, molti sistemi forniscono agganci per un'ulteriore elaborazione della posta elettronica. Una funzionalità particolarmente preziosa per molti utenti della posta elettronica è la possibilità di impostare **filtri**. Si tratta di regole controllate alla ricezione della posta elettronica o all'avvio dell'agente utente. Ogni regola specifica una condizione e un'azione. Per esempio, una regola potrebbe affermare che qualsiasi messaggio ricevuto dal capo deve essere spostato nella cartella di posta 1, i messaggi da un gruppo selezionato di amici vanno nella cartella di posta 2, mentre i messaggi contenenti parole equivoche nella riga Oggetto devono essere scartati senza commenti.

Alcuni ISP forniscono un filtro che divide automaticamente in categorie la posta in arrivo, separando i messaggi importanti dallo spam (posta indesiderata) e archiviando ogni messaggio nella cartella corrispondente. Tali filtri lavorano generalmente controllando prima se la fonte è uno spammer noto. Poi esaminano generalmente la riga dell'oggetto. Se cen-

tinaia di utenti hanno ricevuto un messaggio con la stessa riga dell'oggetto, probabilmente si tratta di spam. Per il rilevamento dello spam vengono utilizzate anche altre tecniche. Un'altra funzionalità di consegna fornita spesso è la capacità di inoltrare temporaneamente la posta in arrivo a un indirizzo diverso. Questo indirizzo può anche essere relativo a un computer gestito da un servizio commerciale, che inoltra le pagine via radio o via satelliti visualizzando la riga *subject*: sul cercapersone dell'utente.

Un'altra utile funzionalità di consegna finale è la possibilità di installare un **daemon per le assenze**. Si tratta di un programma che esamina ogni messaggio in arrivo e invia al mittente una risposta insipida come

Ciao. Sono in vacanza! Tornerò il 24 di agosto. Buone ferie!

Tali risposte possono anche specificare come gestire questioni urgenti nel frattempo, altre persone da contattare per problemi specifici e così via. La maggior parte dei daemon per le assenze tiene traccia delle persone a cui ha già inviato la risposta ed evita di inviare alla stessa persona una replica identica. I più validi controllano anche se il messaggio in arrivo è stato inviato a una mailing list, e in questo caso non inviano la risposta automatica. In effetti, le persone che durante l'estate inviano messaggi a mailing list molto frequentate non desiderano ricevere centinaia di risposte relative alle vacanze degli altri membri. L'autore ha incontrato una volta una forma estrema di elaborazione delle consegne quando ha inviato un messaggio di posta elettronica a una persona che si lamenta di ricevere 600 messaggi al giorno. Qui la sua identità non sarà divulgata, affinché i lettori di questo libro non gli invino altre e-mail. Lo chiameremo John.

John ha installato un robot di posta elettronica che controlla ogni messaggio in arrivo per vedere se proviene da un nuovo corrispondente. In questo caso, il robot invia una risposta automatica spiegando che John non può più leggere personalmente tutti i suoi messaggi. Ha quindi prodotto un documento di FAQ (*Frequently Asked Questions*) personale che risponde a molte delle domande che gli vengono poste. Di solito sono i newsgroup a offrire delle FAQ, non le persone.

Le FAQ di John comunicano il suo indirizzo, il numero di fax e il numero di telefono, e spiegano come contattare la sua società. Egli spiega come richiedere la sua partecipazione a una conferenza e descrive dove trovare i suoi documenti. Offre anche alcuni puntatori ai software che ha scritto, alle conferenze tenute, agli standard di cui è l'editor e così via. Questo approccio è necessario, ma forse le FAQ personali diventeranno uno status symbol.

Webmail

Un ultimo argomento da menzionare è la Webmail, o posta Web. Alcuni siti Web, per esempio Hotmail e Yahoo!, offrono un servizio di posta elettronica a chiunque lo richieda. Tali servizi funzionano come segue. Essi dispongono di normali agenti di trasferimento dei messaggi in ascolto sulla porta 25 per le connessioni SMTP in ingresso. Per contattare Hotmail, per esempio, occorre acquisire il loro record MX DNS, per esempio digitando

`host -a -v hotmail.com`

su un sistema UNIX. Supponiamo che il server di posta sia chiamato *mx10.hotmail.com*; digitando

`telnet mx10.hotmail.com 25`

è possibile stabilire una connessione TCP su cui inviare i comandi SMTP nel modo solito. Finora niente di strano, tranne il fatto che questi grandi server sono spesso occupati, per cui potrebbero essere necessari diversi tentativi prima che una connessione TCP venga accettata.

La parte interessante riguarda la consegna della posta elettronica. Fondamentalmente, quando l'utente osserva la pagina Web della posta elettronica, vede un modulo che chiede il nome di login e la password. Quando l'utente fa clic su Sign In o Accedi, il nome di login e la password vengono inviati al server per la convalida. Se il login ha successo, il server trova la casella di posta dell'utente e crea un elenco simile a quello della Figura 7.8, ma formattato come una pagina Web in HTML. La pagina Web viene quindi inviata al browser per la visualizzazione. Molti degli elementi sulla pagina sono selezionabili, per consentire la lettura o l'eliminazione dei messaggi.

7.3 Il World Wide Web

Il World Wide Web è un'infrastruttura per l'accesso ai documenti collegati sparsi su milioni di macchine che appartengono a Internet. In 10 anni, si è evoluto da un modo per distribuire dati sulla fisica delle alte energie a un'applicazione che milioni di persone considerano come "Internet". La sua enorme popolarità è dovuta alla piacevole interfaccia grafica, di facile utilizzo per i principianti; inoltre, fornisce moltissime informazioni su qualsiasi argomento, dagli abati agli Zulu.

Il Web (noto anche come WWW) nacque nel 1989 presso CERN, il centro europeo per la ricerca nucleare. CERN ha diversi team di scienziati provenienti da tutte le nazioni europee aderenti al progetto, che svolgono ricerche sulla fisica delle particelle. Questi team sono composti spesso da membri di più di una decina di nazioni. La maggior parte degli esperimenti sono molto complessi e richiedono anni per la pianificazione e la costruzione dell'attrezzatura. Il Web è cresciuto grazie all'esigenza di collaborare di questi team di ricercatori dispersi tra le varie nazioni, che utilizzano una raccolta in costante cambiamento di rapporti, documenti, disegni e fotografie.

La proposta iniziale per una rete di documenti collegati fu opera del fisico Tim Berners-Lee nel marzo del 1989. Il primo prototipo (basato su testo) fu operativo 18 mesi dopo. Nel dicembre 1991, fu eseguita una presentazione pubblica alla conferenza Hypertext '91 a San Antonio, Texas.

Questa dimostrazione e la sua pubblicità attrassero l'attenzione di altri ricercatori, che portarono Marc Andreessen dell'università dell'Illinois a iniziare a sviluppare il primo browser grafico, Mosaic, che fu rilasciato nel febbraio del 1993. Mosaic era così popolare che, un anno dopo, Andreessen creò una nuova società, Netscape Communications Corp., il cui obiettivo era sviluppare client, server e altri programmi per il Web. Quando

Netscape entrò in borsa nel 1995, gli azionisti investirono un miliardo e mezzo di dollari, pensando che questa società sarebbe diventata la prossima Microsoft. Questo record fu sorprendente perché la società offriva un solo prodotto, operava in rosso e aveva annunciato nel suo prospetto di non prevedere di trarre un profitto nel futuro prossimo. Per i tre anni successivi, Netscape Navigator e Microsoft Internet Explorer ingaggiarono la "guerra dei browser", cercando di aggiungere nuove caratteristiche (e quindi più bug) rispetto al prodotto concorrente. Nel 1998, America Online acquistò Netscape Communications Corp. per 4,2 miliardi di dollari, ponendo termine alla breve vita di Netscape come società autonoma.

Nel 1994 CERN e M.I.T. firmarono un accordo per creare **World Wide Web Consortium** (abbreviato in **W3C**), un'organizzazione che ha come scopo sviluppare ulteriormente il Web, standardizzare i protocolli e incoraggiare l'interoperabilità tra i siti. Berners-Lee divenne il direttore. Da allora, centinaia di università e società sono entrate a far parte del consorzio. Anche se ora esistono moltissimi libri sul Web, il luogo migliore dove trovare informazioni aggiornate sul Web è naturalmente il Web stesso. La homepage del consorzio si trova all'indirizzo www.w3.org. I lettori interessati possono consultarla per seguire i collegamenti alle pagine che descrivono i numerosi documenti e attività del consorzio.

7.3.1 Una panoramica dell'architettura

Dal punto di vista degli utenti, il Web è una vasta raccolta mondiale di documenti o **pagine Web**, spesso definite solamente **pagine** per brevità. Ogni pagina può contenere collegamenti ad altre pagine situate ovunque nel mondo.

Gli utenti possono seguire un collegamento facendo clic su di esso e raggiungendo così a pagina indicata; il processo può essere ripetuto indefinitamente. L'idea di fare in nodo che ogni pagina puntasse a un'altra (**l'ipertesto**) venne a un visionario professore di ingegneria elettronica del M.I.T., Vannevar Bush, nel 1945, molto prima che fosse inventata Internet.

Le pagine sono visualizzate con un programma chiamato **browser**, di cui Internet Explorer e Netscape Navigator sono le forme più popolari. Il browser preleva la pagina richiesta, interpreta il testo e i comandi di formattazione, quindi visualizza la pagina correttamente formattata sullo schermo. Un esempio è mostrato nella Figura 7.18(a). Come molte pagine Web, questa inizia con un titolo, contiene alcune informazioni e termina con l'indirizzo e-mail di chi gestisce la pagina. Le stringhe di testo che sono collegamenti ad altre pagine, o **collegamenti ipertestuali**, sono spesso evidenziate da una sottolineatura, da un colore speciale o entrambe le cose. Per seguire un collegamento, l'utente posiziona il puntatore del mouse sull'area evidenziata: il puntatore cambia forma e gli può fare clic. Esistono browser non grafici, come Lynx, ma non sono popolari quando i browser grafici, pertanto ci concentreremo su questi ultimi. Sono in fase di sviluppo anche browser basati sulla voce.

Gli utenti interessati al dipartimento di psicologia animale possono saperne di più accedendo clic sul suo nome (sottolineato). Il browser recupera la pagina collegata al

BENVENUTI NELLA HOME PAGE DELL'UNIVERSITÀ DI EAST PODUNK

- Informazioni sul campus
 - [Informazioni sull'ammissione](#)
 - [Mappa del campus](#)
 - [Direttive per il campus](#)
 - [Il corpo studentesco UEP](#)

- Dipartimenti accademici
 - [Dipartimento di psicologia animale](#)
 - [Dipartimento di studi alternativi](#)
 - [Dipartimento di cucina microbiotica](#)
 - [Dipartimento di studi non tradizionali](#)
 - [Dipartimento di studi tradizionali](#)

Webmaster@eastpodunk.edu

(a)

DIPARTIMENTO DI PSICOLOGIA ANIMALE

- [Informazioni generiche](#)
- Personale
 - [Membri della facoltà](#)
 - [Studenti laureati](#)
 - [Personale non accademico](#)
- Progetti di ricerca
- Posizioni disponibili
- I corsi più popolari
 - [Lavorare con gli erbivori](#)
 - [Gestione dei cavalli](#)
 - [Trattare con gli animali domestici](#)
 - [Costruzione di cuccie per cani](#)
 - [Elenco completo dei corsi](#)

Webmaster@animalpsych.eastpodunk.edu

(b)

Figura 7.18. (a) Una pagina Web. (b) La pagina raggiunta facendo clic su [Dipartimento di psicologia animale](#).

nome e la visualizza, come mostrato nella Figura 7.18(b). Naturalmente è possibile fare clic sugli elementi sottolineati per raggiungere altre pagine. La nuova pagina può trovarsi sulla stessa macchina della prima o su una macchina dall'altra parte del globo; l'utente non può saperlo. Il recupero della pagina è svolto dal browser, senza alcun aiuto da parte dell'utente. Se l'utente torna alla pagina principale, i collegamenti già

visitati potrebbero essere visualizzati con una sottolineatura tratteggiata (e eventualmente in un colore diverso) per distinguerli dai collegamenti che non sono stati visitati. Occorre notare che la riga *Informazioni sul campus* nella pagina principale non ha alcuna funzione. Non è sottolineata, pertanto si tratta di testo normale che non è collegato ad altre pagine.

Il modello di base del funzionamento del Web è mostrato nella Figura 7.19. Il browser visualizza una pagina Web sulla macchina client. Quando l'utente fa clic su una riga di testo collegata a una pagina sul server *abcd.com*, il browser segue il collegamento ipertestuale inviando un messaggio al server *abcd.com* per richiedere la pagina. All'arrivo della pagina, questa viene visualizzata. Se la pagina contiene un collegamento ipertestuale a una pagina sul server *xyz.com* e l'utente vi fa clic, il browser invia una richiesta a tale macchina, proseguendo così indefinitamente.

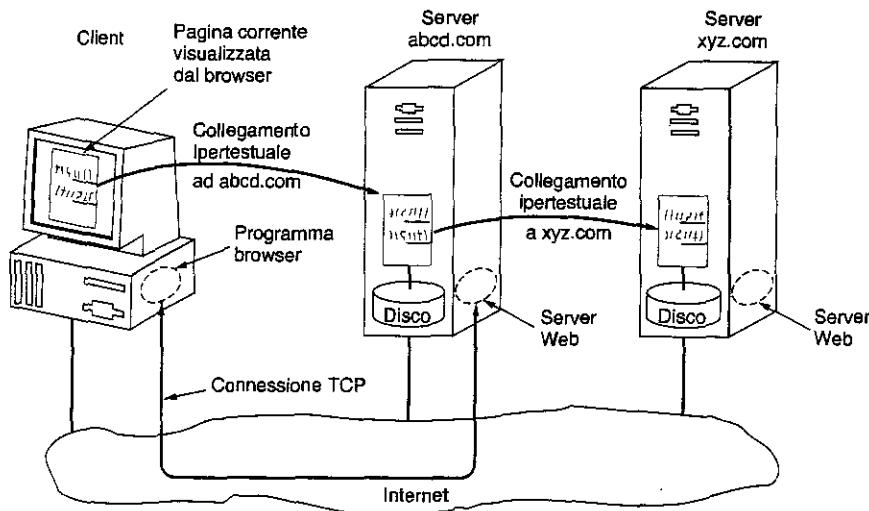


Figura 7.19. Le parti del modello Web.

Il lato client

Esaminiamo ora in dettaglio il lato client della Figura 7.19. In pratica, un browser è un programma che può visualizzare una pagina Web e rilevare i clic del mouse sugli elementi della pagina visualizzata. Quando viene selezionato un elemento, il browser segue il collegamento ipertestuale e preleva la pagina selezionata. Di conseguenza, il collegamento ipertestuale contenuto nella pagina ha necessità di un metodo per fare riferimento alle altre pagine sul Web. Le pagine sono denominate utilizzando gli **URL** (*Uniform Resource Locator*). Un URL tipico è

<http://www.abcd.com/products.html>

Lo strato applicazione

Spiegheremo gli URL in seguito in questo capitolo. Per il momento, è sufficiente sapere che un URL è composto di tre parti: il nome del protocollo (*http*), il nome DNS della macchina su cui è situata la pagina (*www.abcd.com*) e solitamente il nome del file contenente la pagina (*products.html*).

Quando un utente fa clic su un collegamento ipertestuale, il browser esegue una serie di passaggi per recuperare la pagina indicata. Supponiamo che un utente stia esplorando il Web e trovi un collegamento sulla telefonia Internet relativo alla home page di ITU, il cui indirizzo è *http://www.itu.org/home/index.html*. Vediamo i passaggi seguiti alla selezione di questo collegamento.

1. Il browser determina l'URL (osservando che cosa è stato selezionato).
2. Il browser chiede a DNS l'indirizzo IP di *www.itu.org*.
3. DNS risponde con 156.106.192.32.
4. Il browser esegue una connessione TCP alla porta 80 su 156.106.192.32.
5. Invia quindi una richiesta per il file */home/index.html*.
6. Il server *www.itu.org* invia il file */home/index.html*.
7. La connessione TCP viene rilasciata.
8. Il browser visualizza tutto il testo in */home/index.html*.
9. Il browser preleva e visualizza tutte le immagini in questo file.

Molti browser visualizzano il passaggio che stanno eseguendo in una riga di stato nella parte inferiore dello schermo. In questo modo, se le prestazioni sono scadenti, l'utente può vedere se la causa è DNS che non risponde, il server che non risponde o una semplice congestione della rete durante la trasmissione della pagina.

Per essere in grado di visualizzare la nuova pagina (o qualsiasi pagina), il browser deve comprenderne il formato. Per consentire a tutti i browser di comprendere tutte le pagine, le pagine Web sono scritte in un linguaggio standardizzato chiamato HTML, che descrive le pagine Web. Ne parleremo in dettaglio in seguito nel capitolo.

Anche se un browser è fondamentalmente un interprete HTML, la maggior parte offre pulsanti e funzioni per facilitare l'esplorazione del Web: un pulsante per tornare alla pagina precedente, un pulsante per passare alla successiva (operativo solo se l'utente è prima tornato indietro di una pagina) e un pulsante per passare direttamente alla pagina iniziale predefinita. Molti browser hanno un pulsante o una voce di menu per impostare un segnalibro su una data pagina, e un altro per visualizzare l'elenco dei segnalibri, consentendo di rivisitare le pagine con pochi clic del mouse. Le pagine possono anche essere salvate su disco o stampate. Generalmente sono disponibili molte opzioni per controllare il layout dello schermo e impostare le preferenze dell'utente.

Oltre a disporre di testo ordinario (non sottolineato) e collegamenti ipertestuali (sottolineati), le pagine Web possono anche contenere icone, disegni vettoriali, mappe e fotografie.

fie. Ogni elemento può facoltativamente essere collegato a un'altra pagina. Facendo clic su uno di questi elementi il browser preleva la pagina collegata e la visualizza sullo schermo, proprio come avviene per il testo. Con immagini come mappe e fotografie, la pagina visualizzata in seguito dipende dalla parte dell'immagine selezionata.

Non tutte le pagine contengono HTML. Una pagina può consistere solo di un documento formattato in formato PDF, di un'icona in formato GIF, di una fotografia in formato JPEG, di un brano in formato MP3, di un video in formato MPEG o di centinaia di altri tipi di file. Dal momento che le pagine HTML standard possono collegarsi a tutti questi elementi, il browser può avere problemi quando incontra una pagina che non è in grado di interpretare.

Anziché rendere i browser sempre più grandi incorporandovi interpreti per una raccolta di tipi di file in rapida crescita, la maggior parte dei produttori ha scelto una soluzione più generica. Quando un server restituisce una pagina, comunica anche alcune informazioni aggiuntive, che comprendono il tipo MIME della pagina (Figura 7.12). Le pagine di tipo *text/html* sono visualizzate direttamente, come le pagine di altri tipi incorporati. Se il tipo MIME non corrisponde a uno di quelli incorporati, il browser consulta la sua tabella di tipi MIME per capire come visualizzare la pagina. Questa tabella associa un tipo MIME a un visualizzatore.

Esistono due possibilità: plug-in e applicazioni helper. Un **plug-in** è un modulo di codice che il browser preleva da una directory speciale sul disco e installa come estensione a se stesso, come illustrato nella Figura 7.20(a). Dal momento che i plug-in sono eseguiti nel browser, hanno accesso alla pagina corrente e possono modificare il suo aspetto. Quando il plug-in ha svolto il suo lavoro (solitamente dopo che l'utente è passato a una pagina Web diversa), il plug-in viene rimosso dalla memoria del browser.

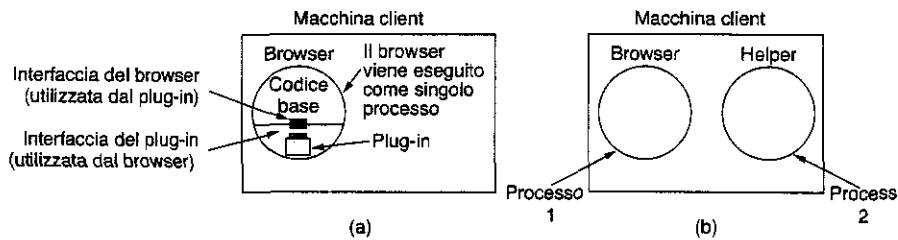


Figura 7.20. (a) Un plug-in del browser. (b) Un'applicazione helper.

Ogni browser possiede un set di procedure che tutti i plug-in devono implementare affinché il browser possa chiamare il plug-in. Per esempio, solitamente il codice base del browser chiama una procedura per fornire al plug-in i dati da visualizzare. Questo set di procedure è l'**interfaccia del plug-in** ed è specifico del browser.

Inoltre, il browser crea un set di proprie procedure per il plug-in, al fine di fornirgli servizi. Le procedure tipiche nell'interfaccia del browser riguardano l'allocazione e la liberazione della memoria, la visualizzazione di un messaggio sulla riga di stato del browser, e l'interrogazione del browser in merito ai parametri.

Prima di poter utilizzare un plug-in, occorre installarlo. L'abituale procedura d'installa-

zione prevede che l'utente si colleghi al sito Web del plug-in e scarichi un file di installazione. Su Windows, si tratta generalmente di un file zip autoestraente con estensione *.exe*. Dopo avere fatto doppio clic sul file zip, viene eseguito un piccolo programma allegato al file zip. Questo programma decomprime il plug-in e lo copia nella directory plug-in del browser. Esegue quindi le chiamate appropriate per registrare il tipo MIME del plug-in e per associarvi il plug-in. Su UNIX, l'installatore è spesso uno script della shell che gestisce la copia e la registrazione.

L'altro modo per estendere un browser consiste nell'utilizzare un **applicazione helper**. Si tratta di un programma completo, eseguito come processo separato, e illustrato nella Figura 7.20(b). Dal momento che l'helper è un programma separato, non offre un'interfaccia al browser e non utilizza i servizi del browser. Al contrario, solitamente accetta il nome di un file temporaneo, lo apre e ne visualizza il contenuto. Generalmente, gli helper sono grandi programmi che esistono in modo indipendente dal browser, come Adobe Acrobat Reader per la visualizzazione dei file PDF o Microsoft Word. Alcuni programmi (come Acrobat) dispongono di un plug-in che invoca l'helper stesso.

Molte applicazioni helper utilizzano il tipo MIME *application*. È stato definito un numero considerevole di sottotipi, per esempio *application/pdf* per i file PDF e *application/msword* per i file di Word. In questo modo, un URL può puntare direttamente a un file di Word o PDF; quando l'utente fa clic, vengono automaticamente avviati Acrobat o Word, cui viene passato il nome del file temporaneo con il contenuto da visualizzare. Di conseguenza, i browser possono essere configurati per gestire un numero virtualmente illimitato di tipi di documenti senza bisogno di modifiche. I server Web moderni sono spesso configurati con centinaia di combinazioni tipo/sottotipo, e ne vengono aggiunte di nuove ogni volta che viene installato un nuovo programma.

Le applicazioni helper non devono necessariamente utilizzare il tipo MIME *application*. Adobe Photoshop utilizza *image/x-photoshop* mentre RealOne Player è in grado di gestire *audio/mp3*, per esempio.

Su Windows, quando viene installato un programma sul computer, questo registra i tipi MIME che desidera gestire. Il meccanismo porta a dei conflitti quando sono disponibili più visualizzatori per un sottotipo, come *video/mpg*. In pratica, l'ultimo programma registrato sovrascrive le associazioni esistenti (tipo MIME, applicazione helper), acquisendo la gestione del tipo. Di conseguenza, l'installazione di un nuovo programma può cambiare il modo in cui un browser gestisce i tipi esistenti.

Su UNIX, questo processo di registrazione generalmente non è automatico. L'utente deve aggiornare manualmente alcuni file di configurazione. Questo approccio richiede più lavoro ma genera meno sorprese.

I browser possono anche aprire i file locali, anziché prelevarli da server Web remoti. Dal momento che i file locali non possiedono tipi MIME, il browser necessita di un modo per determinare quale plug-in o helper utilizzare per i tipi diversi da quelli incorporati come *text/html* e *image/jpeg*. Per gestire i file locali, gli helper possono essere associati a un'estensione di file oltre che a un tipo MIME. Nella configurazione standard, l'apertura di *foo.pdf* avvia Acrobat mentre l'apertura di *bar.doc* avvia Word. Alcuni browser utilizzano

il tipo MIME, l'estensione del file e persino le informazioni prese dal file stesso per indovinare il tipo MIME. In particolare, Internet Explorer fa affidamento più sull'estensione del file che sul tipo MIME, quando ha a disposizione entrambi.

Anche qui possono sorgere conflitti, perché molti programmi desiderano (o meglio bramano) appropriarsi della gestione di particolari file, per esempio *.mpg*. Durante l'installazione, i programmi destinati ai professionisti spesso visualizzano caselle di controllo per i tipi MIME e le estensioni che sono in grado di gestire, consentendo all'utente di scegliere quelle appropriate e di non sovrascrivere accidentalmente le associazioni esistenti. I programmi mirati al mercato dei consumatori presumono che l'utente non sappia che cos'è un tipo MIME, e acquisiscono tutto ciò che possono senza riguardo per quanto fatto dai programmi installati in precedenza.

La capacità di estendere il browser con un elevato numero di nuovi tipi è conveniente, ma porta anche alcuni problemi. Quando Internet Explorer preleva un file con estensione *exe*, comprende che questo file è un programma eseguibile e di conseguenza non ha un helper. L'azione ovvia è eseguire il programma. Tuttavia, questo potrebbe portare un enorme buco nella protezione. Un sito Web maligno potrebbe produrre una pagina Web con immagini di stelle dello spettacolo o dello sport, collegate a un virus. Un singolo clic su un'immagine potrebbe provocare lo scaricamento del file e l'esecuzione sul computer dell'utente di un programma eseguibile sconosciuto e potenzialmente ostile. Per impedire ospiti indesiderati come questo, Internet Explorer può essere configurato per segnalare l'esecuzione di programmi sconosciuti, ma non tutti gli utenti sanno come gestire la configurazione. Su UNIX può esistere un problema analogo con gli script della shell, ma l'utente deve installare volontariamente la shell come helper. Fortunatamente, questa installazione è abbastanza complicata da scongiurare la possibilità che qualcuno la esegua accidentalmente (e ben poche persone possono farlo intenzionalmente).

Il lato server

Finora abbiamo parlato del lato client; adesso osserviamo il lato server. Come affermato in precedenza, quando l'utente digita un URL o fa clic su una riga dell'ipertesto, il browser analizza sintatticamente l'URL e interpreta la parte tra *http://* e la barra successiva come il nome DNS da ricercare. Armato dell'indirizzo IP del server, il browser stabilisce una connessione TCP sulla relativa porta 80. Invia poi un comando contenente il resto dell'URL, vale a dire il nome di un file sul server, che quindi restituisce il file per la visualizzazione da parte del browser.

In prima approssimazione, un server Web è simile al server della Figura 6.6. A tale server, come un vero server Web, viene fornito il nome di un file da cercare e restituire. In entrambi i casi, i passaggi eseguiti dal server nel suo ciclo principale sono i seguenti:

1. accettare una connessione TCP da un client (un browser)
2. ottenere il nome del file richiesto
3. ottenere il file (dal disco)

4. restituire il file al client
5. rilasciare la connessione TCP.

I server Web moderni hanno anche altre funzionalità, ma in sostanza le operazioni di un server Web sono queste.

Un problema di questa struttura riguarda il fatto che ad ogni richiesta deve essere eseguito un accesso al disco per recuperare il file. Il risultato è che il server Web non può servire un numero di richieste al secondo superiore al numero degli accessi al disco. Un disco SCSI high-end ha un tempo di accesso medio di circa 5 msec, che limita il server a circa 200 richieste al secondo (o meno, se devono essere letti spesso file grandi). Per un sito Web grande, questo valore è troppo ridotto.

Un miglioramento ovvio (utilizzato da tutti i server Web) consiste nel mantenere in memoria in una cache gli *n* file utilizzati più di recente. Prima di accedere al disco per recuperare un file, il server controlla la cache. Se il file è presente, può essere servito direttamente dalla memoria eliminando l'accesso al disco. Anche se l'utilizzo effettivo della cache richiede un'elevata quantità di memoria principale e un tempo di elaborazione extra per controllare la cache e gestirne il contenuto, il risparmio in termini di tempo vale sempre l'overhead e la spesa.

Il passaggio successivo per la costruzione di un server veloce richiede di rendere il server multithreaded. In una possibile struttura di questo tipo, il server consiste di un modulo front-end che accetta tutte le richieste in ingresso e di *k* moduli di elaborazione, mostrati nella Figura 7.21. I *k* + 1 thread appartengono tutti allo stesso processo, pertanto tutti i moduli di elaborazione hanno accesso alla cache all'interno dello stesso spazio di indirizzamento del processo. Quando arriva una richiesta, il front-end la accetta e costruisce un breve record descrittivo. Trasmette quindi il record a uno dei moduli di elaborazione. Una struttura alternativa elimina il front-end e ogni modulo di elaborazione cerca di acquisire le proprie richieste; è tuttavia necessario un protocollo di blocco per impedire i conflitti.

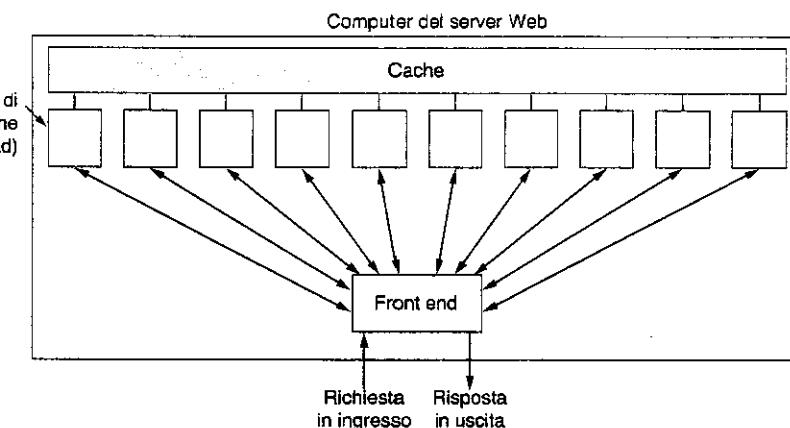


Figura 7.21. Un server Web multithreaded con un front-end e moduli di elaborazione.

Il modulo di elaborazione controlla per prima cosa la cache per vedere se contiene il file necessario. In questo caso, aggiorna il record per includervi un puntatore al file. Se non è presente, il modulo di elaborazione avvia un'operazione su disco per la lettura (scartando altri file nella cache, per creare spazio). Quando il file viene recuperato dal disco, viene inserito nella cache e restituito al client.

Il vantaggio di questo schema è il seguente: mentre uno o più moduli di elaborazione sono bloccati in attesa del completamento di un'operazione su disco (senza consumare tempo della CPU), altri moduli possono lavorare attivamente su altre richieste. Naturalmente, per ottenere reali miglioramenti rispetto al modello a thread singolo, è necessario disporre di più dischi, in modo che nello stesso tempo si possano avere più dischi occupati. Con k moduli di elaborazione e k dischi, la velocità dei dati può essere k volte superiore a quella ottenibile con un server monodisco a thread singolo.

In teoria, anche un server single-threaded e k dischi possono far guadagnare un fattore k di prestazioni, ma il codice e l'amministrazione sono più complicati perché non permettono di utilizzare le chiamate di sistema READ per accedere al disco. Con un server multithreaded, è possibile utilizzarle perché READ blocca solo il thread che ha eseguito la chiamata, non l'intero processo.

I server Web moderni svolgono altre operazioni, oltre ad accettare i nomi dei file e a restituire i file, e l'elaborazione effettiva di ogni richiesta può divenire complessa. Per questo motivo, in molti server ogni modulo di elaborazione esegue una serie di passaggi. Il front-end passa ogni richiesta in ingresso al primo modulo disponibile, che la trasporta utilizzando un sottoinsieme dei seguenti passaggi (che dipende da quali sono necessari per esaudire la particolare richiesta):

1. risolvere il nome della pagina Web richiesta
2. autenticare il client
3. eseguire il controllo di accesso sul client
4. eseguire il controllo di accesso sulla pagina Web
5. controllare la cache
6. prelevare la pagina richiesta dal disco
7. determinare il tipo MIME da includere nella risposta
8. occuparsi di questioni varie
9. restituire la risposta al client
10. creare una voce nel registro del server.

Il passaggio 1 è necessario perché la richiesta in ingresso potrebbe non contenere il nome effettivo del file come stringa letterale. Per esempio, si consideri l'URL <http://www.cs.vu.nl>, che presenta un nome file vuoto da espandere in qualche nome file predefinito. Inoltre, i browser moderni possono specificare la lingua predefinita dell'utente (per esempio italiano o inglese), in modo che il server possa selezionare una pagina Web in tale lingua, se è disponibile. In generale l'espansione del nome non è così semplice come può sembrare, a causa di numerose convenzioni sui nomi dei file.

Il passaggio 2 consiste nel verificare l'identità del client. Questo passaggio è necessario per le pagine che non sono disponibili al pubblico. Spiegheremo come procedere in seguito nel capitolo.

Il passaggio 3 controlla se vi sono limitazioni alla soddisfazione della richiesta in base all'identità e alla posizione del client. Il passaggio 4 controlla se vi sono limitazioni di accesso alla pagina stessa. Se è presente un particolare file (per esempio *.htaccess*) nella directory dove è situata la pagina desiderata, questo file potrebbe limitare l'accesso a particolari domini, per esempio solo agli utenti interni alla società.

I passaggi 5 e 6 riguardano il recupero della pagina. Il passaggio 6 deve essere in grado di gestire più letture del disco contemporaneamente.

Il passaggio 7 riguarda la determinazione del tipo MIME dall'estensione del file, dalle prime parole del contenuto, dal file di configurazione e possibilmente da altre fonti. Il passaggio 8 riguarda numerose attività, come la creazione di un profilo utente o la raccolta di dati statistici.

Il passaggio 9 restituisce i risultati mentre il 10 crea una voce nel registro di sistema a scopo amministrativo. Tali registri possono essere utilizzati in seguito per ricavare informazioni preziose sul comportamento degli utenti, per esempio in merito all'ordine in cui le persone accedono alle pagine.

Se ogni secondo giungono troppe richieste, la CPU non sarà in grado di gestire il carico di elaborazione, indipendentemente dal numero di dischi utilizzati in parallelo. La soluzione è aggiungere altri nodi (computer), possibilmente con dischi replicati per evitare che i dischi diventino un collo di bottiglia. Questo porta al modello di **server farm** della Figura 7.22. Un front-end accetta le richieste in ingresso ma le divide su più CPU, anziché su più thread, per ridurre il carico su ogni computer. Le singole macchine possono a loro volta essere multithreaded e organizzate in pipeline come in precedenza.

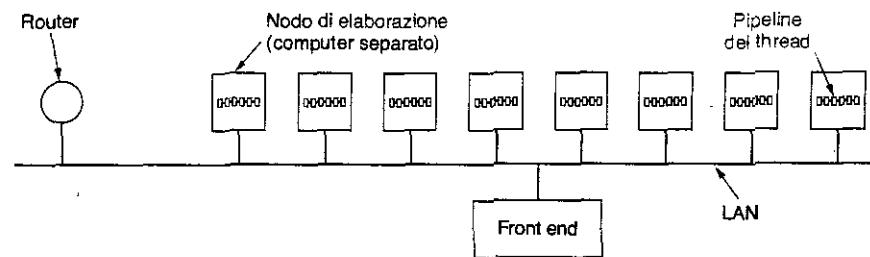


Figura 7.22. Una server farm.

Un problema delle server farm riguarda la mancanza di una cache condivisa, perché ogni nodo di elaborazione possiede la sua memoria (a meno che venga utilizzato un costoso multiprocessore a memoria condivisa). Un modo per contrastare questa perdita di presta-

zioni richiede un front-end che tenga traccia del luogo in cui è stata inviata ogni richiesta, e in seguito invii sempre allo stesso nodo le richieste di quella pagina. Ogni nodo diventa quindi uno specialista per determinate pagine, e lo spazio della cache non viene sprecato dalla presenza di ogni file in ogni cache.

Un altro problema delle server farm riguarda il fatto che la connessione TCP del client termina nel front-end, per cui la risposta deve passare attraverso il front-end. Questa situazione è mostrata nella Figura 7.23(a), dove la richiesta in ingresso (1) e la risposta in uscita (4) passano attraverso il front-end. A volte viene utilizzato un trucco, chiamato **TCP handoff**, per aggirare il problema. Con questo trucco, il punto finale di TCP viene passato al nodo di elaborazione, in modo che possa rispondere direttamente al client, come mostrato nella Figura 7.23(b). Questo handoff viene svolto in modo trasparente rispetto al client.

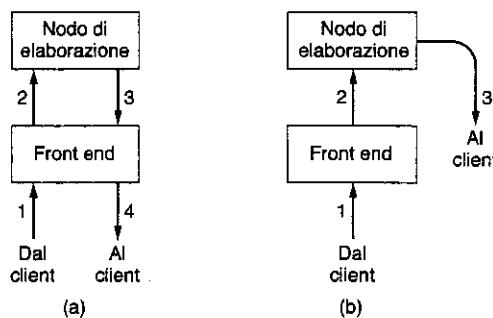


Figura 7.23. (a) Una normale sequenza di messaggi richiesta/risposta.
(b) La sequenza con l'utilizzo di TCP handoff.

URL (Uniform Resource Locator)

Abbiamo ripetuto più volte che le pagine Web possono contenere puntatori ad altre pagine Web. È il momento di vedere più in dettaglio come sono implementati questi puntatori. Quando fu creato il Web, divenne immediatamente chiaro che la disponibilità di pagine che facessero riferimento l'una all'altra richiedeva un meccanismo per denominare e individuare le pagine. In particolare, occorre rispondere a tre domande prima di poter visualizzare una pagina selezionata:

1. com'è chiamata la pagina?
2. dove è situata la pagina?
3. come è possibile accedere alla pagina?

Se a ogni pagina fosse assegnato un nome univoco non vi sarebbero ambiguità nell'identificazione delle pagine, ma il problema non sarebbe comunque risolto. Si consideri un

parallelo tra le persone e le pagine. In Italia, quasi tutti possiedono un codice fiscale, che è un identificatore univoco (due persone non dovrebbero avere mai lo stesso). Ciò nonostante, se si possiede solo il codice fiscale, non c'è modo per individuare l'indirizzo del proprietario e ovviamente non c'è modo di sapere se a questa persona è possibile scrivere in italiano, spagnolo o cinese. Il Web presenta fondamentalmente gli stessi problemi.

La soluzione scelta identifica le pagine in un modo che risolve tutti e tre i problemi contemporaneamente. A ogni pagina è assegnato un **URL (Uniform Resource Locator)** che serve effettivamente come nome mondiale per la pagina. Gli URL sono composti di tre parti: il protocollo (noto anche come **schemma**), il nome DNS della macchina su cui è situata la pagina e un nome locale che indica in modo univoco la pagina specifica (solitamente un nome file sulla macchina in cui risiede). Come esempio, il sito Web del dipartimento dell'autore contiene video sull'università e la città di Amsterdam. L'URL per la pagina del video è

<http://www.cs.vu.nl/video/index-en.html>

Questo URL è composto di tre parti, il protocollo (*http*), il nome DNS dell'host (*www.cs.vu.nl*) e il nome file (*video/index-en.html*), con alcuni segni di punteggiatura che separano i pezzi. Il nome file è un percorso relativo alla directory Web predefinita di *cs.vu.nl*.

Molti siti dispongono di scorciatoie per i nomi file. In molti casi, un nome file mancante corrisponde alla home page principale dell'organizzazione. Generalmente, quando il file indicato è una directory, implica un file denominato *index.html*. Per finire, *~user/* potrebbe essere associato alla directory WWW di *user* e al file *index.html* in tale directory. Di conseguenza, la home page dell'autore può essere raggiunta all'indirizzo

<http://www.cs.vu.nl/~ast>

anche se il nome file effettivo è *index.html* in una determinata directory predefinita. Ora possiamo conoscere il funzionamento dei collegamenti ipertestuali. Per rendere selezionabile un testo, l'autore della pagina deve fornire due informazioni: il testo selezionabile da visualizzare e l'URL della pagina da aprire quando viene selezionato. Spiegheremo la sintassi del comando in seguito in questo capitolo.

Quando il testo è selezionato, il browser cerca il nome host utilizzando DNS. Una volta noto l'indirizzo IP dell'host, il browser stabilisce una connessione TCP all'host. Su tale connessione, invia il nome file utilizzando il protocollo specificato. Bene! Viene restituita la pagina.

Questo schema URL è aperto, nel senso che consente ai browser di utilizzare più protocolli per raggiungere tipi diversi di risorse. In effetti, sono stati definiti URL per altri protocolli comuni. Alcune forme semplificate dei più comuni sono elencate nella Figura 7.24. Vediamo brevemente l'elenco. Il protocollo *http* è il linguaggio nativo del Web, quello "parlato" dai server Web. **HTTP** sta per **HyperText Transfer Protocol**. Lo esamineremo in dettaglio in seguito nel capitolo.

Nome	Utilizzato per	Esempio
http	Ipertesto (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	File locale	file:///usr/suzanne/prog.c
news	Newsgroup	news:comp.os.minix
news	Articolo delle news	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Invio di posta elettronica	mailto:JohnUser@acm.org
telnet	Login remoto	telnet://www.w3.org:80

Figura 7.24. Alcuni URL comuni.

Il protocollo *ftp* è utilizzato per accedere ai file con FTP, il protocollo di trasferimento file di Internet. FTP è disponibile da più di due decenni ed è ben consolidato. Numerosi server FTP in tutto il mondo consentono alle persone ovunque su Internet di accedere e scaricare i file; il Web non cambia questo stato di cose, ma facilita il recupero dei file, visto che FTP ha un'interfaccia in un certo senso arcana (ma più potente di HTTP, per esempio perché permette a un utente su una macchina *A* di trasferire un file dalla macchina *B* alla macchina *C*). È possibile accedere a un file locale come se fosse una pagina Web, sia utilizzando il protocollo *file* sia, più semplicemente, indicando solo il suo nome. Questo approccio è simile all'utilizzo di FTP ma non richiede la disponibilità di un server. Naturalmente, funziona solo per i file locali, non per quelli remoti. Molto tempo prima di Internet esisteva il sistema di news USENET. Consiste di circa 30.000 newsgroup in cui milioni di persone parlano di numerosi argomenti inviando e leggendo articoli correlati all'argomento del newsgroup. Il protocollo *news* può essere utilizzato per richiamare un articolo delle news come se fosse una pagina Web. Questo significa che un browser Web è allo stesso tempo anche un lettore di news. In effetti, molti browser dispongono di pulsanti o voci di menu per facilitare la lettura delle news USENET, in modo più completo dei lettori di news standard.

Per il protocollo *news* sono supportati due formati. Il primo specifica un newsgroup e può essere utilizzato per ottenere un elenco di articoli da un sito delle news predefinito. Il secondo richiede l'identificatore di un articolo delle news specifico, in questo caso AA0134223112@cs.utah.edu. Il browser preleva quindi l'articolo dato dal suo sito delle news predefinito utilizzando NNTP (*Network News Transfer Protocol*). Non studieremo NNTP in questo libro, comunque è ampiamente basato su SMTP e ha uno stile simile.

Il protocollo *gopher* è utilizzato dal sistema Gopher, progettato dall'università del Minnesota e denominato in base alla squadra di atletica della scuola, i Golden Gophers (è un'espressione gergale per "go for", che in italiano significa "vai per"). Gopher ha monopolizzato il Web per anni. Si tratta di uno schema di recupero delle informazioni, concettualmente simile al Web stesso, ma che supporta solo il testo e non le immagini. Ora è essenzialmente obsoleto e utilizzato raramente.

Lo strato applicazione

Gli ultimi due protocolli non permettono di recuperare le pagine Web, ma sono comunque utili. Il protocollo *mailto* consente agli utenti di inviare posta elettronica da un browser Web. Per farlo è sufficiente fare clic sul pulsante Apri e specificare un URL costituito da *mailto:* seguito dall'indirizzo di posta elettronica del destinatario. La maggior parte dei browser risponde avviando un programma di posta elettronica con l'indirizzo e alcuni campi di intestazione già compilati.

Il protocollo telnet viene utilizzato per stabilire una connessione online a una macchina remota. È utilizzato nello stesso modo di un programma telnet: questo non deve sorprendere, perché la maggior parte dei browser chiama il programma telnet come applicazione helper.

In breve, gli URL sono stati progettati non solo per consentire agli utenti di esplorare il Web, ma anche per gestire FTP, news, Gopher, posta elettronica e telnet, rendendo inutili tutti i programmi specializzati di interfaccia utente per gli altri servizi e integrando tutte le funzioni di accesso a Internet in un singolo programma, il browser Web. Se non fosse per il fatto che questa idea deriva da un ricercatore di fisica, potrebbe tranquillamente sembrare il prodotto del reparto pubblicitario di una società di software.

Nonostante tutte queste belle proprietà, l'utilizzo crescente del Web ha evidenziato una debolezza nello schema degli URL. Un URL fa riferimento a un host specifico. Per le pagine visitate in modo "pesante", è utile disporre di più copie ben distanti per ridurre il traffico di rete. Il problema è che gli URL non forniscono un modo per fare riferimento a una pagina senza indicare contemporaneamente dove si trova. Non c'è modo per affermare: "Voglio la pagina xyz, ma non importa da dove viene presa". Per risolvere questo problema e consentire la replica delle pagine, IETF sta lavorando su un sistema di URN (*Universal Resource Name*). Un URN può essere visto come una sorta di URL generalizzato. Questo argomento è ancora oggetto di ricerca, anche se una proposta di sintassi è fornita in RFC 2141.

La mancanza di stato e i cookie

Come abbiamo detto ripetutamente, il Web è fondamentalmente privo di stato. Non esiste il concetto di sessione di login. Il browser invia una richiesta a un server e ottiene il file, dopo di che il server dimentica per sempre di avere visto quel particolare client.

All'inizio, quando il Web era utilizzato solo per recuperare documenti disponibili pubblicamente, questo modello era perfettamente adeguato. Tuttavia, quando il Web ha iniziato ad acquisire altre funzioni, ha provocato problemi. Per esempio, alcuni siti Web richiedono ai clienti di registrarsi (e magari di pagare) per l'utilizzo. Come possono i server distinguere tra le richieste degli utenti registrati e quelle di altri utenti? Un secondo esempio è relativo all'e-commerce. Se gli utenti visitano un negozio elettronico, aggiungendo elementi al carrello di volta in volta, come può il server tenere traccia del contenuto del carrello? Un terzo esempio è dato dai portali Web personalizzati come Yahoo! Gli utenti possono impostare una pagina iniziale dettagliata con le informazioni desiderate (per esempio le loro azioni e le loro squadre sportive preferite); come può il server visualizzare la pagina corretta se non sa chi è l'utente?

A prima vista si potrebbe pensare che i server possano tenere traccia degli utenti osservandone gli indirizzi IP. Tuttavia, questa idea non funziona. Prima di tutto, molti utenti lavorano su computer condivisi, specialmente in azienda: gli indirizzi IP, quindi, identificano solamente il computer, non l'utente. In secondo luogo, molti ISP utilizzano NAT, pertanto tutti i pacchetti in uscita provenienti da tutti gli utenti contengono lo stesso indirizzo IP. Dal punto di vista del server, tutte le centinaia di clienti dell'ISP utilizzano lo stesso indirizzo IP.

Per risolvere questo problema, Netscape ha proposto una tecnica molto criticata: i **cookie**. Il nome deriva da un antico gergo dei programmati, in cui un programma chiama una procedura e ottiene qualcosa che potrebbe in seguito dover presentare per svolgere un lavoro. In questo senso, un descrittore di file UNIX o l'handle di un oggetto Windows possono essere considerati cookie. I cookie sono stati formalizzati in RFC 2109.

Quando un client richiede una pagina Web, il server può fornire informazioni aggiuntive insieme alla pagina richiesta. Questa informazione può includere un cookie, vale a dire un piccolo file (o stringa) che non supera i 4 KB. I browser archiviano i cookie in un'apposita directory sul disco rigido del client, a meno che l'utente non li abbia disattivati. I cookie sono solo file o stringhe, non programmi eseguibili. Un cookie potrebbe teoricamente contenere un virus, ma visto che i cookie sono trattati come dati, non esiste un modo ufficiale per mandare in esecuzione l'ipotetico virus e quindi causare danni. Tuttavia, non si può escludere che qualche hacker sfrutti un bug del browser per provocare l'attivazione.

Un cookie può contenere fino a cinque campi, come mostrato nella Figura 7.25. **Domain** (dominio) indica da dove proviene il cookie. I browser dovrebbero verificare che i server non mentano in merito al loro dominio. Su ogni client non possono essere mantenuti un numero superiore a venti cookie provenienti dallo stesso dominio. **Path** (percorso) è un percorso nella struttura delle directory del server che identifica quale parte della struttura di file del server può utilizzare il cookie. Spesso corrisponde a /, che indica l'intera struttura.

Dominio	Percorso	Contenuto	Scadenza	Sicuro
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	Sì
joes-store.com	/	Cart=1-00501;1-07031;2-13721	11-10-02 14:22	No
aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jets	31-12-10 23:59	No
sneaky.com	/	UserID=3627239101	31-12-12 23:59	No

Figura 7.25. Alcuni esempi di cookie.

tura.

Il campo **content** (contenuto) assume la forma **nome = valore**. **Nome** e **valore** possono assumere qualsiasi valore scelto dal server; qui viene archiviato il contenuto del cookie. Il campo **expires** (scadenza) specifica quando scade il cookie. Se il campo è assente, il browser scarta il cookie alla chiusura; un cookie di questo tipo è definito **cookie non persistente**. Se vengono fornite una data e un'ora, il cookie è **persistente** ed è conservato fino

alla scadenza. I tempi di scadenza sono forniti secondo l'ora di Greenwich. Per rimuovere un cookie dal disco rigido del client, un server deve inviarlo di nuovo ma con un tempo di scadenza già trascorso.

Per finire, il campo **secure** (sicuro) può essere impostato per indicare che il browser può restituire il cookie solo a un server sicuro. Questa funzionalità è utilizzata per l'e-commerce, il remote banking e altre applicazioni sicure.

Abbiamo visto come sono acquisiti i cookie, ma come vengono utilizzati? Il browser, prima di inviare una richiesta per una pagina di qualche sito Web, controlla nella directory dei cookie se sono presenti cookie inseriti dal dominio a cui rivolge la richiesta. In questo caso, tutti i cookie inseriti da tale dominio vengono inclusi nel messaggio di richiesta. Quando il server li riceve, può interpretarli come desiderato.

Esaminiamo alcuni possibili utilizzi dei cookie. Nella Figura 7.25, il primo cookie è impostato da *toms-casino.com* ed è utilizzato per identificare il cliente. Quando il cliente esegue il login la settimana successiva per buttare via altri soldi, il browser invia il cookie in modo che il server riconosca l'utente. Armato dell'ID del cliente, il server può ricercare il suo record in un database e utilizzare l'informazione per costruire una pagina Web appropriata. A seconda delle abitudini del cliente, la pagina potrebbe presentare una mano di poker, un elenco delle corse ippiche del giorno o una slot machine.

Il secondo cookie proviene da *joes-store.com*. In questo caso lo scenario è un cliente che visita il negozio, cercando gli articoli da acquistare. Quando trova un affare e fa clic su di esso, il server costruisce un cookie contenente la quantità e il codice del prodotto, quindi lo restituisce al client. Il cookie viene nuovamente restituito a ogni richiesta di pagina durante la visita del negozio; all'accumularsi di altri acquisti, il server li aggiunge al cookie. Nella figura il carrello contiene tre articoli, dove l'ultimo è richiesto in due esemplari. Per finire, quando il cliente fa clic su *Procedi con il pagamento*, il cookie (che ora contiene l'intero elenco di acquisti) viene inviato insieme alla richiesta. In questo modo il server sa esattamente che cosa è stato acquistato.

Il terzo cookie è per un portale Web. Quando il cliente fa clic su un collegamento del portale, il browser invia il cookie, che comunica al portale di costruire una pagina contenente i prezzi delle azioni per Sun Microsystems e Oracle, nonché i risultati dei New York Jets. Dal momento che un cookie può raggiungere una dimensione di 4 KB, c'è molto spazio per preferenze dettagliate relativi a titoli di giornale, previsioni del tempo, offerte speciali e così via.

I cookie possono essere utilizzati anche a vantaggio del server. Per esempio, supponiamo che un server voglia tenere traccia del numero di visitatori unici e del numero di pagine osservate da ognuno prima di lasciare il sito. Quando giunge la prima richiesta, questa non è accompagnata da un cookie, pertanto il server restituisce un cookie contenente **counter = 1**. I clic successivi in tale sito restituiranno il cookie al server, e ogni volta il contatore viene incrementato e restituito al client. Tenendo traccia dei contatori, il server può sapere quante persone hanno abbandonato il sito dopo avere visto la prima pagina, quanti hanno guardato due pagine e così via.

I cookie si prestano anche ad utilizzi abusivi. I cookie dovrebbero essere restituiti solo al sito di origine, ma gli hacker hanno sfruttato numerosi bug nei browser per acquisire cookie non destinati a loro. Dal momento che alcuni siti di e-commerce pongono i numeri delle carte di credito nei cookie, il potenziale dell'abuso è chiaro.

Un utilizzo controverso dei cookie riguarda la raccolta segreta di informazioni sulle abitudini di esplorazione del Web degli utenti. Funziona in questo modo. Un'agenzia di pubblicità, per esempio Sneaky Ads, contatta i principali siti Web e pone banner pubblicitari per i prodotti delle aziende partner sulle loro pagine, pagando una piccola quota ai proprietari del sito. Anziché fornire al sito un file GIF o JPEG da inserire in ogni pagina, comunicano un URL da aggiungere alle pagine. Ogni URL contiene un numero univoco nella parte relativa al file, per esempio

<http://www.sneaky.com/382674902342.gif>

Quando un utente visita per la prima volta una pagina *P* contenente tale annuncio, il browser preleva il file HTML. Il browser analizza quindi il file HTML e trova il collegamento al file d'immagine su www.sneaky.com, pertanto invia una richiesta per l'immagine. Viene restituito un file GIF insieme a un cookie che contiene un ID utente univoco (3627239101 nella Figura 7.25). Sneaky in questo modo registra il fatto che l'utente con questo ID ha visitato la pagina *P*; ciò è facilmente realizzabile perché il file richiesto (*382674902342.gif*) è referenziato solo dalla pagina *P*. Naturalmente, l'annuncio effettivo può apparire su centinaia di pagine, ma ogni volta con un nome file diverso. Sneaky probabilmente raccoglie un paio di centesimi dal produttore dell'articolo ogni volta che mostra l'annuncio.

In seguito, quando l'utente visita un'altra pagina Web contenente uno degli annunci di Sneaky, il browser preleva il file HTML dal server, vede il collegamento a <http://www.sneaky.com/493654919923.gif> (per esempio) e richiede il file. Visto che possiede già un cookie dal dominio *sneaky.com*, il browser include il cookie di Sneaky contenente l'ID utente. Sneaky ora conosce una seconda pagina visitata dall'utente.

Nel tempo, Sneaky può costruire un profilo completo delle abitudini di esplorazione dell'utente, anche se l'utente non ha mai fatto clic su uno degli annunci. Naturalmente non possiede ancora il nome dell'utente, ma dispone dell'indirizzo IP, che può essere sufficiente per dedurre il nome da altri database. Se l'utente fornisce il suo nome a un qualsiasi sito che coopera con Sneaky, diventa disponibile un profilo completo di nome che viene venduto a chiunque voglia acquistarlo. La vendita di queste informazioni può essere abbastanza proficua da consentire a Sneaky di inserire altri annunci su più siti Web, e quindi di raccogliere più informazioni. La parte più insidiosa di questo business è che la maggior parte degli utenti è completamente inconsapevole della raccolta di informazioni, e può pensare di essere al sicuro perché non fa clic sugli annunci.

Se Sneaky volesse davvero essere meschina, l'annuncio non apparirebbe nemmeno come un classico banner. Un "annuncio" costituito da un singolo pixel nel colore di sfondo (e quindi invisibile) ha esattamente lo stesso effetto di un banner pubblicitario: richiede al

browser di prelevare l'immagine GIF di 1 x 1 pixel e di inviare tutti i cookie originati dal dominio del pixel.

Per mantenere una parvenza di privacy, alcuni utenti configurano i loro browser per rifiutare tutti i cookie, ma questa scelta provoca problemi con i siti Web legittimi che utilizzano cookie. Per risolvere il problema, a volte gli utenti installano software di eliminazione dei cookie. Si tratta di programmi speciali che esaminano ogni cookie in ingresso dopo l'arrivo, e lo accettano o lo scartano in base alle scelte dell'utente (per esempio se il sito Web è considerato di fiducia). L'utente dispone quindi di un controllo preciso sui cookie accettati e rifiutati; i browser moderni come Mozilla (www.mozilla.org) offrono di serie sofisticate regolazioni per i cookie.

7.3.2 Documenti Web statici

La base del Web è il trasferimento di pagine Web dal server al client. Nella forma più semplice le pagine Web sono statiche, cioè sono file che risiedono sul server in attesa di essere recuperati. In questo contesto, anche un video è una pagina Web statica, perché si tratta solamente di un file. In questa sezione osserveremo le pagine Web statiche in dettaglio; nella successiva esamineremo il contenuto dinamico.

HTML (*HyperText Markup Language*)

Oggi le pagine Web sono scritte in un linguaggio chiamato **HTML (*HyperText Markup Language*)**. HTML consente agli utenti di produrre pagine Web che contengono testo, grafica, e puntatori ad altre pagine Web. HTML è un linguaggio di markup, ovvero un linguaggio per descrivere come devono essere formattati i documenti. Il termine "markup" deriva dai tempi antichi, quando gli impaginatori contrassegnavano (marking-up) i documenti per comunicare allo stampatore quali tipi di caratteri utilizzare e così via. I linguaggi di markup contengono quindi comandi esplicativi di formattazione. Per esempio, in HTML, **** indica l'inizio della modalità grassetto, mentre **** indica di terminare l'utilizzo del grassetto. Il vantaggio di un linguaggio di markup su un linguaggio senza markup esplicito sta nella semplicità con cui è possibile scrivere un browser per esso: il browser, infatti, deve semplicemente comprendere i comandi di markup. TeX e troff sono altri esempi ben noti di linguaggi di markup.

Incorporando tutti i comandi di markup all'interno di ogni file HTML e standardizzandoli, qualsiasi browser Web può leggere e riformattare qualsiasi pagina Web. Essere in grado di riformattare le pagine Web dopo averle ricevute è fondamentale, perché una pagina potrebbe essere stata prodotta in una finestra 1.600 x 1.200 con colore a 24 bit, ma potrebbe dover essere visualizzata in una finestra 640 x 320 configurata per il colore a 8 bit.

Di seguito è fornita una breve introduzione ad HTML, per dare un'idea del suo scopo. Anche se è possibile scrivere documenti HTML con un editor standard, come fanno molte persone, si possono utilizzare editor HTML specializzati o elaboratori di testo che svolgono la maggior parte del lavoro (offrendo però un minore controllo sul risultato finale).

Una pagina Web consiste di un'intestazione e un corpo, entrambi racchiusi dai tag `<html>` e `</html>` (comandi di formattazione), tuttavia molti browser non si lamentano se questi mancano. Come è possibile osservare nella Figura 7.26(a), l'intestazione è racchiusa tra i tag `<head>` ed `</head>` mentre il corpo è racchiuso tra i tag `<body>` e `</body>`. Le stringhe all'interno dei tag sono chiamate **direttive**. La maggior parte dei tag HTML utilizza questo formato, vale a dire che utilizzano `<qualcosa>` per segnalare l'inizio di qualcosa e `</qualcosa>` per indicarne la fine. La maggior parte dei browser dispone di una voce di menu *Visualizza origine* o simile. La selezione di questa voce mostra il codice sorgente HTML della pagina, anziché il suo output formattato.

I tag possono essere scritti sia in maiuscolo sia in minuscolo. Di conseguenza, `<head>` ed `<HEAD>` indicano la stessa cosa, ma le versioni più nuove dello standard esigono l'uso delle sole minuscole. Il layout effettivo del documento HTML è irrilevante. Gli analizzatori sintattici HTML ignorano spazi e ritorni carrello extra, perché devono riformattare il testo per adattarlo all'area di visualizzazione corrente, pertanto è possibile aggiungere spazi per rendere i documenti HTML più leggibili (visto che la maggior parte dei documenti è piuttosto carente da questo punto di vista). Come altra conseguenza, non si possono utilizzare righe vuote per separare i paragrafi, perché vengono ignorate. È richiesto un tag esplicito.

Alcuni tag hanno parametri denominati, chiamati **attributi**. Ecco un esempio:

```

```

rappresenta il tag ``, con il parametro `src` impostato ad *abc* e il parametro `alt` impostato a *foobar*. Per ogni tag, lo standard HTML offre un elenco di parametri consentiti e del loro significato. Dal momento che ogni parametro è denominato, l'ordine di inserimento dei parametri non è significativo.

Tecnicamente, i documenti HTML sono scritti con il set di caratteri ISO 8859-1 Latin-1; tuttavia, per gli utenti le cui tastiere supportano solo ASCII, sono fornite sequenze di escape per caratteri speciali, come “è”. L'elenco di caratteri speciali è fornito nello standard. Tutti iniziano con una & commerciale e terminano con un punto e virgola. Per esempio produce uno spazio, è produce è ed ´ produce é. Visto che <, > e & hanno significati speciali, possono essere espressi solo con le loro sequenze di escape, rispettivamente <, > ed &.

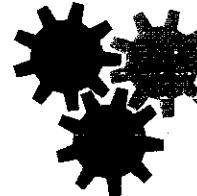
L'elemento principale dell'intestazione è il titolo, delimitato da `<title>` e `</title>`, ma possono essere presenti anche altri tipi di metainformazioni; il titolo vero e proprio non è visualizzato sulla pagina: alcuni browser lo utilizzano per intitolare la finestra della pagina.

Diamo ora un'occhiata ad alcune delle caratteristiche illustrate nella Figura 7.26; tutti i tag utilizzati sono mostrati nella Figura 7.27. Le intestazioni sono generate da un tag `<hn>`, dove *n* è una cifra compresa tra 1 e 6. Di conseguenza, `<h1>` è il titolo più importante, mentre `<h6>` è quello meno importante. Sta al browser rappresentarli in modo appropriato sullo schermo. Generalmente i titoli con numerazione inferiore sono visualizzati con un carattere più grande e più spesso. Il browser può anche scegliere di utilizzare colori diversi per ogni livello di titolo. Di norma, i titoli `<h1>` sono grandi e in grassetto, con almeno una riga vuota sopra e sotto. I titoli `<h2>`, invece, sono in un carattere più piccolo con meno spazio sopra e sotto.

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Benvenuti nella home page di AWI </h1>
 <br>
Siamo lieti che abbiate scelto di visitare la home page di <b> Amalgamated Widget</b>. Speriamo che <i> voi </i> possiate trovare tutte le informazioni necessarie.
<p>Di seguito trovate collegamenti alle informazioni sui nostri prodotti.
Potete ordinarli per via elettronica (WWW), per telefono o via fax. </p>
<hr>
<h2> Informazioni sui prodotti </h2>
<ul>
  <li> <a href="http://widget.com/products/big"> Aggeggi grandi </a>
  <li> <a href="http://widget.com/products/little"> Aggeggi piccoli </a>
</ul>
<h2> Numeri di telefono </h2>
<ul>
  <li> 1-800-WIDGETS
  <li> 1-415-765-4321
</ul>
</body>
</html>
```

(a)

Benvenuti nella home page di AWI



Siamo lieti che abbiate scelto di visitare la home page di **Amalgamated Widget**.
Speriamo che *voi* possiate trovare tutte le informazioni necessarie.

Informazioni sui prodotti

- [Aggeggi grandi](http://widget.com/products/big)
- [Aggeggi piccoli](http://widget.com/products/little)

Numeri di telefono

- 1-800-WIDGETS
- 1-415-765-4321

(b)

Figura 7.26. (a) Il codice HTML per una pagina Web di esempio. (b) La pagina formattata.

I tag **** e **<i>** sono utilizzati rispettivamente per attivare le modalità grassetto e corsivo. Se il browser non è in grado di visualizzare grassetti e corsivi, deve utilizzare un altro metodo di rappresentazione, per esempio un colore diverso per ognuno o l'inversione del video.

HTML fornisce diversi meccanismi per creare elenchi, anche nidificati. Gli elenchi iniziano con **** oppure ****, ed in entrambi i casi **** è utilizzato per indicare l'inizio degli elementi. Il tag **** apre un elenco non ordinato. I singoli elementi, contrassegnati con il tag **** nel codice sorgente, appaiono preceduti da punti elenco (•). Una variazione di questo meccanismo è ****, dedicato agli elenchi ordinati. Quando viene utilizzato questo tag, le voci **** sono numerate dal browser. A parte l'utilizzo di tag diversi di inizio e fine, **** e **** presentano la stessa sintassi e risultati simili.

I tag **
, **<p> e **<hr>** indicano un confine tra le sezioni del testo; il formato preciso può essere determinato dal foglio di stile (spiegato in seguito) associato alla pagina. Il tag **
** forza semplicemente un "a capo". Generalmente i browser non inseriscono una riga vuota dopo **
. Al contrario, **<p> inizia un paragrafo, che per esempio potrebbe inserire una riga vuota e un rientro. Teoricamente, **</p>** esiste per contrassegnare la fine di un paragrafo, ma è utilizzato raramente; molti autori HTML non sanno nemmeno che esiste. Per finire, **<hr>** forza un'interruzione e disegna una linea orizzontale sullo schermo.

HTML consente l'inclusione d'immagini in una pagina Web. Il tag **** specifica che un'immagine deve essere visualizzata nella posizione corrente nella pagina. Può disporre di diversi parametri. Il parametro **src** restituisce l'URL dell'immagine. Lo standard HTML non specifica quali formati grafici sono consentiti. In pratica, tutti i browser supportano i file GIF e JPEG. I browser sono liberi di supportare altri formati, ma questa estensione è una lama a doppio taglio. Se un utente è abituato a un browser che supporta file BMP, per esempio, potrebbe includerli nelle pagine Web e restare sorpreso dal fatto che altri browser ignorano le sue immagini.

Altri parametri di **** sono **align**, che controlla l'allineamento dell'immagine rispetto alla linea di base del testo (*top*, *middle*, *bottom*); **alt**, che fornisce il testo da utilizzare al posto dell'immagine se l'utente ha disattivato le immagini; **ismap**, un flag che indica se l'immagine è una mappa attiva (vale a dire se è selezionabile).

Per finire, dobbiamo parlare dei collegamenti ipertestuali, che utilizzano i tag **<a>** (ancoraggio) e ****. Come ****, **<a>** presenta numerosi parametri, compresi **href** (l'URL) e **name** (il nome del collegamento ipertestuale). Il testo tra **<a>** e **** è quello che viene visualizzato. Se viene selezionato, il browser esegue il collegamento ipertestuale verso un'altra pagina. È anche consentito inserire un'immagine in questa posizione: in questo caso, facendo clic sull'immagine si attiva il collegamento ipertestuale.

Come esempio, si consideri il seguente frammento HTML:

```
<a href="http://www.nasa.gov"> Home page della NASA </a>
```

Quando viene visualizzata una pagina con questo frammento, sullo schermo appare

[Home page della NASA](http://www.nasa.gov)

Tag	Descrizione
<html> ... </html>	Dichiara che la pagina Web è scritta in HTML
<head> ... </head>	Delimita l'intestazione della pagina
<title> ... </title>	Definisce il titolo (non visualizzato sulla pagina)
<body> ... </body>	Delimita il corpo della pagina
<hr> ... </hr>	Delimita un titolo di livello n
 ... 	Imposta ... in grassetto
<i> ... </i>	Imposta ... in corsivo
<center> ... </center>	Centra ... orizzontalmente nella pagina
 ... 	Racchiude un elenco non ordinato (puntato)
 ... 	Racchiude un elenco numerato
 ... 	Racchiude un elemento in un elenco ordinato o numerato

	Forza un'interruzione di riga
<p>	Avvia un paragrafo
<hr>	Inserisce una linea orizzontale
	Visualizza un'immagine
 ... 	Definisce un collegamento ipertestuale

Figura 7.27. Una selezione di tag HTML comuni. Alcuni possono accettare parametri aggiuntivi.

Se l'utente fa successivamente clic su questo testo, il browser preleva immediatamente la pagina il cui URL è <http://www.nasa.gov> e la visualizza.

Come secondo esempio, si consideri

```
<a href="http://www.nasa.gov">  </a>
```

Quando viene visualizzata, questa pagina mostra un'immagine (per esempio dello space shuttle); facendo clic su di essa si passa alla home page della NASA, proprio come con il testo sottolineato nell'esempio precedente. Se l'utente ha disattivato la visualizzazione automatica delle immagini, sarà visualizzato il testo NASA nella posizione in cui sarebbe dovuta apparire la figura.

Il tag **<a>** accetta il parametro **name** per costituire un collegamento ipertestuale che fa riferimento a un punto preciso di una pagina Web. Per esempio, alcune pagine Web iniziano con un sommario selezionabile. Facendo clic su un elemento in un sommario, l'utente passa alla corrispondente sezione nella pagina.

HTML continua ad evolversi. HTML 1.0 e HTML 2.0 non possedevano le tabelle, che sono state aggiunte in HTML 3.0. Una tabella HTML è composta da una o più righe, ognuna composta da una o più celle. Le celle possono contenere una vasta gamma di materiale, compresi testo, figure, icone, fotografie e persino altre tabelle. Le celle possono essere unite, pertanto un titolo si può espandere su più colonne. Gli autori delle pagine hanno un parziale controllo sul layout, determinando tra l'altro l'allineamento, gli stili del bordo e i margini di cella, ma sono i browser ad avere l'ultima parola nella rappresentazione delle tabelle.

Una definizione di tabella HTML è elencata nella Figura 7.28(a), mentre una possibile rappresentazione è mostrata nella Figura 7.28(b). Questo esempio mostra solo alcune delle caratteristiche basilari delle tabelle HTML. Le tabelle iniziano con il tag `<table>`, e si possono fornire altre informazioni per descrivere le loro proprietà generali.

Il tag `<caption>` si usa per fornire una didascalia. Ogni riga inizia con il tag `<tr>` (Table Row, riga della tabella). Le singole celle sono contrassegnate come `<th>` (Table Header, intestazione della tabella) o `<td>` (table data, dati della tabella). La distinzione consente ai browser di utilizzare rappresentazioni diverse, come visibile nell'esempio.

Per definire le tabelle sono previsti numerosi attributi. Comprendono modi per specificare allineamenti di cella orizzontali e verticali, la giustificazione all'interno di una cella, i bordi, il raggruppamento delle celle, le unità e altro.

In HTML 4.0 sono state aggiunte ulteriori caratteristiche, che comprendono le funzioni di accessibilità per utenti disabili, l'incorporamento degli oggetti (una generalizzazione del tag `` che permette di incorporare nelle pagine anche gli oggetti), il supporto per i linguaggi di scripting (per consentire il contenuto dinamico) e altro ancora.

Quando un sito Web è complesso, composto di molte pagine prodotte da più autori che lavorano per la stessa società, spesso è utile disporre di un modo per impedire che pagine diverse assumano un aspetto disomogeneo. Questo problema può essere risolto utilizzando i **fogli di stile**. Con il loro impiego le singole pagine non utilizzano più gli stili fisici, come grassetto e corsivo. Gli autori di pagina utilizzano invece stili logici come `<dn>` (definizione), `` (enfasi debole), `` (enfasi forte) e `<var>` (variabili di programma). Gli stili logici sono definiti nel foglio di stile, a cui si fa riferimento all'inizio di ogni pagina. In questo modo tutte le pagine utilizzano lo stesso stile; se il Webmaster decide di cambiare `` da 14 punti corsivo blu in 18 punti grassetto rosa, deve eseguire la modifica di una sola definizione per convertire l'intero sito Web. Un foglio di stile può essere comparato a un file `#include` in un programma C: modificando una definizione di macro, questa viene cambiata in tutti i file di programma che includono l'intestazione.

I moduli

HTML 1.0 era fondamentalmente unidirezionale. Gli utenti potevano richiamare le pagine dai provider di informazioni, ma era difficile restituire informazioni nell'altra direzione. Quando un numero sempre maggiore di organizzazioni commerciali ha iniziato a utilizzare il Web, vi fu un'elevata richiesta di traffico bidirezionale. Per esempio, molte società volevano essere in grado di raccogliere ordini tramite le loro pagine Web, i rivenditori di software volevano distribuire i programmi tramite il Web e chiedere ai clienti di compilare elettronicamente le schede di registrazione, le società che offrivano funzioni di ricerca volevano permettere ai loro clienti l'inserimento di parole chiave.

Queste richieste portarono all'inclusione dei **moduli** (*form*) a partire da HTML 2.0. I moduli contengono caselle e pulsanti che permettono agli utenti di inserire informazioni o compiere scelte, che vengono poi inviate al proprietario della pagina. A questo scopo si usa il tag `<input>`, che supporta un gran numero di parametri per determinare la dimen-

```

<html>
<head> <title> Una pagina di esempio con una tabella </title> </head>
<body>
<table border=1 rules=all>
<caption> Alcune differenze tra le versioni di HTML </caption>
<col align=left>
<col align=center>
<col align=center>
<col align=center>
<col align=center>
<tr> <th> Elemento <th> HTML 1.0 <th> HTML 2.0 <th> HTML 3.0 <th> HTML 4.0 </tr>
<tr> <th> Collegamenti ipertestuali <td> x <td> x <td> x <td> x </tr>
<tr> <th> Immagini <td> x <td> x <td> x <td> x </tr>
<tr> <th> Elenchi <td> x <td> x <td> x <td> x </tr>
<tr> <th> Mappe e immagini attive <td> &nbsp; <td> x <td> x <td> x </tr>
<tr> <th> Moduli <td> &nbsp; <td> x <td> x <td> x </tr>
<tr> <th> Equazioni <td> &nbsp; <td> &nbsp; <td> x <td> x </tr>
<tr> <th> Barre degli strumenti <td> &nbsp; <td> &nbsp; <td> x <td> x </tr>
<tr> <th> Tabelle <td> &nbsp; <td> &nbsp; <td> x <td> x </tr>
<tr> <th> Funzioni di accessibilità <td> &nbsp; <td> &nbsp; <td> &nbsp; <td> x </tr>
<tr> <th> Incorporamento degli oggetti <td> &nbsp; <td> &nbsp; <td> &nbsp; <td> x </tr>
<tr> <th> Scripting <td> &nbsp; <td> &nbsp; <td> &nbsp; <td> x </tr>
</table>
</body>
</html>

```

(a)

Alcune differenze tra le versioni di HTML

Elemento	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0
Collegamenti ipertestuali	x	x	x	x
Immagini	x	x	x	x
Elenchi	x	x	x	x
Mappe e immagini attive		x	x	x
Moduli		x	x	x
Equazioni			x	x
Barre degli strumenti			x	x
Tabelle			x	x
Funzioni di accessibilità				x
Incorporamento degli oggetti				x
Scripting				x

Figura 7.28. (a) Una tabella HTML. (b) Una possibile rappresentazione di questa tabella.

sione, la natura e l'utilizzo delle caselle visualizzate. I moduli più comuni contengono campi vuoti per accettare il testo dell'utente, caselle da selezionare, mappe attive e pulsanti di invio. L'esempio della Figura 7.29 illustra alcune opzioni.

Iniziamo lo studio dei moduli seguendo l'esempio. Come tutti i moduli, questo è racchiuso tra i tag `<form>` e `</form>`. Il testo non racchiuso in un tag è solamente visualizzato. Nel modulo possono essere utilizzati tranquillamente tutti i soliti tag (per esempio ``). In questo modulo sono utilizzati tre tipi di caselle di input.

Il primo tipo segue il testo "Nome". La casella è larga 46 caratteri e prevede che l'utente digitì una stringa, che è memorizzata nella variabile `customer` per la successiva elaborazione. Il tag `<p>` indica al browser di visualizzare il testo e le caselle successive sulla prossima riga, anche se c'è spazio nella riga corrente. Utilizzando `<p>` e altri tag di layout, l'autore della pagina può controllare l'aspetto del modulo sullo schermo.

La riga successiva del modulo chiede l'indirizzo dell'utente, offrendo 40 caratteri su una singola riga. Viene poi una riga che chiede la città, la provincia e la nazione. Non vengono utilizzati tag `<p>` tra questi campi, pertanto il browser li visualizza tutti su una riga (se ci stanno). Per il browser questo paragrafo contiene sei elementi: tre stringhe alternate a tre caselle. Sono visualizzate linearmente da sinistra a destra, ma si passa a una nuova riga solo se la riga corrente non può contenere l'elemento successivo. Di conseguenza, è concepibile che su uno schermo 1.600 x 1.200 tutte le stringhe e le caselle corrispondenti appaiano sulla stessa riga, mentre su uno schermo 1.024 x 768 potrebbero essere divise su due righe. Nello scenario peggiore, la parola "Nazione" si trova alla fine di una riga e la sua casella all'inizio della successiva.

La riga seguente chiede il numero della carta di credito e la data di scadenza. La trasmissione dei numeri di carta di credito su Internet dovrebbe essere fatta solo dopo aver preso adeguate misure di protezione. Alcune sono discusse nel Capitolo 8.

Dopo la data di scadenza appare una nuova funzionalità: i pulsanti di scelta, utilizzati quando occorre compiere una scelta tra due o più alternative. Il modello concettuale è quello di un'autoradio con alcuni pulsanti per scegliere le stazioni. Il browser visualizza queste caselle in un modulo che permette all'utente di selezionarle e deselezionarle con un clic (o utilizzando la tastiera). Selezionando un pulsante vengono disattivati tutti gli altri pulsanti dello stesso gruppo; la presentazione visiva è lasciata al browser. Anche per specificare la dimensione dell'aggeggio si utilizzano i pulsanti di scelta; i due gruppi sono distinti dal campo `name`, non dalla posizione statica o dall'utilizzo di qualcosa di simile a `<radiobutton> ... </radiobutton>`.

I parametri `value` vengono utilizzati per indicare quale pulsante di opzione è selezionato. A seconda dell'opzione scelta per la carta di credito, la variabile `cc` viene impostata sulla stringa "mastercard" o "visacard".

Dopo i due gruppi di pulsanti di scelta compare la modalità di spedizione, rappresentata da una casella di tipo `checkbox`. Può essere attivata oppure disattivata. A differenza dei pulsanti di scelta, dove è possibile scegliere un solo valore nel gruppo, una casella `checkbox` può essere attivata o disattivata indipendentemente dalle altre. Per esempio, quando si

```

<html>
<head> <title> MODULO D'ORDINE AWI </title> </head>
<body>
<h1> Modulo d'ordine </h1>
<form ACTION="http://widget.com/cgi-bin/widgetorder" method=POST>
<p> Nome <input name="customer" size=46> </p>
<p> Indirizzo <input name="address" size=40> </p>
<p> Città <input name="city" size=20> Provincia <input name="state" size =4>
Nazione <input name="country" size=10> </p>
<p> N. carta di credito <input name="cardno" size=10>
Scadenza <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard">
VISA <input name="cc" type=radio value="visacard"> </p>
<p> Dimensione aggeggio Grande <input name="product" type=radio value="expensive">
Piccolo <input name="product" type=radio value="cheap">
Spedizione via corriere <input name="express" type=checkbox> </p>
<p><input type=submit value="Invia ordine"> </p>
Grazie per avere ordinato un aggeggio AWI, il migliore che potevate acquistare!
</form>
</body>
</html>

```

(a)

Modulo d'ordine

Nome

Indirizzo

Città Provincia Nazione

N. carta di credito Scadenza M/C VISA

Dimensione aggeggio Grande Piccolo Spedizione via corriere

Grazie per avere ordinato un aggeggio AWI, il migliore che potevate acquistare!

(b)

Figura 7.29. (a) Il codice HTML per un modulo d'ordine. (b) La pagina formattata.

ordina una pizza tramite la pagina Web di Electropizza, l'utente può scegliere sardine e cipolle e ananas, ma non può scegliere piccola e media e grande per la stessa pizza. Gli ingredienti della pizza saranno rappresentati da tre caselle *checkbox* separate, mentre la dimensione della pizza sarà indicata da un set di pulsanti di scelta.

Tra parentesi, per elenchi molto lunghi da cui occorre eseguire una selezione, i pulsanti di scelta possono essere scomodi. A tale scopo sono forniti i tag `<select>` and `</select>`, che racchiudono un elenco di alternative con la semantica dei pulsanti di scelta (a meno che venga passato il parametro *multiple*: in questo caso la semantica è quella delle caselle *checkbox*). Alcuni browser rappresentano gli elementi compresi tra `<select>` e `</select>` come un menù a tendina.

Abbiamo appena visto due tipi incorporati del tag `<input>`: *radio* e *checkbox*; in realtà ne abbiamo visto anche un terzo, *text*. Visto che questo è il tipo predefinito, non è importante includere il parametro `type = text`, ma avremmo potuto farlo. Altri due tipi sono *password* e *textarea*. Una casella *password* equivale a una casella *text*, tranne per il fatto che i caratteri non vengono visualizzati durante la digitazione. Anche una casella *textarea* equivale a una casella *text*, tranne per il fatto che può contenere più righe.

Tornando all'esempio della Figura 7.29, di seguito appare un pulsante *submit*. Quando si fa clic su questo pulsante, le informazioni utente contenute nel modulo vengono inviate alla macchina che ha fornito il modulo. Come tutti gli altri tipi, *submit* è una parola riservata compresa dal browser. La stringa *value* specifica l'etichetta visualizzata sul pulsante. Tutte le caselle possono assumere valori, ma la caratteristica è necessaria solo in questo caso. Per le caselle di testo il contenuto del campo *value* è visualizzato nella casella, ma l'utente può cancellarlo o modificarlo. Le caselle *checkbox* e i pulsanti di scelta si possono invece inizializzare con un campo chiamato *checked* (perché *value* restituisce il testo, ma non indica la scelta).

Quando l'utente fa clic sul pulsante *submit*, il browser comprime le informazioni raccolte in una singola riga e le invia al server per l'elaborazione. Il carattere & viene utilizzato per separare i campi, mentre + rappresenta lo spazio. Nel modulo di esempio, la riga potrebbe somigliare al contenuto della Figura 7.30 (qui divisa in più righe perché la pagina non è abbastanza grande):

```
customer=John+Doe&address=100+Main+St.&city=White+Plains&state=NY&country=USA&
cardno=1234567890&expires=6/98&cc=mastercard&product=cheap&express=on
```

Figura 7.30. Una possibile risposta dal browser al server con le informazioni compilate dall'utente.

La stringa viene restituita al server in una sola riga, ovviamente. Se una casella *checkbox* non è selezionata, viene omessa dalla stringa. È compito del server dare un senso a questa stringa, e spiegheremo come procedere in seguito nel capitolo.

XML e XSL

HTML (con o senza moduli) non fornisce alcuna struttura alle pagine Web, anzi mescola il contenuto alla formattazione. Man mano che e-commerce e applicazioni di altro tipo divengono sempre più comuni, cresce il bisogno di strutturare le pagine Web e separare il contenuto dalla formattazione. Per esempio, un programma che ricerca sul Web il prezzo migliore di un libro o di un CD deve analizzare molte pagine Web alla ricerca del titolo e del prezzo di un articolo. Con le pagine Web in HTML è molto difficile per un programma capire dove si trovano il titolo e il prezzo.

Per questo motivo, W3C ha sviluppato un'estensione di HTML che consente la strutturazione delle pagine Web per l'elaborazione automatica. A tale scopo sono stati sviluppati due nuovi linguaggi. Il primo, **XML** (*eXtensible Markup Language*), descrive il contenuto Web in modo strutturato; il secondo linguaggio, **XSL** (*eXtensible Style Language*), descrive la formattazione in modo indipendente dal contenuto. Entrambi sono argomenti vasti e complicati, pertanto questa breve introduzione ha solo lo scopo di consentire ai lettori di farsi un'idea sul loro funzionamento.

Si consideri il documento XML di esempio della Figura 7.31. Definisce una struttura chiamata *book_list*, vale a dire un elenco di libri (*book*). Ogni libro dispone di tre campi: il titolo, l'autore e l'anno della pubblicazione. Queste strutture sono molto semplici. È possibile utilizzare strutture con campi ripetuti (per esempio più autori), campi facoltativi (per esempio il titolo del CD-ROM incluso) e campi alternativi (per esempio l'URL di una libreria se è disponibile la copia stampata, oppure l'URL di un sito di aste se il libro è fuori produzione).

In questo esempio i tre campi sono un'entità indivisibile, ma è comunque possibile dividerli ulteriormente. Per esempio, il campo autore potrebbe essere rappresentato come segue per consentire migliori funzioni di ricerca e formattazione:

```
<author>
  <first_name> Andrew </first_name>
  <last_name> Tanenbaum </last_name>
</author>
```

Ogni campo può essere diviso in campi secondari, nidificandoli arbitrariamente.

Il file della Figura 7.31 non fa altro che definire un elenco di libri contenente tre volumi. Non dice nulla su come visualizzare la pagina Web sullo schermo. Per fornire le informazioni di formattazione è necessario un secondo file *book_list.xsl*, contenente la definizione XSL. Questo file è un foglio di stile che comunica come visualizzare la pagina (esistono alternative ai fogli di stile, nonché un modo per convertire XML in HTML, ma queste alternative vanno oltre l'ambito di questo libro).

Un semplice file XSL per la formattazione della Figura 7.31 è mostrato nella Figura 7.32. Dopo alcune dichiarazioni necessarie, compreso l'URL dello standard XSL, il file contiene tag che iniziano con `<html>` e `<body>`. Questi definiscono come al solito l'inizio della

```
<?xml version="1.0"?
<?xml-stylesheet type="text/xsl" href="book_list.xsl"?>

<book_list>

<book>
  <title> Computer Networks, 4/e </title>
  <author> Andrew S. Tanenbaum </author>
  <year> 2003 </year>
</book>

<book>
  <title> Modern Operating Systems, 2/e </title>
  <author> Andrew S. Tanenbaum </author>
  <year> 2001 </year>
</book>

<book>
  <title> Structured Computer Organization, 4/e </title>
  <author> Andrew S. Tanenbaum </author>
  <year> 1999 </year>
</book>

</book_list>
```

Figura 7.31. Una semplice pagina Web in XML.

pagina Web. Segue la definizione di una tabella, con le intestazioni per le tre colonne. Occorre notare che oltre ai tag `<th>` esistono anche i tag `</th>`, di cui finora non ci eravamo preoccupati. Le specifiche di XML e XSL sono molto più rigorose delle specifiche di HTML. Esse affermano che è obbligatorio rifiutare i file dalla sintassi non corretta, anche se il browser può determinare che cosa desiderava realmente il Web designer. Un browser che accetta un file XML o XSL sintatticamente non corretti e ripara gli errori non è conforme, e sarà rifiutato dai test di conformità. I browser possono però segnalare gli errori. Questa misura draconiana è necessaria per risolvere il problema delle numerose pagine Web malformate che sono in circolazione.

L'istruzione

```
<xsl:for-each select="book_list/book">
```

è analoga a un'istruzione `for` in C. Il browser deve in pratica iterare nel corpo del ciclo (terminato da `</xsl:for-each>`) una volta per ogni book. Ogni iterazione genera in output cinque righe: `<tr>`, il titolo, l'autore, l'anno e `</tr>`. Dopo il ciclo, vengono emessi in output i tag di chiusura `</body>` e `</html>`. Il risultato dell'interpretazione di questo foglio di stile da parte del browser è identico a quello di una pagina Web che contiene la tabella in linea, ma con questo formato i programmi possono analizzare il file XML e (per esempio)

```
<?xml version='1.0'?
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">

<html>
<body>

<table border="2">
<tr>
  <th> Titolo </th>
  <th> Autore </th>
  <th> Anno </th>
</tr>

<xsl:for-each select="book_list/book">
<tr>
  <td> <xsl:value-of select="title"/> </td>
  <td> <xsl:value-of select="author"/> </td>
  <td> <xsl:value-of select="year"/> </td>
</tr>
</xsl:for-each>
</table>

</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Figura 7.32. Un foglio di stile in XSL.

trovare facilmente i libri pubblicati dopo il 2000. Vale la pena evidenziare che, anche se il nostro file XSL conteneva una sorta di ciclo, le pagine Web in XML e XSL sono statiche, poiché contengono semplicemente delle istruzioni per il browser sul modo di visualizzare la pagina (proprio come fanno le pagine Web). Naturalmente, per utilizzare XML e XSL, il browser deve essere in grado di interpretare XML e XSL, ma molti browser odierni dispongono già di questa funzionalità. Non è ancora chiaro se XSL prenderà il posto dei fogli di stile tradizionali.

Non abbiamo mostrato come farlo, ma con XML il progettista di siti Web può creare file di definizione dove le strutture sono definite in anticipo. Questi file di definizione possono essere inclusi, consentendone l'utilizzo nella costruzione di pagine Web compresse. Per informazioni aggiuntive su queste e molte altre caratteristiche di XML e XSL, si possono consultare i numerosi libri sull'argomento. Due esempi sono (Livingston, 2002 e Williamson, 2001).

Prima di terminare la discussione su XML e XSL, vale la pena commentare una battaglia ideologica in corso tra WWW Consortium e la comunità dei Web designer. L'obiettivo ori-

ginale di HTML era specificare la *struttura* del documento, non il suo *aspetto*. Ecco un esempio:

```
<h1> Fotografie di Deborah </h1>
```

indica al browser di enfatizzare il titolo, ma non comunica nulla sul tipo di carattere, la dimensione in punti o il colore. Questo compito è lasciato al browser, che conosce le proprietà dello schermo (per esempio, il numero di pixel disponibili). Tuttavia, molti disegnatori di pagine Web desideravano il controllo assoluto sull'aspetto delle pagine, pertanto ad HTML furono aggiunti nuovi tag per controllare l'aspetto, per esempio

```
<font face="helvetica" size="24" color="red"> Fotografie di Deborah </font>
```

Furono inoltre aggiunti metodi per controllare il posizionamento accurato sullo schermo. Il problema di questo approccio è che non è portabile. Anche se una pagina può essere rappresentata alla perfezione nel browser in cui è stata progettata, con un altro browser o un'altra release dello stesso browser (o una diversa risoluzione dello schermo) potrebbe essere rappresentata in modo completamente diverso. XML era in parte un tentativo di tornare all'obiettivo originale: specificare la struttura, non l'aspetto di un documento. Tuttavia, viene fornito anche XSL per gestire l'aspetto. Ovviamente qualcuno abuserà di entrambi i formati: potete contarci.

XML può essere utilizzato per scopi diversi dalla descrizione delle pagine Web. È in crescita l'utilizzo come linguaggio di comunicazione tra i programmi applicativi; in particolare **SOAP** (*Simple Object Access Protocol*) è un modo per eseguire RPC tra applicazioni in modo indipendente dal linguaggio e dal sistema. Il client costruisce la richiesta come messaggio XML e la invia al server, utilizzando il protocollo HTTP (descritto sotto). Il server invia una replica come messaggio formattato in XML. In questo modo, le applicazioni su piattaforme eterogenee possono comunicare.

XHTML (*eXtended HyperText Markup Language*)

HTML continua ad evolversi per soddisfare nuove richieste. Molti esperti del settore ritengono che in futuro la maggior parte delle periferiche abilitate per il Web non saranno PC, ma dispositivi palmari wireless come i PDA. Questi dispositivi hanno troppo poca memoria per i grandi browser ricchi di euristica, che cercano di gestire pagine Web con sintassi scorretta; di conseguenza, il successore di HTML 4 è un linguaggio molto pignolo. È chiamato **XHTML** (*eXtended HyperText Markup Language*) anziché HTML 5, perché si tratta fondamentalmente di HTML 4 riformulato in XML. Con questo intendiamo dire che i tag come `<h1>` non hanno un significato intrinseco. Per ottenere l'effetto di HTML 4, è necessaria una definizione nel file XSL. XHTML è il nuovo standard Web e dovrebbe essere utilizzato per tutte le nuove pagine Web, in modo da ottenere la massima portabilità tra piattaforme e browser.

Esistono sei differenze principali e numerose piccole differenze tra XHTML e HTML

4; vediamo rapidamente quelle di maggiore importanza. Per prima cosa, le pagine e i browser XHTML devono essere rigorosamente conformi allo standard. Non esisteranno più pagine Web malformate! Questa proprietà è stata ereditata da XML.

In secondo luogo, tutti i tag e gli attributi devono essere in minuscolo. I tag come `<HTML>` non sono validi in XHTML. L'utilizzo di tag come `<html>` ora è obbligatorio. Allo stesso modo, è proibito anche l'utilizzo di ``, perché contiene un attributo in maiuscolo. Terzo, sono richiesti i tag di chiusura, persino per `<p>`. Per i tag che non hanno un tag di chiusura naturale, come `
`, `<hr>` e ``, occorre far precedere la parentesi angolare di chiusura da una barra, come in

```

```

Quarto, gli attributi devono essere racchiusi tra virgolette. Ecco un esempio di sintassi che non è più consentita:

```
<img SRC="pic001.jpg" height=500 />
```

500 deve essere racchiuso tra virgolette, proprio come il nome del file JPEG, anche se è solo un numero.

Quinto, i tag devono essere nidificati correttamente. In passato, non era richiesta una sintassi corretta, a patto di raggiungere il corretto stato finale. Per esempio:

```
<center> <b> Fotografie delle vacanze </center> </b>
```

era legale; in XHTML non lo è più. I tag devono essere chiusi nell'ordine inverso a quello in cui sono stati aperti.

Sesto, ogni documento deve specificare il proprio tipo di documento. Lo abbiamo visto nella Figura 7.32, per esempio. Per una discussione di tutte le modifiche, grandi o piccole, visitare www.w3.org.

7.3.3 I documenti Web dinamici

Fino a questo punto il modello utilizzato è quello della Figura 6.6: il client invia un nome di file al server, che restituisce il file. Agli esordi del Web, infatti, tutto il contenuto era statico come questo (semplici file). Negli ultimi tempi buona parte del contenuto è divenuta dinamica, vale a dire generata su richiesta anziché archiviata su disco. La generazione del contenuto può avvenire sia sul lato server sia sul lato client; vediamo entrambi i casi.

La generazione di pagine Web dinamiche sul lato server

Per scoprire perché è necessaria la generazione di contenuto sul server, consideriamo l'utilizzo dei moduli descritto in precedenza. Quando un utente compila un modulo e fa clic sul pulsante di invio, al server viene inviato un messaggio con il contenuto del modulo, vale a dire i campi compilati dall'utente. Questo messaggio non è il nome di un file da restituire. È invece necessario consegnare il messaggio a un programma o uno script per

l'elaborazione. Solitamente, il processo usa le informazioni fornite dall'utente per cercare un record in un database sul disco del server, e generare una pagina HTML personalizzata da restituire al client. Per esempio, in un'applicazione di e-commerce, dopo che l'utente ha fatto clic su *Procedi con il pagamento* il browser restituisce il cookie contenente gli articoli del carrello della spesa; tuttavia, è necessario invocare un programma o uno script sul server per elaborare il cookie e generare una pagina HTML in risposta. La pagina HTML potrebbe visualizzare un modulo contenente l'elenco di articoli nel carrello e l'ultimo indirizzo di spedizione noto, insieme a una richiesta di verificare le informazioni e di specificare il metodo di pagamento. I passaggi necessari all'elaborazione dell'informazione di un modulo HTML sono mostrati nella Figura 7.33.

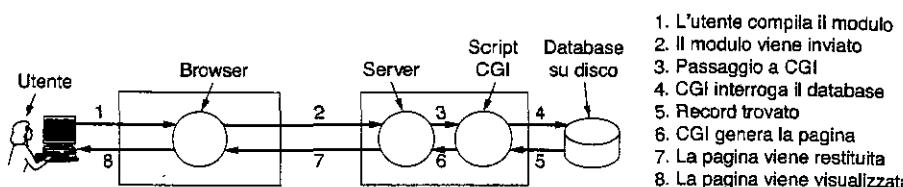


Figura 7.33. I passaggi per elaborare le informazioni di un modulo HTML.

Il metodo tradizionale per gestire i moduli e altre pagine interattive è un sistema chiamato **CGI** (*Common Gateway Interface*). È un'interfaccia standardizzata che consente ai server Web di comunicare con script e programmi di back-end che possono accettare l'input (per esempio dai moduli) e generare pagine HTML in risposta. Solitamente, questi back-end sono script scritti nel linguaggio di scripting Perl, perché gli script Perl sono più facili e veloci da scrivere rispetto ai programmi (per lo meno per chi sa programmare in Perl). Per convenzione, risiedono in una directory chiamata *cgi-bin*, che è visibile nell'URL. A volte al posto di Perl viene utilizzato un altro linguaggio di scripting, Python.

Come esempio del funzionamento di CGI, si consideri il caso di un prodotto della società Truly Great Products Company, venduto senza una scheda di registrazione per la garanzia. Il cliente deve invece visitare www.tgpc.com per eseguire la registrazione online. Su tale pagina, trova un collegamento ipertestuale che afferma

Fare clic qui per registrare il prodotto

Questo collegamento punta a uno script Perl, diciamo www.tgpc.com/cgi-bin/reg.perl. Quando lo script viene invocato senza parametri, restituisce una pagina HTML contenente il modulo di registrazione. Quando l'utente compila il modulo e fa clic su *Invia*, viene restituito un messaggio allo script, contenente i valori inseriti utilizzando lo stile della Figura 7.30. Lo script Perl analizza sintatticamente i parametri, crea una voce nel database per il nuovo cliente e restituisce una pagina HTML con un numero di registrazione e un numero di telefono relativo all'help desk. Questo non è l'unico modo per gestire i moduli, ma è tra i più usati. Esistono molti libri sulla creazione di script CGI e la programmazione in Perl. Alcuni esempi sono (Hanegan, 2001; Lash, 2002; Meltzer e Michalski, 2001).

Gli script CGI non sono l'unico modo per generare contenuto dinamico sul lato server. Un altro metodo largamente utilizzato consiste nell'incorporare piccoli script nelle pagine HTML, e di farli eseguire dal server stesso per generare la pagina. Un celebre linguaggio utilizzato per la scrittura degli script è **PHP** (*PHP: Hypertext Preprocessor*). Per utilizzarlo il server deve comprendere PHP, proprio come un browser deve comprendere XML per interpretare le pagine scritte in XML. Solitamente, i server si aspettano che le pagine Web contenenti PHP abbiano un'estensione *php* anziché *html* o *htm*.

Un breve script PHP è mostrato nella Figura 7.34; dovrebbe funzionare con qualsiasi server con PHP installato. Contiene codice HTML normale, tranne per lo script PHP racchiuso nel tag `<?php ... ?>`. Il suo scopo è generare una pagina Web che mostra informazioni sul browser che la invoca. I browser di norma inviano alcune informazioni insieme alle loro richieste (e agli eventuali cookie), inserite nella variabile *HTTP_USER_AGENT*. Se questo script viene inserito nel file *test.php* memorizzato nella directory WWW presso la società ABCD, digitando l'URL www.abcd.com/test.php viene creata una pagina Web che comunica all'utente il browser, la lingua e il sistema operativo utilizzati.

```
<html>
<body>
<h2> Ecco che cosa so di te </h2>
<?php echo $HTTP_USER_AGENT ?>

</body>
</html>
```

Figura 7.34. Esempio di pagina HTML che incorpora codice PHP.

PHP è particolarmente valido nella gestione dei moduli, ed è più semplice rispetto all'utilizzo di uno script CGI. Come esempio del funzionamento con i moduli si consideri la Figura 7.35(a), che contiene una normale pagina HTML al cui interno si trova un modulo. L'unica cosa insolita di questo modulo è la prima riga, che specifica che il file *action.php* deve essere invocato per gestire i parametri dopo che l'utente ha compilato e inviato il modulo. La pagina visualizza due caselle di testo: la prima richiede un nome e la seconda un'età. Dopo che le due caselle sono state compilate e il modulo inviato, il server analizza la stringa restituita (simile a quella della Figura 7.30) inserendo il nome nella variabile *name* e l'età nella variabile *age*; subito dopo avvia l'elaborazione del file *action.php*, mostrato nella Figura 7.35(b). Durante l'elaborazione del file vengono eseguiti i comandi PHP. Se l'utente ha inserito "Barbara" e "24" nelle caselle, il file HTML restituito sarà quello mostrato nella Figura 7.35(c). Di conseguenza, utilizzando PHP la gestione dei moduli diventa estremamente semplice.

