

RETI DI COMPUTER

NUOVA EDIZIONE

Indice

PREFAZIONE

XIII

PREMESSA

XV

Capitolo 1 INTRODUZIONE

1

1.1 Reti di calcolatori

2

1.1.1 Reti per aziende (2) - 1.1.2 Reti per le persone (4) - 1.1.3 Aspetti sociali (5)

1.2 Hardware delle reti

7

1.2.1 Reti locali (8) - 1.2.2 Reti metropolitane (10) - 1.2.3 Reti geografiche (10) - 1.2.4 Reti senza filo (13) - 1.2.5 Inter-reti (15)

1.3 Software delle reti

16

1.3.1 Gerarchie di protocolli (16) - 1.3.2 Aspetti progettuali per i livelli (20) - 1.3.3 Interfacce e servizi (21) - 1.3.4 Servizi orientati alla connessione e servizi privi di connessione (22) - 1.3.5 Primitive dei servizi (23) - 1.3.6 Relazioni fra servizi e protocolli (26)

1.4 Modelli di riferimento

27

1.4.1 Il modello di riferimento OSI (27) - 1.4.2 Il modello di riferimento TCP/IP (33) - 1.4.3 Un confronto fra i modelli di riferimento OSI e TCP/IP (36) - 1.4.4 Una critica al modello e ai protocolli OSI (38) - 1.4.5 Una critica al modello di riferimento TCP/IP (41)

1.5 Esempi di reti

42

1.5.1 Novell NetWare (43) - 1.5.2 ARPANET (45) - 1.5.3 NSFNET (48) - 1.5.4 Internet (50) - 1.5.5 Reti sperimentali con velocità dell'ordine del gigabit (52)

1.6 Esempi di servizi per la comunicazione dei dati	54
1.6.1 SMDS – Switched Multimegabit Data Service (54) - 1.6.2 Reti X.25 (57) - 1.6.3 Frame relay (58) - 1.6.4 Reti Broadband-ISDN e reti ATM (58) - 1.6.5 Confronto fra i servizi (63)	
1.7 Standardizzazione delle reti	64
1.7.1 Chi c'è nel mondo delle comunicazioni (64) - 1.7.2 Chi c'è nel mondo degli standard internazionali (66) - 1.7.3 Chi c'è nel mondo degli standard per Internet (68)	
1.8 Struttura del seguito del libro	69
1.9 Riassunto	70
Esercizi	71
 Capitolo 2	
IL LIVELLO FISICO	75
2.1 Le basi teoriche per la comunicazione dei dati	75
2.1.1 Analisi di Fourier (75) - 2.1.2 Segnali a larghezza di banda limitata (76) - 2.1.3 Il tasso di dati massimo per un canale (78)	
2.2 Mezzi di trasmissione	80
2.2.1 Mezzi magnetici (80) - 2.2.2 Il doppino (twisted pair) (80) - 2.2.3 Cavo coassiale a banda base (81) - 2.2.4 Cavo coassiale a larga banda (82) - 2.2.5 Fibre ottiche (84)	
2.3 La trasmissione senza filo	91
2.3.1 Lo spettro elettromagnetico (91) - 2.3.2 Trasmissione radio (94) - 2.3.3 Trasmissione a microonde (95) - 2.3.4 Onde infrarosse e millimetriche (96) - 2.3.5 Trasmissione a onde luminose (97)	
2.4 Il sistema telefonico	98
2.4.1 Struttura del sistema telefonico (99) - 2.4.2 La politica dei telefoni (102) - 2.4.3 Il circuito locale (104) - 2.4.4 Dorsali e multiplexing (113) - 2.4.5 Commutazione (125)	
2.5 ISDN a banda stretta (N-ISDN)	134
2.5.1 Servizi ISDN (134) - 2.5.2 Architetture del sistema ISDN (135) - 2.5.3 L'interfaccia ISDN (137) - 2.5.4 Prospettiva su N-ISDN (138)	

2.6 ISDN a larga banda e ATM	138
2.6.1 Circuiti virtuali contro commutazione di circuito (139) - 2.6.2 Trasmissione in reti ATM (141) - 2.6.3 Comutatori ATM (142)	
2.7 Radio cellulare	149
2.7.1 Sistemi di rintracciamento (paging) (150) - 2.7.2 Telefoni senza filo (150) - 2.7.3 Telefoni cellulari analogici (151) - 2.7.4 Telefoni cellulari digitali (156) - 2.7.5 Servizi di comunicazione personale (156)	
2.8 Satelliti per comunicazioni	157
2.8.1 Satelliti geosincroni (158) - 2.8.2 Satelliti a orbita bassa (160) - 2.8.3 Confronto tra satelliti e fibre (162)	
2.9 Riassunto	163
Esercizi	164
 Capitolo 3	
IL LIVELLO DATA LINK	169
3.1 Principi di progettazione del livello data link	169
3.1.1 I servizi forniti al livello rete (169) - 3.1.2 Impacchettamento (framing) (172) - 3.1.3 Controllo degli errori (175) - 3.1.4 Controllo del flusso (176)	
3.2 Rilevazione e correzione degli errori	177
3.2.1 Codici di correzione degli errori (177) - 3.2.2 Codici di rilevazione di errori (180)	
3.3 Protocolli elementari data link	184
3.3.1 Un protocollo simplex non limitato (188) - 3.3.2 Un protocollo simplex stop-and-wait (188) - 3.3.3 Un protocollo simplex per un canale disturbato (191)	
3.4 I protocolli a finestra scorrevole (sliding window)	194
3.4.1 Un protocollo sliding window di un solo bit (196) - 3.4.2 Un protocollo go back n (200) - 3.4.3 Un protocollo con ripetizione selettiva (204)	
3.5 Specifica e verifica di protocolli	209
3.5.1 I modelli a macchine a stati finiti (209) - 3.5.2 I modelli a reti di Petri (214)	

3.6 Esempi di protocolli data link	216
3.6.1 HDLC – High-level Data Link Control (216) - 3.6.2 Il livello data link in Internet (219) - 3.6.3 Il livello data link di ATM (225)	
3.7 Riassunto	228
Esercizi	229
Capitolo 4	
IL SOTTOLIVELLO DI ACCESSO AL MEZZO	233
4.1 I problemi di allocazione del canale	233
4.1.1 Allocazione statica di canali nelle LAN e nella WAN (234) - 4.1.2 Allocazione dinamica di canali nelle LAN e nelle WAN (235)	
4.2 Protocolli di accesso multiplo	236
4.2.1 ALOHA (236) - 4.2.2 Protocolli multiaccesso con rilevamento della portante (240) - 4.2.3 Protocolli esenti da collisione (243) - 4.2.4 Protocolli a contesa limitata (246) - 4.2.5 Protocolli multiaccesso a suddivisione di lunghezza d'onda (249) - 4.2.6 Protocollo per LAN senza filo (252) - 4.2.7 Radio cellulari digitali (255)	
4.3 Standard IEEE 802 per LAN e MAN	264
4.3.1 Lo standard IEEE 802.3 ed Ethernet (265) - 4.3.2 Lo standard IEEE 802.4: token bus (275) - 4.3.3 Lo Standard IEEE 802.5: token ring (281) - 4.3.4 Confronti tra 802.3, 802.4 e 802.5 (288) - 4.3.5 Lo standard IEEE 802.6: Distributed Queue Dual Bus (289) - 4.3.6 Lo standard 802.2: Logical Link Control (292)	
4.4 I bridge	293
4.4.1 Bridge da 802.x a 802.y (294) - 4.4.2 Bridge trasparenti (298) - 4.4.3 Bridge di instradamento da sorgente (302) - 4.4.4 Confronti tra i bridge 802 (304) - 4.4.5 Bridge remoti (305)	
4.5 LAN ad alta velocità	306
4.5.1 FDDI (306) - 4.5.2 Fast Ethernet (309) - 4.5.3 HIPPI – interfaccia parallela ad alte prestazioni (312) - 4.5.4 Canali a fibra (314)	
4.6 Reti satellitari	315
4.6.1 Polling (316) - 4.6.2 ALOHA (316) - 4.6.3 FDM (317) - 4.6.4 TDM (318) - 4.6.5 CDMA (320)	

4.7 Riassunto	320
Esercizi	322
Capitolo 5	
IL LIVELLO RETE	327
5.1 Caratteristiche di progetto del livello rete	327
5.1.1 Servizi offerti al livello trasporto (327) - 5.1.2 Organizzazione interna del livello rete (329) - 5.1.3 Confronto fra reti basate su circuito virtuale e reti basate su datagram (331)	
5.2 Algoritmi di routing	333
5.2.1 Il principio di ottimalità (335) - 5.2.2 Routing lungo il cammino minimo (336) - 5.2.3 Flooding (338) - 5.2.4 Routing basato su flusso (340) - 5.2.5 Routing basato su vettori di distanza (342) - 5.2.6 Routing basato sullo stato dei canali (346) - 5.2.7 Routing gerarchico (352) - 5.2.8 Routing per host mobili (354) - 5.2.9 Broadcast routing (357) - 5.2.10 Multicast routing (358)	
5.3 Algoritmi di controllo della congestione	360
5.3.1 Principi generali del controllo di congestione (362) - 5.3.2 Politiche di prevenzione della congestione (364) - 5.3.3 Normalizzazione del traffico (365) - 5.3.4 Specifiche di flusso (371) - 5.3.5 Controllo delle congestioni in reti basate su circuito virtuale (372) - 5.3.6 Pacchetti regolatori (373) - 5.3.7 Caduta di carico (376) - 5.3.8 Controllo del jitter (378) - 5.3.9 Controllo di congestione per il multicasting (379)	
5.4 Internetworking	381
5.4.1 In cosa differiscono le reti (385) - 5.4.2 Circuiti virtuali concatenati (386) - 5.4.3 Internetworking senza connessioni (387) - 5.4.4 Tunneling (389) - 5.4.5 Routing all'interno di Internet (390) - 5.4.6 Frammentazione (392) - 5.4.7 Firewall (394)	
5.5 Il livello rete in Internet	397
5.5.1 Il protocollo IP (398) - 5.5.2 Indirizzi IP (401) - 5.5.3 Sottoreti (403) - 5.5.4 Protocolli di controllo di Internet (404) - 5.5.5 Il protocollo di routing tra gateway interni: OSPF (409) - 5.5.6 Il protocollo di routing tra gateway esterni: BGP (414) - 5.5.7 Internet multicasting (416) - 5.5.8 IP mobile (417) - 5.5.9 CIDR – routing interdominio senza classe (419) - 5.5.10 IPv6 (422)	

5.6 Il livello rete nelle reti ATM	434
5.6.1 Formati delle celle (436) - 5.6.2 Creazione delle connessioni (438) - 5.6.3 Instradamento e commutazione (440) - 5.6.4 Categorie di servizi (443) - 5.6.5 Qualità del servizio (445) - 5.6.6 Normalizzazione e pattugliamento del traffico (447) - 5.6.7 Controllo della congestione (451) - 5.6.8 LAN basate su ATM (455)	
5.7 Riassunto	457
Esercizi	458
Capitolo 6	
IL LIVELLO TRASPORTO	463
6.1 Il servizio di trasporto	463
6.1.1 I servizi forniti ai livelli superiori (463) - 6.1.2 Qualità del servizio (465) - 6.1.3 Primitive del servizio di trasporto (467)	
6.2 Elementi del protocollo di trasporto	471
6.2.1 Indirizzamento (472) - 6.2.2 Creare una connessione (476) - 6.2.3 Chiudere una connessione (481) - 6.2.4 Controllo di flusso e gestione dei buffer (485) - 6.2.5 Multiplexing (489) - 6.2.6 Ripristino dai guasti (491)	
6.3 Un semplice protocollo di trasporto	493
6.3.1 Le primitive del servizio di esempio (493) - 6.3.2 L'entità di trasporto di esempio (495) - 6.3.3 L'esempio visto come una macchina a stati finiti (503)	
6.4 I protocolli di trasporto di Internet (TCP e UDP)	504
6.4.1 Il modello di servizio TCP (506) - 6.4.2 Il protocollo TCP (507) - 6.4.3 Il preambolo del segmento TCP (508) - 6.4.4 Gestione delle connessioni TCP (511) - 6.4.5 Politica di trasmissione di TCP (514) - 6.4.6 Controllo di congestione di TCP (519) - 6.4.7 Gestione dei timer di TCP (521) - 6.4.8 UDP (525) - 6.4.9 TCP e UDP senza filo (525)	
6.5 I protocolli del livello AAL di ATM	527
6.5.1 Struttura del livello AAL (529) - 6.5.2 AAL 1 (530) - 6.5.3 AAL 2 (532) - 6.5.4 AAL 3/4 (532) - 6.5.5 AAL 5 (535) - 6.5.6 Confronto dei protocolli AAL (536) - 6.5.7 SSCOP (537)	
6.6 Problematiche relative alle prestazioni	538
6.6.1 Problemi di prestazione nelle reti di computer (539) - 6.6.2 Misurare le prestazioni di rete (540) - 6.6.3 Progettazione di sistema per migliorare	

le prestazioni (543) - 6.6.4 Elaborazione veloce dei TPDU (546) - 6.6.5 Protocolli per reti ad alta velocità (549)	
6.7 Riassunto	554
Esercizi	554
Capitolo 7	
IL LIVELLO DELLE APPLICAZIONI	559
7.1 La sicurezza in rete	559
7.1.1 La crittografia tradizionale (561) - 7.1.2 Due principi fondamentali di crittografia (567) - 7.1.3 Algoritmi a chiave segreta (568) - 7.1.4 Algoritmi a chiave pubblica (578) - 7.1.5 Protocolli di autenticazione (582) - 7.1.6 Firme digitali (594) - 7.1.7 Aspetti sociali (601)	
7.2 DNS – Domain Name System	603
7.2.1 Lo spazio dei nomi del DNS (603) - 7.2.2. Descrittori di risorsa (605) - 7.2.3 Name server (609)	
7.3 SNMP – Simple Network Management Protocol	611
7.3.1 Il Modello SNMP (611) - 7.3.2. ASN.1 – Abstract Syntax Notation 1 (613) - 7.3.3 SMI – Struttura delle informazioni di gestione (619) - 7.3.4 Il MIB – Management Information Base (621) - 7.3.5 Il protocollo SNMP (622)	
7.4 La posta elettronica	623
7.4.1 Architettura e servizi (625) - 7.4.2 L'agente utente (User Agent) (626) - 7.4.3 Formato dei messaggi (630) - 7.4.4 Trasferimento di messaggi (637) - 7.4.5 Posta elettronica privata (643)	
7.5 Le news di USENET	648
7.5.1 USENET per l'utente (649) - 7.5.2 Come è realizzata USENET (654)	
7.6 Il World Wide Web	659
7.6.1 La parte client (661) - 7.6.2 La parte server (664) - 7.6.3 Scrivere una pagina Web in HTML (671) - 7.6.4 Java (685) - 7.6.5 Come localizzare le informazioni sul Web (698)	
7.7 Il multimediale	702
7.7.1 Audio (702) - 7.7.2 Video (705) - 7.7.3 Compressione dei dati (708) - 7.7.4 Video su richiesta (721) - 7.7.5 MBone (733)	

7.8 Riassunto	736
----------------------	------------

Esercizi	737
-----------------	------------

Capitolo 8	
RIFERIMENTI BIBLIOGRAFICI	743

8.1 Suggerimenti per letture ulteriori	743
---	------------

8.1.1 Introduzione e lavori di ordine generale (743) - 8.1.2 Il livello fisico (744) - 8.1.3 Il livello data link (746) - 8.1.4 Il sottolivello medium access control (746) - 8.1.5 Il livello rete (747) - 8.1.6 Il livello trasporto (747) - 8.1.7 Il livello delle applicazioni (748)

8.2 Bibliografia	750
-------------------------	------------

Indice analitico	767
-------------------------	------------

PREMESSA

Questo libro è giunto alla terza edizione. Ciascuna edizione corrisponde a un diverso modo di usare le reti di computer. Quando la prima edizione apparve nel 1980, le reti erano una curiosità accademica. Quando apparve la seconda edizione nel 1988 le reti erano usate da università e grandi società. Quando è apparsa la terza edizione nel 1996 le reti, specie la Internet mondiale, sono diventate una realtà quotidiana per milioni di persone.

Inoltre, l'hardware e il software di rete sono cambiati completamente dai tempi della seconda edizione. Nel 1988 quasi tutte le reti si basavano su filo di rame. Oggi molte usano cavi in fibra ottica o comunicazioni senza filo. Le reti commerciali, quali la SNA, sono diventate molto meno importanti delle reti pubbliche, quali la Internet. I protocolli OSI sono svaniti silenziosamente, mentre il protocollo TCP/IP è oggi dominante. In effetti tutto è talmente cambiato che il libro è stato quasi completamente riscritto.

Sebbene il capitolo 1 svolga la stessa funzione introduttiva come nella seconda edizione, il suo contenuto è stato completamente rivisto e aggiornato. Per esempio, invece di basare il libro sul modello OSI a sette livelli, nel capitolo 1 viene introdotto un modello ibrido a 5 livelli (vedi figura 1-21). Anche se non è esattamente uguale al modello TCP/IP, questo nuovo modello è molto più vicino per spirito a TCP/IP di quello OSI che venne usato nella seconda edizione. Inoltre, qui introduciamo i nuovi esempi principali usati in tutto il libro, Internet e ATM, oltre ad alcune reti a gigabit e altri modelli famosi.

Nel capitolo 2 presentiamo invece del cavo di rame la fibra ottica e la comunicazione senza filo, perché queste sono le tecnologie del futuro. Il sistema telefonico è diventato quasi interamente digitale negli ultimi dieci anni, quindi la sua descrizione è stata largamente riscritta, con l'aggiunta di nuovo materiale sul sistema ISDN a larga banda. Il materiale sulla radio cellulare è stato espanso fortemente, con l'aggiunta di nuovo materiale sui satelliti a orbita bassa.

L'ordine di presentazione tra il livello data link e il sottolivello MAC è stato invertito, perché l'esperienza diretta con gli studenti ha dimostrato che essi capiscono meglio il sottolivello MAC dopo aver studiato il livello data link. I protocolli usati come esempio sono stati conservati, perché sono molto popolari, ma sono stati riscritti in C. È stato aggiunto nuovo materiale sui livelli data link di Internet e ATM.

Nel capitolo 4 i principi di progettazione del sottolivello MAC sono stati rivisti per poter descrivere alcuni nuovi protocolli, che includono il multiplexing a divisione di lunghezza d'onda, le LAN senza filo e la radio digitale. La discussione dei bridge è stata rivista, e abbiamo aggiunto nuovo materiale sulle LAN ad alta velocità.

La maggior parte degli algoritmi di instradamento del capitolo 5 sono stati rimpiazzati da

esempi più moderni, compreso il vettore di distanza e il routing link state. I paragrafi sul controllo della congestione sono stati completamente rifatti, mentre il materiale su Internet e ATM è completamente nuovo.

Il capitolo 6 riguarda ancora il livello di trasporto, ma anche qui sono stati fatti grossi cambiamenti, soprattutto aggiungendo nuovo materiale su Internet, ATM e le prestazioni delle reti.

Il capitolo 7, sul livello delle applicazioni, è ora il capitolo più lungo del libro. Il materiale sulla sicurezza delle reti è stato raddoppiato in lunghezza, e abbiamo aggiunto nuovo materiale su DNS, SNMP, la posta elettronica, USENET, il World Wide Web, HTML, Java, il multimediale, il video a richiesta, e MBone.

Delle 395 figure di questa terza edizione, 276 (il 70%) è completamente nuovo, mentre altre sono state riviste. Dei 370 riferimenti bibliografici, 281 (76%) riguardano libri e articoli pubblicati dopo la pubblicazione della seconda edizione. Di questi, oltre 100 riguardano lavori pubblicati nel 1995 e 1996. Complessivamente, probabilmente il 75% del libro è completamente nuovo, e parti del restante 25% sono state pesantemente riviste. Siccome questo è effettivamente un libro nuovo, la copertina è stata rifatta per evitare confusioni con la seconda edizione.

I libri di informatica sono ricchi di acronimi. Questo non fa eccezione. Alla fine del suo studio, il lettore dovrebbe aver acquisito familiarità con le sigle che seguono: AAL, AMPS, ARP, ASN, ATM, BGP, CDMA, CDPD, CSMA, DQDB, DNS, FAQ, FDM, FTP, FITC, FITH, GSM, HDLC, HEC, HIPPI, IAB, ICMP, IDEA, IETF, IPv6, ISO, ITU, LATA, MAC, MACA, MAN, MIB, MIME, NAP, NNTP, NSA, NSAP, OSI, OSPF, PCM, PCN, PCS, PEM, PGP, PPP, PSTN, PTT, PVC, QAM, RARP, RFC, RSA, SABME, SAP, SAR, SDH, SDLC, SHA, SMI, SNA, SNMP, SNRME, SPX, TCP, UDP, VHF, VLF, VSAT, WARC, WDM, WWW, e WWW. Non preoccupatevi. Tutti saranno accuratamente definiti prima di essere usati.

Per aiutare i docenti a usare questo libro come testo di riferimento, l'autore ha preparato tre supporti didattici:

- Un manuale di soluzione dei problemi.
- Un file PostScript contenente tutte le figure (per generare lucidi da proiezione).
- Un simulatore scritto in C dei protocolli di esempio del capitolo 3.

Il file con le figure e il simulatore sono disponibili via WWW all'indirizzo <http://www.cs.vu.nl/~ast/>.

Il libro è stato formattato usando Troff e il tipo di carattere Times Roman, che dopo tutti questi anni è ancora la miglior soluzione. Anche se Troff non è di moda come i sistemi WYSIWYG, invitiamo il lettore a paragonare la qualità tipografica di questo libro con quella dei libri prodotti mediante sistemi WYSIWYG. La mia unica concessione ai personal computer e all'editoria elettronica da scrivania è che per la prima volta le figure sono state prodotte mediante Adobe Illustrator, invece di essere disegnate su carta. Per la prima volta il libro è stato prodotto completamente in forma elettronica. L'output PostScript di Troff è stato spedito alla tipografia via Internet, dove sono state prodotte le pellicole per la stampa

offset. Di solito si usa una copia cartacea che viene fotografata, ma stavolta ne abbiamo fatto a meno.

Sono stato aiutato da molte persone per questa terza edizione. Voglio ringraziare specialmente Chase Bailey, Saniya Ben Hassen, Nathaniel Borenstein, Ron Cocchi, Dave Crocker, Wiebren de Jonge, Carl Ellison, M. Rasit Eskicioglu, John Evans, Mario Gerla, Mike Goguen, Paul Green, Dick Grune, Wayne Hathaway, Franz Hauck, Jack Holtzman, Gerard Holzmann, Philip Homburg, Peter Honeyman, Raj Jain, Dave Johnson, Charlie Kaufman, Vinay Kumar, Jorg Liebeherr, Paul Mockapetris, Carol Orange, Craig Partridge, Charlie Perkins, Thomas Powell, Greg Sharp, Anne Steegstra, George Swallow, Mark Taylor, Peter van der Linden, Hans van Staveren, Maarten van Steen, Kees Verstoep, Stephen Walters, Michael Weintraub, Joseph Wilkes e Stephen Wolff.

Un ringraziamento speciale va a Radia Perlman per molti suggerimenti costruttivi. Anche i miei studenti mi hanno aiutato in molti modi. Mi piace ricordare per la loro assistenza Martijn Bot, Wilbert de Graaf, Flavio del Pomo e Arnold de Wit.

Mary Franz è stata il mio assistente editoriale presso la Prentice Hall, e mi ha fornito più materiale di lettura di quanto ne abbia potuto consumare negli ultimi dieci anni. Mi ha aiutato anche in molti altri modi, piccoli, medi, grandi e giganteschi. Camille Trentacoste è stata il produttore editoriale che mi ha insegnato con umiltà molte cose importanti, anche se aveva a che fare con un autore pestifero e un piano di lavorazione a brevi scadenze. Infine, ringrazio le persone più importanti. Suzanne, Barbara, Marvin e persino il piccolo Bram avevano già sperimentato tutto questo in passato. Mi sopportano con infinita pazienza e buona grazia. Grazie.

Capitolo 1

INTRODUZIONE

Ciascuno degli ultimi tre secoli è stato dominato da una sola tecnologia. Il XVIII secolo fu il tempo dei grandi sistemi meccanici che accompagnarono la rivoluzione industriale. Il XIX secolo fu l'età della macchina a vapore. Durante il XX secolo, la tecnologia chiave è stata la raccolta di informazioni, la loro elaborazione e distribuzione. Fra altri sviluppi, abbiamo visto l'installazione della rete telefonica mondiale, l'invenzione di radio e televisione, la nascita e la crescita senza precedenti dell'industria dei calcolatori, e il lancio di satelliti per la comunicazione.

A causa del progresso tecnologico, queste aree stanno rapidamente convergendo, e le differenze fra raccolta, memorizzazione ed elaborazione di informazioni stanno scomparendo velocemente. Organizzazioni con centinaia di uffici sparsi su di una vasta area geografica prevedono d'abitudine di esaminare lo stato corrente anche del loro più remoto avamposto solo premendo un pulsante. Man mano che la nostra capacità di raccolta, elaborazione e distribuzione di informazione aumenta, la domanda di elaborazione di informazione sempre più sofisticata aumenta ancor più velocemente.

Sebbene l'industria dei calcolatori sia giovane a confronto di altre industrie (per esempio automobili e trasporti aerei), i calcolatori hanno fatto progressi spettacolari in breve tempo. Durante le prime due decadi della loro esistenza, i sistemi di calcolatori erano fortemente centralizzati, normalmente in una singola grande stanza. Non di rado, questa stanza aveva pareti di vetro, attraverso le quali i visitatori potevano osservare la grande meraviglia elettronica. Una azienda di medie dimensioni o una università poteva avere uno o due calcolatori, mentre grandi istituti ne avevano al più qualche dozzina. L'idea che entro venti anni calcolatori con uguale potenza più piccoli di un francobollo potessero essere prodotti in massa a milioni, era pura fantascienza.

La fusione di calcolatori e comunicazione ha avuto una profonda influenza sul modo con cui i sistemi di elaboratori sono organizzati. La nozione di "centro di calcolo" come stanza con un grande calcolatore al quale gli utenti portano il proprio lavoro da elaborare, è oggi totalmente obsoleta. Il modello obsoleto di un singolo calcolatore che risolve tutte le necessità computazionali di una organizzazione è stato rimpiazzato da uno in cui un gran numero di calcolatori *indipendenti* ma *interconnessi* eseguono il lavoro. Questi sistemi sono chiamati **reti di calcolatori**. Il progetto e l'organizzazione di tali reti sono gli argomenti di questo libro.

Nel libro si userà il termine "rete di calcolatori" per indicare una collezione di calcolatori autonomi collegati. Si dice che due calcolatori sono collegati se sono capaci di scambiare informazioni. Il collegamento non deve obbligatoriamente passare attraverso filo di rame; possono essere utilizzate anche fibre ottiche, microonde o comunicazioni

satellitari. Richiedendo che i calcolatori siano autonomi, si desidera escludere dalla nostra definizione sistemi in cui esiste una chiara relazione padrone/schiavo (master/slave). Se un calcolatore può forzatamente accendere, bloccare o controllarne un altro, i calcolatori non sono autonomi. Un sistema con un'unità adibita a controllo e diverse componenti sottomesse non è una rete; e non lo è nemmeno un grosso elaboratore con terminali e stampanti remote.

C'è una considerevole confusione nella letteratura fra reti di calcolatori e **sistemi distribuiti**. La distinzione principale è che in un sistema distribuito l'esistenza di molteplici calcolatori autonomi è trasparente (cioè non visibile) all'utente. Quando si scrive un comando per eseguire un programma questo viene eseguito. È compito del sistema operativo selezionare il miglior processore, trovarlo e trasportare tutti gli archivi di ingresso a quel processore, e inserire i risultati nella posizione appropriata. In altre parole, l'utente di un sistema distribuito non è a conoscenza del fatto che esistono molteplici processori; appare tutto come un unico processore virtuale. L'allocazione delle elaborazioni ai processori e degli archivi sui dischi, il movimento degli archivi da dove sono memorizzati a dove sono richiesti, e tutte le altre funzioni di sistema devono essere automatiche.

Con una rete, gli utenti devono *esplicitamente* collegarsi a una macchina, *esplicitamente* richiedere elaborazioni remote, *esplicitamente* spostare archivi e generalmente occuparsi personalmente della gestione della rete. Con un sistema distribuito, nulla deve essere fatto esplicitamente; è tutto fatto automaticamente dal sistema operativo senza che l'utente ne sia a conoscenza.

In concreto, un sistema distribuito è un sistema software costruito per una rete. Il software fornisce un alto grado di trasparenza e coesione. Quindi la distinzione fra una rete e un sistema distribuito sta nel software (specialmente nel sistema operativo), più che nell'hardware.

Ciò nonostante, esiste una considerevole intersezione fra i due argomenti. Per esempio, sia i sistemi distribuiti che le reti di calcolatori hanno bisogno di spostare archivi. La differenza sta in chi invoca lo spostamento: il sistema o l'utente. Anche se questo libro tratta in principal modo le reti, molti degli argomenti sono importanti anche per i sistemi distribuiti. Per maggiori informazioni sui sistemi distribuiti si vedano Couloris *et al.*, (1994); Mullender (1993); Tanenbaum (1995).

1.1 Reti di calcolatori

Prima di iniziare a esaminare in dettaglio gli aspetti tecnici, val la pena spendere un po' di tempo per discutere i motivi per cui siamo interessati alle reti di calcolatori e a come si possono usare.

1.1.1 Reti per aziende

Molte organizzazioni impiegano un notevole numero di elaboratori, spesso posti in luoghi molto lontani fra loro. Per esempio, un'azienda con molte fabbriche può avere un elaboratore in ogni fabbrica per gestire il magazzino, per controllare la produzione, e per preparare localmente gli stipendi. Inizialmente, ognuno di questi elaboratori lavorava in modo isolato dagli altri, ma a un certo punto, la direzione generale potrebbe decidere di collegarli allo scopo di estrarre e correlare informazioni riguardanti l'intera azienda.

Detto in termini più generali, lo scopo in questo caso è di **condividere risorse**, e di rendere disponibili a tutti i programmi, le attrezzature, e specialmente i dati sulla rete indipendentemente dalla posizione fisica delle risorse e dell'utente. In altre parole, il fatto che un utente possa essere 1000 km lontano dai suoi dati non dovrebbe impedirgli di poterli usare pensandoli come locali. Questo scopo può essere sintetizzato dicendo che è un tentativo per far terminare la "tirannia della geografia".

Un secondo scopo è di raggiungere **alta affidabilità** mediante fonti alternative di supporto. Per esempio, tutti gli archivi potrebbero essere replicati su due o tre macchine, così se una di esse risultasse non disponibile (a causa di un guasto hardware), potrebbero essere usate le altre copie. In più, la presenza di più processori significa che se uno non funziona, gli altri potrebbero essere in grado di sobbarcarsi il suo lavoro, a costo di una riduzione delle prestazioni. Per le strutture militari, le banche, il controllo del traffico aereo, la sicurezza dei reattori nucleari, e per tante altre applicazioni, la capacità di continuare a operare in presenza di problemi hardware è di vitale importanza.

Un ulteriore scopo è quello di **risparmiare denaro**. I piccoli elaboratori hanno un rapporto costo/prestazioni migliore di quelli più grandi. I mainframe (calcolatori aventi le dimensioni di una stanza) sono circa dieci volte più veloci dei personal computer, ma costano mille volte tanto. Questo sbilanciamento ha spinto molti progettisti di sistemi a costruire sistemi costituiti da personal computer, uno per ciascun utente, con dati memorizzati su una o più macchine (**file server**) condivise. In questo modello, gli utenti sono chiamati **client** (*client*), e il tipo di organizzazione è detto **modello client-server**. Esso è illustrato in figura 1-1.

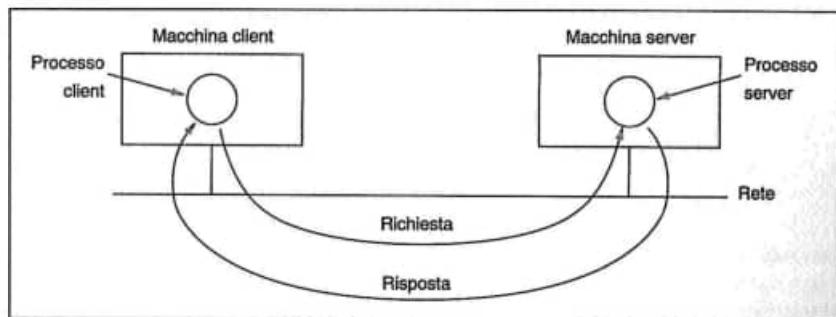


Fig. 1-1 Il modello client-server.

Nel modello client-server la comunicazione generalmente ha la forma di un messaggio a un server da parte di un client che chiede di eseguire un certo lavoro. Il server allora esegue il lavoro e spedisce indietro la risposta. Normalmente ci sono molti client che usano un piccolo numero di server.

Un altro scopo delle reti è la **scalabilità**, cioè l'abilità di incrementare le prestazioni del sistema in modo graduale con l'incrementare del carico di lavoro aggiungendo più pro-

cessori. Con i mainframe centralizzati, quando il sistema è saturo, deve essere sostituito da uno più grande, normalmente più costoso e anche più faticoso da imparare per gli utenti. Con il modello client-server, nuovi client possono essere aggiunti a richiesta. Un altro scopo per cui si crea una rete di calcolatori ha poco a che vedere con la tecnologia. Una rete di calcolatori può fornire un potente **mezzo di comunicazione** fra impiegati molto lontani. Usando una rete, è facile per due o più persone che vivono molto lontano scrivere insieme una relazione. Quando una effettua una modifica a un documento, l'altra può vedere la variazione immediatamente, invece di aspettare diversi giorni per una lettera. Questo incremento di velocità fa diventare la cooperazione fra gruppi di persone facile proprio dove in precedenza era impossibile. A lungo andare, l'utilizzo di reti per migliorare la comunicazione uomo-a-uomo sarà probabilmente più importante di altri obiettivi quali il miglioramento dell'affidabilità.

1.1.2 Reti per le persone

Le motivazioni date in precedenza per la costruzione di reti di calcolatori sono essenzialmente di carattere economico e di natura tecnologica. Se fossero disponibili mainframe sufficientemente grandi e potenti a un prezzo accettabile, la maggior parte delle aziende potrebbe scegliere di mantenervi tutti i propri dati e di fornire agli impiegati terminali collegati a essi. Negli anni settanta e nei primi anni ottanta, la maggior parte delle aziende operarono in questo modo. Le reti di calcolatori divennero popolari solamente quando le reti di personal computer offrirono un notevole vantaggio in termini di costi/prestazioni rispetto ai mainframe.

A partire dagli anni novanta le reti di calcolatori iniziarono a fornire servizi a singoli individui. Questi servizi e le motivazioni per utilizzarli sono parzialmente differenti da quelli del modello "societario" descritto nel paragrafo precedente. Nel seguito si discutono tre tra le motivazioni più interessanti:

1. Accesso a informazioni remote.
2. Comunicazione uomo-a-uomo.
3. Intrattenimento interattivo.

L'accesso a informazioni remote si ha in varie forme. Un'area in cui sta già avvenendo è l'accesso a servizi finanziari. Molta gente paga le fatture, gestisce il proprio conto corrente, e si occupa dei propri investimenti elettronicamente. Anche gli acquisti eseguiti da casa stanno diventando popolari, con la possibilità di ispezionare i cataloghi in linea di centinaia di aziende. Alcuni di questi cataloghi forniscono la possibilità di visionare istantaneamente un filmato del prodotto semplicemente selezionando il suo nome.

I giornali saranno anch'essi in linea e personalizzati. Sarà possibile chiedere al giornale che si desidera qualsiasi cosa riguardante politici corrotti, grandi incendi, scandali su celebrità o epidemie, ma niente calcio, grazie. Di notte, mentre si dorme, il giornale verrà scaricato sul disco del proprio elaboratore o stampato sulla stampante laser. In piccola scala questo tipo di servizi già esiste. Il prossimo passo verso giornali (o riviste e giornali scientifici) è la libreria digitale in linea. A causa del costo, dimensione, e peso di elaboratori aventi la dimensione di un libro, i libri stampati diventeranno obsoleti. Gli scettici

dovrebbero pensare agli effetti che il sistema di stampa ebbe sui manoscritti medievali miniati.

Un'altra applicazione che rientra in questa categoria è l'accesso ai sistemi di informazioni quali il World Wide Web attuale, che contiene informazioni a proposito di arte, affari, cucina, politica, salute, passatempi, scienze, sport, viaggi, e tanti altri argomenti.

Tutte le applicazioni suddette coinvolgono interazioni fra una persona e una base di dati remota. La seconda grande categoria di utilizzo di reti saranno le interazioni uomo-a-uomo, praticamente la risposta del XXI secolo al telefono del XIX secolo. La posta elettronica o **email** è già largamente usata da milioni di persone e presto conterrà audio e video così come testo. Ci vorrà un po' più di tempo per inserire anche odori nei messaggi.

La posta elettronica in tempo reale permetterà a utenti remoti di comunicare senza ritardi, forse anche vedendosi e ascoltandosi. Questa tecnologia rende possibile incontri virtuali, chiamati **videoconferenze**, fra gente lontana. Di tanto in tanto si dice che i trasporti e la comunicazione stanno facendo una gara, e chi di loro vincerà renderà l'altro obsoleto. Gli incontri virtuali potrebbero essere usati per scuole geograficamente remote, per avere opinioni mediche da specialisti distanti, o per numerose altre applicazioni.

I gruppi di discussione a livello mondiale (**newsgroup**), con discussioni su ogni argomento immaginabile sono già luoghi frequentati fra un gruppo selezionato di persone, che aumenteranno fino a includere tutta la popolazione. Queste discussioni, in cui una persona invia un messaggio e tutti i sottoscrittori di un certo gruppo di discussione possono leggerlo, coprono l'intera gamma degli argomenti possibili, dall'umorismo alla passione. La terza categoria è l'intrattenimento, che è una grande industria in crescita. L'applicazione cruciale (quella che potrebbe dominare su tutto il resto) è il video a richiesta. Entro una decina d'anni o poco più, potrà essere possibile selezionare un qualsiasi film o un programma televisivo prodotto ovunque, in una nazione qualsiasi, e vederlo visualizzato sul proprio video istantaneamente. I nuovi film potranno essere interattivi, in cui l'utente verrà occasionalmente interpellato per decidere il seguito della storia (dovrebbe Macbeth assassinare Duncan adesso oppure aspettare un momento più opportuno?) con scenari alternativi forniti nei diversi casi. La televisione in diretta potrebbe anch'essa diventare interattiva, con gli spettatori che partecipano a spettacoli a quiz, scegliendo fra vari correnti, e così via.

D'altra parte, forse l'applicazione cruciale non sarà il video a richiesta. Forse potrebbero essere i giochi. Già esistono giochi di simulazione in tempo reale per più persone, come esplorazioni di prigioni sotterranee, o simulatori di volo multiutente con i giocatori di una squadra che cercano di abbattere i giocatori della squadra opposta. Usando visualizzazioni in grafica tridimensionale in tempo reale, e qualità fotografica di immagini in movimento, si avrà un tipo di realtà virtuale condivisa in tutto il mondo.

In breve, la possibilità di offrire insieme informazioni, comunicazioni, e intrattenimenti darà sicuramente vita a una nuova industria di massa basata su reti di calcolatori.

1.1.3 Aspetti sociali

La diffusione su vasta scala delle reti introdurrà nuovi problemi sociali, etici, e politici (Laudon, 1995). Ne menzioniamo brevemente alcuni; uno studio più approfondito riche-

derebbe un intero libro, come minimo. Una caratteristica popolare di molte reti sono i *gruppi di discussione* (newsgroup), o *bacheche elettroniche* (bulletin board), in cui gli utenti possono scambiare messaggi con altri utenti che hanno i medesimi interessi. Fino a quando gli argomenti sono ristretti a temi tecnici o passatempi quali il giardinaggio, non sorgono troppi problemi.

I problemi iniziano quando i gruppi di discussione riguardano temi che interessano particolarmente la gente quali politica, religione o sesso. Opinioni inviate a questi gruppi potrebbero essere fortemente offensive per qualcuno. Inoltre, i messaggi non si limitano a testi. Fotografie in alta risoluzione e anche brevi filmati possono essere già trasmessi su rete. Alcune persone seguono una filosofia del tipo vivi-e-lascia-vivere, ma altri pensano che inviare certo materiale (per esempio pornografia infantile) è inaccettabile. Quindi il dibattito infuria.

Sono stati chiamati in causa gli operatori delle reti, affermando che sono responsabili del contenuto di quello che esse trasportano, alla stessa stregua di giornali e riviste. La risposta inevitabile è che una rete è come una società telefonica o un ufficio postale e che non ci si può attendere che si vigili su cosa i propri utenti esprimono. In termini ancora più forti, se gli operatori delle reti fossero eletti a censori di messaggi ciò causerebbe probabilmente la cancellazione di qualsiasi cosa che possa anche minimamente chiamarli in giudizio, e questo andrebbe a violare i diritti degli utenti di parlare liberamente. È probabilmente giusto asserire che tale dibattito andrà avanti ancora per molto.

Un'altra area riguarda i diritti degli impiegati rispetto ai diritti dei datori di lavoro. Molti impiegati leggono e scrivono messaggi di posta elettronica ogni giorno al lavoro. Alcuni datori di lavoro reclamano il diritto di leggere e possibilmente censurare i messaggi dei propri impiegati, inclusi i messaggi inviati da un terminale a casa fuori orario di lavoro. Non tutti gli impiegati sono d'accordo (Sipior, Ward, 1995).

Anche se i datori di lavoro hanno potere sui propri impiegati, questo tipo di relazione può essere estesa a università e studenti? E per quanto riguarda le scuole superiori? Nel 1994, l'università Carnegie-Mellon decise di bloccare il flusso dei messaggi entranti riguardanti diversi gruppi di discussione che avevano a che fare con il sesso in quanto l'università aveva la sensazione che il materiale fosse non appropriato per i minorenni (cioè gli studenti aventi meno di 18 anni). Le conseguenze di questo evento richiederanno anni per stabilizzarsi.

Le reti di calcolatori offrono la potenzialità di spedire messaggi anonimi. In alcuni casi, questa possibilità potrebbe essere desiderabile. Per esempio, si fornisce la possibilità agli studenti, ai soldati, agli impiegati, e ai cittadini di segnalare comportamenti illegali da parte dei professori, ufficiali, superiori, e politici senza il rischio di repressioni. D'altra parte, negli Stati Uniti e in molte altre democrazie, la legge permette a un accusato il diritto di confrontarsi e scontrarsi con il proprio accusatore in tribunale. Le accuse anonymously non possono essere utilizzate come prove.

In breve, le reti di calcolatori, come la carta stampata 500 anni fa, permettono a cittadini ordinari di distribuire le proprie idee in diversi modi e a differenti ascoltatori rispetto a quanto fosse possibile in precedenza. Questa nuova libertà porta con sé molti problemi irrisolti a livello sociale, politico e morale. La soluzione di questi problemi è lasciata come esercizio al lettore.

1.2 Hardware delle reti

Si può ora spostare la nostra attenzione dalle applicazioni e dai problemi sociali delle reti agli aspetti tecnologici che coinvolgono la loro progettazione. Non esiste una tassonomia generalmente accettata in cui tutte le reti possono essere catalogate, ma due aspetti risultano avere particolare importanza: la tecnologia di trasmissione e la scala. Esamineremo nel seguito ciascuna di queste separatamente.

Comunemente parlando, esistono due tipi di tecnologia per la trasmissione:

1. Reti a diffusione globale (broadcast).
2. Reti punto-a-punto (point-to-point).

Le reti a broadcast hanno un unico canale di comunicazione che è condiviso da tutte le macchine della rete. Brevi messaggi, chiamati **pacchetti** in alcuni contesti, inviati da una qualsiasi macchina vengono ricevuti da tutte le altre. Un campo indirizzo all'interno del pacchetto indica a chi esso è diretto. Dopo la ricezione di un pacchetto, una macchina controlla il campo indirizzo. Se il pacchetto è diretto alla macchina stessa, esso lo elabora; se il pacchetto è diretto a un'altra macchina, esso viene ignorato.

Per analogia, si può immaginare a un capo di un corridoio sul quale si affacciano tante stanze qualcuno che urla: "Watson, vieni qua. Ti voglio". Anche se il pacchetto può essere ricevuto (sentito) da molta gente, solo Watson risponde. Un altro esempio è un annuncio all'aeroporto che richiede a tutti i passeggeri del volo 644 di portarsi alla porta 12.

I sistemi a broadcast generalmente permettono anche la possibilità di indirizzare un pacchetto a *tutte* le destinazioni usando un codice speciale nel campo indirizzo. Quando viene trasmesso un pacchetto con questo codice, esso viene ricevuto ed elaborato da tutte le macchine sulla rete. Questo modo di operare viene chiamato **broadcasting**. Alcuni sistemi a broadcast supportano anche trasmissione a un sottoinsieme delle macchine, concetto conosciuto come **multicasting**. Uno schema possibile consiste nel riservare un bit per indicare multicasting. I rimanenti $n-1$ bit di indirizzo possono contenere un numero di gruppo. Ogni macchina si può "iscrivere" a uno o a tutti i gruppi. Quando un pacchetto viene inviato a un certo gruppo, esso è spedito a tutte le macchine iscritte a tale gruppo. Al contrario, le reti **point-to-point** consistono di molte connessioni fra coppie individuali di macchine. Per andare dal mittente al destinatario, un pacchetto su questo tipo di rete potrebbe dover visitare una o più macchine intermedie. Spesso sono possibili parecchi cammini di diversa lunghezza, quindi nelle reti punto a punto giocano un ruolo importante gli algoritmi di ricerca del cammino. Come regola generale (ci sono tuttavia diverse eccezioni), reti piccole, geograficamente localizzate usano il broadcast, mentre le reti più grandi sono normalmente punto-a-punto.

Un criterio alternativo per classificare le reti è legato alla loro scala. In figura 1-2 si fornisce una classificazione di sistemi con parecchi processori in base alla loro dimensione fisica. Al livello superiore ci sono le **macchine a flusso di dati** (macchine data flow), calcolatori altamente paralleli con molte unità funzionali che lavorano sul medesimo programma. Poi vengono i **multicomputer**, sistemi che comunicano inviando messaggi lungo canali molto corti e molto veloci. Oltre i multicomputer ci sono le vere reti, fatte di calcolatori che comunicano mediante scambio di messaggi lungo lunghi cavi. Le reti

Distanza fra processori	Processori situati nella stessa	Esempio
0.1 m	Circuito	Macchina data flow
1 m	Sistema	Multicomputer
10 m	Stanza	Rete locale (LAN)
100 m	Edificio	
1 km	Università	Rete metropolitana (MAN)
10 km	Città	
100 km	Nazione	Rete geografica (WAN)
1.000 km	Continente	
10.000 km	Pianeta	Internet

Fig. 1-2 Classificazione della scala di interconnessione.

possono essere divise in reti locali, metropolitane, e geografiche. In ultimo, la connessione di due o più reti è chiamata **internetwork**. La rete mondiale Internet è un esempio ben conosciuto di internetwork. La distanza è un importante metro di classificazione perché tecniche differenti sono usate su diverse scale. In questo libro ci si concentrerà solamente con le vere reti e le loro interconnessioni. Di seguito si fornisce una breve introduzione sull'argomento dell'hardware delle reti.

1.2.1 Reti locali

Le **reti locali**, generalmente chiamate LAN (local area network), sono reti private all'interno di un singolo edificio o una università, di dimensione al più di qualche chilometro. Esse sono molto utilizzate per collegare i personal computer e le stazioni di lavoro degli uffici o degli stabilimenti delle aziende per permettere la condivisione di risorse (per esempio stampanti) e lo scambio di informazioni. Le LAN si distinguono dagli altri tipi di rete per tre caratteristiche: (1) le loro dimensioni, (2) la loro tecnologia di trasmissione e (3) la loro topologia.

Le LAN sono di dimensioni ridotte, che significa che il tempo peggiore di comunicazione è limitato e conosciuto a priori. La conoscenza di questo limite permette l'uso di alcuni tipi di schemi di progetto che non sarebbe possibile utilizzare altrimenti. Inoltre, permette di semplificare la gestione della rete.

Le LAN spesso usano una tecnologia di trasmissione che usa un solo cavo al quale tutte le macchine sono collegate, come la linea telefonica autonoma locale una volta usata nelle aree rurali. Le LAN tradizionali permettono velocità di trasmissione che vanno da 10 a 100 Mbps, hanno un basso ritardo (decine di microsecondi), e fanno pochi errori. Le LAN più recenti possono operare a velocità più elevate, fino a centinaia di megabit/s. In questo libro si aderisce alla tradizione misurando le velocità di linea

in megabit/s (Mbps), e non megabyte/s (MB/s). Un megabit è 1.000.000 di bit, non 1.048.576 (2^{20}) bit.

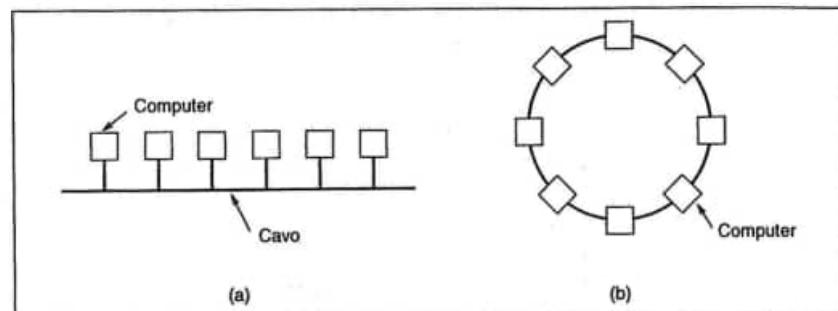


Fig. 1-3 Due reti broadcast. (a) Bus. (b) Anello.

Sono possibili differenti tipologie per le reti broadcast. La figura 1-3 ne mostra due. In una rete a bus (cioè a cavo lineare), in ogni istante una sola macchina è il master e le è permesso di trasmettere. A tutte le altre macchine è richiesto di trattenersi dal trasmettere. È necessario un meccanismo di arbitraggio per risolvere conflitti quando due o più macchine vogliono trasmettere simultaneamente. Il meccanismo può essere centralizzato o distribuito. IEEE 802.3, popolarmente chiamato **Ethernet**, è una rete broadcast basata su bus con controllo decentralizzato operante a 10 o 100 Mbps. I calcolatori su una Ethernet possono trasmettere ogni qualvolta lo desiderino: se due o più pacchetti generano una collisione, ogni calcolatore attende un tempo casuale e ritenta la trasmissione.

Un secondo tipo di sistema broadcast è l'anello (ring). In un anello, ogni bit viene diffuso per suo conto, senza aspettare il resto del pacchetto a cui esso appartiene. Tipicamente, ogni bit circumnaviga l'intero anello nel tempo richiesto per trasmettere pochi bit, spesso prima che l'intero pacchetto venga spedito. Come tutti gli altri sistemi broadcast, qualche regola è necessaria per arbitrare accessi simultanei all'anello. Metodi vari sono in uso e saranno discussi più avanti nel libro. IEEE 802.5 (la IBM token ring), è una LAN popolare basata su anello che opera a 4 e 16 Mbps.

Le reti broadcast possono essere ulteriormente catalogate in statiche e dinamiche, a seconda di come il canale viene allocato. Un'allocazione statica tipica è suddividere il tempo in intervalli discreti (quanti) ed eseguire un algoritmo che permette a ogni macchina il broadcast solamente quando si presenta il suo quanto. L'allocazione statica spreca banda di trasmissione quando una macchina non ha nulla da spedire durante il suo quanto; per questo motivo, la maggior parte dei sistemi cerca di allocare i canali dinamicamente (cioè su richiesta).

I metodi di allocazione dinamica per un canale condiviso sono centralizzati o decentralizzati. Nei metodi di allocazione centralizzati c'è una singola entità, per esempio un'unità di arbitraggio su bus, che determina chi deve essere il prossimo a proseguire. Questo può essere realizzato accettando richieste e prendendo decisioni seguendo un qualche algorit-

mo interno. Nei metodi di allocazione dei canali decentralizzati non esiste un'entità centrale; ogni macchina deve decidere autonomamente se spedire oppure no. Si potrebbe pensare che questo porti al caos, ma non è vero. Più avanti si studieranno molti algoritmi progettati per eliminare caos potenziali.

L'altro tipo di LAN viene costruito utilizzando linee punto-a-punto. Linee individuali collegano una macchina specifica a un'altra. Tale LAN è in realtà una rete geografica in miniatura. Questa si studierà più avanti.

1.2.2 Reti metropolitane

Una **rete metropolitana**, o **MAN** (metropolitan area network), è sostanzialmente una versione ingrandita della LAN e normalmente usa tecnologie simili. Essa può coprire un gruppo di uffici vicini della medesima azienda oppure una città e può essere privata o pubblica. Una MAN può supportare sia dati che voci, e può anche essere collegata alla rete locale di televisione via cavo. Una MAN ha solamente uno o due cavi e non contiene elementi di scambio, per spedire i pacchetti su una fra le più potenziali linee di uscita. Il fatto di non dover effettuare queste azioni di scelta semplifica il progetto.

La ragione principale per distinguere anche le MAN come categorie speciali è che è stato adottato uno standard, che è stato implementato. Esso è chiamato **DQDB** (**Distributed Queue Dual Bus**), o per chi preferisce i numeri alle lettere, 802.6 (il numero dello standard IEEE che lo identifica). DQDB include due bus (cavi) a cui tutti i calcolatori sono collegati, come mostra la figura 1-4. Ogni bus ha un terminatore, un dispositivo che inizializza l'attività di trasmissione. Il traffico che è destinato a un calcolatore alla destra del mittente usa il bus superiore. Il traffico verso sinistra usa quello inferiore.

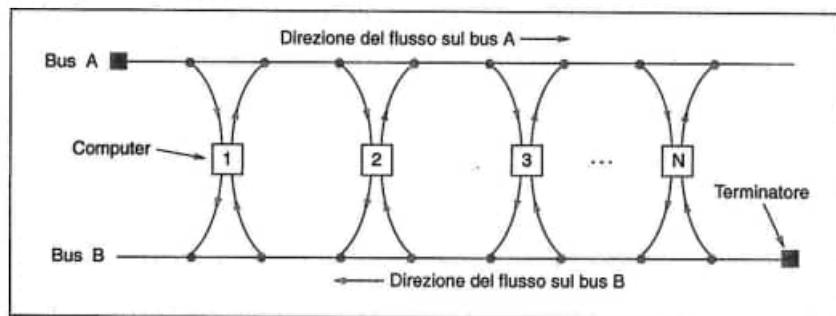


Fig. 1-4 Architettura della rete metropolitana DQDB.

Un aspetto chiave di una MAN è che esiste un mezzo di broadcast (per 802.6, a due cavi) a cui tutti i calcolatori sono connessi. Questo semplifica moltissimo la progettazione in confronto ad altri tipi di rete. Si discuterà in dettaglio il DQDB nel capitolo 4.

1.2.3 Reti geografiche

Una **rete geografica**, o **WAN** (wide area network), copre una grande area geografica, spesso una nazione o un continente. Essa contiene una collezione di macchine adibite

all'esecuzione di programmi (cioè applicazioni) per gli utenti. Si seguirà la trattazione tradizionale e si chiameranno queste macchine **host** (ospiti). Anche il termine **sistema terminale** è usato qualche volta in letteratura. Gli host sono collegati da una **sottorete di comunicazione**, o più brevemente una **sottorete**. Il compito di una sottorete è di trasportare messaggi da host a host, così come il sistema telefonico trasporta parole fra colui che parla e colui che ascolta. Separare gli aspetti di pura comunicazione della rete (la sottorete) dagli aspetti applicativi (gli host), la progettazione globale della rete risulta notevolmente semplificata.

Nella maggior parte delle reti geografiche la sottorete è costituita da due componenti distinte: le linee di trasmissione e gli elementi di commutazione. Le linee di trasmissione (chiamate anche **circuiti**, **canali** o **dorsali**) spostano bit fra le varie macchine.

Gli elementi di commutazione sono calcolatori specializzati usati per collegare due o più linee di trasmissione. Quando i dati arrivano su una linea di ingresso, l'elemento di commutazione deve scegliere una linea di uscita per farli proseguire. Sfortunatamente, non c'è una terminologia standard per denominare questi calcolatori. Essi sono chiamati **nodi per lo scambio dei pacchetti**, **sistemi intermedi**, **commutatori di dati** e in molti altri modi. Un termine generico che useremo per i calcolatori commutatori è **router**, ma informiamo il lettore che non c'è consenso su questa terminologia. In questo modello, mostrato in figura 1-5, ogni host è generalmente collegato a una LAN su cui è presente un router. L'insieme costituito dalle linee di comunicazione e dai router (e non dagli host) forma una sottorete.

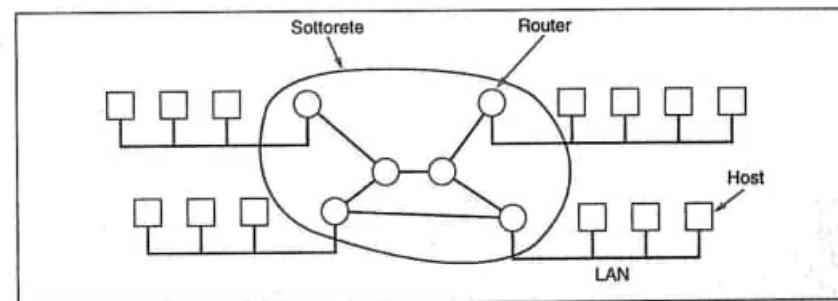


Fig. 1-5 Relazione fra host e sottorete.

Val la pena di dire qualcosa a proposito del termine "sottorete". Originariamente, il suo unico significato era legato alla collezione dei router e delle linee di comunicazione che muovevano i pacchetti dall'host mittente all'host destinatario. Tuttavia, qualche anno dopo, esso acquistò un secondo significato in congiunzione con le reti a indirizzamento (che discuteremo nel capitolo 5). Quindi il termine risulta ambiguo. Sfortunatamente, non ci sono termini alternativi largamente diffusi per il significato iniziale, così con qualche esitazione si farà uso del termine in entrambi i significati. Il suo significato risulterà chiaro dal contesto.

Nella maggior parte delle WAN la rete contiene numerosi cavi o linee telefoniche, una

per coppia di router. Se due router che non condividono un cavo desiderano comunicare, essi devono farlo in modo indiretto, attraverso altri router. Quando un pacchetto è spedito da un router a un altro attraverso uno o più router intermedi, il pacchetto viene ricevuto interamente da ogni router intermedio, memorizzato fino a che la linea in uscita richiesta non risulti libera, e quindi viene fatto proseguire. Una sottorete che usa questo principio è detta **point-to-point** (reti punto a punto), **store-and-forward** (memorizza-e-fai-proseguire) o sottorete a **packed switched** (commutazione pacchetto). Quasi tutte le reti geografiche (eccetto quelle che utilizzano i satelliti) hanno reti del tipo store-and-forward. Quando i pacchetti sono piccoli e tutte delle stesse dimensioni sono spesso chiamati **celle**. Quando viene usata una rete punto-a-punto, un importante aspetto del progetto riguarda la topologia di interconnessione dei router. La figura 1-6 mostra diverse possibili topologie. Le reti locali hanno di solito una topologia simmetrica. Invece le reti geografiche hanno tipicamente topologie irregolari.

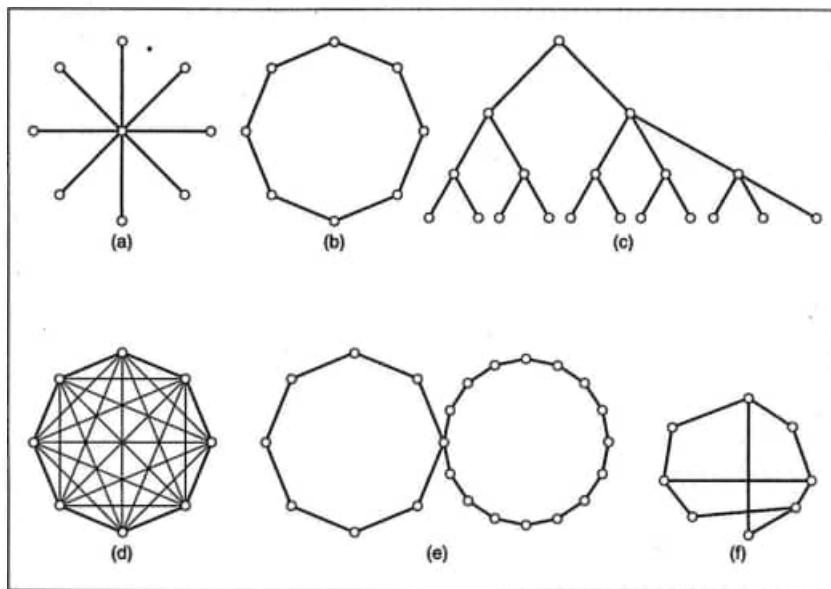


Fig. 1-6 Alcune topologie per sottoreti punto-a-punto. (a) Stella. (b) Anello. (c) Albero. (d) Completa. (e) Anelli secanti. (f) Irregolare.

Una seconda possibilità per una WAN è quella di utilizzare un satellite o il sistema radio terrestre. Ogni router ha una antenna attraverso cui spedire e ricevere. Tutti i router possono ascoltare le trasmissioni *del satellite*, e in alcuni casi anche quelle dei singoli router *verso il satellite*. Qualche volta i router sono collegati a una sottorete point-to-point, e solo alcuni di essi possiedono una antenna satellitare. Le reti a satellite sono fondamen-

talmente a broadcast e sono tanto più utili quanto più risulta importante la proprietà di broadcast.

1.2.4 Reti senza filo

I calcolatori portatili, come i notebook o i personal digital assistant (PDA), sono il segmento con maggior crescita dell'industria dei calcolatori. Molti dei possessori di questi calcolatori hanno macchine sulla loro scrivania dell'ufficio collegate a LAN o WAN e desiderano collegarsi a esse anche quando sono a casa propria o in viaggio. Visto che una connessione a filo è impossibile in macchina o in aeroplano, c'è un forte interesse per le reti senza filo. In questo paragrafo si introduce brevemente questo argomento. (Nota: per paragrafo intendiamo una porzione di libro con un numero costituito da tre parti come ad es. 1.2.4.)

In realtà, le comunicazioni digitali senza filo non sono un'idea nuova. Nel 1901, il fisico italiano Guglielmo Marconi presentò un telegrafo senza fili fra nave e terra ferma usando il codice Morse (punti e linee sono simboli binari, dopo tutto). I moderni sistemi digitali senza fili hanno prestazioni migliori, ma l'idea di base è la medesima. Informazioni addizionali su questi sistemi possono essere trovate in Garg, Wilkes (1996); Pahlavan *et al.* (1995).

Le reti senza filo hanno molti usi. Un uso comune è l'ufficio portatile. La gente che si trova in viaggio spesso vuole usare il proprio equipaggiamento elettronico portatile per spedire e ricevere chiamate telefoniche, fax, posta elettronica, leggere archivi remoti, collegarsi a calcolatori remoti, e così via, e lo vuole fare da qualsiasi parte su terra, mare o in aria.

Reti senza filo sono molto utili a gruppi di autocarri, taxi, autobus e personale di manutenzione per tenersi in collegamento. Un altro uso è per i soccorritori di eventi catastrofici (incendi, inondazioni, terremoti ecc.) in cui il sistema telefonico è stato distrutto. In queste occasioni i calcolatori possono spedire messaggi, conservare informazioni, e così via.

Infine, reti senza filo sono molto utili ai militari. Se occorre essere capaci di combattere una guerra ovunque sulla Terra con breve preavviso, contare sull'utilizzo delle infrastrutture locali non è certo una buona idea. È meglio portarsene dietro.

Anche se le reti senza filo e i calcolatori portatili sono spesso collegati, essi non sono la stessa cosa, come mostra la figura 1-7. I calcolatori portatili sono qualche volta collegati con filo. Per esempio, se un viaggiatore collega il proprio calcolatore portatile a una presa telefonica in un hotel, si ha mobilità ma non una rete senza filo. Un altro esempio è qualcuno che usa un computer portatile per ispezionare problemi tecnici su un treno. In questo caso può trascinarsi dietro un lungo cavo (modello aspirapolvere).

D'altra parte, alcuni calcolatori senza filo non sono portatili. Un importante esempio in questo caso è una azienda che possiede un vecchio stabile che non ha cavi di collegamento e desidera collegare i propri calcolatori. L'installazione di una LAN senza filo potrebbe richiedere poco più che comprare una piccola scatola con dell'elettronica e installare qualche antenna. Questa soluzione potrebbe essere più economica che riempire di cavi lo stabile.

Anche se le LAN senza filo sono facili da installare, esse hanno qualche svantaggio. Tipicamente esse hanno una capacità di 1-2 Mbps, che è più bassa delle LAN con filo.

Senza filo	Mobile	Applicazioni
No	No	Postazioni di lavoro fisse negli uffici
No	Sì	Uso di un portatile in un hotel; manutenzione su un treno
Sì	No	LAN in vecchi edifici senza fili di collegamento
Sì	Sì	Ufficio portatile; PDA per inventario in un negozio

Fig. 1-7 Combinazioni di reti senza filo e calcolatori portatili.

Inoltre, il rapporto di errore è spesso più alto, e le trasmissioni fra due calcolatori possono interferire con un terzo.

Ma sicuramente esistono anche le applicazioni intrinsecamente mobili e senza filo, che includono l'ufficio portatile e persone che camminano in un negozio con un PDA facendone l'inventario. In molti aeroporti, gli impiegati che ricevono le macchine a nolo, lavorano fuori nel parcheggio con calcolatori portatili senza filo. Essi digitano la targa della macchina che viene restituita, e il loro portatile, che ha una stampante all'interno, chiama il calcolatore principale, prende le informazioni di noleggio, e stampa la ricevuta al momento. La computazione mobile viene discussa approfonditamente in Forman, Zahorjan (1994).

Le reti senza filo si presentano in molte forme. Alcune università stanno già installando antenne per permettere agli studenti di sedersi sotto un albero e consultare il catalogo della biblioteca. In questo caso i calcolatori comunicano direttamente con la LAN senza filo in formato digitale. Un'altra possibilità è di utilizzare un telefono cellulare (portatile) con un tradizionale modem analogico. Il servizio digitale cellulare, chiamato CDPD (Cellular Digital Packed Data) sta diventando disponibile in molte città. Verrà studiato nel capitolo 4.

In ultimo, è possibile avere varie combinazioni di reti con e senza filo. Per esempio, in figura 1-8 (a), si mostra un aeroplano con un certo numero di persone che usano modem collegati a telefoni posti presso ogni postazione del velivolo per chiamare l'ufficio. Ogni chiamata è indipendente dalle altre. Una opzione molto più efficiente, invece, è la LAN in volo di figura 1-8 (b). In questo caso ogni posto sull'aereo viene equipaggiato con un connettore Ethernet a cui i passeggeri possono collegare i propri calcolatori. Un unico router sul velivolo mantiene un collegamento radio su qualche router sulla terra, cambiando router durante il volo. Questa configurazione è quella di una LAN convenzionale, eccetto che le sue connessioni verso l'esterno finiscono per essere collegamenti radio invece di linee hardware.

Mentre molti credono che i calcolatori portatili senza filo sono l'affare del futuro, almeno una voce contraria si è fatta sentire. Bob Metcalfe, l'inventore di Ethernet, ha scritto: "I calcolatori mobili senza filo sono come i bagni mobili senza tubature. Essi saranno utili sui veicoli, nei cantieri, ai concerti rock. Il mio consiglio è quello di collegare con un filo la tua casa e di restare in casa" (Metcalfe, 1995). Saranno molti a seguire il consiglio di Metcalfe? Solo il tempo potrà dirlo.

1.2 Hardware delle reti

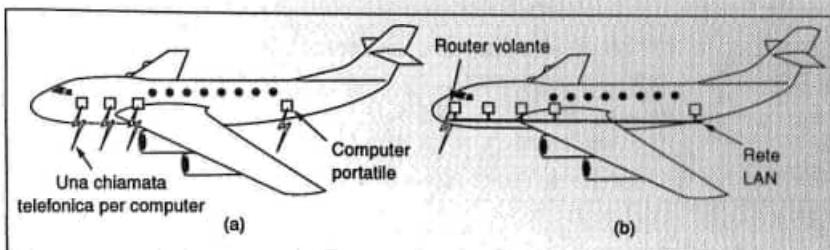


Fig. 1-8 (a) Calcolatori portatili individuali. (b) Una LAN volante.

1.2.5 Internetwork

Esistono molte reti, spesso con hardware e software differenti. Coloro che si collegano a una rete spesso desiderano comunicare con persone collegate ad altre reti. Questo desiderio richiede di collegare insieme reti diverse e spesso incompatibili, qualche volta usando macchine chiamate **gateway** per realizzare la connessione e provvedere alle necessarie traduzioni, sia in termini di hardware che di software. Una collezione di reti collegate viene chiamata una **internet** (*internetwork*).

Una forma comune di internet è una collezione di LAN collegate da una WAN. Infatti, se si rimpiazzasse l'etichetta "sottoretete" nella figura 1-5 con "WAN", null'altro sarebbe da cambiare nella figura. L'unica vera distinzione fra una sottoretete e una WAN in questo caso è dove sono gli host. Se il sistema nella linea chiusa contiene solamente router, essa è una sottoretete. Se contiene sia router che host, essa è una WAN.

Per evitare confusione, bisogna notare che la parola "internet" sarà sempre usata in questo libro con un senso generico. Al contrario, **Internet** (con la I maiuscola) indica un'internet mondiale specifica largamente utilizzata per collegare università, uffici di governo, aziende e ultimamente anche privati. Ci sarà da dire molto altro a proposito di internet e Internet nel seguito del libro.

Sottoreti, reti, e internetwork vengono spesso confuse. Le sottoreti hanno senso nel contesto di reti geografiche, dove ci si riferisce a collezioni di router e linee di comunicazione possedute da operatori di rete, per esempio aziende come America Online e CompuServe. Per analogia, il sistema telefonico comprende cabine per la trasmissione dei segnali telefonici collegate fra loro con linee ad alta velocità e collegate alle case e agli uffici da linee a bassa velocità. Queste linee e attrezzature, possedute e gestite dalle società telefoniche, formano la sottoretete del sistema telefonico. I telefoni (gli host in questa analogia) non sono parte della sottoretete. La combinazione di una sottoretete e dei suoi host forma una rete. Nel caso di una LAN, il cavo e gli host costituiscono la rete. Non esiste una sottoretete.

Si forma una internet quando reti distinte vengono collegate insieme. Nella visione di questo testo, la connessione fra una LAN e una WAN o la connessione di due LAN formano una internet, ma nell'industria non c'è accordo a proposito della terminologia in questo campo.

1.3 Software delle reti

La prima rete di calcolatori venne progettata pensando l'hardware come obiettivo principale, mentre il software veniva considerato un problema secondario. Questa strategia non vale più. Il software delle reti è attualmente altamente strutturato. Nel seguito del paragrafo si esamina in dettaglio la tecnica di strutturazione del software. Il metodo descritto forma la chiave di volta dell'intero libro e apparirà ripetutamente nel seguito.

1.3.1 Gerarchie di protocolli

Per ridurre la complessità di progettazione, la maggior parte delle reti è organizzata come una serie di strati o **livelli**, ognuno costruito su quello inferiore. Il numero di livelli, il nome di ciascun livello, il suo contenuto e le funzionalità differiscono da una rete all'altra. Tuttavia, in tutte le reti, lo scopo di ogni livello è di offrire certi servizi al livello superiore, schermendo quel livello dai dettagli di come i servizi offerti sono realizzati.

Il livello n su una macchina permette una conversazione con il livello n di un'altra macchina. Le regole e le convenzioni usate in queste conversazioni sono generalmente conosciute come il **protocollo** del livello n . Fondamentalmente, un protocollo è un accordo fra i partecipanti di una comunicazione su come la comunicazione deve procedere. Per analogia, quando una donna viene presentata a un uomo, può scegliere se allungare la mano. Lui, invece, può decidere se stringerla o baciarla, rispettivamente se è un'avvocatessa a un incontro d'affari o una principessa a un ballo regale. La violazione del protocollo può rendere la comunicazione più difficile se non impossibile.

Una rete a cinque livelli è mostrata in figura 1-9. Le entità che comprendono il corrispondente livello su macchine differenti sono chiamate **pari** (peer). In altre parole, sono i pari che comunicano utilizzando il protocollo.

In realtà, nessun dato viene trasferito dal livello n su una macchina al livello n su un'altra. Invece, ogni livello passa al livello immediatamente sotto di sé dati e informazioni di controllo, fino a che viene raggiunto il livello più basso. Sotto il livello 1 c'è il **mezzo fisico** attraverso il quale avviene la comunicazione. In figura 1-9, la comunicazione virtuale è mostrata da linee tratteggiate, mentre la comunicazione fisica da linee continue.

Fra ogni coppia di livelli adiacenti c'è una **interfaccia**. L'interfaccia definisce quali operazioni primitive e servizi offre il livello sottostante a quello superiore. Quando i progettisti di reti decidono quanti livelli includere in una rete e che cosa fa ciascuno, una delle considerazioni più importanti è definire interfacce chiare fra i diversi livelli. Fare questo, però, richiede che ogni livello esegua una collezione specifica di funzioni ben comprensibili. Oltre che a minimizzare la quantità di informazioni che deve essere passata fra i vari livelli, interfacce chiare semplificano la sostituzione dell'implementazione di un livello con un'implementazione completamente diversa (per esempio, tutte le linee telefoniche sono sostituite con canali satellitari), in quanto tutto ciò che è richiesto alla nuova implementazione è che offra esattamente lo stesso insieme di servizi ai suoi livelli superiori, come faceva la vecchia implementazione.

Un insieme di livelli e protocolli è chiamato **architettura di rete**. La specifica di un'architettura deve contenere abbastanza informazione per permettere a un implementatore di scrivere il programma o costruire l'hardware per ciascun livello così che esso risponda correttamente al protocollo appropriato. Né i dettagli dell'implementazione e nemmeno

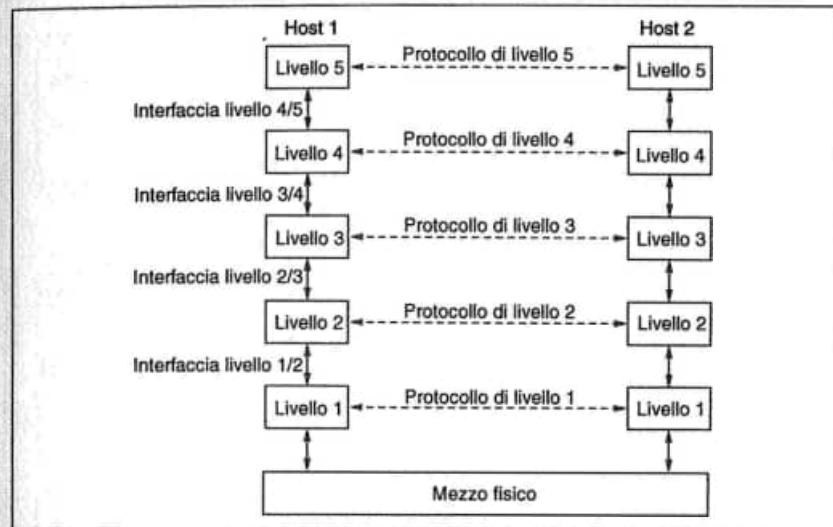


Fig. 1-9 Livelli, protocolli e interfacce.

la specifica delle interfacce sono parte dell'architettura perché sono nascosti all'interno delle macchine e non sono visibili all'esterno. Non è inoltre necessario che le interfacce su tutte le macchine in una rete siano le stesse, purché ogni macchina usi correttamente i protocolli. Una lista di protocolli usati da un certo sistema, un protocollo per livello, è chiamata **pila di protocolli**. Le architetture delle reti e le pile di protocolli sono il principale argomento di questo libro.

Un'analogia può aiutare nello spiegare l'idea di comunicazione multilivello. Si immaginino due filosofi (processi pari del livello 3), uno dei quali parla indostano (la lingua Indù) e inglese e l'altro che parla cinese e francese. Sino a che non hanno un linguaggio comune, essi coinvolgono un interprete (processi pari di livello 2), che a sua volta contatta una segretaria (processi pari di livello 1). Il filosofo 1 desidera comunicare la sua viva simpatia per l'*oryctolagus cuniculus* al suo pari. Per far questo, egli passa un messaggio (in inglese) attraverso l'interfaccia 2/3, al suo interprete, dicendo "I like rabbits", come illustrato in figura 1-10. L'interprete si è accordato su un linguaggio neutro, l'olandese, e così il messaggio viene tradotto in "Ik hou von konijnen". La scelta del linguaggio è il protocollo di livello 2 e viene fatta dai processi pari di livello 2.

L'interprete quindi consegna il messaggio a una segretaria per la trasmissione, utilizzando, per esempio, un fax (il protocollo di livello 1). Quando il messaggio arriva, esso viene tradotto in francese a passo lungo l'interfaccia 2/3 al filosofo 2. Si noti che ogni protocollo è completamente indipendente dall'altro così come le interfacce. Gli interpreti possono passare dall'olandese al finlandese a piacimento, a condizione che entrambi siano d'accordo, e questo non cambia l'interfaccia né con il livello 1 né con il livello 3. Allo stesso modo le segretarie possono sostituire il fax con la posta elettronica o il telefono

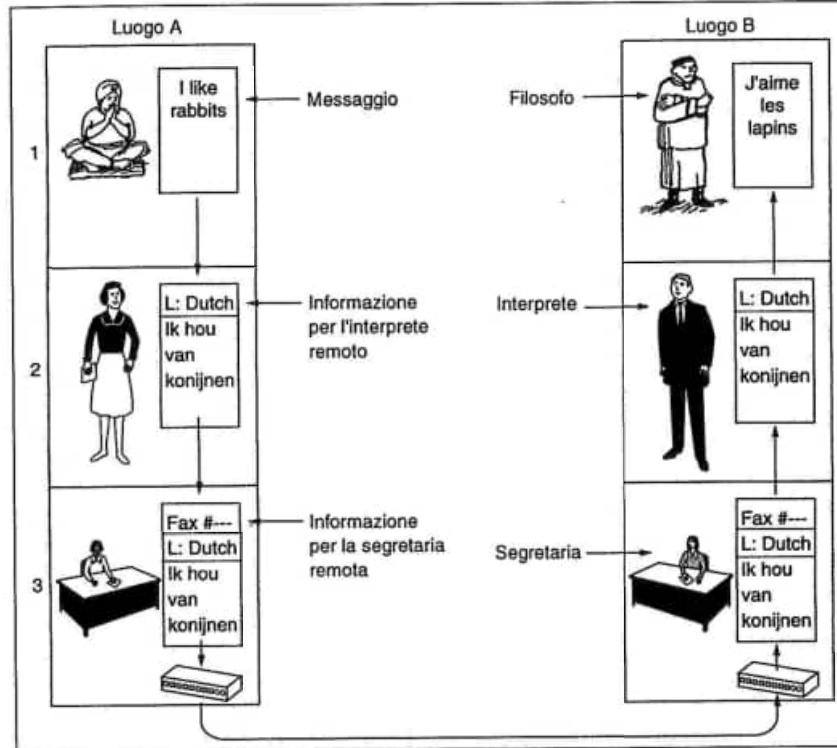


Fig. 1-10 L'architettura filosofo-interprete-segretaria.

senza disturbare (e nemmeno informare) gli altri livelli. Ogni processo può aggiungere delle informazioni per il proprio pari, che non vengono passate al livello superiore.

Ora si consideri un esempio più tecnico: come permettere la comunicazione al livello superiore della rete a cinque livelli di figura 1-11. Un messaggio M , viene prodotto da un processo applicativo al livello 5 e passato al livello 4 per essere trasmesso. Il livello 4 inserisce una **intestazione** davanti al messaggio per identificare il messaggio e passa il risultato al livello 3. L'intestazione include informazioni di controllo, tipo numeri in sequenza, per permettere al livello 4 sulla macchina destinataria di ricostruire i messaggi nel giusto ordine se i livelli più bassi non preservano l'ordine. In alcuni livelli, le intestazioni contengono anche dimensione, ora, e altri campi di controllo.

In molte reti, non c'è limite alla dimensione di un messaggio trasmesso con il protocollo a livello 4, ma c'è quasi sempre un limite imposto dal protocollo di livello 3. Di conseguenza, il livello 3 deve spezzare i messaggi in arrivo in pacchetti, unità più piccole, aggiungendo una intestazione di livello 3 a ogni pacchetto. In questo esempio, M è diviso in due parti, M_1 e M_2 .

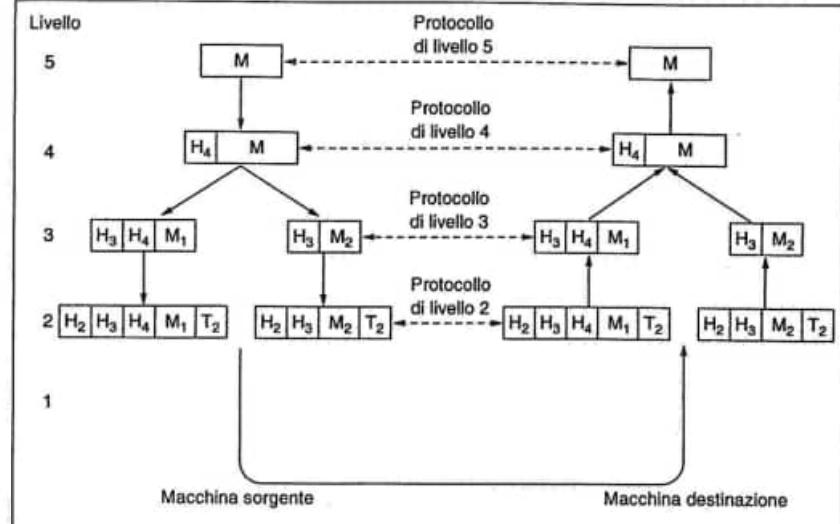


Fig 1-11 Esempio di flusso di informazioni per supportare la comunicazione virtuale al livello 5.

Il livello 3 decide quale delle linee in uscita usare e passa i pacchetti al livello 2. Il livello 2 inserisce non solo un'intestazione in ogni pacchetto, ma anche un cammino, e passa le unità al livello 1 per la trasmissione fisica. Alla macchina destinataria il messaggio giunge salendo, livello per livello, con le intestazioni che vengono eliminate di volta in volta. Nessuna delle intestazioni per i livelli inferiori a n raggiungono il livello n .

La cosa importante da capire della figura 1-11 è la relazione fra comunicazione virtuale e reale e la differenza fra protocolli e interfacce. I processi pari a livello 4, per esempio, pensano che la loro comunicazione sia "orizzontale", usando il protocollo di livello 4. È come se ognuno avesse una procedura chiamata *InviaDaAltraParte* e *RiceviDaAltraParte* anche se queste procedure comunicano in realtà con i livelli sottostanti attraverso l'interfaccia 3/4, e non direttamente con il proprio pari.

L'astrazione dei processi pari è cruciale in tutti i progetti di reti. Usando questa astrazione, il compito difficilmente gestibile di progettare l'intera rete può essere spezzato in diversi problemi di progettazione più trattabili, in particolare la progettazione di un singolo livello.

Anche se il paragrafo 1.3 si intitola "Software delle Reti", vale la pena sottolineare che i livelli inferiori di una gerarchia di protocolli sono frequentemente implementati in hardware o firmware. Ciò nonostante, sono coinvolti algoritmi di protocolli complessi, anche se essi sono realizzati (completamente o in parte) in hardware.

1.3.2 Aspetti progettuali per i livelli

Alcuni degli aspetti chiave che ci sono nelle reti di calcolatori sono presenti in livelli diversi. Nel seguito, se ne menzionano alcuni fra i più importanti.

Ogni livello richiede un meccanismo per identificare il mittente e il destinatario. Visto che normalmente le reti hanno molti calcolatori, alcuni dei quali hanno molti processi, occorre un meccanismo per specificare il processo con il quale si desidera colloquiare. In conseguenza del fatto che si possono avere destinazioni diverse, è necessaria qualche forma di indirizzamento per denotare una destinazione specifica.

Un altro insieme di decisioni progettuali riguarda le regole per il trasferimento dei dati. In alcuni sistemi, i dati viaggiano in una direzione (**comunicazione simplex**). In altri casi essi possono viaggiare in entrambe le direzioni, ma non simultaneamente (**comunicazione half-duplex**). In altri ancora i dati viaggiano in entrambe le direzioni nello stesso tempo (**comunicazione full-duplex**). Il protocollo deve anche determinare a quanti canali logici corrispondono le connessioni, e quali sono le loro proprietà. Molte reti hanno almeno due canali logici per connessione, uno per dati normali e uno per dati urgenti.

Il controllo degli errori è un aspetto importante perché i circuiti di comunicazione fisica non sono perfetti. Esistono molti codici di rilevamento e correzione degli errori, ma entrambi i capi di una connessione devono accordarsi su quale di questi deve essere utilizzato. In più, il ricevente deve avere un meccanismo per dire al mittente quali messaggi sono stati correttamente ricevuti e quali invece no.

Non tutti i canali di comunicazione preservano l'ordine dei messaggi inviati lungo di essi. Per gestire una possibile perdita della sequenzializzazione, il protocollo deve fornire al ricevente la possibilità di ricostruire correttamente i pezzi ricevuti. Una soluzione ovvia è di numerare i pezzi, ma questa soluzione lascia ancora aperto il problema di che cosa fare con pacchetti che giungono fuori ordine.

Un aspetto che si ha a ogni livello è come evitare che un mittente veloce inondi di dati un ricevitore lento. Diverse soluzioni sono state proposte e verranno discusse nel seguito. Alcune di queste coinvolgono un meccanismo di risposta dal ricevente al mittente, diretto o indiretto, a proposito della sua situazione. Altri limitano il mittente a una data velocità di trasmissione.

Un altro problema che deve essere risolto a diversi livelli è l'incapacità di tutti i processi di accettare messaggi arbitrariamente lunghi. Questa proprietà porta a meccanismi di decomposizione, trasmissione e ricomposizione dei messaggi. Un aspetto collegato è cosa fare quando i processi insistono a trasmettere dati in unità che sono così piccole che il loro invio separato risulta inefficiente. In questo caso la soluzione è di fondere insieme diversi messaggi intestati a una destinazione comune in un unico grande messaggio e di smembrare tale messaggio al momento della sua ricezione.

Quando è poco conveniente o costoso instaurare una connessione separata per ogni coppia di processi comunicanti, il livello sottostante potrebbe decidere di usare la medesima connessione per molte conversazioni indipendenti. Fino a quando questa condivisione di canale avviene in modo trasparente, può essere utilizzata da ogni livello. La condivisione è richiesta al livello fisico, per esempio, dove tutto il traffico per tutte le connessioni deve essere inviato lungo al più pochi circuiti fisici.

Quando ci sono più cammini fra sorgente e destinazione, deve esserne scelto uno. Qualche

volta questa decisione viene divisa fra due o più livelli. Per esempio, per spedire dati da Londra a Roma, una decisione ad alto livello sceglie se passare attraverso la Francia o la Germania a seconda delle singole leggi dei due stati, mentre una decisione a basso livello sceglie uno dei possibili circuiti in funzione del loro carico di traffico.

1.3.3 Interfacce e servizi

La funzione di ogni livello è di fornire dei servizi al livello superiore. In questo paragrafo studiamo più in dettaglio che cos'è un servizio, ma prima dobbiamo definire alcuni termini.

Gli elementi attivi in ogni livello sono spesso chiamati **entità**. Una entità può essere software (un processo), o hardware (un componente intelligente di gestione dell'input/output). Le entità allo stesso livello su macchine diverse sono chiamate **entità pari**. Le entità al livello n implementano servizi usati al livello $n+1$. In questo caso il livello n è detto **fornitore di servizi** e il livello $n+1$ è detto **utente di servizi**. Il livello n può usare i servizi del livello $n-1$ per fornire i propri. Ogni livello può fornire diversi tipi di servizi, per esempio, comunicazione veloce ma costosa e comunicazione lenta ed economica.

I servizi sono disponibili presso i **SAP – Service Access Point** (punti di accesso dei servizi). I SAP del livello n sono i luoghi in cui il livello $n+1$ può accedere ai servizi che vengono offerti. Ogni SAP ha un indirizzo che lo identifica univocamente. Per chiarire questo punto, i SAP nel sistema telefonico sono le prese dove possono essere inseriti i telefoni, e gli indirizzi SAP sono i numeri telefonici di queste prese. Per chiamare qualcuno, bisogna conoscere l'indirizzo di SAP. In modo analogo, nel sistema postale gli indirizzi di SAP sono i numeri civici e i numeri delle cassette postali. Per spedire una lettera, bisogna conoscere l'indirizzo di SAP.

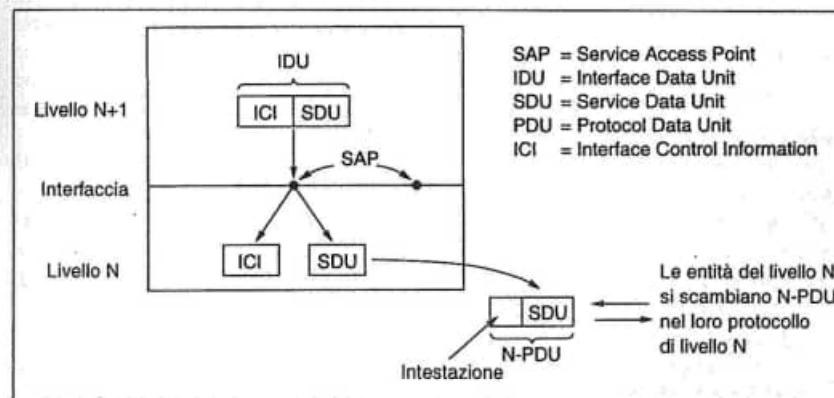


Fig. 1-12 Relazioni fra livelli in una interfaccia.

Per permettere lo scambio di informazioni fra due livelli, bisogna definire un accordo sull'insieme di regole riguardanti le interfacce. A una interfaccia tipica, l'entità al livello

$n+1$ passa un **IDU – Interface Data Unit** (unità dati di interfaccia) all'entità di livello n attraverso il SAP come viene rappresentato in figura 1-12. L'IDU è costituito da un **SDU – Service Data Unit** (unità dati di servizio) e alcune informazioni di controllo. L'SDU è l'informazione passata attraverso la rete all'entità pari e in seguito al livello $n+1$. L'informazione di controllo è richiesta per aiutare il livello inferiore a realizzare il proprio lavoro (per esempio il numero di byte nell'SDU), ma non è parte degli stessi dati.

Per trasferire l'SDU, l'entità di livello n potrebbe dover frammentarla in diversi pezzi, a ognuno dei quali viene data una intestazione. Questi pacchetti separati vengono poi spediti come **PDU – Protocol Data Unit** (unità dati di protocollo). Le intestazioni dei PDU sono usate dalle entità pari per portare avanti il proprio protocollo. Esse identificano quali PDU contengono dati e quali contengono informazioni di controllo, forniscono numeri di sequenze e contatori, e così via.

1.3.4 Servizi orientati alla connessione e servizi privi di connessione

I livelli possono offrire due diversi tipi di servizi ai livelli superiori: i servizi orientati alla connessione e quelli privi di connessione. In questo paragrafo presentiamo questi tipi di servizio ed esaminiamo le loro differenze.

I **servizi orientati alla connessione** sono modellati sul sistema telefonico. Per parlare con qualcuno si alza il telefono, si compone il numero, si parla e infine si ripone il telefono. In modo similare, per usare un servizio di rete orientato alla connessione, l'utente del servizio prima stabilisce una connessione, la utilizza e infine la rilascia. L'aspetto essenziale di una connessione è che agisce come una tubatura: il mittente introduce oggetti (bit) a partire da una estremità, e il ricevente li riprende fuori nel medesimo ordine all'altra estremità.

Al contrario, i **servizi privi di connessione** sono modellati sul sistema postale. Ogni messaggio (lettera) porta con sé l'indirizzo completo di destinazione, e ognuno è condotto lungo il sistema indipendentemente da ogni altro messaggio. Normalmente, quando due messaggi sono inviati alla medesima destinazione, il primo inviato sarà anche il primo ad arrivare. Tuttavia, è possibile che il primo inviato possa essere ritardato così che il secondo arrivi prima. Con un servizio orientato alla connessione questo non è possibile.

Ogni servizio può essere caratterizzato da una **qualità del servizio** (Quality of Service – QoS). Alcuni servizi sono affidabili nel senso che non perdono mai dei dati. Normalmente, un servizio affidabile è realizzato attraverso un messaggio di avvenuta ricezione inviato dai destinatari ogni qualvolta ricevono un nuovo messaggio, in modo tale che il mittente si assicuri della ricezione dei propri messaggi. Il processo di invio del messaggio di avvenuta ricezione introduce lavoro ulteriore e ritardi, che sono spesso utili ma che sono a volte indesiderati.

Una situazione tipica in cui è appropriato un servizio affidabile orientato alla connessione è il trasferimento di un archivio. Il possessore dell'archivio vuole essere certo che tutti i bit arrivino correttamente e nello stesso ordine in cui erano stati inviati. Pochi utenti del trasferimento di archivi preferirebbero un servizio che occasionalmente scambia o perde alcuni bit, anche se risulta molto più veloce.

I servizi affidabili orientati alla connessione hanno due varianti minori: sequenze di messaggi e successione di byte. Nel primo caso, i limiti di dimensione dei messaggi sono

1.3 Software delle reti

preservati. Quando sono inviati due messaggi da 1 KB, essi arrivano come due distinti messaggi da 1 KB, ma come un unico messaggio da 2 KB. Nel secondo caso, la connessione è solamente una successione di byte, senza limiti per la lunghezza dei messaggi. Quando 2 KB arrivano al ricevente, non c'è modo per dire se essi erano stati inviati come un unico messaggio da 2 KB, due messaggi da 1 KB, o 2048 messaggi da 1 byte. Se le pagine di un libro vengono inviate lungo una rete a un tipografo attraverso messaggi separati, risulta importante preservare i limiti di lunghezza dei messaggi. D'altra parte, tutto quel che serve a un terminale che si collega a un sistema condiviso è una successione di byte dal terminale al calcolatore.

Come abbiamo visto prima, per alcune applicazioni, i ritardi introdotti dal messaggio di avvenuta ricezione sono inaccettabili. Una di queste applicazioni è il traffico di voci digitalizzate. Gli utenti telefonici preferiscono sentire un po' di rumore sulla linea o una parola poco chiara di tanto in tanto piuttosto che subire il ritardo necessario per trasmettere il messaggio di avvenuto ricevimento. In modo analogo, quando si trasmette un filmato, se alcuni pixel sono sbagliati non è un problema, ma vedere il filmato a scatti per poter correggere gli errori può essere molto irritante.

Non tutte le applicazioni richiedono la connessione. Per esempio, con l'avvento della posta elettronica, la posta trasmessa in pezzi separati potrà essere un'applicazione utile. Colui che invia posta in parti separate probabilmente non vuole avere il problema di inizializzare e più tardi annullare una connessione solamente per inviare un pezzo della propria posta. Non è richiesta nemmeno un'affidabilità del 100%. Ciò che è richiesto è un mezzo per inviare un singolo messaggio che ha alta probabilità di essere ricevuto, seppure senza certezza. Servizi privi di connessione non affidabili (nel senso che il messaggio di avvenuto ricevimento non è utilizzato) vengono spesso chiamati **servizi datagramma**, in analogia con il servizio di telegrammi, che non prevede un messaggio di ricevimento da parte del mittente.

In altre situazioni, la convenienza di non dover stabilire una connessione per spedire un messaggio piccolo a piacere è desiderabile, ma l'affidabilità è essenziale. Per queste applicazioni può essere fornito il servizio datagramma **con messaggio di avvenuto ricevimento** o messaggio di riscontro (ack). È come inviare una raccomandata e richiedere la ricevuta di ritorno. Quando la ricevuta ritorna indietro, il mittente è assolutamente sicuro che la lettera inviata ha raggiunto il destinatario e non è andata persa lungo il viaggio.

Un ulteriore servizio è il **servizio richiesta-risposta**. In questo servizio il mittente trasmette un singolo datagramma contenente una richiesta: il messaggio di ritorno contiene la risposta. Per esempio, ricade in questa categoria una richiesta alla libreria locale in cui si chiede dove è citato Uighur. La richiesta-risposta è comunemente usata per implementare comunicazione nel modello client-server: il client invia una richiesta e il server gli risponde. La figura 1-13 riassume i tipi di servizi presentati.

1.3.5 Primitive dei servizi

Un servizio è formalmente specificato da un insieme di **primitive** (operazioni) disponibili a un utente o ad altre entità per accedere ai servizi stessi. Queste primitive dicono al servizio di eseguire delle azioni o di fare rapporto su una azione realizzata da una entità

	Servizio	Esempio
Orientamento alla connessione	Serie di messaggi affidabili	Sequenza di pagine
	Serie di byte affidabili	Collegamento remoto
	Connessione non affidabile	Voce digitalizzata
	Datagramma non affidabile	Pezzi di posta elettronica
	Datagramma con riscontro	Posta con ricevuta
	Richiesta-risposta	Interrogazione di database

Fig. 1-13 Sei diversi tipi di servizi.

pari. Un modo per classificare le primitive dei servizi è di dividerle in quattro classi come mostra la figura 1-14.

Primitiva	Significato
Request	Entità che richiede che il servizio faccia qualcosa
Indication	Un'entità viene informata di un evento
Response	Un'entità vuole rispondere a un evento
Confirm	È arrivata la risposta a una richiesta precedente

Fig. 1-14 Quattro classi di primitive per i servizi.

Per illustrare l'uso delle primitive, si consideri il modo in cui una connessione viene stabilita e rilasciata. L'entità che effettua l'inizializzazione effettua una CONNECT.request che permette a un pacchetto di essere spedito. Il ricevente riceve una CONNECT.indication che annuncia che una entità da qualche parte desidera attivare una connessione. L'entità che riceve la CONNECT.indication utilizza la primitiva CONNECT.response per dire se vuole accettare o meno la connessione proposta. In ogni caso, l'entità che aveva inviato l'iniziale CONNECT.request verifica che cosa è avvenuto attraverso una primitiva CONNECT.confirm.

Le primitive possono avere parametri, e la maggior parte di esse ne ha. I parametri di una CONNECT.request potrebbero specificare la macchina a cui collegarsi, il tipo di servizio desiderato e la massima dimensione di messaggi che si può utilizzare lungo la connessione. I parametri di una CONNECT.indication potrebbero contenere l'identità del chiamante, il tipo di servizio desiderato, e la proposta di massima dimensione di messaggio. Se l'entità chiamata non concorda con la proposta dimensione massima, può fare una controproposta attraverso la propria primitiva response, che verrà resa nota al chiamante attraverso la primitiva confirm. I dettagli di questa **negoziazione** fanno parte del protocollo. Per esempio, nel caso di due proposte conflittuali sulla massima dimensione dei

1.3 Software delle reti

messaggi, il protocollo potrebbe specificare che venga scelta sempre la dimensione minore.

Per quanto riguarda la terminologia, faremo attenzione a evitare i termini "aprire una connessione" e "chiudere una connessione" perché per gli ingegneri elettronici un "circuito aperto" è un circuito con una interruzione. L'elettricità può solamente circolare lungo "circuiti chiusi". Gli informatici potrebbero non essere d'accordo ad avere informazioni che viaggiano lungo circuiti chiusi. Per mantenere la pace fra i due campi, si useranno i termini "stabilire una connessione" e "rilasciare una connessione".

I servizi possono essere sia **confermati** che **non confermati**. In un servizio confermato, ci sono le primitive *request*, *indication*, *response* e *confirm*. In un servizio non confermato, ci sono solo le primitive *request* e *indication*. Il servizio CONNECT è sempre confermato in quanto il pari remoto deve accordarsi per stabilire la connessione. Il trasferimento dei dati, d'altra parte, può essere sia confermato che non confermato, a seconda che il mittente desideri un messaggio di avvenuta ricezione oppure no. Entrambi i tipi di servizi sono utilizzati nelle reti.

Per rendere il concetto di servizio più concreto, si considera come esempio un semplice servizio orientato alla connessione con otto primitive di servizio:

1. CONNECT.request – Richiede di stabilire una connessione.
2. CONNECT.indication – Trasmette un segnale all'entità chiamata.
3. CONNECT.response – Usata dall'entità chiamata per accettare o rifiutare le chiamate.
4. CONNECT.confirm – Comunica all'entità chiamante se la chiamata è stata accettata.
5. DATA.request – Richiede la spedizione dei dati.
6. DATA.indication – Segnala l'arrivo dei dati.
7. DISCONNECT.request – Richiede il rilascio di una connessione.
8. DISCONNECT.indication – Informa il proprio pari a proposito delle richieste.

In questo esempio, CONNECT è un servizio confermato (viene richiesta esplicitamente una risposta), mentre DISCONNECT è non confermato (senza risposta).

Per vedere come queste primitive vengono utilizzate è utile fare una analogia con il sistema telefonico. Si considerino i passi richiesti per chiamare Zia Millie al telefono e invitarla per un tè.

1. CONNECT.request – Componi il numero telefonico di Zia Millie.
2. CONNECT.indication – Il suo telefono squilla.
3. CONNECT.response – Lei alza la cornetta.
4. CONNECT.confirm – Senti il termine dello squillo.
5. DATA.request – La inviti per un tè.
6. DATA.indication – Lei ascolta il tuo invito.
7. DATA.request – Lei dice che sarebbe lieta di venire.
8. DATA.indication – Tu ascolti la sua accettazione.
9. DISCONNECT.request – Tu chiudi la comunicazione.
10. DISCONNECT.indication – Lei sente che hai interrotto la comunicazione e la interrompe a sua volta.

La figura 1-15 mostra questa stessa sequenza di passi come una serie di primitive di servizio, inclusa la conferma di disconnessione finale. Ogni passo coinvolge una interazione fra i due livelli su un calcolatore. Ogni primitiva *request* o *response* causa poco dopo una primitiva *indication* o *confirm* dall'altra parte. In questo esempio, gli utenti del servizio (tu e Zia Millie) sono al livello $n+1$ mentre il fornitore del servizio (il sistema telefonico) è al livello n .

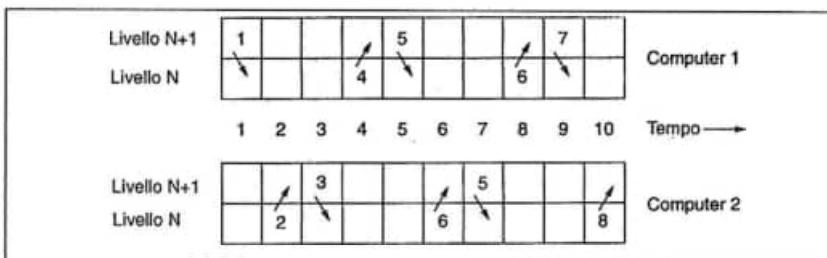


Fig. 1-15 Un calcolatore invita Zia Millie per un tè. I numeri vicino a ogni freccia fanno riferimento alle otto primitive di servizio discusse in questo paragrafo.

1.3.6 Relazioni fra servizi e protocolli

Servizi e protocolli sono nozioni distinte, tuttavia sono frequentemente confuse. Questa distinzione è così importante, tuttavia, che la sottolineiamo ulteriormente. Un *servizio* è un insieme di primitive (operazioni) che un livello fornisce al livello superiore. Il servizio definisce quali operazioni il livello è pronto a eseguire per conto dei propri utenti, ma non dice nulla su come tali operazioni vengono implementate. Un servizio fa parte di una interfaccia fra due livelli, il livello inferiore che è il fornitore del servizio e il livello superiore il fruitore.

Un *protocollo*, invece, è un insieme di regole che governano il formato e il significato dei blocchi di informazione, dei pacchetti o dei messaggi che vengono scambiati fra le entità pari di un certo livello. Le entità utilizzano i protocolli per implementare le definizioni dei propri servizi. Esse sono libere di modificare i propri protocolli in futuro, ammesso che non cambino i servizi visibili ai propri utenti. In questo modo, i servizi e i protocolli sono completamente separati.

È utile un'analogia con i linguaggi di programmazione. Un servizio è come un tipo di dato astratto o un oggetto in linguaggi orientati agli oggetti. Esso definisce operazioni che possono essere eseguite su un oggetto ma non specificano come queste operazioni devono essere implementate. Un protocollo fa riferimento all'*implementazione* del servizio e come tale non è visibile all'utente del servizio.

Molti vecchi protocolli non facevano distinzione fra servizio e protocollo. Concretamente, un livello tipico poteva avere una primitiva di servizio SEND PACKET e l'utente doveva fornire un puntatore a un pacchetto completamente assemblato. Questo meccanismo implicava che tutti i cambiamenti al protocollo fossero immediatamente visibili agli utenti.

1.4 Modelli di riferimento

La maggior parte dei progettisti di reti pensano oggi che questo modo di progettare sia un errore grossolano.

1.4 Modelli di riferimento

Ora che si sono presentate in modo astratto le reti stratificate, è tempo di parlare di qualche esempio. Nei prossimi due paragrafi presentiamo due importanti architetture di reti, il modello di riferimento OSI e il modello di riferimento TCP/IP.

1.4.1 Il modello di riferimento OSI

Il modello OSI è presentato in figura 1-16 (meno il mezzo fisico). Questo modello è basato su una proposta sviluppata dalla Organizzazione per gli Standard Internazionali (ISO – International Standard Organization) come un primo passo verso la standardizzazione internazionale dei protocolli utilizzati nei vari livelli (Day, Zimmermann, 1983). Il modello è chiamato **modello di riferimento ISO-OSI (Open System Interconnection – interconnessione di sistemi aperti)** perché si interessa di collegare sistemi aperti – cioè, sistemi che sono aperti per comunicazioni con altri sistemi. Tale modello verrà chiamato modello OSI per brevità.

Il modello OSI ha sette livelli. I principi che sono stati seguiti per arrivare a sette livelli sono i seguenti:

1. Un livello dovrebbe essere creato ogni volta che viene richiesto un diverso livello di astrazione.
2. Ogni livello dovrebbe realizzare una ben determinata funzione.
3. La funzione di ciascun livello dovrebbe essere scelta con un occhio rivolto alla definizione di protocolli internazionali standardizzati.
4. I limiti dei livelli dovrebbero essere scelti per minimizzare il flusso delle informazioni attraverso le interfacce.
5. Il numero di livelli deve essere abbastanza ampio per permettere a funzioni distinte di non essere inserite forzatamente nel medesimo livello senza che sia necessario, e abbastanza piccolo per permettere che le architetture non diventino pesanti e poco maneggevoli.

Nel seguito si presenta separatamente ciascun livello del modello, iniziando dal livello inferiore. Si noti che il modello OSI da solo non è un'architettura di rete perché non specifica con precisione i servizi e i protocolli che devono essere usati in ogni livello. Esso dice solo che cosa dovrebbe fare ciascun livello. Tuttavia, ISO ha prodotto degli standard per tutti i livelli, anche se questi non fanno parte del modello di riferimento. Ognuno di questi è stato pubblicato come uno standard internazionale separato.

Il livello fisico (physical)

Il livello fisico fa riferimento alla trasmissione dei bit lungo un canale di comunicazione.

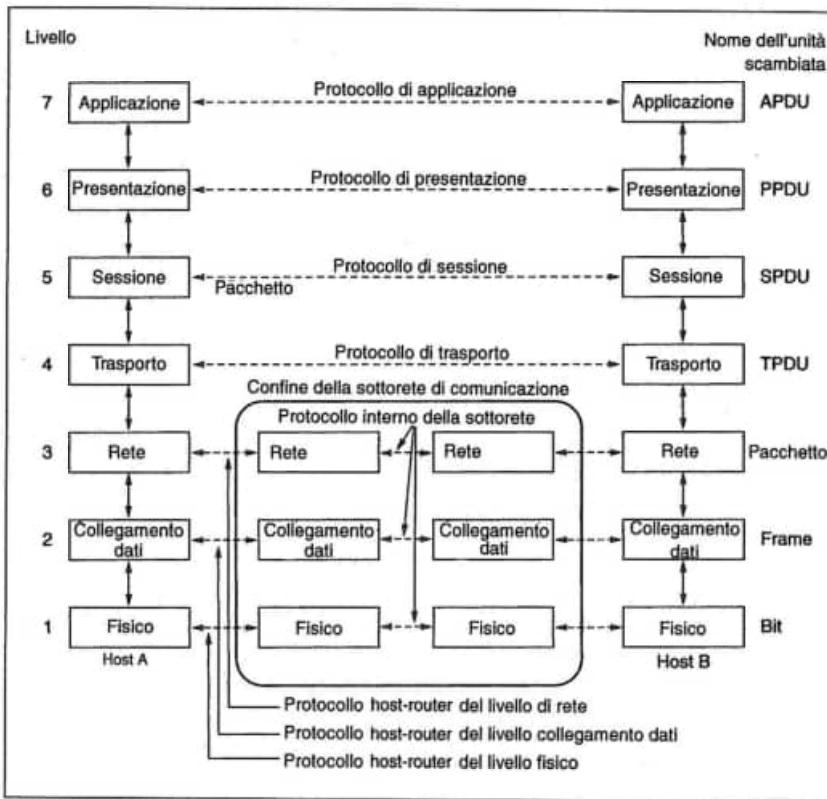


Fig. 1-16 Il modello di riferimento OSI.

Gli aspetti di progettazione hanno a che fare con l'assicurarsi che quando una estremità invia un bit 1 esso venga ricevuto dall'altra come un 1, e non come uno 0. Problemi tipici a questo livello sono quanti volt devono essere utilizzati per rappresentare un 1 e quanti per uno 0, quanti microsecondi richiede un bit, se la trasmissione può procedere simultaneamente in entrambe le direzioni, come si stabilisce la connessione iniziale e come la si termina quando a entrambe le estremità la comunicazione è finita, quanti connettori devono essere presenti sul cavo di rete e come ogni connessione deve essere utilizzata. Gli aspetti di progettazione hanno spesso a che fare con interfacce meccaniche, elettriche e procedurali, e il mezzo di trasmissione fisica, che si trova al di sotto del livello fisico.

Il livello data-link (collegamento dei dati)

Lo scopo principale del livello è di trasformare una trasmissione grezza in una linea per il livello superiore che appaia libera da errori di trasmissione non segnalati. Tale livello

realizza questo scopo facendo decomporre al mittente i dati da spedire in pacchetti (data frame – normalmente composti da alcune centinaia o migliaia di byte), che vengono spediti in sequenza attendendo poi il messaggio di avvenuta ricezione (ack) del pacchetto inviato dal ricevente. Visto che il livello fisico accetta e trasmette sequenze di bit senza far riferimento al loro significato o alla loro struttura, è compito del livello data link creare e riconoscere i limiti dei pacchetti. Questo può essere realizzato attaccando dei bit speciali all'inizio e alla fine del pacchetto. Se questi bit si presentano in modo accidentale nei dati, bisogna fare particolare attenzione per evitare che vengano interpretati erroneamente come delimitatori di pacchetto.

Un'interferenza sulla linea potrebbe completamente distruggere un pacchetto. In questo caso, il software del livello data link sulla macchina del mittente può ritrasmettere il pacchetto stesso. Tuttavia, trasmissioni multiple del medesimo pacchetto introducono la possibilità di pacchetti duplicati. Un pacchetto duplicato potrebbe essere spedito se il messaggio di avvenuto ricevimento è andato perso. È compito di questo livello risolvere i problemi causati dal danneggiamento, perdita o duplicazione di pacchetti. Il livello data link può offrire diverse classi di servizi al livello superiore, ognuno di diversa qualità e avente un diverso costo.

Un altro aspetto importante al livello data link (e anche in molti altri livelli) è come mantenere veloce una trasmissione evitando di sommergere un ricevitore di dati lento. Deve essere stabilito qualche meccanismo di regolazione del traffico per permettere al trasmittitore di conoscere quanto spazio ha a disposizione il ricevente per memorizzare i propri messaggi. Spesso, la regolazione di questo flusso e la gestione degli errori sono integrati.

Se la linea può essere utilizzata per trasmettere dati in entrambe le direzioni, questo introduce una nuova complicazione che il software del livello data link deve gestire. Il problema è che il messaggio di avvenuto ricevimento dei pacchetti che viaggiano da A a B compete per l'utilizzo della linea con i pacchetti che viaggiano da B ad A. È stata proposta una soluzione efficiente (piggybacking); verrà presentata in dettaglio nel seguito. Le reti broadcast hanno un ulteriore particolarità per quanto riguarda il livello data-link: occorre controllare l'accesso al canale condiviso. Uno speciale sottolivello per il livello data-link, il sottolivello di accesso al mezzo di comunicazione, ha a che fare con questo tipo di problema.

Il livello di rete (network)

Il livello di rete ha a che fare con il controllo delle operazioni di sottorete. Un aspetto chiave di progettazione è determinare come i pacchetti percorrono la rete dalla sorgente alla sua destinazione. I cammini possono essere basati su tabelle statiche che sono inserite all'interno della rete che vengono cambiate raramente. Essi possono essere determinati anche all'inizio di ogni conversazione, per esempio una sessione di terminale. Infine, possono essere determinati in modo dinamico per ciascun pacchetto, per prendere in considerazione il corrente carico di lavoro.

Se troppi pacchetti sono presenti nello stesso istante nella sottorete, daranno vita in ogni caso a qualche forma di congestione. Il controllo di tale congestione è responsabilità del livello rete.

Visto che gli operatori di sottorete possono avere bisogno di una numerazione dei pacchetti per realizzare i propri scopi, c'è spesso qualche funzione di enumerazione inserita all'interno del livello rete. Nel dettaglio, il software deve contare quanti pacchetti, o caratteri o bit vengono inviati da ogni utente, per produrre le informazioni di riepilogo. Quando un pacchetto attraversa un confine nazionale, con diverse velocità fra le due parti, il conteggio può diventare complicato.

Quando un pacchetto deve viaggiare da una rete a un'altra per raggiungere la propria destinazione, possono nascere molti problemi. L'indirizzamento utilizzato nella seconda rete può essere diverso da quello utilizzato nella prima. La seconda potrebbe non accettare il pacchetto perché troppo lungo. I protocolli potrebbero essere differenti, e così via. È compito del livello rete superare tutti questi tipi di problemi per permettere il collegamento di reti eterogenee.

Nelle reti broadcast, il problema del cammino da scegliere per far viaggiare i pacchetti (problema del routing) è semplice, così il livello rete è spesso sottile o assente.

Il livello di trasporto (transport)

La funzione di base del **livello di trasporto** è di accettare dati dal livello superiore, spezzarli in piccole unità se è necessario, passare queste al **livello rete**, e assicurarsi che tutti i frammenti giungano correttamente a destinazione. Inoltre, tutto questo deve avvenire in modo efficiente, e in modo tale da isolare i livelli superiori da inevitabili cambiamenti nella tecnologia hardware.

Sotto condizioni normali, il livello trasporto crea una connessione di rete per ogni connessione di trasporto richiesta dal livello superiore. Se la connessione di trasporto richiede un alto flusso, tuttavia, il livello trasporto può creare più connessioni di rete, dividendo i dati fra le varie connessioni per aumentare la capacità di trasmissione. D'altra parte, se mantenere o creare una connessione di rete risulta costoso, per ridurre i costi il livello trasporto può far condividere a più connessioni a livello trasporto la medesima connessione a livello rete. In ogni caso, il livello trasporto deve rendere la condivisione trasparente al livello superiore.

Il livello trasporto determina inoltre il tipo di servizi da fornire ai livelli superiori. Il più popolare tipo di connessione a livello trasporto è un canale punto-a-punto libero da errori che consegna messaggi o byte nell'ordine in cui sono stati inviati. Tuttavia, altri tipi di servizi a questo livello sono il trasporto di messaggi isolati senza garanzia sull'ordine di ricezione, o la diffusione del medesimo messaggio a destinazioni multiple. Il tipo di servizio è determinato quando la connessione viene stabilita.

Il livello trasporto è davvero punto-a-punto, dalla sorgente alla destinazione. In altre parole, un programma sulla macchina sorgente mantiene una conversazione con un programma simile sulla macchina destinazione, usando le intestazioni dei messaggi e i messaggi di controllo. Ai livelli inferiori, i protocolli sono situati fra ogni macchina e l'immediato vicino, e non fra la prima macchina sorgente e l'ultima macchina destinazione, che possono essere separati da diversi router. La differenza fra i livelli da 1 a 3, che sono collegati, e i livelli da 4 a 7, che sono in realtà punto-a-punto, è illustrata in figura 1-16. Molti host sono multiprogrammati, cosa che comporta che connessioni multiple possano entrare e uscire dall'host. C'è bisogno di qualche meccanismo per poter dire quali mes-

saggi appartengono a una certa connessione. L'intestazione a livello trasporto (H_4 in figura 1-11) può essere il luogo in cui inserire questo tipo di informazione.

Inoltre, per far condividere lo stesso canale a più sequenze di messaggi, il livello trasporto deve fare attenzione a stabilire ed eliminare connessioni lungo la rete. Questo richiede un meccanismo di denominazione, in modo tale che i processi su una macchina abbiano un modo per descrivere con chi desiderano conversare. Ci deve inoltre essere un meccanismo per regolare il flusso di informazioni, in modo tale che un host veloce non sovraccarichi uno lento. Questo meccanismo è chiamato **controllo-del-flusso** e gioca un ruolo chiave nel livello di trasporto (come in altri livelli). Il controllo del flusso fra host è distinto dal controllo del flusso fra router, tuttavia si vedrà più avanti che principi simili si applicano a entrambi.

Il livello di sessione (session)

Il **livello di sessione** permette agli utenti su macchine diverse di stabilire sessioni. Una sessione permette il trasporto ordinario di dati, realizzato dal livello trasporto, ma permette anche servizi avanzati utili a talune applicazioni. Una sessione potrebbe essere utilizzata per permettere a un utente di collegarsi a un sistema condiviso remoto o di trasferire archivi fra due macchine.

Uno dei servizi del **livello sessione** è di gestire il controllo del dialogo. Le sessioni possono permettere al traffico di viaggiare in entrambe le direzioni allo stesso tempo, o solamente in una direzione alla volta. Se il traffico può muoversi solamente in una direzione alla volta (in modo analogo a un unico binario su una linea ferroviaria), il livello sessione tiene traccia di chi è il turno attuale.

Un servizio legato al **livello sessione** è la **gestione dei token**. Per alcuni protocolli, è essenziale che entrambe le estremità non cerchino di effettuare la medesima operazione contemporaneamente. Per gestire queste attività, il livello sessione fornisce dei token che possono essere scambiati. Solamente l'estremità in possesso del token può eseguire l'operazione critica.

Un altro servizio del livello sessione è la **sincronizzazione**. Si consideri il problema che si può avere quando si cerca di effettuare un trasferimento di un archivio che impiega due ore fra due macchine che hanno un'ora come tempo medio fra guasti. Se un trasferimento si interrompe e fallisce, l'intero trasferimento dovrebbe iniziare daccapo e potrebbe a sua volta fallire. Per eliminare questo problema, il livello sessione fornisce un modo per inserire dei punti di controllo in una sequenza di dati, in modo tale che, dopo un guasto, solamente i dati trasferiti dopo l'ultimo punto di controllo debbano essere ritrasmessi.

Il livello di presentazione (presentation)

Il **livello di presentazione** realizza alcune funzioni che sono richieste abbastanza spesso da richiedere una soluzione generale, invece che lasciare a ogni utente il compito di risolvere tali problemi in modo autonomo. In particolare, diversamente dagli altri livelli sottostanti, che sono interessati solamente a muovere bit in modo affidabile da un punto a un altro, il livello presentazione fa riferimento alla sintassi e alla semantica delle informazioni trasmesse.

Un esempio tipico di un servizio del livello di presentazione è la **codifica-dei-dati** in un

modo standard riconosciuto. La maggior parte degli utenti non scambia stringhe casuali di bit binari. Scambiano nomi di persone, date, quantità di denaro e fatture. Queste informazioni sono rappresentate da stringhe di caratteri, interi, numeri a virgola mobile e strutture dati composte di diversi componenti più semplici. Calcolatori differenti hanno codici differenti per rappresentare stringhe di caratteri (ad es. ASCII e Unicode), interi (ad es. complemento a 1 e complemento a 2), e così via. Per rendere possibile la comunicazione fra calcolatori con rappresentazioni differenti, le strutture dati che devono essere scambiate devono essere definite in modo astratto, in modo tale che una codifica standard possa essere utilizzata in modo dinamico. Il livello presentazione gestisce queste strutture di dati astratte e converte dalla rappresentazione usata all'interno del calcolatore alla rappresentazione standard della rete e viceversa.

Il livello di applicazione (application)

Il livello di applicazione contiene una varietà di protocolli che sono normalmente necessari. Per esempio, ci sono centinaia di tipi di terminali non compatibili nel mondo. Si consideri la situazione limite di un editor a tutto schermo che deve lavorare su una rete con molti tipi di terminali differenti, ognuno con una diversa gestione dello schermo, diverse sequenze di escape per l'inserimento e la cancellazione di testo, diversi modi per muovere il cursore ecc.

Un modo per risolvere questo problema è di definire l'astrazione di un **terminale virtuale di rete** per cui possono essere scritti editor e altri programmi. Per gestire ogni tipo di terminale, una parte di software deve essere scritta per realizzare il terminale virtuale sul terminale reale. Per esempio, quando l'editor muove il cursore del terminale virtuale nell'angolo superiore sinistro dello schermo, questo software deve inviare il comando appropriato al terminale reale per far spostare in quel punto il suo cursore. Tutto il software del terminale virtuale costituisce il livello applicazione.

Un'altra funzione del livello applicazione è il trasferimento dei file. File system differenti hanno differenti convenzioni per nominare gli archivi, diversi modi per rappresentare linee di testo, e così via. Il trasferimento di un file fra due differenti sistemi richiede di gestire queste e altre incompatibilità. Anche questo tipo di lavoro è a carico del livello applicazione, come la posta elettronica, la gestione remota di processi, la ricerca di directory, e molte altre funzioni di uso generale o specifico.

Trasmissione dei dati nel modello OSI

La figura 1-17 mostra un esempio di come possono essere trasmessi i dati usando il modello OSI. Il processo mittente vuole spedire alcuni dati a un processo destinatario. Esso passa i dati al livello applicazione, che attacca l'intestazione AH (che potrebbe essere nulla), all'inizio dei dati stessi e che passa il blocco di dati ottenuto al livello presentazione.

Il livello presentazione può trasformare questo blocco di dati in diversi modi e forse aggiunge un'intestazione all'inizio, passando poi il risultato al livello sessione. È importante rendersi conto che il livello presentazione non è a conoscenza di quale porzione di dati costituisca AH, se questa è presente, o quali siano i veri dati dell'utente.

Questo processo viene ripetuto fino a quando i dati raggiungono il livello fisico, dove

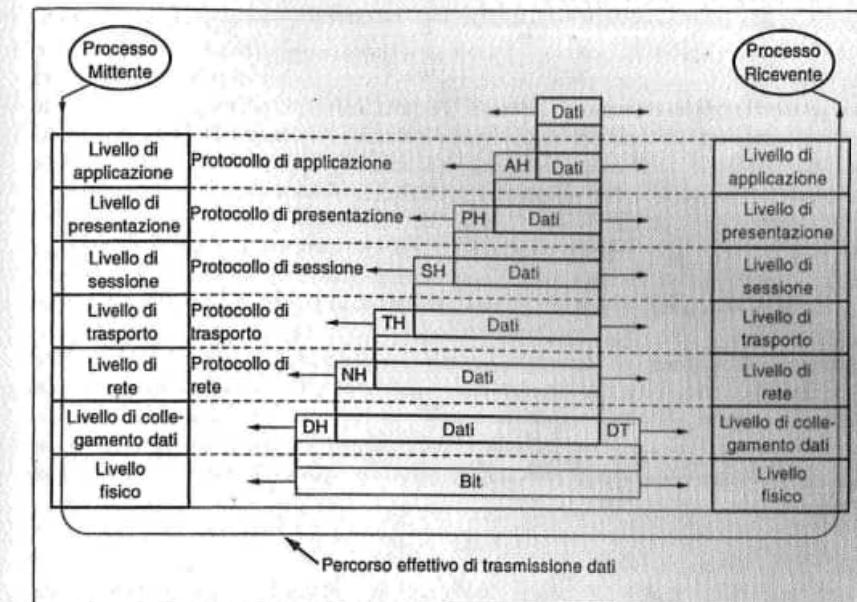


Fig. 1-17 Un esempio di come funziona il modello OSI. Alcune delle intestazioni potrebbero essere nulle (H.C. Folts. Utilizzato con permesso.).

sono finalmente trasmessi alla macchina destinazione. Su tale macchina le diverse intestazioni sono eliminate una per una mentre il messaggio si propaga lungo i vari livelli fino ad arrivare al processo destinatario.

L'idea chiave è che sebbene i dati vengano in realtà trasmessi verticalmente nella figura 1-17, ogni livello è programmato come se la comunicazione avvenisse in modo orizzontale. Quando il livello trasporto mittente, per esempio, riceve un messaggio dal livello sessione, esso vi attacca la propria intestazione e passa il tutto al relativo livello trasporto destinatario. Dal suo punto di vista, il fatto che debba passare il messaggio al livello rete della propria macchina è un aspetto tecnico trascurabile. Analogamente, quando un diplomatico parla alle Nazioni Unite nella propria lingua, egli pensa di parlare agli altri diplomatici. Il fatto che stia parlando solamente al proprio interprete è solo un dettaglio tecnico.

1.4.2 Il modello di riferimento TCP/IP

Si passa ora dal modello di riferimento OSI al modello di riferimento usato nella progenitrice di tutte le reti di calcolatori, le rete ARPANET, e nella sua erede, la rete mondiale Internet. Anche se si presenterà una breve storia della rete ARPANET nel seguito, è utile menzionare subito alcuni aspetti chiave. ARPANET era una rete di ricerca sponsorizzata dal Dipartimento della Difesa Statunitense. Essa collegava centinaia di università e strut-

ture governative utilizzando le linee telefoniche. Quando vennero aggiunte le reti satellitari o via radio, i protocolli esistenti diedero problemi di compatibilità, e venne richiesto un nuovo modello di riferimento. Quindi sin dall'inizio l'abilità di collegare più reti fra loro in modo semplice si presentò come uno dei principali aspetti di progettazione. Questo tipo di architettura divenne poi nota come **modello di riferimento TCP/IP**, a causa dei suoi due protocolli principali. Fu definito per la prima volta in Cerf, Kahn (1974). Una successiva presentazione è in Leiner *et al.* (1985). La filosofia di progettazione legata a questo modello è discussa in Clark (1988).

A causa dei timori del Dipartimento della difesa che qualcuno dei suoi preziosi host, router, o collegamenti fra reti venisse distrutto senza preavviso, un altro fra gli obiettivi principali fu di rendere la rete capace di sopravvivere al guasto di parti di hardware, in modo tale che le conversazioni esistenti non fossero danneggiate. In altre parole, il Dipartimento della Difesa desiderava che le connessioni rimanessero intatte fino a quando la macchina mittente e quella destinataria fossero ancora funzionanti, anche se alcune delle macchine o delle linee di trasmissione intermedie fossero messe fuori uso. Inoltre, il Pentagono richiese un'architettura flessibile, in quanto si stavano studiando parecchi tipi diversi di applicazioni, dal trasferimento di archivi fino alle trasmissioni audio in tempo reale.

Il livello internet

Tutte queste richieste portarono alla scelta di una rete a commutazione di pacchetto basata su un livello privo di connessione. Questo livello, chiamato **il livello internet**, è il perno che mantiene assieme l'intera architettura. Il suo compito è di permettere a un host di inserire pacchetti in una qualsiasi rete in modo tale che questi viaggino indipendentemente verso la destinazione (potenzialmente anche su una rete differente). Essi possono arrivare anche in un ordine diverso rispetto a quello con cui erano stati inviati, in questo caso sarà compito di un qualche livello superiore riordinarli. Si noti che "internet" viene utilizzato in senso generico, anche se tale livello è presente nell'Internet.

L'analogia in questo caso è con il sistema postale. Una persona può imbucare una sequenza di lettere internazionali in una cassetta in una certa nazione, e con un po' di fortuna, molte di esse saranno consegnate all'indirizzo corretto nella nazione di destinazione. Probabilmente le lettere viaggeranno attraverso una o più nazioni lungo il loro cammino, ma questo è completamente trasparente agli utenti. Inoltre, viene nascosto agli utenti il fatto che ogni nazione (cioè ogni rete) possa avere i propri francobolli, le proprie dimensioni preferite, o le proprie regole di spedizione.

Il livello internet definisce un formato di pacchetto ufficiale e un protocollo chiamato **IP (Internet Protocol – protocollo internet)**. Lo scopo del livello internet è di consegnare i pacchetti IP dove si supponga debbano andare. La scelta del cammino dei pacchetti in questo caso è la questione principale, così come evitare la congestione. Per questi motivi, è ragionevole dire che il livello internet TCP/IP è molto simile nelle sue funzionalità al livello rete del modello OSI. La figura 1-18 mostra questa corrispondenza.

Il livello di trasporto

Il livello superiore al livello internet nel modello TCP/IP è chiamato **il livello di traspor-**

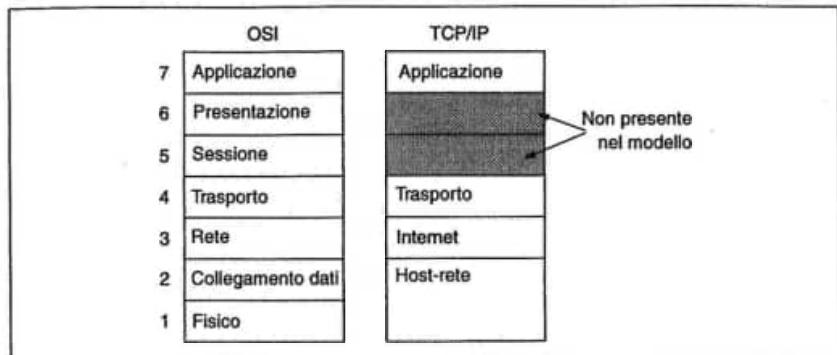


Fig. 1-18 Il modello di riferimento TCP/IP.

to. Serve per permettere alle entità pari livello sugli host sorgente e destinazione di portare avanti una conversazione, come nel livello trasporto del modello OSI. Due protocolli di collegamento sono definiti in questo livello. Il primo, **TCP (Transmission Control Protocol – protocollo di controllo della trasmissione)** è un protocollo orientato alla connessione affidabile che permette a sequenze di byte originate su una macchina di essere consegnate senza errori su una qualsiasi altra macchina della rete. Essò frammenta la sequenza entrante di byte in messaggi e li passa al livello internet. Sulla destinazione, il processo TCP ricevente riassembra i messaggi ricevuti nella sequenza in uscita. Il protocollo TCP gestisce anche il flusso del controllo per essere sicuro che un mittente veloce non possa sovraccaricare un ricevente lento con più messaggi di quelli che è in grado di gestire.

Il secondo protocollo in questo livello, **UDP (User Datagram Protocol – protocollo per datagrammi degli utenti)**, è un protocollo inaffidabile, privo di connessione, per applicazioni che non desiderano la sequenzializzazione o il controllo del flusso del protocollo TCP e che desiderano gestire tutto questo in modo autonomo. È spesso utilizzato per comunicazioni veloci, per richieste e risposte fra un client e un server, o applicazioni in cui la prontezza nella consegna è più importante che la sua accuratezza, come nel caso di trasmissioni audiovisive. La relazione fra IP, TCP e UDP è mostrata in figura 1-19. Da quando il modello fu sviluppato, IP è stato implementato su molte altre reti.

Il livello delle applicazioni

Il modello TCP/IP non ha i livelli presentazione e sessione. Non sono necessari, quindi sono stati esclusi. L'esperienza con il modello OSI ha provato la correttezza di tale punto di vista: questi livelli sono poco utili nella maggior parte delle applicazioni.

Subito sopra al livello trasporto si trova il **livello applicazione**. Esso contiene tutti i protocolli ad alto livello. I più antichi sono il terminale virtuale (TELNET), il trasferimento di archivi (FTP), e la posta elettronica (SMTP), come mostra la figura 1-19. Il protocollo di terminale virtuale permette a un utente su una macchina di collegarsi su una macchina remota e di lavorare su di essa. Il protocollo per il trasferimento di archivi

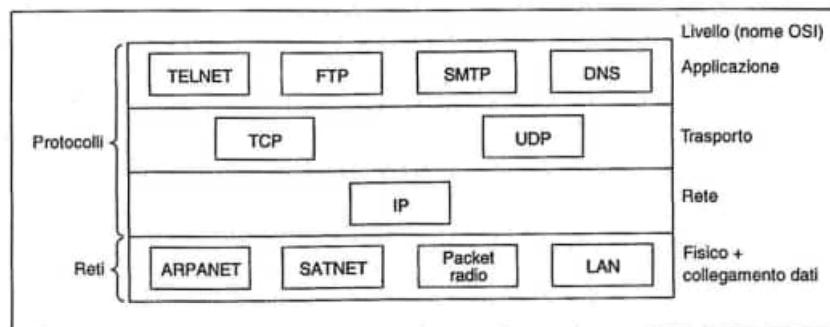


Fig. 1-19 Protocolli e reti originali nel modello TCP/IP.

fornisce un meccanismo per spostare gli archivi in modo efficiente da una macchina a un'altra. La posta elettronica era in origine solo un tipo di trasferimento di archivi, poi venne sviluppato un protocollo più specializzato. Molti altri protocolli sono stati aggiunti a questi nell'arco degli anni, come il Domain Name Service (DNS – il servizio per i nomi del dominio) per associare a ogni host il proprio indirizzo di rete, NNTP, il protocollo per muovere i messaggi dei gruppi di discussione, e HTTP, il protocollo utilizzato per caricare pagine sul World Wide Web, e molti altri.

Il livello host-to-network

Sotto al livello internet c'è un grande vuoto. Il modello di riferimento TCP/IP non dice molto a proposito di quello che avviene a questo livello, eccetto che l'host deve connettersi alla rete utilizzando un protocollo in modo da inviare pacchetti IP lungo di essa. Questo protocollo non viene definito e varia da host a host e da rete a rete. Gli articoli riguardanti il modello TCP/IP raramente discutono questo aspetto.

1.4.3 Un confronto fra i modelli di riferimento OSI e TCP/IP

I modelli di riferimento OSI e TCP/IP hanno molto in comune. Essi sono entrambi basati sul concetto di una pila di protocolli indipendenti e sovrapposti. Anche le funzionalità dei vari livelli sono abbastanza simili. Per esempio, in entrambi i modelli i livelli fino al livello trasporto hanno il compito di fornire ai processi che desiderano comunicare un servizio di trasporto indipendente dalla rete sottostante. Questi livelli costituiscono il fornitore del trasporto. Inoltre, in entrambi i modelli i livelli superiori al livello trasporto sono costituiti da servizi orientati alle applicazioni che utilizzano i servizi di trasporto. Nonostante queste fondamentali similitudini, i due modelli hanno anche molte differenze. In questo paragrafo analizziamo le differenze fondamentali fra i due modelli di riferimento. È importante notare che si analizzeranno i *modelli di riferimento*, e non le *pile di protocolli*. I protocolli stessi verranno analizzati nel seguito. Un intero libro che confronta TCP/IP e OSI è quello di Piscitello, Chapin (1993).

Tre concetti sono centrali nel modello OSI:

1. Servizi.
2. Interfacce.
3. Protocolli.

Probabilmente il principale contributo del modello OSI è di distinguere chiaramente fra questi tre concetti. Ogni livello realizza dei servizi per il livello superiore. La definizione del *servizio* dice cosa fa il livello in questione, non dice nulla di come le entità accedano al servizio o di come esso venga realizzato.

Una *interfaccia* di livello dice come i processi dei livelli superiori possono accedere al livello stesso. Essa specifica quali devono essere i parametri e quali risultati ci si deve attendere. Essa non dice ancora nulla di come il livello lavora al proprio interno.

Infine, i *protocolli* usati a un certo livello risultano essere le vere attività del livello. Il livello può utilizzare ogni protocollo a sua disposizione, al fine di portare a termine il proprio lavoro (cioè realizzare i servizi forniti). Esso può inoltre cambiare tali protocolli sempre che non vengano intaccati i programmi ai livelli superiori.

Queste idee corrispondono alle moderne idee a proposito della programmazione orientata agli oggetti. Un oggetto, come un livello, ha un insieme di metodi (operazioni) che i processi esterni all'oggetto possono invocare. La semantica di questi metodi definisce l'insieme dei servizi che l'oggetto offre. I parametri dei metodi e i relativi risultati formano l'interfaccia dell'oggetto. Il codice interno all'oggetto coincide col suo protocollo e non è visibile agli oggetti esterni.

Il modello TCP/IP non distingueva in origine fra servizi, interfacce e protocolli, tuttavia si è cercato a posteriori di renderlo più simile al modello OSI. Per esempio, gli unici veri servizi offerti dal livello internet sono INVIA PACCHETTO IP e RICEVI PACCHETTO IP. Di conseguenza, i protocolli nel modello OSI sono più nascosti rispetto al modello TCP/IP e possono essere rimpiazzati in modo relativamente semplice al variare della tecnologia. Essere in grado di fare queste variazioni è uno dei principali motivi per cui avere protocolli a livelli.

Il modello di riferimento OSI venne concegnato *prima* che i protocolli venissero inventati. Questo ordine indica che il modello non presentava preferenze per qualche insieme particolare di protocolli, rendendolo così il più generale possibile. L'aspetto negativo di questo ordine è che i progettisti non avevano una grossa esperienza delle problematiche di comunicazione e non ebbero idee chiare riguardo alle funzionalità da inserire nei vari livelli.

Per esempio, il livello data link originariamente si interessava solamente di reti punto-a-punto. Quando si realizzarono reti a broadcast, un nuovo sottolivello venne inserito nel livello. Quando si iniziò a realizzare reti utilizzando il modello OSI e i protocolli esistenti, si scoprì che essi non soddisfacevano le specifiche dei servizi (meraviglia delle meraviglie), quindi si dovettero inserire dei sottolivelli nel modello in modo da soddisfare queste differenze. Infine, il comitato originariamente progettò che ogni nazione avesse la sua rete, utilizzata dal governo e basata su protocolli OSI, e non si pensò alle problematiche di collegamento delle singole reti. Ma le cose, per dirla in breve, non andarono realmente in questo modo.

Con il modello TCP/IP risulta vero l'inverso: prima venivano progettati i protocolli, e il modello risultava una descrizione dei protocolli esistenti. Non c'era nessun problema nel far corrispondere modello e protocolli. Essi combaciavano perfettamente. L'unico problema era che il *modello* non si adattava a nessun'altra pila di protocolli. Di conseguenza, non era utile per descrivere reti non TCP/IP.

Spostandosi da considerazioni di tipo filosofico verso considerazioni più tecniche, una differenza ovvia fra i due modelli è il numero di livelli: il modello OSI ha sette livelli mentre il TCP/IP ne ha quattro. Entrambi hanno i livelli rete, trasporto e applicazione, ma gli altri livelli sono diversi.

Un'altra differenza è nel campo delle comunicazioni che possono essere orientate alla connessione o prive di connessione. Il modello OSI fornisce entrambe le connessioni a livello rete, ma solamente comunicazione orientata alla connessione a livello trasporto, dove conta (in quanto i servizi del livello trasporto sono quelli visibili all'utente). Il modello TCP/IP ha solamente un modo a livello rete (privo di connessione) ma fornisce entrambi i modi al livello trasporto, dando la possibilità di scegliere all'utente. Questa scelta è importante per semplici protocolli del tipo richiesta-risposta.

1.4.4 Una critica al modello e ai protocolli OSI

Né il modello o i protocolli OSI, né quelli TCP/IP sono perfetti. Si possono criticare entrambi i modelli. In questo e nel prossimo paragrafo presentiamo alcune delle critiche più importanti. Iniziamo col modello OSI per esaminare poi TCP/IP.

Quando venne pubblicata la seconda edizione di questo libro (1989), ai più esperti del settore sembrava che il modello OSI e i suoi protocolli fossero pronti per invadere il mondo spazzando via tutto ciò che non fosse compatibile con essi. Questo non avvenne. Perché? Può essere utile una riflessione sulle ragioni per cui questo non avvenne. Tali ragioni possono essere riassunte come segue:

1. Cattiva scelta del momento.
2. Cattiva tecnologia.
3. Cattive implementazioni.
4. Cattiva politica.

Cattiva scelta del momento

Analizziamo la prima motivazione. Il momento in cui uno standard viene definito è una caratteristica chiave ai fini del suo successo. David Clark del MIT ha una teoria degli standard che egli chiama *l'apocalisse dei due elefanti*, che è illustrata in figura 1-20.

Questa figura mostra la quantità di attività che coinvolge un nuovo soggetto. Quando il soggetto viene scoperto, c'è un'esplosione di attività di ricerca nella forma di discussioni, articoli e incontri. Dopo un po' questa diminuisce, le industrie scoprono il soggetto, e arriva l'onda dei miliardi di dollari di investimenti.

È essenziale che lo standard venga definito nell'intervallo fra i due "elefanti". Se essi vengono definiti troppo presto, prima che la ricerca abbia termine, il soggetto può essere ancora poco conosciuto, cosa che comporta standard sbagliati. Se essi vengono definiti troppo tardi, molte industrie potrebbero aver già effettuato i loro maggiori investimenti

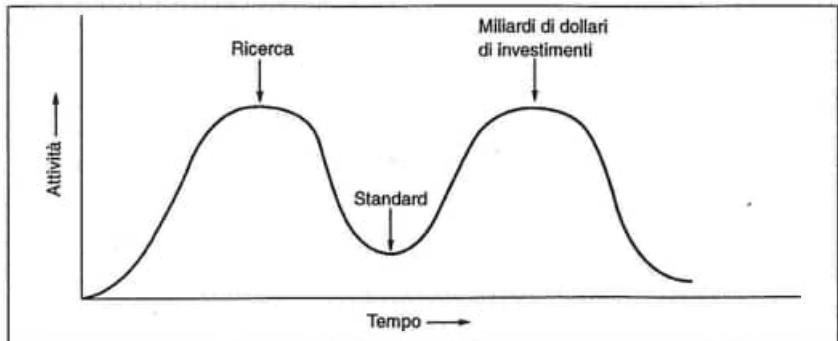


Fig. 1-20 L'apocalisse dei due elefanti.

senza aver tenuto in considerazione gli standard. Se l'intervallo fra i due elefanti è molto breve (perché si ha fretta di iniziare), coloro che sviluppano gli standard potrebbero fallire. Adesso appare chiaro perché i protocolli standard OSI sono falliti. I protocolli antagonisti TCP/IP erano già largamente utilizzati dalle università che facevano ricerca ai tempi in cui apparirono i protocolli OSI. Mentre l'onda dei miliardi di dollari non era ancora arrivata, il mercato accademico era comunque ampio a sufficienza per permettere a molte aziende di offrire, anche se cautamente, i loro prodotti TCP/IP. Quando apparve il modello OSI, esse non erano interessate a realizzare una seconda serie di protocolli a meno che non ne fossero costrette, così non ci fu un'offerta iniziale. Ogni azienda attese le altre, ma nessuna azienda si offrì di realizzarlo, e OSI non venne realizzato.

Cattiva tecnologia

La seconda ragione per cui OSI non venne mai realizzato è che sia il modello che i protocolli sono difettosi. Molte discussioni a proposito del modello a sette livelli danno l'impressione che il numero e il contenuto degli stessi fosse il più naturale o il più ovvio. Questo non è vero. Il livello sessione è scarsamente utile nella maggior parte delle applicazioni, e il livello presentazione è quasi vuoto. Infatti la proposta inglese all'ISO aveva solamente cinque livelli, non sette. In contrasto con i livelli sessione e presentazione, i livelli data link e rete sono talmente pieni che lavori successivi hanno diviso tali livelli in molteplici sottolivelli, ognuno con funzioni differenti.

Anche se di rado viene pubblicamente ammesso, la ragione reale per cui il modello OSI ha sette livelli è che ai tempi in cui venne progettato, l'IBM era proprietaria di un protocollo a sette livelli chiamato SNA (Systems Network Architecture – architettura per sistemi di rete). A quei tempi, l'IBM aveva un dominio nell'industria tale che chiunque altro, incluse le società telefoniche, le società competitive, e anche i principali governi, avevano una grossa paura che l'IBM volesse usare il suo mercato per costringere chiunque a utilizzare SNA, che avrebbe potuto poi essere modificato ogni volta l'IBM l'avesse voluto. L'idea alla base del modello OSI era di produrre una pila di protocolli e un modello

di riferimento simile a quello IBM, che potesse diventare uno standard mondiale, e che fosse controllato non da una singola azienda, ma da una organizzazione neutrale, l'ISO. Il modello OSI, assieme alle sue definizioni di servizi e protocolli, è straordinariamente complesso. Quando viene stampato in modo completo e su modulo continuo, la stampa di questo standard occupa una pila di carta alta quasi un metro. I protocolli sono inoltre difficili da implementare e i servizi sono inefficienti. In questo contesto, viene in mente un indovinello di Paul Mockapetris citato in Rose (1993):

D: Cosa ottieni quando utilizzi uno standard internazionale?

R: Un'offerta che non si riesce a capire.

Oltre a essere incomprensibile, un altro problema del modello OSI è che alcune funzioni, come l'indirizzamento, il controllo del flusso e il controllo degli errori appaiono di volta in volta nei diversi livelli. Saltzer *et al.* (1984), per esempio, hanno mostrato che, per essere efficace, il controllo degli errori deve essere realizzato al livello più alto, in quanto ripeterlo lungo ognuno dei livelli inferiori è spesso inefficiente e non necessario.

Un altro aspetto è che la decisione di inserire alcune caratteristiche in particolari livelli non è sempre ovvia. La gestione del terminale virtuale (adesso inserita nel livello applicazione) era nel livello presentazione durante la maggior parte dello sviluppo dello standard. Venne spostato al livello applicazione perché il comitato ebbe problemi per decidere quali erano gli scopi del livello presentazione. La sicurezza dei dati e la crittografia erano problemi così controversi che non si riuscirono ad accordare sul livello in cui inserirle, così vennero lasciate fuori entrambe. La gestione delle reti venne anch'essa omessa dal modello per ragioni similari.

Un'ulteriore critica allo standard originale è che esso ignorava del tutto i servizi e i protocolli privi di connessione, anche se molte reti locali lavoravano in questo modo. Appendici successive (conosciute nel mondo della programmazione come fissaggio degli errori) corressero questo problema.

Forse la critica più seria è che il modello è dominato da una mentalità legata alla comunicazione. La relazione fra computazione e comunicazione appare un po' ovunque, e alcune delle scelte che sono state fatte sono completamente inappropriate rispetto al modo in cui lavorano i calcolatori e i programmi. Per esempio, si considerino le primitive OSI, elencate in figura 1-14. In particolare, analizziamo attentamente le primitive e come dovrebbero essere utilizzate nei linguaggi di programmazione.

La primitiva CONNECT.request è semplice. Si può immaginare una procedura di libreria, *connect*, che i programmi invocano per stabilire una connessione. Adesso si pensi alla primitiva CONNECT.indication. Quando arriva un messaggio, il processo destinatario deve riceverne segnalazione. Concretamente, deve ricevere un'interruzione – una nozione poco adeguata ai programmi scritti in qualsiasi moderno linguaggio ad alto livello. Sicuramente, al livello inferiore avviene un'interruzione che segnala l'arrivo del messaggio. Se il programma è in attesa di un messaggio in arrivo, potrebbe richiamare una procedura di libreria *receive* per bloccarsi. Ma in questo caso, perché non viene utilizzata la primitiva *receive* invece di *indication*? *Receive* è chiaramente orientata verso il modo in cui lavorano i calcolatori, mentre *indication* è allo stesso modo orientata al modo in cui funziona i telefoni. I calcolatori sono diversi dai telefoni. I telefoni suonano; i calcolatori non suonano. In breve, il modello semantico di un sistema comandato dalle interruzioni è concettualmente una pessima idea e totalmente lontana da tutte le moderne filosofie di programmazione strutturata. Questo e problemi similari vengono discussi da Langsford (1984).

nano i telefoni. I calcolatori sono diversi dai telefoni. I telefoni suonano; i calcolatori non suonano. In breve, il modello semantico di un sistema comandato dalle interruzioni è concettualmente una pessima idea e totalmente lontana da tutte le moderne filosofie di programmazione strutturata. Questo e problemi similari vengono discussi da Langsford (1984).

Cattive implementazioni

Data l'enorme complessità del modello e dei protocolli, non dovrebbe sorprendere che le implementazioni iniziali risultassero pesanti, poco maneggevoli e lente. Chiunque le provasse ne rimaneva deluso. Non ci volle molto tempo perché si associasse "OSI" con "bassa qualità". Anche se col passar del tempo i prodotti vennero migliorati, l'immagine rimase danneggiata.

Al contrario, una delle prime implementazioni del TCP/IP fu una componente dello UNIX di Berkeley e fu abbastanza buona. Gli utenti la accettarono velocemente, e questo fatto creò una vasta comunità, che portò a miglioramenti, che a loro volta allargarono la comunità. In questo caso la spirale crebbe verso l'alto anziché verso il basso.

Cattiva politica

A causa della sua prima implementazione, in molti, specialmente in ambito accademico, pensavano che il TCP/IP fosse parte di UNIX, e UNIX negli anni ottanta era per l'ambiente accademico qualcosa come la mamma o la torta di mele.

OSI, invece, veniva visto come la creatura dei ministri delle telecomunicazioni europei, della Comunità Europea, e più tardi anche del Governo degli Stati Uniti. Questa credenza era in parte veritiera, ma l'idea di un gruppo di burocrati governativi che cercavano di imporre ai programmati e ai ricercatori che stavano sviluppando le reti di calcolatori uno standard tecnicamente inferiore, non aiutava molto. Alcuni videro questo sviluppo nello stesso modo in cui venne visto l'annuncio dato dall'IBM negli anni sessanta che il PL/I doveva essere il linguaggio del futuro, oppure il Dipartimento della difesa degli Stati Uniti che più tardi annunciava la medesima cosa per Ada.

Nonostante il fatto che il modello e i protocolli OSI non ebbero un gran successo, ci sono ancora alcune organizzazioni interessate a esso, principalmente le PTT (Poste, Telegrafi e Telecomunicazioni) europee che hanno tuttora il monopolio delle telecomunicazioni. A causa di questo, è stato fatto un piccolo sforzo per aggiornare il modello OSI, sforzo che ha prodotto un modello rivisitato pubblicato nel 1994. Per le cose che sono state cambiate (poche) e quelle che avrebbero dovuto subire variazioni (molte), vedi Day (1995).

1.4.5 Una critica al modello di riferimento TCP/IP

Il modello e i protocolli TCP/IP hanno anch'essi i loro problemi. Innanzitutto, il modello non distingue chiaramente i concetti di servizi, interfacce e protocolli. Una buona ingegnerizzazione del software richiede una differenziazione fra specifica e implementazione, una cosa che OSI fa con molta attenzione, e TCP/IP non fa. Di conseguenza, il modello TCP/IP è solo una guida per chi desidera progettare nuove reti usando nuove tecnologie. Secondo, il modello TCP/IP non è del tutto generale e non è adatto a descrivere una

qualsiasi pila di protocolli oltre al TCP/IP. Descrivere SNA utilizzando il modello TCP/IP potrebbe essere impossibile, per esempio.

Terzo, il livello host-to-network non è un livello nel vero senso del termine. Esso è un'interfaccia (fra i livelli rete e data link). La distinzione fra un'interfaccia e un livello è cruciale, e non si dovrebbe essere superficiali rispetto a un problema del genere.

Quarto, il modello TCP/IP non distingue il livello fisico dal livello data link, e nemmeno li menziona. I due livelli sono completamente diversi. Il livello fisico deve avere a che fare con le caratteristiche di trasmissione, con i fili di rame, con le fibre ottiche o le comunicazioni senza fili. Il compito del livello data link è di limitare l'inizio e la fine dei blocchi di dati e di spostarli da una parte all'altra della rete con il richiesto grado di affidabilità. Un modello appropriato li dovrebbe contenere entrambi come livelli separati. Il modello TCP/IP non fa questo.

Infine, anche se i protocolli IP e TCP vennero studiati con attenzione, e correttamente implementati, molti altri protocolli risultarono specifici, generalmente prodotti da studenti di dottorato che li realizzavano nel loro tempo libero. Le implementazioni venivano così distribuite liberamente, e divennero largamente utilizzate, fortemente radicate, tanto da diventare difficili da rimpiazzare. Alcune di queste sono oggi sorgenti di imbarazzo. Il protocollo di terminale virtuale, TELNET, per esempio, venne progettato per un terminale meccanico telescrivente da dieci caratteri al secondo. Non si interessa di interfacce grafiche o del mouse. Ciò nonostante, 25 anni dopo, esso è ancora largamente utilizzato. Riassumendo, nonostante i suoi problemi, il *modello* OSI (meno i livelli sessione e presentazione) è dimostrabilmente molto utile per discutere di reti di calcolatori. Al contrario, i *protocolli* OSI non sono molto popolari. È vero l'inverso per TCP/IP: il *modello* è praticamente inesistente, ma i *protocolli* sono largamente usati. In questo libro si utilizza un modello OSI modificato anche se ci si concentrerà principalmente sui protocolli TCP/IP e correlati, come anche di protocolli più nuovi come SMDS, frame relay, SONET e ATM. In effetti, verrà utilizzato il modello ibrido di figura 1-21 come base per questo libro.



Fig. 1-21 Il modello riferimento ibrido utilizzato in questo libro.

1.5 Esempi di reti

Numerose reti stanno attualmente operando in tutto il mondo. Alcune di queste sono pubbliche gestite da imprese o dalle PTT (Poste, Telegrafi e Telecomunicazioni), altre sono reti per ricercatori, altre sono reti cooperative gestite dagli stessi utenti, e altre ancora

sono reti commerciali o private. Nel prossimo paragrafo si presenteranno alcune reti attuali o storiche per avere una idea di come sono (o sono state) e di come esse differiscono l'una dall'altra.

Le reti differiscono nella loro storia, amministrazione, servizi forniti, progettazione tecnica e comunità di utenti. La storia e l'amministrazione possono variare da una rete pianificata attentamente da una singola organizzazione con uno scopo ben preciso, a una collezione di elaboratori che sono stati collegati l'un l'altro durante gli anni senza una precisa pianificazione o un'amministrazione centrale. I servizi forniti spaziano dalla comunicazione interprocesso alla posta elettronica, il trasferimento di archivi, il collegamento e l'esecuzione remota. La progettazione tecnica può cambiare nel mezzo di trasmissione utilizzato, il processo di denominazione e gli algoritmi per il cammino dei messaggi, e i protocolli utilizzati. Infine, la comunità degli utenti può variare dalla singola società a tutti gli informatici accademici del mondo industrializzato.

Nel prossimo paragrafo si presenteranno alcuni esempi. Questi sono il pacchetto commerciale Novell NetWare, la rete mondiale Internet (inclusi i suoi predecessori, le reti ARPANET e NSFNET) e le prime reti con velocità dell'ordine dei gigabit.

1.5.1 Novell NetWare

La rete più popolare nel mondo dei PC è Novell NetWare. Venne progettata per l'uso da parte di aziende che volevano passare da mainframe a PC. In questi sistemi, ogni utente ha il proprio PC che funziona come cliente. Un certo numero di potenti PC operano invece come server, fornendo i servizi per gli archivi, le basi di dati e altri servizi forniti ai propri client. In altre parole, Novell NetWare è basato sul modello client-server. NetWare utilizza una propria pila di protocolli illustrata in figura 1-22. Essa si basa sul vecchio Xerox Network System, XNS, ma con varie modifiche. Novell NetWare è antecedente al modello OSI e non è basato su di esso. Somiglia più a TCP/IP che al modello OSI.

Livello	SAP	File server	...
Applicazione	SAP	File server	...
Trasporto	NCP		SPX
Rete	IPX		
Collegamento dati	Ethernet	Token ring	ARCnet
Fisico	Ethernet	Token ring	ARCnet

Fig. 1-22 Il modello di riferimento Novell NetWare.

I livelli fisico e data link possono essere scelti fra diversi standard industriali, inclusi Ethernet, IBM token ring e ARCnet. Il livello rete esegue un protocollo di comunicazione non affidabile privo di connessione chiamato IPX. Esso trasporta pacchetti in modo trasparente da sorgente a destinazione, anche se la sorgente e la destinazione sono su reti

differenti. IPX è nelle sue funzioni simile all'IP, solo che utilizza indirizzi da 10 byte invece di quelli da 4 byte. La saggezza di questa scelta si chiarirà nel capitolo 5.

Sopra all'IPX si trova un protocollo orientato alla connessione chiamato **NCP (Network Core Protocol)**. NCP fornisce anche diversi servizi oltre al trasporto dei dati ed è il vero cuore di NetWare. È disponibile un secondo protocollo, **SPX**, ma fornisce solamente trasporto. TCP è un'altra opzione. Le applicazioni possono sceglierne una. Per esempio, il file system utilizza NCP mentre Lotus Notes usa SPX. I livelli sessione e presentazione non esistono. Diversi protocolli applicativi sono presenti nel livello applicazione.

Come nel modello TCP/IP, la chiave dell'intera architettura è il pacchetto che viaggia sulla rete; ogni altra cosa viene costruita su di esso. Il formato di un pacchetto IPX è mostrato in figura 1-23. Il campo *Checksum* è usato raramente, in quanto il sottostante livello data link fornisce un servizio equivalente. Il campo *Packet length* dice quanto è lungo l'intero pacchetto, intestazione più dati. Il campo *Transport control* conta quante reti sono state attraversate dal pacchetto. Quando questo campo eccede un valore massimo, il pacchetto viene scartato. Il campo *Packet type* è usato per marcare diversi pacchetti di controllo. I due indirizzi contengono ciascuno un numero di rete di 32 bit, un numero di macchina da 48 bit (l'indirizzo 802 LAN) e l'indirizzo locale della macchina (socket) da 16 bit. Infine, vengono i dati, che occupano il resto del pacchetto, la cui dimensione massima viene definita dalla rete sottostante.

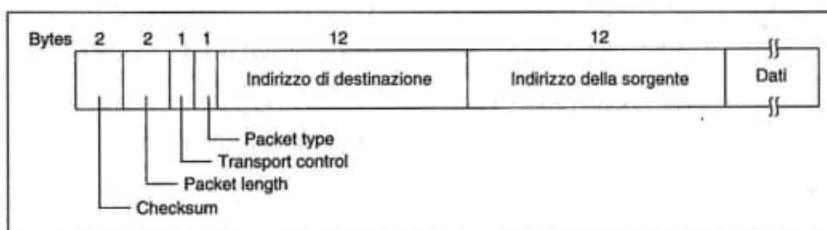


Fig. 1-23 Un pacchetto IPX della Novell NetWare.

Circa una volta al minuto, ogni server diffonde un pacchetto con cui fornisce il proprio indirizzo e descrive i servizi che offre. Questa diffusione utilizza il **SAP (Service Advertising Protocol** – protocollo di pubblicizzazione dei servizi). I pacchetti vengono visti e collezionati da processi agenti attivi sulle macchine router. Questi agenti usano le informazioni contenute in essi per costruire una base di dati dei server che sono attivi. Quando una macchina cliente viene inizializzata, essa diffonde una richiesta in cui chiede dove si trovano i server più vicini. L'agente sulla macchina router riceve tale richiesta, guarda nella propria base di dati, e determina i migliori server. La scelta dei server da utilizzare è quindi inviata al cliente. Il cliente è ora in grado di stabilire una connessione NCP con il server. Utilizzando tale connessione, il cliente e il server negoziano la massima dimensione del pacchetto. Da questo punto in poi, il cliente può accedere al file system e ad altri servizi utilizzando questa connessione. Può inoltre richiedere alla base di dati del server di cercare altri server (più distanti).

1.5 Esempi di reti

1.5.2 ARPANET

Passiamo adesso dalle reti locali alle reti geografiche. A metà degli anni sessanta, ai tempi della guerra fredda, il Dipartimento della difesa degli Stati Uniti voleva costruire una rete di comando e di controllo che potesse sopravvivere alla guerra nucleare. Le tradizionali reti telefoniche a commutazione erano considerate troppo vulnerabili, in quanto la perdita di una linea o di un commutatore avrebbe interrotto tutte le comunicazioni che li stavano utilizzando e avrebbe potuto frazionare la rete. Per risolvere questo problema, il Dipartimento della difesa si rivolse al suo istituto dedicato alla ricerca, ARPA (in seguito DARPA, e ora di nuovo ARPA), la (Defense) Advanced Research Projects Agency – agenzia per i progetti di ricerca avanzata (di difesa).

ARPA venne creato come risposta al lancio dello Sputnik sovietico nel 1957. Aveva la missione di incoraggiare il progresso nelle tecnologie che potessero essere utili all'esercito. ARPA non possiede scienziati o laboratori; infatti, ha solamente un ufficio e un budget basso (per gli standard del Pentagono). L'istituto realizzò i propri obiettivi offrendo fondi di ricerca e contratti alle università e alle aziende le cui idee sembravano promettenti.

Inizialmente, diversi fondi arrivarono alle università per investigare l'allora nuova idea della comunicazione a commutazione di pacchetti, che era stata suggerita da Paul Baran in una serie di rapporti della RAND Corporation pubblicati nei primi anni Sessanta. Dopo alcune discussioni con diversi esperti, ARPA decise che la rete di cui necessitava il Dipartimento della difesa doveva essere una rete a commutazione di pacchetto, costituita da sottoreti e calcolatori host.

Le sottoreti dovevano consistere di minicalcolatori chiamati **IMP (Interface Message Processors** – processori dei messaggi di interfaccia) collegati a delle linee di trasmissione. Per avere alta affidabilità, ogni IMP sarebbe stato connesso ad almeno altri due IMP. La sottorette doveva essere una sottorette a datagramma, così che se alcune linee o IMP fossero stati distrutti, i messaggi avrebbero potuto passare su cammini alternativi.

Ogni nodo della rete doveva consistere di un IMP e di un sistema host, nella stessa stanza, collegati da cavi corti. Un host avrebbe dovuto inviare messaggi fino a 8063 bit al proprio IMP, che li avrebbe spezzati in pacchetti fino a 1008 bit e li avrebbe inviati in modo indipendente verso la loro destinazione. Ogni pacchetto veniva ricevuto interamente prima di essere spedito, così la sottorete fu la prima rete elettronica a commutazione di pacchetto del tipo store-and-forward (memorizza e fai proseguire).

ARPA fece quindi un'offerta per la costruzione della sottorete. Dodici aziende fecero proposte. Dopo la valutazione delle proposte, ARPA scelse BBN, una ditta di consulenze a Cambridge, Massachusetts, e nel dicembre 1968 stipulò un contratto per la costruzione della sottorete e per la scrittura dei relativi programmi. BBN scelse di utilizzare come IMP degli speciali minicalcolatori Honeywell DDP-316 modificati, con una memoria centrale da 12K di parole da 16 bit. Gli IMP non avevano dischi, in quanto parti mobili erano considerate poco affidabili. Gli IMP erano collegati attraverso linee da 56 kbps noleggiate dalle aziende telefoniche.

I programmi vennero divisi in due parti: host e sottorete. I programmi per la sottorete consistevano della parte di gestione dell'IMP, per quanto riguardava la connessione host-IMP, il protocollo IMP-IMP e un protocollo sorgente IMP-destinazione IMP, progettato

per incrementare l'affidabilità. Il progetto originale della ARPANET è mostrato in figura 1-24.

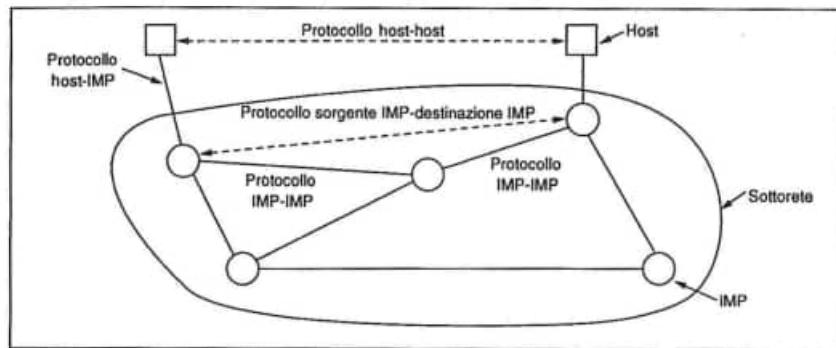


Fig. 1-24 La struttura originale di ARPANET.

A parte la gestione di rete erano necessari altri programmi, ovvero la parte di gestione dell'host per quanto riguardava la connessione host-IMP, il protocollo host-host e i programmi applicativi. Fu chiaro abbastanza presto che la BBN riteneva il proprio compito limitato a trasferire un messaggio da un cavo di collegamento host-IMP al cavo IMP-host destinazione.

Larry Roberts di ARPA convocò un incontro per ricercatori sulle reti, in maggior parte studenti di dottorato, a Snowbird, Utah, nell'estate del 1969, sul tema dei programmi per host. I dottorandi si aspettavano la presenza di qualche esperto di reti che avrebbe esposto il progetto della rete e dei suoi programmi e che avrebbe assegnato a ciascuno il compito per scrivere parte di essi. Essi furono stupiti quando videro che non c'erano esperti di reti e nemmeno un progetto ben definito. Loro stessi dovevano definire in modo autonomo cosa realizzare.

Una rete sperimentale venne completata nel dicembre 1969 con quattro nodi: UCLA, UCSB, SRI, e l'università di Utah. Queste quattro istituzioni vennero scelte perché tutte avevano un gran numero di contratti ARPA, e tutte possedevano calcolatori completamente incompatibili (tanto per rendere il gioco più divertente). La rete si ampliò velocemente dopo la consegna e installazione di nuovi IMP; presto coprì gli interi Stati Uniti. La figura 1-25 mostra quanto la rete si allargò nei suoi primi tre anni di vita.

Più tardi i programmi IMP vennero modificati per permettere ai terminali di collegarsi direttamente a uno speciale IMP, chiamato TIP (Terminal Interface Processor – processore delle interfacce per terminali), senza dover passare per un host. Altre varianti inclusero la possibilità di avere host multipli per ogni IMP (per risparmiare denaro), host capaci di colloquiare con IMP multipli (per salvaguardarsi da guasti degli IMP), e host e IMP separati da ampie distanze (per permettere host lontani dalla sottorete).

Oltre ad aiutare la crescita della neonata ARPANET, ARPA finanziò ricerche sulle reti satellitari e le reti radiomobili a pacchetti. In una famosa dimostrazione, un camion che

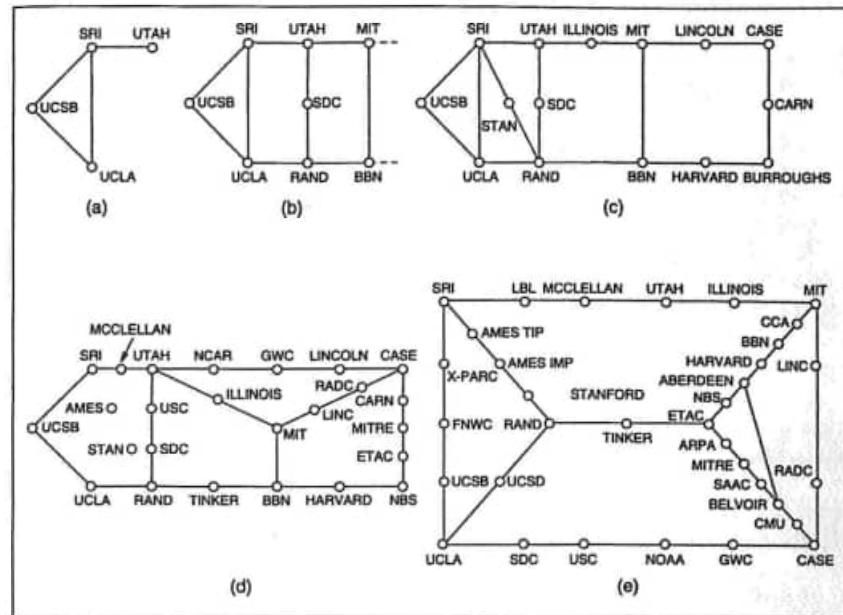


Fig. 1-25 Crescita di ARPANET. (a) Dicembre 1969. (b) Luglio 1970. (c) Marzo 1971. (d) Aprile 1972. (e) Settembre 1972.

viaggiava lungo le strade della California usò la rete radio per spedire messaggi all'SRI, messaggi che vennero fatti proseguire lungo ARPANET fino alla costa orientale, dove vennero poi spediti all'University College a Londra attraverso una rete satellitare. Questo permise a un ricercatore su di un camion di usare un calcolatore a Londra mentre viaggiava in California.

Questo esperimento mostrò soprattutto che i protocolli esistenti per ARPANET non erano adatti per essere eseguiti lungo reti diverse. Questa osservazione spinse verso ulteriori ricerche sui protocolli, culminanti con l'invenzione del modello e dei protocolli TCP/IP (Cerf, Kahn, 1974). TCP/IP fu progettato specificatamente per gestire comunicazione fra differenti reti, una funzione che divenne sempre più importante con il collegamento di nuove reti ad ARPANET.

Per incoraggiare l'adozione di questi nuovi protocolli, ARPA stipulò numerosi contratti con la BBN e con l'università di Berkeley in California per integrarli nel Berkeley UNIX. I ricercatori a Berkeley svilupparono un'ottima interfaccia fra programmi e rete (i socket) e scrissero molte applicazioni, programmi di utilità e gestione per facilitare l'utilizzo della rete.

La scelta del momento fu perfetta. Molte università avevano appena acquistato un secondo o un terzo calcolatore VAX e una rete locale per collegarsi a essi, ma non avevano nessun tipo di programma per la gestione della rete. Appena 4.2BSD fu disponibile, con il pro-

tocollo TCP/IP, i socket e tante utilità per la rete, il pacchetto completo venne adottato immediatamente. Inoltre, con TCP/IP era facile per le reti locali collegarsi ad ARPANET, e molti lo fecero.

Nel 1983, la ARPANET era ormai stabile e molto popolare, con oltre 200 IMP e centinaia di host. A questo punto, ARPA passò la gestione della rete alla agenzia di comunicazione della difesa (DCA – Defense Communication Agency), per renderla una rete operativa. La prima cosa che DCA fece fu di separare la porzione militare (circa 160 IMP di cui 110 negli Stati Uniti e 50 all'estero) in una sottorete separata, **MILNET**, con delle vie di accesso più restrittive fra MILNET e il resto della sottorete di ricerca.

Durante gli anni ottanta, altre reti, specialmente reti locali, vennero collegate ad ARPANET. Con l'incremento delle dimensioni, la ricerca degli host destinazione diventava sempre più costosa, così venne creato il **DNS (Domain Naming System** – sistema per la denominazione del dominio) per organizzare macchine in domini e mappare i nomi degli host in indirizzi IP. Da allora, DNS è diventato un sistema di gestione di una base di dati generalizzata e distribuita capace di memorizzare una varietà di informazioni legate alla denominazione. Lo studieremo in dettaglio nel capitolo 7.

Nel 1990, ARPANET fu superata da reti più recenti che essa stessa aveva creato, e venne quindi chiusa e smantellata, ma essa vive nei cuori e nelle menti dei ricercatori delle reti. MILNET continua invece a operare.

1.5.3 NSFNET

Alla fine degli anni settanta, la NSF (National Science Foundation – la fondazione per le scienze degli Stati Uniti) vide l'enorme impatto che ARPANET stava avendo sulla ricerca universitaria, permettendo agli scienziati di vari paesi di condividere dati e di collaborare a progetti di ricerca. Tuttavia, per far parte di ARPANET, un'università doveva avere un contratto di ricerca con il Dipartimento della difesa, cosa che in molte non avevano. Questa mancanza di accesso universale spinse l'NSF a installare una rete virtuale, **CSNET**, centralizzata su una singola macchina presso il BBN che supportava linee a cui collegarsi per connettersi ad ARPANET e ad altre reti. Utilizzando CSNET, i ricercatori accademici potevano effettuare una chiamata e lasciare messaggi di posta elettronica per altre persone che si sarebbero collegate in seguito. Il meccanismo era semplice ma funzionale.

Nel 1984 la NSF iniziò a progettare un successore di ARPANET ad alta velocità, che sarebbe stato aperto a tutti i gruppi accademici. Per avere qualcosa di concreto da cui partire, NSF decise di costruire una rete che facesse da spina dorsale per collegare i suoi sei centri di supercalcolatori, a San Diego, Boulder, Champaign, Pittsburgh, Ithaca e Princeton. A ogni supercalcolatore veniva assegnato un piccolo fratello, che consisteva in un microcalcolatore LSI-11 chiamato **fuzzball**. I fuzzball erano collegati con linee affittate da 56 kbps formanti la sottorete, la stessa tecnologia hardware utilizzata da ARPANET. La tecnologia per i programmi era invece diversa: i fuzzball utilizzavano TCP/IP sin dall'inizio, cosa che li rendeva la prima rete geografica basata su TCP/IP.

NSF finanziò anche alcune (quasi venti) reti regionali collegate alla spina dorsale per permettere agli utenti di migliaia di università, laboratori di ricerca, biblioteche e musei di accedere a qualunque supercalcolatore e di comunicare fra loro. Le rete completa,

inclusa la sua dorsale e le reti regionali, venne chiamata **NSFNET**. Essa era collegata ad ARPANET attraverso un collegamento fra un IMP e un fuzzball nella stanza delle macchine presso Carnegie-Mellon. La prima dorsale di NSFNET è illustrata in figura 1-26.

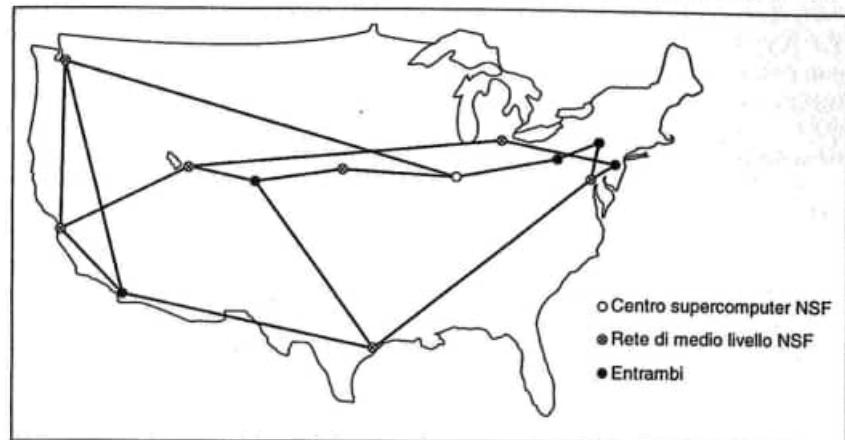


Fig. 1-26 La dorsale di NSFNET nel 1988.