Anche se PHP è facile da utilizzare, si tratta di un potente linguaggio di programmazione orientato all'interfacciamento tra il Web e un database su server. Possiede variabili, stringhe, array e la maggior parte delle strutture di controllo presenti in C, ma dispone di funzioni di I/O più potenti di *printf*. PHP è un codice open source ed è disponibile gratuitamente.

```
<html>
<body>
<form action="action.php" method="post">
<p> Come ti chiami: <input type="text" name="name"> </p>
<p> Quanti anni hai: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

(a)

```
<html>
<body>
<h1> Risposta: </h1>
Ciao <?php echo $name; ?>.
Previsione: l'anno prossimo avrai <?php echo $age + 1; ?> anni
</body>
</html>
```

(b)

```
<html>
<body>
<h1> Risposta: </h1>
Ciao Barbara.
Previsione: l'anno prossimo avrai 25 anni
</body>
</html>
```

(c)

Figura 7.35. (a) Una pagina Web contenente un modulo. (b) Uno script PHP per la gestione dell'output del modulo. (c) L'output dello script PHP quando i valori di input sono rispettivamente "Barbara" e 24.

È progettato in modo specifico per funzionare bene con Apache, che è anch'esso open source ed è il server Web più utilizzato al mondo. Per ulteriori informazioni su PHP, vedere (Valade, 2002).

Abbiamo visto due diversi modi per generare pagine HTML dinamiche: gli script CGI e il codice PHP incorporato. Esiste anche una terza tecnica, chiamata **JSP** (*JavaServer Pages*) e simile a PHP, tranne per il fatto che la parte dinamica è scritta nel linguaggio di programmazione Java, anziché in PHP. Le pagine che utilizzano questa tecnica presentano l'estensione del file *jsp*. Una quarta tecnica, **ASP** (*Active Server Pages*), è la versione Microsoft di PHP e JavaServer Pages. Per la generazione del contenuto dinamico utilizza il linguaggio di scripting proprietario di Microsoft, Visual Basic Script. Le pagine che utilizzano questa tecnica hanno l'estensione *asp*. La scelta tra **PHP**, **JSP** e **ASP** di solito ha a

che fare più con la politica (open source contro Sun contro Microsoft) che con la tecnologia, perché i tre linguaggi sono in un certo senso comparabili.

L'insieme di tecnologie per la generazione di contenuto dinamico a volte è chiamata **HTML dinamico**.

La generazione di pagine Web dinamiche sul lato client

Gli script CGI, PHP, JSP e ASP risolvono il problema della gestione di moduli e interazioni con i database sul server. Possono accettare le informazioni in ingresso dai moduli, cercare le informazioni in uno o più database e generare pagine HTML con i risultati. Ciò che non possono fare è rispondere ai movimenti del mouse o interagire direttamente con gli utenti. A questo scopo, è necessario disporre di script incorporati nelle pagine HTML, che sono eseguiti sulla macchina client anziché sulla macchina server. Iniziando da HTML 4.0, tali script possono essere utilizzati con il tag **<script>**. Il linguaggio di scripting più popolare per il lato client è **JavaScript**, a cui ora daremo una rapida occhiata.

JavaScript è un linguaggio di scripting, molto liberamente ispirato da alcune idee del linguaggio di programmazione Java. Non si tratta comunque di Java. Come altri linguaggi di scripting, è un linguaggio di livello molto alto. Per esempio, con una sola riga di codice JavaScript è possibile aprire una finestra di dialogo, attendere l'input di testo e archiviare la stringa risultante in una variabile. Le funzioni di alto livello come questa rendono JavaScript ideale per la progettazione di pagine Web interattive. D'altra parte, il fatto che non sia standardizzato e stia mutando più rapidamente di un moscerino della frutta intrappolato in una macchina a raggi X rende davvero difficile scrivere programmi JavaScript capaci di funzionare su tutte le piattaforme, ma forse un giorno si stabilizzerà.

Come esempio di programma in JavaScript, si consideri la Figura 7.36. Come nel caso della Figura 7.35(a), mostra un modulo che chiede un nome e un'età, prevedendo quanti anni avrà la persona in questione l'anno successivo. Il corpo è quasi equivalente all'esempio di PHP; la differenza principale sta nella dichiarazione del pulsante di invio e nell'istruzione di assegnazione al suo interno. Questa istruzione di assegnazione comunica al browser di invocare lo script *response* al clic di un pulsante, e di passargli come parametro il modulo stesso.

La parte completamente nuova riguarda la dichiarazione della funzione JavaScript *response* nell'intestazione del file HTML, un'area normalmente riservata a titoli, colori di sfondo e così via. Questa funzione estrae il valore del campo *name* dal modulo e lo memorizza nella variabile *person* come stringa. Estraie anche il valore del campo *age*, lo converte in un intero utilizzando la funzione *eval*, vi somma 1 e memorizza il risultato in *years*. Apre quindi un documento per l'output, esegue quattro scritture utilizzando il metodo *writeln* e chiude il documento. Il documento è un file HTML, come è possibile osservare dai tag HTML che contiene. Il browser visualizza quindi il documento sullo schermo.

È molto importante comprendere che anche se le Figure 7.35 e 7.36 sembrano simili, sono elaborate in modo completamente diverso. Nella Figura 7.35, dopo che l'utente ha fatto clic sul pulsante di invio, il browser raccoglie le informazioni in una lunga stringa simile

```

<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
    var person = test_form.name.value;
    var years = eval(test_form.age.value) + 1;
    document.open();
    document.writeln("<html> <body>");
    document.writeln("Ciao " + person + "<br>");
    document.writeln("Previsione: l'anno prossimo avrai " + years + " anni.");
    document.writeln("</body> </html>");
    document.close();
}
</script>
</head>

<body>
<form>
Come ti chiami <input type="text" name="name">
<p>
Quanti anni hai <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>

```

Figura 7.36. L'utilizzo di JavaScript per l'elaborazione di un modulo.

alla Figura 7.30 e la invia al server che ha inviato la pagina. Il server osserva il nome del file PHP e lo esegue. Lo script PHP produce una nuova pagina HTML; tale pagina viene restituita al browser per la visualizzazione. Nella Figura 7.36, quando l'utente fa clic sul pulsante di invio, il browser interpreta una funzione JavaScript contenuta nella pagina. Tutto il lavoro viene svolto in locale, all'interno del browser. Non esistono contatti con il server. Di conseguenza, il risultato è visualizzato quasi immediatamente; al contrario con PHP può esserci un ritardo di diversi secondi prima che il codice HTML risultante arrivi al client. La differenza tra scripting sul lato server e scripting sul lato client è mostrata nella Figura 7.37, insieme ai passaggi coinvolti. In entrambi i casi, i passaggi numerati iniziano dopo la visualizzazione del modulo. Il passaggio 1 consiste nell'accettare l'input dell'utente. Segue poi l'elaborazione dell'input, differente nei due casi.

Questa differenza non significa che JavaScript sia migliore di PHP. I loro utilizzi sono completamente diversi. PHP (e di conseguenza JSP e ASP) sono utilizzati quando è necessaria l'interazione con un database remoto; JavaScript viene utilizzato quando l'interazione avviene con l'utente sul computer client. È certamente possibile (e frequente) avere

Lo strato applicazione

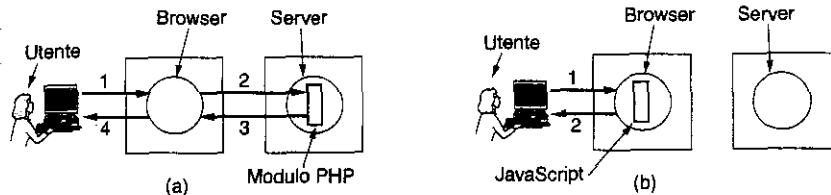


Figura 7.37. (a) Lo scripting sul lato server con PHP. (b) Lo scripting sul lato client con JavaScript.

pagine HTML che utilizzano sia PHP sia JavaScript, anche se (ovviamente) le due tecnologie non possono svolgere lo stesso compito o controllare lo stesso pulsante.

JavaScript è un linguaggio di programmazione completo, con tutta la potenza di C o Java. Possiede variabili, stringhe, array, oggetti, funzioni e strutture di controllo. Possiede inoltre numerose caratteristiche specifiche per le pagine Web, compresa la capacità di gestire finestre e frame, impostare e ottenere cookie, lavorare con i moduli e gestire i collegamenti ipertestuali. Un esempio di programma JavaScript che utilizza una funzione ricorsiva è mostrato nella Figura 7.38.

```

<html>
<head>
<script language="javascript" type="text/javascript">

function response(test_form) {
    function factorial(n) {if (n == 0) return 1; else return n * factorial(n - 1);}
    var r = eval(test_form.number.value);           // r = digitato nell'argomento
    document.myform.mytext.value = "Ecco i risultati:\n";
    for (var i = 1; i <= r; i++) // stampa una riga alla volta da 1 a r
        document.myform.mytext.value += (i + "!" + factorial(i) + "\n");
}

</script>
</head>

<body>
<form name="myform">
Inserire un numero: <input type="text" name="number">
<input type="button" value="Calcola il fattoriale" onclick="response(this.form)">
</p>
<textarea name="mytext" rows=25 cols=50> </textarea>
</form>
</body>
</html>

```

Figura 7.38. Un programma JavaScript per il calcolo e la stampa dei fattoriali.

JavaScript può anche tenere traccia del movimento del mouse sugli oggetti dello schermo. Molte pagine Web JavaScript fanno accadere qualcosa quando il puntatore del mouse viene spostato su un testo o un'immagine. Spesso l'immagine cambia, o appare improvvisamente un menu. Questo tipo di comportamento è facile da programmare in JavaScript e produce pagine Web animate. Un esempio è mostrato nella Figura 7.39.

```
<html>
<head>
<script language="javascript" type="text/javascript">
if (!document.myurl) document.myurl = new Array();
document.myurl[0] = "http://www.cs.vu.nl/ast/im/kitten.jpg";
document.myurl[1] = "http://www.cs.vu.nl/ast/im/puppy.jpg";
document.myurl[2] = "http://www.cs.vu.nl/ast/im/bunny.jpg";
function pop(m) {
    var urx = "http://www.cs.vu.nl/ast/im/cat.jpg";
    popupwin = window.open(document.myurl[m],"mywind","width=250,height=250");
}
</script>
</head>

<body>
<p><a href="#" onMouseover="pop(0); return false;"> Gattini </a> </p>
<p><a href="#" onMouseover="pop(1); return false;"> Cagnolini </a> </p>
<p><a href="#" onMouseover="pop(2); return false;"> Coniglietti </a> </p>
</body>
</html>
```

Figura 7.39. Una pagina Web interattiva che risponde ai movimenti del mouse.

JavaScript non è l'unico modo per rendere le pagine Web altamente interattive. Un altro metodo popolare prevede l'utilizzo delle **applet**, piccoli programmi Java compilati in codice macchina per un computer virtuale chiamato **JVM (Java Virtual Machine)**. Le applet possono essere incorporate nelle pagine HTML (tra `<applet>` e `</applet>`) e interpretate dai browser che supportano JVM. Dal momento che le applet Java sono interpretate, e non eseguite direttamente, l'interprete Java può impedire loro di svolgere operazioni nocive. Questo almeno in teoria: in pratica gli autori di applet hanno trovato nelle librerie I/O di Java una quantità di bug da sfruttare.

Microsoft ha risposto alle applet Java di Sun permettendo alle pagine Web di contenere **controlli ActiveX**, vale a dire programmi compilati nel linguaggio macchina Pentium ed eseguiti direttamente sull'hardware. Questa caratteristica li rende particolarmente veloci e più flessibili delle applet Java interpretate, perché possono svolgere qualsiasi operazione eseguita da un programma. Quando Internet Explorer vede un controllo ActiveX in una pagina, lo scarica, ne verifica l'identità e lo esegue. Tuttavia, con il download e l'esecuzione di programmi esterni sorgono problemi di protezione, di cui parleremo nel Capitolo 8.

Dal momento che quasi tutti i browser possono interpretare sia i programmi Java sia JavaScript, un webmaster che desidera produrre una pagina Web molto interattiva può scegliere tra almeno due tecniche; se la portabilità su piattaforme diverse non è importante, può prendere in considerazione anche ActiveX. Come regola generale, i programmi JavaScript sono facili da scrivere, le applet Java vengono eseguite velocemente e i controlli ActiveX sono eseguiti ancora più rapidamente. Inoltre, visto che tutti i browser implementano esattamente la stessa JVM ma non la stessa versione di JavaScript, le applet Java sono più portabili dei programmi JavaScript. Per ulteriori informazioni su JavaScript esistono molti libri, tutti con un buon numero di pagine (spesso più di 1.000). Alcuni esempi sono (Easttom, 2001; Harris, 2001; McFedries, 2001).

Prima di lasciare l'argomento del contenuto Web dinamico, riepiloghiamo brevemente quanto affermato finora. Le pagine Web possono essere generate all'istante da vari script sulla macchina server. Una volta ricevute dal browser, sono trattate come normali pagine HTML e visualizzate. Gli script possono essere scritti in Perl, PHP, JSP o ASP, come mostrato nella Figura 7.40.

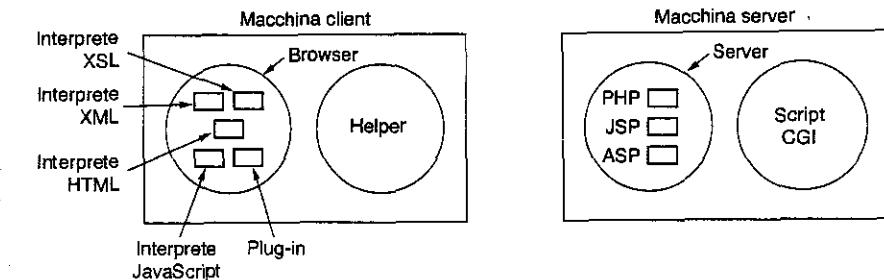


Figura 7.40. I molti modi per generare e visualizzare il contenuto.

La generazione di contenuto dinamico è possibile anche sul lato client. Le pagine Web possono essere scritte in XML e poi convertite in HTML in base a un file XSL. I programmi JavaScript possono eseguire calcoli arbitrari. Per finire, plug-in e applicazioni helper possono essere utilizzati per visualizzare il contenuto in un gran numero di formati.

7.3.4 HTTP (*HyperText Transfer Protocol*)

Il protocollo di trasferimento usato sul World Wide Web è **HTTP (HyperText Transfer Protocol)**, che specifica quali messaggi i client possono inviare ai server e quali risposte questi ultimi possono restituire. Ogni interazione consiste di una richiesta ASCII, seguita da una risposta MIME RFC 822. Tutti i client e i server devono utilizzare questo protocollo, definito in RFC 2616. In questa sezione vedremo alcune delle sue proprietà più importanti.

Le connessioni

Il modo solitamente utilizzato da un browser per contattare un server prevede di stabilire una connessione TCP alla porta 80 sulla macchina server, anche se questa procedura non è formalmente richiesta. Il vantaggio dell'utilizzo di TCP sta nel fatto che né i browser né i server devono preoccuparsi di messaggi persi, duplicati, messaggi lunghi o acknowledgement. Tutti questi dettagli sono gestiti dall'implementazione di TCP.

In HTTP 1.0, dopo avere stabilito la connessione, viene inviata una singola richiesta e restituita una singola risposta; a questo punto la connessione TCP viene rilasciata. In un mondo in cui la tipica pagina Web era composta interamente da testo HTML, questo metodo era adeguato. Nel giro di pochi anni, la pagina Web media ha iniziato a contenere numerose icone, immagini e altri abbellimenti, pertanto stabilire una connessione TCP per trasportare una singola icona diventa un modo molto costoso di operare.

Questa osservazione ha portato a HTTP 1.1, che supporta le **connessioni persistenti**. Con questa tecnologia è possibile stabilire una connessione TCP, inviare una richiesta, ottenere una risposta, e poi inviare altre richieste e ottenere nuove risposte. Ammortizzando l'impostazione e il rilascio di TCP su più richieste, il corrispondente overhead dovuto a TCP è inferiore. È inoltre possibile concatenare le richieste, vale a dire inviare la richiesta 2 prima dell'arrivo della risposta alla richiesta 1.

I metodi

Anche se HTTP fu progettato per l'utilizzo sul Web, è stato intenzionalmente reso più generico del necessario con un occhio alle future applicazioni orientate agli oggetti. Per questo motivo sono supportate operazioni diverse dalla richiesta di una pagina Web, chiamate **metodi**; questa generalità ha permesso la nascita di SOAP. Ogni richiesta consiste di una o più righe di testo ASCII, con la prima parola della prima riga che rappresenta il nome del metodo richiesto. I metodi incorporati sono elencati nella Figura 7.41, e per accedere agli oggetti generici sono disponibili altri metodi specifici per ciascun oggetto. I nomi sono *case sensitive*, pertanto *GET* è un metodo legale ma *get* non lo è.

Il metodo *GET* richiede al server di inviare la pagina, che nel caso più generale si tratta di un oggetto, ma in pratica è un file. La pagina viene codificata in MIME. La maggior parte delle richieste ai server Web sono di tipo *GET*. La forma normale di *GET* è

```
GET nomefile HTTP/1.1
```

dove *nomefile* è il nome della risorsa (file) da prelevare mentre 1.1 è la versione del protocollo utilizzata.

Il metodo *HEAD* chiede solamente l'intestazione del messaggio, senza la pagina effettiva. Questo metodo può essere utilizzato per ottenere la data dell'ultima modifica di una pagina, per raccogliere informazioni a scopo di indicizzazione o per verificare la validità di un URL.

Metodo	Descrizione
GET	Richiede di leggere una pagina Web
HEAD	Richiede di leggere l'intestazione di una pagina Web
PUT	Richiede di memorizzare una pagina Web
POST	Accoda alla risorsa indicata (per esempio una pagina Web)
DELETE	Rimuove la pagina Web
TRACE	Mostra la richiesta in ingresso
CONNECT	Riservato per utilizzi futuri
OPTIONS	Interroga determinate opzioni

Figura 7.41. I metodi di richiesta HTTP incorporati.

Il metodo *PUT* è l'inverso di *GET*: anziché leggere la pagina, scrive su di essa. Questo metodo permette di costruire una raccolta di pagine Web su un server remoto. Il corpo della richiesta contiene la pagina, e può essere codificato utilizzando MIME. In questo caso, le righe che seguono *PUT* possono contenere intestazioni di autenticazione e *Content-Type*, per provare che il chiamante ha l'autorizzazione per eseguire l'operazione richiesta. Il metodo *POST* assomiglia a *PUT*. Anch'esso trasporta un URL, ma anziché sostituire i dati esistenti, i nuovi dati vengono "accodati" in senso generalizzato. In questo contesto la pubblicazione di un messaggio in un newsgroup o l'aggiunta di un file a una bacheca elettronica sono esempi di accodamento. In pratica, né *PUT* né *POST* vengono utilizzati spesso [NdR - tranne che con i moduli e HTML dinamico].

DELETE fa proprio ciò che ci si aspetta: rimuove la pagina. Come con *PUT*, l'autenticazione e le autorizzazioni giocano un ruolo fondamentale. Non vi sono garanzie del successo di *DELETE*, perché anche se il server HTTP remoto desidera eliminare la pagina, il file sottostante potrebbe essere in una modalità che impedisce la modifica o la rimozione da parte del server HTTP.

Il metodo *TRACE* è utilizzato per il debug. Indica al server di mostrare la richiesta. Questo metodo è utile quando le richieste non vengono elaborate correttamente, e il client desidera sapere quale richiesta è stata effettivamente ricevuta dal server.

Il metodo *CONNECT* oggi non è utilizzato. È riservato per il futuro.

Il metodo *OPTIONS* offre al client un modo per interrogare il server sulle sue proprietà o su quelle di un file specifico.

Ogni richiesta ottiene una risposta costituita da una riga di stato e magari da altre informazioni (per esempio una pagina Web o parte di essa). La riga di stato contiene un codice di stato di tre cifre che indica se la richiesta è stata soddisfatta e, in caso contrario, la causa del fallimento. La prima cifra è utilizzata per dividere le risposte in cinque gruppi principali, come mostrato nella Figura 7.42. I codici 1xx sono utilizzati raramente nella pratica. I codici 2xx indicano che la richiesta è stata gestita con successo e che il contenuto viene restituito.

Codice	Significato	Esempi
1xx	Informazione	100 = il server accetta di soddisfare la richiesta del client
2xx	Successo	200 = richiesta eseguita con successo; 204 = nessun contenuto presente
3xx	Reindirizzamento	301 = pagina spostata; 304 = pagina nella cache ancora valida
4xx	Errore del client	403 = pagina vietata; 404 = pagina non trovata
5xx	Errore del server	500 = errore interno del server; 503 = riprovare più tardi

Figura 7.42. I gruppi di risposte e i corrispondenti codici di stato.

I codici 3xx indicano al client di cercare altrove, sia in un URL differente sia nella cache (discussa in seguito). I codici 4xx indicano che la richiesta è fallita a causa di un errore del client, come una richiesta non valida o una pagina inesistente. Infine, gli errori 5xx indicano che il server stesso ha un problema, dovuto a un errore nel proprio codice o a un sovraccarico temporaneo.

Le intestazioni del messaggio

La riga della richiesta (per esempio la riga con il metodo *GET*) può essere seguita da altre righe con ulteriori informazioni. Sono chiamate **intestazioni di richiesta**. Queste informazioni si possono assimilare ai parametri della chiamata a una procedura. Anche le risposte possono avere **intestazioni di risposta**. Alcune intestazioni possono essere utilizzate in entrambe le direzioni; un riepilogo delle più importanti è mostrato nella Figura 7.43.

L'intestazione *User-Agent* è usata dal client per passare al server informazioni relative al browser, al sistema operativo e altre proprietà. Nella Figura 7.34 abbiamo visto che il server dispone per magia di queste informazioni, e può produrle a richiesta in uno script PHP. Questa intestazione è utilizzata dal client per fornire al server le informazioni.

Le quattro intestazioni *Accept* comunicano al server ciò che il client è in grado di accettare, se dispone di un insieme limitato di elementi accettabili. La prima intestazione specifica i tipi MIME accettati (per esempio *text/html*); la seconda indica il set di caratteri (per esempio ISO-8859-5 o Unicode-1-1). La terza ha a che fare con il metodo di compressione (per esempio *gzip*), mentre la quarta indica una lingua naturale (per esempio lo spagnolo). Se il server ha più pagine tra cui scegliere, può utilizzare questa informazione per fornire ciò che il client sta cercando. Se non è in grado di soddisfare la richiesta, viene restituito un codice di errore e la richiesta fallisce. L'intestazione *Host* denomina il server. È presa dall'URL ed è obbligatoria [NdR - a partire dalla versione 1.1 di HTTP]. Viene utilizzata perché alcuni indirizzi IP possono servire più nomi DNS, e il server deve in qualche modo capire quale host deve gestire la richiesta. L'intestazione *Authorization* è necessaria per le pagine protette. In questa situazione il client può avere l'obbligo di dimostrare che possiede i diritti per vedere la pagina richiesta; per fare ciò si usa l'intestazione. Anche se i cookie sono esaminati in RFC 2109 anziché in RFC 2616, dispongono anch'essi di due intestazioni. L'intestazione *Cookie* è utilizzata dai client per restituire al server un cookie, inviato in precedenza da qualche macchina nel dominio del server.

Lo strato applicazione

Intestazione	Tipo	Contenuto
User-Agent	Richiesta	Informazioni sul browser e sulla piattaforma
Accept	Richiesta	Il tipo di pagine che il client può gestire
Accept-Charset	Richiesta	I set di caratteri accettabili per il client
Accept-Encoding	Richiesta	Le codifiche di pagina che il client può gestire
Accept-Language	Richiesta	Le lingue naturali che il client può gestire
Host	Richiesta	Il nome DNS del server
Authorization	Richiesta	Un elenco delle credenziali del client
Cookie	Richiesta	Invia al server un cookie impostato in precedenza
Date	Entrambi	La data e l'ora di invio del messaggio
Upgrade	Entrambi	Il protocollo che il mittente desidera usare
Server	Risposta	Le informazioni sul server
Content-Encoding	Risposta	Come viene codificato il contenuto (per esempio con <i>gzip</i>)
Content-Language	Risposta	La lingua naturale utilizzata nella pagina
Content-Length	Risposta	La lunghezza della pagina in byte
Content-Type	Risposta	Il tipo MIME della pagina
Last-Modified	Risposta	La data e l'ora in cui la pagina è stata modificata
Location	Risposta	Un comando che indica al client di inviare altrove la sua richiesta
Accept-Ranges	Risposta	Il server accetterà richieste di intervalli di byte
Set-Cookie	Risposta	Il server desidera che il client salvi un cookie

Figura 7.43. Alcune intestazioni dei messaggi HTTP.

L'intestazione *Date* può essere utilizzata in entrambe le direzioni e contiene la data e l'ora di invio del messaggio. L'intestazione *Upgrade* viene utilizzata per facilitare la transizione a una futura versione (magari incompatibile) del protocollo HTTP. Consente al client di annunciare cosa è in grado di supportare, e al server di affermare che cosa sta utilizzando.

Passiamo ora alle intestazioni utilizzate solo dal server in risposta alle richieste. La prima, *Server*, consente al server di affermare la sua identità e di comunicare alcune delle sue proprietà.

Le quattro intestazioni successive, che iniziano tutte con *Content-*, permettono al server di descrivere le proprietà della pagina che sta inviando.

L'intestazione *Last-Modified* comunica quando è stata modificata per l'ultima volta una pagina. Questa intestazione gioca un ruolo importante nel caching della pagina.

L'intestazione *Location* è utilizzata dal server per informare il client che dovrebbe provare un URL diverso. Può essere utilizzata se la pagina è stata spostata, o per consentire che più URL facciano riferimento alla stessa pagina (possibilmente su server diversi). È utilizzata anche dalle società che hanno una pagina Web principale nel dominio *.com*, ma che reindirizzano i client verso una pagina nazionale o regionale in base all'indirizzo IP o alla lingua preferita.

Se una pagina è molto grande, un client di capacità limitate potrebbe non volere vederla tutta in una volta. Alcuni server accettano richieste di intervalli di byte, in modo che la pagina possa essere prelevata in più unità piccole. L'intestazione *Accept-Ranges* annuncia la disponibilità del server a gestire questo tipo di richieste parziali di pagina.

La seconda intestazione per i cookie, *Set-Cookie*, permette ai server di inviare i cookie ai client. Il client deve salvare il cookie e restituirlo al server nelle richieste successive.

Un esempio di utilizzo di HTTP

Visto che HTTP è un protocollo ASCII, è abbastanza facile per una persona a un terminale (o anche per un browser) comunicare direttamente con i server Web. È sufficiente una connessione TCP alla porta 80 sul server. I lettori sono incoraggiati a provare questo scenario personalmente (preferibilmente da un sistema UNIX, perché molti altri sistemi non restituiscono lo stato della connessione). Provare la sequenza di comandi mostrata di seguito:

```
telnet www.ietf.org 80 >log
GET /rfc.html HTTP/1.1
Host: www.ietf.org

close
```

La sequenza di comandi avvia una connessione telnet (o TCP) alla porta 80 del server Web di IETF, www.ietf.org. Il risultato della sessione viene reindirizzato al file *log* per una successiva analisi. Segue il comando *GET* per dare un nome al file e al protocollo. La riga successiva è l'intestazione *Host* obbligatoria. È richiesta anche la riga vuota, che segnala al server che non vi sono altre intestazioni di richiesta. Il comando *close* indica al programma telnet di interrompere la connessione [NdR - si noti che il comando *close* non è un comando HTTP, quindi non viene trasmesso sulla connessione TCP. Corrisponde invece ad un comando dato all'applicazione telnet per iniziare la chiusura della connessione (la sequenza di tasti da premere potrebbe essere ad esempio *Ctrl-/* per entrare in modalità comando, seguito da *close*)]. In conseguenza del comando *close* l'applicazione telnet invia il segmento FIN con la richiesta di chiusura della connessione].

Il file *log* può essere esaminato con qualsiasi editor. Dovrebbe iniziare in modo simile al listato della Figura 7.44, a meno che IETF non lo abbia cambiato di recente.

Le prime tre righe sono l'output del programma telnet, non del sito remoto. La riga che inizia con HTTP/1.1 è la risposta di IETF, che comunica la possibilità di parlare in HTTP/1.1 con l'utente. Seguono diverse intestazioni e poi il contenuto. Abbiamo visto tutte le intestazioni a eccezione di *ETag*, un identificatore di pagina univoco relativo al caching, e *X-Pad*, che non è standard e serve probabilmente per alcuni browser ricchi di bug.

7.3.5 Il miglioramento delle prestazioni

La popolarità del Web è sempre stata la sua rovina. Server, router e linee sono spesso sovraccarichi. Molte persone hanno iniziato a chiamare il WWW con il nome World Wide Wait ("wait" in inglese significa "aspettare"). Come conseguenza di questi ritardi interminabili, i ricercatori hanno sviluppato varie tecniche per migliorare le prestazioni. Ora ne esamineremo tre: il caching, la replica del server e le reti di consegna del contenuto.

```
Trying 4.17.168.6...
Connected to www.ietf.org.
Escape character is ']'.
HTTP/1.1 200 OK
Date: Wed, 08 May 2002 22:54:22 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.5a
Last-Modified: Mon, 11 Sep 2000 13:56:29 GMT
ETag: "2a79d-c8b-39bce48d"
Accept-Ranges: bytes
Content-Length: 3211
Content-Type: text/html
X-Pad: avoid browser bug

<html>
<head>
<title>IETF RFC Page</title>

<script language="javascript">
function url() {
var x = document.form1.number.value
if (x.length == 1) {x = "000" + x}
if (x.length == 2) {x = "00" + x}
if (x.length == 3) {x = "0" + x}
document.form1.action = "/rfc/rfc" + x + ".txt"
document.form1.submit
}
</script>
</head>
```

Figura 7.44. L'inizio dell'output di www.ietf.org/rfc.html.

Il caching

Un modo piuttosto semplice per migliorare le prestazioni consiste nel salvare le pagine richieste, per riutilizzarle in seguito. Questa tecnica è particolarmente efficiente con pagine che vengono visitate molto spesso, come www.yahoo.com e www.cnn.com. L'accumulo delle pagine per un utilizzo successivo è definito **caching**.

La normale procedura coinvolge un processo, chiamato **proxy**, che deve mantenere la cache. Per utilizzare il caching, occorre configurare un browser in modo che tutte le richieste di pagina vengano rivolte al proxy, anziché al server reale. Se il proxy possiede la pagina, questa viene restituita immediatamente; in caso contrario la pagina viene prelevata dal server, aggiunta alla cache per l'utilizzo futuro e restituita al client che l'ha richiesta.

Due domande importanti relative al caching sono le seguenti:

1. chi dovrebbe svolgere il caching?
2. per quanto tempo le pagine devono essere archiviate nella cache?

Esistono diverse risposte alla prima domanda. I singoli PC spesso eseguono dei proxy, in modo da poter cercare rapidamente nella pagine visitate in precedenza. Su una LAN aziendale, il proxy è spesso una macchina condivisa da tutte le macchine sulla LAN, pertanto se un utente cerca una determinata pagina e un altro utente della stessa LAN desidera la medesima pagina, entrambi possono recuperarla dalla cache del proxy. Anche molti ISP gestiscono proxy per accelerare l'accesso da parte dei loro clienti. Spesso tutte queste cache lavorano in contemporanea: le richieste vengono per prima cosa inviate al proxy locale; se questo fallisce, il proxy locale interroga il proxy della LAN, e in caso di fallimento il proxy della LAN interroga il proxy dell'ISP. L'operazione successiva è il recupero della pagina dalla cache dell'ISP, da uno di livello superiore o dal server vero e proprio. Uno schema che coinvolge più cache in sequenza è definito **caching gerarchico**. Una possibile implementazione è mostrata nella Figura 7.45.

Stabilire per quanto tempo le pagine devono essere archiviate nella cache è più complesso. Alcune pagine non dovrebbero mai essere archiviate, per esempio una pagina con i prezzi delle 50 azioni più attive al momento (perché cambia ogni secondo). Se fossero memorizzati nella cache, un utente che preleva una copia dalla cache otterrebbe dati stan-

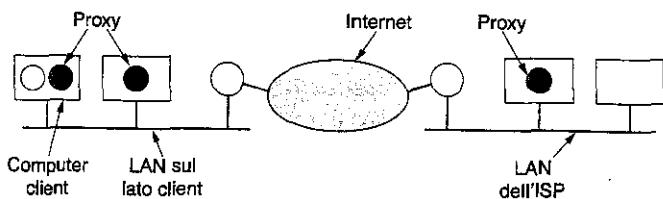


Figura 7.45. Il caching gerarchico con tre proxy.

ti (vale a dire obsoleti). D'altra parte, una volta chiuso lo scambio di azioni per la giornata, la pagina rimane valida per ore o giorni, fino all'inizio della successiva sessione di contrattazioni. L'archiviabilità di una pagina nella cache può quindi variare nel tempo. La questione fondamentale da risolvere, quando bisogna stabilire se una pagina va eliminata dalla cache, sta nel determinare la quantità di "vecchiaia" sopportata dagli utenti; infatti, poiché le pagine della cache sono conservate su disco, lo spazio consumato diventa raramente un problema. Se un proxy elimina rapidamente le pagine, restituirà raramente una pagina stantia ma non sarà nemmeno efficiente (perché troverà poche corrispondenze). Se conserva le pagine troppo a lungo, potrebbe trovare molte corrispondenze, restituendo però frequentemente pagine stantie.

Esistono due approcci per risolvere il problema. Il primo utilizza l'euristica per indovinare per quanto tempo tenere ogni pagina. Una scelta comune prevede di basare il tempo di conservazione sull'intestazione *Last-Modified* (Figura 7.43). Se una pagina è stata modificata un'ora fa, viene conservata nella cache per un'ora. Se è stata modificata un anno fa, si tratta senza dubbio di una pagina molto stabile (per esempio un elenco degli dei della

mitologia greca e romana), pertanto può essere archiviata nella cache per un anno, presumendo che non cambi. Anche se l'euristica spesso funziona bene in pratica, di tanto in tanto restituisce pagine stantie.

L'altro approccio è più costoso, ma elimina la possibilità di pagine stantie utilizzando alcune caratteristiche speciali di RFC 2616 che hanno a che fare con la gestione della cache. Una delle caratteristiche più utili è l'intestazione di richiesta *If-Modified-Since*, che un proxy può inviare a un server. Specifica la pagina desiderata dal proxy e l'ora in cui la pagina nella cache è stata modificata per l'ultima volta (dall'intestazione *Last-Modified*). Se da quel momento la pagina non è più stata modificata, il server restituisce un breve messaggio *Not Modified* (codice di stato 304 nella Figura 7.42), che indica al proxy di utilizzare le pagine archiviate nella cache. Se invece la pagina è stata modificata, viene restituita la nuova pagina. Questo approccio richiede sempre un messaggio di richiesta e uno di risposta, ma il messaggio di risposta sarà molto breve quando la voce nella cache è ancora valida.

Questi due approcci possono essere facilmente combinati. Per il primo *DT* dopo il prelievo della pagina, il proxy la restituisce al client che l'ha richiesta. Dopo che la pagina è disponibile da qualche tempo, il proxy utilizza i messaggi *If-Modified-Since* per controllarne la freschezza. La scelta del valore di *DT* implica inevitabilmente l'uso di euristica, basata sul tempo da cui la pagina è stata modificata per l'ultima volta.

Le pagine Web che hanno contenuto dinamico (per esempio generate da uno script PHP) non dovrebbero mai essere messe nella cache, perché i parametri possono cambiare di volta in volta. Per gestire questo e altri casi, esiste un meccanismo generale utilizzabile da un server per istruire tutti i proxy sul percorso verso il client, comunicando loro di non utilizzare mai la pagina corrente senza prima verificarne la freschezza. Questo meccanismo può essere utilizzato anche per qualsiasi pagina che dovrebbe cambiare rapidamente. In RFC 2616 sono definiti molti altri meccanismi di controllo della cache.

Un altro approccio per migliorare le prestazioni è il caching proattivo. Quando un proxy preleva una pagina da un server, può analizzarla per vedere se contiene collegamenti ipertestuali. In questo caso, può emettere richieste ai server corrispondenti per precaricare la cache con le pagine indicate, che magari saranno richiamate in seguito. Questa tecnica può ridurre il tempo di accesso delle richieste successive, ma può anche intasare le linee di comunicazione con pagine che non saranno mai necessarie.

Chiaramente, il Web caching non è semplice, e sull'argomento vi sarebbe molto da dire. In effetti sono stati scritti molti libri relativi solo a questo soggetto, per esempio (Rabinovich e Spatscheck, 2002; Wessels, 2001). Ora è tempo di passare all'argomento successivo.

La replica del server

Il caching è una tecnica sul lato client per migliorare le prestazioni; tuttavia, esistono anche tecniche da eseguire sul lato server. L'approccio più comune per migliorare le prestazioni dei server richiede di replicare il loro contenuto in più posizioni distinte e lontane tra loro. Questa tecnica è detta a volte **mirroring**.

Un tipico utilizzo del mirroring riguarda la pagina Web principale di una società, che contiene poche immagini con collegamenti, per esempio, ai siti Web per le aree occidentale, orientale, settentrionale e meridionale. L'utente può fare clic su quello più vicino; da quel momento tutte le richieste vengono rivolte al server selezionato.

I siti mirror sono generalmente statici. La società decide dove vuole posizionare i mirror, organizza un server in ogni regione e inserisce in ciascuno più o meno tutto il contenuto (magari omettendo le pale da neve dal sito di Miami e i surf da quello dell'Alaska). La scelta dei siti resta di norma stabile per mesi o anni.

Sfortunatamente, il Web subisce un fenomeno noto come **flash crowds**, in cui un sito Web che in precedenza rappresentava una zona depressa e sconosciuta diventa improvvisamente il centro dell'universo conosciuto. Per esempio, fino al 6 novembre 2000, il sito Web del Segretario di stato della Florida, www.dos.state.fl.us, forniva brevi informazioni sulle riunioni del Governo di Stato della Florida e istruzioni su come divenire notai in Florida. Tuttavia, dal 7 novembre 2000, quando il voto per la presidenza degli Stati Uniti poteva essere ribaltato da poche centinaia di voti delle contee della Florida, divenne rapidamente uno dei cinque siti Web più visitati al mondo. Inutile dire che non resse il carico di lavoro.

È quindi necessario un modo che permetta al sito Web che nota un aumento massiccio del traffico di riprodursi in più posizioni, mantenendo questi siti operativi fino al passaggio della tempesta e disattivandoli subito dopo (tutti o solo in parte). Per disporre di questa capacità è necessario un accordo anticipato con alcune società che dispongono di molti siti di hosting, che preveda la possibilità di creare repliche su richiesta e di pagare lo spazio effettivamente utilizzato.

Una strategia ancora più flessibile prevede la creazione di repliche dinamiche di singole pagine, a seconda del luogo da cui proviene il traffico. Alcune ricerche sull'argomento sono riportate in (Pierre et al., 2001; Pierre et al., 2002).

Le reti di consegna del contenuto

La forza del capitalismo è dimostrata dal fatto che qualcuno ha scoperto come fare soldi anche con il World Wide Wait. Funziona in questo modo. Alcune società chiamate CDN (Content Delivery Network, reti di consegna del contenuto) si accordano con i provider di contenuti (siti musicali, quotidiani e altri siti che desiderano che il loro contenuto sia disponibile in modo facile e veloce) e promettono di consegnare questi contenuti in modo efficiente in cambio di una piccola quota. Dopo la firma del contratto, il proprietario consegna alla CDN il contenuto del suo sito Web per la pre-elaborazione (discussa in seguito) e quindi la distribuzione.

A questo punto, la CDN parla con molti ISP e si offre di pagare per ottenere l'autorizzazione a inserire un server gestito in remoto sulle loro LAN. Questa non è solo una fonte di reddito, ma offre anche ai clienti dell'ISP un eccellente tempo di risposta per il recupero del contenuto della CDN, dando così all'ISP un vantaggio competitivo su altri ISP che non collaborano con la CDN. A queste condizioni, la sottoscrizione di un contratto con

una CDN è più che conveniente per l'ISP. Di conseguenza, le CDN più grandi dispongono di più di 10.000 server distribuiti in tutto il mondo.

Con il contenuto replicato in migliaia di siti nel mondo, vi sono ottime probabilità di migliorare le prestazioni. Tuttavia, affinché la tecnica funzioni, occorre un modo per reindirizzare le richieste del client al server CDN più vicino, preferibilmente quello collocato presso l'ISP del client. Inoltre, questo reindirizzamento deve essere svolto senza modificare DNS o altre parti dell'infrastruttura standard di Internet. Ecco una descrizione notevolmente semplificata dell'operato di Akamai, la CDN più grande.

L'intero processo inizia quando il provider di contenuti consegna alla CDN il suo sito Web. La CDN passa ogni pagina in un preprocessore che sostituisce tutti gli URL con indirizzi modificati. Il modello operativo alla base di questa strategia si basa sul fatto che il sito Web del provider di contenuti è composto da molte pagine piccole (solo testo HTML), ma che queste pagine sono spesso collegate a file grandi, come immagini, audio e video. Le pagine HTML modificate sono memorizzate sul server del provider di contenuti e sono recuperate nel solito modo; le immagini, l'audio e il video vengono invece trasferiti sui server della CDN.

Per vedere il reale funzionamento di questo schema, si consideri la pagina Web di Furry Video nella Figura 7.46(a). Dopo la pre-elaborazione, viene trasformata come appare nella Figura 7.46(b) e posta sul server di Furry Video con l'indirizzo www.furryvideo.com/index.html.

Quando un utente digita l'URL www.furryvideo.com, DNS restituisce l'indirizzo IP del sito Web di Furry Video, consentendo il normale recupero della pagina principale (HTML). Quando l'utente fa clic su uno dei collegamenti ipertestuali, il browser chiede a DNS di cercare cdn-server.com. Il browser invia quindi una richiesta HTTP a questo indirizzo IP, pensando di ottenere un file MPEG.

Questo non avviene, perché cdn-server.com non ospita alcun contenuto. Infatti, si tratta del server HTTP fittizio della CDN. Esamina il nome file e il nome del server per scoprire quale pagina è necessaria, e presso quale provider di contenuti. Esamina anche l'indirizzo IP della richiesta in ingresso e cerca nel suo database per determinare dove potrebbe trovarsi l'utente. Armata di queste informazioni, determina quale dei server di contenuto della CDN può offrire all'utente il servizio migliore. La decisione è difficile perché quello più vicino geograficamente potrebbe non essere il più vicino in termini di topologia di rete, mentre il più vicino in termini di topologia di rete può essere molto occupato al momento. Dopo avere eseguito una scelta, cdn-server.com restituisce una risposta con il codice di stato 301 e un'intestazione *Location* con l'URL del file sul server di contenuto della CDN più vicino al client. Per questo esempio, presumiamo che l'URL sia www.CDN-0420.com/furryvideo/bears.mpg. Il browser elabora questo URL nel solito modo per recuperare il file MPEG vero e proprio.

I passaggi necessari sono illustrati nella Figura 7.47. Il primo passo consiste nel cercare www.furryvideo.com per ottenere il suo indirizzo IP. A questo punto, la pagina HTML può essere prelevata e visualizzata nel solito modo. La pagina contiene tre collegamenti ipertestuali a [cdn-server](http://cdn-server.com), come evidenziato dalla Figura 7.46(b). Quando (per esempio) viene

```

<html>
<head> <title> Furry Video </title> </head>
<body>
<h1> Elenco di prodotti di Furry Video </h1>
<p> Fate clic per alcuni campioni gratuiti. </p>

<a href="bears.mpg"> Orsacchiotti </a> <br>
<a href="bunnies.mpg"> Coniglietti </a> <br>
<a href="mice.mpg"> Topolini </a> <br>
</body>
</html>

```

(a)

```

<html>
<head> <title> Furry Video </title> </head>
<body>
<h1> Elenco di prodotti di Furry Video </h1>
<p> Fate clic per alcuni campioni gratuiti. </p>

<a href="http://cdn-server.com/furryvideo/bears.mpg"> Orsacchiotti </a> <br>
<a href="http://cdn-server.com/furryvideo/bunnies.mpg"> Coniglietti </a> <br>
<a href="http://cdn-server.com/furryvideo/mice.mpg"> Topolini </a> <br>
</body>
</html>

```

(b)

Figura 7.46. (a) La pagina Web originale. (b) La stessa pagina dopo la trasformazione.

selezionato il primo, il suo indirizzo DNS viene ricercato (passo 5) e restituito (passo 6). Quando una richiesta di *bears.mpg* viene inviata a *cdn-server* (passo 7), al client viene comunicato di passare a *CDN-0420.com* (passo 8). Quando svolge l'operazione indicata (passo 9), gli viene consegnato il file dalla cache del proxy (passo 10). La proprietà che permette al meccanismo di funzionare sta nel passo 8, vale a dire nella presenza del server HTTP fittizio che reindirizza il client a un proxy CDN vicino al client.

Il server CDN verso cui è rediretto il client è di norma un proxy, con una grande cache pre-caricata con il contenuto più importante. Se qualcuno chiede un file che non si trova nella cache, questo viene prelevato dal vero server e posto nella cache per un utilizzo successivo. Trasformando il server di contenuto in un proxy, anziché eseguire una replica completa, la CDN ha la possibilità di creare compromessi tra la dimensione del disco, il tempo di precaricamento e i parametri relativi alle prestazioni.

Per ulteriori informazioni sulle reti di consegna del contenuto, vedere (Hull, 2002; Rabinovich e Spatscheck, 2002).

7.3.6 Il Web wireless

C'è un considerevole interesse verso i piccoli dispositivi portatili in grado di accedere al Web tramite un collegamento wireless. In effetti, i primi passi in questa direzione sono già

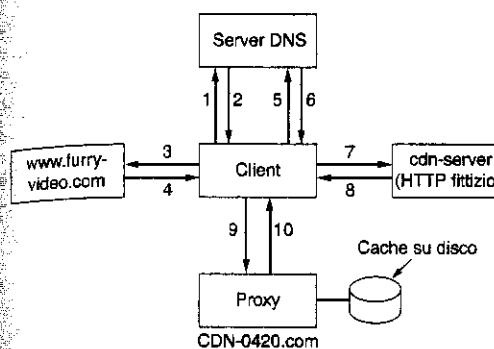


Figura 7.47. I passaggi per la ricerca di un URL quando viene utilizzata una CDN.

stati compiuti. Senza dubbio negli anni a venire questo settore subirà molte modifiche, ma per vedere a che punto siamo vale comunque la pena di esaminare alcune delle idee attuali relative al Web wireless. Ci concentreremo sulle prime due grandi aree dei sistemi Web wireless che hanno raggiunto il mercato: WAP e i-mode.

WAP (Wireless Application Protocol)

Quando Internet e i telefoni cellulari divennero molto comuni, non trascorse molto tempo prima che qualcuno avesse l'idea di unire insieme i due concetti, creando un telefono cellulare capace di accedere in modalità wireless alla posta elettronica e al Web. "Qualcuno" in questo caso era un consorzio cui inizialmente appartenevano Nokia, Ericsson, Motorola e phone.com (formalmente nota come Unwired Planet), e che attualmente conta centinaia di membri. Il sistema è chiamato **WAP** (Wireless Application Protocol).

Un dispositivo WAP può essere un telefono cellulare avanzato, un PDA o un computer notebook privo di funzionalità vocali. Le specifiche comprendono questi e altri dispositivi. L'idea di base è utilizzare l'infrastruttura wireless digitale esistente. Gli utenti possono letteralmente chiamare un gateway WAP attraverso il collegamento wireless e inviare ad esso le richieste di pagine Web. Il gateway dapprima controlla nella sua cache se è presente la pagina richiesta per inviarla, altrimenti la preleva dalla Internet cablata. In sostanza, questo significa che WAP 1.0 è un sistema a commutazione di circuito con alti oneri di connessione per minuto. Le persone non hanno apprezzato la possibilità di accedere a Internet da un piccolo schermo pagando ogni minuto di connessione, e per questo motivo WAP è stato praticamente un flop (ma c'erano anche altri problemi). Tuttavia, WAP e il suo rivale, i-mode (discusso in seguito), sembrano convergere verso una tecnologia simile: WAP 2.0 potrà quindi essere un grande successo. Dal momento che WAP 1.0 è stato il primo tentativo di Internet wireless, vale la pena descriverlo almeno brevemente.

WAP è essenzialmente una pila di protocolli per l'accesso al Web, ottimizzata per connessioni a banda ridotta utilizzando dispositivi wireless con una CPU lenta, poca memoria e uno schermo piccolo. Questi requisiti sono ovviamente diversi da quelli dello scenario standard con PC desktop, pertanto esistono alcune differenze tra i protocolli. Gli strati sono mostrati nella Figura 7.48.

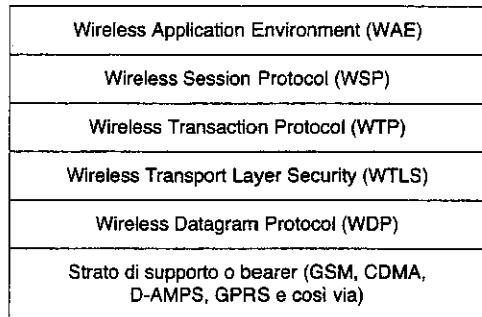


Figura 7.48. La pila di protocolli WAP.

Lo strato più basso supporta tutti i sistemi telefonici mobili esistenti, compresi GSM, D-AMPS e CDMA. La velocità dei dati con WAP 1.0 è pari a 9600 bps. Sopra questo strato si trova il protocollo per i datagrammi, **WDP** (*Wireless Datagram Protocol*), che equivale essenzialmente a UDP. Segue uno strato per la protezione, naturalmente necessario in un sistema wireless. WTLS è un sottoinsieme di SSL di Netscape, che sarà presentato nel Capitolo 8. Sopra questo si trova uno strato per le transazioni, che gestisce richieste e risposte in modo affidabile o inaffidabile. Questo strato sostituisce TCP, che non viene utilizzato per i collegamenti "aerei" per motivi di efficienza. Segue uno strato di sessione simile a HTTP/1.1, ma con alcune limitazioni e modifiche a scopo di ottimizzazione. In cima si trova il microbrowser (WAE).

A parte il costo, l'altro aspetto che ha sicuramente ostacolato l'accettazione di WAP è il fatto che non utilizza HTML. Lo strato WAE utilizza infatti un linguaggio di markup chiamato **WML** (*Wireless Markup Language*), che non è altro che un'applicazione di XML. Come conseguenza, in linea di principio un dispositivo WAP può accedere solo a quelle pagine che sono state convertite in WML. Tuttavia, visto che questa limitazione riduce notevolmente il valore di WAP, l'architettura ha richiesto un filtro HTML/WML per aumentare il numero di pagine disponibili. Questa architettura è illustrata nella Figura 7.49.

In tutta onestà, WAP è stato probabilmente rilasciato prima del tempo. Al suo esordio, XML non era conosciuto all'esterno di W3C, pertanto la stampa annunciava **WAP NON UTILIZZA HTML**. Un titolo più accurato avrebbe potuto essere: **WAP UTILIZZA GIÀ IL NUOVO STANDARD HTML**. Una volta compiuto il danno, però, fu difficile ripararlo e WAP 1.0 non ebbe mai successo. Rivisiteremo WAP dopo avere analizzato il suo principale rivale.

Lo strato applicazione

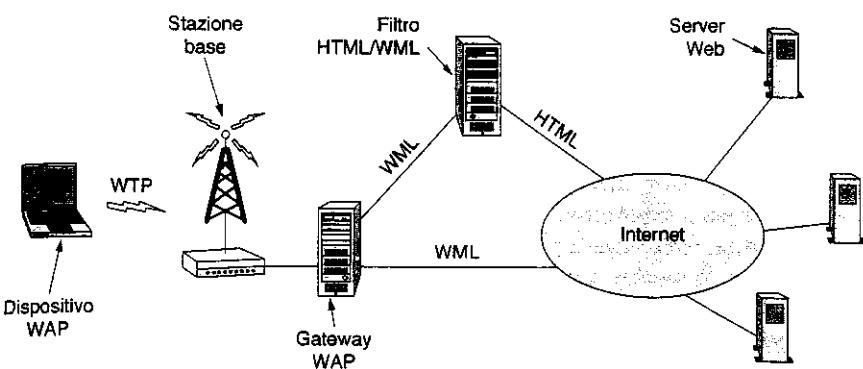


Figura 7.49. L'architettura di WAP.

I-mode

Mentre un consorzio di società informatiche e di telecomunicazioni era occupato a sviluppare uno standard aperto utilizzando la più avanzata versione di HTML disponibile, in Giappone venivano sviluppate altre tecnologie. Una donna giapponese, Mari Matsunaga, inventò un approccio diverso al Web wireless chiamato **i-mode** (information-mode). Riuscì a convincere la consociata wireless della compagnia telefonica giapponese che il suo approccio era corretto; nel febbraio del 1999, NTT DoCoMo (Japanese Telephone and Telegraph Company) lanciò il servizio in Giappone. Nel giro di 3 anni superò i 35 milioni di abbonati, che potevano accedere a più di 40.000 siti Web specifici per i-mode. La maggior parte delle società mondiali di telecomunicazioni parlò a lungo del suo successo finanziario, specialmente alla luce del fatto che WAP sembrava non riscuotere risultati. Diamo ora un'occhiata a i-mode e al suo funzionamento.

I-mode ha tre componenti principali: un nuovo sistema di trasmissione, un nuovo tipo di telefono e un nuovo linguaggio per la progettazione di pagine Web. Il sistema di trasmissione è composto di due reti separate: la rete telefonica mobile esistente a commutazione di circuito (comparabile con D-AMPS) e una nuova rete a commutazione di pacchetto costruita specificamente per i servizi i-mode. La modalità vocale utilizza la rete a commutazione di circuito, e il costo è calcolato per minuto di connessione. I-mode utilizza la rete a commutazione di pacchetto ed è sempre attivo (come ADSL o una connessione via cavo), pertanto non vi sono costi per il tempo di connessione ma per ogni pacchetto inviato. Attualmente non è possibile utilizzare entrambe le reti contemporaneamente.

Il telefono somiglia a un comune telefono cellulare, con l'aggiunta di un piccolo schermo. NTT DoCoMo pubblicizza i dispositivi i-mode come telefoni cellulari evoluti e non come terminali Web wireless, anche se in effetti è proprio quello che sono. Probabilmente la maggior parte dei clienti non è nemmeno consapevole di trovarsi su Internet: considera il suo dispositivo i-mode come un telefono cellulare con servizi avanzati. Questo modello

prevede che i-mode sia un servizio, pertanto i telefoni non sono programmabili dall'utente; tuttavia, contengono l'equivalente di un PC del 1995 e potrebbero probabilmente eseguire Windows 95 o UNIX.

Quando il telefono i-mode viene acceso, all'utente viene mostrato un elenco delle categorie con i servizi ufficialmente approvati. Esistono più di 1.000 servizi divisi in 20 categorie. Ogni servizio, che in pratica è un piccolo sito Web i-mode, è svolto da una società indipendente. Le categorie principali del menu ufficiale comprendono posta elettronica, notizie, previsioni del tempo, sport, giochi, shopping, mappe, oroscopi, intrattenimento, viaggi, guide regionali, suonerie, ricette, giochi d'azzardo, home banking e prezzi delle aziende. Il servizio è in qualche modo destinato ad adolescenti e giovani, che di norma adorano gli aggeggi elettronici (specialmente se hanno colori alla moda). Il solo fatto che esistano oltre 40 società che vendono suonerie dice molto. L'applicazione più popolare è la posta elettronica, che consente messaggi fino a 500 byte: è un notevole miglioramento rispetto a SMS (Short Message Service), che permette messaggi fino a 160 byte. Anche i giochi sono molto popolari.

Esistono oltre 40.000 siti Web i-mode, e per accedervi è necessario digitare l'URL corrispondente (invece di effettuare la selezione da un menu). In un certo senso, l'elenco ufficiale dei siti i-mode è come un portale Internet che consente l'accesso ad altri siti Web mediante clic, anziché tramite la digitazione di un URL.

NTT DoCoMo controlla in modo rigoroso i servizi ufficiali. Per essere incluso nell'elenco, un servizio deve soddisfare numerosi criteri noti pubblicamente. Per esempio, un servizio non deve avere un'influenza negativa sulla società, i dizionari giapponese-inglese devono contenere un numero sufficiente di parole, i servizi per le suonerie devono aggiungere spesso nuovi toni di chiamata, nessun sito deve ispirare comportamenti maniacali o riflettersi in maniera negativa su NTT DoCoMo (Frengle, 2002). I 40.000 siti Internet possono fare ciò che vogliono.

Il modello commerciale di i-mode è così diverso da quello della Internet convenzionale che vale la pena spiegarlo. La tariffa di sottoscrizione base di i-mode è pari a pochi dollari al mese. Visto che occorre pagare per ogni pacchetto ricevuto, l'abbonamento base comprende un numero di pacchetti limitato. Il cliente può anche scegliere una sottoscrizione con maggiori pacchetti gratuiti: la tariffa per pacchetto scenderà sensibilmente passando a 1 a 10 MB al mese. Se i pacchetti gratuiti vengono esauriti a metà del mese, è possibile acquistare online altri pacchetti.

Per utilizzare un servizio occorre eseguire la sottoscrizione, generalmente facendo clic su un pulsante o un collegamento e inserendo il codice PIN. La maggior parte dei servizi ufficiali costa l'equivalente di circa 1 o 2 euro al mese. NTT DoCoMo somma l'importo alla bolletta del telefono e ne trasferisce il 91% al provider di servizi, tenendo per sé il 9%. Se un servizio non ufficiale ha un milione di clienti, deve inviare ogni mese un milione di fatture di 1 euro circa; se il servizio diventa ufficiale, NTT DoCoMo gestisce la fatturazione e trasferisce ogni mese 910.000 euro sul conto bancario del servizio. Il fatto di non dover gestire la fatturazione è un incentivo che spinge i provider di servizi a divenire ufficiali, generando maggiori ricavi per NTT DoCoMo. Inoltre, l'ufficialità garantisce l'inserimento del servizio nel menu iniziale, facilitandone l'individuazione. La bolletta telefonica del-

l'utente comprende le telefonate, la tariffa di sottoscrizione a i-mode, le tariffe di sottoscrizione ai servizi e le spese per i pacchetti aggiuntivi.

Nonostante il suo successo in Giappone, non è chiaro se sia possibile ottenere lo stesso risultato in Europa o negli Stati Uniti. Per certi aspetti l'ambiente giapponese è diverso da quello dell'occidente. In primo luogo, in occidente la maggior parte dei potenziali clienti (adolescenti, studenti universitari e uomini d'affari) dispone già di un PC a casa, ottenendo quindi connessioni a Internet con una velocità di almeno 56 kbps (ma spesso superiore). In Giappone poche persone possiedono PC connessi a Internet a casa, in parte a causa della mancanza di spazio, ma soprattutto per i costi esorbitanti chiesti da NTT per i servizi telefonici locali (circa 700 euro per l'installazione di una linea e circa 1,50 euro all'ora per le chiamate locali). Per la maggior parte degli utenti, i-mode è l'unica connessione a Internet.

In secondo luogo, in Europa e in America le persone non sono abituate a pagare 1 euro al mese per accedere al sito Web della CNN, 1 euro al mese per il sito Web di Yahoo, 1 euro al mese per il sito Web di Google e così via, per non parlare delle spese aggiuntive per MB scaricato. La maggior parte dei provider Internet richiede un canone fisso indipendentemente dall'utilizzo effettivo, in gran parte come risposta alle richieste dei clienti.

Terzo, molti giapponesi utilizzano i-mode mentre si spostano da o verso il luogo di lavoro o la scuola, in treno o in metropolitana. In Europa, le persone che si spostano in treno sono molte meno che in Giappone; negli Stati Uniti sono ancora meno. L'utilizzo di i-mode a casa, accanto a un computer con un monitor da 17 pollici e una connessione ADSL a 1 Mbps con quantità illimitata di megabyte gratuiti, non ha molto senso. Ciò nonostante, nessuno ha predetto l'immensa popolarità dei telefoni cellulari, pertanto i-mode potrebbe attecchire anche nel mondo occidentale.

Come affermato in precedenza, i telefoni i-mode utilizzano la rete a commutazione di circuito esistente per la voce e una nuova rete a commutazione di pacchetto per i dati. La rete per i dati è basata su CDMA e trasmette pacchetti di 128 byte a 9.600 bps. Uno schema della rete è presentato nella Figura 7.50. I telefoni utilizzano LTP (*Lightweight Transport Protocol*) per il collegamento aereo a un gateway di conversione del protocollo. Il gateway dispone di una connessione in fibra ottica a banda larga verso il server i-mode, che è connesso a tutti i servizi. Quando l'utente seleziona un servizio dal menu ufficiale, la richiesta viene inviata al server i-mode, che archivia nella cache la maggior parte delle pagine per migliorare le prestazioni. Le richieste ai siti non presenti nel menu ufficiale oltrepassano il server i-mode e si collegano direttamente a Internet.

I telefoni attuali possiedono CPU 100 a MHz, diversi megabyte di Flash ROM, circa 1 MB di RAM e un piccolo schermo incorporato. I-mode richiede che lo schermo abbia una dimensione di almeno 72 x 94 pixel, ma alcuni dispositivi high-end utilizzano schermi di 120 x 160 pixel. Gli schermi utilizzano in genere profondità di colore a 8 bit (che permette quindi 256 colori). Non è sufficiente per le fotografie, ma è adeguata per disegni e semplici cartoni animati. Visto che non esiste il mouse, la navigazione sullo schermo viene eseguita con i tasti di direzione.

La struttura del software è mostrata nella Figura 7.51. Lo strato inferiore consiste di un semplice sistema operativo in tempo reale per il controllo dell'hardware. Segue un modulo

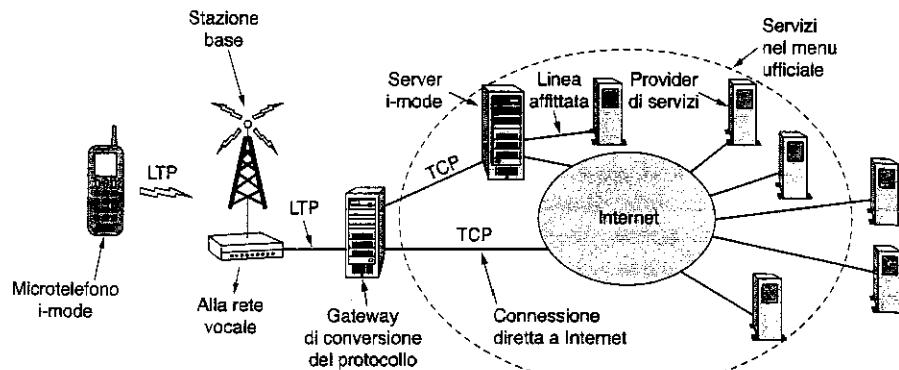


Figura 7.50. La struttura della rete dati di i-mode con i protocolli di trasporto.

per eseguire la comunicazione di rete, che utilizza il protocollo LTP di proprietà di NTT DoCoMo. Al di sopra c'è un semplice gestore di finestre che si occupa del testo e della grafica (file GIF). Con schermi che non superano i 120 x 160 pixel, non c'è altro da gestire.



Figura 7.51. La struttura del software i-mode.

Il quarto strato contiene l'interprete delle pagine Web (vale a dire il browser). I-mode non utilizza HTML ma un suo sottoinsieme chiamato **cHTML** (compact HTML, HTML compatto), basato su HTML 1.0. Questo strato consente anche l'utilizzo di applicazioni helper e plug-in, in modo simile ai browser per PC. Un'applicazione helper standard è un interprete per una versione leggermente modificata di JVM.

Nella parte superiore si trova un modulo di interazione con l'utente, che si occupa della comunicazione con l'utente.

Ora osserviamo più da vicino cHTML, che per quanto detto equivale approssimativamente ad HTML 1.0, con poche omissioni e alcune estensioni per l'utilizzo su un telefono mobile. È stato inviato a W3C per la standardizzazione, ma W3C ha mostrato poco interesse verso questo linguaggio, che probabilmente resterà un prodotto proprietario.

Il linguaggio accetta la maggior parte dei tag HTML di base, compresi <html>, <head>, <title>, <body>, <hn>, <center>, , , <menu>, ,
, <p>, <hr>, , <form> e <input>. I tag e <i> non sono permessi.

Il tag <a> è utilizzato per il collegamento ad altre pagine, ma con lo schema aggiuntivo *tel* per la composizione dei numeri telefonici. In tal senso *tel* è analogo a *mailto*. Quando viene selezionato un collegamento ipertestuale che utilizza lo schema *mailto*, il browser visualizza un modulo per inviare un messaggio di posta elettronica alla destinazione indicata nel collegamento. Quando viene selezionato un collegamento ipertestuale che utilizza lo schema *tel*, il browser compone il numero telefonico. Per esempio, una rubrica potrebbe contenere semplicemente le fotografie di varie persone. Selezionandone una, il telefono farebbe partire una chiamata a quella persona. RFC 2806 prende in esame gli URL telefonici.

Il browser cHTML è limitato in altri modi. Non supporta JavaScript, i frame, i fogli di stile, i colori di sfondo e le immagini di sfondo. Non supporta nemmeno le immagini JPEG, perché richiedono troppo tempo per la decompressione. Le applet Java sono consentite, ma sono attualmente limitate a 10 KB a causa della ridotta velocità di trasmissione sul collegamento aereo.

Anche se NTT DoCoMo ha rimosso alcuni tag HTML, ne ha aggiunti altri nuovi. Il tag <blink> fa lampeggiare il testo attivandolo e disattivandolo. Anche se può sembrare incoerente proibire (per il presupposto che i siti Web non dovrebbero gestire l'aspetto) e poi aggiungere <blink> (che ha a che fare solo con l'aspetto), questi sono i fatti. Un altro nuovo tag è <marquee>, che fa scorrere il suo contenuto sullo schermo in modo simile agli annunci di borsa su certe insegne.

Un'altra nuova caratteristica è l'attributo *align* del tag
. È necessario perché, con uno schermo tipico di 6 righe per 16 caratteri, esiste un elevato pericolo che le parole vengano spezzate, come mostrato nella Figura 7.52(a). *Align* aiuta a ridurre questo problema per ottenere qualcosa di simile alla Figura 7.52(b). È interessante notare che il giapponese non risente del problema delle parole divise su più righe. Per il testo kanji, lo schermo viene diviso in una griglia rettangolare di celle di 9 x 10 pixel o di 12 x 12 pixel, a seconda del tipo di carattere supportato. Ogni cella contiene esattamente un carattere kanji, che è l'equivalente di una parola in inglese. Le interruzioni di riga tra le parole sono sempre consentite in giapponese.

The time has com
e the walrus sai
d to talk of man
y things. Of sho
es and ships and
sealing wax of c

(a)

The time has
come the walrus
said to talk of
many things. Of
shoes and ships
and sealing wax

(b)

Figura 7.52. Un brano di Lewis Carroll su uno schermo 16 x 6.

Anche se la lingua giapponese ha decine di migliaia di kanji, NTT DoCoMo ne ha inventati 166 nuovi, chiamati **emoji**, con un alto fattore di divertimento: si tratta in pratica di pittogrammi come gli smiley della Figura 7.6. Comprendono simboli per segni zodiacali,

birra, hamburger, parchi di divertimento, compleanni, telefoni cellulari, cani, gatti, Natale, cuori spezzati, baci, stati d'animo, sonno e così via.

Un altro nuovo attributo dà agli utenti la possibilità di selezionare i collegamenti ipertestuali utilizzando la tastiera: è chiaramente una proprietà importante su un computer privo di mouse. Un esempio di utilizzo di questo attributo è mostrato nel file cHTML della Figura 7.53.