NSFNET fu un successo istantaneo e si saturò immediatamente. NSF iniziò subito a progettare il suo successore e stipulò un contratto con il consorzio MERIT per realizzarlo. Canali a fibra ottica da 448 kbps vennero affittati dall'MCI per fornire la seconda versione della dorsale. Degli IBM RS6000 vennero utilizzati come router. Anche questa, si saturò, e negli anni novanta, la spina dorsale venne potenziata a 1,5 Mbps.

Assieme alla sua crescita, NSF intuì che il governo non avrebbe continuato a finanziare reti in eterno. Inoltre, molte organizzazioni commerciali desideravano collegarsi ma veniva loro impedito dallo statuto della NSF. Di conseguenza, NSF incoraggiò MERIT, MCI e IBM a formare una società senza scopo di lucro, **ANS (Advanced Networks and Services** – servizi e reti avanzate), come primo passo verso la commercializzazione. Nel 1990, ANS rilevò NSFNET e aggiornò i collegamenti da 1,5 Mbps a 45 Mbps per formare la **ANSNET**.

Nel dicembre del 1991, il congresso degli Stati Uniti emise un decreto che autorizzava **NREN (National Research and Educational Network** – la rete per l'educazione e la ricerca nazionale), la rete di ricerca del successore di NSFNET, che doveva raggiungere velocità dell'ordine dei gigabit. Lo scopo era quello di avere una rete nazionale che avesse una velocità di 3 Gbps prima del nuovo millennio. Questa rete è da realizzare come un prototipo della molto discussa autostrada dell'informazione.

Nel 1995, la dorsale di NSFNET non era più necessaria per collegare le reti regionali NSF perché numerose aziende già utilizzavano reti commerciali IP. Quando ANSNET venne venduta a "America Online" nel 1995, le reti regionali NSF dovettero uscire e acquistare servizi IP commerciali per potersi collegare.

Per facilitare la transizione e assicurarsi che ogni rete regionale potesse comunicare con ogni altra rete regionale, NSF offrì contratti a quattro diversi operatori di rete per stabilire un **NAP** (Network Access Point – punto di accesso alla rete). Questi operatori erano PacBell (San Francisco), Ameritech (Chicago), MFS (Washington, D.C.), e Sprint (New York City, dove per finalità NAP, Pennsauken, N.J. coincide con New York City). Ogni operatore di rete che desiderava fornire servizi di base alle reti regionali NSF doveva collegarsi a tutti i NAP. Questo meccanismo significava che un pacchetto originario di una qualsiasi rete regionale NSF poteva scegliere quale collegamento utilizzare per spostarsi dal proprio NAP a quello di destinazione. Di conseguenza le aziende che fornivano i servizi di collegamento erano forzate a competere per entrare in affari con le reti regionali sulla base della qualità dei propri servizi e dei propri prezzi. Oltre ai NAP di NSF, anche vari NAP governativi (per esempio FIX-E, FIX-W, MAE-East e MAE-West) e NAP commerciali (per esempio CIX) sono stati creati, così il concetto di un'unica dorsale venne rimpiazzato da infrastrutture in competizione secondo le leggi di mercato. Anche altre nazioni e regioni stanno costruendo reti simili a NSFNET. In Europa, per esempio, EBONE è una dorsale IP per organizzazioni di ricerca ed EuropaNET è invece una rete più orientata ad aspetti commerciali. Entrambe collegano numerose città in Europa con linee da 2 Mbps. Si stanno realizzando degli aggiornamenti a 34 Mbps. Ogni nazione in Europa ha una o più reti nazionali, che grosso modo sono paragonabili alle reti regionali NSF.

1.5.4 Internet

Il numero di reti, macchine e utenti collegati ad ARPANET crebbe rapidamente dopo che TCP/IP divenne l'unico protocollo ufficiale dall'1 gennaio 1983. Quando NSFNET e ARPANET vennero collegate, la crescita divenne esponenziale. Molte reti regionali si collegarono, e collegamenti vennero realizzati con il Canada, l'Europa e il Pacifico. A metà degli anni ottanta, si iniziò a vedere la collezione delle reti come una internetwork, e in seguito come Internet, anche se non ci fu mai una inaugurazione ufficiale con un uomo politico che rompesse una bottiglia di champagne contro un fuzzball. La crescita continuò in modo esponenziale, e nel 1990 Internet raggiunse 3.000 reti e 200.000 calcolatori. Nel 1992 venne collegato il milionesimo host. Nel 1995, vi erano molteplici dorsali, centinaia di reti di livello medio (per esempio regionali), decine di migliaia di reti locali, milioni di host e decine di milioni di utenti. La dimensione raddoppia approssimativamente ogni anno (Paxson, 1994).

La maggior parte delle crescita deriva dal collegamento a Internet di reti esistenti. Fra queste SPAN, rete per i fisici spaziali della NASA, HEPNET, una rete per fisici energetici, BITNET, rete dei supercalcolatori IBM, EARN, una rete accademica europea attualmente largamente utilizzata nell'Europa dell'Est, e tante altre. Vengono impiegati numerosi collegamenti transatlantici che viaggiano da 64 kbps a 2 Mbps.

La colla che tiene insieme l'intera Internet è il modello di riferimento TCP/IP e la relativa pila di protocolli. TCP/IP permette la realizzazione di servizi universali e può essere comparato al sistema telefonico o all'adozione dello scartamento normale decisa dalle ferrovie nel XIX secolo.

Cosa significa in realtà essere su Internet? La nostra definizione è che una macchina è su Internet se utilizza una pila di protocolli TCP/IP, ha un suo indirizzo IP, e ha la capacità

di spedire pacchetti IP a tutte le macchine su Internet. La sola capacità di spedire e ricevere posta elettronica non è sufficiente, in quanto la posta elettronica può essere diffusa anche da tante altre reti esterne a Internet. Tuttavia, la definizione è resa ancora più complicata dal fatto che molti personal computer hanno la possibilità di chiamare un fornitore di servizi Internet utilizzando un modem, di ottenere un indirizzo IP temporaneo e di spedire pacchetti IP ad altri host Internet. È sensato considerare tali macchine come presenti su Internet per il tempo in cui sono collegate al router che fornisce loro i servizi.

Con la sua crescita esponenziale, il vecchio modo informale di gestire Internet non funziona più. Nel gennaio del 1992 venne fondata la **Internet Society** (la società per la rete Internet), per promuovere l'uso di Internet e forse anche per controllarne la gestione. Tradizionalmente, la rete Internet aveva quattro applicazioni principali:

- 1. Posta elettronica.** La possibilità di comporre, spedire e ricevere posta elettronica c'era sin dai primi giorni di ARPANET e risulta tuttora estremamente popolare. Molte persone ricevono dozzine di messaggi al giorno e considerano la posta elettronica il loro modo primario per interagire con il mondo esterno, ben oltre il telefono e la lenta posta tradizionale. Oggigiorno, programmi per la gestione di posta elettronica sono disponibili più o meno per ogni tipo di calcolatore.
- 2. News.** I gruppi di discussione (newsgroup) sono forum specializzati in cui utenti che condividono un interesse possono scambiarsi messaggi. Esistono migliaia di gruppi di discussione, su argomenti tecnici e non, inclusi i calcolatori, le scienze, i passatempi e la politica. Ogni gruppo di discussione ha la propria etichetta, stile e abitudini, e sventurato sia chiunque li violi.
- 3. Collegamento remoto.** Utilizzando Telnet, Rlogin o altri programmi, qualsiasi utente di Internet può collegarsi su qualsiasi altra macchina su cui abbia un proprio account.
- 4. Trasferimento di archivi.** Utilizzando il programma FTP, è possibile copiare archivi da una macchina su Internet a un'altra. In questo modo è disponibile un enorme numero di articoli, banche di dati e altre informazioni.

Fino dall'inizio degli anni novanta, Internet era popolata principalmente dai ricercatori accademici, governativi e industriali. Una nuova applicazione, il **WWW** (World Wide Web – ragnatela attorno al mondo) cambiò tutto e richiamò alla rete milioni di nuovi utenti non accademici. Questa applicazione, inventata dal fisico del CERN Tim Berners-Lee, non cambiò nessuno dei servizi di base, ma li rese più semplici da utilizzare. Insieme al visualizzatore Mosaic, scritto al NCSA (National Center for Supercomputer Applications – centro nazionale per le applicazioni per supercalcolatori), il WWW rendeva possibile installare pagine di informazioni contenenti testo, figure, suoni e anche immagini in movimento, contenenti anche ulteriori collegamenti ad altre pagine. Premendo il pulsante del mouse su un iperpuntatore, l'utente viene immediatamente trasportato alla pagina denotata da tale puntatore. Per esempio, molte aziende possiedono una pagina con indicazioni alle pagine contenenti informazioni sui propri prodotti, listini prezzi, punti vendita,

supporto tecnico, comunicazioni agli impiegati, informazioni per i magazzinieri e altro ancora.

Tanti altri tipi di pagine sono stati definiti in breve tempo, incluse cartine, tabelle di mercato, cataloghi di libri, programmi radio registrati, e anche una pagina che punta al testo completo di libri i cui diritti d'autore sono scaduti (Mark Twain, Charles Dickens ecc.). Molte persone hanno anche una propria pagina personale (home page).

Nel primo anno successivo alla realizzazione di Mosaic, il numero di server WWW crebbe da 100 a 7000. Una crescita enorme continuerà senza dubbio anche nei prossimi anni, e probabilmente sarà la forza trainante per la tecnologia e l'utilizzo di Internet nel prossimo millennio.

Molti libri trattano Internet e i suoi protocolli. Per ulteriori informazioni, si vedano Black (1995); Carl-Mitchell, Quarterman (1993); Comer (1995); Santifallar (1994).

1.5.5 Reti sperimentali con velocità dell'ordine del gigabit

Le dorsali di Internet operano nell'ordine del megabit al secondo. Per chi vuole far avanzare lo sviluppo tecnologico, il prossimo passo saranno le reti con dorsali dell'ordine del gigabit. A ogni incremento delle prestazioni della rete, diventano realizzabili nuove applicazioni, e le reti dell'ordine del gigabit non sono una eccezione. In questo paragrafo presentiamo alcune applicazioni iniziali per tali reti, menzionandone due, e in seguito si elencano alcuni esempi di reti sperimentali che sono state realizzate.

Le reti dell'ordine del gigabit permettono una maggiore banda di comunicazione rispetto alle reti attuali, ma non sempre permettono ritardi inferiori. Per esempio, per spedire un pacchetto da 1 KB da New York a San Francisco a 1 Mbps si richiede 1 ms per l'invio dei dati e 20 ms per il ritardo necessario per attraversare il continente, per un totale di 21 ms. Con una rete da 1 Gbps si può ridurre questo ritardo a 20,001 ms. Mentre i bit hanno la possibilità di essere inviati più velocemente, il ritardo per attraversare il continente rimane lo stesso, in quanto la velocità della luce nelle fibre ottiche (o nei fili di rame) è vicina ai 200.000 km/s indipendentemente dalla velocità con cui vengono spediti i dati. Quindi per applicazioni geografiche in cui la velocità è un punto critico, utilizzare velocità superiori potrebbe non essere di grande aiuto. Fortunatamente, per certe applicazioni quello che conta è la banda passante, e queste sono le applicazioni per cui le reti dell'ordine del gigabit faranno una grossa differenza.

Un'applicazione è la telemedicina. In molti pensano che un modo per ridurre i costi medici sia quello di reintrodurre i medici di famiglia e le cliniche familiari su larga scala, così risulta più comodo l'accesso alle cure mediche di base. Nel caso in cui si presenti un problema medico più grave, il dottore di famiglia può richiedere esami di laboratorio o immagini mediche, come raggi X, TAC o risonanza elettromagnetica. I risultati degli esami e le immagini possono essere spediti elettronicamente a uno specialista che può fare una diagnosi. I medici oggi di solito si rifiutano di fare diagnosi utilizzando le immagini prodotte da un calcolatore. Questo comportamento continuerà almeno fino a quando la qualità delle immagini spedite elettronicamente non sarà buona quanto quella delle immagini originali. Questo significa immagini da $4K \times 4K$ pixel, con 8 bit per pixel (per immagini in bianco e nero) o 24 bit per pixel (per immagini a colori). Visto che molte analisi richiedono fino a 100 immagini (per esempio diverse viste in sezione dell'organo in questione),

un'unica serie per un paziente può generare 40 gigabit. Immagini in movimento (per esempio il battito cardiaco) generano ancora più dati. La compressione può aiutare, ma i medici sono titubanti a utilizzarla in quanto anche gli algoritmi più efficienti riducono la qualità delle immagini. Inoltre, tutte le immagini devono essere archiviate per anni per essere disponibili per eventuali urgenze mediche. Gli ospedali non desiderano riempirsi di calcolatori, così si rende necessario un archivio esterno e un meccanismo di ricerca elettronica ad alte prestazioni.

Un'altra applicazione consiste negli incontri virtuali. Ogni stanza di ritrovo contiene una telecamera sferica e una o più persone. Le sequenze di bit che arrivano da ogni telecamera vengono combinate elettronicamente per dare l'illusione che tutti si trovino nella stessa stanza. Ognuno vede queste immagini utilizzando occhiali per realtà virtuale. In questo modo l'incontro può avvenire senza viaggiare, ma anche in questo caso la velocità con cui vengono creati dati è stupefacente.

A partire dal 1989, ARPA e NSF unitamente si accordarono di finanziare un certo numero di reti sperimentali dell'ordine del gigabit gestiti da industrie e università, in seguito nell'ambito del progetto NREN. In alcune di queste, la velocità era di 622 Mbps, così solo se si sommano i dati che viaggiano in entrambe le direzioni si ottiene un gigabit. Questo tipo di gigabit viene spesso chiamato "gigabit governativo". (Alcuni cinici lo chiamano gigabit tassato.) Di seguito vengono brevemente menzionati i primi cinque progetti. Essi hanno completato il loro corso, ma meritano credito come pionieri, alla stessa stregua di ARPANET.

1. **Aurora** era una rete sperimentale che collegava quattro città del Nord-Est: MIT, l'università della Pennsylvania, il laboratorio T. J. Watson dell'IBM e Bellcore (Morristown, N.J.) a 622 Mbps utilizzando fibre ottiche fornite da MCI, Bell Atlantic e NYNEX. Aurora venne progettato principalmente per aiutare il test dei commutatori Sunshine della Bellcore e plaNET di proprietà dell'IBM utilizzando reti parallele. Gli aspetti di ricerca includevano la tecnologia di commutazione, protocolli che richiedessero velocità dell'ordine del gigabit, algoritmi per il controllo del trasporto dei dati sulla rete, controllo delle reti, memorie distribuite virtuali e collaborazioni utilizzando video-conferenze. Per ulteriori informazioni, si vedano Clark *et al.* (1993).
2. **Blanca** era un progetto di ricerca originariamente chiamato XUNET che coinvolgeva i laboratori AT&T Bell, Berkeley e l'università del Wisconsin. Nel 1990 vennero aggiunti alcuni nuovi siti (LBL, Cray Research e l'università dell'Illinois) e ottenne dei fondi NSF/ARPA. Alcuni collegamenti viaggiavano a 622 Mbps, ma altri erano più lenti. Blanca era l'unica rete nazionale, le altre erano regionali. Di conseguenza, la maggior parte della ricerca riguardava gli effetti dei ritardi dovuti alla velocità della luce. L'interesse era rivolto ai protocolli, specialmente protocolli per il controllo della rete, interfacce per gli host e applicazioni che richiedevano velocità di trasmissione dell'ordine del gigabit come immagini mediche, modellazioni meteorologiche e astronomiche. Per ulteriori informazioni, si vedano: Catlett (1992); Fraser (1993).
3. **CASA** aveva lo scopo di fare ricerca sulle applicazioni per supercalcolatori, specialmente quelle in cui parte del problema si adatta a un tipo di supercalcolatore (per

esempio supercalcolatori vettoriali Cray) e partì su un tipo diverso (ad es. un supercalcolatore parallelo). Le applicazioni analizzate riguardavano la geologia (utilizzando immagini Landsat), la modellazione del clima e la comprensione di reazioni chimiche. La rete operò in California e New Mexico e collegava Los Alamos, Cal Tech, JPL e il San Diego Supercomputer Center.

4. Nectar differiva dalle tre reti sperimentali già descritte in quanto essa era una rete metropolitana da gigabit che collegava CMU al Pittsburgh Supercomputer Center. I progettisti erano interessati ad applicazioni che coinvolgevano il flusso e le operazioni dei processi chimici, così come agli strumenti per poterli "testare".
5. VISTAnet era una piccola rete che operava a Research Triangle Park, North Carolina e collegava l'università di North Carolina, la North Carolina State University ed MCNM. In questo caso l'interesse era centrato su un prototipo per una rete a comunicazione pubblica che avesse dei commutatori con linee capaci di trasportare centinaia di gigabit, nel senso che i commutatori dovevano avere la capacità di elaborare un terabit per secondo. La ricerca scientifica si sviluppò utilizzando immagini tridimensionali per pianificare terapie che utilizzano radiazioni per pazienti affetti da cancro, con l'oncologo messo in grado di variare i parametri del fascio e istantaneamente vedere la quantità richiesta di radiazioni raggiungere il tumore e i tessuti sottostanti (Ransom, 1992).

1.6 Esempi di servizi per la comunicazione dei dati

Le società telefoniche hanno iniziato a offrire servizi di rete a ogni organizzazione che ne faccia richiesta. La sottorete è posseduta dall'operatore di rete, il quale fornisce servizi di comunicazione ai terminali e agli host clienti. Questo sistema viene chiamato **rete pubblica**. È analogo al sistema telefonico, e spesso ne è anche parte. Si è già presentato uno di questi nuovi servizi, DQDB, in figura 1-4. Nel paragrafo seguente si studieranno quattro esempi di altri servizi, SMDS, X.25, frame relay e broadband-ISDN.

1.6.1 SMDS – Switched Multimegabit Data Service

Il primo servizio che presentiamo, **SMDS** (**S**witched **M**ultimegabit **D**ata **S**ervice – servizio di comunicazione dati da multimegabit a commutazione), venne progettato per collegare fra loro molteplici reti locali, tipicamente i vari uffici o stabilimenti di una singola azienda. Venne progettata dalla Bellcore negli anni ottanta e sviluppata nei primi anni novanta con collegamenti regionali di breve distanza. Lo scopo era di produrre un servizio di comunicazione dati ad alta velocità che potesse essere installato con il minimo numero di problemi. SMDS rappresenta il primo servizio commutato a banda ampia (cioè ad alta velocità) offerto al pubblico.

Per vedere una situazione in cui SMDS potrebbe risultare utile, si consideri un'azienda con quattro uffici in quattro diverse città, ognuno con una propria rete locale. L'azienda potrebbe desiderare di collegare insieme le varie reti, in modo tale che i pacchetti siano in grado di muoversi da una rete all'altra. Una soluzione potrebbe essere di affittare sei linee ad alta velocità per poter collegare completamente le reti come mostrato in figura 1-27(a). Questa soluzione è certamente possibile, ma costosa.

1.6 Esempi di servizi per la comunicazione dei dati

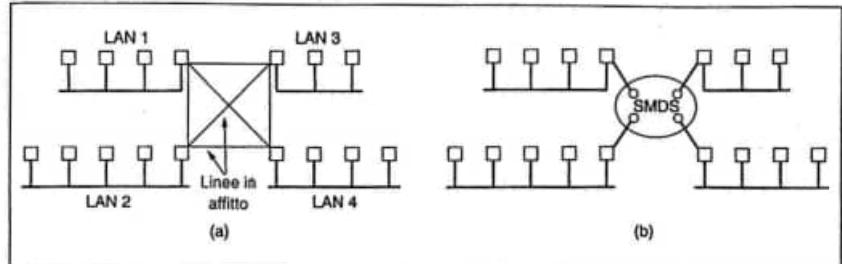


Fig. 1-27(a) Quattro reti locali connesse attraverso linee in affitto. (b) Collegamento utilizzando SMDS.

Una soluzione alternativa è di utilizzare SMDS, come in figura 1-27 (b). La rete SMDS agisce come una dorsale ad alta velocità, che permette ai pacchetti di una rete locale di raggiungere ogni altra rete. Tra le reti locali, negli uffici dei clienti, e la rete SMDS, negli uffici della società telefonica, c'è una (breve) linea di accesso affittata dalla società telefonica. Di solito questa linea è una MAN che utilizza DQDB, ma possono essere disponibili altre opzioni.

Mentre la maggior parte dei servizi delle società telefoniche è stata progettata per il traffico continuativo, SMDS è stato progettato per gestire traffico intermittente. In altre parole, di tanto in tanto ci sono pacchetti che devono essere trasportati da una rete locale all'altra velocemente, ma per la maggior parte del tempo non c'è traffico fra le varie reti locali. La soluzione con linee affittate di figura 1-27 (a) ha il problema di bollette mensili molto elevate; una volta installate, l'utilizzatore deve pagare le linee anche se non vengono utilizzate in modo continuativo. Nel caso di traffico intermittente, le linee affittate sono una soluzione costosa, e SMDS può economicamente competere con esse. Con n reti locali, una rete a linee affittate completamente connessa richiede $n(n-1)/2$ linee in qualche caso lunghe (quindi costose), mentre SMDS richiede solamente l'affitto di n brevi linee di accesso al router SMDS più vicino.

Visto che lo scopo di SMDS è di permettere alle LAN di comunicare fra loro, deve essere veloce a sufficienza per realizzare questo compito. La velocità è normalmente di 45 Mbps, anche se qualche volta velocità inferiori sono disponibili. Le MAN possono operare a 45 Mbps, ma esse non sono commutate, quindi per collegare quattro LAN utilizzando MAN, la società telefonica dovrebbe installare un singolo cavo da LAN 1 a LAN 2 a LAN 3 a LAN 4, cosa che risulta possibile solamente se le LAN sono situate nella medesima città. Con SMDS, ogni LAN si collega a un punto di connessione di una società telefonica che indirizza i pacchetti attraverso la rete SMDS nel modo richiesto per raggiungere la destinazione, probabilmente attraversando molteplici commutatori durante il cammino.

Il servizio base di SMDS è la semplice consegna di pacchetti utilizzando meccanismi privi di connessione. Il formato del pacchetto è mostrato in figura 1-28. Esso ha tre campi: la destinazione (dove deve andare il pacchetto), la sorgente (chi lo ha spedito), e uno spazio utile per i dati dell'utente di lunghezza variabile fino a un massimo di 9188 byte. La macchina sulla LAN mittente che è collegata alla linea di accesso mette il pacchetto su tale linea, e SMDS fa il possibile per consegnarlo alla corretta destinazione. Non vengono fornite garanzie.

Bytes	8	8	≤ 9188
Indirizzo di destinazione	Indirizzo della sorgente	Dati utente	

Fig. 1-28 Il formato del pacchetto SMDS.

Gli indirizzi di sorgente e destinazione consistono di un codice a 4 bit seguito da un numero di telefono di massimo 15 cifre decimali. Le singole cifre sono codificate in campi separati da 4 bit. I numeri di telefono contengono il prefisso della nazione, il prefisso locale, il numero dell'abbonato, in modo tale che il servizio possa essere fornito a livello internazionale. Si pensava che ponendo i numeri telefonici come numeri di indirizzi di rete, si sarebbe riusciti a rendere la nuova offerta più familiare anche agli utenti più scettici.

Quando un pacchetto arriva alla rete SMDS, il primo router controlla, per impedire frodi, che l'indirizzo sorgente corrisponda alla linea entrante. Se l'indirizzo non è corretto, il pacchetto viene scartato. Se è corretto, il pacchetto viene spedito verso la propria destinazione.

Una caratteristica utile di SMDS è il broadcast. L'utente può specificare una lista di numeri di telefono SMDS, e assegnare un numero speciale a tale lista. Ogni pacchetto inviato a quel numero è consegnato a tutti i membri della lista. La National Association of Securities Dealers utilizza questo servizio SMDS per diffondere i nuovi listini prezzi ai propri 5000 membri.

Un'ulteriore funzionalità è la visualizzazione degli indirizzi, sia per pacchetti entranti che uscenti. Con la visualizzazione per i pacchetti uscenti, è possibile definire una lista di numeri di telefono e specificare che nessun pacchetto venga indirizzato a numeri fuori da questa lista. Con la visualizzazione per i pacchetti in ingresso, è possibile accettare solamente i pacchetti provenienti da un insieme prestabilito di numeri di telefono. Quando entrambe le funzionalità vengono utilizzate, l'utente può effettivamente creare una rete privata priva di connessioni SMDS al mondo esterno. Per aziende con dati confidenziali, questa possibilità ha un grande valore.

Il carico utile del pacchetto può contenere una qualsiasi sequenza di byte desiderata dall'utente, fino a 9188 byte. SMDS non si interessa di tali dati. Può contenere un pacchetto Ethernet, un pacchetto IBM token ring, un pacchetto IP, o qualsiasi altra cosa. Qualsiasi cosa sia presente nel campo dati viene trasportata senza modifiche dalla rete sorgente a quella destinazione.

SMDS gestisce il traffico come segue. Il router collegato a ogni linea di accesso contiene un contatore che viene incrementato a intervalli costanti, per esempio ogni 10 µs. Quando un pacchetto arriva al router, si controlla se il contatore è più grande della lunghezza in byte del pacchetto. Se così è, il pacchetto viene spedito senza alcun ritardo e il contatore viene decrementato della lunghezza del pacchetto. Se la lunghezza del pacchetto è maggiore del contatore, il pacchetto viene scartato.

Concretamente, con un intervallo di incremento di 10 µs, l'utente può spedire a una velocità media di 100.000 byte al secondo, ma la velocità effettiva può essere superiore. Se, per esempio, la linea è rimasta inutilizzata per 10 ms, il contatore sarà 1.000, e

l'utente potrà spedire 1 kb alla massima velocità di 45 Mbps, così sarà trasmesso in circa 180 µs. Con una linea affittata da 100.000 byte al secondo, lo stesso kilobyte avrebbe richiesto 10 ms. Quindi SMDS offre brevi ritardi per blocchi di dati indipendenti spediti a intervalli ampi, nel caso in cui la quantità media rimanga inferiore al valore accordato. Questo meccanismo fornisce risposte veloci quando è necessario, evitando che gli utenti utilizzino più della banda passante per cui si sono accordati con la società telefonica.

1.6.2 Reti X.25

Molte reti pubbliche più antiche, specialmente fuori dagli Stati Uniti, seguono uno standard chiamato X.25. Esso venne sviluppato durante gli anni settanta dalla CCITT per fornire un'interfaccia fra le reti a commutazione di pacchetto e i relativi clienti.

Il protocollo del livello fisico, chiamato X.21, specifica l'interfaccia fisica, elettrica e procedurale fra l'host e la rete. Poche reti pubbliche attualmente supportano questo standard, perché esso richiede segnale digitale anziché analogico sulle linee telefoniche. Come soluzione provvisoria, è stata definita una interfaccia analogica simile al noto standard RS-232.

Lo standard a livello data link ha un certo numero di possibili varianti (a volte incompatibili). Esse sono progettate per gestire errori di trasmissione della linea telefonica fra le attrezzature dell'utente (host o terminale) e la rete pubblica (router).

Il protocollo del livello rete gestisce l'indirizzamento, il controllo del flusso, la conferma della consegna, le interruzioni e gli aspetti correlati. Fondamentalmente esso permette all'utente di stabilire circuiti virtuali a quindi spedire pacchetti contenenti fino a 128 byte di informazione. Tali pacchetti vengono consegnati in modo affidabile e ordinato. La maggior parte delle reti X.25 lavorano a velocità fino a 64 kbps, cosa che le rende obsolete per molte applicazioni. Ciò nonostante, esse sono ancora molto diffuse, così che il lettore potrebbe essere interessato alla loro esistenza.

X.25 è orientata alla connessione e supporta sia circuiti virtuali commutati che circuiti permanenti. Un **circuito virtuale commutato** è creato quando un calcolatore invia un pacchetto alla rete chiedendo di fare una chiamata a un calcolatore remoto. Una volta stabilita, i pacchetti possono essere inviati lungo la connessione, e arriveranno sempre in modo ordinato. X.25 fornisce controllo di flusso, per assicurarsi che un mittente più veloce non possa sovraccaricare un ricevente più lento o impegnato.

Un **circuito virtuale permanente** è utilizzato nello stesso modo di uno commutato, ma viene instaurato in modo anticipato attraverso un'intesa fra il cliente e colui che fornisce i collegamenti di rete. Esso è costantemente presente, e nessuna chiamata a procedure di installazione è richiesta per poterlo utilizzare. È analogo alle linee affittate.

Visto che il mondo è ancora pieno di terminali che non sono in grado di utilizzare lo standard X.25, un altro insieme di standard è stato definito per descrivere come un terminale ordinario (non intelligente) possa comunicare con una rete pubblica X.25. Concretamente, l'utente o l'operatore di rete installa una "scatola nera" a cui i terminali vengono collegati. La scatola nera è chiamata PAD (Packet Assembler Disassembler – assemblatore e disassemblatore di pacchetti), e le sue funzioni sono spiegate in un documento conosciuto come X.3. Un protocollo standard è stato definito fra il terminale e il

PAD, chiamato **X.28**; un altro protocollo standard esiste fra il PAD e la rete, chiamato **X.29**. Insieme, queste tre indicazioni sono chiamate **tripla X**.

1.6.3 Frame relay

Frame relay è un servizio destinato a chi desidera un meccanismo orientato alla connessione, privo di dorsale, per muovere bit da A a B a una velocità e un costo ragionevoli (Smith, 1993). La sua esistenza è dovuta a varianti tecnologiche sviluppate negli ultimi venti anni. Vent'anni fa, la comunicazione attraverso le linee telefoniche era lenta, analogica e inaffidabile, e i calcolatori erano costosi e lenti. Come risultato, per mascherare gli errori erano richiesti protocolli complessi, e i calcolatori per gli utenti erano troppo costosi per permettere loro di realizzare questi collegamenti affidabili.

La situazione è oggi completamente diversa. Linee telefoniche affittate sono ora veloci, digitali e affidabili, e i calcolatori sono veloci e non costosi. Questo suggerisce l'uso di protocolli semplici, con la maggior parte del lavoro che viene fatto dai calcolatori degli utenti, invece che dalla rete. È in questo ambito che si inserisce frame relay.

Un sistema frame relay può essere pensato come una linea virtuale affittata. Il cliente affitta un circuito virtuale permanente fra due punti e può quindi inviare fra di essi pacchetti lunghi fino a 1600 byte. È inoltre possibile affittare circuiti virtuali permanenti fra un certo sito e molti altri, in modo tale che i pacchetti contengano un numero da 10 bit che indichi quale circuito virtuale utilizzare.

La differenza fra una linea affittata concreta e una virtuale è che in quella concreta l'utente può spedire per tutto il giorno alla massima velocità. Con una linea virtuale, piccoli blocchi di dati possono essere spediti alla massima velocità, ma la velocità media di trasmissione di grossi quantità di dati sarà inferiore. Però, il costo di una linea virtuale è inferiore.

Oltre a competere con linee affittate, frame relay compete anche con circuiti permanenti virtuali X.25, con il risultato che opera a velocità superiori, normalmente 1,5 Mbps, e fornisce meno servizi.

Frame relay fornisce un servizio minimale, fondamentalmente la possibilità di determinare l'inizio e la fine di ogni pacchetto, e il riconoscimento di errori di trasmissione. Se viene ricevuto un pacchetto sbagliato, il servizio di frame relay lo scarta. È compito dell'utente scoprire se un pacchetto viene perso e agire di conseguenza per ripristinare la situazione. Diversamente da X.25, frame relay non fornisce informazioni di avvenuta ricezione o normale controllo del flusso. Esso ha un bit nell'intestazione, tuttavia, che una estremità della connessione può utilizzare per indicare all'altra che ci sono dei problemi. L'utilizzo di questo bit è a discrezione dell'utente.

1.6.4 Reti Broadband-ISDN e reti ATM

Man mano che i servizi esposti sopra diventano popolari, le società telefoniche hanno a che fare con un problema fondamentale: le reti multiple. POTS (Plain Old Telephone Service – ordinari vecchi servizi telefonici) e Telex utilizzano la vecchia rete a commutazione di circuito. Ognuno dei nuovi servizi per i dati come SMDS e frame relay utilizzano una specifica rete a commutazione di pacchetto. DQDB è diversa da questi, e la rete per la gestione delle chiamate interne della società telefonica (SSN 7) è un'altra rete

ancora. Il mantenimento di tutte queste reti separate è un rompicapo, e inoltre esiste una ulteriore rete, quella della televisione via cavo, che le società telefoniche tuttora non controllano anche se desidererebbero farlo.

La soluzione è di inventare una singola nuova rete per il futuro che rimpiazzi l'intero sistema telefonico e tutte le reti specializzate con una singola rete integrata per il trasferimento di tutti i tipi di informazione. Questa nuova rete avrà una grossa quantità di dati in confronto alle reti e ai servizi esistenti e renderà possibile l'offerta di una grande varietà di nuovi servizi. Questo non è un progetto piccolo, e di certo non si realizzerà nell'arco di una notte, ma è attualmente in esecuzione.

Il nuovo servizio capace di ricoprire una vasta area, è chiamato **B-ISDN** (Broadband Integrated Services Digital Network – rete digitale per servizi integrati ad ampia banda). Esso offrirà video a richiesta, televisione in diretta da molte sorgenti, posta elettronica multimediale in movimento, musica con la qualità dei CD, collegamento fra reti locali, trasporto di dati ad alta velocità per il mondo scientifico e industriale e tanti altri servizi che non sono stati ancora nemmeno pensati, tutti lungo le linee telefoniche.

Il sistema sottostante che rende possibile B-ISDN si chiama **ATM** (Asynchronous Transfer Mode) perché non è sincrono (vincolato a un orologio globale), così come molte linee telefoniche a lunga distanza. Si noti che l'acronimo ATM in questo caso non ha nulla a che fare con la "Automated Teller Machine" (macchina dispensatrice automatica – analoga al Bancomat, NdT) fornita da molte banche, anche se una macchina ATM può utilizzare una rete ATM per parlare con la propria banca.

Una grossa mole di lavoro è già stata fatta su ATM e sul sistema B-ISDN che lo utilizza, anche se ne occorre molto altro. Per ulteriori informazioni su questo argomento, si vedano: Fischer et al. (1994); Gasman (1994); Goralski (1995); Kim et al. (1994); Kyas (1995); McDysan, Spohn (1995); Stallings (1995a).

L'idea base di ATM è di trasmettere tutte le informazioni in piccoli pacchetti di dimensione fissa chiamati **celle**. Le celle sono lunghe 53 byte, di cui 5 byte di intestazione e 48 di carico utile, come in figura 1-29. ATM è sia una tecnologia (nascosta all'utente) che potenzialmente un servizio (visibile agli utenti). A volte il servizio viene chiamato **cell relay**, in analogia con frame relay.

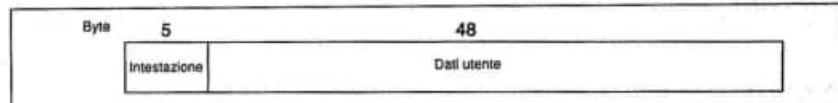


Fig. 1-29 Una cella ATM.

L'utilizzo di una tecnologia a commutazione di celle rappresenta una gigantesca rottura con la centenaria tradizione dei circuiti commutati (che stabiliscono un cammino di rame) per quanto riguarda il sistema telefonico. Esistono molte ragioni per cui è stata scelta la commutazione di celle, fra le quali ci sono le seguenti. Innanzitutto, la commutazione di celle è altamente flessibile e può gestire comodamente sia il traffico a velocità costante (audio, video) che il traffico a velocità variabile (dati). Secondo, alle alte velocità raggiunte (l'ordine del gigabit al secondo non è lontano), la commutazione digitale delle

celle è più facile rispetto all'utilizzo di tradizionali tecniche di commutazione, specialmente utilizzando fibre ottiche. Terzo, per la distribuzione telefonica, la diffusione completa è essenziale; la commutazione di celle può realizzarla mentre circuiti commutati non possono.

Le reti ATM sono orientate alla connessione. Realizzare una chiamata richiede prima la spedizione di un messaggio per instaurare la connessione. Dopo di che, le celle seguenti seguono tutte lo stesso cammino verso la destinazione. La consegna delle celle non è garantita, ma l'ordine sì. Se le celle 1 e 2 sono spedite in questo ordine, allora se entrambe arrivano, esse arriveranno nel medesimo ordine, mai verrà consegnata prima 2 e poi 1.

Le reti ATM sono organizzate come le tradizionali reti geografiche, con linee e commutatori (router). Le velocità delle reti ATM sono di 155 o 622 Mbps, con la possibilità più avanti di raggiungere velocità dell'ordine del gigabit al secondo. La velocità di 155 Mbps è stata scelta in quanto è la velocità richiesta per la trasmissione della televisione in alta definizione. La scelta esatta di 155,52 Mbps venne presa per compatibilità con il sistema di trasmissione SONET della AT&T. La velocità di 622 Mbps è stata scelta per permettere a quattro canali da 155 Mbps di essere trasmessi su di essa. Ora dovrebbe essere chiaro perché alcune delle reti sperimentali dell'ordine del gigabit operano a 622 Mbps: esse usano ATM.

Quando ATM venne proposto, la discussione riguardava la possibilità di fornire il servizio di video su richiesta e di sostituire il sistema telefonico, come descritto sopra. Da allora, altri sviluppi sono stati importanti. Molte organizzazioni hanno superato la larghezza di banda a propria disposizione e sono forzate a spostarsi verso un tipo di sistema commutato che fornisca una banda passante superiore a quella delle singole LAN. Anche nelle computazioni client-server alcune applicazioni richiedono l'abilità di comunicare ad alcuni server ad alte velocità. ATM è certamente uno dei candidati maggiori per entrambe le applicazioni. Ciò nonostante, è un po' più facile sostituire l'intero sistema telefonico analogico a bassa velocità avendo a disposizione un sistema digitale ad alta velocità, che non collegare tutte le Ethernet presenti. L'interconnessione di LAN utilizzando ATM viene discussa in Kavak (1995); Newman (1994); Truong *et al.* (1995).

È inoltre ovvio sottolineare che le diverse organizzazioni coinvolte nella realizzazione delle reti ATM hanno diversi interessi (finanziari). Le aziende che forniscono servizi telefonici hanno interesse a utilizzare ATM per sostituire le obsolete linee telefoniche in modo tale da competere con le società delle televisioni via cavo nella distribuzione elettronica dei video. I venditori di calcolatori vedono il campo ATM – LAN come la maggior fonte di guadagno (per loro stessi). Tutti questi interessi in competizione non rendono certo il processo di standardizzazione più veloce, più facile o più coerente. Anche i politici e coloro che detengono il potere all'interno delle organizzazioni che stanno definendo lo standard ATM (il Forum ATM) hanno una considerevole influenza su come evolverà ATM.

Il modello di riferimento ATM B-ISDN

Si ritorna ora alla tecnologia delle reti ATM, specialmente vista come tecnologia usata nel (futuro) sistema telefonico. Broadband-ISDN utilizza ATM come suo modello di riferimento, modello diverso da quello OSI e anche diverso dal TCP/IP. Tale modello è

mostrato in figura 1-30. Esso consiste di tre livelli: fisico, ATM e adattamento, più qualunque cosa che l'utente desideri inserirvi al di sopra.

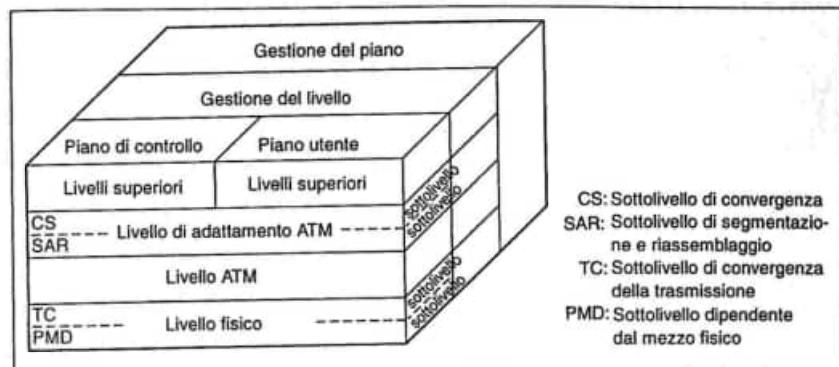


Fig. 1-30 Il modello di riferimento ATM B-ISDN.

Il livello fisico ha a che fare con il mezzo fisico: voltaggi, temporizzazione dei bit e altri aspetti. ATM non prescrive un particolare insieme di regole, ma indica invece che le celle ATM possono essere spedite su fili di rame o fibre ottiche per conto proprio, o possono anche essere impacchettate all'interno del carico utile di un pacchetto spedito secondo un qualche altro sistema. In altre parole, ATM è stato progettato per essere indipendente dal meccanismo di trasmissione.

Il livello ATM ha a che fare con celle e trasporto di celle. Esso definisce la composizione di una cella e indica il significato dei campi di intestazione. Esso ha a che fare anche con lo stabilimento e il rilascio dei circuiti virtuali. Il controllo della congestione è anch'esso situato in questo livello.

Visto che la maggior parte delle applicazioni non desidera lavorare direttamente con le celle (anche se potrebbe), è stato definito un livello al di sopra di ATM per permettere agli utenti di spedire pacchetti più lunghi delle singole celle. L'interfaccia ATM segmenta questi pacchetti, trasmette le celle individualmente e le riassembra dalla parte del ricevente. Questo livello è AAL (ATM Adaptation Layer – livello di adattamento ATM).

Diversamente dai modelli di riferimento bidimensionali, il modello ATM è tridimensionale, come mostra la figura 1-30. Il piano utente gestisce il trasporto dei dati, il controllo del flusso, la correzione degli errori e altre funzioni dell'utente. In contrasto, il piano del controllo ha a che fare con la gestione della connessione. Le funzioni di gestione del piano hanno a che fare con la gestione delle risorse e la coordinazione fra i vari livelli. I livelli fisico e AAL sono entrambi divisi in due sottolivelli, uno inferiore che realizza i lavori, e un sottolivello di convergenza al di sopra che fornisce la appropriata interfaccia al livello superiore. Le funzioni dei livelli e dei sottolivelli sono mostrate in figura 1-31. Il sottolivello PMD (Physical Medium Dependent – dipendente dal mezzo fisico) interfa il cavo vero e proprio. Esso muove i bit e gestisce la loro temporizzazione. Questo livello sarà differente per diversi cavi o sistemi di trasporto.

Livello OSI	Livello ATM	Sottolivello ATM	Funzionalità
3/4	AAL	CS	Fornisce l'interfaccia standard (convergenza)
		SAR	Segmentazione e riassemblaggio
2/3	ATM		Flusso di controllo Generazione / estrazione dell'intestazione delle celle Gestione del circuito / cammino virtuale Multiplexing / demultiplexing delle celle
2		TC	Disaccoppiamento della sequenza di celle Generazione e verifica del checksum dell'intestazione Generazione delle celle Imballaggio / disimballaggio delle celle dall'involucro Generazione dei frame
1	Fisico	PMD	Temporizzazione dei bit Accesso fisico alla rete

Fig. 1-31 I livelli e i sottolivelli ATM e le loro funzioni.

L'altro sottolivello del livello fisico è il sottolivello TC (Transmission Convergence – convergenza di trasmissione). Quando vengono trasmesse le celle, il livello TC le invia come stringhe di bit al livello PMD. È facile farlo: dalla parte del ricevente, il sottolivello TC riceve una sequenza pura di bit entranti dal sottolivello PMD. Il suo compito è di convertire questa sequenza di bit in una sequenza di celle per il livello ATM. Esso gestisce tutti gli aspetti riguardanti la definizione di dove le celle iniziano e finiscono nella sequenza di bit. Nel modello ATM, questa funzionalità è nel livello fisico. Nel modello OSI e quasi in tutte le altre reti, il compito di trasformare una sequenza grezza di bit in una sequenza di pacchetti o celle, è assegnato al livello data link. Per questa ragione in questo libro discuteremo questo problema assieme al livello data link, non con il livello fisico.

Come abbiamo detto in precedenza, il livello ATM gestisce celle, inclusa la loro generazione e il trasporto. Molti degli aspetti interessanti di ATM si trovano a questo livello. È una miscela dei livelli data link e rete del modello OSI, ma non viene diviso in sottolivelli.

Il livello AAL viene diviso in un sottolivello SAR (Segmentation And Reassembly – segmentazione e riassemblaggio) e in un sottolivello CS (Convergence Sublayer – sottolivello di convergenza). Il sottolivello più in basso rompe i pacchetti in celle dalla parte del mittente e li riunisce dalla parte del destinatario. Il livello superiore rende possibile ai sistemi ATM di offrire diversi tipi di servizi a differenti applicazioni (per esempio, il trasferimento di file e il servizio di video a richiesta hanno diversi requisiti riguardanti la gestione degli errori, la temporizzazione ecc.).

Prospettive di ATM

Da un certo punto di vista, ATM è un progetto inventato dalle industrie telefoniche in quanto dopo l'avvento di Ethernet, l'industria dei calcolatori non si interessava alla definizione di nuovi standard. Le società telefoniche colmarono questo vuoto con ATM, e nell'ottobre 1991, anche molti vendori di calcolatori si unirono alle società telefoniche per far nascere il **Forum ATM**, un comitato industriale che guiderà il futuro di ATM.

Anche se ATM promette di consegnare informazioni ovunque alla velocità di circa 1 Gbps, mantenere questa promessa non sarà facile. ATM è fondamentalmente commutazione di pacchetto ad alta velocità, una tecnologia con cui le società telefoniche hanno poca esperienza. Quello che esse hanno è un investimento immenso in una tecnologia differente (a commutazione di circuito) rimasta concettualmente intatta dai giorni di Alexander Graham Bell. È facile prevedere che questa transizione non avverrà velocemente, in quanto più che un'evoluzione è una rivoluzione, e le rivoluzioni non sono mai facili.

Deve essere considerato anche il fattore economico nell'installazione di ATM in tutto il mondo. Una frazione considerevole del sistema telefonico esistente deve essere sostituita. Chi pagherà? Quanto sarebbero disposti a pagare i clienti per avere elettronicamente un film su richiesta, possono averne una copia economica in videocassetta dal più vicino rivenditore? Infine, è cruciale la questione di dove deve essere fornita la maggior parte dei servizi avanzati. Se essi sono provvisti dalla rete, il profitto andrà alle società telefoniche. Se essi sono invece forniti dai calcolatori collegati alla rete, i profitti andranno ai costruttori e gli operatori di questi dispositivi. Tutto questo potrebbe non interessare all'utente, ma alle società telefoniche e ai rivenditori di calcolatori certamente sì, e questo sicuramente influisce anche sul loro interesse alla realizzazione di ATM.

1.6.5 Confronto fra i servizi

Il lettore può chiedersi perché esistano così tanti servizi incompatibili e in sovrapposizione, inclusi DBDQ, SMDS, X.25, frame relay, ATM e molti altri. La ragione di fondo è la decisione del 1984 di smantellare AT&T e incoraggiare la competizione nell'industria delle telecomunicazioni. Diverse aziende con diversi interessi e tecnologie sono ora libere di offrire qualsiasi servizio di cui pensano ci sia domanda e molte di esse lo stanno facendo in modo molto aggressivo.

Per ricapitolare alcuni dei servizi che si sono presentati in questo capitolo, DQDB è una tecnologia per rete metropolitana senza commutazione che permette a celle da 53 byte (di cui 44 utili) di essere spedite lungo cavi all'interno della medesima città. SMDS è una tecnologia a datagramma con commutazione per spedire datagrammi ovunque in una rete a 45 Mbps. X.25 è una più vecchia tecnologia di rete orientata alla connessione per la trasmissione di piccoli pacchetti di dimensione variabile a 64 kbps. Frame relay è un servizio che fornisce linee affittate virtuali a velocità attorno a 1,5 Mbps. Infine, ATM è progettata per sostituire l'intero sistema telefonico a commutazione di circuito con un sistema a commutazione di celle per essere in grado di gestire sia dati che segnali televisivi. Alcune differenze fra questi antagonisti vengono riassunte nella figura 1-32.

Caratteristica	DQDB	SMDS	X.25	Frame Relay	ATM AAL
Orientato alla connessione	Si	No	Si	Si	Si
Velocità normale	45	45	.064	1.5	155
Commutata	No	Si	Si	No	Si
Carico utile fisso	Si	No	No	No	No
Carico utile massimo	44	9188	128	1600	Variabile
VC permanenti	No	No	Si	Si	Si
Multicasting	No	Si	No	No	Si

Fig. 1-32 Diversi servizi di rete.

1.7 Standardizzazione delle reti

Esistono molti venditori e fornitori di reti, ciascuno con le proprie idee su come le cose dovrebbero essere fatte. Se non si raggiunge l'accordo, continuerà a regnare il caos. L'unica possibilità per uscirne è di accordarsi su degli standard per le reti.

La definizione di standard non solo permette a calcolatori differenti di comunicare, ma incrementa anche il mercato dei prodotti che aderiscono a un determinato standard, porta alla produzione di massa, a implementazioni VLSI e ad altri benefici che permettono di decrementare il prezzo e incrementare ulteriormente la richiesta. Nei paragrafi che seguono daremo uno sguardo veloce all'importante, ma poco conosciuto, mondo degli standard internazionali.

Gli standard si dividono in due categorie: "de facto" e "de jure". Gli standard **de facto** (in latino "di fatto") sono quelli che sono capitati senza nessun tipo di pianificazione. I PC dell'IBM e i loro successori sono degli standard de facto per calcolatori per piccoli uffici perché dozzine di costruttori hanno scelto di copiare le macchine IBM. UNIX è lo standard de facto per quanto riguarda i sistemi operativi nei dipartimenti di informatica universitari.

Gli standard **de jure** (in latino "per legge"), invece, sono standard formali, legali, adottati da una struttura autorizzata alla definizione di standard. Le autorità internazionali di standardizzazione sono generalmente divise in due classi: quelle stabilite con trattati fra governi nazionali, e organizzazioni spontanee. Nell'area degli standard per reti di calcolatori, ci sono parecchie organizzazioni di entrambi i tipi, e vengono discusse nel seguito.

1.7.1 Chi c'è nel mondo delle comunicazioni

Lo status legale delle società telefoniche varia considerevolmente da nazione a nazione. A un estremo ci sono gli Stati Uniti, che hanno 1500 diverse società telefoniche private. Prima di essere smantellata nel 1984, AT&T, a quei tempi la più grande società mondiale, dominava completamente la scena. Essa forniva servizi telefonici a quasi l'80% degli americani, coprendo più della metà dell'area geografica degli Stati Uniti, lasciando alle

1.7 Standardizzazione delle reti

altre società i rimanenti clienti (per la maggior parte disseminati in aree rurali). Dalla sua chiusura, AT&T continua comunque a fornire servizi a lunga distanza, anche se ora si trova in competizione con altre aziende. Le sette Regional Bell Operating Companies nate da AT&T e 1500 aziende indipendenti forniscono servizi telefonici locali e cellulari. Alcune di queste aziende indipendenti, come la GTE, sono molto grandi.

Le aziende che forniscono servizi di comunicazione negli Stati Uniti sono chiamate **aziende comuni**. Le loro offerte e i loro prezzi sono descritti da un documento, detto **tariffario**, che deve essere approvato dalla Commissione federale per le comunicazioni interne e internazionali, e dalle commissioni pubbliche per il traffico interno.

All'altro estremo ci sono nazioni in cui il governo nazionale detiene il monopolio completo su tutte le comunicazioni, incluse poste, telegrafi, telefoni e spesso anche radio e televisioni. La maggior parte delle nazioni del mondo rientra in questa categoria. In alcuni casi l'autorità di gestione delle telecomunicazioni è una azienda statale, mentre in altre è una branca governativa, normalmente conosciuta come **PTT** (amministrazione delle **poste, telegrafi e telefoni**). In tutto il mondo la tendenza è verso la liberalizzazione e la competizione al di fuori del monopolio del governo.

Con tutti questi servizi e fornitori differenti, c'è una necessità chiara di fornire compatibilità su scala mondiale per assicurare alle persone (e ai calcolatori) di una nazione di poter chiamare e comunicare con la relativa controparte in un'altra nazione. Effettivamente, questa necessità è esistita per un lungo tempo. Nel 1865, si incontrarono i rappresentanti di molti governi europei per formare il predecessore dell'attuale **ITU** (**International Telecommunication Union** – unione per le telecomunicazioni internazionali). Il compito dell'ITU fu di standardizzare le telecomunicazioni internazionali, che a quei tempi erano rappresentate dal telegrafo. Anche allora fu chiaro che sarebbero nati dei problemi se metà delle nazioni avesse usato il codice Morse e l'altra metà un altro codice. Quando il telefono divenne un servizio internazionale, ITU si accollò il compito di standardizzare anche la telefonia. Nel 1947, ITU divenne una agenzia delle Nazioni Unite. ITU include tre settori principali:

1. Settore radiocomunicazioni (ITU-R).
2. Settore di standardizzazione delle telecomunicazioni (ITU-T).
3. Settore sviluppo (ITU-D).

ITU-R riguarda l'allocazione delle frequenze radio mondiali. Noi ci interesseremo principalmente dell'ITU-T, che riguarda la telefonia e i sistemi di comunicazione dei dati. Dal 1956 al 1993, ITU-T fu nota come **CCITT**, un acronimo francese: "Comité Consultatif International Télégraphique et Téléphonique" (comitato internazionale di consulenza telegrafica e telefonica). L'1 marzo del 1993, CCITT venne riorganizzato per svincolarlo dalla burocrazia e venne ridenominato per rispecchiare il suo nuovo ruolo. Sia ITU-T che CCITT davano indicazioni nell'area della telefonia e delle telecomunicazioni. Alcune di queste che vennero proposte dal CCITT come il CCITT X.25, dopo il 1993 passarono sotto l'etichetta ITU-T.

ITU-T ha cinque classi di membri:

1. Amministrazioni (le PTT nazionali).
2. Operatori privati riconosciuti (ad es. AT&T, MCI, British Telecom).
3. Organizzazioni regionali di telecomunicazione (ad es. la European ETSI).
4. Industrie di telecomunicazioni e organizzazioni scientifiche.
5. Altre organizzazioni interessate (ad es. istituti bancari o compagnie aeree).

ITU-T comprende approssimativamente 200 amministrazioni, 100 operatori privati e diverse centinaia di altri membri. Solo le amministrazioni hanno il diritto di voto, ma tutti i membri possono partecipare ai lavori dell'ITU-T. Visto che gli Stati Uniti non hanno un PTT, qualcun altro li deve rappresentare presso l'ITU-T. Questo compito spetta al Dipartimento di Stato, in quanto ITU-T ha a che fare con nazioni straniere, che è la specialità del Dipartimento di Stato.

Il compito di ITU-T è di dare indicazioni a proposito di telefonia, telegrafi e telecomunicazioni. Queste divengono spesso standard internazionali riconosciuti, per esempio, V.24 (conosciuto anche come EIA RS-232 negli Stati Uniti), che specifica posizione e significato dei collegamenti all'interno dei connettori usati nella maggior parte dei terminali asincroni.

Ricordiamo che le indicazioni ITU-T sono tecnicamente solo proposte che i governi possono adottare o ignorare, a loro piacimento. In pratica, una nazione che desidera adottare uno standard telefonico diverso dal resto del mondo è libera di farlo, ma al prezzo di autoescludersi da qualsiasi altro. Questo forse si potrebbe accettare per l'Albania, ma in altri casi potrebbe generare grossi problemi. L'abitudine di chiamare gli standard ITU-T "indicazioni" era ed è tuttora necessaria per placare le spinte nazionalistiche.

Il vero compito di ITU-T viene realizzato in *gruppi di studio*, spesso allargati fino a 400 persone. Per riuscire a lavorare, i gruppi di studio si dividono in *gruppi di lavoro*, che vengono a loro volta divisi in *gruppi esperti*, che sono ulteriormente divisi in gruppi specifici. Una volta c'era molta burocrazia, adesso pure.

Nonostante tutto questo, ITU-T riesce a lavorare davvero. In media ogni anno sforna circa 5000 pagine di indicazioni. I membri offrono i fondi per coprire i costi di gestione di ITU-T. Le nazioni ricche e di grosse dimensioni pagano fino a 30 unità di contribuzione; le nazioni piccole e povere se la possono cavare con 1/16 di una unità di contribuzione (tale unità è di circa 250.000 dollari). Anche se tale contributo è completamente volontario, questo testimonia il valore di ITU-T, visto che quasi tutti pagano per il suo sostentamento.

Quando le telecomunicazioni completeranno la trasformazione iniziata negli anni ottanta diventando internazionali da completamente nazionali, gli standard diventeranno ancora più importanti, e ulteriori organizzazioni desidereranno essere coinvolte in questa trasformazione. Per ulteriori informazioni a proposito di ITU, si veda Irmer (1994).

1.7.2 Chi c'è nel mondo degli standard internazionali

Gli standard internazionali sono prodotti da ISO (**International Standards Organization** – organizzazione per gli standard internazionali), una organizzazione su base volontaria fondata nel 1946. I suoi membri sono le organizzazioni degli standard nazionali degli 89 paesi membri. Questi membri includono ANSI (Stati Uniti), BSI (Gran Bretagna), AFNOR (Francia), DIN (Germania), più altri 85.

ISO emette standard su un numero enorme di soggetti, che variano dai dadi e bulloni (letteralmente) ai rivestimenti dei pali telefonici. Sono stati definiti oltre 5000 standard, incluso l'OSI. ISO ha quasi 200 comitati tecnici (TC), numerati nell'ordine della loro creazione. Ciascuno tratta un soggetto specifico: TC1 tratta di dadi e bulloni (standardizzandone le filettature); TC97 ha a che fare con calcolatori ed elaborazione dell'informazione. Ciascun comitato tecnico ha dei sottocomitati (SC) a loro volta divisi in gruppi di lavoro (WG).

Il lavoro vero è fatto principalmente nei gruppi di lavoro da oltre 100.000 volontari di tutto il mondo. Molti di questi "volontari" sono costretti a lavorare sugli standard ISO dai propri datori di lavoro, i cui prodotti sono stati standardizzati. Altri sono rappresentanti ufficiali di governo interessati a imporre come standard internazionale il modo di operare del proprio paese. Anche esperti accademici sono attivi all'interno dei gruppi di lavoro. Su problemi di standard per le telecomunicazioni, ISO e ITU-T spesso cooperano (ISO è un membro di ITU-T) per evitare l'assurdità di avere due standard ufficiali e mutualmente incompatibili.

Il rappresentante degli Stati Uniti all'interno di ISO è **ANSI (American National Standards Institute** – istituto americano per gli standard nazionali), che malgrado il suo nome, è un organo privato, non governativo e non avente fini di lucro. I suoi membri sono fabbricanti, aziende e altri parti interessate. Gli standard ANSI sono adottati frequentemente da ISO come standard internazionali.

La procedura usata da ISO per adottare standard è progettata per raggiungere il più largo consenso possibile. Il processo comincia quando una delle organizzazioni nazionali sente il bisogno di uno standard internazionale in una qualche area. Un gruppo di lavoro è quindi formato per realizzare un **CD (Committee Draft** – bozza del comitato). Il CD viene fatto circolare fra tutti i membri, che hanno sei mesi per analizzarlo. Se una maggioranza sostanziale lo approva, un documento riveduto, chiamato **DIS (Draft International Standard** – bozza di standard internazionale) è prodotto e viene fatto circolare per ricevere commenti e votazioni. A seconda dei risultati di questo turno, il testo finale dell'**IS (International Standard** – standard internazionale) viene preparato, approvato e pubblicato. In aree fortemente controverse, un CD o un DIS possono attraversare diverse versioni prima di acquisire abbastanza voti, e l'intero processo può richiedere anni.

Il **NIST (National Institute of Standards and Technology** – Istituto nazionale degli standard e della tecnologia) è un'agenzia del Dipartimento del commercio americano. Esso era inizialmente conosciuto come il **National Bureau of Standards**. Esso emette standard che sono obbligatori per acquisti fatti dal governo degli Stati Uniti, escluso il Dipartimento della difesa, che ha i propri standard.

Un altro ente che agisce nel mondo degli standard è **IEEE (Institute of Electrical and Electronics Engineers** – Istituto di ingegneria elettrica ed elettronica), la più grande organizzazione professionale nel mondo. Oltre alla pubblicazione di riviste e l'organizzazione di numerose conferenze ogni anno, IEEE ha un gruppo di standardizzazione nell'area dell'ingegneria elettrica e del calcolo. Lo standard IEEE 802 per reti locali è lo standard chiave per le LAN. È stato successivamente preso da ISO come la base per ISO 8802.

1.7.3 Chi c'è nel mondo degli standard per Internet

L'Internet mondiale ha i suoi meccanismi di standardizzazione, molto diversi da quelli di ITU-T e ISO. La differenza può essere grosso modo riassunta dicendo che coloro che provengono dagli incontri per la definizione degli standard ITU o ISO indossano completi eleganti. Coloro che provengono dalle riunioni per gli standard di Internet indossano jeans o uniformi militari.

Le riunioni ITU-T e ISO sono frequentate da diplomatici per i quali la standardizzazione è un lavoro. Essi vedono la standardizzazione come una cosa buona e dedicano a essa le loro vite. Coloro che utilizzano Internet, d'altra parte, preferiscono decisamente l'anarchia come principio di base, ma qualche volta gli accordi sono necessari. Così gli standard, anche se con dispiacere, sono di quando in quando necessari.

Quando ARPANET venne installata, il Dipartimento della difesa creò un comitato informale per supervisionarla. Nel 1983, il comitato venne ridevominato **IAB (Internet Activities Board** – comitato per le attività di Internet) e gli venne affidata la missione di tenere uniti, si fa per dire, i ricercatori coinvolti in ARPANET e Internet, un'attività non dissimile dall'accalappiacani. Il significato dell'acronimo IAB venne più tardi cambiato in **Internet Architecture Board** (comitato per l'architettura di Internet).

Ciascuno dei circa dieci membri dello IAB capeggiava un gruppo di lavoro su problemi aventi una certa importanza. Lo IAB impiegava a volte anni per discutere e comunicare i propri risultati al Dipartimento della difesa o all'NSF, che a quei tempi fornivano la maggior parte dei fondi. Quando veniva richiesto uno standard (ad es. un nuovo algoritmo di instradamento), i membri dello IAB lo progettavano e fornivano le specifiche agli studenti di dottorato, che scrivevano effettivamente i programmi. Le specifiche venivano presentate attraverso una serie di rapporti tecnici chiamati **RFC (Request For Comments** – richiesta di commenti). Gli RFC sono memorizzati in linea e possono essere recuperati da chiunque sia interessato. Sono numerati in ordine cronologico di creazione. Ne esistono quasi 2000.

Nel 1989, Internet era divenuta così grande che questo modo di lavorare estremamente informale non andava più bene. Molti vendori da tempo offrivano prodotti TCP/IP e non volevano cambiarli solo perché dieci ricercatori avevano pensato a qualcosa di migliore. Nell'estate del 1989, lo IAB venne nuovamente riorganizzato. I ricercatori vennero spostati all'**IRTF (Internet Research Task Force** – gruppo di ricerca su Internet), che era un nuovo organo dello IAB insieme con l'**IETF (Internet Engineering Task Force** – gruppo per l'ingegnerizzazione di Internet). L'IAB venne ripopolato con persone che rappresentavano una serie più ampia di organizzazioni rispetto alla comunità scientifica. Era inizialmente un gruppo costituito da membri con cariche della durata di due anni, che nominavano alla scadenza i propri successori. Più tardi, venne creata l'**Internet Society**, e venne popolata da gente interessata a Internet. L'Internet Society è in un certo senso comparabile ad ACM o IEEE. È governata da amministratori nominati dai membri dello IAB.

L'idea alla base di questa divisione era che l'IRTF si concentrasse sulla ricerca a lungo termine, mentre l'IETF avrebbe gestito i problemi dell'ingegnerizzazione a breve termine. L'IETF fu diviso in gruppi di lavoro, ciascuno con un problema specifico da risolvere. I presidenti di questi gruppi inizialmente si incontrarono in un comitato di governo per

1.8 Struttura del seguito del libro

dirigere lo sforzo di ingegnerizzazione. I temi discussi includevano nuove applicazioni, informazioni all'utente, integrazione OSI, instradamento e indirizzamento, sicurezza, gestione della rete e standardizzazione. In seguito si formarono così tanti gruppi di lavoro (più di 70) da dover essere raggruppati in aree, e i presidenti di ogni area costituirono singoli comitati di governo.

Inoltre, venne adottato un processo formale di standardizzazione, modellato su ISO. Per diventare uno **standard proposto**, l'idea di base deve essere spiegata completamente in un RFC e generare sufficiente interesse nella comunità per essere preso in considerazione. Per passare alla posizione di **standard preliminare**, ci deve essere una implementazione funzionante che viene esaminata completamente da almeno due siti indipendenti per quattro mesi. Se lo IAB è convinto che l'idea sia corretta e che il programma funzioni, l'RFC viene dichiarato **standard di Internet**. Alcuni standard di Internet sono diventati standard anche per il Dipartimento della difesa (MIL-STD), condizione obbligatoria per i fornitori del Dipartimento della difesa. David Clark una volta ha fatto un commento ora famoso a proposito degli standard di Internet, i quali sono composti da "un po' di consenso e da programmi funzionanti".

1.8 Struttura del seguito del libro

Questo libro discute i principi sia teorici che pratici legati alle reti di calcolatori. La maggior parte dei capitoli inizia con una discussione dei principi rilevanti, seguita da una serie di esempi che illustrano questi principi. Due reti sono usate come esempi portanti in tutto il testo: l'Internet e le reti ATM. Da un certo punto di vista le due reti sono complementari: ATM è interessata soprattutto ai livelli più bassi, mentre Internet ai livelli superiori. Nel futuro, Internet potrà viaggiare su una dorsale ATM, in modo tale che le due reti coesistano. Altri esempi verranno presentati ove necessario.

Il libro è strutturato secondo il modello ibrido di figura 1-21. A partire dal capitolo 2, analizziamo la gerarchia di protocolli iniziando dal basso. Il secondo capitolo fornisce alcune informazioni fondamentali nel campo della comunicazione dei dati. Si parla di trasmissione analogica e digitale, della commutazione e del passato, presente e futuro sistema telefonico. Questo materiale è legato al livello fisico, benché si coprano unicamente gli aspetti architettonici piuttosto che gli aspetti hardware. Vengono discussi diversi esempi riguardanti il livello fisico, come SONET o il sistema radiocellulare.

Il capitolo 3 discute il livello data link e i suoi protocolli attraverso parecchi esempi sempre più complessi. Viene presentata anche l'analisi di questi protocolli. Nel seguito vengono discussi alcuni importanti protocolli del mondo reale, inclusi HDLC (usato in reti a bassa o media velocità), SLIP, e PPP (usati nell'Internet), e ATM (usato in B-ISDN).

Il capitolo 4 riguarda il sottolivello di accesso al mezzo fisico, che è parte del livello data link. La questione principale che trattiamo è come determinare chi potrà usare la rete nel caso in cui essa consista di un singolo canale condiviso, come avviene nella maggior parte delle LAN e in alcune reti satellitari. Molti esempi vengono dalle LAN, dalle reti in fibre ottiche e dalle reti satellitari. Vengono discussi anche i meccanismi per collegare fra loro LAN indipendenti.

Il capitolo 5 tratta il livello rete, specialmente il problema dell'instradamento, del con-

trollo della congestione e della connessione di reti eterogenee. Si discutono gli algoritmi di instradamento sia statici che dinamici. Viene discusso anche il broadcast. Gli effetti di instradamento e la congestione, vengono discussi in dettaglio. Connettere reti eterogenee per formare una unica inter-rete genera numerosi problemi che sono discussi in tale capitolo. Il livello rete e le reti ATM vengono studiate in dettaglio.

Il capitolo 6 descrive il livello trasporto. L'enfasi è sui protocolli orientati alla connessione, visto che molte applicazioni hanno bisogno di questo tipo di connessione. Discutiamo approfonditamente un esempio di servizio del livello trasporto e della sua realizzazione. Vengono descritti i protocolli del livello trasporto sia di Internet (TCP e UDP) che delle reti ATM (AAL 1-5).

I livelli sessione e presentazione del modello OSI non vengono presentati in questo libro in quanto spesso non vengono assolutamente utilizzati.

Il capitolo 7 tratta il livello delle applicazioni, i suoi protocolli e le applicazioni vere e proprie. In questo capitolo analizziamo la sicurezza, le tecniche di indirizzamento, la posta elettronica, i gruppi di discussione, la gestione della rete, il World Wide Web e le applicazioni multimediali.

Il capitolo 8 contiene un elenco di letture suggerite per approfondire gli argomenti trattati nei vari capitoli. Il suo scopo è di aiutare il lettore che desidera proseguire i propri studi riguardanti le reti. Il capitolo contiene anche una bibliografia ordinata in ordine alfabetico di tutti i riferimenti citati in questo libro.

1.9 Riassunto

Le reti di calcolatori possono essere usate per numerosi servizi, sia per aziende che per individui. Per le aziende, le reti di personal computer che usano serventi condivisi spesso forniscono flessibilità e un buon rapporto prezzo/prestazioni. Per gli individui le reti offrono accesso a una varietà di informazioni e risorse di intrattenimento.

Parlando in modo approssimativo, le reti possono essere divise in LAN, MAN, WAN, e inter-reti, ciascuna con le proprie caratteristiche, proprie tecnologie, proprie velocità e proprie estensioni. Le LAN coprono un edificio, le MAN coprono una città e le WAN coprono una nazione o un continente. Le LAN e le MAN sono reti prive di commutazione (cioè non ci sono router); le WAN sono commutate.

I programmi per le reti costituiscono i protocolli, che definiscono le regole secondo le quali i processi possono comunicare. I protocolli possono essere sia orientati alla connessione che privi di connessione. La maggior parte delle reti supporta gerarchie di protocolli, in cui ciascun livello fornisce servizi ai livelli superiori isolandoli dai dettagli dei protocolli utilizzati dai livelli inferiori. Le pile di protocolli sono normalmente basate sul modello OSI o il modello TCP/IP. Entrambi i modelli hanno i livelli rete, trasporto e applicazione, ma differiscono per gli altri livelli.

Le reti più note includono la Novell NetWare, l'ARPANET (ora defunta), l'NSFNET, l'Internet, e diverse reti sperimentali con velocità dell'ordine del gigabit. I servizi forniti dalle reti includono DQDB, SMDS, X.25, frame relay e B-ISDN. Tutte queste reti e questi servizi sono disponibili sul mercato, offerti da una varietà di fornitori. Sarà il mercato a determinare quali sopravviveranno e quali no.

1.9 Riassunto

Esercizi

- Nel futuro, quando tutti avranno un terminale a casa propria collegato a una rete di calcolatori, saranno possibili referendum pubblici in tempo reale su importanti problemi legislativi. Certi meccanismi legislativi potranno essere eliminati, per permettere l'espressione diretta della volontà popolare. Gli aspetti positivi di una tale democrazia diretta sono abbastanza ovvi; si discutano alcuni degli aspetti negativi.
- Un'alternativa a una LAN è un sistema centralizzato condiviso con un terminale per ogni utente. Si presentino due vantaggi di un sistema client-server basato su una LAN.
- Cinque router devono essere collegati in una sottorete punto-a-punto. Tra ciascuna coppia di router, i progettisti possono mettere una linea ad alta velocità, una a media velocità, una a bassa velocità o nessuna linea. Se sono richiesti 100 millisecondi di tempo di calcolatore per generare e ispezionare ogni topologia, quanto tempo richiede ispezionare tutte le topologie possibili per trovare quella migliore rispetto al carico previsto?
- Un gruppo di $2^n - 1$ router è collegato attraverso un albero binario centralizzato, a un router su ogni nodo dell'albero. Il router i comunica con il router j spedendo un messaggio alla radice dell'albero. La radice quindi spedisce il messaggio verso il router j . Si deduca un'indicazione approssimata del numero medio atteso di passaggi per la consegna di un messaggio supponendo equiprobabile ogni coppia di router.
- Uno svantaggio del broadcast su una sottorete è la possibilità che diversi host cerchino di accedere al canale condiviso contemporaneamente. Per semplificare le cose, si supponga che il tempo venga diviso in intervalli discreti, con ciascuno degli n host che tentano di usare il canale con probabilità p durante ciascun intervallo. Quanti intervalli con collisioni si attendono?
- A cosa corrispondono gli indirizzi SAP nella diffusione radio in FM?
- Qual è la differenza principale tra comunicazione orientata e non orientata alla connessione?
- Due reti forniscono entrambe un servizio affidabile orientato alla connessione. Una offre una sequenza di byte affidabile mentre l'altra fornisce una sequenza di messaggi. Questi servizi sono equivalenti? Se non lo sono, fornire un esempio in cui essi differiscono.
- Qual è la differenza tra un servizio confermato e uno non confermato? Per ciascuno dei seguenti servizi, dire se può essere confermato, non confermato, entrambe le cose o nessuna delle due.
 - Stabilimento del collegamento.
 - Trasmissione dei dati.
 - Rilascio del collegamento.

10. Cosa significa "negoziare" quando si discute di protocolli di rete? Si fornisca un esempio.
11. Quali sono due possibili ragioni per usare protocolli a livelli?
12. Si elenchino due aspetti in comune fra il modello di riferimento OSI e il modello di riferimento TCP/IP. Si elenchino anche due aspetti che li differenziano.
13. Il presidente della Società delle Vernici Speciali ha l'idea di lavorare con un fabbricante locale di birra allo scopo di produrre un barattolo di birra invisibile (un rimedio al problema dei rifiuti). Il presidente chiede ai propri dirigenti di investigare, e a sua volta il problema passa in mano al capo degli ingegneri, che chiama il suo pari presso l'altra società per discutere degli aspetti tecnici del progetto. Gli ingegneri forniscono un rapporto ai rispettivi dirigenti che si accordano telefonicamente con i dirigenti dell'altra azienda rispetto agli aspetti legali. Infine i due presidenti discutono il lato finanziario della collaborazione. È questo un esempio di un protocollo multilivello nel senso del modello OSI?
14. Nella maggior parte delle reti, il livello data link gestisce gli errori di trasmissione richiedendo una nuova spedizione dei pacchetti danneggiati. Se la probabilità che un pacchetto venga danneggiato è p , qual è il numero medio di trasmissioni richieste per spedire un pacchetto supponendo che i messaggi di ack non vengano persi?
15. Quali livelli OSI hanno a che fare con i seguenti aspetti:
 - (a) Divisione della sequenza di bit in pacchetti.
 - (b) Determinazione del cammino da seguire lungo la rete.
16. TPDU incapsula pacchetti o utilizza altri meccanismi? Si discuta questo aspetto.
17. Un sistema ha una gerarchia di protocolli a n livelli. Le applicazioni generano messaggi lunghi M byte. Ogni livello aggiunge una intestazione di h byte. Quale frazione della banda passante della rete è occupata dalle intestazioni?
18. Qual è la differenza principale tra TCP e UDP?
19. L'architettura della Novell NetWare assomiglia di più a X.25 o a Internet? Si giustifichi la risposta.
20. L'Internet raddoppia quasi ogni 18 mesi. Benché nessuno realmente ne sia sicuro, si valuta il numero di host presenti nel gennaio 1996 pari a 7.000.000. Si usi questo dato per calcolare il numero atteso di host su Internet nell'anno 2008.
21. Perché SMDS venne progettata come una rete non orientata alla connessione mentre frame relay come una rete orientata alla connessione?
22. Immagina di aver addestrato un cane San Bernardo, Bernie, a portare una scatola contenente tre nastri da 8 mm invece del fiasco di brandy (quando il disco si riempie, si considera un'emergenza e arriva il cane). Questi nastri contengono ciascuno 7 gigabyte. Il cane può viaggiare al tuo fianco, dovunque tu possa essere, a 18 Km

- all'ora. Per quali distanze ha una capacità di trasporto superiore ai 155 Mbps delle linee ATM?
23. Quando si trasferisce un archivio tra due calcolatori, possono essere adottate almeno due diverse strategie per i messaggi che riscontrano l'avvenuto ricevimento (ack). Nel primo caso l'archivio viene spezzato in pacchetti, per ognuno dei quali si utilizza un messaggio di ack, mentre nessun ack viene spedito per l'avvenuto ricevimento dell'intero archivio. Nel secondo caso non vengono usati messaggi di ack per i singoli pacchetti ma solamente per la ricezione dell'intero archivio. Si confrontino questi due metodi.
 24. Si immagini che il pacchetto SMDS di figura 1-28 venga incorporato in una gerarchia di protocolli OSI. In quale livello apparirebbe?
 25. Si citi un vantaggio e uno svantaggio del frame relay su linee telefoniche affittate.
 26. Perché ATM usa celle di lunghezza piccola e prefissata?
 27. Si elenchino due vantaggi e due svantaggi di avere standard internazionali per protocolli di rete.
 28. Quando un sistema ha una parte permanente e una parte rimovibile, come un lettore di dischi o i dischi stessi, è importante che il sistema sia standardizzato, così che le aziende possano fare sia le parti permanenti che quelle rimovibili. Si presentino tre esempi al di fuori del mondo dei calcolatori in cui esistono standard internazionali. Si descrivano anche tre aree, esclusa l'industria dei calcolatori, dove tali standard non esistono.

Capitolo 2

IL LIVELLO FISICO



In questo capitolo studieremo il livello più basso della gerarchia di figura 1-21. Inizieremo con una analisi teorica della trasmissione dei dati, solo per scoprire che madre natura fissa alcuni limiti su ciò che può essere inviato su un canale. Quindi esamineremo i mezzi di trasmissione, sia guidati (fili di rame e fibre ottiche) che non guidati (senza cavo). Questo materiale fornirà dei riferimenti concettuali sulle principali tecnologie di trasmissione usate nelle reti moderne. Il resto del capitolo è dedicato a esempi di sistemi di comunicazione che usano i sottostanti mezzi di trasmissione. Inizieremo con i sistemi telefonici esaminandone tre versioni differenti: Il sistema attuale parzialmente analogico, un potenziale sistema digitale per il prossimo futuro (N-ISDN), e un probabile sistema digitale per il futuro lontano (ATM). Infine daremo uno sguardo a due sistemi senza filo, la radio cellulare e i satelliti di comunicazione.

2.1 Le basi teoriche per la comunicazione dei dati

L'informazione può essere trasmessa su filo modulando una variabile fisica come la tensione o la corrente. Rappresentando i valori della tensione o della corrente come una funzione del tempo $f(t)$, possiamo modellare il comportamento del segnale e analizzarlo matematicamente. Quest'analisi sarà l'argomento dei successivi paragrafi.

2.1.1 Analisi di Fourier

All'inizio del XIX secolo, il matematico francese Jean-Baptiste Fourier dimostrò che ogni funzione periodica $g(t)$ a comportamento ragionevole e con periodo T può essere costruita sommando un numero di seni e coseni (eventualmente infinito):

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

dove $f=1/T$ è la frequenza fondamentale e a_n e b_n sono le ampiezze di seno e coseno della n -esima armonica (termini). Tale decomposizione è chiamata **serie di Fourier**. Dalle serie di Fourier si può ricostruire la funzione; cioè, se il periodo, T , è noto e sono date le ampiezze, si può trovare la funzione del tempo originaria calcolando le somme dell'equazione (2-1).

Un segnale dati che ha una durata finita (che tutti hanno) può essere gestito immaginando

che esso ripeta l'intero schema per sempre (cioè l'intervallo da T a $2T$ è lo stesso da 0 a T ecc.). L'ampiezza a_n può essere calcolata per ogni $g(t)$ moltiplicando entrambi i membri dell'equazione (2-1) per $\sin(2\pi kf t)$ e poi integrando fra 0 e T .

Dal momento che

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

rimane un solo termine della sommatoria: a_n . La sommatoria su b_n scompare completamente. Allo stesso modo, moltiplicando l'equazione (2-1) per $\cos(2\pi kf t)$ e integrando fra 0 e T , possiamo ricavare b_n . Integrando entrambi i membri dell'equazione così com'è possiamo trovare c . I risultati del calcolo di queste operazioni sono i seguenti:

$$a_n = \frac{2}{T} \int_0^T g(t) \sin(2\pi nf t) dt$$

$$b_n = \frac{2}{T} \int_0^T g(t) \cos(2\pi nf t) dt$$

$$a_n = \frac{2}{T} \int_0^T g(t) dt$$

2.1.2 Segnali a larghezza di banda limitata

Per vedere cosa tutto ciò abbia a che fare con la comunicazione dati, consideriamo un esempio specifico: la trasmissione del carattere ASCII "b" codificato in un byte di 8 bit. La sequenza di bit che deve essere trasmessa è 01100010.

La parte sinistra della figura 2-1 (a) mostra il voltaggio in uscita del computer trasmittente. L'analisi di Fourier di questo segnale riporta i coefficienti seguenti:

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

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

$$c = 3/4$$

La media quadratica delle ampiezze, $\sqrt{a_n^2 + b_n^2}$, è raffigurata per i primi termini sulla parte destra di figura 2-1 (a). Questi valori sono interessanti perché i loro quadrati sono proporzionali all'energia trasmessa alla frequenza corrispondente.

Nessun mezzo di trasmissione può trasmettere segnali senza perdere un po' di potenza. Se tutte le componenti di Fourier fossero attenuate allo stesso modo, il segnale risultante sarebbe ridotto in ampiezza ma non distorto (cioè, avrebbe la stessa piacevole forma quadrata come in figura 2-1 (a)). Sfortunatamente, tutti i mezzi di trasmissione abbattano

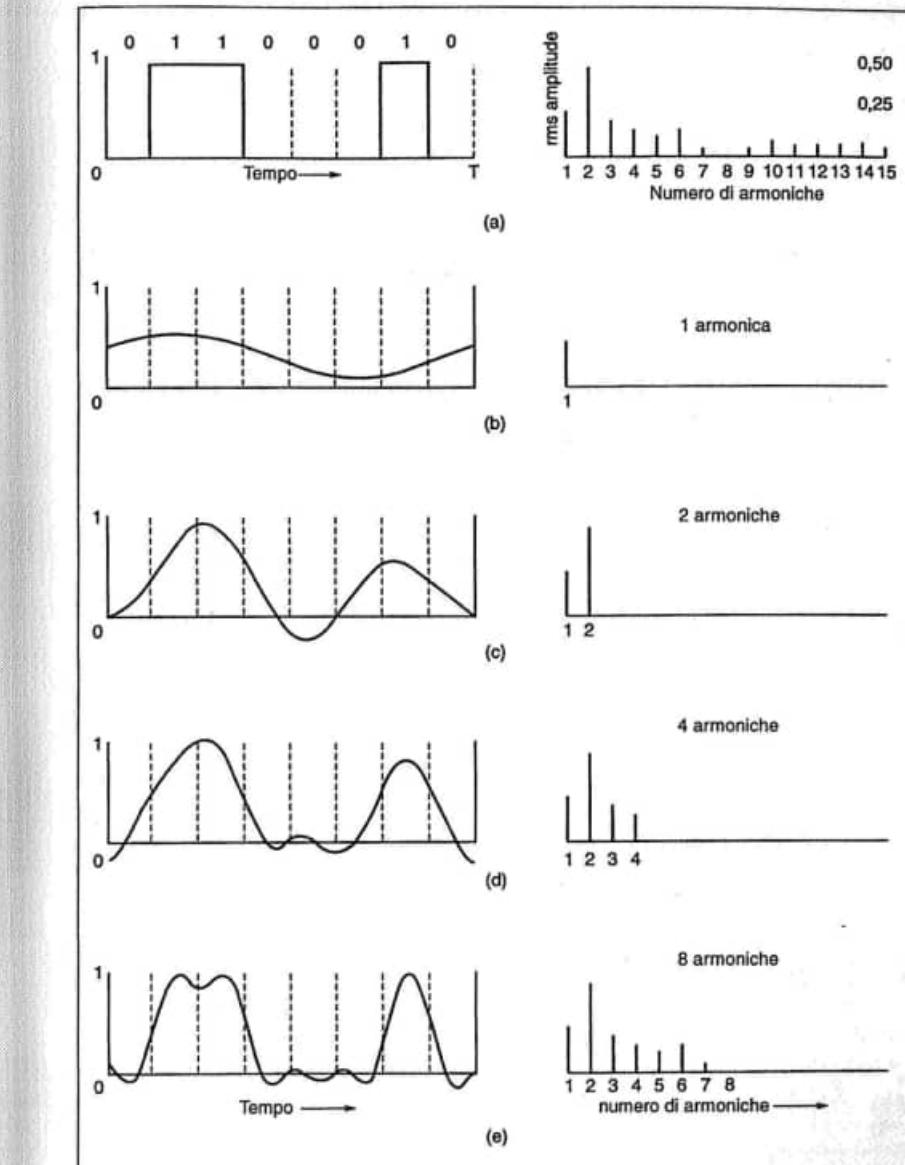


Fig. 2-1 (a) Un segnale binario e la media geometrica delle ampiezze di Fourier. (b)-(e) Approssimazioni successive del segnale originario.

le diverse componenti di Fourier di quantità diverse, introducendo così distorsione. Di solito, le ampiezze sono trasmesse intatte da 0 fino a una frequenza f_c (misurata in cicli/secondo o Hertz (Hz)) mentre tutte quelle al di sopra di questa frequenza di taglio sono fortemente attenuate. In alcuni casi questa è una proprietà fisica del mezzo di trasmissione mentre in altri casi si introduce intenzionalmente un filtro nel circuito per limitare la quantità di larghezza di banda disponibile per ogni cliente.

Consideriamo ora come apparirebbe il segnale di figura 2-1 (a) se la banda fosse così bassa da poter trasmettere solo le frequenze più basse (cioè se la funzione fosse approssimata dai primi termini dell'equazione (2-1)). La figura 2-1 (b) mostra il segnale che risulta da un canale che permette di passare solo alla prima armonica (la fondamentale, f). Allo stesso modo la figura 2-1 (c)-(e) mostra gli spettri e le funzioni ricostruiti per canali con larghezza di banda superiore.

Il tempo T richiesto per trasmettere il carattere dipende sia dal metodo di codifica che dalla velocità di segnalazione (il numero di volte al secondo che il segnale cambia valore – per esempio il suo voltaggio). Il numero di cambiamenti al secondo si misura in **baud**. Una linea da b baud non necessariamente trasmette b bit/s, poiché ogni segnale potrebbe trasportare diversi bit. Se fossero usati i voltaggi 0,1,2,3,4,5,6 e 7, ogni valore del segnale potrebbe essere usato per trasportare 3 bit, sicché il tasso dei bit sarebbe uguale a 3 volte il tasso di baud. Data una velocità di b bit/s, il tempo richiesto per spedire 8 bit (per esempio) è $8/b$ s, così la frequenza della prima armonica è $b/8$ Hz. Una linea telefonica ordinaria, spesso chiamata **linea voice-grade**, ha una frequenza di taglio introdotta artificialmente vicina ai 3000 Hz. Questa restrizione significa che il numero della più alta armonica che la attraversa è $3000/(b/8)$ o approssimativamente $24.000/b$ (il taglio non è preciso). Per alcune velocità di dati, i numeri sono quelli mostrati in figura 2-2. Da questi numeri è chiaro che cercare di trasmettere a 9600 bps su una linea telefonica voice-grade trasformerà la figura 2-1 (a) in qualcosa che appare come la figura 2-1 (c), rendendo difficile la corretta ricezione del flusso di bit originario. Dovrebbe risultare evidente che a tassi molto più alti di 38,4 kbps non c'è nessuna speranza per segnali **binari**, anche se l'attrezzatura di trasmissione è del tutto senza rumore. In altre parole, limitare la larghezza di banda limita la velocità dei dati, anche su canali perfetti. Tuttavia esistono sofisticati schemi di codifica che usano livelli di tensione differenti e possono raggiungere tassi di dati più alti. Ne discuteremo più avanti, in questo capitolo.

2.1.3 Il tasso di dati massimo per un canale

Fin dal 1924, H. Nyquist si rese conto dell'esistenza di un limite fondamentale e derivò un'equazione che esprimeva il tasso massimo di dati per una canale senza rumore a larghezza di banda finita. Nel 1948, Claude Shannon portò avanti il lavoro di Nyquist e lo estese al caso di un canale soggetto a rumore casuale (cioè termodinamico) (Shannon, 1948). Riassumeremo qui brevemente i loro classici risultati.

Nyquist provò che se un segnale arbitrario è stato fatto passare attraverso un filtro passabasso di larghezza di banda H , il segnale filtrato può essere completamente ricostruito facendo solo $2H$ (esatti) campioni al secondo. Campionare la linea più velocemente di $2H$ volte al secondo è inutile poiché le componenti di frequenza più alta che tale campio-

Bps	T (ms)	Prima armonica (Hz)	# Armoniche inviate
300	26,67	37,5	80
600	13,33	75	40
1200	6,67	150	20
2400	3,33	300	10
4800	1,67	600	5
9600	0,83	1200	2
19200	0,42	2400	1
38400	0,21	4800	0

Fig. 2-2 Relazione tra tasso dei dati e armatiche.

namento potrebbe recuperare sono già state filtrate via. Se il segnale consiste di V livelli discreti, il teorema di Nyquist stabilisce che:

$$\text{massimo tasso dati} = 2H \log_2 V \text{ bit/s}$$

Per esempio, un canale senza rumore a 3 kHz non può trasmettere segnali binari (cioè a due livelli) a un tasso superiore a 6000 bps.

Finora abbiamo considerato solo canali senza rumore. Se è presente del rumore casuale, la situazione si deteriora rapidamente. La quantità di rumore termico si misura col rapporto fra la potenza del segnale e la potenza del rumore, chiamato **rapporto segnale-rumore**. Se indichiamo con S la potenza del segnale e con N la potenza del rumore, il rapporto segnale-rumore è S/N . Di solito, non si dà il rapporto; invece si indica la quantità $10 \log_{10} S/N$. Queste unità sono chiamate **decibel** (dB). Un rapporto S/N di 10 è 10 dB, un rapporto di 100 è 20 dB, un rapporto di 1000 è 30 dB e così via. I produttori di amplificatori stereo caratterizzano la larghezza di banda (la gamma di frequenza) sulla quale il loro prodotto è lineare dando la frequenza a 3 dB su ogni estremità. Questi sono i punti ai quali il fattore di amplificazione è stato approssimativamente dimezzato.

Il maggior risultato di Shannon è che il massimo tasso di dati su un canale rumoroso la cui banda è H Hz, e il cui rapporto segnale-rumore è S/N , è dato da

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

Per esempio, un canale con larghezza di banda 3000 Hz, e un rapporto tra segnale e rumore termico di 30 dB (parametri tipici delle componenti analogiche di un sistema telefonico) non può mai trasmettere a molto più di 30.000 bps, non importa quanti livelli di segnale siano usati e quanto più o meno spesso siano presi i campioni. Il risultato di Shannon fu derivato usando argomenti di teoria dell'informazione e applicato ad ogni canale soggetto a rumore (termico) gaussiano. I controesempi dovrebbero essere trattati

allo stesso modo di macchine per il moto perpetuo. Si noti, comunque, che questo è solo un limite superiore e raramente i sistemi reali lo raggiungono.

2.2 Mezzi di trasmissione

Lo scopo del livello fisico è di trasportare un flusso grezzo di bit da una macchina a un'altra. Per la trasmissione reale possono essere usati vari mezzi fisici. Ciascuno ha la sua nicchia in termini di larghezza di banda, ritardo, costo e facilità di installazione e manutenzione. I mezzi sono grosso modo classificabili in mezzi *guidati*, come fili di rame e fibre ottiche, e mezzi *non guidati*, come radio e laser via etere. Li studieremo in questo e nel successivo paragrafo.

2.2.1 Mezzi magnetici

Uno dei modi più comuni per trasportare dati da un computer all'altro è di scriverli su un nastro magnetico o su floppy disk, portare fisicamente il nastro o i dischi alla macchina di destinazione, e leggerli di nuovo. Sebbene questo metodo non sia così sofisticato come usare un satellite di comunicazione geosincrono, è spesso molto più efficace nei costi, specialmente per applicazioni in cui il fattore chiave è un'alta larghezza di banda o il costo per bit trasportato.

Un semplice calcolo chiarirà questo punto. Un nastro video standard a 8 mm (ad es. Exabyte) può contenere 7 gigabytes. Una scatola di dimensioni 50×50×50 cm può contenere circa 1000 di questi nastri per una capacità totale di 7000 gigabytes. Una scatola di nastri può essere consegnata dovunque negli Stati Uniti in 24 ore dalla Federal Express e da altre società. La larghezza di banda effettiva di questa trasmissione è di 56.000 gigabit/86.400 s o 648 Gbps, che è leggermente migliore della versione ad alta velocità di ATM (622 Mbps). Se la destinazione è solo a un'ora di strada, la larghezza di banda aumenta fino a più di 15 Gbps.

Per una banca con gigabytes di dati da salvare giornalmente su una seconda macchina (in modo che la banca possa continuare a funzionare anche in occasione di un grande alluvione o di un terremoto) è probabile che nessun'altra tecnologia di trasmissione possa nemmeno iniziare ad avvicinarsi alle prestazioni di un nastro magnetico.

Se analizziamo i costi, otteniamo un quadro analogo. Il costo di 1000 nastri video è forse di 5000 dollari se comprati all'ingrosso. Un nastro video può essere riusato almeno dieci volte, in questo modo il costo diventa 500 dollari. Aggiungiamo a questo altri 200 dollari per la spedizione e otteniamo un costo di circa 700 dollari per spedire 7000 gigabyte. Ciò equivale a 10 cent al gigabyte. La morale della favola è:

Non sottostimare la larghezza di banda di un camioncino che passa a razzo sull'autostrada.

2.2.2 Il doppino (twisted pair)

Sebbene le caratteristiche di larghezza di banda dei supporti magnetici siano eccellenti, le caratteristiche di ritardo sono assai mediocri. Il tempo di trasmissione si misura in minuti o in ore, non in millisecondi. Per molte applicazioni invece è necessaria una

2.2 Mezzi di trasmissione

connessione in diretta. Il più vecchio e ancora il più comune mezzo di trasmissione è il **doppino**. Un doppino consiste di due fili di rame isolati, tipicamente spessi 1 mm. I fili sono intrecciati insieme in una forma elicoidale, proprio come una molecola di DNA. La ragione per cui i fili si intrecciano è che si riducono le interferenze elettriche provenienti da coppie simili poste nelle vicinanze (due fili paralleli costituiscono una semplice antenna: due fili intrecciati no).

L'applicazione più comune del doppino è il sistema telefonico. Quasi tutti i telefoni sono collegati agli uffici della società telefonica da un doppino. I doppini corrono senza amplificazione per diversi chilometri, ma per distanze più lunghe sono necessari dei ripetitori. Quando molti doppiini corrono in parallelo per una distanza notevole, così come tutti i fili che provengono da un palazzo fino all'ufficio della società telefonica, essi sono legati insieme e raggruppati in guaine protettive. Le coppie in questi fasci provocherebbero interferenze una con l'altra se non fosse per l'intreccio. In alcune parti del mondo dove le linee telefoniche corrono su pali in superficie, è normale vedere fasci di diversi centimetri di diametro.

I doppiini possono essere usati sia per trasmissioni analogiche che digitali. La larghezza di banda dipende dallo spessore del filo e dalla distanza percorsa, ma in molti casi si possono raggiungere diversi megabit/s per distanze di chilometri. Grazie alle loro prestazioni adeguate e al basso costo, i doppiini vengono largamente usati e così è probabile che sia per molti anni a venire.

I doppiini hanno diverse varietà, due delle quali sono importanti per le reti di computer. I doppiini di **categoria 3** consistono di due fili isolati delicatamente intrecciati. Quattro coppie sono in genere raggruppate insieme in una guaina di plastica per ottenerne otto fili. Prima del 1988, la maggior parte degli uffici aveva un cavo di categoria 3 che correva da un **vano di fili** centrale su ogni piano in ogni ufficio. Questo schema permetteva fino a quattro telefoni o a due telefoni multilinea in ogni ufficio di connettersi all'apparecchiatura telefonica della società nel vano fili.

A partire dal 1988, furono introdotti i più avanzati doppiini di **categoria 5**. Essi sono simili ai doppiini di categoria 3, ma con più intrecciamenti per centimetro e isolamento in Teflon, che porta a meno interferenze e una migliore qualità del segnale su distanze più lunghe, rendendoli più adatti a comunicazioni ad alta velocità fra computer. Entrambi questi tipi di cavi sono indicati come **UTP (Unshielded Twisted Pair)**, cioè doppiini non schermati, per distinguerli dai voluminosi, costosi, cavi schermati IBM introdotti all'inizio degli anni ottanta, che però non sono diventati popolari al di fuori delle installazioni IBM.

2.2.3 Cavo coassiale a banda base

Un altro comune mezzo di trasmissione è il **cavo coassiale** (noto ai suoi molti amici semplicemente come "coax"). Offre una migliore protezione rispetto al cavo intrecciato, e può attraversare distanze più lunghe a una velocità maggiore. Sono ampiamente usati due tipi di cavo coassiale. Un tipo, il cavo a 50 Ω, è comunemente usato per le trasmissioni digitali ed è l'argomento di questo paragrafo. L'altro tipo, il cavo a 75 Ω, è comunemente usato per le trasmissioni analogiche e sarà descritto nel paragrafo successivo. Questa distinzione si basa su fattori storici piuttosto che tecnici. (Per esempio le prime antenne

a dipolo avevano una impedenza di 300Ω ed era facile costruire trasformatori con una impedenza 4:1).

Un cavo coassiale consiste di un filo di rame rigido come nucleo, circondato da materiale isolante. L'isolante è racchiuso da un conduttore cilindrico, spesso come una maglia fittamente intrecciata. Il conduttore esterno è ricoperto da una guaina protettiva di plastica. Una vista in sezione di un cavo coassiale è mostrata in figura 2-3.

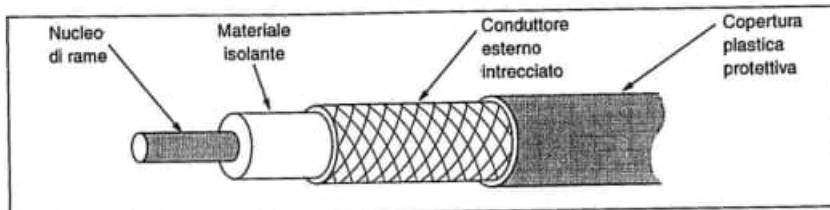


Fig. 2-3 Un cavo coassiale.

La costruzione e la schermatura del cavo coassiale dà una buona combinazione di alta larghezza di banda e di eccellente immunità al rumore. La larghezza di banda possibile dipende dalla lunghezza del cavo. Per cavi da 1 km, è ammisible una velocità di dati da 1 fino a 2 Gbps. Si possono usare anche cavi più lunghi ma solo con una velocità di dati più bassa o con amplificatori. I cavi coassiali sono di solito largamente usati nei sistemi telefonici ma adesso sono stati in gran parte rimpiazzati sulle linee a lungo percorso da fibre ottiche. Solo negli Stati Uniti, sono installati ogni giorno 1000 km di fibre ottiche (contando un fascio di 100 km con 10 cavi di fibre come 1000 km). Sprint è già al 100% in fibra ottica, e le altre principali società si stanno rapidamente avvicinando a questo limite. Il cavo coassiale è comunque ancora largamente usato per la televisione via cavo e per alcune reti locali.

2.2.4 Cavo coassiale a larga banda

L'altro tipo di sistema a cavo coassiale usa la trasmissione analogica sul cablaggio standard della televisione via cavo. È chiamato a larga banda (**broadband**). Sebbene il termine "a larga banda" provenga dal mondo della televisione, dove si riferisce a tutto ciò che è a più di 4 kHz, nel mondo delle reti di computer "cavo a larga banda" denota una rete di cavi che impiega la trasmissione analogica (si veda Cooper, 1986).

Dal momento che le reti a larga banda usano la tecnologia standard della televisione via cavo, i cavi possono essere usati fino a 300 MHz (e spesso fino a 450 MHz) e possono correre per quasi 100 km grazie alla segnalazione analogica che è molto meno critica della segnalazione digitale. Per trasmettere segnali digitali su una rete analogica, ogni interfaccia deve contenere un circuito per convertire il flusso di bit in uscita in un segnale analogico, e il segnale analogico in entrata in un flusso di bit. A seconda del tipo di circuito, 1 bps può occupare circa 1 Hz di larghezza di banda. A frequenze più alte, sono possibili tassi di molti bit per Hz impiegando tecniche di modulazione avanzata.

I sistemi a larga banda sono divisi in canali multipli, di solito canali da 6 MHz usati per

le trasmissioni televisive. Ogni canale può essere usato per la televisione analogica, audio di qualità CD (1,4 Mbps) o un flusso di bit digitale, per esempio a 3 Mbps, indipendentemente dagli altri. Televisione e dati possono essere mescolati su un solo cavo. Una differenza chiave tra banda base e larga banda è che i sistemi a larga banda tipicamente coprono uno spazio maggiore e perciò hanno bisogno di amplificatori analogici per rafforzare il segnale periodicamente. Questi amplificatori possono trasmettere solo segnali in una direzione, così un computer che emette un pacchetto non potrà raggiungere i computer "in alto" se un amplificatore si trova fra di loro. Per superare questo problema, sono stati sviluppati due tipi di sistemi a larga banda: sistemi a doppio e a singolo cavo. I sistemi a cavo doppio hanno due cavi identici stesi in parallelo, uno vicino all'altro. Per trasmettere dati, un computer invia dati nel cavo 1, che porta a un dispositivo chiamato **head-end** alla radice dell'albero del cavo. Tale dispositivo trasferisce poi il segnale al cavo 2 per la trasmissione giù nell'albero. Tutti i computer trasmettono sul cavo 1 e ricevono sul cavo 2. Un sistema di cavo doppio è mostrato in figura 2-4 (a). L'altro schema alloca bande di frequenza differenti per le comunicazioni in entrata e in uscita su un singolo cavo (vedi fig. 2-4 (b)).

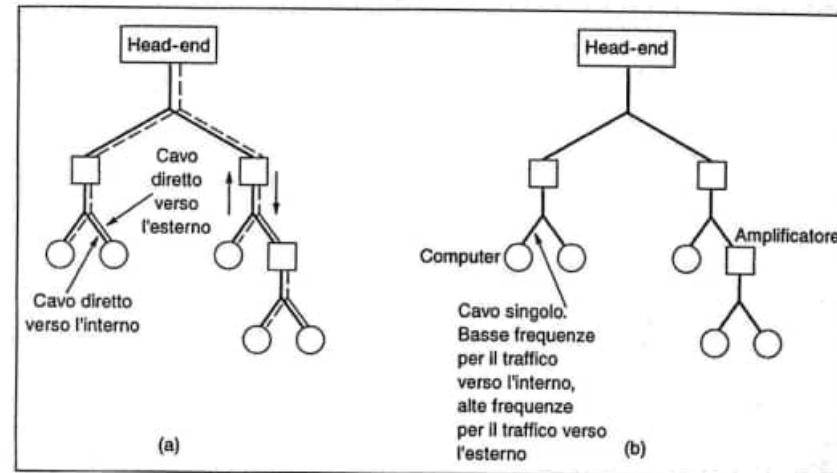


Fig. 2-4 Reti a larga banda. (a) Cavo doppio. (b) Cavo singolo.

La banda a bassa frequenza è usata per la comunicazione dai computer al dispositivo head-end, che poi sposta il segnale alla banda ad alta frequenza e lo ritrasmette. Nel sistema **subsplit**, le frequenze da 5 a 30 MHz sono usate per il traffico in entrata e le frequenze da 40 a 300 MHz sono usate per il traffico in uscita.

Nel sistema **midsplit**, la banda in entrata è da 5 a 116 MHz e la banda in uscita è da 168 a 300 MHz. La scelta di queste bande di frequenza è storica, in quanto deriva dal modo in cui la Commissione federale per le comunicazioni (FCC) degli USA aveva assegnato le frequenze per le trasmissioni televisive, per le quali era stata progettata

la larga banda. Entrambi i sistemi richiedono una head-end attiva che accetti i segnali in entrata su una banda e li ritrasmetta in uscita su un'altra. Queste tecniche e frequenze sono state sviluppate per la televisione via cavo e sono state riusate senza modifiche per le reti grazie alla disponibilità di hardware affidabile e relativamente poco costoso. La larga banda può essere usata in diversi modi. Si può assegnare ad alcune copie di computer un canale fisso per loro uso esclusivo. Altri computer possono richiedere un canale per una connessione temporanea su un canale di controllo e poi usare tale canale per la durata della connessione. Ancora un'altra strategia consiste nel porre tutti i computer in competizione per l'accesso a un singolo canale o a un gruppo di canali, usando tecniche esaminate nel capitolo 4.

Tecnicamente, per spedire dati digitali il cavo a larga banda è inferiore al cavo a banda base (cioè a canale singolo), ma ha il vantaggio che in genere è già in sito. In Olanda, per esempio, il 90% di tutte le case hanno una connessione alla TV via cavo. Negli Stati Uniti, la TV via cavo copre oltre l'80% di tutte le case. Di queste circa il 60% hanno davvero una connessione attiva. Con la competizione fra società telefoniche e società di TV via cavo già a pieno ritmo, possiamo aspettarci che i sistemi di TV via cavo presto inizino a operare come MAN per offrire sempre più spesso anche servizi telefonici e altro. Per maggiori informazioni sull'uso dei cavi TV come supporto a reti di computer si vedano Karshmer, Thomas (1992).

2.2.5 Fibre ottiche

Molti nell'industria informatica sono assai orgogliosi della velocità con la quale progredisce la tecnologia dei computer. Negli anni settanta, un computer veloce (ad es. CDC 6600) poteva eseguire un'istruzione in 100 ns. Venti anni più tardi, un veloce computer Cray poteva eseguire un'istruzione in 1 ns, un fattore di miglioramento di 10 per ciascun decennio. Niente male.

Nello stesso periodo, la comunicazione dati passò da 56 kbps (la ARPANET) a 1 Gbps (con le moderne comunicazioni ottiche), un guadagno maggiore di un fattore di 100 per decennio, mentre nello stesso tempo il tasso di errore si ridusse da 10^{-5} per bit a quasi zero. Inoltre, le CPU singole stanno iniziando ad avvicinarsi ai limiti fisici, come la velocità della luce e problemi di dissipazione del calore. Per contro, con l'*attuale* tecnologia a fibra, la larghezza di banda raggiungibile è certamente superiore a 50.000 Gbps (50 Tbps) e molti ricercatori stanno sviluppando materiali migliori. Agli effetti pratici il limite attuale di banda di circa 1 Gbps è dovuto alla nostra incapacità a convertire più velocemente segnali elettrici in segnali ottici. In laboratorio, 100 Gbps sono stati raggiunti su brevi tratte. Una velocità di 1 terabit/s sarà raggiungibile fra pochissimi anni. Sono a portata di mano sistemi completamente ottici, compreso l'ingresso e l'uscita dai computer (Miki, 1994a).

La corsa fra calcolatori e comunicazione è stata dunque vinta dalla comunicazione. Le implicazioni di una larghezza di banda in sostanza infinita (sebbene non a costo zero) non sono state ancora completamente esplorate da una generazione di scienziati e ingegneri informatici educati a pensare in termini dei ristretti limiti di Nyquist e Shannon imposti dal filo in rame. Il nuovo senso comune dovrebbe essere che tutti i computer sono lenti senza speranza, e le reti dovrebbero cercare di evitare la computazione a tutti i costi, non importa quanta larghezza di banda si perda.

2.2 Mezzi di trasmissione

In questo paragrafo studieremo le fibre ottiche per vedere come funziona la tecnologia di trasmissione.

Un sistema di trasmissione ottico ha tre componenti: la sorgente di luce, il mezzo di trasmissione e il rilevatore. Conventionalmente, un impulso di luce indica un bit 1 e l'assenza di luce indica un bit 0. Il mezzo di trasmissione è una fibra di vetro ultrasottile. Il rilevatore genera un impulso elettrico quando la luce cade su di esso. Attaccando una sorgente di luce a una estremità di una fibra ottica e un rilevatore all'altra, abbiamo un sistema di trasmissione di dati unidirezionale che accetta un segnale elettrico, lo converte e lo trasmette mediante impulsi luminosi e poi lo riconverte all'uscita in un segnale elettrico all'estremità ricevente. Questo sistema di trasmissione perderebbe luce e sarebbe in pratica inutile tranne che per un interessante principio di fisica. Quando un raggio di luce passa da un mezzo all'altro, per esempio dal silicio fuso all'aria, il raggio viene rifratto (curvato) al confine silicio/aria come mostrato in figura 2-5. Qui si vede un raggio di luce incidente sul confine a un angolo α_1 , che emerge a un angolo β_1 . La quantità di rifrazione dipende dalle proprietà dei due mezzi (in particolare, dai loro indici di rifrazione). Per angoli di incidenza sopra un certo valore critico, la luce viene riflessa di nuovo nel silicio; nessuno di essi fugge nell'aria. Così un raggio di luce incidente a o al di sopra dell'angolo critico rimane intrappolato all'interno della fibra, come mostrato in figura 2-5 (b) e si può propagare per molti chilometri virtualmente senza alcuna perdita.

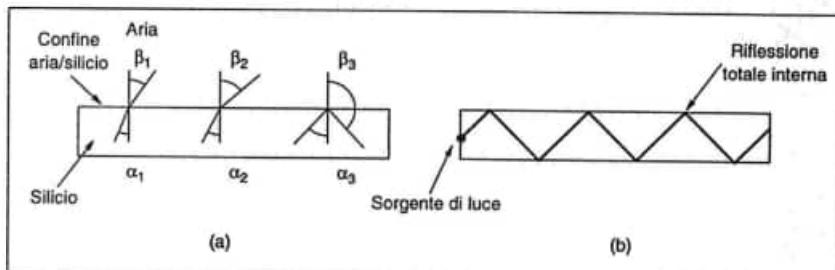


Fig. 2-5 (a) Tre esempi di un raggio di luce dall'interno di una fibra di silicio che sbatte contro il confine silicio/aria ad angoli diversi. (b) Luce intrappolata da riflessione totale interna.

Il disegno di figura 2-5 (b) mostra un solo raggio intrappolato, ma poiché qualsiasi raggio di luce incidente sul confine al di sopra dell'angolo critico sarà riflesso internamente, molti raggi diversi rimbalzeranno con angoli diversi. Si dice che ogni raggio ha un modo differente, quindi una fibra che ha questa proprietà viene chiamata **fibra multimodo**. Tuttavia, se il diametro della fibra si riduce a poche onde di lunghezza di luce, la fibra agisce come un'onda guidata e la luce si può propagare solo in linea retta, senza rimbalzare, portando a una **fibra a modo singolo**. Le fibre a modo singolo sono più costose ma possono essere usate per distanze più lunghe. Le fibre a modo singolo attualmente disponibili possono trasmettere dati a diversi Gbps per 30 km. Ogni velocità di dati più alta è stata raggiunta in laboratorio per distanze minori. Esperimenti hanno dimostrato che laser

potenti possono guidare una fibra lunga 100 km senza ripetitori, sebbene a velocità più basse. La ricerca su fibre a erbio-dopato promette percorsi più lunghi senza ripetitori.

Trasmissione della luce attraverso la fibra

Le fibre ottiche sono fatte di vetro, che, a sua volta, è fatto di sabbia, un materiale grezzo non costoso disponibile in quantità illimitata. L'arte del vetro era nota agli antichi Egizi, ma il loro vetro non era più spesso di 1 mm o la luce non poteva attraversarlo brillando. Il vetro trasparente per finestre venne sviluppato durante il Rinascimento. Il vetro usato per le fibre ottiche moderne è così trasparente che se gli oceani fossero pieni di esso invece che d'acqua, da un aeroplano il fondo del mare sarebbe visibile quanto la superficie della terra in un giorno limpido.

L'attenuazione della luce che attraversa il vetro dipende dalla lunghezza d'onda della luce stessa. Nel caso del vetro utilizzato nelle fibre, l'attenuazione è mostrata in figura 2-6 in decibel per chilometro lineare di fibra. L'attenuazione in decibel è data dalla formula

$$\text{Attenuazione in decibel} = 10 \log_{10} (\text{potenza trasmessa} / \text{potenza ricevuta})$$

Per esempio, un fattore di perdita 2 produce un'attenuazione di $10 \log_{10} 2 = 3$ dB. La figura mostra la parte vicino all'infrarosso dello spettro, che è quella che è usata in pratica. La luce visibile ha lunghezze d'onda leggermente più corte, da 0,4 a 0,7 μ (micron) ($1 \mu = 10^{-6}$ m).

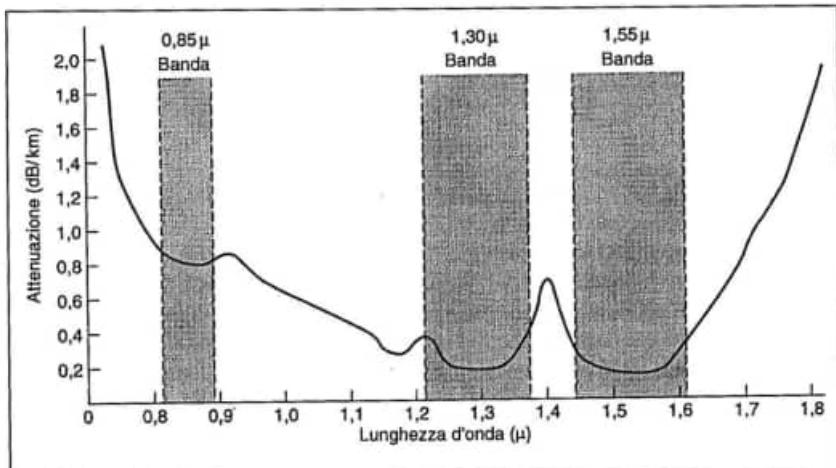


Fig. 2-6 Attenuazione della luce attraverso la fibra nella regione infrarossa.

Per la comunicazione si usano tre bande di lunghezza d'onda. Esse sono centrate a 0,85, 1,30 e 1,55 μ , rispettivamente. L'ultima ha buone proprietà di attenuazione (perdita di meno del 5% per chilometro). La banda a 0,85 μ ha attenuazione più alta, ma la buona

2.2 Mezzi di trasmissione

proprietà che a quella lunghezza d'onda, i laser e l'elettronica possono essere fatte dello stesso materiale (gallio arsenide). Tutte le tre bande sono larghe da 25.000 a 30.000 GHz. Gli impulsi di luce inviati su una fibra si diffondono in lunghezza quando si propagano. Questa diffusione viene chiamata dispersione. La sua quantità dipende dalla lunghezza d'onda. Un modo per contenere questi impulsi di diffusione dalla sovrapposizione è di aumentare la distanza fra loro, ma questo si può fare solo riducendo la velocità di segnalazione. Fortunatamente, si è scoperto che dando una speciale forma agli impulsi in relazione all'inverso del coseno iperbolico, tutti gli effetti della dispersione si cancellano, ed è possibile inviare impulsi per migliaia di chilometri senza apprezzabili distorsioni di forma. Questi impulsi vengono chiamati solitoni. È in corso al momento una considerevole ricerca allo scopo di far uscire i solitoni dai laboratori e portarli sul campo.

Cavi di fibre

I cavi di fibra ottica sono simili ai coax, eccetto che per la maglia. La figura 2-7 (a) mostra una singola fibra vista di lato. Al centro c'è il nucleo di vetro attraverso il quale si propaga la luce. Nelle fibre multimodo, il nucleo è di 50 μ di diametro, circa lo spessore di un cappello umano. Nelle fibre a modo singolo, il nucleo varia da 8 a 10 μ .

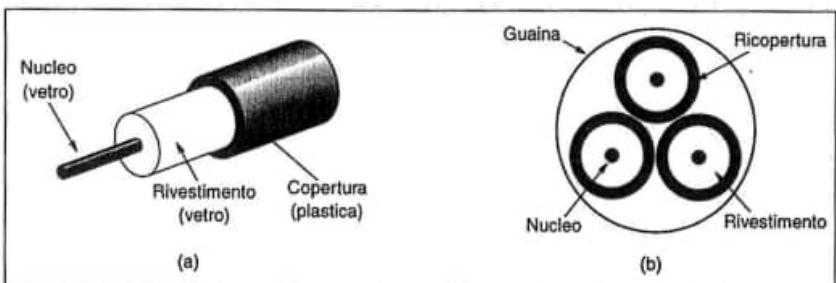


Fig. 2-7 (a) Vista laterale di una singola fibra. (b) Estremità di una guaina con tre fibre.

Il nucleo è circondato da un rivestimento di vetro con indice di rifrazione più basso di quello del nucleo, per mantenere tutta la luce nel nucleo. Vicino c'è una sottile copertura in plastica per proteggere il rivestimento. Le fibre sono tipicamente raggruppate insieme in fasci, e protette da una guaina esterna. La figura 2-7 (b) mostra una guaina con tre fibre. I fasci di fibre terrestri sono posati nel terreno normalmente a un metro sotto la superficie, dove sono occasionalmente soggetti ad attacchi da parte di zuppe o talpe. Vicino alle spiagge, i fasci di fibre transoceaniche sono sepolti in trincee da una specie di aratro marino. In acque profonde, essi giacciono sul fondo, dove possono essere ostacolati dalla pesca a strascico o mangiati dagli squali.

Le fibre possono essere connesse in tre modi diversi. Primo, possono terminare in connettori ed essere inserite in prese per fibre. I connettori perdono circa dal 10 al 20% di luce, ma semplificano la riconfigurazione del sistema.

Secondo, possono essere unite meccanicamente. Le giunzioni meccaniche poggiano ac-

curatamente le due estremità una vicina all'altra in una manica speciale e le stringono insieme. L'allineamento può essere migliorato facendo passare la luce attraverso la giunzione e quindi facendo piccoli aggiustamenti per massimizzare il segnale. Le giunzioni meccaniche richiedono circa cinque minuti a personale addestrato e risultano in una perdita del 10% di luce.

Terzo, due pezzi di fibra possono essere fusi per formare una connessione solida. Una giunzione di fusione è tanto buona quanto una singola fibra tirata dritta, ma anche qui c'è una piccola quantità di attenuazione. Per tutti e tre i tipi di giunzione, ci può essere riflessione nel punto di giunzione e l'energia riflessa può interferire col segnale.

Si possono usare due tipi di sorgenti di luce per trasmettere segnali, LED (Light Emitting Diodes) e laser semiconduttori. Essi hanno proprietà diverse, come mostrato in figura 2-8. Possono essere regolati nella lunghezza d'onda inserendo interferometri di Fabry-Perot o di Mach-Zehnder tra la sorgente e il cavo. Gli interferometri di Fabry-Perot sono semplici cavità risonanti che consistono di due specchi paralleli. La luce è incidente perpendicolarmente agli specchi. La lunghezza della cavità seleziona quelle lunghezze d'onda che si adattano all'interno un numero intero di volte. Gli interferometri di Mach-Zehnder separano la luce in due fasci che viaggiano a distanze diverse. Sono ricombinati alla fine e restano in fase solo per certe lunghezze d'onda.

L'estremità ricevente di una fibra ottica consiste di un fotodiodo, che restituisce un im-

Caratteristica	LED	Laser semiconduttore
Velocità dati	Bassa	Alta
Modo	Multimodo	Multimodo o a modo singolo
Distanza	Corta	Lunga
Durata	Vita lunga	Vita breve
Sensibilità alla temperatura	Minore	Sostanziale
Costo	Economica	Costosa

Fig. 2-8 Un confronto fra diodi semiconduttori e LED come sorgenti di luce.

pulso elettrico quando è colpito dalla luce. Il tempo tipico di risposta di un fotodiodo è di 1 ns, il che limita il flusso di dati a circa 1 Gbps. Il rumore termico è anche un problema, così un impulso di luce deve portare abbastanza energia da essere rilevato. Facendo gli impulsi abbastanza potenti, il tasso di errore può essere reso arbitrariamente piccolo.

Reti di fibre ottiche

Le fibre ottiche possono essere usate per le LAN così come per trasmissioni a lungo percorso, sebbene utilizzarle sia più complesso che connettersi a una Ethernet. Un modo di evitare il problema è quello di rendersi conto che una rete ad anello è in realtà solo un insieme di connessioni punto-a-punto, come in figura 2-9. L'interfaccia di ciascun com-

2.2 Mezzi di trasmissione

puter fa passare il flusso di impulsi di luce alla connessione successiva e serve anche come giunzione a T per permettere al computer di spedire e ricevere messaggi.

Si usano due tipi di interfaccia. Un'interfaccia passiva consiste di due prese fuse insieme nella fibra principale. Una presa ha un LED o un diodo laser all'estremità (per trasmettere) e l'altra ha un fotodiodo per ricevere. La stessa presa è completamente passiva ed è così estremamente affidabile perché un LED o un fotodiodo rotto non rompono l'anello, ma piuttosto sconnettono il computer.

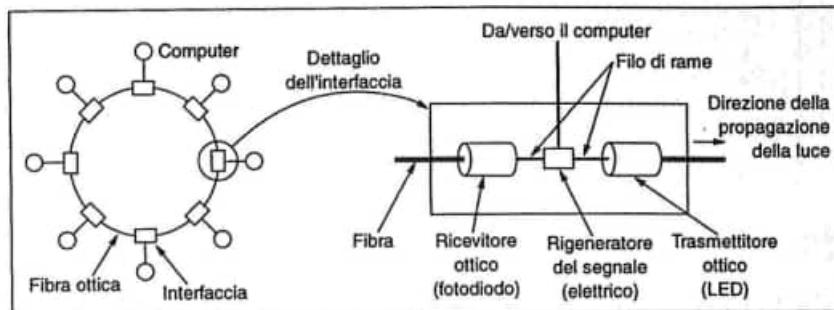


Fig. 2-9 Un anello in fibra ottica con ripetitori attivi.

L'altro tipo di interfaccia, mostrato in figura 2-9, è il ripetitore attivo. La luce in entrata è convertita in un segnale elettrico, rigenerato a piena potenza se è stata indebolita, e ritrasmessa come luce. L'interfaccia col computer è un normale filo di rame che entra nel rigeneratore del segnale. Ora sono usati anche ripetitori solo ottici. Questi dispositivi non richiedono la conversione ottica-elettrica-ottica, il che significa che possono operare a larghezze di banda estremamente alte.

Se un ripetitore attivo si guasta, l'anello si rompe e la rete va giù. D'altra parte, poiché il segnale è rigenerato a ogni interfaccia, le connessioni computer a computer possono essere lunghe chilometri, senza limite alla dimensione totale dell'anello. Invece, le interfacce passive perdono luce a ogni giunzione, sicché il numero di computer e la lunghezza totale dell'anello sono fortemente limitati.

La topologia ad anello non è l'unico modo di costruire una LAN usando fibre ottiche. È anche possibile usare la struttura a **stella passiva** di figura 2-10. In questo schema, ogni interfaccia ha una fibra che corre dal suo trasmettitore a un cilindro di silicio, in cui le fibre in entrata sono fuse a una estremità del cilindro. Allo stesso modo, le fibre sono fuse all'altra estremità del cilindro e arrivano a ciascuno dei riceventi. Quando un'interfaccia emette un impulso luminoso, questo viene diffuso all'interno della stella passiva e illumina tutti i riceventi: un broadcast. In effetti, la stella passiva raccoglie tutti i segnali in entrata e trasmette il risultato fuso su tutte le linee. Dal momento che l'energia in entrata si divide fra tutte le linee in uscita, il numero di nodi nella rete è limitato dalla sensibilità del fotodiodo.

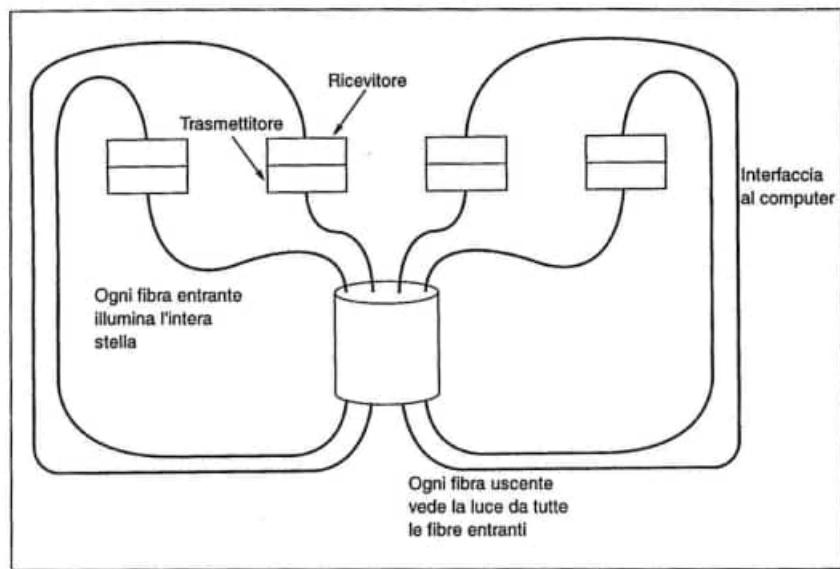


Fig. 2.10 Una connessione a stella passiva in una rete a fibra ottica.

Confronto fra fibre ottiche e filo di rame

È istruttivo confrontare la fibra col rame: la fibra ha molti vantaggi. Per cominciare, essa può gestire larghezze di banda superiori al rame. Questo fatto da solo giustifica il suo uso nelle reti veloci. Grazie alla bassa attenuazione, i ripetitori lungo le linee sono necessari solo circa ogni 30 km, contro i circa 5 km del rame, un risparmio sostanziale. La fibra ha anche il vantaggio di non essere soggetta a picchi improvvisi di corrente, interferenze elettromagnetiche o mancanze di energia. E neppure viene intaccata da corrosivi chimici nell'aria, il che la rende ideale per ambienti di fabbrica.

Ma le società telefoniche preferiscono la fibra ancora altre ragioni: è sottile e poco pesante. Molti degli attuali condotti di cavi sono completamente pieni, sicché non c'è abbastanza spazio per aggiungere nuovi cavi. Rimuovendo tutto il rame e rimpiazzandolo con le fibre i condotti si svuotano e il rame si può rivendere vantaggiosamente ai raffinatori che lo usano come un minerale grezzo di alta qualità. La fibra è inoltre più leggera del rame. Un migliaio di fili intrecciati lunghi 1 km pesano 8000 kg. Due fibre hanno più capacità e pesano solo 100 kg, il che riduce enormemente il bisogno di costosi sistemi di supporto meccanici che richiedono manutenzione. Se si devono stendere ex novo, le fibre vincono a mani basse grazie al loro costo di installazione molto minore. Infine, le fibre non si infiltrano facilmente e sono molto difficili da intercettare. Ciò dà una grande sicurezza contro potenziali intercettatori.

Il motivo per cui la fibra è migliore del rame riguarda la fisica di base. Quando gli elettroni si muovono in un filo, essi si influenzano uno con l'altro e sono essi stessi influenzati da

2.3 La trasmissione senza filo

elettroni all'esterno del filo. I fotoni in una fibra non si influenzano (non hanno carica elettrica) e non sono influenzati da fotoni vaganti all'esterno della fibra.

D'altra parte, la fibra è una tecnologia poco familiare che richiede abilità che la maggior parte degli ingegneri non ha. Poiché la trasmissione ottica è inerentemente monodirezionale, la comunicazione a due vie richiede o due fibre o due bande di frequenza su una fibra. Infine, le interfacce per la fibra costano più delle interfacce elettriche. Non di meno il futuro di tutte le comunicazioni fisse per distanze di più di pochi metri è chiaramente con la fibra. Per una discussione dettagliata di tutti gli aspetti delle reti a fibra ottica si veda Green (1993).

2.3 La trasmissione senza filo

Il nostro tempo ha dato origine a drogati dell'informazione: persone che hanno bisogno di essere collegate ogni momento. Per questi utenti mobili, il filo intrecciato, il cavo coassiale e la fibra ottica non hanno alcuna utilità. Essi hanno bisogno di ricevere dati nei loro computer laptop, portatili, da tasca, palmari o da polso senza collegarsi all'infrastruttura di comunicazione terrestre. Per tali utenti, la soluzione è la comunicazione senza filo. In questo paragrafo daremo uno sguardo alla comunicazione senza filo in generale, dal momento che essa ha molte altre importanti applicazioni, oltre a fornire connettività a utenti che vogliono leggere la posta elettronica su un aereo.

Alcuni credono che il futuro ci porterà a solo due tipi di comunicazioni: fibra e senza filo. I computer fissi (cioè non mobili), i telefoni, fax, e così via saranno su fibra e tutti quelli mobili senza cavo.

Tuttavia la comunicazione senza fili è vantaggiosa in alcune circostanze persino per i dispositivi fissi. Per esempio, se risulta difficile stendere una fibra a causa della conformazione del terreno (montagne, giungle, paludi ecc.) la comunicazione senza filo può essere preferibile. Vale la pena di notare che la comunicazione digitale moderna senza filo iniziò nelle isole Hawaii, dove grandi tratti dell'Oceano Pacifico dividevano gli utenti e il sistema telefonico era inadeguato.

2.3.1 Lo spettro elettromagnetico

Quando gli elettroni si muovono, creano delle onde elettromagnetiche che si possono propagare attraverso lo spazio libero (persino nel vuoto). Queste onde furono previste dal fisico inglese James Clerk Maxwell nel 1865 e per la prima volta prodotte e osservate dal fisico tedesco Heinrich Hertz nel 1887. Il numero di oscillazioni al secondo di un'onda elettromagnetica si chiama **frequenza**, f , e si misura in **Hz** (in onore di Heinrich Hertz). La distanza fra due massimi consecutivi (o minimi) si chiama **lunghezza d'onda**, che è universalmente indicata dalla lettera greca λ (lambda).

Collegando un'antenna di dimensioni appropriate a un circuito elettrico, le onde elettromagnetiche possono essere trasmesse efficientemente e ricevute da un ricevitore a una certa distanza. Tutta la comunicazione senza filo si basa su questo principio.

Nel vuoto, tutte le onde elettromagnetiche viaggiano alla stessa velocità, non importa la loro frequenza. Questa velocità, detta di solito **velocità della luce**, c , è approssimativamente 3×10^8 m/s, o all'incirca un piede (30 cm) al nanosecondo. Nei fili di rame o in

fibra la velocità rallenta fino a 2/3 di questo valore e diventa parzialmente dipendente dalla frequenza. La velocità della luce è il limite di velocità estremo. Nessun oggetto o segnale si può muovere più velocemente.

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

$$\lambda f = c \quad (2-2)$$

Poiché c è una costante, se conosciamo f possiamo trovare λ e viceversa. Per esempio, le onde di 1 MHz sono lunghe circa 300 m e le onde di 1 cm hanno una frequenza di 30 GHz.

Lo spettro elettromagnetico è mostrato in figura 2.11. Le porzioni di radio, microonde, infrarossi e luce visibile dello spettro possono tutte essere usate per trasmettere informazione modulando l'ampiezza, la frequenza o la fase delle onde. La luce ultravioletta, i raggi X e gamma sarebbero ancora meglio per le loro più alte frequenze, ma sono difficili da produrre, modulare, non si propagano bene attraverso gli edifici e sono pericolosi per gli esseri viventi. Le bande elencate alla fine della figura 2.11 sono i nomi ufficiali ITU e si basano sulla lunghezza delle onde, così la banda LF va da 1 km a 10 km (approssimativamente da 30 kHz a 300 kHz). I termini LF, MF e HF si riferiscono rispettivamente a bassa, media e alta frequenza. Chiaramente, quando i nomi furono assegnati, nessuno si aspettava di andare al di sopra di 10 MHz, così le bande più alte furono più tardi chiamate Very (VHF), Ultra (UHF), Super (SHF), Extremely (EHF) e Tremendously High Frequency (THF). Non esistono altri nomi, ma Incredibly, Astonishingly and Prodigiou-sly High Frequency (IHF, AHF e PHF) suonerebbero bene.

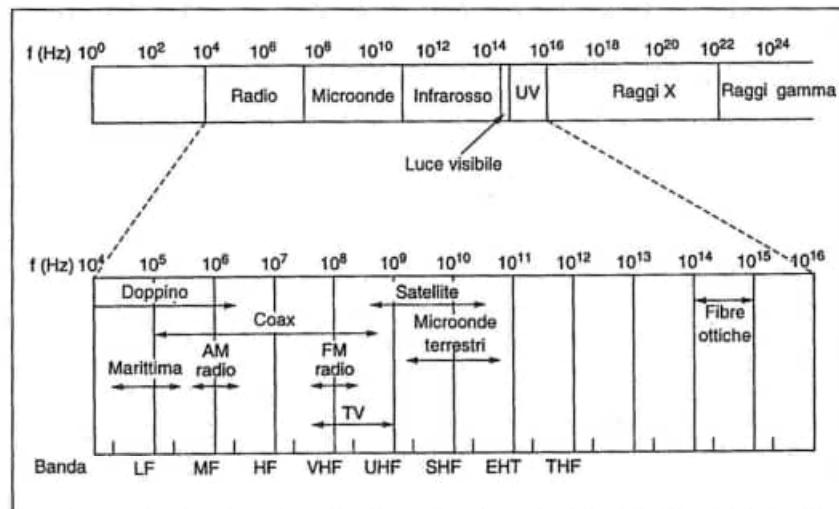


Fig 2.11 Lo spettro elettromagnetico e il suo uso per la comunicazione.

La quantità di informazione che può trasportare un'onda elettromagnetica è relativa alla sua larghezza di banda. Con la tecnologia attuale, è possibile codificare pochi bit per Hertz a basse frequenze, ma fino a 40 sotto certe condizioni ad alte frequenze, quindi un cavo con una banda di 500 MHz può trasportare diversi gigabit/s. Dalla figura 2-11 dovrebbe risultare chiaro perché gli utenti della rete gradiscono tanto le fibre ottiche. Se risolviamo l'equazione. (2-2) in f e differenziamo rispetto a λ otteniamo

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

Se ora andiamo alle differenze finite invece dei differenziali e guardiamo solo ai valori assoluti, otteniamo

$$\Delta f = \frac{c \Delta \lambda}{\lambda^2} \quad (2-3)$$

Cioè, data l'ampiezza della banda di lunghezza d'onda, $\Delta\lambda$, possiamo calcolare la banda corrispondente di frequenza, Δf , e da questa la velocità di dati che la banda può produrre. Più grande è la banda, maggiore è la velocità dei dati. Per esempio, consideriamo la banda di 1,30 μ di figura 2-6. Qui abbiamo

$$\lambda = 1,3 \times 10^{-6} \text{ e } \Delta\lambda = 1,7 \times 10^{-6}$$

quindi Δf è circa 30 THz.

Per evitare il caos totale, esistono accordi nazionali e internazionali per assegnare i permessi di usare le frequenze. Poiché tutti desiderano un maggiore tasso di dati, tutti vogliono più spettro. Negli Stati Uniti, la FCC assegna porzioni di spettro per radio AM ed FM, televisione e telefoni cellulari, così come per società telefoniche, polizia, marittimi, navigazione, militari, governo e molti altri utenti in concorrenza. In tutto il mondo, una agenzia della ITU-R (WARC) fa questo lavoro. Nell'incontro in Spagna del 1991, per esempio, WARC assegnò un po' di spettro agli strumenti portatili di comunicazione personale. Sfortunatamente, la FCC che non è vincolata dalle raccomandazioni della WARC, scelse una parte di spettro differente (perché coloro che negli Stati Uniti occupavano la banda WARC scelsero di non abbandonarla ed ebbero abbastanza potere politico per evitare di perderla). Di conseguenza, i comunicatori personali costruiti per il mercato USA non funzioneranno in Europa o Asia e viceversa.

La maggior parte delle trasmissioni usa una banda di frequenza ristretta (cioè $\Delta f/f \ll 1$) per ottenere una migliore ricezione (molti Watt/Hertz (W/Hz)). Comunque, in alcuni casi, il trasmettitore salta di frequenza in frequenza con uno schema regolare oppure le trasmissioni sono intenzionalmente diffuse su una banda di frequenza più larga. Questa tecnica è chiamata **spettro diffuso** (Kohno et al., 1995). È popolare per le comunicazioni militari poiché rende difficile rilevare le trasmissioni e quasi impossibile disturbarle. Il salto di frequenza non ci interessa molto (oltre a notare che fu coinventato dall'attrice

cinematografica Hedy Lamarr). Lo spettro diffuso reale, qualche volta chiamato **spettro diffuso a sequenza diretta**, sta guadagnando popolarità nel mondo commerciale e ci ritorneremo nel capitolo 4. Per una affascinante e dettagliata storia della comunicazione a spettro diffuso, si veda Scholtz (1982).

Per il momento, supponiamo che tutte le trasmissioni usino una banda ristretta di frequenza. Ora discuteremo come vengono usate le varie parti dello spettro, iniziando dalla radio.

2.3.2 Trasmissione radio

Le onde radio sono facili da generare, possono viaggiare per lunghe distanze e penetrano facilmente negli edifici, cosicché sono largamente usate per la comunicazione sia interna che esterna. Le onde radio sono anche omnidirezionali; ciò significa che viaggiano dalla sorgente in tutte le direzioni, così il trasmettitore e il ricevitore non devono essere fisicamente allineati.

Qualche volta la radio omnidirezionale è un bene, ma qualche volta è un male. Negli anni settanta, la General Motors decise di equipaggiare le sue nuove Cadillac con freni anti-bloccaggio controllati da computer. Quando il guidatore premeva il pedale del freno, il computer attivava e disattivava i freni invece di bloccarli bruscamente. Un bel giorno un poliziotto stradale dell'Ohio cominciò a usare la sua radio mobile per chiamare l'ufficio centrale e improvvisamente la Cadillac a lui vicina cominciò a comportarsi come un cavallo che sgroppava. Quando l'ufficiale si accostò alla macchina, il guidatore affermò che egli non aveva fatto niente e che la macchina era impazzita.

Alla fine cominciò a emergere uno schema: le Cadillac qualche volta sarebbero montate su tutte le furie, ma solo sulle principali autostrade dell'Ohio e quando la polizia stradale stava a guardare. Per lungo, lungo tempo, la General Motors non riuscì a capire perché le Cadillac si comportavano bene in tutti gli altri stati, e anche sulle strade minori in Ohio. Solo dopo una considerevole quantità di ricerche scoprirono che i fili delle Cadillac costituivano una buona antenna per la frequenza del sistema radio usato dalla polizia stradale dell'Ohio.

Le proprietà delle onde radio dipendono dalla frequenza. A basse frequenze, le onde radio passano bene attraverso ostacoli, ma la potenza decade precisamente con la distanza, circa come $1/r^3$ nell'aria. Ad alte frequenze, le onde radio tendono a viaggiare in linea retta e rimbalzano sugli ostacoli. A tutte le frequenze, le onde radio sono soggette a interferenza con i motori e con altre apparecchiature elettriche. Grazie alla capacità della radio di viaggiare a lunghe distanze, l'interferenza fra gli utenti è un problema. Per questo motivo, tutti i governi concedono avaramente la licenza agli utenti di trasmettitori radio, con una eccezione (discussa sotto).

Nelle bande VLF, LF, e MF, le onde radio seguono il terreno, come è illustrato nella figura 2-12 (a). Queste onde possono essere rilevate per circa 1000 km alle frequenze più basse, meno a quelle più alte. La trasmissione radio AM usa la banda MF, il che spiega perché le stazioni radio AM di Boston non si possono facilmente ascoltare a New York. Le onde radio in queste bande passano facilmente attraverso gli edifici, il che spiega perché le radio portatili funzionano all'interno. Il problema principale con queste bande per la comunicazione dati è l'ampiezza di banda relativamente bassa che offrono (vedi eq. (2-2)).

2.3 La trasmissione senza filo

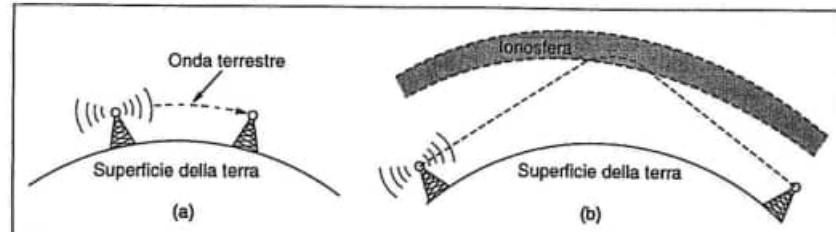


Fig. 2-12 (a) Nelle bande VLF, VF e MF le onde radio seguono la curvatura terrestre. (b) In HF le onde rimbalzano sulla ionosfera.

Nelle bande HF e VHF, le onde tendono a essere assorbite dal suolo.

Tuttavia le onde che raggiungono la ionosfera, uno strato di particelle cariche che circonda la Terra a una altezza da 100 a 500 km, sono rifatte e rispedite alla Terra, come mostrato in figura 2-12 (b). Sotto certe condizioni atmosferiche, i segnali possono rimbalzare parecchie volte. I radioamatori usano queste bande per comunicare a lunga distanza. Anche i militari comunicano sulle bande HF e VHF.

2.3.3 Trasmissione a microonde

Sopra i 100 MHz, le onde viaggiano in linea retta e perciò possono essere fortemente direzionate. Concentrando tutta l'energia in un piccolo raggio usando un'antenna parabolica (come un disco per la TV satellitare) si ha un rapporto segnale/rumore molto migliore, ma le antenne riceventi e trasmissenti devono essere accuratamente allineate. Inoltre, questa direzionalità permette a trasmettitori multipli allineati in fila di comunicare con ricevitori multipli in fila senza interferenza. Prima delle fibre ottiche, per decenni queste microonde hanno formato il cuore del sistema di trasmissione telefonica a lunga distanza. Infatti, il nome della società MCI stava per Microwave Communications Inc., poiché l'intero sistema era originariamente costruito su torri a microonde (da allora le parti principali della rete sono state sostituite da fibre).

Poiché le microonde viaggiano in linea retta, se due torri sono troppo distanti, la terra sarà un ostacolo (si pensi a un collegamento da San Francisco ad Amsterdam). Di conseguenza servono dei ripetitori. Più alte sono le torri, più distanti possono essere. La distanza fra i ripetitori cresce all'incirca come la radice quadrata dell'altezza delle torri. Per torri alte 100 m, i ripetitori possono essere distanziati di 80 km.

A differenza delle onde radio a bassa frequenza, le microonde non attraversano bene gli edifici. Inoltre, anche se il raggio può essere ben direzionato sul trasmettitore, esiste comunque un po' di dispersione nello spazio. Alcune onde possono essere rifatte dagli strati bassi dell'atmosfera e possono metterci un po' di più ad arrivare delle onde dirette. Le onde ritardate possono arrivare fuori fase rispetto alle onde dirette e così cancellano il segnale. Questo effetto è chiamato **multipath fading** ed è spesso un problema serio. Esso dipende dal tempo e dalla frequenza. Alcuni operatori mantengono libero il 10% dei propri canali come riserva su cui passare quando il multipath fading cancella temporaneamente alcune bande di frequenza.

La richiesta di sempre maggiore spettro induce a migliorare la tecnologia, infatti le trasmissioni tendono a usare frequenze sempre più alte. Le bande fino a 10 GHz sono adesso di uso comune, ma fino a circa 8 GHz si pone un nuovo problema: l'assorbimento da parte dell'acqua. Queste onde sono lunghe solo pochi centimetri e sono assorbite dalla pioggia. Questo effetto sarebbe utile se uno progettasse di costruire un enorme forno a microonde all'aperto, ma per la comunicazione costituisce un problema difficile. Come col multipath fading, l'unica soluzione è chiudere i collegamenti sotto la pioggia e trovare un percorso alternativo.

Riassumendo, la comunicazione a microonde è così tanto sfruttata per la comunicazione telefoni a lungo percorso, telefoni cellulari, trasmissioni televisive e altri usi, che si è sviluppata una grossa domanda di spettro. Essa ha molti vantaggi sulle fibre. Il principale è che non ha bisogno di diritti di passaggio e comprando un piccolo pezzo di terreno ogni 50 km e piazzandovi sopra una torre a microonde, si può saltare il sistema telefonico e comunicare direttamente. Questo è il modo in cui la MCI è riuscita a partire così rapidamente come società telefonica a lunga distanza (Sprint usò un sistema diverso: venne creata dalla Southern Pacific Railroad che già possedeva una grossa quantità di diritti di passaggio e interrò fibre vicino ai binari).

Le microonde sono anche relativamente poco costose. Mettere insieme due semplici torri (possono essere solo due grossi pali con quattro tiranti) e piazzare antenne su ciascuno può essere più economico di interrare 50 km di fibra attraverso un'area urbana congestionata o su una montagna, e può essere anche più economico che affittare la fibra di una società telefonica, specialmente se questa non ha ancora pagato tutto il rame che ha strappato quando ha messo la fibra.

Oltre ad essere usate per trasmissioni a lunga distanza, le microonde hanno un altro importante utilizzo, cioè le bande **industriali-scientifiche-mediche**. Queste bande sono l'unica eccezione alle regole di rilascio della licenza: i trasmettitori che le usano non hanno bisogno della licenza del governo. In tutto il mondo è allocata la banda 2,400-2,484 GHz. Inoltre, negli Stati Uniti e in Canada sono assegnate anche le bande 902-928 MHz e 5,725-5,850 GHz. Tali bande sono usate per i telefoni portatili, telecomandi delle porte dei garage, altoparlanti senza fili, cancelli di sicurezza ecc. La banda a 900 MHz funziona meglio ma è affollata e le apparecchiature che la usano possono funzionare solo in Nord America. Le bande più alte richiedono elettronica più costosa e sono soggetti a interferenze con i fornì a microonde e con le installazioni radar. Non di meno, queste bande sono diffuse per varie forme di reti senza cavo a piccola distanza perché evitano il problema della licenza.

2.3.4 Onde infrarosse e millimetriche

Le onde a infrarossi e millimetriche non guidate sono ampiamente usate per comunicazioni a piccola distanza. I telecomandi usati per le televisioni, i videoregistratori e gli stereo usano tutti comunicazioni infrarosse. Sono relativamente direzionali, economici e facili da costruire, ma hanno un difetto principale: non passano attraverso oggetti solidi (provate a stare fra il telecomando e il televisore e vedete se funziona ancora). In generale, man mano che ci spostiamo sullo spettro dalle onde radio lunghe verso la luce visibile, le onde si comportano sempre più come luce e sempre meno come radio.

D'altra parte, il fatto che le onde infrarosse non passino bene attraverso muri solidi è anche un vantaggio. Significa che un sistema a infrarossi in una stanza non interferirà con un sistema simile in una stanza adiacente. Inoltre, la sicurezza di un sistema a infrarossi contro le intercettazioni è migliore di quella di un sistema radio proprio per questo motivo. Per queste ragioni, non occorre nessuna licenza governativa per far funzionare un sistema a infrarossi, al contrario dei sistemi radio che devono avere una licenza.

Queste proprietà hanno reso l'infrarosso un candidato interessante per LAN interne senza cavo. Per esempio i computer e gli uffici in un edificio possono essere equipaggiati con trasmettitori e ricevitori a infrarossi relativamente non direzionali (cioè in qualche modo omnidirezionali). In questo modo i computer portatili con capacità a infrarossi possono entrare sulla LAN locale senza connettersi fisicamente. Quando molte persone si incontrano per una riunione, possono sedersi nella sala conferenza ed essere pienamente connesse, senza doversi collegare fisicamente. La comunicazione a infrarossi non può essere usata all'esterno perché il sole splende negli infrarossi tanto quanto nello spettro visibile. Per maggiori informazioni sulla comunicazione a infrarossi si vedano Adams et al. (1993); Bantz, Bauchot (1994).

2.3.5 Trasmissione a onde luminose

I segnali ottici non guidati sono stati in uso per secoli. Paul Revere usò una segnalazione ottica binaria dalla Vecchia Chiesa del Nord proprio prima della sua famosa cavalcata. Un'applicazione più moderna è di connettere le LAN in due edifici attraverso dei laser montati sui tetti. La segnalazione ottica coerente usando dei laser è inerentemente unidirezionale, così ogni edificio ha bisogno del suo laser e del suo fotorilevatore. Questo schema offre una grande larghezza di banda ed è molto poco costoso. È anche relativamente facile da installare e a differenza delle microonde non richiede una licenza della FCC.

La potenza del laser, un raggio molto sottile, è anche la sua debolezza. Puntare un raggio laser largo 1 mm su un bersaglio di 1 mm a 500 m di distanza richiede un'abilità nel tiro pari a quella di Annie Oakley. Di solito, nel sistema si mettono delle lenti per sfocare leggermente il raggio.

Uno svantaggio è che il raggio laser non può passare attraverso la pioggia o una fitta nebbia, ma può lavorare normalmente bene nei giorni di sole. Comunque, l'autore una volta andò a una conferenza in Europa alla quale gli organizzatori premurosamente fornirono una stanza piena di terminali per i partecipanti per leggere la loro posta elettronica durante le presentazioni noiose. Poiché le poste locali non volevano installare un grande numero di linee telefoniche per solo tre giorni, gli organizzatori misero un laser sul tetto e lo puntarono verso l'edificio del Dipartimento di informatica a pochi chilometri di distanza. Lo provarono la notte prima e funzionava perfettamente. Alle 9 della mattina successiva, in un luminoso giorno di sole, la connessione non funzionava affatto e fu giù per tutto il giorno. Quella sera, gli organizzatori la provarono di nuovo molto attentamente e una volta ancora funzionava in modo assolutamente perfetto. Lo schema si ripeté per due giorni allo stesso modo. Dopo la conferenza, gli organizzatori scoprirono il problema. Il calore del sole durante il giorno causava correnti di convezione che si sollevavano dal tetto dell'edificio, come in Fig 2-13. Quest'aria turbolenta deviava il raggio e lo faceva

ballare intorno al rilevatore. Effetti atmosferici come questi fanno scintillare le stelle (ecco perché gli astronomi piazzano i loro telescopi sulla cima delle montagne, per andare più in alto possibile nell'atmosfera). Sono anche causa del tremolio delle strade in un giorno caldo e delle immagini tremolanti quando si guarda vicino a un radiatore bollente.

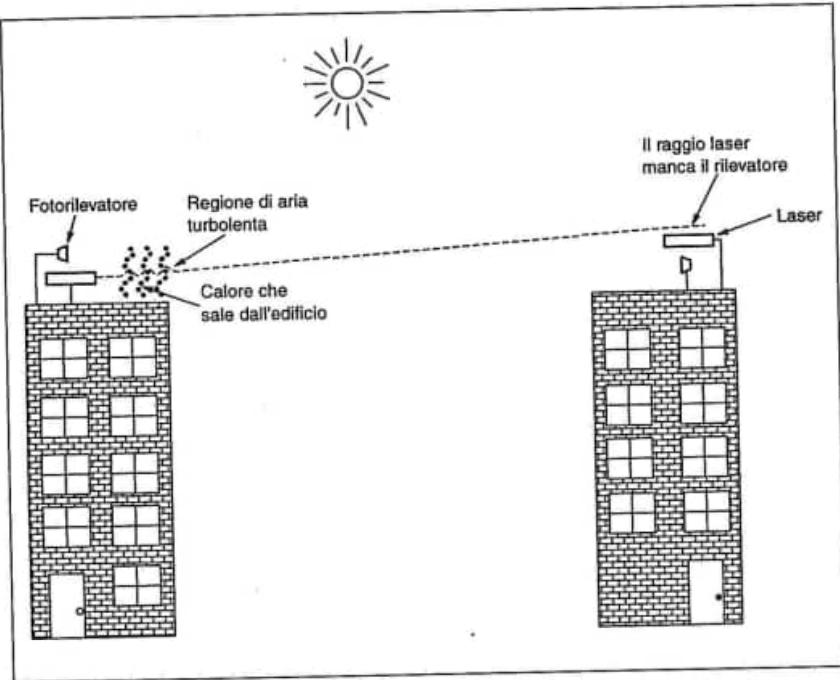


Fig 2-13 Le correnti di convezione possono interferire con i sistemi di comunicazione laser. Qui è illustrato un sistema bidirezionale con due laser.

2.4 Il sistema telefonico

Quando due computer posseduti dalla stessa società od organizzazione e collocati uno vicino all'altro hanno bisogno di comunicare, spesso la cosa più facile è di connetterli con un cavo. Le LAN funzionano in questo modo. Tuttavia, quando le distanze sono grandi, o ci sono molti computer, o i cavi dovrebbero passare attraverso una strada pubblica o altro, i costi di stesura di cavi privati sono di solito proibitivi. Inoltre, in quasi tutti i paesi nel mondo, stendere delle linee di trasmissione private attraverso (o sotto) la proprietà pubblica è illegale. Di conseguenza, i progettisti di reti devono contare sulle attrezzature per la telecomunicazione esistenti.

Queste attrezzature, specialmente la PSTN, (rete telefonica pubblica commutata), furono di solito progettate molti anni fa, con un obiettivo completamente diverso: trasmettere la voce umana in una forma più o meno riconoscibile. La loro adattabilità per l'utili-

2.4 Il sistema telefonico

lizzo nella comunicazione fra computer e computer è spesso tutt'al più marginale, ma la situazione sta cambiando con l'introduzione di fibre ottiche e tecnologia digitale. In ogni caso, il sistema telefonico è così strettamente interconnesso con le reti di computer (area geografica) che vale la pena di dedicare un tempo considerevole al suo studio.

Per comprendere la dimensione del problema, facciamo un confronto approssimativo ma illuminante delle proprietà di una tipica connessione fra computer e computer via cavo locale e via linea telefonica commutata. Un cavo fra due computer può trasferire dati a velocità di memoria, di solito da 10^7 a 10^8 bps. Il tasso di errore è di solito così basso da essere difficile da misurare, ma un errore al giorno sarebbe considerato nella media della maggior parte delle installazioni. Un errore al giorno a queste velocità è equivalente a un errore ogni 10^{12} o 10^{13} bit spediti.

Al contrario, una linea commutata ha una velocità massima di dati dell'ordine di 10^4 bps e un tasso di errore di 1 ogni 10^5 bit spediti, con oscillazioni dovute all'età dell'apparecchiatura di commutazione telefonica considerata. La prestazione combinata di velocità di bit per tasso di errore di un cavo locale è quindi 11 ordini di grandezza migliore di una linea telefonica vocale. Per fare una analogia nel campo dei trasporti, il rapporto fra il costo dell'intera missione Apollo, che fece atterrare gli uomini sulla Luna, e il costo di una corsa di un autobus in città è di circa 11 ordini di grandezza (in dollari del 1965: 40.000.000.000 contro 40 centesimi).

Il problema, naturalmente, è che i progettisti informatici lavorano di solito con sistemi di computer, e quando improvvisamente si confrontarono con un altro sistema le cui prestazioni (dal loro punto di vista) erano 11 ordini di grandezza peggiori, non è sorprendente che molto tempo e molti sforzi siano stati dedicati a cercare di capire come usarlo efficacemente. D'altra parte, le società telefoniche hanno fatto grandi progressi nell'ultimo decennio per aggiornare le apparecchiature e migliorare il servizio in certe aree. Nei paragrafi successivi descriveremo il sistema telefonico e mostreremo cos'era una volta e cosa sta diventando. Per informazioni aggiuntive sul sistema telefonico si veda Bellamy (1991).

2.4.1 Struttura del sistema telefonico

Quando Alexander Graham Bell brevettò il telefono nel 1876 (solo poche ore prima del suo rivale, Elisha Gray), ci fu un'enorme richiesta per la sua nuova invenzione. Il mercato iniziale fu per la vendita dei telefoni che si compravano a coppia. Stava al cliente disporre un singolo filo fra loro. Gli elettroni ritornavano attraverso la terra. Se un possessore di un telefono voleva parlare a n possessori di telefono, si dovevano stendere fili separati verso ognuna delle n case. Entro un anno, le città furono ricoperte da fili che passavano sulle case e sugli alberi in un groviglio selvaggio. Divenne immediatamente ovvio che il modello di connessione di tutti a tutti, mostrato in figura 2-14 (a), non avrebbe funzionato.

A suo merito, va detto che Bell previde questo fatto e fondò la Bell Telephone Company, che aprì il suo primo ufficio di commutazione a New Haven, Connecticut, nel 1878. La società fece correre un filo alla casa o all'ufficio di ogni cliente. Per fare una chiamata, il cliente girava la manovella del telefono per fare uno squillo nell'ufficio della società telefonica e richiamare l'attenzione di un operatore, che poi manualmente connetteva il

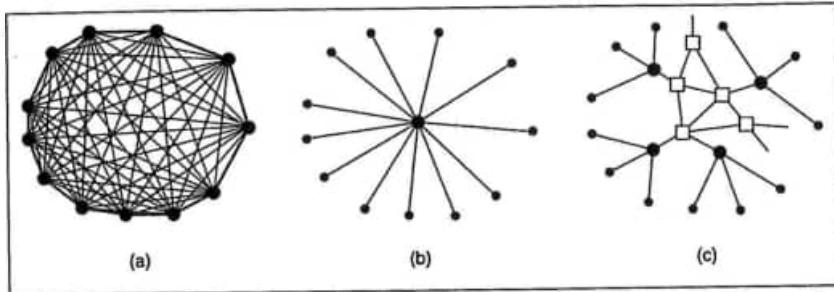


Fig. 2-14 (a) Rete pienamente interconnessa. (b) Comutazione combinata. (c) Gerarchia a due livelli.

chiamante col chiamato usando un cavo di giunzione. Il modello di un ufficio a singola commutazione è illustrato in figura 2-14 (b).

Molto presto, uffici di commutazione della Bell System sorsero ovunque. La gente voleva fare chiamate fra città a lunga distanza, così la Bell System cominciò a connettere gli uffici di commutazione. Il problema originale si ripresentò: connettere ogni ufficio di commutazione all'altro per mezzo di un filo divenne rapidamente insostenibile, così furono inventati uffici di secondo livello. Dopo un po' di tempo, furono necessari uffici multipli di secondo livello, come in figura 2-14 (c). Alla fine, la gerarchia crebbe fino a cinque livelli.

A partire dal 1890, le tre componenti principali del sistema telefonico furono pronte: i centralini, i cavi fra i clienti e i centralini (da allora doppini bilanciati, isolati invece di fili aperti col ritorno a terra), e le connessioni a lunga distanza tra i centralini. Anche se da allora in tutte e tre le aree ci sono stati dei miglioramenti, il modello Bell System rimase essenzialmente invariato per più di cento anni. Per una breve storia tecnica del sistema telefonico si veda Hawley (1991).

Allo stato attuale, il sistema telefonico è organizzato in una gerarchia multilivello altamente ridondante. La seguente descrizione è molto semplificata ma non di meno ne mostra l'essenza. Ogni telefono ha due fili di rame che escono e vanno direttamente al più vicino **ufficio terminale** (end office) della società telefonica (detto anche **ufficio centrale locale**). La distanza è tipicamente da 1 a 10 km, minore in città rispetto alle aree rurali.

Solo negli Stati Uniti ci sono circa 19.000 uffici terminali. La concatenazione del codice dell'area e le prime tre cifre del numero telefonico specificano univocamente un ufficio terminale, il che spiega perché la struttura tariffaria usa questa informazione. Le connessioni con due fili fra ogni abbonato telefonico e l'ufficio terminale sono note commercialmente come **circuiti locali**. Se i circuiti locali del mondo fossero allungati da una estremità all'altra, si estenderebbero fino alla Luna in andata e ritorno per mille volte. In un certo periodo, l'80% del capitale dalla AT&T era costituito dal rame dei circuiti locali. La AT&T era allora, in effetti, la più grande miniera di rame al mondo. Fortunatamente questo fatto non fu largamente risaputo nella comunità di investimenti. Se fosse

stato noto, qualche scalatore di società avrebbe potuto acquistare la AT&T, strappar via tutti i fili e venderli a un raffinatore di rame per ottenere un rapido guadagno. Se un abbonato collegato a un ufficio terminale chiama un altro abbonato collegato allo stesso ufficio, il meccanismo di commutazione all'interno dell'ufficio predispone una connessione elettrica diretta fra i due circuiti locali. Questa connessione rimane intatta per la durata della chiamata.

Se il telefono chiamato è collegato a un altro ufficio terminale, si deve usare una procedura diversa. Ogni ufficio terminale ha un numero di linee uscenti verso uno o più centri di commutazione vicini, chiamati **uffici di pedaggio** (o se sono all'interno della stessa area locale **uffici tandem**). Queste linee sono chiamate **dorsali di connessione di pedaggio**. Se capita che entrambi gli uffici terminale del chiamante e del chiamato abbiano una dorsale di pedaggio allo stesso ufficio di pedaggio (un evento probabile se sono relativamente vicini), la connessione può essere stabilita all'interno dell'ufficio. In figura 2-14 (c) è illustrata una rete telefonica consistente solo di telefoni (i punti piccoli), di uffici terminali (i punti grandi) e di uffici di pedaggio (i quadrati).

Se il chiamato e il chiamante non hanno un ufficio di pedaggio in comune, il percorso deve essere stabilito in qualche posto più in alto nella gerarchia. Ci sono uffici primari, di sezione, regionali, che formano una rete attraverso la quale sono connessi gli uffici di pedaggio. Gli scambi di pedaggio, primari, di sezione e regionali comunicano uno con l'altro per mezzo di **dorsali interpedaggio ad alta banda** (chiamati anche **dorsali interufficio**). Il numero dei diversi tipi di uffici di commutazione e la loro topologia (ad es. due uffici di sezione hanno una connessione diretta o devono passare attraverso un ufficio regionale?) varia da paese a paese a seconda della densità telefonica. La figura 2-15 mostra come può essere instradata una connessione a media distanza.



Fig. 2-15 Diversi mezzi di trasmissione sono usati per la telecomunicazione.

I circuiti locali consistono al giorno d'oggi di doppini, sebbene nei primi giorni della telefonia fossero comuni fili non isolati distanziati di 25 cm sui pali telefonici. Tra gli uffici di commutazione si usano cavi coassiali, microonde e specialmente fibre ottiche. In passato, la segnalazione attraverso il sistema telefonico era analogica, con il segnale vocale presente trasmesso come voltaggio elettrico dalla sorgente alla destinazione. Con l'avvento dell'elettronica digitale e dei computer è diventata possibile la segnalazione digitale. In questo sistema sono permessi solo due voltaggi, ad esempio -5 e +5 V. Questo schema ha diversi vantaggi sulla segnalazione analogica. Il primo è che, sebbene

l'attenuazione e la distorsione siano più forti quando si spediscono segnali a due livelli che quando si usano dei modem, è più facile calcolare quanto lontano un segnale si può propagare restando riconoscibile. Un rigeneratore digitale può essere inserito nella linea per riportare il segnale al valore originario, poiché ci sono solo due possibilità. Un segnale digitale può passare attraverso un numero arbitrario di rigeneratori con nessuna perdita nel segnale e così percorrere lunghe distanze con nessuna perdita di informazione. Al contrario, i segnali analogici soffrono sempre di perdita di informazione quando sono amplificati, e questa perdita è cumulativa. Il risultato netto è che si può far avere un basso tasso di errore alla trasmissione digitale.

Un secondo vantaggio della trasmissione digitale è che si possono alternare voce, dati, musica e immagini (ad es. televisione, fax e video) per rendere più efficiente l'uso di circuiti e apparecchiature. Un altro vantaggio è che sono possibili velocità di dati molto più alte usando le linee esistenti.

Un terzo vantaggio è che la trasmissione digitale è più economica della trasmissione analogica, poiché non è necessario riprodurre accuratamente una forma d'onda dopo che questa è passata potenzialmente fra centinaia di amplificatori in una chiamata intercontinentale. Basta esser capaci di distinguere fra uno 0 e un 1. Infine, la manutenzione di un sistema digitale è più facile della manutenzione di un sistema analogico. Un bit trasmesso è ricevuto correttamente oppure no, e questo rende più semplice scovare i problemi.

Di conseguenza, si stanno rapidamente convertendo al digitale tutti le dorsali a lunga distanza nel sistema telefonico. Il vecchio sistema usava la trasmissione analogica su fili di rame; il nuovo usa la trasmissione digitale su fibre ottiche. Riassumendo, il sistema telefonico consiste di tre componenti principali:

1. Circuiti locali (doppini, segnalazione analogica).
2. Dorsali (fibre ottiche o microonde, principalmente digitali).
3. Centralini.

Dopo una breve digressione sulla politica dei telefoni, torneremo in dettaglio a ciascuna di queste tre componenti. Per il circuito locale, saremo interessati a come spedire dati digitali su di essi (risposta rapida: si usa un modem). Per le dorsali a lungo percorso, la questione principale è come raccogliere insieme molte chiamate e come spedirle insieme. Questo argomento è chiamato multiplexing, e studieremo tre modi diversi per affrontarlo. Infine, ci sono due diversi sistemi fondamentali per fare commutazione, così daremo uno sguardo a entrambi.

2.4.2 La politica dei telefoni

Per decenni, prima del 1984, la Bell System fornì servizi sia locali che a lunga distanza nella maggior parte degli Stati Uniti. Negli anni settanta, il governo americano si convinse che questo fosse un monopolio illegale e citò in giudizio la AT&T per romperlo. Il governo vinse, e il primo gennaio 1984 la AT&T fu divisa in AT&T Long Lines, 23 BOC (Bell Operating Companies) e un po' di altri pezzi. Le 23 BOC furono raggruppate insieme in 7 BOC regionali (RBOC) per renderle economicamente vitali. L'intera natura della

telecomunicazione negli Stati Uniti fu cambiata tutto a un tratto dall'ordine del tribunale (non da un atto del Congresso).

I dettagli esatti del giudizio furono descritti nel cosiddetto MFJ (Modified Final Judgment), un ossimoro se mai ce ne fosse uno (se il giudizio poteva essere modificato, chiaramente non era finale). Questo evento portò a una maggiore concorrenza, a un miglior servizio e a prezzi più bassi per consumatori e affari. Molti altri paesi stanno ora considerando di introdurre la concorrenza in modo analogo.

Per chiarire chi poteva farlo, gli Stati Uniti furono divisi in circa 160 LATA (Local Access and Transport Areas). Dentro ogni LATA, normalmente c'è una LEC (Local Exchange Carrier) che ha il monopolio sui servizi telefonici tradizionali all'interno della LATA. Le LEC più importanti sono le BOC, sebbene qualche LATA contenga una o più delle 1500 società telefoniche indipendenti che operano come LEC. Nelle LATA geograficamente più grandi (nel West) le LEC possono gestire chiamate a lunga distanza nella propria LATA ma non possono gestire chiamate che vanno a una LATA diversa.

Tutto il traffico inter-LATA è gestito da un altro tipo di società, una IXC (Inter Exchange Carrier). Originariamente, le lunghe linee della AT&T erano le uniche IXC, ma ora MCI e SPRINT sono concorrenti ben affermate nell'affare IXC. Una delle questioni al momento della rottura era di assicurare che tutte le IXC fossero trattate equamente in termini di qualità di linea, tariffe e numero di cifre che l'utente avrebbe dovuto digitare per usarle. Questo sistema di gestione è illustrato in figura 2-16. Qui vediamo tre esempi di LATA, ognuno con diversi uffici terminali. Le LATA 2 e 3 hanno anche una piccola gerarchia con uffici tandem (uffici di pedaggio intra-LATA).

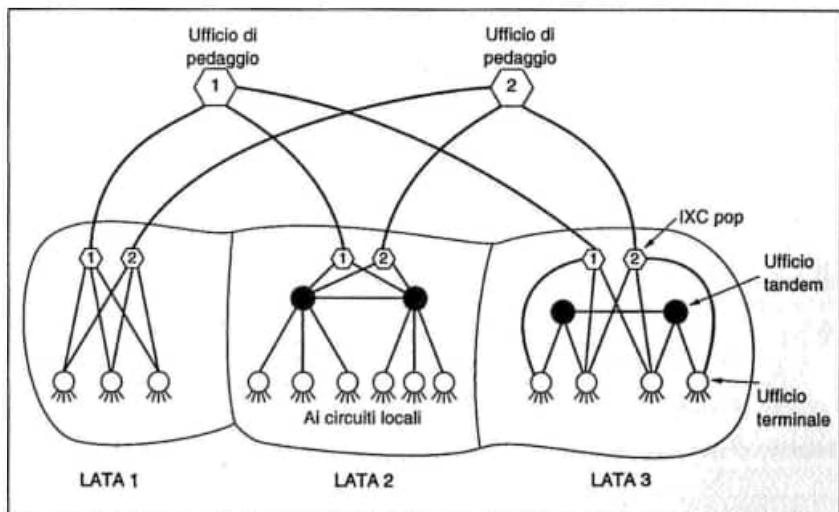


Fig. 2-16 La relazione fra LATA, LEC e IXC. Tutti i cerchi sono uffici di commutazione LEC. Ogni esagono appartiene all'IXC contraddistinta dal proprio numero.

Ogni IXC che desideri gestire le chiamate che si originano in una LATA può costruire lì un ufficio di commutazione chiamato **POP (Point of Presence)**. Alle LEC è richiesto di connettere ogni IXC a ogni ufficio terminale, o direttamente come nelle LATA 1 e 3, o indirettamente come nella LATA 2. Inoltre i termini della connessione, sia tecnici che finanziari, devono essere identici per tutte le IXC. In questo modo, un abbonato, diciamo nella LATA 1, può scegliere quale IXC usare per chiamare gli abbonati nella LATA 3.

Come parte del MFJ, alle IXC fu proibito di offrire servizio telefonico locale e alle LEC fu proibito di offrire servizio telefonico inter-LATA, sebbene entrambe fossero libere di entrare in altri settori industriali, come per esempio ristoranti che producono pollo arrosto. Nel 1984, quella fu una dichiarazione giustamente non ambigua.

Sfortunatamente, la tecnologia ha un modo speciale di rendere la legge obsoleta. Né la televisione via cavo, né i telefoni cellulari furono previsti dall'accordo. Quando la televisione via cavo da passiva diventò interattiva, e i telefoni cellulari esplosero in popolarità, sia le LEC che le IXC cominciarono a comprare o a fondersi con operatori di cavo e cellulari. Nel 1995, il Congresso vide che cercare di mantenere una distinzione fra le varie specie di società non era più a lungo sostenibile e approvò una legge per permettere alle società di TV via cavo, alle società di telefoni locali, ai trasporti a lunga distanza e agli operatori cellulari di entrare uno nel settore dell'altro. L'idea era che ogni società avrebbe potuto offrire ai suoi clienti un singolo pacchetto integrato contenente TV via cavo, telefoni e servizi di informazione, e che società diverse sarebbero state in concorrenza per servizi e prezzi. La legge entrò in vigore nel febbraio 1996. Come risultato, il paesaggio delle telecomunicazioni degli USA è attualmente sotto una radicale ristrutturazione.

2.4.3 Il circuito locale

Nell'ultimo secolo la trasmissione analogica ha dominato tutta la comunicazione. In particolare, il sistema telefonico era originariamente basato interamente sulla segnalazione analogica. Mentre le linee interurbane a lunga distanza sono adesso in gran parte digitali nei paesi più avanzati, i circuiti locali sono ancora analogici e probabilmente rimarranno così

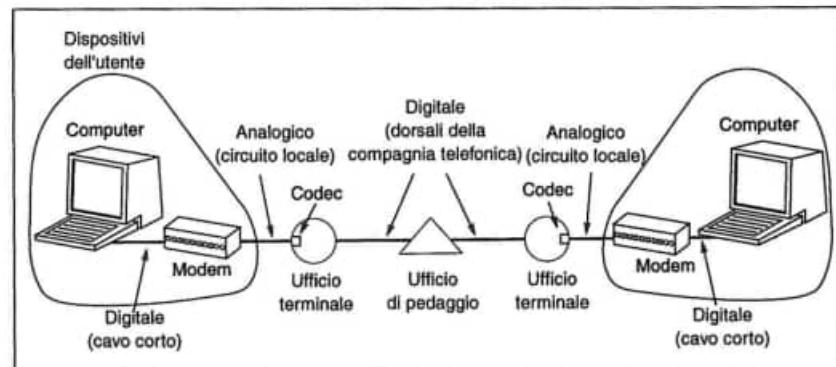


Fig. 2-17 L'uso di trasmissione sia analogica che digitale per una chiamata da computer a computer.

per almeno uno o due decenni, a causa dell'enorme costo per convertirle. Di conseguenza, quando un computer vuole mandare dei dati digitali su una linea commutata, i dati devono essere prima convertiti in forma analogica per la trasmissione sul circuito locale, poi convertiti in forma digitale per la trasmissione sulle linee a lungo percorso, e poi indietro in analogico sul circuito locale al terminale ricevente, e finalmente di nuovo in digitale da un altro modem per la memorizzazione nel computer di destinazione. Questo sistema è mostrato in figura 2-17.

La conversione è fatta dai modem e dai codec. Anche se questa situazione non è ideale, così è al momento, e gli studenti di reti dovrebbero avere qualche conoscenza di trasmissione sia analogica che digitale, così come delle funzioni di conversione. Con le linee in affitto è possibile andare in digitale dall'inizio alla fine, ma sono costose e utili solo per reti private di una società. Nei paragrafi seguenti esamineremo i problemi della trasmissione analogica e vedremo come i modem rendano possibile trasmettere dati digitali su circuiti analogici. Daremo uno sguardo anche a due comuni interfacce di modem, RS-232-C ed RS-449.

Danneggiamenti della trasmissione

La segnalazione analogica consiste nel modulare il voltaggio nel tempo per rappresentare un flusso di informazione. Se il mezzo di trasmissione fosse perfetto, il ricevitore riceverebbe esattamente lo stesso segnale che il trasmettitore invia. Sfortunatamente, i mezzi non sono perfetti, così il segnale ricevuto non è lo stesso del segnale trasmesso. Per i dati digitali questa differenza può causare errori.

Le linee di trasmissione presentano tre problemi principali: attenuazione, distorsione di ritardo e rumore. L'**attenuazione** è la perdita di energia mentre il segnale si propaga all'esterno. Sui mezzi guidati (ad es. fili e fibre ottiche), il segnale decade logaritmicamente con la distanza. La perdita è espressa in decibel per chilometro. La quantità di energia perduta dipende dalla frequenza. Per capire l'effetto di questa dipendenza dalla frequenza, immaginiamo un segnale non come una semplice forma d'onda, ma come una serie di componenti di Fourier. Ogni componente è attenuata da una diversa quantità, il che risulta in uno spettro di Fourier diverso al ricevitore, e quindi in un segnale diverso. Se l'attenuazione è eccessiva, il ricevitore può non essere capace nemmeno di rilevare il segnale, o il segnale può cadere sotto il livello del rumore. In molti casi, le proprietà di attenuazione di un mezzo sono note, quindi possono essere inseriti degli amplificatori per cercare di compensare l'attenuazione dipendente dalla frequenza. Tutto questo aiuta ma non può mai riportare il segnale esattamente alla sua forma originaria.

Il secondo danneggiamento di trasmissione è la **distorzione da ritardo**. È causata dal fatto che componenti di Fourier diverse viaggiano a diversa velocità. Per dati digitali, le componenti veloci di un bit possono raggiungere e superare le componenti lente di un bit che lo precede, mischiando i due bit e aumentando la probabilità di una ricezione non corretta.

Il terzo danneggiamento è il **rumore**, che è energia non voluta, prodotta da sorgenti diverse dal trasmettitore. Il rumore termico è causato dal movimento casuale di elettroni in un filo, ed è inevitabile. Il cross talk è causato dall'accoppiamento induttivo fra due fili che sono uno vicino all'altro. A volte quando si parla al telefono, si può ascoltare un'altra conversazione in sottofondo: questo è il cross talk. Infine, c'è il rumore di im-

pulso, causato da picchi sulla linea di corrente o altre cause. Nel caso di dati digitali, il rumore di impulso può cancellare uno o più bit.