```
<html>
<body>
<h1> Selezionare un'opzione </h1>
<a href="messages.chtml" accesskey="1"> Controlla posta vocale </a> <br>
<a href="mail.chtml" accesskey="2"> Controlla posta elettronica </a> <br>
<a href="games.chtml" accesskey="3"> Partecipa a un gioco </a>
</body>
</html>
```

Figura 7.53. Un file cHTML.

Anche se il lato client è in un certo senso limitato, il server i-mode è un computer completo. Supporta CGI, Perl, PHP, JSP, ASP e tutte le tecnologie normalmente supportate dai server Web.

Un breve confronto tra WAP e i-mode, come sono implementati attualmente nei sistemi di prima generazione, è mostrato nella Figura 7.54. Anche se alcune differenze possono sembrare minime, spesso sono importanti. Per esempio, i quindicenni non possiedono carte di credito, pertanto la possibilità di acquistare prodotti online addebitando l'importo sulla bolletta del telefono può far propendere l'interesse verso un sistema piuttosto che verso l'altro. Per ulteriori informazioni su i-mode, consultare (Frengle, 2002; Vacca, 2002).

Il Web wireless di seconda generazione

WAP 1.0, basato su standard internazionali riconosciuti, doveva essere uno strumento serio per uomini d'affari costretti a spostarsi spesso, ma fu un fallimento. I-mode era un giocattolo elettronico per adolescenti giapponesi che usava ovunque standard proprietari, e fu un enorme successo. Cosa accadde dopo? Ogni parte imparò qualcosa dalla prima generazione di Web wireless. Il consorzio WAP ha imparato l'importanza del contenuto: la mancata disponibilità di un elevato numero di siti Web che utilizzano il linguaggio di markup prescelto è fatale. NTT DoCoMo ha imparato che un sistema proprietario chiuso, associato a piccoli telefoni e alla cultura giapponese, non è un prodotto esportabile. La conclusione tratta da entrambe la parti è stata che, per convincere molti siti Web a trasferire il loro contenuto in un formato specifico, occorre disporre di un linguaggio di markup aperto e stabile, accettato in tutto il mondo. La guerra tra formati non è un vantaggio per il business.

Entrambi i servizi stanno per entrare nella seconda generazione della tecnologia Web wireless. WAP 2.0 è uscito per primo, pertanto lo useremo come esempio. In WAP 1.0 molte

Caratteristica	WAP	i-mode
Che cos'è	Pila di protocolli	Servizio
Dispositivi supportati	Telefono, PDA, notebook	Telefono
Accesso	Su linea commutata	"Always on"
Rete sottostante	Commutazione di circuito	Due: circuito + pacchetto
Velocità dei dati	9600 bps	9600 bps
Schermo	Monocromatico	A colori
Linguaggio di markup	WML (applicazione di XML)	cHTML
Linguaggio di scripting	WMLscript	Nessuno
Tariffe di utilizzo	Per minuto	Per pacchetto
Pagamento acquisti	Carta di credito	Bolletta del telefono
Pittogrammi	No	Sì
Standardizzazione	Open standard di WAP forum	Proprietà di NTT DoCoMo
Dove è usato	Eropa, Giappone	Giappone
Utente tipico	Uomo d'affari	Giovane donna

Figura 7.54. Confronto tra i-mode e WAP di prima generazione.

cose funzionavano bene e quindi sono state mantenute. Da una parte, WAP può essere utilizzato su una varietà di reti diverse. La prima generazione utilizzava le reti a commutazione di circuito, ma le reti a commutazione di pacchetto hanno sempre rappresentato una possibilità aggiuntiva prevista. I sistemi di seconda generazione utilizzano la commutazione di pacchetto, per esempio GPRS. Dall'altra parte, WAP inizialmente mirava al supporto di una vasta gamma di dispositivi, dai telefoni portatili ai computer notebook; anche questa caratteristica è stata conservata.

WAP 2.0 offre anche nuove funzionalità. Le più significative sono le seguenti:

1. modello "push" oltre al modello "pull"
2. supporto per l'integrazione della telefonia nelle applicazioni
3. messaggistica multimediale
4. inclusione di 264 pittogrammi
5. interfaccia verso una periferica di archiviazione
6. supporto per i plug-in nel browser.

Il modello "pull" è ben noto: il client richiede una pagina e la ottiene. Il modello "push" supporta l'arrivo di dati senza precedenti richieste, per esempio informazioni sui prezzi delle azioni o avvisi sul traffico.

Voce e dati stanno iniziando a fondersi assieme, e WAP 2.0 li supporta entrambi in diversi modi. Abbiamo già visto un esempio con la capacità di i-mode di collegare un'icona o

un testo sullo schermo a un numero telefonico da chiamare. Insieme alla posta elettronica e alla telefonia, viene supportata la messaggistica multimediale.

La grande popolarità degli emoji di i-mode ha stimolato il consorzio WAP a inventare i propri 264. Le categorie comprendono animali, apparecchiature, abbigliamento, emozioni, cibo, corpo umano, sesso, mappe, musica, piante, sport, periodi, attrezzi, veicoli, armi e tempo. È abbastanza interessante il fatto che lo standard denomina ogni pittogramma ma non fornisce l'effettiva mappa di bit, probabilmente per la paura che la rappresentazione di "addormentato" o "abbraccio" in una cultura possa apparire insultante per un'altra cultura. I-mode non aveva questo problema perché era destinato a una singola nazione.

La presenza di un'interfaccia per l'archiviazione non significa che ogni telefono WAP 2.0 include un capiente disco rigido; anche le Flash ROM sono periferiche di archiviazione. Una fotocamera wireless abilitata per WAP può utilizzare la Flash ROM per la memorizzazione temporanea delle immagini, prima di inviare le fotografie migliori su Internet. Per finire, i plug-in possono estendere le capacità del browser. Viene anche fornito un linguaggio di scripting.

In WAP 2.0 ci sono anche molte differenze tecniche. Le due principali riguardano la pila dei protocolli e il linguaggio di markup. WAP 2.0 continua a supportare la vecchia pila di protocolli della Figura 7.48, ma supporta anche quella standard di Internet con TCP e HTTP/1.1. Tuttavia, sono state apportate quattro piccole (ma compatibili) modifiche a TCP (per semplificare il codice): (1) utilizzo di una finestra fissa di 64 KB, (2) nessun avvio lento, (3) MTU massima di 1.500 byte, (4) algoritmo di ritrasmissione leggermente differente. TLS è il protocollo per la sicurezza dello strato di trasporto standardizzato da IETF, che esamineremo nel Capitolo 8. Molti dispositivi, almeno all'inizio, conterranno probabilmente entrambe le pile come mostrato nella Figura 7.55.

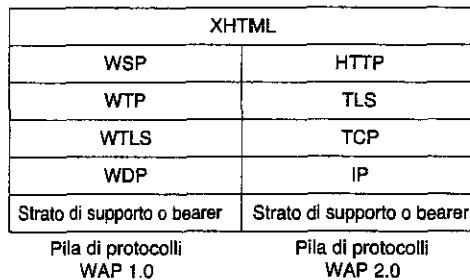


Figura 7.55. WAP 2.0 supporta due pile di protocolli.

L'altra differenza tecnica con WAP 1.0 è il linguaggio di markup. WAP 2.0 supporta XHTML Basic, destinato alle periferiche wireless miniaturizzate. Dal momento che anche NTT DoCoMo ha deciso di supportare questo sottoinsieme, chi progetta siti Web può utilizzare questo formato sapendo che le pagine saranno visibili sulla Internet fissa e su qualsiasi dispositivo wireless. Queste decisioni termineranno la guerra dei formati per il linguaggio di markup, che stava rallentando la crescita dell'industria Web wireless.

Lo strato applicazione

È il caso di spendere alcune parole su XHTML Basic, uno standard destinato a telefoni cellulari, televisioni, PDA, distributori automatici, cercapersone, automobili, videogiochi e persino orologi. Per questo motivo non supporta fogli di stile, script o frame ma accetta la maggior parte dei tag standard, che sono raggruppati in 11 moduli: alcuni obbligatori ed altri facoltativi. Tutti sono definiti in XML. I moduli e alcuni tag di esempio sono elencati nella Figura 7.56. Non spiegheremo tutti i tag di esempio; ulteriori informazioni sono disponibili all'indirizzo www.w3.org.

Modulo	Obbligatorio	Funzione	Tag di esempio
Structure	Sì	Struttura del documento	body, head, html, title
Text	Sì	Informazione	br, code, dfn, em, hn, kbd, p, strong
Hypertext	Sì	Collegamenti ipertestuali	a
List	Sì	Elenchi di voci	dl, dt, dd, ol, ul, li
Forms	No	Moduli da compilare	form, input, label, option, textarea
Tables	No	Tabelle rettangolari	caption, table, td, th, tr
Image	No	Immagini	img
Object	No	Applet, mappe e così via	object, param
Meta-information	No	Informazioni aggiuntive	meta
Link	No	Simile ad <a>	link
Base	No	Punto di partenza per gli URL	base

Figura 7.56. I moduli e i tag di XHTML Basic.

Nonostante l'accordo sull'utilizzo di XHTML Basic, su WAP e i-mode incombe una minaccia: 802.11. Il Web wireless di seconda generazione dovrebbe essere eseguito a 384 kbps, più velocemente dei 9.600 bps della prima generazione, ma molto più lentamente degli 11 Mbps o 54 Mbps offerti da 802.11. Naturalmente 802.11 non è ovunque, ma visto che molti ristoranti, hotel, negozi, società, aeroporti, stazioni degli autobus, musei, università, ospedali e altre organizzazioni (soprattutto negli Stati Uniti) stanno installando le stazioni base per dipendenti e clienti, presto nelle aree urbane vi sarà una copertura sufficiente a invogliare le persone a fare due passi per sedersi in un bar dotato di 802.11 per sorseggiare un caffè e leggere la posta elettronica. Negozi e locali pubblici possono inserire il logo di 802.11 accanto a quelli che mostrano le carte di credito accettate, proprio con il medesimo scopo: attrarre i clienti. Le cartine delle città (scaricabili, naturalmente) potrebbero mostrare le aree coperte in verde e quelle non coperte in rosso, in modo che le persone si possano spostare di stazione base in stazione base, proprio come i nomadi camminano da oasi a oasi nel deserto.

Anche se bar e fast-food potrebbero installare rapidamente le stazioni base 802.11, è difficile che gli agricoltori facciano questa scelta: la copertura sarà quindi irregolare e limitata alle aree centrali delle città, a causa dell'estensione limitata di 802.11 (poche centinaia di metri al massimo). Questo potrebbe portare a dispositivi wireless dual-mode, che utilizzano 802.11 se riescono a captare un segnale e ricorrono a WAP in mancanza di copertura.

7.4 Multimedia

Il Web wireless è un nuovo sviluppo interessante, ma non è l'unico. Per molte persone, il multimedia è il sacro graal delle reti. Quando viene citata la parola, la salvezza di cervelloni e uomini d'affari aumenta vertiginosamente. I primi vedono immense sfide tecniche nel fornire video interattivo su richiesta ad ogni abitazione. I secondi vedono ugualmente immensi vantaggi. Visto che il multimedia richiede una banda elevata, farlo funzionare su connessioni fisse è già piuttosto difficile; la trasmissione wireless di un video anche di qualità VHS potrà essere eseguita solo tra qualche anno, perciò la nostra descrizione riguarderà i sistemi cablati.

Letteralmente, multimedia riguarda l'utilizzo di due o più supporti (*media* in inglese). Se l'autore di questo libro volesse aderire al tanto reclamizzato "multimedia", potrebbe affermare che utilizza una tecnologia multimediale. Dopo tutto, contiene già due supporti: il testo e la grafica (le figure). Tuttavia, quando la maggior parte delle persone parla di multimediale, in genere intende la combinazione di due o più **supporti continui**, vale a dire media che devono essere riprodotti in un intervallo di tempo ben definito, solitamente con una certa interazione da parte dell'utente. In pratica, i due supporti sono normalmente audio e video, vale a dire un suono accompagnato da immagini in movimento. Tuttavia, spesso molte persone utilizzano il termine per fare riferimento all'audio puro, come la telefonia o le radio Internet: naturalmente questo utilizzo non è corretto. In effetti, un termine migliore è **streaming media**; tuttavia, seguiremo l'abitudine acquisita e considereremo l'audio in tempo reale come "multimedia". Nei paragrafi seguenti esamineremo il modo in cui i computer elaborano audio e video, le modalità di compressione e alcune applicazioni di rete per queste tecnologie. Per una descrizione completa (in tre volumi) sul multimediale in rete, vedere (Steinmetz e Nahrstedt, 2002; Steinmetz e Nahrstedt, 2003a; Steinmetz e Nahrstedt, 2003b).

7.4.1 Introduzione all'audio digitale

Un'onda audio (sonora) è un'onda acustica (di pressione) monodimensionale. Quando un'onda acustica entra nell'orecchio, il tamburo vibra provocando la successiva vibrazione delle piccole ossa interne all'orecchio, che inviano impulsi nervosi al cervello percepiti come suoni dall'ascoltatore. In modo simile, quando un'onda acustica colpisce un microfono, il microfono genera un segnale elettrico, che rappresenta l'ampiezza del suono come funzione del tempo. La rappresentazione, l'elaborazione, la memorizzazione e la trasmissione di questi segnali audio sono i principali argomenti di studio per i sistemi multimediali.

L'intervallo di frequenze percepite dall'orecchio umano è compreso tra 20 e 20.000 Hz. Alcuni animali, in particolare i cani, possono udire frequenze superiori. L'orecchio "ascolta" in modo logaritmico, pertanto il rapporto tra due suoni con potenze A e B è espresso convenzionalmente in **dB (decibel)** secondo la formula:

$$\text{dB} = 10 \log_{10}(A/B)$$

Se definiamo il limite inferiore di udibilità (una pressione di circa 0,0003 dyne/cm²) per un'onda sinusoidale di 1 kHz come 0 dB, una conversazione ordinaria avviene a circa 50 dB, mentre la soglia del dolore è a circa 120 dB: l'intervallo dinamico presenta un fattore pari a un milione.

L'orecchio è sorprendentemente sensibile alle variazioni del suono in pochi millisecondi. L'occhio, al contrario, non nota le modifiche leggere del livello luminoso che avvengono in pochi millisecondi. Il risultato di questa osservazione è che un jitter di pochi millisecondi durante una trasmissione multimediale influenza sulla qualità percepita del suono più di quanto influenzino la qualità percepita dell'immagine.

Le onde audio possono essere convertite in forma digitale da un **convertitore analogico/digitale** (ADC, *Analog-Digital Converter*). Un ADC riceve in input una tensione elettrica e genera in output un numero binario. Nella Figura 7.57(a) è mostrato un esempio di onda sinusoidale. Per rappresentare questo segnale in forma digitale, è possibile eseguire un campionamento ogni DT secondi, come mostrato dalle barre nella Figura 7.57(b). Se un'onda audio non è un'onda sinusoidale pura ma una sovrapposizione lineare di onde sinusoidali dove la componente di frequenza più alta è f , allora il teorema di Nyquist (vedere il Capitolo 2) afferma che è sufficiente per creare campioni alla frequenza $2f$. Un campionamento più rapido non ha alcun valore, perché non sono presenti le frequenze più alte che il campionamento potrebbe rilevare.

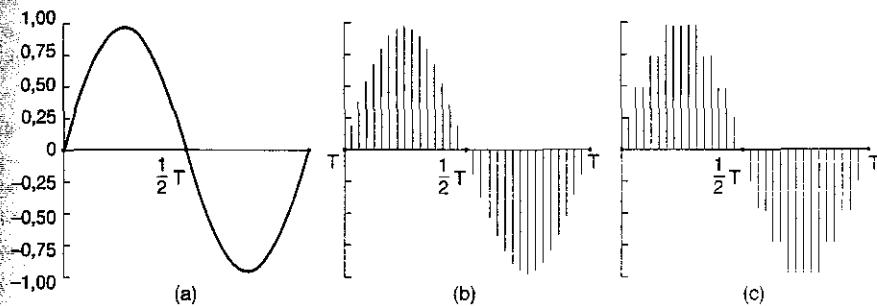


Figura 7.57. (a) Un'onda sinusoidale. (b) Il campionamento dell'onda sinusoidale. (c) La quantizzazione dei campioni a 4 bit.

I campioni digitali non sono mai esatti. I campioni della Figura 7.57(c) consentono solo nove valori, da -1,00 a +1,00 con passi di 0,25. Un campione di 8 bit consentirebbe 256 valori distinti. Un campione di 16 bit consentirebbe 65.536 valori distinti. L'errore introdotto dal numero finito di bit per campione è chiamato **rumore di quantizzazione**. Se è troppo grande, l'orecchio è in grado di rilevarlo. Due esempi ben noti di utilizzo dei suoni campionati sono il telefono e i compact disc audio. La modulazione a impulsi, utilizzata dal sistema telefonico, utilizza campioni di 8 bit presi 8.000 volte al secondo. In Nord America e in Giappone, 7 bit sono per i dati e 1 per il controllo; in Europa tutti gli 8 bit sono per i dati. Questo sistema offre una velocità dei dati di 56.000 oppure 64.000 bps. Con solo 8.000 campioni al secondo, le frequenze superiori a 4 kHz vengono perse.

I CD audio sono digitali, con una frequenza di campionamento pari a 44.100 campioni al secondo, sufficienti per acquisire le frequenze fino a 22.050 Hz: sono sufficienti per le persone, ma non per i cani amanti della musica. I campioni sono di 16 bit ognuno e sono lineari nella gamma di ampiezze. Occorre notare che i campioni di 16 bit permettono solo 65.536 valori distinti, anche se la gamma dinamica dell'orecchio è pari a circa un milione se misurata in passi con ampiezza uguale al più piccolo suono udibile. Di conseguenza, l'utilizzo di soli 16 bit per campione introduce un certo rumore di quantizzazione (anche se l'intera gamma dinamica non è coperta: i CD non sono pensati per arrivare alla soglia del dolore). Con 44.100 campioni al secondo di 16 bit ognuno, un CD audio richiede una banda di 705,6 kbps per l'audio mono e 1,411 Mbps per l'audio in stereo. Anche se è un valore inferiore a quello necessario per il video, richiede comunque un canale T1 completo per trasmettere in tempo reale suoni stereo di qualità CD non compressi.

Il suono digitalizzato può essere facilmente elaborato via software dai computer. Esistono decine di programmi per personal computer che permettono agli utenti di registrare, visualizzare, modificare, mixare e memorizzare onde audio da più fonti. In pratica oggi tutti gli strumenti professionali di registrazione ed editing del suono sono digitali.

La musica, naturalmente, è solo un tipo speciale di audio, anche se molto importante. Un altro tipo importante è il parlato. Il parlato umano è compreso quasi interamente nell'intervallo di frequenze tra 600 e 6.000 Hz. Un discorso è composto di vocali e consonanti, che hanno proprietà diverse. Le vocali sono prodotte quando il tratto vocale non è ostruito, generando risonanze le cui frequenze fondamentali dipendono dalla forma e della dimensione del sistema vocale, nonché dalla posizione della lingua e della mascella di chi sta parlando. Questi suoni sono quasi periodici per intervalli di circa 30 msec. Le consonanti sono prodotte quando il tratto vocale è parzialmente bloccato. Questi suoni sono meno regolari delle vocali.

Alcuni sistemi di trasmissione e generazione del parlato fanno uso di modelli del sistema vocale per ridurre un discorso a pochi parametri (per esempio le dimensioni e le forme di varie cavità), anziché campionare semplicemente la forma d'onda del suono. Il funzionamento di questi vocoder va però oltre l'ambito di questo libro.

7.4.2 La compressione audio

Abbiamo già visto che l'audio di qualità CD richiede una banda di trasmissione di 1,411 Mbps. Chiaramente, è necessaria una forte compressione per rendere fattibile la trasmissione su Internet; per questo motivo, sono stati sviluppati molti algoritmi di compressione audio. Probabilmente il più popolare è l'audio MPEG, che presenta tre livelli (o varianti) di cui MP3 (*MPEG audio layer 3*) è il più potente e conosciuto. Su Internet è disponibile moltissima musica in formato MP3: non è tutta legale, e ciò ha stimolato numerose cause legali da parte degli artisti e dei proprietari del copyright. MP3 appartiene alla porzione audio dello standard di compressione video MPEG. Discuteremo la compressione video in seguito nel capitolo; vediamo per ora la compressione audio.

La compressione audio può essere eseguita in due modi. Nella **codifica per forma d'onda** (*waveform coding*), il segnale viene convertito matematicamente da una trasformata di

Fourier nelle sue componenti nel dominio della frequenza. L'esempio della Figura 2.1(a) mostra una funzione nel dominio del tempo e le sue ampiezze di Fourier. L'ampiezza di ogni componente viene poi codificata con la minima quantità di bit; lo scopo è riprodurre accuratamente all'altro capo la forma d'onda con il minor numero di bit possibile.

L'altro metodo, la **codifica percettiva** (*perceptual coding*), sfrutta alcuni limiti del sistema uditivo umano per codificare un segnale in modo tale che sembri lo stesso a un ascoltatore umano, pur apparendo piuttosto diverso su un oscilloscopio. La codifica percettiva è basata sulla scienza della **psicoacustica**, che studia il modo in cui le persone percepiscono i suoni. MP3 è basato sulla codifica percettiva.

La proprietà fondamentale della codifica percettiva riguarda il fatto che alcuni suoni possono **mascherare** altri suoni. Si supponga di voler trasmettere un concerto per flauto dal vivo, in un caldo giorno d'estate. Immediatamente una folla di operai attiva i martelli pneumatici e inizia a rompere la strada. Nessuno potrebbe più udire i flauti: il loro suono è stato mascherato dai martelli pneumatici. Ai fini della trasmissione, ora è sufficiente codificare solo la banda di frequenza utilizzata dai martelli pneumatici, perché gli ascoltatori non possono comunque udire i flauti. La tecnica è chiamata **mascheramento della frequenza** e indica la capacità di un suono forte in una banda di nascondere un suono più tenue in un'altra banda di frequenza, che sarebbe stato udibile in assenza del suono più forte. In effetti, anche dopo che i martelli pneumatici terminano il loro lavoro, i flauti non saranno udibili per un breve periodo di tempo, perché all'avviarsi dei martelli pneumatici l'orecchio deve abbassare il suo guadagno e richiede un tempo finito per riportarlo a valori normali. Questo effetto è definito **mascheramento temporale**.

Per rendere gli effetti più quantitativi, eseguiamo un primo esperimento. Una persona in una stanza silenziosa indossa le cuffie connesse alla scheda audio del computer. Il computer genera un'onda sinusoidale pura a 100 Hz, inizialmente tenue ma che aumenta gradualmente la potenza. La persona deve premere un tasto quando riesce a udire il tono. Il computer registra il livello di potenza corrente e poi ripete l'esperimento a 200 Hz, 300 Hz e a tutte le altre frequenze, fino al limite dell'udito umano. Dopo avere ripetuto l'esperimento su diverse persone per calcolare una media, il grafico su scala bilogaritmica della potenza necessaria per rendere udibile un tono assume l'aspetto della Figura 7.58(a). Una conseguenza diretta di questa curva è che non è mai necessario codificare le frequenze la cui potenza non ricade all'interno della soglia di udibilità. Per esempio, se la potenza a 100 Hz fosse 20 dB nella Figura 7.58(a), potrebbe essere omessa dall'output con una perdita di qualità non percettibile, perché 20 dB a 100 Hz è un valore che si trova sotto alla soglia di udibilità.

Vediamo ora un secondo esperimento. Il computer ripete l'esperimento 1, questa volta con un'onda sinusoidale ad ampiezza costante a 150 Hz, sovrapposta alla frequenza di test. È possibile scoprire che la soglia di udibilità per le frequenze vicine a 150 Hz è sollevata, come mostrato nella Figura 7.58(b).

La conseguenza di questa nuova osservazione è che, tenendo traccia di quali segnali vengono mascherati dai segnali più potenti in bande di frequenza vicine, è possibile risparmiare bit omettendo frequenze nel segnale codificato. Nella Figura 7.58, il segnale di 125

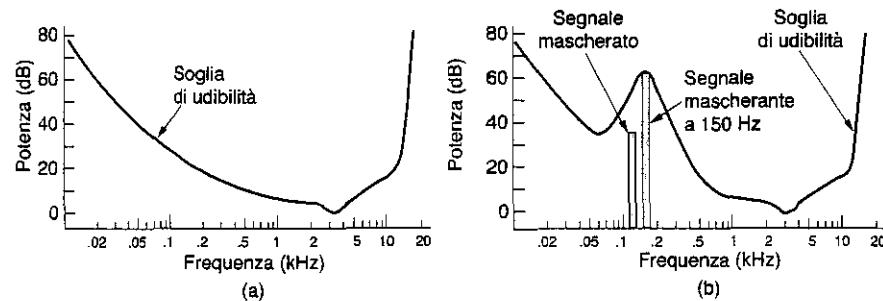


Figura 7.58. (a) La soglia di udibilità in funzione della frequenza. (b) L'effetto del mascheramento.

Hz può essere completamente omesso dall'output e nessuno sarà in grado di notare la differenza. Anche dopo che in qualche banda di frequenza viene a cessare un segnale potente, la conoscenza delle proprietà di mascheramento temporali consente di continuare a omettere le frequenze mascherate per il tempo di recupero dell'orecchio. L'idea alla base di MP3 sta nell'eseguire una trasformazione di Fourier del suono per ottenere la potenza a ogni frequenza, e quindi trasmettere solo le frequenze non mascherate, codificandole con il minor numero di bit possibile.

Con queste informazioni di base possiamo vedere come viene eseguita la codifica. La compressione audio viene svolta campionando la forma d'onda a 32 kHz, 44,1 kHz o 48 kHz. Il campionamento può essere eseguito su uno o due canali, scegliendo fra quattro possibili configurazioni:

1. monofonico (un singolo flusso di input)
2. monofonico doppio (per esempio una colonna sonora in inglese e giapponese)
3. stereo disgiunto (ogni canale è compresso separatamente)
4. stereo unito (la ridondanza intercanale è sfruttata completamente).

In primo luogo si deve scegliere il bit-rate di uscita. MP3 può comprimere un CD di rock 'n roll stereo a 96 kbps con una piccola perdita percettibile di qualità, anche per i fan del rock 'n roll con un udito perfetto, ma per un concerto per pianoforte sono necessari almeno 128 kbps. Questa differenza esiste perché il rapporto segnale/rumore per il rock 'n roll è più elevato che per un concerto per pianoforte (almeno in senso ingegneristico). È anche possibile scegliere velocità di output inferiori e accettare una certa perdita di qualità. I campioni vengono poi elaborati in gruppi di 1.152 (che corrispondono a circa 26 msec). Ogni gruppo viene preventivamente passato attraverso 32 filtri digitali per ottenere 32 bande di frequenza. Allo stesso tempo, l'input viene inserito in un modello psicoacustico per determinare le frequenze mascherate. A questo punto, ognuna delle 32 bande di frequenza è ulteriormente trasformata per fornire una risoluzione spettrale più fine.

Nella fase successiva, il budget dei bit viene ripartito tra le bande, allocando il massimo dei bit alle bande con la maggiore potenza spettrale non mascherata, meno bit alle bande non mascherate ma con meno potenza spettrale, e nessun bit alle bande mascherate. Per finire, i bit vengono codificati con la codifica di Huffman, che assegna codici brevi ai numeri che appaiono spesso e codici lunghi a quelli che appaiono meno spesso.

Ci sarebbe altro da aggiungere alla storia. Per la riduzione del disturbo, l'antialiasing e lo sfruttamento della ridondanza intercanale si utilizzano varie tecniche, che però vanno oltre l'ambito di questo libro. Un'introduzione matematica più formale al processo è data in (Pan, 1995).

7.4.3 L'audio in streaming

Passiamo ora dalla tecnologia dell'audio digitale a tre delle sue applicazioni di rete. La prima è l'audio in streaming, che permette l'ascolto dei suoni su Internet. È chiamato anche "musica su richiesta". Le due successive sono rispettivamente le radio Internet e Voice over IP.

Internet è ricca di siti Web musicali, molti dei quali elencano titoli di brani su cui gli utenti possono fare clic per riprodurre le canzoni. Alcuni siti sono gratuiti (per esempio quelli di nuovi gruppi in cerca di pubblicità), ed altri esigono un pagamento per ascoltare la musica, anche se spesso offrono campioni gratuiti (relativi per esempio ai primi 15 secondi di un brano). Il modo più diretto per riprodurre la musica è illustrato nella Figura 7.59.

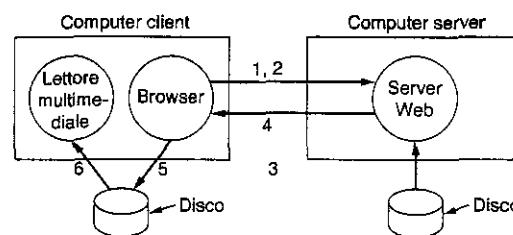


Figura 7.59. Un modo elementare per richiamare musica facendo clic su una pagina Web.

Il processo inizia quando l'utente fa clic su un brano, e il browser passa all'azione. Il passaggio 1 permette di stabilire una connessione TCP al server Web su cui si trova il brano collegato. Il passaggio 2 invia una richiesta *GET* in HTTP per richiedere il brano. Successivamente (passaggi 3 e 4), il server preleva il brano (che è un semplice file in MP3 o altri formati) dal disco e lo restituisce al browser. Se il file è più grande della memoria del server, quest'ultimo potrebbe prelevare e inviare la musica un blocco alla volta. Utilizzando il tipo MIME (per esempio *audio/mp3*) o l'estensione del file, il browser cerca di capire come deve visualizzare il file. Di norma esiste un'applicazione helper, come RealOne Player, Windows Media Player o Winamp, associata a questo tipo di file. Visto

che normalmente un browser comunica con un helper scrivendo il contenuto in un file temporaneo, l'intero file musicale è preventivamente salvato come file temporaneo su disco (passaggio 5). A questo punto viene avviato il lettore multimediale, passandogli il nome del file temporaneo. Nel passaggio 6, il lettore multimediale inizia a prelevare e riprodurre la musica, un blocco alla volta.

In principio questo approccio è assolutamente corretto e sufficiente per riprodurre la musica; l'unico problema è che l'intero brano deve essere trasmesso sulla rete prima dell'inizio della riproduzione. Se il brano è di 4 MB (dimensione tipica per un brano MP3) e il modem è a 56 kbps, l'utente sarà accolto da almeno 10 minuti di silenzio durante il download della canzone. Non tutti gli amanti della musica approvano questo comportamento, specialmente se anche il brano successivo richiede altri 10 minuti di tempo di download. Per risolvere il problema senza modificare il funzionamento del browser, i siti musicali hanno provato a utilizzare il seguente schema. Il file collegato al titolo del brano non è il vero e proprio file musicale. Si tratta invece di un **metafile**, vale a dire un file molto breve il cui scopo è solo quello di assegnare un nome alla musica. Un tipico metafile potrebbe contenere una sola riga di testo ASCII simile alla seguente:

`rtsp://joes-audio-server/song-0025.mp3`

Quando il browser riceve il file di una riga, lo scrive su disco in un file temporaneo, avvia il lettore multimediale come helper gli passa come al solito il nome del file. Il lettore multimediale legge il file e vede che contiene un URL; contatta quindi *joes-audio-server* e richiede il brano. Occorre notare che ora il browser non fa più parte del ciclo.

Nella maggior parte dei casi, il server nominato nel metafile non è il server Web. In effetti, di norma non si tratta di un server HTTP, ma di un server multimediale specializzato. In questo esempio, il server multimediale utilizza **RTSP** (*Real Time Streaming Protocol*), come indicato dallo schema di denominazione *rtsp*. È descritto in RFC 2326.

Il lettore multimediale deve svolgere quattro operazioni fondamentali:

1. gestire l'interfaccia utente
2. gestire gli errori di trasmissione
3. decomprimere la musica
4. eliminare il jitter.

La maggior parte dei lettori multimediali odierni presenta un'interfaccia utente accattivante, che ricorda uno stereo con pulsanti, leve, cursori e display visivi. Spesso sono disponibili pannelli frontali intercambiabili, chiamati **skin**, che l'utente può applicare al lettore. Il lettore multimediale deve gestire tutto questo e interagire con l'utente.

Il suo secondo compito è gestire gli errori. La trasmissione di musica in tempo reale utilizza raramente TCP, perché un errore e una ritrasmissione potrebbero introdurre una lacuna inaccettabilmente lunga nella musica. Al contrario, la trasmissione è di solito svolta con un protocollo come RTP, che abbiamo studiato nel capitolo 6. Come la maggior parte dei

protocolli in tempo reale, RTP è basato su UDP, pertanto si potrebbe verificare la perdita di pacchetti. È compito del lettore gestire la situazione.

In alcuni casi, la musica è intercalata (*interleaved*) per facilitare la gestione degli errori. Per esempio, un pacchetto potrebbe contenere 220 campioni stereo, ognuno contenente una coppia di numeri di 16 bit, di norma validi per 5 msec di musica. Tuttavia, il protocollo potrebbe inviare tutti i campioni dispari per un intervallo di 10 msec in un pacchetto, e tutti i campioni pari nel pacchetto successivo. Un pacchetto perso non rappresenta quindi un vuoto di 5 msec nella musica, ma la perdita dei campioni per 10 msec. Questa perdita può essere gestita facilmente chiedendo al lettore multimediale di eseguire un'interpolazione (stima dei valori mancanti) utilizzando i campioni precedenti e successivi. L'utilizzo di questi metodi per ottenere il recupero degli errori è mostrato nella Figura 7.60. Ogni pacchetto contiene i soli campioni pari oppure dispari per un intervallo di 10 msec. Di conseguenza, la perdita del pacchetto 3 (come mostrato) non provoca un vuoto nella musica, ma riduce la risoluzione temporale per un certo intervallo di tempo; i valori mancanti possono essere interpolati per fornire musica continua. Questo particolare schema funziona solo con campioni non compressi, ma mostra che una codifica intelligente può trasformare un pacchetto perso in una riduzione di qualità anziché una pausa. Ad ogni modo, RFC 3119 offre uno schema che funziona con l'audio compresso.

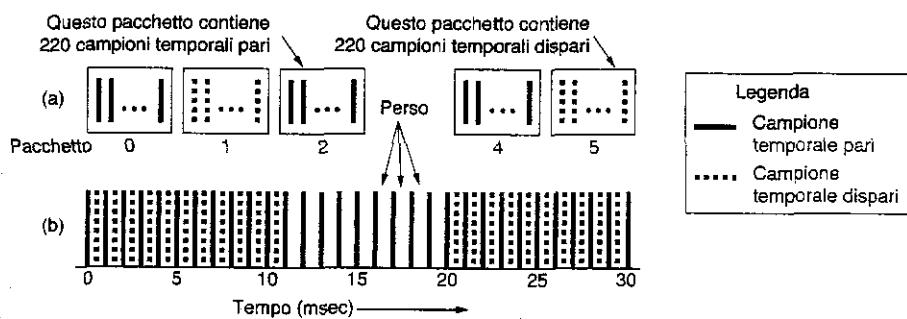


Figura 7.60. Quando i pacchetti trasportano campioni alternati, la perdita di un pacchetto riduce la risoluzione temporale evitando la comparsa di una lacuna.

Il terzo compito del lettore multimediale consiste nel decomprimere la musica. Anche se questa attività sfrutta in modo intensivo le risorse, è piuttosto semplice.

Il quarto compito consiste nell'eliminare lo jitter, che è la rovina tutti i sistemi in tempo reale. Tutti i sistemi audio in streaming iniziano eseguendo il buffering di circa 10-15 secondi di musica prima di iniziare la riproduzione, come mostrato nella Figura 7.61. Teoricamente il server continuerà a riempire il buffer alla esatta frequenza di "richiamo" del lettore multimediale; in realtà questo non accade, pertanto è utile una retroazione nel ciclo. Per mantenere sempre pieno il buffer vengono utilizzati due approcci. Con un **server pull**,

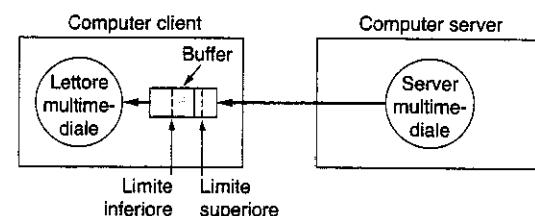


Figura 7.61. Il lettore multimediale inserisce nel buffer l'input del server multimediale, ed esegue la riproduzione dal buffer anziché direttamente dalla rete.

finché nel buffer esiste spazio per un altro blocco, il lettore multimediale continua a inviare richieste al server per ottenere un blocco aggiuntivo; il suo obiettivo è mantenere il buffer pieno. Lo svantaggio di un server pull sta nelle richieste di dati non necessarie. Il server sa di avere spedito l'intero file, quindi perché il lettore continua a chiederlo? Per questo motivo è utilizzato raramente.

Con un **server push**, il lettore multimediale invia una richiesta *PLAY* e il server continua semplicemente a inviare dati. Esistono due possibilità: il server multimediale lavora a velocità pari alla normale velocità di riproduzione, oppure spedisce i dati più rapidamente. In entrambi i casi, alcuni dati vengono inseriti nel buffer prima dell'inizio della riproduzione. Se il server lavora alla normale velocità di riproduzione, i dati in arrivo vengono accodati alla fine del buffer e il lettore rimuove i dati dalla parte iniziale del buffer per la riproduzione. Finché tutto funziona perfettamente, la quantità di dati nel buffer rimane costante nel tempo. Questo schema è semplice perché non sono richiesti messaggi di controllo nelle due direzioni.

L'altro schema "push" prevede che il server carichi i dati più velocemente del necessario. Il vantaggio sta nel fatto che, se il server non può garantire l'esecuzione a una velocità regolare, ha l'opportunità di recuperare nel caso resti indietro. Un problema deriva invece dai potenziali overrun del buffer, che potrebbero verificarsi se il server invia i dati troppo velocemente rispetto al consumo (e deve essere in grado di farlo, per evitare la formazione di lacune).

La soluzione consiste nel chiedere al lettore multimediale di definire un **limite inferiore** e un **limite superiore** nel buffer. Fondamentalmente, il server invia i dati fino a quando il buffer arriva a riempirsi al livello del limite superiore. Il lettore multimediale chiede quindi di fare una pausa. Visto che i dati continuano ad affluire fino a quando il server ha ricevuto la richiesta di pausa, la distanza tra il limite superiore e la fine del buffer deve essere superiore al prodotto banda-ritardo della rete. Una volta fermato il server, il buffer inizia a svuotarsi. Una volta raggiunto il limite inferiore, il lettore multimediale comunica al server di riprendere. Il limite inferiore deve essere scelto in modo che non si verifichino underrun del buffer.

Per operare con un server push, il lettore multimediale ha bisogno di un meccanismo per il suo controllo remoto: questo è proprio lo scopo di RTSP. È definito in RFC 2326 e fornisce il meccanismo che permette al lettore di controllare il server. Non specifica però la gestione del flusso dei dati, lasciata di norma a RTP. I comandi principali supportati da RTSP sono elencati nella Figura 7.62.

Comando	Azione del server
DESCRIBE	Elenca i parametri del file multimediale
SETUP	Stabilisce un canale logico tra il lettore e il server
PLAY	Inizia a inviare dati al client
RECORD	Inizia ad accettare dati dal client
PAUSE	Interrompe temporaneamente l'invio dei dati
TEARDOWN	Rilascia il canale logico

Figura 7.62. I comandi RTSP dal lettore al server.

7.4.4 Le radio Internet

Dopo aver reso possibile lo streaming dell'audio su Internet, le stazioni radio commerciali hanno avuto l'idea di trasmettere il loro contenuto su Internet, oltre che nell'etere. Poco tempo dopo, anche le stazioni radio dei college hanno iniziato a porre il loro segnale su Internet. Questo significa che gli *studenti* dei college hanno creato le loro stazioni radio: con la tecnologia attuale, chiunque può teoricamente creare una stazione radio. L'intera area delle radio Internet è nuova e in fase sperimentale, ma vale la pena parlarne.

Esistono due approcci generali alle radio Internet. Nel primo, i programmi sono preregistrati e archiviati su disco. Gli ascoltatori possono connettersi agli archivi della stazione radio, scegliere un programma e scaricarlo per l'ascolto. In effetti, questa tecnica è identica allo streaming audio appena discusso. È anche possibile memorizzare ogni programma subito dopo la trasmissione del vivo, in modo che l'archivio sia in esecuzione solo per una mezz'ora, per esempio. I vantaggi di questo approccio sono la facilità di gestione (tutte le tecniche discusse finora funzionano anche in questo caso) e il fatto che gli ascoltatori possono scegliere tra i diversi programmi nell'archivio. L'altro approccio consiste nel trasmettere dal vivo su Internet. Alcune stazioni trasmettono sull'etere e su Internet contemporaneamente, ma un numero sempre maggiore di stazioni radio esiste solo su Internet. Alcune delle tecniche applicabili all'audio in streaming sono utili anche per le radio Internet dal vivo, ma vi sono alcune differenze fondamentali.

Un punto che non cambia riguarda la necessità di archiviare i dati nel buffer dell'utente per attenuare lo jitter. Archiviando 10 o 15 secondi di radio prima di iniziare la riproduzione, l'audio può proseguire ininterrotto anche in caso di gravi jitter sulla rete. Finché i pacchetti arrivano prima dell'istante in cui sono necessari, il preciso momento di arrivo non conta.

Una differenza fondamentale riguarda il fatto che l'audio in streaming può essere inviato a una velocità superiore a quella di riproduzione, perché il ricevitore può fermarsi quando raggiunge il limite superiore. Potenzialmente, in questo modo si ha il tempo di ritrasmettere i pacchetti persi, anche se la strategia non è utilizzata comunemente. Al contrario, la radio dal vivo è sempre trasmessa alla velocità di generazione e riproduzione.

Un'altra differenza riguarda il fatto che una stazione radio dal vivo possiede di norma centinaia o migliaia di ascoltatori simultanei, mentre l'audio in streaming è di tipo punto a punto. In queste circostanze, la radio Internet dovrebbe utilizzare il multicasting con i protocolli RTP/RTSP. È chiaramente il metodo operativo più efficiente.

Attualmente, le radio Internet non funzionano in questo modo: l'utente stabilisce una connessione TCP alla stazione e i dati vengono inviati sulla connessione TCP. Naturalmente questo crea vari problemi, come l'interruzione del flusso quando la finestra è piena, il timeout e la ritrasmissione dei pacchetti persi, e così via.

Esistono tre motivi per cui viene utilizzato l'unicasting TCP al posto del multicasting RTP. Innanzitutto, pochi ISP supportano il multicasting, che pertanto non è un'opzione pratica. In secondo luogo, RTP è meno conosciuto di TCP: spesso le stazioni radio sono piccole e hanno poca esperienza informatica, pertanto è più facile utilizzare un protocollo ben compreso e supportato da tutti i pacchetti software. Infine, molte persone ascoltano le radio Internet al lavoro: in pratica, questo spesso significa che lavorano attraverso un firewall. La maggior parte degli amministratori di sistema configura il firewall per proteggere la LAN dai visitatori sgraditi. Solitamente sono ammesse le connessioni TCP dalla porta remota 25 (SMTP per la posta elettronica), i pacchetti UDP dalla porta remota 53 (DNS) e le connessioni TCP dalla porta remota 80 (HTTP per il Web). Tutto il resto viene bloccato, compreso RTP. Di conseguenza, l'unico modo per far passare il segnale radio attraverso il firewall richiede che il sito Web finga di essere un server HTTP (almeno per il firewall) e di utilizzare i server HTTP, che comunicano con TCP. Queste misure di protezione molto severe, pur offrendo solo una protezione minima, obbligano le applicazioni multimediali a seguire modalità operative molto meno efficienti.

Visto che la radio Internet è un nuovo supporto, la guerra dei formati è in pieno svolgimento. RealAudio, Windows Media Audio e MP3 sono in competizione su questo mercato per divenire il formato dominante per le radio Internet. Un nuovo arrivo è Vorbis, tecnicamente simile a MP3 ma di tipo "open source" e abbastanza diverso da non utilizzare il brevetto su cui è basato MP3.

Una tipica stazione radio su Internet possiede una pagina Web che ne elenca la pianificazione, informazioni su DJ e speaker, e molti annunci pubblicitari. Esistono inoltre una o più icone che elencano i formati audio supportati (o semplicemente un'icona ASCOLTA se viene supportato un solo formato). Queste icone sono metafile del tipo discusso in precedenza.

Quando un utente fa clic su una delle icone, viene inviato il breve metafile. Il browser utilizza il suo tipo MIME o l'estensione del file per determinare l'helper appropriato per il metafile, e cioè il lettore multimediale. Scrive quindi il metafile su un file temporaneo nel disco fisso, avvia il lettore multimediale e passa il nome del file. Il lettore multimediale legge il file temporaneo, vede l'URL contenuto (solitamente con lo schema *http* anziché *rtsp*, per aggirare il firewall e per il fatto che molte popolari applicazioni multimediali lavorano in questo modo), contatta il server e inizia ad agire come una radio.

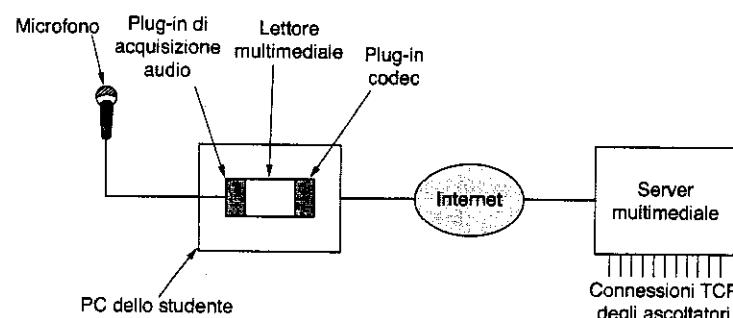


Figura 7.63. Una stazione radio per studenti.

Tra parentesi, l'audio presenta un solo flusso, quindi *http* funziona bene; per il video, che utilizza almeno due flussi, *http* fallisce ed è realmente necessario qualcosa di simile a *rtsp*.

Un altro interessante sviluppo nell'area delle radio Internet è una soluzione dove chiunque, persino uno studente, può attivare una stazione radio. I componenti principali sono illustrati nella Figura 7.63. La base della stazione è un PC ordinario con una scheda audio e un microfono. Il software consiste di un lettore multimediale, come Winamp o Freeamp, con un plug-in per l'acquisizione audio e un codec per il formato di output selezionato, per esempio MP3 o Vorbis.

Il flusso audio generato dalla stazione viene poi inviato su Internet a un grande server, che gestisce la distribuzione a un elevato numero di connessioni TCP. Il server supporta generalmente molte piccole stazioni. Gestisce anche una guida delle stazioni che gestisce, con indicazione di ciò che è attualmente in onda su ognuna. I potenziali ascoltatori visitano il server, scelgono una stazione, e ottengono i dati tramite TCP. Esistono pacchetti software commerciali per la gestione dell'intera soluzione, nonché pacchetti open source come icecast. Esistono inoltre server che gestiscono a pagamento la distribuzione.

7.4.5 Voice over IP

Una volta, il sistema telefonico commutato pubblico era utilizzato principalmente per il traffico vocale, con qualche trasmissione di dati qua e là. Il traffico dati cresceva sempre e, nel 1999, il numero di bit di dati trasmessi equivaleva al numero di bit vocali. Nel 2002, il volume del traffico dati era notevolmente superiore al volume di traffico vocale e continuava a crescere esponenzialmente, al contrario del traffico vocale che presentava una crescita quasi piatta (5% l'anno).

Come conseguenza di questi numeri, molti operatori delle reti a commutazione di pacchetto iniziarono a interessarsi al trasporto della voce sulle reti di dati. La quantità di banda aggiuntiva richiesta per la voce è minima, perché le reti a pacchetti sono dimensionate per

il traffico dati. Tuttavia, la bolletta telefonica di un utente medio è probabilmente superiore alla bolletta relativa a Internet, pertanto gli operatori delle reti di dati hanno visto la telefonia Internet come un modo per fare altri soldi senza dover stendere nuovi cavi. Nacque così la **telefonia Internet** (detta anche *voice over IP*).

H.323

Una cosa che fu chiara a tutti fin dall'inizio fu il fatto che, se ogni produttore avesse progettato la propria pila di protocolli, il sistema non avrebbe mai funzionato. Per evitare questo problema, alcune delle parti interessate si unirono sotto gli auspici di ITU per elaborare gli standard. Nel 1996 ITU rilasciò la raccomandazione **H.323** intitolata "Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service" (sistema audiovisivo telefonico ed apparati per reti locali che offrono una qualità di servizio senza garanzie). Solo l'industria telefonica avrebbe potuto pensare a un nome del genere. Questa raccomandazione fu rivista nel 1998; H.323 divenne così la base per i primi sistemi telefonici Internet diffusi.

H.323 non descrive un vero e proprio protocollo, ma rappresenta più che altro una panoramica architettonica della telefonia Internet. Fa riferimento a diversi protocolli specializzati per la codifica del parlato, l'impostazione delle chiamate, la segnalazione, il trasporto di dati e altre aree, anziché emettere specifiche proprie. Il modello generale è mostrato nella Figura 7.64. Al centro si trova il **gateway** che connette Internet al sistema telefonico. Utilizza i protocolli H.323 sul lato Internet e i protocolli PSTN sul lato della telefonia. I dispositivi di comunicazione sono chiamati **terminali**. Una LAN può disporre di un **gatekeeper**, che controlla i punti finali sotto la sua giurisdizione, chiamata **zona**.

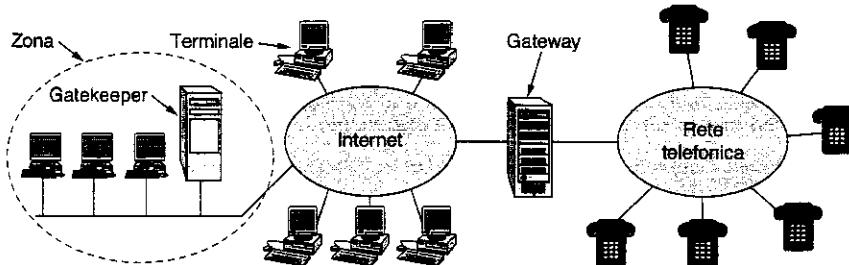


Figura 7.64. Il modello architettonico H.323 per la telefonia Internet.

Una rete telefonica richiede diversi protocolli. Per iniziare, esiste un protocollo per la codifica e la decodifica del parlato. Il sistema PCM studiato nel Capitolo 2 è definito nella raccomandazione ITU **G.711**. Codifica un singolo canale vocale eseguendo il campionamento 8000 volte al secondo con un campione di 8 bit, per fornire parlato non compresso a 64 kbps. Tutti i sistemi H.323 devono supportare G.711, ma sono consentiti (non obbligatori) anche altri protocolli di compressione vocale che utilizzano diversi algoritmi, presentando compromessi differenti tra la qualità e la banda. Per esempio, **G.723.1** prende un

blocco di 240 campioni (30 msec di discorso) e utilizza la codifica predittiva per ridurlo a 24 o 20 byte. Questo algoritmo offre una velocità di output pari a 6,4 o 5,3 kbps (fattore di compressione rispettivamente 10 e 12), con una piccola perdita nella qualità percepita. Sono consentiti anche altri codec.

Visto che sono permessi più algoritmi di compressione, è necessario un protocollo per dare ai terminali la possibilità di negoziarne l'utilizzo. Questo protocollo è chiamato **H.245**, e negozia anche altri aspetti della connessione, come il bit-rate. RTCP è necessario per il controllo dei canali RTP. È inoltre richiesto un protocollo per stabilire e rilasciare le connessioni, fornire i toni di composizione, produrre gli squilli e occuparsi di altri aspetti della telefonia standard: in questa area si usa ITU **Q.931**. I terminali necessitano di un protocollo per comunicare con il gatekeeper, se è presente, ed a questo scopo viene utilizzato **H.225**. Il canale PC/gatekeeper che gestisce è chiamato canale **RAS** (*Registration/Admission/Status*, registrazione/ammissione/stato). Questo canale consente ai terminali di entrare nella zona e di lasciarla, di richiedere e restituire banda, nonché di fornire aggiornamenti dello stato. Per finire, è necessario un protocollo per la trasmissione dei dati vera e propria: è stato stabilito di usare RTP, gestito come al solito da RTCP. La posizione di tutti questi protocolli è mostrata nella Figura 7.65.

Parlato	Controllo			
	G.7xx	RTCP	H.225 (RAS)	Q.931 (segnalazione chiamata)
RT P				
	UDP			
	TCP			
	IP			
	Protocollo dello strato data link			
	Protocollo dello strato fisico			

Figura 7.65. La pila di protocolli H.323.

Per vedere come questi protocolli operano insieme, si consideri il caso di un terminale PC su una LAN (con un gatekeeper) che chiama un telefono remoto. Il PC deve prima scoprire il gatekeeper, pertanto trasmette in broadcast un pacchetto UDP di rilevamento del gatekeeper sulla porta 1718.

Quando il gatekeeper risponde, il PC apprende il relativo indirizzo IP. Ora il PC si registra presso il gatekeeper inviando un messaggio RAS in un pacchetto UDP. Dopo l'accettazione, il PC invia al gatekeeper un messaggio di ammissione RAS richiedendo un po' di banda. Solo dopo che la banda è stata garantita può iniziare l'impostazione della chiamata. La richiesta anticipata della banda consente al gatekeeper di limitare il numero di chiamate, per impedire che la linea in uscita sia sottoscritta in eccesso e quindi fornire la qualità del servizio necessaria.

Il PC ora stabilisce una connessione TCP al gatekeeper per iniziare l'impostazione della chiamata. L'impostazione utilizza i protocolli della rete telefonica esistenti, che sono orientati alla connessione, pertanto è necessario TCP. Il sistema telefonico non dispone di qualcosa di simile a RAS per consentire ai telefoni di annunciare la loro presenza, pertanto i progettisti di H.323 sono stati liberi di utilizzare UDP o TCP per RAS: hanno così scelto UDP, a causa del suo overhead inferiore.

Dopo l'allocazione della banda, il PC può inviare un messaggio *SETUP* Q.931 sulla connessione TCP. Questo messaggio specifica il numero del telefono da chiamare (o l'indirizzo IP e la porta, se occorre chiamare un computer). Il gatekeeper risponde con un messaggio *CALL PROCEEDING* Q.931 per confermare la corretta ricezione della richiesta. Il gatekeeper inoltra quindi il messaggio *SETUP* al gateway.

Il gateway, che per metà è un computer e per metà un commutatore telefonico, esegue una telefonata ordinaria al telefono desiderato. Il telefono chiamato squilla e invia un messaggio *ALERT* Q.931 per comunicare al PC chiamante che il telefono ha iniziato a squillare. Quando la persona all'altro capo alza la cornetta, la centrale telefonica terminale restituisce un messaggio *CONNECT* Q.931 per segnalare al PC che possiede la connessione.

Una volta stabilita la connessione il gatekeeper non si trova più nel ciclo (al contrario del gateway, naturalmente). I pacchetti successivi scavalcano il gatekeeper e raggiungono direttamente l'indirizzo IP del gateway. A questo punto, si dispone di un semplice tubo tra due parti: si tratta solamente di una connessione allo strato fisico per lo spostamento dei bit. Ogni lato non conosce nulla sull'altro.

Il protocollo H.245 ora è utilizzato per negoziare i parametri della chiamata. Utilizza il canale di controllo H.245, che è sempre aperto. Ogni lato inizia annunciando le sue capacità, per esempio se è in grado di gestire il video (H.323 può gestire il video) o le chiamate in conferenza, quali codec supporta e così via. Quando ogni lato conosce ciò che l'altro può gestire, vengono impostati due canali dati unidirezionali, assegnando un codec e altri parametri a ognuno. Dal momento che ogni lato può avere attrezzature diverse, è possibile che i codec nei canali diretto e inverso siano differenti. Una volta completate tutte le negoziazioni, può iniziare il flusso dei dati che utilizza RTP. È gestito tramite RTCP, che svolge un ruolo importante nel controllo delle congestioni. Se è presente il video, RTCP gestisce la sincronizzazione audio/video. I canali sono mostrati nella Figura 7.66. Quando le parti riagganciano, il canale di segnalazione della chiamata Q.931 viene utilizzato per abbattere la connessione.

Una volta terminata la chiamata, il PC chiamante contatta di nuovo il gatekeeper con un messaggio RAS che chiede di rilasciare la banda assegnata. In alternativa, può eseguire un'altra chiamata.

Non abbiamo parlato della qualità del servizio (QoS, *Quality of Service*), anche se è un aspetto essenziale per il successo di Voice over IP. Il motivo è che QoS ricade all'esterno dell'ambito di H.323. Se la rete sottostante è in grado di produrre una connessione stabile e priva di jitter dal PC chiamante al gateway (per esempio utilizzando le tecniche discusse nel capitolo 5), allora il valore QoS della chiamata sarà buono; in caso contrario sarà scarso. La parte telefonica utilizza PCM ed è sempre priva di jitter.

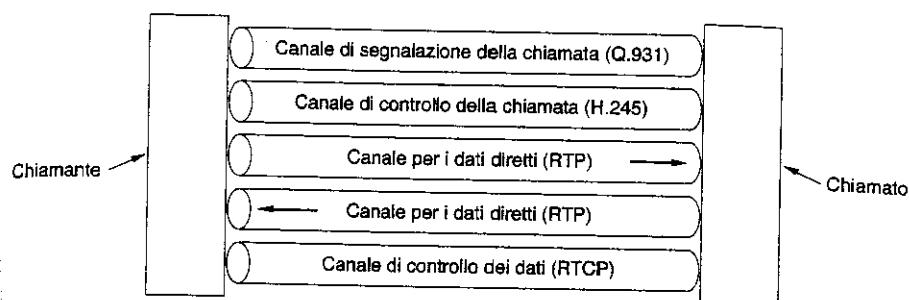


Figura 7.66. I canali logici tra chiamante e chiamato durante una chiamata.

SIP (*Session Initiation Protocol*)

H.323 è stato progettato da ITU. Molti appartenenti alla comunità Internet lo hanno considerato come un tipico prodotto per le telecomunicazioni: grande, complesso e poco flessibile. Di conseguenza, IETF ha creato un comitato per progettare un modo più semplice e modulare per gestire Voice over IP. Il miglior risultato a oggi è SIP (*Session Initiation Protocol*), descritto in RFC 3261. Questo protocollo descrive come impostare le telefonate Internet, le video conferenze e altre connessioni multimediali. A differenza di H.323, che è una suite di protocolli completa, SIP è un modulo singolo progettato per lavorare con applicazioni Internet esistenti. Per esempio definisce i numeri telefonici come URL, in modo che possano essere contenuti nelle pagine, consentendo di fare clic su un collegamento per avviare una telefonata. È un modo di procedere simile allo schema *mailto*, che consente di fare clic su un collegamento per aprire il programma per l'invio di posta elettronica.

SIP può stabilire sessioni tra due parti (telefonate ordinarie), sessioni tra più parti (dove chiunque può ascoltare e parlare), nonché sessioni multicast (un mittente, molti ricevitori). Le sessioni possono contenere audio, video o dati; questi ultimi servono (per esempio) per giochi multiplayer in tempo reale. SIP gestisce solamente l'impostazione, la gestione e la terminazione delle sessioni. Altri protocolli, come RTP/RTCP, sono utilizzati per il trasporto dei dati. SIP è un protocollo dello strato applicazione e può essere eseguito su UDP o TCP.

SIP supporta numerosi servizi, tra cui l'individuazione del chiamato (che potrebbe non trovarsi sul computer della propria abitazione) e la determinazione delle capacità del chiamato, nonché la gestione dei meccanismi di impostazione e termine della chiamata. Nel caso più semplice, SIP imposta una sessione dal computer del chiamante al computer del chiamato, quindi esamineremo per primo questo caso.

In SIP i numeri telefonici sono rappresentati come URL utilizzando lo schema *sip*, per esempio *sip:ilse@cs.university.edu* per un utente chiamato Ilse all'host specificato dal nome DNS *cs.university.edu*. Gli URL SIP possono anche contenere indirizzi IPv4, indirizzi IPv6 o numeri di telefono veri e propri.

Il protocollo SIP è un protocollo di testo modellato su HTTP. Una parte invia un messaggio di testo ASCII costituito da un nome di metodo sulla prima riga, seguito da altre righe con le intestazioni per il passaggio dei parametri. Molte intestazioni sono prese da MIME, per consentire a SIP di lavorare con applicazioni Internet esistenti. I sei metodi definiti dalle specifiche sono elencati nella Figura 7.67.

Metodo	Descrizione
INVITE	Richiede l'inizio di una sessione
ACK	Conferma l'inizio di una sessione
BYE	Richiede la fine di una sessione
OPTIONS	Interroga un host sulle sue capacità
CANCEL	Annulla una richiesta in sospeso
REGISTER	Informa un server di reindirizzamento sulla posizione corrente dell'utente

Figura 7.67. I metodi SIP definiti nella specifica di base.

Per stabilire una sessione, il chiamante crea una connessione TCP al chiamato e invia un messaggio *INVITE* su esso, oppure invia il messaggio *INVITE* in un pacchetto UDP. In entrambi i casi, le intestazioni della seconda riga e di quelle successive descrivono la struttura del corpo del messaggio, che contiene le capacità del chiamante, i tipi di media e i formati. Se il chiamato accetta la chiamata, risponde con un codice di risposta HTTP (un numero di tre cifre in base ai gruppi della Figura 7.42, 200 per l'accettazione). Dopo la riga con il codice di risposta, il chiamato può fornire informazioni sulle sue capacità, sui tipi di media e sui formati.

La connessione viene svolta con un handshake a tre vie, pertanto il chiamante risponde con un messaggio *ACK* per terminare il protocollo e confermare la ricezione del messaggio 200.

Entrambe le parti possono richiedere la fine di una sessione inviando un messaggio contenente il metodo *BYE*. Quando l'altro lato lo riconosce, la sessione viene terminata.

Il metodo *OPTIONS* è utilizzato per interrogare una macchina sulle sue capacità. È generalmente utilizzato prima dell'impostazione di una sessione per scoprire se la macchina è in grado di utilizzare Voice over IP o per capire quali tipi di sessione sono possibili.

Il metodo *REGISTER* è correlato alla capacità di SIP di tenere traccia di un utente lontano da casa e di collegarsi ad esso. Questo messaggio viene inviato a un location server SIP, che tiene traccia della posizione di ogni utente. Il server può essere successivamente interrogato per trovare la posizione corrente dell'utente. L'operazione di reindirizzamento è mostrata nella Figura 7.68. Qui il chiamante invia il messaggio *INVITE* a un server proxy, per nascondere il possibile reindirizzamento. Il proxy cerca l'utente e invia il messaggio *INVITE* verso la posizione individuata, poi agisce come un ripetitore per i successivi messaggi nell'handshake a tre vie. I messaggi *LOOKUP* e *REPLY* non sono parte di SIP: è possibile utilizzare qualsiasi protocollo adatto, in base al tipo di location server utilizzato.

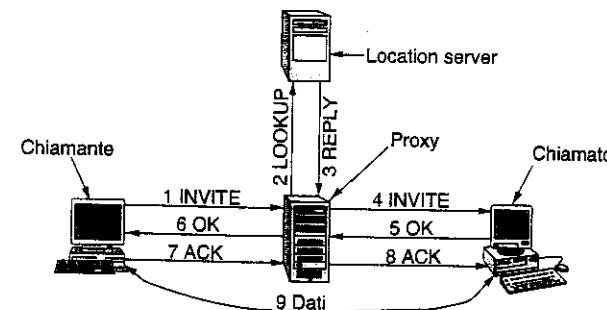


Figura 7.68. Utilizzo dei server di reindirizzamento e proxy con SIP.

SIP ha molte altre caratteristiche che non sono qui descritte, come l'avviso di chiamata, la selezione delle chiamate, la crittografia e l'autenticazione. Ha inoltre la capacità di eseguire chiamate da un computer a un telefono normale, se è disponibile un gateway adatto tra Internet e il sistema telefonico.

Un confronto tra H.323 e SIP

H.323 e SIP presentano molte somiglianze, ma anche alcune differenze. Entrambi consentono chiamate a due o più parti, utilizzando come punti finali computer e telefoni. Entrambi supportano la negoziazione dei parametri, la crittografia e i protocolli RTP/RTCP. Un riepilogo delle somiglianze e delle differenze è mostrato nella Figura 7.69. Anche se l'insieme delle funzionalità è simile, i due protocolli differiscono notevolmente nella filosofia. H.323 è un tipico standard dell'industria telefonica, che specifica tutta la pila di protocolli e definisce con precisione che cosa è consentito e che cosa è negato. Questo approccio tende a generare protocolli ben definiti in ogni strato, facilitando l'interoperabilità. Il prezzo pagato è uno standard grande, rigido e complesso, difficile da adattare alle applicazioni future.

In contrasto, SIP è un tipico protocollo Internet basato sullo scambio di brevi righe di testo ASCII. È un modulo leggero che interopera bene con altri protocolli Internet, ma meno bene con i protocolli di segnalazione del sistema telefonico esistente. Visto che il modello IETF di Voice over IP è altamente modulare, risulta flessibile e può essere adattato facilmente alle nuove applicazioni. Lo svantaggio è rappresentato da potenziali problemi di interoperabilità, che possono essere risolti dalle frequenti riunioni dove gli implementatori verificano i loro sistemi.

Voice over IP è un argomento promettente. Di conseguenza, esistono già diversi libri sull'argomento. Alcuni esempi sono (Collins, 2001; Davidson e Peters, 2000; Kumar et al., 2001; Wright, 2001). Il numero di maggio/giugno 2002 di *Internet Computing* contiene diversi articoli correlati.

Elemento	H.323	SIP
Progettato da	ITU	IETF
Compatibilità con PSTN	Sì	In gran parte
Compatibilità con Internet	No	Sì
Architettura	Monoitica	Modulare
Completezza	Pila di protocolli completa	SIP gestisce solo l'impostazione
Negoziazione dei parametri	Sì	Sì
Segnalazione della chiamata	Q.931 su TCP	SIP su TCP o UDP
Formato dei messaggi	Binario	ASCII
Trasporto	RTP/RTCP	RTP/RTCP
Chiamate di più parti	Sì	Sì
Conferenze multimediali	Sì	No
Indirizzamento	Numero di telefono o host	URL
Termino della chiamata	Esplicito o rilascio di TCP	Esplicito o timeout
Messaggistica immediata	No	Sì
Crittografia	Sì	Sì
Dimensione degli standard	1.400 pagine	250 pagine
Implementazione	Ampia e complessa	Moderata
Stato	Ampliamente utilizzato	In fase di sviluppo

Figura 7.69. Un confronto tra H.323 e SIP.

7.4.6 Introduzione al video

Abbiamo parlato a lungo dell'orecchio, ora passeremo all'occhio (tranquilli, questa sezione non è seguita da quella sul naso). L'occhio umano ha la proprietà di mantenere per qualche millisecondo l'immagine che appare sulla retina. Se una sequenza di immagini viene disegnata riga per riga a 50 immagini al secondo, l'occhio non capisce che sta osservando delle immagini discrete. Tutti i sistemi video (come la televisione) sfruttano questo principio per produrre immagini in movimento.

I sistemi analogici

Per comprendere il video è meglio iniziare con un semplice televisore in bianco e nero. Per rappresentare l'immagine bidimensionale con una tensione monodimensionale in funzione del tempo, la telecamera spazzola rapidamente con un fascio di elettroni l'immagine in senso orizzontale, facendo scorrere lentamente dall'alto verso il basso la riga di scansione mentre registra l'intensità della luce. Alla fine della scansione, chiamata **fotogramma** o **frame**, il fascio ritorna verso l'alto. L'intensità in funzione del tempo è una trasmissione, e i ricevitori ripetono il processo di scansione per ricostruire l'immagine. Lo sche-

ma di scansione utilizzato dalla telecamera e dal ricevitore è mostrato nella Figura 7.70 (tra parentesi, le telecamere CCD eseguono un'integrazione e non una scansione, ma alcuni modelli - e tutti i monitor - eseguono la scansione).

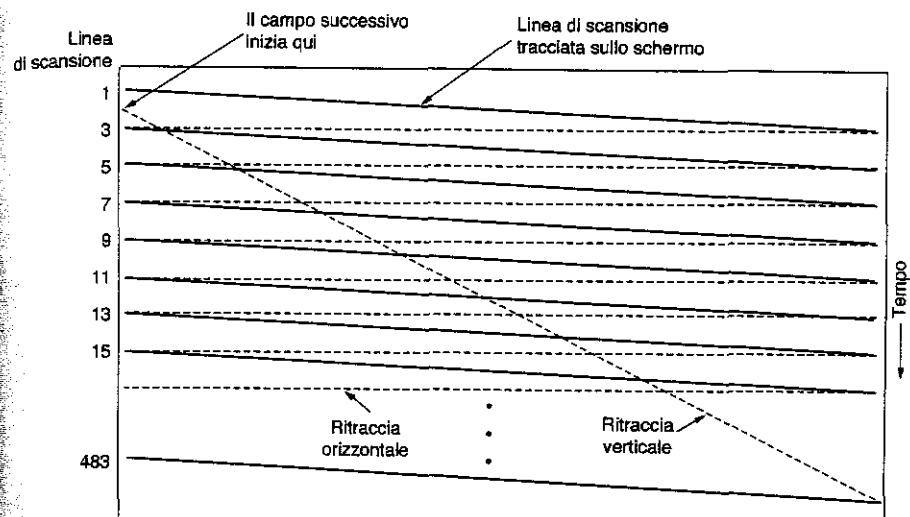


Figura 7.70. Lo schema di scansione utilizzato per la televisione e il video NTSC.

I parametri di scansione variano da nazione a nazione. Il sistema utilizzato in America e Giappone ha 525 linee di scansione, un rapporto orizzontale/verticale pari a 4:3, e 30 fotogrammi al secondo. Il sistema europeo utilizza 625 linee di scansione, lo stesso rapporto 4:3 e 25 fotogrammi al secondo. In entrambi i sistemi, le prime linee superiori e inferiori non sono visualizzate (per approssimare un'immagine rettangolare sui CRT arrotondati). Vengono quindi visualizzate solo 483 delle 525 linee di scansione NTSC (e 576 delle 625 linee di scansione PAL/SECAM). Il fascio viene disattivato mentre viene ripercorso il tracciato verticale, pertanto molte stazioni (specialmente in Europa) utilizzano questo tempo per trasmettere il televideo (pagine di testo con informazioni sulla cronaca, il tempo, lo sport, le azioni e così via).

Anche se 25 fotogrammi al secondo sono sufficienti per acquisire un movimento fluido, a questa frequenza dei fotogrammi molte persone (specialmente le più anziane) percepiscono un certo tremolio dell'immagine, perché la vecchia immagine è già scomparsa dalla retina prima della comparsa di quella nuova. Anziché aumentare la frequenza dei fotogrammi, che richiederebbe l'utilizzo di altra preziosa banda, si segue un approccio diverso. Anziché visualizzare le linee di scansione in ordine, vengono prima visualizzate tutte le linee di scansione dispari e poi quelle pari. Ogni elemento rappresentato da metà dei fotogrammi è chia-

mato campo (**field**). Alcuni esperimenti hanno mostrato che anche le persone che notano un tremolio a 25 fotogrammi al secondo non lo rilevano a 50 campi al secondo. Questa tecnica è chiamata **interlacciamento**. Le televisioni o il video non interlacciati sono detti **progressivi**. Occorre notare che i film vengono eseguiti a 24 f/s, ma ogni fotogramma resta completamente visibile per 1/24 di secondo.

Il video a colori utilizza lo stesso schema di scansione del monocromatico (bianco e nero), tranne per il fatto che utilizza tre fasci in movimento all'unisono per visualizzare l'immagine (anziché uno solo). Ogni fascio viene utilizzato per uno dei tre colori primari additivi: rosso, verde e blu (RGB). Questa tecnica funziona perché qualsiasi colore può essere costruito da una sovrapposizione lineare di rosso, verde e blu con le intensità appropriate. Tuttavia, per la trasmissione su un singolo canale, i tre segnali di colore devono essere combinati in un singolo segnale **composito**.

Quando fu inventata la televisione a colori, erano disponibili più metodi per visualizzare il colore; ogni nazione fece la sua scelta, producendo sistemi tuttora incompatibili (occorre notare che queste scelte non hanno a che fare con il confronto tra VHS, Betamax e P2000, che sono metodi di registrazione). In tutte le nazioni, un requisito politico prevedeva che i programmi trasmessi a colori fossero ricevibili anche sui televisori in bianco e nero esistenti. Di conseguenza, lo schema più semplice (la codifica separata dei segnali RGB) non era accettabile; per giunta RGB non è lo schema più efficiente.

Il primo sistema a colori fu standardizzato negli Stati Uniti da **National Television Standards Committee**, che assegnò come nome allo standard il suo acronimo: **NTSC**. La televisione a colori fu introdotta in Europa diversi anni dopo: durante quel periodo la tecnologia si era notevolmente evoluta, portando a sistemi con maggiore immunità dai disturbi e colori migliori. Questi sistemi sono chiamati **SECAM** (*Sequentiel Couleur Avec Memoire*), utilizzato in Francia e nell'Europa dell'est, e **PAL** (*Phase Alternating Line*), utilizzato nel resto dell'Europa. La differenza nella qualità del colore tra NTSC e PAL/SECAM ha prodotto un divertente aneddoto industriale: sembra che NTSC significhi in realtà *Never Twice the Same Color*, nel senso che lo stesso colore non appare mai due volte.

Per consentire la visione delle trasmissioni a colori su ricevitori in bianco e nero, tutti i tre sistemi combinano linearmente i segnali RGB in un segnale di **luminanza** (luminosità) e due segnali di **crominanza** (colore), pur utilizzando coefficienti diversi per costruire questi segnali dai segnali RGB. Stranamente l'occhio è più sensibile al segnale di luminanza che ai segnali di crominanza, che per questo motivo non hanno bisogno di essere trasmessi accuratamente. La conseguenza è che il segnale di luminanza può essere trasmesso alla stessa frequenza del vecchio segnale in bianco e nero, in modo tale da poter essere ricevuto su televisori in bianco e nero. I due segnali di crominanza sono trasmessi in bande ridotte, a frequenze elevate. Alcuni televisori dispongono di controlli denominati luminosità, tonalità e saturazione (o luminosità, tinta e colore) per controllare questi tre segnali separatamente. Comprendere la distinzione tra luminanza e la crominanza è necessario per capire come funziona la compressione del video.

Negli ultimi anni, c'è stato un notevole interesse per **HDTV** (*High Definition TeleVision*), che produce immagini più nitide raddoppiando il numero di linee di scansione. Gli Stati Uniti, l'Europa e il Giappone hanno già sviluppato sistemi HDTV, tutti diversi e tutti mutuamente

incompatibili. Potevamo aspettarci altrimenti? I principi di base di HDTV in termini di scansione, luminanza, crominanza e così via sono simili a quelli dei sistemi esistenti. Tuttavia, i tre formati hanno un rapporto comune di 16:9 anziché 4:3, affinché corrispondano meglio al formato utilizzato per i film (registrati su pellicola a 35 mm, con un rapporto di 3:2).

I sistemi digitali

La rappresentazione più semplice del video digitale è una sequenza di fotogrammi, ognuno composto da una griglia rettangolare di elementi chiamati **pixel**. Ogni pixel può essere un singolo bit, per rappresentare il bianco o il nero. La qualità di un sistema di questo tipo assomiglia a quella ottenibile inviando una fotografia a colori tramite fax: orribile! Per farsi un'idea, se non si dispone di un apparecchio fax, provare a fotocopiare una fotografia a colori con una fotocopiatrice tradizionale che non svolge la rasterizzazione.

Il passaggio successivo è utilizzare 8 bit per pixel, per rappresentare 256 livelli di grigio. Questo schema garantisce un video in bianco e nero di alta qualità. Per il video a colori i sistemi di buona qualità utilizzano 8 bit per ogni colore RGB, anche se quasi tutti li mescolano in un segnale video composito per la trasmissione. Anche se l'utilizzo di 24 bit per pixel limita il numero di colori a circa 16 milioni, è un numero comunque superiore alle capacità di analisi dell'occhio umano. Le immagini a colori digitali sono prodotte utilizzando tre fasci di scansione, uno per colore. La geometria è la stessa del sistema analogico della Figura 7.70, tranne per il fatto che le linee di scansione continue vengono sostituite da righe di pixel discreti.

Per produrre un movimento fluido, il video digitale (come quello analogico) deve visualizzare almeno 25 fotogrammi al secondo. Tuttavia, visto che i monitor informatici di buona qualità spesso ripetono la scansione dello schermo almeno 75 volte al secondo, l'interlacciamento non è necessario e di conseguenza non viene utilizzato. È sufficiente ridisegnare lo stesso fotogramma tre volte di seguito per eliminare il tremolio.

In altre parole, il movimento fluido è determinato dal numero di immagini *diverse* al secondo, mentre il tremolio è determinato dal numero di aggiornamenti dello schermo per secondo. Questi due parametri sono diversi. Un'immagine statica ridisegnata a 20 fotogrammi al secondo non mostra un movimento a scatti, ma tremerà perché il fotogramma abbandona la retina prima della comparsa della successiva immagine. Un filmato con 20 fotogrammi al secondo, ognuno ridisegnato quattro volte di seguito, non sarà tremolante, ma il movimento apparirà a scatti.

Il significato di questi due parametri diventa chiaro quando si considera la banda richiesta per trasmettere il video digitale su una rete. Gli attuali monitor dei computer devono utilizzare il rapporto 4:3 per sfruttare gli economici cinescopi prodotti per il mercato dei televisori. Le configurazioni più comuni sono 1.024 x 768, 1.280 x 960 e 1.600 x 1.200 pixel. Anche la più bassa, con 24 bit per pixel e 25 fotogrammi al secondo, deve essere alimentata a 472 Mbps. È necessaria una portante SONET OC-12 per questo scopo, ma naturalmente non è una soluzione a portata degli utenti domestici. Raddoppiare questa velocità per evitare i tremoli è ancora meno attraente. Una soluzione migliore richiede di trasmettere 25 fotogrammi al secondo e fare in modo che il computer memorizzi ciascuno di essi, disegnandolo due volte. Le stazioni televisive non utilizzano questa strategia, perché i tele-

visori non dispongono di memoria, ma anche se l'avessero i segnali analogici non possono essere memorizzati nella RAM senza la conversione in forma digitale, e questo richiederebbe altro hardware. Di conseguenza, l'interlacciamento è necessario per la televisione ordinaria ma non per il video digitale.

7.4.7 La compressione video

Ora dovrebbe essere ovvio che la trasmissione di video non compresso è assolutamente irrealizzabile; l'unica speranza è che sia possibile una forte compressione. Fortunatamente, negli scorsi decenni molti ricercatori hanno messo a punto algoritmi e tecniche di compressione che rendono possibile la trasmissione video. In questa sezione studieremo come viene svolta la compressione.

Tutti i sistemi di compressione richiedono due algoritmi: uno per la compressione dei dati all'origine e l'altro per la decompressione alla destinazione. Questi algoritmi sono definiti rispettivamente algoritmo di **codifica** e algoritmo di **decodifica**. Useremo questa terminologia nella parte restante del capitolo.

Questi algoritmi presentano alcune asimmetrie che è importante comprendere. Innanzitutto, per molte applicazioni (per esempio un documento multimediale) un filmato viene codificato una volta sola, quando viene memorizzato sul server multimediale, ma sarà decodificato migliaia di volte, quando viene visto dai clienti. Questa asimmetria segnala che è accettabile che l'algoritmo di codifica sia lento e richieda hardware costoso, purché l'algoritmo di decodifica sia rapido e non richieda hardware costoso. Dopo tutto, l'operatore di un server multimediale può essere disposto a noleggiare un supercomputer parallelo per qualche settimana per codificare la sua intera libreria di video, ma chiedere ai consumatori di noleggiare un supercomputer per due ore solo per guardare un video non è certo una buona idea. Molti sistemi di compressione usati nella realtà hanno reso la decodifica semplice e veloce, anche al prezzo di una codifica lenta e complessa.

Tuttavia, per i file multimediali in tempo reale (come le videoconferenze), una codifica lenta non è più accettabile. La codifica deve avvenire all'istante, in tempo reale. Di conseguenza, il multimedia in tempo reale utilizza algoritmi o parametri diversi da quelli impiegati per memorizzare i video su disco, spesso con una compressione notevolmente inferiore.

Una seconda asimmetria riguarda il fatto che non è necessario rendere reversibile il processo di codifica/decodifica. Quando si comprime un file, lo si trasmette e lo si decomprime, l'utente si aspetta di ricevere l'originale, accurato fino all'ultimo bit. Con il multimedia questo requisito non esiste. Solitamente è accettabile che il segnale video che è stato codificato e poi decodificato sia leggermente diverso dall'originale. Quando l'output decodificato non è esattamente uguale all'input originale, si dice che il sistema è di tipo **con perdite**. Se l'input e l'output sono identici, il sistema è **senza perdite**. I sistemi con perdite sono importanti perché accettando una piccola perdita di informazioni si può ottenere un rapporto di compressione elevato.

Lo standard JPEG

Un video è semplicemente una sequenza di immagini (con l'audio). Se trovassimo un valido algoritmo per la codifica di una singola immagine, potrebbe essere applicato a ogni immagine in successione per ottenere la compressione del video. Esistono validi algoritmi di compressione delle immagini statiche: il nostro studio della compressione video inizia proprio da qui. Lo standard **JPEG** (*Joint Photographic Experts Group*) per la compressione di immagini statiche a toni continui (le fotografie) è stato sviluppato da esperti di fotografia sotto gli auspici di ITU, ISO e IEC (un altro gruppo per la standardizzazione). È importante per il multimediale perché, in prima approssimazione, lo standard multimediale per il trasferimento d'immagini MPEG equivale alla codifica JPEG di ogni singolo fotogramma, con alcune funzionalità extra per la compressione tra fotogrammi e il rilevamento del movimento. JPEG è definito nello standard internazionale 10918.

JPEG presenta quattro modalità e molte opzioni; è più simile a una lista della spesa che a un singolo algoritmo. Per i nostri scopi, però, è rilevante solo la modalità sequenziale con perdite, quella illustrata nella Figura 7.71. Inoltre, ci concentreremo sul modo in cui JPEG viene normalmente utilizzato per codificare le immagini video RGB a 24 bit e tralascieremo alcuni dettagli per semplicità.



Figura 7.71. Il funzionamento di JPEG nella modalità sequenziale con perdite.

Il passaggio 1 della codifica di un'immagine con JPEG è la preparazione del blocco. Per semplicità, supponiamo che l'input JPEG sia un'immagine RGB 640 x 480 con 24 bit per pixel, come mostrato nella Figura 7.72(a). Visto che l'utilizzo della luminanza e della crominanza offre una compressione migliore, calcoleremo prima la luminanza, Y , e poi le due crominanze, I e Q (per NTSC), secondo le seguenti formule:

$$Y = 0,30R + 0,59G + 0,11B$$

$$I = 0,60R - 0,28G - 0,32B$$

$$Q = 0,21R - 0,52G + 0,31B$$

Per PAL, le crominanze sono chiamate U e V e i coefficienti sono diversi, ma l'idea è la stessa. SECAM è diverso sia da NTSC sia da PAL.

Per Y , I e Q vengono costruite matrici separate, ognuna con elementi nell'intervallo da 0 a 255. Successivamente, viene calcolata la media di blocchi quadrati di 4 pixel nelle matrici I e Q , al fine di ridurle a 320 x 240. Questa riduzione è con perdite, ma l'occhio lo nota poco perché apprezza la luminanza più della crominanza. Ciò nonostante, il totale dei dati viene compresso di un fattore pari a due. Ora viene sottratto 128 da ogni elemento di tutte

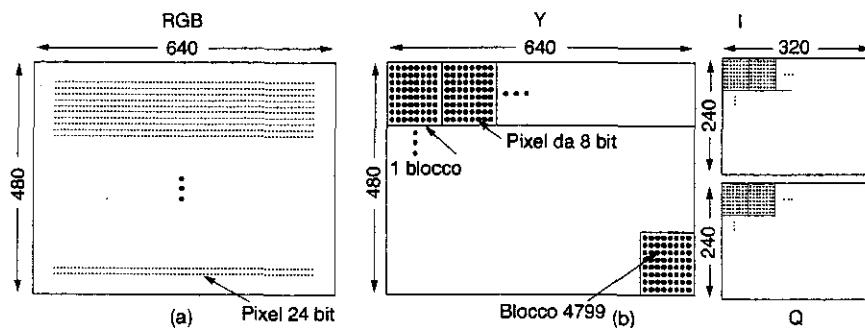


Figura 7.72. (a) I dati di input RGB. (b) Dopo la preparazione del blocco.

le tre matrici per porre 0 al centro dell'intervallo. Infine, ogni matrice è divisa in blocchi di 8×8 . La matrice Y ha 4.800 blocchi e le altre due contengono 1200 blocchi ognuna, come appare in Figura 7.72(b).

La seconda fase di JPEG consiste nell'applicare una **DCT** (Discrete Cosine Transformation, trasformata coseno discreta) a ognuno dei 7200 blocchi separatamente. L'output di ogni DCT è una matrice 8×8 di coefficienti DCT, dove l'elemento DCT (0, 0) è il valore medio del blocco, e gli altri elementi indicano la quantità di potenza spettrale che è presente per ogni frequenza spaziale. In teoria DCT è una trasformazione senza perdite, ma in pratica l'utilizzo dei numeri in virgola mobile e delle funzioni trascendentali introduce un errore di arrotondamento che provoca una piccola perdita di informazioni. Normalmente questi elementi diminuiscono rapidamente all'aumento della distanza dall'origine (0, 0), come suggerito dalla Figura 7.73.

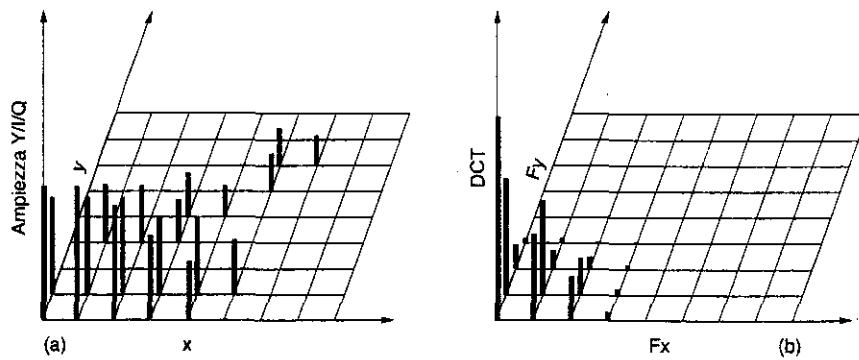


Figura 7.73. (a) Un blocco della matrice Y . (b) I coefficienti DCT.

Una volta completata la DCT, JPEG passa alla fase 3, chiamata **quantizzazione**, in cui vengono eliminati i coefficienti DCT meno importanti. Questa trasformazione con perdite viene eseguita dividendo ogni coefficiente nella matrice DCT 8×8 per un peso estrat-

to da una tabella. Se tutti i pesi corrispondono a 1, la trasformazione non ha effetti. Tuttavia, se i pesi aumentano in modo netto dall'origine, le frequenze spaziali alte vengono rapidamente eliminate.

Un esempio di questo passaggio è mostrato nella Figura 7.74. Qui vediamo la matrice DCT iniziale, la tabella di quantizzazione e il risultato ottenuto dividendo ogni elemento DCT per il corrispondente elemento della tabella di quantizzazione. I valori nella tabella di quantizzazione non fanno parte dello standard JPEG; ogni applicazione deve fornire i propri, e ciò permette di controllare i compromessi della compressione con perdite.

Coefficienti DCT	Tabella di quantizzazione	Coefficienti di quantizzazione
150 80 40 14 4 2 1 0	1 1 2 4 8 16 32 64	150 80 20 4 1 0 0 0
92 75 36 10 6 1 0 0	1 1 2 4 8 16 32 64	92 75 18 3 1 0 0 0
52 38 26 8 7 4 0 0	2 2 2 4 8 16 32 64	26 19 13 2 1 0 0 0
12 8 6 4 2 1 0 0	4 4 4 4 8 16 32 64	3 2 2 1 0 0 0 0
4 3 2 0 0 0 0 0	8 8 8 8 8 16 32 64	1 0 0 0 0 0 0 0
2 2 1 0 0 0 0 0	16 16 16 16 16 16 32 64	0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0	32 32 32 32 32 32 32 64	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	64 64 64 64 64 64 64 64	0 0 0 0 0 0 0 0

Figura 7.74. Il calcolo dei coefficienti DCT quantizzati.

Il passaggio 4 riduce il valore (0, 0) di ogni blocco (quello nell'angolo in alto a sinistra) sostituendolo con la quantità di cui differisce dall'elemento corrispondente nel blocco precedente. Visto che questi elementi sono la media dei rispettivi blocchi, dovrebbero cambiare lentamente, pertanto la scelta dei valori differenziali dovrebbe ridurrne la maggior parte a valori piccoli. Nessun differenziale è elaborato per gli altri valori. I valori (0, 0) sono definiti componenti DC, gli altri valori sono i componenti AC.

La fase 5 linearizza i 64 elementi e applica la codifica run-length all'elenco. La scansione del blocco da sinistra a destra e poi dall'alto verso il basso non concentrerà gli zero insieme, pertanto viene utilizzato uno schema di scansione a zig zag, come mostrato nella Figura 7.75. In questo esempio, lo schema a zig zag produce 38 zero consecutivi alla fine della matrice. Questa stringa può essere ridotta a un singolo conteggio che afferma la presenza di 38 zero: la tecnica è nota come **codifica run-length**.

Ora abbiamo un elenco di numeri che rappresentano l'immagine (nello spazio di trasformazione). Il passaggio 6 applica ai numeri la codifica di Huffman per l'archiviazione o la trasmissione, assegnando ai numeri più frequenti codici più brevi di quelli meno frequenti.

JPEG può sembrare complicato, e in effetti lo è, tuttavia è molto diffuso perché spesso realizza una compressione con rapporto 20:1 o anche migliore. La decodifica di un'immagine JPEG richiede l'esecuzione inversa dell'algoritmo. JPEG è approssimativamente simmetrico: la decodifica richiede circa lo stesso tempo della codifica. Questa proprietà non vale per tutti gli algoritmi di compressione, come vedremo tra poco.

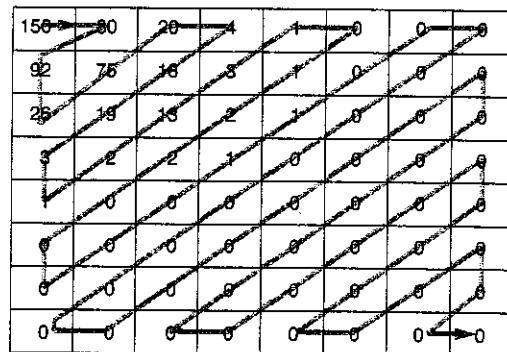


Figura 7.75. L'ordine in cui sono trasmessi i valori quantizzati.

Lo standard MPEG

Eccoci al cuore della questione: gli standard **MPEG** (*Motion Picture Experts Group*). Si tratta degli algoritmi principali utilizzati per comprimere i video e sono divenuti standard internazionali da 1993. Dal momento che i filmati possono contenere immagini e suoni, MPEG può comprimere sia l'audio sia il video. Abbiamo già esaminato la compressione di audio e immagini statiche: ora vedremo la compressione del video.

Il primo standard finalizzato fu MPEG-1 (International Standard 11172). Il suo scopo era produrre un output di qualità simile a quella di un videoregistratore (cioè 352 x 240 nel caso di NTSC) utilizzando un bit rate pari a 1,2 Mbps. Un'immagine di 352 x 240 pixel con 24 bit per pixel e 25 fotogrammi al secondo richiede 50,7 Mbps, pertanto portarla a 1,2 Mbps non è poi così semplice: serve un fattore di compressione pari a 40. MPEG-1 può essere trasmesso su brevi distanze usando linee di trasmissione a doppini, ed è utilizzato anche per memorizzare filmati su CD-ROM.

Lo standard successivo nella famiglia MPEG è MPEG-2 (International Standard 13818), che in origine è stato progettato per comprimere video di qualità broadcast a bit-rate compresi tra 4 e 6 Mbps, per poter essere inserito in un canale di trasmissione NTSC oppure PAL. In seguito, MPEG-2 fu espanso per supportare risoluzioni superiori, compreso HDTV. Ora è molto comune, visto che forma le basi per i DVD e le televisioni satellitari digitali.

I principi di base di MPEG-1 e MPEG-2 sono simili, ma i dettagli sono differenti. A una prima approssimazione MPEG-2 è un superset di MPEG-1, con funzionalità, formati e opzioni di codifica aggiuntivi. Parleremo prima di MPEG-1 e poi di MPEG-2.

MPEG-1 è composto da tre parti: audio, video e sistema, che integra le altre due come mostrato nella Figura 7.76. I codificatori audio e video lavorano in modo indipendente, facendo nascere il problema relativo alla sincronizzazione dei due flussi nel ricevitore. Questo problema viene risolto con un clock di sistema a 90 kHz che ermette in output il valore del tempo corrente verso entrambi i codificatori. Questi valori sono a 33 bit, per

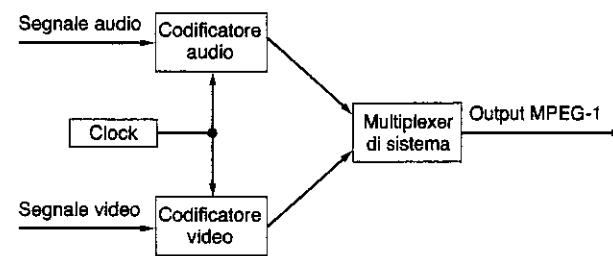


Figura 7.76. La sincronizzazione dei flussi audio e video in MPEG-1.

supportare film con durata massima di 24 ore senza traboccati. Questi indicatori di data e ora sono inclusi nell'output codificato e propagati al ricevitore, che può utilizzarli per sincronizzare i flussi audio e video.

Consideriamo ora la compressione video di MPEG-1. Nei filmati esistono due tipi di ridondanze: spaziale e temporale; MPEG-1 le utilizza entrambe. La ridondanza spaziale può essere utilizzata semplicemente codificando ogni fotogramma separatamente con JPEG. Questo approccio è utilizzato occasionalmente, specialmente quando è necessario l'accesso casuale a ogni fotogramma, come nelle attività di montaggio video. In questa modalità, è raggiungibile una banda compressa nell'intervallo 8-10 Mbps.

Una compressione aggiuntiva può essere raggiunta traendo vantaggio dal fatto che i fotogrammi consecutivi sono quasi identici. Questo effetto è minore di quanto potrebbe sembrare, perché molti produttori eseguono un taglio tra le scene ogni 3 o 4 secondi (è possibile contare il tempo e le scene per verificare). Ciò nonostante, anche un insieme di 75 fotogrammi molto simili offre il potenziale per una riduzione maggiore rispetto alla codifica separata di ogni fotogramma con JPEG.

Nelle scene in cui la cinepresa e lo sfondo sono statici e uno o due attori si spostano lentamente, quasi tutti i pixel saranno identici di fotogramma in fotogramma. In questo caso è sufficiente sottrarre ogni fotogramma dal precedente ed eseguire JPEG sulla differenza. Tuttavia, questa tecnica non è applicabile per le scene dove la cinepresa esegue una panoramica o uno zoom; serve perciò un modo per compensare questo movimento. Questo è precisamente lo scopo di MPEG, ed è la differenza principale tra MPEG e JPEG.

L'output di MPEG-1 è composto di quattro tipi di fotogrammi:

1. fotogrammi I (intracodificati): immagini statiche auto-contenute codificate con JPEG
2. fotogrammi P (predittivi): differenze blocco per blocco con l'ultimo fotogramma
3. fotogrammi B (bidirezionali): differenze tra l'ultimo fotogramma e il successivo
4. fotogrammi D (codificati DC): medie dei blocchi, utilizzati per l'avvolgimento rapido.

I fotogrammi I sono semplici immagini statiche codificate utilizzando una variante di JPEG, che utilizza la luminanza alla risoluzione completa e la crominanza a mezza risoluzione su entrambi gli assi. È necessario che i fotogrammi I appaiano periodicamente nel flusso di output per tre motivi. Innanzitutto MPEG-1 può essere utilizzato per una trasmissione multicast, dove gli utenti si sintonizzano in qualunque momento. Se tutti i fotogrammi dipendessero dai predecessori, fino al primo fotogramma, chi si è perso il primo fotogramma non potrà mai decodificare i successivi. Secondo, se si verificasse un errore su un fotogramma ricevuto, non sarebbe possibile un'ulteriore decodifica. Terzo, senza i fotogrammi I, durante un avvolgimento rapido o un riavvolgimento il decodificatore dovrebbe calcolare ogni fotogramma saltato per conoscere il valore completo di quello su cui viene fermato. Per questi motivi, i fotogrammi I sono inseriti nell'output una o due volte al secondo.

I fotogrammi P, invece, codificano le differenze intrafotogramma. Sono basati sull'idea dei **macroblocchi**, che coprono 16×16 pixel nello spazio della luminanza e 8×8 pixel nello spazio della crominanza. Un macroblocco viene codificato ricercando il fotogramma precedente o qualcosa che sia solo leggermente diverso.

Un esempio di utilità dei fotogrammi P è mostrato nella Figura 7.77. Qui sono mostrati tre fotogrammi consecutivi con lo stesso sfondo, ma posizioni diverse della persona. I macroblocchi contenenti la scena dello sfondo corrispondono esattamente, mentre quelli con la persona sono scostati di una quantità sconosciuta e devono essere rintracciati.

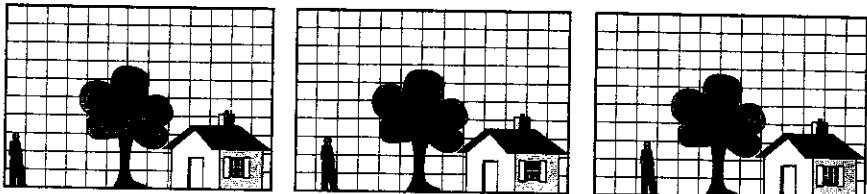


Figura 7.77. Tre fotogrammi consecutivi.

Lo standard MPEG-1 non specifica come cercare, dove cercare o il grado di corrispondenza necessario. Tutto questo è lasciato all'implementazione. Per esempio, un'implementazione potrebbe cercare un macroblocco nella posizione corrente del fotogramma precedente, e in tutte le altre posizioni scostate di $\pm Dx$ nella direzione x e $\pm Dy$ nella direzione y . Per ogni posizione, deve essere elaborato il numero di corrispondenze nella matrice della luminanza. La posizione con il punteggio più alto sarà decretata vincitrice, purché si trovi sopra una soglia predefinita, altrimenti il macroblocco viene considerato mancante. Naturalmente sono disponibili anche algoritmi più sofisticati.

Se viene trovato un macroblocco, questo viene codificato calcolandone la differenza con il valore nel fotogramma precedente per la luminanza ed entrambe le crominanze. Queste matrici delle differenze sono poi soggette alla trasformazione coseno discreta, alla quantizzazione, alla codifica run-length e alla codifica di Huffman, proprio come JPEG. Il valore per il macroblocco nel flusso di output è quindi il vettore di movimento (la distanza di

cui il macroblocco è stato spostato dalla posizione precedente in ogni direzione), seguito da un elenco di numeri con codifica Huffman. Se il macroblocco non è situato nel fotogramma precedente, il valore corrente è codificato con JPEG, come per un fotogramma I. Chiaramente, questo algoritmo è altamente asimmetrico. Un'implementazione è libera di provare ogni posizione plausibile nel fotogramma precedente, se lo desidera, in un tentativo disperato di individuare ogni macroblocco precedente, indipendentemente dalla posizione in cui è stato spostato. Questo approccio minimizzerà il flusso MPEG-1 codificato a spese di una codifica molto lenta. L'approccio può essere valido per codificare un film, ma sarebbe terribile per una videoconferenza in tempo reale.

Allo stesso modo, ogni implementazione è libera di decidere che cosa costituisce un macroblocco "individuato". Questa libertà permette agli implementatori di competere sulla qualità e la velocità dei loro algoritmi, ma produce sempre implementazioni compatibili con MPEG-1. Indipendentemente dall'algoritmo di ricerca utilizzato, l'output finale è la codifica JPEG del macroblocco corrente o la codifica JPEG della differenza tra il macroblocco corrente e quello nel fotogramma precedente, a un offset specificato rispetto a quello corrente.

Fino a questo punto la decodifica di MPEG-1 è elementare. La decodifica dei fotogrammi I equivale alla decodifica delle immagini JPEG; la decodifica dei fotogrammi P richiede al decoder di inserire nel buffer il fotogramma precedente e di costruire quello attuale in un secondo buffer, in base ai macroblocchi completamente codificati e ai macroblocchi che contengono differenze rispetto al fotogramma precedente. Il nuovo fotogramma viene assemblato macroblocco per macroblocco.

I fotogrammi B sono simili ai fotogrammi P, tranne per il fatto che consentono al macroblocco di riferimento di trovarsi sia in un fotogramma precedente sia in un fotogramma successivo. Questa libertà aggiuntiva permette una compensazione del movimento migliorata ed è utile quando gli oggetti passano davanti o dietro ad altri oggetti. Per la codifica dei fotogrammi B, il codificatore deve disporre di tre fotogrammi decodificati in memoria: il precedente, quello attuale e il successivo. Anche se i fotogrammi B offrono la migliore compressione, non tutte le implementazioni li supportano.

I fotogrammi D sono utilizzati solo per consentire la visualizzazione di un'immagine a bassa risoluzione durante un riavvolgimento o un avvolgimento rapido. Svolgere la normale decodifica MPEG-1 in tempo reale è abbastanza difficile. Aspettarsi che il decoder svolga l'operazione mentre riproduce il video a una velocità dieci volte superiore a quella normale è chiedere un po' troppo. I fotogrammi D sono invece utilizzati per produrre immagini a bassa risoluzione. Ogni fotogramma D rappresenta solamente il valor medio di un blocco, senza ulteriore codifica, in modo da facilitarne la visualizzazione in tempo reale. Questa caratteristica è importante per consentire alle persone di analizzare un video ad alta velocità alla ricerca di una scena particolare. I fotogrammi D sono generalmente posti appena prima dei corrispondenti fotogrammi I, in modo che all'interruzione di un avvolgimento veloce sia possibile iniziare la visualizzazione alla velocità normale.

Terminata l'analisi di MPEG-1, è possibile passare a MPEG-2. La codifica MPEG-2 è sostanzialmente simile alla codifica MPEG-1, con fotogrammi I, P e B. I fotogrammi D, invece, non sono supportati. Inoltre, la trasformazione coseno discreta utilizza un blocco di 10×10 anziché un blocco di 8×8 , per offrire il 50% di coefficienti in più (producendo così una qualità

migliore). Visto che MPEG-2 è destinato alle televisioni in broadcast e ai DVD, supporta sia le immagini progressive sia quelle interallacciate, al contrario di MPEG-1 che supporta solo le immagini progressive. Altri dettagli minori sono diversi tra i due standard.

Anziché supportare un solo livello di risoluzione, MPEG-2 ne supporta quattro: low (352 x 240), main (720 x 480), high-1440 (1.440 x 1.152) e high (1.920 x 1.080). La risoluzione low (bassa) è per i VCR e per la compatibilità con MPEG-1. Main è la risoluzione normale per il broadcasting NTSC; le altre due sono destinate a HDTV. Per un output di alta qualità, MPEG-2 viene normalmente eseguito a 4-8 Mbps.

7.4.8 Il video on demand

A volte il video su richiesta (*video on demand*) è paragonato a una videoteca elettronica. L'utente (o il cliente) seleziona un video tra quelli disponibili e lo porta a casa per vederlo, ma con il video su richiesta la selezione viene eseguita da casa utilizzando il telecomando del televisore, evitando i viaggi verso il negozio; inoltre il video inizia immediatamente. Ovviamente, l'implementazione del video su richiesta è più complicata della sua descrizione. In questa sezione vedremo una panoramica delle idee di base e della loro implementazione.

Il video su richiesta è davvero simile al noleggio di un video, o è più simile alla scelta di un filmato da guardare su un sistema via cavo a 500 canali? La risposta presenta importanti implicazioni tecniche. In particolare, gli utenti della videoteca sono abituati all'idea di poter fermare il video, recarsi per un istante in cucina o in bagno, e poi riprendere la riproduzione da dove era stata interrotta. I telespettatori invece non si aspettano di poter mettere i programmi in pausa.

Se il video su richiesta deve competere con successo con le videoteche, potrebbe essere necessario consentire agli utenti di fermare, avviare e riavvolgere i video. Dando agli utenti questa capacità, si obbliga il fornitore del video a trasmetterne una copia separata a ognuno.

D'altra parte, se il video su richiesta è visto come una sorta di televisione evoluta, potrebbe essere sufficiente fare in modo che il fornitore del video lo avvii per esempio ogni 10 minuti, facendolo procedere senza interruzioni. Un utente che desidera vedere un video potrebbe dover attendere per 10 minuti il suo inizio. Anche se in questo caso non è possibile mettere in pausa e riprendere, un visitatore che torna in soggiorno dopo una breve pausa può scegliere un altro canale dove viene riprodotto lo stesso video a 10 minuti di distanza. Parte del materiale sarà ripetuto, ma non mancherà nulla. Questo schema è chiamato **near video on demand**. Offre il potenziale del video on demand a un costo inferiore, perché la stessa emissione del server video è distribuita a molti utenti contemporaneamente. La differenza con il video su richiesta vero e proprio è simile alla differenza tra guidare la propria automobile e prendere l'autobus.

La possibilità di richiedere un video è solo uno dei potenziali nuovi servizi messi a disposizione dal networking su banda larga. Il modello generale che viene generalmente utiliz-

zato è illustrato nella Figura 7.78. La figura mostra al centro del sistema una rete dorsale estesa (nazionale o internazionale) con una banda elevata. A questa sono collegate centinaia di reti di distribuzione locali, per esempio sistemi di distribuzione delle società telefoniche o della TV via cavo. I sistemi di distribuzione locale raggiungono le abitazioni degli utenti, dove terminano all'interno di **set-top box**, che sono potenti personali computer specializzati.

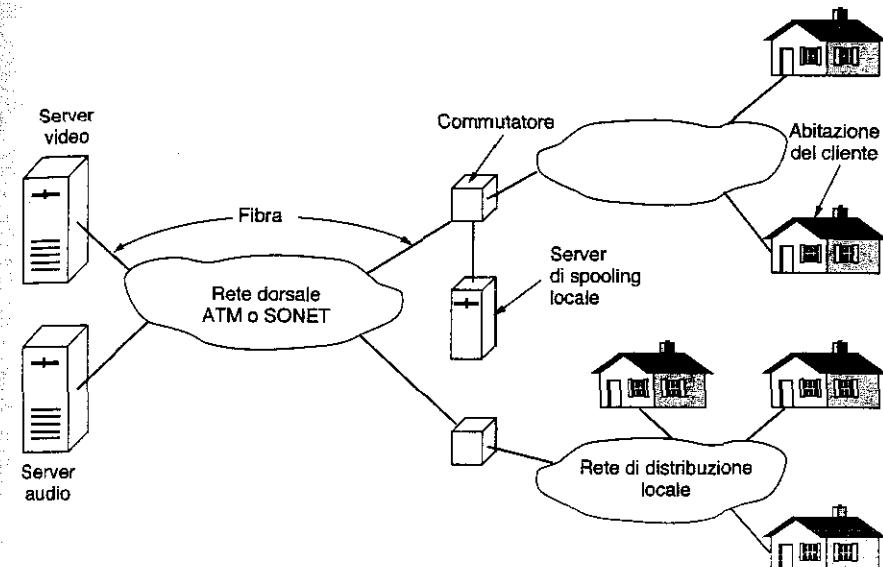


Figura 7.78. Una panoramica di un sistema per il video su richiesta.

Molti fornitori di informazioni sono collegati alla dorsale, mediante fibre ottiche a banda larga. Alcuni offrono video pay-per-view o CD audio pay-per-hear; altri offrono servizi specializzati, come lo shopping da casa (che permette agli utenti di ruotare una zuppiera, di eseguire uno zoom sull'elenco degli ingredienti, o di osservare un clip video su come guidare un tosaerba a gasolio). Senza dubbio, saranno rapidamente disponibili sport, notiziari, repliche di film, accesso WWW e molte altre possibilità.

Il sistema comprende anche i server di spooling locali, che permettono di collocare (preventivamente) i video vicino agli utenti, per risparmiare banda nelle ore di punta. Il modo in cui le parti del sistema sono collegate tra di loro, e la definizione di chi possiede cosa, sono oggetto di numerosi dibattiti nell'industria del settore. Di seguito esaminiamo la struttura dei componenti principali del sistema: i server video e la rete di distribuzione.

I server video

Per disporre del video su richiesta, sono necessari **server video** in grado di archiviare ed emettere in output un elevato numero di filmati contemporaneamente. Il numero totale di film girati fino ad oggi è stimato intorno a 65.000 (Minoli, 1995). Quando viene compresso in MPEG-2, un film tipico occupa circa 4 GB di spazio, pertanto 65.000 film richiederebbero qualcosa come 260 terabyte. A questi si aggiungono i vecchi programmi televisivi, i programmi sportivi, i telegiornali, le pubblicità e così via: è chiaro che siamo di fronte a un importante problema di memorizzazione.

Il modo più economico di memorizzare grandi volumi di informazioni prevede l'utilizzo di nastri magnetici. Probabilmente questo tipo di supporto sarà utilizzato ancora per molto tempo. Un nastro Ultrium può memorizzare 200 GB (50 filmati) a un costo pari a 1 o 2 euro per filmato. In commercio sono disponibili grandi server meccanici che gestiscono migliaia di nastri e dispongono di un braccio per prelevare i nastri e inserirli nelle unità. Il problema di questi sistemi è costituito dal tempo di accesso (specialmente per il cinquantesimo filmato sul nastro), dalla velocità di trasferimento e dal numero limitato di unità a nastro (per servire n filmati alla volta, il sistema necessita di n unità).

Fortunatamente, l'esperienza di videoteche, biblioteche e altre organizzazioni simili mostra che non tutti gli elementi sono richiesti con la stessa frequenza. Sperimentalmente, quando sono disponibili N filmati, la frazione di tutte le richieste per il filmato che occupa la posizione k nella graduatoria di popolarità equivale approssimativamente a C/k . C viene elaborato in modo da normalizzare la somma a 1:

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + \dots + 1/N)$$

Di conseguenza, il filmato più popolare è sette volte più popolare del settimo filmato. Questo risultato è noto come **legge di Zipf** (Zipf, 1949).

Il fatto che alcuni filmati siano più popolari di altri suggerisce una possibile soluzione sotto forma di una gerarchia di archiviazione, come mostrato nella Figura 7.79. Le prestazioni aumentano mentre ci si sposta verso l'alto nella gerarchia.

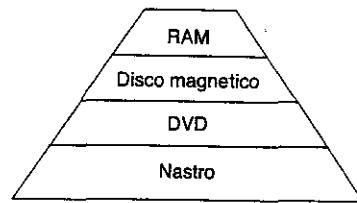


Figura 7.79. Una gerarchia di archiviazione per un server video.

Un'alternativa al nastro è l'archiviazione ottica. Gli attuali DVD contengono 4,7 GB, perfetti per un film, ma la successiva generazione ne conterrà due. Anche se i tempi di ricerca sono limitati in confronto ai dischi magnetici (50 msec contro 5 msec), il loro basso

costo e l'alta affidabilità rendono i juke-box ottici contenenti centinaia di DVD una valida alternativa al nastro per i filmati più utilizzati.

Il passo successivo sono i dischi magnetici. Presentano tempi di accesso ridotti (5 msec), velocità di trasferimento elevate (320 MB/sec per SCSI 320) e capacità importanti (più di 100 GB), pertanto sono adatti a conservare filmati in corso di trasmissione (a differenza di quelli archiviati nel caso qualcuno li desideri). Lo svantaggio principale è l'alto costo per l'archiviazione di filmati cui si accede raramente.

In cima alla piramide della Figura 7.79 si trova la RAM. La RAM è il supporto di archiviazione più veloce, ma anche il più costoso. Quando il prezzo della RAM scenderà intorno a 50 euro per GB, un filmato di 4 GB occuperà 200 euro di RAM; pertanto, per archiviare 100 filmati nella RAM sarà necessaria memoria per 20.000 euro. Tuttavia, per un server video che offre 100 filmati, la memorizzazione di tutti i dati nella RAM sta iniziando a sembrare fattibile. Se il server video ha 100 clienti che, collettivamente, guardano solo 20 filmati diversi, la soluzione non è più solo fattibile ma valida.

Dal momento che un server video è in pratica un dispositivo di I/O in tempo reale, necessita di un'architettura hardware e software diversa da un PC o una workstation UNIX. L'architettura hardware di un tipico server video è mostrata nella Figura 7.80. Il server ha una o più CPU ad alte prestazioni, ognuna con una memoria locale, una memoria principale condivisa, una grande cache RAM per i filmati più richiesti, diverse periferiche di archiviazione per i filmati e alcuni componenti hardware di rete: normalmente un'interfaccia ottica a una dorsale SONET o ATM a OC-12. Questi sottosistemi sono connessi da un bus ad alta velocità (almeno 1 GB/sec).

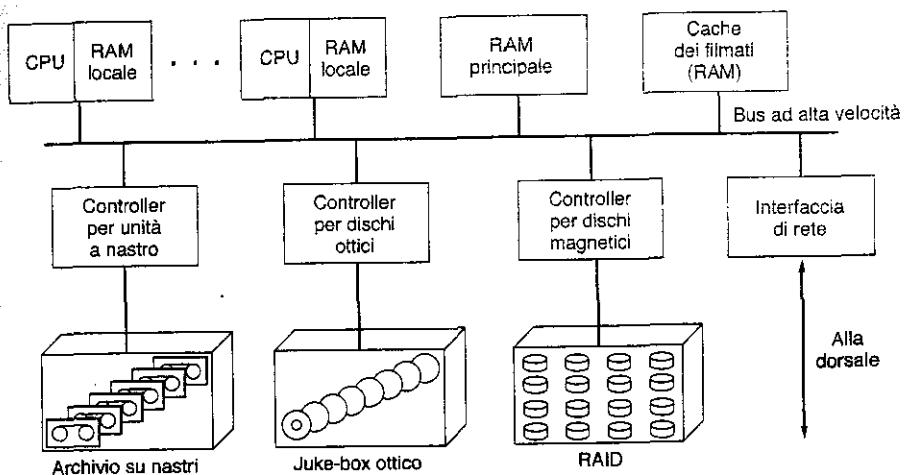


Figura 7.80. L'architettura hardware di un tipico server video.

Diamo ora un'occhiata al software del server video. Le CPU sono utilizzate per accettare le richieste degli utenti, individuare i filmati, spostare i dati tra le periferiche, emettere le fatture per i clienti e molte altre funzioni. Per alcune il tempo non è un fattore critico, al contrario di molte altre: ecco perché alcune (o magari tutte) le CPU devono eseguire un sistema operativo real-time, per esempio un microkernel real-time. Questi sistemi normalmente suddividono il lavoro in attività più piccole, ognuna con una scadenza nota, perciò lo scheduler può eseguire un algoritmo del tipo nearest deadline next o rate monotonic (Liu e Layland, 1973).

Il software della CPU definisce anche la natura dell'interfaccia che il server presenta ai client (server di spooling e set-top box). Si usano solitamente due architetture. La prima è un tradizionale file system, in cui i clienti possono aprire, leggere, scrivere e chiudere i file. A parte le complicazioni introdotte dalla gerarchia di archiviazione e dalle considerazioni sulla modalità real-time, il server può avere un file system modellato su quello di UNIX.

Il secondo tipo di interfaccia è basato sul modello del videoregistratore. I comandi per il server richiedono di aprire, riprodurre, mettere in pausa, avvolgere e riavvolgere i file. La differenza con il modello UNIX sta nel fatto che, una volta imparito un comando *PLAY*, il server continua a inviare dati a una velocità costante, senza la necessità di nuovi comandi.

Il cuore del software del server video è la parte che gestisce il disco. Possiede due compiti principali: porre i filmati sul disco magnetico quando devono essere estratti dall'archivio ottico o dal nastro, e gestire le richieste per molti flussi di output. Il posizionamento dei filmati è importante perché può influire notevolmente sulle prestazioni.

Due dei modi per organizzare l'archiviazione su disco sono le disk farm e i disk array. In una **disk farm**, ogni unità a disco contiene alcuni filmati interi. Per motivi di prestazioni e affidabilità, ogni filmato dovrebbe essere presente su almeno due unità. L'altra organizzazione per l'archivio è il **disk array o RAID** (*Redundant Array of Inexpensive Disks*), in cui ogni filmato è diviso su più unità. Per esempio, il blocco 0 è sull'unità 0, il blocco 1 sull'unità 1 e così via, con il blocco $n-1$ sull'unità $n-1$; il ciclo viene ripetuto, con il blocco n sull'unità 0 e così via. Questa organizzazione è chiamata **striping**.

Un disk array con striping presenta diversi vantaggi rispetto alla disk farm. Innanzitutto, le n unità possono lavorare in parallelo, migliorando le prestazioni di un fattore n . In secondo luogo, può essere reso ridondante con l'aggiunta di un'altra unità a ogni gruppo di n : l'unità ridondante contiene l'OR esclusivo blocco per blocco delle altre unità, per consentire un pieno recupero dei dati in caso di errore di un'unità. Infine, risolve il problema del bilanciamento del carico, poiché non è necessario il posizionamento manuale per evitare che tutti i filmati più richiesti si trovino sulla stessa unità. D'altra parte, l'organizzazione del disk array è più complicata della disk farm ed è sensibile ai guasti multipli. È anche poco adatta alle operazioni del tipo videoregistratore, come il riavvolgimento o l'avvolgimento rapido di un filmato.

L'altro compito del software che gestisce il disco è servire tutti i flussi di output in tempo reale, soddisfacendo i vincoli di temporizzazione. Solo pochi anni fa ciò richiedeva complessi algoritmi di pianificazione dei dischi, ma grazie alla diminuzione dei

prezzi della memoria oggi è possibile seguire un approccio più semplice. Per ogni flusso da servire, nella RAM viene conservato un buffer di circa 10 secondi di video (5 MB). Il buffer è riempito da un processo del disco e svuotato da un processo di rete. Con 500 MB di RAM, è possibile alimentare 100 flussi direttamente dalla RAM. Naturalmente, il sottosistema a dischi deve avere una velocità dei dati sostenuta di 50 MB/sec per tenere pieno il buffer, ma un RAID costruito da dischi SCSI high-end può gestire tranquillamente questo requisito.

La rete di distribuzione

La rete di distribuzione è l'insieme di switch e linee tra l'origine e la destinazione. Come è stato visto nella Figura 7.78, consiste di una dorsale connessa a una rete di distribuzione locale. Solitamente la dorsale è commutata mentre la rete di distribuzione locale non lo è.

Il requisito principale imposto sulla dorsale è la banda elevata. Un tempo anche il jitter ridotto era un requisito, ma ora che persino i PC più piccoli sono in grado di archiviare 10 secondi di video MPEG-2 di alta qualità, questa non è più un requisito importante.

La distribuzione locale è molto caotica: diverse società sperimentano reti differenti per ogni regione. Le compagnie telefoniche, le società di TV via cavo e le new entry (come i fornitori di energia elettrica) sono convinte che chi arriva per primo sarà il vincitore. Di conseguenza, possiamo osservare una proliferazione di tecnologie installate. In Giappone, alcune società che gestiscono le reti fognarie si sono lanciate nel business di Internet, partendo dal presupposto che dispongono più ampio condotto che entra nelle abitazioni (fanno scorrere una fibra ottica al suo interno, ma devono prestare attenzione a dove emerge il cavo). I quattro schemi di distribuzione locale per il video su richiesta utilizzano gli acronimi ADSL, FTTC, FTTH e HFC; esaminiamoli uno alla volta.

ADSL è la debuttante dell'industria telefonica nella corsa alla distribuzione locale. Abbiamo studiato ADSL nel capitolo 2 e quindi non ci ripeteremo. L'idea è che quasi tutte le abitazioni negli Stati Uniti, in Europa e in Giappone dispongono già di un doppino in rame per il servizio telefonico analogico. Se questi cavi potessero essere utilizzati per il video su richiesta, le società telefoniche potrebbero vincere.

Il problema, naturalmente, sta nel fatto che questi cavi non possono supportare MPEG-1 sulla loro tipica lunghezza di 10 Km, figurarsi MPEG-2. Il video a colori ad alta risoluzione richiede 4-8 Mbps, a seconda della qualità desiderata. ADSL non è abbastanza veloce, eccetto che per i collegamenti locali più brevi.

La seconda proposta delle aziende telefoniche è **FTTC** (*Fiber To The Curb*). In FTTC, la compagnia telefonica stende una fibra ottica dalla centrale a ogni complesso residenziale delle vicinanze, che termina in un dispositivo chiamato **ONU** (*Optical Network Unit*). Ad un'ONU possono fare capo circa 16 collegamenti locali in rame, così brevi che su di essi è possibile utilizzare T1 o T2 full duplex, consentendo la trasmissione di filmati MPEG-1 e MPEG-2. Inoltre, FTTC è simmetrico e perciò supporta le videoconferenze tra chi lavora a casa e le piccole imprese.

per tutte queste reti di distribuzione locale, è possibile che ogni quattro si la equipaggia-
to di uno o più server di spooling, che sono versioni ridotte dei server video discusse in
precedenza. Il grande vantaggio di questi server locali è che spesso non parte del carico
ogni abitazione, e viene chiamata **FTTB** (*Fiber To The Home*). In questo schema tutti dis-
pongono di una portante OC-1, OC-3 o per uno superiore, se fosse necessaria. FTTB è
molto costoso e non sarà utilizzato per molti anni ancora, ma chiaramente apre un nuovo
scenario. Nella Figura 7.63 abbiamo visto come chiunque possa creare la sua stazione
radio. Che cosa ne pensate del fatto che ogni membro della famiglia potrebbe gestire la
sua stazione televisiva personale? ADSL, FTTC e FTTB sono tutte reti di distribuzione
locali punto a punto: non bisogna sorprendersi, visto che questa è l'attuale organizzazione
ne del sistema telefonico.

Un apprezzabile connettore diverso è **FTC** (*Hybrid Fiber Coax*), la soluzione attual-
mente prescelta da molti operatori di TV via cavo, illustrata nella Figura 2.47(a). Già attuali cavi
coassiali a 300-450 MHz sono stati sostituiti da cavi coassiali a 750 MHz, aggiornando la
capacità da 50-75 canali per la trasmissione analogica.

I 50 nuovi canali saranno modulati con QAM-256, che fornisce circa 40 Mbps per cana-
le, offrendo un totale di 2 Gbps di nuova banda. L'headend sarà spostato nelle vicinanze,
che a ogni abitazione può essere allocato un canale dedicato di 4 Mbps, che può gestire un
fotogramma estensibile a paragonabile alle operazioni televisive, la
grandeza di nuova infrastruttura è in breve, sostituire l'intero sistema. Di conseguenza,
gli amplificatori multirazionali: in breve, sostituire l'intero sistema. Di conseguenza, la
cablaggio estensibile a 750 MHz, instaurare nuovi headend e rimuovere tutti
i canali se sembra fantastico, ciò richiede ai fornitori della TV via cavo di sostituire tutto il
fotogramma mediana Internet, Mbone viene utilizzata per la visualizzazione di filmati precomprese
nichiesa, dove l'enfasi è posta sulla richiesta e visualizzazione di filmati precomprese
Mbone può essere pensato come una televisione via Internet. A differenza del video su
panoramica del suo funzionamento.

Anche se sembra fantastico, ciò richiede ai fornitori della TV via cavo di sostituire tutto il
cablaggio estensibile a 750 MHz, instaurare nuovi headend e rimuovere tutti i
fotogramma mediana Internet, Mbone viene utilizzata per la visualizzazione di filmati precomprese
nichiesa, dove l'enfasi è posta sulla richiesta e visualizzazione di filmati precomprese
Mbone può essere pensato come una televisione via Internet. A differenza del video su
panoramica del suo funzionamento.

Tecnicamente, queste due sistemi non sono molto diversi da come li fanno appari-
re i concetti di dati Rolling Stones è per alcuni filmati relativi al festival del film di Cannes; si
fotogramma mediana Internet, Mbone è stata utilizzata anche per un
se molte conferenze scientifiche, specialmente le riunioni di IETF, nonché eventi scientifici
fotogramma mediana Internet, Mbone viene utilizzata per la trasmissione di video in
archiviazioni su un server, Mbone viene utilizzata per la trasmissione di filmati precomprese
nichiesa, dove l'enfasi è posta sulla richiesta e visualizzazione di filmati precomprese
Mbone può essere pensato come una televisione via Internet. A differenza del video su
panoramica del suo funzionamento.

Ciò nonostante, esiste una reale differenza che vale la pena sottolineare. HFC utilizza un
supporto condizioso senza switch e router. Qualiasi informazione posta sul cavo può esse-
re rimessa da qualsiasi abbonato senza ulteriori preoccupazioni. FTC, che è completa-
mente comunitario, non ha questa proprietà. Di conseguenza, gli operatori di HFC deside-
rano che i servizi possano vedere, gli operatori di FTC non desiderano la crittografia,
perché agevolano complessità, riduce le prestazioni e non fornisce una protezione agguantu-
re del loro sistema. Dal punto di vista della società che esegue il servizio, è una buona
idea di meno utilizzare la crittografia? Un server gestito da una compagnia telefonica o da
una delle società che non hanno paghi
maido l'efficienza come motivo ma provocando perdite economiche ai suoi rivali HFC.

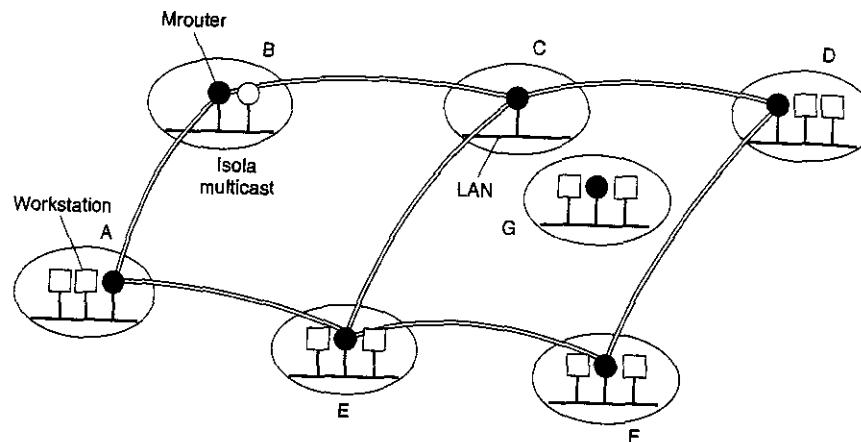


Figura 7.81. MBone consiste di isole multicast connesse da tunnel.

I tunnel sono configurati manualmente. Solitamente un tunnel segue il percorso di una connessione fisica, ma non si tratta di un requisito. Se a causa di un guasto il percorso fisico sottostante a un tunnel s'interrompe, gli mrouter che utilizzano il tunnel non lo noteranno nemmeno, visto che Internet instrada automaticamente su altre linee il traffico IP che scorre tra essi.

Quando appare una nuova isola che desidera unirsi a MBone, come *G* nella Figura 7.81, il suo amministratore invia un messaggio che ne annuncia l'esistenza alla mailing list di MBone. Gli amministratori dei siti vicini lo contattano poi per organizzare l'impostazione dei tunnel. A volte i tunnel esistenti sono rimescolati per ottimizzare la topologia, traendo vantaggio dalle nuove isole. Dopo tutto, i tunnel non hanno un'esistenza fisica ma sono definiti da tabelle nei mrouter e possono essere aggiunti, eliminati o spostati semplicemente modificando le tabelle. Generalmente, ogni nazione su MBone utilizza una dorsale, a cui sono collegate le isole regionali. Di norma, MBone è configurato con uno o due tunnel che attraversano gli oceani Atlantico e Pacifico, rendendolo globale.

In ogni istante MBone utilizza una topologia specifica composta di isole e tunnel, indipendente dal numero di indirizzi multicast attualmente in uso e da chi li sta ascoltando o osservando. La situazione è molto simile a una normale sottorete (fisica), pertanto possono essere applicati i normali algoritmi di routing. Per questo motivo MBone usava inizialmente un algoritmo di routing, **DVMRP** (*Distance Vector Multicast Routing Protocol*), basato sull'algoritmo per i vettori di distanza Bellman-Ford. Per esempio, nella Figura 7.81, l'isola *C* può eseguire l'instradamento ad *A* sia tramite *B* sia tramite *E* (o volendo anche tramite *D*). La scelta viene compiuta prendendo i valori che questi nodi forniscono per le rispettive distanze da *A* e quindi sommando ad essi la propria distanza; in questo modo ogni isola determina la via migliore verso ogni altra. Nella realtà i percorsi non sono però gestiti in questo modo, come vedremo tra poco.

Ora esaminiamo come avviene effettivamente il multicasting. Per il multicast di un programma audio o video, un'origine deve acquisire un indirizzo multicast di classe D, che agisce come una frequenza o un numero di canale. Gli indirizzi di classe D vengono riservati da un programma che esegue ricerche in un database di indirizzi multicast liberi. In ogni istante possono avvenire molti multicast, quindi un host può sintonizzarsi su quello a cui è interessato ascoltando l'indirizzo multicast appropriato.

Periodicamente, ogni mrouter invia un pacchetto di broadcast IGMP limitato alla sua isola, chiedendo chi è interessato a quel canale. Gli host che desiderano (continuare a) ricevere uno o più canali inviano un altro pacchetto IGMP come risposta. Queste risposte sono sfalsate nel tempo, per evitare di sovraccaricare la LAN locale. Ogni mrouter conserva una tabella dei canali che deve porre sulla sua LAN, per non sprecare banda con attività di multicast dei canali che nessuno desidera.

Il multicast si propaga tramite MBone come segue. Quando una fonte audio o video genera un nuovo pacchetto, ne esegue il multicasting sull'isola locale utilizzando la relativa infrastruttura hardware. Il pacchetto è prelevato dal mrouter locale, che lo copia in tutti i tunnel a cui è connesso.

Ogni mrouter che ottiene questo pacchetto tramite un tunnel lo controlla per vedere se viene dal percorso migliore, vale a dire da quello che la tabella indica di utilizzare per raggiungere la fonte (come se fosse una destinazione). Se il pacchetto proviene dalla via migliore lo mrouter copia il pacchetto in tutti gli altri tunnel, in caso contrario viene scartato. Per esempio, nella Figura 7.81, se le tabelle di *C* affermano che occorre utilizzare *B* per raggiungere *A*, quando un pacchetto multicast di *A* raggiunge *C* tramite *B*, il pacchetto viene copiato in *D* ed *E*. Tuttavia, quando un pacchetto multicast da *A* raggiunge *C* tramite *E* (quindi non dal percorso migliore), viene semplicemente scartato. Questo algoritmo equivale all'algoritmo di inoltro dal percorso inverso (reverse path forwarding) presentato nel capitolo 5. Anche se non è perfetto, è abbastanza valido e molto semplice da implementare.

Per prevenire il flooding di Internet, oltre al reverse path forwarding viene utilizzato il campo *time to live* di IP per limitare l'ambito del multicasting. Ogni pacchetto inizia con qualche valore (determinato dall'origine), e a ogni tunnel è assegnato un peso. Un pacchetto viene trasferito tramite un tunnel solo se dispone di un peso sufficiente, altrimenti viene scartato. Per esempio, i tunnel transoceanei sono di norma configurati con un peso di 128, pertanto i pacchetti possono essere limitati al continente di origine assegnando una durata minore o uguale a 127. Dopo il passaggio nel tunnel, il campo *time to live* (vita utile) viene decrementato del peso del tunnel.

Anche se l'algoritmo di routing di MBone funziona, sono in corso numerose ricerche per migliorarlo. Una proposta conserva l'idea di routing con vettore di distanza, ma rende l'algoritmo gerarchico raggruppando i siti MBone in regioni ed eseguendo il primo instradamento verso essi (Thyagarajan e Deering, 1995).

Un'altra proposta prevede di utilizzare una forma modificata di routing basato sullo stato del collegamento, anziché sul vettore della distanza. In particolare, un gruppo di lavoro di

IETF ha modificato OSPF per adattarlo al multicasting all'interno di un singolo sistema autonomo. L'OSPF multicast risultante è chiamato **MOSPF** (Moy, 1994). Le modifiche fanno in modo che l'intera mappa costruita da MOSPF tenga traccia delle isole e dei tunnel multicast, oltre che delle solite informazioni di routing. Armati della topologia completa, è facile elaborare il percorso migliore da un'isola all'altra utilizzando i tunnel. È possibile utilizzare l'algoritmo di Dijkstra, per esempio.

Una seconda area di ricerca riguarda il routing inter-AS. È stato sviluppato un algoritmo chiamato **PIM** (*Protocol Independent Multicast*), da un altro gruppo di lavoro di IETF. PIM è fornito in due versioni, a seconda della densità delle isole (sono dense quando chiunque desidera osservare, sparse se invece ci sono pochi spettatori). Entrambe le versioni utilizzano le tabelle di routing unicast standard, anziché creare una topologia sovrapposta come DVMRP e MOSPF.

In PIM-DM (*Dense Mode*) l'idea è di sfoltire i percorsi inutili. Lo sfoltimento funziona come segue. Quando un pacchetto multicast giunge attraverso il tunnel "sbagliato", viene inviato un pacchetto di sfoltimento tramite lo stesso tunnel, che comunica al mittente di interrompere l'invio dei pacchetti dall'origine in questione. Quando un pacchetto giunge tramite il tunnel "giusto", viene copiato in tutti gli altri tunnel che non sono stati sfoltiti in precedenza. Se tutti gli altri tunnel si sono sfoltiti e il canale non ha alcun interesse all'interno dell'isola locale, lo mrouter invia un messaggio di sfoltimento tramite il canale "corretto". In questo modo, il multicast si adatta automaticamente e procede solo dove è desiderato.

PIM-SM (*Spare Mode*), descritto in RFC 2362, funziona in modo differente. L'idea è quella d'impedire la saturazione di Internet se tre persone a Berkeley vogliono fare una chiamata in conferenza su un indirizzo di classe D. PIM-SM funziona impostando punti di rendez-vous. Ogni origine del gruppo multicast PIM-SM invia i suoi pacchetti ai punti di rendez-vous, e qualsiasi sito interessato ad unirsi chiede a uno dei punti di rendez-vous di impostare un tunnel verso di lui. In questo modo, tutto il traffico PIM-SM viene trasportato da unicast, anziché multicast. PIM-SM sta diventando popolare e MBone sta migrando verso il suo utilizzo. Quando PIM-SM sarà utilizzato in modo diffuso, MOSPF inizierà gradualmente a scomparire. D'altra parte, MBone stesso sembra essere stagnante e probabilmente non avrà mai un enorme successo.

Ciò nonostante, il networking multimediale è un campo eccitante e in rapido movimento. Ogni giorno vengono annunciate nuove tecnologie e applicazioni. Il multicasting e la qualità del servizio procedono di pari passo, come spiegato in (Striegel e Manimaran, 2002). Un altro argomento caldo è il multicast wireless (Gossain et al., 2002). L'intera area del multicasting e di tutto ciò che vi è legato rimarrà importante per gli anni a venire.

7.5 Sommario

L'assegnazione dei nomi in Internet utilizza uno schema gerarchico chiamato sistema dei nomi di dominio (DNS). Al primo livello si trovano i domini generici ben noti, tra cui figurano *com*, *edu*, e altri 200 domini nazionali. DNS è implementato come un sistema di database distribuito con server in tutto il mondo. DNS mantiene i record con gli indirizzi IP,

quelli di mail exchange e altre informazioni. Interrogando un server DNS, un processo può associare un nome di dominio Internet all'indirizzo IP utilizzato per comunicare con il dominio.

La posta elettronica è una delle applicazioni fondamentali per Internet. Ormai la utilizzano tutti, dai bambini ai nonni. La maggior parte dei sistemi e-mail nel mondo utilizza il sistema di posta elettronica definito in RFC 2821 e 2822. I messaggi inviati in accordo a questo standard utilizzano le intestazioni ASCII di sistema per definire le proprietà del messaggio. Usando MIME si possono inviare molti tipi di contenuto. I messaggi sono inviati con SMTP, che lavora con una connessione TCP dall'host di origine a quello di destinazione, consegnando direttamente la posta elettronica sulla connessione TCP.

L'altra applicazione fondamentale di Internet è il World Wide Web. Il Web è un sistema per il collegamento di documenti ipertestuali. In origine, ogni documento era una pagina scritta in HTML, con collegamenti ipertestuali ad altri documenti. Oggi XML sta gradualmente iniziando a prendere il posto di HTML, e una grande quantità di contenuto è generata dinamicamente utilizzando script sul lato server (PHP, JSP e ASP) nonché script sul lato client (in particolare JavaScript). Un browser può visualizzare un documento stabilendo una connessione TCP al suo server, chiedendo il documento e poi chiudendo la connessione. I messaggi di richiesta contengono diverse intestazioni per fornire informazioni aggiuntive. Il caching, la replica e le reti di consegna del contenuto sono largamente impiegate per migliorare le prestazioni del Web.

Il Web wireless è ai suoi esordi. I primi sistemi a disposizione sono WAP e i-mode, con schermi piccoli e banda limitata, ma la successiva generazione sarà più potente.

Multimedia è un'altra stella nascente nel firmamento delle reti. Consente la digitalizzazione e il trasporto elettronico di audio e video. L'audio richiede meno banda, pertanto è molto più evoluto. L'audio in streaming, le radio Internet e Voice over IP sono già una realtà; nuove applicazioni vengono prodotte ogni giorno. Il video su richiesta è un'area promettente verso cui c'è molto interesse. Per finire, MBone è un servizio sperimentale per una televisione digitale mondiale trasmessa in tempo reale su Internet.

Problemi

1. Molti computer aziendali dispongono di tre distinti identificatori univoci mondiali. Quali sono?
2. Secondo le informazioni della Figura 7.3, *little-sister.cs.vu.nl* si trova in una rete di classe A, B o C?
3. Nella Figura 7.3 non è presente il punto dopo *rowboat*? Perché no?
4. Provare a indovinare il significato dello smiley :-X (a volte scritto come :-#).
5. DNS utilizza UDP al posto di TCP. Se viene perso un pacchetto DNS, non esiste il recupero automatico. Questo provoca un problema? Se sì, come viene risolto?
6. Oltre a essere soggetti a perdite, i pacchetti UDP hanno una lunghezza massima che in alcuni casi non supera i 576 byte. Che cosa accade se un nome DNS da ricercare supera questa lunghezza? Può essere inviato in due pacchetti?
7. Una macchina con un singolo nome DNS può utilizzare più indirizzi IP? Come può succedere?

8. Un computer può avere due nomi DNS che ricadono in domini di primo livello diversi? Se sì, indicare un esempio plausibile; se no, spiegarne il motivo.
9. Il numero di società con un sito Web è notevolmente aumentato negli ultimi anni. Come risultato, centinaia di società si sono registrate con il dominio *com*, provocando un elevato carico sul server di primo livello per questo dominio. Suggerire un modo per alleviare il problema senza cambiare lo schema di denominazione (vale a dire senza introdurre nuovi nomi di dominio di primo livello). Sono ammesse soluzioni che richiedono modifiche al codice del client.
10. Alcuni sistemi di posta elettronica supportano un campo di intestazione *Content Return*: che specifica se il corpo di un messaggio deve essere restituito in caso di mancata consegna. Questo campo appartiene all'involucro o all'intestazione?
11. I sistemi di posta elettronica hanno bisogno delle directory dove ricercare gli indirizzi di posta elettronica delle persone. Per costruire queste directory, occorre dividere i nomi in componenti standard (per esempio nome e cognome) per rendere possibile la ricerca. Discutere alcuni problemi che devono essere risolti affinché uno standard mondiale sia accettabile.
12. L'indirizzo di posta elettronica di una persona corrisponde al suo nome di login seguito da @ e dal nome di un dominio DNS con un record *MX*. I nomi di login possono essere nomi di battesimo, cognomi, iniziali e ogni altro genere di nome. Si supponga che una grande società stabilisca che troppi messaggi di posta elettronica vengono persi perché le persone non conoscono il nome di login del destinatario. Esiste un modo per risolvere il problema senza cambiare DNS? Se sì, formulare una proposta e spiegarne il funzionamento; se no, spiegare perché non è possibile.
13. Un file binario è lungo 3.072 byte. Quanto sarà lungo se viene codificato con la codifica base64, con una coppia CR+LF inserita ogni 80 byte inviati e alla fine?
14. Si consideri lo schema di codifica MIME quoted-printable. Indicare un problema non discusso nel testo e proporre una soluzione.
15. Indicare cinque tipi MIME non elencati nel libro. È possibile utilizzare il browser e Internet per cercare informazioni.
16. Si supponga di voler inviare un file MP3 a un amico, ma la dimensione del file MP3 è 4 MB e l'ISP dell'amico limita la quantità di posta in arrivo a 1 MB. Esiste un modo per gestire questa situazione utilizzando RFC 822 e MIME?
17. Si supponga che qualcuno imposti un daemon per le assenze e invii un messaggio appena prima di disconnettersi. Sfortunatamente, il destinatario è in vacanza per una settimana e dispone anch'egli di un daemon per le assenze. Cosa accade dopo? Le risposte automatiche proseguiranno avanti e indietro fino al ritorno di uno dei due?
18. In qualsiasi standard, come RFC 822, è necessaria una grammatica precisa delle operazioni consentite, in modo che diverse implementazioni possano interoperate. Anche gli elementi semplici devono essere definiti attentamente. Le intestazioni SMTP consentono gli spazi bianchi (white space) tra i token. Elenicare *due* definizioni plausibili alternative dello spazio bianco tra i token.

19. Il daemon per le assenze è parte dell'agente utente o dell'agente di trasferimento dei messaggi? Naturalmente si configura utilizzando l'agente utente, ma è quest'ultimo a inviare realmente le risposte? Spiegare la risposta.
20. POP3 consente agli utenti di prelevare e scaricare la posta elettronica da una casella di posta remota. Questo significa che, per fare in modo che qualsiasi programma POP3 sul lato client possa leggere la casella di posta su qualsiasi mail server, il formato interno delle caselle di posta deve essere standardizzato? Spiegare la risposta.
21. Dal punto di vista di un ISP, POP3 e IMAP differiscono per un aspetto importante. Gli utenti di POP3 generalmente svuotano le caselle di posta ogni giorno; gli utenti di IMAP conservano la loro posta sul server all'infinito. Si supponga di essere chiamati per consigliare a un ISP quale protocollo dovrebbe supportare. Quali considerazioni sarebbero opportune?
22. La Webmail utilizza POP3, IMAP o altri protocolli? Perché è stata fatta questa scelta? Se non vengono usati POP3 o IMAP, qual è quello più vicino come teoria?
23. Quando le pagine Web vengono inviate, sono precedute dalle intestazioni MIME. Perché?
24. Quando sono necessari i visualizzatori esterni? Come fa un browser a sapere quale deve utilizzare?
25. È possibile che, quando un utente fa clic su un collegamento in Netscape venga avviato un particolare helper, ma facendo clic sullo stesso collegamento in Internet Explorer venga avviato un helper completamente diverso, anche se il tipo MIME restituito in entrambi i casi è identico? Spiegare la risposta.
26. Un server Web multithreaded è organizzato come mostrato nella Figura 7.21. Richiede 500 μ sec per accettare una richiesta e controllare la cache. Nella metà dei casi il file viene trovato nella cache e restituito immediatamente; nell'altra metà il modulo si deve bloccare per 9 msec mentre la richiesta del disco viene accodata ed elaborata. Quanti moduli dovrebbe avere il server per mantenere la CPU sempre occupata (presumendo che il disco non sia un collo di bottiglia)?
27. L'URL standard *http* presume che il server Web sia in ascolto sulla porta 80. Tuttavia, è possibile che un server Web esegua l'ascolto su un'altra porta. Indicare una sintassi ragionevole per un URL che accede a un file su una porta non standard.
28. Anche se non è stata menzionata nel testo, una forma alternativa per gli URL prevede di utilizzare l'indirizzo IP al posto del nome DNS. Un esempio di utilizzo di un indirizzo IP è *http://192.31.231.66/index.html*. Come fa il browser a sapere se il nome che segue lo schema è un nome DNS o un indirizzo IP?
29. Si supponga che un membro del reparto di scienze informatiche di Stanford abbia scritto un nuovo programma che desidera distribuire con FTP. Inserisce pertanto il file nella directory *FTP/ftp/pub/freebies/newprog.c*. Quale potrebbe essere l'URL completo per questo programma?
30. Nella Figura 7.25, *www.aportal.com* tiene traccia delle preferenze dell'utente in un cookie. Uno svantaggio di questo schema deriva dal fatto che i cookie sono limitati a 4 KB, pertanto se le preferenze sono numerose (per esempio molte azioni, squadre sportive, tipi di notizie, previsioni meteo per diverse città, offerte speciali per molte categorie di prodotti e così via) il limite di 4 KB può essere raggiunto. Progettare un metodo alternativo per tenere traccia delle preferenze, che non presenti questo problema.

31. Sloth Bank desidera facilitare il banking online per i suoi clienti pigri: dopo che un utente ha eseguito l'accesso ed è autenticato da una password, la banca restituisce un cookie contenente il numero ID del cliente. In questo modo, il cliente non deve identificare sé stesso o digitare una password nelle future visite alla banca online. Come viene giudicata questa idea? Funzionerà? È una buona idea?
32. Nella Figura 7.26, il parametro *ALT* viene impostato nel tag ``. A quali condizioni sarà utilizzato dal browser, e come?
33. Come è possibile rendere selezionabile un'immagine in HTML, in modo che risponda a un clic fatto su di essa? Mostrare un esempio.
34. Creare il tag `<a>` necessario per trasformare la stringa "ACM" in un collegamento ipertestuale a <http://www.acm.org>.
35. Progettare un modulo per una nuova società, Interburger, che consente di ordinare hamburger via Internet. Il modulo dovrebbe comprendere il nome del cliente, l'indirizzo e la città, nonché una scelta per la dimensione (gigante o immenso) e un'opzione per il formaggio. Gli hamburger saranno pagati in contanti alla consegna, per cui non sono necessarie informazioni sulla carta di credito.
36. Progettare un modulo che richieda all'utente di digitare due numeri. Quando l'utente fa clic sul pulsante Invia, il server restituisce la loro somma. Scrivere il codice sul lato server come script PHP.
37. Per ognuna delle seguenti applicazioni, spiegare se sarebbe (1) possibile e (2) preferibile utilizzare uno script PHP o JavaScript, e perché.
- Visualizzare un calendario per qualsiasi mese richiesto, a partire dal settembre 1752.
 - Visualizzare l'orario dei voli da Amsterdam a New York.
 - Tracciare il grafico di un polinomio con i coefficienti forniti dall'utente.
38. Scrivere un programma in JavaScript che accetta un intero maggiore di 2 e indica se si tratta di un numero primo. Occorre notare che JavaScript dispone di istruzioni if e while con la stessa sintassi di C e Java. L'operatore di modulo è `%`. Per calcolare la radice quadrata di x è possibile utilizzare `Math.sqrt(x)`.
39. La seguente è una pagina HTML:
- ```
<html> <body>
 Fare clic qui per informazioni
</body> </html>
```
- Se l'utente fa clic sul collegamento ipertestuale, viene aperta una connessione TCP e alcune righe vengono inviate al server. Elencare tutte le righe inviate.
40. L'intestazione *If-Modified-Since* può essere utilizzata per controllare se una pagina nella cache è ancora valida. Le richieste possono riguardare pagine contenenti immagini, suoni, video e altro ancora, oltre ad HTML. L'efficacia di questa tecnica per le immagini JPEG è maggiore o minore rispetto all'utilizzo per HTML? Pensare attentamente al significato di "efficacia" e spiegare la risposta.

41. Nel giorno in cui si tiene un importante evento sportivo, come la finale di un campionato di qualche sport, molte persone visitano il sito Web ufficiale. È possibile considerarlo un flash crowd come per l'elezione della Florida nel 2000? Perché (o perché no)?
42. È sensato che un singolo ISP lavori come una CDN? Se sì, come dovrebbe operare? Se no, che cosa c'è di sbagliato nell'idea?
43. In quali condizioni l'utilizzo di una CDN è una cattiva idea?
44. I terminali Web wireless hanno una banda ridotta, che rende importante una codifica efficace. Progettare uno schema per trasmettere il testo normale in modo efficiente su un collegamento wireless a un dispositivo WAP. È possibile ipotizzare che il terminale disponga di diversi megabyte di ROM e di una CPU moderatamente potente. Suggerimento: pensare a come si trasmette il giapponese, in cui ogni simbolo è una parola.
45. Un compact disc può contenere 650 MB di dati. Per i CD audio viene utilizzata la compressione? Spiegare il ragionamento.
46. Nella Figura 7.57(c) il rumore di quantizzazione si verifica a causa dell'utilizzo di campioni di 4 bit per rappresentare nove valori di segnale. Il primo campione, a 0, è esatto, ma gli altri no. Qual è l'errore percentuale per i campioni a 1/32, 2/32 e 3/32 del periodo?
47. Un modello psicoacustico può essere utilizzato per ridurre la banda necessaria per la telefonia Internet? Se sì, quali eventuali condizioni devono essere soddisfatte per il funzionamento? Se no, perché no?
48. Un server di streaming audio presenta una distanza unidirezionale di 50 msec con un lettore multimediale. L'output avviene a 1 Mbps. Se il lettore multimediale ha un buffer di 1 MB, che cosa si può dire sulla posizione del limite inferiore e del limite superiore?
49. L'algoritmo di interleaving della Figura 7.60 presenta il vantaggio di poter sopravvivere alla perdita occasionale di un pacchetto senza introdurre una lacuna nella riproduzione. Tuttavia, se viene utilizzato per la telefonia Internet, presenta anche un piccolo svantaggio. Quale?
50. Voice over IP presenta gli stessi problemi con i firewall dell'audio in streaming? Spiegare la risposta.
51. Qual è il bit-rate per la trasmissione di fotogrammi a colori non compressi di 800 x 600 pixel, con 8 bit per pixel e 40 fotogrammi al secondo?
52. Un errore di 1 bit in un fotogramma MPEG può influire su più fotogrammi, oltre a quello in cui si verifica? Spiegare la risposta.
53. Si consideri un server video per 100.000 clienti, dove ogni cliente guarda due filmati al mese. Metà dei filmati viene servita alle 20:00. Quanti filmati deve trasmettere contemporaneamente il server in questo periodo di tempo? Se ogni filmato richiede 4 Mbps, quante connessioni OC-12 sono necessarie?
54. Si supponga che valga la legge di Zipf per gli accessi a un server video con 10.000 filmati. Tenendo presente che il server conserva i 1.000 filmati più popolari su un disco magnetico e i rimanenti 9.000 su un disco ottico, scrivere un'espressione per la frazione delle richieste che sono soddisfatte dal disco magnetico. Scrivere un breve programma per valutare numericamente questa espressione.

55. Alcuni cybersquatter hanno registrato nomi di dominio corrispondenti ad errori ortografici di famosi siti aziendali, per esempio [www.microsoft.com](http://www.microsoft.com). Creare un elenco di almeno cinque domini di questo tipo.
56. Molte persone hanno registrato nomi DNS simili a [www.parola.com](http://www.parola.com), dove *parola* è una parola comune. Per le seguenti categorie, elencare cinque siti Web e riempilarne brevemente il contenuto (per esempio, [www.stomach.com](http://www.stomach.com) è un gastroenterologo di Long Island). Ecco l'elenco delle categorie: animali, cibi, soprammobili, parti del corpo. Per l'ultima categoria, limitarsi alle parti del corpo sopra la cintura.
57. Progettare alcuni emoji personali utilizzando bitmap 12 x 12. Rappresentare i termini fidanzata, fidanzato, professore e politico.
58. Scrivere un server POP3 che accetta i seguenti comandi: *USER*, *PASS*, *LIST*, *RETR*, *DELE* e *QUIT*.
59. Riscrivere il server della Figura 6.6 come un vero server Web utilizzando il comando *GET* di HTTP 1.1. Dovrebbe accettare anche il messaggio *Host*. Il server dovrebbe mantenere una cache di file recentemente prelevati dal disco e servire le richieste dalla cache, quando possibile.

## 8

## Sicurezza delle reti

Nei primi decenni della loro esistenza, le reti di computer vennero utilizzate prevalentemente dai ricercatori universitari per inviare e ricevere e-mail, e dalle aziende per condividere le stampanti. In quelle applicazioni non si prestava particolare attenzione alla sicurezza. Oggi milioni di persone usano le reti per fare acquisti, lavorare con la banca oppure completare la dichiarazione dei redditi, quindi la sicurezza delle reti sta apparendo all'orizzonte come un problema potenzialmente molto vasto. In questo capitolo studieremo la sicurezza delle reti da diversi punti di vista, indicheremo molti punti critici e discuteremo un gran numero di algoritmi e protocolli che servono a rendere le reti più sicure.

La sicurezza è un argomento ampio, che copre una moltitudine di problemi. Nella sua forma più semplice, riguarda come fare in modo che intrusi non riescano a leggere (o, peggio ancora, modificare di nascosto) i messaggi destinati a terzi. Si occupa inoltre di impedire che determinate persone possano accedere a servizi remoti che non sono autorizzate a usare. La sicurezza si occupa anche di come accertarsi dell'identità dei mittenti dei messaggi, di come impedire l'intercettazione e la ripetizione di messaggi legittimi catturati sulla rete, e di come perseguire chi afferma di non aver mai spedito certi messaggi.

La maggior parte dei problemi di sicurezza sono causati da persone malintenzionate, che tentano di guadagnare qualcosa, catturare l'attenzione oppure danneggiare qualcuno. Una lista dei profili più comuni è elencata nella Figura 8.1. Dall'elenco dovrebbe essere chiaro che rendere sicura una rete richiede uno sforzo decisamente maggiore della semplice eliminazione degli errori di programmazione. Bisogna infatti affrontare persone intelligenti, tenaci, e in alcuni casi anche ben finanziate. Dovrebbe anche essere evidente che misure adeguate per sconfiggere avversari occasionali avranno un impatto limitato su

quelli più pericolosi. I rapporti di polizia dimostrano che la maggior parte degli attacchi non vengono da personaggi esterni che intercettano linee telefoniche, ma da personale interno spinto dal risentimento: i sistemi di sicurezza vanno progettati tenendo in mente anche questi fatti.

Nemico	Scopo
Studente	Divertirsi a leggere le e-mail degli altri
Cracker	Provare i sistemi di sicurezza di qualcuno; rubare dati
Venditore	Vantarsi di rappresentare tutta l'Europa, non solo Andorra
Uomo d'affari	Scoprire il piano di marketing strategico del concorrente
Ex-impiegato	Vendetta per il licenziamento
Contabile	Vendetta per il licenziamento
Agente di borsa	Negare un accordo fatto al cliente via e-mail
Commesso	Rubare numeri di carte di credito per rivenderli
Spira	Scoprire i segreti militari o industriali del nemico
Terrorista	Rubare i progetti di un'arma biologica

Figura 8.1. Alcuni personaggi che creano problemi di sicurezza, con le motivazioni tipiche.

I problemi di sicurezza delle reti si possono orientativamente dividere in quattro aree interconnesse fra loro: segretezza, autenticazione, non-ripudio (*nonrepudiation*) e controllo dell'integrità. La segretezza, detta anche confidenzialità, si occupa di mantenere le informazioni fuori dalla portata degli utenti non autorizzati: è ciò che la gente di solito associa al concetto di sicurezza delle reti. L'autenticazione si occupa di stabilire l'identità del soggetto con cui stiamo comunicando, prima di rivelare informazioni sensibili o concludere transazioni commerciali. Il termine non-ripudio riguarda le firme: come si può dimostrare che il cliente ha veramente inviato un ordine elettronico per dieci milioni di gadget a 89 cent l'uno, quando poi dichiara che il prezzo era 69 cent oppure nega di aver inviato l'ordine? Infine, come si fa a essere sicuri che il messaggio che abbiamo ricevuto è realmente quello spedito e non è stato modificato in transito o inventato di sana pianta da avversari malintenzionati?

Tutti questi problemi (segretezza, autenticazione, non-ripudio e controllo dell'integrità) esistono anche nei sistemi tradizionali, ma con delle differenze significative. L'integrità e la segretezza si raggiungono, nei sistemi tradizionali, usando posta raccomandata e chiudendo a chiave i documenti. Assaltare il treno postale è più difficile ai giorni nostri che ai tempi di Jesse James...

Di solito le persone riescono a vedere la differenza tra un documento cartaceo originale e una fotocopia e ne tengono conto. Come test, pensiamo di fare la fotocopia di un assegno valido. Il lunedì andiamo in banca a incassare l'assegno originale, e il martedì proviamo a incassare la fotocopia: sicuramente il comportamento della banca non è lo stesso. Con gli

assegni elettronici, l'originale e la copia sono indistinguibili. È necessario del tempo perché le banche imparino a trattare anche questo tipo di transazioni.

Le persone autenticano altre persone riconoscendo le loro facce, la voce e la calligrafia. L'autenticazione può essere realizzata con una firma su un documento, un sigillo, ecc. Le falsificazioni sono generalmente identificabili dagli esperti in calligrafia, inchiostri e carta. Nessuna di queste possibilità esiste nel mondo elettronico, quindi servono altre soluzioni. Prima di parlare delle soluzioni, è utile una digressione per considerare quali sono gli strati e i protocolli interessati dalla sicurezza. Probabilmente non c'è un solo posto, ogni strato può contribuire con qualcosa. Nello strato fisico, l'ascolto del canale può essere evitato racchiudendo le linee di trasmissione dentro a un tubo sigillato che contiene gas ad alta pressione. Ogni tentativo di forare il tubo rilascerà il gas, riducendo la pressione e facendo quindi scattare un allarme. Alcuni sistemi militari usano questa tecnica.

Nello strato data link, i pacchetti di una linea punto-punto possono essere cifrati quando lasciano un computer e poi decifrati appena entrano in un altro. Tutti i dettagli sono gestiti dallo strato data link, mentre gli strati superiori non si accorgono di queste operazioni. Purtroppo la soluzione non è attuabile quando i pacchetti devono attraversare più router. Il motivo è che i pacchetti dovrebbero essere decifrati in ogni router, rimanendo quindi vulnerabili ad attacchi dall'interno del router. Inoltre questo metodo permette di proteggere solo alcune sessioni (per esempio quelle relative ad acquisti on-line con carta di credito) mentre altre rimangono in chiaro. Questo metodo si chiama cifratura del canale o **link encryption** e in ogni caso è spesso utile e può essere facilmente aggiunto in qualsiasi rete. Nello strato network si possono installare firewall per filtrare i pacchetti buoni e respingere quelli cattivi. Anche la IP security agisce a questo strato.

Nello strato trasporto, l'intera connessione può essere cifrata da capo a capo, cioè da processo sorgente a processo ricevente. Ciò è necessario per ottenere la massima sicurezza. Infine, problemi come l'autenticazione degli utenti e il non-ripudio possono essere trattati solo allo strato applicativo.

Poiché la sicurezza non si può attribuire in modo netto a uno strato specifico, non è stata discussa in nessuno dei capitoli precedenti ma le viene dedicato un capitolo a parte. Sebbene questo capitolo sia lungo, tecnico ed essenziale, oggi è praticamente ininfluente. Per esempio, è ben documentato che la maggior parte delle falliche di sicurezza presso le banche sono dovute a personale incompetente, procedure di sicurezza deboli, o frodi dall'interno; e non a geni criminali che ascoltano le linee telefoniche e decrittano i messaggi in codice. Se una persona può entrare in una filiale qualunque di una banca con una carta bancomat che ha trovato per la strada, dicendo di aver perso il codice PIN e perciò ottenere uno nuovo, tutta la crittografia del mondo non aiuterà a prevenire gli abusi. Il libro di Ross Anderson è una vera rivelazione a riguardo, infatti documenta centinaia di casi di falliche nella sicurezza in molti settori industriali, praticamente tutti dovuti a quelle che possiamo chiamare educatamente "pratiche operative deboli", o mancanze di attenzione alla sicurezza (Anderson, 2001). A ogni modo siamo fiduciosi che, con il diffondersi del commercio elettronico, le aziende correggeranno le loro procedure operative, eliminando le falliche e portando gli aspetti tecnici della sicurezza al centro dell'attenzione.

Eccezione fatta per la sicurezza dello strato fisico, praticamente tutto il resto si basa su principi crittografici. Per questo motivo cominceremo il nostro studio della sicurezza esaminando alcuni dettagli della crittografia. Nel Paragrafo 8.1 esamineremo alcuni principi base. Nei Paragrafi da 8.2 a 8.5, vedremo alcuni algoritmi fondamentali e strutture dati usate in crittografia, e studieremo poi come questi concetti possono essere applicati alla sicurezza delle reti. Concluderemo con alcune osservazioni sulla tecnologia e la società. Prima di cominciare vediamo anche cosa non verrà discusso. Abbiamo tentato di focalizzarci sui problemi legati alle reti piuttosto che ai sistemi operativi o le applicazioni, anche se a volte è difficile distinguerne il confine. Per esempio in questo capitolo non c'è niente sull'autenticazione degli utenti con tecniche biometriche, sulla sicurezza delle password, sugli attacchi di tipo buffer overflow, i cavalli di Troia, lo spoofing dei login, le bombe logiche, i virus, i worm, ecc. Questi argomenti sono trattati in dettaglio nel Capitolo 9 di *Modern Operating Systems* (Tannenbaum, 2001). Il lettore interessato può far riferimento a questo testo per gli aspetti sistematici della sicurezza.

## 8.1 Crittografia

**Crittografia** deriva da due parole di greco antico che significano "scrittura segreta". La crittografia ha una storia lunga e avventurosa che risale a migliaia di anni fa. In questo paragrafo vedremo solo alcuni dei punti principali, che ci serviranno come base per la comprensione del seguito. Per una storia completa della crittografia si raccomanda la lettura del libro di Kahn (1995). Un trattato completo e aggiornato della crittografia con i suoi algoritmi, protocolli e applicazioni è (Kaufman et al. 2002). Per un approccio più matematico vedi (Stinson, 2002). Per un approccio meno matematico (Burnett e Paine, 2001).

I professionisti distinguono i termini cifrario e codice. Con **cifrario** s'intende una trasformazione carattere per carattere (o bit per bit), senza considerare la struttura linguistica del messaggio. Al contrario, un **codice** rimpiazza ogni parola con un'altra parola o simbolo. I codici non vengono più utilizzati, anche se hanno alle loro spalle una storia gloriosa. Il codice di maggior successo fu usato dalle forze armate americane nel pacifico durante la seconda guerra mondiale: si impiegavano indiani Navajo che comunicavano tra loro con le parole Navajo che definiscono i termini militari. Per esempio, *chay-da-gahi-nail-tsaidi* (letteralmente: ammazza tartarughe) indicava le armi anticarro. La lingua Navajo è molto tonale, estremamente complessa e non ha una forma scritta, inoltre nessuno in Giappone ne conosceva l'esistenza.

Nel settembre 1945, il *San Diego Union* descrisse il codice dicendo "per tre anni, ovunque sbucavano i marines, i giapponesi ascoltavano una serie di strani suoni gutturali inframmezzati con altri suoni che rassomigliavano a un misto fra la voce di un monaco tibetano e il suono del boiler dell'acqua calda che si svuota". I giapponesi non riuscirono mai a interpretare il codice, e molti codificatori Navajo ricevettero gli alti onori militari per

servizio straordinario e il loro coraggio. Il fatto che gli Stati Uniti decrittarono il codice giapponese, mentre il Giappone non decrittò mai quello dei Navajo, ebbe un ruolo cruciale nella vittoria americana nel pacifico.

### 8.1.1 Introduzione alla crittografia

Praticamente quattro gruppi di persone hanno usato e dato il loro contributo all'arte della crittografia: i militari, il corpo diplomatico, gli scrittori di diari e gli amanti. Fra tutti questi, i militari hanno avuto il ruolo più importante e hanno dato forma alla disciplina nel corso dei secoli. All'interno delle organizzazioni militari, i messaggi da cifrare erano tradizionalmente assegnati ad addetti di basso livello e bassa paga che avevano il compito di cifrare e poi trasmetterli. Il notevole volume di messaggi faceva sì che non si potesse usare una squadra di specialisti di alto livello per questo lavoro.

Primo all'avvento dei computer, uno dei principali limiti della crittografia era la capacità degli addetti alla codifica di riuscire a compiere le operazioni necessarie, spesso anche in campo di battaglia e utilizzando un equipaggiamento minimo. Un vincolo aggiuntivo era anche la difficoltà nel cambiare velocemente da un metodo crittografico a un altro, in quanto questo significava dover formare una gran quantità di persone. Comunque, il pericolo che un addetto alla codifica potesse essere catturato dal nemico rendeva essenziale la possibilità di cambiare il metodo crittografico all'istante, se necessario. Questi requisiti in conflitto fra loro hanno dato origine al modello della Figura 8.2.

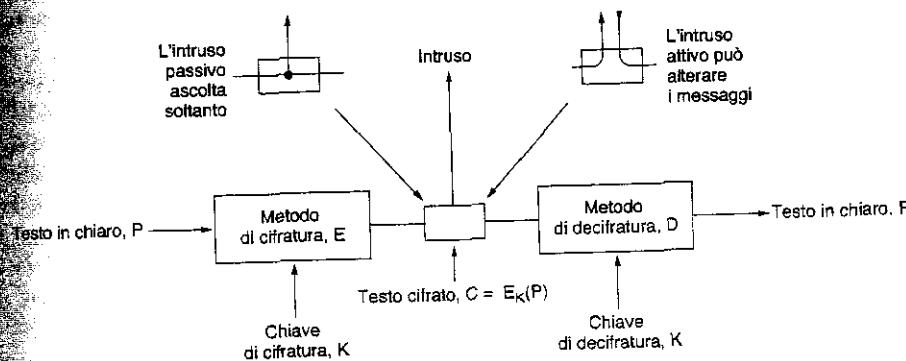


Figura 8.2. Il modello di cifratura per un cifrario a chiave simmetrica.

I messaggi da cifrare, detti **testo in chiaro**, sono trasformati tramite una funzione parametrizzata da una **chiave**. L'output del processo di cifratura, noto come **testo cifrato**, viene generalmente trasmesso tramite un messaggero o la radio. Assumiamo che il nemico, o **intruso**, ascolti accuratamente e trascriva completamente il testo cifrato. Al contrario del destinatario legittimo, l'intruso non conosce la chiave di decifrazione e quindi non

riesce facilmente a decifrare il testo cifrato. In alcuni casi l'intruso, oltre ad ascoltare il canale di comunicazione (intruso passivo), riesce anche a registrare i messaggi per poi ritrasmetterli aggiungendo delle informazioni, oppure per modificare i messaggi legittimi prima che raggiungano il destinatario (intruso attivo). L'arte di decriptare i cifrari, chiamata **criptoanalisi**, e l'arte di inventarli (crittografia) sono note sotto il nome collettivo di **crittologia**. **Decriptare** è l'attività di decifrazione da parte di un criptoanalista (intruso), mentre **decifrare** è l'operazione legittima di lettura di un messaggio cifrato. È utile avere una notazione per relazionare fra loro il testo in chiaro, quello cifrato e le chiavi. Useremo la notazione  $C = E_K(P)$  per indicare che la cifratura del testo in chiaro  $P$  usando la chiave  $K$  genera il testo cifrato  $C$ . Analogamente,  $P = D_K(C)$  indica la decifrazione di  $C$  per estrarre il testo in chiaro. Segue quindi che

$$D_K(E_K(P)) = P$$

Questa notazione suggerisce che  $E$  e  $D$  sono semplicemente delle funzioni matematiche. Il punto difficile è che si tratta di funzioni di due parametri. Abbiamo scritto uno dei parametri (la chiave) come indice per distinguerlo dall'altro argomento (il messaggio). Una regola fondamentale della crittografia afferma che bisogna sempre assumere che il criptoanalista conosca il metodo usato per la cifratura e la decifrazione. In altre parole si deve supporre che il criptoanalista conosca nei dettagli il funzionamento del metodo  $E$  di cifratura e  $D$  di decifrazione. Lo sforzo necessario per inventare, testare e installare un nuovo algoritmo ogni volta che quello vecchio è compromesso (o si pensa che lo sia stato) ha sempre reso impraticabile la strada di tenere segreto l'algoritmo di cifratura. Credere di avere un algoritmo segreto quando non è vero produce più danno che guadagno. Qui entrano in gioco le chiavi. La chiave crittografica consiste di una stringa (relativamente) corta che identifica una particolare cifratura fra molte possibilità. Diversamente dal metodo generale, che può essere cambiato solo una volta ogni diversi anni, la chiave può essere cambiata tutte le volte che si ritiene necessario. Quindi il nostro modello base è dato da un metodo generale stabile e noto pubblicamente, però parametrizzato da un chiave segreta che può essere facilmente cambiata. L'idea che il criptoanalista conosca gli algoritmi e che il segreto stia esclusivamente nella chiave è detto **principio di Kerckhoff**, da nome del crittografo fiammingo Auguste Kerckhoff, che per primo formulò il principio nel 1883 (Kerckhoff, 1883). Riassumendo:

**Principio di Kerckhoff:** tutti gli algoritmi devono essere pubblici, solo le chiavi sono segrete.

La mancanza di segratezza dell'algoritmo è fondamentale. Cercare di tenere segreto l'algoritmo, metodo noto in gergo come **sicurezza per occultamento**, non funziona mai. Inoltre, pubblicizzando l'algoritmo il crittografo ottiene gratuitamente la consulenza di un gran numero di accademici desiderosi di forzare il sistema, per poter pubblicare degli articoli in cui dimostrano quanto sono intelligenti. Se un gran numero di esperti ha tentato di forzare l'algoritmo per 5 anni dalla sua pubblicazione e nessuno c'è riuscito, probabilmente si tratta di un algoritmo decisamente solido.

Visto che la vera segratezza sta nella chiave, la sua lunghezza è un argomento importante nella progettazione di un sistema crittografico. Consideriamo, per esempio, un semplice lucchetto a combinazione. Il principio generale è che le cifre vengono immesse in sequen-

za, il che è noto, mentre la chiave è segreta. Una chiave di due cifre significa che ci sono 100 possibilità. Una chiave di tre cifre significa 1.000 possibilità, mentre per sei cifre ci sono un milione di possibilità. Più lunga è la chiave, più alto è il **fattore lavoro** che il criptoanalista deve mettere in conto: il fattore lavoro per rompere il sistema con la ricerca completa dello spazio delle chiavi cresce esponenzialmente con la lunghezza della chiave. La segratezza viene dall'avere un algoritmo forte (ma pubblico) e una chiave lunga. Per evitare che il vostro fratellino legga le e-mail private che non lo riguardano, può bastare una chiave di 64 bit. Per il normale uso commerciale si dovrebbero usare almeno 128 bit. Per la sicurezza delle comunicazioni fra enti governativi sono auspicabili chiavi di almeno 256 bit.

Dal punto di vista del criptoanalista, i problemi di criptoanalisi hanno tre variazioni principali. Quando si ha solo testo cifrato e nessun testo in chiaro si parla di un problema con **solo testo cifrato**. I criptogrammi che si trovano nella sezione enigmistica dei giornali sono di questo tipo. Quando invece si ha a disposizione del testo cifrato con il corrispondente testo in chiaro, si parla di problema con **testo in chiaro noto**. Infine, quando il criptoanalista ha la possibilità di cifrare dei pezzi di testo in chiaro a scelta, parliamo di problema con **testo in chiaro a scelta**. I criptogrammi dei giornali potrebbero essere risolti facilmente se il criptoanalista potesse avere risposta a domande del tipo: qual è il testo cifrato di ABCDEFGHIJKL?

I principianti che si accostano alla crittografia spesso ipotizzano che se un cifrario riesce a resistere a un attacco di tipo solo testo cifrato, allora è sicuro. Questa ipotesi è troppo semplicistica. Infatti, in molti casi, i criptoanalisti riescono a indovinare informazioni importanti riguardo al testo in chiaro. Per esempio, la prima cosa che molti computer dicono quando vengono contattati è "login:". A questo punto, conoscendo testo in chiaro e testo cifrato corrispondente il lavoro del criptoanalista diventa molto più semplice. Il crittografo dovrebbe perciò cautelarsi, in modo da essere sicuro che il sistema non sia forzabile neanche se l'altra parte riesce a cifrare una quantità arbitraria di testo in chiaro.

I metodi crittografici sono storicamente divisi in due categorie: cifrari a sostituzione e cifrari a trasposizione. Vedremo velocemente entrambi i casi, il che ci servirà come introduzione alla crittografia moderna.

### 8.1.2 Cifrari a sostituzione

In un cifrario a sostituzione, ogni lettera o gruppo di lettere viene rimpiazzato da un'altra lettera o gruppo di lettere per mascherare il messaggio. Uno dei cifrari più antichi che si conoscano è il **cifrario di Cesare**, attribuito a Giulio Cesare. In questo metodo, *a* diventa *D*, *b* diventa *E*, *c* diventa *F*, ... e *z* diventa *C*. Per esempio, la parola *attacco* diventa *DWWDFRR*. Negli esempi il testo in chiaro è scritto in minuscolo e quello cifrato in maiuscolo.

Una semplice generalizzazione del cifrario di Cesare consiste nello spostare l'alfabeto del testo cifrato di  $k$  lettere, dove  $k$  vale tre per il cifrario di Cesare descritto sopra. In questo caso  $k$  diventa la chiave del metodo generale, che consiste nell'avere un alfabeto spostato circolarmente. Il cifrario di Cesare può anche aver tratto in inganno Pompeo, ma da allora non ha più ingannato nessuno.

Il miglioramento successivo consiste nel far sì che ognuno dei simboli del testo in chiaro (diciamo le lettere dell'alfabeto, per semplicità) corrisponda ad altre lettere. Per esempio:

Testo in chiaro: a b c d e f g h i j k l m n o p q r s t u v w x y z  
 Testo cifrato: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

Il sistema generale per la sostituzione simbolo a simbolo viene chiamato **sostituzione monoalfabetica**, dove la chiave è la stringa di lettere che corrisponde all'intero alfabeto. Con la chiave di cui sopra, la parola *attacco* diventa *QZZQEEG*.

A prima vista questo può sembrare un sistema sicuro, perché anche se il criptoanalista conosce il metodo (sostituzione lettera a lettera), non conosce però quale delle  $26! = 4 \times 10^{26}$  possibili chiavi viene usata. In questo caso, diversamente dal cifrario di Cesare, provare tutte le chiavi non è un approccio promettente. Anche alla velocità di 1 nsec per chiave, un computer impiegherebbe  $10^{10}$  anni per provare tutte le chiavi.

A ogni modo, data una quantità anche molto piccola di testo cifrato, il cifrario può essere forzato facilmente. L'attacco base fa leva sulle proprietà statistiche dei linguaggi naturali. In inglese, per esempio, la *e* è la lettera più comune, seguita da *t*, *o*, *a*, *n*, *i*, ecc. Le combinazioni di due lettere, o **digrammi** più comuni sono *th*, *in*, *er*, *re*, *an*. Le combinazioni di tre lettere, o **trigrammi** più comuni sono *the*, *ing*, *and* e *ion*.

Un criptoanalista che voglia forzare un cifrario monoalfabetico comincerebbe contando la frequenza relativa di tutte le lettere nel testo cifrato. A quel punto può procedere per tentativi, associando la lettera più comune con la *e*, quella successiva nell'ordine alla *t*. Potrebbe poi guardare i trigrammi per arrivare a scoprirne uno frequente della forma *tXe*, il che gli suggerisce che *X* è *h*. Analogamente scoprirà che la sequenza *thYt* appare spesso, quindi la *Y* è probabilmente una *a*. Con queste informazioni può cercare i trigrammi nella forma *aZW*, che molto probabilmente corrisponde a *and*. Il criptoanalista ricostruisce per tentativi il testo in chiaro, lettera per lettera, facendo delle ipotesi su lettere, digrammi e trigrammi più comuni e conoscendo le sequenze più probabili di vocali e consonanti.

Un altro approccio consiste nell'indovinare una parola o una frase che probabilmente si trovano dentro al testo. Per esempio, consideriamo il testo cifrato riportato qui sotto e preso da un'azienda che si occupa di contabilità (riportato in gruppi di cinque caratteri):

CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ  
 QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ  
 DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW

Una parola che molto probabilmente appare nel messaggio, vista la sua sorgente, è *financial*. Sapendo che *financial* ha una lettera (*i*) ripetuta con quattro lettere fra una posizione e l'altra, cerchiamo nel testo cifrato le lettere ripetute con questa distanza fra di loro. Troviamo 12 possibilità alle posizioni: 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 e 82. Solo due di queste, 31 e 42 hanno la lettera successiva (corrispondente a *n* nel testo in chiaro) ripetuta al posto giusto. Di queste, solo 31 ha la *a* nella posizione corretta, in questo

modo sappiamo che *financial* comincia alla posizione 30. Da questo punto, si può dedurre facilmente la chiave usando le statistiche di frequenza delle lettere per i testi in inglese.

### 8.1.3 Cifrari a trasposizione

I cifrari a sostituzione conservano l'ordine dei simboli del testo in chiaro, limitandosi a mascherare la loro apparenza. I **cifrari a trasposizione**, al contrario, riordinano le lettere ma non le mascherano. La Figura 8.3 mostra un tipico cifrario a trasposizione, il cifrario a trasposizione colonna. La cifra ha come chiave una parola o una frase senza lettere ripetute. Nell'esempio la chiave è **MEGABUCK**. Lo scopo della chiave è quello di numerare le colonne: la colonna numero 1 è quella sotto la lettera della chiave più vicina all'inizio dell'alfabeto, e così via. Il testo in chiaro viene scritto orizzontalmente, per righe, fino a riempire la matrice, eventualmente usando anche alcuni caratteri di riempimento allo scopo. Il testo cifrato viene letto per colonna, cominciando con quella che ha lettera chiave più bassa.

M E G A B U C K	
Z 4 5 1 2 8 3 6	
p t e a s e t r	Testo in chiaro
a n s f e r o n	please transfer one million dollarsto
e m i l l i o n	my swissbankaccountsixtwo
d o l l a r s t	Testo cifrato
o m y s w i s s	AFLLSKSOSELAWAIAATOOSCTCLNMOMANT
b a n k a c c o	ESILYNTWRNNTSOWDPAEDOBUCERIRICXB
u n t s i x t w	
o t w o a b c d	

Figura 8.3. Cifrario a trasposizione.

Per forzare un cifrario a trasposizione il criptoanalista, come prima cosa, deve sapere che sta trattando con un cifrario a trasposizione. Guardando alla frequenza delle lettere *E*, *T*, *A*, *O*, *I*, *N*, ecc è facile vedere se corrispondono alla normale frequenza per un testo in chiaro. Se questo è vero, si tratta chiaramente di un cifrario a trasposizione, infatti in un cifrario di questo tipo ogni lettera rappresenta sé stessa e la distribuzione di frequenza rimane intatta.

Il passo successivo consiste nel fare un'ipotesi sul numero di colonne. In molti casi si riesce a indovinare dal contesto una parola o frase che con buona probabilità appaiono nel testo. Per esempio, supponiamo che il criptoanalista sospetti che il testo in chiaro contenga da qualche parte la stringa *milliondollars*. Notiamo che i digrammi *MO*, *IL*, *LL*, *LA*, *IR* e *OS* appaiono nel testo come risultato della cifratura di tale stringa. Nel testo cifrato la lettera *O* segue la lettera *M* (cioè le due lettere sono verticalmente adiacenti nella colonna 4), e questo ci fa ipotizzare che, all'interno della stringa *milliondollars*, queste siano separate da una distanza pari alla lunghezza della chiave. Se si fosse utilizzata una chiave di lunghezza sette, avremmo trovato invece i digrammi *MD*, *IO*, *LL*, *LA*, *OR* e *NS*. In

effetti, per ogni possibile lunghezza della chiave possiamo trovare un differente insieme di digrammi nel testo cifrato. Il criptoanalista è spesso in grado di determinare la lunghezza della chiave semplicemente provando tutte le possibilità.

Il passo conclusivo consiste nel trovare l'ordine delle colonne. Quando il numero delle colonne  $k$  è piccolo, ognuna delle  $k(k-1)$  possibili coppie di colonne può essere esaminata per vedere se le frequenze di occorrenza dei suoi digrammi corrispondono a quelle tipiche dell'inglese. Si suppone poi che la coppia con il miglior risultato in questo test sia posizionata correttamente. A questo punto si prosegue con le colonne rimanenti, finché non si identifica la colonna successiva alla coppia: questa sarà la colonna che produce la miglior distribuzione di frequenza dei digrammi e dei trigrammi. Analogamente si procede per trovare la colonna che precede la coppia. Questo processo viene ripetuto finché non si identifica un possibile ordinamento delle colonne. A questo punto c'è già una buona possibilità che appaia il testo in chiaro (per esempio, se leggiamo *milloin* sapremo chiaramente dove è stato fatto un errore).

Alcuni cifrari a trasposizione prendono un blocco di lunghezza fissa in input e producono in output un altro blocco di lunghezza fissa. Questi cifrari possono essere descritti completamente dalla lista che indica l'ordine in cui i caratteri sono mandati in output. Per esempio, il cifrario della Figura 8.3 è un cifrario a blocchi di 64 bit e l'ordine dei caratteri in output è 4, 12, 20, 28, 36, 44, 52, 60, 5, 12, ..., 62. In altre parole, il quarto carattere in input,  $a$ , è il primo in output, seguito dal dodicesimo,  $f$ , e così di seguito.

#### 8.1.4 Blocci monouso

Costruire un cifrario imbattibile è molto facile e la tecnica per farlo è nota da decenni. Per prima cosa, si prende una chiave formata da una stringa di bit generati a caso. Poi, si converte il testo in chiaro in un'altra stringa di bit, per esempio usando la rappresentazione ASCII per i caratteri. Infine, si calcola lo XOR (OR esclusivo) delle due stringhe, bit per bit. Il testo cifrato risultante non può essere forzato, in quanto per ogni pezzo sufficientemente grande del testo cifrato, ogni lettera apparirà con la stessa frequenza, lo stesso vale per i digrammi e i trigrammi, ecc. Questo metodo, noto come **blocco monouso (one-time pad)**, è immune a tutti gli attacchi presenti e futuri indipendentemente da quanta potenza di calcolo l'intruso abbia a disposizione. La ragione di questo fatto viene dalla teoria dell'informazione: il messaggio cifrato non contiene nessuna informazione in quanto contiene tutti i possibili testi in chiaro di una data lunghezza con eguale probabilità. Un esempio di come sono usati i blocchi monouso è dato dalla Figura 8.4. Nell'esempio, il messaggio 1 "I love you" viene convertito in ASCII a 7 bit. A quel punto il blocco monouso 1 viene scelto e usato per l'operazione di XOR con il messaggio per produrre il testo cifrato. Un criptoanalista potrebbe provare tutte le possibili combinazioni di blocchi monouso per vedere quale testo in chiaro emerga da ognuna di esse. Per esempio potrebbe usare il blocco monouso indicato con il numero 2 in figura: il risultato sarebbe il testo in chiaro 2 "Elvis lives", che potrebbe anche essere plausibile (l'argomento va oltre lo scopo di questo libro). In effetti, per ogni testo ASCII di 11 caratteri in chiaro, possiamo trovare un blocco monouso che lo genera. Questo è quello che intendiamo dicendo che non c'è informazione nel testo cifrato: partendo da esso possiamo leggere qualunque messaggio della lunghezza corretta.