Modem

A causa dei problemi già discussi, specialmente per il fatto che le velocità di attenuazione e di propagazione sono dipendenti dalla frequenza, non è desiderabile avere un grande intervallo di frequenze nel segnale. Sfortunatamente le onde quadrate come nei dati digitali hanno un'ampio spettro e così sono soggette a una forte attenuazione e distorsione da ritardo. Questi effetti rendono la segnalazione a banda base (DC) non adatta, tranne che a basse velocità e su distanze piccole.

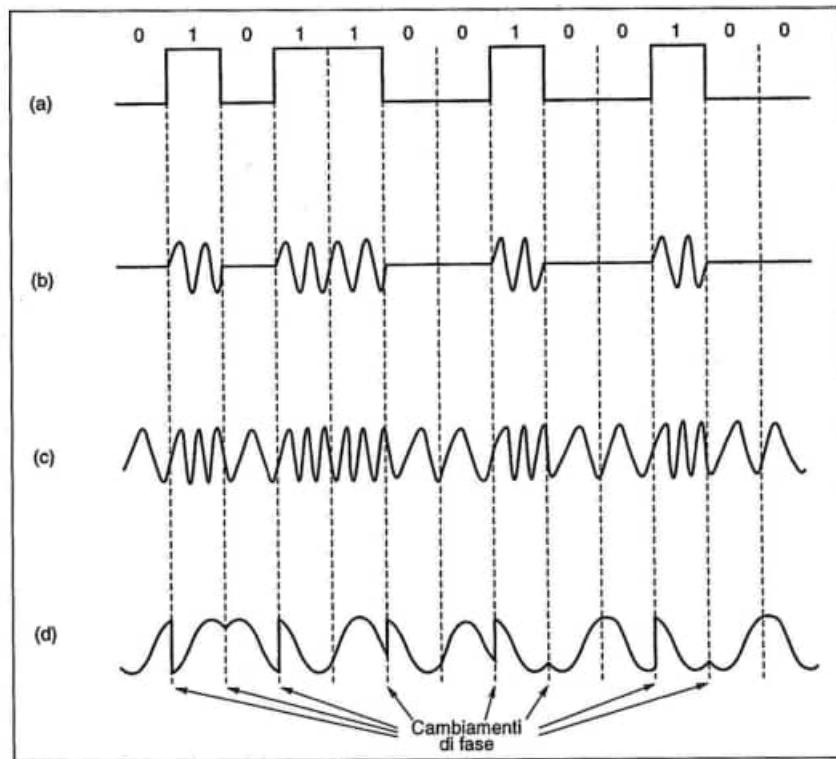


Fig. 2-18 (a) Un segnale binario. (b) Modulazione d'ampiezza. (c) Modulazione di frequenza. (d) Modulazione di fase.

Per aggirare i problemi associati con la segnalazione DC, specialmente sulle linee telefoniche, si usa la segnalazione AC. Si introduce un tono continuo nell'intervallo da 1000 a 2000 Hz, chiamata **portante sinusoidale**. La sua ampiezza, frequenza o fase può essere modulata

per trasmettere informazione. Nella **modulazione d'ampiezza**, sono usati due diversi livelli di tensione per rappresentare 0 e 1. Nella **modulazione di frequenza**, anche nota come **frequency shift keying**, si usano due (o più) toni. Nella più semplice forma della **modulazione di fase**, l'onda portante è sistematicamente spostata di 45, 135, 225, 315 gradi a intervalli uniformemente distanziati. Ogni fase trasmette due bit di informazione. La figura 2-18 illustra le tre forme di modulazione. Un dispositivo che accetta un flusso di bit come input e produce una portante modulata come output (o viceversa) è detto **modem** (modulatore-demodulatore). Il modem sta fra il computer (digitale) e il sistema telefonico (analogico). Per andare a velocità sempre più alte, non è possibile incrementare a piacere il tasso di campionamento. Il teorema di Nyquist dice che persino con una linea a 3000 Hz perfetta (non è decisamente il caso di una linea telefonica a combinazione), non c'è un punto nel campionamento più veloce di 6000 Hz. Così tutta la ricerca per modem più veloci è focalizzata sull'ottenere più bit per campione (cioè per baud).

I modem più avanzati usano una combinazione di tecniche di modulazione per trasmettere più bit per baud. In figura 2-19 (a) vediamo dei punti a 0, 90, 180, e 270°, con due livelli di ampiezza per cambiamento di fase. In figura 2-19 (b) vediamo un diverso schema di modulazione, nel quale si usano sedici diverse combinazioni di ampiezza e di spostamento di fase. Così la figura 2-19 (a) ha otto combinazioni valide e può essere usata per trasmettere 3 bit per baud. Al contrario, la figura 2-19 (b) ha sedici combinazioni valide e può così essere usata per trasmettere 4 bit per baud. Lo schema di figura 2-19 (b) quando usato per trasmettere su una linea a 2400 bd è chiamato **QAM (Quadrature Amplitude Modulation)**.

Diagrammi come quelli di Fig 2-19, che mostrano le combinazioni legali di ampiezza e fase, sono chiamati **schemi a costellazione**. Ogni modem ad alta velocità ha il proprio schema a costellazione e può parlare solo con altri modem che ne usano uno uguale (sebbene la maggior parte dei modem possa emulare quelli più lenti). Per esempio il modem standard ITU V.32 9600 bps usa la costellazione di Fig 2-19 (b).

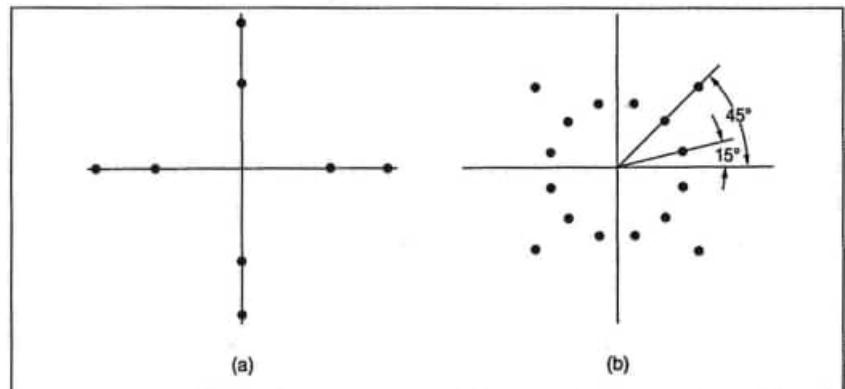


Fig. 2-19 (a) Modulazione a 3 bit/baud. (b) Modulazione a 4 bit/baud.

Il passo successivo oltre i 9600 bps è 14.400 bps (**V.32 bis**). Questa velocità è raggiunta trasmettendo 6 bit per campione a 2400 Bd. La sua costellazione ha 64 punti. I modem fax usano questa velocità per trasmettere pagine che sono state convertite in mappe di bit. Dopo il V.32 bis viene il **V.34**, che va a 28.800 bps.

Con così tanti punti nella costellazione, anche una piccola quantità di rumore nella fase o nell'ampiezza rilevata può risultare in un errore e potenzialmente in 6 bit sbagliati. Per ridurre la possibilità di errore, molti modem aggiungono un bit di parità, mettendo 128 punti nella costellazione. La codifica dei punti è fatta attentamente per massimizzare la possibilità di rilevare errori, e si chiama **codifica trellis**.

Un approccio completamente differente per la trasmissione ad alta velocità è dividere lo spettro disponibile a 3000 Hz in 512 piccole bande e trasmettere 20 bps in ciascuna, per esempio. Questo schema richiede un processore potente nel modem, ma ha il vantaggio di essere capace di disabilitare bande di frequenza che sono troppo rumorose. I modem che usano questo approccio hanno normalmente anche le funzioni V.32 o V.34, in modo da poter dialogare con i modem standard.

Molti modem adesso includono direttamente in hardware anche compressione e correzione d'errore. Il grande vantaggio di questo approccio è che queste funzioni migliorano la velocità effettiva senza richiedere alcun cambiamento al software. Uno schema di compressione popolare è **MNP 5**, che usa la codifica di stringa per comprimere serie di bit identici. Anche i modem fax usano la codifica di stringa, dal momento che le serie di 0 (silenzio) sono molto comuni. Un altro schema è il **V.42 bis**, che usa l'algoritmo di compressione Ziv-Lempel usato anche nel programma compress e in altri programmi (Ziv, Lempel, 1977).

Anche quando si usano tali modem ci può essere un altro problema sulle linee telefoniche: l'eco. Su una linea lunga, quando il segnale arriva alla destinazione finale, un po' di energia può essere riflessa, come capita con l'eco acustica in montagna. Come esempio di eco elettromagnetica, si provi a far splendere la luce di un flash da una stanza oscura attraverso una finestra chiusa di notte. Si vedrà il riflesso della luce del flash nella finestra (cioè una parte dell'energia è stata riflessa nella giunzione aria-vetro e rimandata indietro). La stessa cosa accade nelle linee di trasmissione, specialmente nel punto in cui i circuiti locali terminano nell'ufficio terminale.

L'effetto dell'eco è che una persona che parla al telefono sente le proprie parole dopo un piccolo ritardo. Studi psicologici hanno mostrato che questo disturba molte persone, spesso rendendole balbettanti o confuse. Per eliminare il problema dell'eco, vengono installati **soppressori di eco** sulle linee più lunghe di 2000 km (sulle linee corte l'eco torna indietro così velocemente che la gente non è disturbata). Un soppressore di eco è un dispositivo che rileva le parole che vengono da un capo della connessione ed elimina tutti i segnali che vanno nell'altro senso. Fondamentalmente è un amplificatore che può essere acceso o spento da un segnale di controllo prodotto da un circuito per la rilevazione delle parole. Quando la prima persona smette di parlare e inizia la seconda, il soppressore di eco cambia direzione. Un buon soppressore di eco può invertirsi da 2 a 5 ms. Mentre funziona, comunque, l'informazione può viaggiare solo in una direzione; le eco non possono tornare indietro al mandante. La figura 2-20 (a) mostra lo stato del soppressore di eco mentre A parla a B. La figura 2-20 (b) mostra lo stato dopo che B ha iniziato a parlare.

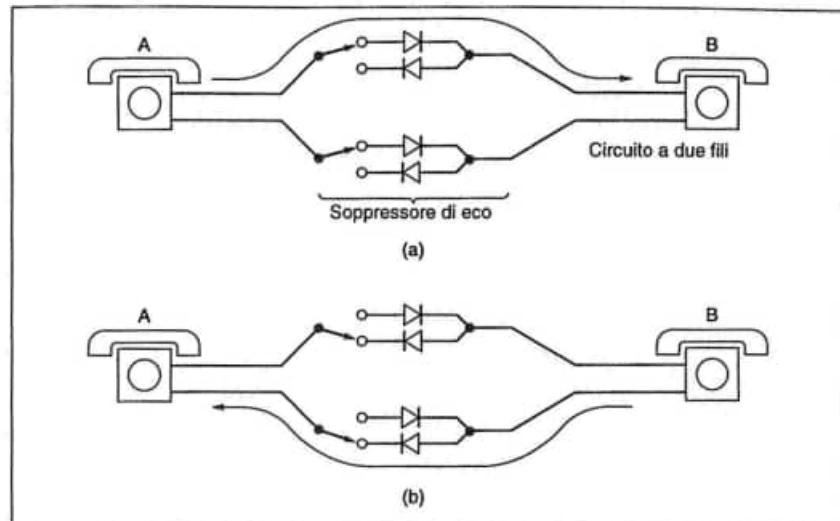


Fig. 2-20 (a) A parla a B. (b) B parla ad A.

I soppressori di eco hanno diverse proprietà che non sono desiderabili per la comunicazione di dati. Primo, se non fossero presenti, sarebbe possibile trasmettere in entrambe le direzioni allo stesso tempo usando una diversa banda di frequenza per ogni direzione. Questo approccio è chiamato trasmissione **full-duplex**. Con i soppressori di eco, la trasmissione full-duplex è impossibile. L'alternativa è la trasmissione **half-duplex**, nella quale la comunicazione può andare in ogni senso, ma solo uno per volta. Un binario è half-duplex. Anche se la trasmissione half-duplex è adeguata, è un fastidio poiché il tempo richiesto per cambiare direzione può essere rilevante. Inoltre i soppressori di eco sono progettati per cambiare direzione quando rilevano voci, non dati digitali. Per alleviare questi problemi, sui circuiti telefonici con soppressori di eco è stata fornita una via di fuga. Quando i soppressori di eco sentono un tono puro a 2100 Hz, essi si fermano e rimangono bloccati fino a quando è presente una portante. Questo è uno dei molti esempi di **segnalazione in banda**, così detta perché i segnali di controllo giacciono nella banda accessibile all'utente. In generale la tendenza è lontana dalla segnalazione in banda, per prevenire gli utenti dall'interferire con le operazioni dello stesso sistema. Negli Stati Uniti, la maggior parte della segnalazione in banda è finita, ma in altri paesi ancora esiste. Una alternativa ai soppressori di eco, sono i cancellatori di eco. Questi sono circuiti che simulano l'eco, la stimano, e la sottraggono dal segnale consegnato, senza il bisogno di relay meccanici. Quando si usano cancellatori di eco, sono possibili operazioni full-duplex. Per questo motivo, cancellatori di eco stanno rapidamente sostituendo i soppressori di eco negli Stati Uniti e in altri paesi.

RS-232-C e RS-449

L'interfaccia fra il computer o il terminale e il modem è un esempio di protocollo di livello fisico. Essa deve specificare in dettaglio l'interfaccia meccanica, elettrica, funzionale e procedurale. Daremo adesso uno sguardo da vicino a due ben noti strati fisici standard: la RS-232-C e sua succeditrice RS-449. Partiamo con la **RS-232-C**, la terza revisione dello standard RS-232. Lo standard fu compilato dall'Associazione delle industrie elettroniche, un'organizzazione commerciale di fabbricanti di elettronica, ed è propriamente indicato come EIA RS-232-C. La versione internazionale è data nella raccomandazione CCITT V. 24, che è simile ma differisce leggermente su qualcuno dei circuiti raramente usati. Negli standard, il terminale o il computer è chiamato ufficialmente **DTE** (Data Terminal Equipment) e il modem è chiamato ufficialmente **DCE** (Data Circuit-Terminating Equipment).

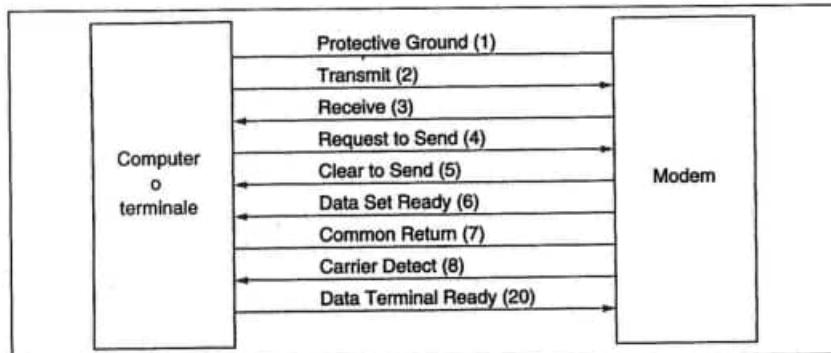


Fig. 2-21 Alcuni dei principali circuiti RS-232-C. I numeri di pin sono tra parentesi.

La specifica meccanica è per un connettore a 25 pin largo $47,04 \pm 13$ mm (da centro vite a centro vite), con tutte le altre dimensioni egualmente ben specificate. La riga superiore ha i piedini numerati da 1 a 13 (da sinistra a destra); la riga inferiore ha i piedini numerati da 14 a 25 (anch'essa da sinistra a destra).

La specifica elettrica per la RS-232-C è che una tensione inferiore a -3 V è un 1 binario e una tensione superiore a $+4$ V è uno 0 binario. Sono permesse velocità di dati fino a 20 kbps, così come cavi fino a 15 m.

La specifica funzionale dice quali circuiti sono connessi a ciascuno dei 25 piedini, e cosa significano. La figura 2-21 mostra 9 piedini che sono quasi sempre implementati. I restanti sono frequentemente omessi. Quando il terminale o il computer viene acceso, esso assegna (cioè pone a un 1 logico) Data Terminal Ready (piedino 20). Quando viene acceso il modem, viene settato Data Set Ready (piedino 6). Quando il modem rileva una portante sulla linea telefonica, setta Carrier Detect (piedino 8). Request to Send (piedino 4) indica che il terminale vuole spedire dati. Clear to Send (piedino 5) significa che il modem è pronto a ricevere dati. I dati sono trasmessi sul circuito Transmit (piedino 2) e ricevuti sul circuito Receive (piedino 3).

Ci sono altri circuiti per selezionare la velocità dei dati, testare il modem, sincronizzare i dati, rilevare il segnale di chiamata, e spedire dati nella direzione inversa su un canale secondario. In pratica sono raramente usati. La specifica procedurale è il protocollo, cioè la sequenza legale di eventi. Il protocollo è basato su coppie azione-reazione. Quando il terminale afferma Request to Send, per esempio, il modem risponde con Clear to Send, se può ricevere dati. Simili coppie azione-reazione esistono anche per gli altri circuiti. Capita spesso che due computer debbano essere connessi via RS-232-C. Dal momento che nessuno dei due è un modem c'è un problema di interfaccia. Questo problema è risolto connettendoli con un dispositivo detto **modem nullo**, che connette la linea di trasmissione di una macchina alla linea di ricezione dell'altra. Inoltre incrocia altre linee in modo simile. Un modem nullo sembra un cavo corto.

La RS-232-C è stata usata per anni. A poco a poco, il limite di velocità dei dati a non più di 20 kbps e la lunghezza del cavo a non più di 15 m sono diventati sempre più fastidiosi. La EIA dibatté a lungo se fosse meglio un nuovo standard compatibile con quello vecchio (ma tecnicamente non molto avanzato) o uno incompatibile che andasse incontro a tutti i bisogni per gli anni a venire. Alla fine raggiunsero un compromesso scegliendo entrambe le soluzioni. Il nuovo standard, la **RS-449**, è in realtà tre standard in uno. Le interfacce meccaniche, funzionali e procedurali sono nella RS-449, ma l'interfaccia elettrica è definita da due standard diversi. Il primo, la **RS-423-A**, è simile alla RS-232-C nel fatto che tutti i suoi circuiti condividono una massa comune. Questa tecnica è detta trasmissione **sbilanciata**. Il secondo standard elettrico, la **RS-422-A**, al contrario, usa la trasmissione **bilanciata**, nella quale ognuno dei circuiti principali richiede due fili, senza massa comune. Il risultato è che la RS-422-A può essere usata a velocità fino a 2 Mbps su cavi oltre ai 60 m.

I circuiti usati nella RS-449 sono mostrati in figura 2-22. Diversi circuiti nuovi non presenti nella RS-232-C sono stati aggiunti. In particolare, sono stati inclusi circuiti per testare il modem sia localmente che a distanza. A causa dell'inclusione di un certo numero di circuiti a due fili (quando si usa la RS-422-A), il nuovo standard ha bisogno di più piedini, così il familiare connettore a 25 piedini è stato abbandonato. Al suo posto c'è un connettore a 37 piedini e un connettore a 9 piedini. Il connettore a 9 piedini è richiesto solo se si usa il secondo canale (inverso).

Fibra nel circuito locale

Per servizi futuri avanzati, quali il video a richiesta, il canale a 3 kHz attualmente usato non servirà. La discussione su cosa fare in questo caso si incentra su due soluzioni. Quella più naturale – stende una fibra dall'ufficio terminale alle nostre case – è detta **FTTH** (Fiber To The Home). Questa soluzione si adatterebbe bene al sistema attuale ma non sarebbe economicamente accettabile per decenni. È troppo costosa. Una soluzione alternativa che è molto più economica è la **FTTC** (Fiber To The Curb). In questo modello, la società telefonica fa correre una fibra da ogni ufficio terminale a ciascun quartiere (Paff, 1995). La fibra termina in una scatola di giunzione nella quale entrano tutti i circuiti locali. Poiché questi sono ora molto più corti (diciamo 100 m invece di 3 km), essi possono andare a velocità più alte, probabilmente 1 Mbps, che è appena quanto basta per trasmettere video compresso. Questo schema è mostrato in figura 2-23 (a). In questo modo, si possono riversare sulla fibra video multipli (o altri canali di informa-

RS-232-C			CCITT V.24			RS-449			
Code	Pin	Circuit	Code	Pin	Circuit	Code	Pin	Circuit	
AA	1	Protective ground	101	1	Protective ground	SG	1	Signal ground	
AB	7	Signal ground	102	7	Signal ground	SC	19	Send common	
						RC	37	Receive common	
BA	2	Transmitted data	103	2	Transmitted data	SD	4, 22	Send data	
BB	3	Received data	104	3	Received data	RD	6, 24	Receive data	
CA	4	Request to send	105	4	Request to send	RS	7, 25	Request to send	
CB	5	Clear to send	106	5	Ready for sending	CS	9, 27	Clear to send	
CC	6	Data set ready	107	6	Data set ready	DM	11, 29	Data mode	
CD	20	Data terminal ready	108	20	Data terminal ready	TR	12, 30	Terminal ready	
CE	22	Ring indicator	125	22	Calling indicator	IC	15	Incoming call	
CF	8	Line detector	109	8	Line detector	RR	13, 31	Receiver ready	
CG	21	Signal quality	110	21	Signal quality	SQ	33	Signal quality	
CH	23	DTE rate	111	23	DTE rate	SR	16	Signaling rate	
CI	18	DCE rate	112	18	DCE rate	SI	2	Signaling indicators	
			136		New signal	IS	28	Terminal in service	
			126	11	Select frequency	NS	34	New signal	
						SF	16	Select frequency	
DA	24	DTE timing	113	24	DTE timing	TT	17, 25	Terminal timing	
DB	15	DCE timing	114	15	DCE timing	ST	5, 23	Send timing	
DD	17	Receiver timing	115	17	Receiver timing	RT	8, 26	Receive timing	
Canale Secondario	SBA	14	Transmitted data	118	14	Transmitted data	SSD	3	Send data
	SBB	16	Received data	119	16	Received data	SRD	4	Receive data
	SCA	19	Request to send	120	19	Line signal	SRS	7	Request to send
	SCB	13	Clear to send	121	13	Channel ready	SCS	8	Clear to send
	SCF	12	Line detector	122	12	Line detector	SRR	2	Receiver ready
						LL	10	Local loopback	
						RL	14	Remote loopback	
						TM	18	Test mode	
						SS	32	Select standby	
						SB	36	Standby indicator	

Fig. 2-22 Confronto fra RS-232-C, V.24 ed RS-449.

zione) ad alta velocità che alla fine si riversano sui doppini. Condividendo fibra da 1 Gbps fra 100-1000 utenti, il costo per utente si può ridurre, e si può fornire una larghezza di banda considerevolmente più alta dell'attuale. Andare parecchio sopra 1 Mbps per lunghe distanze con i doppini esistenti è impossibile. Quindi, a lungo termine, tutti i doppini dovranno essere sostituiti dalla fibra. Se la soluzione intermedia FTTC debba essere usata per il momento o la FTTH debba essere l'obiettivo sin dall'inizio, è una questione dibattuta all'interno dell'industria telefonica.

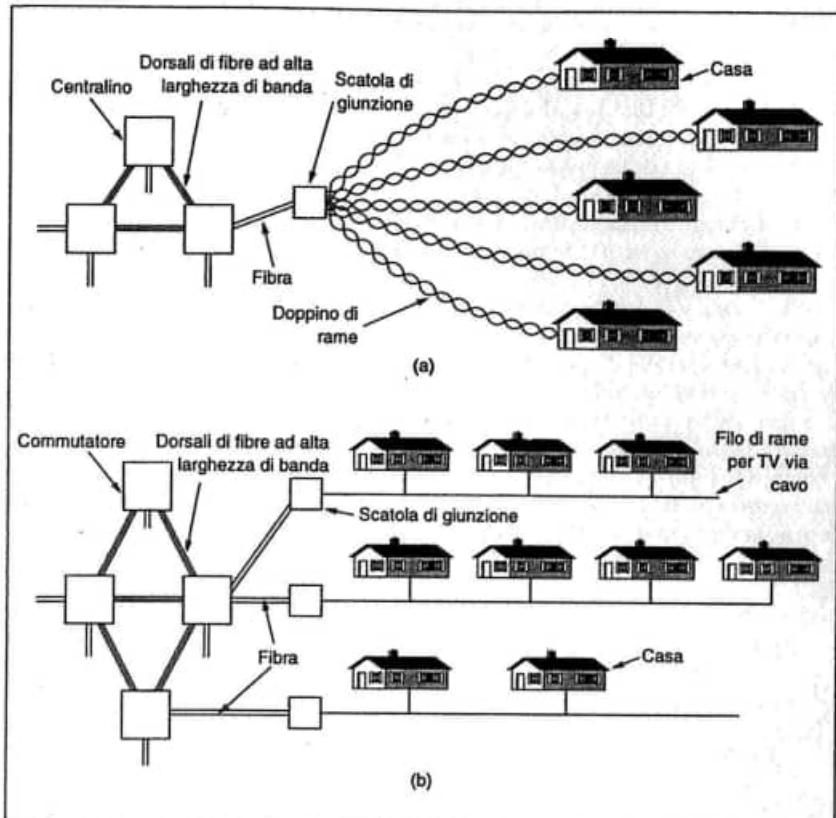


Fig. 2-23 Fibra nelle case. (a) Usando la rete telefonica. (b) Usando la rete TV via cavo.

Uno schema alternativo che usa l'infrastruttura esistente della TV via cavo è mostrato in figura 2-23 (b). Qui si usa un cavo a caduta multipla (multidrop) invece del sistema punto-a-punto caratteristico del sistema telefonico. È probabile che entrambi gli schemi in figura 2-23 (a) e figura 2-23 (b) coesisteranno nel futuro, quando le società telefoniche e gli operatori di TV via cavo entreranno in concorrenza per servizi vocali, dati ed eventualmente anche televisivi. Per maggiori informazioni su questo argomento si vedano Cook, Stern (1994); Miki (1994b); Mochida (1994).

2.4.4 Dorsali e multiplexing

L'economia di scala gioca un ruolo importante nel sistema telefonico. Costa essenzialmente lo stesso installare e mantenere fra due uffici di commutazione tanto una dorsale ad alta banda quanto una a bassa banda (cioè i costi derivano dal dover scavare le tracce

e non dal filo di rame o dalla fibra ottica). Di conseguenza, le società telefoniche hanno sviluppato tecnologie atte a immettere molte conversazioni simultaneamente su una singola dorsale fisica. Tali schemi multiplex possono essere divisi in due categorie base: **FDM** (Frequency Division Multiplexing – a partizione di frequenza) e **TDM** (Time Division Multiplexing – a partizione di tempo).

In FDM lo spettro di frequenza è diviso fra più canali logici e ogni utente ha possesso esclusivo di una certa banda di frequenza. Nel TDM invece gli utenti a turno periodicamente ottengono l'intera larghezza di banda per un piccolo quanto di tempo.

La diffusione radio in AM fornisce un esempio di entrambi i tipi di multiplexing. Lo spettro allocato è di circa 1 MHz, più o meno da 500 a 1500 kHz. A canali logici diversi (stazioni) sono assegnate frequenze diverse, ognuna operante in una porzione dello spettro, con la separazione fra i canali abbastanza grande da prevenire le interferenze. Questo sistema è un esempio di multiplexing a divisione di frequenza. In aggiunta (in alcuni paesi), le stazioni individuali hanno due sottocanali logici: musica e pubblicità. Questi due si alternano nel tempo sulla stessa frequenza, prima un po' di musica, poi un po' di annunci, poi ancora musica e così via. Questo è il multiplexing a divisione di tempo, che esamineremo sotto. Dopo di che vedremo come FDM può essere applicato alle fibre ottiche (multiplexing a divisione di lunghezza d'onda). Poi passeremo a TDM e finiremo con un sistema avanzato TDM usato per le fibre ottiche (SONET).

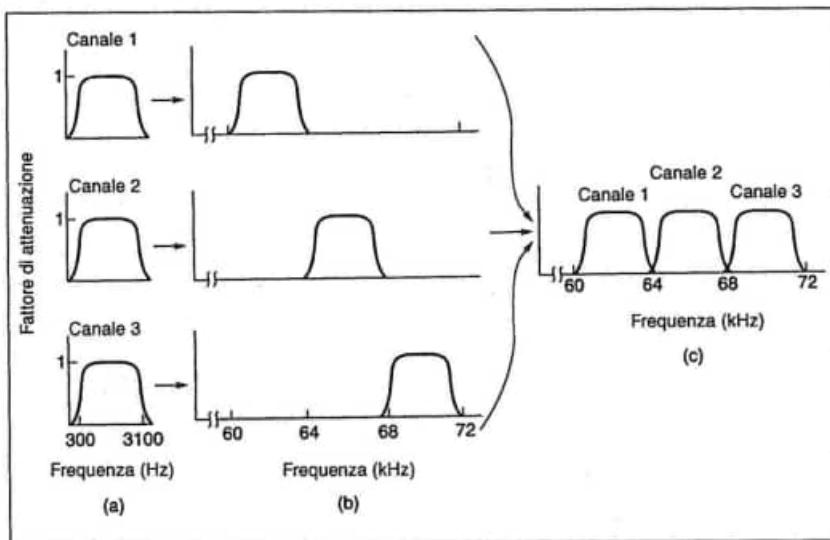


Fig. 2-24 Multiplexing a divisione di frequenza. (a) Le larghezze di banda originali. (b) Le larghezze di banda aumentate in frequenza. (c) Il canale multiplexato.

Multiplexing a divisione di frequenza

La figura 2-24 mostra tre canali telefonici voice-grade gestiti usando FDM. I filtri limitano la larghezza di banda utilizzabile fino a circa 3000 Hz per canale voice-grade. Quando molti canali sono combinati insieme, sono allocati 4000 Hz a ciascuno per tenerli ben separati. Prima i canali vocali vengono aumentati in frequenza ognuno in una diversa quantità. Poi possono essere combinati, dal momento che adesso due canali non occupano la stessa porzione dello spettro. Si noti che anche se ci sono dei salti (bande di guardia) fra i canali, c'è una certa sovrapposizione fra canali adiacenti, poiché i filtri non hanno dei confini precisi. Questa sovrapposizione significa che un picco alto al confine di un canale sarà sentito in quello adiacente come rumore non termico.

Gli schemi FDM usati nel mondo sono standardizzati fino a un certo punto. Uno standard diffuso è il 12 canali vocali a 4000 Hz (3000 Hz per l'utente più due bande di guardia di 500 Hz ciascuna) multiplexati nella banda da 60 a 108 kHz. Questa unità è chiamata **gruppo**. La banda da 12 a 60 kHz è qualche volta usata per un altro gruppo. Molte società offrono agli utenti un servizio di affitto di linea da 48 a 56 kbps, basato sul gruppo. Cinque gruppi (60 canali vocali) possono essere multiplexati per formare un **supergruppo**. L'unità successiva è il **gruppo principale**, che consiste di cinque supergruppi (standard CCITT) o dieci supergruppi (sistema Bell). Esistono altri standard fino a 230.000 canali vocali.

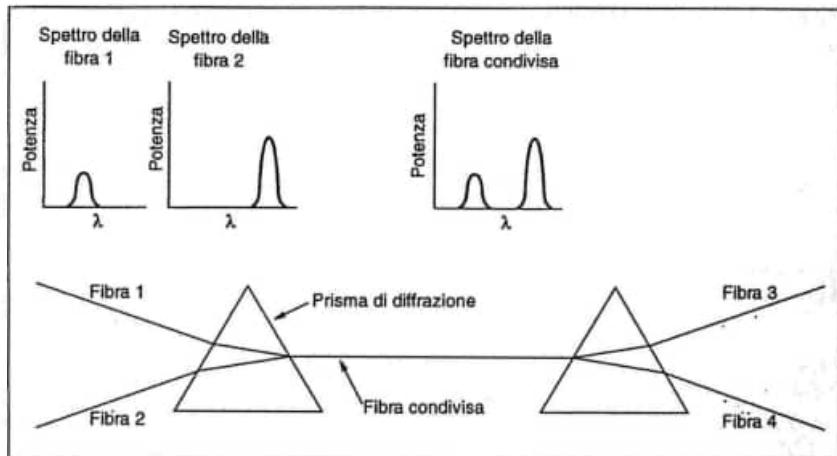


Fig. 2-25 Multiplexing a divisione di lunghezza d'onda.

Multiplexing a divisione di lunghezza d'onda

Per i canali di fibra ottica si usa una variante del multiplexing a divisione di frequenza. È detto **WDM** (Wavelength Division Multiplexing). Un modo semplice per ottenere FDM sulle fibre è illustrato in figura 2-25. Qui due fibre convergono insieme in un prisma (o più probabilmente, una grata di diffrazione), ognuna con la sua energia in una banda

differenti. I due raggi sono fatti passare attraverso il prisma o la grata e combinati in una singola fibra condivisa per la trasmissione a una destinazione lontana, dove sono di nuovo divisi.

Qui non c'è realmente nulla di nuovo. Fino a quando ogni canale ha il proprio intervallo di frequenza, e tutti gli intervalli sono disgiunti, essi possono essere multiplexati insieme sulla fibra a lungo percorso. La sola differenza con FDM elettrico è che il sistema ottico che usa una grata di diffrazione è completamente passivo, e così altamente affidabile. Si dovrebbe notare che la ragione per cui il WDM è popolare è che l'energia su una singola fibra è larga tipicamente solo pochi gigahertz poiché è attualmente impossibile convertire più velocemente fra mezzo elettrico e ottico. Poiché la larghezza di banda su una singola fibra è di circa 25.000 GHz (vedi figura 2-6), esiste un grande potenziale per multiplexare molti canali insieme su rotte a lungo raggio. Una condizione necessaria, comunque, è che i canali in entrata usino frequenze diverse. Un'applicazione potenziale del WDM è nei sistemi FTTC descritti prima. Inizialmente, una società telefonica stendeva una singola fibra da un ufficio terminale a una scatola di giunzione di quartiere dove si incontrava con i doppini dalle case. Adesso, che il costo della fibra è diminuito e la richiesta è aumentata, i doppini possono essere rimpiazzati dalla fibra e tutti i circuiti locali si uniscono nella fibra che corre all'ufficio terminale usando WDM.

Nell'esempio della figura 2-25, abbiamo fissato un sistema a lunghezza d'onda. I bit dalla fibra 1 vanno alla fibra 3 e bit dalla fibra 2 vanno alla fibra 4. Non è possibile avere bit che vanno dalla fibra 1 alla fibra 4. Tuttavia, è anche possibile costruire sistemi WDM commutati. In tale dispositivo, ci sono molte fibre input e molte fibre output, e i dati da ogni fibra input possono andare in qualsiasi fibra output. Tipicamente, l'accoppiatore è una stella passiva, con la luce che illumina la stella da ogni fibra input. Sebbene diffondere l'energia su n output la diluisca di un fattore n , tali sistemi sono pratici per centinaia di canali.

Naturalmente, se la luce da una delle fibre in entrata è a 1,50206 μ e deve poter andare in qualsiasi fibra output, tutte le fibre output hanno bisogno di filtri modulabili cosicché quello selezionato si può configurare a 1,50206 μ. Tali filtri ottici modulabili possono essere costruiti dagli interferometri di Fabry-Perot o di Mach-Zehnder. Alternativamente, le fibre input possono essere modulabili e quelle output fisse. Averle entrambe modulabili è una spesa superflua e raramente ne vale la pena.

Multiplexing a divisione di tempo

Sebbene la FDM sia ancora usata sui fili di rame o sui canali a microonde, essa richiede circuiterie analogiche e non è possibile che essa venga fatta da un computer. Al contrario, il TDM può essere gestito interamente da elettronica digitale, e per questa reagire si è molto diffuso negli ultimi anni. Sfortunatamente, si può usare solo per dati digitali. Poiché i circuiti locali producono segnali analogici, c'è bisogno di una conversione dall'analogico al digitale negli uffici terminali, dove arrivano tutti i circuiti locali per essere combinati nelle dorsali uscenti. Daremo ora uno sguardo a come vengono digitalizzati i segnali vocali multipli analogici e combinati in una singola dorsale digitale uscente (anche i dati di un computer spediti su un modem sono analogici quando arrivano all'ufficio terminale). I segnali analogici sono digitalizzati nell'ufficio terminale da un dispositivo chiamato codec (codificatore-decodificatore), producendo un numero di 7 od 8 bit (vedi figura 2-17). Il codec

cattura 8000 campioni al secondo (125 μs/campione) secondo il teorema di Nyquist che dice che ciò è sufficiente per catturare tutta l'informazione da un canale telefonico di larghezza di banda 4 kHz. A un tasso di campionamento più basso, l'informazione verrebbe perduta; a uno più alto, nessuna informazione extra verrebbe guadagnata.

Questa tecnica si chiama PCM (Pulse Code Modulation). PCM è il cuore del moderno sistema telefonico. Di conseguenza, virtualmente tutti gli intervalli di tempo all'interno del sistema telefonico sono multipli di 125 μs.

Quando la trasmissione digitale cominciò a emergere come una tecnologia accettabile, la CCITT fu incapace di raggiungere un accordo su uno standard internazionale per la PCM. Di conseguenza, adesso c'è una varietà di schemi incompatibili in uso in paesi differenti in tutto il mondo. Le connessioni internazionali fra paesi incompatibili richiedono delle "scatole nere" (spesso costose) per convertire il sistema del paese originario in quello del paese ricevente.

Un metodo che è largamente in uso in Nord America e in Giappone è la portante T1, illustrata in figura 2-26. (Parlando tecnicamente, il formato è chiamato DS1 e la portante è chiamata T1, ma qui non faremo questa sottile distinzione). La portante T1 consiste di 24 canali vocali multiplexati insieme. Di solito, i segnali analogici sono campionati in modo round-robin e il flusso analogico risultante è passato al codec, piuttosto di avere 24 codec separati e poi fondere il risultato digitale.

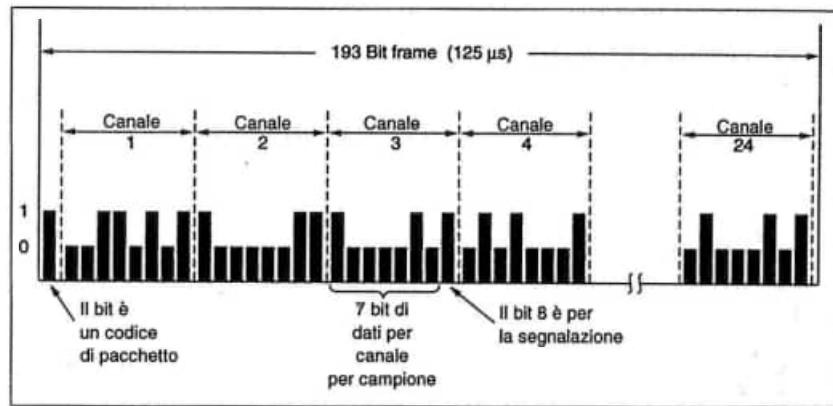


Fig. 2-26 La portante T1 (1,544 Mbps).

Un pacchetto consiste di $24 \times 8 = 192$ bit, più un bit extra, che porta a 193 bit ogni 125 μs. Si raggiunge una velocità di dati complessiva di 1,544 Mbps. Il 193-esimo bit è usato per la sincronizzazione fra i pacchetti. Quest'ultimo prende come valore la configurazione 01010101... Normalmente il ricevitore continua a controllare questo bit per essere sicuro di non perdere la sincronizzazione. Se va fuori sincronismo, il ricevitore può cercare tale configurazione per riuscire a risincronizzarsi. Gli utenti analogici non possono generare lo schema di bit, perché questo corrisponde a un'onda sinusoidale a 4000 Hz,

che sarebbe filtrata. Gli utenti digitali, naturalmente, possono generare questo schema, ma le probabilità che questo sia presente quando il pacchetto decade sono basse. Quando si sta usando un sistema T1 interamente per i dati, si usano solo 23 canali. Il 24-esimo è usato per uno speciale schema di sincronizzazione, per permettere un recupero più veloce nell'occasione in cui il pacchetto decade.

Quando alla fine la CCITT raggiunse un accordo, ritenne che 8000 bps per la segnalazione di informazione fosse troppo, quindi il suo standard a 1,544 Mbps si basa su dati a 8 bit piuttosto che a 7 bit; cioè, il segnale analogico è quantizzato in 256 invece che 128 livelli discreti. Esistono due varianti incompatibili. Nella **segnalazione a segnale-comune**, il bit extra (che è attaccato sul retro piuttosto che sul davanti del pacchetto di 193 bit) prende il valore 10101010... nei pacchetti dispari e contiene informazioni per la segnalazione per tutti i canali nei pacchetti pari.

Con l'altra variante, **segnalazione associata al canale**, ogni canale ha il proprio sottocanale di informazione privato. Un sottocanale privato è definito allocando uno degli 8 bit utente in ogni sesto pacchetto per scopi di segnalazione, cosicché cinque dei sei campioni sono grandi 8 bit, e l'altro solo 7 bit. La CCITT ha anche una raccomandazione per una portante PCM a 2,048 Mbps chiamata E1. Questa portante ha 32 campioni di dati a 8 bit raggruppati nel pacchetto base di 125 µs. Trenta canali sono usati per l'informazione e due per la segnalazione. Ogni gruppo di 4 pacchetti fornisce 64 bit di segnalazione, metà dei quali sono usati per la sincronizzazione dei pacchetti o sono riservati in ogni paese per l'uso che si desidera. Escludendo il Nord America e il Giappone, la portante a 2,048 Mbps è di uso molto diffuso.

Una volta che il segnale vocale è stato digitalizzato, si usano tecniche statistiche per ridurre il numero di bit necessari per canale. Queste tecniche sono appropriate non solo per codificare le parole, ma per la digitalizzazione di ogni segnale analogico. Tutti i

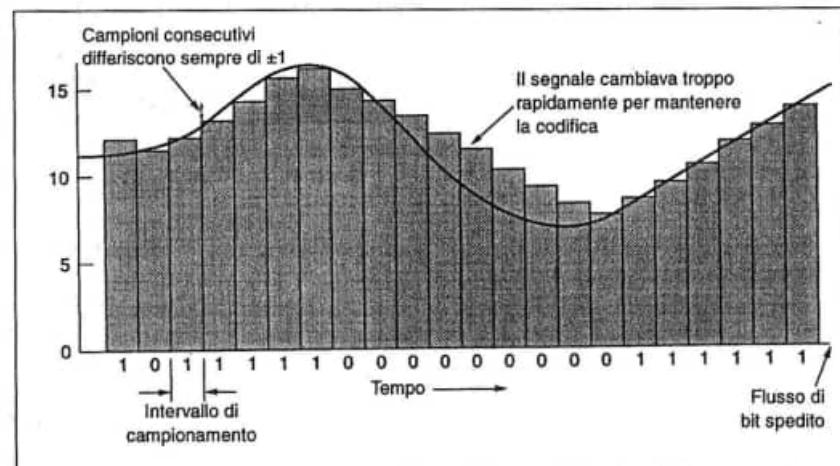


Fig. 2-27 Modulazione delta.

metodi di compattamento si basano sul principio che il segnale cambia in modo relativamente lento rispetto alla frequenza di campionamento, cosicché molta dell'informazione al livello digitale di 7 o 8 bit è ridondante.

Un metodo, chiamato **modulazione a codice di impulso differenziale**, consiste nell'emettere non l'ampiezza digitalizzata, ma la differenza tra il valore corrente e il precedente. Dal momento che i salti di ± 16 o più su una scala di 128 sono improbabili, 5 bit dovrebbero bastare invece di 7. Se il segnale occasionalmente salta bruscamente, la logica di codifica potrebbe richiedere diversi periodi di campionamento per "farcela". Per le parole, l'errore introdotto può essere ignorato.

Una variante di questo metodo di compattamento richiede che ogni valore campionato differisca dal suo predecessore o di 1 o di -1. Si trasmette un singolo bit, dicendo se il nuovo campione è sopra o sotto il precedente. Questa tecnica detta **modulazione delta**, è illustrata in figura 2-27. Come tutte le tecniche di compattamento che assumono piccoli livelli di cambio fra campioni consecutivi, la codifica delta si trova nei guai se il segnale cambia troppo velocemente, come mostrato nella figura. Quando questo accade, l'informazione viene perduta.

Un miglioramento al PCM differenziale consiste nell'estrapolare alcuni valori precedenti per prevedere il valore successivo e poi codificare la differenza fra il segnale previsto e quello effettivo. Il trasmittente e il ricevente devono usare lo stesso algoritmo di predizione, naturalmente. Tali schemi vengono detti **codifica predittiva**. Sono utili perché possono ridurre la dimensione dei numeri da codificare, quindi il numero di bit da inviare. Sebbene PCM sia ampiamente diffuso sulle dorsali fra gli uffici, l'utente riceve un beneficio relativamente piccolo se tutti i dati devono essere spediti all'ufficio terminale nella forma di un'onda sinusoidale analogica modulata a 28,8 kbps. Sarebbe bello se la portante collegasse il circuito locale direttamente al sistema a dorsali PCM, cosicché il computer potrebbe emettere dati digitali direttamente nel circuito locale a 1,544 o 2,048 Mbps. Sfortunatamente i circuiti locali non possono reggere queste velocità molto a lungo.

Il multiplexing a divisione di tempo permette a più portanti T1 di essere multiplexate in portanti di ordine più alto. La figura 2-28 mostra come ciò possa essere fatto. A sinistra vediamo quattro canali T1 che vengono multiplexati in un canale T2. Il multiplexing a T2 e oltre, è fatto bit per bit con 24 canali vocali che costituiscono un pacchetto T1. Quattro flussi T1 a 1,544 Mbps dovrebbero generare 6,176 Mbps, ma la T2 va in realtà

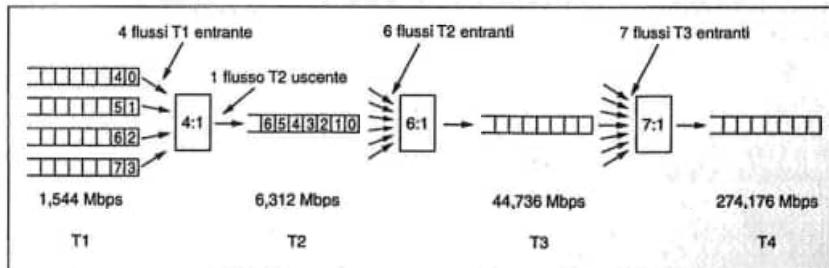


Fig. 2-28 Multiplexing di flussi T1 su portanti più alte.

a 6,312 Mbps. I bit extra sono usati per il framing e la riparazione dei guasti, nel caso che la portante cada.

Al livello successivo, sei flussi T2 sono combinati per bit per formare un flusso T3. Questi sette flussi T3 sono congiunti per fornire un flusso T4. Una certa quantità di superlavoro si aggiunge a ogni passo per il framing e il recupero.

Proprio come c'è poco accordo sulla portante base fra gli Stati Uniti e il resto del mondo, c'è egualmente poco accordo su come si debba multiplexare su portanti con bande più alte. Lo schema degli USA di scalare in passi di 4, 6 e 7 non ha convinto nessun altro, così lo standard CCITT cerca di combinare a ogni livello quattro flussi in uno. Anche il framing e il recupero sono diversi. La gerarchia CCITT per 32, 128, 512, 2048 e 8192 canali va a velocità di 2,048, 8,848, 34,304, 139,264 e 565,148 Mbps.

SONET/SDH

Nei primi giorni di impiego delle fibre ottiche, ogni società telefonica aveva il proprio sistema ottico TDM privato. Dopo lo smembramento della AT&T nel 1984, le società telefoniche locali dovettero connettersi a diverse società a lunga distanza, tutte con un diverso sistema ottico TDM, così divenne necessaria una standardizzazione. Nel 1985, la Bellcore, il braccio di ricerca della RBOC, cominciò a lavorare a uno standard, chiamato SONET (*Synchronous Optical NETwork*). Più tardi, la CCITT si unì allo sforzo, e i risultati furono uno standard SONET e un insieme parallelo di raccomandazioni CCITT (G.707, G.708 e G.709) nel 1989. Le raccomandazioni CCITT sono chiamate SDH (*Synchronous Digital Hierarchy*) ma differiscono solo in maniera minore dal SONET.

Virtualmente tutto il traffico telefonico a lunga distanza negli Stati Uniti e in gran parte altrove usa adesso dorsali che eseguono SONET nel livello fisico. Quando i processori SONET diverranno più economici, le schede di interfaccia SONET potranno essere più diffuse, e sarà più facile per le società collegare i loro computer direttamente al cuore della rete telefonica sotto speciali condizioni di noleggio delle linee. Più avanti discuteremo brevemente gli obiettivi e il progetto di SONET. Per informazioni aggiuntive si vedano: Bellamy (1991); Omidyar, Aldridge (1993).

Il progetto SONET aveva quattro obiettivi principali. Innanzitutto SONET doveva far sì che diverse portanti lavorassero insieme. Raggiungere questo obiettivo richiedeva di definire uno standard comune di segnalazione rispetto a lunghezza d'onda, temporizzazione, struttura dei pacchetti e altre questioni.

Secondo, c'era bisogno di alcuni mezzi per unificare i sistemi digitali statunitensi, europei e giapponesi, che erano tutti basati su canali PCM a 64 kbps, ma tutti combinati in modi differenti (e incompatibili).

Terzo, SONET doveva fornire un modo per combinare insieme diversi canali digitali. Al tempo in cui SONET fu concepito, la portante digitale più veloce e ancor oggi molto usata negli Stati Uniti era la T3, a 44,736 Mbps. La T4 era definita, ma non molto usata, e niente era definito al di sopra della velocità della T4. Parte della missione del SONET era di continuare la gerarchia fino ai gigabit/secondo e oltre. C'era anche bisogno di un modo standard per multiplexare canali più lenti in un solo canale SONET.

Quarto, SONET doveva fornire supporto per le operazioni, l'amministrazione e la manutenzione. I sistemi precedenti non lo facevano molto bene.

Una prima decisione fu di fare di SONET un tradizionale sistema TDM, dedicando l'intera larghezza di banda della fibra a un solo canale suddiviso tra i vari sottocanali. In questo senso, SONET è un sistema sincrono. È controllato da un orologio principale con una precisione di 1 su 10^9 . I bit su una linea SONET sono spediti a intervalli estremamente precisi, sotto il controllo dell'orologio principale.

Quando in seguito si propose la commutazione di cella come base per ISDN a banda diffusa, il fatto che essa permettesse l'arrivo di celle irregolari la etichettò come modo di trasferimento *asincrono* (cioè ATM) in contrasto con l'operazione sincrona di SONET. Un sistema SONET consiste di commutatori, multiplexer e ripetitori, tutti connessi da fibra. Un percorso da una sorgente a una destinazione con un multiplexer intermedio e un ripetitore intermedio è mostrato in figura 2-29. Nella terminologia SONET, una fibra che va direttamente da ogni dispositivo ad ogni altro dispositivo, è chiamata **sezione**. Un percorso fra due multiplexer (eventualmente con uno o più ripetitori nel mezzo) è detta **linea**. Infine, la connessione fra la sorgente e la destinazione (eventualmente con uno o più multiplexer e ripetitori) è chiamato **percorso**. La topologia SONET può essere una mesh, ma spesso è un anello duale.

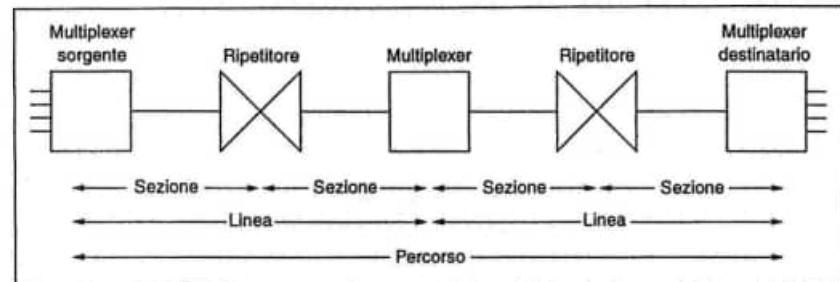


Fig. 2-29 Un percorso SONET.

Il pacchetto base SONET è un blocco di 810 byte emessi ogni 125 µs. Poiché SONET è sincrono, i pacchetti vengono emessi sia che ci siano o no dati utili da spedire. Così si trasmettono $8 \times 810 = 6480$ bit 8000 volte al secondo, per un complessivo tasso di dati a 51,48 Mbps. Questo è il canale base SONET ed è chiamato STS-1 (*Synchronous Transport Signal-1*). Tutte le dorsali SONET sono multiple di STS-1.

Le prime tre colonne di ogni pacchetto sono riservate all'informazione della gestione del sistema, come illustrato in figura 2-30. Le prime tre righe contengono l'overhead di sezione; le sei successive contengono l'overhead di linea. L'overhead di sezione è controllato e generato all'inizio e alla fine di ogni linea.

Le 87 colonne rimanenti contengono $87 \times 9 \times 8 \times 9000 = 50,112$ Mbps di dati utili. Tuttavia, i dati dell'utente, detti SPE (*Synchronous Payload Envelope*) non iniziano sempre nella riga 1, colonna 4. SPE può iniziare dovunque all'interno di un pacchetto. Nella prima riga della linea in alto è contenuto un puntatore al primo byte. La prima colonna di SPE è l'overhead di percorso (cioè l'intestazione per il protocollo del sottolivello per il percorso da punto-a-punto).

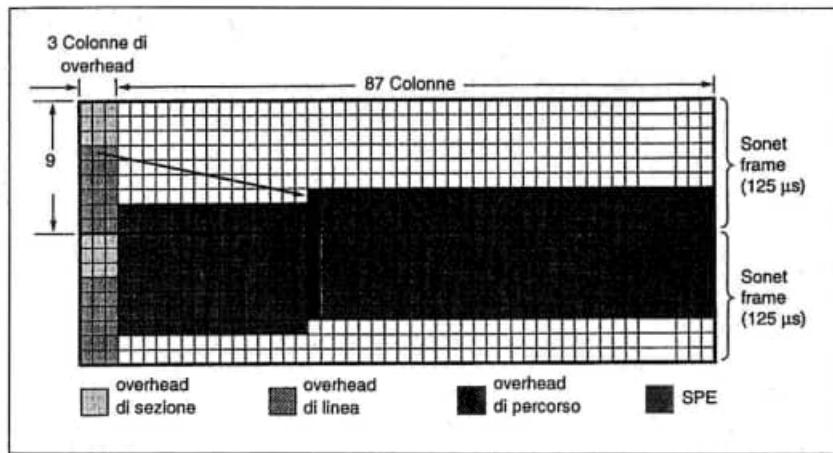


Fig. 2-30 Due pacchetti SONET in sequenza.

La capacità di SPE di iniziare dovunque all'interno del pacchetto SONET e persino di estendersi per due pacchetti, come in figura 2-30, aggiunge flessibilità al sistema. Per esempio, se arriva un carico utile alla sorgente mentre si sta costruendo un pacchetto SONET fittizio, esso può essere inserito nel pacchetto corrente, invece di essere ritardato fino all'inizio del successivo. Questo fattore serve anche quando un carico utile non si adatta esattamente in un pacchetto, come nel caso di celle ATM di 53 byte. La prima riga

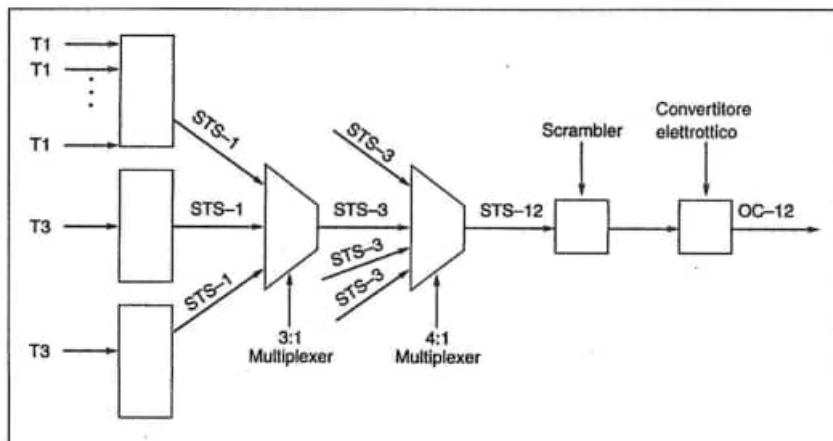


Fig. 2-31 Multiplexing in SONET.

della linea in alto può così puntare all'inizio della prima cella piena, per fornire sincronizzazione.

Le parti superiori della sezione, della linea e del percorso contengono bit in abbondanza usati per operazioni, amministrazione e manutenzione. Poiché ogni byte arriva 8000 volte al secondo, esso rappresenta un canale PCM. Tre di questi byte sono infatti usati per procurare dei canali vocali per il personale della manutenzione della sezione, della linea e del percorso. Altri byte sono usati per framing, parità, monitoraggio dell'errore, ID, temporizzazione, sincronizzazione e altre funzioni. Bellamy (1991) descrive tutti i campi in dettaglio.

Il multiplexing di multipli flussi di dati, chiamati **affluenti**, gioca un ruolo importante in SONET. Il multiplexing è illustrato in figura 2-31. A sinistra, partiamo con vari flussi input a bassa velocità, che sono convertiti alla velocità base SONET STS-1, nella maggior parte dei casi aggiungendo dei riempitivi per arrotondare a 51,84 Mbps. Successivamente, tre affluenti STS-1 sono multiplexati in flusso output STS-3 a 155,52 Mbps. Questo flusso è a sua volta multiplexato con altri tre in un flusso finale di uscita che ha 12 volte la capacità del flusso STS-1. A questo punto il segnale è disturbato, per prevenire lunghe occorrenze di 0 e di 1 dall'interferenza con la temporizzazione, e convertito da segnale elettrico a segnale ottico.

Il multiplexing è fatto byte per byte. Per esempio, quando tre affluenti STS-1 a 51,84 Mbps sono fusi in flusso STS-3 a 155,52 Mbps, il multiplexer emette prima un byte dall'affluente 1, poi 1 dall'affluente 2, e infine 1 dall'affluente 3 prima di tornare indietro all'1. La figura dell'STS-3 analoga alla figura 2-30 mostra (da sinistra a destra) colonne dagli affluenti 1, 2 e 3 in quell'ordine, poi un'altra tripla e così via fino alla colonna 270. Uno di questi pacchetti di 270×90 byte viene spedito ogni 125 μs, portando il tasso di trasmissione dati a 155,52 Mbps.

SONET		SDH	Velocità dei dati(Mbps)		
Elettrico	Ottico	Ottico	Lordo	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	1244,16	1202,688	1188,864
STS-36	OC-36	STM-12	1866,24	1804,032	1783,296
STS-48	OC-48	STM-16	2488,32	2405,376	2377,728

Fig. 2-32 Tassi di multiplexing per SONET e SDH.

In figura 2-32 è mostrata la gerarchia di multiplexing di SONET. Sono state definite le velocità da STS-1 a STS-48. La portante ottica corrispondente all'STS-*n* è detta OC-*n* ma è la stessa bit per bit eccetto che per il disturbo mostrato in figura 2-31. I nomi SDH sono differenti e iniziano da OC-3 poiché i sistemi basati su CCITT non hanno un tasso vicino a 51,84 Mbps. La portante OC-9 è attuale perché essa combacia strettamente con la velocità di una linea principale ad alta velocità usata in Giappone.

OC-18 e OC-36 saranno usate in futuro in Giappone. La velocità di dati complessiva include tutto l'overhead. Il flusso SPE esclude l'overhead di linea e di sezione. La velocità di dati utile esclude tutto l'overhead e conta solo le 86 colonne disponibili per il carico utile.

Per di più, quando una portante, come l'OC-3, non è multiplexata, ma porta i dati solo da una singola sorgente, la lettera *c* (come concatenazione) viene aggiunta al nome, così OC-3 indica una portante a 155,52 Mbps fatta di tre separate portanti OC-1, mentre OC-3c indica un flusso di dati da una singola sorgente a 155,52 Mbps. I tre flussi OC-1 all'interno di un flusso OC-3c sono intervallati da colonne, prima la colonna 1 dal flusso 1, poi la colonna 2 dal flusso 2, poi la colonna 1 dal flusso 3, seguita dalla colonna 2 dal flusso 1, e così via, portando a un pacchetto grande 270 colonne e profondo 9 righe.

La quantità di dati utenti reali in flusso OC-3c è leggermente superiore che in un flusso OC-3 (149,760 Mbps contro 148,608 Mbps) perché la colonna di sovraccarico di percorso è inclusa nello SPE solo una volta, invece di tre volte, come avverrebbe con tre flussi indipendenti OC-1. In altre parole 260 delle 270 colonne sono disponibili per dati utenti in OC-3c, mentre solo 258 colonne sono disponibili per dati utente in OC-3. Esistono anche pacchetti di alto grado concatenati (OC-12).

Adesso dovrebbe essere chiaro perché l'ATM va a 155 Mbps: l'intenzione è di portare celle ATM su dorsali SONET OC-3c. Dovrebbe anche essere chiaro che l'ampiamente citata cifra di 155 Mbps è quella complessiva, incluso il sovraccarico SONET. Inoltre, qualcuno ha arrotondato non correttamente 155,52 Mbps a 155 Mbps, invece che a 156 Mbps, e adesso tutti sbagliano.

Il livello fisico SONET è diviso in quattro sottolivelli, come in figura 2-33. Quello più basso è il sottolivello fotonico. Esso riguarda la specifica di proprietà fisiche della luce e della fibra che deve essere usata.

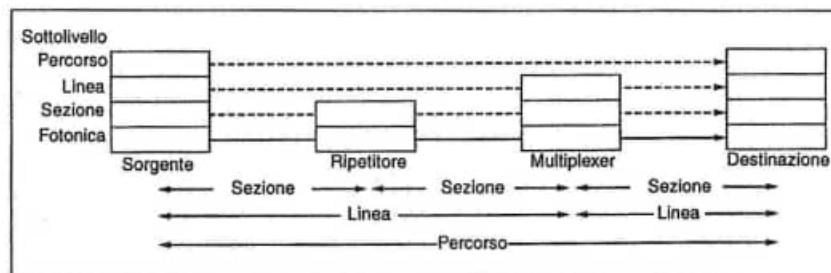


Fig. 2-33 L'architettura SONET.

I tre sottolivelli restanti corrispondono alle sezioni, alle linee e ai percorsi. Il sottolivello della sezione gestisce un percorso singolo di una fibra punto-a-punto, generando un pacchetto standard a una estremità ed elaborandolo all'altra. Le sezioni possono partire e finire ai ripetitori, che amplificano e rigenerano i bit, ma non li cambiano o li elaborano in alcun modo.

Il sottolivello della linea riguarda il multiplexing di affluenti multipli in una singola linea e il demultiplexing all'altra estremità. I ripetitori sono trasparenti al sottolivello della linea. Quando un multiplexer emette dei bit su una fibra, si aspetta che essi arrivino al multiplexer successivo non cambiati, non importa quanti ripetitori intermedi siano usati. Il protocollo nel sottolivello della linea è così fra due multiplexer e concerne questioni come quanti input siano multiplexati insieme e come. Al contrario, il sottolivello e il protocollo del percorso trattano gli aspetti punto-a-punto.

2.4.5 Comutazione

Dal punto di vista dell'ingegnere telefonico medio, il sistema fonico si divide in due parti: l'impianto esterno (il circuito locale e le dorsali, poiché essi sono all'esterno degli uffici di commutazione) e l'impianto interno (commutatori). Abbiamo già dato uno sguardo all'impianto esterno. Ora è tempo di esaminare l'impianto interno.

Due differenti tecniche di commutazione sono usate nel sistema telefonico: la commutazione a circuito e quella a pacchetto. Prima di tutto daremo una breve introduzione a ciascuna. Poi spiegheremo in dettaglio la commutazione a circuito, perché così funziona l'attuale sistema telefonico. Quindi esamineremo in dettaglio la commutazione a pacchetto nel contesto del sistema telefonico di prossima generazione, ISDN a banda larga.

Comutazione di circuito

Quando qualcuno, uomo o computer, fa una chiamata telefonica, l'apparecchiatura all'interno del sistema telefonico cerca un percorso fisico di "rame" (inclusi fibra e radio) fino al telefono del ricevente. Questa tecnica viene detta **commutazione di circuito** ed è mostrata schematicamente in figura 2-34 (a). Ognuno dei sei rettangoli rappresenta un ufficio di commutazione di portante (ufficio terminale, di pedaggio ecc.). In questo esempio, ogni ufficio ha tre linee entranti e tre linee uscenti. Quando una chiamata passa attraverso un ufficio di commutazione, si stabilisce (concretualmente) una connessione fisica tra la linea sulla quale è entrata la chiamata e una delle linee in uscita, come mostrato dalle linee tratteggiate.

Nei primi giorni della telefonia la connessione veniva fatta dall'operatore di un centralino che inseriva un cavo di giunzione nelle prese di entrata e di uscita passando così le linee. A questo proposito, si narra una piccola storia sorprendente associata all'invenzione dell'apparecchiatura di commutazione di circuito automatica. Questa invenzione si deve a un impresario di pompe funebri chiamato Almon B. Strowger, nel XIX secolo. Strowger non era il solo impresario di pompe funebri nella sua città, ve n'era un altro la cui moglie era proprio l'operatrice telefonico al centralino. Rapidamente Strowger capì che o inventava l'apparecchiatura di commutazione automatica o sarebbe uscito dagli affari. Scelse la prima opzione. Per quasi 100 anni, l'apparecchiatura di commutazione di circuito usata

in tutto il mondo fu nota come congegno Strowger (la storia non ci dice se la centralinista disoccupata trovò poi un lavoro a un centro informazioni, rispondendo a domande come: "Qual è il numero telefonico di un'impresa di pompe funebri?").

Il modello mostrato in figura 2-34 (a) è molto semplificato naturalmente, poiché parti del percorso di "rame" fra i due telefoni possono essere, infatti, connessioni a microonde con migliaia di chiamate multiplexate. Non di meno, l'idea base è valida: una volta che una chiamata è stata composta, esiste un percorso fra i due terminali e continuerà a esistere fino alla fine della chiamata.

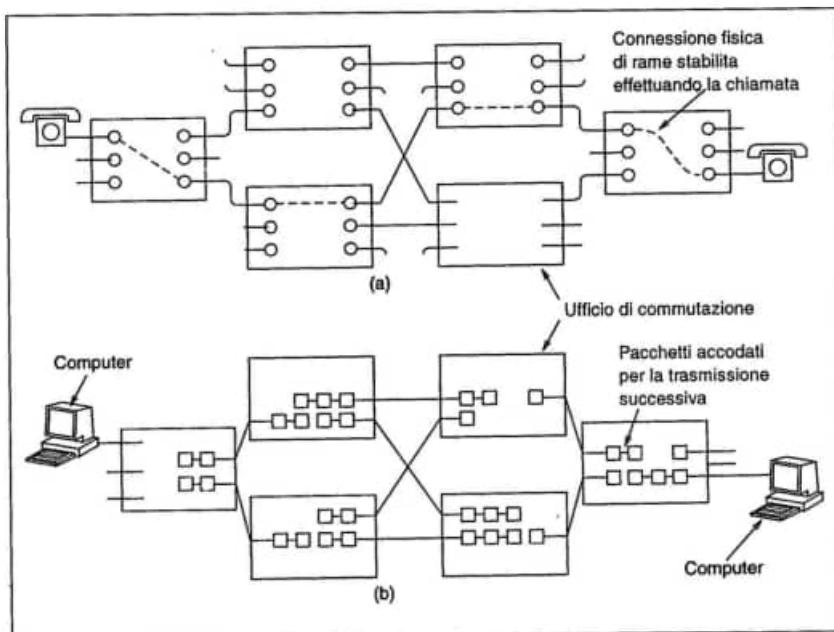


Fig. 2-34 (a) Commutazione di circuito. (b) Commutazione di pacchetti.

Una importante proprietà della commutazione di circuito è la necessità di disporre un percorso da estremo a estremo prima che ogni dato possa essere spedito. Il tempo trascorso fra la fine della composizione del numero e l'inizio dello squillo facilmente può essere di 10 s, di più sulle chiamate a lunga distanza o internazionali. Durante questo intervallo di tempo, il sistema telefonico cerca un percorso di rame, come illustrato in figura 2-35 (a). Si noti che prima che la trasmissione dei dati possa iniziare, il segnale di richiesta di chiamata si deve propagare lungo tutta la via fino alla destinazione, ed essere riconosciuto. Per molte applicazioni informatiche (ad es. verifica di crediti nei punti di vendita), non sono desiderabili tempi lunghi di inizializzazione.

Siccome esiste un percorso di rame fra le parti, una volta completata la connessione il

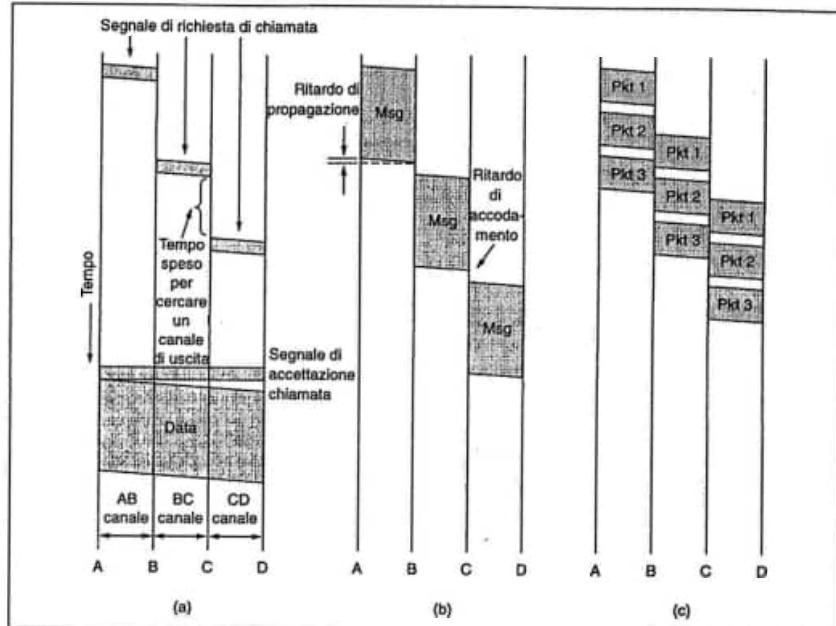


Fig. 2-35 Tempi degli eventi in (a) commutazione di circuito, (b) commutazione di messaggio, (c) commutazione di pacchetto.

solo ritardo per i dati è il tempo di propagazione per il segnale elettromagnetico, circa 5 ms per 1000 km. Sempre in conseguenza del percorso stabilito, non c'è pericolo di congestione, cioè una volta che la chiamata è stata fatta passare, non si ottiene mai segnale di occupato, anche se si potrebbe ottenerlo prima che la connessione sia stata stabilita a causa della mancanza di capacità di commutazione o di linea.

Una strategia alternativa di commutazione è la **commutazione di messaggio**, mostrata in figura 2-35 (b). Quando si usa questa forma di commutazione, non si stabilisce nessun percorso fisico di rame in anticipo fra il chiamante e il ricevente. Invece, quando il mittente ha un blocco di dati da spedire, esso viene immagazzinato nel primo ufficio di commutazione (cioè un router) e poi spedito, un passo per volta. Ogni blocco è ricevuto nella sua interezza, ispezionato in cerca di errori e poi ritrasmesso. Una rete che usa questa tecnica è detta **store-and-forward**, come abbiamo visto nel capitolo 1.

I primi sistemi di telecomunicazioni elettroniche usavano la commutazione di messaggi, cioè per i telegrammi. Il messaggio veniva perforato su un nastro di carta nell'ufficio mittente e poi letto e ritrasmesso su una linea di comunicazione fino all'ufficio successivo lungo il percorso, dove era perforato di nuovo su un nastro di carta. Lì un operatore strappava il nastro e lo leggeva in uno dei tanti lettori di nastri, uno per linea

in uscita. Questi uffici di commutazione erano chiamati uffici del nastro strappato (**torn tape offices**).

Con la commutazione di messaggio, non c'è limite sulla dimensione del blocco, il che significa che i router (in un sistema moderno) devono avere dei dischi per memorizzare lunghi blocchi. Significa anche che un singolo blocco può legare una linea da un router all'altro per minuti, rendendo la commutazione di messaggi non utile per il traffico interattivo. Per superare questi problemi, venne inventata la **commutazione di pacchetto**. Le reti a commutazione di pacchetto pongono uno stretto limite superiore sulla dimensione del blocco, permettendo ai pacchetti di essere memorizzati nella memoria principale del router invece che su disco. Siccome assicurano che nessun utente possa monopolizzare una linea di trasmissione molto a lungo (millisecondi), le reti a commutazione di pacchetto sono adatte alla gestione del traffico interattivo.

Un vantaggio ulteriore della commutazione di pacchetto sulla commutazione di messaggio è mostrato in figura 2-35 (b) e (c): il primo pacchetto di un messaggio su pacchetti multipli può essere inoltrato prima che il secondo sia completamente arrivato, riducendo il ritardo e migliorando il bilanciamento del carico. Per queste ragioni, le reti di computer sono di solito a commutazione di pacchetto, a volte a commutazione di circuito, ma mai a commutazione di messaggio.

La commutazione di circuito e la commutazione di pacchetto differiscono in molti aspetti. La differenza chiave è che la commutazione di circuito riserva statisticamente in anticipo la larghezza di banda richiesta, mentre la commutazione di pacchetto la ottiene e la rilascia al bisogno. Con la commutazione di circuito, qualsiasi larghezza di banda non usata su un circuito allocato è persa. Con la commutazione di pacchetto essa può essere utilizzata da altri pacchetti provenienti da altre sorgenti e diretti verso altre destinazioni, poiché i circuiti non sono dedicati. Tuttavia, proprio poiché i circuiti non sono dedicati, un picco improvviso di traffico in entrata può sovraccaricare un router, superando la sua capacità di memorizzazione e causando la perdita di pacchetti.

Al contrario della commutazione di circuito, quando si usa la commutazione di pacchetto è naturale che i router siano veloci e aggiungano conversione di codice. Essi in qualche misura forniscono anche correzione di errore. In alcune reti a commutazione di pacchetto, tuttavia, i pacchetti possono essere consegnati a destinazione in un ordine sbagliato. Il riordino dei pacchetti non può accadere mai con la commutazione di circuito.

Un'altra differenza è che la commutazione di circuito è completamente trasparente. Il mittente e il ricevente possono usare qualsiasi tasso di bit, formato o metodo di impacchettamento. La portante non lo sa o non se ne cura. Con la commutazione di pacchetto la portante determina alcuni parametri di base. Un'analogia grossolana è la strada rispetto alla ferrovia. Nella prima, l'utente determina la dimensione, la velocità e la natura del veicolo; nell'ultima lo fa la portante. È questa trasparenza che permette alla voce, ai dati e ai fax di coesistere all'interno del sistema telefonico.

Un'ultima differenza fra la commutazione di pacchetto e la commutazione di circuito è l'algoritmo di addebito. Le società di comunicazione a commutazione di pacchetto di solito basano il loro prezzo sia sul numero di byte (o di pacchetti) trasportati che sul tempo di connessione. Inoltre, la distanza di trasmissione di solito non importa, tranne forse a

livello internazionale. Con la commutazione di circuito, il prezzo è basato solo sulla distanza e sul tempo, non sul traffico. Le differenze sono riassunte in figura 2-36.

Caratteristica	Commutazione di circuito	Commutazione di pacchetto
Percorso di rame dedicato	Sì	No
Larghezza di banda disponibile	Fissa	Dinamica
Larghezza di banda potenzialmente persa	Sì	No
Trasmissione store and forward	No	Sì
Ogni pacchetto segue lo stesso cammino	Sì	No
Messa a punto della chiamata	Richiesta	Non necessaria
Quando può avversi congestione	Al tempo	Su ogni pacchetto
Addebito	Per minuto	Per pacchetto

Fig. 2-36 Un confronto fra reti a commutazione di circuito e a commutazione di pacchetto.

Sia la commutazione di circuito che la commutazione di pacchetto sono così importanti che ci torneremo tra breve e descriveremo in dettaglio le varie tecnologie usate.

La gerarchia di commutazione

Vale la pena spendere qualche parola su come viene fatto il routing tra le commutazioni all'interno dell'attuale sistema telefonico a commutazione di circuito. Qui descriveremo il sistema AT&T, ma altre società e nazioni usano gli stessi principi generali.

Il sistema telefonico ha cinque classi di uffici di commutazione, come in figura 2-37. Ci sono 10 uffici regionali di commutazione ed essi sono completamente interconnessi da 45 linee a fibre ottiche ad alta larghezza di banda. Sotto gli uffici regionali ci sono 67 uffici di sezione, 230 uffici primari, 1300 uffici di pedaggio e 19.000 uffici terminali. I quattro livelli più bassi erano originariamente connessi ad albero.

Le connessioni sono generalmente stabilite al livello più basso possibile. Così se un abbonato connesso all'ufficio terminale 1 chiama un altro abbonato connesso all'ufficio terminale 1, la chiamata verrà completata in quell'ufficio. Tuttavia, una chiamata da un utente collegato a un ufficio terminale 1 a un utente attaccato a un ufficio terminale 2 dovrà andare all'ufficio di pedaggio 1 (vedi figura 2-37). Inoltre una chiamata da un ufficio terminale 1 a un ufficio terminale 4 dovrà salire fino all'ufficio primario 1 e così via. Con un albero puro, esiste un solo percorso minimo, che verrebbe impiegato normalmente.

Durante anni di attività, le società telefoniche hanno notato che alcune rotte erano più occupate di altre. Per esempio, c'erano molte chiamate da New York a Los Angeles. Piuttosto di salire per tutta la gerarchia, essi installarono delle **dorsali dirette** per le rotte affollate. Alcune sono mostrate in figura 2-37 come linee tratteggiate. Di conseguenza, molte chiamate possono ora essere instradate lungo diversi percorsi. Il percorso reale scelto è in genere il più diretto, ma se i percorsi diretti sono occupati, si sceglie un'alternativa. Questo instradamento complesso è ora possibile poiché una macchina di commu-

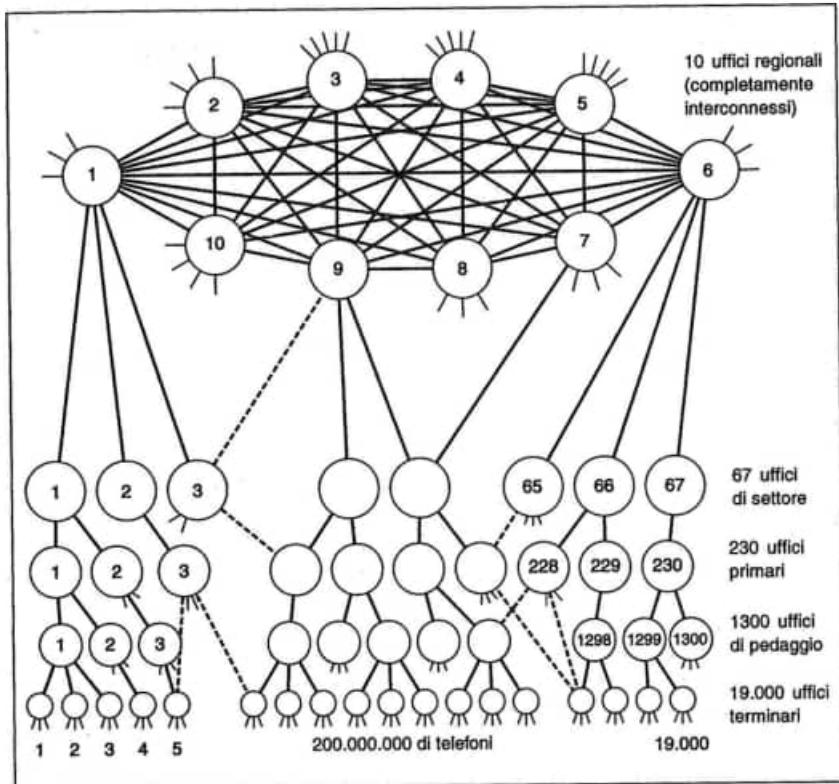


Fig. 2-37 La gerarchia telefonica della AT&T. Le linee tratteggiate sono dorsali dirette.

tazione, come la AT&T 5 ESS, è in realtà un computer di uso generale, anche se possiede una grande quantità di dispositivi per I/O specializzato.

Commutatori crossbar

Passiamo adesso dal modo in cui le chiamate vengono instradate fra le commutazioni al modo in cui i commutatori individuali lavorano internamente. Il sistema telefonico si avvale (o si avvalesse) di diverse specie di commutatori. Il tipo più semplice è il **commutatore crossbar** (a griglia), mostrato in figura 2-38. In un commutatore con n linee input e n linee output (cioè n linee full duplex), la griglia ha n^2 intersezioni, chiamate **punti di incrocio**, dove una linea input e una linea output possono essere connesse da un interruttore a semiconduttore, come in figura 2-38 (a). In figura 2-38 (b) vediamo un esempio nel quale la linea 0 è collegata alla linea 4, la linea 1 è collegata alla linea 7 e la linea 2 è collegata alla linea 6. Le linee 3 e 5 non sono connesse. Tutti i bit che arrivano al commutatore dalla linea 4, per esempio, sono immediatamente trasmesse sulla linea 0.

Quindi il crossbar implementa la commutazione di circuito costruendo una connessione elettrica diretta, proprio come i cavi di giunzione nei commutatori della prima generazione, solo che lo fa automaticamente e in tempi dell'ordine dei millisecondi.

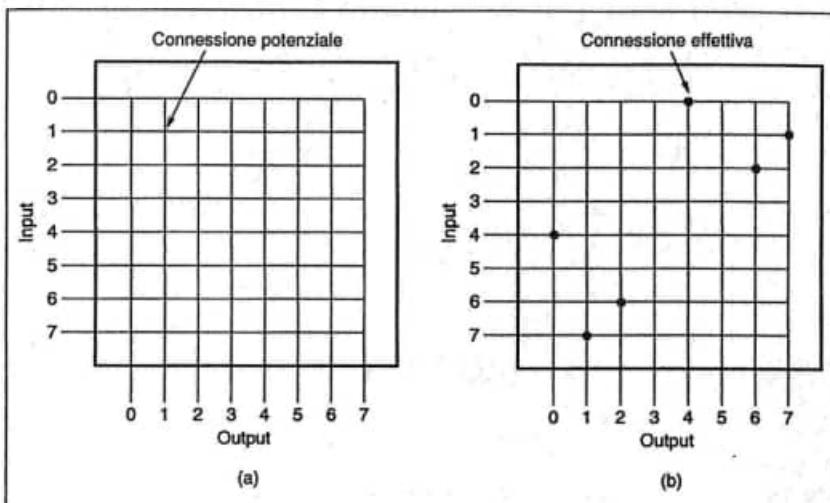


Fig. 2-38 (a) Un commutatore crossbar con nessuna connessione. (b) Un commutatore trasversale con tre connessioni: 0 con 4, 1 con 7, 2 con 6.

Il problema con un commutatore crossbar è che il numero di linee trasversali cresce con il quadrato del numero delle linee nel commutatore. Se supponiamo che tutte le linee siano full duplex e che non ci siano autoconnessioni, sono necessari solo i punti di incrocio sulle diagonali. Sono necessari tuttavia $n(n-1)/2$ punti di incrocio. Per $n = 1000$, abbiamo bisogno di 499.500 punti di incrocio. Anche se è possibile costruire un processore VLSI con questo numero di commutatori a transistor, non è altrettanto possibile mettere 1000 piedini su un chip non lo è. Così un commutatore crossbar è utile solo per piccoli uffici terminali.

Commutatori a divisione di spazio

Dividendo i commutatori crossbar in piccoli pezzi e interconnettendoli, è possibile costruire commutatori multistadio con molti punti di incrocio in meno. Questi sono chiamati **commutatori a divisione di spazio**. Due configurazioni sono illustrate in figura 2-39. Per mantenere semplice il nostro esempio, considereremo solo commutatori a tre stadi, ma sono possibili anche commutatori con più stadi. In questi esempi abbiamo un totale di N input e N output. Invece di costruire un singolo commutatore crossbar $N \times N$, lo costruiamo con più piccoli crossbar rettangolari. Nel primo stadio, ogni commutatore ha n input, quindi abbiamo bisogno di N/n commutatori per gestire tutte le N linee entranti. Il secondo stadio ha k commutatori, ognuno con N/n entrate e N/n uscite. Il terzo stadio

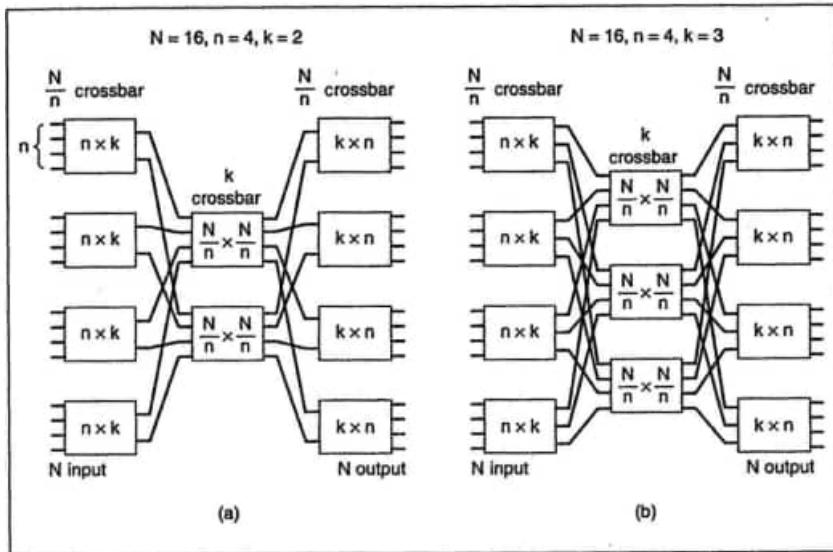


Fig. 2-39 Due commutatori a divisione di spazio con parametri differenti.

è una ripetizione del primo stadio, ma invertito da sinistra a destra. Ogni commutatore intermedio è connesso a ciascun commutatore di entrata e ad ognuno in uscita. Di conseguenza, è possibile connettere ogni entrata ad ogni uscita usando o il primo commutatore intermedio in figura 2-39 (a) o il secondo. Infatti, ci sono due percorsi disgiunti da ogni input ad ogni output, dipendenti dalla scelta del commutatore intermedio. In figura 2-39 (b) ci sono tre percorsi per ogni coppia di entrata/uscita. Con k stadi intermedi (k è un parametro di progetto), ci sono k percorsi disgiunti.

Calcoliamo ora il numero di punti di incrocio necessari per un commutatore a tre stadi. Nel primo stadio, ci sono N/n commutatori, ognuno con nk punti di incrocio, per un totale di Nk . Nel secondo stadio, ci sono k commutatori, ognuno con $(N/n)^2$ punti di incrocio. Il terzo stadio è come il primo. Sommando i tre stadi, otteniamo

$$\text{Numero di punti di incrocio} = 2kN + k(N/n)^2$$

Per $N = 1000$, $n = 50$ e $k = 10$, abbiamo bisogno di soli 24.000 punti di incrocio invece dei 499.500 richiesti da un commutatore 1000×1000 a singolo stadio.

Sfortunatamente, come al solito, non esistono pranzi gratis. Il commutatore può bloccarsi. Si consideri di nuovo la figura 2-39. Lo stadio 2 ha 8 ingressi, così possono connettersi al massimo 8 chiamate per volta. Quando arriva la nona chiamata, dovrà ottenere un segnale di occupato, anche se la destinazione è disponibile. Il commutatore di figura 2-39 (b) è migliore, potendo gestire un massimo di 12 chiamate invece di 8, ma usa un maggior numero di punti di incrocio. Qualche volta, quando si fa una chiamata telefonica, si può

ottenere un segnale di occupato prima di finire di comporre il numero. Ciò è probabilmente causato dal blocco di parte del percorso attraverso la rete.