```

Messaggio 1: 1001001 0100000 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110
Blocco 1: 1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011
Testo cifrato: 0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101

Blocco 2: 1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110
Testo cifrato 2: 1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011

```

Figura 8.4. Un blocco monouso per la cifratura dà la possibilità di ottenere qualsiasi testo in chiaro dal testo cifrato cambiando il blocco.

I blocchi monouso sono ottimi in teoria, ma hanno diversi svantaggi nella pratica. Per cominciare, la chiave non può essere imparata a memoria, quindi sia il mittente sia il destinatario devono averne una copia. Se pensiamo che uno dei due soggetti possa essere catturato, vediamo che avere delle chiavi scritte non è desiderabile. Inoltre, la quantità di dati che può essere trasmessa è limitata dalla lunghezza della chiave. Se una spia ha fortuna e scopre una gran quantità di dati, si può trovare nella situazione di non poterli trasmettere al quartier generale perché ha esaurito la chiave a sua disposizione. Un altro problema del metodo è la debolezza nel trattare i caratteri persi o inseriti per sbaglio. Se a un certo punto della comunicazione il mittente e il destinatario perdono la sincronizzazione, il resto dei dati sarà completamente illeggibile.

Con l'avvento dei computer, la crittografia a blocchi monouso può diventare una strada praticabile per alcune applicazioni. La sorgente della chiave potrebbe essere uno speciale DVD contenente diversi gigabyte d'informazioni, da trasportare in una custodia da film e costruito con l'accortezza di aggiungere alcuni minuti di video all'inizio per sviare i sospetti. Chiaramente, alla velocità delle reti gigabit ethernet, dover inserire un nuovo DVD ogni 30 secondi può diventare molto fastidioso. Inoltre il DVD dovrebbe essere trasportato di persona dal mittente al destinatario prima di cominciare a trasmettere, il che limita severamente l'utilità pratica di questo metodo.

#### Crittografia quantistica

Un'interessante risposta all'esigenza di trasmettere i blocchi monouso attraverso la rete arriva, inaspettatamente, dalla meccanica quantistica. Questa soluzione è ancora sperimentale, ma molto promettente. Se si riuscirà a perfezionare il metodo rendendolo efficiente, praticamente tutta la crittografia finirà per essere realizzata con i blocchi monouso, visto che sono provatamente sicuri. Nel seguito spiegheremo come funziona questo metodo, detto **crittografia quantistica**. In particolare descriveremo un protocollo, detto **BB84** dal nome dei suoi autori e dell'anno di pubblicazione (Bennet e Brassard, 1984).

Un utente, Alice, vuole stabilire un blocco monouso con un secondo utente, Bob. Alice e Bob sono detti i **protagonisti**, gli attori principali della nostra storia. Per esempio, Bob è un banchiere con cui Alice vuole fare affari. I nomi "Alice" e "Bob" vengono usati per indicare i protagonisti in tutti gli articoli e i libri di crittografia degli ultimi dieci anni. I crittografi amano la tradizione. Se usassimo "Andy" e "Barbara" come nomi dei protagonisti, nessuno crederebbe a una riga di questo capitolo: per questo motivo ci atterremo alla tradizione. Se Alice e Bob riuscissero a stabilire un blocco monouso, lo potrebbero usare per comu-

nicare in modo sicuro. La domanda è: come possono farlo senza essersi scambiati dei DVD in precedenza? Immaginiamo che Alice e Bob siano agli estremi opposti di una fibra ottica, che possono usare per inviare e ricevere impulsi luminosi. Un intruso attivo, che chiameremo Trudy, può inserirsi nella fibra per violare la sicurezza della comunicazione. Trudy può leggere tutti i bit e mandare messaggi falsi in entrambe le direzioni. La situazione può sembrare un vicolo cieco per Alice e Bob, invece con la crittografia quantistica si riescono ad avere dei nuovi risultati.

La crittografia quantistica si basa sul fatto che la luce viene trasportata in piccoli pacchetti, chiamati **foton**, che hanno delle proprietà particolari. Un altro fatto importante è che la luce può essere polarizzata facendola passare attraverso dei filtri polarizzatori, un fatto ben noto ai fotografi e a chi indossa occhiali da sole. Se un raggio di luce (cioè un flusso di fotoni) viene fatto passare attraverso un polarizzatore, tutti i fotoni emergenti saranno polarizzati nella direzione dell'asse del filtro polarizzatore (per esempio, un asse verticale). Se il raggio viene fatto passare attraverso un secondo filtro polarizzatore, l'intensità della luce emergente dal secondo filtro è proporzionale al quadrato del coseno dell'angolo fra gli assi dei due polarizzatori. Se i due assi sono perpendicolari, nessun fotone riesce a passare. L'orientamento assoluto dei due filtri non è importante, l'unica cosa rilevante è l'angolo fra i loro assi.

Per generare un blocco monouso, Alice ha bisogno di due insiemi di filtri. Il primo contiene un filtro verticale e uno orizzontale. Questa scelta è chiamata una **base rettilinea**. Una base è un sistema di coordinate. Il secondo insieme è simile al primo, solo con una rotazione aggiuntiva di 45 gradi. In questo modo l'asse di un filtro è orientato da in basso a sinistra a in alto a destra, mentre l'altro va da in alto a sinistra a in basso a destra. Questa scelta è detta **base diagonale**. Quindi Alice ha due basi che può inserire nel raggio di luce, supponiamo senza limitazioni di velocità. In realtà Alice non ha quattro filtri separati, ma un cristallo la cui polarizzazione può essere cambiata elettronicamente in modo rapido in ognuna delle quattro direzioni indicate sopra. Bob ha la stessa attrezzatura di Alice. Il fatto che Bob e Alice abbiano a disposizione due basi ciascuno è essenziale per la crittografia quantistica.

Per ogni base, Alice identifica il bit 0 con una certa direzione e l'1 con la direzione ortogonale allo 0. Nell'esempio che segue assumeremo che lo 0 sia la direzione verticale, mentre l'1 quella orizzontale. Un discorso analogo vale per la base diagonale: Alice assegna lo 0 alla direzione dal basso a sinistra verso l'alto a destra e l'1 all'altra direzione. Alice trasmette queste scelte in chiaro a Bob.

A questo punto Alice genera un blocco monouso, per esempio basandosi su un generatore di numeri casuali (un argomento complesso da discutere a parte) e lo trasmette a Bob bit per bit, scegliendo per ogni bit una delle due basi in modo casuale. Per esempio, Alice può scegliere di usare per il primo bit la base diagonale, per il secondo quella rettilinea, poi rettilinea di nuovo, ecc. Per trasmettere il blocco monouso 1001110010100110 con queste basi, Alice dovrebbe inviare i fotoni come nella Figura 8.5(a). Dato il blocco monouso e l'ordine delle basi, la polarizzazione da utilizzare per ogni bit è determinata univocamente. I bit inviati come singoli fotoni sono chiamati **qubits**.

Bob non sa quale base utilizzare, quindi ne sceglie una a caso per ogni fotone in arrivo, come mostrato nella Figura 8.5(b). Se Bob sceglie la base corretta, avrà anche il bit corretto. Se invece sceglie la base sbagliata, otterrà un valore casuale. Infatti, quando un fotone attraver-

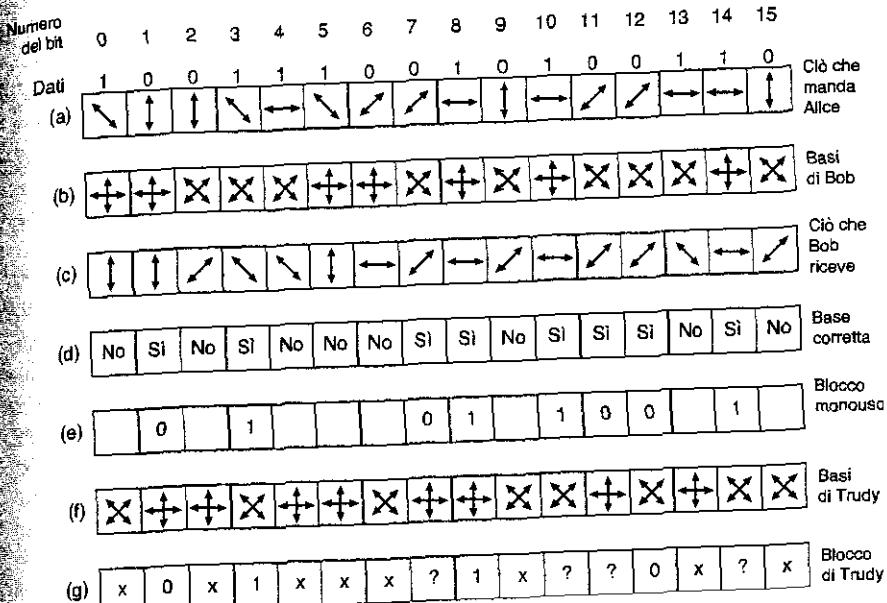


Figura 8.5. Un esempio di crittografia quantistica.

sa un filtro polarizzatore formando un angolo di 45 gradi fra la sua polarizzazione e l'asse del polarizzatore, il risultato è che il fotone collassa in modo casuale e con eguale probabilità o alla polarizzazione del filtro o alla polarizzazione a esso ortogonale. Questo risultato deriva dalle leggi fondamentali della meccanica quantistica. Il risultato è che alcuni bit sono ricevuti correttamente mentre altri hanno dei valori casuali, come mostrato nella Figura 8.5(c). Bob non è ancora in grado di sapere quali sono i bit corretti e quali quelli casuali. Come può fare Bob per avere le informazioni necessarie a sapere quando ha usato la base giusta e quando quella sbagliata? Semplicemente comunica in chiaro ad Alice quali basi ha utilizzato e Alice gli risponde, di nuovo in chiaro, dicendogli quali sono giuste e quali sbagliate, come mostrato nella Figura 8.5(d). Usando queste informazioni, riescono entrambi a costruire una stringa di bit trasmessi correttamente, come nella Figura 8.5(e). Questa stringa, che avrà una lunghezza media pari alla metà della stringa di bit originale, è nota a tutti e due i protagonisti e quindi potrà essere effettivamente usata come blocco monouso. Notiamo che per stabilire un blocco monouso di una certa lunghezza Alice deve semplicemente inviare una stringa di bit leggermente più lunga del doppio necessario, e il problema è risolto.

Abbiamo finora trascurato il ruolo di Trudy. Supponiamo che Trudy sia curiosa di sapere cosa dice Alice e quindi esegua una deviazione nella fibra ottica per inserire i suoi apparati di ricezione e trasmissione. Sfortunatamente, per Trudy, nemmeno lei riesce a sapere quali basi usare per ogni fotone entrante. La cosa migliore che può fare è scegliere a caso le basi, così come fa Bob. Un esempio delle possibili scelte di Trudy è mostrato nella Figura 8.5(f). Quando poi Bob trasmette (in chiaro) ad Alice l'elenco delle basi che ha usato e Alice gli

comunica (in chiaro) quali sono giuste e quali no, anche Trudy riesce a sapere quando ha usato le basi giuste e quando ha sbagliato. Nella Figura 8.5 vediamo che Trudy ha indovinato le basi per i bit 0, 1, 2, 3, 4, 6, 8, 12 e 13. Però Trudy riesce a sapere dalla risposta di Alice nella Figura 8.5(d) che solamente i bit 1, 3, 7, 8, 10, 11, 12 e 14 fanno parte del blocco monouso. Per quattro di questi bit aveva indovinato la sequenza delle basi (1, 3, 8 e 12), ma per gli altri quattro (7, 10, 11 e 14) aveva sbagliato e quindi non ne conosce i valori. Quindi Bob sa che il blocco monouso comincia con 01011001, come nella Figura 8.5(e), mentre Trudy sa che il blocco vale 01?1??0?, come nella Figura 8.5(g).

Ovviamente Bob e Alice sono a conoscenza del fatto che Trudy può aver catturato parte del loro blocco monouso, quindi desiderano ridurre ulteriormente la quantità di informazione che Trudy ha a disposizione. Riescono a farlo operando una trasformazione sui dati del blocco. Per esempio, possono dividere il blocco monouso in blocchi di 1.024 bit e calcolare il quadrato di ciascun blocco per formare dei numeri di 2.048 bit, e poi usare la concatenazione di questi numeri a 2.048 bit per formare il blocco monouso che utilizzeranno per la comunicazione. Trudy, con la sola conoscenza parziale del blocco, non riesce a calcolare i quadrati dei blocchi e quindi si ritrova senza niente in mano. La trasformazione che porta dal blocco originale a quello finale, riducendo così l'informazione in mano a Trudy, è detta **amplificazione della privacy**. In pratica, al posto del calcolo del quadrato, vengono usate delle trasformazioni complesse in cui tutti i bit dell'output dipendono da tutti i bit dell'input. I guai per Trudy non sono finiti qui. Infatti, non solo non riesce a sapere qual è il blocco monouso, ma la sua presenza viene anche rilevata. Per poter ascoltare la comunicazione, facendo pensare a Bob che la comunicazione sia in realtà con Alice, Trudy deve ritrasmettere tutti i qubit ricevuti. Il punto è che il massimo che può fare è trasmettere i bit ricevuti usando la polarizzazione che ha usato per riceverli, che però sarà sbagliata in media la metà delle volte. Dal punto di vista di Bob, questo si rifletterà in una frequenza elevata di errori di ricezione.

Quando Alice comincia a inviare i dati, li codifica usando dei metodi forti di correzione degli errori. Per Bob, un errore di 1 bit nel blocco monouso è la stessa cosa di un errore di trasmissione: in entrambi i casi ottiene un bit sbagliato. Se la codifica per la correzione degli errori è sufficientemente forte, Bob riesce a correggere gli errori e a ripristinare il messaggio originale. Bob può facilmente contare il numero di errori di trasmissione e confrontarli con la frequenza di errore che ci si aspetta per la linea in questione. Se scopre una quantità di errori anomala, saprà che Trudy ha ascoltato la linea e potrà agire di conseguenza (per esempio potrà dire ad Alice di usare un canale radio, oppure chiamare la polizia, ecc.). Se Trudy potesse clonare i fotoni, così da avere un fotone da esaminare e uno identico da trasmettere a Bob, potrebbe evitare di essere scoperta, ma al momento non sono noti metodi per clonare i fotoni. Anche se Trudy riuscisse a clonare i fotoni, questo non ridurrebbe minimamente l'utilità della crittografia quantistica per stabilire dei blocchi monouso.

La crittografia quantistica è stata già dimostrata per distanze fino a 60 km su fibra ottica, anche se, allo stato attuale, l'attrezzatura necessaria rimane complessa e costosa. A ogni modo, l'idea è promettente. Per ulteriori informazioni sulla crittografia quantistica si rimanda a (Mullins, 2002).

### 8.1.5 Due principi crittografici fondamentali

Due principi crittografici fondamentali sono alla base dello studio dei diversi sistemi crittografici che discuteremo nei paragrafi successivi.

#### Ridondanza

Il primo principio afferma che tutti i messaggi cifrati devono contenere una qualche forma di ridondanza, cioè d'informazione non necessaria alla comprensione del messaggio. Utilizziamo un esempio per chiarire come mai serve questa informazione aggiuntiva. Consideriamo un'azienda di vendita per corrispondenza, la "The Couch Potato" (TCP) con un catalogo di 60.000 prodotti. I programmati della TCP, pensando di aumentare l'efficienza, decidono che i messaggi contenenti gli ordini devono consistere di 16 byte con il nome del cliente, seguiti da 3 byte per i dati (1 byte per la quantità e 2 byte per il numero del prodotto). Gli ultimi 3 byte vengono cifrati usando una chiave molto lunga e nota solo alla TCP e ai suoi clienti.

A prima vista questo metodo potrebbe sembrare sicuro, e in un certo modo lo è, visto che gli intrusi passivi non riescono a decifrare i messaggi. Sfortunatamente però, ha un difetto fatale che lo rende inutile. Supponiamo che un'impiegata licenziata di recente voglia punire la TCP per averla licenziata. Prima di lasciare l'azienda, si scarica la lista dei clienti, lavora tutta la notte e scrive un programma che genera degli ordini finti usando però i veri nomi dei clienti. Visto che non ha la lista delle chiavi, l'ex impiegata inserisce dei numeri casuali negli ultimi 3 byte dei messaggi, quindi invia centinaia di ordini alla TCP. Quando questi messaggi arrivano a destinazione, il computer della TCP usa il nome del cliente per trovare la chiave e decifrare il messaggio. Sfortunatamente per la TCP, praticamente tutti i messaggi di 3 byte sono validi e quindi il computer comincia a stampare le istruzioni per l'invio della merce. Anche se può sembrare strano che un cliente ordini 837 altalene per bambini e 540 scatole di sabbia, per quanto ne sa il computer, il cliente potrebbe anche essere in procinto di aprire una catena in franchising di parchi giochi. In questo modo un intruso attivo (l'ex impiegata) può causare una grande quantità di danni, anche se non riesce a decifrare i messaggi che vengono generati dal suo computer.

Questo problema può essere risolto aggiungendo della ridondanza a tutti i messaggi. Per esempio, se i messaggi degli ordini vengono estesi a 12 byte, di cui i primi 9 devono essere a 0, allora questo tipo di attacco non funziona più, in quanto l'ex impiegata non riesce più a generare un flusso di messaggi validi. La morale della storia è che tutti i messaggi devono contenere una quantità considerevole di ridondanza, in modo da evitare che intrusi attivi possano generare dati casuali e che questi vengano in seguito interpretati come messaggi validi.

D'altra parte, aggiungere della ridondanza rende più facile il lavoro del criptoanalista che tenta di violare il codice. Supponiamo che l'azienda di vendita per corrispondenza sia molto competitiva e che il suo principale concorrente, "The Sofa Tube", desideri fortemente sapere quante scatole di sabbia la TCP sta vendendo. Per fare questo intercetta le comunicazioni sulla linea telefonica della TCP. Nello schema originale, con i messaggi da 3 byte, la criptoanalista era praticamente impossibile. Infatti, se il criptoanalista tentava delle chiavi a caso, non aveva

comunque modo di capire se l'ipotesi era giusta o sbagliata. Questo perché praticamente tutti i messaggi erano tecnicamente buoni. Con il nuovo schema a 12 byte, per il criptoanalista è più facile capire se un messaggio è valido oppure no.

#### *Principio crittografico 1: i messaggi devono contenere ridondanza.*

In altre parole, il destinatario, dopo aver decifrato il messaggio, deve essere in grado di stabilire la validità con un semplice esame del contenuto o con un breve calcolo. Questo tipo di ridondanza è necessario per prevenire gli attacchi degli intrusi attivi, che consistono nell'inviare dei messaggi generati a caso per ingannare il destinatario, che li decifra come se fossero validi. Il rovescio della medaglia è che la stessa ridondanza rende più semplice, per gli intrusi passivi, il compito di decifrare i messaggi. Inoltre, la ridondanza non dovrebbe mai presentarsi nella forma di  $n$  valori a zero posti all'inizio o alla fine del messaggio: passare messaggi così formati attraverso un algoritmo crittografico rende il risultato più regolare, facilitando quindi il lavoro dei criptoanalisti. La soluzione migliore consiste nell'usare dei codici hash crittografici, come vedremo in seguito.

Tornando per un attimo alla crittografia quantistica, vediamo che la ridondanza ha un ruolo anche in quel caso. Visto che Trudy ha intercettato dei fotoni, alcuni bit nei blocchi monouso di Bob sono sbagliati. Bob ha bisogno di avere ridondanza nei messaggi in arrivo per capire che si sono verificati degli errori. Una forma molto semplice di ridondanza consiste nel ripetere il messaggio due volte. Se le due copie non sono identiche, Bob individua un'anomalia: o c'è molto rumore nella trasmissione in fibra, oppure qualcuno sta ascoltando la comunicazione. Ovviamente, trasmettere il messaggio due volte è un'esagerazione, un metodo più efficiente per identificare gli errori è dato dall'uso della codifica di Hamming o di Reed-Solomon. Da questa discussione deve essere chiaro che l'uso di qualche forma di ridondanza è necessario per distinguere i messaggi validi da quelli che non lo sono, specialmente se ci troviamo di fronte a intrusi attivi.

#### Attualità

Il secondo principio crittografico afferma che è necessario avere la possibilità di verificare che ogni messaggio ricevuto sia attuale, cioè trasmesso di recente. Questo serve per evitare che intrusi attivi possano inviare messaggi vecchi spacciandoli per nuovi. Se non ci sono misure di questo tipo, la nostra ex impiegata potrebbe ascoltare la linea telefonica di TCP e mandare delle semplici ripetizioni dei messaggi validi che ha intercettato. Possiamo quindi enunciare il secondo principio.

#### *Principio crittografico 2: sono necessari dei metodi per prevenire gli attacchi di tipo ripetizione.*

Una possibilità consiste nell'includere un timestamp (data e ora) valido, per esempio, solamente per 10 secondi. Il destinatario può quindi tenere traccia dei messaggi degli ultimi 10 secondi, per confrontarli con i messaggi in arrivo e filtrare i duplicati. I messaggi più vecchi di 10 secondi vengono scartati, in quanto troppo vecchi. Altri metodi, oltre a questo del timestamp, verranno discussi nel seguito.

## 8.2 Algoritmi a chiave simmetrica

La crittografia moderna usa le stesse idee base della crittografia classica (come la trasposizione e la sostituzione), ma con un'enfasi diversa. Tradizionalmente i crittografi hanno sempre usato degli algoritmi semplici. Al giorno d'oggi è vero il contrario: l'idea è quella di rendere l'algoritmo di cifratura così complesso e involuto per fare in modo che, anche se il criptoanalista avesse a disposizione una vasta quantità di testo cifrato di sua scelta, non riesca comunque a capirci nulla senza avere la chiave crittografica.

La prima classe di algoritmi di cifratura che studieremo in questo capitolo va sotto il nome di **algoritmi a chiave simmetrica**, in quanto usano la stessa chiave per cifrare e decifrare i messaggi. La Figura 8.2 illustra l'uso di un algoritmo a chiave simmetrica. In particolare ci concentreremo sui **cifrari a blocco**, che prendono un blocco di  $n$  bit dal testo in chiaro e li trasformano usando la chiave in  $n$  bit cifrati.

Gli algoritmi crittografici possono essere implementati in hardware (per aumentarne la velocità) o nel software (per aumentarne la flessibilità). Anche se nel nostro discorso ci concentreremo principalmente sugli algoritmi e i protocolli, che sono indipendenti dall'implementazione, può essere interessante spendere alcune parole su come si costruisce l'hardware crittografico. La trasposizione e la sostituzione si possono implementare con semplici circuiti elettrici. La Figura 8.6(a) mostra un dispositivo, noto come **P-box** (abbreviazione di "scatola di permutazione"), usato per la trasposizione con input a 8 bit. Se i bit vengono numerati dall'alto al basso con 01234567, l'output della P-box è 24506713. La P-box può compiere qualunque tipo di trasposizione, semplicemente impostando in modo opportuno i collegamenti interni. Inoltre, la P-box lavora praticamente alla velocità della luce, in quanto non deve compiere nessun calcolo, ma solamente propagare il segnale in ingresso. La sua struttura segue la legge di Kerckhoff: l'intruso sa che il principio generale consiste nel permutare i bit, però non sa come vengono permutati i bit, cioè non conosce la chiave.

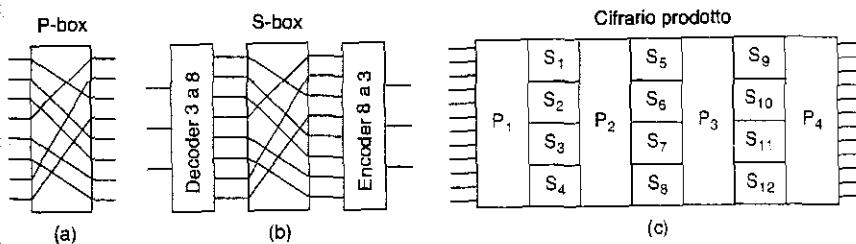


Figura 8.6. Blocchi elementari dei cfrari prodotto. (a) P-box. (b) S-box. (c) Prodotto.

Le sostituzioni sono fatte da **S-box**, come mostrato nella Figura 8.6(b). In questo esempio abbiamo in input un testo in chiaro di 3 bit, e in output un testo cifrato di 3 bit. I 3 bit dell'input scelgono una delle 8 linee che escono dal primo stadio e li impostano a 1, tutte le altre linee sono a 0. Il secondo stadio è una P-box. Il terzo stadio codifica le linee di input selezionate riportandole in binario. Con i collegamenti mostrati nella figura, se i numeri

da 0 a 7 arrivano in input uno alla volta (01234567), la sequenza di output sarà: 24506713. In altre parole, 0 è stato rimpiazzato da 2, 1 da 4, ecc. Per quanto esposto, cambiando i collegamenti interni della P-box che si trova dentro alla S-box possiamo ottenere qualunque tipo di sostituzione. Inoltre, un dispositivo di questo tipo può essere costruito in hardware per raggiungere notevoli velocità, visto che la codifica e decodifica hanno dei ritardi di gate inferiori al nanosecondo, mentre la propagazione nella P-box può essere anche inferiore al picosecondo.

La vera potenza di questi elementi base si vede quando si collegano in cascata una serie di P-box e S-box per formare un **cifrario prodotto**, come mostrato nella Figura 8.6(c). In questo esempio, 12 linee di input subiscono nel primo stadio ( $P_1$ ) una trasposizione. In teoria sarebbe possibile avere, come secondo stadio, una S-box che trasforma univocamente un qualunque numero a 12 bit in un altro a 12 bit. Il punto è che questo tipo di dispositivo richiederebbe  $2^{12} = 4.096$  incroci di fili nel suo stadio intermedio. Più praticamente, si divide l'input in quattro gruppi di 3 bit ciascuno, ognuno dei quali è soggetto a una sostituzione indipendente da quella degli altri. Anche se questo metodo è meno generale, continua a essere molto potente. Includendo un numero sufficiente di stadi nel cifrario prodotto, l'output generato diventa una funzione estremamente complicata dell'input.

I cifrari prodotto che operano su input di  $k$  bit per produrre output di  $k$  bit sono molto comuni. Tipicamente,  $k$  vale 64 o 256 e l'implementazione hardware consta di almeno 18 stadi fisici, al posto dei 7 mostrati nella Figura 8.6(c). Un'implementazione software viene programmata con un loop di almeno 8 iterazioni. Ogni iterazione opera una sostituzione di tipo S-box su dei sottoblocchi dei dati di grandezza pari a 64 o 256 bit, seguita da una permutazione che mescola gli output delle S-box. Spesso si trova una permutazione speciale all'inizio e alla fine del calcolo. Nella letteratura, queste iterazioni sono dette **round**.

### 8.2.1 DES (Data Encryption Standard)

Nel gennaio 1977 il governo degli Stati Uniti adottò come standard per le informazioni non riservate un cifrario prodotto sviluppato da IBM. Questo cifrario, **DES (Data Encryption Standard, standard per la cifratura dei dati)**, fu quindi largamente utilizzato anche dall'industria per i prodotti riguardanti la sicurezza. DES non è più sicuro nella sua forma originale, ma in una forma modificata è ancora utile.

Uno schema del funzionamento di DES è mostrato nella Figura 8.7(a). Il testo in chiaro viene cifrato in blocchi di 64 bit, che generano 64 bit di testo cifrato. L'algoritmo è parametrizzato da una chiave a 56 bit e ha 19 stadi distinti. Il primo stadio è indipendente dalla chiave e consiste nella trasposizione dei 64 bit del testo in chiaro. L'ultimo stadio è esattamente l'inverso del primo. Il penultimo stadio consiste nello scambiare i 32 bit più a sinistra con i 32 più a destra. I rimanenti 16 stadi sono funzionalmente identici, ma sono parametrizzati da diverse funzioni della chiave. L'algoritmo è stato progettato per permettere la decifrazione con la stessa chiave usata per la cifratura, proprietà necessaria in un algoritmo a chiave simmetrica. I passi vengono semplicemente percorsi in ordine inverso nei due casi.

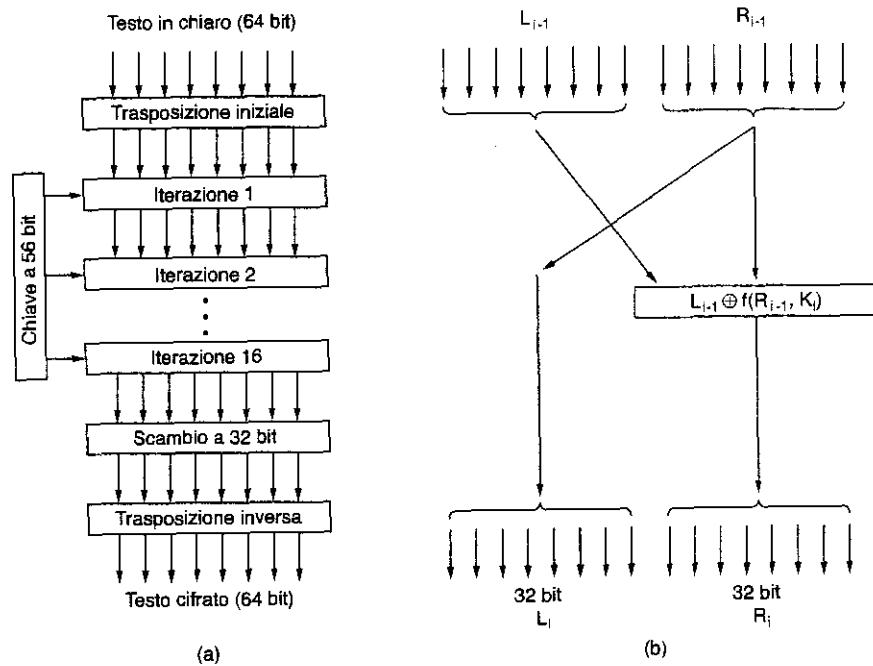


Figura 8.7. Standard per la cifratura dei dati. (a) Struttura generale. (b) Dettaglio di una iterazione. La + inclusa in un cerchio significa OR esclusivo.

Il funzionamento di uno degli stadi intermedi è illustrato nella Figura 8.7(b). Ogni stadio prende due ingressi a 32 bit e produce due uscite a 32 bit. L'output di sinistra è semplicemente una copia dell'output di destra. L'output di destra consiste nello XOR bit per bit dell'input di sinistra con una funzione dell'input di destra e della chiave per questo stadio,  $K_i$ . Tutta la complessità è in questa funzione.

La funzione consiste di quattro passi eseguiti in sequenza. Come prima cosa si costruisce un numero di 48 bit,  $E$ , espandendo i 32 bit di destra,  $R_{i-1}$ , usando una regola fissa di trasposizione e duplicazione. Nel secondo passo si esegue l'operazione di XOR fra  $E$  e  $R_{i-1}$ . Questo output viene diviso in otto gruppi di 6 bit ciascuno, ognuno dei quali viene passato attraverso una diversa S-box. Ognuno dei 64 possibili input alla S-box viene mappato in un output di 4 bit. Infine si fanno passare gli 8 gruppi di 4 bit attraverso una P-box.

Si usa una chiave differente per ognuna delle 16 iterazioni. Una trasposizione di 56 bit è applicata alla chiave prima di cominciare a operare con l'algoritmo. Subito prima di ogni iterazione si divide la chiave in due unità di 28 bit, ciascuna delle quali viene ruotata a sinistra di un numero di bit che dipende dal numero dell'iterazione. Ogni valore di  $K_i$  è determinato da questa chiave ruotata, applicandogli un'ulteriore trasposizione di 56 bit. A ogni round si estrae e permuta un diverso sottoinsieme di 48 bit presi dalla chiave a 56 bit.

Una tecnica che viene talvolta utilizzata per rafforzare DES si chiama **sbiancamento** (*whitening*). Consiste nell'operare lo XOR di una chiave casuale a 64 bit su ogni blocco di dati del testo in chiaro, usare il risultato come input per il DES, e infine eseguire lo XOR di una seconda chiave a 64 bit; il risultato finale è il testo cifrato da trasmettere. Lo sbiancamento si rimuove facilmente eseguendo le operazioni inverse (se il destinatario ha le due chiavi di sbiancamento). Questa tecnica rende la ricerca completa nello spazio delle chiavi molto più lunga, poiché in definitiva aggiunge nuovi bit alla chiave stessa. Notiamo che per ogni blocco viene utilizzata la stessa chiave di sbiancamento (cioè esiste una sola chiave di sbiancamento).

DES è stato circondato da controversie sin dalla sua nascita. Fu basato su un cifrario sviluppato e brevettato da IBM chiamato Lucifer, con la differenza che il cifrario IBM usava una chiave di 128 bit al posto dei 56 di DES. Quando il governo federale degli Stati Uniti volle creare un cifrario standard per i documenti non riservati, "invitò" IBM per "discutere" la questione con NSA, la parte del governo degli Stati Uniti che si occupa di forzare i cifrari e che è anche il più grande datore di lavoro al mondo per matematici e criptoanalisti. NSA è così segreta che per scherzo si dice:

Domanda: Che cosa significa l'acronimo NSA?

Risposta: No Such Agency (non esiste questa agenzia).

A parte gli scherzi, NSA sta per *National Security Agency* (Agenzia per la Sicurezza Nazionale).

Dopo questa serie di discussioni, IBM ridusse la lunghezza della chiave da 128 a 56 bit e decise di tenere segreto il processo seguito per sviluppare DES. Molti sospettarono che la lunghezza della chiave fosse stata ridotta per garantire a NSA la possibilità di forzare DES, cosa che sarebbe rimasta fuori dalla portata di organizzazioni con un budget inferiore. Riguardo alla segretezza della struttura, si supponeva che questa nascondesse una back door che poteva rendere ancora più facile la forzatura di DES da parte di NSA. Le polemiche furono ulteriormente alimentate, quando un dipendente di NSA chiese in modo discreto a IEEE di cancellare una conferenza sulla crittografia già programmata. NSA ha sempre negato tutto.

Nel 1977, due ricercatori di Stanford, Diffie e Hellman progettarono una macchina per forzare DES e stimarono che potesse essere costruita con circa 20 milioni di dollari. Dati un piccolo pezzo di testo in chiaro e il testo cifrato corrispondente, questa macchina poteva trovare la chiave con un ricerca completa nello spazio delle  $2^{56}$  chiavi in meno di 1 giorno. Al giorno d'oggi una macchina di questo tipo costerebbe molto meno di 1 milione di dollari.

### DES triplo

Fin dal 1979 IBM si rese conto che la chiave di DES era troppo corta e trovò un modo per incrementarla in modo efficace, usando la cifratura tripla (Tuchman, 1979). Il metodo scelto, che è stato poi incorporato nello standard internazionale 8732, è illustrato nella Figura 8.8. Vengono usate due chiavi e tre stadi. Nel primo stadio, il testo in chiaro viene cifrato con DES nel modo solito usando la chiave  $K_1$ . Nel secondo stadio, DES viene usato in modalità di decifrazione usando la chiave  $K_2$ . Infine un'altra cifratura viene fatta con la chiave  $K_1$ .

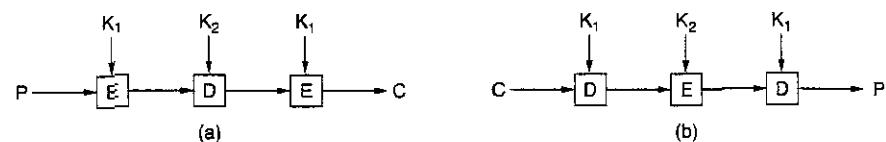


Figura 8.8. (a) Cifratura tripla con DES. (b) Decifrazione.

Questa struttura fa immediatamente nascere due domande. La prima è: perché vengono usate due chiavi e non tre? La seconda è: perché si usa EDE (*Encrypt Decrypt Encrypt*) al posto di EEE (*Encrypt Encrypt Encrypt*)? La ragione per cui vengono usate due chiavi è che anche il più paranoico dei crittografi ritiene che 112 bit offrano una lunghezza adeguata per le normali applicazioni commerciali, ai giorni nostri (fra i crittografi la paranoia è considerata un pregio e non un difetto). Usare 168 bit vorrebbe dire aggiungere del lavoro per gestire e trasportare un'altra chiave senza ottenere un reale vantaggio.

La sequenza cifra, decifra, cifra è stata scelta per compatibilità all'indietro con i sistemi DES a chiave singola. Sia la funzione di cifratura sia quella di decifrazione sono mappe fra insiemi di 64 bit. Da un punto di vista crittografico, le due mappe sono egualmente forti. Usando EDE, al posto di EEE, un computer che usa la cifratura tripla può parlare con uno che usa la cifratura singola, semplicemente impostando  $K_1 = K_2$ . Questa proprietà permette di introdurre la cifratura tripla per gradi, qualcosa che non interessa troppo ai crittografi accademici, ma che è di grande importanza per IBM e i suoi clienti.

### 8.2.2 AES (*Advanced Encryption Standard*)

Poiché DES stava raggiungendo la fine della sua vita utile, anche nella variante triplo DES, NIST (*National Institute of Standards and Technology*) decise che il governo aveva bisogno di un nuovo standard per le comunicazioni non riservate. NIST è l'agenzia del dipartimento del commercio, che ha il compito di approvare gli standard federali per il governo degli Stati Uniti. NIST conosceva bene le diatribe sorte intorno a DES e sapeva che, se avesse annunciato un nuovo standard, chiunque sapesse qualcosa di crittografia avrebbe pensato automaticamente che NSA aveva una back door per questo standard, e avrebbe potuto leggere tutti i messaggi. In queste condizioni probabilmente nessuno avrebbe usato il nuovo standard, che sarebbe quindi certamente finito nel dimenticatoio. Per questo motivo NIST adottò un approccio diverso dal quello della normale burocrazia governativa e sponsorizzò un concorso crittografico. Nel gennaio 1997 i ricercatori di tutto il mondo furono invitati a inviare delle proposte per un nuovo standard che avrebbe preso il nome di AES (*Advanced Encryption Standard*). Le regole del concorso erano:

1. l'algoritmo doveva essere un cifrario simmetrico a blocchi
2. la struttura doveva essere interamente pubblica

3. doveva supportare chiavi di lunghezza di 128, 192 e 256 bit
4. dovevano essere possibili implementazioni software e hardware
5. l'algoritmo doveva essere pubblico o rilasciato senza vincoli di alcun tipo.

A questo concorso furono presentate quindici proposte serie, discusse in conferenze pubbliche dove i presenti furono incoraggiati a trovarne i possibili difetti. Nell'agosto 1998, NIST selezionò cinque finalisti sulla base dei seguenti criteri: sicurezza, efficienza, semplicità, flessibilità e requisiti di memoria (importante per i sistemi embedded). Ci furono quindi ulteriori conferenze e discussioni critiche su questi cifrari. Durante l'ultima di queste conferenze si fece una votazione non vincolante, con i seguenti risultati:

1. Rijndael (di Joan Daemen e Vincent Rijmen, 86 voti)
2. Serpent (di Ross Anderson, Eli Biham e Lars Knudsen, 59 voti)
3. Twofish (di un team guidato da Bruce Schneier, 31 voti)
4. RC6 (dei laboratori RSA, 23 voti)
5. MARS (di IBM, 13 voti).

Nell'ottobre 2000 NIST annunciò di aver votato per Rijndael, e nel novembre 2001 Rijndael divenne uno standard del governo degli Stati Uniti, pubblicato nel Federal Information Processing Standard FIPS 197. Vista la straordinaria apertura della competizione, le proprietà tecniche di Rijndael e il fatto che il team vincente fosse una coppia di giovani crittografi Belgi (che difficilmente possono aver costruito una back door solo per far contenta NSA), ci si può attendere che Rijndael diventerà lo standard crittografico dominante nel mondo per almeno un decennio. Il nome Rijndael, pronunciato Rain-dòl (all'incirca), deriva dal cognome dei due autori Rijmen + Daemen.

Rijndael supporta blocchi e chiavi con dimensioni da 128 a 256 bit in passi di 32 bit. La lunghezza della chiave e del blocco possono essere scelti indipendentemente. AES, invece, specifica che la dimensione del blocco deve essere 128 bit, mentre la chiave può essere 128, 192 o 256 bit. Sembra improbabile che vengano mai usate chiavi da 192 bit, quindi in pratica AES ha due varianti: 128 bit di blocco con chiave a 128 bit e 128 bit di blocco con chiave a 256 bit.

Nella nostra discussione tratteremo solamente il caso 128/128, in quanto è quello che probabilmente diventerà lo standard commerciale. Una chiave di 128 bit genera uno spazio delle chiavi con  $2^{128} \approx 3 \times 10^{38}$  chiavi. Anche se NSA riuscisse a costruire una macchina con 1 miliardo di processori paralleli, ognuno in grado di valutare una chiave per picosecondo, ci vorrebbero  $10^{10}$  anni per eseguire una ricerca su tutto lo spazio delle chiavi. Nel frattempo il sole avrà esaurito la sua energia, così il risultato dovrà essere letto a lume di candela...

### Rijndael

Da un punto di vista matematico Rijndael è basato sulla teoria dei campi di Galois, e ciò gli fornisce alcune proprietà di sicurezza dimostrabili. Possiamo prendere anche un altro punto di vista e considerare Rijndael come un programma in C, senza entrare nelle sue complessità matematiche.

Analogamente a DES, Rijndael usa sostituzioni e permutazioni attuate in molteplici round. Il numero dei round dipende dalla dimensione della chiave e del blocco: si va da 10 per chiavi a 128 bit con blocchi da 128 bit, fino a 14 per la chiave e il blocco più lunghi. A differenza di DES, tutte le operazioni coinvolgono interi byte, questo per avere un'implementazione efficiente sia hardware sia software. Una versione schematica del codice è riportata nella Figura 8.9.