Dovrebbe essere ovvio che più grande è k , più costoso è il commutatore e più bassa è la probabilità di un blocco. Nel 1953, Clos dimostrò che quando $k = 2n-1$, il commutatore non si blocca mai (Clos, 1953). Altri ricercatori hanno analizzato schemi di chiamate in grande dettaglio per costruire dei commutatori che in teoria potrebbero bloccarsi ma in pratica lo fanno solo raramente.

Commutatori a divisione di tempo

Un tipo completamente differente di commutatore è quello a **divisione di tempo**, mostrato in figura 2-40. Con la commutazione a divisione di tempo, le n linee in entrata sono scandite in sequenza per costruire un pacchetto d'ingresso a n posizioni. Ogni posizione ha k bit. Per commutatori T1, gli slot sono di 8 bit, elaborando 8000 pacchetti al secondo.

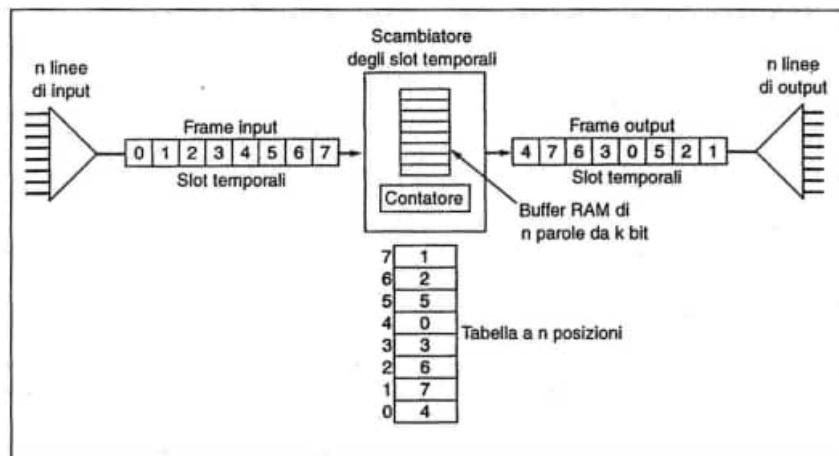


Fig. 2-40 Un commutatore a divisione di tempo.

Il cuore del commutatore a divisione di tempo è l'**intercambiatore di slot di tempo**, che accetta pacchetti input e produce pacchetti output in cui gli slot di tempo sono stati riordinati. In figura 2-40, lo slot input 4 è emesso per primo, poi lo slot 7 e così via. Alla fine il pacchetto in uscita è demultiplexato, con lo slot output 0 (slot di input 4) che va alla linea 0 e così via. In sostanza, il commutatore ha spostato un byte dalla linea di entrata 4 alla linea di uscita 0, un altro byte dalla linea di input 7 alla linea di uscita 1 e così via. Visto dall'esterno, l'intero sistema è un commutatore di circuito, anche se non ci sono connessioni fisiche.

Lo scambiatore di slot di tempo lavora come segue: quando un pacchetto input è pronto per essere elaborato, ogni slot (cioè ogni byte nel pacchetto input) è scritto nel buffer

RAM all'interno dello scambiatore. Gli slot sono scritti in ordine, così la parola di buffer i contiene lo slot i .

Dopo che tutti gli slot dei pacchetti input sono stati immagazzinati nel buffer, il pacchetto output viene costruito rileggendo le parole, ma in un ordine diverso. Un contatore va da 0 a $n - 1$. Al passo j , il contenuto della parola j di una tabella viene letto e usato per l'indirizzo della tabella RAM. Così se la parola 0 della tabella contiene un 4, la parola 4 del buffer RAM sarà letta per prima e il primo slot del pacchetto output sarà lo slot 4 del pacchetto input. Così i contenuti della tabella determinano quale permutazione del pacchetto input sarà generata in output e quale linea input sarà connessa a quale linea output.

I commutatori a divisione di tempo usano tabelle lineari, piuttosto che quadratiche, nel numero di linee, ma hanno un altro limite. È necessario immagazzinare nel buffer RAM n slot e poi leggerli di nuovo in un periodo di pacchetto di $125 \mu s$. Se ognuno di questi accessi in memoria impiega $T \mu s$, il tempo necessario per un elaborare un pacchetto è $2nT \mu s$, così abbiamo $2nT = 125$ o $n = 125/2T$. Per una memoria con 100 ns cicli di tempo, possiamo supportare al massimo 625 linee. Possiamo anche invertire tale relazione e determinare il ciclo di memoria richiesto per supportare un dato numero di linee. Come nel caso dei commutatori trasversali, è possibile concepire commutatori multistadio che dividono il lavoro in diverse parti e poi combinano i risultati per gestire un maggior numero di linee.

2.5 ISDN a banda stretta (N-ISDN)

Per più di un secolo l'infrastruttura primaria per la telecomunicazione internazionale è stata il sistema telefonico pubblico a commutazione di circuito. Questo sistema fu progettato per la trasmissione della voce, e oggi è inadeguato ai bisogni della comunicazione moderna. Anticipando le considerevoli richieste per un servizio totalmente digitale (cioè non come in figura 2-17 che è in parte digitale e in parte analogico), le società telefoniche mondiali e la PTT si riunirono nel 1984 sotto gli auspici della CCITT e si accordarono per costruire un nuovo sistema telefonico completamente digitale entro la prima parte del XXI secolo. Questo nuovo sistema, chiamato **ISDN (Integrated Services Digital Network)**, ha come obiettivo primario l'integrazione di servizi vocali e dati. È già disponibile in molti luoghi e il suo uso sta crescendo lentamente. Nei paragrafi successivi descrivremo cosa fa e come funziona. Per ulteriori informazioni, si vedano Dagdeviren et al. (1994); Kessler (1993).

2.5.1 Servizi ISDN

Il servizio principale dell'ISDN continuerà ad essere la voce, anche se verranno aggiunte molte funzioni avanzate. Per esempio molti manager di società hanno un pulsante di intercomunicazione sul loro telefono che suona istantaneamente ai segretari (senza sprecare tempo per preparare la connessione). Una caratteristica dell'ISDN sono i telefoni con più pulsanti per chiamate istantanee a telefoni ovunque nel mondo. Un'altra funzione sono i telefoni che quando squillano mostrano sul quadrante il numero telefonico, l'indirizzo e il nome del chiamante. Una versione più sofisticata di questa funzione permette a un

telefono di connettersi a un computer, in modo che un record dell'archivio del chiamante venga mostrato sullo schermo non appena arriva la chiamata. Per esempio, un agente di borsa potrebbe volere che, quando risponde al telefono, il portafoglio del chiamante sia già sullo schermo con i prezzi correnti di tutti i titoli del chiamante. Altri servizi vocali avanzati includono l'inoltramento della chiamata e chiamate a conferenza in tutto il mondo.

Alcuni servizi avanzati non vocali sono ad esempio la lettura a distanza di contatori di elettricità; oppure allarmi medici, antifurto e antincendio, che automaticamente chiamano rispettivamente l'ospedale, la polizia e i vigili del fuoco, e danno il loro indirizzo per velocizzare la risposta.

2.5.2 Architetture del sistema ISDN

È tempo ora di esaminare in dettaglio l'architettura ISDN, in particolare l'apparecchiatura dell'utente e l'interfaccia fra utente e società telefonica. L'idea chiave dietro ISDN è quella del **condotto digitale di bit (digital bit pipe)**, un canale virtuale tra l'utente e la società attraverso il quale scorrono i bit. Non importa se i bit partono da un telefono digitale, un terminale digitale, un fax digitale o da qualche altro dispositivo. Tutto quello che importa è che i bit possono scorrere in entrambe le direzioni lungo il canale.

Il condotto digitale supporta di norma più canali indipendenti per mezzo del multiplexing a divisione di tempo del flusso di bit. Il formato esatto del flusso di bit e il suo multiplexing è una parte della specifica di interfaccia del condotto digitale definita con molta attenzione. Sono stati sviluppati due standard principali per il condotto, uno standard a bassa banda per uso domestico, e uno a più alta banda per uso professionale che supporta più canali che sono identici al canale di uso domestico. Inoltre, le aziende possono avere più condotti se hanno bisogno di una capacità maggiore di quella che può fornire quello standard.

In figura 2-41 (a) vediamo la configurazione normale per uso domestico o piccole aziende. La società dei telefoni piazza un dispositivo di terminazione di rete, **NT1**, in base alle richieste dell'utente e lo connette all'ingresso ISDN nell'ufficio di servizio, a diversi chilometri di distanza, usando il doppino che era precedentemente usato per il telefono. La scatola NT1 ha un connettore nel quale può essere inserito un cavo a bus passivo. Al cavo si possono connettere fino a 8 telefoni, terminali, allarmi e altri dispositivi ISDN, in modo simile a come i dispositivi vengono connessi a una LAN. Dal punto di vista dell'utente, il confine della rete è il connettore su NT1.

Se l'azienda è grande, il modello di figura 2-41 (a) è inadeguato perché di solito ci sono conversazioni telefoniche in corso simultaneamente in numero maggiore di quante il bus ne possa gestire. Perciò si usa il modello di figura 2-41 (b). In questo modello troviamo un dispositivo, **NT2**, chiamato **PBX (Private Branch eXchange)**, che è connesso all'NT1 e che fornisce la vera interfaccia per telefoni, terminali e altre apparecchiature. Un PBX ISDN concettualmente non è molto differente da un commutatore ISDN, anche se questo è di solito più piccolo e non può gestire così tante conversazioni allo stesso tempo.

CCITT ha definito quattro **punti di riferimento**, chiamati R, S, T e U, fra i vari dispositivi. Questi sono evidenziati in figura 2-41. Il punto di riferimento U è la connessione fra il commutatore ISDN nell'ufficio della società e l'NT1. Al momento è un doppino a due

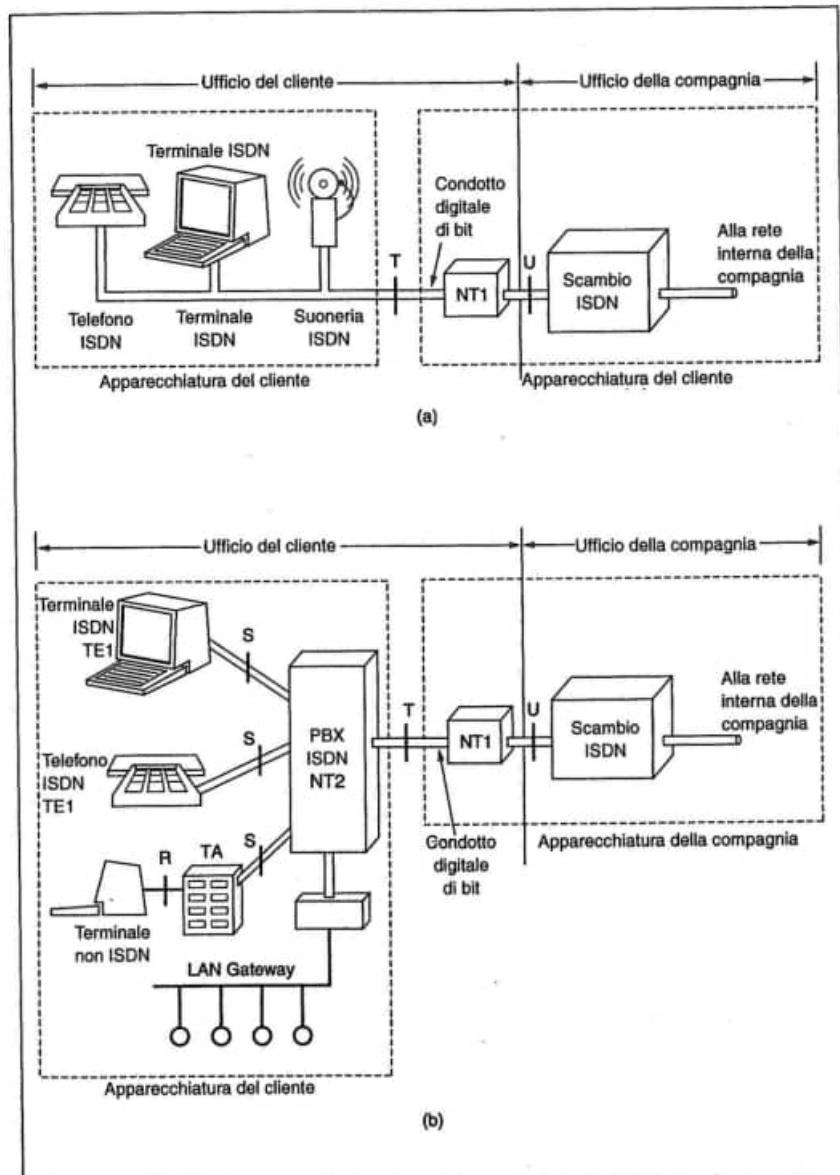


Fig. 2-41 (a) Esempio di sistema ISDN per uso domestico. (b) Esempio di un sistema ISDN con un PBX per uso in grandi società.

fili di rame, ma a un certo momento nel futuro potrà essere sostituito da fibre ottiche. Il punto di riferimento T è quello che il connettore su NT1 fornisce all'utente. Il punto di riferimento S è l'interfaccia fra il PBX ISDN e i terminali ISDN. Il punto di riferimento R è la connessione fra l'adattatore per il terminale e terminali non ISDN. Molti diversi tipi di interfacce saranno usati a R.

2.5.3 L'interfaccia ISDN

Il condotto ISDN supporta canali multipli in multiplexing a divisione di tempo. Sono stati standardizzati diversi tipi di canali:

- A – Canale telefonico analogico a 4 kHz.
- B – Canale digitale PCM a 64 kbps per voce o dati.
- C – Canale digitale a 8 o 16 kbps.
- D – Canale digitale a 16 kbps per segnalazione fuori banda.
- E – Canale digitale a 64 kbps per segnalazioni interne ISDN.
- H – Canale digitale a 384, 1536 o 1920 kbps.

Non era intenzione della CCITT permettere una combinazione arbitraria dei canali sul condotto digitale. Finora sono state standardizzate tre combinazioni:

1. **Tasso base:** 2B + 1D.
2. **Tasso primario:** 23B + 1D (USA e Giappone) o 30B + 1D (Europa).
3. **Irido:** 1A + 1C.

I canali a tasso base e primario sono illustrati in figura 2-42.

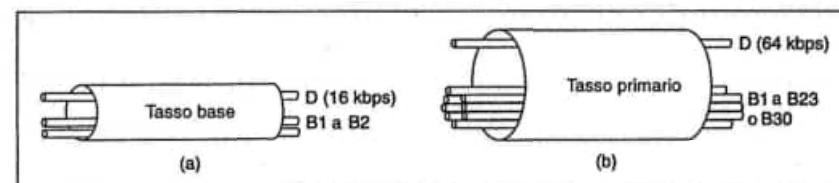


Fig. 2-42 (a) Canale digitale a tasso base. (B) Canale digitale a tasso primario.

Il tasso base dovrebbe essere visto come un rimpiazzamento del POTS (Plain Old Telephone Service) per uso domestico o piccole società. Ognuno dei canali B a 64 kbps può gestire un singolo canale a voce PCM con campioni di 8 bit presi 8000 volte al secondo (64 kbps qui significa 64.000, non 65.536). La segnalazione è su un canale separato D a 16 kbps, così 64 kbps pieni sono disponibili per l'utente (come nel sistema CCITT a 2.048 Mbps e a differenza del sistema americano e giapponese T1). Poiché l'ISDN è così incentrata su canali a 64 kbps, ci riferiamo ad esso come N-ISDN (ISDN a banda stretta), per distinguerlo da ISDN a larga banda che sarà discusso dopo.

L'interfaccia a tasso primario è destinata per l'uso al punto di riferimento T per società

che possiedono un PBX. Ha 23 canali B e 1 canale D (a 64 kbps) negli Stati Uniti e nel Giappone e 30 canali B e 1 canale D (a 64 kbps) in Europa. La scelta 23B + 1D fu fatta per permettere a un pacchetto ISDN di adattarsi bene nel sistema T1 di AT&T, così come la scelta 30B + 1D fu fatta per permettere a un pacchetto ISDN di adattarsi bene nel sistema CCITT a 2,048 Mbps. Lo slot di tempo di 32 s nel sistema CCITT è usato per il framing e la manutenzione generale della rete. La quantità di canali D per canale B nel tasso primario è molto minore che nel tasso base, perché non ci si aspetta che lì ci siano molti pacchetti di dati a bassa banda.

2.5.4 Prospettiva su N-ISDN

N-ISDN fu un tentativo massiccio di sostituire il sistema telefonico analogico con uno digitale adatto per il traffico sia vocale che dati. Raggiungere un accordo mondiale sullo standard per l'interfaccia per il tasso base si supponeva portasse a una grande richiesta di utenti per apparecchiature ISDN, portando così a una produzione di massa, economie di scala e chip VLSI ISDN non costosi. Sfortunatamente, mentre il processo di standardizzazione impiegò degli anni a svilupparsi, la tecnologia in quest'area si mosse molto rapidamente, così una volta che ci si accordò sullo standard, esso era già obsoleto.

Per uso domestico, la richiesta più grande per nuovi servizi sarà senza dubbio per il video a richiesta. Sfortunatamente, il tasso base ISDN manca della necessaria larghezza di banda di due ordini di grandezza. Per uso affari la situazione è persino più desolata. Allo stato attuale le LAN disponibili offrono almeno 10 Mbps e vengono rimpiazzate da LAN a 100 Mbps. Offrire un servizio a 64 kbps per affari era una proposta seria negli anni ottanta. Negli anni novanta è uno scherzo.

Abbastanza stranamente, ISDN può ancora essere salvata, ma da un'applicazione totalmente inaspettata: l'accesso a Internet. Varie società adesso vendono adattatori ISDN che combinano i canali 2B + D in un singolo canale digitale a 144 kbps. Molti fornitori di servizi Internet supportano anche questi adattatori. Il risultato è che gli utenti possono accedere a Internet su una connessione a 144 kbps totalmente digitale, invece di un connessione analogica via modem a 28,8 kbps. Per molti utenti Internet, guadagnare un fattore di cinque per scaricare pagine World Wide Web piene di grafica è un servizio che vale la pena di avere. Mentre la B-ISDN è anche meglio, la N-ISDN a 144 kbps è ora a un prezzo affrontabile e quella può essere la sua nicchia principale per i prossimi anni.

2.6 ISDN a larga banda e ATM

Quando finalmente la CCITT comprese che ISDN a banda stretta non avrebbe entusiasmato il mondo, cercò di pensare a un nuovo servizio. Il risultato fu **ISDN a larga banda (B-ISDN)**, un circuito digitale virtuale per spostare pacchetti di dimensione fissata (celle) dalla sorgente alla destinazione a 155 Mbps (in realtà 156 Mbps, come detto prima). Dal momento che questo tasso di dati è sufficiente persino per HDTV (non compressa), sarà possibile soddisfare anche i più grandi ingordi di banda per almeno qualche anno. Mentre ISDN a banda stretta è un primo timido passo nell'era digitale, ISDN a larga banda è un coraggioso salto verso l'ignoto. I benefici sono enormi, un aumento di banda

sulla ISDN a banda stretta di un fattore di 2500, ma la sfida è ugualmente molto grande (Ambruster, 1995).

Per iniziare, la ISDN a larga banda si basa sulla tecnologia ATM che, come abbiamo visto brevemente nel capitolo 1, è fondamentalmente una tecnologia a commutazione di pacchetto, non a commutazione di circuito (sebbene possa emularla molto bene). Al contrario, sia la PSTN esistente che la ISDN a banda stretta sono tecnologie a commutazione di circuito. Una enorme quantità di esperienza ingegneristica nella commutazione di circuito sarà resa obsoleta da questo cambiamento. Passare dalla commutazione di circuito a quella di pacchetto è davvero un mutamento di paradigma.

Come se non bastasse, ISDN a larga banda non può sfruttare l'esistente doppino per qualsiasi distanza sostanziale. Ciò significa che introdurla richiederà di sostituire la maggior parte dei circuiti locali e mettere ovunque doppi di categoria 5 o la fibra ottica (Stephens, Banwell, 1995). Inoltre, non si può usare per la commutazione di pacchetto la divisione di spazio o la divisione di tempo. Occorreranno nuovi commutatori basati su principi differenti che vadano a velocità molto più elevate. Le sole cose che possono essere salvate sono le fibre ad area geografica e le dorsali.

In breve, gettare via le conoscenze accumulate in 100 anni più gli investimenti in impianti esterni e interni che valgono centinaia di miliardi di dollari non è esattamente un piccolo passo da intraprendere con leggerezza. Non di meno, alle società telefoniche è chiaro che se non lo fanno loro, lo faranno probabilmente le società della televisione via cavo, pensando al video a richiesta. Anche se è probabile che sia la PSTN esistente, sia l'ISDN a banda stretta rimarranno con noi per i prossimi dieci anni o forse anche più a lungo, il futuro a lungo termine è probabilmente ATM, perciò la studieremo in gran dettaglio in questo libro, a partire dal livello fisico in questo capitolo.

2.6.1 Circuiti virtuali contro commutazione di circuito

Il servizio ISDN a larga banda è un compromesso tra la pura commutazione di circuito e la pura commutazione di pacchetto. L'attuale servizio offerto è orientato alla connessione, ma è implementato internamente con la commutazione di pacchetto, non con quella di circuito.

Sono offerti due tipi di connessione: circuiti virtuali permanenti e circuiti virtuali commutati. I circuiti virtuali permanenti sono richiesti dal cliente (ad es. spedendo un fax alla società) e tipicamente sopravvivono per mesi o anni. I circuiti virtuali commutati sono come le chiamate telefoniche: essi vengono predisposti dinamicamente a richiesta e cancellati immediatamente dopo.

In una rete a commutazione di circuito, fare una connessione significa in realtà che un percorso fisico è stabilito dalla sorgente alla destinazione attraverso la rete, certamente quando si usano commutatori a divisione di spazio (con i commutatori a divisione di tempo, il concetto di "percorso fisico" diventa un po' confuso). In una rete a circuito virtuale, come ATM, quando si stabilisce un circuito, quello che accade veramente è che il tragitto scelto dalla sorgente alla destinazione e tutti i router lungo la strada costruiscono delle voci in una tabella in modo da instradare qualsiasi pacchetto su quel circuito virtuale. Essi hanno anche l'opportunità di riservare delle risorse per un nuovo circuito. La figura 2-43 mostra un singolo circuito virtuale dall'host H_1 all'host H_5 attraverso i router A, E, C e D.

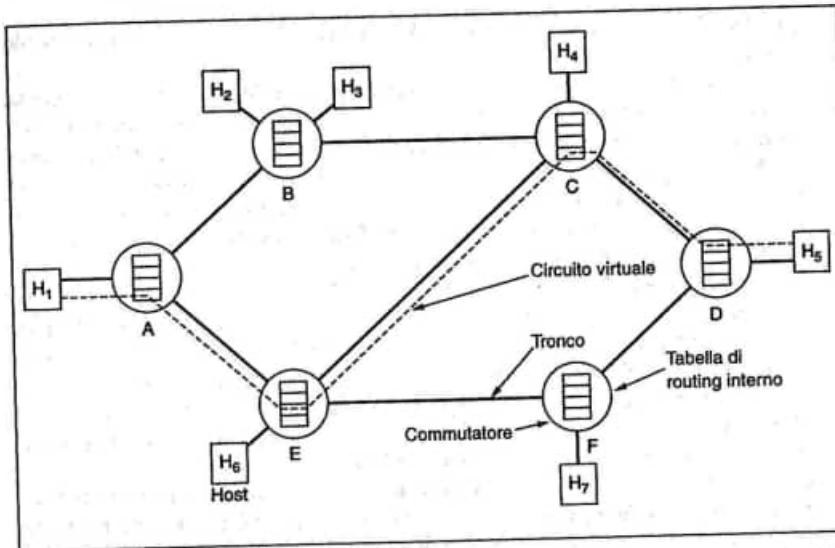


Fig. 2-43 Le linee punteggiate mostrano un circuito virtuale. È definito dalle voci della tabella all'interno dei commutatori.

Quando arriva un pacchetto, il commutatore ispeziona l'intestazione del pacchetto per trovare a quale circuito virtuale appartenga. Poi cerca quel circuito virtuale nelle sue tabelle per determinare su quale linea di comunicazione trasmetterlo. Esamineremo questo processo in maggior-dettaglio nel capitolo 5.

Il significato del circuito virtuale permanente tra H_1 e H_5 in figura 2-43 dovrebbe adesso essere chiaro. È un accordo fra il cliente e la società che i commutatori tengano sempre delle entrate in una tabella per una destinazione particolare, anche quando non ci sia stato traffico per mesi. Chiaramente, un tale accordo costa risorse (certamente spazio nella tabella all'interno dei commutatori e anche eventualmente banda riservata e buffer), perciò c'è sempre una tariffa mensile per circuito virtuale permanente. Il vantaggio su un circuito virtuale comunitato è che non c'è l'intervallo necessario alla configurazione. I pacchetti possono muoversi istantaneamente. Per alcune applicazioni, come verifica di carte di credito, risparmiare pochi secondi su ogni transazione, può facilmente valere la spesa.

Al contrario, una linea a noleggio da H_1 ad H_5 in una rete a circuito comunitato con la topologia di figura 2-43 e commutatori a divisione di tempo manterebbe in realtà i punti di incrocio chiusi per mesi e riserverebbe permanentemente larghezza di banda sulle dorsali, sia su bande FDM che a slot di tempo (una "linea" a noleggio può essere segmentata se nessuna linea diretta è disponibile). Tale soluzione è ovviamente molto più dispendiosa in termini di risorse quando non è utilizzata da un circuito virtuale.

2.6.2 Trasmissione in reti ATM

Come abbiamo visto, ATM sta per modo di trasferimento *asincrono*. Questo modo può essere confrontato alla portante sincrona T1 illustrata in figura 2-44 (a). Un pacchetto T1 viene generato precisamente ogni 125 µs. Questo battito è governato da un orologio principale. Lo slot k di ogni pacchetto contiene un byte di dati dalla stessa sorgente. T1 è sincrono.

ATM, al contrario, non ha bisogno che le celle si alternino rigidamente fra le varie sorgenti. La figura 2-44 (b) mostra le celle su una linea da varie fonti, con nessuno schema particolare. Le celle arrivano casualmente da sorgenti diverse.

Inoltre, non si richiede nemmeno che il flusso di celle provenienti da un computer sia continuo. Sono possibili buchi fra le celle di dati, che vengono riempiti da speciali celle inattive.



Fig. 2-44 (a) Modo di trasmissione sincrona. (b) Modo di trasmissione asincrona.

ATM non standardizza il formato per le celle trasmittenti. Piuttosto, specifica che è permesso solo spedire celle individuali ma specifica anche che le celle possono essere racchiuse in una portante come la T1, T3, SONET o FDDI (una LAN a fibra ottica). Per questi esempi, esistono degli standard che stabiliscono come le celle vadano racchiuse nei pacchetti che questi sistemi forniscono.

Nello standard originale ATM, il tasso primario era 155,52 Mbps, con un tasso aggiuntivo a quattro volte tale velocità (622,08 Mbps). Questi tassi furono scelti per essere compatibili con SONET, il framing standard usato con le connessioni a fibra ottica nel sistema telefonico. È anche previsto ATM su T3 (44,736 Mbps) ed FDDI (100 Mbps).

Il mezzo di trasmissione per ATM sono normalmente le fibre ottiche, ma per percorsi sotto i 100 m, sono accettabili anche il cavo coassiale o il doppino di categoria 5. Le fibre possono correre per molti chilometri. Ogni collegamento va da un computer a un commutatore ATM o tra due commutatori ATM. In altre parole, tutte le connessioni ATM sono punto-a-punto (non come le LAN, che ha molti mittenti e molti riceventi sullo stesso cavo). Il multicasting è ottenuto mediante una cella che entra in un commutatore su una linea ed esce su linee multiple. Ogni connessione punto a punto è unidirezionale. Per le

operazioni full-duplex occorrono due connessioni parallele, una per il traffico in ciascuna direzione.

Il sottostrato ATM dipendente dal mezzo fisico riguarda il recupero dei bit su e fuori dal filo. Hardware differente è richiesto per eavi e fibre differenti, a seconda della velocità e della codifica della linea. Lo scopo del sottolivello per la convergenza della trasmissione è quello di fornire un'interfaccia uniforme per il livello ATM in entrambe le direzioni. All'esterno, il livello ATM fornisce una sequenza di celle e il sottolivello PMD le codifica come richiesto e le emette in forma di flusso di bit.

All'interno, il sottolivello PMD prende i bit provenienti dalla rete e consegna un flusso di bit al sottolivello TC. I confini della cella non sono segnati in alcun modo. È compito del sottolivello TC capire dove finisce una cella e dove comincia la successiva. Questo lavoro non è solo difficile, ma teoricamente impossibile. Perciò il sottolivello TC deve affrontare un lavoro molto impegnativo. Poiché il sottolivello TC sta effettuando il framing della cella, è una funzione data link, quindi la discuteremo nel capitolo 3. Per informazioni aggiuntive sul livello fisico ATM, si vedano Rao, Hatamian (1995).

2.6.3 Commutatori ATM

In letteratura sono stati descritti molti progetti di commutatore di cella ATM. Alcuni di questi sono stati implementati e "testati". In questo paragrafo daremo una breve introduzione ai principi di progetto di commutatore di cella ATM e li illustreremo con alcuni esempi. Per maggiori informazioni, si vedano De Prycker (1993); Garcia-Haro, Jajszczyk (1994); Partridge (1994). Per un commutatore ATM ottimizzato per far eseguire IP su ATM, si vedano Parulkar et al. (1995).

Il modello generale per un commutatore di cella ATM è mostrato in figura 2-45. Esso ha un certo numero di linee in entrata e un certo numero di linee in uscita, quasi sempre lo stesso numero (poiché le linee sono bidirezionali). I commutatori ATM sono generalmente sincroni nel senso che durante un ciclo, da ogni linea input viene presa una cella (se è presente), passata alla struttura di commutazione interna e alla fine trasmessa su una linea output appropriata.

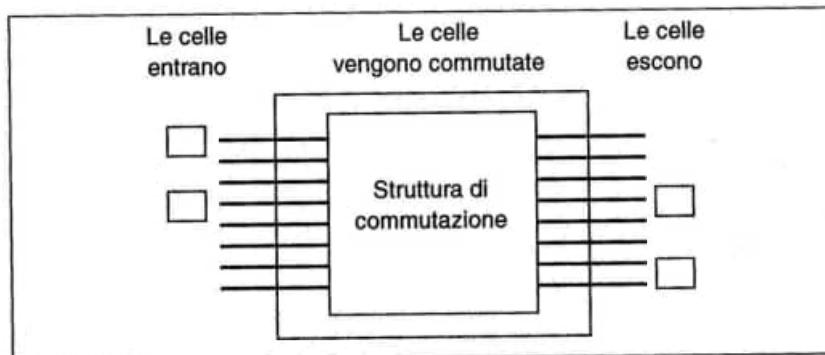


Fig. 2-45 Un generico commutatore ATM.

2.6 ISDN a larga banda e ATM

I commutatori possono essere connessi in serie, cioè si possono impiegare diversi cicli prima che una cella in entrata si presenti su una linea output. Le celle in realtà arrivano sulle linee input asincronicamente, così c'è un orologio principale che segna l'inizio del ciclo. Una cella completamente arrivata al momento del battito può essere commutata durante tale ciclo. Ogni cella non arrivata completamente deve aspettare fino al ciclo successivo.

Le celle arrivano a velocità ATM, circa 150 Mbps. Questo produce poco più di 360.000 celle/s, il che significa che il tempo di un ciclo del commutatore deve essere di circa 2,7 μs. Un commutatore commerciale potrebbe avere da qualche parte da 16 a 1024 linee input, il che significa che deve essere preparato ad accettare e cominciare a commutare un gruppo da 16 fino a 1024 celle ogni 2,7 μs. A 622 Mbps, un nuovo gruppo di celle viene inserito nella struttura di commutazione ogni circa 700 ns. Il fatto che le celle siano di lunghezza fissata e corta (53 byte) rende possibile costruire tali commutatori. Con i pacchetti a lunghezza variabile più lunga, la commutazione ad alta velocità sarebbe più complessa, il che spiega perché ATM usa celle corte di lunghezza fissa.

Tutti i commutatori ATM hanno due obiettivi:

1. Commutare tutte le celle col minimo tasso di scarto.
2. Non riordinare le celle su un circuito virtuale.

L'obiettivo 1 dice che è possibile perdere delle celle in caso di emergenza, ma che il tasso di scarto dovrebbe essere il più piccolo possibile. È probabilmente accettabile un tasso di scarto di una cella su 10^{12} . Su un commutatore grande, questo tasso è di circa 1 o 2 celle all'ora.

L'obiettivo 2 dice che le celle che arrivano su un circuito virtuale in un certo ordine devono anche partire in quell'ordine con nessuna eccezione, mai. Questo vincolo rende il progetto di un commutatore considerevolmente più difficile, ma è richiesto dallo standard ATM. Un problema che capita in tutti i commutatori ATM è cosa fare se le celle che arrivano a due o più linee input vogliono andare alla stessa porta nello stesso ciclo. Risolvere questo problema è una delle questioni chiave nel progetto di tutti i commutatori ATM. Una non soluzione è di prendere una cella da consegnare e scartare il resto. Dal momento che questo algoritmo viola l'obiettivo 1, non lo possiamo usare.

Il successivo tentativo è di fornire una coda per ogni linea input. Se due o più celle sono in conflitto, se ne sceglie una per la consegna, e le restanti sono mantenute per i cicli successivi. La scelta può essere fatta a caso, o ciclicamente, ma non si dovrebbe favorirne nessuna; per esempio, non bisogna favorire la linea col numero più basso per evitare di dare alle linee con numero basso un servizio migliore di quello per le linee con numeri alti. La figura 2-46 (a) illustra la situazione all'inizio del ciclo 1, in cui le celle sono arrivate su quattro linee input, destinate rispettivamente alle linee 2, 0, 2 e 1. Poiché c'è conflitto per la linea 2, può essere scelta solo una cella. Supponiamo che sia la uno sulla linea input 0. All'inizio del ciclo 1, mostrata in figura 2-46 (b), sono state emesse tre celle, ma la cella sulla linea 2 è stata mantenuta e sono arrivate due celle in più. Solo all'inizio del ciclo 4, mostrato in figura 2-46 (d), abbiamo che tutte le celle hanno liberato il commutatore.

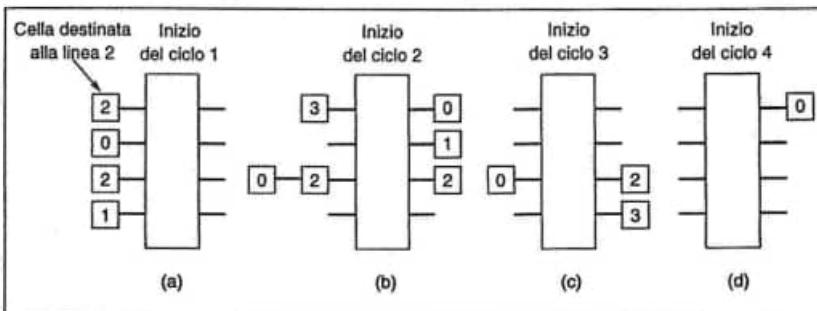


Fig. 2-46 Accodamento all'ingresso di un commutatore ATM.

Il problema con la coda input è che quando una cella deve essere tenuta in sospeso, questa blocca l'avanzare delle celle dietro di essa, anche se queste potrebbero essere altrimenti commutate. Questo effetto è chiamato **blocco della testa della linea**. È qualcosa di più complicato di quello mostrato qui, poiché in un commutatore con 1024 linee input, i conflitti non possono essere notati fino a che le celle sono effettivamente nel commutatore e competono per la linea output. Mantenere una cella sulla sua coda input fino a quando un segnale torna indietro dicendo che lo ha fatto attraverso il commutatore richiede logica extra, un percorso inverso di segnalazione e maggior ritardo. A volte si mettono le celle perdenti su un bus di ricircolo che le spedisce indietro dal lato input, ma il commutatore deve stare attento a dove le mette, per evitare di consegnare fuori ordine le celle dello stesso circuito virtuale.

Uno schema alternativo, che non subisce il blocco di testa della linea, costruisce la coda sul lato output, come in figura 2-47. Qui abbiamo lo stesso schema di arrivo della cella, ma ora quando due celle vogliono andare alla stessa linea output nello stesso ciclo, entrambe passano attraverso il commutatore. Una di loro è posta sulla linea output e l'altra è messa in coda sulla linea output, come in figura 2-47 (b). Qui, commutare tutti i pacchetti richiede solo tre cicli, invece di quattro. Karol e altri (1987) hanno dimostrato che la coda in output è generalmente più efficiente della coda in input.

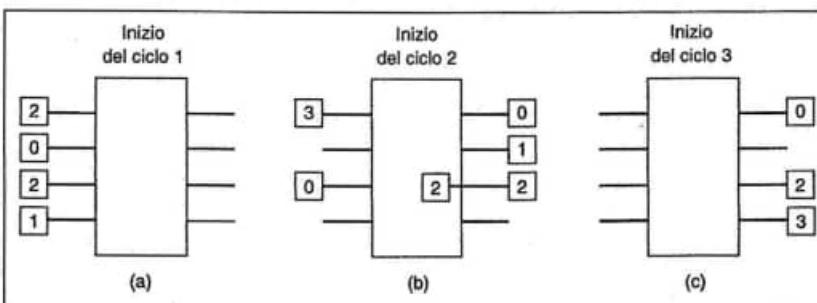


Fig. 2-47 Coda in output a un commutatore ATM.

Il commutatore knockout

Diamo adesso uno sguardo con maggior attenzione al progetto di un commutatore ATM che usa la coda in output. È chiamato **commutatore knockout** (Yeh et al., 1987) ed è illustrato in figura 2-48 per otto linee input e quattro linee output. Ogni linea input è connessa a un bus sul quale le celle entranti sono diffuse nel ciclo in cui arrivano. Un gestore per ogni bus semplifica considerevolmente il progetto e la temporizzazione.

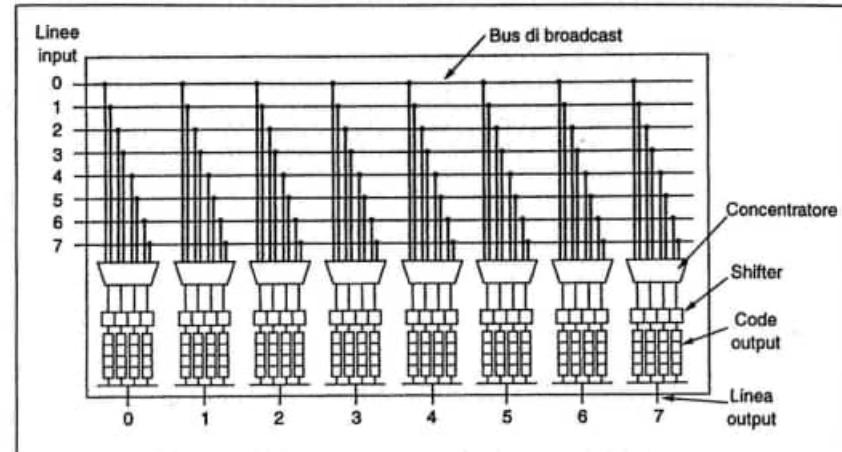


Fig. 2-48 Diagramma semplificato del commutatore knockout.

Per ogni cella che arriva, l'hardware esamina l'intestazione della cella per trovare informazioni sul suo circuito virtuale, lo cerca nella tabella di routing (vedi figura 2-43) e abilita il corretto punto di incrocio. La cella quindi viaggia lungo il suo bus fino a quando arriva al punto di incrocio abilitato, e quel punto punta a sud verso la sua linea di output. È possibile che più celle, anche tutte, vadano alla stessa linea output. È anche possibile che una cella venga trasmessa a diverse linee output solo abilitando diversi punti di incrocio sul suo bus di diffusione.

Il modo più semplice di gestire le collisioni sarebbe quello di mettere in un buffer tutte le celle dal lato output. Comunque, con un commutatore con 1024 linee input, nel caso peggiore sarebbero necessari 1024 buffer output. In pratica, questa situazione è molto improbabile che accada, così una ottimizzazione ragionevole è fornire molti meno buffer output, per esempio n .

Nell'eventualità improbabile che in un ciclo arrivino più celle di quante ne possano essere gestite, il concentratore su ogni linea seleziona n celle per la coda, scartando il resto. Il concentratore è un circuito che necessita di intelligenza, per eseguire correttamente questa selezione, usando un torneo a eliminazione (knockout) simile ai quarti di finale, semifinali e finali in molti tornei sportivi.

Concettualmente, tutte le celle selezionate vanno in una singola coda output (a meno che sia piena, nel cui caso le celle vengono scartate). Tuttavia, mettere tutte le celle in una

coda singola nel tempo allocato non è possibile, così la coda di output è simulata da code multiple. Le celle selezionate vanno in uno shifter, che dopo le distribuisce uniformemente sulle n code output usando un token per tenere traccia di quale sia la coda successiva. Variando n , i progettisti possono cambiare il costo del commutatore rispetto il tasso atteso di scarto di celle.

Il commutatore Batcher-banyan

Il problema col commutatore knockout è che esso è fondamentalmente un crossbar, quindi il numero di punti di incrocio cresce col quadrato del numero di linee. Questo fatto, che è un problema per la commutazione di circuito, lo è anche per quella a pacchetto. La soluzione per la commutazione di circuito è la divisione di tempo, che riduce di molto il numero di punti di incrocio, al punto di richiedere un commutatore multistadio. Una soluzione simile è disponibile per la commutazione di pacchetto.

Questa soluzione è chiamata commutatore **Batcher-banyan**. Come i commutatori knockout, quelli Batcher-banyan sono sincroni e accettano un insieme di celle (zero o più per linea input) in ciascun ciclo. Persino un semplice Batcher-banyan è più complicato dei commutatori a divisione di spazio di figura 2-39, così lo presenteremo in dettaglio. In figura 2-49 (a) abbiamo un commutatore banyan 8×8 a tre stadi, così chiamato perché i suoi fili somigliano alle radici di un albero di baniano. In tutti i commutatori banyan esiste solo un percorso da ogni linea input a ogni linea output. Il routing è fatto cercando la linea output per ciascuna cella (usando l'informazione del circuito virtuale e le tabelle). Questo numero binario a 3 bit è poi posto davanti alla cella, poiché sarà usata per il routing attraverso il commutatore.

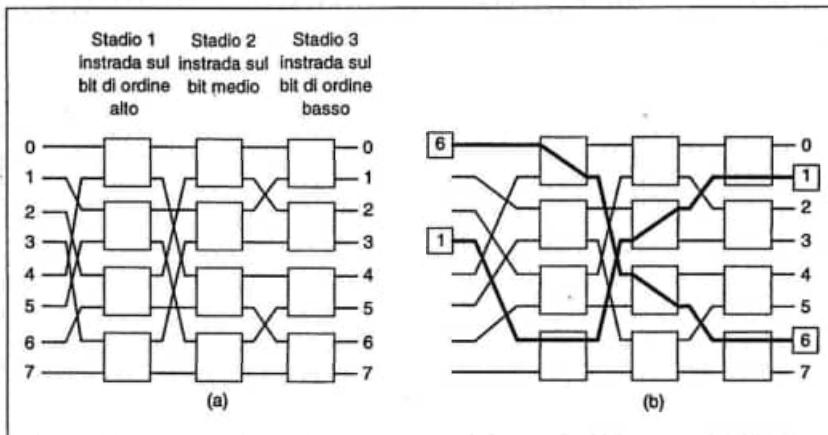


Fig. 2-49 (a) Un commutatore banyan con otto linee input e otto linee output. (b) I percorsi che due celle seguono attraverso il commutatore banyan.

Ciascuno dei 12 elementi di commutazione nel commutatore banyan ha due ingressi e

due uscite. Quando una cella arriva all'elemento di commutazione, viene esaminato un bit del numero della linea output e, basandosi su quello, la cella viene instradata o alla porta 0 (quella superiore) o alla porta 1 (quella inferiore). Nel caso di collisione, una cella viene instradata e l'altra scartata.

Un commutatore banyan analizza il numero della linea output da sinistra a destra, così lo stadio 1 esamina il bit più a sinistra (cioè quello di ordine più alto), lo stadio 2 esamina il bit in mezzo e lo stadio 3 esamina il bit più a destra (cioè quello di ordine più basso).

In figura 2-49 (b) abbiamo due celle: una cella sulla linea input 0 che si dirige verso la linea output 6 e una cella sulla linea input 3 che si dirige verso la linea output 1. Per la prima cella l'indirizzo binario di output è 110, così passa attraverso i tre stadi usando rispettivamente le porte inferiore, inferiore e superiore. Allo stesso modo, l'altra cella, etichettata in binario 001, usa rispettivamente le porte superiore, superiore e inferiore.

Sfortunatamente, capita una collisione in un commutatore banyan quando due celle in entrata vogliono uscire da un elemento di commutazione attraverso la stessa porta allo stesso tempo. In figura 2-50 (a) è illustrata una serie di tali collisioni. Nello stadio 1, le collisioni coinvolgono le celle che sono dirette verso le seguenti coppie di linee output: (5, 7), (0, 3), (6, 4), (2, 1). Supponiamo che queste collisioni siano risolte a favore di 5, 0, 4 e 1. Nel secondo stadio otteniamo delle collisioni tra (0, 1) e (5, 4). Qui lasciamo vincere 1 e 5 e queste sono instradate alle linee output correttamente.

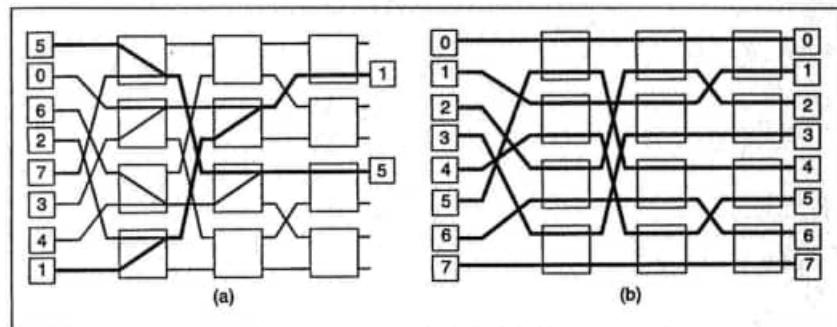


Fig. 2-50 (a) Celle che collidono in un commutatore banyan. (b) Routing senza collisione attraverso un commutatore banyan.

Adesso esaminiamo la figura 2-50 (b). Tutte le otto celle passano senza alcuna collisione. La conclusione è: a seconda dell'input, il commutatore banyan può fare un buono o un cattivo lavoro di instradamento.

L'idea che sta alla base del commutatore Batcher banyan è di porre un commutatore davanti al banyan per permutare le celle in una configurazione che il banyan può gestire senza perdite. Per esempio, se le celle in entrata sono ordinate per destinazione e presentate sulle linee input 0, 2, 4, 6, 1, 3, 5 e 7, in quest'ordine per quanto necessario (secondo quante celle ci sono), allora il commutatore banyan non perde celle.

Per ordinare le celle in entrata usiamo un Batcher, inventato da K.E. Batcher (1968). Come

i banyan e i knockout, anch'esso è sincrono e lavora con cicli discreti. Un Batcher è costituito da 2×2 elementi di commutazione, ma questi lavorano differentemente da quelli nel banyan. Quando un elemento riceve due celle, confronta numericamente i loro indirizzi output (così non solo un bit) e instrada la cella con indirizzo più alto alla porta nella direzione della freccia e quella con il più basso nell'altro senso. Se c'è solo una cella, essa va verso la porta nella direzione opposta a quella che punta la freccia.

Un Batcher per otto linee è illustrato sulla sinistra in figura 2-51. Lo stadio 1 ordina le celle entranti a coppie. I due stadi successivi fanno una fusione a quattro vie. I tre stadi finali fanno una fusione a otto vie. In generale, per n linee, la complessità di un Batcher cresce come $n \log^2 n$. Quando sono presenti k celle sulle linee input, il Batcher ordina le celle sulle prime k linee output.

Dopo essere uscite dal Batcher, le celle subiscono un rimescolamento e sono poi inserite in un commutatore banyan. Il risultato finale è che ogni cella appare sulla linea output corretta all'estremità lontana del commutatore banyan.

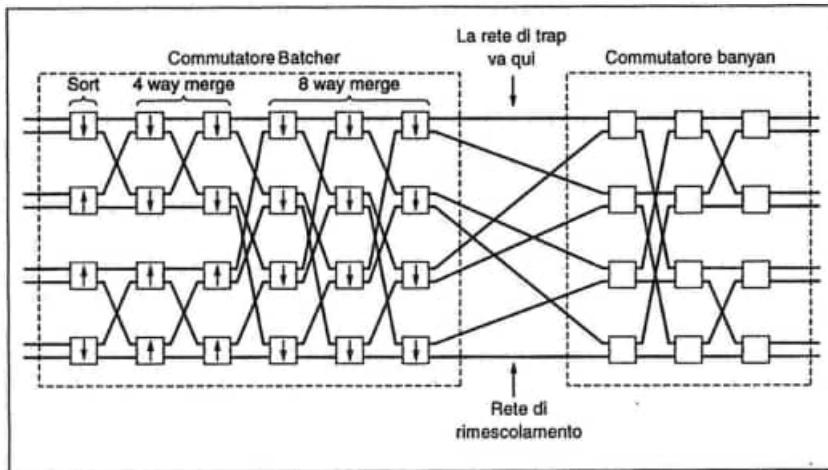


Fig. 2-51 La struttura di commutazione per un Batcher-banyan.

Un esempio di come lavora la struttura di commutazione combinata Batcher-banyan è in figura 2-52. Qui le celle sono presenti sulle linee input 2, 3, 4 e 5, dirette per le linee output 6, 5, 1, e 4 rispettivamente. Inizialmente, le celle per 5 e 6 entrano nello stesso elemento di commutazione. La cella 6 ha un indirizzo più alto, così esce nella direzione della freccia; la cella 5 va nell'altra direzione. Qui non avvengono scambi. Con le celle 1 e 4 avviene uno scambio: con la cella 4 che entra dal basso ma esce dall'alto. Le linee marcate mostrano i percorsi lungo tutta la strada fino alla fine.

Si noti che alla fine del Batcher, le quattro celle sono ammucchiate in ordine nella parte superiore. Poi sono fatte passare attraverso una rete di mescolamento e inserite nel banyan, che è capace di elaborare senza collisioni.

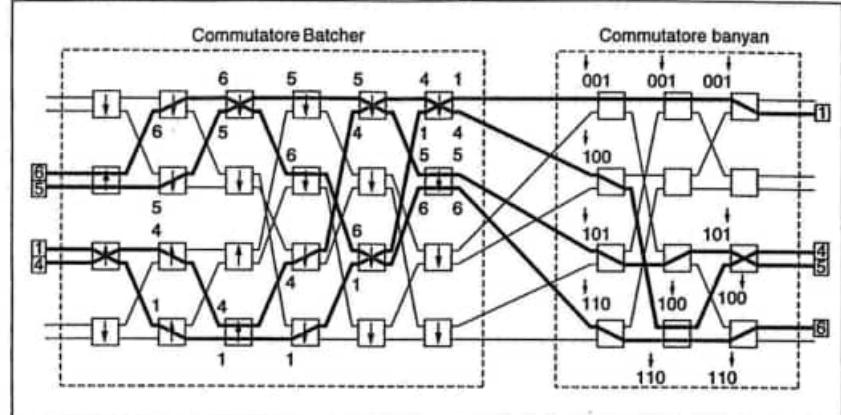


Fig. 2.52 Un esempio con quattro celle usando il commutatore Batcher banyan.

In teoria, il Batcher banyan è un buon commutatore ATM, ma ci sono due complicazioni che abbiamo ignorato: collisioni sulla linea output e multicasting. Se due o più celle sono dirette alla stessa linea output, il Batcher banyan non può gestirle, così dobbiamo usare qualche tipo di buffer. Un modo per risolvere questo problema è di inserire una rete trappola fra il commutatore Batcher e il commutatore banyan. Il compito della rete trappola è di filtrare i duplicati e farli ricircolare nei cicli susseguenti, mantenendo nel frattempo l'ordine delle celle sul circuito virtuale (ora dovrebbe essere chiaro che il requisito di ordinamento costituisce un problema molto più di quanto potesse apparire prima). I commutatori commerciali devono gestire anche il multicast.

Il primo Batcher-banyan fu progettato da Huang e Knauer (1984). Era chiamato Starlite. Poi venne Moonshine (Hui, 1987) e Sunshine (Giacopelli et al., 1991). Si deve ammettere che questi ragazzi hanno senso dell'humor. Starlite, Moonshine e Sunshine differiscono principalmente nel circuito tranello e nella gestione del multicast.

2.7 Radio cellulare

Il sistema telefonico tradizionale (anche quando ISDN a banda larga sarà pienamente in esercizio) non sarà capace di soddisfare un gruppo importante di utenti: le persone in viaggio. Di conseguenza, assistiamo alla corsa a costruire un sistema per le comunicazioni che usi onde radio invece di fibre e fili. Questo sistema giocherà negli anni a venire un ruolo di crescente importanza nell'interconnessione di computer portatili, telefonini e assistenti personali digitali. Nei paragrafi seguenti esamineremo i sistemi di rintracciamento satellitari, telefoni senza fili, telefoni cellulari e tecnologie simili. Questi sistemi si stanno adesso fondendo, producendo dei computer portatili capaci di mandare e ricevere chiamate telefoniche, fax e posta elettronica, così come di cercare informazioni su database remoti in qualsiasi luogo sulla terra.

Tali dispositivi stanno già creando un mercato enorme. Molte società nei computer, telefoni, satelliti e altre industrie vogliono un pezzo dell'azione. Il risultato è un mercato caotico, con numerosi prodotti e servizi incompatibili o che si sovrappongono, tutti in rapido cambiamento e tipicamente differenti in ogni paese. Non di meno, le descrizioni date più avanti dovrebbero fornire almeno una conoscenza fondamentale delle tecnologie elencate. Per maggiori informazioni, si vedano Bates (1994); Goodman (1991); Macario (1993); Padgett et al. (1995); Seybold (1994).

2.7.1 Sistemi di rintracciamento (paging)

I primi sistemi di rintracciamento usavano altoparlanti all'interno di un singolo edificio. In un ospedale è facile sentire annunci come questo: "il dottor Johnson chiami per favore l'interno 4321". Al giorno d'oggi, la gente che vuole essere rintracciata indossa dei piccoli allarmi, di solito con piccoli schermi per mostrare dei brevi messaggi in arrivo.

Una persona che vuole rintracciarne un'altra che indossa un cercapersone (beeper) può chiamare la società del beeper e inserire un codice di sicurezza, il numero del beeper è il numero che la persona che lo indossa deve chiamare (o un altro messaggio breve). Il computer che riceve la richiesta poi lo trasmette sulle linee terrestri a una antenna in cima a una collina, che lo diffonde direttamente (per il rintracciamento locale) oppure lo manda a un satellite in cielo (per rintracciamento a lunga distanza) che poi lo ritrasmette. Quando il beeper rileva il suo numero unico nel flusso radio in ingresso, suona e mostra il numero da chiamare. È anche possibile rintracciare simultaneamente un intero gruppo di persone con una singola chiamata telefonica.

I sistemi più avanzati di beeper si collegano direttamente a un computer e possono ricevere non solo un numero telefonico, ma un messaggio più lungo. Il computer può in seguito elaborare i dati quando arrivano. Per esempio, una società potrebbe mantenere aggiornata la lista dei prezzi nei computer portatili dei suoi rappresentanti usando questa forma di chiamata.

La maggior parte degli attuali sistemi di chiamata è a senso unico, da un singolo computer a un grande numero di riceventi. Non si pone il problema di chi parlerà dopo e non c'è conflitto fra i molti utenti in concorrenza per un piccolo numero di canali, poiché c'è solo un mittente nell'intero sistema.

I sistemi di paging richiedono piccola larghezza di banda poiché ogni messaggio richiede solo una scarica di forse 30 byte. A questo tasso di dati, un canale satellitare di 1 Mbps può gestire più di 240.000 messaggi al minuto. I vecchi sistemi di paging vanno a varie frequenze nella banda 150-174 MHz. La maggior parte di quelli moderni va nella banda 930-932 MHz. La figura 2-53 (a) mostra la natura a senso unico di un sistema di paging, con tutte le comunicazioni che vengono trasmesse su una singola frequenza. In seguito confronteremo questo sistema con i telefoni mobili, che sono a doppio senso e usano due frequenze per chiamata, con coppie di frequenze differenti usate per chiamate differenti, come in figura 2-53 (b). Queste differenze rendono il sistema di paging molto più semplice e meno costoso da far funzionare.

2.7.2 Telefoni senza filo

I telefoni senza filo (cordless) vennero introdotti per permettere di camminare per casa

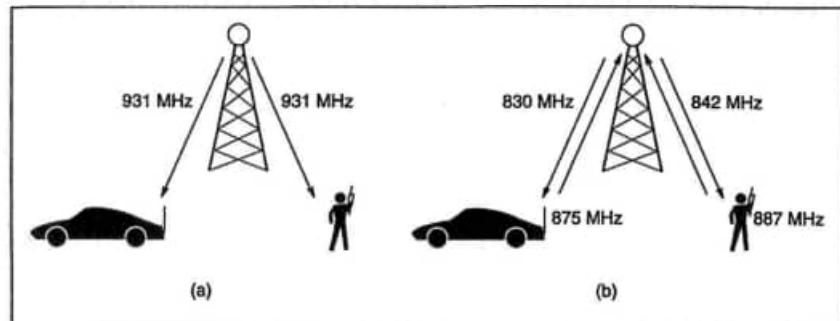


Fig. 2-53 (a) I sistemi di paging sono a senso unico. (b) I telefoni mobili sono a doppio senso.

mentre si parla al telefono. Un telefono senza filo consiste di due parti, che sono sempre vendute insieme: una stazione base e un telefono. La stazione base ha un connettore telefonico standard nella parte posteriore, così può essere connessa (da un filo) alla rete telefonica. I telefoni comunicano con la stazione base con una radio a bassa potenza. La portata è tipicamente da 100 a 300 m.

Poiché ci si aspettava che i primi telefoni senza filo comunicassero solo con le proprie stazioni base, non c'era bisogno di standardizzazione. Alcuni dei modelli più economici usavano una frequenza fissata, selezionata alla fabbrica. Se, per caso, accadeva che il tuo telefono cordless usasse la stessa frequenza di quello del tuo vicino, ciascuno poteva ascoltare la chiamata dell'altro. Modelli più costosi evitavano questo problema permettendo all'utente di selezionare la frequenza di trasmissione.

La prima generazione di telefoni di questo tipo, nota come CT-1 negli Stati Uniti e CEPT-1 in Europa, era interamente analogica. Potevano, e spesso capitava, causare interferenze con le radio e le televisioni. La scarsa ricezione e la mancanza di sicurezza portarono l'industria a sviluppare uno standard digitale, CT-2, che nacque in Inghilterra. I primi dispositivi CT-2 potevano fare chiamate ma non riceverle, ma non appena fu venduto il primo, il fabbricante ricevette una risposta negativa dal mercato e il sistema fu rapidamente riprogettato. Come la versione CT-1, ogni telefono doveva stare entro pochi metri dalla propria stazione base, rendendolo utile in casa o in ufficio, ma inutile in auto o camminando in città.

Nel 1992, venne introdotta una terza generazione, la CT-3 o DECT, che supporta la gestione dinamica sulle stazioni base (roaming). Questa tecnologia è simile ai telefoni cellulari, che adesso descriveremo.

2.7.3 Telefoni cellulari analogici

I radiotelefoni mobili venivano usati sporadicamente per comunicazioni marittime e militari durante i primi decenni del XX secolo. Nel 1946, fu messo a punto a St. Louise il primo sistema per telefoni sulle auto. Questo sistema usava un singolo grande trasmettitore sulla parte superiore di un alto edificio e aveva un singolo canale, usato sia per spedire

che ricevere. Per parlare, l'utente doveva premere un pulsante che abilitava il trasmettente e disabilitava il ricevente. Tali sistemi, noti come **sistemi premi-per-parlare** (push to talk), furono installati in diverse città a cominciare dalla fine degli anni cinquanta. Le radio CB, i taxi e le macchine della polizia dei programmi televisivi spesso usano questa tecnologia.

Negli anni sessanta venne installato **IMTS (Improved Mobile Telephone System)**. Anch'esso usava un trasmettitore ad alta potenza (200 W), sulla cima di una collina, ma ora aveva due frequenze, una per trasmettere e una per ricevere, così non era più necessario il pulsante premi-per-parlare. Poiché tutte le comunicazioni dai telefoni mobili venivano dirette su un canale diverso da quello che ascoltavano i telefoni, gli utenti mobili non potevano ascoltarsi l'un l'altro (non come il sistema premi-per-parlare usato nei taxi). IMTS supportava 23 canali da 150 MHz a 450 MHz. A causa del piccolo numero di canali, gli utenti dovevano spesso attendere a lungo per ottenere un segnale di linea libera. Inoltre, a causa della grande potenza del trasmettitore sulla vetta della collina, i sistemi adiacenti dovevano essere distanti centinaia di chilometri per evitare interferenze. In breve, il sistema non era pratico a causa della limitata capacità.

Sistema telefonico mobile avanzato

Tutto cambiò con l'**AMPS (Advanced Mobile Phone System)**, inventato dalla Bell Labs e installato la prima volta negli Stati Uniti nel 1982. In Inghilterra viene chiamato TACS, e in Giappone MCS-L1. L'AMPS ha una regione geografica divisa in **celle**, tipicamente da 10 a 20 km in larghezza, e usa in ognuna un certo insieme di frequenze. L'idea chiave che dà all'AMPS una capacità molto maggiore di tutti i sistemi precedenti è l'uso di celle relativamente piccole e il riuso di frequenze di trasmissione nelle celle vicine (ma non adiacenti). Mentre un sistema IMTS di 100 km in larghezza può avere una chiamata su ogni frequenza, un sistema AMPS potrebbe avere 100 celle di 10 km nella stessa area ed essere capace di avere da 5 a 10 chiamate sulla stessa frequenza, in celle ampiamente separate. Inoltre, celle più piccole significano meno potenza, il che porta a dispositivi più piccoli e più economici. I telefoni tenuti in mano emettono 0,6 W; i trasmettitori nelle macchine sono tipicamente di 3 W, il massimo permesso dalla FCC.

L'idea del riuso della frequenza è illustrata in figura 2-54 (a). Le celle sono normalmente quasi circolari, ma sono più facili da modellare come esagoni. In figura 2-54 (a), le celle sono della stessa dimensione. Esse sono raggruppate insieme in unità di sette celle. Ogni lettera indica un gruppo di frequenze. Per ogni insieme di frequenze, c'è un buffer largo circa due celle dove quella frequenza non è riusata, per avere una buona separazione e bassa interferenza.

Trovare postazioni a elevate altitudini per piazzare antenne per la stazione base è una questione fondamentale. Questo problema ha portato alcune società di telecomunicazioni a fare alleanze con la Chiesa Cattolica Romana, poiché quest'ultima possiede un gran numero di postazioni per antenne di elevato potenziale, tutte convenientemente sotto una singola gestione.

In un'area in cui il numero di utenti è cresciuto fino al punto di sovraccaricare il sistema, la potenza viene ridotta e le celle sovraccaricate sono divise in celle più piccole per

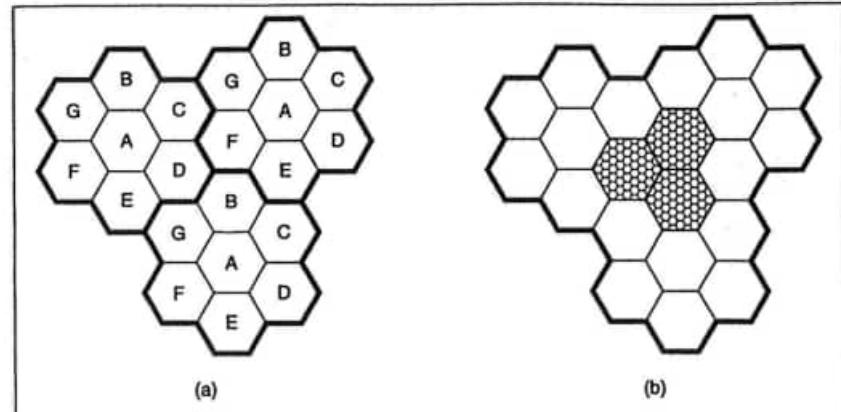


Fig. 2-54 Le frequenze non sono riusate nelle celle adiacenti. (b) Per aggiungere più utenti, possono essere usate celle più piccole.

permettere un maggior riuso di frequenza, come mostrato in figura 2-54 (b). Quanto dovrebbero essere grandi le celle è un problema complesso ed è trattato in Hac (1995). Al centro di ciascuna cella c'è una stazione base alla quale trasmettono tutti i telefoni nella cella. La stazione base è un computer e un trasmettitore/ricevitore connesso a una antenna. In un sistema piccolo, tutte le stazioni base sono connesse a un singolo dispositivo chiamato **MTSO (Mobile Telephon Switching Office)** o **MSC (Mobile Switching Center)**. In uno più grande sono necessari diversi MTSO tutti connessi a un MTSO di secondo livello e così via. Gli MTSO sono essenzialmente uffici terminali come nel sistema telefonico e sono, infatti, connessi ad almeno un ufficio terminale del sistema telefonico. Gli MTSO comunicano con la stazione base, uno con l'altro e con la PSTN usando una rete a commutazione di pacchetto.

In ogni momento, ogni telefono mobile si trova logicamente in una cella specifica e sotto il controllo della stazione base di quella cella. Quando un telefono mobile lascia una cella, la sua stazione base nota il segnale del telefono che si affievolisce e chiede a tutte le stazioni base intorno quanta potenza stanno ricevendo da esso. La stazione base allora trasferisce il controllo alla cella che riceve il segnale più forte, cioè quella in cui il telefono è ora posizionato. Il telefono è quindi informato, e, se una chiamata è in corso, sarà richiesto di commutare a un nuovo canale (poiché il vecchio non è riusato in nessuna delle celle adiacenti). Questo processo è chiamato **handoff** e impiega circa 300 ms. L'assegnazione del canale viene fatta dall'MTSO, che è il centro nervoso del sistema. Le stazioni base sono in realtà solo ponti radio.

Canali

Il sistema AMPS usa 832 canali full-duplex, ognuno consistente di una coppia di canali simplex. Ci sono 832 canali simplex di trasmissione da 824 a 849 MHz, e 832 canali

simplex di ricezione da 869 a 894 MHz. Ognuno di questi canali simplex è largo 30 kHz. Quindi AMPS usa FDM per separare i canali.

Nella banda a 800 MHz, le onde radio sono lunghe circa 40 cm e viaggiano per linee rette. Esse vengono assorbite dagli alberi e dalle piante e rimbalzano sul terreno e sugli edifici. È possibile che un segnale inviato da un telefono mobile raggiunga la stazione base attraverso un percorso diretto, ma anche leggermente più tardi dopo aver rimbalzato sul terreno o su un edificio. Questo può causare un effetto eco o distorsione del segnale. Qualche volta è anche possibile ascoltare una conversazione distante rimbalzata diverse volte.

Negli Stati Uniti, gli 832 canali in ciascuna città sono gestiti dalla FCC. Di questi, metà sono assegnati alla società telefonica locale, la società **wireline** o società **B-side**. L'altra metà viene assegnata a un nuovo entrato nel mercato dei cellulari, la società **A-side**. L'idea è che ci siano in concorrenza almeno due società di cellulari, per aumentare l'efficienza e diminuire i prezzi.

Comunque, la distinzione fra una società telefonica e una di telefonia cellulare è adesso confusa, dal momento che la maggior parte delle società telefoniche ha un partner cellulare. Nel 1994 la AT&T si fuse con la McCaw Cellular, il più grande operatore di cellulari. Accade di frequente che una società sia A-side in alcuni mercati e B-side in altri. Altra confusione avviene poiché una società può affittare o vendere tutte o in parte le sue 416 licenze di canale.

Gli 832 canali sono divisi in quattro categorie:

1. Controllo (base verso mobile) per gestire il sistema.
2. Paging (base verso mobile) per avvertire un utente mobile.
3. Accesso (bidirezionale) per predisposizione di chiamate e assegnazione del canale.
4. Dati (bidirezionali) per voce, fax e dati.

Ventuno canali sono riservati per controllo e questi sono installati in una PROM in ogni telefono. Poiché le stesse frequenze non possono essere riusate nelle celle vicine, il numero reale di canali vocali disponibili per cella è molto più piccolo di 832, tipicamente circa 45.

Gestione della chiamata

Ogni telefono mobile AMPS ha un numero seriale a 32 bit e un numero telefonico di 10 cifre nella sua PROM. Il numero telefonico è rappresentato come un codice di area a 3 cifre, su 10 bit, e un numero di abbonato di 7 cifre, su 24 bit. Quando si accende, un telefono scandisce una lista preprogrammata di 21 canali di controllo per trovare il segnale più potente. Il telefono mobile è predisposto per scandire solo A-side, o solo B-side, A-side preferiti o B-side preferiti, a seconda dei servizi ai quali l'utente si è abbonato. Dal canale di controllo esso impara il numero di paging e i canali di accesso.

Il telefono quindi trasmette il suo numero seriale a 32 bit e il numero telefonico a 34 bit. Come tutte le informazioni di controllo nell'AMPS, questo pacchetto viene spedito in forma digitale, più volte e con un codice a correzione d'errore, anche se i canali vocali stessi sono analogici.

Quando la stazione base sente l'annuncio, lo comunica all'MTSO, che registra l'esistenza del suo nuovo utente e informa anche l'MTSO di base dell'utente della sua nuova locazione. Durante le normali operazioni, il telefono mobile si reregistra circa una volta ogni 15 min.

Per fare una chiamata, un utente mobile accende il telefono, digita sulla tastiera il numero da chiamare e preme il pulsante di SEND. Il telefono poi trasmette il numero da chiamare e la sua identità sul canale di accesso. Se avviene una collisione, ritenta più tardi. Quando la stazione base riceve la richiesta, informa l'MTSO. Se il chiamante è un utente della società dell'MTSO (o uno dei suoi partner), l'MTSO cerca un canale non in uso per la chiamata. Il telefono mobile poi automaticamente commuta al canale vocale selezionato e aspetta fino a quando il chiamato solleva il telefono.

Per le chiamate in arrivo il sistema funziona diversamente. Per iniziare, tutti i telefoni non in uso ascoltano in continuazione il canale di chiamata per rilevare messaggi a loro diretti. Quando arriva una chiamata per un telefono mobile (sia da un telefono fisso o da un altro telefono mobile), viene trasmesso un pacchetto all'MTSO di casa del chiamato per trovare dov'è. Viene poi mandato un pacchetto alla stazione base nella sua cella corrente, che spedisce un messaggio sul canale di chiamata: "Unità 14, sei lì?" Il telefono chiamato allora risponde con un "Sì" sul canale di controllo. Poi la base dice qualcosa come: "Unità 14, chiamata per te sul canale 3". A questo punto, il telefono del chiamato si commuta sul canale 3 e inizia a squillare.

Questioni di sicurezza

I telefoni analogici cellulari non garantiscono segretezza. Chiunque possieda un ricevitore multibanda (scanner) può sintonizzarsi e ascoltare tutto quello che succede in una cella. La principessa Diana e il suo amante furono una volta sorpresi in questo modo, il che produsse gran titoli di giornale in tutto il mondo. Poiché la maggior parte degli utenti dei cellulari non capisce quanto sia insicuro il sistema, spesso gli affida i numeri di carta di credito e altre informazioni confidenziali.

Un altro problema principale è il furto di tempo di trasmissione. Con un ricevitore su tutte le bande collegato a un computer, un ladro può monitorare il canale di controllo e registrare il numero seriale di 32 bit e i numeri telefonici a 34 bit di tutti i telefoni mobili che ascolta. Andandosene in giro per un paio di ore, egli può costruire un grosso archivio di numeri. Il ladro può allora prendere un numero e usarlo per le sue chiamate. Questo trucco funzionerà fino a quando la vittima riceverà la bolletta, settimane dopo, nel momento in cui il ladro userà semplicemente un altro numero.

Alcuni ladri offrono un servizio telefonico a basso costo facendo chiamate per i loro utenti usando numeri rubati. Altri riprogrammano (clonano) i telefoni mobili con numeri rubati e li vendono come telefoni che possono fare chiamate senza pagare.

Alcuni di questi problemi possono essere risolti dalla crittografia, ma allora la polizia non potrebbe intercettare facilmente i criminali che comunicano col cellulare. Questa materia è molto controversa ed è discussa nel capitolo 7 in maggior dettaglio.

Un'altra questione nell'area della sicurezza è il vandalismo e il danneggiamento alle antenne e alle stazioni base. Tutti questi problemi sono molto difficili e comportano la perdita di centinaia di milioni di dollari all'anno per l'industria del cellulare.

2.7.4 Telefoni cellulari digitali

La prima generazione di sistemi cellulari era analogica. La seconda generazione è digitale. Negli Stati Uniti c'era fondamentalmente un solo sistema: AMPS. Quando fu il momento del digitale, emersero tre o quattro concorrenti e cominciò una lotta per la sopravvivenza. Sembra adesso che sopravviveranno due sistemi. Il primo è compatibile all'indietro con lo schema di allocazione di frequenza AMPS ed è specificato negli standard IS-54 e IS-135. L'altro si basa sulla sequenza diretta a spettro diffuso ed è specificato nello standard IS-95. IS-54 è a doppio modo (analogico e digitale) e usa gli stessi canali di 30 kHz dell'AMPS. Esso impacchetta 48,6 kbps in ciascun canale e lo condivide con tre utenti simultanei. Ogni utente riceve 13 kbps; il resto è controllo e sovraccarico di temporizzazione. Celle, stazioni base e MTSO funzionano lo stesso come nell'AMPS. Solo la segnalazione digitale e la codifica digitale della voce è differente. Il sistema IS-95 è piuttosto nuovo. Lo discuteremo quando arriveremo all'allocazione del canale nel capitolo 4.

In Europa osserviamo il processo inverso. Erano in uso cinque diversi sistemi analogici, in paesi diversi, così qualcuno con un telefono britannico non poteva usarlo in Francia e così via. Questa esperienza portò la PTT europea ad accordarsi su un sistema digitale comune, chiamato **GSM (Global Systems for Mobile communications)**, che fu dispiegato prima di qualsiasi altro sistema americano concorrente. Il sistema giapponese è diverso da tutti quelli sopra.

Poiché i sistemi europei erano tutti diversi, fu più semplice renderli operanti in puro digitale su una nuova banda di frequenza (1,8 GHz), in aggiunta a riadattare la banda a 900 MHz dove possibile. GSM usa sia FDM che TDM. Lo spettro disponibile è diviso in 50 bande da 200 kHz. All'interno di ciascuna banda TDM è usata per multiplex più utenti.

Alcuni telefoni GSM usano le carte intelligenti (smart card), cioè dispositivi della dimensione di una carta di credito che contengono una CPU. Il numero seriale e il numero del telefono sono contenuti sulla carta, non nel telefono, migliorando il livello di sicurezza (se rubi il telefono senza la carta non avrai il numero). Si usa anche la crittografia. Discuteremo il GSM nel capitolo 4.

2.7.5 Servizi di comunicazione personale

Il Sacro Graal del mondo dei telefoni è un piccolo telefono senza fili che si può usare in casa e portare in qualsiasi parte del mondo. Dovrebbe rispondere allo stesso numero telefonico, non importa dove si trovi, cosicché chiunque avrebbe un solo numero telefonico (con l'AMPS, il tuo telefono di casa e il tuo telefono mobile hanno numeri diversi). Questo sistema è attualmente in forte sviluppo (Lipper, Rumsewicz, 1994). Negli Stati Uniti viene chiamato **PCS (Personal Communication Services)**. Dovunque viene chiamato **PCN (Personal Communication Network)**. Nel mondo della telefonia, gli Stati Uniti hanno una tradizione a marciare a un ritmo diverso da ogni altro. Fortunatamente, la maggior parte dei dettagli tecnici sono gli stessi.

PCS userà tecnologia cellulare, ma con microcelle, larghe forse da 50 a 100 m. Questo consentirà una potenza molto bassa (1/4 W), che renderà possibile la costruzione di telefoni molto piccoli e leggeri. D'altra parte esso richiederà molte più microcelle delle celle da 20 km dell'AMPS. Supponendo che una microcella sia 1/200 del diametro di una cella AMPS, sarebbero necessarie 40.000 celle per coprire la stessa area. Anche se queste

2.8 Satelliti per comunicazioni

celle sono molto più economiche delle celle AMPS, è chiaro che costruire un sistema PCS completo dal nulla richiederà un investimento d'infrastruttura molto più massiccio di quello dell'AMPS.

Alcune società telefoniche hanno capito che i loro pali del telefono sono basi eccellenze per posizionare le stazioni che hanno la dimensione di un tostapane, dal momento che i pali e i fili già esistono, riducendo così enormemente i costi di installazione. Queste piccole stazioni base sono talvolta chiamate **telepoint**. Quante installarne e dove metterle è una questione complicata (Steele et al. 1995a, 1995b).

Il governo degli USA (specificamente la FCC) sta usando la PCS per far soldi. Nel 1994-95 esso mise all'asta licenze per usare lo spettro PCS (da 1,7 a 2,3 GHz). L'asta portò 7.700.000.000 di dollari per il governo. Questa asta sostituì il precedente sistema di assegnare le bande di frequenza con una lotteria, eliminando così la pratica delle società con nessun interesse nelle telecomunicazioni che entravano nella lotteria. Quando una di queste società vinceva una frequenza poteva istantaneamente venderla a uno dei perdenti per milioni di dollari.

Sfortunatamente, non esistono pranzi gratis, nemmeno per il governo. La banda da 1,7 a 2,3 GHz è già completamente allocata ad altri utenti. A questi utenti sarà assegnato uno spettro da qualche altra parte e sarà detto di spostarsi lì. Il guaio è che la dimensione dell'antenna dipende dalla frequenza, così questa forzata riallocazione della frequenza, richiederà miliardi di dollari di investimento per sostituire antenne, trasmettitori ecc. Orde di lobbisti si aggirano per Washington, con suggerimenti su chi dovrebbe pagare per tutto questo. Il risultato netto è che il PCS non può essere ampiamente dispiegato in questo millennio. Per un modo più razionale di trattare lo spettro, si vedano Youssef et al. (1995).

2.8 Satelliti per comunicazioni

Negli anni cinquanta e all'inizio degli anni sessanta, si cercava di mettere a punto un sistema di comunicazione facendo rimbalzare i segnali su palloni meteorologici metallizzati. Sfortunatamente, i segnali ricevuti erano troppo deboli per essere di uso pratico. In seguito, la Marina degli Stati Uniti notò nel cielo una specie di pallone meteorologico permanente – la Luna – e costruì un sistema per le comunicazioni da rispedire a terra facendovi rimbalzare i segnali.

L'ulteriore progresso nel campo della comunicazione celeste dovette attendere fino al lancio del primo satellite per comunicazioni nel 1962. La differenza principale fra un satellite artificiale e uno naturale è che in quello artificiale si possono amplificare i segnali prima di rimandarli indietro, convertendo una curiosità scientifica in un potente sistema di comunicazione.

I satelliti per comunicazioni hanno alcune interessanti proprietà che li rendono attraenti per molte applicazioni. Un satellite per comunicazioni può essere pensato come un grande ripetitore di microonde nel cielo. Esso contiene diversi **trasponditori**, ognuno dei quali ascolta una certa porzione di spettro, amplifica il segnale in entrata e lo ritrasmette su un'altra frequenza per evitare interferenza col segnale che arriva. Il raggio diretto verso il basso può essere diffuso, coprendo una frazione sostanziale della superficie terrestre, o stretto, coprendo un area di solo pochi chilometri di diametro.

2.8.1 Satelliti geosincroni

Secondo la legge di Keplero, il periodo orbitale di un satellite varia secondo il raggio dell'orbita alla potenza 3/2. Vicino la superficie della Terra, il periodo è di circa 90 min. Queste basse altitudini sono problematiche perché un satellite per comunicazione resta in vista di una stazione terrestre solo per un breve intervallo di tempo.

Tuttavia, a un'altitudine di circa 36.000 km sull'equatore, il periodo del satellite è di 24 h (per i puristi, la velocità di rotazione è il giorno siderale: 23 h, 56 min, 4,09 s), quindi esso ruota alla stessa velocità della Terra. Un osservatore che guarda il satellite in una orbita circolare equatoriale lo vede in un posto fisso del cielo, apparentemente fermo. Vedere il satellite fermo in cielo è estremamente desiderabile, poiché altrimenti sarebbe necessaria una costosa antenna direzionale per seguirlo.

Con la tecnologia attuale, non è saggio avere satelliti collocati più vicino di 2° nel piano equatoriale a 360°, per evitare interferenze. Con una separazione di 2°, ci possono essere solo $360/2 = 180$ satelliti geosincroni per comunicazioni. Alcune di queste parti di orbita sono riservate per altre classi di utenti (ad es. trasmissioni televisive, uso governativo e militare ecc.).

Fortunatamente, i satelliti che usano diverse parti dello spettro non sono in concorrenza, così ciascuno dei 180 possibili satelliti potrebbe avere dei flussi di dati che vanno su e giù contemporaneamente. In alternativa, due o più satelliti potrebbero occupare la stessa parte di orbita se operassero a frequenze diverse.

Banda	Frequenze	Downlink (GHz)	Uplink (GHz)	Problemi
C	4/6	3,7–4,2	5,925–6,425	Interferenza terrestre
Ku	11/14	11,7–12,2	14,0–14,5	Pioggia
Ka	20/30	17,7–21,7	27,5–30,5	Pioggia; costi apparecchiatura

Fig. 2-55 Le principali bande satellitari.

Per prevenire il caos totale nel cielo, esistono accordi internazionali su chi può usare quali orbite e frequenze. Le bande commerciali principali sono elencate in figura 2-55. La banda C fu la prima a essere designata per il traffico di satelliti commerciali. In essa sono assegnati due intervalli di frequenze, il più basso per il traffico diretto in basso (dal satellite) e quello più alto per il traffico diretto in alto (verso il satellite). Per un canale di connessione full-duplex si richiedono entrambe le direzioni. Queste bande sono già sovrapposte perché sono usate anche dalle società per i collegamenti terrestri a microonde. La banda successiva più alta disponibile per le società di telecomunicazioni commerciali è la banda Ku. Questa banda non è molto congestionata e a queste frequenze i satelliti possono avvicinarsi fino a 1°. Tuttavia, esiste un altro problema: la pioggia. L'acqua è un ottimo assorbente per queste brevi microonde. Fortunatamente, i forti temporali sono di solito locali, così usando diverse stazioni terrestri ampiamente distanziate invece di una sola, il problema può essere superato al prezzo di antenne extra, cavo extra ed elettronica extra per commutare rapidamente le stazioni.

Su Ka è stata anche allocata banda per traffico commerciale satellitare, ma l'apparecchiatura necessaria per usarla è ancora costosa. In aggiunta a queste bande commerciali, esistono anche molte bande governative e militari.

Un satellite tipico ha 12-20 transponditori, ognuno con una larghezza di banda di 36-50 MHz. Un transponditore a 50 Mbps può essere usato per codificare un singolo flusso di dati a 50 Mbps, 800 canali digitali vocali a 64 kbps, o varie altre combinazioni. Inoltre, due transponditori diversi possono usare differenti polarizzazioni del segnale, così possono usare lo stesso intervallo di frequenza senza interferire. Nei primi satelliti la divisione dei canali nei transponditori era statica, dividendo la larghezza di banda in bande di frequenza fissate (FDM). Al giorno d'oggi, si usa anche il multiplexing a divisione di tempo a causa della sua maggiore flessibilità.

I primi satelliti con un singolo raggio spaziale illuminavano l'intera Terra. Con l'enorme discesa dei prezzi, dimensioni e requisiti di potenza della microelettronica, è diventata possibile una strategia di trasmissione molto più sofisticata. Ogni satellite è equipaggiato con più antenne e più transponditori. Ogni raggio diretto verso il basso può essere focalizzato su una piccola area geografica, così possono aver luogo simultaneamente più trasmissioni verso l'alto e verso il basso. Questi così detti **raggi puntiformi (spot beam)** sono di solito di forma ellittica e possono avere un diametro piccolo alcune centinaia di chilometri. Un satellite per comunicazioni per gli Stati Uniti dovrebbe tipicamente avere un largo raggio per i 48 stati, più fasci puntiformi per l'Alaska e le Hawaii.

Un nuovo sviluppo per il mondo della comunicazione satellitare è lo sviluppo di microstazioni a basso costo chiamate **VSAT** (*Very Small Aperture Terminals*) (Ivancic et al. 1994). Questi piccoli terminali hanno antenne di 1 m e possono emettere circa 1 watt di potenza. Il collegamento verso l'alto è in genere buono per 19,2 kbps, ma il collegamento verso il basso è maggiore, spesso 512 kbps. In molti sistemi VSAT, le microstazioni non hanno abbastanza potenza per comunicare direttamente con un'altra (via satellite, naturalmente). Invece, una speciale stazione terrestre, l'**hub**, con una grande antenna a grande guadagno è necessaria per trasmettere il traffico fra i VSAT, come in figura 2-56. In questo sistema di funzionamento, il ricevitore o il mittente hanno una grande antenna e un potente amplificatore. L'alternativa è fra un ritardo più lungo nel ritorno e stazioni terminali utenti più economiche.

I satelliti per comunicazioni hanno diverse proprietà che sono radicalmente differenti dai collegamenti terrestri punto a punto. Per iniziare, anche se i segnali da e per un satellite viaggiano alla velocità della luce (circa 300.000 km/s), la grande distanza di andata e ritorno introduce un ritardo sostanziale. A seconda della distanza fra l'utente e le stazioni terrestri e l'altezza del satellite sull'orizzonte, il tempo di transito da capo a capo è tra 250 e 300 ms. Un valore tipico è di 270 ms (540 ms per un sistema VSAT con uno hub). Per fare un confronto, i collegamenti a microonde terrestri hanno un ritardo di propagazione di quasi 3 μ s/km e il cavo coassiale o la fibra ottica hanno un ritardo di approssimativamente 5 μ s/km (i segnali elettromagnetici viaggiano più velocemente nell'aria che nei materiali solidi).

Un'altra importante proprietà dei satelliti è che essi sono intrinsecamente dei mezzi a diffusione. Non costa di più spedire un messaggio a migliaia di stazioni nell'ombra di un

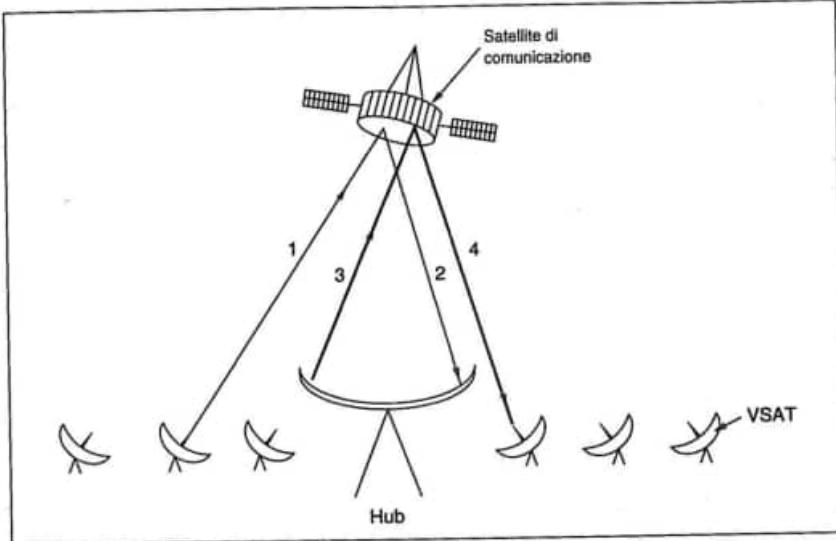


Fig. 2-56 VSAT che usa uno hub.

trasponditore di quanto costi spedirlo a una sola. Per alcune applicazioni questa proprietà è molto utile. Anche quando la trasmissione può essere simulata usando una linea punto-a-punto, la diffusione satellitare può essere molto più economica. D'altra parte, dal punto di vista della sicurezza e del segreto, i satelliti sono un completo disastro: chiunque può ascoltare tutto. La crittografia è essenziale quando si richiede la segretezza.