```
#define LENGTH 16
#define NROWS 4
#define NCOLS 4
#define ROUNDS 10
typedef unsigned char byte;
/* # bytes in un blocco dati o chiave */
/* numero di righe in uno stato */
/* numero di colonne in uno stato */
/* numero di iterazioni */
/* unsigned 8-bit integer */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
 int r; /* indice del loop */
 byte state[NROWS][NCOLS]; /* stato attuale */
 struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1]; /* chiavi del round */
 /* costruisce le chiavi del round */
 /* init stato attuale */
 /* XOR di chiave su stato */
 /* applica S-box ad ogni byte */
 /* ruota riga i di i byte */
 /* funzione mix */
 /* XOR di chiave su stato */
 /* rende il risultato */

 expand-key(key, rk);
 copy-plaintext-to-state(state, plaintext);
 xor-roundkey-into-state(state, rk[0]);
 for (r = 1; r <= ROUNDS; r++) {
 substitute(state);
 rotate-rows(state);
 if (r < ROUNDS) mix-columns(state);
 xor-roundkey-into-state(state, rk[r]);
 }
 copy-state-to-ciphertext(ciphertext, state);
}
/* rende il risultato */
```

Figura 8.9. Schema di Rijndael.

La funzione *rijndael* ha tre parametri: *plaintext*, un array di 16 byte che contiene i dati in input, *ciphertext*, un array di 16 byte per l'output del testo cifrato, e *key* (la chiave a 16 byte). Durante il calcolo lo stato corrente dei dati è mantenuto in array di byte, *state*, la cui

dimensione è  $NROWS \times NCOLS$ . Per i blocchi da 128 bit questo array è di  $4 \times 4$  byte; 16 byte permettono di memorizzare tutti i 128 bit dei dati.

L'array *state* è inizializzato in modo da contenere il testo in chiaro e viene poi modificato in ogni passo del calcolo. In alcuni passi viene eseguita una sostituzione byte per byte. In altri, i byte sono permutati all'interno dell'array. Vengono usate anche altre trasformazioni. Alla fine, il contenuto di *state* contiene il dato cifrato ritornato dalla funzione.

Il codice comincia espandendo la chiave in 11 array con dimensioni pari a quella di *state*, memorizzati in *rk* che è un array di struct. Ogni struct contiene un array di state. Uno di questi verrà usato all'inizio del calcolo, mentre gli altri 10 saranno usati a turno, uno per ogni round. Il calcolo delle chiavi per ogni round a partire dalla chiave crittografica è troppo complicato per descriverlo qui. Basti sapere che le chiavi sono prodotte dalla ripetizione di rotazioni e XOR fra vari gruppi di bit della chiave. Per i dettagli vedi (Daemen e Rijmen, 2002).

Il passo successivo consiste nel copiare il testo in chiaro nell'array *state* in modo che possa essere elaborato a ogni round. Viene copiato in ordine di colonna, con i primi 4 byte che entrano nella colonna 0, i successivi 4 nella colonna 1, ecc. Colonne e righe sono numerate a partire da 0, mentre i round sono numerati a partire da 1. La configurazione iniziale dei 12 array di dimensione  $4 \times 4$  è illustrata nella Figura 8.10.

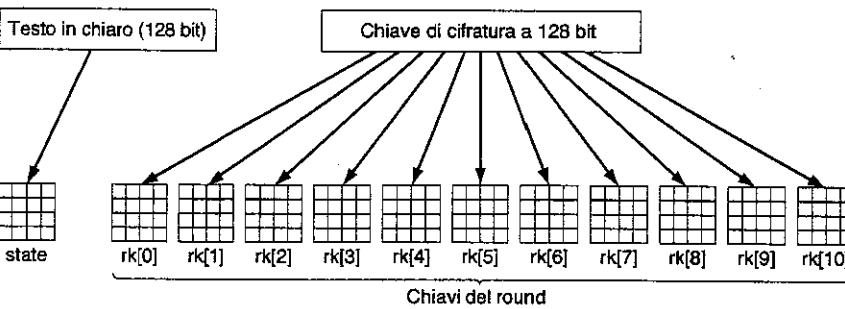


Figura 8.10. Creazione degli array state ed rk.

È ancora un passo prima dell'inizio della parte principale dell'algoritmo: viene calcolato XOR byte per byte fra *rk[0]* e *state*. In altre parole, ognuno dei 16 byte di *state* viene riempito con lo XOR fra se stesso e il byte corrispondente in *rk[0]*.

Nella parte centrale dell'algoritmo si esegue un ciclo di 10 iterazioni (una per ogni round) che trasformano *state*. Ogni round si articola in quattro passi: nel primo viene fatta una sostituzione di *state* byte per byte. Ogni byte è usato a turno come indice di una S-box, e riempito con il contenuto della S-box. Questo passo è una semplice cifratura a sostituzione monoalfabetica. Rijndael usa una sola S-box, a differenza di DES che ne utilizza diverse.

Il secondo passo ognuna delle quattro righe è ruotata verso sinistra. La riga 0 viene ruotata di 0 byte (cioè non viene cambiata), la riga 1 viene ruotata di 1 byte, la riga 2 di 2 byte

e la riga 3 di 3 byte. Questo passo diffonde i contenuti dei dati correnti all'interno del blocco, in maniera simile alle permutazioni della Figura 8.6.

Il passo 3 mescola ogni colonna indipendentemente dalle altre. Il mescolamento è fatto usando una matrice di moltiplicazione, nella quale ogni nuova colonna è il prodotto della vecchia colonna per una matrice costante. La moltiplicazione usa il campo finito di Galois  $GF(2^8)$ . L'operazione, in apparenza complicata, viene in realtà svolta da un algoritmo che permette di calcolare i valori delle nuove colonne operando due ricerche in tabella e tre operazioni di XOR (Daemen e Rijmen, 2002, Appendice E). Infine, il passo 4 esegue l'operazione XOR tra la chiave per l'iterazione corrente e l'array *state*.

Ogni passo è reversibile, perciò la decifrazione può essere fatta semplicemente eseguendo l'algoritmo all'indietro; esiste però un trucco per eseguire la decifrazione usando l'algoritmo di cifratura, ma con tabelle differenti.

L'algoritmo è stato progettato non solo per fornire una grande sicurezza, ma anche per essere molto veloce. Una buona implementazione software su un computer a 2 GHz dovrebbe raggiungere una velocità di cifratura pari a 700 Mbps, che è adeguata per cifrare più di 100 video MPEG-2 in tempo reale. Le implementazioni hardware sono ancora più veloci.

### 8.2.3 Modalità di cifratura

Nonostante la sua complessità, AES (o DES, o ogni altro tipo di cifrario a blocchi) è essenzialmente un cifrario a sostituzione monoalfabetica che usa dei caratteri lunghi (caratteri da 128 bit per AES e da 64 bit per DES). Quando lo stesso testo in chiaro viene preso come input, abbiamo lo stesso testo cifrato in output. Se cifriamo il testo in chiaro *abcdefg* per 100 volte con la stessa chiave DES, otteniamo 100 volte l'identico testo cifrato. Un intruso può utilizzare questa proprietà per cercare di forzare il cifrario.

#### Modalità ECB (Electronic Code Book)

Per vedere come questa proprietà del cifrario a sostituzione monoalfabetica possa essere usata per forzare parzialmente il cifrario stesso, discutiamo un esempio usando il (triplo) DES, in quanto è più semplice rappresentare blocchi di 64 bit piuttosto che 128 bit; comunque AES ha esattamente lo stesso tipo di problema. La modalità d'impiego più ovvia del DES per cifrare un lungo pezzo di testo in chiaro consiste nel prendere il testo e suddividerlo in blocchi di 8 byte (64 bit), che vengono poi cifrati uno dopo l'altro usando la stessa chiave. L'ultimo pezzo di testo in chiaro, se necessario, viene riempito fino a far gli raggiungere la lunghezza di 64 bit. Questa tecnica è nota come **modalità ECB** (*Electronic Code Book*, libro elettronico dei codici) per analogia con i vecchi libri dei codici dove si trovavano gli elenchi delle parole in chiaro seguite dal testo cifrato (di solito un numero decimale di cinque cifre).

Nella Figura 8.11 è mostrato l'inizio di un file che contiene l'elenco dei bonus annuali che un'azienda ha deciso di dare ai suoi dipendenti. Il file è costituito da una serie di record a 32 bit, uno per ogni dipendente, nel formato mostrato nella figura: 16 byte per il nome, 8

byte per la mansione e 8 byte per il bonus. Ognuno dei sedici blocchi da 8 byte (numerati da 0 a 15) è cifrato con il (triplo) DES.

Nome	Ruolo	Bonus
Adams, Lesile	Clerk	\$1,110
Black, Robin	Boss	\$500,000
Collins, Kim	Manager	\$110,000
Davies, Bobbie	Janitor	\$1,55

Byte → 16 ← 8 ← 8 ← 8 →

Figura 8.11. Il testo in chiaro di un file cifrato come 16 blocchi DES.

Lesile ha avuto di recente una brutta discussione con il capo, quindi non si aspetta di ricevere un gran bonus. Kim, al contrario, è il favorito del capo, come sanno tutti in azienda. Lesile è in grado di accedere al file solo dopo che è stato cifrato, ma prima che venga spedito alla banca. È possibile che Lesile riesca a sistemare questa situazione di squilibrio avendo a disposizione solo il file cifrato?

Non è difficile: Lesile deve solamente fare una copia del dodicesimo blocco di testo cifrato (che contiene il bonus di Kim) e usarlo per rimpiazzare il quarto blocco cifrato (che contiene il bonus di Lesile). Anche senza sapere che cosa contiene il dodicesimo blocco, Lesile si può comunque attendere un Natale più allegro quest'anno. Copiare l'ottavo blocco di testo è un'altra possibilità, ma è più facile che la frode venga scoperta, e inoltre Lesile non è così attaccata al denaro.

#### Modalità cipher block chaining

Per evitare questo tipo di attacchi si collegano tutti i blocchi cifrati in diversi modi. Così facendo, quando si tenta di rimpiazzare dei pezzi di testo cifrato, come nel caso di Lesile, l'effetto in fase di decifrazione sarà quello di avere dei dati senza significato a partire dal punto in cui è stata operata la sostituzione. Un modo per collegare i blocchi è il **cipher block chaining** (collegamento dei blocchi cifrati). In questo metodo, mostrato nella Figura 8.12, ogni blocco di testo in chiaro è messo in XOR con il precedente blocco cifrato prima di eseguire la cifratura vera e propria. Così facendo a blocchi di testo in chiaro uguali non corrispondono più blocchi di testo cifrato identici, e la cifratura non è più costituita da un cifrario a sostituzione monoalfabetica. Per il primo blocco lo XOR viene calcolato con un blocco di dati casuali, detto IV (*Initialization Vector*, vettore di inizializzazione), che è trasmesso (in chiaro) insieme al testo cifrato.

Nell'esempio della Figura 8.12 possiamo vedere come funziona la modalità cipher block chaining. Cominciamo calcolando  $C_0 = E(P_0 \text{ XOR } IV)$ , seguito da  $C_1 = E(P_1 \text{ XOR } C_0)$  e così via. La decifrazione usa di nuovo lo XOR per rovesciare il processo, cioè per calcolare  $P_0 = IV \text{ XOR } D(C_0)$ , ecc. Notiamo che la cifratura del blocco  $i$  è una funzione di tutto il testo in chiaro a partire dal blocco 0 fino a  $i - 1$ , quindi lo stesso testo in chiaro dà origine a diversi testi cifrati a seconda della sua posizione. Una trasformazione come quella fatta da Lesile

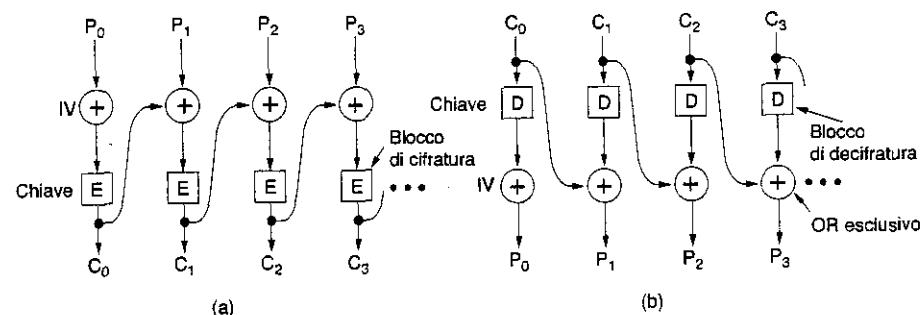


Figura 8.12. Cipher block chaining. (a) Cifratura. (b) Decifrazione.

produrrà due blocchi di dati senza significato, a partire dal campo bonus di Lesile. Un investigatore astuto utilizzerà sicuramente questo dato per cominciare le sue indagini. Il cipher block chaining ha un altro vantaggio: non produce lo stesso testo cifrato a partire da blocchi di testo in chiaro uguali, rendendo quindi la criptoanalisi più difficile. Questa è proprio il motivo principale per cui viene utilizzato.

#### Modalità cipher feedback

La modalità cipher block chaining ha lo svantaggio di richiedere che un intero blocco di 64 bit di testo cifrato arrivi a destinazione prima che la decifrazione possa cominciare. Questo tipo di comportamento non è accettabile per i terminali interattivi, dove gli utenti possono digitare delle linee anche più corte di otto caratteri per poi fermarsi in attesa di una risposta. Con la cifratura byte per byte viene usata la modalità **cipher feedback** con (triplo) DES, come mostrato nella Figura 8.13. Per AES l'idea è esattamente la stessa, con la sola differenza che si utilizza un registro di shift a 128 bit. Nella figura lo stato della macchina per cifrare è mostrato dopo che i byte da 0 a 9 sono già stati cifrati e trasmessi. Quando arriva il byte 10 del testo in chiaro, come nella Figura 8.13(a), l'algoritmo DES agisce sui 64 bit del registro di shift per generare 64 bit di testo cifrato. Il byte più a sinistra del testo cifrato viene estratto, si calcola lo XOR con  $P_{10}$  e il risultato è trasmesso sulla linea di comunicazione. Inoltre il registro di shift viene spostato a sinistra di 8 bit, quindi  $C_2$  esce dal registro mentre  $C_{10}$  entra nella posizione lasciata vacante da  $C_9$ . Notiamo che il contenuto del registro di shift dipende dall'intera storia precedente del testo in chiaro, quindi una combinazione di caratteri ripetuta nel testo in chiaro darà origine a valori differenti nel testo cifrato. Come nel caso del cipher block chaining, per iniziare la comunicazione serve un vettore d'inizializzazione.

Con la modalità cipher feedback la decifrazione funziona come la cifratura. In particolare, il contenuto del registro di shift viene *cifrato* e non *decifrato*, così il byte che viene calcolato in XOR con  $C_{10}$  per ottenere  $P_{10}$  è lo stesso che, in origine, veniva calcolato in XOR con  $P_{10}$  per generare  $C_{10}$ . La decifrazione funziona correttamente se i due registri di shift rimangono identici, come illustrato nella Figura 8.13(b).

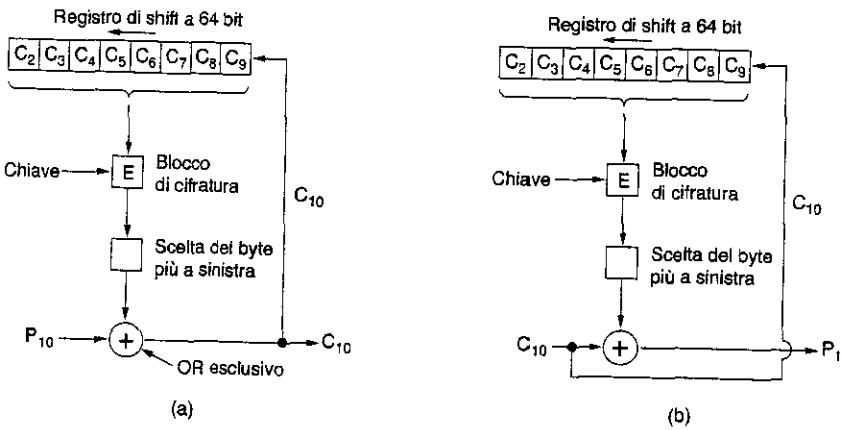


Figura 8.13. Modalità cipher feedback. (a) Cifratura. (b) Decifratura.

problema della modalità cipher feedback si verifica quando un bit del testo cifrato viene accidentalmente invertito durante la trasmissione: gli 8 byte che vengono decifrati mentre nel registro di shift si trova il byte corrotto saranno pure loro corrotti. Una volta che il byte corrotto è uscito dal registro di shift, la decifrazione procederà di nuovo in modo corretto, quindi l'effetto dell'inversione di un singolo bit risulta relativamente localizzata e non riesce a rovinare il resto del messaggio, ma comunque rovina tanti byte quanto il registro di shift.

#### Modalità stream cipher

tono delle applicazioni per cui è inaccettabile il fatto che un errore di trasmissione di uno bit possa rovinare 64 bit di testo in chiaro. Per questo tipo di applicazioni esiste una quarta opzione: la modalità **stream cipher**. Questa modalità funziona cifrando un vettore di inizializzazione con una chiave crittografica per ottenere un blocco in uscita. Questo blocco viene cifrato per produrre un secondo blocco in uscita, quindi si procede analogamente con il terzo, ecc. La sequenza (di lunghezza arbitraria) di blocchi cifrati in uscita, chiamata il **keystream** (flusso delle chiavi), viene utilizzata come un blocco di cifratura e applicata in XOR sul testo in chiaro per ottenere il testo cifrato, come mostrato nella Figura 8.14(a). Notiamo che IV è usato solamente nel primo passo, nei passi successivi l'output è cifrato. Notiamo anche che il keystream è indipendente dai dati, così che può essere calcolato in anticipo, se necessario, ed è anche immune da errori di trasmissione. La decifrazione è mostrata nella Figura 8.14(b).

La cifratura viene eseguita generando lo stesso keystream dal lato del ricevente. Visto che il keystream dipende solo da IV e dalla chiave, non è sensibile agli errori di trasmissione del testo cifrato. Quindi, un errore di 1 bit nel testo cifrato trasmesso genera solamente un errore di 1 bit nel testo decifrato.

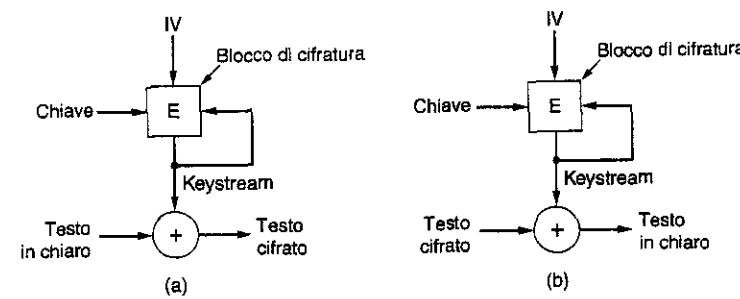


Figura 8.14. Stream cipher. (a) Cifratura. (b) Decifratura.

È essenziale che nella modalità stream cipher non venga mai riutilizzata la coppia (chiave, IV), perché questo vorrebbe dire generare più volte lo stesso keystream. L'uso ripetuto dello stesso keystream espone il testo cifrato ad **attacchi di tipo keystream riutilizzato**. Immaginiamo che un blocco di testo in chiaro,  $P_0$ , sia cifrato con un keystream per ottenere  $P_0 \oplus K_0$ . Più tardi si utilizza ancora lo stesso keystream per cifrare un secondo blocco di testo  $Q_0$ , calcolando  $Q_0 \oplus K_0$ . Un intruso che riesca a catturare entrambi questi blocchi cifrati può semplicemente calcolare lo XOR dei due testi in chiaro:  $P_0 \oplus Q_0$ , il che elimina la chiave. Se uno dei due blocchi è noto, oppure può essere indovinato, l'intruso ha trovato il valore dell'altro blocco. Inoltre, lo XOR di due testi in chiaro può essere attaccato usando le proprietà statistiche del messaggio. Prendendo per esempio i messaggi in inglese, il carattere più comune nel flusso di dati sarà probabilmente lo XOR di due spazi, seguito dallo XOR di uno spazio e della lettera "e", ecc. In conclusione, avendo a disposizione lo XOR di due testi in chiaro, il criptoanalista ha eccellenti probabilità di dedurli entrambi.

#### Modalità contatore

Un problema comune a tutte le modalità, fatta eccezione per electronic code book, è quello di rendere impossibile un accesso casuale ai dati. Per esempio, supponiamo che un file sia trasmesso tramite la rete e poi memorizzato su disco in un formato cifrato; ciò potrebbe essere utile se il computer del ricevente è un portatile a rischio di furto. Memorizzare i file crittati riduce ampiamente il potenziale danno dovuto alla fuga di notizie che avviene se il computer cade nelle mani sbagliate. Spesso si accede ai file su disco in un ordine non sequenziale (random), specialmente quando si tratta di database. Con un file cifrato usando il cipher block chaining, l'accesso random a un blocco richiede che prima siano decifrati tutti i blocchi precedenti, un'operazione troppo onerosa. Per questo motivo è stata inventata un'altra modalità: la **modalità contatore**, mostrata nella Figura 8.15. In questo caso il testo in chiaro non viene cifrato.

to direttamente; si cifra invece un vettore d'inizializzazione con una costante, e il risultato è messo in XOR con il testo in chiaro. Incrementando di 1 il vettore d'inizializzazione a ogni blocco, diventa facile riuscire a decifrare un blocco in qualunque posizione si trovi, senza dover decifrare prima tutti i suoi predecessori.

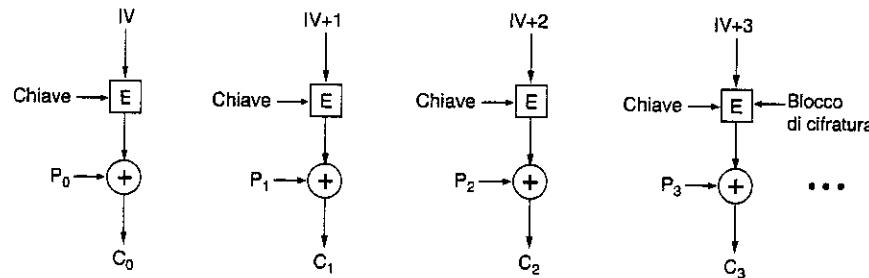


Figura 8.15. Cifratura in modalità contatore (counter).

Anche se la modalità contatore è utile, ha una debolezza che vogliamo esplicitare. Supponiamo che la stessa chiave,  $K$ , venga usata di nuovo in futuro (con un testo in chiaro differente, ma con lo stesso IV) e un intruso riesca a ottenere tutto il testo cifrato per entrambi i casi. Essendo il keystream lo stesso nei due casi, il sistema è esposto ad attacchi di tipo keystream riutilizzato, che abbiamo discusso a proposito della modalità stream cipher. Il criptoanalista deve fare un semplice XOR dei due testi cifrati, e il risultato sarà uguale allo XOR dei due testi in chiaro, con soppressione di tutta la protezione crittografica. Questa debolezza non significa che la modalità contatore sia sbagliata, ma che vettore d'inizializzazione e chiave vanno scelti indipendentemente e in modo casuale. Anche se la stessa chiave viene usata due volte accidentalmente, quando l'IV è differente nei due casi il testo in chiaro è salvo.

#### 8.2.4 Altri cifrari

DES e Rijndael sono i più noti algoritmi crittografici a chiave simmetrica. Esistono comunque anche molti altri algoritmi a chiave simmetrica, alcuni di questi sono inseriti all'interno di diversi prodotti. Alcuni dei più comuni sono elencati nella Figura 8.16.

#### 8.2.5 Criptoanalisi

Prima di abbandonare l'argomento della crittografia a chiave simmetrica, è utile menzionare alcuni sviluppi della criptoanalisi. Il primo sviluppo è la **criptoanalisi differenziale** (Biham e Shamir, 1993), una tecnica che può essere usata per attaccare i cifrari a blocco. Si comincia con una coppia di blocchi di testo in chiaro che differiscono solo per un piccolo numero di bit e si guarda attentamente a cosa accade in ogni iterazione interna, mentre la cifratura avanza. In molti casi, alcune combinazioni di bit risultano più comuni di altre. Questo tipo di osservazioni è di aiuto per gli attacchi di tipo probabilistico.

Algoritmo	Autore	Lunghezza della chiave	Commenti
Blowfish	Bruce Schneier	1–448 bits	Vecchio e lento
DES	IBM	56 bits	Troppo debole per l'uso attuale
IDEA	Massey and Xuejia	128 bits	Valido ma brevettato
RC4	Ronald Rivest	1–2.048 bits	Attenzione: alcune chiavi sono deboli
RC5	Ronald Rivest	128–256 bits	Valido ma brevettato
Rijndael	Daemen and Rijmen	128–256 bits	La scelta migliore
Serpent	Anderson, Biham, Knudsen	128–256 bits	Molto robusto
DES triplo	IBM	168 bits	Seconda scelta
Twofish	Bruce Schneier	128–256 bits	Molto robusto; ampiamente usato

Figura 8.16. Alcuni noti algoritmi crittografici a chiave simmetrica.

Il secondo sviluppo interessante è la **criptoanalisi lineare** (Matsui, 1994). Questa tecnica riesce a forzare il DES solamente con la conoscenza di  $2^{43}$  testi in chiaro. Funziona calcolando lo XOR di certe combinazioni di bit del testo in chiaro con quello cifrato ed esaminandone il risultato. Quando questa operazione è eseguita ripetutamente, metà dei bit dovrebbero essere 0 e metà 1. Capita spesso che i cifrari introducano uno sbilanciamento in una direzione piuttosto che l'altra. Queste piccole differenze possono essere usate per ridurre il fattore lavoro necessario a forzare il cifrario. I dettagli si possono trovare nell'articolo di Matsui.

Il terzo sviluppo consiste nell'analizzare il consumo dell'alimentazione elettrica per trovare le chiavi segrete. I computer tipicamente usano 3 Volt per rappresentare i bit a 1 e 0 Volt per i bit a 0. Quindi l'elaborazione di un 1 richiede più energia dell'elaborazione di uno 0. Se un algoritmo crittografico contiene un ciclo in cui i bit della chiave vengono elaborati in ordine, un possibile attacco consiste nel rimpiazzare il clock principale a  $n$ -GHz con uno più lento (per esempio 100 Hz) e attaccare due connettori a coccodrillo sull'alimentazione della CPU e sulla terra. In questo modo si riesce a misurare la potenza consumata da ogni singola istruzione della macchina. Da questi dati, dedurre la chiave è sorprendentemente semplice. Questo tipo di criptoanalisi può essere sconfitto solamente con una programmazione attenta a livello di linguaggio assembly, per essere sicuri che il consumo elettrico sia indipendente dalla chiave e anche indipendente dalle singole chiavi per ogni round.

Il quarto sviluppo è l'analisi dei tempi. Gli algoritmi crittografici sono pieni di istruzioni per il test dei bit nelle chiavi di ogni iterazione. Se la parte "then" e quella "else" delle istruzioni di test impiegano tempi di esecuzione differenti, è possibile dedurre le chiavi di iterazione rallentando il clock e misurando quanto tempo viene impiegato da ogni passo. Quando si conoscono tutte le chiavi delle iterazioni, è possibile ricostruire anche la chiave originale. Le analisi del consumo e della temporizzazione si possono usare simultaneamente per semplificare il lavoro. Questo tipo di attacco può sembrare bizzarro, ma in realtà si tratta di tecniche molto potenti che riescono a forzare quei cifrari non progettati per resistervi.

### 8.3 Algoritmi a chiave pubblica

Storicamente la distribuzione delle chiavi è sempre stato l'anello debole della maggior parte dei sistemi crittografici. Indipendentemente dalla sua forza, ogni sistema crittografico diventava automaticamente inutile nel momento in cui un intruso riusciva a rubarne la chiave. I criptoanalisti hanno sempre assunto che le chiavi di cifratura e decifrazione fossero le stesse (oppure fossero facilmente derivabili l'una dall'altra). Rimaneva il fatto che la chiave doveva essere distribuita a tutti gli utenti del sistema, quindi sembrava esserci un problema intrinseco nel sistema. Le chiavi dovevano essere protette dai furti; ma al tempo stesso dovevano essere distribuite, e non semplicemente custodite in una cassetta di sicurezza.

Nel 1976 due ricercatori dell'università di Stanford, Diffie e Hellman, proposero un sistema crittografico radicalmente innovativo, in cui le chiavi di cifratura e decifrazione erano differenti e la chiave di decifrazione non era facilmente derivabile da quella di cifratura. Nella loro proposta, l'algoritmo di cifratura  $E$ , e quello di decifrazione  $D$  dovevano soddisfare a tre requisiti:

1.  $D(E(P)) = P$
2. risulta estremamente difficile dedurre  $D$  da  $E$
3.  $E$  non può essere forzato da un attacco di tipo "testo in chiaro a scelta".

Primo requisito afferma che se applichiamo  $D$  a un messaggio cifrato  $E(P)$ , otteniamo il testo in chiaro originale,  $P$ . Senza questa proprietà, il destinatario legittimo del messaggio non riuscirebbe a decifrarlo. Il secondo requisito è autoesplicativo. Il terzo requisito è necessario perché, come vedremo fra poco, gli intrusi possono comunque sperimentare un algoritmo a piacimento. Sotto queste condizioni non c'è ragione per non rendere pubblica la chiave di cifratura.

Il metodo funziona così: una persona, per esempio Alice, vuole ricevere dei messaggi segreti e quindi inventa due algoritmi con le proprietà elencate sopra. L'algoritmo di cifratura e la chiave di Alice vengono resi pubblici, da qui il nome di **crittografia pubblica**. Alice può mettere la sua chiave pubblica, per esempio, nella sua home page sul Web. Saranno la notazione  $E_A$  per indicare l'algoritmo di cifratura parametrizzato dalla chiave pubblica di Alice. Analogamente, l'algoritmo (segreto) di decifrazione parametrizzato dalla chiave privata di Alice sarà  $D_A$ . Bob esegue le operazioni analoghe ad Alice, cioè ha pubblico  $E_B$ , mentre tiene segreto  $D_B$ .

Supponiamo se riusciamo a risolvere il problema di come stabilire una canale sicuro fra Alice e Bob, che non hanno mai avuto nessun contatto in precedenza. Supponiamo che la chiave di cifratura di Alice  $E_A$  e quella di Bob  $E_B$  siano dei file leggibili pubblicamente. Alice invia il suo primo messaggio,  $P$ , calcola  $E_B(P)$  e lo manda a Bob. Bob decifra il messaggio usando la sua chiave segreta  $D_B$  [cioè calcola  $D_B(E_B(P))=P$ ]. Nessun altro riesce a leggere il messaggio cifrato,  $E_B(P)$ , in quanto il sistema di cifratura è forte e quindi risultare troppo difficile derivare  $D_B$  dalla chiave pubblica  $E_B$ . Per inviare una risposta,  $R$ , Bob invia  $E_A(R)$ . Ora Alice e Bob riescono a comunicare in modo sicuro.

Può essere utile una nota sulla terminologia. La crittografia a chiave pubblica richiede che ciascun utente abbia due chiavi: una chiave pubblica, usata dal mondo intero per cifrare i messaggi da mandare a quell'utente, e una chiave privata, che l'utente usa per decifrare i messaggi. Chiameremo queste due chiavi rispettivamente *pubblica* e *privata*, per distinguere dalle chiavi *segrete* usate nella crittografia convenzionale a chiave simmetrica.

#### 8.3.1 RSA

L'unico problema è che dobbiamo trovare degli algoritmi che riescano effettivamente a soddisfare le tre proprietà richieste. Visti i potenziali vantaggi della crittografia a chiave pubblica, molti ricercatori stanno lavorando intensamente su questo argomento e alcuni algoritmi sono già stati pubblicati. Un ottimo metodo è stato scoperto da un gruppo del M.I.T (Rivest et al., 1978), ed è noto con le iniziali RSA dei suoi tre ideatori (Rivest, Shamir e Adleman). È sopravvissuto a tutti i tentativi di forzatura per più di un quarto di secolo, ed è considerato un algoritmo molto robusto. Una gran parte delle applicazioni pratiche nel campo della sicurezza si basa su RSA. Il suo maggior svantaggio è che richiede chiavi di almeno 1.024 bit per poter offrire una buona sicurezza (contro i 128 bit degli algoritmi a chiave simmetrica), il che lo rende abbastanza lento.

RSA si basa su alcuni principi di teoria dei numeri. Faremo riepilogo sull'uso di questo metodo, mentre per i dettagli consigliamo di consultare l'articolo originale.

1. Scegliamo due numeri primi,  $p$  e  $q$  (tipicamente di 1.024 bit).
2. Calcoliamo  $n = p \times q$  e  $z = (p - 1) \times (q - 1)$ .
3. Scegliamo un numero primo relativamente a  $z$ , detto  $d$ .
4. Troviamo  $e$  tale che  $e \times d = 1 \text{ mod } z$ .

Avendo calcolato questi parametri in anticipo, possiamo cominciare la cifratura. Dividiamo in blocchi il testo in chiaro (visto come una stringa di bit), in modo che ogni messaggio in chiaro,  $P$ , cada nell'intervallo  $0 \leq P < n$ . Per fare questo, raggruppiamo il testo in chiaro in blocchi di  $k$  bit, dove  $k$  è il più grande intero per cui vale  $2^k < n$ . Per cifrare il messaggio  $P$ , calcoliamo  $C = P^e \pmod{n}$ . Per decifrare  $C$ , calcoliamo  $P = C^d \pmod{n}$ . Possiamo dimostrare che per ogni  $P$  nell'intervallo specificato, le funzioni di cifratura e decifrazione sono una l'inversa dell'altra. Per cifrare abbiamo bisogno di  $e$  e di  $n$ ; per decifrare invece ci servono  $d$  e  $n$ . Quindi la chiave pubblica consiste nella coppia  $(e, n)$ , mentre la chiave privata consiste in  $(d, n)$ .

La sicurezza del metodo è basata sulla difficoltà di scomporre in fattori i numeri molto grandi. Se il criptoanalista riuscisse a fattorizzare il numero (noto pubblicamente)  $n$ , allora riuscirebbe anche a trovare  $p$  e  $q$ , e da questi anche  $z$ . Con la conoscenza di  $z$ , si possono trovare  $e$  e  $d$  tramite l'algoritmo di Euclide. Fortunatamente i matematici si occupano della fattorizzazione dei grandi numeri da più di 300 anni, e dalla conoscenza acquisita si sa che questo è un problema veramente difficile.

Secondo Rivest e colleghi, fattorizzare un numero di 500 cifre usando la forza bruta richiede  $10^{25}$  anni, anche assumendo di usare il miglior algoritmo noto e un computer che elabora le istruzioni in 1 msec. Se i computer continueranno ad aumentare la loro velocità di un ordine di grandezza ogni decade, passeranno secoli prima di rendere fattibile il problema della fattorizzazione di un numero di 500 cifre. A quel punto però, i nostri discendenti possono semplicemente scegliere  $p$  e  $q$  ancora più grandi.

Un semplice esempio didattico del funzionamento dell'algoritmo RSA è dato nella Figura 8.17. Per questo esempio abbiamo scelto  $p = 3$  e  $q = 11$ , quindi  $n = 33$  e  $e = 20$ . Un valore adeguato per  $d$  è  $d = 7$ , visto che 7 e 20 non hanno divisori comuni. Con queste scelte,  $e$  può essere trovato risolvendo l'equazione  $7e \equiv 1 \pmod{20}$ , il che dà  $e = 3$ . Il testo cifrato,  $C$ , per il testo in chiaro,  $P$ , è dato da  $C = P^e \pmod{n}$ . Il testo cifrato viene decifrato dal ricevente usando la regola  $P = C^d \pmod{n}$ . La figura nell'esempio mostra la cifratura del testo "SUZANNE".

Testo in chiaro (P)		Testo cifrato (C)		Dopo la decifratura	
Simbolico	Numerico	$P^3$	$P^3 \pmod{33}$	$C^7$	$C^7 \pmod{33}$
S	19	6859	28	13492928512	19
U	21	9261	21	1801088541	21
Z	26	17576	20	1280000000	26
A	01	1	1	1	01
N	14	2744	5	78125	14
N	14	2744	5	78125	14
E	05	125	26	8031810176	05

Calcoli del mittente                                    Calcoli del destinatario

Figura 8.17. Un esempio dell'algoritmo RSA.

Siccome per questo esempio abbiamo scelto dei numeri primi molto piccoli,  $P$  deve essere inferiore a 33, quindi ogni blocco di testo può contenere un solo carattere. Il risultato è un cifrario a sostituzione monoalfabetica, il che non è molto impressionante. Se invece, avessimo scelto  $p$  e  $q \approx 2^{512}$ , avremmo avuto  $n \approx 2^{1024}$ , quindi ogni blocco potrebbe essere lungo fino a 1.024 bit, oppure 128 caratteri da 8 bit ciascuno, contro gli 8 caratteri di DES e i 16 di AES.

Notiamo che RSA, per come l'abbiamo descritto, è simile a un algoritmo simmetrico in modalità ECB, cioè blocchi di input uguali originano lo stesso blocco di output: anche in questo caso occorre qualche forma di concatenamento. In pratica, però, la maggior parte dei sistemi crittografici basati su RSA usa la crittografia a chiave pubblica solo una tantum, per distribuire le chiavi segrete da usare con un algoritmo di crittografia simmetrica, come AES o triplo DES. RSA è troppo lento per poterlo usare nella cifratura di grandi volumi di dati, mentre è spesso impiegato per la distribuzione delle chiavi.

### 8.3.2 Altri algoritmi a chiave pubblica

Anche se è largamente utilizzato, RSA non è certamente l'unico algoritmo a chiave pubblica noto. Il primo algoritmo a chiave pubblica fu l'algoritmo "a zaino" (Merkle e Hellman, 1978). L'idea è che qualcuno possiede una gran quantità di oggetti, ognuno con un peso differente. Il proprietario cifra il messaggio, scegliendo casualmente un sottinsieme di oggetti e mettendoli nello zaino. Il peso totale degli oggetti nello zaino è reso pubblico, così come la lista dei possibili oggetti. La lista degli oggetti nello zaino viene tenuta segreta. Con alcune restrizioni aggiuntive, il problema da risolvere per forzare il codice diventa quello di trovare la possibile lista di oggetti con peso stabilito. Si pensava che tale problema fosse computazionalmente inaffrontabile, quindi era stato scelto come base per l'algoritmo a chiave pubblica.

L'inventore dell'algoritmo, Ralph Merkle, era così sicuro che questo algoritmo non poteva essere forzato che offrì una ricompensa di 100 \$ a chiunque riuscisse nell'impresa. Adi Shamir (la "S" in RSA) riuscì in poco tempo a forzare il codice e prese la ricompensa. Senza scomporsi, Merkle rafforzò l'algoritmo e offrì una ricompensa di 1.000 \$ a chi fosse riuscito a forzare questa nuova versione. Ronald Rivest (la "R" in RSA) riuscì dopo poco tempo nell'impresa, e prese la ricompensa. Merkle non osò offrire 10.000 \$ per la successiva versione, così "A" (Leonard Adleman) non poté essere premiato a sua volta. A ogni modo, l'algoritmo a zaino non è considerato sicuro e non viene praticamente più usato. Altri schemi a chiave pubblica sono basati sulla difficoltà di calcolare i logaritmi discreti. Algoritmi di questo tipo sono stati inventati da El Gamal (1985) e Schnorr (1991). Esistono anche degli altri schemi, come quelli basati sulle curve ellittiche (Menezes e Vanstone, 1993), ma le due categorie principali si basano sulla difficoltà di fattorizzare i numeri primi e sul calcolo dei logaritmi discreti modulo un grande numero primo. Questi problemi sono ritenuti veramente difficili da risolvere, inoltre molti matematici ci hanno lavorato sopra per diversi anni senza mai trovare delle vere soluzioni.

### 8.4 Firme digitali

L'autenticità di molti documenti legali, finanziari, ecc., è basata sulla presenza o assenza di una firma autografa autorizzata. Le fotocopie non contano. È necessario trovare un metodo per far sì che i documenti elettronici possano essere firmati in modo non falsificabile, se vogliamo che i messaggi computerizzati rimpiazzino la carta e l'inchiostro. Il problema di trovare un sostituto per la firma autografa è veramente difficile. Essenzialmente, serve un sistema con il quale il mittente possa mandare un messaggio firmato al destinatario in modo che valgano le seguenti condizioni:

1. il destinatario può verificare l'identità dichiarata dal mittente
2. il mittente non può ripudiare in un secondo tempo il contenuto del messaggio
3. il destinatario non può falsificare i messaggi del mittente.

Il primo requisito è necessario, per esempio, per i sistemi finanziari. Quando il computer di un cliente ordina al computer della banca di comprare una tonnellata di oro, il computer della banca deve poter essere sicuro che il computer che ha mandato l'ordine appartenga realmente all'azienda sul cui conto verrà addebitato. In altre parole, la banca deve poter autenticare il cliente (e il cliente deve poter autenticare la banca).

Il secondo requisito è necessario per proteggere la banca dalle frodi. Supponiamo che la banca compri una tonnellata d'oro e, immediatamente dopo, il prezzo dell'oro cali bruscamente. Un cliente disonesto potrebbe far causa alla banca, dicendo che non ha mai mandato l'ordine di comprare dell'oro. Quando la banca porterà il messaggio in tribunale come prova, il cliente negherà di averlo spedito. La proprietà per cui nessuna delle parti contraenti un contratto è in grado in un secondo momento di negare di averlo stipulato si chiama **non-ripudio** (*nonrepudiation*). Gli schemi di firma digitale che vedremo nel seguito permettono la sua implementazione.

Il terzo requisito è necessario per proteggere il cliente, se il prezzo dell'oro sale e la banca tentasse di costruire un messaggio firmato in cui il cliente chiedeva una barra d'oro al posto di una tonnellata. In questo scenario di frode, la banca si tiene per sé il resto dell'oro.

#### 8.4.1 Firme a chiave simmetrica

Un approccio alle firme digitali è la creazione di un'autorità centrale che le conosce tutte e di cui tutti hanno fiducia: chiamiamolo Big Brother (*BB*). Ogni utente sceglie una chiave segreta e la porta a mano nell'ufficio di *BB*. Quindi, solo Alice e *BB* conoscono la chiave segreta  $K_A$ , e così via.

Quando Alice vuole mandare un messaggio in chiaro firmato,  $P$ , al suo banchiere, Bob, genera  $K_A(B, R_A, t, P)$ , dove  $B$  è l'identità di Bob,  $R_A$  è un numero casuale scelto da Alice,  $t$  è il timestamp che assicura la freschezza del messaggio e  $K_A(B, R_A, t, P)$  è il messaggio cifrato con la chiave  $K_A$ . A questo punto Alice manda il messaggio, come nella Figura 8.18. *BB* vede il messaggio da Alice, lo decifra e lo manda a Bob, come mostrato. Il messaggio per Bob contiene il testo in chiaro con il messaggio di Alice e anche il messaggio firmato  $K_{BB}(A, t, P)$ . Infine Bob compie le azioni richieste da Alice.

Cosa succede se, in un secondo tempo, Alice nega di aver spedito il messaggio? Il passo 1 è che ognuno denuncia tutti gli altri (per lo meno questo è quello che accade negli Stati Uniti). Infine, quando il caso arriva in tribunale e Alice continua a negare risolutamente di aver mandato il messaggio in questione a Bob, il giudice chiederà a Bob come può essere sicuro che il messaggio proveniva da Alice e non da Trudy. Bob, come primo argomento, fa notare che *BB* non accetta messaggi da Alice se non sono cifrati con  $K_A$ , quindi non è possibile che Trudy abbia mandato a *BB* un messaggio falso spacciandosi per Alice, senza che *BB* se ne sia accorto subito.

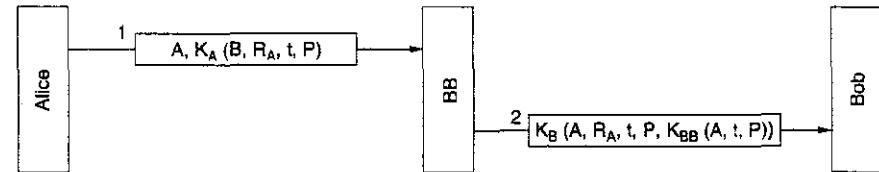


Figura 8.18. Firme digitali usando Big Brother.

Bob a questo punto può presentare la prova A:  $K_{BB}(A, t, P)$ . Bob afferma che questo è un messaggio firmato da *BB*, che prova che Alice ha inviato  $P$  a Bob. Il giudice chiede quindi a *BB* (di cui tutti si fidano) di decifrare la prova A. Quando *BB* testimonia che Bob ha detto la verità, il giudice dà ragione a Bob. Il caso è chiuso.

Un potenziale problema con il protocollo di firma della Figura 8.18 è dato dagli attacchi di tipo ripetizione, che Trudy può eseguire con entrambi i messaggi. Per ridurre l'impatto di questo problema, vengono usati i timestamp. Inoltre Bob può controllare tutti i messaggi recenti per vedere se  $R_A$  era già stato usato. In caso affermativo, il messaggio viene scartato come possibile attacco di ripetizione. Notiamo che, basandosi sul timestamp, Bob rifiuterà i messaggi troppo vecchi. Per proteggersi contro attacchi di ripetizione istantanea, Bob deve solamente controllare che  $R_A$  di ciascun messaggio non corrisponda a quello di un messaggio ricevuto nell'ultima ora da Alice. In questo caso Bob può assumere che il messaggio sia una nuova richiesta.

#### 8.4.2 Firme a chiave pubblica

Il problema strutturale della crittografia a chiave simmetrica per la firma digitale è dato dal fatto che tutti devono fidarsi di Big Brother, il quale può anche leggere i messaggi di tutti. I candidati più logici per gestire un server Big Brother sono il governo, le banche, i contabili e gli avvocati. Sfortunatamente, nessuna di queste organizzazioni riesce a suscitare un senso di sicurezza totale in tutti i cittadini. Quindi è auspicabile avere un metodo per firmare i documenti che non richieda un'autorità fidata.

Fortunatamente, la crittografia a chiave pubblica può dare un contributo importante in quest'area. Assumiamo che gli algoritmi di cifratura e decifrazione abbiano la proprietà che  $E(D(P)) = P$ , oltre alla proprietà solita  $D(E(P)) = P$ . RSA ha questa proprietà, quindi sappiamo già che la nostra richiesta è ragionevole. Sotto queste ipotesi, Alice può trasmettere un messaggio in chiaro,  $P$ , a Bob inviando  $E_B(D_A(P))$ . È importante notare che Alice conosce sia la sua chiave privata,  $D_A$ , sia la chiave pubblica di Bob,  $E_B$ , in questo modo riesce a costruire il messaggio da inviare.

Bob, dopo aver ricevuto il messaggio, lo trasforma come al solito usando la sua chiave privata, ottenendo quindi  $D_A(P)$  come mostrato nella Figura 8.19. Bob memorizza il messaggio in un posto sicuro e poi applica  $E_A$  per ottenere il messaggio in chiaro originale.

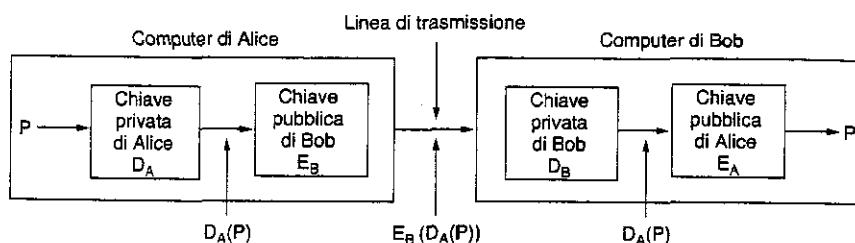


Figura 8.19. Firme digitali usando la crittografia a chiave pubblica.

Per vedere come funziona la proprietà di firma, supponiamo che Alice in un secondo momento voglia negare di aver mandato il messaggio  $P$  a Bob. Quando il caso arriva in tribunale, Bob può presentare sia  $P$  sia  $D_A(P)$ . Il giudice può facilmente verificare che Bob ha in effetti un messaggio valido che è stato cifrato con  $D_A$ , semplicemente applicandogli  $E_A$ . Visto che Bob non conosce la chiave privata di Alice, l'unico modo in cui Bob può aver ricevuto un messaggio cifrato con quella chiave è una trasmissione da parte di Alice. Alice avrà diverso tempo libero per pensare a nuovi schemi crittografici a chiave pubblica mentre sarà in prigione per frode e falsa testimonianza...

La crittografia a chiave pubblica per la firma digitale è uno schema elegante, che però presenta alcuni problemi legati principalmente al contesto in cui viene utilizzata, piuttosto che agli algoritmi utilizzati. Come primo punto, Bob può provare che un messaggio è stato realmente mandato da Alice solo se la chiave  $D_A$  rimane segreta. Se Alice rende pubblica la sua chiave segreta, il discorso non vale più, in quanto chiunque, incluso Bob, potrebbe aver inviato il messaggio.

Per esempio il problema può verificarsi se Bob è l'agente di borsa di Alice. Alice dice a Bob di comprare alcune azioni e obbligazioni. Quasi immediatamente il prezzo cala bruscamente. Con lo scopo di ripudiare il messaggio inviato a Bob, Alice corre dalla polizia dicendo che la sua casa è stata svaligiata e che il PC con la chiave è stato rubato. A seconda delle leggi in vigore nel suo stato di residenza, Alice può essere considerata responsabile del messaggio inviato oppure no; ciò può dipendere anche dal fatto che Alice dichiari di non aver scoperto il furto fino al suo rientro a casa dal lavoro, diverse ore dopo.

Un altro problema con lo schema della firma elettronica si ha quando Alice decide di cambiare la sua chiave. Questa operazione è sicuramente legale, inoltre è una buona idea farla periodicamente. Se in seguito si dovesse andare in tribunale come descritto sopra, il giudice userebbe la chiave corrente  $E_A$  per verificare  $D_A(P)$  e scoprirebbe che non produce  $P$ . A questo punto Bob sarebbe in una posizione indifendibile.

In teoria, qualunque algoritmo a chiave pubblica può essere usato per la firma digitale. Lo standard de facto, utilizzato da una grande quantità di prodotti per la sicurezza, è l'algoritmo RSA. Nel 1991 NIST propose una variante dell'algoritmo a chiave pubblica El Gamal, per il nuovo DSS (Digital Signature Standard, standard per la firma digitale). El Gamal si basa sulla difficoltà di calcolare i logaritmi discreti, mentre RSA si basa sulla scomposizione in fattori.

Come al solito, quando il governo tenta di dettare legge sugli standard crittografici si alza un polverone. DSS è stato criticato per essere:

1. troppo segreto (NSA ha progettato il protocollo per usare El Gamal)
2. troppo lento (da 10 a 40 volte più lento di RSA per controllare le firme)
3. troppo nuovo (El Gamal non è ancora stato esaminato fino in fondo)
4. troppo insicuro (la chiave è di lunghezza fissa pari a 512 bit).

In una revisione successiva è stato superato il quarto punto introducendo chiavi a 1.024 bit.

#### 8.4.3 Message digest

Spesso ai metodi di firma si rivolge una critica: nel loro funzionamento riuniscono due funzioni distinte, l'autenticazione e la segretezza. In molti casi è necessaria l'autenticazione, ma non la segretezza; inoltre è più facile ottenere le licenze per l'esportazione della tecnologia se un sistema fornisce solo autenticazione e non segretezza. Di seguito descriviamo uno schema di autenticazione che non richiede la cifratura dell'intero messaggio. Questo schema consiste nel costruire su una funzione di tipo hash monodirezionale, che prende come input un testo in chiaro di lunghezza arbitraria e calcola una stringa di bit di lunghezza fissa. La funzione hash  $MD$ , detta *message digest* (riassunto del messaggio), ha quattro proprietà importanti:

1. dato  $P$ , è facile calcolare  $MD(P)$
2. dato  $MD(P)$ , è praticamente impossibile trovare  $P$
3. dato  $P$ , nessuno è in grado di trovare  $P'$ , tale che  $MD(P') = MD(P)$
4. se l'input cambia anche di 1 bit, l'output diventa completamente diverso.

Per soddisfare il criterio 3 l'hash dovrebbe essere lungo almeno 128 bit, e preferibilmente di più. Per il criterio 4, l'hash deve modificare in modo radicale i bit in input, in modo analogo agli algoritmi di cifratura a chiave simmetrica che abbiamo discusso.

Calcolare un message digest dal testo in chiaro è molto più veloce che non cifrare il testo stesso con un algoritmo a chiave pubblica, per questo motivo i message digest possono essere usati per velocizzare gli algoritmi di firma digitale. Vediamo un esempio, considerando di nuovo il protocollo della Figura 8.18. Al posto di firmare  $P$  con  $K_{BB}(A, t, P)$ , BB calcola il message digest applicando  $MD$  a  $P$ , cioè calcolando  $MD(P)$ . BB quindi inserisce  $K_{BB}(A, t, MD(P))$  come quinto elemento della lista cifrata con  $K_B$  che viene mandata a Bob, al posto di  $K_{BB}(A, t, P)$ .

Se dovesse nascere una disputa, Bob può mostrare sia  $P$  che  $K_{BB}(A, t, MD(P))$ . Big Brother è in grado di decifrare questo messaggio per il giudice, a questo punto Bob ha  $MD(P)$ , che è garantito genuino, e il messaggio oggetto della disputa,  $P$ . Siccome è impossibile che Bob possa trovare un qualunque altro messaggio che produce l'hash in questione, il giudice

dice sarà facilmente convinto che Bob dice il vero. Usare i message digest in questo modo fa risparmiare tempo di cifratura e costi di trasporto.

I message digest funzionano anche con i sistemi di crittografia a chiave pubblica, come mostrato nella Figura 8.20. In questo caso, Alice prima calcola il message digest del testo in chiaro, poi firma il message digest e manda sia il testo in chiaro sia il digest firmato. Se Trudy rimpiazza  $P$  durante la trasmissione, Bob se ne accorgerà nel momento in cui calcolerà  $MD(P)$  dal messaggio originale.

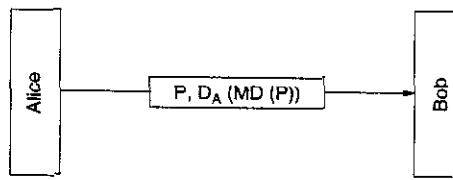


Figura 8.20. Firme digitali usando i message digest.

## MD5

Fra le molte proposte che sono state avanzate per le funzioni message digest, le più usate nella pratica sono MD5 (Rivest, 1992) e SHA-1 (NIST, 1993). **MD5** è il quinto di una serie di message digest progettati da Ronald Rivest. MD5 mescola i bit d'ingresso in una maniera sufficientemente complicata da ottenere che ogni bit di uscita sia influenzato da tutti i bit d'ingresso. In sintesi, l'algoritmo inizia riempiendo il messaggio fino a una lunghezza di 448 bit (modulo 512). Il valore della lunghezza originale del messaggio viene aggiunto sotto forma d'intero a 64 bit, per raggiungere un input di lunghezza 512 bit, e si inizializza a un valore prefissato un buffer di 128 bit.

A questo punto comincia il calcolo vero e proprio. Ogni iterazione prende dall'input un blocco di 512 bit, che viene profondamente alterato usando il buffer a 128 bit. Come extra, viene aggiunta anche una tabella costruita con la funzione seno. L'uso di una funzione ben conosciuta come il seno, al posto di un generatore di numeri casuali, serve per allontanare qualunque sospetto che l'autore abbia inserito delle back door sofisticate, che solo lui riesce a usare. Ricordiamo che i rifiuti da parte di IBM di rendere pubblici i principi alla base delle S-box nel DES generarono una gran quantità di discussioni su possibili back door. Rivest volle evitare questo tipo di sospetti. Per ogni blocco in input si effettuano quattro iterazioni, e il processo prosegue fino a che tutti i blocchi sono stati elaborati. Il contenuto del buffer a 128 bit costituisce il message digest.

MD5 è in vita da più di un decennio e molti hanno tentato di forzarlo. Sono state trovate alcune vulnerabilità, ma non è stato ancora forzato. Se le barriere nella sua struttura interna che ancora resistono dovessero cadere, l'algoritmo potrebbe un giorno essere forzato. A ogni modo, al momento in cui è pubblicato questo libro MD5 resiste ancora.

## SHA-1

L'altra principale funzione di message digest è **SHA-1** (*Secure Hash Algorithm*, algoritmo di hash sicuro), sviluppata da NSA e benedetta da NIST nel FIPS 180-1. Come MD5, anche SHA-1 elabora i dati in input a blocchi di 512 bit, però al contrario di MD5 genera dei message digest di 160 bit. La Figura 8.21 mostra un tipico esempio di come Alice può mandare a Bob un messaggio non segreto ma firmato. Il testo in chiaro viene usato come input per l'algoritmo SHA-1 per ottenere l'hash da 160 bit. Alice firma l'hash con la sua chiave privata RSA e trasmette a Bob sia il testo in chiaro sia l'hash.

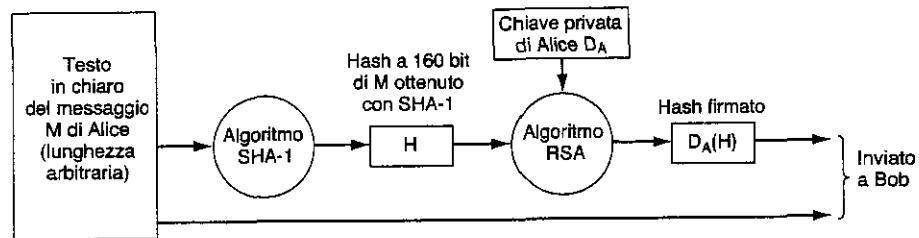


Figura 8.21. Uso di SHA-1 e RSA per firmare messaggi non segreti.

Dopo aver ricevuto il messaggio, Bob calcola l'hash SHA-1 dal testo in chiaro e applica la chiave pubblica di Alice all'hash firmato per ottenere l'hash originale,  $H$ . Se i due sono d'accordo, il messaggio viene considerato valido. Trudy non è in grado di modificare il testo (in chiaro) mentre è in transito per produrne un altro che abbia lo stesso hash  $H$ , quindi Bob riesce facilmente a identificare i cambiamenti eventualmente apportati da Trudy. Lo schema della Figura 8.21 è ampiamente usato per trasmettere messaggi dove l'integrità è importante, ma i contenuti non sono segreti. Con un costo di calcolo relativamente basso, questo schema riesce a garantire una probabilità molto alta di rilevare le modifiche al testo in transito.

Vediamo brevemente come funziona SHA-1. Si comincia con il riempimento del messaggio aggiungendo un bit a 1 in coda, seguito da tanti 0 quanti sono necessari per renderne la lunghezza multipla di 512 bit. Si calcola un numero a 64 bit contenente la lunghezza del messaggio prima del riempimento, e lo si mette in OR con i 64 bit di ordine più basso. Nella Figura 8.22 il messaggio è mostrato con riempimento a destra, in quanto il testo e le figure si leggono da sinistra verso destra (cioè la posizione in basso a destra è normalmente considerata come la fine della figura). Per i computer, l'orientamento illustrato corrisponde alle macchine big-endian come le SPARC; comunque SHA-1 riempie sempre la fine del messaggio, indipendentemente dal tipo di computer utilizzato.

L'algoritmo di SHA-1 utilizza cinque variabili a 32 bit dove viene accumulato l'hash, da  $H_0$  a  $H_4$ , come mostrato nella Figura 8.22(b). Queste variabili sono inizializzate da costanti specificate nello standard.

I blocchi da  $M_0$  a  $M_{n-1}$  sono elaborati uno dopo l'altro. Per il blocco corrente le 16 word

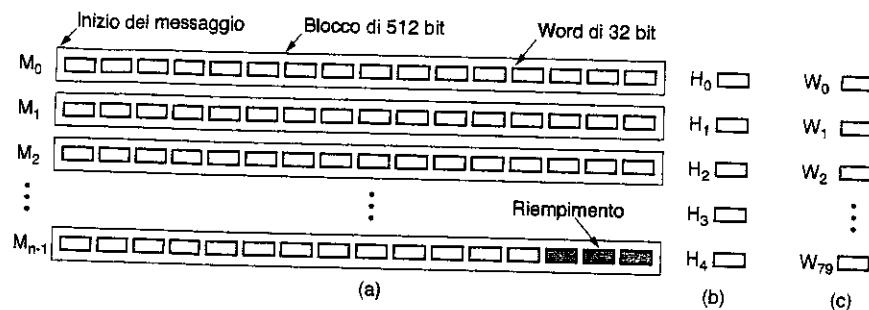


Figura 8.22. (a) Messaggio esteso a un multiplo di 512 bit. (b) Variabili di uscita. (c) L'array word.

sono copiate nella parte iniziale di un array ausiliario di 80 word,  $W$ , come mostrato nella Figura 8.22(c); le altre 64 word di  $W$  si riempiono applicando la formula

$$W_i = S^b(W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \quad (16 \leq i \leq 79)$$

Dove  $S^b(W)$  rappresenta la rotazione circolare verso sinistra della word a 32 bit,  $W$ , di  $b$  bit. Cinque variabili di servizio, da  $A$  ad  $E$ , vengono inizializzate da  $H_0$  fino ad  $H_4$ , rispettivamente.

Il calcolo effettuato può essere espresso in pseudo-C nel modo seguente:

```
for (i = 0; i < 80; i++) {
 temp = S5(A) + fi(B, C, D) + E + Wi + Ki;
 E = D; D = C; C = S30(B); B = A; A = temp;
}
```

dove le costanti  $K_i$  sono definite nello standard. Le definizioni delle funzioni di mescolamento  $f_i$  sono:

$$\begin{aligned} f_i(B, C, D) &= (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D) & (0 \leq i \leq 19) \\ f_i(B, C, D) &= B \text{ XOR } C \text{ XOR } D & (20 \leq i \leq 39) \\ f_i(B, C, D) &= (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) & (40 \leq i \leq 59) \\ f_i(B, C, D) &= B \text{ XOR } C \text{ XOR } D & (60 \leq i \leq 79) \end{aligned}$$

Quando tutte le 80 iterazioni del ciclo sono completate, le variabili da  $A$  ad  $E$  vengono aggiunte ad  $H_0$  fino ad  $H_4$ , rispettivamente.

Dopo aver elaborato il primo blocco di 512 bit si comincia con quello successivo. L'array  $W$  è inizializzato usando il nuovo blocco, mentre  $H$  rimane invariata. Quando anche questo blocco è stato elaborato, si continua con quello successivo, e così via fino a esaurimento dei blocchi in input. Quando anche l'ultimo blocco è stato elaborato, le cinque word a 32 bit nell'array  $H$  rappresentano l'output a 160 bit dell'hash crittografico. Il codice completo in C per SHA-1 è riportato in RFC 3174.

Sono in corso di sviluppo nuove versioni di SHA-1 per produrre hash crittografici di 256, 384 e 512 bit.

#### 8.4.4 L'attacco del compleanno

Nel mondo della crittografia, niente è mai quello sembra. Si può pensare che ci voglia un numero di operazioni dell'ordine di  $2^m$  per forzare un message digest di  $m$  bit. In realtà, spesso possono bastare  $2^{m/2}$  operazioni, usando l'**attacco del compleanno**, un approccio pubblicato da Yuval (1979) in un articolo diventato un classico, intitolato "How to Swindle Rabin".

L'idea per questo attacco viene da una tecnica che i professori di matematica usano spesso nei loro corsi di probabilità. La domanda è: quanti studenti ci devono essere in una classe perché la probabilità che due persone abbiano il compleanno lo stesso giorno superi  $1/2$ ? Molti studenti si aspettano che la risposta sia un numero molto maggiore di 100. In realtà la teoria della probabilità ci dice che ne bastano 23. Non ne daremo una dimostrazione rigorosa, ma ragioneremo intuitivamente. Con 23 persone possiamo formare  $(23 \times 22)/2 = 253$  coppie differenti, ognuna ha la probabilità di  $1/365$  di essere quella buona. Con questa osservazione, il risultato non è più sorprendente.

Generalizzando, se c'è una funzione fra input e output con  $n$  valori di input (persone, messaggi, ecc) e  $k$  possibili valori di output (compleanni, message digest, ecc), ci sono  $n(n-1)/2$  coppie di input. Se  $n(n-1)/2 > k$ , la possibilità di avere almeno una coppia con lo stesso output è decisamente buona. Quindi, approssimativamente, per avere due output uguali basta avere  $n > \sqrt{k}$ . Questo risultato significa che un message digest di 64 bit può essere forzato, con una buona probabilità, generando  $2^{32}$  messaggi e cercandone due con lo stesso message digest.

Vediamo un esempio pratico. Il dipartimento di informatica alla state university ha una cattedra vacante e due candidati, Tom e Dick. Tom è entrato nel dipartimento due anni prima di Dick e quindi viene valutato per primo. Se ottiene il posto, Dick sarà scartato. Tom sa che la direttrice del dipartimento, Marilyn, ha un'alta considerazione del suo lavoro, quindi le chiede di scrivere una lettera di raccomandazione al rettore, che dovrà decidere dell'assunzione di Tom. Una volta inviate, tutte le lettere diventano confidenziali.

Marilyn chiede alla sua segretaria, Ellen, di scrivere una lettera al rettore, dicendole gli argomenti da inserire. Quando la lettera sarà pronta, Marilyn la controllerà, calcolerà il digest a 64 bit e la invierà al rettore; dopo Ellen potrà inviare la lettera per e-mail.

Sfortunatamente per Tom, Ellen è innamorata di Dick e vorrebbe sabotare Tom, quindi scrive la seguente lettera con le 32 opzioni elencate fra parentesi.

Egregio Rettore,

Questa [lettera | missiva] ha lo scopo di fornirle la mia [onesta | franca] opinione sul Prof. Tom Wilson, che [è candidato | concorre] per la cattedra [attualmente | quest'anno]. [Conosco | Ho lavorato con] Wilson per [quasi | già] sei anni. Egli è un [eccellente | ammirabile] ricercatore di grande [talento | capacità] noto [in tutto il mondo | internazionalmente] per la sua [brillante | creativa] conoscenza in [molti | svariati] problemi di [grande difficoltà | impegnativa soluzione].

È inoltre un [insegnante | docente] [molto | notevolmente] [stimato | apprezzato]. I suoi studenti danno [ai suoi corsi | alle sue lezioni] voti [spettacolari | massimi]. [Nel nostro dipartimento | nella nostra facoltà] è l' [insegnante | docente] più [popolare | stimato].

[Inoltre | Per giunta] il Prof. Wilson è [abile | efficace] nella ricerca di fondi. I suoi [contatti | progetti] hanno portato un [notevole | significativo] finanziamento [al nostro dipartimento | alla nostra università]. Questi [fondi | finanziamenti] ci hanno [permesso | consentito] di [portare avanti | sviluppare] molti [importanti | significativi] progetti, [tra cui | per esempio] il progetto Stato 2000. Senza questi fondi [non saremmo in grado | non potremmo] proseguire il progetto, che è così [importante | vitale] per noi. La invito caldamente ad affidargli la cattedra.

Sfortunatamente per Tom, dopo aver composto e battuto la lettera in questione, Ellen ne scrive una seconda:

Egregio Rettore,  
 Questa [lettera | missiva] ha lo scopo di fornirle la mia [onesta | franca] opinione sul Prof. Tom Wilson, che [è candidato | concorre] per la cattedra [attualmente | quest'anno]. [Conosco | Ho lavorato con] Wilson per [quasi | già] sei anni. Egli è un [mediocre | svolgiano] ricercatore poco noto nella sua [disciplina | area]. Le sue ricerche [raramente | quasi mai] mostrano [competenza | comprensione] dei problemi [principali | vitali] dell'attività [che svolgiamo | in corso oggi]. Inoltre, non è un [insegnante | docente] [stimato | apprezzato]. I suoi studenti danno [ai suoi corsi | alle sue lezioni] voti [bassi | insufficienti]. [Nel nostro dipartimento | Nella nostra facoltà] è l' [insegnante | docente] meno [popolare | stimato], noto [soprattutto | principalmente] nel [dipartimento | gruppo di lavoro] per la sua [tendenza | propensione] a [ridicolizzare | imbarazzare] gli studenti che [imprudentemente | inutilmente] fanno domande alle sue lezioni.  
 [Inoltre | Per giunta] il Prof. Wilson è [incapace | inefficace] nella ricerca di fondi. I suoi [contatti | progetti] hanno portato un [minimo | insignificante] finanziamento [al nostro dipartimento | alla nostra università]. Se non verranno trovati presto [nuovi fondi | nuovi finanziatori] dovremo cancellare alcuni programmi vitali, come "Stato 2000". Sfortunatamente, a causa di queste [condizioni | circostanze] non posso raccomandare [in coscienza | in buona fede] Wilson per [una cattedra | una posizione permanente].

A questo punto Ellen programma il suo computer per calcolare, durante la notte, i  $2^{32}$  message digest di ciascuna delle due lettere. È probabile che un digest della prima lettera possa essere uguale a un digest della seconda lettera. Altrimenti, Ellen può aggiungere qualche altra opzione e provare di nuovo durante il weekend. Supponiamo che trovi una coppia corretta. Chiamiamo A la lettera "buona" e B quella "cattiva".

Ellen manda per e-mail la lettera A a Marilyn per approvazione. La lettera B viene invece tenuta segreta. Marilyn, ovviamente, approva e calcola il suo digest a 64 bit, lo firma e manda per e-mail il digest firmato al rettore Smith. Indipendentemente, Ellen manda per e-mail la lettera B al rettore (non la lettera A, come dovrebbe).

Dopo aver ricevuto la lettera e il digest firmato, il rettore calcola il message digest della lettera B, vede che è uguale a quello inviato da Marilyn e licenzia Tom. Il rettore non si rende conto che Ellen è riuscita a generare due lettere con lo stesso message digest e che ne ha inviata una differente da quella che Marilyn aveva visto e approvato (finale alterna-

tivo: Ellen dice a Dick quello che ha fatto. Dick è scandalizzato e la lascia. Ellen è furiosa e si confessa con Marilyn. Marilyn chiama il rettore. Tom riceve la cattedra). L'attacco del compleanno è difficile con MD5, in quanto anche elaborando 1 miliardo di digest al secondo, ci vorrebbero più di 500 anni per calcolare tutti i  $2^{64}$  digest di due lettere con 64 varianti ciascuna. Anche in quel caso, comunque, non c'è garanzia di riuscire a trovare una coppia di messaggi con lo stesso digest. Ovviamente, con 5.000 computer che lavorano in parallelo, 500 anni diventano 5 settimane. SHA-1 è migliore, in quanto produce un message digest più lungo.

## 8.5 Gestione delle chiavi pubbliche

La crittografia a chiave pubblica consente la comunicazione sicura fra persone che non condividono una chiave comune e permette la firma dei messaggi senza la presenza di una terza parte fidata. Infine, i message digest firmati abilitano e facilitano la verifica dell'integrità dei messaggi.

Rimane però un problema su cui abbiamo sorvolato troppo velocemente: se Alice e Bob non si conoscono, come fanno a procurarsi le rispettive chiavi pubbliche per cominciare la comunicazione? La soluzione ovvia, quella di mettere la chiave sul proprio sito Web, non funziona per il seguente motivo. Supponiamo che Alice voglia cercare la chiave pubblica di Bob sul suo sito Web. Come procede? Comincia digitando l'URL di Bob. Il suo browser cerca sul DNS l'indirizzo della home page di Bob e manda la richiesta GET, come mostrato nella Figura 8.23. Sfortunatamente, Trudy intercetta la richiesta e risponde con una home page falsificata, probabilmente una copia della home page di Bob con la differenza che la chiave pubblica di Bob è stata sostituita con quella di Trudy. Quando Alice cifra il suo primo messaggio usando  $E_T$ , Trudy lo decifra, lo legge, lo cifra di nuovo con la chiave pubblica di Bob e lo invia a Bob, il quale è ignaro del fatto che Trudy sta leggendo i messaggi in arrivo. A peggiorare le cose, rimane il fatto che Trudy potrebbe anche modificare i messaggi prima di cifrarli e spedirli a Bob. Chiaramente è necessario un qualche meccanismo per assicurarsi che le chiavi pubbliche possano essere scambiate in modo sicuro.

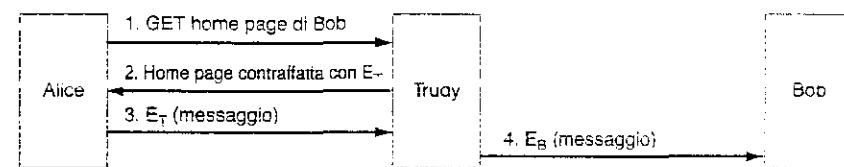


Figura 8.23. Come Trudy può forzare la cifratura a chiave pubblica

### 8.5.1 Certificati

Come primo tentativo per distribuire le chiavi pubbliche in modo sicuro, possiamo immaginare l'esistenza di un centro di distribuzione aperto 24 ore su 24 per fornire le chiavi pubbliche su richiesta. Uno dei problemi associati a questo tipo di soluzione è la mancan-

za di scalabilità, e quindi il fatto che il centro di distribuzione delle chiavi diventerebbe presto il collo di bottiglia del sistema. Inoltre, se questo centro dovesse andare fuori servizio la sicurezza di Internet subirebbe un'improvvisa battuta d'arresto.

Per questo motivo è stata sviluppata una soluzione differente, che non richiede un centro di distribuzione delle chiavi online a tutte le ore. In effetti non deve essere affatto online: quello che deve fare è certificare le chiavi pubbliche che appartengono alle persone, alle aziende e alle altre organizzazioni. Un'organizzazione che certifica le chiavi pubbliche è chiamata una **CA** (*Certificate Authority*, autorità di certificazione).

Supponiamo, per esempio, che Bob voglia autorizzare Alice, insieme ad altre persone, a comunicare con lui in modo sicuro. Bob può rivolgersi alla CA portando la sua chiave pubblica e il suo passaporto, o un altro documento, e chiedere di essere certificato. La CA emette un certificato simile a quello della Figura 8.24, ne calcola l'hash SHA-1 e lo firma con la chiave privata di CA. Bob paga un compenso alla CA e ottiene un floppy disk con il certificato e l'hash firmato.

Certifico che la chiave pubblica 19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A appartiene a Robert John Smith 12345 University Avenue Berkeley, CA 94702 Birthday: July 4, 1958 Email: bob@superduper.net.com
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Hash SHA-1 del certificato di cui sopra firmato con la chiave privata della CA

Figura 8.24. Un esempio di certificato e del suo hash firmato.

Lo scopo principale di un certificato è quello di legare una chiave pubblica al nome di un protagonista (individuo, azienda, ecc.). I certificati, di per sé, non sono né segreti né protetti. Per esempio Bob potrebbe decidere di mettere il suo nuovo certificato sul proprio sito Web, con un link sulla pagina principale del tipo "fare clic qui per vedere il mio certificato con la chiave pubblica". Il link riporterebbe il certificato e la firma (l'hash SHA-1 firmato del certificato).

Torniamo alla situazione della Figura 8.23. Quando Trudy intercetta la richiesta di Alice che vuole leggere la home page di Bob, cosa può fare? Può mettere il suo certificato, con la relativa firma, nella pagina falsificata; ma quando Alice leggerà il certificato capirà immediatamente che non sta parlando con Bob, in quanto il nome di Bob non è sul certificato. Trudy può anche modificare al volo la home page di Bob, rimpiazzando la chiave pubblica di Bob con la sua chiave. Però, quando Alice calcola l'hash SHA-1 del certificato e lo confronta con il valore ottenuto applicando la chiave pubblica di CA alla firma del certificato, vede che sono diversi e quindi c'è qualcosa che non funziona. Operando in questo modo, Alice riesce a essere sicura di avere la chiave pubblica di Bob e non quella di Trudy o di qualcun altro. Come avevamo promesso, questo schema non richiede che la CA sia online durante la verifica, eliminando quindi un potenziale collo di bottiglia.

La funzione principale di un certificato è quella di legare una chiave pubblica con un protagonista. Un certificato può anche essere usato per legare una chiave pubblica a un **attributo**. Per esempio, un certificato può affermare: questa chiave pubblica appartiene a una persona maggiorenne. Possiamo perciò usare un certificato per provare che il proprietario della chiave privata non è minorenne e può accedere a del materiale non adatto ai bambini, ecc; il tutto senza dover esporre l'identità del proprietario. Tipicamente, la persona che possiede il certificato lo invia al sito Web, al protagonista, o al processo interessati a determinare l'età della controparte. Questo sito, protagonista o processo procede generando un numero casuale, che viene usato come chiave della sessione per il resto della comunicazione.

Un altro esempio di un certificato che contiene un attributo è dato da un sistema distribuito orientato agli oggetti. Ogni oggetto normalmente contiene diversi metodi. Il proprietario dell'oggetto può fornire, a ogni client, un certificato contenente l'elenco dei metodi che il client può utilizzare. Questo elenco viene poi legato a una chiave pubblica tramite un certificato. Anche in questo caso il detentore del certificato, una volta che ha provato di essere in possesso della chiave privata, viene autorizzato a eseguire i metodi nella lista. Questo schema ha la proprietà di non richiedere la conoscenza dell'identità del detentore del certificato, che è molto utile nelle situazioni in cui la privacy è importante.

### 8.5.2 X.509

Se tutte le persone interessate alla firma elettronica andassero dalla CA con un tipo diverso di certificato, l'operazione di gestire i diversi formati diventerebbe presto proibitiva. Per risolvere questo problema è stato sviluppato uno standard per i certificati, poi approvato dalla ITU. Lo standard è chiamato **X.509** ed è largamente utilizzato su Internet. Ne sono già state sviluppate tre versioni a partire dallo standard iniziale del 1988; in seguito discuteremo la V3.

X.509 è stato influenzato pesantemente dal mondo OSI, portandosi dietro alcune delle sue peggiori caratteristiche (per esempio la nomenclatura e la codifica). Sorprendentemente IETF ha approvato X.509 anche se, cercando di operare al meglio, ha generalmente ignorato OSI in tutte le altre aree, dai nomi delle macchine ai protocolli di trasporto, agli indirizzi e-mail. La versione di IETF di X.509 è descritta in RFC 3280.

La funzione principale di X.509 è quella di descrivere i certificati. I campi primari di un certificato sono elencati nella Figura 8.25, dove la descrizione dovrebbe dare un'idea generale del significato di ciascun campo. Per ulteriori informazioni, si consiglia di consultare lo standard stesso o RFC 2459.

Per esempio, se Bob lavora alla divisione prestiti della Money Bank, il suo indirizzo X.500 potrebbe essere:

/C=US/O=MoneyBank/OU=Loan/CN=Bob/

dove *C* sta per country (nazione), *O* per organizzazione, *OU* per organizational unit (unità operativa), e *CN* sta per common name (nome comune). Le CA e altre entità sono chiamate in modi simili.

Campo	Significato
Version	Numero della versione di X.509
Serial number	Il certificato è univocamente identificato da questo numero più il nome della CA
Signature algorithm	Algoritmo usato per firmare il certificato
Issuer	Nome X.500 della CA
Validity period	Inizio e fine del periodo di validità
Subject name	L'entità proprietaria della chiave da certificare
Public key	La chiave pubblica del soggetto e l'ID dell'algoritmo che la usa
Issuer ID	Facoltativo: identificativo univoco di chi emette il certificato
Subject ID	Facoltativo: identificativo univoco del soggetto del certificato
Extensions	Sono state definite molte estensioni
Signature	La firma del certificato (firmata dalla chiave privata della CA)

Figura 8.25. I campi essenziali di un certificato X.509.

Un problema sostanziale con i nomi X.500 è il seguente: quando Alice tenta di contattare [bob@moneybank.com](mailto:bob@moneybank.com) e le viene dato un certificato con un nome X.500, per Alice può non essere ovvio che il certificato ottenuto si riferisce proprio al Bob che lei sta cercando. Fortunatamente, a partire dalla versione 3, è permesso usare i nomi DNS al posto di quelli X.500, quindi il problema alla fine potrebbe scomparire. I certificati sono codificati con OSI ASN.1 (*Abstract Syntax Notation 1*, notazione sintattica astratta N.1), che può essere pensata come una sorta di struct del C, a parte una notazione particolarmente prolissa. Altre informazioni su X.509 si trovano in (Ford e Baum, 2000).

### 8.5.3 Infrastrutture a chiave pubblica

Naturalmente, affidarsi a una sola CA per emettere tutti i certificati del mondo è una soluzione che non può funzionare. Collasserebbe sotto il carico di lavoro, e sarebbe anche un punto centrale di rottura del sistema. Una soluzione possibile consiste nell'avere diverse CA, tutte gestite dalla stessa organizzazione e tutte con la stessa chiave privata usata per firmare i certificati. Questa soluzione risolverebbe i problemi del carico e del punto centrale di fallimento, ma ne introdurrebbe anche uno nuovo: la sicurezza della chiave. Se ci sono dozzine di server sparsi per il mondo, tutti con la stessa chiave privata, la possibilità che la chiave venga rubata o comunque trapeli all'esterno risulta molto amplificata. Visto che la compromissione di questa chiave rovinerebbe l'infrastruttura di sicurezza elettronica del mondo, il fatto di avere un'unica CA comporta un grave rischio. Inoltre, quale organizzazione dovrebbe gestire la CA? È difficile immaginare un'autorità che possa essere accettata in tutto il mondo come legittima e degna di fiducia. In alcuni paesi la gente vorrebbe che fosse il governo, in altri invece si desidererebbe in contrario. Per questo motivo si è evoluto un metodo differente per certificare le chiavi pubbliche. Questo metodo va sotto il nome generico di PKI (*Public Key Infrastructure*, infrastruttura per le chiavi pubbliche). In questo paragrafo faremo un riassunto del funzionamento generale, anche se i dettagli probabilmente evolveranno nel tempo sotto la spinta delle diverse proposte che vengono avanzate.

Una PKI ha diversi componenti, fra cui gli utenti, le CA, i certificati e le directory. La funzione della PKI è quella di fornire la struttura per inserire tutte queste componenti e definire gli standard per i diversi tipi di documenti e protocolli. Un esempio particolarmente semplice di PKI è la gerarchia di CA, riportata nella Figura 8.26. In questo esempio abbiamo riportato tre livelli, ma nella realtà possono essere di più o di meno. La CA di livello più alto, la root (radice), certifica le CA di secondo livello, che chiamiamo RA (Regional Authorities, autorità regionali) perché possono coprire delle regioni geografiche, come uno stato o un continente. Questo termine non è standard, in effetti nessun termine per definire i vari livelli dell'albero è veramente standard. A loro volta, le RA certificano le vere e proprie CA, che emettono i certificati X.509 alle organizzazioni e ai privati. Quando la root autorizza una nuova RA, genera un certificato X.509 che ne sancisce l'avvenuta approvazione e che include la chiave pubblica della nuova RA, quindi lo firma e lo passa alla RA. Analogamente, quando una RA approva una nuova CA, produce e firma un certificato che riporta l'avvenuta approvazione e che contiene la chiave pubblica della CA.

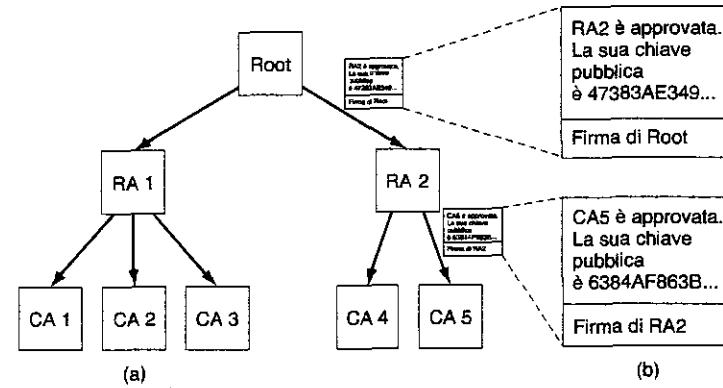


Figura 8.26. (a) PKI gerarchico. (b) Catena di certificati.

La nostra PKI funziona in questo modo. Supponiamo che Alice abbia bisogno della chiave pubblica di Bob per poter comunicare con lui. Alice cerca e trova un certificato che contiene la chiave di Bob firmato da CA 5. Però Alice non ha mai sentito parlare di CA 5. Per quanto ne sa Alice, CA 5 potrebbe essere la figlia di 10 anni di Bob. Alice può andare da CA 5 e chiedere: provami il fatto che tu sei un'autorità legittima. CA 5 risponde con il certificato che ha preso da RA 2, che contiene la chiave pubblica di CA 5. Armato della chiave pubblica di CA 5, Alice può verificare che il certificato di Bob è stato in effetti firmato da CA 5 ed è quindi valido.

Questo è vero a patto che RA 2 non sia il figlio di 12 anni di Bob. Quindi il passo successivo consiste nel chiedere a RA 2 di dimostrare la sua legittimità. La risposta alla richiesta è data dal certificato firmato da root e che contiene la chiave pubblica di RA 2. A questo punto Alice è sicura di avere la chiave pubblica di Bob.

Come fa Alice a trovare la chiave pubblica di root? Si suppone che tutti conoscano la chiave pubblica di root. Per esempio, il browser di Alice può essere stato rilasciato con la chiave pubblica di root già al suo interno.

Bob è una persona collaborativa e non vuole che Alice faccia troppo lavoro. Sa che Alice dovrà controllare CA 5 e RA 2, quindi per risparmiarle del lavoro raccoglie i due certificati necessari e li invia insieme al suo. Alice può usare la sua conoscenza della chiave pubblica di root per verificare il certificato di livello più alto, prendere la chiave che si trova al suo interno e quindi verificare anche il secondo certificato. In questo modo, Alice non ha bisogno di contattare nessuno per effettuare le verifiche. Visto che i certificati sono tutti firmati, Alice può facilmente identificare i tentativi di modificare i loro contenuti. Una catena di certificati che risale fino alla root, come quella descritta, viene a volte chiamata una **catena di fiducia** o un **percorso di certificazione**. Questa tecnica è ampiamente usata nella pratica.

Ovviamente abbiamo ancora il problema di chi debba gestire la root. La soluzione consiste nel non avere una singola root, ma di averne tante, ognuna con i suoi RA e CA. In effetti, i browser moderni contengono già al loro interno le chiavi pubbliche di oltre 100 root, a volte chiamati **trust anchors** (ancore fiduciarie). In questo modo si evita di dover avere una singola autorità fiduciaria per il mondo intero.

Rimane il problema di come viene presa la decisione da parte del produttore del browser relativamente a quali trust anchor sono affidabili e quali no. Alla fine dei conti sarà l'utente del browser a doversi fidare che il produttore abbia fatto la scelta giusta, e non abbia semplicemente incluso tutte le trust anchor che hanno pagato un prezzo per entrare nella lista. La maggior parte dei browser permettono agli utenti di controllare le chiavi delle root (di solito nella forma di certificati firmati dalla root), e di cancellare quelli che sembrano sospetti.

## Directory

Un'altra questione che viene affrontata da tutte le PKI è quella della memorizzazione dei certificati (e delle loro catene fino a qualche trust anchor conosciuta). Una possibilità consiste nel far sì che ciascun utente memorizzi i propri certificati. Questo modo di operare è sicuro (non c'è modo per gli utenti di manomettere i certificati firmati, senza essere scoperti), però è anche scomodo. Una proposta alternativa consiste nell'usare il DNS come directory dei certificati. Bob, prima di contattare Alice, probabilmente deve cercare il suo indirizzo IP usando un DNS: quindi perché non far sì che il DNS ritorni a Bob anche l'intera catena di certificati, insieme all'indirizzo IP?

Alcune persone pensano che questa sia la soluzione giusta, altre invece preferiscono usare dei directory server dedicati alla gestione dei certificati X.509. Tali directory potrebbero fornire servizi di ricerca usando le proprietà dei nomi X.500. Per esempio, in teoria una directory di questo tipo potrebbe rispondere a domande come: "dammi la lista delle persone che si chiamano Alice, lavorano nel dipartimento vendite e si trovano in Canada o negli U.S.A.". LDAP potrebbe essere un candidato per contenere questo tipo di informazioni.

## Revoca

Anche il mondo reale è pieno di certificati, pensiamo ai passaporti e alle patenti di guida. In alcuni casi questi certificati possono essere revocati, per esempio la patente di guida può essere revocata in caso di guida in stato di ebbrezza o per altre infrazioni. Lo stesso problema esiste nel mondo digitale: il garante che ha emesso un certificato può decidere di revocarlo, in quanto la persona o l'organizzazione che l'ha ricevuto ne ha abusato in qualche modo. Il certificato può essere revocato anche nel caso in cui la chiave privata del ricevente sia stata esposta. Un'altra situazione, ancora più grave, si ha se la chiave privata della CA è stata compromessa.

Un primo passo in questa direzione consiste nel far sì che ogni CA periodicamente emetta un **CRL** (*Certificate Revocation List*, lista dei certificati revocati) contenente il numero seriale di tutti i certificati che ha revocato. Visto che i certificati contengono una data di scadenza, la CRL deve contenere solo i numeri seriali dei certificati da revocare che non sono ancora scaduti. Un certificato perde automaticamente validità dopo la sua data di scadenza, quindi non è necessario distinguere fra i certificati scaduti e quelli effettivamente revocati. In entrambi i casi i certificati non possono più essere usati.

Sfortunatamente, introdurre le CRL significa che un utente che sta per usare un certificato deve anche prendere la CRL e controllare se il certificato in questione è stato revocato; in caso affermativo non dovrebbe usarlo. Anche se il certificato non è nella lista, potrebbe essere stato revocato poco dopo la pubblicazione della lista stessa. Quindi l'unico modo per esserne sicuri è quello d'interrogare la CA. La CA va interrogata anche in occasione del successivo uso di questo stesso certificato, in quanto (nel frattempo) il certificato potrebbe essere stato revocato.

Un'altra complicazione è data dal fatto che il certificato revocato potrebbe anche essere reso di nuovo valido, per esempio se era stato revocato in attesa di un pagamento dovuto che poi è stato saldato. Gestire la revoca (e magari anche il reinserimento) annulla una delle migliori proprietà dei certificati, cioè il fatto che possono essere usati senza dover contattare la CA.

Dove vanno memorizzate le CRL? Un buon posto sarebbe quello in cui sono memorizzati i certificati stessi. Una strategia consiste nel prevedere una pubblicazione periodica delle CRL da parte delle CA. Le directory procedono quindi a elaborare le CRL semplicemente rimuovendo i certificati revocati. Se invece non si usano le directory per memorizzare i certificati, le CRL possono essere tenute in cache in varie posizioni di facile accesso all'interno della rete. Visto che una CRL è di per sé un documento firmato, ogni tentativo di manomissione viene facilmente scoperto.

Se i certificati hanno una vita lunga, anche le CRL saranno lunghe. Per esempio, la lista delle carte di credito revocate sarà molto più lunga nel caso di carte con validità di 5 anni, che non nel caso di carte con validità 3 mesi. Un metodo standard per operare con le CRL consiste nell'emettere sporadicamente una lista principale, seguita però da una serie di aggiornamenti frequenti. Con questo meccanismo si riduce la banda necessaria a distribuire le CRL.

## 6 Sicurezza delle comunicazioni

biamo finito il nostro studio degli strumenti del mestiere, trattando le tecniche e i problemi più importanti. Il resto del capitolo esamina le applicazioni pratiche e il modo in cui queste tecniche sono sfruttate per fornire sicurezza alle reti. Alla fine del capitolo si discuteranno anche alcuni aspetti sociali legati alla sicurezza.

I quattro paragrafi seguenti affrontano il tema della sicurezza delle comunicazioni. Vedremo quindi come sia possibile trasferire un flusso di bit dalla sorgente alla destinazione, in modo sicuro, senza che ci siano modifiche e tenendo alla larga i bit indesiderati. Queste non sono le sole problematiche di sicurezza delle reti, ma sono certamente gli argomenti più importanti e quindi costituiscono un buon punto di partenza.

### 1 IPsec

Fra sapeva da tempo che Internet mancava di sicurezza, ma aggiungerla non era facile, in quanto era in corso una guerra per decidere dove fosse meglio inserirla. La maggior parte degli esperti ritiene che per rendere Internet veramente sicura bisogna aggiungere la cifratura e i controlli di integrità da capo a capo (cioè nello strato applicativo). In questo modo il processo sorgente cifra e/o protegge l'integrità dei dati e poi li invia al processo di destinazione che si prende in carico la decifrazione e/o la verifica di integrità. Ogni manipolazione dei dati effettuata fra questi due processi può essere rilevata, anche se viene attivata a livello del sistema operativo. Il problema di questo approccio è che richiede la modifica di tutte le applicazioni per poterle usare con la sicurezza. In questa ottica, il secondo miglior tentativo consiste nel mettere la sicurezza allo strato di trasporto oppure in un nuovo strato, intermedio fra lo strato applicativo e quello di trasporto. Anche così si ottiene una sicurezza da capo a capo, ma senza cambiamento delle applicazioni.

L'approccio alternativo consiste nel partire dal presupposto che gli utenti non capiscono la sicurezza, non sono in grado di utilizzarla in modo corretto e che quindi nessuno vuole modificare i programmi esistenti in alcun modo. In questa ottica autenticare e/o cifrare i pacchetti diventa compito dello strato network, senza coinvolgere gli utenti. Dopo anni di dibattaglie, questa filosofia ha raggiunto abbastanza consenso da stimolare la definizione di standard per la sicurezza dello strato network. La spiegazione del successo consiste, in parte, nel fatto che avere la cifratura a livello dello strato network non impedisce agli utenti orientati alla sicurezza di operare comunque nel modo più sicuro; allo stesso tempo non impedisce in qualche modo anche quella parte di utenti che ignorano la sicurezza.

Il risultato di questa guerra fu chiamato **IPsec** (*IP security*), che è descritto in RFC 2401, 2402, 2406 e altri documenti. Non tutti gli utenti vogliono la cifratura (perché è pesante dal punto di vista computazionale). Invece di rendere la cifratura opzionale, fu deciso di obbligarla comunque, ma di permettere l'uso di un algoritmo nullo. L'algoritmo nullo è descritto nell'RFC 2410, dove si trovano anche i dettagli della sua ammirabile semplicità di implementazione e grande velocità.

Il progetto completo dell'IPsec definisce un'infrastruttura per fornire molteplici servizi, con diversi criteri e granularità. La ragione per avere molteplici servizi è che non tutti vogliono pagare il prezzo per avere tutti i servizi disponibili in ogni istante, quindi alcuni servizi

vengono resi disponibili su richiesta. I servizi principali sono: la segretezza, l'integrità dei dati e la protezione dagli attacchi di tipo ripetizione (dove l'intruso invia di nuovo un vecchio messaggio). Tutti questi servizi sono basati sulla crittografia a chiave simmetrica, in quanto le performance sono cruciali.

La ragione per avere più algoritmi è data dal fatto che un algoritmo che oggi crediamo sicuro potrebbe essere forzato in futuro. Rendendo l'IPsec indipendente dall'algoritmo, l'infrastruttura può sopravvivere anche se un particolare algoritmo viene forzato. Prevedendo diverse granularità, infine, è possibile proteggere una singola connessione TCP, come tutto il traffico fra due host, oppure tutto il traffico fra due router sicuri, o altre possibilità.

Un aspetto parzialmente sorprendente dell'IPsec è il fatto che è orientato alla connessione, anche se si trova allo strato IP. In definitiva questo fatto non è poi così sorprendente, in quanto per avere una qualche forma di sicurezza bisogna stabilire una chiave da usare per un certo periodo di tempo: ciò è essenzialmente una sorta di connessione. Inoltre, le connessioni distribuiscono i costi dell'inizializzazione su una serie di pacchetti. Una "connessione" nel contesto dell'IPsec è chiamata **SA** (*Security Association*).

Una SA è una connessione simplex fra due estremi e ha un identificatore di sicurezza associato. Se è necessario stabilire un traffico sicuro nelle due direzioni, saranno necessarie due SA. Gli identificatori di sicurezza sono trasportati da pacchetti che viaggiano su queste connessioni sicure, e vengono usati (all'arrivo dei pacchetti sicuri) per la ricerca delle chiavi e di altre informazioni rilevanti.

Le specifiche tecniche di IPsec contengono due parti principali. La prima parte descrive due nuove intestazioni che possono essere aggiunte ai pacchetti per contenere l'identificatore di sicurezza, i dati di controllo dell'integrità e altre informazioni. L'altra parte, **ISAKMP** (*Internet Security Association and Key Management Protocol*) concerne lo scambio delle chiavi.

Non tratteremo ulteriormente ISAKMP, in quanto (1) è estremamente complesso, (2) il suo protocollo principale, **IKE** (*Internet Key Exchange*), contiene degli errori molto gravi e quindi deve essere rimpiazzato (Perlman e Kaufman, 2000).

IPsec può essere utilizzato in una modalità scelta fra le due possibili. Nella **modalità trasporto** l'intestazione IPsec viene inserita subito dopo quella dell'IP. Il campo *Protocol* nell'intestazione IP è cambiato in modo da indicare che un'intestazione IPsec segue quella solita dell'IP (prima dell'intestazione TCP). L'intestazione IPsec contiene le informazioni di sicurezza: principalmente l'identificatore SA, un nuovo numero di sequenza, ed eventualmente un controllo d'integrità del campo payload. Nella **modalità tunnel** l'intero pacchetto IP, con l'intestazione compresa, viene incapsulato nel corpo di un nuovo pacchetto IP con un'intestazione IP completamente diversa. La modalità tunnel è utile quando il tunnel arriva in un posto diverso dalla sua destinazione finale. In alcune situazioni, il tunnel arriva a una macchina con un gateway sicuro, per esempio il firewall dell'azienda. Operando in questo modo, il firewall incapsula e decapsula i pacchetti che lo attraversano. Facendo terminare il tunnel su questa macchina sicura, le machine sulla LAN dell'azienda non devono essere a conoscenza dell'IPsec. Solo il firewall sarà a conoscenza del tunnel IPsec.

La modalità tunnel è utile anche nella situazione in cui si vuole aggregare un'insieme di connessioni TCP, per poi gestirle come un unico flusso cifrato. In questo modo si evita che un

intruso possa avere informazioni sul traffico: per esempio chi sta mandando dei pacchetti, a chi li sta mandando e quanti ne sta inviando. Consideriamo il caso in cui, durante una crisi militare, il traffico fra il Pentagono e la Casa Bianca diminuisca di colpo. Simultaneamente, il traffico fra il Pentagono e alcune installazioni militari nelle Montagne Rocciose aumenta della stessa quantità di cui il primo è diminuito. Un intruso potrebbe trarre delle informazioni utili da questi dati. Lo studio della distribuzione del flusso dei pacchetti, anche se cifrati, è chiamato **analisi del traffico**. La modalità tunnel fornisce un modo per aggirare parzialmente questo tipo di analisi. Lo svantaggio della modalità tunnel è dato dal fatto che aggiunge delle ulteriori intestazioni IP e quindi aumenta la dimensione del pacchetto in modo sostanziale. Al contrario, la modalità trasporto non cambia di molto la grandezza del pacchetto. La prima ulteriore intestazione è detta AH (*Authentication Header*). AH garantisce il controllo dell'integrità e la sicurezza contro gli attacchi di ricezione, ma non la segretezza (cioè non ha cifratura). L'uso di AH nella modalità trasporto è illustrato nella Figura 8.27. Nell'IPv4, AH viene messo fra l'intestazione IP (incluse le eventuali opzioni) e l'intestazione TCP. Nell'IPv6, AH è gestito come un'altra estensione dell'intestazione e trattata di conseguenza. In effetti, il suo formato è simile a quello di un'estensione standard dell'intestazione IPv6. Il campo payload può essere riempito fino a raggiungere una particolare lunghezza per facilitare l'algoritmo di autenticazione, come mostrato.

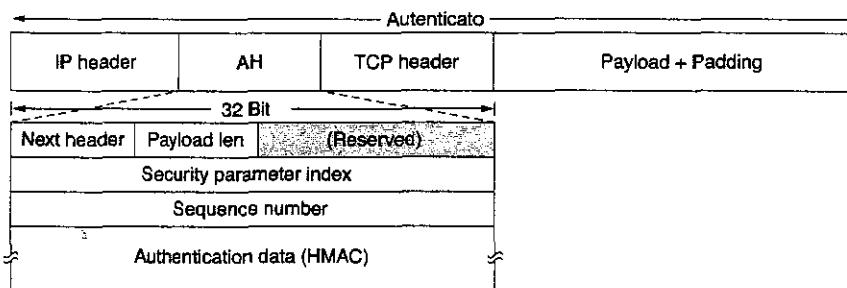


Figura 8.27. L'header di autenticazione IPsec nella modalità trasporto per IPv4.

Nell'intestazione AH, il campo *Next header field* serve per memorizzare il valore originale che aveva il campo *IP Protocol* prima di essere rimpiazzato con il valore 51. Quest'ultimo è usato per indicare che segue un'intestazione AH. Nella maggior parte dei casi, il valore sarà il codice per il TCP (6). Il campo *Payload length* contiene il numero di word a 32 bit nell'intestazione AH meno 2.

Il *Security parameter index* è l'identificatore della connessione. Viene inserito dal mittente per indicare un particolare record nel database del ricevente. Questo record contiene la chiave condivisa usata per questa connessione e altre informazioni sempre relative alla connessione. Se questo protocollo fosse stato inventato da ITU al posto di IETF, questo campo sarebbe stato chiamato *Virtual circuit number*.

Il campo *Sequence number* viene usato per attribuire un numero a tutti i pacchetti inviati in un SA. A ogni pacchetto viene attribuito un numero univoco, incluse le ritrasmissioni.

In altre parole, un pacchetto ritrasmesso prende un numero di sequenza diverso dall'originale (anche se il suo numero di sequenza TCP è lo stesso). Lo scopo di questo campo è quello di evitare attacchi di tipo ripetizione. Questi numeri di sequenza non sono riutilizzati in circolo: se vengono usate tutte le  $2^{32}$  possibili combinazioni, dovrà essere stabilito un nuovo SA per poter continuare la comunicazione.

Infine abbiamo *Authentication data*, che è un campo a lunghezza variabile che contiene la firma digitale del payload. L'algoritmo di firma elettronica da utilizzare è negoziato quando viene stabilito l'SA. Di solito in questo stadio non viene usata la crittografia a chiave pubblica, perché i pacchetti devono essere elaborati in modo estremamente rapido, mentre tutti gli algoritmi a chiave pubblica conosciuti sono troppo lenti per questo compito. Per questo motivo IPsec è basato sulla crittografia a chiave simmetrica. Il mittente e il destinatario stabiliscono una chiave comune prima d'instaurare la SA, che viene usata nel calcolo della firma. Un metodo semplice consiste nel calcolare l'hash sui contenuti del pacchetto e della chiave insieme. Ovviamente, la chiave condivisa non viene trasmessa.

Uno schema di questo tipo è chiamato HMAC (*Hashed Message Authentication Code*). È molto più veloce da calcolare della strategia alternativa, che consiste nell'usare prima SHA-1 per poi passare elaborare il risultato con RSA.

L'intestazione AH non permette la cifratura dei dati, quindi è utile principalmente quando è necessario un controllo d'integrità ma non serve la sicurezza. Una funzione importante di AH è data dal fatto che il controllo d'integrità copre alcuni campi dell'intestazione IP, in particolare quelli che non cambiano mentre il pacchetto viene trasmesso da router a router. Per esempio, il campo *Time to live* cambia in ogni salto di router e quindi non può essere incluso nel controllo d'integrità. L'indirizzo IP della sorgente viene incluso nel controllo, in questo modo diventa impossibile per un intruso falsificare l'origine del pacchetto.

L'intestazione IPsec alternativa è ESP (*Encapsulating Security Payload*). Il suo uso, sia per la modalità trasporto sia per quella tunnel, viene mostrato nella Figura 8.28.

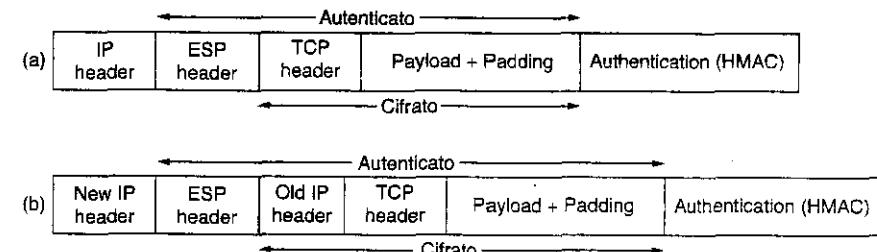


Figura 8.28. (a) ESP in modalità trasposto. (b) ESP in modalità tunnel.

L'intestazione ESP consiste di due word a 32 bit, che costituiscono i campi *security parameters index* e *sequence number*, con funzioni analoghe a quanto abbiamo discusso per AH. Una terza word che di solito segue le prime due (ma che tecnicamente non è parte dell'intestazione) è *initialization vector*, usato per la cifratura dei dati. Se non viene usata la cifratura, questo campo è omesso.

ESP fornisce un controllo d'integrità HMAC come AH, ma diversamente da quest'ultimo non lo include nell'intestazione; lo aggiunge invece in coda al campo payload, come mostrato nella Figura 8.28. Utilizzare HMAC in coda al payload comporta dei vantaggi nelle implementazioni hardware. In questo caso, HMAC può essere calcolato mentre i bit sono trasmessi verso l'interfaccia di rete, e poi aggiunto alla fine: per lo stesso motivo Ethernet e altre LAN hanno il CRC dopo la fine dei dati, piuttosto che all'inizio. Con AH, invece, il pacchetto va messo in un buffer e la firma calcolata prima di poterlo trasmettere, pertanto questo modo di operare può avere l'effetto di ridurre il numero di pacchetti/sec che possono essere trasmessi.

ESP riesce a fare tutto quello che AH può fare, ha anche delle funzioni aggiuntive ed è più efficiente. La domanda a questo punto è: perché interessarsi ad AH? La risposta ha origini soprattutto storiche: in origine AH si occupava solo d'integrità e ESP solo di segretezza. Successivamente, l'integrità è stata aggiunta a ESP. A questo punto, però, le persone che avevano progettato AH non volevano vederlo morire dopo tutto il lavoro che avevano fatto. Il loro unico punto a favore è che AH controlla parte dell'intestazione IP, mentre ESP non lo fa, ma questo è un argomento debole. Un'altra giustificazione debole è che un prodotto che supporti AH ma non ESP potrebbe avere minori problemi a ottenere una licenza per l'export in quanto non presenta funzioni di cifratura. AH verrà probabilmente eliminato in un prossimo futuro.

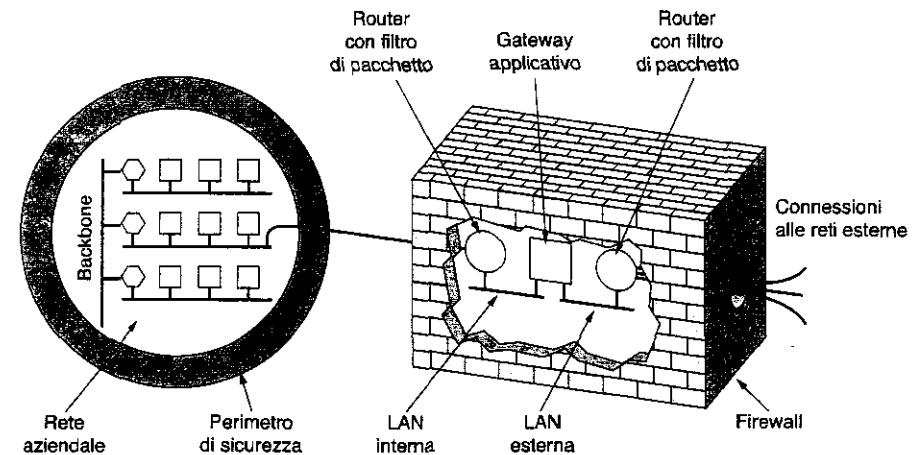
## 8.6.2 Firewall

La possibilità di connettere qualunque computer che si trova in qualunque posto a qualunque altro computer in una qualsiasi località, può essere visto allo stesso tempo come una comodità e come una fonte di problemi. Per le persone che si collegano da casa e navigano su Internet, si tratta di un divertimento. Per i manager della sicurezza nelle aziende è invece un incubo. La maggior parte delle aziende possiede una gran quantità d'informazioni online: segreti industriali, piani di sviluppo di prodotti, strategie di marketing, analisi finanziarie, ecc. La divulgazione di queste informazioni alla concorrenza può avere pesanti ripercussioni.

Oltre al pericolo di vedere fuoriuscire delle informazioni, ci sono anche i pericoli legati alla diffusione di informazioni verso l'interno. In particolare, virus, worm e altre "pesti digitali" possono aprire una falla nella sicurezza e distruggere dati importanti, oppure far sprecare una gran quantità del tempo agli amministratori di sistema che devono rimediare ai danni. Spesso questi problemi sono portati all'interno della rete da un dipendente disattento, che voleva semplicemente provare qualche nuovo bel gioco.

La conseguenza è la necessità d'instaurare dei meccanismi per tenere dentro i bit "buoni" e fuori quelli "cattivi". Un metodo è quello di usare IPsec. Questo approccio protegge i dati in transito fra i siti sicuri. IPsec non riesce però a fare nulla per tenere alla larga dalle LAN aziendali le pesti digitali e gli intrusi. Per vedere come si può realizzare questo obiettivo, dobbiamo parlare di firewall.

I **firewall** (letteralmente, muro di fuoco) sono una versione moderna del vecchio rimedio medievale per le emergenze di sicurezza: scavare un profondo fossato attorno al proprio castello. Questa struttura obbligava chiunque volesse entrare o uscire dal castello a passare per un singolo ponte levatoio, dove la polizia poteva facilmente eseguire le sue ispezioni.



**Figura 8.29.** Un firewall formato da due router con filtro di pacchetto (*packet filter*) e un gateway applicativo (*application gateway*).

ni. Con le reti è possibile realizzare lo stesso trucco: un'azienda può avere molte LAN connesse in modo arbitrario, ma tutto il traffico da e per l'azienda viene convogliato attraverso un ponte levatoio (il firewall), come mostrato nella Figura 8.29.

In questa configurazione il firewall ha due componenti: due router che fanno il filtraggio dei pacchetti e un gateway applicativo. Esistono anche delle configurazioni più semplici, ma questa struttura ha il vantaggio di obbligare ogni pacchetto a transitare attraverso due filtri e un gateway applicativo sia per entrare sia per uscire. Non esistono altre strade. Chi pensa che un solo controllo di sicurezza possa bastare, chiaramente non si è imbarcato su un volo di linea internazionale negli ultimi tempi.

Ogni **packet filter** (filtro di pacchetti) è un router standard equipaggiato con funzionalità extra. Queste funzionalità permettono di ispezionare ogni pacchetto in arrivo o in uscita. I pacchetti che rispondono a certi criteri vengono fatti passare normalmente, quelli che falliscono il test vengono scartati. Nella Figura 8.29 è molto probabile che il packet filter verso l'interno della LAN controlli i pacchetti in uscita, mentre quello sul lato esterno della LAN controlli i pacchetti in ingresso. I pacchetti, dopo aver attraversato il primo ostacolo, proseguono verso il gateway applicativo per un'ulteriore ispezione. Si utilizzano due packet filter su LAN differenti per essere sicuri che nessun pacchetto possa entrare o uscire senza essere passato attraverso il gateway applicativo: non c'è nessun percorso che gli passa intorno.

I packet filter sono tipicamente gestiti tramite delle tabelle configurate dall'amministratore di sistema. Queste tabelle elencano sorgenti e destinazioni accettabili e quelle bloccate, inoltre gestiscono le regole predefinite riguardo a cosa fare con i pacchetti che vengono o vanno verso altre macchine.

Nel caso comune delle impostazioni TCP/IP, la sorgente e la destinazione consistono di un indirizzo IP e di una porta. Le porte indicano quale servizio è desiderato. Per esempio, la porta TCP 23 è per il telnet, la porta 79 è per il finger, la 119 è per le news di USENET. Un'azienda può bloccare tutti i pacchetti entranti per tutti gli indirizzi IP in combinazione

con una o tutte queste porte. In questo modo, nessuno al di fuori dell'azienda può effettuare una login con telnet oppure cercare le persone con il daemon finger. Inoltre l'azienda evita che i suoi dipendenti spendano tutto il loro tempo a leggere le news di USENET. Bloccare i pacchetti in uscita è più difficile in quanto, sebbene molti siti si attengano alle convenzioni sui numeri standard delle porte, non c'è l'obbligo di farlo. Inoltre, per alcuni servizi come l'FTP (*File Transfer Protocol*), i numeri di porta vengono assegnati dinamicamente. Bloccare le connessioni UDP è ancora più difficile rispetto a quelle TCP, questo perché a priori si riescono ad avere troppo poche informazioni sull'attività delle connessioni UDP. Per questo motivo, molti packet filter sono configurati semplicemente per bandire completamente il traffico UDP. La seconda parte del firewall è costituita dal **gateway applicativo**. Il gateway non guarda più i pacchetti in quanto tali, ma opera allo strato applicativo. Per esempio, è possibile installare un gateway di posta, per esaminare tutti i messaggi in arrivo o in uscita. Il gateway esamina ciascun messaggio e decide se trasmetterlo o scartarlo in base ai campi di intestazione, alla dimensione del messaggio, o anche a seconda del contenuto (per esempio, in un'installazione militare, la presenza delle parole "nucleare" o "bomba" possono far scatenare alcune misure particolari).

Ogni sito è libero d'installare uno o più gateway applicativi per applicazioni specifiche. È prassi comune, per le organizzazioni molto sensibili alla sicurezza, permettere l'ingresso e l'uscita della posta elettronica e l'uso del Web, ma di proibire tutto il resto, in quanto considerato troppo rischioso. Questa soluzione, in combinazione con la crittografia e il packet filtering, offre una certa sicurezza, al costo di qualche scomodità.

Anche quando il firewall è configurato perfettamente, rimangono ancora molti problemi di sicurezza. Per esempio, se un firewall è configurato per permettere l'ingresso dei pacchetti solo da alcune specifiche reti (per esempio le altre sedi dell'azienda), un intruso dall'esterno del firewall può inserire un indirizzo di sorgente falso e quindi aggirare il controllo. Se una persona, dall'interno, vuole spedire all'esterno dei documenti segreti, li può cifrare oppure fotografare per poi spedirli come file JPEG, che aggirano qualunque filtro sulle parole. Vedremo più avanti che il 70% degli attacchi arrivano da dentro il firewall, spesso da parte di dipendenti malcontenti (Schneier, 2000).

Inoltre esiste un'intera classe di attacchi che i firewall non riescono a trattare. L'idea alla base dei firewall è quella di prevenire l'ingresso degli intrusi e l'uscita dei dati segreti. Sfortunatamente, ci sono persone che non hanno niente di meglio da fare che cercare di bloccare la funzionalità di certi siti. Questo viene fatto inviando una grande quantità di pacchetti legittimi ai siti bersaglio, finché questi non collassano sotto il peso del carico di lavoro. Per esempio, per "azzoppare" un sito, un intruso può inviare un pacchetto TCP SYN, per stabilire una connessione. Il sito allocherà quindi uno slot per la connessione e invierà come risposta un pacchetto SYN + ACK. Se l'intruso non risponde, lo slot viene mantenuto per alcuni secondi finché non si ottiene un timeout. Se l'intruso invia migliaia di richieste di connessione, tutti gli slot vengono riempiti e le connessioni legittime non possono più essere aperte. Gli attacchi che hanno come scopo quello di fermare l'operatività dell'obiettivo, e non quello di rubare dei dati, sono chiamati **DoS (Denial of Service)**. Normalmente i pacchetti con le richieste hanno un indirizzo sorgente falso, in questo modo l'intruso non può essere identificato facilmente.

Una variante più pericolosa è possibile quando un intruso è già penetrato in centinaia di macchine sparse per il mondo, e ordina a tutte queste macchine di attaccare simultaneamente un solo obiettivo. Questo approccio aumenta il potere di fuoco dell'intruso, oltre a ridurre le possibilità di una sua identificazione, visto che i pacchetti arrivano da una gran quantità di macchine che appartengono a utenti ignari della situazione. Attacchi di questo tipo sono detti attacchi **DDoS (Distributed Denial of Service)**. È difficile proteggersi contro questo tipo di attacchi. Anche se la macchina attaccata identifica le false richieste, ci vuole sempre un po' di tempo per elaborarle e poi scartarle. Se questo tipo di richieste arrivano con una frequenza abbastanza alta, la CPU della macchina attaccata sarà costretta a spendere tutto il suo tempo per gestirle.

### 8.6.3 VPN, reti private virtuali

Molte aziende hanno uffici e stabilimenti situati in diverse città, a volte anche in diversi stati. Prima che esistessero le reti pubbliche per i dati, le aziende affittavano dalla compagnia telefonica le linee per collegare fra loro alcune o tutte le loro sedi. Alcune aziende lo fanno ancora. Una rete costruita con i computer aziendali e le linee telefoniche affittate è detta **rete privata**. Un esempio di rete privata che connette tre siti è mostrata nella Figura 8.30(a).

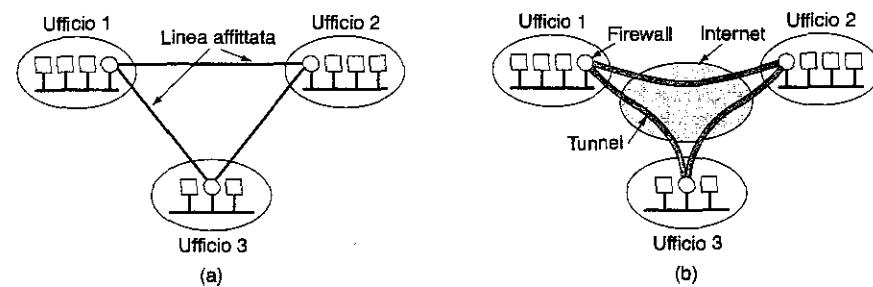


Figura 8.30. (a) Rete privata con linee affittate. (b) Virtual Private Network.

Le reti private funzionano bene e sono molto sicure. Quando le uniche linee presenti sono affittate non è possibile che il traffico di rete possa fuoriuscire dall'azienda, salvo che un intruso riesca a intercettare fisicamente le comunicazioni, il che non è affatto facile. Il problema delle reti private è dato dal loro costo. Infatti, affittare una singola linea T1 può costare migliaia di dollari al mese, affittare una T3 molto di più. Quando arrivarono sulla scena le reti pubbliche per i dati, e successivamente Internet, molte aziende vollero spostare il loro traffico dati (e se possibile anche vocale) sulle reti pubbliche, ma senza abbandonare la sicurezza delle linee private.

Questa domanda portò in breve tempo all'invenzione delle **VPN (Virtual Private Network)**. Le VPN permettono di sovrapporre delle reti sopra le reti pubbliche, ma senza abbandonare le proprietà di sicurezza tipiche delle reti private. Sono chiamate "virtuali" perché sono una mera illusione, così come i circuiti virtuali non sono veri circuiti e la memoria virtuale non è vera memoria.

Le VPN possono essere costruite sopra le reti ATM (o frame relay), ma un approccio che diventa sempre più comune consiste nel costruire le VPN direttamente sopra Internet. Una tipica architettura consiste nell'avere un firewall per ogni ufficio e creare dei tunnel attraverso Internet fra tutte le coppie di uffici, come mostrato nella Figura 8.30(b). Se viene usato l'IPsec per il tunneling, è possibile aggregare tutto il traffico fra ogni possibile coppia di uffici in una singola SA con autenticazione e cifratura. Questo fornisce controllo dell'integrità, segretezza e anche una considerevole immunità all'analisi del traffico.

Ogni coppia di firewall deve negoziare allo startup i parametri della sua SA: i servizi, le modalità, gli algoritmi e le chiavi. Molti firewall sono progettati per gestire le VPN, e ciò vale anche per alcuni normali router. Visto che i firewall sono impiegati principalmente nell'ambito della sicurezza, è naturale che i tunnel delle VPN comincino e finiscano su dei firewall. In questo modo si realizza una chiara separazione fra Internet e l'azienda. Firewall, VPN e IPsec su ESP in modalità tunnel costituiscono una combinazione naturale e molto usata nella pratica. Il traffico comincia a scorrere dopo che sono state stabilite le SA. Per un router su Internet, un pacchetto che viaggia attraverso un tunnel VPN non è altro che un normale pacchetto. L'unica cosa strana è la presenza di un'intestazione IPsec dopo la normale intestazione dell'IP. Visto che le intestazioni extra non hanno effetto sulla procedura di forwarding, i router semplicemente le ignorano.

Impostare le VPN in questo modo presenta il vantaggio fondamentale della completa trasparenza delle operazioni nei confronti degli utenti e del loro software. I firewall impostano e gestiscono le SA. L'unica persona che è al corrente di queste impostazioni è l'amministratore di sistema che deve configurare e gestire i firewall. Per tutti gli altri utenti, è come se ci fosse una rete privata su linea dedicata. Per ulteriori informazioni sulle VPN si consiglia (Brown, 1999; e Izzo, 2000).

#### 8.6.4 Sicurezza wireless

È sorprendentemente facile disegnare un sistema con VPN e firewall che risulta completamente sicuro dal punto di vista logico, mentre in pratica fa acqua da tutte le parti. Questa situazione si può realizzare quando alcuni computer coinvolti sono wireless, quindi usano onde radio che attraversano il firewall in entrambe le direzioni. La portata delle reti 802.11 è normalmente di alcune centinaia di metri. Chiunque volesse spiare le comunicazioni di un'azienda può quindi parcheggiare la sua macchina nel parcheggio dei dipendenti al mattino, lasciare in macchina un laptop con accesso all'802.11 impostato per registrare tutte le comunicazioni che riesce a ricevere, e quindi allontanarsi indisturbato. Alla fine della giornata l'hard disk sarà pieno di materiale interessante. In teoria questa fuga di dati non dovrebbe accadere. In teoria, però, non dovrebbero accadere neanche le rapine alle banche.

Molti dei problemi di sicurezza delle reti wireless sono causati dal fatto che i produttori delle stazioni di trasmissione wireless (gli access point) vogliono rendere questi prodotti semplici da usare per gli utenti finali. Di solito, gli access point sono costruiti per funzionare direttamente alla prima accensione e senza particolari configurazioni. Questo si ottiene al costo di una scarsa sicurezza e del pericolo di vedere i propri dati trasmessi a tutti quelli che si trovano nel raggio di portata dell'apparecchio. Se l'access point è collegato alla ethernet, avremo come risultato che tutto il traffico ethernet sarà disponibile anche nel

parcheggio. Le reti wireless sono una manna per le spie: dati a volontà senza dover fare nessun lavoro. È chiaro, a questo punto, che la sicurezza diventa ancora più importante per i sistemi wireless di quanto non lo sia già per le reti cablate. In questo paragrafo vedremo come alcune reti wireless gestiscono i problemi di sicurezza. Per ulteriori approfondimenti si rimanda a (Nichols e Lekkas, 2002).

#### Sicurezza di 802.11

Lo standard 802.11 prescrive un protocollo di sicurezza per lo strato data link chiamato **WEP** (*Wired Equivalent Privacy*), che è stato progettato per portare la sicurezza di una rete LAN wireless allo stesso livello di quella delle reti cablate. Visto che lo standard di sicurezza per una rete cablata corrisponde a non avere nessuna sicurezza, l'obiettivo è facile da raggiungere e WEP ci riesce, come vedremo nel seguito.

Quando la sicurezza di 802.11 viene attivata, il risultato è che ogni stazione stabilisce una chiave segreta condivisa con la stazione base. Il modo in cui debbano essere distribuite le chiavi non è specificato nello standard. Per esempio, possono essere già inserite nell'apparecchio dal costruttore, oppure sono scambiate tramite un rete cablata prima d'iniziare la comunicazione. Un'ultima possibilità consiste nell'avere la stazione base, oppure una macchina utente, che genera un numero casuale e lo invia all'altra parte con una trasmissione wireless cifrata, usando la chiave pubblica del ricevente. Una volta stabilite, le chiavi rimangono stabili per mesi o anche anni. La cifratura WEP usa uno stream cipher basato sull'algoritmo RC4. RC4 fu sviluppato da Ronald Rivest e tenuto segreto finché fu possibile, cosa che terminò con la sua pubblicazione su Internet nel 1994. Come abbiamo avuto modo di dire in precedenza, è praticamente impossibile mantenere segreti gli algoritmi, anche se l'obiettivo è solamente la salvaguardia della proprietà intellettuale (come in questo caso), piuttosto che ottenere una sicurezza per occultamento delle informazioni (che comunque non riguardava RC4). Nel WEP, RC4 serve per generare un keystream che viene applicato in XOR al testo in chiaro per produrre il testo cifrato.

Il payload di ogni pacchetto viene cifrato con il metodo della Figura 8.31. Per prima cosa viene calcolato il checksum del payload usando il CRC-32 polinomiale. Il checksum viene aggiunto al payload per formare il testo in chiaro da inviare all'algoritmo di cifratura. A questo punto al testo in chiaro viene applicata in XOR una parte del keystream pari alla sua lunghezza. Il risultato finale è il testo cifrato. L'IV utilizzato per inizializzare il keystream RC4 viene inviato insieme al testo cifrato. Il destinatario, quando riceve il pacchetto, per prima cosa estrae il payload cifrato, poi genera il keystream a partire dalla chiave segreta condivisa e dall'IV che ha appena ricevuto, quindi calcola lo XOR fra il testo cifrato e il keystream. In questo modo il destinatario ottiene il testo in chiaro e può procedere a verificarne il checksum per assicurarsi che non sia stato oggetto di modifiche indesiderate.

Anche se a un primo esame questo approccio può sembrare sicuro, è già stato scoperto e pubblicato un metodo per violarlo (Borisov et al., 2001). Nel seguito riassumeremo i risultati di tale articolo. Come prima cosa, notiamo che un numero sorprendente d'installazioni usa la stessa chiave condivisa per tutti gli utenti, in questo modo ogni utente può leggere tutto il traffico degli altri. Questo è analogo a ciò che accade anche con ethernet, ma non è certamente molto sicuro.

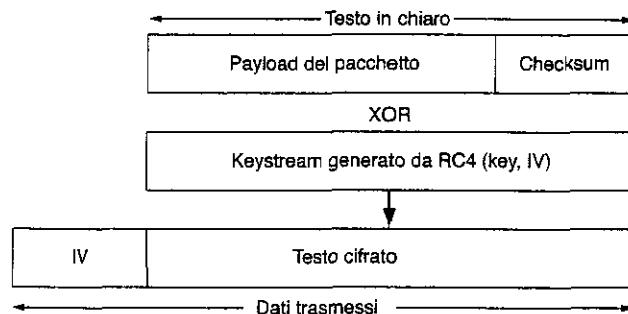


Figura 8.31. Incapsulamento dei pacchetti con WEP.

WEP può essere attaccato anche nel caso in cui ogni utente abbia la sua chiave. Visto che le chiavi rimangono stabili per un lungo periodo di tempo, lo standard WEP raccomanda (ma non obbliga) che l'IV venga cambiato a ogni pacchetto. Questo per evitare l'attacco per riutilizzo del keystream discusso nel Paragrafo 8.2.3. Sfortunatamente, molte schede 802.11 per notebook effettuano il reset a 0 dell'IV ogni volta che la scheda viene inserita nel computer, e poi incrementano IV di 1 all'invio di ogni pacchetto. Visto che questo tipo di schede viene spesso disconnesso e reinserito, il risultato è che sono più frequenti i pacchetti con valori bassi di IV. Se Trudy riesce a collezionare diversi pacchetti inviati dallo stesso utente con un dato valore di IV (valore che viene trasmesso in chiaro insieme al pacchetto), riuscirà a calcolare lo XOR di due particolari testi in chiaro e probabilmente potrà usare questa informazione per forzare il cifrario.

Anche nel caso in cui la scheda 802.11 selezioni un numero casuale per gli IV di ogni pacchetto, visto che l'IV usa 24 bit, dopo aver inviato  $2^{24}$  pacchetti ci sarà sicuramente il riutilizzo di alcuni valori di IV. La situazione nel caso di IV generati casualmente è anche peggiore, in quanto il numero medio di pacchetti che devono essere trasmessi prima di avere una ripetizione è solamente 5.000, come può essere calcolato con la stessa logica spiegata nel Paragrafo 8.4.4 per l'attacco del compleanno. Quindi, se Trudy ascolta la rete per qualche minuto, è praticamente sicura di riuscire a catturare due pacchetti con lo stesso IV e la stessa chiave. Calcolando lo XOR dei testi cifrati, Trudy ottiene anche lo XOR dei testi in chiaro corrispondenti. Da questi dati è possibile ottenere i testi in chiaro usando diverse tecniche di attacco. Con un po' più di lavoro è possibile anche ottenere il keystream corrispondente per quel dato IV. Trudy può continuare questo tipo di attacco per un po' di tempo e quindi arrivare a compilare un dizionario dei keystream corrispondenti a diversi valori di IV. Quando un IV è stato forzato, tutti i pacchetti che in futuro verranno inviati con quell'IV possono essere completamente decifrati (questo vale anche per i pacchetti già trasmessi).

Trudy, dopo aver determinato una coppia valida di (IV, keystream), può usare l'informazione per generare tutti i pacchetti che desidera e quindi interferire con le comunicazioni. Ciò è possibile perché gli IV della comunicazione legittima vengono comunque generati in modo casuale. Teoricamente, il ricevente potrebbe notare che un gran numero di pacchetti improvvisamente utilizzano lo stesso IV, ma (1) WEP lo permette, (2) nessuno con-

trolla questo tipo di comportamento. Infine notiamo che il CRC non serve a molto in questo caso, in quanto Trudy può cambiare tutto il payload ed effettuare anche il corrispondente cambiamento del CRC, senza neanche dover rimuovere la cifratura. Riassumendo: abbiamo visto che forzare la sicurezza di 802.11 richiede una procedura relativamente semplice, e questo senza neanche aver elencato tutti gli attacchi descritti da Borisov et al. Nell'agosto del 2001, un mese dopo la pubblicazione dell'articolo di Borisov et al., fu pubblicato un altro devastante attacco al WEP (Fluhrer et al., 2001). In questo caso gli autori descrivevano una debolezza crittografica dell'RC4. Fluhrer et al. avevano scoperto che per molte chiavi era possibile derivare alcuni bit della chiave stessa a partire dal keystream. Quando questo tipo di attacco viene ripetuto, diventa possibile ottenere l'intera chiave con uno sforzo modesto. Essendo essenzialmente orientati alla teoria, Fluhrer et al. non tentarono di mettere in pratica le loro scoperte su un caso reale di LAN 802.11.

Un approccio differente fu preso da uno studente e due ricercatori dei laboratori AT&T che, venuti a conoscenza dei risultati di Fluhrer et al., decisero di metterli in pratica (Stubblefield et al. 2002). Nell'arco di una settimana arrivarono al punto di riuscire a forzare una chiave a 128 bit su una LAN 802.11 di produzione. Questo risultato fu ottenuto nonostante la maggior parte del tempo fosse stata spesa per cercare la scheda 802.11 più economica, chiedere il permesso per acquistarla, quindi installarla e testarla. Il tempo speso effettivamente a programmare l'attacco fu di sole due ore.

Quando i tre annunciarono il loro risultato, la CNN fece uscire una storia intitolata "La cifratura wireless sconfitta da un attacco accessibile a tutti", dove alcuni guru del settore tentavano di minimizzare la portata di questo lavoro affermando che si trattava di un'operazione banale, dati i risultati Fluhrer et al. Anche se questo commento era tecnicamente corretto, rimane il fatto che gli sforzi combinati dei due team avevano dimostrato una falla mortale per la sicurezza di WEP e 802.11.

Il 7 settembre 2001, IEEE reagì al fatto che ormai WEP era stato forzato completamente con una breve nota in cui venivano esposti 6 punti che possono essere approssimativamente riassunti come segue:

1. vi avevamo già detto che la sicurezza di WEP non era migliore di quella di Ethernet
2. una minaccia ben peggiore è data dal non abilitare del tutto la sicurezza
3. provate a usare altri tipi di sicurezza (per esempio, sicurezza nello strato di trasporto)
4. la prossima versione, 802.11i, avrà una sicurezza migliorata
5. le certificazioni future richiederanno l'uso di 802.11i
6. vedremo cosa è possibile fare in attesa dell'arrivo di 802.11i.

Abbiamo voluto descrivere questa storia nei dettagli per dimostrare come non sia facile implementare in modo corretto la sicurezza, anche per degli esperti.

### Sicurezza di Bluetooth

Bluetooth ha un raggio di azione considerevolmente più corto di 802.11, quindi non può essere attaccato dal parcheggio. Comunque la sicurezza rimane un punto importante da dis-

cutere anche in questo caso. Immaginiamo, per esempio, che il computer di Alice abbia una tastiera Bluetooth senza fili. Supponiamo anche che Trudy si trovi nell'ufficio adiacente. In assenza di sicurezza, Trudy potrà leggere tutto quello che Alice scrive, incluse tutte le mail che spedisce. Potrebbe anche intercettare tutto quello che il computer di Alice invia alla stampante Bluetooth che si trova nei paraggi (per esempio, le e-mail in arrivo e i documenti confidenziali). Fortunatamente, Bluetooth ha uno schema di sicurezza complesso per cercare di sviare tutte le Trudy del mondo. Ne descriveremo le principali caratteristiche.

Bluetooth ha tre modalità di sicurezza, che vanno da nessuna sicurezza a una completa cifratura dei dati e controllo dell'integrità. Come nel caso di 802.11, se la sicurezza viene disabilitata (il comportamento predefinito) non c'è nessuna sicurezza. La maggior parte degli utenti mantiene la sicurezza disabilitata finché non viene scoperta un'importante fuga di dati, a quel punto attiva la sicurezza. Nel mondo agricolo, questo approccio è noto come chiudere la stalla quando i buoi sono già scappati.

Bluetooth presenta soluzioni di sicurezza su più strati. Nello strato fisico, il salto di frequenza (*frequency hopping*) fornisce un po' di sicurezza. Visto che ogni dispositivo Bluetooth che si muove all'interno di una piconet deve essere informato della sequenza di salto di frequenza, questa sequenza ovviamente non è segreta. La vera sicurezza inizia quando un nuovo dispositivo slave chiede un canale al master. Si suppone che i due dispositivi abbiano già effettuato uno scambio di chiave segreta condivisa. In alcuni casi questa è già preconfezionata in entrambi dispositivi (per esempio per gli auricolari venduti in abbinamento al telefono cellulare); in altri casi, un dispositivo (per esempio l'auricolare) ha una chiave già inserita al suo interno che l'utente deve inserire anche dentro l'altro dispositivo (per esempio il telefono cellulare) sotto forma di un numero decimale. Queste chiavi condivise sono dette **passkey**.

Per stabilire un canale, master e slave effettuano il controllo per vedere se l'altro conosce la passkey. In caso affermativo, negoziano se il canale deve essere cifrato e se bisogna effettuare il controllo d'integrità, o entrambe le cose. Quindi scelgono una chiave di sessione casuale a 128 bit, di cui alcuni bit possono essere resi pubblici. Lo scopo di questo indebolimento della chiave è di rendere il protocollo utilizzabile anche in quegli stati per cui la legge impone delle restrizioni all'esportazione o proibisce l'uso di chiavi più lunghe di quelle che le autorità governative preposte riescono a forzare.

La cifratura usa uno stream cipher detto  $E_0$ , il controllo di integrità usa **SAFER+**. Entrambi sono cifrari a blocco tradizionali con chiave simmetrica. SAFER+ fu inviato anche al concorso per AES, ma venne eliminato al primo round perché più lento degli altri candidati. Bluetooth fu finalizzato prima della decisione definitiva sul cifrario AES, altrimenti sarebbe stato scelto molto probabilmente Rijndael anche per Bluetooth.

Il meccanismo di cifratura con lo stream cipher è mostrato nella Figura 8.14, in cui si vede come il testo in chiaro viene applicato in XOR con il keystream per generare il testo cifrato. Sfortunatamente,  $E_0$  stesso (come RC4) potrebbe avere delle debolezze fatali (Jakobsson e Wetzel, 2001). Anche se non è ancora stato forzato, al momento in cui scriviamo questo libro preoccupano parecchio (Biryukov et al. 2000) le sue similitudini con il cifrario A5/1, il cui spettacolare fallimento compromette tutto il traffico telefonico GSM. Spesso la gente si sorprende (incluso anche l'autore di questo libro) di come nel perenne gioco al gatto e al topo fra i crittografi e i criptoanalisti, i criptoanalisti siano così spesso la parte vincente.

Un altro punto riguardante la sicurezza è dato dal fatto che Bluetooth autentica solo i dispositivi e non gli utenti, quindi il furto di un dispositivo Bluetooth può dare al ladro l'accesso ai dati finanziari, o comunque riservati, dell'utente. Bluetooth però implementa la sicurezza anche negli strati superiori, quindi nel caso in cui ci sia una violazione allo strato data link, rimane ancora un po' di sicurezza, specialmente per le applicazioni che richiedono l'inserimento manuale dalla tastiera di un PIN per poter completare la transazione.

### Sicurezza del WAP 2.0

In linea generale, il WAP Forum ha imparato dal WAP 1.0 le conseguenze dell'adozione di protocolli non standard per i vari strati. WAP 2.0 usa in larga misura dei protocolli standard per tutti gli strati, e la sicurezza non fa eccezione. Visto che WAP 2.0 è basato su IP, supporta in pieno l'uso di IPsec nello strato network. Nello strato trasporto le connessioni TCP si possono proteggere con TLS, uno standard IETF che studieremo più avanti in questo capitolo. A un livello più alto, WAP 2.0 usa la connessione client http, come definita in RFC 2617. Librerie crittografiche allo strato applicativo forniscono il controllo d'integrità e di non-riprudio. Visto che WAP 2.0 è basato su standard, possiamo affermare che in linea generale i suoi servizi di sicurezza, e in particolare privacy, autenticazione, controllo d'integrità e non-riprudio possono avere un miglior successo dei corrispondenti servizi per 802.11 e Bluetooth.

## 8.7 Protocolli di autenticazione

L'**autenticazione** è la tecnica usata dai processi per verificare che la loro controparte nella comunicazione sia veramente chi dice di essere, e non un impostore. La verifica dell'identità di un processo remoto, per combattere gli intrusi attivi e in malafede, è un compito sorprendentemente difficile, che richiede dei protocolli crittografici complessi. In questo paragrafo studieremo alcuni dei molti protocolli di autenticazione usati per le reti insicure. Per inciso, alcune persone confondono l'autorizzazione con l'autenticazione. L'autenticazione riguarda il problema di assicurare l'identità del processo con cui si sta comunicando. L'autorizzazione si occupa, invece, di stabilire che cosa può fare un processo. Per esempio, consideriamo un processo client che contatta un file server e gli comunica: sono il processo di Scott e voglio cancellare il file *ricettario.old*. Il file server ha bisogno delle risposte a queste due domande:

1. la richiesta viene veramente da un processo di Scott (autenticazione)?
2. Scott è autorizzato a cancellare *ricettario.old* (autorizzazione)?

La richiesta può essere evasa soltanto dopo che entrambe le domande hanno ottenuto una risposta certa e affermativa. La prima domanda è quella veramente cruciale. Una volta che il file server sa con chi sta parlando, il controllo dell'autorizzazione consiste solamente in una ricerca dentro tabelle locali o in un database. Per questo motivo, in questo paragrafo ci concentreremo sull'autenticazione.

Il modello generale utilizzato da tutti i protocolli di autenticazione è il seguente: Alice comincia inviando un messaggio a Bob o a una terza persona considerata fidata, detta **KDC** (*Key Distribution Center*).

*istribution Center). In seguito è previsto lo scambio di molti altri messaggi, in varie direzioni, che possono essere intercettati, modificati o rinviati da parte di Trudy per ingannare (o comunque ostacolare) Alice e Bob. Quando il protocollo è stato completato, Alice però è sicura di essere in comunicazione con Bob e Bob è sicuro di essere in comunicazione con Alice. Inoltre, in molti protocolli, Alice e Bob stabiliscono anche una chiave segreta, detta **chiave di sessione (session key)**, da usare per continuare la loro conversazione. Nella realtà per motivi di prestazioni tutto il traffico è cifrato con la crittografia a chiave simmetrica (di solito con AES o DES triplo), mentre la crittografia a chiave pubblica è usata largamente nei protocolli di autenticazione per stabilire la chiave di sessione.*

Le ragioni per cui si usa una differente chiave casuale per ciascuna nuova connessione sono molteplici: minimizzare la quantità di traffico inviato usando le chiavi permanenti degli utenti (segrete oppure pubbliche), ridurre la quantità di testo cifrato che può cadere nelle mani di un intruso, e minimizzare i rischi connessi alla situazione in cui un processo va in crash e il relativo core dump cade nelle mani sbagliate. In questo caso, si spera che l'unica chiave presente nel dump sia la chiave di sessione. Tutte le chiavi permanenti dovrebbero essere già state attentamente azzerate dopo che la sessione è stata attivata.

## 1 Autenticazione basata su un segreto condiviso

Il nostro primo protocollo di autenticazione, supponiamo che Alice e Bob abbiano già una chiave segreta nota a entrambi,  $K_{AB}$ . La chiave è stata condivisa (per esempio) per telefono o di persona, ma comunque non sulla rete, che si suppone insicura.

Questo protocollo si basa su un principio comune a molti altri protocolli di autenticazione: un utente invia un numero casuale all'altro, che trasforma il numero con un algoritmo partire e quindi trasmette il risultato. Questo tipo di protocolli è detto **challenge-response**. La funzione che useremo per questo protocollo e per i successivi è:

$A, B$  sono le identità di Alice e Bob

$R_i$  sono le richieste (*challenge*), dove l'indice  $i$  identifica il richiedente

$K_i$  sono le chiavi, dove  $i$  indica il possessore

$K_S$  è la chiave di sessione.

La sequenza dei messaggi scambiati nel nostro primo protocollo di autenticazione a chiave condivisa è mostrata nella Figura 8.32. Nel messaggio 1, Alice invia a Bob la sua identità  $A$  in una modalità che Bob è in grado di capire. Bob, ovviamente, non può sapere se il messaggio viene da Alice o da Trudy, quindi crea una richiesta per Alice nella forma di un numero casuale grande,  $R_B$ , che invia in chiaro nel messaggio 2. I numeri casuali usati una sola volta per un protocollo challenge-response come quello descritto sono detti **nonce**. Alice produce il messaggio servendosi della chiave condivisa con Bob e inviando il messaggio 3 con il testo cifrato  $K_{AB}(R_B)$ . Quando Bob riceve il messaggio, sa immediatamente che proviene da Alice, in quanto Trudy non conosce  $K_{AB}$  e quindi non può aver generato il messaggio 3. Visto che  $R_B$  è scelto in modo casuale da uno spazio molto grande (per esempio, amo pensare a un numero casuale di 128 bit) è molto improbabile che Trudy possa aver indovinato la risposta corretta alla richiesta contenuta nel messaggio 2.

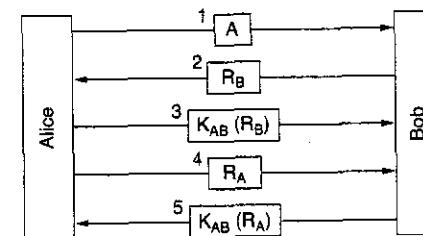


Figura 8.32. Autenticazione a due vie con protocollo challenge-response.

A questo punto Bob è sicuro di essere in comunicazione con Alice, ma Alice non è ancora sicura di niente. Per quanto ne sa Alice, Trudy potrebbe aver intercettato il messaggio 1 e aver inviato la risposta  $R_B$ . Magari Bob è morto la notte prima. Per scoprire con chi sta parlando, Alice sceglie un numero casuale  $R_A$ , e lo invia a Bob in chiaro nel messaggio 4. Quando Bob risponde con  $K_{AB}(R_A)$ , Alice sa di essere veramente in comunicazione con Bob. A questo punto è possibile stabilire una chiave di sessione: Alice ne sceglie una,  $K_S$ , e la manda a Bob cifrata con  $K_{AB}$ .

Il protocollo della Figura 8.32 contiene cinque messaggi. Vediamo se riusciamo a eliminare qualche messaggio in modo intelligente. Un approccio è illustrato nella Figura 8.33, dove Alice inizia il protocollo challenge-response senza aspettare che sia Bob a farlo. Analogamente, Bob invia la sua richiesta mentre risponde a quella di Alice. L'intero protocollo si riduce quindi a tre messaggi, invece di cinque.

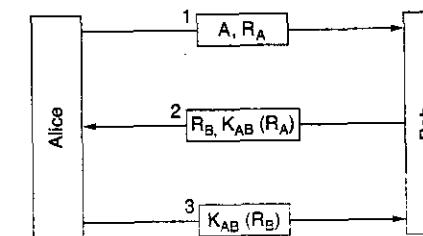


Figura 8.33. Un protocollo di autenticazione a due vie ridotto.

Il nuovo protocollo è migliore di quello originale? In un certo senso sì: è più corto. Sfortunatamente è anche sbagliato. In certe circostanze, Trudy riesce ad attaccare il protocollo usando una tattica nota come **attacco per riflessione**. In particolare, Trudy può attaccare il protocollo se è possibile aprire più sessioni simultanee con Bob. Questa situazione si verifica, per esempio, se Bob è una banca pronta ad accettare molte connessioni simultanee dagli sportelli bancomat.

Un attacco per riflessione è mostrato nella Figura 8.34. Comincia con Trudy che finge di essere Alice e invia  $R_T$ . Bob risponde come al solito con una richiesta,  $R_B$ . Adesso Trudy è bloccata. Cosa può fare, dato che non conosce  $K_{AB}(R_B)$ ?

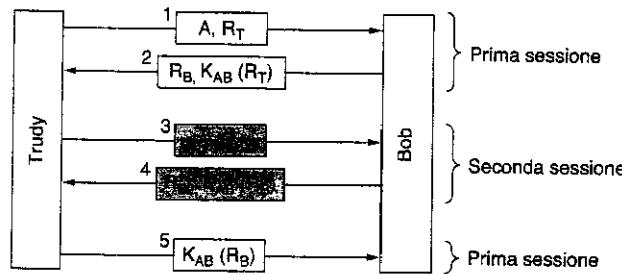


Figura 8.34. L'attacco per riflessione.

Può aprire una seconda sessione con il messaggio 3, fornendo come richiesta la  $R_B$  presa dal messaggio 2. Bob cifra con calma la  $R_B$  e invia  $K_{AB}(R_B)$  nel messaggio 4. Nella figura, i messaggi della seconda sessione sono in un rettangolo ombreggiato per evidenziarli. A questo punto Trudy ha l'informazione mancante, e può completare la prima sessione e interrompere la seconda. Bob adesso è convinto che Trudy sia Alice, perciò quando questa domanda il suo estratto conto, Bob lo trasmette senza indugi. Quando Trudy chiede a Bob di trasferire tutti i soldi su un conto segreto in Svizzera, Bob eseguirà anche questo ordine senza esitazioni. Il morale della storia è:

*Progettare un protocollo di autenticazione corretto è più difficile di quanto non sembri a prima vista.*

*Le quattro regole generali seguenti sono spesso di aiuto.*

1. Fare in modo che chi inizia l'autenticazione provi la sua identità prima che lo faccia chi risponde. Nel nostro caso, Bob invia delle informazioni importanti prima che Trudy abbia dato chiara evidenza della sua identità.
2. Fare sì che i due soggetti, cioè chi inizia l'autenticazione e chi risponde, usino due chiavi differenti per provare la loro identità, anche se ciò significa usare due chiavi condivise  $K_{AB}$  e  $K'_{AB}$ .
3. Far sì che i due soggetti utilizzino, per le richieste, numeri presi da insiemi diversi. Per esempio, chi inizia l'autenticazione può usare i numeri pari e chi risponde quelli dispari.
4. Rendere il protocollo resistente ad attacchi che coinvolgono una seconda sessione parallela, dove le informazioni ottenute da una sessione possono essere usate per l'altra.

Se anche una sola di queste regole non è rispettata, il protocollo può essere spesso forzato. Nel nostro caso vengono violate tutte e quattro le regole, con conseguenze disastrose.

Torniamo indietro e diamo un'occhiata più da vicino alla Figura 8.32. Siamo sicuri che il protocollo non sia soggetto ad attacchi per riflessione? La risposta è: dipende. La questione è delicata. Trudy era in grado di forzare il nostro protocollo con un attacco per riflessione, in quanto le era possibile aprire una seconda sessione con Bob per ingannarlo e quindi farlo rispondere alla sua stessa domanda. Cosa succederebbe nel caso in cui Alice fosse un computer in grado di accettare più sessioni, e non una persona seduta davanti a un computer? Vediamo quello che Trudy può fare.

Per vedere come funziona l'attacco di Trudy, guardiamo la Figura 8.35. Alice comincia annunciando la sua identità nel messaggio 1. Trudy intercetta il messaggio e comincia la sua sessione con il messaggio 2, pretendendo di essere Bob. Di nuovo, abbiamo inserito i messaggi della sessione 2 in un rettangolo ombreggiato. Alice risponde al messaggio 2 dicendo: tu dici di essere Bob? Provalo con il messaggio 3. A questo punto Trudy è bloccata, perché non riesce a provare di essere Bob.

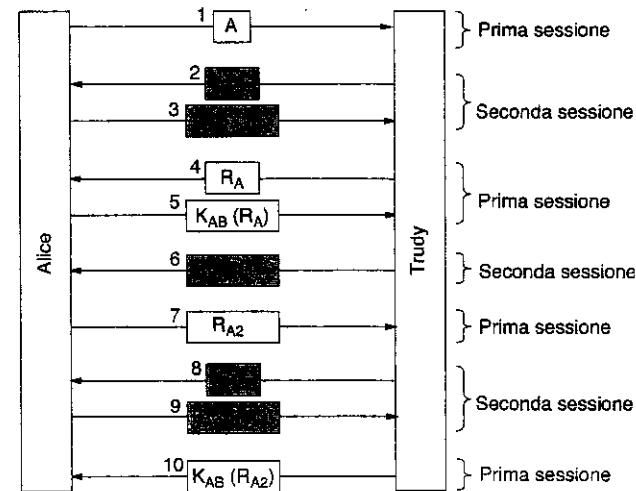


Figura 8.35. Attacco per riflessione al protocollo della figura 8.32.

Cosa fa Trudy adesso? Ritorna alla prima sessione, dove è arrivato il suo turno di inviare la richiesta, quindi invia  $R_A$  che ha ricevuto nel messaggio 3. Alice risponde gentilmente con il messaggio 5, il che fornisce a Trudy l'informazione che le serviva per mandare il messaggio 6 nella sessione 2. A questo punto Trudy è a campo vinto, perché ha risposto con successo alla richiesta di Alice nella sessione 2. Trudy può interrompere la sessione 1, inviare un qualunque vecchio numero per il resto della sessione 2, e avrà una sessione autenticata con Alice nella sessione 2.

Trudy vuole accanirsi nel suo attacco. Al posto di inviare un qualunque vecchio numero per completare la sessione 2, Trudy aspetta che Alice trasmetta il messaggio 7, la richiesta di Alice per la sessione 1. Ovviamente Trudy non sa come rispondere, quindi usa di nuovo l'attacco per riflessione: risponde con  $R_{A2}$  nel messaggio 8. Alice cifra  $R_{A2}$  nel mes-

saggio 9. Trudy quindi torna alla sessione 1 e trasmette ad Alice il messaggio 10 con il numero richiesto, semplicemente copiandolo da quello inviato da Alice nel messaggio 9. A questo punto Trudy ha due sessioni autenticate con Alice.

Il risultato di questo attacco è in qualche modo differente da quello contro il protocollo a tre messaggi della Figura 8.34. Questa volta Trudy ha due connessioni autenticate con Alice, mentre nell'esempio precedente aveva una connessione autenticata con Bob. Se avessimo applicato le regole generali per i protocolli di autenticazione discusse in precedenza, questo attacco sarebbe stato fermato. Una discussione dettagliata di questo tipo di attacchi e di come combatterli è data da (Bird et al., 1993). In quella trattazione viene anche mostrato come si possono sistematicamente costruire dei protocolli che sono corretti in modo dimostrabile. Il più semplice di tali protocolli è comunque un po' complicato, perciò adesso discuteremo una classe differente di protocolli più semplici ma ugualmente funzionanti.

Il nuovo protocollo di autenticazione è mostrato nella Figura 8.36. (Bird et al., 1993). Utilizza un HMAC del tipo che abbiamo visto studiando l'IPsec. Alice inizia inviando a Bob nel messaggio 1 un nonce  $R_A$ . Bob risponde scegliendo il suo nonce  $R_B$ , che invia ad Alice con un HMAC. L'HMAC è formato costruendo una struttura dati che consiste nel nonce di Alice, quello di Bob, le loro identità e la chiave segreta condivisa  $K_{AB}$ ; per ottenere l'HMAC si calcola l'hash di questa struttura, per esempio usando SHA-1. Quando riceve il messaggio 2, Alice conosce  $R_A$  (che aveva scelto lei),  $R_B$  (che le è arrivato come testo in chiaro), le due identità, e la chiave segreta  $K_{AB}$  (che conosceva fin dall'inizio). Alice può quindi procedere indipendentemente al calcolo dell'HMAC e confrontare il risultato con quello ricevuto nel messaggio 2. Se i due valori coincidono, Alice sa di parlare con Bob, perché Trudy non conosce  $K_{AB}$  e quindi non riesce a calcolare l'HMAC corretto. Alice risponde a Bob con un HMAC che contiene due nonce.

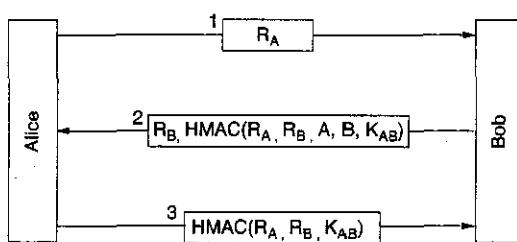


Figura 8.36. Autenticazione con gli HMAC.

Trudy riesce in qualche modo a forzare il protocollo? No, perché non riesce a indurre nessuna delle due parti a cifrare o calcolare l'hash di un valore a sua scelta, come invece accadeva nella Figura 8.34 e nella Figura 8.35. Entrambi gli HMAC includono dei valori scelti dalla destinazione, cioè qualcosa che Trudy non può controllare.

Gli HMAC non sono l'unico modo per sfruttare questa idea. Uno schema alternativo, spesso usato invece del calcolo dell'HMAC per una serie di oggetti, consiste nel cifrare gli oggetti in sequenza con la modalità cipher block chaining.

### 8.7.2 Come stabilire una chiave condivisa: lo scambio di chiave di Diffie-Hellman

Fino ad ora abbiamo supposto che Alice e Bob condividano una chiave segreta. Supponiamo che questo non sia vero (in quanto non esiste ancora una PKI universalmente accettata per firmare e distribuire i certificati). Come si può stabilire una chiave condivisa? Alice potrebbe chiamare Bob al telefono e dettargli la chiave, ma probabilmente la telefonata comincerebbe con Bob che chiede: come faccio a sapere che tu sei Alice e non Trudy? Potrebbero allora organizzare un incontro, dove ognuno porta il passaporto, la patente di guida e tre carte di credito dei circuiti maggiori. Essendo persone impegnate, può darsi che non riescano a fissare una data accettabile per mesi a venire. Fortunatamente e incredibilmente, esiste un metodo per stabilire una chiave segreta condivisa fra perfetti sconosciuti; questo metodo funziona senza segreti, anche se Trudy ascolta e registra attentamente ogni messaggio scambiato.

Il protocollo che permette a degli sconosciuti di scambiarsi una chiave segreta è chiamato **scambio di chiave di Diffie-Hellman** (Diffie e Hellman, 1976). Questo protocollo richiede come prima azione che Alice e Bob si mettano d'accordo su due numeri grandi,  $n$  e  $g$ , dove  $n$  è un numero primo,  $(n - 1)/2$  è pure primo e  $g$  soddisfa a certe condizioni particolari. Questi numeri possono essere pubblici, quindi uno dei due può scegliere la coppia di numeri e poi semplicemente la comunica all'altro. A questo punto Alice sceglie un numero grande (per esempio di 512 bit),  $x$ , e lo tiene segreto. Analogamente Bob sceglie un numero grande e segreto,  $y$ .

Alice inizia il protocollo di scambio della chiave inviando a Bob un messaggio che contiene  $(n, g, g^x \bmod n)$ , come mostrato nella Figura 8.37. Bob risponde inviando ad Alice un messaggio che contiene  $g^y \bmod n$ . Alice quindi prende il numero che gli ha inviato Bob e lo eleva alla potenza  $x$ , modulo  $n$ , cioè calcola  $(g^y \bmod n)^x \bmod n$ . Bob esegue un calcolo simile per ottenere  $(g^x \bmod n)^y \bmod n$ . Per le regole dell'aritmetica modulare, entrambe le espressioni valgono  $g^{xy} \bmod n$ . Il gioco è fatto: ora Alice e Bob hanno una chiave segreta condivisa:  $g^{xy} \bmod n$ .

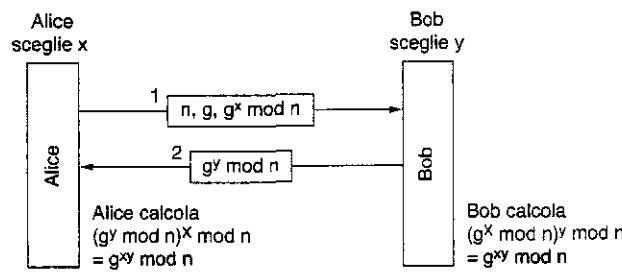


Figura 8.37. Lo scambio di chiavi Diffie-Hellman.

Trudy, ovviamente, ha visto entrambi i messaggi, quindi conosce  $g$  e  $n$  dal messaggio 1. Se riuscisse a calcolare  $x$  e  $y$ , potrebbe conoscere la chiave segreta. Il punto è che, dato solamente  $g^x \bmod n$ , non riesce a trovare  $x$ . Non sono noti algoritmi che riescano a calcolare in modo efficiente i logaritmi discreti con modulo un numero primo molto grande.

Per rendere l'esempio più concreto, useremo i valori (completamente non realistici) di  $n = 47$  e  $g = 3$ . Alice sceglie  $x = 8$  e Bob sceglie  $y = 10$ . Entrambi questi numeri sono tenuti segreti. Il messaggio di Alice per Bob è  $(47, 3, 28)$ , in quanto  $3^8 \text{ mod } 47 = 28$ . Il messaggio di Bob per Alice è  $(17)$ . Alice calcola  $17^8 \text{ mod } 47$ , che vale 4. Bob calcola  $28^{10} \text{ mod } 47$ , che vale 4. Alice e Bob hanno determinato in modo indipendente la chiave segreta, che è 4. Trudy deve risolvere l'equazione  $3^x \text{ mod } 47 = 28$ : per dei numeri piccoli come questi il problema può essere risolto con una ricerca esaustiva, ma non certo quando tutti i numeri in gioco sono lunghi centinaia di bit. Tutti gli algoritmi oggi conosciuti impiegano un tempo troppo lungo, anche usando dei computer a parallelismo massiccio.

Nonostante la sua eleganza, l'algoritmo di Diffie-Hellman ha un problema: quando Bob riceve la tripla  $(47, 3, 28)$ , come fa a sapere che viene da Alice e non da Trudy? Non c'è modo per Bob di saperlo. Sfortunatamente, Trudy può utilizzare questo fatto per ingannare Alice e Bob, come illustrato nella Figura 8.38. Mentre Alice e Bob scelgono rispettivamente  $x$  e  $y$ , Trudy sceglie il suo numero casuale  $z$ . Alice invia il messaggio 1 per Bob, Trudy lo intercetta e manda il messaggio 2 a Bob, usando i valori corretti per  $g$  e  $n$  (che comunque sono pubblici), ma con il valore  $z$  al posto di  $x$ . Manda anche indietro il messaggio 3 ad Alice. Successivamente Bob invia ad Alice il messaggio 4, che Trudy intercetta di nuovo e trattiene.

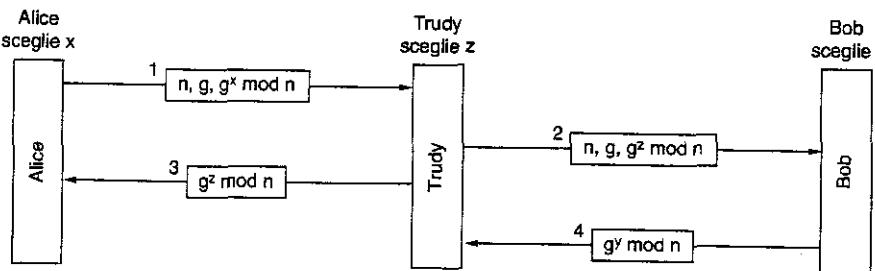


Figura 8.38. L'attacco bucket brigade o uomo nel mezzo.

A questo punto tutte le parti coinvolte eseguono i loro calcoli con l'aritmetica modulare. Alice calcola la chiave segreta  $g^{xz} \text{ mod } n$ . Questo calcolo viene fatto anche da Trudy (per i messaggi con Alice). Bob calcola  $g^{yz} \text{ mod } n$ , un calcolo che fa anche Trudy (lo userà per trasmettere con Bob). Alice pensa di parlare con Bob e quindi stabilisce una chiave di sessione con Trudy; lo stesso vale per Bob. Ogni messaggio che Alice invia sulla sessione cifrata è catturato da Trudy, memorizzato, modificato se necessario, ed eventualmente passato a Bob. Lo stesso vale per la direzione opposta. Trudy vede tutto e può modificare tutti i messaggi se lo vuole, mentre Bob e Alice hanno la falsa impressione di avere un canale sicuro di comunicazione. Questo tipo di attacco è noto come **attacco bucket brigade** (attacco del passaggio dei secchi), in quanto ricorda vagamente il vecchio metodo usato dai vigili del fuoco volontari, che si passavano l'un l'altro i secchi d'acqua dal camion cisterna fino all'incendio. Questo tipo di attacco viene chiamato anche **man-in-the-middle** (attacco dell'uomo nel mezzo).

### 8.7.3 Autenticazione usando un centro di distribuzione delle chiavi

Siamo quasi riusciti a far funzionare il metodo per stabilire un segreto condiviso con uno sconosciuto, ma ancora non ci siamo. Ripensandoci meglio, dopotutto forse questo algoritmo non serve a nulla: per parlare con  $n$  persone in questo modo servono  $n$  chiavi. Per le persone molto ricercate la gestione delle chiavi diventerebbe un vero problema, specialmente se ogni chiave va memorizzata in una smart card distinta.

Un approccio differente si ottiene introducendo un centro di distribuzione delle chiavi, o **KDC** (*Key Distribution Center*). In questo modello, ogni utente ha una singola chiave condivisa con il KDC, e la gestione dell'autenticazione e della chiave di sessione sono gestite dal KDC. Nella Figura 8.39 è mostrato il più semplice fra i protocolli noti per l'autenticazione KDC con due soggetti e un KDC fidato.

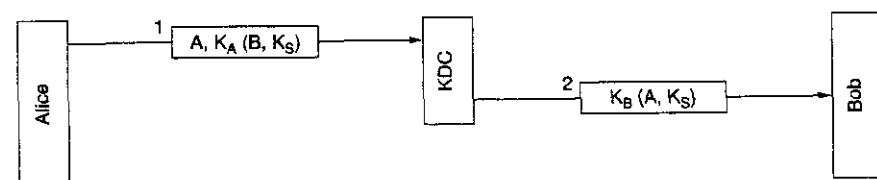


Figura 8.39. Primo tentativo per realizzare un protocollo di autenticazione usando un KDC.

L'idea dietro al protocollo è la seguente: Alice sceglie una chiave di sessione  $K_S$ , e dice al KDC che vuole comunicare con Bob usando  $K_S$ . Questo messaggio viene cifrato usando la chiave segreta che Alice condivide (solamente) con KDC,  $K_A$ . Il KDC decifra il messaggio, estrae l'identità di Bob e la chiave di sessione, poi costruisce un nuovo messaggio contenente l'identità di Alice e la chiave di sessione, infine spedisce il nuovo messaggio a Bob. Questa cifratura è fatta con  $K_B$ , la chiave segreta che Bob condivide con KDC: quando Bob decifrerà il messaggio, saprà che Alice desidera parlare con lui e saprà anche la chiave che Alice vuole usare.

In questo scenario l'autenticazione si ottiene gratuitamente. Il KDC sa che il messaggio 1 proviene necessariamente da Alice, infatti nessun altro avrebbe potuto cifrarlo con la chiave privata di Alice; analogamente Bob sa che il messaggio 2 proviene per forza dal KDC (di cui ha fiducia), in quanto nessun altro conosce la sua chiave segreta.

Sfortunatamente questo protocollo ha un difetto grave. Trudy ha bisogno di soldi, quindi s'inventa qualche servizio legittimo che può fornire ad Alice, presenta una buona offerta e ottiene il lavoro. Dopo aver finito il lavoro, Trudy chiede educatamente ad Alice di essere pagata con un bonifico bancario. Alice stabilisce perciò una chiave di sessione con il suo banchiere, Bob, a cui invia il messaggio di richiesta di un bonifico a favore di Trudy.

Nel frattempo, Trudy è tornata alle sue vecchie abitudini e intercetta i messaggi sulla rete. Copia sia il messaggio 2 della Figura 8.39, sia la successiva richiesta di bonifico.

Più tardi Trudy manda nuovamente entrambi i messaggi a Bob. Bob li riceve e pensa: Alice deve aver assunto di nuovo Trudy, chiaramente lavora bene. Bob procede di nuovo al trasferimento della stessa somma di denaro dal conto di Alice a quello di Trudy. Dopo aver

ricevuto la cinquantesima copia di messaggi uguali, Bob va a cercare Trudy per offrirle un bel prestito per espandere quello che è ormai un business di sicuro successo. Questo problema si chiama **attacco di ripetizione** (*replay attack*).

Esistono svariate soluzioni per bloccare gli attacchi di ripetizione. Il primo consiste nell'aggiungere a ciascun messaggio un contrassegno temporale (*timestamp*); i messaggi scaduti vengono immediatamente scartati dal destinatario. Il problema di questo approccio è che su una rete gli orologi non sono mai esattamente sincronizzati, quindi bisogna prevedere un intervallo di tempo in cui il timestamp rimane valido. Trudy può ripetere il messaggio durante questo intervallo e farla franca.

La seconda soluzione consiste nel mettere un nonce in ciascun messaggio. Ogni soggetto deve ricordarsi tutti i nonce precedenti e scartare ogni messaggio che contiene un nonce già usato. I nonce vanno ricordati per sempre, altrimenti Trudy può ripetere un messaggio vecchio di 5 anni. Inoltre, se qualche macchina va in crash e dimentica la lista dei nonce, può diventare vulnerabile ad attacchi di ripetizione. Timestamp e nonce si possono combinare per limitare l'intervallo di tempo in cui ricordare i nonce, ma chiaramente questo rende il protocollo molto più complicato.

Un approccio più sofisticato alla mutua autenticazione consiste nell'usare un protocollo challenge-response a più vie. Un noto esempio è il **protocollo di autenticazione di Needham-Schroeder** (Needham e Schroeder, 1978), di cui la Figura 8.40 mostra una variante.

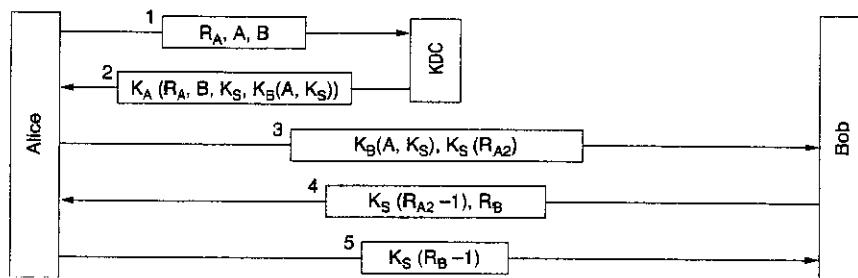


Figura 8.40. Il protocollo di autenticazione Needham-Schroeder.

Il protocollo comincia con Alice che dice al KDC che vuole parlare con Bob. Questo messaggio contiene un numero casuale grande,  $R_A$ , usato come nonce. Il KDC manda indietro il messaggio 2 che contiene il numero casuale di Alice, una chiave di sessione e un ticket che Alice può inviare a Bob. Il numero casuale  $R_A$  serve per rassicurare Alice che il messaggio 2 è fresco e non è una ripetizione. L'identità di Bob è inclusa nel messaggio, per prevenire un attacco in cui Trudy tenta di rimpiazzare l'identità di  $B$  con la propria identità all'interno del messaggio 1: se nel messaggio non fosse presente l'identità di Bob, il KDC eseguirebbe la cifratura del ticket (alla fine del messaggio 2) usando  $K_T$  al posto di  $K_B$ . Il ticket cifrato con  $K_B$  è incluso nel messaggio cifrato per evitare che Trudy possa facilmente sostituirlo durante la trasmissione ad Alice.

Successivamente, Alice invia a Bob un ticket e un nuovo numero casuale,  $R_{A2}$ , cifrato con

la chiave di sessione  $K_S$ . Nel messaggio 4, Bob risponde con  $K_S(R_{A2} - 1)$  per dimostrare ad Alice che sta parlando veramente con Bob. La trasmissione di  $K_S(R_{A2})$  non è accettabile, perché Trudy potrebbe rubare questa informazione dal messaggio 3.

Dopo aver ricevuto il messaggio 4, Alice si convince che sta veramente parlando con Bob e che non è stato fatto nessun attacco di ripetizione: infatti è stata Alice a generare  $R_{A2}$  solo alcuni millisecondi prima. Il messaggio 5 serve per convincere Bob dell'identità di Alice e dell'assenza di attacchi di ripetizione. Entrambi i soggetti della comunicazione hanno generato una richiesta di tipo challenge-response, e ciò elimina la possibilità che sia avvenuto un qualunque tipo di attacco di ripetizione.

Questo protocollo può sembrare molto solido, però ha una piccola debolezza. Infatti Trudy sarà in grado di stabilire una sessione con Bob se riesce a ottenere una vecchia chiave di sessione nel testo in chiaro. Per fare questo le basterà ripetere il messaggio 3 che corrisponde alla chiave compromessa, e Bob sarà convinto di essere in comunicazione con Alice (Denning e Sacco, 1981). Questa volta Trudy riesce a prosciugare il conto corrente di Alice senza dover eseguire neanche una volta un servizio legittimo.

Successivamente Needham e Schroeder hanno pubblicato un protocollo che corregge questo tipo di problemi (Needham e Schroeder, 1987). Nello stesso numero di quella rivista, Otway e Rees (1987) hanno pubblicato un altro articolo che risolve il problema in un modo più conciso. La Figura 8.41 mostra una versione leggermente modificata del protocollo Otway-Rees.

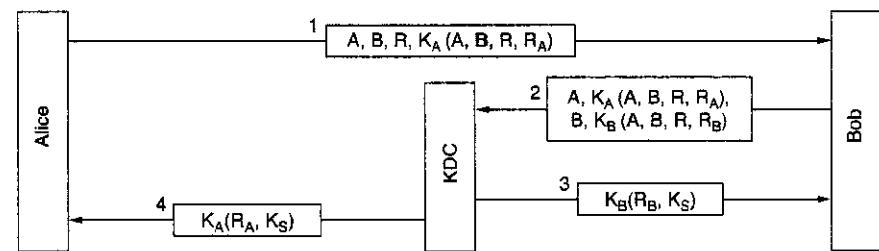


Figura 8.41. Il protocollo di autenticazione Otway-Rees (leggermente semplificato).

Nel protocollo di Otway-Rees, Alice inizia generando una coppia di numeri casuali:  $R$  (che verrà usata come identificatore comune) e  $R_A$  (che Alice invierà come richiesta a Bob). Quando Bob riceve questo messaggio, ne costruisce uno nuovo usando la parte cifrata del messaggio di Alice e una parte analoga costruita da lui stesso. Entrambe le parti, cifrate con  $K_A$  e  $K_B$ , identificano Alice e Bob, contengono l'identificatore comune e una richiesta (*challenge*).

Il KDC controlla che  $R$  in entrambe le parti abbia lo stesso valore. Potrebbe anche non essere vero, se Trudy ha manomesso  $R$  nel messaggio 1 oppure alterato parte del messaggio 2. Se le due  $R$  coincidono, il KDC considera valido il messaggio di richiesta di Bob, perciò genera una chiave di sessione e la cifra due volte: una per Alice e una per Bob. Ogni messaggio contiene il numero casuale del ricevitore, come prova che il messaggio è stato

generato dal KDC e non da Trudy. A questo punto Alice e Bob sono in possesso della stessa chiave di sessione e possono cominciare a comunicare. Al primo scambio di dati, ognuno dei due può vedere che l'altro è in possesso di un'identica copia di  $K_s$ , quindi l'autenticazione è completata.

#### 8.7.4 Autenticazione con Kerberos

Un protocollo di autenticazione usato in molti sistemi reali (incluso Windows 2000) è Kerberos, basato su una variante di Needham-Schroeder. Il suo nome deriva dalla mitologia greca, dove Cerbero (Kerberos in inglese) è il cane a più teste messo a guardia dell'ingresso dell'Ade (presumibilmente per tenere fuori le persone non desiderate). Kerberos fu progettato al M.I.T. per fornire un accesso sicuro alle risorse di rete agli utenti di workstation. La principale differenza rispetto a Needham-Schroeder sta nel fatto che Kerberos suppone che tutti gli orologi siano sincronizzati in modo abbastanza preciso. Il protocollo ha subito uno sviluppo con diverse iterazioni. La versione maggiormente utilizzata nell'industria è la V4, quindi sarà quella che discuteremo, ma avremo modo di fare anche qualche commento sul suo successore V5. Per ulteriori informazioni si rimanda a (Steiner et al., 1988).

Kerberos utilizza tre server in aggiunta ad Alice (la workstation client):

*Authentication Server (AS):* verifica gli utenti durante il login

*Ticket-Granting Server (TGS):* emette ticket che provano l'identità di chi li possiede

Il server Bob: è quello che compie il lavoro che Alice ha richiesto.

AS è simile a un KDC, nel senso che condivide una password segreta con ogni utente. Il TGS ha il compito di emettere dei ticket (biglietti) da usare per convincere i server veri e propri del fatto che il loro possessore è realmente chi dichiara di essere.

Alice inizia una sessione da una qualunque workstation pubblica, inserendo il suo nome. La workstation invia il nome di Alice in chiaro all'AS, come mostrato nella Figura 8.42. La risposta sarà formata da una chiave di sessione e da un ticket per il TGS,  $K_{TGS}(A, K_s)$ . Questi dati vengono impacchettati e cifrati usando la chiave segreta di Alice, così solamente Alice sarà in grado di decifrarli. La workstation chiederà la password di Alice solo dopo aver ricevuto il messaggio 2. La password viene quindi usata per generare  $K_A$ , che a sua volta serve per decifrare il messaggio 2 e ottenere la chiave di sessione e il ticket TGS al suo interno. A questo punto la workstation sovrascrive la password di Alice, assicurandosi così che sia rimasta all'interno della workstation stessa solamente per pochi millisecondi. Se Trudy tenta di fare il login con il nome di Alice, la password che inserirà sarà sbagliata e la workstation potrà accorgersene in quanto la parte standard del messaggio 2 non sarà corretta.

Dopo il login, Alice può chiedere alla workstation di contattare il file server Bob. La workstation invia il messaggio 3 al TGS chiedendogli un ticket da usare con Bob. L'elemento chiave di questa richiesta è  $K_{TGS}(A, K_s)$ , che è cifrato con la chiave segreta del TGS ed è usato come prova che il mittente è Alice. Il TGS risponde creando una chiave di sessione  $K_{AB}$ , che Alice potrà usare con Bob. Ne vengono spedite indietro due versioni: la prima è cifrata solamente con  $K_s$ , così che Alice la possa leggere, mentre la seconda è cifrata con la chiave di Bob  $K_B$ , così che Bob riesca a leggerla.

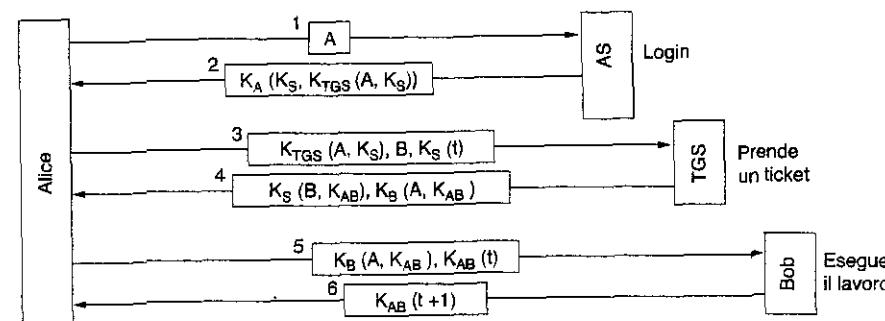


Figura 8.42. Funzionamento di Kerberos V4.

Trudy può copiare il messaggio 3 e provare a usarlo di nuovo, ma sarà bloccata dal timestamp cifrato  $t$ , che è inviato insieme al messaggio. Trudy non può rimpiazzare il timestamp con uno più recente, in quanto non conosce  $K_s$ , la chiave di sessione che Alice usa per parlare con il TGS. Se Trudy rimanda il messaggio 3 molto velocemente, l'unica cosa che otterrebbe sarebbe un'altra copia del messaggio 4, che non è riuscita a decifrare la prima volta e che quindi non sarà neppure in grado di decifrare questa volta.

Adesso Alice può inviare  $K_{AB}$  a Bob e stabilire una sessione con lui; anche questo scambio è accompagnato da un timestamp. Per Alice la risposta è una prova del fatto che sta effettivamente parlando con Bob e non con Trudy.

Dopo una serie di scambi, Alice riesce a parlare con Bob usando la copertura fornita da  $K_{AB}$ . Se successivamente Alice decide di parlare con un altro server, Carol, deve semplicemente inviare il messaggio 3 al TGS indicando  $C$  al posto  $B$ . Il TGS risponderà subito con un ticket cifrato con  $K_C$ , che Alice può inviare a Carol e che Carol accetterà come prova della provenienza da Alice.

Lo scopo di tutto questo lavoro è rendere Alice in grado di accedere ai server su tutta la rete in modo sicuro senza che la sua password debba mai essere trasmessa sulla rete. Infatti la password è rimasta sulla sua workstation, e solo per pochi millisecondi. Notiamo comunque che ogni server si prende carico delle proprie autorizzazioni. Quando Alice presenta a Bob il suo ticket, questo serve solo a dimostrare a Bob l'identità di Alice. Stabilire che cosa Alice sia autorizzata a fare è un compito che viene lasciato a Bob.

I progettisti di Kerberos non si aspettavano certo che tutto il mondo potesse fidarsi di un unico server di autenticazione, per questo motivo hanno previsto la possibilità di avere più **realm**, ognuno con il proprio AS e TGS. Per ottenere un ticket per un realm remoto, Alice deve chiedere al suo TGS un ticket accettato dal TGS nel realm remoto. Se il TGS remoto si è registrato con il TGS locale (allo stesso modo in cui lo fanno i server locali), il TGS locale darà ad Alice un ticket valido per il TGS remoto. Alice può quindi operare nel realm remoto, per esempio ottenendo ticket per i server di quel realm. Notare che per far sì che due soggetti situati in realm diversi possano operare uno con l'altro, è necessario che abbiano fiducia ognuno nel TGS dell'altro.

Kerberos V5 è una versione migliorata di V4, con più ridondanza. Inoltre usa OSI ASN.1 (*Abstract Syntax Notation 1*, notazione di sintassi astratta 1) per descrivere i tipi di dati e ha alcuni piccoli cambiamenti nei protocolli. Prevede una vita più lunga per i ticket, permette il rinnovo dei ticket e può fornire ticket postdatati. In aggiunta, almeno in teoria, V5 non è dipendente dal DES come V4, e supporta realm multipli con il meccanismo di delega dei ticket a più ticket server.

### 8.7.5 Autenticazione con la crittografia a chiave pubblica

La mutua autenticazione si può eseguire anche usando la crittografia a chiave pubblica. Per cominciare, Alice ha bisogno della chiave pubblica di Bob. Se esiste una PKI con un directory server che fornisce i certificati per le chiavi pubbliche, Alice gli può chiedere il certificato di Bob, come mostrato nella Figura 8.43 per il messaggio 1. La risposta, il messaggio 2, è un certificato X.509 che contiene la chiave pubblica di Bob. Alice verifica la correttezza della firma, quindi manda a Bob un messaggio che contiene la sua identità e un nonce.

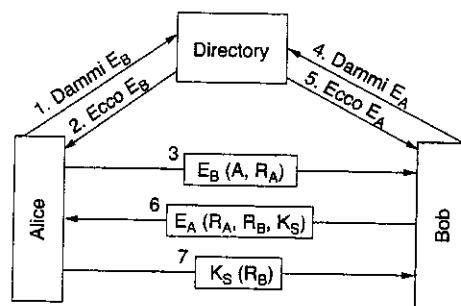


Figura 8.43. Mutua autenticazione con crittografia a chiave pubblica.

Quando Bob riceve questo messaggio non sa ancora se proviene da Alice o da Trudy, ma prosegue chiedendo al directory server la chiave pubblica di Alice (messaggio 4) che riceve rapidamente (messaggio 5). Quindi Bob manda ad Alice il messaggio 6 che contiene l' $R_A$  di Alice, il suo nonce  $R_B$  e una proposta per la chiave di sessione,  $K_S$ .