I satelliti hanno anche la proprietà che il costo di trasmettere un messaggio è indipendente dalla distanza attraversata. Una chiamata attraverso l'oceano non costa più di una chiamata che attraversa la strada. I satelliti hanno anche un basso tasso di errore e possono essere configurati quasi istantaneamente, una considerazione importante per la comunicazione militare.

2.8.2 Satelliti a orbita bassa

Per i primi trent'anni dell'era spaziale, i satelliti a orbita bassa erano raramente usati per la comunicazione, perché entravano e uscivano dalla visuale troppo velocemente. Nel 1990, la Motorola iniziò una nuova attività presentando alla FCC una richiesta per il lancio di 77 satelliti a orbita bassa per il progetto Iridium (l'elemento 77 è l'iridio). Il piano fu più tardi revisionato per usare solo 66 satelliti, così il progetto avrebbe dovuto essere rinominato Dysprosium (l'elemento 66), ma probabilmente questa parola suonava come una malattia. L'idea era che non appena un satellite fosse uscito dalla visuale, un altro lo avrebbe sostituito. Questa proposta scatenò una gran frenesia fra le altre società di comunicazioni. All'improvviso, ognuna voleva lanciare una catena di satelliti a orbita bassa. Qui descriveremo brevemente il sistema Iridium, ma gli altri sono simili.

L'obiettivo base di Iridium sarebbe quello di fornire un servizio mondiale di telecomunicazioni mediante dispositivi portatili che comunicano direttamente con i satelliti. Iridium fornirebbe ovunque sulla Terra servizi per voce, dati, paging, fax e navigazione. Questo servizio sarebbe in concorrenza diretta con il PCS/PCN e lo renderebbe inutile. Iridium sfrutterebbe l'idea dalla radio cellulare, ma con un importante modifica. Normalmente, le celle sono fisse ma gli utenti sono mobili. Qui, ogni satellite avrebbe un numero sostanziale di fasci puntiformi che scandirebbero la Terra al muoversi del satellite. Così sia le celle che gli utenti sarebbero mobili in questo sistema, ma le tecniche di handoff usate per la radio cellulare sono parimenti applicabili al caso della cella che lascia l'utente così come al caso dell'utente che lascia la cella.

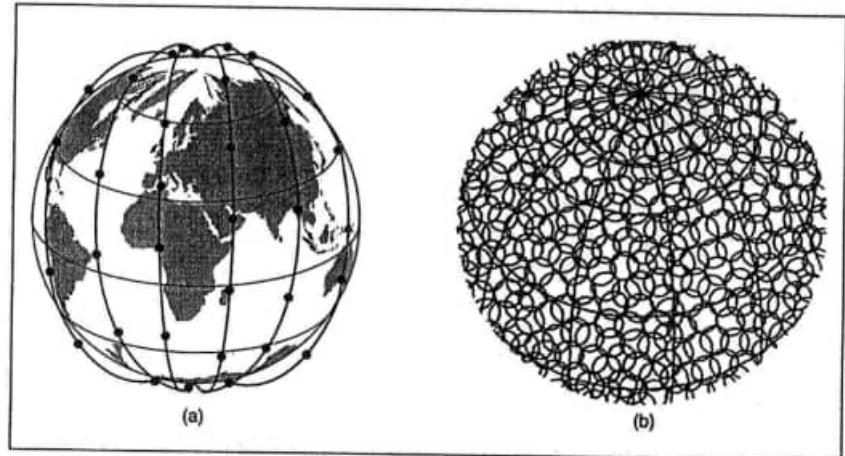


Fig. 2-57 I satelliti Iridium formano sei collane attorno alla Terra. (B) La Terra coperta da 1628 celle mobili.

I satelliti dovrebbero essere posizionati a una altitudine di 750 km, in orbita sui circoli polari. Essi sarebbero disposti in collane da nord a sud, con un satellite ogni 32° di latitudine. L'intero pianeta sarebbe coperto con sei collane di satelliti, come in figura 2-57 (a). La gente che non sa molto di chimica può immaginare questa disposizione come un atomo molto grande di disperso, in cui la Terra è il nucleo e i satelliti sono elettroni. Ogni satellite dovrebbe avere un massimo di 48 raggi puntiformi, per un totale di 1628 celle sulla superficie della Terra, come in figura 2-57 (b). Le frequenze potrebbero essere riusate ogni due celle, come nella radio cellulare convenzionale. Ogni cella avrebbe 174 canali full-duplex, per un totale di 283.272 canali in tutto il mondo. Alcuni di questi sarebbero usati per il paging e la navigazione, che richiedono appena una minima larghezza di banda (i dispositivi di paging mostrano due linee di testo alfanumerico). I collegamenti diretti verso l'alto e verso il basso dovrebbero operare nella banda L a 1,6 GHz, rendendo possibile comunicare con un satellite usando un dispositivo dotato di una piccola batteria. I messaggi ricevuti da un satellite ma destinati a uno remoto sarebbero

ritrasmessi fra i satelliti nella banda Ka. Larghezza di banda sufficiente è disponibile nella spazio esterno per collegamenti intersatellitari. I fattori limitanti sarebbero i segmenti diretti verso l'alto/basso. La Motorola stima che 200 MHz sarebbero sufficienti per l'intero sistema.

Il costo previsto per l'utente finale è di circa 3 dollari al minuto. Se questa tecnologia a quel prezzo potrà fornire servizio universale in qualsiasi posto sulla Terra, è improbabile che il progetto muoia per mancanza di clienti. Gli agenti d'affari e gli altri viaggiatori che vogliono essere sempre raggiungibili, anche in aree non sviluppate, lo sottoscriverebbero in massa. Comunque, nelle aree sviluppate, Iridium incontrerebbe la dura concorrenza del PCS/PCN.

2.8.3 Confronto tra satelliti e fibre

È istruttivo un confronto fra le comunicazioni satellitari e quelle terrestri. Venti anni fa, si poteva prevedere che il futuro delle comunicazioni sarebbe stato nei satelliti per comunicazioni. Dopo tutto, il sistema telefonico era cambiato poco nei cento anni precedenti e non mostrava alcun segno di cambiamento per i cento anni successivi. Questa assenza di prospettive evolutive era causata principalmente dall'ambiente protetto politicamente per cui ci si aspettava che le società telefoniche fornissero un buon servizio vocale a prezzi ragionevoli (cosa che facevano) e ottenessero in ritorno un profitto garantito sui loro investimenti. Per la gente che voleva trasmettere dati erano disponibili modem a 1200 bps. Questo era quasi tutto quello che c'era.

L'arrivo della concorrenza negli Stati Uniti nel 1984 e più tardi in Europa cambiò tutto radicalmente. Le società telefoniche cominciarono a rimpiazzare le loro reti a lungo percorso con le fibre e introdussero servizi a grande larghezza di banda come SMDS e B-ISDN. Esse interuppero anche la loro antica consuetudine di far pagare artificialmente alti prezzi per gli utenti a lunga distanza per sovvenzionare il servizio locale.

All'improvviso, le connessioni terrestri su fibra sembrano il vincitore a lungo termine. Non di meno, i satelliti per comunicazioni conservano alcune nicchie di mercato che la fibra non può indirizzare. Daremo adesso uno sguardo ad alcune di queste.

Mentre una singola fibra ha, in teoria, maggior larghezza di banda di tutti i satelliti mai lanciati, questa larghezza di banda non è disponibile alla maggior parte degli utenti. Le fibre che si stanno installando adesso sono utilizzate dal sistema telefonico per gestire simultaneamente molte chiamate a lunga distanza, non per fornire agli utenti singoli una grande larghezza di banda. Inoltre, pochi utenti hanno accesso a un canale a fibra poiché il circuito locale costituito dal fedele doppino è ancora in uso.

Chiamare l'ufficio terminale della società telefonica locale a 28,8 kbps non darà mai una larghezza di banda maggiore di 28,8 kbps, non importa quanto grande sia il collegamento intermedio. Con i satelliti, è pratico per gli utenti alzare una antenna sul tetto di un palazzo e saltare completamente il sistema telefonico. Per molti utenti, saltare il circuito locale è una motivazione fondamentale.

Per gli utenti che (qualche volta) necessitano di 40 o 50 Mbps, una possibilità è di noleggiare una portante T3 (44,736 Mbps). Purtroppo questo è un impegno costoso. Se tutta questa banda serve solo ogni tanto, SMDS può essere una soluzione adatta, ma non è disponibile dovunque mentre il satellite lo è.

Una seconda nicchia è la comunicazione mobile. Molte persone al giorno d'oggi vogliono comunicare mentre corrono, guidano, sono in mare o volano. I collegamenti terrestri a fibra ottica non sono loro di alcuna utilità, mentre i collegamenti satellitari lo sono potenzialmente. È possibile, comunque, che una combinazione di radio cellulare e fibra risulterà adeguata per la maggior parte degli utenti (ma probabilmente non per quelli in aereo o in mare).

Una terza nicchia sono le situazioni in cui è essenziale la diffusione. Un messaggio spedito dal satellite può essere ricevuto da migliaia di stazioni terrestri allo stesso tempo. Per esempio, un'organizzazione che trasmette un flusso di prezzi di azioni, titoli o beni a migliaia di commercianti potrebbe trovare un sistema satellitare molto più economico che simulare una trasmissione a terra.

Una quarta nicchia è per le comunicazioni su un terreno ostile o una infrastruttura terrestre poco sviluppata. L'Indonesia, per esempio, ha il proprio satellite per il traffico telefonico domestico. Lanciare una satellite era molto più facile che tendere migliaia di cavi sotto il mare fra tutte le isole dell'arcipelago.

Una quinta nicchia di mercato per i satelliti si apre quando il diritto di passaggio per porre le fibre è difficile o eccessivamente dispendioso. La sesta è quando un dispiegamento rapido è critico, come nei sistemi di comunicazione militare in tempo di guerra, in cui i satelliti vincono facilmente.

In breve, sembra che il flusso principale di comunicazione del futuro sarà fibre ottiche terestre combinate con radio cellulari, ma per alcuni usi speciali i satelliti sono migliori. Tuttavia, c'è un monito a tutto questo: economia. Sebbene le fibre offrano una maggiore larghezza di banda, è certamente possibile che la comunicazione terrestre e satellitare saranno in concorrenza aggressiva sui prezzi. Se gli avanzamenti tecnologici ridurranno radicalmente il costo di lanciare un satellite (ad es., una navicella spaziale in futuro potrà lasciare dozzine di satelliti in un colpo solo), o i satelliti a bassa orbita avranno successo, non è sicuro che la fibra vinca su tutti i mercati.

2.9 Riassunto

Il livello fisico è la base di tutte le reti. La natura impone due limiti fondamentali su tutti i canali e questi determinano la loro larghezza di banda. Questi limiti sono il limite di Nyquist, che tratta canali senza rumore, e il limite di Shannon, per canali rumorosi.

I mezzi di trasmissione possono essere guidati o non. I mezzi guidati principali sono il doppino, il cavo coassiale e la fibra ottica. Mezzi non guidati includono radio, microonde, infrarossi e laser attraverso l'aria.

Un elemento chiave nella maggior parte delle reti geografiche è il sistema telefonico. I suoi componenti principali sono i circuiti locali, le dorsali e i commutatori. I circuiti locali sono circuiti analogici a doppino, che richiedono modem per trasmettere dati digitali. Le dorsali sono digitali e possono essere multiplexate in diversi modi includendo FDM, TDM e WDM. I commutatori includono i crossbar, i commutatori a divisione di tempo e i commutatori a divisione di spazio. Sia la commutazione a circuito che la commutazione a pacchetto sono importanti.

In futuro il sistema telefonico sarà tutto digitale e trasporterà sulle stesse linee traffico

vocale e dati. Sono state introdotte due versioni di questo nuovo sistema, chiamato ISDN. L'ISDN a banda stretta è un sistema digitale a circuito commutato che è un miglioramento incrementale del sistema attuale. Al contrario, l'ISDN a larga banda rappresenta un cambio di paradigma, poiché si basa sulla tecnologia ATM a commutazione di cella. Esistono diversi tipi di commutatori ATM, inclusi il commutatore knockout e il commutatore Batcher-banyan.

Per applicazioni mobili, il sistema telefonico a fili rigidi non è adatto. Le alternative includono la radio cellulare e i satelliti per comunicazioni. La radio cellulare è adesso utilizzata per i telefoni portatili ma presto si diffonderà anche per il traffico dati. La generazione attuale di sistemi cellulari (es. AMPS) è analogica, ma la generazione successiva (ad es. PCS/PCN) sarà completamente digitale. I satelliti tradizionali per comunicazioni sono geosincroni, ma c'è molto interesse ora nei sistemi satellitari a bassa orbita come Iridium.

Esercizi

- Calcolare i coefficienti di Fourier per la funzione $f(t) = t$ ($0 \leq t \leq 1$).
- Un canale senza rumore a 4 kHz è campionato ogni millisecondo. Qual è il massimo tasso di dati?
- I canali televisivi sono ampi 6 MHz. Quanti bit/s possono essere spediti se si usano segnali digitali a quattro livelli? Si supponga che il canale sia senza rumore.
- Se un segnale binario è spedito su un canale a 3 kHz il cui rapporto segnale-rumore è di 20 dB, qual è il massimo tasso di dati raggiungibile?
- Qual è il rapporto segnale-rumore necessario per mettere una portante T1 su una linea a 50 kHz?
- Qual è la differenza fra una stella passiva e un ripetitore attivo in una rete a fibra ottica?
- Quanta larghezza di banda c'è in uno spettro di $0,1 \mu$ a una lunghezza d'onda di 1μ ?
- Si desidera spedire una sequenza di immagini di schermi di computer su una fibra ottica. Lo schermo è di 480×640 pixel, con ogni pixel di 24 bit. Ci sono 60 immagini di schermo al secondo. Quanta larghezza di banda è necessaria e quanti micron di lunghezza d'onda sono necessari per questa banda a $1,30 \mu$?
- Il teorema di Nyquist è valido anche per la fibra ottica o solo per il filo di rame?
- In figura 2-6 la banda a sinistra è più stretta delle altre. Perché?
- Le antenne radio funzionano meglio quando il diametro dell'antenna è uguale alla lunghezza d'onda delle onde radio. Antenne ragionevoli variano da 1 cm a 5 m di diametro. Quale intervallo di frequenza coprono?
- Il fading multipath è massimizzato quando arrivano due raggi sfasati di 180° . Che

2.9 Riassunto

differenza di percorso è richiesta per massimizzare l'evanescenza per un collegamento di 50 km a microonde di 1 Ghz?

- Un raggio laser largo 1 mm è diretto a un rilevatore largo 1 mm a 100 m di distanza sul tetto di un edificio. Che deviazione angolare (in gradi) il laser deve avere prima che manchi il rilevatore?
- Un semplice sistema telefonico consiste di due uffici terminali e un singolo ufficio di pedaggio al quale ogni ufficio terminale è connesso da una dorsale di 1 MHz full-duplex. Il telefono medio è usato per fare quattro chiamate per un giorno lavorativo di 8 h. La durata media di chiamata è di 6 min. Il 10% delle chiamate sono a lunga distanza (cioè passano attraverso l'ufficio di pedaggio). Qual è il massimo numero di telefoni che un ufficio terminale può supportare? (Supporre 4 kHz per circuito).
- Una società telefonica regionale ha 10.000.000 di abbonati. Ognuno dei loro telefoni è connesso a un ufficio centrale da un doppino di rame. La lunghezza media di questi doppini è di 10 km. Quanto vale il rame dei circuiti locali? Si supponga che la sezione trasversale di ogni filo è un cerchio di 1 mm di diametro, il peso specifico del rame è 9,0 e che il rame si vende a 3 dollari al chilogrammo.
- Il costo di un potente microprocessore è calato al punto da poterne includere uno in ogni modem. Come influenza questo fatto la gestione degli errori di linea telefonica?
- Un diagramma di un modem a costellazione simile alla figura 2-19 ha punti di dati alle seguenti coordinate: (1, 1), (1, -1), (-1, 1) e (-1, -1). Quanti bps può raggiungere un modem con questi parametri a 1200 Bd?
- Un diagramma di un modem a costellazione simile alla figura 2-19 ha punti di dati a (0, 1) e (0, 2). Il modem usa modulazione di fase o modulazione di ampiezza?
- La FTTH si adatta al modello della società telefonica con uffici terminali, di pedaggio e così via o il modello deve essere cambiato in maniera radicale? Spiegare la risposta.
- A basso livello, il sistema telefonico è a forma di stella, con tutti i circuiti locali di un vicinato che convergono in un ufficio terminale. Al contrario, la televisione via cavo consiste di un singolo lungo cavo che serpeggia lungo la strada passando in tutte le case nel vicinato. Si supponga che la futura TV via cavo sia fibra a 10 Gbps invece di rame. Potrebbe essere usato il modello in cui tutti hanno la propria linea privata fino all'ufficio terminale? Se così, quante case con un solo telefono potrebbero essere agganciate a una singola fibra?
- Un sistema di TV via cavo, ha 100 canali commerciali, tutti che alternano programmi con annunci. Questo è più simile al TDM o al FDM?
- Perché il tempo di campionamento del PCM è stato fissato a $125 \mu\text{s}$?
- Qual è la percentuale di scarto su una portante T1: cioè, quale percentuale dei 1,544 Mbps non è consegnata all'utente finale?
- Confrontare il massimo tasso di dati di un canale senza rumore che usa:

- (a) La codifica analogica con 2 bit per campione.
 (b) Il sistema T1 PCM.
25. Se un sistema a portante T1 cade e perde traccia del collegamento, cerca di risincronizzarsi usando il primo bit di ogni pacchetto. Quanti pacchetti devono essere esaminati in media per risincronizzarsi con una probabilità di errore di 0,001?
26. Qual è la differenza, se c'è, tra la parte demodulatore di un modem e la parte di codifica di un codec? (Dopo tutto, entrambi convertono segnali analogici in digitali).
27. Un segnale è trasmesso digitalmente su un canale senza rumore a 4 kHz con un campione ogni 125 μ s. Quanti bit per secondo sono in realtà spediti per ognuno dei seguenti metodi di decodifica?
- (a) CCITT standard a 2,048 Mbps.
 - (b) DPCM con un valore di segnale relativo a 4 bit.
 - (c) Modulazione delta.
28. Un'onda sinusoidale pura di ampiezza A è codificata usando la modulazione delta, con x campioni/s. Un output di +1 corrisponde a un cambiamento di segnale di $+A/8$ e un segnale output di -1 corrisponde a un cambiamento di segnale di $-A/8$. Qual è la più alta frequenza che può essere tracciata senza errori cumulativi?
29. Gli orologi SONET hanno una deriva di circa una parte su 10^9 . Quanto ci vuole perché la deriva egualgi la larghezza di un bit? Quali sono le implicazioni di questo risultato?
30. In figura 2-32, il tasso dati utile per OC-3 è stabilito 148,608 Mbps. Mostrare come questo numero può essere derivato dai parametri SONET OC-3.
31. Qual è la larghezza di banda disponibile all'utente in una connessione OC-12c?
32. Tre reti a commutazione di pacchetto contengono ciascuna n nodi. La prima rete ha una topologia a stella con un commutatore centrale, la seconda è un anello (bidirezionale) e la terza è completamente interconnessa, con un filo da ogni nodo a ogni altro nodo. Quali sono il caso migliore, medio e peggiore di percorso di trasmissione?
33. Confrontare il ritardo nello spedire un messaggio di x bit su un percorso a k salti in una rete commutata a circuito e in una rete (leggermente sovraccarica) a pacchetto. Il tempo di messa a punto del circuito è s sec, il tempo di propagazione è di d s per salto, la dimensione del pacchetto è di p bit e il tasso di dati è b bps. Sotto quali condizioni la rete a pacchetto ha un ritardo minore?
34. Si supponga che x bit di dati utente debbano essere trasmessi su un percorso a k salti in una rete commutata a pacchetto come una serie di pacchetti, ciascuno contenente p bit di dati e h bit di intestazione, con $x > p+h$. Il tasso di bit delle linee è di b bps e il ritardo di propagazione è trascurabile. Quale valore di p minimizza il ritardo totale?
35. Quanti punti di incrocio hanno i commutatori di figura 2-39 (a) e di figura 2-39 (b)? Confrontare con un commutatore crossbar a singolo stadio 16x16.

36. Nel commutatore a divisione di spazio di figura 2-39 (a), qual è il più piccolo numero di connessioni esistenti che possono bloccare una chiamata in uscita?
37. Uno schema alternativo a quello di figura 2-39 (a) è quello in cui le 16 linee sono divise in due blocchi di otto, invece che in quattro blocchi di quattro (cioè $n = 8$ invece di $n = 4$). Tale progetto dovrebbe risparmiare sui costi dell'hardware, poiché solo due concentratori sarebbero necessari sui versanti input e output. Qual è l'argomentazione più forte contro questa alternativa?
38. Quante linee può gestire un commutatore a divisione di tempo se il tempo di accesso alla RAM è di 50 ns?
39. Quanti bit di un buffer RAM sono necessari per uno scambiatore a commutazione di tempo se i campioni della linea input sono di 10 bit e ci sono 80 linee input?
40. La commutazione a divisione di tempo introduce necessariamente un ritardo minimo a ogni stadio di commutazione? Se sì, di quanto?
41. Quanto si impiega a trasmettere un'immagine di 8 in per 10 in via facsimile su un canale ISDN larga banda? Il fax digitalizza l'immagine a 300 pixel per pollice e assegna 4 bit per pixel. Le macchine fax attuali vanno più veloci sulle ordinarie linee telefoniche. Come ci riescono?
42. Citare un vantaggio e uno svantaggio di NT12 (contrapposto alla NT1 e NT2) in una rete ISDN.
43. In figura 2-50 (a) vediamo delle collisioni tra le celle che attraversano un commutatore banyan. Queste collisioni accadono nel primo e nel secondo stadio. Possono accadere collisioni anche nel terzo stadio? Se sì, a quali condizioni?
44. Occorre instradare alcune celle attraverso un commutatore ATM Batcher banyan passo dopo passo. Quattro celle sono presenti sulle linee input da 0 a 3, dirette verso 3, 5, 2 e 1 rispettivamente. Per ognuno dei sei stadi nel commutatore Batcher e i quattro passi nel commutatore banyan (includendo l'input e l'output), elenca quali celle ci sono come ottupla (cella sulla linea 0, cella sulla linea 1 e così via). Indicare le linee senza celle con -.
45. Ripetere il precedente problema partendo da (7, -, 6, -, 5, -, 4, -).
46. Un commutatore ATM ha 1024 linee input e 1024 linee output. Le linee operano alla velocità SONET di 622 Mbps, che produce un tasso utile di approssimativamente 594 Mbps. Quale larghezza di banda aggregata è necessaria al commutatore per gestire il carico? Quante celle al secondo deve essere capace di elaborare?
47. In un tipico sistema telefonico cellulare con celle esagonali, è proibito il riuso di una banda di frequenza in una cella adiacente. Se è disponibile un totale di 840 frequenze, quante ne possono essere usate in una singola cella?
48. Dare una stima approssimata del numero di microcelle PCS di 100 m di diametro che ci vorrebbero per coprire San Francisco (120 km²).

49. Qualche volta quando un utente attraversa il confine da una cella all'altra, la chiamata corrente viene bruscamente interrotta, anche se tutti i trasmettitori e i ricevitori funzionano perfettamente. Perché?
50. I 66 satelliti a bassa orbita nel progetto Iridium sono divisi in 6 collane attorno alla Terra. All'altitudine che usano il periodo è di 90 minuti. Qual è l'intervallo medio per l'handoff per un trasmettitore stazionario?

Capitolo 3

IL LIVELLO DATA LINK

In questo capitolo studieremo la struttura del secondo livello, data link (collegamento dati). Introdurremo gli algoritmi che permettono di conseguire una comunicazione affidabile ed efficiente tra due macchine adiacenti nel livello data link. Per adiacenti intendiamo due macchine fisicamente connesse da un canale di comunicazione che agisca concettualmente come un cavo (ad es. un cavo coassiale o una linea telefonica). La proprietà fondamentale di un canale che agisce come cavo è che i bit vengono recapitati esattamente nello stesso ordine in cui sono stati spediti.

A prima vista potrebbe sembrare che questo problema sia tanto semplice da non richiedere alcun software: la macchina A mette semplicemente i bit sul cavo e la macchina B altrettanto semplicemente li riceve. Purtroppo ogni tanto i circuiti di comunicazione commettono errori. Inoltre essi hanno una velocità di trasmissione finita e si ha un ritardo di propagazione non nullo tra l'istante in cui un bit viene spedito e l'istante in cui viene ricevuto. Queste limitazioni hanno una notevole influenza sull'efficienza del trasferimento dei dati. I protocolli utilizzati per la comunicazione devono tenere conto di tutti questi fattori. Tali protocolli saranno l'argomento di questo capitolo.

Dopo un'introduzione alle caratteristiche fondamentali di progetto del livello data link, studieremo i suoi protocolli considerando la natura degli errori, le loro cause e i modi in cui possono essere individuati e corretti. Studieremo poi una serie di protocolli con complessità crescente che risolvono, in modo sempre più completo, i problemi presenti a questo livello. Concluderemo infine con un'analisi della struttura e della correttezza dei protocolli e daremo alcuni esempi di protocolli data link.

3.1 Princìpi di progettazione del livello data link

Il livello data link deve svolgere diverse funzioni specifiche. Queste funzioni includono una buona interfaccia di servizio al livello rete, il raggruppamento dei bit del livello fisico in pacchetti, la gestione degli errori di trasmissione e la regolazione del flusso dei pacchetti in modo che i riceventi lenti non siano travolti dai pacchetti dei mittenti rapidi. Nel paragrafo che segue esamineremo ognuna di queste caratteristiche.

3.1.1 I servizi forniti al livello rete

Lo scopo del livello data link è quello di fornire servizi al livello rete. Il servizio principale è quello di trasferire dati dal livello rete della macchina sorgente al livello rete della macchina destinazione. Nel livello rete della macchina sorgente esiste un'entità, chiamata

mola processo, che passa alcuni bit al livello data link per la trasmissione alla destinazione. Il compito del livello data link è quello di trasmettere i bit alla macchina destinataria in modo che possano essere passati al livello rete, come si vede in figura 3.1 (a).

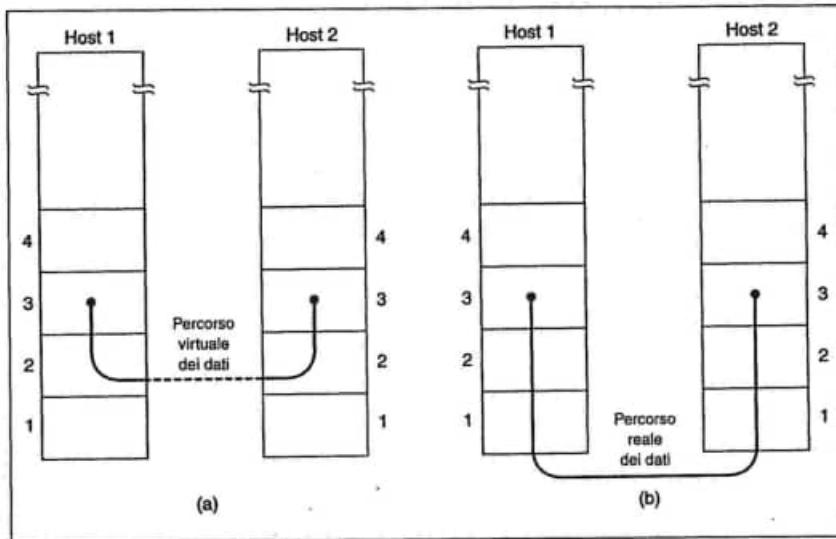


Fig. 3.1 (a) Comunicazione virtuale. (b) Comunicazione reale.

La trasmissione vera segue il percorso descritto in figura 3.1 (b), ma è più semplice pensare in termini di due processi del livello data link che comunicano mediante un protocollo data link. Per questo motivo in tutto il capitolo useremo implicitamente il modello della figura 3.1 (a).

Il livello data link può essere progettato per offrire vari servizi. I servizi effettivamente offerti possono variare da sistema a sistema. Tre possibilità ragionevoli di servizi comunemente offerti sono:

1. Servizio senza connessione e senza riscontro (ack).
2. Servizio senza connessione e con riscontro.
3. Servizio orientato alla connessione con riscontro.

Consideriamoli ora singolarmente.

Il servizio senza connessione e senza riscontro consiste di una macchina mittente che spedisce pacchetti indipendenti alla macchina destinataria senza che questa ne debba accusare la ricezione. Nessuna connessione è stabilita in precedenza o rilasciata successivamente. Se un pacchetto viene perso a causa di un disturbo sulla linea, nessun tentativo di recupero viene attuato a livello data link. Questo genere di servizio è appropriato

quando il tasso di errore è molto basso: la gestione degli errori eventuali viene lasciata ai livelli superiori. È appropriato anche per trasmissioni real-time, come dialoghi, nei quali dati ritardati sono più dannosi di dati scorretti. La maggior parte delle LAN usa un servizio senza connessione e senza riconoscimento per il livello data link.

Il livello successivo in termini di affidabilità è il servizio senza connessione ma con riscontro. Quando il servizio è di questo tipo, come nel caso precedente non si ha connessione, ma per ogni pacchetto spedito viene dato riscontro del ricevimento. In questo modo il mittente riesce a sapere se il pacchetto è arrivato a destinazione. Se non è arrivato entro un certo intervallo di tempo il pacchetto può essere rispedito. Questo servizio è utile quando si hanno canali inaffidabili, come nel caso di sistemi senza filo (wireless).

Vale forse la pena di fare notare che produrre il riscontro nel livello data link è solo una questione di ottimizzazione, non un requisito. Il livello di trasporto può sempre spedire un messaggio e aspettare il riscontro. Se il riscontro non arriva a tempo debito il mittente può rispedire l'intero messaggio. Il problema di questa strategia è che se un messaggio medio è suddiviso in diciamo 10 pacchetti e se ne perde il 20%, il tempo di spedizione del messaggio è notevole. Se invece viene dato il riscontro dei singoli pacchetti ed essi vengono poi rispediti, il messaggio viene consegnato molto più velocemente. Su canali affidabili, come quelli in fibra, il sovraccarico di un protocollo data link pesante può non essere necessario, ma su canali senza filo tali protocolli convengono data l'inaffidabilità intrinseca di tali sistemi.

Ritornando ai servizi, il più sofisticato che il livello data link può fornire al livello rete è quello orientato alla connessione. Con questo servizio la macchina sorgente e la macchina destinazione stabiliscono una connessione prima del trasferimento dei dati. Ogni pacchetto spedito sulla connessione viene numerato e il livello data link garantisce che ogni pacchetto spedito venga anche ricevuto. Inoltre garantisce che ogni messaggio sia ricevuto una sola volta e che tutti i pacchetti siano ricevuti nell'ordine esatto. Con un servizio senza connessione, invece, è immaginabile che un riscontro perso causi la spedizione e la ricezione multipla del pacchetto. Diversamente, il servizio orientato alla connessione fornisce ai processi del livello rete l'equivalente di un flusso affidabile di bit.

Quando si utilizza un servizio orientato alla connessione, i trasferimenti si svolgono in tre fasi distinte. Nella prima viene stabilita la connessione nella quale entrambe le parti inizializzano le variabili e i contatori necessari per tenere traccia dei pacchetti che sono stati ricevuti e di quelli che invece non lo sono stati. Nella seconda fase uno o più pacchetti sono realmente trasmessi. Nella terza e ultima fase viene rilasciata la connessione liberando le variabili, i buffer e le altre risorse utilizzate per mantenere la connessione.

Si consideri questo tipico esempio: una sottorete WAN composta di router connessi punto-a-punto su linea telefonica. Quando un pacchetto raggiunge un router, l'hardware verifica il checksum (somma di controllo) e passa il pacchetto al software del livello data link (che può essere inglobato in un chip della scheda di rete). Il software del livello data link controlla se quello è il pacchetto atteso e, se è così, consegna il pacchetto contenuto nel campo riservato al carico (payload) al software di instradamento. Il software di instradamento sceglie la giusta linea di uscita a ripassa il pacchetto al software del livello data link che quindi lo trasmette. Il flusso tra due router è mostrato in figura 3-2.

Il codice di instradamento spesso richiede che il processo sia corretto, cioè che le connessioni

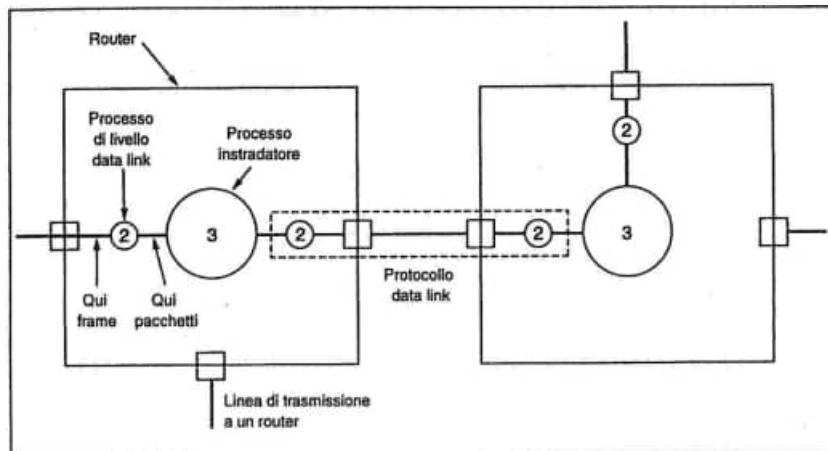


Fig 3-2 Posizione del protocollo data link.

siano affidabili e sequenzializzate su ognuna delle linee punto-a-punto. Non vuole avere a che fare troppo spesso con pacchetti persi per strada. È compito del protocollo data link, mostrato nel rettangolo tratteggiato, rendere perfette, o almeno piuttosto buone, le linee di comunicazione inaffidabili. Questa proprietà è particolarmente importante per collegamenti senza filo, che sono intrinsecamente molto inaffidabili. Da notare che sebbene mostriamo per ogni router copie differenti del software del livello data link, in realtà una stessa copia gestisce tutte le linee, con tabelle e strutture dati diverse per ognuna.

Anche se questo capitolo tratta principalmente il livello data link e i protocolli data link, molti dei principi che studieremo qui, come il controllo degli errori e il controllo di flusso, si ritrovano anche in protocolli di livello trasporto e in altri.

3.1.2 Impacchettamento (framing)

Al fine di fornire servizi al livello rete, il livello data link deve usufruire dei servizi forniti dal livello fisico. Il livello fisico accetta un flusso di bit allo stato grezzo e cerca di consegnarli a destinazione. Questo flusso di bit non è garantito come esente da errori. Il numero di bit ricevuti potrebbe essere minore, uguale o maggiore del numero dei bit trasmessi e i bit possono avere valori diversi. È compito del livello data link quello di individuare e, se necessario, correggere gli errori.

L'approccio consueto del livello data link è quello di dividere il flusso di bit in pacchetti e di calcolarne la checksum (gli algoritmi di checksum verranno trattati più avanti in questo stesso capitolo). Quando un pacchetto arriva a destinazione la checksum viene ricalcolata. Se il nuovo risultato è diverso da quello contenuto nel pacchetto, il livello data link riconosce che deve essere stato commesso un errore e prende adeguati provvedimenti (ad es. scarta il pacchetto e spedisce in risposta un messaggio di errore).

Suddividere il flusso di bit in pacchetti è più difficile di quanto sembri. Un modo per

ottenere questa suddivisione è quello di inserire degli intervalli temporali tra i pacchetti, un po' come gli spazi tra le parole in un testo. Le reti, comunque, difficilmente danno delle garanzie temporali, è quindi possibile che questi intervalli siano eliminati, oppure che altri intervalli siano inseriti durante la trasmissione.

Poiché è troppo rischioso affidarsi al tempo per determinare l'inizio e la fine di ogni pacchetto, sono stati inventati altri metodi. In questo paragrafo considereremo quattro metodi:

1. Conteggio di caratteri.
2. Caratteri di inizio e fine, con riempimento di caratteri.
3. Indicatori (flag) di inizio e fine, con riempimento di bit.
4. Violazioni di codifica del livello fisico.

Il primo metodo di impacchettamento utilizza un campo nell'intestazione per specificare il numero di caratteri del pacchetto. Quando il livello data link della destinazione legge il contatore di caratteri, viene informato di quanti caratteri seguiranno e, di conseguenza, sa dove trovare la fine del pacchetto. Questa tecnica è mostrata in figura 3-3 (a) per quattro pacchetti di rispettivamente 5, 5, 8 e 8 caratteri.

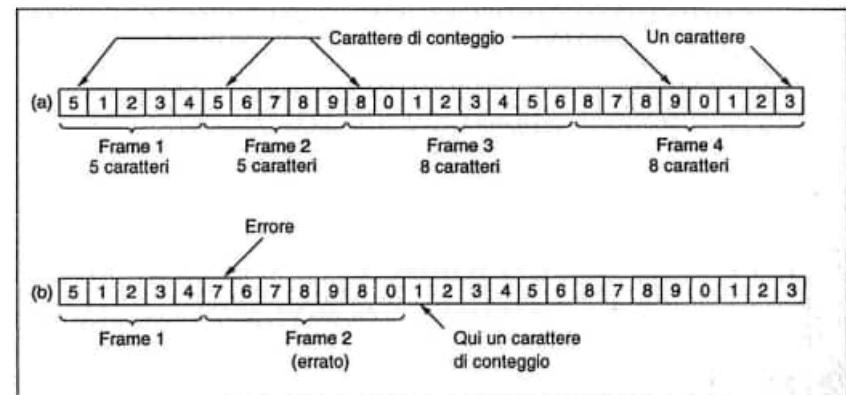


Fig. 3-3 Un flusso di caratteri. (a) Senza errori. (b) Con un errore.

Il problema di questo algoritmo è che il conteggio può essere alterato da errori di trasmissione. Per esempio, se il carattere di conteggio 5 del secondo pacchetto nella figura 3-3 (b) diventa 7, la destinazione perderà la sincronizzazione con la sorgente e non sarà più capace di determinare l'inizio del pacchetto successivo. Anche se la checksum è scorretta e quindi la destinazione sa che il pacchetto è alterato, non ha comunque modo di sapere dove inizia il pacchetto successivo. Non è una soluzione neppure rimandare un pacchetto alla sorgente chiedendo la ritrasmissione, poiché la destinazione non sa quanti caratteri deve saltare prima di ricevere la ritrasmissione. Per questo motivo il metodo di conteggio di caratteri non è più molto usato.

Il secondo metodo aggira il problema della risincronizzazione dopo un errore facendo iniziare ogni pacchetto con la sequenza di caratteri ASCII, DLE STX e facendolo finire con la sequenza DLE ETX. (DLE sta per Data Link Escape, STX per Start of TeXt e ETX per End of TeXt). In questo modo se la destinazione perdesse memoria dei confini del pacchetto, tutto quello che dovrebbe fare sarebbe di cercare la stringa DLE STX oppure DLE ETX per risincronizzarsi.

Un grave problema compare, con questo metodo, se vengono trasmessi dati binari, come programmi oggetto o numeri in virgola mobile. Può facilmente succedere infatti, che le stringhe DLE STX o DLE ETX che compaiono nei dati interferiscano con la frammentazione. Un modo per risolvere questo problema è quello di avere un livello data link che inserisca un carattere DLE ASCII appena prima di ogni occorrenza accidentale di DLE nei dati. Il livello data link della destinazione deve quindi rimuovere i DLE prima di passare i dati al livello rete. Questa tecnica viene detta di riempimento di caratteri. In questo modo una sequenza di controllo DLE STX o DLE ETX può essere distinta da una sua occorrenza nei dati, dall'assenza o presenza di un singolo DLE. I DLE nei dati sono sempre duplicati. La figura 3-4 mostra un esempio di sequenza di dati prima del riempimento, dopo il riempimento e dopo l'eliminazione dei caratteri di riempimento.

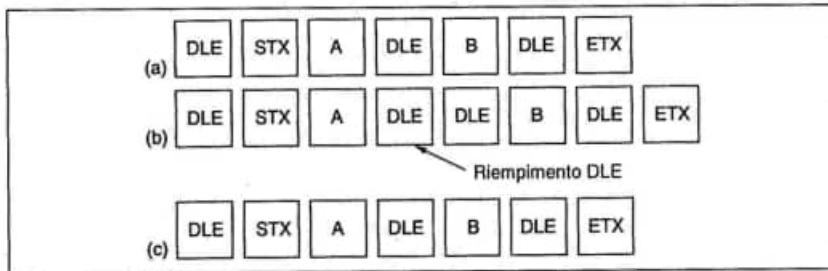


Fig. 3-4 (a) Dati spediti dal livello rete. (b) Dati dopo il riempimento di caratteri del livello data link. (c) Dati passati al livello rete della macchina destinazione.

Uno dei maggiori svantaggi dell'utilizzo di questa tecnica è che essa è molto legata ai caratteri di 8 bit in generale, e al codice di caratteri ASCII in particolare. Con lo sviluppo delle reti, lo svantaggio di incorporare il codice di caratteri nel meccanismo di impacchettamento è divenuto sempre più ovvio. Quindi, è stata sviluppata una nuova tecnica per permettere l'impiego di caratteri di dimensioni arbitrarie.

La nuova tecnica ammette che i pacchetti di dati contengano un numero arbitrario di caratteri con un numero arbitrario di bit per carattere. Il funzionamento viene descritto di seguito. Ogni pacchetto inizia e finisce con una sequenza speciale di bit, 0111110, chiamata **byte indicatore** (flag byte). Quando il livello data link mittente incontra cinque bit 1 consecutivi nei dati, automaticamente inserisce un bit 0 nel flusso di bit in uscita. Questo **riempimento di bit** è analogo al riempimento di caratteri, in cui la sequenza DLE viene inserita nel flusso di caratteri in uscita prima di un DLE nei dati.

Quando il ricevente vede arrivare cinque bit 1 consecutivi seguiti da uno 0, automaticamente

elimina il bit 0. Proprio come nel caso di riempimento di caratteri, questa tecnica è completamente trasparente al livello rete su entrambe le macchine. Se i dati contengono la sequenza speciale, 0111110, questa viene trasmessa come 01111010 ma memorizzata nella memoria del ricevente come 0111110. La figura 3-5 mostra un esempio di riempimento di bit.

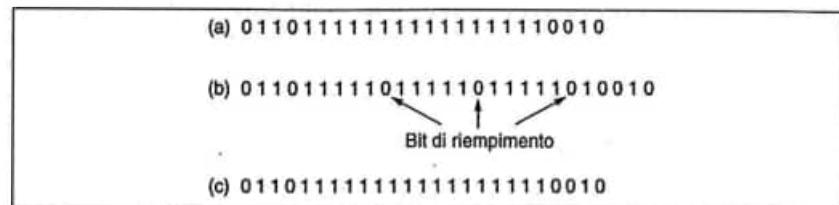


Fig. 3-5 Riempimento di bit. (a) I dati originali. (b) I dati come appaiono sulla linea. (c) I dati memorizzati dal ricevente dopo l'eliminazione dei caratteri di riempimento.

Con il riempimento di bit, il confine tra due pacchetti può essere determinato in modo non ambiguo attraverso la sequenza speciale di bit. In questo modo se il ricevente perde traccia della posizione, tutto quello che deve fare è di scorrere l'input cercando la sequenza speciale, poiché essa non può comparire nei dati.

L'ultimo metodo di impacchettamento è applicabile solo a reti in cui la codifica su mezzo fisico contenga ridondanze. Per esempio, alcune LAN codificano il bit 1 nei dati usando due bit fisici. Normalmente, il bit 1 è una coppia alto-basso e lo 0 è una coppia basso-alto. Le combinazioni alto-alto e basso-basso non sono usate per i dati. Questo schema significa che ogni bit di dati ha una transizione nel mezzo ed è facile per il ricevente determinare i confini del bit. Questo impiego di codici fisici non validi è parte dello standard LAN 802 che studieremo nel capitolo 4.

Una nota finale sull'impacchettamento: molti protocolli data link usano una combinazione di conteggio di caratteri con uno degli altri metodi per ottenere ulteriori garanzie di sicurezza. Quando arriva un pacchetto, il campo per il conteggio è utilizzato per memorizzarne la fine. Solo se è presente l'apposito limite in quella posizione e la checksum è corretta il pacchetto viene accettato come valido; altrimenti il flusso di input viene scandito alla ricerca del successivo delimitatore.

3.1.3 Controllo degli errori

Risolto il problema della delimitazione di ogni pacchetto, arriviamo al problema successivo: come essere sicuri che tutti i pacchetti siano prima o poi consegnati al livello rete della destinazione e nel giusto ordine. Supponiamo che il mittente spedisca i messaggi senza accertarsi che questi arrivino correttamente. Questo potrebbe andare bene per un tipo di servizio senza connessione e senza riscontro, ma non sarebbe opportuno per un tipo di servizio orientato alla connessione affidabile.

Il modo consueto per assicurare una consegna affidabile è quello di fornire al mittente un riscontro di quello che sta accadendo all'altro capo della linea. Tipicamente il protocollo richiede che il ricevente rispedisca alcuni speciali pacchetti di controllo con valore posi-

tivo o negativo a seconda dei pacchetti ricevuti. Se il mittente riceve un riscontro positivo su di un pacchetto spedito, sa che esso è arrivato correttamente. Se invece ottiene un riscontro negativo significa che qualcosa è andato male e che occorre ritrasmettere il pacchetto.

Una complicazione aggiuntiva viene dalla possibilità che problemi hardware causino la sparizione totale del pacchetto (ad es. una raffica di rumore). In questo caso il ricevente non reagirà in alcun modo poiché non ne ha motivo. Dovrebbe essere chiaro che in un tipo di protocollo dove si trasmette un pacchetto e quindi si aspetta un riscontro, positivo o negativo, se un pacchetto scompare per problemi di malfunzionamento dell'hardware, il mittente aspetterà all'infinito.

Questa possibilità viene affrontata introducendo un timer a livello data link. Quando il mittente trasmette un pacchetto inizializza anche un timer. Il timer è programmato per scadere dopo un tempo sufficiente perché il pacchetto possa raggiungere la destinazione, qui essere elaborato e il pacchetto di conferma possa ritornare al mittente. Normalmente il pacchetto viene ricevuto correttamente e l'ack viene rispedito prima dello scadere del timer: in questo caso il timer viene bloccato.

Tuttavia, se il pacchetto o il messaggio di riscontro vengono persi, il timer scade avvertendo il mittente del potenziale problema. La soluzione più ovvia è quella di ritrasmettere il pacchetto. Però, quando i pacchetti possono essere trasmessi più volte c'è il pericolo che il ricevente accetti e passi al livello rete lo stesso pacchetto due o più volte. Per prevenire questo problema, è necessario assegnare numeri progressivi ai pacchetti spediti in modo che il ricevente possa distinguere le ritrasmissioni dai pacchetti originali.

La gestione dei timer e dei numeri progressivi in modo da assicurare che ogni pacchetto sia passato al livello rete della destinazione esattamente una volta, né più né meno, è uno dei compiti più importanti del livello data link. Più avanti nel capitolo studieremo in dettaglio come questa gestione venga effettuata utilizzando una serie di esempi via via più complicati.

3.1.4 Controllo del flusso

Un altro importante problema di progettazione che si ritrova nel livello data link (e anche negli strati più alti) è quello di decidere cosa fare di un mittente che sistematicamente tende a trasmettere pacchetti più velocemente di quanto il ricevente li accetti. Questa situazione può facilmente essere riscontrata quando il mittente è dislocato su una macchina veloce (o a basso carico) e il ricevente su una macchina lenta (o ad alto carico). Il mittente continua a spedire pacchetti ad alta velocità fino a quando il ricevente non è completamente sopraffatto. Anche se la trasmissione è esente da errori, a un certo punto il ricevente non sarà più in grado di gestire i pacchetti in arrivo e inizierà a perderli. Chiaramente occorre escogitare qualcosa per prevenire questo tipo di situazione.

La tipica soluzione è quella di introdurre un controllo di flusso per obbligare il mittente a rispettare la velocità del ricevente nello spedire i pacchetti. Questa imposizione solitamente richiede un certo tipo di meccanismo di riscontro in modo che il mittente possa essere avvisato se il ricevente è in grado o meno di ricevere.

Sono note diverse tecniche di controllo ma la maggior parte usano lo stesso principio di base. Il protocollo contiene regole ben definite su come il mittente può trasmettere il

pacchetto successivo. Queste regole spesso proibiscono la spedizione di pacchetti fino a quando il ricevente non concede il permesso, implicito o esplicito. Per esempio, quando viene stabilita una connessione, il ricevente potrebbe dire: "Adesso puoi inviarmi n pacchetti, ma dopo averli spediti non inviare nulla fino a quando non ti dico di continuare". In questo capitolo studieremo diversi meccanismi di controllo di flusso basati su questo principio. Nel capitolo seguente studieremo altri meccanismi.

3.2 Rilevazione e correzione degli errori

Come abbiamo visto nel capitolo 2 il sistema telefonico è composto da tre parti: i commutatori, le dorsali regionali e i circuiti locali. Negli Stati Uniti e in diversi altri paesi le prime due sono quasi completamente digitalizzate. Quasi ovunque invece i circuiti locali sono ancora analogici, in coppie di fili di rame attorcigliati (doppini) e continueranno ad essere così per decenni, data l'enorme spesa che comporterebbe il loro rimpiazzamento. Mentre gli errori sono rari sulle parti digitali, sono invece molto frequenti sui circuiti locali. Inoltre la comunicazione senza filo sta divenendo sempre più comune, e il grado di errore in questo caso è più elevato che nelle dorsali regionali in fibra. Se ne deduce che gli errori di trasmissione continueranno ad essere un fatto della vita per molti anni a venire.

A causa del processo fisico che li genera, gli errori su certi mezzi (come per esempio le onde radio) tendono a presentarsi a raffiche piuttosto che isolati. Questo ha degli svantaggi e dei vantaggi. Sul piano dei vantaggi, notiamo che i dati vengono sempre inviati come blocchi di bit. Supponiamo che la dimensione del blocco sia di 1000 bit e che il tasso di errore sia di 0,001 per bit. Se gli errori fossero indipendenti, molti blocchi conterrebbero errori. Se invece gli errori avvenissero a raffiche di 100, in media, solo uno o due blocchi su 100 presenterebbero errori. Lo svantaggio di avere gli errori a raffiche è che è più difficile individuarli e correggerli.

3.2.1 Codici di correzione degli errori

I progettisti di rete hanno sviluppato due strategie di base per gestire gli errori. La prima consiste nell'includere, nel blocco di dati da spedire, sufficiente informazione ridondante per permettere al ricevente di dedurre quale doveva essere il carattere trasmesso. L'altra strategia consiste nell'includere nel blocco sufficiente informazione perché il ricevente possa dedurre se è stato commesso un errore, ma non il tipo di errore, e chiedere quindi la ritrasmissione del blocco. La prima strategia utilizza **codici di correzione di errore** e la seconda **codici di rilevamento di errore**.

Per capire come possono essere gestiti gli errori è necessario osservare bene cosa è un errore in realtà. Generalmente un pacchetto consiste di m bit di dati (cioè un messaggio) e di r bit di controllo (o ridondanti). Supponiamo che la lunghezza totale sia n (cioè $n = m + r$). Una unità di n bit contenenti dati e bit di controllo viene solitamente detta **parola di codice** (codeword) su n bit.

Date due parole di codice, diciamo 10001001 e 10110001, è possibile determinare quanti bit corrispondenti differiscono. In questo caso differiscono 3 bit. Per determinare quanti bit differiscono, basta effettuare l'OR esclusivo delle due stringhe e contare il numero di bit 1

del risultato. Il numero di posizioni in cui differiscono le due parole viene detta **distanza di Hamming** (Hamming, 1950). Il suo significato è il seguente: se due stringhe hanno distanza di Hamming d , saranno necessari d errori di singoli bit per convertire l'una nell'altra. In molte applicazioni di trasmissioni di dati, tutte le possibili stringhe 2^m di bit sono legali, ma per il modo in cui i bit di controllo sono calcolati, non sono utilizzate tutte le possibili codeword 2^n . Dato l'algoritmo per il calcolo dei bit di controllo, è possibile costruire una lista completa delle parole legali e da questa lista determinare le due la cui distanza di Hamming è minima. Questa distanza è la distanza di Hamming dell'intero codice. Le proprietà di un codice di rilevare e di correggere gli errori dipende dalla sua distanza di Hamming. Per rilevare d errori è necessario un codice con distanza $d+1$ poiché con questo codice non c'è modo che d errori di bit singolo mutino una codeword valida in un'altra codeword valida. Quando il ricevente trova una codeword non valida può dire che si è avuto un errore di trasmissione. Similmente, per correggere d errori è necessario un codice con distanza di $2d+1$ poiché le sue parole valide sono così distanti che anche con d mutazioni, la parola originale è ancora la più vicina valida e può essere quindi individuata.

Come esempio di codice di rilevamento di errore consideriamone uno in cui un singolo **bit di parità** venga appeso ai dati. Il bit di parità è scelto in modo che il numero di bit 1 nella stringa sia fisso (pari o dispari). Per esempio, quando viene spedita con un codice con bit di parità pari la stringa 10110101, aggiungendo un bit 1 in fondo, essa diventa 101101011, mentre in un codice con bit di parità dispari sarebbe diventata 101101010. Un codice con un bit singolo di parità ha una distanza 2 poiché ogni errore di bit singolo produce una parola di codice con la parità sbagliata. Esso può essere utilizzato per rilevare errori singoli.

Come esempio di correzione di errori consideriamo un codice con solo quattro parole valide:

0000000000, 0000011111, 1111100000, e 1111111111

Questo codice ha distanza 5, che significa che può correggere errori doppi. Se arriva la stringa 0000000111, il ricevente sa che quella originale doveva essere 0000001111. Se invece un errore triplo muta 0000000000 in 0000000111, l'errore non potrà essere corretto in modo appropriato.

Immaginiamo di voler progettare un codice con m bit di messaggio ed r bit di controllo, che consenta la correzione di tutti gli errori singoli. Ognuno dei messaggi legali 2^m possiede n stringhe illegali a distanza 1. Queste stringhe sono ottenibili invertendo sistematicamente ciascuno degli n bit della stringa di n bit originale. Così ognuno dei messaggi legali 2^m necessita di $n+1$ combinazioni di bit dedicate. Poiché il numero totale di combinazioni è 2^n , si devono avere $(n+1) 2^m \leq 2^n$. Poiché $n=m+r$ questa condizione diventa $(m+r+1) \leq 2^r$. Dato m , questo definisce un limite inferiore al numero di bit di controllo necessari per correggere un errore singolo.

Questo limite inferiore teorico può essere ottenuto utilizzando una tecnica dovuta ad Hamming (1950). I bit della parola di codice vengono numerati progressivamente, ponendo il bit 1 all'estremità sinistra. I bit che sono potenza di 2 (1, 2, 4, 8, 16...) sono bit di controllo. Nei rimanenti bit (3, 5, 6, 7, 9...) vengono sistemati gli m bit di dati. Ogni

bit di controllo forza la parità di un insieme di bit, incluso se stesso, ad essere fissa (pari o dispari). Un bit può essere incluso in diversi insiemi di parità. Per sapere su quali bit di parità influisce il bit di dati k , basta riscrivere k come somma di potenze di 2. Per esempio, $11 = 1+2+8 + 29 = 1+4+8+16$. Un bit è controllato da tutti e soli i bit di controllo che appartengono alla sua espansione (cioè per esempio il bit 11 è controllato dai bit 1, 2 e 8).

Quando arriva una parola di codice, il ricevente inizializza un contatore a zero. Inizia poi a esaminare ogni bit di controllo k ($k = 1, 2, 4, 8, \dots$), per vedere se la stringa ha parità corretta. Se no, esso aggiunge k al contatore. Se il contatore è zero dopo aver esaminato tutti i bit di controllo (cioè se erano tutti corretti), la parola di codice è accettata come valida. Se il contatore invece è diverso da zero, la parola conterrà alcuni bit scorretti. Per esempio, se i bit di controllo 1, 2 e 8 sono scorretti il bit invertito sarà l'11, poiché esso è l'unico bit controllato da tutti e soli i bit di controllo 1, 2 e 8. La figura 3-6 mostra alcuni caratteri di 7 bit codificati come stringa di 11 bit usando il codice di Hamming. Si ricordi che i bit di dati si trovano nelle posizioni 3, 5, 6, 7, 9, 10 e 11.

Car.	ASCII	Bit di controllo
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	11111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	00101011111
d	1100100	11111001100
e	1100101	00111000101

Bit in ordine di trasmissione

Fig. 3-6 Uso del codice di Hamming per correggere errori a raffica.

I codici di Hamming possono correggere solo errori singoli. Tuttavia c'è un trucco che permette ai codici di Hamming di correggere anche errori a raffica. Una sequenza di k stringhe consecutive viene sistemata in una matrice, una parola per riga. Generalmente, i dati sarebbero trasmessi una parola per volta, da destra a sinistra. Per correggere errori a raffica, i dati dovrebbero invece essere trasmessi una colonna per volta, partendo da quella più a sinistra. Quando tutti i k bit sono stati spediti, viene inviata la seconda colonna e così via. Quando un pacchetto arriva al ricevente, la matrice viene ricostruita, una colonna per volta. Se si presenta una raffica di errori di lunghezza k , un bit al massimo per ognuna delle k parole risulta alterato, e poiché il codice di Hamming è in grado di correggere un errore per parola, tutto il blocco potrà essere recuperato. Questo metodo utilizza kr bit di controllo per rendere immuni da errori a raffiche di lunghezza k o minore, km bit di dati.

3.2.2 Codici di rilevazione di errori

I codici di correzione di errore sono usati alcune volte per la trasmissione di dati, per esempio, quando il canale è simplex, e quindi la ritrasmissione non può essere richiesta, ma più spesso si preferisce la rilevazione degli errori con conseguente ritrasmissione perché è più efficiente. Come semplice esempio si consideri un canale sul quale si possano avere errori isolati e in cui il tasso di errore sia di 10^{-6} per bit; sia la lunghezza dei blocchi di 1000 bit. Per garantire correzione degli errori per blocchi di 1000 bit, sono necessari 10 bit di controllo; un megabit di dati richiederebbe 10.000 bit di controllo. Invece per riconoscere errori in un blocco scorretto, è sufficiente un solo bit di parità per blocco. Ogni 1000 blocchi deve essere trasmesso un solo blocco extra (1001 bit). L'appendimento totale per un metodo di riconoscimento degli errori e la ritrasmissione è dunque solo di 2001 bit per megabit di dati contro 10.000 bit del codice di Hamming.

Se viene aggiunto a un blocco un solo bit di parità e il blocco è stato modificato da una lunga raffica di errori, la probabilità che l'errore sia riconosciuto è di 0,5, che è un valore non accettabile. Questi dati possono essere notevolmente migliorati considerando ogni blocco spedito come matrice rettangolare larga n e alta k . Un bit di parità viene calcolato separatamente per ogni colonna e appeso alla matrice come ultima colonna. Quindi la matrice viene trasmessa una riga per volta. Quando arriva il blocco, il ricevente controlla tutti i bit di parità. Se uno di essi è scorretto richiede la trasmissione del blocco.

Questo metodo permette di riconoscere errori a raffica di lunghezza n , poiché sarà cambiato solo un bit per colonna. Tuttavia, una raffica di lunghezza $n+1$ non sarebbe riconosciuta, se il primo bit fosse invertito e tutti gli altri corretti. (Una raffica di errori non implica che tutti i bit siano mutati; ma solo che almeno il primo e l'ultimo lo siano). Se il blocco viene modificato da una lunga raffica di errori o da corte raffiche multiple, la probabilità che qualcuna delle n colonne abbia il bit di parità giusto per caso, è di 0,5, quindi la probabilità che un blocco corrotto venga accettato quando non dovrebbe è di 2^{-n} .

Sebbene il metodo precedente qualche volta possa essere adeguato, in genere viene comunemente utilizzato un altro metodo: il codice polinomiale (noto anche come codice di ridondanza ciclica o CRC code). I codici polinomiali considerano le stringhe di bit come rappresentazioni di polinomi con coefficienti 0 e 1. Un pacchetto di k bit viene visto come una lista di coefficienti di un polinomio a k termini compresi fra x^{k-1} e x^0 . Questo polinomio è detto di grado $k-1$. Il bit di livello più alto (il più a sinistra) è il coefficiente di x^{k-1} ; il bit successivo è coefficiente di x^{k-2} , e così via. Per esempio, 110001 ha 6 bit e quindi rappresenta un polinomio con sei termini con coefficienti 1, 1, 0, 0, 0 e 1: $x^5 + x^4 + x^0$.

L'aritmetica polinomiale viene sviluppata in modulo 2, in accordo con le regole della teoria dei campi algebrici. Non ci sono riporti per l'addizione o prestiti per la sottrazione. Entrambe, l'addizione e la sottrazione, sono identiche all'OR esclusivo. Per esempio:

$$\begin{array}{r} 10011011 \quad 00110011 \quad 11110000 \quad 01010101 \\ + 11001010 \quad + 11001101 \quad - 10100110 \quad - 10101111 \\ \hline 01010001 \quad 11111110 \quad 01010110 \quad 11111010 \end{array}$$

3.2 Rilevazione e correzione degli errori

La divisione lunga viene svolta come se fosse binaria a eccezione del fatto che la sottrazione è eseguita in modulo 2, come sopra. Un divisore è "contenuto" nel dividendo se il dividendo ha tanti bit quanti il divisore.

Quando viene impiegato il metodo del codice polinomiale, il mittente e il ricevente devono accordarsi in anticipo su un generatore polinomiale, $G(x)$. Entrambi i bit di più alto e più basso ordine devono essere uno. Per calcolare la somma di controllo (checksum) di alcuni pacchetti di m bit, corrispondenti al polinomio $M(x)$, il pacchetto deve essere più lungo del generatore polinomiale. L'idea è quella di appendere una checksum alla fine del pacchetto in modo che il polinomio rappresentato dal pacchetto su cui si è verificata la checksum, sia divisibile per $G(x)$. Quando il ricevente ottiene il pacchetto verificato, prova a dividerlo per $G(x)$. Se la divisione dà un resto ci deve essere stato un errore di trasmissione.

L'algoritmo per il calcolo della checksum è il seguente:

1. Sia r il grado di $G(x)$. Si appendano r bit 0 dopo il bit di ordine più basso del pacchetto in modo che esso arrivi a contenere $m+r$ bit e corrisponda al polinomio $x^r M(x)$.
2. Si divida la stringa di bit corrispondente a $G(x)$ per la stringa di bit corrispondente a $x^r M(x)$ usando la divisione modulo 2.
3. Si sottragga il resto (che è sempre r bit o meno) dalla stringa di bit corrispondente a $x^r M(x)$ usando la sottrazione modulo 2. Il risultato è il pacchetto verificato da trasmettere. Chiamiamo il suo polinomio $T(x)$.
4. La figura 3-7 mostra il calcolo per il pacchetto 1101011011 e $G(x) = x^4 + x + 1$.

Dovrebbe essere chiaro che $T(x)$ è divisibile (modulo 2) per $G(x)$. In ogni problema di divisione, se si sottrae il resto dal dividendo, quello che rimane è divisibile per il divisore. Per esempio, in base 10, se si divide 210.278 per 10.941, il resto è 2399. Sottraendo 2399 da 210.278, resta 207.879 che è divisibile per 10.941.

Ora analizziamo le potenzialità di questo metodo. Che tipo di errori è in grado di individuare? Immaginiamo che sia avvenuto un errore di trasmissione: invece di ricevere la stringa di bit $T(x)$, si riceve $T(x) + E(x)$. Ogni 1 in $E(x)$ corrisponde a un bit invertito. Se ci sono k bit 1 in $E(x)$, sono avvenuti k errori di un singolo bit. Una sola raffica di errori è caratterizzata da un 1 iniziale, una sequenza di 0 e 1, e da un 1 finale, mentre gli altri bit sono tutti 0.

Dopo aver ricevuto il pacchetto verificato, il ricevente lo divide per $G(x)$; cioè calcola $[T(x) + E(x)] / G(x)$. $T(x) / G(x) = 0$ quindi il risultato della computazione è $E(x) / G(x)$. Gli errori che corrispondono a polinomi contenenti il fattore $G(x)$ non saranno rilevati; tutti gli altri sì.

Se ci sono stati errori di singoli bit, $E(x) = x^i$, dove i determina il bit errato. Se $G(x)$ contiene due o più termini, non sarà mai in grado di dividere $E(x)$, quindi tutti gli errori di singoli bit saranno riconosciuti.

Se sono avvenuti due errori isolati di un singolo bit, si ha $E(x) = x^i + x^j$, dove $i > j$. questo può anche essere scritto come $E(x) = x^i(x^{i-j} + 1)$. Se assumiamo che $G(x)$ non sia

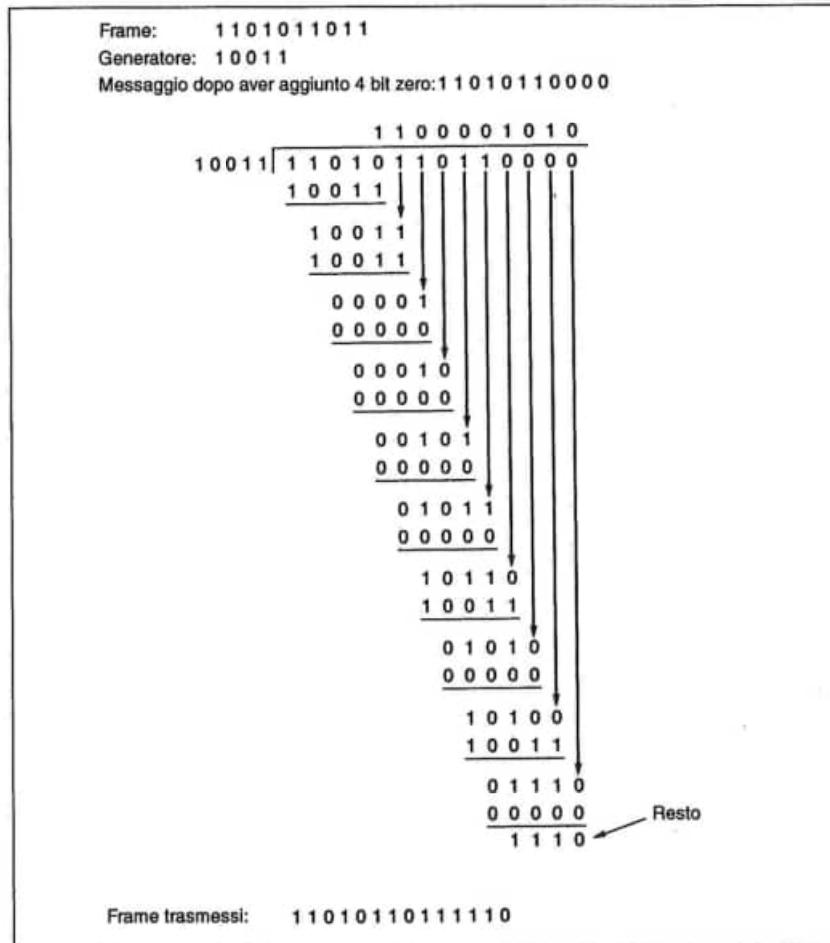


Fig. 3-7 Calcolo della somma di controllo del codice polinomiale.

divisibile per x , una condizione sufficiente affinché tutti gli errori doppi siano riconosciuti è che $G(x)$ non divida $x^k + 1$ per ogni k fino al valore massimo di $i-j$ (cioè fino alla lunghezza massima del pacchetto). Sono noti polinomi semplici e di basso grado che danno protezione a lunghi pacchetti. Per esempio, $x^{15} + x^{14} + 1$ non divide $x^k + 1$ per ogni k minore di 32.768.

Se c'è un numero dispari di bit errati, $E(x)$ conterrà un numero dispari di termini (per esempio, $x^5 + x^2 + 1$, ma non $x^2 + 1$). Da notare che non esiste un polinomio con un numero dispari di termini che abbia $x+1$ come fattore nel sistema a modulo 2. Rendendo

$x+1$ fattore di $G(x)$, possiamo determinare tutti gli errori consistenti in un numero dispari di bit invertiti.

Per verificare che nessun polinomio con un numero dispari di termini è divisibile per $x+1$, assumiamo che $E(x)$ abbia un numero dispari di termini e che sia divisibile per $x+1$. Fattorizziamo $E(x)$ come $(x+1)Q(x)$. Calcoliamo ora $E(1) = (1+1)Q(1)$. Poiché $1+1=0$ (modulo 2), $E(1)$ deve essere 0. Se $E(x)$ ha un numero dispari di termini, sostituendo x con 1 ovunque otterremo 1 come risultato. Quindi nessun polinomio con un numero dispari di termini è divisibile per $x+1$.

Ultima e più importante nota, un codice polinomiale con r bit di controllo riconoscerà tutte le raffiche di errori di lunghezza (\leq). Una raffica di errori di lunghezza k può essere rappresentata da $x^i(x^{k-1} + \dots + 1)$, dove i indica la distanza dall'estremità destra del pacchetto ricevuto della raffica di errori. Se $G(x)$ contiene un termine x^0 non avrà x^i come fattore, quindi, se il grado dell'espressione parametrizzata è minore del grado di $G(x)$, il resto non potrà mai essere zero.

Se la lunghezza della raffica è di $r+1$, il resto della divisione per $G(x)$ sarà nullo se e solo se la raffica è identica a $G(x)$. Per definizione di raffica, il primo e l'ultimo bit devono essere 1, di conseguenza l'identità dipende dagli $r-1$ bit intermedi. Se tutte le combinazioni vengono considerate equiprobabili, la probabilità che un tale pacchetto errato venga accettato come valido è di $(1/2)^{r-1}$.

Si può anche dimostrare che quando si ha una raffica di errori di lunghezza maggiore a $r+1$, oppure una serie di raffiche più corte, la probabilità che un pacchetto scorretto passi inosservato è di $(1/2)^r$, assumendo che tutte le combinazioni di bit siano equiprobabili.

Questi tre polinomi sono diventati lo standard internazionale:

$$\text{CRC-12} = x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$$

$$\text{CRC-16} = x^{16} + x^{15} + x^2 + 1$$

$$\text{CRC-CCITT} = x^{16} + x^{12} + x^5 + 1$$

Tutti e tre contengono $x+1$ come fattore primo. CRC-12 viene utilizzato quando la lunghezza del carattere è di 6 bit. Gli altri invece per caratteri di 8 bit. Una checksum di 16 bit come CRC-16 oppure CRC-CCITT, riconosce errori singoli e doppi, tutti gli errori con numero dispari di bit, tutte le raffiche di errori di lunghezza minore o uguale a 16, il 99,997 % delle raffiche di lunghezza 17 bit e il 99,998 % delle raffiche di 18 bit o più lunghe.

Nonostante la computazione richiesta per il calcolo della checksum sembra complicata, Peterson e Brown (1961) hanno dimostrato che si può costruire un semplice circuito con registri di shift (spostamento) per calcolare e verificare in hardware la somma di controllo. In pratica tale hardware è utilizzato quasi sempre.

Per decenni si è assunto che i pacchetti su cui si doveva calcolare la checksum contenessero bit casuali. Tutte le analisi degli algoritmi di checksum si sono basate su questa assunzione. Un'analisi più recente sui dati ha però dimostrato che questa assunzione era sbagliata. Come conseguenza, si è notato che sotto particolari circostanze gli errori non

riconosciuti hanno molte più caratteristiche comuni di quanto pensasse (Partridge et al. 1995).

3.3 Protocolli elementari data link

Per introdurre l'argomento protocolli inizieremo considerandone tre con complessità crescente. Per gli interessati è disponibile su WWW un simulatore per tali protocolli (vedi la prefazione). Prima di introdurre i protocolli è necessario fare alcune assunzioni sul modello di comunicazione. Primo, assumeremo che nel livello fisico, e in quelli data link e rete ci siano processi indipendenti che comunicano passandosi messaggi avanti e indietro. In alcuni casi i processi dei livelli fisico e data link gireranno su di un processore in uno speciale processore di I/O di rete e quelli del livello rete sulla CPU principale, ma sono possibili anche altre implementazioni (per esempio tre processi su un singolo processore di I/O; i livelli fisico e data link come procedure chiamate dal processo del livello rete e così via). In ogni caso, trattare i tre livelli come processi separati rende le spiegazioni concettualmente più chiare e aiuta a enfatizzare l'indipendenza dei livelli.

Un'altra assunzione chiave è che una macchina *A* che voglia spedire una lunga sequenza di dati a una macchina *B* usi un servizio affidabile orientato alla connessione. Più avanti considereremo il caso in cui anche *B* voglia spedire dati ad *A* nello stesso momento. Si assuma anche che *A* abbia una risorsa infinita di dati da spedire e che non debba mai aspettare che i dati vengano prodotti. Quando il livello data link di *A* richiede dati, il livello rete è sempre in grado di soddisfare la sua richiesta. (Anche questa restrizione sarà eliminata in seguito).

Per quanto riguarda il livello data link, il pacchetto passatogli attraverso l'interfaccia dal livello rete consiste di puri dati che devono essere riconsegnati al livello rete della destinazione. Il fatto che il livello rete della destinazione possa interpretare parte dei dati come intestazione non riguarda il livello data link.

Quando il livello data link accetta un pacchetto, lo incapsula aggiungendovi un'intestazione (header) e una coda (trailer): si veda la figura 1-11. In questo modo un pacchetto consiste di dati, più alcune informazioni di controllo che si trovano nella intestazione. Il pacchetto viene quindi trasmesso all'altro livello data link. Assumeremo che esistano delle apposite procedure di libreria *to_physical_layer* per inviare un pacchetto e *from_physical_layer* per riceverlo. L'hardware di trasmissione calcola e appende la checksum in modo che il software del livello data link non se ne debba preoccupare. Per esempio, potrebbe essere utilizzato l'algoritmo polinomiale discusso precedentemente.

All'inizio il ricevente non ha nulla da fare: attende che succeda qualcosa. Nei protocolli di esempio di questo capitolo indichiamo che il livello data link del ricevente non ha nulla da fare mediante la procedura *wait_for_event (&event)*. Tale procedura termina quando qualcosa è successo (ad es. è arrivato un pacchetto). Quando la procedura termina la variabile *event* indica cosa è successo. L'insieme di tutti i possibili eventi differisce per i diversi protocolli che descriveremo e verrà definito volta per volta. Da notare che, in situazioni più realistiche, il livello data link non starebbe in attesa attiva di un evento, come abbiamo suggerito, ma riceverebbe un'interruzione che causerebbe la sospensione di tutte le attività che stava svolgendo e inizierebbe a gestire il pacchetto ricevuto. Ad

3.3 Protocolli elementari data link

ogni modo per semplicità ignoreremo i dettagli dell'attività concorrente nel livello data link e assumeremo che esso sia completamente dedicato a gestire a tempo pieno quel- l'unico canale.

Quando un pacchetto arriva al ricevente, l'hardware calcola la checksum. Se è scorretto (cioè se c'è stato un errore di trasmissione) il livello data link ne viene informato (*event=cksum_err*). Anche se il pacchetto arriva integro il livello link viene informato (*event=frame_arrival*), in modo che possa prelevare il pacchetto per ispezionarlo usando *from_physical_layer*. Quando il livello data link del ricevente ha prelevato il pacchetto non danneggiato analizza le informazioni di controllo dell'intestazione e, se tutto è a posto, la parte contenente i dati viene passata al livello rete. Al livello rete non viene mai passata l'intestazione.

C'è una buona ragione per cui il livello rete non riceve mai l'intestazione di un pacchetto: in questo modo infatti i protocolli di livello rete e data link rimangono completamente separati. Poiché il livello rete non sa nulla del protocollo data link o della struttura del pacchetto, questa può essere modificata senza influire sul software del livello rete. Fornendo una rigida interfaccia tra i due livelli si semplifica notevolmente la struttura del software, poiché i protocolli di comunicazione di livelli diversi possono evolvere in modo indipendente.

La figura 3-8 mostra alcune dichiarazioni (in C) comuni a molti protocolli di cui parleremo più avanti. Vengono definite cinque strutture dati: *boolean*, *seq_nr*, *packet*, *frame_kind* e *frame*. Un *boolean* può essere *vero* o *falso*. Un *seq_nr* è un piccolo intero usato per numerare i pacchetti in modo da identificarli. I numeri vanno da 0 a *MAX_SEQ* che è definito in ogni protocollo che lo utilizza. Un *packet* è una unità di informazione scambiata tra il livello rete e il livello data link della stessa macchina o tra livelli rete paritari. Nel nostro modello un *packet* conterrà sempre *MAX_PKT* byte, ma più realisticamente dovrebbe essere di lunghezza variabile.

Un *pacchetto* (*frame*) è composto di quattro campi: *kind*, *seq*, *ack* e *info*; i primi tre contengono informazioni di controllo mentre l'ultimo può contenere i dati veri da trasferire. Questi campi di controllo sono globalmente detti **intestazione del pacchetto (frame header)**. Il campo *field* indica se ci sono dati nel pacchetto, poiché alcuni dei protocolli distinguono i pacchetti contenenti solo informazioni di controllo da quelli contenenti anche i dati. I campi *seq* e *ack* sono usati rispettivamente per i numeri di sequenza e i riscontri (*ack*); descriveremo dettagliatamente in seguito il loro utilizzo. Il campo *info* contiene di solito un singolo pacchetto; il campo *info* di un pacchetto di controllo non viene utilizzato. Una realizzazione più realistica userebbe un campo *info* di lunghezza variabile, omettendolo completamente nei pacchetti di controllo.

È importante capire la relazione tra un pacchetto a livello rete e pacchetto a livello data link (*frame*). Il livello rete costruisce un pacchetto prendendo un messaggio dal livello di trasporto e aggiungendovi un'intestazione. Il pacchetto viene passato al livello data link per essere incluso nel campo *info* di un frame da spedire. Quando il frame giunge a destinazione, il livello data link estrae il pacchetto dal frame e lo passa al livello rete. In questo modo il livello rete può comportarsi come se le due macchine si scambiassero pacchetti direttamente.

Diverse altre procedure sono elencate in figura 3-8: routine di libreria contenenti dettagli

```

#define MAX_PKT 1024           /* determines packet size in bytes */

typedef enum {false, true} boolean;    /* boolean type */
typedef unsigned int seq_nr;          /* sequence or ack numbers */
typedef struct {unsigned char data[MAX_PKT];} packet; /* packet definition */
typedef enum {data, ack, nak} frame_kind; /* frame_kind definition */

typedef struct {                         /* frames are transported in this layer */
    frame_kind kind;                   /* what kind of a frame is it? */
    seq_nr seq;                       /* sequence number */
    seq_nr ack;                      /* acknowledgement number */
    packet info;                     /* the network layer packet */
} frame;

/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);

/* Stop the clock and disable the timeout event. */
void stop_timer(seq_nr k);

/* Start an auxiliary timer and enable the ack_timeout event. */
void start_ack_timer(void);

/* Stop the auxiliary timer and disable the ack_timeout event. */
void stop_ack_timer(void);

/* Allow the network layer to cause a network_layer_ready event. */
void enable_network_layer(void);

/* Forbid the network layer from causing a network_layer_ready event. */
void disable_network_layer(void);

/* Macro inc is expanded in-line: Increment k circularly. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0

```

Fig. 3-8 Alcune definizioni che serviranno nei protocolli del seguito. Queste definizioni sono contenute nel file *protocol.h*.

legati all'implementazione e il cui funzionamento interno non verrà ulteriormente descritto; la procedura *wait_for_event* che resta in attesa che qualcosa accada, come già descritto; le procedure *to_network_layer* e *from_network_layer* che sono utilizzate dal livello data link rispettivamente per passare o accettare pacchetti al/dal livello rete.

Si noti che *from_physical_layer* e *to_physical_layer* vengono usate per passare pacchetti tra il livello data link e il livello fisico, mentre le procedure *to_network_layer* e *from_network_layer* per passare pacchetti tra il livello data link e il livello rete. In altre parole, *to_network_layer* e *from_network_layer* hanno a che fare con l'interfaccia tra il livello 2 e il livello 3, mentre *from_physical_layer* e *to_physical_layer* con l'interfaccia tra il livello 1 e il livello 2.

Nella maggior parte dei protocolli assumiamo di avere un canale non affidabile, che ogni tanto perde interi pacchetti. Per gestire queste situazioni il livello data link del mittente deve inizializzare un timer o un orologio interno ogni volta che invia un pacchetto. Se non riceve una risposta entro un determinato intervallo di tempo, il timer scade e il livello data link riceve un segnale di interruzione.

Nei protocolli considerati questa situazione è gestita permettendo che la procedura *wait_for_event* possa restituire *event = timeout*. Le procedure *start_timer* e *stop_timer* sono rispettivamente usate per inizializzare e arrestare il timer. Si possono avere timeout solo quando il timer è attivo. Viene permesso esplicitamente di invocare *start_timer* quando il timer è attivo; una chiamata di questo tipo azzerà il timer in modo che possa scadere nuovamente dopo un intero intervallo di timer (a meno che non sia riazzzerato o spento nel frattempo).

Le procedure *start_ack_timer* e *stop_ack_timer* vengono utilizzate per gestire un timer ausiliario per la generazione di rete sotto certe condizioni.

Le procedure *enable_network_layer* e *disable_network_layer* sono utilizzate nei protocolli più sofisticati dove non si assume che il livello rete abbia sempre dei pacchetti da spedire. Quando il livello data link abilita il livello rete, questo ha il permesso di interromperlo quando ha un pacchetto da spedire. Indichiamo questo evento restituendo *event = network_layer_ready*. Quando il livello rete è disabilitato non può causare questo evento. Prestando attenzione a quando abilitare o disabilitare il livello rete, il livello data link evita di essere travolto dai pacchetti del livello rete quando non ha più buffer per essi.

I numeri di sequenza dei pacchetti sono sempre compresi tra 0 e *MAX_SEQ* (incluso), dove *MAX_SEQ* è diverso a seconda dei diversi protocolli. È spesso necessario incrementare i numeri di sequenza in modo circolare (cioè *MAX_SEQ* è seguito da 0). La macro *inc* effettua tale incremento. Viene definita come macro poiché viene utilizzata "in linea" durante il percorso critico. Come meglio avremo modo di vedere in seguito, il fattore limitante delle prestazioni della rete è spesso la struttura del protocollo, quindi la definizione come macro di operazioni semplici come queste non influenza sulla leggibilità del codice ma migliora le prestazioni. Inoltre, poiché *MAX_SEQ* assume diversi valori a seconda dei protocolli, costruendo una macro è possibile includere tutti i protocolli in uno stesso file binario senza conflitti. Questa possibilità è utile al simulatore.

Le dichiarazioni della figura 3-8 sono parte di ognuno dei protocolli che seguono. Per risparmiare spazio e fornire un riferimento utile, esse sono state estratte ed elencate tutte insieme, ma concettualmente dovrebbero essere associate ai protocolli. In C questa asso-

ciazione viene fatta mettendo le definizioni in uno speciale file di intestazione, in questo caso *protocol.h*, e usando `#include` che ordina al preprocessore C di includere l'intestazione nei file dei protocolli.

3.3.1 Un protocollo simplex non limitato

Come primo esempio consideriamo un protocollo molto semplice. I dati vengono trasmessi in una sola direzione. I livelli di rete sia in trasmissione che in ricezione sono sempre pronti entrambi. Il tempo di elaborazione si può ignorare. È disponibile infinito spazio di buffer. Meglio di tutto, il canale di comunicazione tra i livelli data link non perde nessun frame, né lo danneggia. Questo protocollo del tutto irrealistico che chiameremo "utopia" è mostrato in figura 3-9.

Il protocollo è composto di due procedure distinte, un mittente e un ricevente. Il mittente gira nel livello data link della macchina origine e il ricevente gira nel livello data link della macchina destinazione. Non vengono utilizzati né numeri di sequenza né riscontri, quindi *MAX_SEQ* non serve. Il solo tipo di evento possibile è *frame_arrival* (cioè l'arrivo di un frame non danneggiato).

Il mittente è composto da un ciclo infinito che spedisce i dati sulla linea più velocemente possibile. Il corpo del ciclo consiste di tre azioni: prendere un pacchetto dal livello rete (sempre pronto), costruire un frame illimitato usando la variabile *s*, e spedire il frame per la sua strada. In questo protocollo viene utilizzato solo il campo *info* poiché gli altri campi hanno a che vedere con il controllo degli errori e del flusso e qui non si hanno né controllo di errori né di flusso.

Il ricevente è altrettanto semplice. Per prima cosa esso aspetta che avvenga qualcosa, l'unica possibilità è che possa arrivare un frame non danneggiato. Prima o poi il frame arriva e la procedura *wait_for_event* termina con *event = frame_arrival* (che viene comunque ignorato). La chiamata a *from_physical_layer* preleva il frame appena ricevuto dal buffer hardware e lo pone nella variabile *r*. Infine, la porzione di dati viene passata al livello rete e il livello data link ritorna ad aspettare un altro frame, in pratica sospendendosi fino al suo arrivo.

3.3.2 Un protocollo simplex stop-and-wait

Ora eliminiamo la limitazione meno realistica introdotta nel primo protocollo: la capacità del livello rete del ricevente di elaborare dati in ingresso in modo infinitamente veloce (o, equivalentemente, la presenza nel livello data link ricevente di una quantità infinita di spazio di buffer in cui si memorizzano tutti i frame in arrivo in attesa del loro turno di elaborazione). Il canale di comunicazione è tuttavia ancora esente da errori e il traffico di dati simplex.

Il problema principale è quello di impedire al mittente di travolgere il ricevente con dati che arrivano più velocemente di quanto quest'ultimo riesca a elaborarli. In sostanza, se il ricevente ha bisogno di un tempo Δt per eseguire *from_physical_layer* e *to_network_layer*, il mittente dovrà trasmettere a una velocità minore di un pacchetto per Δt . Di più, se assumiamo che non si abbiano buffer e accodamento automatico nell'hardware del ricevente, il mittente non dovrà trasmettere un nuovo pacchetto finché il precedente non sia stato caricato da *from_physical_layer*, affinché uno nuovo non sovrascriva il vecchio.

3.3 Protocolli elementari data link

```
/* Protocol 1 (utopia) provides for data transmission in one direction only, from
   sender to receiver. The communication channel is assumed to be error free,
   and the receiver is assumed to be able to process all the input infinitely fast.
   Consequently, the sender just sits in a loop pumping data out onto the line as
   fast as it can. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
    frame s;                                /* buffer for an outbound frame */
    packet buffer;                           /* buffer for an outbound packet */

    while (true) {
        from_network_layer(&buffer); /* go get something to send */
        s.info = buffer;                /* copy it into s for transmission */
        to_physical_layer(&s);         /* send it on its way */
    }                                         /* Tomorrow, and tomorrow, and tomorrow,
                                                Creeps in this petty pace from day to day
                                                To the last syllable of recorded time
                                                - Macbeth, V, v */
}

void receiver1(vcld)
{
    frame r;                                /* filled in by wait, but not used here */
    event_type event;

    while (true) {
        wait_for_event(&event);           /* only possibility is frame_arrival */
        from_physical_layer(&r);          /* go get the inbound frame */
        to_network_layer(&r.info);        /* pass the data to the network layer */
    }
}
```

Fig. 3-9 Un protocollo simplex non limitato.