Alice riceve il messaggio 6 e lo decifra usando la sua chiave privata. Vede che contiene  $R_A$ , il che la rende contenta. Il messaggio deve quindi venire da Bob, visto che Trudy non ha modo di determinare  $R_A$ . Inoltre deve essere un messaggio fresco e non una ripetizione, in quanto Alice ha appena inviato  $R_A$  a Bob. Alice conferma di essere d'accordo sulla chiave di sessione con il messaggio 7. Quando Bob riceve il messaggio e vede che contiene  $R_B$  cifrato con la chiave di sessione che ha appena generato, sa che Alice ha ricevuto correttamente il messaggio 6 e ha verificato  $R_A$ .

Cosa può fare Trudy per cercare di forzare questo protocollo? Può fabbricare il messaggio 3 e cercare d'ingannare Bob facendogli spedire un messaggio ad Alice. Alice, però, vedrà

un  $R_A$  che non ha spedito e quindi non procederà ulteriormente. Trudy non può falsificare il messaggio 7 di ritorno a Bob, perché non conosce né  $R_B$  né  $K_S$ , e non può saperli senza avere la chiave privata di Alice. Quindi Trudy non riesce a fare niente.

## 8.8 Sicurezza dell'e-mail

Un messaggio e-mail scambiato fra due siti molto distanti fra loro transita tipicamente attraverso dozzine di macchine, prima di giungere a destinazione. Ognuna di queste macchine può leggere il messaggio e registrarne per usi futuri. In pratica la privacy non esiste, al contrario di quanto molti pensano. Tuttavia molti vorrebbero poter inviare delle e-mail leggibili solo dal destinatario legittimo, e da nessun altro: neppure il loro capo o il governo. Questo desiderio ha stimolato persone e gruppi ad applicare alla posta elettronica i principi crittografici che abbiamo studiato in precedenza per ottenere e-mail sicure. Nei seguenti paragrafi, studieremo un sistema di e-mail sicura molto usato nella pratica: PGP. Faremo anche menzione di altri due sistemi: PEM e S/MIME. Per ulteriori informazioni sulle e-mail sicure si consiglia di fare riferimento a (Kaufman et al. 2002, Schneier, 1995).

### 8.8.1 PGP (*Pretty Good Privacy*)

Il nostro primo esempio, PGP (*Pretty Good Privacy*, privacy decisamente buona) è essenzialmente il frutto del lavoro di una sola persona, Phil Zimmermann (Zimmermann, 1995a, 1995b). Zimmermann è uno strenuo difensore della privacy, e il suo motto è: se la privacy è messa fuori legge, allora solo i fuori legge avranno la privacy. PGP è un pacchetto completo per la sicurezza della posta elettronica, rilasciato nel 1991, che fornisce privacy, autenticazione, firme digitali e compressione; il tutto in una forma semplice da usare. L'intero pacchetto, con i codici sorgente, è distribuito gratuitamente su Internet. Oggi PGP è largamente usato per le sue doti di qualità, prezzo (nullo) e immediata disponibilità per le piattaforme Unix, Linux, Windows e Mac OS.

PGP usa un cifrario a blocchi per i dati chiamato IDEA (*International Data Encryption Algorithm*, algoritmo internazionale per la cifratura dei dati), con una chiave a 128 bit. IDEA è stato sviluppato in Svizzera, quando DES era considerato compromesso e AES non era ancora stato inventato. Concettualmente, IDEA è simile a DES e AES: lavora mescolando i bit in una serie di iterazioni, però i dettagli delle funzioni di mescolamento sono diversi da quelli di DES e AES. La gestione delle chiavi si appoggia a RSA mentre l'integrità dei dati è fornita da MD5, che sono argomenti già discussi.

PGP è stato circondato da controversie sin dalla sua nascita (Levy, 1993). Zimmermann non aveva fatto niente per impedire che altre persone pubblicassero PGP su Internet, dove tutto il mondo poteva averne accesso, perciò il governo degli Stati Uniti accusò Zimmermann di aver violato le leggi che proibivano l'esportazione di armi. L'inchiesta su Zimmermann da parte del governo USA durò 5 anni, ma alla fine fu chiusa, probabilmente per due motivi. Il primo è che Zimmermann non aveva messo direttamente PGP su Internet, quindi il suo avvocato dichiarò che Zimmermann non aveva mai esportato nien-

te (rimane anche da discutere il fatto che creare un sito Web costituisca un'exportazione). Il secondo è che il governo degli USA si rese conto che vincere il processo voleva dire convincere una giuria che un sito Web che fornisce il download di un programma per la privacy è vietato dalla legge che proibisce il traffico d'armi e l'exportazione di materiale di guerra come, per esempio, carri armati, sottomarini, aerei militari e armi nucleari. Infine, diversi anni di pubblicità negativa hanno probabilmente avuto una certa influenza. Per inciso, le regole sull'exportazione sono quanto meno bizzarre, per non dire altro. Mettere del codice su un sito Web era considerato illegale dal governo, che quindi perseguì Zimmermann per circa 5 anni. D'altra parte, quando qualcuno pubblicò un libro con il sorgente C completo di PGP (con un font grande e il checksum su ogni pagina, in modo da facilitare la scansione) e poi ha esportato il libro, la cosa era perfettamente accettabile per il governo perché i libri non sono considerati armi. La spada è più potente della penna, per lo meno per lo Zio Sam.

Un altro problema che PGP dovette affrontare, era legato alla violazione dei brevetti. L'azienda che deteneva il brevetto per RSA, la RSA Security Inc., dichiarò che l'uso dell'algoritmo RSA in PGP violava il suo brevetto. Questo fu risolto a partire dalla versione 2.6 di PGP. Nei primi tempi ci furono anche dei problemi con un altro algoritmo crittografico brevettato, IDEA.

Gruppi e singoli individui si sono cimentati in modifiche di PGP, vista la sua natura open source, e ciò ha prodotto una quantità di modifiche e versioni. Alcune di queste modifiche erano mirate ad aggirare la legge sul traffico di armi, altre a evitare l'uso di algoritmi brevettati, altre ancora tentavano di far diventare PGP un prodotto commerciale non più open source. Nel frattempo le leggi sulle armi sono state leggermente liberalizzate (altrimenti i prodotti che utilizzano AES non sarebbero esportabili dagli USA) e il brevetto di RSA è scaduto nel settembre 2000. Comunque, come conseguenza delle prime polemiche, adesso sono in circolazione (con nomi diversi) più versioni di PGP incompatibili tra loro. La discussione che segue è focalizzata sul PGP classico, nella sua versione più semplice e antica. Un'altra versione molto diffusa, Open PGP, è descritta in RFC 2440; un'altra ancora è *GNU Privacy Guard* (guardiano della privacy GNU).

PGP utilizza di proposito algoritmi crittografici esistenti, al posto di inventarne dei nuovi. Gli algoritmi utilizzati, infatti, hanno superato esaustive ricerche da parte di esperti del campo e non sono stati influenzati da nessuna agenzia governativa che poteva avere interesse a indebolirli. Per quelle persone che tendono a non fidarsi del governo, questa proprietà è considerata molto positivamente.

PGP supporta la compressione del testo, la segretezza, le firme digitali e contiene funzioni evolute per la gestione delle chiavi. Stranamente, però, PGP non ha funzioni per la gestione delle e-mail: si tratta più che altro di un preprocessore, che prende in input un testo in chiaro e produce in output un testo cifrato in base64. Naturalmente questo output può essere inviato per e-mail. Come ultimo passaggio, alcune implementazioni di PGP prevedono la chiamata a un agente che invia il messaggio.

Per vedere come funziona PGP, consideriamo l'esempio della Figura 8.44. Alice vuole inviare a Bob in modo sicuro un messaggio in chiaro firmato,  $P$ . Alice e Bob hanno entram-

bi una chiave RSA privata ( $D_X$ ) e una pubblica ( $E_X$ ). Ipotizziamo che ognuno dei due conosca la chiave pubblica dell'altro (vedremo fra poco la gestione delle chiavi con PGP).

$K_M$  : Chiave di messaggio monouso per IDEA

$\otimes$  : Concatenazione

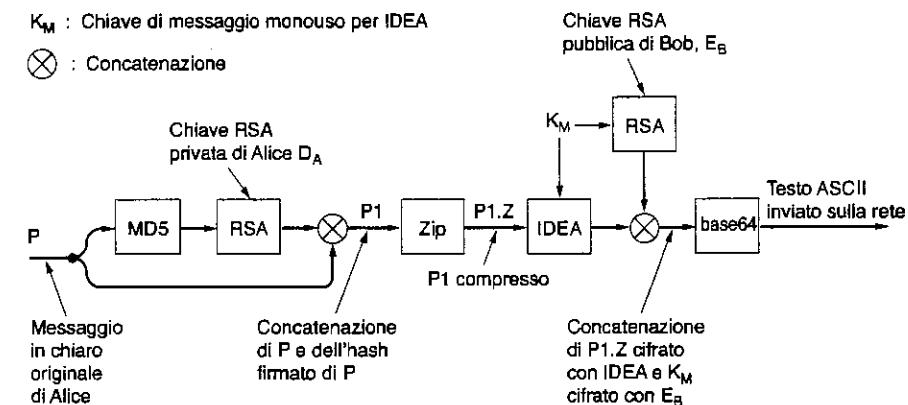


Figura 8.44. Invio di un messaggio con PGP.

Alice inizia lanciando il programma PGP sul suo computer. PGP calcola innanzitutto l'hash del messaggio  $P$  usando MD5, e poi cifra il risultato usando la sua chiave RSA privata  $D_A$ . Quando Bob riceve il messaggio, può decifrare l'hash usando la chiave pubblica di Alice e quindi verificarne la correttezza. La robustezza di MD5 garantisce che una terza persona (per esempio *Trudy*), capace di intercettare l'hash e decifrarlo con la chiave pubblica di Alice, non sarà in grado di produrre un messaggio con lo stesso hash MD5 (è un problema computazionalmente troppo difficile).

L'hash cifrato e il messaggio originale sono concatenati in un singolo messaggio  $P1$ , e compressi usando il programma ZIP, che usa l'algoritmo di Ziv-Lempel (Ziv e Lempel, 1977). Chiamiamo l'output di questo passo  $P1.Z$ .

Successivamente PGP chiede ad Alice un numero casuale in input. Il contenuto del messaggio e la velocità di battitura sono usati per generare una chiave di messaggio IDEA a 128 bit,  $K_M$  (viene chiamata chiave di sessione nella letteratura PGP, ma in realtà il nome non è appropriato, in quanto non esiste una sessione).  $K_M$  serve a cifrare  $P1.Z$  con IDEA in modalità cipher feedback; inoltre  $K_M$  è cifrata con la chiave pubblica di Bob,  $E_B$ . Ora queste due componenti vengono concatenate e convertite in base64, come discusso nel paragrafo sul MIME del Capitolo 7. Il messaggio risultante contiene solo lettere, cifre e i simboli + / =, quindi si può inserire in un body conforme a RFC 822 per inviarlo a destinazione senza temere alterazioni durante il transito.

Quando Bob riceve il messaggio, esegue a rovescio la codifica base64 e decifra la chiave IDEA usando la sua chiave privata; con essa decifra il messaggio e ottiene  $P1.Z$ . Quindi passa alla decompressione, separa il testo in chiaro dall'hash cifrato e decifra l'hash usando la chiave pubblica di Alice. Adesso Bob controlla se l'hash del testo in chiaro coincide

con il valore del suo calcolo MD5: in caso affermativo, sa con certezza che  $P$  è il messaggio originale inviato da Alice.

È utile notare che RSA si usa solamente in due momenti: per cifrare l'hash MD5 a 128 bit e per cifrare la chiave IDEA a 128 bit. Anche se RSA è lento, deve cifrare solamente 256 bit, quindi una quantità limitata di dati. D'altra parte, tutti i 256 bit di dati cifrati con RSA sono altamente casuali, quindi Trudy dovrebbe fare una gran quantità di lavoro già per stabilire se ha indovinato correttamente una chiave. La cifratura della maggior parte dei dati è fatta usando IDEA, che è diversi ordini di grandezza più veloce di RSA. Quindi PGP fornisce sicurezza, compressione e firma digitale, e lo fa in un modo molto più efficiente dello schema illustrato nella Figura 8.19.

PGP supporta quattro diverse lunghezze di chiavi RSA, così l'utente può scegliere quella che ritiene più appropriata:

1. *Casual* (384 bit): può essere forzata facilmente, al giorno d'oggi
2. *Commercial* (512 bit): può essere forzata dalle ben note organizzazioni governative
3. *Military* (1.024 bit): non può essere forzata da nessuno sulla terra
4. *Alien* (2.048 bit): non può essere forzata da nessuno neanche in altri pianeti.

Visto che RSA è usato solo per due piccoli calcoli, tutti dovrebbero usare sempre il livello "alien".

Il formato di un messaggio PGP classico è mostrato nella Figura 8.45, ma si usano numerosi altri formati. Il messaggio PGP ha tre parti, che contengono la chiave IDEA, la firma e il messaggio vero e proprio. La parte con la chiave contiene anche un identificatore della chiave, visto che gli utenti possono usare più chiavi pubbliche.

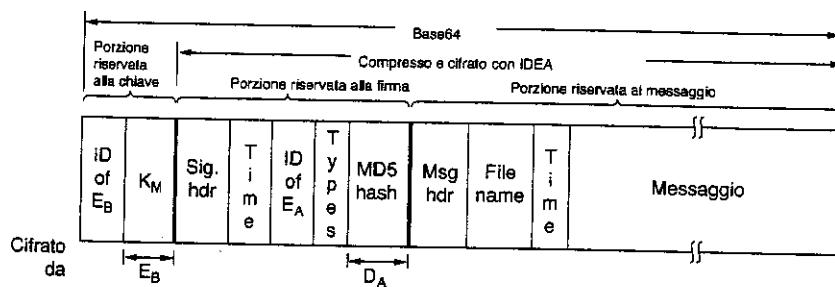


Figura 8.45. Messaggio PGP.

La parte del messaggio PGP che contiene la firma è composta da un'intestazione (che in questo contesto non ci interessa), da un timestamp, dall'identificatore della chiave pubblica usata dal mittente (che serve per decifrare la firma hash), da alcune informazioni che identificano gli algoritmi usati (per consentire l'uso di MD6 e RSA2, quando saranno inventati), e infine dall'hash cifrato.

La parte con il messaggio è formata da intestazione, nome di default per il file (nel caso che il destinatario voglia scriverlo su disco), timestamp relativo alla creazione del messaggio, e infine dal messaggio vero e proprio.

La gestione delle chiavi ha ricevuto una gran quantità di attenzioni durante lo sviluppo di PGP, poiché rappresenta il tallone di Achille di tutti i sistemi di sicurezza. Ogni utente gestisce localmente due strutture dati: un insieme di chiavi private e un insieme di chiavi pubbliche. L'**insieme di chiavi private** (portachiavi privato) contiene una o più coppie personali di chiavi private e pubbliche. Per ogni utente sono supportate coppie multiple di chiavi, per consentire agli utenti di cambiare le loro chiavi pubbliche con cadenza periodica o quando pensano che una chiave sia compromessa, senza che questo annulli la validità dei messaggi in preparazione o in transito. A ogni coppia di chiavi è associato un identificatore, così il mittente può comunicare al destinatario quale chiave pubblica ha usato per cifrare il messaggio. Gli identificatori di messaggio sono formati dai 64 bit di ordine più basso della chiave pubblica, e gli utenti hanno la responsabilità di evitare possibili conflitti fra gli identificatori delle loro chiavi pubbliche. Le chiavi private sono memorizzate su disco in forma cifrata utilizzando una password (di lunghezza arbitraria), per proteggerle contro eventuali furti.

Per capire il problema più grave della gestione delle chiavi, consideriamo l'esempio in cui le chiavi pubbliche sono gestite da una bulletin board. Trudy, per leggere le e-mail segrete di Bob, potrebbe attaccare la bulletin board e rimpiazzare la chiave pubblica di Bob con una di sua scelta. Quando più tardi Alice legge la chiave che crede appartenere a Bob, Trudy può far scattare un attacco di tipo bucket brigade con Bob.

Per prevenire questo tipo di attacchi, o comunque minimizzarne le conseguenze, Alice deve sapere quanto si può fidare di quella cosa chiamata "chiave di Bob" che si ritrova nel suo insieme di chiavi. Per esempio la fiducia sarebbe al massimo valore nel caso in cui Bob avesse personalmente passato ad Alice un floppy contenente la sua chiave. È questo approccio di gestione delle chiavi decentralizzato e controllato dagli utenti che rende PGP essenzialmente diverso dagli schemi di PKI centralizzati.

In ogni modo, in alcune occasioni le chiavi si ottengono con una richiesta a un server fidato. Dopo la standardizzazione di X.509, PGP supporta anche questo tipo di certificati, oltre al meccanismo tradizionale dell'insieme delle chiavi. Tutte le versioni correnti di PGP supportano X.509.

### 8.8.2 PEM (*Privacy Enhanced Mail*)

A differenza di PGP, che fu il frutto di una persona sola, il nostro secondo esempio è uno standard ufficiale di Internet descritto in quattro RFC, da 1421 a 1424. Si chiama **PEM** (*Privacy Enhanced Mail*, posta con privacy maggiorata), e venne sviluppato nei tardi anni 1980. PEM offre circa gli stessi servizi di PGP: privacy e autenticazione per i sistemi di e-mail basati su RFC 822. Le differenze fra PEM e PGP si trovano nell'approccio e nella tecnologia utilizzati.

I messaggi inviati usando PEM vengono dapprima convertiti in una forma canonica, in modo da seguire le stesse convenzioni sul white space (per esempio tabulatori e spazi in

fondo alla frase). Successivamente si calcola un hash usando MD2 o MD5; la concatenazione dell'hash con il messaggio viene cifrata con DES. Visto che è risaputo che le chiavi a 56 bit sono deboli, questa scelta è certamente sospetta. Il messaggio cifrato è infine codificato in base64 e trasmesso al destinatario.

Come nel caso di PGP, la chiave con cui è cifrato il messaggio non viene mai riutilizzata ed è trasmessa insieme al messaggio stesso; può essere protetta con RSA o con il triplo DES in modalità EDE.

La gestione delle chiavi è più strutturata che nel caso di PGP. Le chiavi sono garantite da certificati X.509 emessi dalle CA, e gestiti con una gerarchia rigida a partire da una singola root. Il vantaggio di questo schema è che la revoca dei certificati può essere gestita facendo sì che la root emetta periodicamente delle CRL.

L'unico problema di PEM è che nessuno l'ha mai utilizzato, e quindi giace da tempo nel dimenticatoio. Il problema era essenzialmente politico: chi dovrebbe gestire la root e a quali condizioni? Non c'era certamente mancanza di candidati, ma la gente non voleva affidare la sicurezza di tutto il sistema a una sola entità o azienda. Il candidato più serio, RSA Security Inc., voleva far pagare i certificati emessi. Ad alcune organizzazioni l'idea non piaceva; in particolare il governo USA può usare gratuitamente tutti i brevetti nazionali, mentre le aziende al di fuori degli USA si erano abituate a usare l'algoritmo RSA gratis (RSA si era scodata di brevettare l'algoritmo fuori dagli USA). Quindi nessuno di questi due gruppi era entusiasta all'idea di dover pagare RSA Security Inc. per qualcosa che avevano sempre fatto senza spendere. Alla fine, non fu trovata nessuna root e PEM collassò.

### 8.8.3 S/MIME

Il successivo approccio di IETF alla sicurezza delle e-mail, chiamato **S/MIME** (*Secure/MIME*) è descritto in RFC da 2632 a 2643. Analogamente a PEM, S/MIME fornisce autenticazione, integrità dei dati, segretezza e non ripudio; inoltre è un sistema flessibile che supporta diversi algoritmi crittografici. Visto il suo nome, non dovrebbe sorprendere la notizia che S/MIME si integra bene con MIME, il che gli permette di proteggere tutti i tipi di messaggi; sono state introdotte molte intestazioni MIME per gestire, per esempio, le firme digitali.

IETF aveva certamente imparato una lezione dall'esperienza di PEM. S/MIME non ha una struttura rigida di certificati che si diparte da una singola root, al contrario gli utenti possono avere molti punti di riferimento. Per essere considerato valido un certificato deve semplicemente avere un percorso che lo fa risalire a un punto di riferimento di cui l'utente si fida. S/MIME usa gli algoritmi e i protocolli standard che abbiamo discusso finora, perciò non esamineremo altri dettagli, per i quali rimandiamo agli RFC citati.

## 8.9 Sicurezza del Web

Abbiamo appena studiato due aree importanti in cui la sicurezza è necessaria: le comunicazioni e l'e-mail. Possiamo pensare a quelle due aree come al primo e all'antipasto. È arrivato il momento del piatto forte: la sicurezza del Web. Il Web è il posto dove ormai le Trudy si radunano sempre più spesso per fare il loro "sporco" lavoro. Nei paragrafi seguenti vedremo alcuni problemi legati alla sicurezza del Web.

La sicurezza del Web può essere divisa in tre filoni principali:

1. come si possono gestire in modo sicuro i nomi degli oggetti e delle risorse?
2. come si può stabilire una connessione autenticata?
3. che cosa succede quando un sito Web invia a un client un programma eseguibile?

Prima di affrontare questi problemi, vedremo alcune delle possibili minacce alla sicurezza.

### 8.9.1 Minacce alla sicurezza

I giornali riportano con cadenza quasi settimanale i problemi legati alla sicurezza del Web; la situazione è decisamente grave. Vediamo alcuni esempi di fatti già accaduti. Per iniziare, la home page di diverse organizzazioni è stata attaccata e rimpiazzata con una scelta dai cracker. Per inciso, la stampa popolare chiama "hacker" le persone che penetrano nei sistemi informatici senza autorizzazione. Molti programmatore, invece, riservano questo termine per indicare i colleghi veramente in gamba. Preferiamo quindi usare il termine "cracker". Fra i siti che hanno subito l'attacco della home page citiamo: Yahoo, U.S. Army, CIA, NASA e il New York Times. In molti casi, i cracker hanno solamente aggiunto qualche scritta divertente e il danno è stato poi riparato con qualche ora di lavoro.

Ci sono stati anche casi più seri. Diversi siti sono stati bloccati da attacchi di tipo denial of service, nei quali il cracker sommerge di traffico il sito in modo che non riesca più a rispondere alle richieste legittime. Spesso questi attacchi vengono lanciati da una gran quantità di macchine che il cracker ha già attaccato (attacchi DDos). Ormai questo tipo di attacchi è così comune che non fa più notizia, però può costare all'attaccato diverse migliaia di dollari di perdite causate dall'interruzione del servizio.

Nel 1999, un cracker svedese entrò nel sito Hotmail di Microsoft e creò un sito mirror che permetteva a chiunque di digitare il nome di un utente Hotmail e leggere tutta la posta corrente e archiviata di quell'utente.

In un altro caso, un cracker russo di 19 anni chiamato Maxim, entrò in un sito di e-commerce e rubò 300.000 numeri di carte di credito. A questo punto contattò i gestori del sito minacciandoli di pubblicare su Internet tutti i numeri di carte di credito se non gli avesse pagato 100.000 dollari. Il pagamento non fu fatto e il cracker pubblicò i numeri delle carte di credito, infliggendo un danno a migliaia di vittime innocenti.

Una storia differente è quella di uno studente californiano di 23 anni che inviò per e-mail un comunicato a un'agenzia di stampa, dove dichiarava falsamente che la Emulex Corporation stava per pubblicare delle forti perdite trimestrali e che il C.E.O. avrebbe rassegnato immediatamente le dimissioni. Nel giro di poche ore le azioni della compagnia scesero del 60%, causando una perdita per gli azionisti di oltre 2 miliardi di dollari. Lo studente guadagnò un quarto di milione di dollari contrattando le azioni appena prima dell'annuncio. Anche se questo non è un canonico attacco a un sito Web, è chiaro che mettere degli annunci di questo tipo sulla home page di una qualunque grande corporation sortirebbe degli effetti simili. Potremmo (sfortunatamente) andare avanti con queste notizie per molte pagine. A questo punto vogliamo però discutere alcune delle questioni tecniche legate alla sicurezza del Web. Per maggiori informazioni sui problemi di sicurezza di qualunque tipo si rimanda a (Anderson, 2001; Garfinkel e Spafford, 2002; Schneier, 2000). Una ricerca su Internet può essere utile per trovare un vasto numero di casi specifici.

### 8.9.2 Sicurezza del naming

Cominciamo con qualcosa di molto semplice: Alice vuole visitare il sito Web di Bob. Per fare questo digita l'URL di Bob nel browser e, dopo alcuni secondi, appare una pagina Web. È veramente la pagina di Bob? Forse sì e forse no. Trudy potrebbe essere in azione con qualcuno dei suoi vecchi trucchi. Per esempio, Trudy potrebbe intercettare ed esaminare tutti i pacchetti in uscita da Alice. Quando intercetta una richiesta http *GET* indirizzata al sito di Bob, Trudy potrebbe andare a prendere la pagina di Bob, modificarla a piacimento e poi inviare ad Alice la versione alterata. Alice non si accorgerebbe di niente. Peggio ancora, Trudy potrebbe alterare i prezzi del sito di e-commerce di Bob in modo da renderli molto convenienti, quindi Alice viene ingannata e indotta a inviare il suo numero di carta di credito per comprare merce da Bob. Uno svantaggio di questo classico attacco "uomo nel mezzo" è che Trudy deve poter intercettare tutto il traffico in uscita da Alice, ma anche modificare il traffico in ingresso. In pratica, Trudy deve intercettare la linea telefonica di Alice o quella di Bob, visto che queste operazioni sono decisamente difficili su un backbone in fibra. Intercettare le linee telefoniche è possibile, ma richiede diverso lavoro e si dà il caso che Trudy, oltre che intelligente, è anche pigra. Inoltre esistono dei metodi più semplici per ingannare Alice.

#### DNS spoofing

Per esempio, supponiamo che Trudy sia in grado di entrare nel sistema del DNS, magari anche solo nella cache del DNS dell'ISP di Alice, dove rimpiazza l'indirizzo IP di Bob (diciamo, 36.1.2.3) con il proprio indirizzo IP (diciamo, 42.9.9.9). In questo modo si realizza l'attacco illustrato nella Figura 8.46(a). La sequenza dei passi è la seguente: (1) Alice chiede al DNS l'indirizzo IP di Bob, (2) lo ottiene, (3) chiede a Bob la sua home page, (4) la ottiene. Dopo che Trudy ha modificato il record di Bob nel DNS in modo da contenere l'indirizzo di Trudy, otteniamo la situazione della Figura 8.46(b). In questo caso, quando Alice cerca l'indirizzo IP di Bob, ottiene invece quello di Trudy, quindi tutto il traffico che dovrebbe andare a Bob va verso Trudy, che può far partire un attacco "uomo nel mezzo" senza doversi scomodare a intercettare la linea telefonica. Le è bastato penetrare nel server DNS e cambiare un record, un compito molto più semplice.

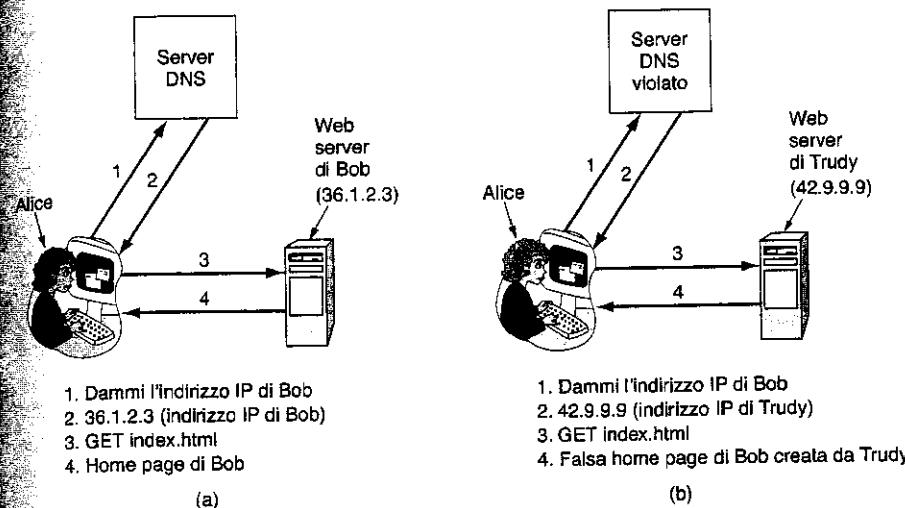


Figura 8.46. (a) Situazione normale. (b) Un attacco basato sull'aggressione del DNS e modifica del record di Bob.

Come fa Trudy a ingannare il DNS? In realtà è abbastanza semplice. Riassumendo brevemente, Trudy può ingannare il DNS dell'ISP di Alice inviandogli una query per cercare l'indirizzo di Bob. Sfortunatamente, visto che DNS usa UDP, il server DNS non ha modo di sapere chi fornisce la risposta alla query. Trudy utilizza questa opportunità per falsificare la risposta alla query e quindi inserire un indirizzo IP falso nella cache del server DNS. Per semplicità assumiamo che l'ISP di Alice inizialmente non abbia informazioni sull'IP del sito Web di Bob, *bob.com*; in caso contrario Trudy può aspettare che questa informazione scada per timeout e quindi riprovare (oppure può usare degli altri trucchi). Trudy comincia inviando all'ISP di Alice la richiesta dell'indirizzo IP di *bob.com*. Visto che non ci sono informazioni per questo nome DNS, la richiesta viene passata al server top-level per il dominio *com*. Invece, quello che succede è che Trudy batte in velocità il server *com*, inviando una risposta falsa che dice: "*bob.com* è 42.9.9.9", dove l'indirizzo IP è quello di Trudy. Se la falsa risposta arriva all'ISP di Alice prima di quella vera, entrerà nella cache e causerà il rifiuto della risposta corretta, che sarà scartata come non desiderata visto che la query è stata già soddisfatta. Ingannare un server DNS per installare un indirizzo IP falso è detto **DNS spoofing** (inganno del DNS). Una cache che contiene un indirizzo IP intenzionalmente falsificato, come in questo esempio, è detta **Poisoned cache** (cache avvelenata).

Nella realtà le cose non sono così semplici. Come prima cosa, l'ISP di Alice controlla che la risposta contenga come indirizzo IP sorgente quello del server top-level. Trudy però aggira facilmente il test, visto che è in grado di mettere quello che vuole in quel campo sorgente dell'IP e che gli indirizzi IP dei server top-level sono pubblici.

Un'altra difficoltà è generata dal fatto che tutte le richieste contengono un numero di sequenza, per consentire ai server DNS di far corrispondere le richieste con le risposte.

Questo implica che, per riuscire a ingannare l'ISP di Alice, Trudy deve conoscere il numero di sequenza corretto. Il modo più semplice che Trudy ha a disposizione per venire a conoscenza del numero di sequenza corrente è quello di registrare un proprio dominio, diciamo *trudy-the-intruder.com*. Supponiamo che l'indirizzo IP associato sia di nuovo 42.9.9.9. Trudy crea anche un server DNS per il suo dominio appena nato, *dns.trudy-the-intruder.com*. Anche questo server usa l'indirizzo IP 42.9.9.9, visto che Trudy ha un solo computer. Adesso Trudy deve far conoscere il suo DNS server all'ISP di Alice. Questa è un'operazione semplice, basta che chieda all'ISP di Alice di risolvere *qualcosa.trudy-the-intruder.com*. L'ISP andrà a cercare chi risolve gli indirizzi del dominio di Trudy con una ricerca sul server top-level *com*.

Avendo messo nella cache dell'ISP di Alice il dns *dns.trudy-the-intruder.com*, Trudy può lanciare il suo attacco con una query verso l'ISP per l'indirizzo *www.trudy-the-intruder.com*. L'ISP invierà a sua volta una query al DNS server di Trudy per risolvere il nome, che contiene il numero di sequenza che Trudy cercava. Veloce come un lampo, Trudy chiede all'ISP di Alice di cercare l'indirizzo di Bob e immediatamente invia una risposta alla sua stessa domanda inviando all'ISP una risposta falsificata, che dovrebbe, in teoria, provenire dal server top-level *com* e che afferma: "bob.com è il 42.9.9.9". Questa risposta falsificata contiene un numero di sequenza pari a quello ricevuto in precedenza, ma incrementato di una unità. Per avere una maggior probabilità di riuscita, Trudy invia anche un altro pacchetto con il numero di sequenza incrementato di due e, se vuole, anche un altro gruppo di pacchetti con numeri di sequenza crescenti. Solo uno di questi pacchetti avrà il numero di sequenza corretto, gli altri pacchetti saranno semplicemente scartati. La risposta falsificata viene messa in cache una volta arrivata a destinazione, mentre la risposta legittima è scartata in quanto arriva quando la query è stata ormai risolta.

Adesso quando Alice cerca *bob.com* viene indirizzata verso l'IP 42.9.9.9, l'indirizzo di Trudy, che quindi è riuscita in un attacco di tipo "uomo nel mezzo" restando nel suo salotto. I passi dell'attacco sono illustrati dalla Figura 8.47. Per complicare la faccenda, questo non è il solo modo per ingannare un DNS, ce ne sono infatti molti altri.

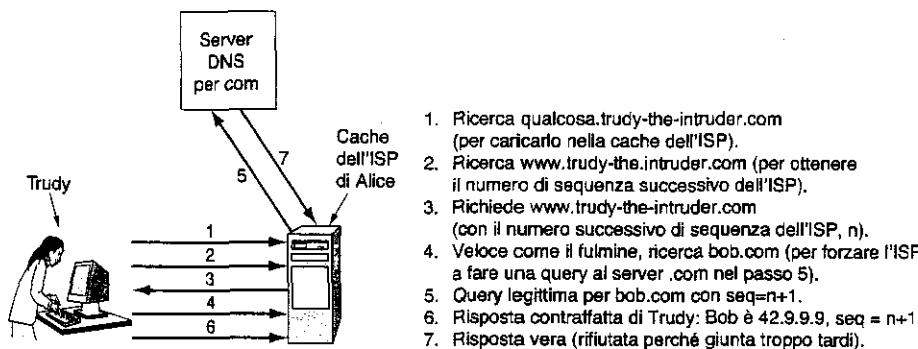


Figura 8.47. Ecco come Trudy forza l'ISP di Alice

### DNS sicuro

Questo specifico tipo di attacco può essere evitato se il DNS usa dei numeri di sequenza casuali nelle query, invece di utilizzare semplicemente dei numeri progressivi. In questo modo si tappa una falla di sicurezza, ma nel sistema emergono altri buchi. Il vero problema del DNS è che fu progettato quando Internet era usata come supporto alla ricerca da poche centinaia di università, quindi né Alice, né Bob, né tanto meno Trudy erano sulla scena. La sicurezza non era un problema a quei tempi, l'unica cosa importante era riuscire a far funzionare Internet. Le condizioni sono cambiate radicalmente negli anni, così nel 1994 IETF decise di formare un gruppo di lavoro per rendere DNS fondamentalmente sicuro. Questo progetto è conosciuto col nome di **DNSsec (DNS security)** ed è presentato nell'RFC 2535. Sfortunatamente, DNSsec non è ancora usato da tutti, con la conseguenza che svariati server DNS sono tuttora vulnerabili con attacchi di spoofing.

DNSsec è basato sulla crittografia a chiave pubblica ed è concettualmente molto semplice. Ogni zona DNS (nel senso della Figura 7.4) ha una coppia di chiavi pubblica/privata. Tutte le informazioni inviate da un DNS sono firmate con la chiave privata della zona d'origine, quindi il ricevente ne può verificare l'autenticità.

DNSsec offre tre servizi fondamentali:

1. la prova del luogo di provenienza dei dati
2. la distribuzione della chiave pubblica
3. l'autenticazione della transazione e della richiesta.

Il primo è il servizio principale, poiché garantisce che i dati di ritorno sono stati approvati dal proprietario della zona. Il secondo è impiegato per memorizzare e utilizzare in modo sicuro le chiavi pubbliche. Il terzo è necessario per salvaguardarsi contro attacchi di tipo ripetizione o spoofing. Notiamo che la segretezza non è offerta come servizio, in quanto l'informazione contenuta nel DNS è considerata pubblica. Si stima che serviranno alcuni anni per completare l'attivazione di tutti i DNSsec, per questo motivo è essenziale che i server attenti alla sicurezza siano in grado di comunicare con quelli che ignorano questi problemi: ciò implica che il protocollo non può essere cambiato. Vedremo nel seguito alcuni dettagli dell'implementazione di DNSsec.

I record DNS sono raggruppati in insiemi chiamati **RRSet (Resource Record Set)**, dove tutti i record di un insieme hanno lo stesso nome, classe e tipo. Un RRSet può contenere più A-record se, per esempio, un nome DNS viene risolto con un indirizzo IP primario e uno secondario. Gli RRSet sono stati estesi con diversi nuovi tipi di record, discussi più avanti. Per ogni RRSet si calcola un hash crittografico (per esempio usando MD5 o SHA-1) e l'hash viene firmato con la chiave privata della zona (per esempio usando RSA); quindi l'unità di trasmissione verso i client è l'RRSet firmato. Alla ricezione di un RRSet, il client può verificare se è stato firmato con la chiave privata della zona di origine, perciò il dato viene accettato solo se la firma è integra. Visto che ogni RRSet contiene la propria firma, gli RRSet possono essere messi in cache ovunque, anche su server non fidati, senza che questo comprometta la sicurezza.

DNSsec introduce nuovi tipi di record. Il primo di questi è il record **KEY**, che contiene la chia-

ve pubblica di una zona, utente, host o altro protagonista; l'identificatore dell'algoritmo crittografico utilizzato per la firma; il protocollo usato per la trasmissione, e alcuni altri bit. La chiave pubblica viene memorizzata nuda, senza usare un certificato X.509, per via del peso aggiuntivo che questo comporterebbe. Il campo con l'algoritmo contiene un 1 per le firme MD5/RSA (la scelta preferita), mentre valori diversi indicano altre combinazioni di algoritmi. Il campo del protocollo indica se vengono utilizzati IPsec o altri protocolli di sicurezza. Il secondo nuovo tipo di record è *SIG*, e contiene l'hash firmato con l'algoritmo specificato dal record *KEY*. La firma si applica a tutti i record dell'RRSet, quindi sono inclusi tutti i record *KEY* ma è esclusa la firma stessa (cioè *SIG*). *SIG* contiene anche l'orario in cui la firma comincia il suo periodo di validità, l'orario di scadenza, il nome di chi firma e alcune altre informazioni.

La struttura di DNSsec permette di mantenere la chiave privata della zona fuori dalla rete. Una o due volte al giorno, i contenuti del database di zona possono essere trasportati manualmente (per esempio su CD-ROM) su una macchina non connessa alla rete, dove è memorizzata la chiave privata. Tutti gli RRSet possono essere firmati su questa macchina, quindi i record *SIG* calcolati vengono trasportati di nuovo sul server primario della zona usando un altro CD-ROM. Operando in questo modo, la chiave privata può essere conservata in un CD-ROM che rimane chiuso in cassaforte per tutto il giorno, eccetto che per il tempo necessario alle operazioni di firma dei nuovi RRSet sulla macchina disconnessa dalla rete. Dopo che le operazioni di firma sono concluse, tutte le copie della chiave vengono cancellate dalla memoria e il CD-ROM ritorna in cassaforte. Questa procedura riduce il problema della sicurezza elettronica a quello della sicurezza fisica, qualcosa che la gente capisce e riesce a gestire.

Questo metodo, che genera RRSet firmati in anticipo, aiuta a velocizzare le risposte alle query, visto che elimina la necessità di effettuare calcoli crittografici al momento della risposta. Ha però lo svantaggio di richiedere grandi quantità di spazio disco, per memorizzare tutte le chiavi e le loro firme dentro al database DNS. Alcuni record possono crescere fino a dieci volte la loro dimensione originale, una volta firmati. Quando un processo client riceve un RRSet firmato, deve applicare la chiave pubblica della zona d'origine per decifrare l'hash ricevuto, calcolare il proprio hash e confrontare i due valori. I dati sono considerati validi se questi coincidono. Tuttavia questa procedura si porta dietro il problema di come il client possa prelevare la chiave pubblica della zona. Un modo può essere quello di acquisirla da un server fidato, usando una connessione sicura (per esempio usando IPsec).

Nella pratica, si suppone che i client siano preconfigurati con le chiavi pubbliche di tutti i domini top-level. Se Alice vuole visitare il sito Web di Bob, può chiedere al DNS l'RRSet di *bob.com*, che contiene l'indirizzo IP e un record *KEY* con la chiave pubblica di Bob. Questo RRSet sarà firmato dal dominio top-level *com*, il che permette ad Alice di verificarne facilmente la validità. Un esempio del contenuto di questo RRSet è mostrato nella Figura 8.48.

Avendo un copia della chiave pubblica di Bob, Alice può procedere chiedendo al DNS server di Bob (che è gestito da Bob stesso) l'indirizzo IP di *www.bob.com*. Questo RRSet sarà firmato con la chiave privata di Bob, quindi Alice può verificare la firma dell'RRSet che Bob gli ha restituito. Se in qualche modo Trudy riuscisse a inserire un falso RRSet in una qualunque delle cache del sistema, Alice sarebbe facilmente in grado di rilevare la contraffazione, perché i contenuti del record *SIG* risulterebbero non corretti.

Domain name	Time to live	Class	Type	Value
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C...

Figura 8.48. Esempio di RRSet per *bob.com*. Il record *KEY* è la chiave pubblica di Bob. Il record *SIG* è l'hash (firmato dal server *com* di livello più alto) dei record *A* e *KEY*, che ne conferma l'autenticità.

DNSsec fornisce anche un metodo crittografico per legare le risposte alle query che le hanno generate. Ciò serve per prevenire lo spoofing che Trudy era riuscita a perpetrare nel caso della Figura 8.47. Questo meccanismo (opzionale) di antispoofing funziona aggiungendo alla risposta un hash del messaggio di query firmato con la chiave privata di chi risponde. Dato che Trudy non conosce la chiave privata del server top-level *com*, non riesce neanche a falsificare la risposta alla query che è stata inviata dall'ISP di Alice. Trudy può sicuramente far sì che la sua risposta raggiunga l'ISP per prima, ma verrà poi scartata in quanto non ha una firma valida dell'hash della query originale. DNSsec supporta anche altri tipi di record. Per esempio, il record *CERT* può essere usato per memorizzare certificati (per esempio X.509). Questo tipo di record è stato inserito perché c'è chi vuole trasformare DNS in una PKI, ma al momento questo progetto è ancora tutto da pensare. Concludiamo qui la nostra discussione su DNSsec, rimandando all'RFC 2535 per ulteriori dettagli.

### Nomi autocertificanti

Il DNS sicuro non è l'unica possibilità per mettere in sicurezza la risoluzione dei nomi. Un approccio completamente differente viene utilizzato nel **Secure File System** (Mazières et al. 1999). In questo progetto gli autori hanno disegnato un file system con estensione mondiale, sicuro e scalabile, il tutto senza modificare il DNS (standard) e senza usare certificati o assumere l'esistenza di una PKI. In questo paragrafo vedremo come tutto ciò è applicabile al Web e quindi useremo, nella nostra descrizione, una terminologia legata alle applicazioni Web al posto dei termini usati nell'articolo originale (in cui veniva utilizzata la terminologia tipica dei file system). A scanso di equivoci evidenziamo che, sebbene questo schema possa essere applicato al Web per ottenere una maggiore sicurezza, non è comunque utilizzato al momento, poiché per poterlo utilizzare servirebbero cambiamenti sostanziali nel software.

Cominciamo facendo l'ipotesi che ogni Web server abbia una coppia di chiavi pubblica/privata. Il fulcro dell'idea è che ogni URL contenga al suo interno anche un hash crittografico del nome del server e della chiave pubblica. Per esempio, nella Figura 8.49 vediamo l'URL per la foto di Bob. Questo URL comincia con il solito schema *http* seguito dal nome DNS del server (*www.bob.com*), segue il carattere ":" e un hash che occupa 32 caratteri; alla fine c'è come al solito il nome del file. Questo è un URL standard con in più l'aggiunta dell'hash. Avendo aggiunto l'hash, otteniamo un **URL autocertificante**.

Server	SHA-1 (server, chiave pubblica del server)	Nome del file
http://www.bob.com:2g5hd8bfjkc7mf6hg8dgany23xds4pe6/photos/bob.jpg		

**Figura 8.49.** URL autocertificante che contiene l'hash del nome del server e la sua chiave pubblica.

Da questo discorso viene spontaneo chiedersi a che cosa serve l'hash, che si calcola concatenando il nome DNS del server con la chiave pubblica del server stesso, e passando il risultato attraverso l'algoritmo SHA-1 per ottenere un risultato a 160 bit. In questo schema l'hash rappresenta una sequenza di 32 cifre e lettere minuscole, con l'eccezione delle lettere "I" e "O" e delle cifre "0" e "1" che non sono ammesse per evitare possibili confusioni. Rimangono quindi 32 possibili scelte fra lettere e cifre, e sappiamo che avendo a disposizione un alfabeto di 32 caratteri ogni simbolo può rappresentare una stringa di 5 bit. In altri termini, per rappresentare l'hash SHA-1 a 160 bit si può usare una stringa di 32 caratteri. In effetti non sarebbe indispensabile usare un hash (si potrebbe anche usare la chiave), ma l'hash è conveniente perché permette di ridurre la lunghezza del nome. Il modo più semplice (ma meno conveniente) che Alice potrebbe adottare per vedere la foto di Bob consiste semplicemente nel digitare la stringa della Figura 8.49 nel browser. Il browser manda un messaggio al sito Web di Bob e gli chiede la chiave pubblica. Quando la chiave è consegnata ad Alice, il browser prende questo valore, lo concatena al nome del server e calcola l'hash. Se il valore calcolato corrisponde con l'hash a 32 caratteri contenuto nella URL sicura, il browser è sicuro di aver ottenuto la chiave pubblica di Bob. Questo perché, viste le proprietà di SHA-1, anche se Trudy intercetta la richiesta e falsifica una risposta non riesce comunque a trovare una chiave pubblica che fornisca l'hash atteso: ogni interferenza da parte di Trudy è rilevata. La chiave pubblica di Bob può essere tenuta in cache per usi futuri.

A questo punto Alice deve verificare che Bob abbia anche la chiave privata corrispondente. Lo fa costruendo un messaggio con la proposta per la session key AES, un nonce e un timestamp, quindi cifra il messaggio con la chiave pubblica di Bob e glielo invia. Bob è l'unico che può decifrare il messaggio, usando la conoscenza della chiave privata, perciò può restituire ad Alice il nonce cifrato con la chiave AES. A questo punto Alice è sicura di essere veramente in comunicazione con Bob, inoltre Alice e Bob hanno stabilito un session key AES, che useranno per le successive richieste *GET* e per le risposte.

Quando Alice avrà ottenuto la foto di Bob (o una qualunque altra pagina Web), la potrà mettere nella lista dei siti preferiti del browser, così la prossima volta non dovrà più digitare l'intero URL. Anche gli URL contenuti nelle pagine Web possono essere autocertificanti: è possibile aprirle con un semplice clic, ma con la sicurezza che le pagine visualizzate siano veramente quelle richieste. Un altro modo per evitare di inserire a mano URL autocertificanti all'inizio della navigazione consiste nel prenderle da un server fidato tramite una connessione sicura; oppure si possono memorizzare in un certificato X.509 firmato da una CA.

URL autocertificanti si potrebbero ottenere anche connettendosi a un motore di ricerca fidato, di cui si conosce l'URL autocertificante (che andrà digitata solo alla prima connessione). A questo punto basta seguire il protocollo descritto sopra per ottenere una connessione sicura e autenticata con il motore di ricerca. Successivamente si possono inviare query, che pro-

durranno come risposta una pagina firmata contenente gli URL autocertificanti: a queste pagine si accede con un solo clic, senza dover scrivere l'intera stringa delle loro URL. Vediamo come questo approccio resiste ai tentativi di spoofing di Trudy. Se Trudy riesce ad avvelenare la cache dell'ISP di Alice, le richieste della vittima possono essere falsamente indirizzate verso Trudy invece che verso Bob. Il protocollo adesso richiede che il destinatario del messaggio iniziale (cioè Trudy, in questo scenario) debba restituire la sua chiave pubblica. Quindi Alice sarà in grado di accorgersi immediatamente dell'attacco, poiché l'hash SHA-1 calcolato non corrisponderà a quello dell'URL autocertificante. Se invece Trudy rende la chiave pubblica di Bob, Alice non si accorgerà dell'attacco, però il successivo messaggio di Alice sarà cifrato con la chiave di Bob. Trudy riceverà quindi il messaggio, ma non sarà in grado di decifrarlo per estrarre la chiave AES e il nonce. La conclusione è che l'unico attacco che Alice può fare al DNS con questo nuovo protocollo è un attacco di tipo Dos.

### 8.9.3 SSL (*Secure Socket Layer*)

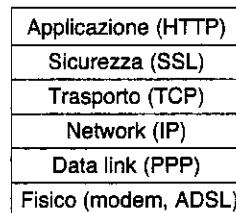
Avere la sicurezza nel DNS è un buon inizio, ma la sicurezza del Web è un argomento molto più vasto. Il passo successivo consiste nel mettere in sicurezza le connessioni. Vedremo adesso come realizzare questo obiettivo.

Durante la sua prima fase di rapida espansione, il Web era usato essenzialmente per distribuire pagine statiche. Poco dopo le aziende hanno avuto l'idea di usarlo per gestire transazioni finanziarie, come l'acquisto di merce con la carta di credito, l'on-line banking e il trading azionario: queste applicazioni hanno creato una domanda per la gestione di connessioni sicure. Nel 1995, Netscape Communications Corp, che allora dominava il mercato dei produttori di browser, rispose alla domanda introducendo un pacchetto per la sicurezza chiamato SSL (*Secure Socket Layer*). Oggi questo software e il suo protocollo sono ampiamente usati anche da Microsoft Internet Explorer, quindi vale sicuramente la pena di esaminarlo.

SSL instaura una connessione sicura fra due socket, con le seguenti caratteristiche:

1. negoziazione dei parametri fra client e server
2. autenticazione mutua fra client e server
3. comunicazione segrete
4. protezione dell'integrità dei dati.

Abbiamo già visto questo tipo di caratteristiche, quindi non le commenteremo ulteriormente. Il posizionamento di SSL nella pila dei protocolli di comunicazione è mostrato nella Figura 8.50: SSL è un nuovo strato interposto fra lo strato applicazione e quello di trasporto. SSL accetta richieste dal browser e le manda al TCP perché siano trasmesse al server; dopo aver stabilito la connessione il compito principale di SSL è quello di gestire la compressione e la cifratura. HTTP usato sopra SSL prende il nuovo nome di **HTTPS** (Secure HTTP), anche se si tratta ancora del protocollo HTTP standard. A volte HTTPS è disponibile su porta diversa (443) dallo standard (80). Per inciso, non c'è nessuna restrizione che lega SSL ai browser Web, ma questa è la sua applicazione più comune.

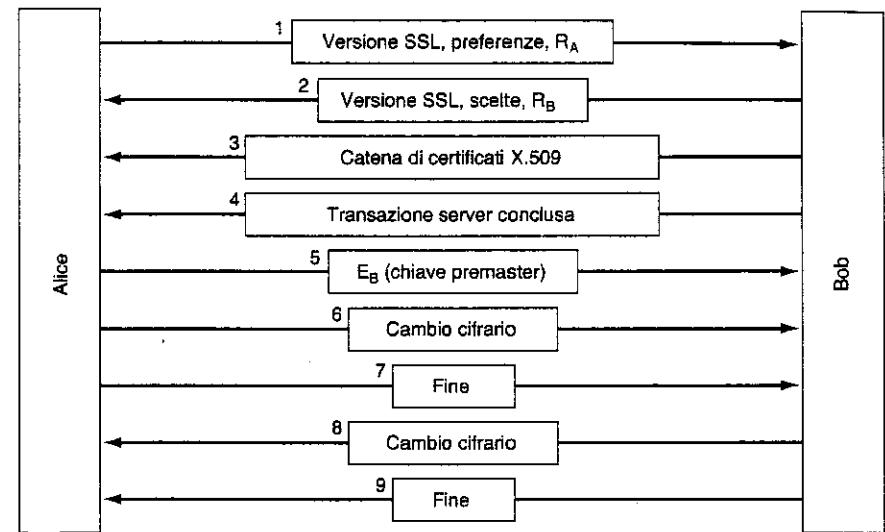


**Figura 8.50.** Strati (e protocolli) per un utente domestico che esegue il browsing del Web con SSL.

sistono diverse versioni del protocollo SSL; discuteremo però solo la versione 3, che è quella più largamente utilizzata. SSL supporta molti algoritmi e opzioni che indicano, per esempio, la presenza o assenza di compressione, gli algoritmi crittografici utilizzati e altri dettagli legati alle restrizioni sull'esportazione di sistemi crittografici. Ciò garantisce che la crittografia veramente forte è utilizzabile solo quando entrambi gli estremi della connessione sono nel territorio degli USA, mentre negli altri casi le chiavi vengono limitate a 40 bit, un livello di sicurezza che i crittografi considerano una specie di scherzo. Netscape è stata obbligata ad aggiungere questa restrizione per poter avere una licenza all'esportazione da parte del governo degli Stati Uniti. SSL consiste di due subprotocolli, uno per stabilire le connessioni sicure e uno per usarle. Cominciamo a vedere come vengono stabilite le connessioni sicure. Il subprotocollo per le connessioni sicure è mostrato nella Figura 8.51. Comincia con il messaggio 1, nel quale Alice manda a Bob la richiesta di stabilire una connessione. La richiesta specifica anche la versione di SSL che Alice utilizza e le sue preferenze per quanto riguarda la compressione e gli algoritmi crittografici. Contiene anche un nonce,  $R_A$ , che verrà usato in seguito.

Ora è il turno di Bob, che manda il messaggio 2 contenente la sua scelta (fatta fra gli algoritmi che Alice supporta) e un nonce  $R_B$ . Poi Bob invia con il messaggio 3 un certificato contenente la sua chiave pubblica. Se il certificato non è firmato da un'autorità nota, Bob invia anche una catena di certificati che permettono di risalire fino a essa. Tutti i browser, incluso quello di Alice, contengono all'installazione già un centinaio di chiavi pubbliche. Alice sarà in grado di verificare la chiave pubblica di Bob se questi riesce a stabilire una catena che fa capo a una chiave pubblica preinstallata nel browser. Ora Bob invia messaggio 4 ad Alice, dicendole che è arrivato il suo turno.

Alice risponde scegliendo una chiave casuale di 384 bit detta **premaster key**, che invia a Bob in forma cifrata usando la chiave pubblica di Bob (messaggio 5). La session key vera e propria è il risultato di un complesso calcolo basato sulla premaster key, in combinazione con i due nonce. Alice e Bob sono entrambi in grado di calcolarla solo dopo che Bob ha ricevuto il messaggio 5. Per questo motivo ora Alice dice a Bob di cominciare a usare session key (messaggio 6) e che il subprotocollo di connessione ha finito il suo compito (messaggio 7). Con i messaggi 8 e 9 Bob conferma la ricezione dei messaggi 6 e 7. A questo punto Alice conosce l'identità di Bob, però Bob non sa chi è Alice (a parte il caso in cui Alice abbia una chiave pubblica e il relativo certificato, situazione strana per un individuo). Il primo messaggio di Bob sarà probabilmente una richiesta ad Alice perché inserisca



**Figura 8.51.** Versione semplificata del subprotocollo SSL per instaurare la connessione.

login name e password, tuttavia il protocollo di login è al di fuori degli scopi dell'SSL. Il trasporto dei dati comincia dopo il login, a prescindere dal modo in cui è effettuato.

SSL supporta diversi algoritmi crittografici. Il più forte utilizza per la cifratura il DES triplo con tre chiavi separate, oltre a SHA-1 per gestire l'integrità dei messaggi. La combinazione di questi algoritmi è relativamente lenta, perciò si usa tipicamente solo per le applicazioni bancarie o comunque là dove è richiesto il massimo grado di sicurezza. Per le ordinarie applicazioni di e-commerce viene impiegato RC4 con una chiave a 128 bit per la cifratura, e MD5 per l'autenticazione d'integrità dei messaggi. RC4 utilizza la chiave a 128 bit come un seme, che viene espanso in un numero più grande utilizzato nel funzionamento interno dell'algoritmo per generare il keystream, poi applicato in XOR con il testo in chiaro per generare il classico flusso cifrato, come abbiamo visto nella Figura 8.14. La versione per l'export utilizza anch'essa RC4 con le chiavi a 128 bit, ma 88 di questi bit sono resi pubblici per rendere il cifrario più facile da forzare.

Per il trasporto vero e proprio si usa un secondo subprotocollo, mostrato nella Figura 8.52. I messaggi provenienti dal browser sono innanzitutto suddivisi in unità di 16 KB. Se la compressione è abilitata, ogni unità viene compressa singolarmente. Si procede con il calcolo di una chiave segreta a partire dai due nonce e dalla premaster key: questa chiave viene concatenata al testo in chiaro e il tutto è passato attraverso l'algoritmo di hash negoziato in precedenza (di solito MD5). Tale hash viene aggiunto a ogni frammento come MAC (*Message Authentication Code*); quindi il frammento compresso e il suo MAC vengono cifrati con l'algoritmo di crittografia simmetrica negoziato in precedenza (di solito questo è fatto calcolando lo XOR con il keystream RC4). Infine, viene aggiunta un'intestazione a ogni frammento e si procede alla sua trasmissione tramite la connessione TCP.

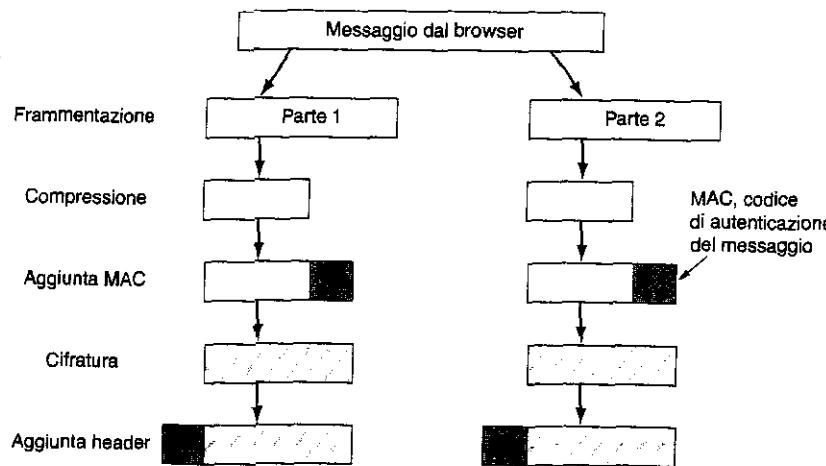


Figura 8.52. Trasmissione dei dati con SSL.

È necessaria una particolare attenzione quando si usa RC4. È stato dimostrato che alcune chiavi deboli di questo algoritmo si possono facilmente analizzare con la criptoanalisi, quindi la sicurezza di SSL con RC4 è un po' traballante (Fluhrer et al. 2001). I browser che permettono all'utente di scegliere gli algoritmi di cifratura utilizzati andrebbero configurati per usare il triplo DES con chiavi a 168 bit e SHA-1, anche se questa combinazione è più lenta di RC4 e MD5.

Un altro problema con SSL è dato dal fatto che i protagonisti possono anche non avere dei certificati; anche se li hanno, non sempre verificano che le chiavi utilizzate corrispondano ai valori inseriti nei certificati.

Nel 1996 Netscape Corp. sottopose SSL all'IETF per la standardizzazione. Il risultato fu TLS (*Transport Layer Security*), che è descritto nell'RFC 2246.

I cambiamenti operati all'SSL furono minimi, ma quanto basta per far sì che SSL versione 3 e TLS non riescano a comunicare fra loro. Per esempio, il modo in cui la session key viene calcolata a partire dai nonce e dalla premaster key fu cambiato per rendere la chiave più forte (cioè più difficile da analizzare con la criptoanalisi).

La versione TLS è anche nota sotto il nome di SSL 3.1. La prima implementazione di TLS è apparsa nel 1999, ma non è ancora chiaro se TLS soppiangerà SSL nella pratica, anche se è leggermente più robusto. Comunque, il problema della debolezza delle chiavi RC4 non è stato risolto.

#### 8.9.4 Sicurezza del codice mobile

Gestione dei nomi e connessioni sono due aree sensibili nello scenario della sicurezza del Web, ma non sono le uniche. Agli inizi, quando le pagine Web erano semplicemente dei file HTML statici, non contenevano del codice eseguibile. Al giorno d'oggi, invece, le pagine Web contengono spesso piccoli programmi come applet Java, controlli ActiveX e

JavaScript. Scaricare ed eseguire questo **codice mobile** è ovviamente un grande rischio per la sicurezza. Per ridurre i rischi sono stati escogitati diversi metodi, quindi daremo un'occhiata veloce ad alcune delle problematiche generate dal codice mobile e agli approcci per affrontarle.

#### Sicurezza delle applet Java

Le applet Java sono dei piccoli programmi Java, compilati in un linguaggio macchina orientato allo stack chiamato JVM (*Java Virtual Machine*). Le applet si possono mettere su una pagina Web, pronte per essere scaricate insieme alla pagina stessa. Dopo lo scaricamento, le applet vengono inserite nell'interprete JVM che si trova dentro al browser, come illustrato nella Figura 8.53.

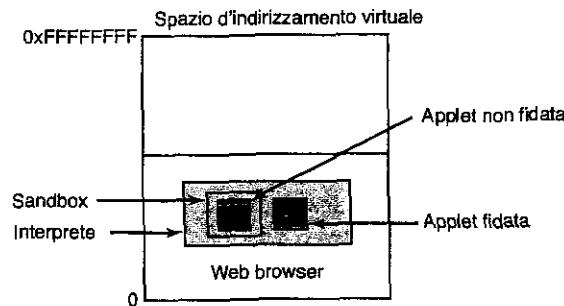


Figura 8.53. Le applet vengono interpretate dal browser Web.

Il vantaggio nell'utilizzare codice interpretato al posto di quello compilato sta nel fatto che ogni istruzione può essere esaminata dall'interprete prima di essere eseguita. Questo dà all'interprete l'opportunità di controllare la validità dell'indirizzo dell'istruzione; inoltre anche le chiamate di sistema possono essere intercettate e interpretate. Se un'applet è fidata (magari perché viene dal disco locale), le sue chiamate di sistema possono essere eseguite senza ulteriori domande. Se invece l'applet non è fidata (per esempio viene da Internet), si può procedere a incapsularla in una cosiddetta **sandbox**, che ne restringe il campo d'azione e intercetta tutti i tentativi di utilizzo delle risorse di sistema.

Quando un'applet cerca di usare una risorsa di sistema, la sua chiamata è passata al monitor della sicurezza per approvazione. Il monitor esamina la chiamata alla luce delle regole locali di sicurezza e decide se permettere o respingere la chiamata. In questo modo è possibile dare alle applet l'accesso ad alcune risorse di sistema, ma non a tutte. Sfortunatamente la realtà è che questo modello di sicurezza funziona male, ed è sempre pieno di errori.

#### ActiveX

I controlli ActiveX sono programmi eseguibili per Pentium, racchiusi dentro alle pagine Web. Quando s'incontra una pagina che contiene un ActiveX, per prima cosa viene fatto un controllo per vedere se deve essere eseguito: se supera il test, viene effettivamente man-

dato in esecuzione. I controlli ActiveX non sono interpretati né eseguiti in una sandbox, cioè hanno lo stesso potere degli eseguibili dell'utente e pertanto possono teoricamente causare dei grandi danni. Tutta la sicurezza sta nel decidere se si deve eseguire il controllo ActiveX oppure no.

Il metodo scelto da Microsoft per prendere questa decisione è basato sull'idea del **code signing**. Ogni controllo ActiveX è accompagnato da una firma digitale, cioè un codice hash firmato dal creatore usando la crittografia a chiave pubblica. Quando compare un controllo ActiveX, il browser verifica subito la firma per essere sicuro che non è stato modificato durante la trasmissione. Se la firma è corretta, il browser controlla le sue tabelle interne per vedere se il creatore del programma è fidato oppure se esiste una catena di certificati che portano a un creatore fidato. Se il creatore è fidato, il programma è eseguito, altrimenti no. Il sistema Microsoft per verificare i controlli ActiveX è chiamato **Authenticode**.

È istruttivo confrontare gli approcci di Java e ActiveX. Nell'approccio di Java non si fa nessun tentativo per determinare l'autore dell'applet; viene invece usato un interprete runtime che ci assicura che l'applet non faccia cose che il proprietario della macchina ha proibito. Al contrario, con il code signing non viene fatto nessun tentativo di verificare se il codice è sicuro o no: se il programmatore che ha scritto il codice *voleva* che durante l'esecuzione venisse formattato l'hard disk e cancellata la flash ROM, in modo che il computer non potrà più essere fatto ripartire, allora il codice verrà eseguito e distruggerà il computer (se i controlli ActiveX sul browser sono abilitati).

Molti pensano che fidarsi di un produttore di software sconosciuto sia troppo pericoloso. Per dimostrare il problema, un programmatore di Seattle fondò una software house e fu certificato come fidato, attività entrambe facili. Realizzò poi un controllo ActiveX che faceva uno shutdown pulito della macchina e lo distribuì in giro. Il risultato fu che molte macchine vennero spente da quel controllo ActiveX (ma ripartirono senza nessun danno). Il programmatore voleva solamente portare all'attenzione del mondo il problema dell'esecuzione degli ActiveX. La risposta ufficiale fu quella di revocare il certificato per quello specifico controllo Activex, il che pose fine all'imbarazzante episodio, ma il problema sottostante è ancora lì, in attesa che qualche programmatore malintenzionato lo voglia sfruttare (Garfinkel e Spafford, 2002). Visto che non c'è modo di controllare migliaia di produttori di software che possono scrivere codice mobile, il risultato è che la tecnica del code signing rappresenta un disastro in attesa di accadere.

## JavaScript

JavaScript non ha nessun modello formale di sicurezza: quello che ha è una lunga storia d'implementazioni con numerose falle. Ogni produttore gestisce la sicurezza in un modo differente, per esempio Netscape Navigator versione 2 usa qualcosa di simile al modello di Java, mentre la versione 4 implementa un modello più simile al code signing.

Il problema fondamentale è che lasciar girare sulla propria macchina codice scritto da sconosciuti vuol dire andare a cercare i problemi. Dal punto di vista della sicurezza, è come invitare un ladro nella propria casa, per poi cercare di tenerlo sotto controllo in modo che

non fugga dalla cucina al salotto. Se accade qualcosa d'inaspettato e vi distraete per un momento, possono accadere cose spiacevoli. Il problema è che il codice mobile permette di generare grafica accattivante e controlli interattivi veloci, quindi molti Web designer pensano che queste funzioni siano molto più importanti della sicurezza, specialmente quando sono le macchine degli altri a essere a rischio.

## Virus

I virus sono un'altra forma di codice mobile. Al contrario degli esempi precedenti, i virus non sono affatto benvenuti. La differenza fra un virus e un tipico codice mobile è che i virus sono scritti per riprodursi. Quando un virus arriva, tramite una pagina Web, un allegato di e-mail o in qualche altro modo, di solito comincia a infettare i programmi eseguibili sull'hard disk. Quando uno di questi programmi viene eseguito, il controllo passa al virus che (di solito) tenta di diffondersi verso altre macchine: per esempio manda per e-mail delle copie di sé stesso a tutte le persone nell'agenda della vittima. Alcuni virus infettano il settore di boot dell'hard disk, in questo modo vanno in esecuzione a ogni riavvio della macchina. I virus sono diventati un enorme problema su Internet e hanno già causato danni per miliardi di dollari. Non ci sono soluzioni semplici. Un aiuto potrebbe arrivare da una nuova generazione di sistemi operativi basati su microkernel sicuri, con una rigida divisione degli utenti in comparti stagni.

## 8.10 Aspetti sociali

Internet e la sua tecnologia per la sicurezza sono un'area dove questioni sociali, leggi e tecnologia s'incontrano, spesso con conseguenze importanti. Nel seguito esamineremo brevemente tre aree: privacy, libertà di parola e copyright. Ovviamente in questa sede possiamo solamente accennare ai problemi fondamentali; per ulteriori approfondimenti si consiglia (Anderson, 2001; Garfinkel e Spafford, 2002; Schneier, 2000). Anche Internet è pieno di materiale sull'argomento, basta digitare in un motore di ricerca le parole "privacy", "censorship" e "copyright". Anche il sito Web di questo libro ha alcuni link interessanti.

### 8.10.1 Privacy

Le persone hanno diritto alla privacy? Questa è una buona domanda. Il quarto emendamento della costituzione degli USA proibisce al governo di perquisire le abitazioni private, i documenti e gli effetti personali senza che esista una buona ragione; inoltre specifica le circostanze in cui un mandato di perquisizione può essere emesso dalle autorità. Possiamo dire che negli Stati Uniti l'argomento della privacy viene discusso pubblicamente da almeno 200 anni.

Ciò che è cambiato nell'ultimo decennio sono da una parte l'efficacia con cui il governo riesce a spiare i cittadini, e dall'altra i metodi a disposizione dei cittadini per prevenire tali operazioni. Nel diciottesimo secolo, se il governo voleva spiare i documenti di un cittadino doveva mandare un poliziotto a cavallo alla fattoria del cittadino e chiedere che gli fos-

sero mostrati certi documenti. Era una procedura lunga e difficile. Al giorno d'oggi, le compagnie telefoniche e gli Internet provider, alla presenza di un mandato, possono fornire velocemente alle autorità la possibilità di ascoltare le linee.

La crittografia cambia questo scenario. Chiunque si prenda il disturbo di scaricare e installare PGP, usando una chiave ben custodita e di lunghezza sufficiente, può essere decisamente sicuro che nessuno nell'universo riuscirà a leggere la sua e-mail, con o senza mandato di perquisizione. I governi hanno capito bene questo fatto e non lo apprezzano per niente. La vera privacy significa che è molto più difficile spiare i criminali di tutte le specie, ma anche che è diventato più difficile spiare i giornalisti e gli avversari politici. Di conseguenza, alcuni governi limitano o proibiscono l'esportazione della crittografia. In Francia, per esempio, prima del 1999 la crittografia era fuori legge, a patto che il governo non fosse a conoscenza delle chiavi.

La Francia non era l'unico caso. Nell'aprile 1993 il governo degli Stati Uniti annunciò l'intenzione di rendere standard per tutte le comunicazioni di rete un criptoprocessore hardware, chiamato **clipper chip**. Veniva anche detto che ciò assicurava la privacy dei cittadini, mentre il chip avrebbe permesso al governo di decifrare tutto il traffico con uno schema chiamato **key escrow** (che dava l'accesso a tutte le chiavi). Naturalmente il governo prometteva che avrebbe ascoltato le comunicazioni solo in presenza di un mandato valido. Ovviamente ciò generò un gran furore, dove i difensori della privacy erano contrari a tutto il progetto, mentre le forze di polizia erano decisamente favorevoli. Alla fine, il governo fece marcia indietro e abbandonò l'idea.

Una gran quantità d'informazioni sulla privacy elettronica è disponibile sul sito Web della Electronic Frontier Foundation, [www.eff.org](http://www.eff.org).

### Remailer anonimi

Varie tecnologie, fra cui PGP e SSL, fanno sì che due soggetti arbitrari possano stabilire una connessione sicura, autenticata e assente da interferenze o sorveglianza esterna. In alcuni casi però la privacy è più tutelata facendo in modo che *non* ci sia autenticazione, cioè rendendo anonima la comunicazione. Questo può essere desiderabile per messaggi punto-punto, per i gruppi di discussione (*newsgroup*), o per entrambi.

Consideriamo alcuni esempi. Primo esempio: i dissidenti politici che vivono sotto un regime autoritario desiderano comunicare in modo anonimo, per evitare di essere messi in prigione oppure assassinati. Secondo esempio: gli scandali che sono scoppiati per le azioni di grandi società, istituzioni governative, università o altri enti spesso nascono da una fuga di notizie dall'interno. Chi divulgava queste notizie spesso preferisce rimanere anonimo. Terzo esempio: le persone con opinioni impopolari sulla società, la politica o la religione, possono voler comunicare fra loro tramite e-mail o newsgroup, senza doversi esporre personalmente. Quarto: alcune persone potrebbero voler discutere nei newsgroup di alcolismo, malattie mentali, molestie sessuali, abusi sui minori, oppure problemi delle minoranze perseguitate, senza esporsi pubblicamente. Ovviamente si potrebbero fare molti altri esempi.

Esaminiamo un caso specifico. Nel 1990 alcuni avversari di un gruppo religioso non tradizionale, pubblicarono i loro punti di vista su un newsgroup di USENET utilizzando un **remailer anonimo**. Questo server permetteva agli utenti di creare degli pseudonimi e quindi mandare le e-mail al server, il quale provvedeva a rispedirle, o fare il post sul newsgroup, utilizzando lo pseudonimo creato in precedenza. In questo modo nessuno poteva sapere da chi provenisse veramente il messaggio. Alcuni messaggi nel newgroup rivelavano delle informazioni che il gruppo religioso considerava dei segreti, o comunque documenti coperti da copyright. Il gruppo religioso rispose andando dalle autorità locali a dichiarare che i suoi segreti erano stati svelati e il copyright su alcuni documenti era stato violato: entrambe le azioni erano perseguibili per legge nello stato in cui si trovava il server. Ne seguì un processo, dove l'operatore del server fu obbligato a fornire la mappa con le informazioni necessarie a svelare la vera identità delle persone che avevano inviato i documenti sul newsgroup (per inciso, non è la prima volta che una religione s'infuria perché qualcuno ha divulgato i suoi segreti: William Tyndale fu arso vivo nel 1536 per aver tradotto la bibbia in inglese).

Una gran parte della comunità Internet fu indignata da questa violazione dell'anonymato. Tutti trassero la conclusione che un remailer anonimo che mantiene la mappa fra gli indirizzi e-mail e gli pseudonimi (detto remailer di tipo 1) non vale molto. Questo caso stimolò diverse persone a sviluppare remailer anonimi capaci di resistere agli effetti di un mandato di comparizione del tribunale (attacco giudiziario).

I nuovi remailer, spesso chiamati **remailer cypherpunk**, funzionano così: l'utente produce un messaggio e-mail, completo con l'intestazione RFC 822 (senza il campo *from*; ovviamente), lo cifra con la chiave pubblica del remailer e lo manda al remailer stesso, il quale estrae l'intestazione RFC 822 più esterna, decifra il contenuto e rispedisce il messaggio. Il remailer non mantiene lo storico della sua attività né log di alcun tipo, quindi anche se venisse confiscato non vi si troverebbe nessuna traccia dei messaggi transitati. Diversi utenti che desiderano mantenere l'anonymità fanno passare le loro richieste attraverso una catena di più remailer anonimi, come mostrato nella Figura 8.54. In questo caso, Alice vuole mandare a Bob un messaggio di San Valentino, veramente e sicuramente anonimo, e per fare questo usa tre remailer. Alice scrive il messaggio  $M$ , e aggiunge un'intestazione con l'indirizzo e-mail di Bob. Quindi cifra il tutto con la chiave pubblica del remailer 3,  $E_3$  (indicata dal motivo a strisce orizzontali), e vi prepone un'intestazione con l'indirizzo e-mail del remailer 3 in chiaro. Questo è il messaggio mostrato nella figura fra i remailer 2 e 3.

Alice prosegue cifrando tale messaggio con la chiave pubblica del remailer 2,  $E_2$  (indicata dal motivo a strisce verticali) e aggiungendo un'intestazione in chiaro contenente l'indirizzo e-mail del remailer 2. Questo è il messaggio mostrato fra 1 e 2 nella Figura 8.54. Infine Alice cifra l'intero messaggio con la chiave pubblica  $E_1$  del remailer 1 e aggiunge in chiaro un'intestazione con l'indirizzo e-mail del remailer 1. Questo è il messaggio mostrato nella figura alla destra di Alice, che viene effettivamente trasmesso.

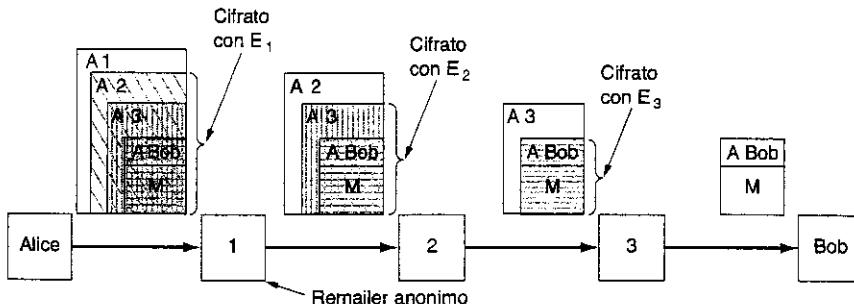


Figura 8.54. Alice usa tre remailer per inviare un messaggio a Bob.

Quando il messaggio arriva al remailer 1, l'intestazione più esterna viene eliminata, il corpo del messaggio decifrato e quindi inviato al remailer 2. Operazioni analoghe avvengono negli altri due remailer.

Rintracciare il mittente dal messaggio finale è estremamente difficile per chiunque. Molti remailer anonimi prendono comunque ulteriori precauzioni; per esempio possono trattenere i messaggi per un tempo di lunghezza casuale, aggiungere o rimuovere della sporcizia alla fine del messaggio oppure riordinare i messaggi. Tutto questo è fatto per rendere più difficile per chiunque il compito di rintracciare la corrispondenza fra i messaggi in uscita e quelli in ingresso, e quindi fare un'analisi del traffico. Per una descrizione di un sistema che rappresenta lo stato dell'arte dei remailer anonimi si rimanda a (Mazières e Kaashoek, 1998).

L'anonymità non è limitata alla e-mail, infatti esistono servizi che permettono anche di navigare il Web in modo anonimo. L'utente configura il suo browser in modo da usare come proxy un server messo a disposizione per rendere anonima la navigazione (battezzato in seguito "anonymizzatore"). In questo modo tutte le richieste passano per l'anonymizzatore, che richiede la pagina e poi la spedisce indietro al richiedente. I siti Web vedono come sorgente l'anonymizzatore invece dell'utente reale. Se l'anonymizzatore non mantiene i log della sua attività, nessuno può identificare chi ha richiesto la pagina.

### 8.10.2 Libertà di parola

La privacy riguarda gli individui e il loro desiderio di limitare le informazioni personali che vogliono rendere accessibili all'esterno. Una seconda questione sociale riguarda la libertà di parola e il suo opposto, la censura. Questa nasce al contrario dal desiderio dei governi di restringere la quantità di informazioni che gli individui possono leggere e pubblicare. Il Web, che contiene milioni e milioni di pagine, è diventato il paradiso del censore. A seconda della natura e dell'ideologia del regime, il materiale proibito può includere siti Web che contengono:

1. materiale inappropriate per bambini o ragazzi
2. odio diretto verso gruppi etnici, religiosi, sessuali, ecc.
3. informazioni sulla democrazia e i valori democratici

4. narrazione di eventi storici che possono contraddirre la versione governativa

5. manuali per aprire le serrature, costruire armi, decifrare messaggi, ecc.

La tipica risposta è quella di bandire i siti cattivi.

Alle volte i risultati sono inaspettati. Per esempio, alcune biblioteche pubbliche hanno installato dei filtri Web sui loro computer per renderli adatti ai ragazzi, bloccando i siti pornografici. I filtri proibiscono l'accesso ai siti inseriti in specifiche *black list* (liste nere) e controllano anche la presenza di parole offensive nelle pagine, prima di mostrarle agli utenti. In un caso accaduto nella contea di Loudoun, in Virginia, il filtro ha bloccato una ricerca legittima di informazioni sul cancro al seno, in quanto il filtro aveva intercettato la parola "seno". L'utente della biblioteca fece causa alla contea di Loudoun. All'estremo opposto, a Livermore in California, un genitore fece causa alla biblioteca pubblica per non aver installato un filtro dopo che il figlio di 12 anni era stato scoperto con del materiale pornografico. Quindi, cosa devono fare le biblioteche?

Molte persone non si rendono pienamente conto che il World Wide Web ha un'estensione mondiale. Non tutti gli stati sono d'accordo nello stabilire cosa è lecito sul Web e cosa no. Per esempio, nel novembre 2000, un tribunale francese ordinò a Yahoo, una corporation californiana, di bloccare l'accesso di tutti gli utenti francesi alle aste di cimeli nazisti che avvenivano sul sito Web di Yahoo, poiché il possesso di tale materiale è in contrasto con le leggi francesi. Yahoo fece ricorso a un tribunale degli Stati Uniti, che le diede ragione. Il problema della legge da applicare in casi come questo è ancora molto lontano da una soluzione finale.

Immaginiamo cosa potrebbe succedere se qualche tribunale negli Utah volesse obbligare la Francia a bloccare siti Web che commerciano vino, perché la loro attività non è conforme alle leggi dello stato degli Utah sul commercio di bevande alcoliche. Supponiamo che la Cina chieda che siano banditi tutti i siti Web che trattano di democrazia, in quanto contrari agli interessi di stato. Le leggi iraniane in termini di religione si possono applicare alla Svezia liberale? L'Arabia Saudita ha il diritto di bloccare i siti Web che trattano dei diritti delle donne? La questione è proprio un vaso di Pandora.

Un commento molto rilevante di John Gilmore a questo proposito è: "la rete interpreta la censura come un guasto e cerca delle rotte per aggirarlo". Per un'interpretazione concreta di questo concetto, consideriamo l'**eternity service** (Anderson, 1996). L'obiettivo è quello di assicurarsi che le informazioni pubblicate non possano essere cancellate o riscritte, come era prassi comune nell'Unione Sovietica durante il governo di Josef Stalin. Nell'**eternity service**, l'utente specifica per quanto tempo vuole che il materiale pubblicato venga preservato, quindi paga una tariffa proporzionale alla durata e alla dimensione del materiale, e infine esegue l'upload. Da quel momento, nessuno potrà rimuovere oppure modificare il materiale, neppure chi ne ha fatto l'upload.

Come si può implementare un servizio di questo tipo? Il modello più semplice consiste nell'usare un sistema peer-to-peer, dove i documenti vengono memorizzati nelle decine di server che fanno parte del sistema. Ogni server riceverà una parte della tariffa pagata, come incentivo per partecipare al sistema. I server dovrebbero essere distribuiti in aree con diverse leggi, per aumentare la robustezza del sistema. Liste contenenti 10 server scelti a caso vengono memorizzate in modo sicuro in diverse locazioni, in questo modo anche se alcune vengono

compromesse ne rimangono delle altre. Un'autorità che voglia distruggere un documento non potrebbe mai avere la sicurezza di aver trovato e distrutto tutte le copie. Il sistema si può rendere autoriparante, nel senso che se alcune copie del documento vengono distrutte, gli altri siti possono cercare nuovi repository dove memorizzare nuove copie per rimpiazzare quelle perse. L'eternity service fu storicamente la prima proposta per un sistema resistente alla censura. Nel frattempo sono stati proposti anche altri sistemi, alcuni dei quali realmente implementati, che svolgono anche nuove funzionalità come la cifratura, l'anonymato e la tolleranza ai guasti. In diversi casi si utilizza la strategia di suddividere i file in tanti frammenti, ognuno dei quali è memorizzato in una varietà di server. Alcuni esempi di sistemi di questo tipo sono Freenet (Clarke et al. 2002), Pasis (Wylie et al., 2000) e Publius (Waldman et al., 2000). Altri lavori di questo tipo si trovano in (Serjantov, 2002).

Un numero crescente di stati sta cercando di stabilire regole per l'esportazione di materiale intangibile, che spesso include siti Web, software, pubblicazioni scientifiche, e-mail, helpdesk telefonici, ecc. Anche la Gran Bretagna, che ha alle spalle una tradizione secolare sulla libertà di parola, sta seriamente pensando di istituire delle leggi restrittive in materia. Questo potrebbe portare, per esempio, alla situazione in cui una discussione tecnica fra un professore inglese e un suo studente straniero della Università di Cambridge sarebbe trattata come esportazione e quindi necessiterebbe dell'autorizzazione governativa (Anderson, 2002). Ovviamente, questo tipo di regolamenti stanno causando molte discussioni.

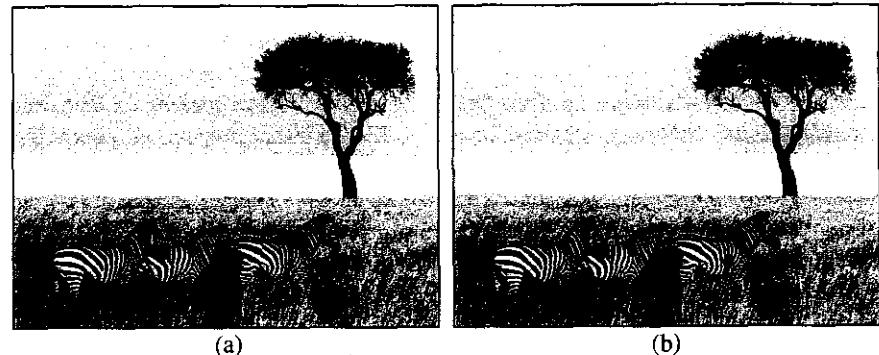
### Steganografia

Negli stati dove abbonda la censura i dissidenti spesso cercano di trovare una scappatoia usando la tecnologia. La crittografia permette di inviare dei messaggi segreti (non necessariamente legali). Se il governo pensa che Alice sia una persona indesiderata, il solo fatto che comuni chi con Bob mette anche lui nella categoria delle persone indesiderate. Questo è il modo in cui i governi repressivi interpretano il concetto di chiusura transitiva, anche senza aver molti matematici alle loro dipendenze. In questo caso di solito vengono in aiuto i remailer anonimi, ma non quando sono banditi a livello nazionale e tutti i messaggi per l'estero richiedono una licenza governativa per l'esportazione. Il Web può risolvere questo problema.

Le persone che vogliono comunicare in modo segreto spesso cercano di nascondere il fatto stesso che ci sia una qualunque comunicazione fra di loro. La scienza che si occupa di nascondere i messaggi è chiamata **steganografia**, dalle parole greche che significano "scrittura nascosta": questa scienza infatti era nota già agli antichi greci. Erodoto scrisse di un generale che fece rasare la testa di un messaggero, tatuò il messaggio sul suo cranio e poi attese la ricrescita dei capelli prima di inviare il messaggero. Le tecniche moderne usano lo stesso concetto, solo con una banda passante molto più alta e con una minore latenza.

Come esempio consideriamo la Figura 8.55(a). Questa fotografia, scattata dall'autore in Kenya, ritrae tre zebre che guardano un albero di acacia. La Figura 8.55(b) sembra rappresentare le stesse tre zebre e l'albero di acacia, ma in realtà contiene anche degli extra. Infatti, contiene al suo interno il testo completo di cinque opere di Shakespeare: *Amleto*, *Macbeth*, *Il mercante di Venezia* e *Giulio Cesare*. In totale queste opere occupano 700 KB di testo.

Come funziona questo canale steganografico? L'immagine originale a colori ha una risoluzione di  $1.024 \times 768$  pixel. Ogni pixel consiste di tre numeri di 8 bit, che determinano



**Figura 8.55.** (a) Tre zebre e un albero. (b) Tre zebre, un albero e il testo integrale di cinque opere di William Shakespeare.

rispettivamente l'intensità del rosso, verde e blu. Il colore di ciascun pixel è dato dalla sovrapposizione lineare dei tre colori. Il metodo di codifica steganografico usa, come canale di comunicazione nascosto, i bit di ordine più basso di ogni colore RGB.

In questo modo ogni pixel può far posto a 3 bit di informazione segreta, uno nel rosso, uno nel blu e uno nel verde. Con un'immagine come quella della Figura 8.55 possiamo nascondere fino a  $1.024 \times 768 \times 3$  bit d'informazione segreta, cioè 294.912 byte.

Il testo completo delle cinque opere di Shakespeare con un breve messaggio introduttivo ammonta a 734.891 byte, ma il testo compresso con un algoritmo standard ha raggiunto la dimensione di circa 274 KB. L'output compresso è stato poi cifrato con IDEA e infine inserito nei bit di ordine più basso di ogni colore. Come si può vedere (o meglio, come non si vede), l'esistenza dell'informazione è completamente invisibile. Anche nella versione originale a pieni colori, non si riesce assolutamente a distinguere l'aggiunta d'informazione steganografica. L'occhio umano non riesce a distinguere facilmente fra un colore a 21 bit e uno a 24 bit. Le due immagini in bianco e nero a bassa risoluzione non rendono giustizia alla potenza di questo metodo.

L'autore ha preparato una dimostrazione che include la versione a colori e alta risoluzione dell'immagine della Figura 8.55(b) con le cinque opere di Shakespeare inserite all'interno, per dare un'idea migliore di come funziona la steganografia. La dimostrazione, assieme agli strumenti per inserire ed estrarre testo dalle immagini, può essere scaricata dal sito Web di questo libro. Alcuni dissidenti potrebbero usare questo metodo per comunicare in modo segreto: si comincia creando un sito Web pieno d'immagini approvate dal regime, per esempio foto del grande condottiero, di sport locale, film, star della televisione, ecc. All'interno di queste immagini si possono inserire dei messaggi con il metodo steganografico. Se i messaggi vengono prima compressi e poi cifrati, anche chi ne sospettasse la presenza avrebbe molte difficoltà nel distinguerli dal semplice rumore bianco. Ovviamente le immagini devono essere degli originali. Se si prendono delle immagini da Internet per poi alterarne dei bit si ottiene una ben misera copertura.

Le immagini non sono il solo mezzo per trasportare messaggi steganografici; anche i file

usicali funzionano bene, e i file video forniscono un'enorme banda steganografica. Anche la disposizione e l'ordine dei tag nei file HTML possono essere usati per trasportare informazione.

abbiamo esaminato la steganografia nel contesto della libertà di parola, ma ne esistono numerosi altri usi. Uno di questi consiste nel codificare messaggi segreti all'interno delle immagini per affermarne la proprietà. Se un'immagine viene rubata e messa su un sito Web, il legittimo proprietario può portare in tribunale il contenuto del messaggio steganografico e quindi dimostrare l'effettivo possesso dell'immagine. Questa tecnica è detta **watermarking** ed è discussa in (Piva et al. 2002).

Per ulteriori discussioni sulla steganografia si veda (Artz, 2001; Johnson e Jajoda, 1998; Atzenbeisser e Petitcolas, 2000; Wayner, 2002).

### 10.3 Copyright

La privacy e la censura sono solamente due delle aree in cui la tecnologia incontra le leggi. Una terza area è il copyright. Il **copyright** garantisce ai creatori di IP (*Intellectual property*, proprietà intellettuale) come scrittori, artisti, compositori, musicisti, fotografi, coreografi il diritto di sfruttare le proprie IP per un certo periodo di tempo; tipicamente è la durata della vita dell'autore più 50 anni, che diventano 75 nel caso di proprietà da parte di aziende. Alla scadenza del copyright, le opere diventano di dominio pubblico e chiunque le può utilizzare o vendere a piacimento. Per esempio il progetto Gutenberg ([www.promo.net/pg](http://www.promo.net/pg)) ha pubblicato sul Web migliaia di opere che sono di pubblico dominio (Shakespeare, Twain, Dickens, ecc.). Nel 1998 il congresso degli USA ha esteso il copyright di altri 20 anni su richiesta di Hollywood, che lamentava il fatto che senza un'estensione nessuno avrebbe creato più niente. Per contrasto, notiamo che i brevetti durano solo 20 anni e la gente continua a produrre nuove invenzioni.

Il problema del copyright è venuto alla ribalta quando Napster, un servizio per lo scambio di musica, aveva 50 milioni di membri. Anche se Napster non realizzava nessuna copia di musica, le autorità giudiziarie ritenevano che il fatto che Napster gestisse un database centrale con le informazioni su chi aveva quale canzone, costituisse di per sé un sostanziale contributo alla violazione della legge sul copyright. In definitiva, i giudici hanno stabilito che Napster aiutava altre persone a infrangere la legge. Anche se nessuno afferma seriamente che il copyright è sbagliato, molti pensano che la durata sia troppo lunga e favorisca le grandi corporazioni a scapito del pubblico. Al contempo sta emergendo una nuova generazione di servizi di scambio musicale, che fanno sorgere diverse questioni etiche. Per esempio, consideriamo una rete peer-to-peer dove gli utenti possono condividere file illegali (musica di pubblico dominio, video privati, trattati religiosi che non siano segreti, ecc.), e magari anche alcuni file che sono coperti da copyright. Supponiamo che tutti gli utenti siano connessi costantemente, per esempio tramite ADSL o cavo. Ogni computer ha un indice dei contenuti del suo disco fisso e una lista dei membri del servizio. Se qualcuno desidera ricercare un file specifico può scegliere un utente a caso per vedere se ha quel file. In caso negativo, la ricerca può procedere controllando tutti i membri elencati nella lista posseduta da quell'utente. I computer sono perfetti per questo tipo di lavoro. Dopo aver trovato il file, il richiedente deve solamente copiarlo sul suo disco fisso.

Se il file è coperto da copyright, è probabile che il richiedente stia infrangendo la legge (anche se per trasferimenti internazionali non è chiaro di quale legge stiamo parlando). Cosa possiamo dire della sorgente? È forse un crimine tenere sul proprio disco fisso della musica che è stata regolarmente acquistata, ma dove anche altre persone possono trovarla? Pensiamo alla situazione in cui una persona che abbia una casetta in campagna, senza serrature alle porte, subisca il furto da parte di un ladro di IP, che penetra in casa portandosi un laptop e uno scanner, copia l'intero contenuto di un libro coperto da copyright e poi scappa. Possiamo ritenere *il proprietario della casa* colpevole del crimine di non aver protetto il copyright del libro? Ci sono anche altri problemi sul fronte del copyright. È in corso un'enorme battaglia fra Hollywood e l'industria dei computer. I primi vogliono una protezione più forte sulla proprietà intellettuale, mentre i secondi non vogliono fare i poliziotti per conto di Hollywood. Nell'ottobre 1998, il congresso USA varò il **DMCA** (*Digital Millennium Copyright Act*), che definisce atto criminale ogni azione volta ad aggirare i meccanismi di protezione presenti sulle opere coperte da copyright, e la divulgazione delle informazioni utili per violare tali protezioni. Legislazioni simili sono state varate anche nell'Unione Europea. Anche se nessuno pensa che i pirati dell'Estremo Oriente debbano essere autorizzati a duplicare il materiale coperto da copyright, molti pensano però che la DMCA sposti troppo l'ago della bilancia dalla parte degli interessi dei proprietari di copyright, contro l'interesse del pubblico.

Un caso emblematico si è realizzato nel settembre del 2000. In quella occasione un consorzio industriale, che aveva il compito di costruire un sistema inattaccabile per vendere musica on-line, decise di sponsorizzare un concorso in cui invitava il pubblico a cercare di forzare il sistema (che è esattamente quello che si deve fare quando si vuole varare un nuovo sistema di sicurezza). Un gruppo di ricercatori nell'ambito della sicurezza provenienti da diverse università e guidato dal Prof. Edward Felten di Princeton riuscì effettivamente a forzare il sistema. Quindi scrissero un articolo sulla loro scoperta, e lo mandarono alla conferenza USENIX sulla sicurezza, dove fu oggetto di peer review e infine accettato. Prima che potesse presentare l'articolo, Felten ricevette una lettera dalla Recording Industry Association of America, in cui l'autore veniva diffidato dal pubblicare l'articolo, altrimenti sarebbe stato denunciato sotto i termini previsti dalla DMCA.

La risposta di Felten fu quella di portare il caso davanti a una corte federale, per avere un giudizio sul fatto che pubblicare degli articoli scientifici fosse ancora un'attività legale. L'industria discografica, temendo che la corte emanasse una sentenza non favorevole, ritirò le accuse e quindi il caso fu chiuso. Senza dubbio questa scelta fu dettata dalla debolezza delle motivazioni che l'industria discografica aveva a suo favore: aveva lanciato un invito a forzare i suoi sistemi e poi aveva minacciato di fare causa a chi aveva accettato la sfida. Dopo che le accuse furono ritirate, l'articolo venne pubblicato (Craver et al., 2001). È certo comunque che ci saranno nuove occasioni di scontro.

Un'altra questione collegata è la **dottrina dell'uso legittimo**, che è stata sancita da sentenze emesse in vari stati. Questa dottrina afferma che chi acquista un'opera coperta da copyright ha il diritto di farne delle copie, con alcuni limiti. Per esempio è possibile citare alcune parti dell'opera per scopi scientifici, usarla come materiale didattico nelle scuole e nelle università, e in alcuni casi è consentito fare copie di backup per uso personale,

da utilizzarsi nel caso in cui il supporto originale non sia più utilizzabile. Per stabilire se una copia entra nella dottrina dell'uso legittimo bisogna anche considerare se: (1) il suo uso è commerciale, (2) quale percentuale dell'opera è stata copiata, (3) l'effetto che la copia ha sulle vendite dell'opera originale. Si dà il caso, però, che DMCA e leggi analoghe approvate in vari stati dell'Unione Europea vietino ogni mezzo atto a eludere gli schemi di protezione, quindi queste leggi proibiscono anche le operazioni legali che vanno sotto la dottrina dell'uso legittimo. A tutti gli effetti, DMCA toglie agli utenti alcuni diritti che avevano storicamente conquistato, e dà quindi più potere a chi vende contenuti. Un confronto netto fra le due opposte posizioni sarà pertanto inevitabile.

Un altro sviluppo che porta su posizioni ancora più estreme lo spostamento di potere da utenti a proprietari di copyright è l'iniziativa **TCPA** (*Trusted Computing Platform Alliance*), guida da Intel e Microsoft. L'idea è quella di far sì che la CPU e il sistema operativo controllino il comportamento dell'utente in modo accurato e attento (per esempio quando l'utente vuole ascoltare della musica pirata) e quindi proibiscano i comportamenti indesiderati. Il sistema permette ai proprietari di contenuti IP di manipolare i PC da remoto per cambiare le regole, nel caso lo ritengano necessario. Inutile a dirlo, le conseguenze di questo schema sono immense. È bello che l'industria finalmente si occupi di sicurezza, ma va biasimato il fatto che tutti gli sforzi siano concentrati nel far rispettare il copyright, piuttosto che affrontare i problemi dei virus, cracker, intrusioni e altre questioni che preoccupano gli utenti dei sistemi informatici.

Nei prossimi mesi legislatori e avvocati saranno decisamente occupati per cercare di bilanciare gli interessi dei possessori di copyright e quelli del pubblico. Il cyberspazio non è diverso dal mondo reale. Anche il cyberspazio crea contrasti fra diversi gruppi di persone, con il risultato di avere lotte per il potere, cause legali e (si spera) alcune soluzioni definitive, almeno fino a quando non arriverà una nuova tecnologia dagli effetti dirompenti.

## 8.11 Sommario

La crittografia è uno strumento che può essere usato per mantenere e assicurare la confidenzialità, l'integrità e l'autenticità delle informazioni. Tutti i sistemi critografici moderni sono basati sul principio di Kerckhoff, che consiste nell'utilizzare un algoritmo noto pubblicamente con una chiave segreta. Molti algoritmi critografici usano delle trasformazioni complesse che contengono sostituzioni e permutazioni usate per trasformare il testo in chiaro in quello cifrato. Tuttavia, se la crittografia quantistica si dimostrerà utilizzabile nella pratica, l'utilizzo di blocchi monouso darà origine a sistemi critografici realmente imbattibili.

Gli algoritmi critografici possono essere divisi in algoritmi a chiave simmetrica e algoritmi a chiave pubblica. Gli algoritmi a chiave simmetrica funzionano mescolando i bit del testo in chiaro con una serie di cicli parametrati dalla chiave; il risultato è il testo cifrato. Gli algoritmi a chiave simmetrica più diffusi oggi sono DES triplo e Rijndael (AES). Questi algoritmi si possono usare con modalità diverse, fra cui: electronic code book, cipher block, chaining, stream cipher, contatore e altri.

Gli algoritmi a chiave pubblica hanno la proprietà di utilizzare delle chiavi diverse per la cifratura e la decifrazione, inoltre la chiave di decifrazione non può essere ricavata da quella di

cifratura. Queste proprietà rendono possibile la pubblicazione della chiave di cifratura. Il principale algoritmo a chiave pubblica è RSA, che deriva la sua forza dalla grande difficoltà computazionale associata al problema della scomposizione in fattori dei grandi numeri.

I documenti commerciali, legali e altri, necessitano di un qualche tipo di firma. Per questo motivo sono stati sviluppati schemi per le firme digitali, che usano sia gli algoritmi a chiave simmetrica sia quelli a chiave asimmetrica. Di norma, i messaggi da firmare vengono prima passati attraverso un algoritmo di hashing, come MD5 o SHA-1, e poi viene firmato l'hash invece del messaggio originale.

La gestione delle chiavi pubbliche può essere fatta tramite l'uso dei certificati, che sono documenti che legano i protagonisti alle loro chiavi crittografiche. I certificati sono firmati da autorità fidate o da qualcuno che è stato certificato da un'autorità fidata. Perché il sistema funzioni, la root della catena deve essere nota in anticipo, infatti normalmente i browser vengono rilasciati con i certificati di diverse root già inseriti.

Questi strumenti crittografici si possono impiegare per rendere sicuro il traffico di rete. IPsec lavora nello strato network, cifrando il flusso dei pacchetti da un host all'altro. I firewall sono usati per controllare il traffico che entra ed esce da una rete aziendale, tramite regole sul protocollo e verifica delle porte utilizzate. Le virtual private network possono simulare il funzionamento delle vecchie linee dedicate, e fornire quindi certe proprietà di sicurezza desiderabili. Infine, le reti wireless necessitano di un buon grado di sicurezza, ma 802.11 e WEP non sono in grado di fornirlo. La situazione dovrebbe migliorare considerevolmente con 802.11i. Due soggetti che vogliono stabilire una sessione per comunicare in modo sicuro, devono avere un metodo per autenticarsi reciprocamente e stabilire una chiave di sessione condivisa in caso di necessità. Esistono svariati protocolli di autenticazione, che utilizzano metodi differenti come una terza parte fidata, Diffie-Hellman, Kerberos e la crittografia a chiave pubblica.

Si può ottenere la sicurezza della e-mail usando una combinazione delle tecniche che abbiamo studiato in questo capitolo. PGP, per esempio, comprime i messaggi, poi li cifra usando IDEA, quindi invia la chiave IDEA cifrandola con la chiave pubblica del ricevente. Inoltre, PGP calcola e trasmette anche l'hash del messaggio ottenendo così una verifica dell'integrità. La sicurezza del Web è un altro argomento importante, che inizia con la sicurezza della risoluzione dei nomi. DNSsec fornisce un metodo per prevenire gli attacchi di DNS spoofing, un altro metodo per risolvere questo problema è l'uso di nomi auto certificanti. La maggior parte dei siti Web di e-commerce utilizza SSL per stabilire sessioni sicure e autenticate fra il client e il server. Per gestire il codice mobile si adottano più tecniche, in particolare sandbox e code signing.

Internet solleva molte questioni dove la tecnologia si trova a interagire con la politica e le leggi. Alcune di queste aree includono la privacy, la libertà di parola e il copyright.

## Problemi

- Forzare il cifrario monoalfabetico riportato qui sotto. Il testo in chiaro, che consiste di sole lettere, è un pezzo di una nota poesia di Lewis Carroll (in lingua inglese).
 

kfd ktbd fzmm eubd kfd pzyiom mztx ku kzg ur bzha kfthcm  
ur mfntm zhx mfudm zhx mdzythc pzq ur ezsszcdm zhx gthcm  
zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthcm

sok pztk z stk kfd uamkdum eitdx sdruid pd fzld uoi efzk  
 rui mubd ur om zid uok ur sidzkf zhx zyy ur om zid rzk  
 hu foia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

2. Decifrare il seguente cifrario a trasposizione di colonna. Il testo in chiaro, in lingua inglese, è preso da un noto libro di testo d'informatica, quindi "computer" è una parola che probabilmente vi appare. Il testo in chiaro consiste di sole lettere (senza spazi). Il testo cifrato è stato suddiviso in blocchi di cinque caratteri per aumentarne la leggibilità.

aauan cvlre runn dltme aeepb ytust iceat npmey iicgo gorch srsoc  
 nntii imiha oofpa gsivt tpsit lboir otoex

3. Trovare il blocco monouso di 77 bit che genera il testo in chiaro "Donald Duck" a partire dal testo cifrato della Figura 8.4.
4. La crittografia quantistica ha bisogno di un cannone spara-fotoni che possa sparare, a comando, un singolo fotone che trasporta 1 bit di informazione. Calcolare quanti fotoni sono trasportati da 1 bit in una linea in fibra a 100 Gbps. Assumere che la lunghezza del fotone sia pari alla sua lunghezza d'onda, che prenderemo pari a 1 micron. La velocità della luce nella fibra è di 20 cm/nsec. (poiché mancano dati sull'intensità, il risultato è chiaramente solo un limite inferiore).
5. Se Trudy raccoglie e rigenera fotoni mentre è in corso una comunicazione con crittografia quantistica, il risultato sarà che i valori raccolti saranno parzialmente errati e che Bob avrà degli errori nel blocco monouso che riceve. In media, che frazione dei bit del blocco monouso di Bob sarà in errore?
6. Un principio crittografico fondamentale afferma che tutti i messaggi devono avere delle ridondanze. Sappiamo però che la ridondanza aiuta gli intrusi a scoprire se la chiave che hanno indovinato è corretta. Consideriamo due forme di ridondanza. La prima consiste nell'avere una sequenza di bit nota nei primi  $n$  bit del testo in chiaro. La seconda consiste nell'aggiungere il codice hash del messaggio negli ultimi  $n$  bit. Da un punto di vista della sicurezza questi due approcci sono equivalenti? Argomentare la risposta.
7. Nella Figura 8.6 le P-box e le S-box si alternano. Anche se questa disposizione è esteticamente attraente, risulta in qualche modo più sicura rispetto ad avere prima tutte le P-box e poi tutte le S-box?
8. Progettare un attacco al DES, basato sulla conoscenza che il testo in chiaro consiste esclusivamente di caratteri ASCII maiuscoli oltre a spazio, virgola, punto, punto e virgola, carriage return e line feed. Non sono note informazioni sulla parità del testo in chiaro.
9. Nel testo del capitolo abbiamo calcolato che una macchina dotata di un miliardo di processori capaci di elaborare una chiave ogni picosecondo, impiegherebbe  $10^{10}$  anni a forzare AES a 128 bit. Allo stato attuale i computer possono arrivare a 1.024 processori e impiegano 1 millisecondo per analizzare una chiave. Per avere delle prestazioni analoghe alla macchina di cui sopra è quindi necessario aumentare la velocità di un fattore  $10^{15}$ . Supponendo che la legge di Moore continui a valere (cioè che la potenza di calcolo raddoppi ogni 18 mesi), quanti anni sarebbero necessari anche solo per costruire una macchina con le prestazioni indicate?
10. AES supporta chiavi di lunghezza pari a 256 bit. Quante sono le possibili chiavi di AES-256? Trovare dei riferimenti in fisica, chimica o astronomia, su numeri di pari grandezza. Internet può essere utile in questo tipo di ricerca. Trarre delle conclusioni da questa ricerca.

11. Prendiamo un messaggio cifrato con DES in modalità block chaining. Supponiamo che un bit del testo cifrato nel blocco  $C_i$  venga accidentalmente trasformato da 0 a 1 durante la trasmissione. Quanto testo in chiaro sarà illeggibile a causa di questo errore?
12. Prendiamo di nuovo un messaggio cifrato in modalità block chaining. Questa volta, al posto di avere un singolo bit a 0 trasformato in 1 per errore, abbiamo il caso in cui un bit a 0 viene aggiunto al flusso di dati cifrati dopo il blocco  $C_i$ . Quanto testo in chiaro sarà illeggibile a seguito di questo errore?
13. Confrontare le modalità cipher block chaining e cipher feedback, per quanto riguarda il numero di operazioni necessarie per trasmettere un file di grandi dimensioni. Quale modalità è più efficiente e di quanto?
14. Consideriamo un sistema crittografico RSA a chiave pubblica, con  $a = 1$ ,  $b = 2$ , ecc.
  - (a) Elencare i primi 5 possibili valori di  $d$ , se  $p = 7$  e  $q = 11$ .
  - (b) Se  $p = 13$ ,  $q = 31$  e  $d = 7$ , trovare  $e$ .
  - (c) Dati  $p = 5$ ,  $q = 11$  e  $d = 27$ , trovare  $e$  e cifrare la stringa "abcdefgij".
15. Supponiamo che un utente, Maria, scopra che la sua chiave privata RSA ( $d1, n1$ ) è uguale alla chiave pubblica di un altro utente ( $e2, n2$ ). In altre parole  $d1 = e2$  e  $n1 = n2$ . Maria dovrebbe cambiare la sua chiave? Argomentare la risposta.
16. Consideriamo la modalità counter, come mostrata nella Figura 8.15 ma con  $IV = 0$ . In generale, l'uso del valore 0 per  $IV$  ha un impatto negativo sulla sicurezza?
17. Il protocollo di firma della Figura 8.18 ha la seguente debolezza: se Bob ha un crash di sistema, può perdere il contenuto della RAM. Che conseguenze può avere questa perdita e che cosa si può fare per prevenirle?
18. Nella Figura 8.20 è illustrato il modo in cui Alice invia un messaggio firmato a Bob. Se Trudy rimpiazza  $P$ , Bob riesce ad accorgersene. Che cosa succede, invece, se Bob rimpiazza sia  $P$  sia la firma?
19. Le firme digitali hanno una potenziale debolezza causata dagli utenti pigri. In una transazione di e-commerce, si può verificare la situazione in cui un contratto viene stilato e l'utente viene chiamato a firmare il relativo codice hash SHA-1. Se l'utente non verifica l'effettiva corrispondenza tra contratto e hash, potrebbe inavvertitamente firmare il contratto sbagliato. Supponiamo che la mafia voglia sfruttare questa debolezza per guadagnare un po' di soldi. Per fare questo installa un sito Web a pagamento (per esempio un sito pornografico, di scommesse, ecc) e chiede a ogni nuovo cliente un numero di carta di credito. A questo punto invia ai clienti un contratto dove si afferma che il cliente vuole effettivamente usare i servizi del sito e che pagherà con la carta di credito. Ai clienti è richiesta la firma dell'hash del contratto, ben sapendo che la maggior parte delle persone firmerà senza controllare che il contratto e l'hash coincidano. Mostrare come, a questo punto, la mafia può comprare dei diamanti da un gioielliere che opera legittimamente su Internet e quindi farli pagare agli ignari clienti.
20. Una classe di matematica ha 20 studenti. Qual è la probabilità che almeno due studenti abbiano lo stesso compleanno? Assumere che nessuno sia nato il 29 febbraio e che ci siano quindi solo 365 possibili compleannni.

1. Dopo che Ellen ha confessato a Marilyn di averla ingannata riguardo alla questione della cattedra di Tom, Marilyn decise di risolvere questo tipo di problemi per il futuro usando il metodo di dettare le lettere in un registratore e quindi facendole trascrivere alla sua nuova segretaria. Marilyn decise anche di esaminare i messaggi a terminale dopo la trascrizione, per essere sicura che contengano le sue esatte parole. La nuova segretaria potrebbe ancora usare l'attacco del compleanno per falsificare i messaggi? Se sì, come può fare? Suggerimento: è possibile.
2. Nella Figura 8.23 consideriamo il tentativo fallito di Alice, che voleva acquisire la chiave pubblica di Bob. Supponiamo che Bob e Alice condividano una chiave segreta, ma Alice continua a volere la chiave pubblica di Bob. Riesce adesso ad ottenerla? Se sì, come?
3. Alice vuole comunicare con Bob usando la crittografia a chiave pubblica. Apre quindi una connessione con qualcuno, che spera sia Bob. Alice richiede a Bob di inviarle la sua chiave pubblica, Bob risponde inviandogliela in chiaro insieme a un certificato X.509 firmato dalla root CA. Alice conosce già la chiave pubblica della root CA. Che passi deve compiere Alice per verificare che sta parlando con Bob? Supponere che a Bob non importi di sapere con chi sta parlando (per esempio, Bob fornisce un qualche servizio pubblico).
4. Supponere che un sistema usi una PKI basata su una gerarchia di CA strutturate ad albero, e che Alice voglia comunicare con Bob. Dopo aver stabilito un canale di comunicazione con Bob, riceve da lui un certificato firmato da una CA  $X$ . Supponere che Alice non abbia mai sentito parlare di  $X$ . Che passi deve compiere Alice per verificare se sta veramente parlando con Bob?
5. Si può usare IPsec con AH in modalità trasporto se una delle macchine è soggetta a NAT? Spiegare la risposta.
6. Spiegare il vantaggio di usare HMAC al posto di RSA per firmare gli hash SHA-1.
7. Dare una ragione per cui può essere conveniente configurare un firewall per ispezionare il traffico in arrivo. Dare una ragione per cui può essere conveniente configurarlo per ispezionare il traffico in uscita. Pensate che il controllo del traffico abbia una buona probabilità di essere efficace?
8. Il formato del pacchetto WEP è mostrato nella Figura 8.31. Supponiamo che il checksum sia di 32 bit, calcolato eseguendo lo XOR di tutte le word di 32 bit del campo payload. Supponiamo anche che i problemi di RC4 siano corretti usando uno stream cipher che non ha debolezze e che le IV siano estensibili fino a 128 bit. È possibile che un intruso riesca a spiare o interferire con il traffico dei dati senza essere scoperto?
9. Supponiamo che un'organizzazione usi le VPN per connettere i suoi siti in modo sicuro attraverso Internet. In questa organizzazione è necessario che un utente, Jim, usi la crittografia o un qualche altro meccanismo di sicurezza per comunicare con l'utente Mary nella stessa organizzazione?
10. Modificare leggermente un messaggio nel protocollo della Figura 8.34 per renderlo resistente ad attacchi di tipo riflessione. Spiegare perché il cambiamento funziona.
11. Lo scambio di chiavi Diffie-Hellman viene usato per stabilire una chiave segreta fra Alice e Bob. Alice manda a Bob (719, 3, 191). Bob risponde con (543). Il numero segreto di Alice è  $x = 16$ . Qual è la chiave segreta?
12. Alice e Bob non si sono mai incontrati, non hanno segreti in comune e nessun certificato. A ogni modo stabiliscono una chiave segreta comune usando l'algoritmo di Diffie-Hellman. Spiegare perché è molto difficile per loro difendersi da un attacco del tipo uomo nel mezzo.

33. Nel protocollo della Figura 8.39, perché  $A$  viene inviato in chiaro insieme alla chiave di sessione cifrata?
34. Nel protocollo della Figura 8.39, abbiamo detto che il fatto di cominciare ogni messaggio in chiaro con 32 bit a 0 è un rischio di sicurezza. Supponiamo che ciascun messaggio inizi con un numero casuale dipendente dall'utente, a tutti gli effetti una seconda chiave segreta nota solo al suo utente e alla KDC. Questo elimina il rischio di un attacco per testo in chiaro noto? Perché?
35. Nel protocollo Needham-Schroeder, Alice genera due challenge  $R_A$  e  $R_{A2}$ . Questo sembrerebbe una ridondanza. Non potrebbe bastarne una sola?
36. Consideriamo il caso di un'organizzazione che usa Kerberos per l'autenticazione. Se AS oppure TGS deventano indisponibili, quali sono gli effetti in termini di sicurezza e disponibilità del servizio?
37. Nel protocollo di autenticazione a chiave privata della Figura 8.43, nel messaggio 7,  $R_B$  viene cifrato con  $K_S$ . È necessario cifrare questo messaggio, oppure sarebbe bastato inviarlo in chiaro? Motivare la risposta.
38. I terminali POS (bancomat) che usano le carte a banda magnetica e i codici PIN hanno un difetto fatale: un venditore disonesto potrebbe modificare il suo lettore di carte in modo da intercettare e memorizzare tutte le informazioni relative alla carta e al codice PIN. In questo modo potrebbe successivamente inviare transazioni falsificate. La prossima generazione di terminali POS utilizzerà carte complete di CPU, tastiera e un piccolo display. Studiare un possibile protocollo per questo sistema, che sia a prova di truffa da parte del venditore.
39. Fornire due motivazioni del perché PGP comprime i messaggi.
40. Assumiamo che tutti gli utenti di Internet usino PGP. Sarebbe possibile inviare un messaggio a un indirizzo arbitrario su Internet in modo che venga sempre decodificato correttamente? Discutere la risposta.
41. L'attacco mostrato nella Figura 8.47 tralascia un passaggio, non necessario per far funzionare lo spoofing, ma se viene incluso riduce la possibilità di lasciare dei sospetti sull'avvenuto attacco. Qual è questo passaggio mancante?
42. Un metodo che è stato proposto per bloccare lo spoofing del DNS fatto tramite il calcolo dell'ID, consiste nel generare ID in modo casuale invece che con un contatore. Discutere gli aspetti di sicurezza di questo approccio.
43. Il protocollo di trasporto dei dati SSL utilizza due nonce e una chiave premaster. Esiste un valore aggiunto nell'usare i nonce? Se sì, quale?
44. L'immagine della Figura 8.55(b) contiene il testo ASCII di cinque opere di Shakespeare. Sarebbe possibile nascondere della musica nell'immagine, al posto del testo? Se sì, come si potrebbe fare e quanta musica si potrebbe celare nell'immagine? Se no, perché?
45. Alice faceva un uso molto frequente dei remailer anonimi di tipo 1. Alice era solita mandare diversi messaggi al suo newsgroup preferito, *alt.fanclub.alice*, usando uno pseudonimo: in questo modo tutti sapevano che i messaggi provenivano effettivamente da Alice. Assumendo che il remailer funzionasse correttamente, Trudy non aveva modo di inviare messaggi fingendo di essere Alice. Dopo che i remailer di tipo 1 sono stati chiusi, Alice ha cominciato ad usare i remailer cypherpunk e ha cominciato un nuovo thread nel suo newsgroup. Escogitare un metodo che Alice può usare per evitare che Trudy invii dei messaggi nel newsgroup impersonando Alice.

46. Cercare su Internet un caso interessante riguardante la privacy, e scrivere un riassunto di circa una pagina.
47. Cercare su Internet alcuni casi giudiziari riguardanti il copyright e l'uso legittimo, e scrivere un riassunto di circa una pagina.
48. Scrivete un programma che esegua la cifratura dell'input con un'operazione di XOR fra l'input in chiaro e un keystream. Trovate o scrivete un generatore di numeri casuali che potete usare per generare il keystream. Il programma dovrebbe funzionare come un filtro che prende il testo in chiaro in input e produce il testo cifrato in output (e viceversa). Il programma deve avere come parametro il numero da usare come seme per il generatore di numeri casuali.
49. Scrivete una procedura che calcola l'hash SHA-1 di un blocco di dati. La procedura deve avere 2 parametri in ingresso: un puntatore al buffer di input e un puntatore al buffer di output lungo 20 byte. Per conoscere le esatte specifiche di SHA-1, cercate su internet FIPS 180-1, che è la specifica completa.

# 9

## Elenco di lettura e bibliografia

Abbiamo terminato il nostro studio sulle reti di computer, ma questo è solo l'inizio. Molti argomenti interessanti non sono stati trattati in dettaglio come meritavano, mentre altri sono stati del tutto omessi per mancanza di spazio. In questo capitolo forniamo alcuni suggerimenti su ulteriori letture e una bibliografia per i lettori che desiderano continuare il loro studio delle reti di computer. I titoli dei libri verranno indicati in corsivo, mentre le pubblicazioni tra virgolette.

### 9.1 Suggerimenti per ulteriori letture

Esiste una letteratura esaurente su tutti gli aspetti delle reti di computer. Tre periodici che pubblicano spesso documenti su questi argomenti sono *IEEE Transactions on Communications*, *IEEE Journal on Selected Areas in Communications* e *Computer Communication Review*. Molti altri periodici pubblicano occasionalmente documenti sull'argomento.

IEEE pubblica anche tre riviste – *IEEE Internet Computing*, *IEEE Network Magazine* e *IEEE Communications Magazine* – che contengono sondaggi, tutorial e studi sulle reti. I primi due enfatizzano l'architettura, gli standard e il software, mentre l'ultimo è rivolto alle tecnologie di comunicazione (fibra ottica, satelliti e così via).

Inoltre si tengono numerose conferenze annuali o biennali che generano numerosi documenti su reti e sistemi distribuiti, in particolare *SIGCOMM Annual Conference*, *The*

International Conference on Distributed Computer Systems e The Symposium on Operating Systems Principles.

seguito sono elencati alcuni suggerimenti per letture supplementari, legate ai capitoli di questo libro. La maggior parte sono tutorial o sondaggi sull'argomento. Altri sono capitoli di testo.

### 1.1 Introduzione e opere generiche

et al., "Wireless Mobile Communications at the Start of the 21<sup>st</sup> Century"

nuovo secolo, una nuova tecnologia: l'idea è buona. Dopo la storia del wireless, i principali argomenti trattati comprendono standard, applicazioni, Internet e tecnologie.

mer, *The Internet Book*

inunque cerchi una rapida introduzione a Internet dovrebbe consultare questo libro. Il libro descrive la storia, la crescita, la tecnologia, i protocolli e i servizi offerti da Internet in termini comprensibili anche dai principianti; tuttavia, gran parte del materiale trattato nel libro interessa anche i lettori più tecnici.

rber, "Will 3G Really Be the Next Big Wireless Technology?"

telefoni cellulari di terza generazione dovrebbero combinare voce e dati, fornendo velocità dei dati fino a 2 Mbps, ma la loro diffusione procede un po' a rilento. Le promesse, i problemi, la tecnologia, i criteri e i risvolti economici dell'utilizzo della comunicazione wireless a banda larga sono esaminati in questo articolo di facile lettura.

IEEE Internet Computing, gennaio/febbraio 2000

Primo numero di *IEEE Internet Computing* nel nuovo millennio ha fatto proprio ciò che gli autori si potrebbero aspettare: chiedere alle persone che hanno aiutato a creare Internet nel millennio precedente di immaginare dove potrebbero portarci nel nuovo millennio. Gli autori sono Paul Baran, Lawrence Roberts, Leonard Kleinrock, Stephen Crocker, Danny Cohen, Bob Metcalfe, Bill Gates, Bill Joy et al.. Per i migliori risultati, aspettate 500 anni o leggete le loro previsioni.

Onnis, "Beating the System: Abuses of the Standards Adoption Process"

I comitati per gli standard cercano di essere imparziali nel loro lavoro, ma sfortunatamente i società che cercano di abusare del sistema. Per esempio, è accaduto ripetutamente che una società aiutasse a sviluppare uno standard e che, dopo l'approvazione, inciasse che lo standard è basato su un brevetto di sua proprietà e che fornirà le licenze solo alle compagnie prescelte, al prezzo che ha deciso. Per conoscere il lato oscuro della standardizzazione, questo articolo è un punto di partenza eccellente.

### Elenco di letture e bibliografia

Kyas e Crawford, *ATM Networks*

ATM era considerato il protocollo di rete del futuro, ed è considerato ancora importante all'interno del sistema telefonico. Questo libro è una guida aggiornata allo stato attuale di ATM, con informazioni dettagliate sui protocolli ATM e sulla loro integrazione con le reti basate su IP.

Kwok, "A Vision for Residential Broadband Service"

Chi desidera sapere che cosa ne pensava Microsoft della consegna di video su richiesta nel 1995 legga questo articolo. Cinque anni dopo la visione era completamente obsoleta. Il valore di questo articolo sta nel fatto che dimostra che persino le persone più informate e motivate non possono leggere con precisione nel futuro, anche prossimo. Dovrebbe essere una lezione per tutti noi.

Naughton, *A Brief History of the Future*

Ma chi ha inventato Internet? Molte persone hanno rivendicato il riconoscimento, e comunque in tanti hanno dato una mano, in modi diversi. Questa storia di Internet spiega come è accaduto tutto questo, in un modo spiritoso e affascinante, ricco di aneddoti (come AT&T che ripeteva sempre che la comunicazione digitale non avrebbe avuto un futuro).

Perkins, "Mobile Networking in the Internet"

Questo articolo è l'ideale per una valida panoramica sul networking mobile, esaminando strato dopo strato. Sono esaminati gli strati da fisico a trasporto, nonché il middleware, la protezione e le reti ad hoc.

Teger e Waks, "End-User Perspectives on Home Networking"

Le reti domestiche non sono come le reti aziendali. Le applicazioni sono diverse (più multimediali), l'attrezzatura proviene da numerosi fornitori e gli utenti non possiedono capacità tecniche (e nemmeno pazienza in caso di errori). Per saperne di più è possibile leggere questo articolo.

Varshney e Vetter, "Emerging Mobile and Wireless Networks"

Un'altra introduzione alla comunicazione wireless. Parla delle LAN wireless, dei collegamenti wireless locali e dei satelliti, nonché di software e applicazioni correlati.

Wetteroth, *OSI Reference Model for Telecommunications*

Anche se i protocolli OSI non sono più utilizzati, il modello a sette strati è divenuto molto noto. Oltre a spiegare in dettaglio OSI, questo libro applica il modello alle reti di telecomunicazioni (anziché di computer), mostrando dove si trovano all'interno della pila i più noti protocolli di telefonia e altri protocolli vocali.

### 9.1.2 Lo strato fisico

Abramson, "Internet Access Using VSATs"

Le piccole stazioni terrestri stanno diventando più popolari per la telefonia rurale e l'accesso a Internet aziendale nelle nazioni sviluppate. Tuttavia, la natura del traffico in questi due casi differisce notevolmente, pertanto sono necessari protocolli diversi per gestire i due casi. In questo articolo, l'inventore del sistema ALOHA discute numerosi metodi di allocazione del canale che possono essere utilizzati per i sistemi VSAT.

Alkhatib et al., "Wireless Data Networks: Reaching the Extra Mile"

Questo documento è un ottimo punto di partenza per una rapida introduzione alle tecnologie e ai termini relativi alle reti wireless, compreso lo spettro distribuito.

Azzam e Ransom, *Broadband Access Technologies*

Il sistema telefonico, le fibre ottiche, la connessione ADSL, le reti via cavo, i satelliti e persino le linee elettriche sono qui descritte come tecnologie di accesso alla rete. Altri argomenti comprendono le reti domestiche, i servizi, le prestazioni di rete e gli standard. Il libro si conclude con le biografie delle principali società di rete e telecomunicazioni, ma con la velocità di cambiamento nell'industria questo capitolo potrebbe avere una vita più breve dei capitoli sulle tecnologie.

Bellamy, *Digital Telephony*

Tutto ciò che si potrebbe voler sapere sul sistema telefonico è contenuto in questo autorevole libro. Di particolare interesse sono i capitoli sulla trasmissione e il multiplexing, la commutazione digitale, le fibre ottiche, la telefonia mobile e DSL.

Berezdivin et al., "Next-Generation Wireless Communications Concepts and Technologies"

Queste persone si trovano un passo avanti rispetto a tutti gli altri. Il termine "next-generation" nel titolo fa riferimento alle reti wireless di quarta generazione, che dovrebbero fornire il servizio IP ovunque con connettività a Internet integrata, banda larga e un'eccellente qualità del servizio. Questi obiettivi possono essere raggiunti tramite un utilizzo dello spettro intelligente, la gestione dinamica delle risorse e i servizi adattati. Tutto questo potrebbe apparire visionario ora, ma i telefoni cellulari sembravano un'utopia ancora nel 1995.

Dutta-Roy, "An Overview of Cable Modem Technology and Market Perspectives"

La TV via cavo si è evoluta dalla semplice CATV a un complesso sistema di distribuzione per TV, Internet e telefonia. Queste modifiche hanno influito considerevolmente sull'infrastruttura. Per una discussione su impianti, standard e marketing, con un occhio a DOCSIS, questo articolo è perfetto.

Farserotu e Prasad, "A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends"

Molti satelliti per la comunicazione dati si trovano in orbita o sui tavoli da disegno, compresi Astrolink, Cyberstar, Spaceway, Skybridge, Teledesic e iSky. Utilizzano varie tecniche, come il bent pipe e la commutazione satellitare. Questo documento è un ottimo punto di partenza per una panoramica sui diversi sistemi satellitari e sulle tecniche relative.

Hu e Li, "Satellite-Based Internet: A Tutorial"

L'accesso a Internet via satellite è diverso dall'accesso mediante le linee terrestri. Non esiste solo la questione del ritardo, ma anche instradamento e commutazione sono differenti. In questo documento, gli autori esaminano alcune delle questioni correlate all'utilizzo dei satelliti per l'accesso a Internet.

Joel, "Telecommunications and the IEEE Communications Society"

Questo articolo rappresenta la storia delle telecomunicazioni, compatta ma sorprendentemente esauriente, che inizia dal telegrafo e termina con 802.11. Parla anche di radio, telefoni, commutazione analogica e digitale, cavi sottomarini, trasmissione digitale, ATM, broadcasting televisivo, satelliti, TV via cavo, comunicazioni ottiche, telefoni cellulari, commutazione di pacchetto, ARPANET e Internet.

Metcalfe, "Computer/Network Interface Design: Lessons from Arpanet & Ethernet"

Anche se gli ingegneri hanno costruito interfacce di rete per decenni, spesso ci si chiede se hanno imparato qualcosa da tutta questa esperienza. In questo documento, il progettista di Ethernet spiega come costruire un'interfaccia di rete e cosa fare con essa dopo averla realizzata. Non risparmia le critiche, ma spiega che cosa ha sbagliato e che cosa invece ha ideato correttamente.

Palais, *Fiber Optic Communication*, 3<sup>a</sup> edizione

I libri sulla tecnologia ottica tendono a essere destinati agli specialisti, ma questo è uno tra i più accessibili. Parla di guide d'onda, fonti di luce, rilevatori di luce, accoppiatori, modulazione, disturbi e di molti altri argomenti.

Pandya, "Emerging Mobile and Personal Communications Systems"

Per una breve introduzione ai sistemi di comunicazione palmari, questo articolo è quello più adatto. Una delle nove pagine contiene un elenco di 70 acronimi utilizzati nelle altre otto pagine.

Sarikaya, "Packet Mode in Wireless Networks: Overview of Transition to Third Generation"

L'idea delle reti cellulari di terza generazione riguarda la trasmissione di dati wireless. Per una panoramica della gestione dei dati da parte delle reti di seconda generazione e dell'e-

voluzione verso la terza generazione, è utile leggere questo articolo. Gli argomenti trattati comprendono GPRS, IS-95B, WCDMA e CDMA2000.

### 9.1.3 Lo strato data link

**Carlson, *PPP Design, Implementation and Debugging*, 2<sup>a</sup> edizione**

Chi è interessato a informazioni dettagliate su tutti i protocolli della suite PPP, compresi CCP (compressione) ed ECP (crittografia) dovrebbe utilizzare questo libro come riferimento. Si concentra in modo particolare su ANU PPP-2.3, una popolare implementazione di PPP.

**Gravano, *Introduction to Error Control Codes***

Gli errori avvengono in tutte le comunicazioni digitali: sono stati sviluppati molti tipi di codici per rilevarli e correggerli. Questo libro spiega alcuni dei più importanti, dai semplici codici lineari di Hamming ai più complessi campi di Galois e ai codici convoluzionali. Cerca di farlo con il minimo di algebra necessario, ma i calcoli sono ancora numerosi.

**Holzmann, *Design and Validation of Computer Protocols***

I lettori interessati agli aspetti più formali dei protocolli di collegamento dati (e simili) dovrebbero dare un'occhiata a questo libro. Le specifiche, la modellazione, la correttezza e la verifica di tali protocolli sono tra gli argomenti del libro.

**Peterson and Davie, *Computer Networks: A Systems Approach***

Il Capitolo 2 contiene materiale su molte questioni relative al collegamento dati, compresi il framing, il rilevamento degli errori, i protocolli stop-and-wait, i protocolli sliding window e le LAN IEEE 802.

**Stallings, *Data and Computer Communications***

Il Capitolo 7 parla dello strato data link e spiega il controllo di flusso, il rilevamento degli errori e i protocolli base per il collegamento dati, compresi stop-and-wait e indietro di n. Sono inoltre descritti i protocolli di tipo HDLC.

### 9.1.4 Il sottostrato Medium Access Control

**Bhagwat, "Bluetooth: Technology for Short-Range Wireless Apps"**

È un ottimo punto di partenza per un'introduzione al sistema Bluetooth. Parla dei protocolli e dei profili di base, delle radio, delle piconet e dei collegamenti; inoltre, presenta un'introduzione ai vari protocolli.

**Bisdikian, "An Overview of the Bluetooth Wireless Technology"**

Come il documento di Bhagwat (sopra), anche questo è un ottimo punto di partenza per saperne di più sul sistema Bluetooth. Tra gli altri argomenti, parla delle piconet, dello stack dei protocolli e dei profili.

**Crow et al., "IEEE 802.11 Wireless Local Area Networks"**

È un ottimo punto di partenza per una semplice introduzione alla tecnologia e ai protocolli di 802.11. L'enfasi è posta sul sottostrato MAC. Vengono esaminati sia il controllo distribuito sia il controllo centralizzato. Il documento si conclude con alcuni studi simulati sulle prestazioni di 802.11 in varie condizioni.

**Eklund et al., "IEEE Standard 802.16: A Technical Overview of the Wireless MAN Air Interface for Broadband Wireless Access"**

Il collegamento wireless locale, standardizzato da IEEE nel 2002 come 802.16, può rivoluzionare il servizio telefonico portando la banda larga nelle abitazioni. In questa panoramica, gli autori spiegano le molte questioni tecnologiche relative a questo standard.

**Kapp, "802.11: Leaving the Wire Behind"**

Questa breve introduzione a 802.11 spiega le nozioni fondamentali, i protocolli e gli standard rilevanti.

**Kleinrock, "On Some Principles of Nomadic Computing and Multi-Access Communications"**

L'accesso wireless su un canale condiviso è più complesso dell'utilizzo di stazioni cablate che condividono un canale. Tra gli altri problemi vi sono le topologie dinamiche, il routing e la gestione energetica. Nell'articolo sono esaminate queste e altre questioni relative all'accesso al canale da parte di dispositivi wireless portatili.

**Miller and Cummins, *LAN Technologies Explained***

Qualcuno vuole saperne di più sulle tecnologie che possono essere utilizzate in una LAN? Questo libro parla di quasi tutte, comprese FDDI e Token ring, oltre alla popolare Ethernet. Anche se le nuove installazioni delle prime due sono ormai rare, molte reti esistenti possono ancora usarle; anche le reti ad anello sono tuttora comuni (per esempio SONET).

**Perlman, *Interconnections*, 2<sup>a</sup> edizione**

Per una descrizione autorevole ma divertente di bridge, router e dell'instradamento in generale, è possibile fare riferimento al libro di Perlman. L'autore ha progettato gli algoritmi utilizzati in IEEE 802 ed è una delle principali autorità nel campo delle reti.

**Webb, "Broadband Fixed Wireless Access"**

I "come" e i "perché" relativi al wireless a banda larga statica sono esaminati in questo documento. La parte sui "perché" presume che le persone non vogliono un indirizzo di posta elettronica per l'abitazione, uno per il lavoro, numeri telefonici separati per l'abitazione, l'ufficio e il cellulare, un account di instant messaging e magari uno o due numeri di fax. Desiderano un singolo sistema integrato che funzioni ovunque. L'enfasi della parte sulla tecnologia è posta sullo strato fisico, con argomenti come il confronto tra TDD e FDD, tra la modulazione fissa e quella adattativa, nonché il numero di portanti.

## 1.5 Lo strato network

hatti and Crowcroft, "QoS Sensitive Flows: Issues in IP Packet Handling"  
uno dei modi per ottenere una migliore qualità del servizio in una rete consiste nel pianificare attentamente le partenze dei pacchetti da ogni router. In questo documento sono descritti in dettaglio numerosi algoritmi di pianificazione dei pacchetti, insieme alle questioni correlate.

Chakrabarti, "QoS Issues in Ad Hoc Wireless Networks"  
i routing nelle reti ad hoc di computer notebook posti vicini tra loro è già abbastanza difficile senza doversi preoccupare anche della qualità del servizio. Ciò nonostante, le persone sono interessate alla qualità del servizio, per cui occorre prestare attenzione a questo argomento. La natura delle reti ad hoc e alcune delle questioni correlate al routing e alla qualità del servizio sono discusse in questo articolo.

Comer, *Internetworking with TCP/IP*, volume 1, 4<sup>a</sup> edizione  
Comer ha scritto l'opera definitiva sulla suite di protocolli TCP/IP. I capitoli da 4 a 11 sono relativi a IP e ai protocolli correlati nello strato network. Gli altri capitoli riguardano principalmente gli strati superiori e vale la pena di leggerli.

Huitema, *Routing in the Internet*  
È il libro adatto a chi desidera sapere tutto il necessario sul routing in Internet. Sono presentati in notevole dettaglio sia gli algoritmi pronunciabili (come RIP, CIDR e MBONE) sia quelli meno pronunciabili (come OSPF, IGRP, EGP e BGP). Presenta anche le nuove funzionalità, come il multicast, IP mobile e la prenotazione delle risorse.

Malhotra, *IP Routing*  
Questo libro contiene molto materiale dettagliato sul routing IP. I protocolli descritti comprendono RIP, RIP-2, IGRP, EIGRP, OSPF e BGP-4.

Metz, "Differentiated Services"  
Le garanzie di qualità del servizio sono importanti per molte applicazioni multimediali. I servizi integrati e i servizi differenziati sono due possibili approcci per raggiungerle. Entrambi sono discussi in questo articolo, che pone l'accento sui servizi differenziati.

Metz, "IP Routers: New Tool for Gigabit Networking"  
La maggior parte degli altri riferimenti per il Capitolo 5 riguarda gli algoritmi di routing. Questo è diverso: parla del funzionamento effettivo dei router. Hanno seguito un processo evolutivo particolare, da workstation generiche a macchine di routing altamente specializzate. Per saperne di più, questo articolo è un ottimo punto di partenza.

Nemeth et al., *UNIX System Administration Handbook*

Il Capitolo 13 di questo libro ha a che fare con il networking in modo più pratico rispetto agli altri riferimenti. Anziché presentare solo i concetti astratti, offre ulteriori consigli su cosa fare per gestire effettivamente una rete.

Perkins, "Mobile Networking through Mobile IP"

Visto che i dispositivi di elaborazione portatili diventano sempre più comuni, Mobile IP sta diventando sempre più importante. Questo tutorial rappresenta una valida introduzione a esso e agli argomenti correlati.

Perlman, *Interconnections: Bridges and Routers*, 2<sup>a</sup> edizione

Nei Capitoli da 12 a 15, Perlman descrive molte delle questioni coinvolte nella progettazione di algoritmi di routing unicast e multicast, sia per le WAN sia per le reti di LAN, e la loro implementazione in vari protocolli. Tuttavia, la parte migliore del libro è il Capitolo 18, in cui l'autore distilla i suoi anni di esperienza con i protocolli di rete in un capitolo informativo e divertente.

Puzmanova, *Routing and Switching: Time of Convergence?*

Alla fine degli anni '90, alcuni produttori di attrezzature di rete hanno iniziato a chiamare ogni dispositivo "switch", mentre molti manager delle grandi reti hanno affermato di essere passati dai router agli switch. Come implica il titolo, questo libro predice il futuro di router e switch e si chiede se stanno realmente convergendo in un dispositivo unico.

Royer and Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks"

L'algoritmo di routing ad hoc AODV discusso nel Capitolo 5 non è l'unico noto. Ne esistono molti altri, compresi DSDV, CGSR, WRP, DSR, TORA, ABR, DRP e SRP: in questo libro sono presentati e confrontati tra loro. Chiaramente, se si prevede di inventare un nuovo protocollo di routing ad hoc, il primo passaggio è pensare a un acronimo di tre o quattro lettere.

Stevens, *TCP/IP Illustrated*, volume 1

I Capitoli da 3 a 10 offrono una descrizione esauriente di IP e dei protocolli correlati (ARP, RARP e ICMP), illustrati con esempi.

Striegel and Manimaran, "A Survey of QoS Multicasting Issues"

Il multicasting e la qualità del servizio sono due argomenti di importanza crescente, visto che i servizi come la televisione e la radio via Internet iniziano a diffondersi. In questo documento, gli autori discutono di come tenere conto di entrambe le questioni negli algoritmi di routing.

### 1.1.5 Lo strato network

Bhatti and Crowcroft, "QoS Sensitive Flows: Issues in IP Packet Handling"

Uno dei modi per ottenere una migliore qualità del servizio in una rete consiste nel pianificare attentamente le partenze dei pacchetti da ogni router. In questo documento sono discussi in dettaglio numerosi algoritmi di pianificazione dei pacchetti, insieme alle questioni correlate.

Chakrabarti, "QoS Issues in Ad Hoc Wireless Networks"

Il routing nelle reti ad hoc di computer notebook posti vicini tra loro è già abbastanza difficile senza doversi preoccupare anche della qualità del servizio. Ciò nonostante, le persone sono interessate alla qualità del servizio, per cui occorre prestare attenzione a questo argomento. La natura delle reti ad hoc e alcune delle questioni correlate al routing e alla qualità del servizio sono discusse in questo articolo.

Comer, *Internetworking with TCP/IP*, volume 1, 4<sup>a</sup> edizione

Comer ha scritto l'opera definitiva sulla suite di protocolli TCP/IP. I capitoli da 4 a 11 sono relativi a IP e ai protocolli correlati nello strato network. Gli altri capitoli riguardano principalmente gli strati superiori e vale la pena di leggerli.

Huitema, *Routing in the Internet*

È il libro adatto a chi desidera sapere tutto il necessario sul routing in Internet. Sono presentati in notevole dettaglio sia gli algoritmi pronunciabili (come RIP, CIDR e MBONE) sia quelli meno pronunciabili (come OSPF, IGRP, EGP e BGP). Presenta anche le nuove funzionalità, come il multicast, IP mobile e la prenotazione delle risorse.

Malhotra, *IP Routing*

Questo libro contiene molto materiale dettagliato sul routing IP. I protocolli descritti comprendono RIP, RIP-2, IGRP, EIGRP, OSPF e BGP-4.

Metz, "Differentiated Services"

Le garanzie di qualità del servizio sono importanti per molte applicazioni multimediali. I servizi integrati e i servizi differenziati sono due possibili approcci per raggiungerle. Entrambi sono discussi in questo articolo, che pone l'accento sui servizi differenziati.

Metz, "IP Routers: New Tool for Gigabit Networking"

La maggior parte degli altri riferimenti per il Capitolo 5 riguarda gli algoritmi di routing. Questo è diverso: parla del funzionamento effettivo dei router. Hanno seguito un processo evolutivo particolare, da workstation generiche a macchine di routing altamente specializzate. Per saperne di più, questo articolo è un ottimo punto di partenza.

Nemeth et al., *UNIX System Administration Handbook*

Il Capitolo 13 di questo libro ha a che fare con il networking in modo più pratico rispetto agli altri riferimenti. Anziché presentare solo i concetti astratti, offre ulteriori consigli su cosa fare per gestire effettivamente una rete.

Perkins, "Mobile Networking through Mobile IP"

Visto che i dispositivi di elaborazione portatili diventano sempre più comuni, Mobile IP sta diventando sempre più importante. Questo tutorial rappresenta una valida introduzione a esso e agli argomenti correlati.

Perlman, *Interconnections: Bridges and Routers*, 2<sup>a</sup> edizione

Nei Capitoli da 12 a 15, Perlman descrive molte delle questioni coinvolte nella progettazione di algoritmi di routing unicast e multicast, sia per le WAN sia per le reti di LAN, e la loro implementazione in vari protocolli. Tuttavia, la parte migliore del libro è il Capitolo 18, in cui l'autore distilla i suoi anni di esperienza con i protocolli di rete in un capitolo informativo e divertente.

Puzmanova, *Routing and Switching: Time of Convergence?*

Alla fine degli anni '90, alcuni produttori di attrezzature di rete hanno iniziato a chiamare ogni dispositivo "switch", mentre molti manager delle grandi reti hanno affermato di essere passati dai router agli switch. Come implica il titolo, questo libro predice il futuro di router e switch e si chiede se stanno realmente convergendo in un dispositivo unico.

Royer and Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks"

L'algoritmo di routing ad hoc AODV discusso nel Capitolo 5 non è l'unico noto. Ne esistono molti altri, compresi DSDV, CGSR, WRP, DSR, TORA, ABR, DRP e SRP: in questo libro sono presentati e confrontati tra loro. Chiaramente, se si prevede di inventare un nuovo protocollo di routing ad hoc, il primo passaggio è pensare a un acronimo di tre o quattro lettere.

Stevens, *TCP/IP Illustrated*, volume 1

I Capitoli da 3 a 10 offrono una descrizione esauriente di IP e dei protocolli correlati (ARP, RARP e ICMP), illustrati con esempi.

Striegel and Manimaran, "A Survey of QoS Multicasting Issues"

Il multicasting e la qualità del servizio sono due argomenti di importanza crescente, visto che i servizi come la televisione e la radio via Internet iniziano a diffondersi. In questo documento, gli autori discutono di come tenere conto di entrambe le questioni negli algoritmi di routing.

Yang and Reddy, "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks"

Gli autori hanno rilevato una tassonomia per gli algoritmi di controllo della congestione. Le categorie principali sono un ciclo aperto con il controllo dell'origine, un ciclo aperto con il controllo della destinazione, un ciclo chiuso con retroazione esplicita e un ciclo chiuso con retroazione implicita. Essi utilizzano questa tassonomia per descrivere e classificare 23 algoritmi esistenti.

### 9.1.6 Lo strato trasporto

Comer, *Internetworking with TCP/IP*, volume 1, 4<sup>a</sup> edizione

Come affermato in precedenza, Comer ha scritto l'opera definitiva sulla suite di protocolli TCP/IP. Il Capitolo 12 riguarda UDP; il Capitolo 13 parla di TCP.

Hall and Cerf, *Internet Core Protocols: The Definitive Guide*

Chi ama trarre le informazioni direttamente dalla sorgente può scegliere questo riferimento per conoscere meglio TCP. Dopo tutto, Cerf lo ha co-inventato. Il Capitolo 7 è un ottimo riferimento su TCP, che mostra come interpretare le informazioni fornite dagli strumenti di analisi del protocollo e gestione della rete. Altri capitoli parlano di UDP, IGMP, ICMP e ARP.

Kurose and Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*  
Il Capitolo 3 riguarda lo strato trasporto e contiene una discreta quantità di materiale su UDP e TCP. Discute anche i protocolli stop-and-wait e go-back-n di cui abbiamo parlato nel Capitolo 3.

Mogul, "IP Network Performance"

Nonostante il titolo, l'articolo riguarda per la maggior parte TCP e le prestazioni di rete in generale, poi si occupa in particolare delle prestazioni di IP. È ricco di utili regole e linee guida.

Peterson and Davie, *Computer Networks: A Systems Approach*

Il Capitolo 5 parla di UDP, TCP e di altri protocolli correlati. Anche le prestazioni di rete sono descritte brevemente.

Stevens, *TCP/IP Illustrated*, volume 1

I Capitoli da 17 a 24 offrono una descrizione esauriente di TCP, illustrata con esempi.

### 9.1.7 Lo strato applicazione

Bergholz, "Extending Your Markup: An XML Tutorial"

Un'introduzione breve e diretta a XML per i principianti.

Cardellini et al., *The State-of-the-Art in Locally Distributed Web-Server Systems*

Il Web diventa sempre più popolare e alcuni siti Web necessitano di grandi server farm per gestire il traffico. La parte complessa della costruzione di una server farm è la distribuzione del carico tra le macchine. Questo documento parla a lungo di questo problema.

Berners-Lee et al., "The World Wide Web"

Una prospettiva sul Web e sul suo futuro, realizzata dalla persona che lo ha inventato e da alcuni suoi colleghi del CERN. L'articolo si concentra sull'architettura del Web, su URL, HTTP e HTML, nonché sulle direttive future; fa inoltre un confronto con altri sistemi informatici distribuiti.

Choudhury et al., "Copyright Protection for Electronic Publishing on Computer Networks"

Anche se molti libri e articoli descrivono gli algoritmi di crittografia, pochi spiegano come utilizzarli per impedire che gli utenti distribuiscano i documenti che non possono decrittare. Questo documento descrive diversi meccanismi che possono aiutare a proteggere i diritti d'autore nell'era elettronica.

Collins, "Carrier Grade Voice over IP"

Questo libro è rivolto a chi ha letto il documento di Varshney et al., e ora desidera conoscere tutti i dettagli su Voice over IP utilizzando H.323. Anche se il libro è lungo e dettagliato, è impostato come tutorial e non richiede conoscenze di ingegneria telefonica.

Davison, "A Web Caching Primer"

Con la crescita del Web, il caching diventa fondamentale per prestazioni ottimali. Per una breve introduzione al Web caching, è possibile consultare questo riferimento.

Krishnamurthy and Rexford, *Web Protocols and Practice*

Sarebbe difficile trovare un libro più esauriente di questo su tutti gli aspetti del Web. Parla di client, server, proxy e caching, come prevedibile. Esistono però anche capitoli sul traffico Web e sulle misurazioni, nonché capitoli sulla ricerca e il miglioramento del Web.

Rabinovich and Spatscheck, *Web Caching and Replication*

Per una descrizione esauriente del caching e della replica Web, si può ricorrere a questo libro. Tra gli argomenti trattati in dettaglio vi sono i proxy, la cache, il pre-fetching, le reti di consegna del contenuto, la selezione dei server e altro ancora.

Shahabi et al., "Yima: A Second-Generation Continuous Media Server"

I server multimediali sono sistemi complessi che devono gestire la pianificazione della CPU, il posizionamento dei file su disco, la sincronizzazione dei flussi e altro ancora. Con

il passare del tempo le persone hanno imparato a progettarli meglio. Questo documento presenta una panoramica dell'architettura di un sistema recente.

#### Tittel et al., *Mastering XHTML*

Due libri in un grande volume, che presenta il nuovo linguaggio di markup standard per il Web. L'opera inizia con un testo che descrive XHTML, concentrandosi sulle differenze con HTML normale. Segue una guida di riferimento esauriente ai tag, ai codici e ai caratteri speciali utilizzati in XHTML 1.0.

#### Varshney et al., "Voice over IP"

Come funziona Voice over IP? Sta per sostituire le reti telefoniche pubbliche commutate? Basta leggere per scoprirla.

#### 9.1.8 La sicurezza delle reti

##### Anderson, "Why Cryptosystems Fail"

Secondo Anderson, la sicurezza dei sistemi bancari è scadente, ma non a causa di intrusi intelligenti che forzano DES sui loro PC. Il problema reale deriva da impiegati disonesti (un impiegato di banca che cambia l'indirizzo di posta di un cliente col proprio, per intercettarne la tessera bancomat e il codice PIN) ed errori di programmazione (che assegnano a tutti i clienti lo stesso codice PIN). È particolarmente interessante l'arrogante risposta fornita dalle banche di fronte a un problema evidente: "I nostri sistemi sono perfetti, pertanto tutti gli errori sono dovuti a sbagli del cliente o a frodi".

##### Anderson, *Security Engineering*

Sotto certi aspetti, questo libro è la versione di 600 pagine di "Why Cryptosystems Fail". È più tecnico di *Secrets and Lies*, ma meno specialistico di *Network Security* (vedere sotto). Dopo un'introduzione alle tecniche di base per la sicurezza, interi capitoli sono dedicati a varie applicazioni, come il remote banking, il controllo e comando in campo nucleare, la stampa, la biometrica, la sicurezza fisica, la guerra elettronica, la sicurezza nelle telecomunicazioni, l'e-commerce e la protezione del copyright. La terza parte del libro riguarda i criteri, la gestione e la valutazione dei sistemi.

##### Artz, "Digital Steganography"

La steganografia ci riporta all'antica Grecia, dove la cera era spalmata su tavolette bianche al fine di applicare al legno sottostante dei messaggi segreti prima di riapplicare la cera. Oggi vengono utilizzate tecniche diverse, ma l'obiettivo è lo stesso. L'articolo presenta le tecniche moderne per nascondere le informazioni nelle immagini, nell'audio o in altre portanti.

##### Brands, *Rethinking Public Key Infrastructures and Digital Certificates*

Questo libro è più che un'introduzione ai certificati digitali: è anche un'esauriente opera di avvocatura. L'autore ritiene che gli attuali sistemi basati su carta per la verifica dell'identità siano obsoleti e inefficienti, e ritiene che i certificati digitali debbano essere utiliz-

zati per applicazioni come il voto elettronico, la gestione dei diritti digitali e anche il cambio di valuta. Avvisa inoltre che, senza PKI e crittografia, Internet potrebbe diventare uno strumento di sorveglianza su vasta scala.

##### Kaufman et al., *Network Security, 2<sup>a</sup> edizione*

Questo autorevole libro è il primo posto dove cercare informazioni tecniche su protocolli e algoritmi per la protezione della rete. Protocolli e algoritmi a chiavi pubbliche e segrete, hash dei messaggi, autenticazione, Kerberos, PKI, IPsec, SSL/TLS, protezione della posta elettronica: tutti gli argomenti sono spiegati attentamente e a lungo, con molti esempi. Il Capitolo 26 sul folclore della sicurezza è una vera perla. Nella sicurezza, il problema sta nei dettagli. Chiunque pianifichi di progettare un sistema di protezione potrà imparare molto da questo capitolo ricco di consigli.

##### Pohlmann, *Firewall Systems*

I firewall sono la prima (e l'ultima) linea di difesa di molte reti. Questo libro spiega come funzionano e cosa fanno, dal più semplice firewall software progettato per proteggere un singolo PC alle applicazioni firewall avanzate che risiedono tra una rete privata e la sua connessione a Internet.

##### Schneier, *Applied Cryptography, 2<sup>a</sup> edizione*

Questo compendio monumentale è il peggior incubo di NSA: un singolo libro che descrive ogni algoritmo crittografico conosciuto. Per peggiorare le cose (o migliorarle, a seconda del punto di vista), il libro contiene la maggior parte degli algoritmi sotto forma di programmi eseguibili (in C). Inoltre, vengono forniti oltre 1.600 riferimenti alla letteratura sulla crittografia. Questo libro non è per i principianti, ma è utile a chi desidera *realmente* mantenere segreti i suoi file.

##### Schneier, *Secrets and Lies*

Chi ha letto *Applied Cryptography* dall'inizio alla fine conosce tutto l'indispensabile sugli algoritmi di crittografia. Se poi legge *Secrets and Lies* dall'inizio alla fine (operazione che richiede molto meno tempo), può scoprire che gli algoritmi di crittografia non sono che l'inizio. La maggior parte delle debolezze nella protezione non sono dovute ad algoritmi errati o a chiavi troppo corte, ma a fallo nell'ambiente di protezione. Sono presentati esempi infiniti su minacce, attacchi, difese, contrattacchi e altro ancora. Per una discussione non tecnica e affascinante sulla protezione dei computer, questo libro è l'ideale.

##### Skoudis, *Counter Hack*

Il modo migliore per fermare un hacker è pensare come lui. Questo libro mostra il modo in cui gli hacker osservano la rete e suppone che la protezione dovrebbe essere una funzione dell'intera struttura di rete, e non aggiunta a posteriori inserendo una tecnologia specifica. Parla di tutti gli attacchi comuni, compresi quelli definiti di "ingegneria sociale", che traggono vantaggio dagli utenti che non hanno familiarità con le misure di protezione dei computer.

- KYLOMENOS, G., POLYZOS, G.C., MAHONEN, P. e SAARANEN, M.: "TCP Performance Issues over Wireless Links", *IEEE Commun. Magazine*, anno 39, pagg. 52-58, aprile 2001.
- WANG, C.-Q. e REDDY, A.V.S.: "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks", *IEEE Network Magazine*, anno 9, pagg. 34-45, luglio/agosto 1995.
- UVAL, G.: "How to Swindle Rabin", *Cryptologia*, anno 3, pagg. 187-190, luglio 1979.
- SACKS, M.: "Antiterrorist Legislation Expands Electronic Snooping", *IEEE Internet Computing*, anno 5, pagg. 8-9, novembre-dicembre 2001.
- TADEH, A.N., JABBARI, B., PICKHOLTZ, R. e VOJCIC, B.: "Self-Organizing Packet Radio Ad Hoc Networks with Overlay (SOPRANO)", *IEEE Commun. Mag.*, anno 40, pagg. 149-157, giugno 2002.
- CHANG, L.: "Comparison of Two Bridge Routing Approaches", *IEEE Network Magazine*, anno 2, pagg. 44-48, gennaio/febbraio 1988.
- CHANG, L.: "RSVP: A New Resource ReSerVation Protocol", *IEEE Network Magazine*, anno 7, pagg. 8-18, settembre/ottobre 1993.
- CHANG, Y. e RYU, B.: "Mobile and Multicast IP Services in PACS: System Architecture, Prototype, and Performance", *Mobile Networks and Applications*, anno 6, pagg. 81-94, gennaio-febbraio 2001.
- GIMMERMANN, P.R.: *The Official PGP User's Guide*, Cambridge, MA, M.I.T. Press, 1995a.
- GIMMERMANN, P.R.: *PGP: Source Code and Internals*, Cambridge, MA, M.I.T. Press, 1995b.
- GIFFE, G.K.: *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Boston, Addison-Wesley, 1949.
- HIV, J. e LEMPEL, Z.: "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. on Information Theory*, anno IT-23, pagg. 337-343, maggio 1977.

## Indice

### Numeri

- 1000Base-x, 288
- 100Base-x, 285
- 10Base-x, 272
- 2.5G, sistema di telefonia cellulare, 168
- 3G, *vedere* Terza generazione, telefono cellulare
- 4B/5B, 285
- 802 (*vedere* IEEE 802.x)
- 802.15 (*vedere* Bluetooth)
- 802.16 (*vedere* IEEE 802.16)
- 802.3 (*vedere* Ethernet)
- 8B/10B, 289
- 8B/6T, 285

### A

- AAL (*vedere* ATM Adaptation Layer)
- AAL-SAP, 494
- Abramson, Norman, 65-66
- Abstract Syntax Notation 1, 768
- Access point, 68
- Accesso remoto, 57
- Accodamento equo pesato, 409
- Accodamento equo, 408-409
- Accordo sul livello di servizio, 400
- ACL (*vedere* Asynchronous ConnectionLess,)
- Active Server Page, 646
- Ad hoc On-demand Distance Vector, routing, 375-380
- Adattativo, algoritmo di routing, 351

- ADC (*vedere* Analog Digital Converter)
- ADCCP (*vedere* Advanced Data Communication Control Procedure)
- Address Resolution Protocol, 450-452
- ARP gratuito, 463
- proxy, 452
- ADSL (*vedere* Asymmetric Digital Subscriber Line)
- Advanced Data Communication Control Procedure, 234
- Advanced Encryption Standard, 741-745
- Rijndael, 743-745
- Advanced Mobile Phone System, 154-157
- Advanced Networks and Services, 55
- Advanced Research Projects Agency, 51
- AES (*vedere* Advanced Encryption Standard)
- Agente di trasferimento dei messaggi, 590
- Agente di trasferimento, e-mail, 590
- Agente esterno, 373
- Agente fisso, 373
- Agente utente, e-mail, 590
- AH (*vedere* Authentication Header)
- Algoritmi basati sul flusso, 409-412
- Algoritmo token bucket, 402-405
- Algoritmo di avvio lento di Jacobson, 549-550
- Algoritmo di codifica, video, 696
- Algoritmo di decodifica, video, 696
- Algoritmo di flooding, 355, 357
- Algoritmo di reverse path forwarding, 369-370
- Algoritmo di Karn, 552
- Algoritmo di Nagle, 545-547

Algoritmo di routing proporzionale, 408  
 Algoritmo di Routing, 20, 347, 350-384  
     adattivo, 351-352  
     ARPANET, 357, 454  
     Bellman-Ford, 357-360, 454  
     flooding, 355-357  
     Ford-Fulkerson, 357-360  
     gerarchico, 366-368  
     host mobile, 372-375  
     IS-IS, 365-366  
     multicast, 370-372  
     non adattivo, 351  
     OSPF, 454-459  
     ottimale, 352-353  
     percorso più corto, 353-356  
     proporzionale, 408  
     rete ad hoc, 375-380  
     reverse path forwarding, 369-370  
     stato dei collegamenti, 360-366  
     vettore delle distanze, 357-360  
 Algoritmo non adattivo, 351  
 Algoritmo RSA, 753-755  
 Alias, posta elettronica, 593  
 Alice, 731  
 Allineamento, TV via cavo, 174  
 Allocazione dinamica del canale, 249-251  
 ALOHA puro, 251-254  
 ALOHA slotted, 254-255  
 American National Standards Institute, 74  
 AMPS (*vedere* Advanced Mobile Phone System)  
 Analisi del traffico, 774  
 Analisi della raggiungibilità, 230  
 Analog Digital Converter, 675  
 Anderson, Ross, 742  
 Andreessen, Mark, 57, 611  
 Anello a chiave privata, 803  
 Anello a chiave pubblica, 803  
 ANS (*vedere* Advanced Networks and Services)  
 ANSI (*vedere* American National Standards Institute)  
 ANSNET, 55  
 AODV (*vedere* Ad hoc On-demand Distance Vector, routing)  
 Applet, 650  
 Applicazione helper, 617  
 Apprendimento inverso, 323  
 Architettura della rete di computer, 28-30  
 Area della dorsale, 456  
 Area proibita, 499-500  
 Area, 456  
 Armatura ASCII, 598  
 Armonica, 86  
 ARP (*vedere* Address Resolution Protocol)

ARP gratuito, 463  
 ARPA (*vedere* Advanced Research Projects Agency)  
 ARPANET, 50-54  
     algoritmo di routing, 357, 454  
 ARQ (*vedere* Automatic Repeat reQuest)  
 Array di dischi, 708  
 AS (*vedere* Autonomous System)  
 ASN.1 (*vedere* Abstract Syntax Notation 1)  
 ASP (*vedere* Active Server Page)  
 Assegnazione dei nomi sicura, 806-813  
 Assegnazione statica del canale, 248-249  
 Assunzione del tempo diviso a intervalli, 250  
 Asymmetric Digital Subscriber Line, 130-134  
     confronto con cavo, 175-176  
 Asynchronous ConnectionLess link, Bluetooth, 315  
 Asynchronous Transfer Mode, 61-65  
 ATM (*vedere* Asynchronous Transfer Mode)  
 ATM Adaptation Layer, 64  
 Attacco alla sicurezza,  
     brigata del secchio, 792  
     compleanno, 763-765, 782  
     consumo di energia, 751  
     DDoS, 778  
     determinazione dei tempi, 751  
     DoS, 778  
     riflessione, 787-790  
     ripetizione, 794  
     riutilizzo del keystream, 749  
     solo testo cifrato, 727  
     testo in chiaro noto, 727  
     testo in chiaro scelto, 727  
     uomo nel mezzo, 792  
 Attacco per riflessione, 787-790  
 Attacco per ripetizione, 794  
 Attenuazione, 125  
 Attributo  
     certificato, 767  
     HTML, 630  
 Audio digitale, 674-676  
 Audio MPEG Layer 3, 676-679  
 Audio, 674-692  
     introduzione al digitale, 674-676  
 Aumento della riservatezza, 734  
 Autenticazione Needham-Schroeder, 794-795  
 Autenticazione, 785  
     chiave condivisa, 786-790  
     chiave pubblica, 798-799  
     Diffie-Hellman, 791-792  
     Kerberos, 796-798  
     Needham-Schroeder, 794-795  
     Otway-Rees, 795-796  
     utilizzo di un centro di distribuzione  
         delle chiavi, 793-796

utilizzo di un HMAC, 790  
 Autenticazione, protocolli, 785-799  
 Authentication Header, 774-775  
 Authenticode, 817  
 Automatic Repeat reQuest, 209  
 Autonomous System, 427, 432, 456-458  
 Autorità regionale, 769  
 Avvio lento, TCP, 549-550  
**B**  
 Backoff esponenziale binario, algoritmo, 278-279  
 Banda larga, 130  
 Banda, 88  
 Bande Industrial, Scientific, Medical, 106, 292-293, 315  
 Baran, Paul, 50  
 Base diagonale, 732  
 Base rettilinea, 732  
 Base64, codifica, 598  
 Baud, 127  
 BB84 quantum, crittografia, 731-734  
 Bell Operating Companies, 122  
 Bell, Alexander Graham, 119  
 Bellman-Ford, algoritmo di routing, 357-360, 454  
 Bent pipe, transponder, 109  
 B-frame, MPEG, 701, 703  
 BGP (*vedere* Border Gateway Protocol)  
 Biham, Eli, 742  
 Bit di parità, 194  
 Bit stuffing, 190  
 Blaatland, Harald, 310  
 Blocco monouso, 730-734  
 Blowfish, 751  
 Bluetooth SIG, 310, 311  
 Bluetooth, 21, 310-317  
     applicazioni, 312-313  
     architettura, 311  
     Asynchronous Connectionless link, 315  
     collegamento sincrono orientato alla  
         connessione, 316  
     collegamento, 315  
     piconet, 311  
     pila di protocolli, 313-314  
     profilo, 312-313  
     scatternet, 311  
     sicurezza, 783-784  
     storia, 310  
     strato banda base, 315-316  
 Bob, 731  
 BOC (*vedere* Bell Operating Companies)  
 BOOTP (*vedere* protocollo BOOT)  
 Border Gateway Protocol, 459-461  
 Bridge remoto, 325-326  
 Bridge Spanning tree, 323-325  
**C**  
 CA (*vedere* Certification Authority)  
 Cablaggio di Categoria 3, 91  
 Cablaggio di Categoria 5, 92  
 Cable Modem Termination System, 173  
 Cache avvelenata, 807  
 Cache Web, 657-659  
     gerarchica, 658-659  
     intestazione if-modified-since, 659  
     intestazione last-modified, 658-659  
     stantia, 658-659  
 Caching gerarchico, Web, 658-659  
 Caching Web gerarchico, 658-659  
 Caching, Web (*vedere* cache Web)  
 Campo, video, 693  
 Canale ad accesso casuale, 162, 247  
 Canale di assegnazione dell'accesso, 162  
 Canale di controllo broadcast, 161  
 Canale di controllo comune, 161  
 Canale di controllo dedicato, 161  
 Canale di paging, 161  
 Canale multiaccesso, 247  
 Carnivore, 13  
 Carrier extension, gigabit Ethernet, 287-288  
 Carrier hotel, 59  
 Carrier Sense Multiple Access  
     1-persistente, 255  
     con rilevamento delle collisioni, 255-258  
     non-persistente, 256  
     p-persistente, 256  
     senza collisioni, 296-297  
 Carrier sense, protocollo, 255

Casella di posta, posta elettronica, 591  
 Catena di trust, 770  
 Cavo coassiale, 92  
 Cavo in rame, confronto con la fibra ottica, 98-99  
 Cavo multidrop, 66  
 Cavo ricetrasmettitore, 272  
 CCITT, 72  
 CD (*vedere* Committee Draft)  
 CD audio, 676  
 CDMA (*vedere* Code Division Multiple Access)  
 CDMA2000, 167  
 CdmaOne, 162  
 CDN (*vedere* Content Delivery Network)  
 Cella  
     HTML, 633  
     telefono cellulare, 154  
 Centrale a nastro strappato, 149  
 Centrale interurbana, 120  
 Centrale locale, 120  
 Centralina, 91  
 Certificate Revocation List, 771  
 Certificati, 765-771  
 Certification Authority, 766  
 Certificato di chiave pubblica, 765-771  
 CGI (*vedere* Common Gateway Interface)  
 Challenge-response, protocollo, 786  
 Character stuffing, 189-190  
 Chat room, 7  
 Checksum, CRC, 197  
 Chiave  
     accordo, 381  
     crittografica, 725  
 Chiave di sessione, 787  
 Chiave Premaster, 814  
 Chiave privata, 753  
 Chiave pubblica, 753  
 Chiave segreta, 753  
 Chip clipper, 820  
 Chord, 380-384  
 HTML (*vedere* Compact HTML)  
 CIDR (*vedere* Classless InterDomain Routing)  
 Cifrario a permutazione, 729-730  
 Cifrario di Cesare, 727  
 Cifrario prodotto, 738  
 Cifratura a blocchi, 737  
 Cifratura a blocchi, modalità block chaining, 746-747  
 Cifratura a sostituzione monoalfabetica, 728  
 Cifratura, 724  
 Cifratura, modalità feedback, 747-748  
 Circuiti virtuali, concatenati, 422-423  
 Circuito virtuale permanente, 62  
 Clark, David, 46  
 Clark, Wesley, 51

Classe A, B, C, D, indirizzi, 437  
 Classi di servizio, IEEE 802.16, 308-309  
 Classless InterDomain Routing, 441-444  
 Clear To Send, 269  
 CLEC (*vedere* Competitive LEC)  
 Client, 4  
 CMTS (*vedere* Cable Modem Termination System)  
 Coda, data link, 184, 200  
 Code Division Multiple Access, 162-166  
 Codec, 140  
 Codeword, 193  
 Codice di correzione degli errori, 193  
 Codice di Gray, 294  
 Codice di Hamming, 193-195, 307  
 Codice di rilevamento degli errori, 193  
 Codice mobile, 817  
 Codice polinomiale, 196  
 Codice Walsh, 164  
 Codice Walsh/Hadamard, 295  
 Codice, 724  
 Codifica a forma d'onda, 676  
 Codifica a sostituzione, 727-729  
 Codifica con perdite, video, 696  
 Codifica Manchester differenziale, 274-5  
 Codifica Manchester, 274-275  
 Codifica percettiva, 677  
 Codifica predittiva, 143  
 Codifica quoted-printable, 598  
 Codifica Run-length, 699  
 Codifica senza perdite, video, 696  
 Codificatore PCM G.711, 686  
 Collegamento interurbano, 120  
 Collegamento ipertestuale, 612  
 Collegamento locale Wireless (*vedere* IEEE 802.16)  
 Collegamento locale, 120, 124  
 Collegamento sincrono orientato alla connessione,  
     Bluetooth, 316  
 Collegamento, Bluetooth, 315  
 Collisione, 250  
 Commercio elettronico, 5  
 Committee draft, 75  
 Common carrier, 72  
 Common Gateway Interface, 644-645  
 Comutazione dello strato data link, 317-336  
 Comutazione di circuito, 147-148  
 Comutazione di etichetta, 415-417  
 Comutazione di messaggio, 148-149  
 Comutazione di pacchetto store-and-forward, 20, 344  
 Comutazione store-and-forward, 149  
 Compact HTML  
     confronto con HTML 1.0, 668  
     esempio, 670  
 Competitive LEC, 134

Composizione, posta elettronica, 590  
 Compressione audio, 676-679  
     codifica percettiva, 677  
     mascheramento delle frequenze, 677  
     mascheramento temporale, 677  
     MP3, 676-679  
     psicoacustica, 677  
 Compressione JPEG, 697-700  
     codifica run-length, 699  
     DCT, 698  
     preparazione del blocco, 697  
     quantizzazione, 698-699  
 Compressione MPEG, 700-704  
     MPEG-1, 700-703  
     MPEG-2, 700, 703-704  
     sincronizzazione audio/video, 701  
     tipi di fotogramma, 701-702  
 Compressione video, 696-704  
     algoritmo di codifica, 696  
     algoritmo di decodifica, 696  
     con perdita, 696  
     JPEG, 697-700  
     MPEG, 700-704  
     senza perdita, 696  
 Computer big endian, 433  
 Computer little endian, 433  
 Concorso di bellezza, 105  
 Condivisione di risorsa, 3  
 Connessione persistente, 652  
 Connessione tra reti, 420-422  
 Connettore a vampiro, Ethernet, 271  
 Consegnare dati, 302  
 Content Delivery Network, 660-662  
     proxy, 662  
 Contesa, 251  
 Conto alla rovescia binario, protocollo, 260-261  
 Controllo ActiveX, 650  
 Controllo degli errori, 31, 191-192  
 Controllo del dialogo, 40  
 Controllo del flusso, 31, 192, 506-510  
     basato sulla retroazione, 192  
     basato sulla velocità, 192  
 Controllo delle congestioni, 384-396  
     bit di allarme, 391  
     controllo del jitter, 395-396  
     pacchetto choke, 391-394  
     principi, 386-388  
     sottoreti a circuito virtuale, 389-390  
     sottoreti a datagrammi, 391-395  
     TCP, 547-548  
 Controllo di ammissione, 389, 406-408  
 Controllo di flusso basato sulla retroazione, 192  
 Controllo di flusso basato sulla velocità, 192

Cookie non persistente, Web, 626  
 Cookie persistente, Web, 626  
 Cookie, Web, 626-629  
 Copyright, 826-828  
 Corpo, posta elettronica, 591  
 Correttezza del protocollo, 229-234  
 Correzione di errori diretta, 193, 307  
 Costituzione della connessione, 496-502  
     TCP, 539-540  
 CRC (*vedere* Cyclic Redundancy Check)  
 Creazione di standard, 71-77  
 Criterio del vino, 394  
 Crittoanalisi differenziale, 750-751  
 Crittoanalisi lineare, 751  
 Crittoanalisi, 726, 750-751  
     consumo elettrico, attacco, 751  
     differenziale, 750-751  
     lineare, 751  
     temporizzazione, attacco, 751  
 Crittografia a chiave pubblica, 752-755  
     algoritmi curva ellittica, 755  
     algoritmo El Gamal, 755  
     algoritmo RSA, 753-755  
 Crittografia a chiave simmetrica, 737-751  
     AES, 741-745  
     DES, 738-741  
     modalità cipher, 745-751  
     Rijndael, 743-745  
 Crittografia dei collegamenti, 723  
 Crittografia quantistica, 731-734  
 Crittografia, 724-755  
     AES, 741-745  
     chiave pubblica, 752-755  
     chiave simmetrica, 737-751  
     crittoanalisi, 726  
     DES, 738-741  
     introduzione, 725-727  
     modalità di cifratura, 745-750  
     one-time pad, 730-731  
     principio di Kerckhoff, 726  
     quantum, 731-735  
     Rijndael, 743-745  
     testo cifrato, 725  
     testo normale, 725  
     tradizionale, 727-730  
 Crittologia, 726  
 CRL (*vedere* Certificate Revocation List)  
 Crominanza, 694  
 CSMA (*vedere* Carrier Sense Multiple Access)  
 CSMA/CA (*vedere* in Carrier Sense Multiple Access)  
 CSMA/CD (*vedere* in Carrier Sense Multiple Access)  
 CTS (*vedere* Clear To Send)  
 Cyclic Redundancy Check, 196

**D**

Daemen, Joan, 742  
 Daemon Internet, 533  
 Daemon per assenze, 610  
 Daemon, 590  
 D-AMPS (*vedere* Digital AMPS)  
 Data Encryption Standard, 738-741, 751  
     DES triplo, 740-741  
     modalità EDE, 741  
     modalità EEE, 741  
 Data Over Cable Service Interface Specification, 173  
 Datagramma, 346  
 Dati urgenti, 535  
 Davide e Golia, 589  
 Davies, Donald, 51  
 DB, 674  
 DCF (*vedere* Distributed Coordination Function)  
 DCF InterFrame Spacing, 299  
 DCT (*vedere* Discrete Cosine Transformation)  
 DDoS, attacco (*vedere* Distributed Denial of Service, attacco)  
 Deadlock, rete di Petri, 232  
 Decibel, 89  
 Del compleanno, attacco, 763-765, 782  
 Demultiplexing, 31  
 Denial of Service, attacco, 778  
 Dense Wavelength Division Multiplexing, 267  
 DES (*vedere* Data Encryption Standard)  
 D-frame, MPEG, 701, 703  
 DHCP (*vedere* Dynamic Host Configuration Protocol)  
 Diagramma a costellazione, 128  
 Differential Pulse Code, 142  
 DIFS (*vedere* DCF InterFrame Spacing)  
 Digital AMPS, 157-159, 665  
 Digital Millennium Copyright Act, 827-828  
 Digital Subscriber Line Access Multiplexer, 134  
 Digital Subscriber Line, 130-134  
     asimmetrica, 130-134  
 Diagramma, 728  
 Direct Sequence Spread Spectrum, 102, 294  
 Direttive, HTML, 630  
 DIS (*vedere* Draft International Standard)  
 Discrete Cosine Transformation, 698-700  
 Discrete MultiTone, 132  
 Disk farm, 708  
 Dispersione cromatica, 95  
 Dispersione di spettro,  
     802.11, 294-295  
     salto di frequenza, 102, 294  
     sequenza diretta, 102, 294  
     storia, 102  
 Dispositivo Set-top, 705

Disposizione, posta elettronica, 590  
 Distance Vector Multicast Routing Protocol, 712-713  
 Distance vector, algoritmo di routing, 357-360  
 Distanza di Hamming, 193  
 Distorsione, 125  
 Distributed Coordination Function, 296, 298-299  
 Distributed Denial of Service, attacco, 778  
 DIX Ethernet, 275-276, 278  
 DMCA (*vedere* Digital Millennium Copyright Act)  
 DMT (*vedere* Discrete MultiTone)  
 DNS (*vedere* Domain Name System)  
 DNS security, 809-811  
 DNS sicuro, 809-811  
 DNSsec (*vedere* DNS security)  
 DOCSIS (*vedere* Data Over Cable Service Interface Specification)  
 Documenti Web dinamici, 643-651  
 Documenti Web statici, 623-643  
 Documento Web,  
     dinamico, 643-651  
     statico, 623-643  
 Domain Name System, 54, 579-588  
     record autorevole, 588  
     server dei nomi, 586-588  
     sicurezza, 809-811  
     sicurezza, 809-811  
     spazio dei nomi, 580-582  
     spoofing, 806-808  
     zona, 586  
 Dominio di collisione, Ethernet, 282  
 Dominio di primo livello, 580  
 Dominio, primo livello, 580  
 Doppino, 91-92  
 Dorsale multicast, 711-714  
 DoS, attacco (*vedere* Denial of Service, attacco)  
 Dottrina dell'utilizzo equo, 827  
 Draft International Standard, 75  
 Draft Standard, 77  
 Drop cable, 272  
 DSLAM (*vedere* Digital Subscriber Line Access Multiplexer)  
 DSSS (*vedere* Direct Sequence Spread Spectrum)  
 DVMRP (*vedere* Distance Vector Multicast Routing Protocol)  
 DWDM (*vedere* Dense Wavelength Division Multiplexing)  
 Dynamic Host Configuration Protocol, 453-454

**E**

ECB (*vedere* modalità Electronic Code Book)  
 E-commerce, 5  
 EDGE (*vedere* Enhanced Data rates for GSM Evolution)

EIFS (*vedere* Extended InterFrame Spacing)  
 Eisenhower, Dwight, 51  
 Elaborazione pipeline, 217  
 Elaborazione rapida delle TPDU, 566-569  
 Elefanti, apocalisse degli, 46-47  
 Elemento di comutazione, 19  
 E-mail (*vedere* Posta elettronica)  
 Emoji, 669  
 Emoticon, 588  
 Encapsulating Security Payload, 775-776  
 Enhanced Data rates for GSM Evolution, 168  
 Entità di trasporto, 482  
 ESP (*vedere* Encapsulating Security Payload)  
 Eternity service, 823  
 Ethernet classica, 286, 327  
 Ethernet commutata, 281-283, 328-336  
 Ethernet fast commutata, 286  
 Ethernet, 17, 65-68, 271-292 (*vedere anche* Fast Ethernet, Gigabit Ethernet)  
     10Base-F, 273  
     10Base-T, 272  
     backoff esponenziale binario, 278-279  
     broadcast, 276  
     cablaggio, 271-274  
     classica, 327  
     codifica Manchester, 274-275  
     commutata, 328-336  
     DIX, 67, 275-276, 278  
     dominio di collisione, 282  
     drop cable, 272  
     fast, 283-285  
     formato del frame, 276  
     Gigabit, 286-290  
     multicast, 276  
     presa a vampiro, 271  
     prestazioni, 279-281  
     protocollo, 275-279  
     retrospettiva, 291-292  
     ripetitore, 274  
     storia, 65-67  
     thick, 271  
     thin, 271  
 Etichetta, HTML, 630  
 Extended HTML, 642-643  
 Extended HTML, Basic, 673  
 Extended Hypertext Markup Language, 642  
 Extended InterFrame Spacing, 299  
 Extensible Markup Language, 639-642  
 Extensible Style Language, 639-642  
 Exterior Gateway Protocol, 427, 454

**F**

FAQ (*vedere* Frequently Asked Questions)

Fascio spot, 112  
 Fast Ethernet, 283-286  
     100Base-FX, 285  
     100Base-T4, 285  
     100Base-TX, 284-285  
     4B/5B, 285  
     8B/6T, 285  
     autonegoziazione, 286  
     cablaggio, 284-286  
     full-duplex, 285  
     hub, 285-286  
     switch, 286  
 Fattore lavoro, 727  
 FDD (*vedere* Frequency Division Duplexing)  
 FDDI (*vedere* Fiber Distributed Data Interface)  
 FDM (*vedere* Frequency Division Multiplexing)  
 FEC (*vedere* Forwarding Equivalence Class)  
 Felten, Edward, 827  
 FHSS (*vedere* Frequency Hopping Spread Spectrum)  
 Fiber Distributed Data Interface, 283  
 Fiber To The Curb, 709-710  
 Fiber To The Home, 710  
 Fibra monomodale, 94  
 Fibra multimodale, 94  
 Fibra ottica, 93-99  
 Fibre channel, 283  
 Fibre ottiche, 93-99  
     confronto con il cavo, 98-99  
     confronto con il satellite, 117-118  
     dispersione cromatica, 95  
     monomodale, 94  
     multimodale, 94  
     solitoni, 96  
 File server Internet di esempio, 488-492  
 File system sicuro, 811  
 File Transfer Protocol, 448, 624  
 Filtro di pacchetto, 777  
 Filtro, posta elettronica, 609  
 Finestra di congestione, TCP, 548-550  
 Finestra di invio, 212  
 Finestra di ricezione, 212  
 Firewall, 776-779  
     filtro a pacchetti, 777  
     gateway, 778  
 Firma del codice, 817  
 Firma digitale, 755-765  
     attacco del compleanno, 763-764  
     chiave pubblica, 757-759  
     chiave simmetrica, 756-757  
     message digest, 759-762  
 Flash crowd, 660  
 Flooding selettivo, 355  
 Flusso, 397

foglio di stile, HTML, 634  
 Ford-Fulkerson, algoritmo di routing, 357-360  
*Forwarding Equivalence Class*, 416

Fotone, 732

Frame

dati, 184

video, 692

Frame bursting, gigabit Ethernet, 288

Frame dati, 38

Frame di acknowledgement, 38

Frame di segnalazione, 298

Frame informativo, 235

Frame non numerato, 235

Frame relay, 61

Frame supervisore, 235

Frammentazione, internetwork, 427-431

Frequency Division Duplexing, 307

Frequency Division Multiplexing, 137-140

Frequency Hopping Spread Spectrum, 102, 294  
 tempo di ciclo, 294

Frequency Shift Keying, 126

Frequently Asked Questions, 610

Frequenza, 100

FTP (*vedere* File Transfer Protocol)

FTTC (*vedere* Fiber To The Curb)

FTTH (*vedere* Fiber To The Home)

Fuzzball, 54

## G

Gatekeeper, H.323, 686

Gateway applicativo, 778

Gateway, 25, 326-328

H.323, 686

internetwork, 422

General Packet Radio Service, 168

Generatore polinomiale, 197-200

Generazione dinamica di pagine Web, 643-651

GEO (*vedere* Geostationary Earth Orbit, satellite)

Geostationary Earth Orbit, satellite, 109-113

Gerarchia di protocollo, 26-30

Gerarchico, algoritmo di routing, 366-368

Gestione del timer, TCP, 550-553

Gestione del Token, 40

Gigabit Ethernet, 286-290

10 Gbps, 290

1000Base-CX, 288

1000Base-LX, 288

1000Base-SX, 288

1000Base-T, 288

8B/10B, 289

cablaggio, 288

estensione della portante, 287-288

frame bursting, 288

modalità operativa, 287  
 UTP, 288

Global Positioning System, 114

Global System for Mobile Communications, 159-162

Globalstar, 115-116

Gopher, 624

GPRS (*vedere* General Packet Radio Service)

GPS (*vedere* Global Positioning System)

Gray, Elisha, 119

Gruppo, 138

GSM (*vedere* Global System for Mobile communications)

## H

H.323, 683-689 (*vedere anche* Voice over IP)  
 gatekeeper, 686

Handoff, 155

hard, 155

soft, 155

Handshake a tre vie, 499-502, 539-540

Hardware di rete, 14-26

Hashed Message Authentication Code, 775, 790

HDLC (*vedere* High-Level Data Link Control)

HDTV (*vedere* High Definition TeleVision)

Headend, 18, 169

Header prediction, 568

Hertz, 100

Hertz, Heinrich, 100

HFC (*vedere* Hybrid Fiber Coax)

High Definition TeleVision, 694-695

High Rate Direct Sequence Spread Spectrum, 295

High-Level Data Link Control, 234-237

HMAC (*vedere* Hashed Message Authentication Code)

Host mobile, 372

Host mobile, algoritmo di routing, 372-375

Host, 19

HTML (*vedere* HyperText Markup Language)

HTML dinamico, 647

HTTPS (*vedere* Secure HTTP)

Hub, 112, 272, 326-327

Hybrid Fiber Coax, 170

HyperText Markup Language, 615, 629-639

attributo, 630

cella, 630

collegamento ipertestuale, 632-633

direttiva, 630

foglio di stile, 634

intestazione, 630

modulo, 634-638

tabella, 633-634

tag, 630

titoli, 630

titolo, 630

Hypertext Transfer Protocol, 41, 623, 651-656

connessione permanente, 652

connessione, 652

esempio di utilizzo, 656

intestazione dei messaggi, 654-656

intestazione di richiesta, 654-656

intestazione di risposta, 654-656

metodo, 652-654

Hz (*vedere* Hertz)

## I

IAB (*vedere* Internet Architecture Board)

ICANN (*vedere* Internet Corp. for Assigned Names and Numbers)

ICMP (*vedere* Internet Control Message Protocol)

IDEA (*vedere* International Data Encryption Algorithm)

Identificatore di nodo, 381

Idioma, umano, 676

IEEE (*vedere* Institute of Electrical and Electronics Engineers)

IEEE 802.11, 292-302

802.11a, 294-295

802.11b, 295

802.11g, 295

burst di frammenti, 298

Clear To Send, 297-298

codice di Walsh-Hadamard, 295

competizione con WAP, 673

confronto con 802.16, 303-304

CSMA/CA, 296-297

DCF InterFrame Spacing, 299

Direct-Sequence Spread Spectrum, 294

Distributed Coordination Function, 296, 298-299

Extended InterFrame Spacing, 299

formato del frame, 299-300

frame di segnalazione, 298

Frequency Hopping Spread Spectrum, 294

multiplexing a divisione di frequenza

ortogonale, 294-295

PCF InterFrame Spacing, 299

pila di protocolli, 292-293

Point Coordination Function, 296, 298-299

problema della stazione esposta, 296

problema della stazione nascosta, 296

protocollo del sottostrato MAC, 295-299

richiesta di invio, 297-298

sequenza di Barker, 294

Short InterFrame Spacing, 299

sicurezza, 781-783

spettro High Rate Direct-Sequence Spread, 295

strato fisico, 293-295

tempo di rotazione, 294

vettore di allocazione della rete, 297

wired equivalent privacy, 300

IEEE 802.11a, 294-295

IEEE 802.11b, 295

IEEE 802.11g, 295

IEEE 802.12, 283

IEEE 802.15 (*vedere* Bluetooth)

IEEE 802.16, 135, 302-310

classi di servizio, 308-309

confronto con 802.11, 303-304

Frequency Division Duplexing, 307

servizio a velocità costante, 308

servizio a velocità variabile in tempo reale, 308

servizio Best effort, 308

sicurezza, 307-308

sottostrato MAC, 307-309

stack di protocolli, 305-306

strato fisico, 306-307

struttura del frame, 309-310

Time Division Duplexing, 307

IEEE 802.1Q, 333-336

IEEE 802.2, 290-291

IEEE 802.3u, 284

IETF (*vedere* Internet Engineering Task Force)

I-frame, MPEG, 701, -702

IGMP (*vedere* Internet Group Management Protocol)

IGP (*vedere* Interior Gateway Protocol)

IKE (*vedere* Internet Key Exchange)

ILEC (*vedere* Incumbent LEC)

IMAP (*vedere* Internet Message Access Protocol)

Imbiancamento, 740

I-mode, 665-670

accettazione in occidente, 667

cHTML, 668-670

confronto con WAP, 671

emoji, 669

fatturazione, 666

handset, 667

modello commerciale, 666

servizi ufficiali, 666

servizi, 666

stack di protocolli, 668

struttura del software, 667-668

IMP, 52

Impronta, 112

Improved Mobile Telephone System, 153

IMT (*vedere* International Mobile Telecommunications)

IMTS (*vedere* Improved Mobile Telephone System)

Incumbent LEC, 134

Indicatore inferiore, 682

Indicatore superiore, 682

Indirizzamento di trasporto, 493-496

Indirizzamento mediante classe, 437

Indirizzamento, 31  
 Indirizzo  
   IP, 436-438, 441-444  
   trasporto, 493-496  
 Indirizzo IPv4, 436-448  
   classe A, B, C, D, 437  
   con classe, 437  
   intestazione, 433-436  
   opzioni, 436  
   privato di classe, 441-444  
   subnet mask, 439-441  
 Indirizzo temporaneo, 463  
 Inetd, 533  
 Information-mode, 665-670  
 Initialization Vector, 746  
 Inoltro accelerato, 413-414  
 Inoltro sicuro, 414-415  
 Inoltro, 350  
 Institute of Electrical and Electronics Engineers, 75  
 InterExchange Carrier, 122  
 Interfaccia, 27  
 Interfaccia, media player, 680  
 Interface Message Processor, 52  
 Interferometro, 97, 266  
 Interior Gateway Protocol, 427, 454  
 Intermediate System-Intermediate System, 365-366  
 International Data Encryption Algorithm, 751, 799  
 International Mobile Telecommunications, 166  
 International Standard, 75  
 International Standards Organization, 74-75  
 International Telecommunication Union, 72-74  
 Internet Activities Board, 75  
 Internet Architecture Board, 75  
 Internet Control Message Protocol, 449-450  
 Internet Corp. for Assigned Names and Numbers (ICANN), 437  
 Internet Engineering Task Force, 76  
 Internet Group Management Protocol, 462, 713  
 Internet Key Exchange, 773  
 Internet Message Access Protocol, 608-609  
   confronto con POP3, 609  
 Internet protocol (IP), 432-444, 464-473 (*vedere anche* IPv4 e IPv6)  
 Internet Research Task Force, 76  
 Internet Security Association e Key Management Protocol, 773  
 Internet Service Provider, 57  
 Internet Society, 77  
 Internet via cavo, 170-176  
   confronto con ADSL, 175-176  
 Internet, 25, 50-59, 237-242, 431-473, 524-556  
   architettura Internet, 58-59  
   indirizzo, 436-448

introduzione, 56-58  
 principi di progettazione, 431-432  
 storia, 50-56  
 strato di rete, 431-473  
 Internetwork, 25-26  
 Internetworking senza connessione, 423-425  
 Internetworking, 418-431  
   circuiti virtuali, 422-423  
   frammentazione, 427-431  
   locale, 322-323  
   routing, 426-427  
   senza connessione, 423-425  
   tunneling, 425-426  
 Interrogazione ricorsiva, 588  
 Interworking tra reti, 418-431  
 Intestazione del frame, 201-203  
 Intestazione di richiesta, 654  
 Intestazione di risposta, 654  
 Intestazione estesa, 469  
 Intestazione If-modified-since, 659  
 Intestazione, 29  
   Ethernet, 275-276  
   frame, 201-203  
   pacchetto IPv4, 433-436  
   pacchetto IPv6, 466-469  
   posta elettronica, 591  
   segmento TCP, 536-539  
 Intranet, 59  
 Intruso, 725  
 Involturo, posta elettronica, 591  
 IP Security, 772-776  
   Authentication Header, 774-775  
   Encapsulating Security Payload, 775-776  
   modalità di trasporto, 773  
   modalità tunnel, 773  
 IP, 432-444, 464-473 (*vedere anche* IPv4 e IPv6)  
 Ipertesto, 612  
 Ipotesi di tempo continuo, 250  
 IPsec (*vedere* IP security)  
 IPv4, 432-444  
 IPv5, 433  
 IPv6, 464-473  
   controversie, 471-473  
   intestazione principale, 466-469  
   intestazioni estese, 469-471  
 Iridium, 114-116  
 IRTF (*vedere* Internet Research Task Force)  
 IS (*vedere* International Standard)  
 ISAKMP (*vedere* Internet Security Association and Key Management Protocol)  
 IS-IS (*vedere* Intermediate System-Intermediate System)  
 ISM (*vedere* Bande Industrial, Scientific, Medical)

ISO, 74  
 ISP (*vedere* Internet Service Provider)  
 ITU (*vedere* International Telecommunication Union)  
 IV (*vedere* Initialization Vector)  
 IXC (*vedere* InterExchange Carrier)  
**J**  
 Japanese Telephone and Telegraph Company, 665-670  
 Java Virtual Machine, 650, 817  
 JavaScript, 647-651  
 JavaServer Page, 646  
 Jitter, 395-396  
 Joint Photographic Experts Group, 697  
 JPEG (*vedere* Joint Photographic Experts Group)  
 JSP (*vedere* JavaServer Page)  
 Jumbogramma, 470, 472  
 JVM (*vedere* Java Virtual Machine)  
**K**  
 KDC (*vedere* Key Distribution Center)  
 Kerberos, 796-798  
 Key Distribution Center, 785  
 Key escrow, 820  
 Keystream reuse, attacco, 749  
 Keystream, 748  
 Knudsen, Lars, 742  
**L**  
 Lamarr, Hedy, 102  
 LAN (*vedere* Local Area Network)  
 LAP (*vedere* Link Access Procedure)  
 LAPB (*vedere* Link Access Procedure B)  
 Last modified, intestazione, 658-659  
 LATA (*vedere* Local Access and Transport Area)  
 LCP (*vedere* Link Control Protocol)  
 LDAP (*vedere* Lightweight Directory Access Protocol)  
 Leasing, 454  
 Leaky bucket, algoritmo, 400-403  
 LEC (*vedere* Local Exchange Carrier)  
 Legge di Zipf, 706  
 LEO (*vedere* Low Earth Orbit, satellite)  
 Lettore multimediale, 680-683  
 Libertà di parola, 822-826  
 Lightweight Directory Access Protocol, 588  
 Lightweight Transport Protocol, 667  
 Limited-contention, protocollo, 261-265  
 Linea di trasmissione, 19  
 Linea full duplex, 129  
 Linea half duplex, 129  
**M**  
 MACA (*vedere* Multiple Access with Collision Avoidance)  
 MACA for Wireless, 270-271  
 MACAW (*vedere* MACA for Wireless)  
 Macchina a stati finiti  
   modello, 229  
   protocollo stop-and-wait, 229-232  
   TCP, 541-543  
 Macchina di Protocollo, 229  
 Macroblocco, 702  
 MAHO (*vedere* Mobile Assisted HandOff)  
 Mailing list, 591  
 MAN (*vedere* Metropolitan Area Network)  
 MAN Wireless (*vedere* IEEE 802.16)  
 MANET (*vedere* Mobile Ad hoc NETwork)  
 Man-in-the-middle, attacco, 792  
 Mantenimento del percorso, reti ad hoc, 380-384  
 Mappa di bit elementare, protocollo, 259-260  
 Marconi, Guglielmo, 21  
 MARS, 742  
 Marshaling, parametro, 527-529  
 Maschera di sottorete, 439  
 Maschera, percettiva, 677  
 Mascheramento delle frequenze, 677  
 Mascheramento temporale, 677  
 Mastergroup, 138

Matsunaga, Mari, 665  
 Maximum Transmission Unit, 535  
 Maxwell, James Clerk, 66  
 MBone, 711-714  
 M-commerce, 11  
 MD5, 760  
 Medium Access Control, sottostrato, 247-342 (*vedere anche* Sottostrato MAC)  
 Medium Earth Orbit, satellite, 113-114  
 MEO (*vedere* Medium Earth Orbit, satellite)  
 Merkle, Ralph, 755  
 Message digest, 759-762  
 MD5, 760  
 SHA-1, 761-762  
 Messaggio di credito, 522  
 Messaggistica immediata, 7  
 Metafile, 680  
 Metcalfe, Bob, 23, 66  
 Metodo, 652  
 Metropolitan Area Network, 18  
 wireless (*vedere* IEEE 802.16)  
 Mezzo di comunicazione, 5  
 Mezzo di trasmissione guidato, 90-99  
 Mezzo fisico, 27  
 MFJ (*vedere* Modified Final Judgment)  
 Michelson-Morley, esperimento, 66  
 Microcella, 154  
 Middleware, 2  
 MIME (*vedere* Multipurpose Internet Mail Extensions)  
 Minislot, 174  
 Misurazione delle prestazioni di rete, 560-562  
 MMDS (*vedere* Multichannel Multipoint Distribution Service)  
 Mobile Ad hoc NETwork, 375  
 Mobile Assisted HandOff, 159  
 Mobile IP, 462-464  
 Mobile Switching Center, 155  
 Mobile Telephone Switching Office, 155  
 Mobile wireless, 10  
 Mobile-commerce, 11  
 Mockapetris, Paul, 47  
 Modalità contatore, 749-750  
 Modalità di cifratura, 745-750  
 Modalità di trasporto, IPsec, 773  
 Modalità EDE, DES, 741  
 Modalità EEE, DES, 741  
 Modalità Electronic Code Book, 745-746  
 Modalità promiscua, 319  
 Modalità stream cipher, 748-749  
 Modalità Tunnel, IPsec, 773  
 Modellazione del traffico, 399-400  
 Modello client-server, 4-5

Modello di gestione della connessione, 541-543  
 macchina a stati, 486  
 Modello di riferimento ISO/OSI, 37  
 Modello di riferimento OSI, 37-41  
 critiche, 46-48  
 rispetto a TCP/IP, 44-46  
 Modello di riferimento TCP/IP, 41-44  
 confronto con OSI, 44-46  
 critiche, 48-49  
 Modello di riferimento, 37-49  
 confronto, 44-46  
 OSI, 37-41  
 TCP/IP, 41-44  
 Modello di stazione, 249  
 Modem via cavo, 173-175  
 Modem, 125-130  
 Modified Final Judgment, 122  
 Modulazione delta, 143  
 Modulazione di ampiezza, 126  
 Modulazione di fase, 126  
 Modulazione di frequenza, 126  
 Modulazione, 142  
 ampiezza, 126  
 delta, 143  
 fase, 126  
 frequenza, 126  
 QAM, 127  
 quadratura, 127  
 trellis-coded, 128  
 Modulo  
 HTML, 634-638  
 PHP, 645-646  
 Web, 634-638  
 Mosaic, 611  
 MOSPF (*vedere* Multicast Open Shortest Path First, routing)  
 Motion Picture Experts Group, 700  
 MP3, 676-679  
 MPEG (*vedere* Motion Picture Experts Group)  
 MPLS (*vedere* MultiProtocol Label Switching)  
 MPLS guidato dai dati, 417  
 MPLS guidato dal controllo, 417  
 Mrouter, 711  
 MSC (*vedere* Mobile Switching Center)  
 MTSO (*vedere* Mobile Telephone Switching Office)  
 MTU (*vedere* Maximum Transmission Unit)  
 Multicast Open Shortest Path First, routing, 714  
 Multicast, 276  
 Multicasting Internet, 461-462  
 Multicasting, 15, 370  
 Internet, 461-462, 712-714  
 Multichannel Multipoint Distribution Service, 135  
 Multimedia, 674-714

audio digitale, 674-676  
 audio in streaming, 679-683  
 audio, 674-692 (*vedere anche* Audio)  
 compressione audio, 676-679  
 compressione video, 696-704  
 introduzione al video, 692-696  
 lettore multimediale, 680-683  
 MBone, 711-714  
 MP3, 676-679  
 radio Internet, 683-685  
 RTSP, 680-683  
 server multimediale, 681-683  
 telefonia Internet, 685-692  
 video su richiesta, 704-711  
 Voice over IP, 685-692 (*vedere anche* Voice over IP)  
 Multipath fading, 69, 104  
 Multiple Access with Collision Avoidance, 269-270  
 Multiplex ascendente, 510  
 Multiplexing discendente, 511  
 Multiplexing, 31, 510-511  
 ascendente, 510  
 discendente, 511  
 MultiProtocol Label Switching, 415-417  
 Multipurpose Internet Mail Extensions, 596-602

**N**

Name server, 495  
 DNS, 586-588  
 NAP (*vedere* Network Access Point)  
 NAT (*vedere* Network Address Translation)  
 National Institute of Standards and Technology, 75, 741  
 National Security Agency, 740  
 National Television Standards Committee, 694  
 NAV (*vedere* Network Allocation Vector)  
 Navajo, 724  
 NCP (*vedere* Network Control Protocol)  
 Near video on demand, 704  
 Negoziazione, 32  
 Network access point, 56  
 Network address translation, 444-448  
 Network allocation vector, 297  
 Network control protocol, 239, 242  
 Network Interface Device, 133  
 Network news transfer protocol, 624  
 Network service access point, 494  
 News, 57  
 NID (*vedere* Network Interface Device)  
 NIST (*vedere* National Institute of Standards and Technology)  
 NNTP (*vedere* Network News Transfer Protocol)  
 Nodo in fibra, 170

Non ripudio, 756  
 Nonce, 786  
 Notazione del punto decimale, 437  
 NSA (*vedere* National Security Agency)  
 NSAP (*vedere* Network Service Access Point)  
 NSFNET, 55  
 NTSC (*vedere* National Television Standards Committee)  
 NTT DoCoMo, 665-670  
 Nyquist, Henry, 89

**O**

Occupazione del canale verificabile, 250  
 OFDM (*vedere* Orthogonal Frequency Division Multiplexing)  
 Olsen, Ken, 6  
 Onda a infrarossi, 106-107  
 Onda millimetrica, 106-107  
 ONU (*vedere* Optical Network Unit)  
 Optical Network Unit, 709  
 Orthogonal Frequency Division Multiplexing, 294  
 Oryctolagus cuniculus, 28  
 OSPF (protocollo Open Shortest Path First)

**P**

Pacchetto choke, 391-394  
 Pacchetto, 15  
 Pacchetto, commutazione, 150-151, 344  
 Pacchetto, pianificazione, 408-409  
 Pagina Web stantia, 658  
 Pagina, Web (*vedere* Web)  
 Pagine Web dinamiche lato client, 647-651  
 Pagine Web dinamiche lato Server, 643-647  
 PAL (*vedere* Phase Alternating Line)  
 PAR (*vedere* Positive Acknowledgement with Retransmission)  
 Passe partout, 784  
 P-box, 737  
 PCF (*vedere* Point Coordination Function)  
 PCF InterFrame Spacing, 299  
 PCM (*vedere* Pulse Code Modulation)  
 PCS (*vedere* Personal Communications Services)  
 Peer to peer, 7-9  
 PEM (*vedere* Privacy Enhanced Mail)  
 Percorso di certificazione, 770  
 Percorso più breve di Dijkstra, algoritmo, 353-355  
 Percorso più corto, 353  
 Perlman, Radia, 324  
 Personal area network, 15  
 Personal Communications Services, 157  
 P-frame, MPEG, 701-702  
 PGP (*vedere* Pretty Good Privacy)  
 Phase Alternating Line, 694

PHP (*vedere PHP Hypertext Preprocessor*)  
 PHP Hypertext Preprocessor, 645–646  
 Piconet, 311  
 PIIFS (*vedere PCF InterFrame Spacing*)  
 Pila di protocolli, 28  
     802.11, 292–293  
     Bluetooth, 313–314  
     H.323, 687  
     IEEE 802.16, 305–306  
     I-mode, 668  
     OSI, 39  
     TCP/IP, 43  
     WAP 1.0, 664  
     WAP 2.0, 672  
 PIM (*vedere Protocol Independent Multicast*)  
 Pixel, 695  
 PKI (*vedere Public Key Infrastructure*)  
 Plain Old Telephone Service, 132  
 Plug-in, 616  
 Point Coordination Function, 296, 298–299  
 Point of Presence, 58, 122  
 Point-to-Point Protocol, 238–242  
 Policing del traffico, 400  
 Politica del latte, 394  
 POP (Post Office Protocol) 3, 605–608  
     POP confronto con IMAP, 609  
 POP (*vedere Point of Presence*)  
 Porta di destinazione, 447  
 Porta nota, 533  
 Porta sorgente, 447  
 Porta, 494, 533  
 Portale, 70  
 Portante E1, 142  
 Portante onda sinusoidale, 126  
 Portante T1, 140–143, 709  
 Portanti T2–T4, 143  
 Positive Acknowledgement with Retransmission, 209  
 Posizione home, 373  
 Posta elettronica, 5, 57, 588–611  
     agente di trasferimento, 590  
     agente utente, 591–594  
     architettura e servizi, 579–591  
     armatura ASCII, 598  
     base64, codifica, 598  
     casella di posta, 591  
     composizione, 590  
     consegna finale, 605–611  
     disposizione, 590  
     filtraggio, 609  
     formato dei messaggi, 594–602  
     funzioni di consegna, 609–610  
     generazione di rapporti, 590  
     intestazioni, 595–596  
     invio, 592–593

lettura, 593–594  
 MIME, 596–602  
 POP3, 605–608  
 profilo utente, 593  
 quoted printable, 598  
 SMTP, 602–605  
 trasferimento dei messaggi, 602–605  
 visualizzazione, 590  
 X.400, 589–590  
 POTS (*vedere Plain Old Telephone Service*)  
 PPP (*vedere Point-to-Point Protocol*)  
 Prenotazione della risorsa, 405–406  
 Pretty Good Privacy, 799–804  
     chiavi, 802  
     formato del messaggio, 802  
     funzionamento, 801  
     IDEA, 799  
 Prima generazione, telefono cellulare, 153–157  
 Primitiva di servizio, 34–36  
 Primitiva, 34  
 Principale, 731  
 Principi crittografici, 735–736  
     freschezza, 736  
     Kerckhoff, 726  
     ridondanza, 735–736  
 Principi di progettazione, Internet, 431–432  
 Principio di Kerckhoff, 726  
 Principio di ottimalità, 352–353  
 Privacy Enhanced Mail, 803–804  
 Problema dei due eserciti, 503–504  
 Problema dei tre orsi, 441  
 Problema del conto all’infinito, 359–360  
 Problema della stazione esposta, 269  
 Problema della stazione nascosta, 269  
 Problema di allocazione del canale, 248–251, 337  
 Problemi di prestazioni, 557–573  
 Problemi di progettazione, 30–31  
     strato data link, 184–192  
     strato network, 343–350  
     strato trasporto, 492–513  
 Problemi sociali, 12–14, 819–828  
 Procedura Perl, 644  
 Prodotto banda-ritardo, 559  
 Produzione di relazioni, e-mail, 590  
 Profilo utente, e-mail, 593  
 Profilo, Bluetooth, 312–313  
 Progetto del sistema  
     per alte prestazioni, 566–569  
 Programmazione del Socket, 488–492  
 Proprietà intellettuale, 826  
 Protezione ActiveX, 817–818  
 Protezione delle applet Java, 817  
 Protezione del codice mobile, 816–819  
 Protezione di JavaScript, 818  
 Protocol Independent Multicast, 714  
 Protocolli ad accesso multiplo, 251–270  
 Protocolli di controllo Internet, 449–454  
 Protocolli di data link di esempio, 204–228  
 Protocolli di data link, 200–228, 234–242  
     elementari, 200–211  
     HDLC, 234–237  
     Internet, 238–242  
     sliding window, 211–228  
 Protocolli gigabit, 569–570  
 Protocolli Internet (*vedere Protocollo*)  
 Protocolli privi di collisioni, 259–270  
 Protocollo 1 (utopia), 204–206  
 Protocollo 2 (stop-and-wait), 206–211  
 Protocollo 3 (PAR), 208–211  
 Protocollo 4 (finestra scorrevole), 211–216  
 Protocollo 5 (torna indietro di n), 216–223  
 Protocollo 6 (ripetizione selettiva), 223–22  
 Protocollo BOOT, 453  
 Protocollo di autenticazione Otway-Rees, 795–796  
 Protocollo di connessione iniziale, 495  
 Protocollo di prenotazione, 260  
 Protocollo di ripetizione selettiva, 223–228  
 Protocollo di telefonia G.723.1, 687  
 Protocollo di telefonia H.225, 687  
 Protocollo di telefonia H.245, 687  
 Protocollo di telefonia Q.931, 687  
 Protocollo di trasporto di esempio, 513–524  
 Protocollo di trasporto, 492  
     codice C, 518–521  
     entità di trasporto, 515–522  
     esempio, 513–524  
     macchina a stati finiti, 522–524  
     primitive, 513–515  
 Protocollo in tempo reale, 529–532  
 Protocollo Indietro di n, 216–223  
 Protocollo LAN Wireless, 265–270  
 Protocollo multicast  
     DVMRP, 712  
     MOSPF, 714  
     PIM, 714  
     PIM-DM, 714  
     PIM-SM, 714  
 Protocollo OSPF (Open Shortest Path First), 454–459  
 Protocollo sliding window, 211–228  
     1-bit, 211–214  
     ripetizione selettiva, 223–228  
     torna indietro di n, 216–223  
 Protocollo stop-and-wait, 206–211  
 Protocollo tree-walk, 263–265  
 Protocollo, 27  
     AODV, 375  
 ARP, 450–452  
 ARQ, 209–211  
 BGP, 459–461  
 BOOTP, 453  
 CSMA, 255–255  
 CSMA/CA, 296–297  
 CSMA/CD, 257–258  
 DHCP, 453–454  
 DVMRP, 712–713  
 Ethernet, 275–279  
 FTP, 448  
 H.323, 683–689  
 HDLC, 234–237  
 HTTP, 41, 623, 651–656  
 ICMP, 449–450  
 IGMP, 462  
 IKE, 773  
 IMAP, 608–609  
 IPv4, 433–444  
 IPv6, 464–473  
 ISAKMP, 773  
 IS-IS, 365–366  
 LAP, 234  
 LAPB, 234–245  
 LCP, 239–242  
 LDAP, 588  
 LTP, 667  
 MACA, 269–270  
 MACAW, 269–270  
 MOSPF, 714  
 NCP, 239, 242  
 NNTP, 624  
 OSPF, 454–459  
 PAR, 209–211  
 PIM, 714  
 POP3, 605–608  
 PPP, 238–242  
 reverse ARP, 453  
 RSVP, 410–412  
 RTCP, 531–532  
 RTP, 529–532  
 RTSP, 680–683  
 SCTP, 556  
 SDLC, 234  
 SIP, 465  
 SMTP, 602–605  
 SOAP, 642  
 TCP, 532–566  
 UDP, 524–532  
 WAP, 663–665, 670–673  
 WDMA, 265–267  
 WDP, 664  
 Proxy ARP, 452

Proxy, Web, 657–659  
 Psicoacustica, 677  
 PSTN (*vedere* Public Switched Telephone Network)  
 PTT (Post, Telegraph & Telephone Administration), 72  
 PTT (*vedere* Post, Telegraph & Telephone administration)  
 Public Key Infrastructure, 768–770  
 Pulse Code Modulation, 140

**Q**

Q.931, 687  
 QAM (*vedere* Quadrature Amplitude Modulation)  
 QoS (*vedere* Qualità del servizio)  
 QPSK (*vedere* Quadrature Phase Shift Keying)  
 Quadrature Amplitude Modulation, 127, 710  
 Quadrature Phase Shift Keying, 127, 306  
 Qualità del servizio, 32  
     accodamento equo, 408–409  
     algoritmi basati sul flusso, 409–412  
     algoritmo token bucket, 402–405  
     algoritmo leaky bucket, 400–403  
     commutazione di etichetta, 415–417  
     controllo di accesso, 406–408  
     inoltro accelerato, 413–414  
     inoltro sicuro, 414–415  
     memorizzazione nei buffer, 399  
     modellazione del traffico, 399–400  
 MPLS, 415–417  
     pianificazione dei pacchetti, 408–409  
     prenotazione delle risorse, 405–406  
     requisiti, 397–398  
     routing proporzionale, 408  
 RSVP, 410–412,  
     servizi differenziati, 412–415  
     servizi integrati, 409–412  
     servizio basato sulle classi, 412–415  
     specifica di flusso, 407  
     strato rete, 397–417  
     tecniche per ottenerla, 398–409  
 Quantizzazione, JPEG, 698  
 Qubit, 733

**R**

Radio Internet, 683–685  
 RAID (*vedere* Redundant Array of Inexpensive Disks)  
 Random Early Detection, 395  
 Rapporto segnale-rumore, 89  
 RARP (*vedere* Reverse ARP)  
 RAS (*vedere* Registration/Admission/Status channel)  
 RC4, 751, 781, 815

RC5, 751  
 RC6, 742  
 Realm, Kerberos, 797  
 Real-time control protocol, 531–532  
 Real-time streaming protocol, 680–683  
 Real-time Transport Control Protocol, 531–532, 687  
 Real-time Transport Protocol, 529–532, 680–683  
 Record autorevole, DNS, 587  
 Record di risorsa, DNS, 582–586  
 Recupero degli errori, multimediale, 681  
 RED (*vedere* Random Early Detection)  
 Redundant Array of Inexpensive Disks, 708  
 Regione, 366  
 Registration/Admission/Status channel, 687  
 Remailer anonimo, 820–821  
 Remailer cypherpunk, 821–822  
 Remote Procedure Call, 526–529  
     ordinamento, 527–529  
     stub client, 527–529  
     stub server, 527–529  
 Replica di Server, Web, 659–660  
 Request For Comments, 76  
 Request to Send, 269  
 Resource Record Set, 809  
 Resource reSerVation Protocol, 410–412  
 Rete (*vedere* rete di computer)  
 Rete ad hoc, 68, 375–380  
 Rete ATM, 61, 65, 417, 418  
 Rete broadcast, 15  
 Rete di computer, 2  
     802.11, 68–71  
     ARPANET, 50–54  
     ATM, 61–65  
     domestica, 6–9, 23–25  
     Ethernet, 65–68  
     gerarchia di protocolli, 26–30  
     hardware, 14–16  
     IEEE 802.11, 68–71  
     modelli di riferimento, 37–49  
     NSFNET, 54–56  
     orientata alla connessione, 59–65  
     software, 26–37  
     standardizzazione, 71–77  
     utilizzo, 3–14  
     wireless, 21–23, 68–71  
     X.25, 61  
 Rete di Petri, 232  
 Rete di transito, 460  
 Rete domestica, 6–9, 23–25  
 Rete geografica, 19–21  
 Rete multiaccesso, 455  
 Rete multiconnessa, 460  
 Rete multimediale, 674–714

Rete privata, 779  
 Rete punto-punto, 15  
 Rete Stub, 460  
 Rete telefonica a commutazione pubblica, 118–151  
 Rete Wireless, 21–23  
 Reti in fibra ottica, 97–98  
 Reverse ARP, 453  
 Revoca, certificato, 771  
 RFC (*vedere anche* Request For Comments)  
 RFC 1034, 580  
 RFC 1048, 453  
 RFC 1058, 360  
 RFC 1084, 453  
 RFC 1106, 539  
 RFC 1112, 462  
 RFC 1122, 532  
 RFC 1323, 532, 539, 570  
 RFC 1379, 556  
 RFC 1424, 803  
 RFC 1519, 442  
 RFC 1550, 465  
 RFC 1644, 556  
 RFC 1661, 238, 241  
 RFC 1662, 239  
 RFC 1663, 239, 240  
 RFC 1700, 435  
 RFC 1771, 461  
 RFC 1889, 529  
 RFC 1939, 605  
 RFC 1958, 431  
 RFC 2045, 597, 599  
 RFC 2060, 608, 609  
 RFC 2109, 626  
 RFC 2131, 453  
 RFC 2132, 453  
 RFC 2141, 625  
 RFC 2205, 409, 410  
 RFC 2210, 407  
 RFC 2211, 407  
 RFC 2246, 816  
 RFC 2251, 588  
 RFC 2326, 682, 680  
 RFC 2328, 455  
 RFC 2362, 714  
 RFC 2401, 772  
 RFC 2410, 772  
 RFC 2440, 800  
 RFC 2459, 767  
 RFC 2460, 466  
 RFC 2535, 809, 811  
 RFC 2597, 414, 415  
 RFC 2616, 651, 654, 659  
 RFC 2617, 785  
 RFC 2632, 804  
 RFC 2806, 669  
 RFC 2821, 605, 715  
 RFC 2822, 594  
 RFC 2993, 448  
 RFC 3003, 600  
 RFC 3022, 445  
 RFC 3023, 599  
 RFC 3119, 681  
 RFC 3168, 549  
 RFC 3174, 762  
 RFC 3194, 469  
 RFC 3246, 413  
 RFC 3261, 689  
 RFC 3280, 767  
 RFC 768, 525  
 RFC 793, 532  
 RFC 821, 589, 594  
 RFC 822, 421, 589, 589, 590, 590, 594, 594, 595, 596, 597, 599, 651, 716, 801, 804, 821  
 RFC 903, 453  
 RFC 951, 453  
 Ricerca inversa, 584  
 Ricerca privata, 59  
 Ricetrasmettitore satellitare, satellite, 109  
 Ricetrasmettitore, 272  
 Riflettrometria a dominio temporale, 272  
 Rijmen, Vincent, 742  
 Rijndael, 743–745, 751  
 Rilascio della connessione, 502–506  
     TCP, 541  
 Rilevamento e correzione degli errori, 192–200  
 Ripetitore attivo, 98  
 Ripetitore, 274, 326–327  
 Ripristino dai crash, 511–513  
 Riservatezza, 302, 819920  
 Risolutore, 580  
 Rivest, Ronald, 302, 751, 754, 755, 781  
 Roberts, Larry, 51  
 Round, 738  
 Route discovery, reti ad hoc, 376–379  
 Router adiacente, 457  
 Router designato, 457  
 Router di area, 457  
 Router multicast, 711–712  
 Router multiprotocollo, 421  
 Router, 19, 326, 328  
 Routing basato sul percorso più corto, 353–356  
 Routing di sessione, 350  
 Routing multicast, 370  
 Routing multidestinazione, 368  
 Routing statico, 351  
 Routing, 31

Routing tra reti, 426–427  
 RPC (*vedere* Remote Procedure Call)  
 RRSets (*vedere* Resource Record Set)  
 RSVP (*vedere* Resource reSerVation Protocol)  
 RTCP (*vedere* Real-time Transport Control Protocol)  
 RTP (*vedere* Real-time Transport Protocol)  
 RTS (*vedere* Request to Send)  
 RTSP (*vedere* Real-Time Streaming Protocol)  
 Rumore di quantizzazione, 675  
 Rumore, 125  
 Ruota di sincronizzazione, 569

**S**

S/MIME, 804  
 SA (*vedere* Security Association)  
 SAFER+, 784  
 Sandbox, 317  
 Satellite (*vedere* satellite per comunicazioni)  
 Satellite di comunicazione, 108–117  
     confronto con la fibra ottica, 117–118  
     GEO, 109–113  
     Globalstar, 115–116  
     Iridium, 114–115  
     LEO, 114–116  
     MEO, 113–114  
     Teledesic, 116  
     VSAT, 112–113  
 S-box, 737  
 Scambio di chiavi Diffie-Hellman, 791–792  
 Scatternet, 311  
 Schema, World Wide Web, 623–625  
     ftp, 624  
     gopher, 624–625  
     http, 623–624  
     mailto, 624–625  
     news, 624  
     rtsp, 684  
     telnet, 624–625  
     URL, 623  
 Schneier, Bruce, 742  
 SCO (*vedere* Synchronous Connection Oriented)  
 Scoperta dei vicini, 361  
 SCTP (*vedere* Stream Control Transmission Protocol)  
 SDH (*vedere* Synchronous Digital Hierarchy)  
 SDLC (*vedere* Synchronous Data Link Control)  
 SECAM (*vedere* Système Electronique Colour Avec Mémoire)  
 Secure Hash Algorithm, 381, 761–762  
 Secure HTTP, 813  
 Secure Sockets Layer, 813–816  
 Security Association, 773  
 Segmentazione e ricostruzione, 65

Segmento, UDP, 525  
 Segnalazione a canale comune, 141  
 Segnalazione associata al canale, 141  
 Separazione, 302  
 Sequenza di Barker, 294  
 Sequenza di chip, 162  
 Sequenza di frammento ortogonale, 164  
 Serie di Fourier, 86–87  
 Serpent, 742, 751  
 Server con mirroring, 659–660  
 Server di directory, 495  
 Server di processo, 495  
 Server farm, 621–622  
 Server multimediale, 681–683  
 Server Pull, multimedia, 681  
 Server Push, multimedia, 682  
 Server stub, 527–529  
 Server Video, 706–709  
     architettura, 707–708  
 Server Web, 618–622  
     copia speculare, 659–660  
     replica, 659–660  
     TCP handoff, 622  
 Server, 4  
 Servizi di IEEE 802.11, 301–302  
     associazione, 301  
     autenticazione, 302  
     consegna dei dati, 302  
     disassociazione, 301  
     distribuzione, 301  
     integrazione, 301  
     privacy, 302  
     riassociazione, 301  
     separazione, 302  
 Servizi differenziati, 412–415  
 Servizi integrati, 409–412  
 Servizi,  
     rete, 344–345  
     strato data link, 184–192  
     trasporto, 481–482  
 Servizio a bit-rate variabile in tempo reale, IEEE 802.16, 308  
 Servizio a bit-rate variabile non in tempo reale, 308  
 Servizio a velocità costante, IEEE 802.16, 308  
 Servizio basato sulla classe, 412–415  
 Servizio best-effort, IEEE 802.16, 308  
 Servizio datagrammi con acknowledgement, 33  
 Servizio datagrammi, 33  
 Servizio di associazione, 802.11, 301  
 Servizio di autenticazione, 802.11, 302  
 Servizio di distribuzione, 802.11, 301  
 Servizio di integrazione, 802.11, 301  
 Servizio di riassociazione, 802.11, 301  
 Servizio di separazione, 802.11, 301  
 Servizio di trasporto, 481–492  
     fornitore, 483  
     primitive, 483–486  
     punto di accesso, 494  
     utente, 483  
 Servizio Eternity, 823  
 Servizio messaggi brevi, 666  
 Servizio orientato alla connessione, 32–33, 347–348  
 Servizio richiesta-risposta, 33  
 Servizio senza connessione, 32–33, 345–347  
 Servizio,  
     orientato alla connessione, 32–33, 347–348  
     relazione con il protocollo, 36–37  
     senza connessione, 32–33, 345–347  
 Session Initiation Protocol, 689–692, (*vedere anche* voice over IP)  
 Sessione, 40  
 SHA-1 (*vedere* Secure Hash Algorithm)  
 Shannon, Claude, 89, 90  
 Short InterFrame Spacing, 299  
 Sicurezza basata sull'offuscamento, 726  
 Sicurezza del Wireless, 780–785  
     802.11, 781–783  
     Bluetooth, 783–784  
     WAP, 785  
 Sicurezza delle comunicazioni, 772–785  
     firewall, 776–779  
     IPsec, 772–776  
     VPN, 779–780  
     wireless, 780–785  
 Sicurezza delle reti, 721–834  
 Sicurezza Web, 805–819  
     assegnazione di nomi sicuri, 806–813  
     codice mobile, 816–819  
     minacce, 805–806  
     SSL, 813–816  
 Sicurezza, 721–834  
     activeX, 817–818  
     algoritmi a chiave pubblica, 752–755  
     algoritmi a chiave simmetrica, 737–751  
     certificati, 765–771  
     codice mobile, 816–819  
     comunicazione, 772–785  
     crittografia, 724–736  
     DNS, 806–811  
     e-mail, 799–804  
     firewall, 776–779  
     firme digitali, 755–765  
     gestione delle chiavi pubbliche, 765–772  
     IPsec, 772–776  
     Java applet, 817  
     JavaScript, 818  
     PGP, 799–803  
 PKI, 769  
 problemi sociali, 819–828  
 protocollti di autenticazione, 785–799  
 SSL, 813–816  
 VPN, 779–780  
 Web, 805–819  
 wireless, 780–785  
 SIFS (*vedere* Short InterFrame Spacing)  
 Simbolo, 127  
 Simple Internet Protocol Plus, 465  
 Sincronizzazione, 41  
 Silly window syndrome, 545–547  
 SIP (*vedere* Session Initiation Protocol) (*vedere anche* Voice over IP)  
 SIPP (*vedere* Simple Internet Protocol Plus)  
 Sistema distribuito, 2  
 Sistema premi e parla, 153  
 Sistema telefonico cellulare, 152–169  
     prima generazione, 153–157  
     seconda generazione, 157–166  
     terza generazione, 166–169  
 Sistema telefonico mobile della seconda generazione, 157–166  
 Sistema telefonico mobile di terza generazione, 166–169  
 Sistema telefonico, 118–151  
     collegamenti locali, 124–137  
     mobile, 152–169  
 multiplex a divisione  
     di frequenza, 137–138  
 multiplex a divisione di lunghezza  
     d'onda, 138–140  
 multiplex a divisione di tempo, 140–143  
 politiche, 122–123  
 struttura, 119–121  
 tronco, 137–143  
 Sito Web, di questo libro, 79  
 Smiley, 588–589  
 SMTP (Simple Mail Transfer Protocol), 602–605  
 Snail mail, 588  
 SOAP (Simple Object Access Protocol), 642  
 SOAP (*vedere* Simple Object Access Protocol)  
 Socket Berkeley, 487–488  
 Socket, 487–492  
 Solitoni, 96  
 Solo testo cifrato, attacco, 727  
 SONET (*vedere* Synchronous Optical NETwork)  
 Sorveglianza della stazione, 111  
 Sottorete a circuito virtuale, 346–347  
     confronto con i datagrammi, 348–350  
     controllo della congestione, 389–390  
 Sottorete a commutazione di pacchetto, 20  
 Sottorete a datagrammi, 345–347

confronto con circuito virtuale, 348-350  
 controllo delle congestioni, 391-395  
 Sottorete di comunicazione, 19, 344  
 Sottorete, 19, 439  
     circuito virtuale, 347-348  
     comunicazione, 344  
     confronto tra datagramma e circuito virtuale, 348-350  
     datagramma, 345-347  
 Sottostrato convergenza di trasmissione, 64  
 Sottostrato di convergenza, 65  
 Sottostrato dipendente dal mezzo fisico ATM, 64  
 Sottostrato MAC, 247-342  
     allocazione dinamica del canale, 249-251  
     allocazione statica del canale, 248-249  
 Bluetooth, 310-317  
 commutazione dello strato data link, 317-336  
 Ethernet, 271-292  
     LAN wireless, 292-302  
     protocolli ad accesso multiplo, 251-270  
     sottostrato, allocazione del canale, 248-251  
 SPE (*vedere* Synchronous Payload Envelope)  
 Specifiche del flusso, 407  
 Spettro elettromagnetico, 100-102  
     politica, 105-106  
 SSL (*vedere* Secure Sockets Layer)  
 Standard de facto, 71  
 Standard de jure, 71  
 Standard per firme digitali, 758-759  
 Standard proposti, 77  
 Standard,  
     standard de facto, 71  
     standard de jure, 71  
 Stato iniziale, macchina a stati finiti, 230  
 Stazione, 249  
 Steganografia, 824-825  
 Stella passiva, 98  
 Strato applicazione, 41, 43, 579-720  
     DNS, 579-588  
     multimedia, 674-714  
     posta elettronica, 588-611  
     World Wide Web, 611-673  
 Strato bearer, WAP, 664  
 Strato data link, 38, 183-246  
     bit stuffing, 190-191  
     byte stuffing, 189-190  
     controllo del flusso, 192  
     gestione degli errori, 192-200  
     problemi di progettazione, 184-192  
     procedure di interfaccia, 202-204  
     protocolli elementari, 200-211  
     protocolli sliding window, 211-228

protocollli, 200-228  
 protocollo HDLC, 234-237  
 protocollo LCP, 239-242  
 protocollo NCP, 239, 242  
 protocollo PPP, 238-242  
 protocollo simplex non limitato, 204-206  
 protocollo stop-and-wait, 206-211  
 verifica dei protocollli, 229-234  
 Strato di rete, 39, 343-480  
     algoritmi di routing, 350-384  
     connessione tra reti, 418-431  
     controllo della congestione, 384-396  
     Internet, 431-473  
     problemi di progetto delle reti, 343-350  
     qualità del servizio, 397-417  
 Strato fisico, layer, 38, 85-182  
     IEEE 802.11, 293-295  
     IEEE 802.16, 306-307  
     Internet via cavo, 169-176  
     mezzi di trasmissione via cavo, 90-99  
     sistema telefonico mobile, 152-169  
     sistema telefonico, 118-151  
     trasmissione via satellite, 109-118  
     trasmissione wireless, 100-108  
 Strato Internet, 42  
 Strato presentazione, 41  
 Strato trasporto, 40, 42, 481-578  
     controllo di flusso, 506-510  
     handshake a tre vie, 499-502  
     impostazione della connessione, 496-502  
     indirizzamento, 493-496  
     Internet, 524-556  
     multiplex, 510-511  
     problemi di prestazioni, 557-573  
     problemi di progettazione, 492-513  
     protocollo di esempio, 513-524  
     recupero dal crash, 511-513  
     rilascio della connessione, 502-506  
     servizio, 481-492  
     sicurezza, 816  
     TCP, 532-566  
     UDP, 524-532  
     utilizzo dei buffer, 506-510  
 Strato, 26  
     applicazione, 41, 579-720  
     data link, 38, 183-246  
     fisico, 38, 85-182  
     presentazione, 41  
     problemi di progettazione, 30-31  
     rete, 39, 343-480  
     sessione, 40  
     trasporto, 481-578

Stream Control Transmission Protocol, 556  
 Striping, 708  
 Struttura basata su core, 372  
 Struttura Sink tree, 352  
 Struttura Spanning tree, 368  
 STS (*vedere* Synchronous Transport Signal)  
 Stub del client, 527-529  
 Supergruppo, 138  
 Supporto continuo, 674  
 Switch cut-through, 328  
 Switch, 326-328  
     Cut-through, 328  
     switch Ethernet, 281  
 Synchronous Connection Oriented, 316  
 Synchronous Data Link Control, 234  
 Synchronous Digital Hierarchy, 144  
 Synchronous Optical NETwork, 144-146  
 Synchronous Payload Envelope, 145  
 Synchronous Transport Signal, 145  
 Systeme Electronique Colour Avec Memoire, 694

**T**

Tabella, HTML, 633-634  
 Tariffa, 72  
 TCM (*vedere* Trellis Coded Modulation)  
 TCP (Transmission Control Protocol), 42, 532-556  
     algoritmo di Jacobson, 549-550  
     algoritmo di Karn, 552  
     algoritmo di Nagle, 545-547  
     controllo della congestione, 547-548  
     criterio di trasmissione, 543-547  
     dati urgenti, 535  
     gestione del timer, 550-553  
     handoff, 622  
     impostazione della connessione, 539-540  
     indiretto, 553-554  
     intestazione, 535-539  
     macchina a stati finiti, 541-543  
     modelli di gestione della connessione, 541-543  
     modello di servizio, 533-535  
     radio Internet, 684-685  
     rilascio della connessione, 541  
     segmento, 535-539  
     silly window syndrome, 545-547  
     transazionale, 555-556  
     wireless, 553-555  
 TCP indiretto, 553-554  
 TCP transazionale, 555-556  
 TCP Wireless, 553-555  
 TCPA (Trusted Computing Platform Alliance), 828  
 TCPA (*vedere* Trusted Computing Platform Alliance)

TDD (*vedere* Time Division Duplexing)  
 TDM (*vedere* Time Division Multiplexing)  
 Teledesic, 116  
 Telefonia Internet, 685-692 (*vedere anche* Voice over IP)  
 Telefonia su Internet (*vedere* voice over IP)  
 Telefono cellulare, 152  
 Telefono cordless, 152  
 Telefono portatile (*vedere* Telefono cellulare)  
 Televisione ad antenna collettiva, 169-170  
 Televisione via cavo, 169-175, 710  
 Tempesta di broadcast, 330, 558  
 Tempo di Dwell, 294  
 Terminale, 249, 686  
 Testo cifrato, 725  
 Testo in chiaro a scelta, attacco, 727  
 Testo in chiaro noto, attacco, 727  
 Testo in chiaro, 725  
 Thread colorato, 417  
 Time Division Duplexing, 307  
 Time Division Multiplexing, 137, 140-143  
 Timer di continuità, 552  
 Timer di ritrasmissione, 550  
 Timer keepalive, 552  
 Tipo MIME, 598-602, 617-618  
 TLS (*vedere* Strato di trasporto, sicurezza)  
 Token, 67  
     Rete di Petri, 232  
 TPDU (*vedere* Transport Protocol Data Unit)  
 Tra pari, 27  
 Traffico combinato, 212  
 Traffico InterAS, 459  
 Transizione, 232  
     macchina a stati finiti, 229  
 Transport Protocol Data Unit, 485  
 Trasferimento di file, 57  
 Trasformazione di Fourier, 676  
 Trasmissione di dati multimediali, 674  
 Trasmissione di flussi di dati audio, 679-683  
     media player, 680-683  
     metafile, 680  
     server pull, 681-682  
     server push, 682  
     tipo MIME, 679-680  
 Trasmissione lightwave, 107-108  
 Trasmissione Radio, 103-104  
 Trasmissione Wireless, 100-108  
 Tratto vocale, 676  
 Tree-walk adattativo, protocollo, 263-265  
 Trellis Coded Modulation, 128  
 Trigramma, 728  
 Triple DES, 740-741, 751  
 Trudy, 732

Trust anchor, 770  
**TSAP** (*vedere servizio di trasporto, punto di accesso*)

Tunnel, 425–427

Twofish, 742, 751

## U

**UDP** (User Datagram Protocol), 43, 524–532

intestazione, 526

segmento, 525–526

wireless, 553–555

**UDP** (User Datagram Protocol), 43, 524–532, (*vedere UDP*)

**UDP Wireless**, 553–555

Ufficio Tandem, 120

**UMTS** (*vedere Universal Mobile Telecommunications System*)

Unicast, 15

Uniform Resource Name, 625

Unità metriche, 77–78

Universal Mobile Telecommunications System, 167

Universal resource name, 625

Unshielded Twisted Pair, 91–92

categoria 3, 91

categoria 5, 92

**URL** (Uniform Resource Locator), 614, 622–625

**URL** (*vedere Uniform Resource Locator*)

URL auto certificante, 811–813

URL Web, 614, 622–625

**URN** (*vedere Uniform Resource Name*)

Utente mobile, 9–12, 372

Utilizzo della filigrana, 826

Utilizzo delle chiavi, 126

**UTP** (*vedere Unshielded Twisted Pair*)

## V

V.32 bis modem, 128

V.34 bis modem, 128

V.34 modem, 128

V.90 modem, 130

V.92 modem, 130

**VC** (*vedere Virtual Circuit*)

Velocità della luce, 100

Velocità massima di un canale

limite di Nyquist, 89

limite di Shannon, 89–90

Verifica di protocollo, 229–234

**Very Small Aperture Terminal**, 112

Vicino attivo, 379

Video a richiesta (on demand), 704–711

Rete di distribuzione, 709–711

Video composito, 694

**Video digitale**, introduzione, 692–696

**Video interlacciato**, 693

**Video progressivo**, 693

**Video**, 692–711, (*vedere anche compressione video*)

campo, 693

codifica, 696

crominanza, 694

fotogramma, 693

HDTV, 694–695

interlacciato, 693

luminanza, 694

NTSC, 693, 694

PAL, 693, 694

progressivo, 693

scansione dei parametri, 693, 695

SECAM, 693, 694

**Virtual Circuit**, 62, 346

**Virtual LAN**, 328–336

**Virus**, 818–819

Visualizzazione, posta elettronica, 590

**VLAN** (*vedere Virtual LAN*)

**Voce aggregata**, 444

**Vocoder**, 158

**Voice over IP**, 685–692

confronto tra H.323 e SIP, 691–692

G.711, 686

G.723.1, 687

H.245, 687

H.323 gatekeeper, 686

H.323 pila di protocolli, 687

H.323, 683–689

impostazione di una chiamata, 687–689

metodi SIP, 690

numeri di telefono SIP, 689

protocollo SIP, 690

Q.931, 687

RAS, 687

RTCP, 687

RTP, 687

SIP, 689–692

**VPN** (Virtual Private Network), 779–780

**VSAT** (*vedere Very Small Aperture Terminal*)

## W

**W3C** (*vedere World Wide Web Consortium*)

**WAE** (*vedere Wireless Application Environment*)

**WAN** (*vedere Wide Area Network*)

**WAP** (*vedere Wireless Application Protocol*)

**WAP 1.0**, 663–665

architettura, 664–665

stack di protocollo, 664

**WAP 2.0**, 670–673

concorrenza con 802.11, 673

confronto con WAP 1.0, 671–672

emoji, 672

stack di protocollo, 672

XHTML base, 673

**WAP 1.1**, 663–665, 670–673, (*vedere anche WAP 1.0, WAP 2.0*)

ambiente di applicazione wireless, 664

architettura, 665–665

confronto con 802.11, 673

confronto con i-mode, 671

emoji, 672

pila di protocolli, 664, 672

protocollo di datagramma wireless, 664

protocollo di sessione wireless, 664

protocollo di transazione wireless, 664

sicurezza dello strato trasporto wireless, 664

sicurezza, 785

strato bearer, 664

utilizzo di XHTML di base, 673

utilizzo di XML, 664

**Watson, Thomas, J.**, 23

**Wavelength Division Multiple Access**, 265–267

**Wavelength Division Multiplexing**, 138–140

**WDM** (*vedere Wavelength Division Multiplexing*)

**WDMA** (*vedere Wavelength Division Multiple Access*)

**WDP** (*vedere Wireless Datagram Protocol*)

**Web** (World Wide Web), 2, 611–673

collegamento ipertestuale, 612

cookie, 625–629

flash crowd, 660

HTML, 615, 629–639

HTTP 41, 623, 651–656

i-mode (*vedere i-mode*)

lato client 614–618

modulo, 634–638

pagina, 612

prestazioni, 665–666

rete di distribuzione del contenuto, 660–662

schema, 623–625

storia, 57–58, 611–612

visione d'insieme sull'architettura, 612–629

wireless, 662–673

XML, 639–642

XSL, 639–642

Web server farm, 621–622

Web Wireless, 662–673

seconda generazione, 670–673

WAP 1.0, 663–665

WAP 2.0, 670–673

Webmail, 610–611

**WEP** (*vedere Wired Equivalent Privacy*)

**WiFi** (*vedere IEEE 802.11*)

**Wired Equivalent Privacy**, 300, 781–783

**Wireless Transaction Protocol**, 664

**Wireless a banda larga** (*vedere IEEE 802.16*)

**Wireless application environment**, 664

**Wireless datagram protocol**, 664

**Wireless fissa**, 10, 135 (*vedere anche IEEE 802.16*)

**Wireless LAN** (*vedere IEEE 802.11*)

**Wireless Session Protocol**, 664

**Wireless Transport Layer Security**, 664

**WLL** (Wireless Local Loop) (*vedere IEEE 802.16*)

**WML** (*vedere Wireless Markup Language*)

**WML** (Wireless Markup Language), 664

**World Wide Wait**, 660

**World Wide Web** (*vedere Web*)

**World Wide Consortium**, 612

**WSP** (*vedere Wireless Session Protocol*)

**WTLS** (*vedere Wireless Transport Layer Security*)

**WTP** (*vedere Wireless Transaction Protocol*)

**WWW** (*vedere Web*)

## X

**X.25**, 61

**X.400**, 589–590

**X.500**, 588

**X.509**, 767–768

**XDSL**, 130

**XHTML** (*vedere eXtensible HyperText Markup Language*)

**XML** (*vedere eXtensible Markup Language*)

**XSL** (*vedere eXtensible Style Language*)

## Z

**Zimmermann, Phil**, 799

**Zona**, 686

DNS, 586