In determinate circostanze (per esempio in caso di trasmissione sincrona e livello data link completamente dedicato a una linea di input), potrebbe essere sufficiente inserire nel protocollo 1 un ritardo del mittente in modo da impedirgli di travolgere il ricevente. Di solito tuttavia il livello data link dovrà gestire più di una linea di input e l'intervallo di tempo tra l'arrivo di un pacchetto e l'inizio della sua elaborazione può variare considerevolmente. Se i progettisti di rete possono calcolare il comportamento del ricevente nel caso peggiore, essi riescono a programmare il mittente in modo che anche se il

```

/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from
   sender to receiver. The communication channel is once again assumed to be error
   free, as in protocol 1. However, this time, the receiver has only a finite buffer
   capacity and a finite processing speed, so the protocol must explicitly prevent
   the sender from flooding the receiver with data faster than it can be handled. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
    frame s;           /* buffer for an outbound frame */
    packet buffer;     /* buffer for an outbound packet */
    event_type event;  /* frame_arrival is the only possibility */

    while (true) {
        from_network_layer(&buffer); /* go get something to send */
        s.info = buffer;             /* copy it into s for transmission */
        to_physical_layer(&s);      /* bye bye little frame */
        wait_for_event(&event);     /* do not proceed until given the go ahead */
    }
}

void receiver2(void)
{
    frame r, s;           /* buffers for frames */
    event_type event;     /* frame_arrival is the only possibility */

    while (true) {
        wait_for_event(&event);   /* only possibility is frame_arrival */
        from_physical_layer(&r); /* go get the inbound frame */
        to_network_layer(&r.info); /* pass the data to the network layer */
        to_physical_layer(&s);   /* send a dummy frame to awaken sender */
    }
}

```

Fig. 3-10 Un protocollo simplex stop-and-wait.

pacchetto subisce il più grosso ritardo non si abbiano problemi. Il guaio con questo approccio è che risulta troppo conservativo, infatti impone un uso della banda che è molto sotto l'ottimo a meno che i casi migliore e peggiore siano molto vicini (cioè la variazione nei tempi di reazione del livello data link sia piccola).

Una soluzione più generale a questo problema è quella di fare in modo che il ricevente restituisca qualche informazione al mittente. Dopo aver passato il pacchetto al livello rete, il ricevente spedisce un piccolo frame fittizio al mittente dandogli praticamente il per-

messo di trasmettere il frame successivo. Dopo avere spedito il frame, il mittente è obbligato dal protocollo ad aspettare finché non arriva il frame fittizio (cioè un riscontro). I protocolli in cui il mittente spedisce un frame e poi aspetta un riscontro prima di procedere sono detti **stop-and-wait**. La figura 3-10 mostra un esempio di un protocollo simplex stop-and-wait.

Come nel protocollo 1, il mittente inizia catturando un pacchetto dal livello rete, impiegandolo per costruire un frame e spedendolo. Quindi, diversamente dal protocollo 1, il mittente aspetta finché non arriva un frame di riscontro, prima di caricare il nuovo pacchetto dal livello rete. Il livello data link del mittente non deve controllare il frame in arrivo in quanto c'è solo una possibilità.

L'unica differenza tra *receiver1* e *receiver2* è che dopo aver consegnato un pacchetto al livello rete, *receiver2* manda un frame di riscontro al mittente prima di ricominciare il suo ciclo. Poiché l'unica cosa importante è l'arrivo del frame al mittente, non il suo contenuto, il ricevente non deve inserirvi particolari informazioni.

Sebbene il traffico di dati dell'esempio sia simplex, diretto dal mittente al ricevente, i frame viaggiano in entrambe le direzioni. Di conseguenza, il canale di comunicazione tra i due livelli data link deve essere in grado di trasferire informazione in entrambe le direzioni. Tuttavia, questo protocollo implica una stretta alternanza di flusso: prima il mittente spedisce un frame, poi tocca al ricevente, quindi al mittente ancora e al ricevente di seguito e così via. In questo caso sarebbe quindi sufficiente un canale fisico half-duplex.

3.3.3 Un protocollo simplex per un canale disturbato

Consideriamo ora una situazione normale di un canale di comunicazione che non è esente da errori. I frame possono danneggiarsi oppure andare completamente perduti. Tuttavia, assumiamo che se un frame viene danneggiato durante il transito, l'hardware del ricevente sa riconoscere l'errore calcolando la checksum. Se il frame venisse danneggiato in modo che la checksum fosse comunque corretta, evento molto improbabile, questo protocollo (ma anche tutti gli altri protocolli) potrebbe fallire (cioè consegnare un pacchetto danneggiato al livello rete).

A prima vista potrebbe sembrare che basti aggiungere un timer nel protocollo 2. Il mittente potrebbe spedire un frame, mentre il ricevente a sua volta invierebbe un riscontro solo nel caso in cui i dati fossero stati ricevuti correttamente. Se venisse consegnato un frame danneggiato il ricevente lo scarterebbe. Dopo un po' di tempo il timer del mittente scadrebbe ed esso invierebbe di nuovo il frame. Questo processo si ripeterebbe finché il frame non arrivasse corretto.

La tecnica sopradescritta presenta un grave difetto. Prima di leggere il seguito si provi a riflettere cercando di individuare cosa non va.

Per capire qual è il problema, occorre ricordare il compito del livello data link al fine di garantire l'assenza di errori, cioè la comunicazione trasparente tra i processi dei livelli rete. Il livello rete della macchina A passa una serie di pacchetti al livello data link sottostante; questo deve assicurare che una serie identica di pacchetti venga consegnata al livello rete della macchina B dal livello data link residente su quella macchina. In particolare il livello rete di B non ha modo di conoscere se un pacchetto viene perso o duplicato, quindi il livello data link deve garantire che nessuna combinazione di errori di

trasmissione, non importa quanto improbabili siano, possa causare la duplicazione di un pacchetto da consegnare al livello rete.

Si consideri la seguente situazione:

- Il livello rete su *A* passa il pacchetto 1 al livello data link. Il pacchetto viene ricevuto correttamente su *B* e passato al livello rete. *B* invia un frame di riscontro ad *A*.
- Il frame di riscontro viene totalmente perso; esso non arriva alla destinazione. La vita sarebbe molto più semplice se il canale danneggiasse o perdesse solo frame di dati e non frame di controllo, ma è triste riscontrare che il canale non è così discriminante.
- Il timer del livello data link di *A* a un certo punto scade. Non avendo ricevuto il frame di riscontro assume (scorrettamente) che il frame di dati che aveva spedito sia stato danneggiato o perso, così rispedisce il frame contenente il pacchetto 1.
- Il frame duplicato arriva anch'esso intatto al livello data link di *B* e viene inconsciamente passato al livello rete. Se *A* stesse spedendo un file a *B*, parte del file verrebbe duplicata (cioè la copia del file fatta da *B* sarebbe scorretta e l'errore non sarebbe riconosciuto). In altre parole, il protocollo fallirebbe.

Chiaramente è necessario che il ricevente sia in qualche modo in grado di distinguere un frame mai visto da uno ritrasmesso. Il modo più ovvio di ottenere ciò è quello di fare in modo che il mittente numeri progressivamente le intestazioni dei frame che spedisce. Quindi il ricevente potrà controllare i numeri dei frame che arrivano per distinguere i nuovi frame da quelli ritrasmessi.

Poiché è desiderabile avere una intestazione di frame piccola, sorge una questione: qual è il minimo numero di bit necessario per i numeri progressivi? L'unica ambiguità in questo protocollo è tra il frame *m* e il suo diretto successore *m+1*. Se il frame *m* viene danneggiato o perso, il ricevente non invierà il riscontro e il mittente proverà a rispedire il frame. Una volta ricevuto correttamente il frame, il ricevente invierà il frame di rete al mittente. E qui può sorgere il problema. A seconda che il frame di rete venga o non venga ricevuto dal mittente, questo spedirà il frame *m* o *m+1*.

L'evento che spinge il mittente a inviare il frame *m+2* è l'arrivo di un frame di ack per *m+1*. Ma questo implica che *m* sia stato ricevuto correttamente, e inoltre che il suo ack sia stato correttamente ricevuto dal mittente (altrimenti il mittente non avrebbe iniziato a gestire *m+1* e tantomeno *m+2*). Di conseguenza, l'unica ambiguità è tra un frame e il suo predecessore o successore, non tra il predecessore e successore stessi.

Un contatore di un bit (0 o 1) è quindi sufficiente. In ogni istante, il ricevente aspetta un determinato numero progressivo. Ogni frame in arrivo che contenga il numero sbagliato viene rifiutato essendo un duplicato. Quando arriva un frame con il corretto numero progressivo, viene accettato, passato al livello rete e il numero atteso viene incrementato modulo 2 (cioè 0 diventa 1 e 1 diventa 0).

Un esempio di questo tipo di protocollo è mostrato in figura 3-11. I protocolli in cui il mittente aspetta un ack positivo prima di gestire l'elemento successivo sono spesso detti PAR (Positive Acknowledgement with Retransmission – ack positivo con ri-

```

/* Protocol 3 (par) allows unidirectional data flow over an unreliable channel. */

#define MAX_SEQ 1           /* must be 1 for protocol 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
    seq_nr next_frame_to_send; /* seq number of next outgoing frame */
    frame s;                  /* scratch variable */
    packet buffer;            /* buffer for an outbound packet */
    event_type event;

    next_frame_to_send = 0;    /* initialize outbound sequence numbers */
    from_network_layer(&buffer); /* fetch first packet */

    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_network_layer(&buffer); /* get the next one to send */
            inc(next_frame_to_send);   /* invert 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) { /* possibilities: frame_arrival, cksum_err */
            /* a valid frame has arrived. */
            from_physical_layer(&r);
            if (r.seq == frame_expected) { /* this is what we have been waiting for. */
                to_network_layer(&r.info); /* pass the data to the network layer */
                inc(frame_expected);    /* next time expect the other sequence nr */
            }
            to_physical_layer(&s);      /* none of the fields are used */
        }
    }
}

```

Fig 3-11 Un protocollo con riscontro positivo con ritrasmessione.

trasmissione) oppure ARQ (Automatic Repeat reQuest – richiesta di ripetizione automatica). Come il protocollo 2, questo trasmette dati in una sola direzione. Sebbene riesca a gestire la perdita di frame (con un timer che scade), ha bisogno che i tempi di scadenza del timer siano piuttosto larghi per prevenire timeout prematuri. Infatti, se il timer del mittente scade troppo presto, e il frame di ack è ancora in viaggio, ritrasmetterà il frame di dati.

Quando il riscontro precedente arriverà, il mittente crederà, sbagliando, che sia il frame in risposta a quello appena spedito e non realizzerà che c'è un altro frame di ack per strada. Se il secondo frame spedito viene perso ma l'ack extra arriva correttamente il mittente non farà la ritrasmissione e il protocollo fallirà. Nel protocollo successivo i frame di ack conterranno informazioni atte a prevenire questo tipo di problemi. Per il momento i frame di ack sono solo finti e assumeremo una stretta alternanza di mittente e ricevente. Il protocollo 3 si distingue dai precedenti nel fatto che sia il mittente che il ricevente hanno una variabile il cui valore resta memorizzato anche mentre il livello data link è in attesa. Il mittente tiene in memoria il numero progressivo del prossimo frame da spedire in *next_frame_to_send*; il ricevente memorizza invece il numero del prossimo frame atteso in *frame_expected*. Ogni protocollo ha una breve fase di inizializzazione prima di iniziare il ciclo infinito.

Dopo aver trasmesso un frame, il mittente inizializza il timer. Se esso era già in azione viene resettato per avere a disposizione un altro intero intervallo di tempo. L'intervallo deve essere scelto in modo che consenta al frame di raggiungere la destinazione, di essere ivi elaborato e di permettere al frame di rete di ritornare al mittente. Solo quando l'intervallo è scaduto è corretto assumere che il frame di dati o quello di rete è stato perso e spedire un duplicato.

Dopo aver trasmesso un frame e inizializzato il timer il mittente aspetta che accada qualcosa di interessante. Ci sono tre possibilità: arriva un frame di rete non danneggiato, arriva un frame di rete danneggiato oppure scade il timer. Nel primo caso il mittente carica il successivo pacchetto dal livello rete e lo mette nel buffer riscrivendo il pacchetto precedente. Incrementa inoltre il numero progressivo. Se invece arriva un frame danneggiato oppure non arriva nulla entro l'intervallo di tempo, non vengono modificati né il buffer né il numero progressivo in modo da essere in grado di spedire un duplicato.

Quando un frame valido arriva al ricevente, viene controllato il numero progressivo per controllare se è un duplicato. Se non lo è esso viene accettato, passato al livello rete e viene generato un ack. I frame duplicati o danneggiati non vengono ovviamente passati al livello rete.

3.4 I protocolli a finestra scorrevole (sliding window)

Nei precedenti protocolli i frame di dati venivano trasmessi in una sola direzione. Nelle situazioni più comuni occorre trasmettere dati in entrambe le direzioni. Un modo per raggiungere la trasmissione duplex dei dati è quella di avere due canali di comunicazione separati e usarli distintamente per traffico simplex (in direzioni opposte). In questo caso, si avranno due circuiti fisici separati, uno per i dati che viaggiano in avanti e l'altro in senso opposto per gli ack. In entrambi i casi la larghezza di banda del canale inverso è

quasi completamente sprecata. Effettivamente l'utente paga il costo di due circuiti per usare alla fine la capacità di uno solo.

Un'idea migliore è quella di usare lo stesso circuito per i dati in entrambe le direzioni. Dopo tutto, nei protocolli 2 e 3 questo sistema è già stato usato per trasmettere frame avanti e indietro e il canale inverso ha in realtà la medesima capacità dell'altro. In questo modello i frame di dati da A a B sono intervallati da frame di ack da A a B. Considerando il campo *kind* dell'intestazione del frame arrivato, il ricevente sa dire se il frame contenga dati o un ack.

Poiché alternare frame di dati e di ack sullo stesso circuito non è altro che un miglioramento rispetto ad avere due circuiti fisici separati, è possibile un'altra miglioria. Quando arriva un frame di dati, invece di inviare immediatamente un frame di controllo, il ricevente attende finché il livello rete non passa un altro pacchetto. L'ack viene attaccato al frame di dati da spedire (usando il campo *ack* nell'intestazione del frame). In questo modo l'ack viaggia, per così dire, gratis, sfruttando la spedizione di un altro frame. La tecnica di ritardare gli ack in modo che possano essere attaccati ad altri frame di dati in uscita si chiama **piggybacking**.

Il principale vantaggio dell'uso del piggybacking rispetto all'avere riscontri distinti è quello di ottenere un miglior uso della larghezza di banda del canale. Il campo *ack* nell'intestazione del frame costa solo pochi bit, mentre un frame intero necessiterebbe di un'intestazione, di un ack e di una somma di controllo. Inoltre, meno frame spediti significa meno interruzioni di frame in arrivo e forse anche meno buffer nel ricevente, a seconda di come è organizzato il suo software. Nel prossimo protocollo il campo di piggybacking costa solo un bit nell'intestazione del frame; raramente comunque esso richiede più di qualche bit.

Il piggybacking introduce tuttavia una complicazione che non era presente con ack distinti. Quanto deve aspettare il livello data link un pacchetto a cui appendere un ack? Se il livello data link aspetta più del tempo atteso dal mittente per l'ack, il frame dati verrà rispedito, facendo venire a mancare l'utilità degli ack. Se il livello data link sapesse prevedere il futuro, conoscerebbe il momento di arrivo di un pacchetto dal livello rete e potrebbe decidere se aspettarlo o se spedire un ack separatamente, a seconda del tempo necessario per attendere il pacchetto. Ovviamente il livello data link non è in grado di prevedere il futuro, quindi deve escogitare qualche altro metodo, come per esempio aspettare un determinato numero di millisecondi. Se un nuovo pacchetto arriva velocemente, l'ack viene agganciato ad esso; altrimenti, se nessun pacchetto arriva entro la fine dell'intervallo di tempo, il livello data link spedisce un frame di ack separatamente.

Oltre ad essere solo simplex, il protocollo 3 può fallire sotto certe precise condizioni che coinvolgono lo scadere prematuro del timer. Sarebbe meglio avere un protocollo che rimanesse sincronizzato indipendentemente da ogni tipo di combinazione di frame danneggiati, persi e di timeout prematuri. I seguenti tre protocolli sono più robusti e resistono anche sotto condizioni patologiche. Appartengono tutti e tre a una classe di protocolli detti protocolli **sliding window** (a finestra scorrevole). Differiscono tra loro in termini di efficienza, complessità ed esigenze di buffer, come spiegheremo in seguito.

In tutti i protocolli sliding window, ogni frame spedito contiene un numero progressivo, che varia da 0 a un massimo. Il massimo è di solito $2^n - 1$ in modo che il numero

progressivo sia contenuto in un campo di n bit. Il protocollo sliding window stop-and-wait usa $n=1$, restringendo i numeri progressivi a 0 e 1, ma versioni più sofisticate possono usare un n arbitrario.

L'essenza di tutti i protocolli sliding window è che a ogni istante, il mittente mantiene un insieme di numeri progressivi corrispondenti ai frame che esso può spedire. Si dice che questi frame ricadono nella **finestra di trasmissione (sending window)**. In modo simile, il ricevente mantiene una **finestra di ricezione (receiving window)** corrispondente all'insieme dei frame che esso può accettare. La finestra del mittente e quella del ricevente non sono obbligate ad avere gli stessi limiti inferiore e superiore, e nemmeno la stessa dimensione. In alcuni protocolli esse sono di dimensione fissa, ma in altri esse possono crescere o diminuire quando i frame vengono spediti o ricevuti.

Sebbene questi protocolli diano al livello data link maggiore libertà riguardo all'ordine di spedizione e ricevimento dei frame, non si è eliminato il requisito che il protocollo debba consegnare i pacchetti al livello rete della destinazione nello stesso ordine in cui essi sono stati passati al livello data link sulla macchina mittente. Non si è nemmeno modificato il requisito che il canale di comunicazione fisico sia un filo conduttore, cioè che debba consegnare i frame nell'ordine di spedizione.

I numeri progressivi della finestra del mittente rappresentano frame spediti ma di cui non si è ancora ricevuto un ack. Quando arriva un nuovo pacchetto dal livello rete, ad esso viene assegnato il maggior numero progressivo disponibile e l'estremità superiore della finestra è aumentata di uno. Quando arriva un ack, il lato inferiore della finestra viene aumentata di uno; in questo modo la finestra mantiene continuamente una lista dei frame di cui non è ancora arrivato un ack.

Poiché i frame che sono indicati nella finestra del mittente possono andare persi o essere danneggiati nel transito, il mittente li mantiene tutti in memoria per possibili ritrasmissioni. Così se la dimensione massima della finestra è n , il mittente ha bisogno di n buffer per memorizzare i frame senza ack. Se la finestra raggiunge la sua massima dimensione, il livello data link del mittente deve temporaneamente proibire al livello rete di passare pacchetti fino a quando non si libera un buffer.

La finestra del livello data link del ricevente corrisponde ai frame che esso può accettare. Ogni frame che non cade nella finestra viene scartato senza indugio. Quando viene ricevuto un frame il cui numero progressivo corrisponde all'estremità inferiore della finestra, esso viene passato al livello rete, viene generato un ack e la finestra viene ruotata di uno. Diversamente dalla finestra del mittente, quella del ricevente rimane sempre di dimensione uguale a quella iniziale. Da notare che una finestra di dimensione 1 significa che il livello data link accetta solo frame ordinati, ma per finestre più grandi questo non vale. Il livello rete, al contrario riceve sempre i dati nell'ordine corretto, indipendentemente dalla dimensione della finestra.

La figura 3-12 mostra un esempio dove la dimensione massima della finestra è 1. Inizialmente nessun frame è in sospeso quindi le estremità inferiore e superiore sono uguali, ma nel tempo, la situazione progredisce come indicato.

3.4.1 Un protocollo sliding window di un solo bit

Prima di trattare il caso generale, esaminiamo un protocollo sliding window con finestre

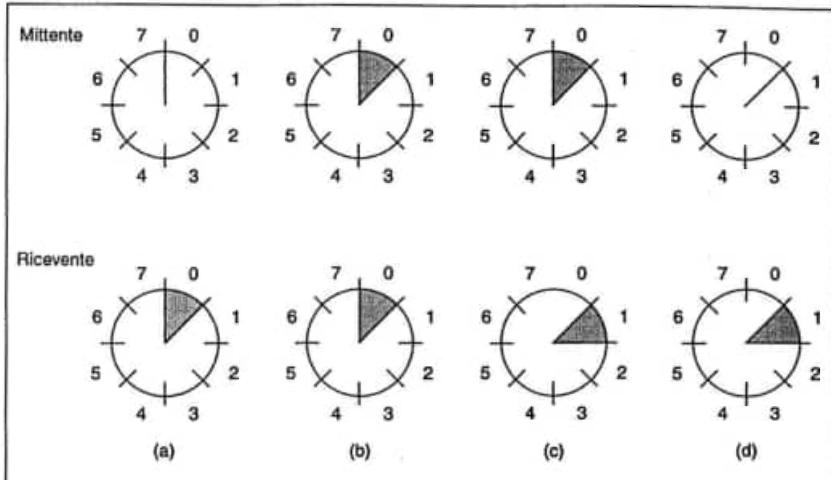


Fig 3-12 Una sliding window di dimensione 1 con numeri progressivi di 3 bit. (a) All'inizio. (b) Dopo la spedizione del primo frame. (c) Dopo la ricezione del primo frame. (d) Dopo la ricezione del primo ack.

di dimensione massima di un solo bit. Questi protocolli utilizzano stop-and-wait, infatti il mittente trasmette un frame e aspetta l'ack prima di spedirne un altro.

La figura 3-13 descrive questo protocollo. Esso inizia, come gli altri, definendo alcune variabili. *Next_frame_to_send* indica quale frame il mittente sta cercando di spedire. In modo simmetrico, *frame_expected* indica quale frame il ricevente sta aspettando. In entrambi i casi le sole possibilità sono 0 o 1.

Generalmente uno dei due livelli data link procede per primo. In altre parole, solo uno dei programmi dei due livelli data link dovrebbe contenere chiamate alle procedure *to_physical_layer* e *start_timer* fuori del ciclo principale. Se i due livelli data link iniziassero insieme, si avrebbe una situazione particolare che discuteremo in seguito. La macchina che parte per prima carica il primo pacchetto dal livello rete, costruisce un frame con esso e lo spedisce. Quando questo (o un altro) frame arriva, il livello data link del ricevente controlla se è un duplice come nel protocollo 3. Se il frame è quello atteso, viene passato al livello rete e la finestra del ricevente viene fatta slittare avanti.

Il campo di ack contiene il numero dell'ultimo frame ricevuto senza errori. Se il numero coincide con quello del frame che il mittente sta cercando di spedire, il mittente capisce di avere terminato la spedizione del frame memorizzato nel suo buffer e può caricare un nuovo pacchetto dal livello rete. Se invece il numero progressivo non coincide deve continuare a rispedire lo stesso frame. Ogni volta che un frame viene ricevuto un altro frame viene spedito indietro.

Esaminiamo ora il protocollo 4 al fine di constatare quanto esso sia robusto nel caso di scenari patologici. Supponiamo che A stia provando a inviare il suo frame 0 a B e che B stia provando

```

/* Protocol 4 (sliding window) is bidirectional and is more robust than protocol 3. */
#define MAX_SEQ 1           /* must be 1 for protocol 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;                      /* scratch variables */
    packet buffer;                   /* current packet being sent */

    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);          /* frame_arrival, cksum_err, or timeout */
        if (event == frame_arrival) { /* a frame has arrived undamaged. */
            from_physical_layer(&r);   /* go get it */

            if (r.seq == frame_expected) {
                /* Handle inbound frame stream. */
                to_network_layer(&r.info); /* pass packet to network layer */
                inc(frame_expected);     /* invert sequence number expected next */
            }

            if (r.ack == next_frame_to_send) { /* handle outbound frame stream. */
                from_network_layer(&buffer); /* fetch new pkt from network layer */
                inc(next_frame_to_send); /* invert sender's sequence number */
            }
        }

        s.info = buffer;                 /* construct outbound frame */
        s.seq = next_frame_to_send;      /* insert sequence number into it */
        s.ack = 1 - frame_expected;    /* seq number of last received frame */
        to_physical_layer(&s);         /* transmit a frame */
        start_timer(s.seq);
    }
}

```

Fig 3-13 Un protocollo sliding window di un solo bit.

a spedire il suo frame 0 ad A. Supponiamo anche che A invii un frame a B ma che l'intervallo del timer di A sia un po' troppo corto. Di conseguenza, il timer di A potrebbe scadere ripetutamente e spedire una serie di frame identici, tutti con $seq=0$ e $ack=1$.

Quando il primo frame valido arriva a B, esso verrà accettato e *frame_expected* sarà posto a 1. Tutti i frame seguenti verranno rifiutati poiché B adesso starà aspettando un frame con numero progressivo 1 e non 0. Inoltre, poiché tutti i duplicati avranno $ack=1$ e B starà ancora aspettando un ack per 0, B non caricherà un nuovo pacchetto dal livello rete. Dopo l'arrivo di ogni duplicato che verrà rifiutato, B spedirà ad A un frame contenente $seq=0$ e $ack=0$. Prima o poi uno di questi frame arriverà correttamente ad A, provocando la spedizione di un nuovo pacchetto da parte di questo. Nessuna combinazione di frame persi o di timer scaduti prematuramente potrà fare in modo che il protocollo consegna pacchetti duplicati ai due livelli rete, oppure che esso salti un pacchetto o, ancora, che entri in una situazione di stallo.

Tuttavia, sorge una situazione particolare se entrambe le parti mandano simultaneamente un pacchetto iniziale. Questa difficoltà di sincronizzazione viene illustrata in figura 3-14. Nel lato (a), viene illustrata l'operazione normale del protocollo. In (b) il caso particolare. Se B aspetta il primo frame di A prima di inviare uno dei suoi, la sequenza è come quella mostrata in (a) e ogni frame viene accettato. Se invece A e B iniziano la comunicazione simultaneamente, i loro primi frame si incrociano e i livelli data link si trovano in situazione (b). In (a) l'arrivo di un frame comporta il passaggio di un pacchetto al livello rete; non ci sono duplicati. In (b) metà dei frame sono duplicati sebbene non avvengano errori di trasmissione. Situazioni simili possono verificarsi come risultato di timer scaduti prematuramente anche quando una parte sia partita per prima. Infatti, se si verifica che il timer scada più volte di seguito, i frame potranno essere ritrasmessi tre o più volte.

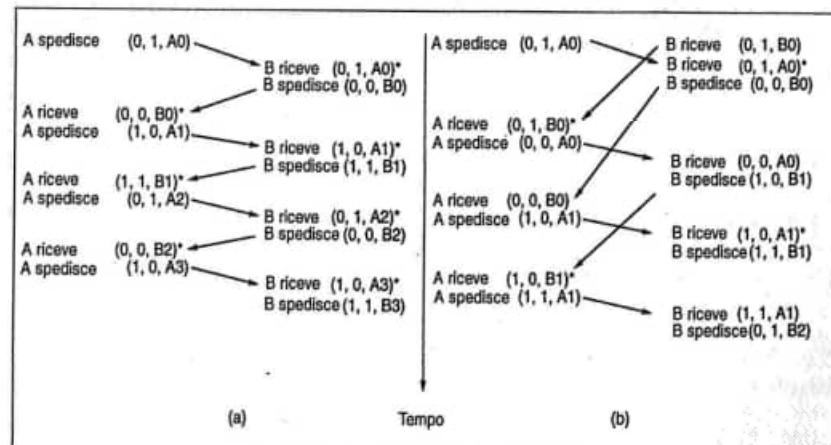


Fig 3-14 Due scenari per il protocollo 4. La notazione è (seq, ack, packet, number). L'asterisco indica l'accettazione del pacchetto da parte del livello rete.

3.4.2 Un protocollo go back n

Fino ad ora abbiamo tacitamente assunto che il tempo richiesto per la trasmissione di un frame e del suo ack fosse trascurabile. Qualche volta però questa assunzione è falsa. In questi casi il tempo impiegato per l'intero ciclo può avere importanti implicazioni sull'efficienza di utilizzazione della larghezza di banda. Ad esempio consideriamo un canale via satellite di 50 bps con un ritardo di propagazione sull'intero ciclo di 500 ms. Immaginiamo di provare a utilizzare il protocollo 4 per inviare frame di 1000 bit via satellite. Al tempo $t=0$ il mittente inizia a spedire il primo frame. Al tempo $t=20$ ms la spedizione del frame è terminata. Nelle migliori circostanze (niente attesa del ricevente e un frame di ack breve), il frame viene ricevuto non prima del tempo $t=270$ ms e l'ack arriva al mittente non prima del tempo $t=520$ ms. Questo significa che il mittente resta bloccato per 500/520 del tempo (96%): cioè viene utilizzata solo per il 4% del tempo la larghezza di banda disponibile. Chiaramente la combinazione di un lungo tempo di trasmissione, un'ampia larghezza di banda e una breve lunghezza dei frame è disastrosa in termini di efficienza.

Il problema appena descritto può essere visto come una conseguenza della regola che richiede che il mittente attenda un ack prima di spedire un altro frame. Eliminando questa restrizione, potremo ottenere una migliore efficienza. Fondamentalmente la soluzione consiste nel permettere al mittente di trasmettere fino a w frame prima di bloccarsi, invece di solo 1. Con una scelta appropriata di w il mittente sarà in grado di trasmettere frame di continuo per un tempo uguale a quello speso per un intero ciclo senza riempire la finestra. Nell'esempio precedente, w dovrebbe essere almeno 26. Il mittente inizia a inviare il frame 0 come sopra. Prima che egli abbia finito di spedire il frame 26, al tempo $t=520$ ms, l'ack del frame 0 sarà appena arrivato. Dopodiché, gli ack arriveranno ogni 20 msec, così il mittente avrà sempre il permesso di continuare quando ne ha bisogno. In ogni momento, ci saranno 25 o 26 frame di cui non si è ancora ricevuto l'ack. Detto in altri termini, la dimensione massima della finestra del mittente è 26.

Questa tecnica è detta **pipelining**. Se la capacità del canale è di b bit/s, la dimensione del frame di l bit e il tempo di propagazione del ciclo è di R s, il tempo impiegato per trasmettere un singolo frame è di l/b . Dopo che l'ultimo bit del frame di dati è stato trasmesso, ci sarà un ritardo di $R/2$ prima che il bit arrivi al ricevente e un altro ritardo di minimo $R/2$ perché l'ack torni indietro, per un ritardo totale di R . Usando stop-and-wait la linea è impegnata per l/b e libera per R , portando a un utilizzo della linea di $l/(l+bR)$. Se $l < bR$ l'efficienza sarà meno del 50%. Poiché si ha sempre un ritardo non nullo perché l'ack venga propagato indietro, in teoria il pipelining può essere utilizzato per tenere la linea impegnata durante questo intervallo, ma se l'intervallo è breve, la complessità aggiuntiva non vale il guadagno.

Il pipelining di frame su di un canale di comunicazione inaffidabile fa sorgere alcune domande. Primo, cosa succede se un frame nel mezzo di una lunga sequenza viene danneggiato? Un gran numero di frame successivi arriveranno al ricevente prima che il mittente si accorga dell'accaduto. Quando un frame danneggiato arriva al ricevente, viene ovviamente scartato, ma che cosa dovrebbe fare il ricevente di tutti i frame successivi corretti? Ricordiamo che il livello data link del ricevente è obbligato a consegnare pacchetti al livello rete in modo ordinato.

3.4 I protocolli a finestra scorrevole (sliding window)

Esistono due approcci fondamentali per trattare errori in presenza di pipelining. Il primo, chiamato **go back n** (indietreggiamento n), consiste nel fare in modo che il ricevente scarti tutti i pacchetti successivi, non spedendo alcun ack per quelli scartati. Questa strategia corrisponde a una finestra ricevente di dimensione 1. In altre parole il livello data link si rifiuta di accettare pacchetti, eccetto quello che, in sequenza, deve passare al livello rete. Se la finestra del mittente si riempie prima che il timer scada, il pipeline inizierà a vuotarsi. Prima o poi il timer del mittente scadrà ed esso ritrasmetterà nell'ordine tutti i pacchetti di cui non ha ricevuto ack, iniziando da quello danneggiato o perso. Questo approccio, descritto in figura 3-15 (a) può sprecare molta larghezza di banda se il tasso di errore è elevato.

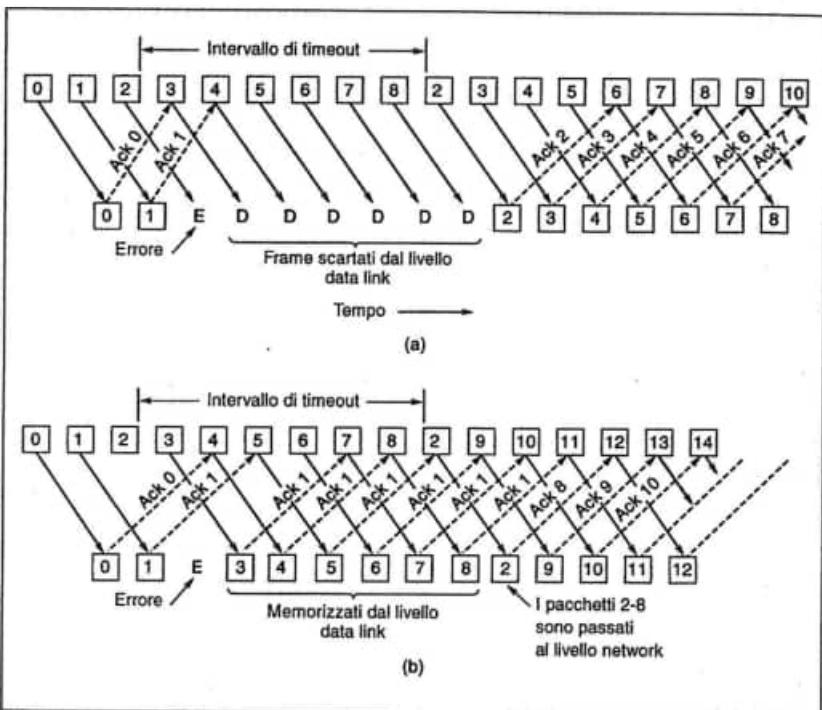


Fig 3-15 (a) L'effetto di un errore nel caso la dimensione della finestra del ricevente sia 1. (b) Effetto di un errore nel caso la dimensione della finestra del ricevente sia grande.

L'altra strategia comune per la gestione degli errori nel caso di utilizzo di pipelining, viene detta **ripetizione selettiva**, e consiste nel fare in modo che il livello data link del ricevente memorizzi tutti i pacchetti corretti che seguono quello danneggiato. Quando il mittente finalmente si accorge del problema, trasmette solo il pacchetto in questione e non tutti i

suoi successori, come in figura 3-15 (b). Se la seconda spedizione riesce, il livello data link ricevente dovrà consegnare velocemente molti pacchetti corretti in successione al livello rete velocemente e mandare il riscontro dell'ultimo numero progressivo.

Questa strategia corrisponde ad avere la finestra del ricevente maggiore di 1. Ogni pacchetto all'interno della finestra può essere accettato e conservato finché tutti i pacchetti precedenti non siano stati passati al livello rete. Questo approccio può richiedere una grossa quantità di memoria per il livello data link se la finestra è grande.

Queste due alternative sono dei compromessi tra la larghezza di banda e lo spazio disponibile per il buffer del livello data link. A seconda di quale risorsa è più preziosa, si userà l'una o l'altra. La figura 3-16 mostra un protocollo di pipeline in cui il livello data link del ricevente accetta solo pacchetti ordinati; i pacchetti che seguono un errore vengono scartati. In questo protocollo, per la prima volta, abbiamo evitato l'ipotesi che il livello rete abbia sempre infinite riserve di pacchetti da passare. Quando il livello rete ha un pacchetto da spedire, può causare l'evento *network_layer_ready*. Tuttavia, al fine di forzare la regola di controllo di flusso ad avere in viaggio in ogni momento non più di *MAX_SEQ* pacchetti di cui non si è avuto ancora riscontro, il livello data link deve essere in grado di proibire al livello rete di passare troppi pacchetti. Le procedure di libreria *enable_network_layer* e *disable_network_layer* hanno questo compito.

Si noti che può essere presente un massimo di *MAX_SEQ* pacchetti e non *MAX_SEQ*+1 in ogni istante, sebbene ci siano *MAX_SEQ*+1 numeri progressivi distinti: 0, 1, 2, ..., *MAX_SEQ*. Per capire il perché di questa restrizione, consideriamo la seguente situazione con *MAX_SEQ* = 7.

1. Il mittente spedisce i pacchetti da 0 a 7.
2. Prima o poi al mittente arriva un ack rispedito con piggybacking per il pacchetto 7.
3. Il mittente spedisce altri otto pacchetti, ancora con numeri progressivi da 0 a 7.
4. Quindi un altro ack spedito con piggybacking raggiunge il mittente.

La domanda è: tutti gli otto pacchetti spediti con il secondo blocco sono arrivati con successo a destinazione oppure sono tutti andati persi (considerando come persi anche quelli scartati dopo un errore)? In entrambi i casi il ricevente avrebbe spedito il pacchetto 7 come ack. Il mittente non ha modo di saperlo. Per questo motivo il massimo numero di pacchetti in viaggio deve essere ristretto a *MAX_SEQ*.

Sebbene il protocollo 5 non conservi i pacchetti che arrivano dopo un errore, non aggira del tutto il problema della bufferizzazione. Poiché un mittente potrà in futuro dover ritrasmettere tutti i pacchetti di cui non ha ancora ricevuto un ack, esso dovrà tenerli in sospeso finché non siano stati accettati dal ricevente. Quando arriva un ack per il pacchetto *n*, vengono considerati ricevuti ack anche per i pacchetti *n-1*, *n-2* ecc. Questa proprietà è particolarmente importante quando alcuni dei pacchetti precedenti che trasportavano degli ack vengono persi o danneggiati. Ogni volta che arriva un ack, il livello data link controlla se qualche buffer può essere rilasciato. Se ci sono buffer che possono essere liberati (cioè c'è spazio nella finestra), il livello rete precedentemente bloccato può essere abilitato a causare l'evento *network_layer_ready*.

Poiché questo protocollo permette di avere più pacchetti in sospeso, ha ovviamente biso-

```

/* Protocol 5 (pipelining) allows multiple outstanding frames. The sender may transmit up to MAX_SEQ frames without waiting for an ack. In addition, unlike the previous protocols, the network layer is not assumed to have a new packet all the time. Instead, the network layer causes a network_layer_ready event when there is a packet to send. */

#define MAX_SEQ 7           /* Should be 2^n - 1 */
typedef enum {frame_arrival, oksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr_t a, seq_nr_t b, seq_nr_t c)
{
    /* Return true if (a <= b) && (b <= c) || (c < a) && (a <= b) || (b <= c) && (c < a) */
    return(true);
}
else
    return(false);
}

static void send_data(seq_nr_t frame_nr, seq_nr_t frame_expected, packet buffer[])
{
    /* Construct and send a data frame. */
    frame_t s;                                /* scratch variable */
    s.info = buffer[frame_nr];                 /* insert packet into frame */
    s.seq = frame_nr;                          /* insert sequence number into frame */
    s.ack = frame_expected + MAX_SEQ % (MAX_SEQ + 1); /* piggyback ack */
    to_physical_layer(&s);
    start_timer(frame_nr);
}

void protocols(void)
{
    seq_nr_t next_frame_to_send;                /* MAX_SEQ > 1; used for outbound stream */
    seq_nr_t old_frame_as yet unacknowledged;   /* oldest frame on inbound stream */
    seq_nr_t next_frame_expected;                /* next frame expected on inbound stream */
    frame_t s;                                  /* scratch variable */
    packet buffer[MAX_SEQ];
    seq_nr_t i;
    seq_nr_t n;
    event_type event;

    enable_network_layer();
    ack_expected = 0;
    next_frame_to_send = 0;
    frame_expected = 0;
    nbuffered = 0;
}

Fig. 3-16 Un protocollo sliding window go back n.

```

gno di avere più timer, uno per ogni pacchetto sospeso. Il timer di un pacchetto scade indipendentemente dagli altri. Tutti questi timer possono facilmente essere simulati via software, usando un unico orologio hardware che causa interruzioni periodiche. I timer in sospeso formano una lista concatenata dove ogni nodo della lista indica quanti *scatti* mancano prima che il timer scada, il pacchetto di riferimento e un puntatore al nodo successivo.

Un esempio di come possono essere realizzati i timer è in figura 3-17. Assumiamo che l'orologio scatti ogni 100 ms. Inizialmente il tempo reale è 10:00:00.0 e ci sono tre timer sospesi, rispettivamente alle 10:00:00.5, 10:00:01.3 e 10:00:01.9. Ogni volta che l'orologio batte, il tempo reale viene aggiornato e viene decrementato il contatore di scatti in testa alla lista. Quando diventa zero, il nodo contenente il timer scaduto viene eliminato come in figura 3-17 (b). Sebbene questa struttura richieda che la lista sia scandita quando sono invocate *start_timer* o *stop_timer*, essa non richiede troppo lavoro per ogni scatto. Nel protocollo 5 entrambe queste routine hanno un parametro che indica il pacchetto da cronometrare.

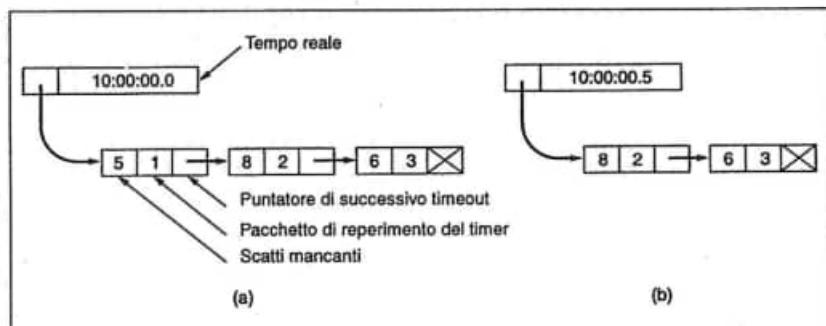


Fig 3-17 Simulazione di timer multipli via software.

3.4.3 Un protocollo con ripetizione selettiva

Il protocollo 5 funziona bene se gli errori sono rari, ma se la linea è scadente, spreca una grossa quantità di larghezza di banda per ritrasmettere i frame. Una strategia alternativa per gestire gli errori è quella di permettere al ricevente di accettare e bufferizzare i frame che ne seguono uno danneggiato o perso. Questo protocollo non scarta i frame solo perché uno precedente è stato danneggiato o perso.

In questo protocollo, il mittente e il ricevente mantengono una finestra di numeri progressivi accettabili. La dimensione della finestra del mittente ha grandezza minima 0 e massima MAX_SEQ . La finestra del ricevente, al contrario, è sempre di dimensione fissa MAX_SEQ . Il ricevente ha un buffer riservato per ogni numero progressivo nella finestra. Associato con ogni buffer c'è un bit (*arrived*) che indica se il buffer è pieno o vuoto. Ogni volta che un frame arriva il suo numero progressivo viene controllato dalla funzione *between*, per vedere se cade all'interno della finestra. Se così è, e se non è ancora stato ricevuto, esso viene accettato e memorizzato. Questa operazione viene fatta senza con-

trollare se esso contenga o meno il pacchetto atteso dal livello rete. Ovviamente, esso deve essere mantenuto nel livello data link e non passato al livello rete fino a ché tutti i frame con numeri più bassi non siano stati consegnati al livello rete nell'ordine corretto. Un protocollo che usa questo algoritmo viene mostrato in figura 3-18.

La ricezione non sequenziale introduce alcuni problemi assenti nei protocolli in cui i frame vengono accettati solo in ordine. Possiamo illustrare il problema più con un esempio. Supponiamo di avere numeri progressivi di 3 bit, così che il mittente possa inviare fino a sette frame prima di dover aspettare un ack. Le finestre iniziali del mittente e del ricevente sono mostrate in figura 3-19 (a). Il mittente quindi trasmette i frame 0..6. La finestra del ricevente permette l'accettazione di frame con numeri progressivi dal 0 al 6 compreso. Tutti e sette i frame arrivano correttamente, il ricevente manda gli ack e fa slittare la sua finestra per accettare la ricezione di 7, 0, 1, 2, 3, 4, o 5, come si vede in figura 3-19 (b). Tutti e sette i buffer sono indicati come vuoti.

È a questo punto che avviene il disastro: un fulmine colpisce un palo telefonico cancellando tutti gli ack. Il timer del mittente prima o poi scade ed esso ritrasmetterà il pacchetto 0. Quando questo pacchetto arriva al ricevente, viene attuato un controllo per vedere se si trova nella finestra del ricevente. Sfortunatamente, nella figura 3-19 (b) il pacchetto 0 è all'interno della nuova finestra, quindi sarà accettato. Il ricevente manda un ack con tecnica piggyback per il pacchetto 6, poiché i pacchetti da 0 a 6 sono stati ricevuti.

Il mittente è felice di sapere che tutti i pacchetti spediti sono arrivati correttamente, avanza la sua finestra e spedisce immediatamente i pacchetti 7, 0, 1, 2, 3, 4, e 5. Il pacchetto 7 sarà accettato dal ricevente e passato al livello rete. Subito dopo, il livello data link del ricevente controlla se c'è già un pacchetto 0 valido, e trovatolo lo passa al livello rete. Di conseguenza il livello rete ottiene un pacchetto errato e il protocollo fallisce.

L'essenza di questo problema è che dopo che il ricevente ha fatto avanzare la sua finestra, il nuovo intervallo di numeri progressivi validi riscrive la vecchia. I successivi pacchetti potrebbero essere o duplicati (se tutti gli ack sono andati persi) oppure nuovi (se tutti gli ack sono stati ricevuti). Il povero ricevente non ha modo di distinguere i due casi.

La soluzione del dilemma consiste nell'assicurarsi che dopo che il ricevente ha fatto avanzare la sua finestra non ci siano sovrapposizioni con la vecchia finestra. Per ottenere questo la dimensione massima della finestra dovrebbe essere al massimo la metà dell'intervallo dei numeri progressivi, come in figura 3-19 (c) e in figura 3-19 (d). Per esempio, se vengono utilizzati 4 bit per i numeri progressivi, essi varieranno tra 0 e 15. Solo otto frame di cui ancora non si è ricevuto ack dovrebbero essere in sospeso in un qualsiasi istante. In questo modo se il ricevente ha appena accettato i frame da 0 a 7 e slitta la finestra in avanti per permettere l'accettazione dei frame da 8 a 15, può in modo non ambiguo dire se i frame successivi sono ritrasmissioni (da 0 a 7) oppure nuovi frame (da 8 a 15). In genere, la dimensione della finestra nel protocollo 6 sarà di $(MAX_SEQ+1)/2$.

Una domanda interessante è: quanti buffer deve avere il ricevente? In nessuna condizione esso accetterà pacchetti il cui numero progressivo sia oltre l'estremità superiore o inferiore della finestra. Di conseguenza, il numero di buffer necessari è uguale alla dimensione della finestra, non all'intervallo dei numeri progressivi. Nell'esempio precedente con numeri progressivi di 4 bit, sono necessari otto buffer, numerati da 0 a 7. Quando arriva il pacchetto i , viene posto nel buffer $i \bmod 8$. Da notare che sebbene i e $(i+8) \bmod 8$

```

/* Protocol 6 (nonsequential receive) accepts frames out of order, but passes packets to the
network layer in order. Associated with each outstanding frame is a timer. When the timer
goes off, only that frame is retransmitted, not all the outstanding frames, as in protocol 5. */

#define MAX_SEQ 7           /* should be 2^n - 1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;
#include "protocol.h"
boolean no_nak = true;      /* no nak has been sent yet */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
    /* Same as between in protocol5, but shorter and more obscure. */
    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[])
{
    /* Construct and send a data, ack, or nak frame. */
    frame s;                  /* scratch variable */

    s.kind = fk;               /* kind == data, ack, or nak */
    if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
    s.seq = frame_nr;          /* only meaningful for data frames */
    s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
    if (fk == nak) no_nak = false; /* one nak per frame, please */
    to_physical_layer(&s);   /* transmit the frame */
    if (fk == data) start_timer(frame_nr % NR_BUFS);
    stop_ack_timer();          /* no need for separate ack frame */
}

void protocol6(void)
{
    seq_nr ack_expected;       /* lower edge of sender's window */
    seq_nr next_frame_to_send; /* upper edge of sender's window + 1 */
    seq_nr frame_expected;     /* lower edge of receiver's window */
    seq_nr too_far;            /* upper edge of receiver's window + 1 */
    seq_nr oldest_frame;       /* which frame timed out? */
    int i;                     /* index into buffer pool */
    frame r;                  /* scratch variable */
    packet out_buf[NR_BUFS];   /* buffers for the outbound stream */
    packet in_buf[NR_BUFS];    /* buffers for the inbound stream */
    boolean arrived[NR_BUFS];  /* inbound bit map */
    seq_nr nbuffered;          /* how many output buffers currently used */

    event_type event;

    enable_network_layer();
    ack_expected = 0;
    next_frame_to_send = 0;
    frame_expected = 0;
    too_far = NR_BUFS;
    nbuffered = 0;              /* initially no packets are buffered */

    for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
}

```

```

wait_for_event(&event);           /* five possibilities: see event_type above */
switch(event) {
    case network_layer_ready:    /* accept, save, and transmit a new frame */
        nbuffered = nbuffered + 1; /* expand the window */
        from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* fetch new packet */
        send_frame(data, next_frame_to_send, frame_expected, out_buf); /* transmit the frame */
        inc(next_frame_to_send); /* advance upper window edge */

    case frame_arrival:          /* a data or control frame has arrived */
        from_physical_layer(&r); /* fetch incoming frame from physical layer */
        if (r.kind == data) {
            /* An undamaged frame has arrived. */
            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)) {
                /* Frames may be accepted in any order. */
                arrived[r.seq % NR_BUFS] = true; /* mark buffer as full */
                in_buf[r.seq % NR_BUFS] = r.info; /* insert data into buffer */
                while (arrived[frame_expected % NR_BUFS]) {
                    /* Pass frames and advance window. */
                    to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                    no_nak = true;
                    arrived[frame_expected % NR_BUFS] = false;
                    inc(frame_expected); /* advance lower edge of receiver's window */
                    inc(too_far); /* advance upper edge of receiver's window */
                    start_ack_timer(); /* to see if a separate ack is needed */
                }
            }
        }
        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; /* handle piggybacked ack */
            stop_timer(ack_expected % NR_BUFS); /* frame arrived intact */
            inc(ack_expected); /* advance lower edge of sender's window */
        }

    case cksum_err:
        if (no_nak) send_frame(nak, 0, frame_expected, out_buf); /* damaged frame */

    case timeout:
        send_frame(data, oldest_frame, frame_expected, out_buf); /* we timed out */

    case ack_timeout:
        send_frame(ack,0,frame_expected, out_buf); /* ack timer expired; send ack */
    }

    if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
}

```

Fig. 3-18 Un protocollo sliding window con ripetizione selettiva.

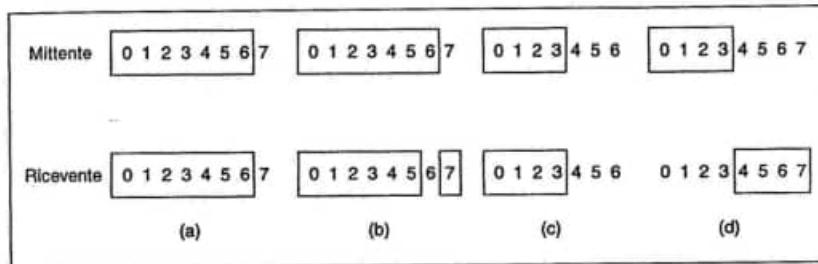


Fig. 3-19 (a) Situazione iniziale con una finestra di dimensione 7. (b) Dopo la spedizione e ricezione ma non il riscontro di 7 pacchetti. (c) Situazione iniziale con una finestra di dimensione 4. (d) Dopo la spedizione e ricezione ma non il riscontro di 4 pacchetti.

competono per lo stesso buffer, essi non sono mai nella stessa finestra nello stesso momento, perché ciò implicherebbe una minima dimensione della finestra di 9.

Per la stessa ragione, il numero di timer necessari è uguale al numero dei buffer e non alla quantità dei numeri progressivi. Esiste in realtà un timer associato a ogni buffer. Quando il timer scade, il contenuto del buffer viene ritrasmesso.

Nel protocollo 5 viene fatta l'assunzione implicita che il canale sia utilizzato pesantemente. Quando arriva un frame, l'ack non viene spedito immediatamente, ma viene agganciato in piggybacking al successivo frame da spedire. Se il traffico in senso inverso è leggero, l'ack resterà sospeso per un lungo periodo di tempo. Se c'è molto traffico in una direzione e niente nell'altra, solo pacchetti *MAX_SEQ* vengono inviati e quindi il protocollo si blocca.

Nel protocollo 6 il problema viene risolto. Dopo la ricezione di un gruppo di frame, viene inizializzato un timer ausiliario con *start_ack_timer*. Se non si ha traffico nella direzione di ritorno che si prenda carico della spedizione dell'ack prima che scada il timer, un frame di ack viene spedito separatamente. Un'interruzione dovuta al timer ausiliario viene detta evento *ack_timeout*. Con questa soluzione, è ora possibile avere flusso di traffico unidirezionale, infatti non è più un ostacolo la mancanza di traffico nel senso inverso per l'agganciamento dei frame di ack. Esiste un solo timer ausiliario e se *start_ack_timer* viene chiamata mentre il timer è attivo, esso viene riinizializzato in modo che abbia a disposizione un intervallo intero.

È essenziale che il tempo di timeout sia piuttosto breve rispetto al timer usato per i frame di dati. Questo perché i frame di ack possano essere correttamente ricevuti prima che il timer del mittente scada ed esso ritrasmetta il frame.

Il protocollo 6 usa una strategia più efficiente del protocollo 5 nel gestire gli errori. Ogni volta che il ricevente ha modo di sospettare che ci sia stato un errore, rimanda un frame di ack negativo (NAK). Questo frame è una richiesta di ritrasmissione di un frame specificato nel NAK. Ci sono due casi in cui il ricevente può sospettare: è arrivato un frame danneggiato oppure è arrivato un frame diverso da quello atteso (un frame si è potenzialmente perso). Per evitare richieste plurime di ritrasmissione dello stesso frame perso, il ricevente dovrebbe tenere traccia del fatto che abbia o non abbia già mandato un NAK.

per quel dato frame. La variabile *no_ack* nel protocollo 6 è vera se nessun NAK è stato spedito da *frame_expected*. Se il NAK viene danneggiato o perso, non accade nulla di grave, infatti prima o poi il timer del mittente scadrà ed esso ritrasmetterà ugualmente il frame. Se arriva il frame sbagliato dopo che un NAK è stato danneggiato o perso, *no_ack* sarà vera e il timer ausiliario inizializzato. Quando scadrà verrà spedito un ack per risincronizzare il mittente con lo stato corrente del ricevente.

In alcune situazioni, il tempo necessario perché il frame venga spedito, ricevuto e analizzato, sia mandato e ricevuto un ack, è quasi costante. In questi casi, il mittente può regolare il suo timer in modo che il suo intervallo sia leggermente maggiore del tempo necessario per l'intero ciclo. Se invece questo tempo è ampiamente variabile, il mittente deve decidere tra scegliere un intervallo piccolo rischiando ritrasmissioni non necessarie, perdendo così larghezza di banda, oppure scegliere un intervallo grande restando senza nulla da fare per molto tempo dopo un errore perdendo in ogni modo larghezza di banda. Se il traffico in senso inverso è raro, il tempo di arrivo di un ack sarà irregolare, più corto quando si ha traffico, più lungo se non ce n'è. Il tempo di elaborazione variabile da parte del ricevente può essere un altro problema. In generale, quando la deviazione standard dell'intervallo di ack è piccola in confronto all'intervallo stesso, il timer può essere regolato in modo stretto e non ci sarà bisogno di NAK. Altrimenti, i timer dovranno essere regolati larghi e i NAK potranno apprezzabilmente rendere più veloce la trasmissione di frame danneggiati e persi.

Una questione fortemente legata all'argomento dei timeout e dei NAK è quella di determinare quale frame abbia causato un timeout. Nel protocollo 5 è sempre *ack_expected*, perché esso è sempre il più vecchio. Nel protocollo 6, non c'è un modo semplice di determinarlo. Supponiamo che i frame dallo 0 al 4 siano stati ritrasmessi, quindi che la lista dei frame sospesi sia 01234, in ordine dal più vecchio al più giovane. Immaginiamo ora che il timer 0 scada, che sia trasmesso 5 (un nuovo frame), che i timer 1 e 2 scadano, e che sia trasmesso 6 (un altro frame). A questo punto la lista dei frame sospesi sarà 3405126 dal più vecchio al più giovane. Se tutto il traffico in arrivo viene perso per qualche tempo, i timer dei sette frame sospesi scadranno nell'ordine. Per fare in modo che l'esempio non si complichi oltre, non mostriamo la gestione dei timer. Assumiamo invece che la variabile *oldest_frame* venga "settata" dopo un timeout a indicare il timer del frame che è scaduto.

3.5 Specifica e verifica di protocolli

I protocolli reali e le loro implementazioni sono spesso piuttosto complicati. Di conseguenza, è stata fatta molta ricerca al fine di trovare strategie matematiche formali per la specifica e verifica dei protocolli. Nei paragrafi che seguono considereremo alcuni modelli e tecniche che, anche se saranno studiati nel contesto del livello data link, sono applicabili anche agli altri livelli.

3.5.1 I modelli a macchine a stati finiti

Un concetto chiave utilizzato in molti protocolli è quello di **macchina a stati finiti**. Con questa tecnica, ogni **macchina di protocollo** (mittente o ricevente che sia) è sempre in

uno stato specifico. Il suo stato è composto dai valori di tutte le sue variabili, incluso il contatore di programma.

In diversi casi, può essere raggruppato ai fini dell'analisi un grosso numero di stati. Per esempio, considerando il ricevente nel protocollo 3, possiamo astrarre da tutti gli stati possibili i due più importanti: l'attesa del frame 0 e quella del frame 1. Tutti gli altri stati possono essere visti come di transito, cioè transizioni tra uno stato e l'altro. Tipicamente gli stati vengono scelti come gli istanti in cui la macchina del protocollo attende il successivo evento (cioè, nel nostro esempio, eseguendo la chiamata alla procedura *wait (event)*). A questo punto lo stato della macchina del protocollo è completamente determinato dallo stato delle sue variabili. Il numero degli stati è quindi 2^n , dove n è il numero di bit necessari per rappresentare tutte le variabili combinate.

Lo stato del sistema completo è dato dalla combinazione di tutti gli stati delle due macchine di protocollo e del canale. Lo stato del canale è determinato dal suo contenuto. Utilizzando ancora il protocollo 3 come esempio, il canale ha quattro possibili stati: un frame 0 o 1 che si muove dal mittente al ricevente, un frame di ack che si sposta in direzione opposta, o il canale vuoto. Se modelliamo il mittente e il ricevente entrambi mediante due soli stati, il sistema completo ha 16 stati distinti.

È necessario parlare un po' del canale; il concetto di avere un frame "nel canale" è ovviamente una astrazione. Quello che veramente si intende è che il frame è stato parzialmente trasmesso, parzialmente ricevuto, ma non del tutto elaborato nella destinazione. Un frame rimane "nel canale" finché la macchina del protocollo non esegue *FromPhysicalLayer* e lo elabora.

Da uno stato sono possibili zero o più **transizioni** verso altri stati. Le transizioni avvengono quando un particolare evento si verifica. Per una macchina di protocollo una transizione si può verificare quando un frame viene spedito, quando un frame arriva, quando un timer scade, quando avviene un'interruzione ecc. Per un canale invece gli eventi tipici sono l'inserimento di un nuovo frame nel canale da una macchina di protocollo, la consegna di un frame a una macchina di protocollo o la perdita di un frame dovuta a una raffica di rumore. Data una descrizione completa della macchina di protocollo e delle caratteristiche del canale, è possibile disegnare un grafo diretto che mostri tutti gli stati come nodi e tutte le transizioni come archi orientati.

Uno stato particolare viene identificato come **stato iniziale**. Questo stato corrisponde alla descrizione del sistema quando inizia l'esecuzione o qualche comodo punto di partenza scelto più avanti. Dallo stato iniziale, alcuni, e forse tutti gli altri stati sono raggiungibili mediante una sequenza di transizioni. Utilizzando tecniche ben conosciute della teoria dei grafi (per esempio calcolando la chiusura transitiva di un grafo), è possibile determinare quali stati sono raggiungibili e quali non lo sono. Questa tecnica viene detta **analisi di raggiungibilità** (Lin et al., 1987). Questo tipo di analisi può risultare utile per determinare se un protocollo è corretto.

Formalmente un modello di macchina a stati finiti di un protocollo può essere visto come una quadrupla (S, M, I, T) dove:

S è l'insieme degli stati in cui possono trovarsi i processi e il canale.

M è l'insieme dei frame che possono essere scambiati nel canale.

I è l'insieme degli stati iniziali dei processi.

T è l'insieme delle transizioni tra gli stati.

All'inizio tutti i processi sono nei loro stati iniziali. Poi iniziano ad accadere eventi, come la generazione di frame disponibili per la trasmissione oppure timer che scadono. Ogni evento può scatenare un'azione da parte dei processi oppure del canale e di conseguenza il cambiamento di stato. Enumerando tutti i possibili successori di ogni stato, si può costruire il grafo di raggiungibilità e analizzare il protocollo.

L'analisi di raggiungibilità può essere utilizzata per riconoscere diversi errori nella specifica del protocollo. Per esempio, se è possibile che un certo frame compaia in un certo stato e la macchina a stati finiti non indica quale azione compiere, la specifica è sbagliata (*incompletezza*). Se esiste un insieme di stati dal quale non si esce e da cui non è possibile progredire in nessun modo (ricezione di frame corretti), si ha uno stallo (*deadlock*). Un errore meno grave è una specifica di protocollo che indica come trattare un evento in uno stato dove l'evento non può verificarsi (transizione estranea). Possono essere rilevati anche altri errori.

Come esempio di un modello a macchina a stati finiti, si consideri la figura 3-20 (a). Questo grafo corrisponde al protocollo 3 descritto precedentemente: ogni macchina di protocollo ha due stati e il canale ne ha quattro. Cioè un totale di 16 stati, non tutti raggiungibili da quello iniziale. Quelli non raggiungibili non vengono mostrati nella figura. Ogni stato ha un'etichetta di tre caratteri, XYZ , dove X è 0 o 1, corrispondentemente al frame che il mittente sta cercando di spedire, Y è anch'esso 0 o 1, corrispondentemente al frame che il ricevente sta aspettando, e Z è 0, 1, A, oppure vuoto (-), corrispondentemente allo stato del canale. In questo esempio lo stato iniziale è stato scelto (000). In altri termini, il mittente ha appena spedito il frame 0, il ricevente sta aspettando quello stesso frame e il frame 0 è attualmente nel canale.

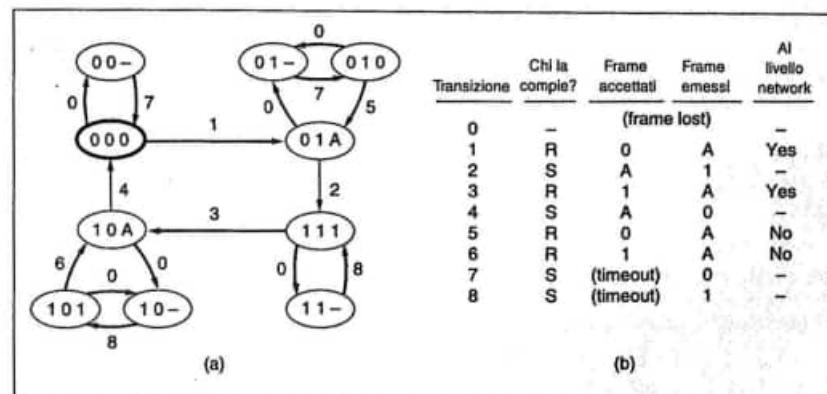


Fig. 3-20 (a) Grafo per il protocollo 3. (b) Transizioni.

In figura 3-20 sono mostrati nove tipi di transizioni. La transizione 0 consiste nel canale che perde il suo contenuto. La transizione 1 consiste nel canale che consegna correttamente il pacchetto 0 al ricevente, con il ricevente che cambia stato per attendere il frame 1 e spedire un ack. La transizione 1 corrisponde anche al ricevente che consegna il pacchetto 0 al livello rete. Le altre transizioni sono elencate in figura 3-20 (b). L'arrivo di un frame con una checksum errata non viene mostrato poiché non si ha cambiamento di stato (nel protocollo 3).

Durante le normali operazioni, le transizioni 1, 2, 3 e 4 vengono ripetute di continuo. In ogni ciclo due pacchetti vengono consegnati, riportando il mittente sullo stato iniziale, intendo a spedire un altro frame con numero progressivo 0. Se il canale perde il frame 0, esso effettua la transizione dallo stato (000) allo stato (00-). Prima o poi, il timer del mittente scadrà (transizione 7) e il sistema tornerà sullo stato (000). La perdita di un ack è più complicata, infatti richiede due transizioni, la 7 e la 5 oppure la 8 e la 6 per riparare il danno.

Una delle proprietà che un protocollo con numeri progressivi di un bit deve avere è che, indipendentemente dalla sequenza di eventi accaduti, il ricevente non consegnerà mai due pacchetti dispari non intervallati da uno pari e viceversa. Dal grafo in figura 3-20 notiamo che questo requisito può essere espresso più formalmente come "non devono esistere cammini dallo stato iniziale dove vengano effettuate due transizioni 1 senza che sia effettuata una transizione 3 tra di esse e viceversa".

Dalla figura si rileva che il protocollo è corretto sotto questo aspetto.

Un altro requisito simile è che non devono esistere cammini nei quali il mittente cambi stato due volte (per esempio da 0 a 1 e ancora a 0) mentre lo stato del ricevente rimane costante. Se così fosse infatti, nella corrispondente sequenza di eventi due frame andrebbero definitivamente persi senza che il ricevente lo noti. La sequenza di pacchetti consegnati avrebbe un salto di due pacchetti.

Un'altra proprietà importante per un protocollo è l'assenza di **stallo**. Lo stallo è una situazione in cui il protocollo non può progredire (cioè consegnare messaggi al livello rete), indipendentemente dalla sequenza di eventi che possono accadere. In termini di grafo, lo stallo è caratterizzato dall'esistenza di un insieme di stati raggiungibile dallo stato iniziale con le seguenti due proprietà:

1. Non ci sono transizioni che portino fuori dall'insieme.
2. Non ci sono transizioni dell'insieme che facciano progredire.

Giunto in una situazione di stallo, il protocollo vi rimane per sempre. Come si nota dal grafo in figura, il protocollo 3 è esente da stallo.

Consideriamo ora una variante del protocollo 3, in cui il canale half-duplex è rimpiazzato da un canale completamente duplex. In figura 3-21 mostriamo gli stati come prodotto degli stati delle due macchine di protocollo e degli stati dei due canali. Da notare che il canale diretto in avanti ora ha tre stati: frame 0, frame 1 o vuoto e che il canale inverso ne ha due, A oppure vuoto. Le transizioni sono le stesse della figura 3-20 (b), eccetto per il fatto che quando un frame di dati e un ack sono nel canale nello stesso istante; si ha una peculiarità: il ricevente non può rimuovere il frame di dati dal canale da solo, poiché

ciò comporterebbe l'avere due ack simultaneamente nel canale, cosa non permessa nel nostro modello (sebbene sia facile progettare un modello che lo consenta). In modo simile il mittente non può prelevare l'ack poiché dovrebbe spedire un altro frame prima che il precedente sia stato accettato. Di conseguenza entrambi gli eventi devono avvenire simultaneamente, per esempio la transizione tra lo stato (000A) e lo stato (111A), etichettata 1+2 nella figura.

Nella figura 3-21 (a) esistono cammini che portano al fallimento del protocollo. In particolare, ci sono cammini nei quali il mittente preleva ripetutamente nuovi pacchetti, anche se i precedenti non sono stati consegnati correttamente. Il problema sorge poiché adesso è possibile che il timer del mittente scada e che esso spedisca un nuovo frame senza disturbare l'ack sul canale inverso. Quando questo ack arriverà, sarà erroneamente considerato come relativo alla trasmissione corrente e non a quella precedente.

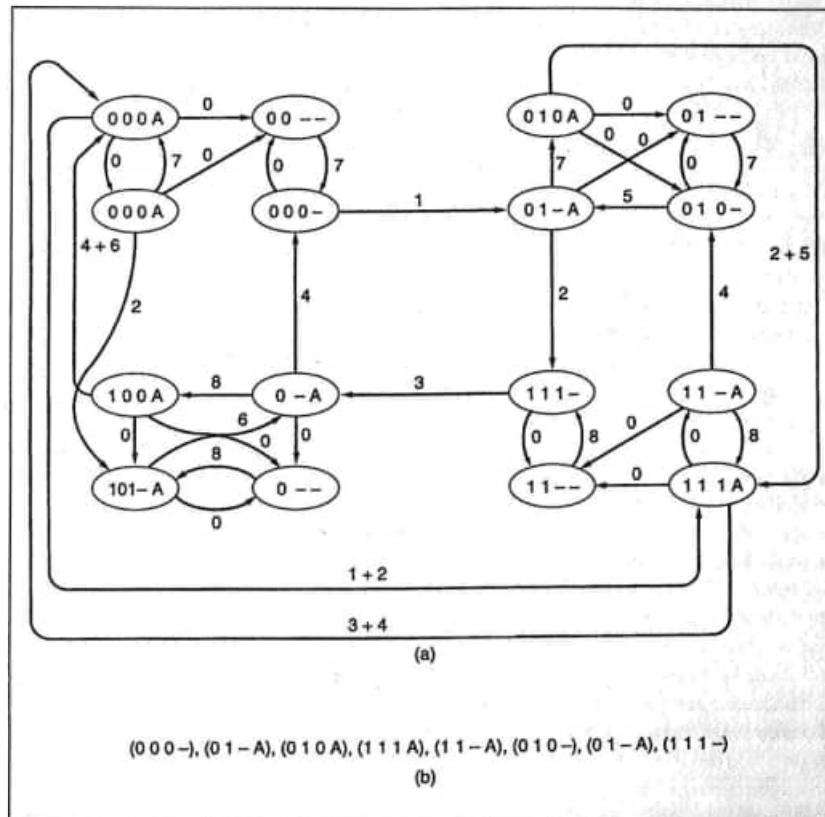


Fig. 3-21 (a) Un grafo per il protocollo 3 e un canale full-duplex. (b) Sequenza di stati che causa fallimento del protocollo.

Una sequenza di stati che causa il fallimento del protocollo è mostrata in figura 3-21 (b). Negli stati quattro e sei della sequenza, il mittente cambia stato, indicando che sta caricando un nuovo pacchetto dal livello rete, mentre il ricevente non cambia stato, quindi non sta consegnando nulla al livello rete.

3.5.2 I modelli a reti di Petri

La macchina a stati finiti non è l'unico modello per la specifica formale di protocolli. In questo paragrafo descriveremo un altro modello, le reti di Petri (Petri Nets: Danthine, 1980). Una rete di Petri è composta da quattro elementi base: i posti, le transizioni, gli archi e i token (marche). Un posto rappresenta uno stato in cui il sistema può trovarsi (o parte di esso). La figura 3-22 mostra una rete di Petri con due posti, A e B, entrambi indicati con un cerchio. Il sistema è attualmente nello stato A, indicato dal token (il puntino) nel posto A. Una transizione viene indicata con una linea orizzontale o verticale. Ogni transizione ha zero o più archi di input, che arrivano dai suoi posti di input, e zero o più archi di output che si dirigono verso i suoi posti di output.

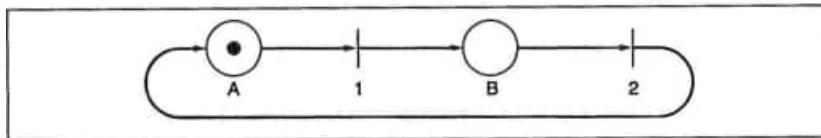


Fig. 3-22 Una rete di Petri con due posti e due transizioni.

Una transizione è **abilitata** se c'è almeno un token di input in ogni suo posto di input. Ogni transizione abilitata può **scattare** quando vuole, rimuovendo un token da ogni suo posto di input e depositandone uno in ogni suo posto di output. Se il numero degli archi di input e quello degli archi di output differiscono, i token non saranno conservati. Se due o più transizioni sono abilitate una qualsiasi di esse può scattare. La scelta di far scattare una transizione è indeterminata, ragione per cui le reti di Petri sono utili modelli per i protocolli. La rete di Petri della figura 3-22 è deterministica e può essere utilizzata per modellare qualsiasi processo a due fasi (ad es. il comportamento di un bambino che mangia, dorme, mangia, dorme, e così via). Come per tutti gli strumenti per costruire modelli, i dettagli sono trascurati.

La figura 3-23 mostra il modello di reti di Petri della figura 3-21. Diversamente dal modello a macchina a stati finiti, qui non si hanno stati composti; lo stato del mittente, quello del canale e quello del ricevitore sono rappresentati separatamente. Le transizioni 1 e 2 corrispondono alla trasmissione del frame 0 da parte del mittente, rispettivamente in situazione generale e allo scadere del timer. Le transizioni 3 e 4 sono le analoghe per il frame 1. Le transizioni 5, 6 e 7 corrispondono rispettivamente alla perdita del frame 0, di un ack e del frame 1. Le transizioni 8 e 9 vengono effettuate quando un frame di dati con un numero progressivo errato arriva al ricevitore. Le transizioni 10 e 11 rappresentano l'arrivo del successivo (nell'ordine) frame al ricevitore e la sua consegna al livello rete. Le reti di Petri possono essere utilizzate per rilevare fallimenti del protocollo in modo simile alle macchine a stati finiti. Per esempio, se qualche sequenza di firing (scatto)

includesse la transizione 10 due volte senza intervallamento con la 11, il protocollo sarebbe scorretto. Il concetto di stallo in una rete di Petri è anch'esso simile a quello per le macchine a stati finiti.

Le reti di Petri possono essere rappresentate in una forma algebrica conveniente che assomiglia a una grammatica. Ogni regola specifica i posti di input e di output di una transizione, per esempio la transizione 1 in figura 3-23 risulta come $BD \rightarrow AC$. Lo stato corrente della rete di Petri è rappresentato da una collezione non ordinata di posti, dove ogni posto è rappresentato nella collezione tante volte quanti sono i token che possiede. Ogni regola i cui posti a sinistra sono presenti può esser fatta scattare, rimuovendo quei posti dallo stato corrente e aggiungendo i suoi posti di output. La marcatura (marking) della figura 3-23 è ACG , quindi la regola 10 ($CG \rightarrow DF$) può essere applicata ma la regola 3 ($AD \rightarrow BE$) non può essere applicata.

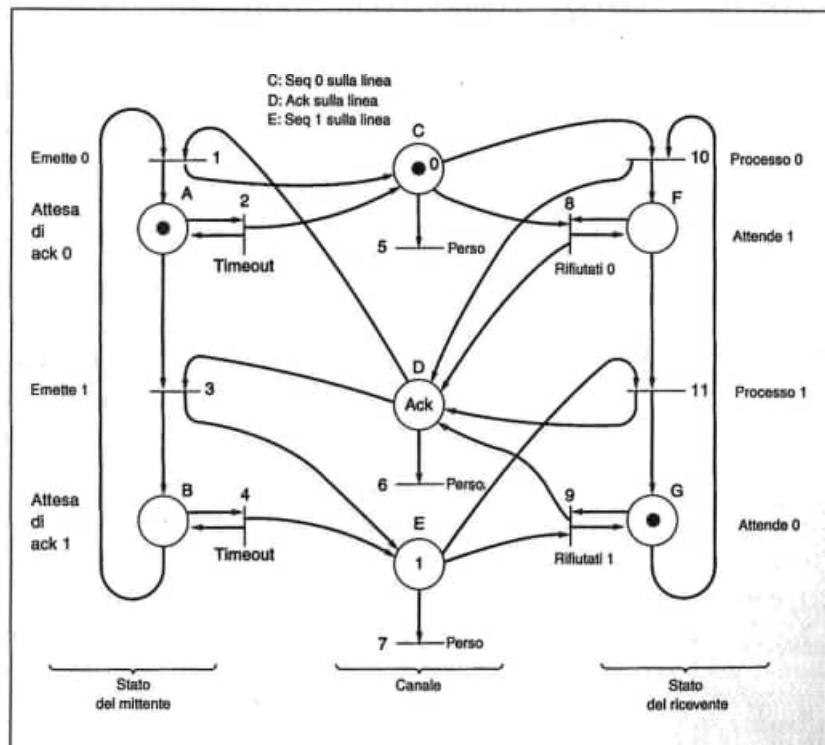


Fig. 3-23 Un modello a rete di Petri per il protocollo 3.

3.6 Esempi di protocolli data link

Nei paragrafi che seguono esamineremo diversi protocolli data link largamente utilizzati. Il primo, HDLC, è di uso frequente sotto X.25 e molte altre reti. Dopodiché, prenderemo in esame i protocolli data link utilizzati nelle reti Internet e ATM. Nei capitoli seguenti, tra l'altro, useremo più frequentemente negli esempi le reti Internet e ATM.

3.6.1 HDLC – High-level Data Link Control

In questo paragrafo esamineremo un gruppo di protocolli correlati, che sono un po' vecchi ma ancora molto utilizzati in molte reti in tutto il mondo. Essi derivano dal protocollo data link utilizzato nell'SNA dell'IBM, chiamato **SDLC** (protocollo **Synchronous Data Link Control** – procedura di controllo sincrono per il collegamento dati). Dopo lo sviluppo di SDLC l'IBM lo sottomise all'ANSI e all'ISO perché fosse rispettivamente accettato come standard statunitense e internazionale. L'ANSI lo modificò e produsse l'**ADCCP (Advanced Data Communication Control Procedure** – procedura di controllo per la comunicazione dati avanzata), e l'ISO lo trasformò in **HDLC (High-level Data Link Control)** – controllo di collegamento dati ad alto livello). Quindi il CCITT adottò e modificò l'HDLC per la sua **LAP (Link Access Procedure** – procedura di accesso al collegamento) come parte dell'interfaccia standard di rete X.25, ma in seguito lo modificò ancora e ottenne il **LAPB**, più compatibile con una versione successiva di HDLC. La cosa bella degli standard è che la scelta è sempre ampia. Inoltre se non ve ne piace nessuno, è sempre possibile attendere il modello dell'anno prossimo.

Tutti questi protocolli sono basati sullo stesso principio; sono tutti orientati al bit e usano il riempimento di bit (bit stuffing) per la trasparenza dei dati. Differiscono in alcuni particolari minori ma comunque fastidiosi. La discussione riguardante i protocolli orientati al bit che segue deve essere considerata come una introduzione generale; per i dettagli dei vari protocolli si consultino le relative definizioni.

Tutti i protocolli orientati al bit usano per i frame la struttura mostrata in figura 3-24. Il campo **Address** è di primaria importanza su linee con terminali multipli, dove viene usato per identificare uno dei terminali. Per le linee punto-a-punto, viene utilizzato qualche volta per distinguere i comandi dalle risposte.

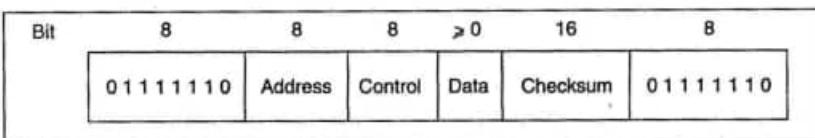


Fig. 3-24 Formato di frame per protocolli orientati al bit.

Il campo **Control** è usato per i numeri progressivi, gli ack e per altri scopi descritti più avanti.

Il campo **Data** può contenere informazioni variabili. Può essere arbitrariamente lungo sebbene l'efficienza della checksum decresca con l'aumentare della lunghezza del frame a causa della probabilità maggiore delle raffiche di errori.

3.6 Esempi di protocolli data link

Il campo **Checksum** è una variazione del ben noto codice di ridondanza ciclica, che usa CRC-CCITT come generatore polinomiale. La variante consiste nel permettere il riconoscimento di byte di flag persi.

Il frame è delimitato da un'altra sequenza di flag (01111110). Sulle linee punto a punto non impegnate, le sequenze di flag vengono trasmesse in continuazione. Il frame più piccolo contiene tre campi e in totale 32 bit, esclusi i flag da entrambe le parti.

Esistono tre tipi di frame: **Information** (di informazione), **Supervisory** (di supervisione) e **Unnumbered** (non numerato). Il contenuto del campo **Control** di questi tre tipi è in figura 3-25. Il protocollo usa una finestra scorrevole con numeri progressivi di 3 bit. Possono essere sospesi fino a sette frame senza ack. Il campo **Seq** mostrato in figura 3-25 (a) è il numero progressivo del frame. Il campo **Next** contiene un ack spedito con piggy-back. Tuttavia, tutti i protocolli aderiscono alla convenzione che invece di usare piggy-backing per il numero dell'ultimo frame ricevuto correttamente, lo si usa per il numero del primo frame non ricevuto (cioè il prossimo atteso). La scelta di usare l'ultimo frame ricevuto oppure il prossimo atteso è arbitraria; non importa quale convenzione viene usata, a patto che venga usata consistentemente.

Bit	1	3	1	3
(a)	0	Seq	P/F	Next
(b)	1	0	Type	P/F
(c)	1	1	Type	P/F
				Modifier

Fig. 3-25 Campo **Control** di (a) un frame di informazione, (b) un frame di supervisione, (c) un frame non numerato.

Il bit **P/F** sta per **Poll/Final**. Esso è utile quando un computer (o concentratore) interroga ciclicamente (polling) un gruppo di terminali. Quando è usato in modo **P**, il computer invita i terminali a spedire dati. Tutti i frame inviati da un terminale, a parte l'ultimo, hanno il bit **P/F** "settato" a **P**. L'ultimo è "settato" a **F**.

In alcuni dei protocolli il bit **P/F** è utilizzato per forzare le altre macchine a spedire un frame **Supervisory** immediatamente, invece di aspettare il traffico in senso inverso e usare piggybacking per spedire le informazioni sulla finestra. Il bit ha anche altri usi minori in relazione ai frame **Unnumbered**.

I diversi tipi di frame **Supervisory** vengono distinti mediante il campo **Type**.

Type = 0 indica un frame di ack (ufficialmente chiamato **RECEIVE READY**: ricevente pronto), usato per determinare il successivo frame atteso. Questo frame è utilizzato quando non si ha traffico in senso inverso per usare piggybacking.

Type = 1 indica un frame di ack negativo (ufficialmente detto **REJECT**: rifiuto). È utilizzato per indicare che si è verificato un errore di trasmissione. Il campo **Next** indica

il primo frame della sequenza non ricevuto correttamente (cioè il frame da ritrasmettere). Al mittente viene richiesto di ritrasmettere tutti i frame sospesi a partire da *Next*. Questa tecnica è simile a quella del nostro protocollo 5 piuttosto che a quella utilizzata nel protocollo 6.

Type = 2 è RECEIVE NOT READY (ricevente non pronto). Esso dà ack di tutti i frame fino a *Next* (escluso), proprio come RECEIVE READY, ma comunica al mittente di bloccare la spedizione. RECEIVE NOT READY intende segnalare certi problemi temporanei del ricevente, come i buffer pieni e non intende essere un'alternativa al controllo di flusso della finestra scorrevole. Quando la situazione si è ristabilita, il ricevente manda un RECEIVE READY oppure REJECT o precisi frame di controllo.

Type = 3 indica SELECTIVE REJECT (rifiuto selettivo). Richiede la trasmissione dei soli frame specificati. In questo senso assomiglia al protocollo 6 più che al 5 ed è per questo il più utile quando la dimensione della finestra del mittente è metà della quantità dei numeri progressivi, o meno. Quindi se il ricevente desidera bufferizzare frame di sequenza per un potenziale uso futuro, può forzare la ritrasmissione di alcuni specificati frame usando il rifiuto selettivo. HDLC e ADCCP prevedono questo tipo di frame al contrario di SDLC e LAPB che non lo prevedono (cioè non hanno rifiuto selettivo) e i frame di tipo 3 sono indefiniti.

La terza classe di frame è composta di frame Unnumbered. Questo tipo di frame viene utilizzato a scopo di controllo ma anche per trasportare dati quando si richieda un servizio inaffidabile senza connessione. I vari protocolli orientati al bit differiscono considerabilmente in questo caso, diversamente dagli altri due tipi, dove la somiglianza è notevole. Sono disponibili cinque bit per indicare il tipo del frame ma non vengono usate tutte e 32 le possibilità.

Tutti i protocolli hanno un comando, DISC (DISConnect: sconnettersi), che permette a una macchina di annunciare che sta per disconnettersi (ad es. per motivi di manutenzione). Essi forniscono inoltre un comando per permettere a una macchina appena tornata in linea di annunciare la sua presenza e di azzerare tutti i numeri progressivi. Questo comando è SNRM (Set Normal Response Mode: impostazione del modo di risposta normale). Sfortunatamente il "modo di risposta normale" è tutt'altro che normale. È un modo sbilanciato (cioè asimmetrico) di risposta in cui un capo della linea è il master (padrone) e l'altro lo slave (servo). SNRM risale ai tempi in cui la comunicazione dati si svolgeva tra un terminale "non intelligente" che comunicava con un computer, situazione totalmente asimmetrica. Per rendere il protocollo più adatto se si hanno due partner equivalenti, HDLC e LAPB hanno un comando aggiuntivo, SABM (Set Asynchronous Balanced Mode: impostazione di modo bilanciato asincrono), che impone la linea e dichiara le due parti uguali. Essi forniscono anche i comandi SABME e SNRME che vengono usati come SABM e SNRM rispettivamente, eccetto per il fatto che essi abilitano un formato esteso di frame che usa numeri progressivi di 7 bit invece che di 3 bit.

Un terzo comando fornito da tutti i protocolli è FRMR (FRaMe Reject: rifiuto di frame), usato per indicare che un frame con checksum corretta ma semantica impossibile, è arrivato. Esempi di semantica impossibile sono: un frame di tipo 3 supervisore in LAPB, un frame più corto di 32 bit, un frame di controllo illegale, un ack di un frame fuori dalla finestra, e così via. I frame FRMR contengono un campo di 24 bit di dati che indica cosa

non andava nel frame. I dati includono il campo di controllo del frame errato, i parametri della finestra e una serie di bit usati per segnalare errori specifici.

I frame di controllo si possono perdere o danneggiare, proprio come i frame di dati, quindi devono anch'essi essere seguiti da un ack. Uno speciale frame di controllo chiamato UA (Unnumbered Ack) viene fornito per questo scopo. Poiché un solo frame può essere sospeso, non c'è mai ambiguità sull'individuazione del frame di controllo di cui è stato mandato l'ack.

I rimanenti frame di controllo hanno a che fare con l'inizializzazione, il polling (interrogazione ciclica) e il resoconto sullo stato. Esiste anche un frame di controllo che può contenere informazione arbitraria, UI (Unnumbered Information). Questi dati non verranno passati al livello rete ma trattenuti dal livello data link del ricevente stesso. Nonostante il suo largo utilizzo, HDLC è ben lontano dall'essere perfetto. Una discussione sui problemi associati ad esso si trova in Fiorini *et al.* (1995).

3.6.2 Il livello data link in Internet

La rete Internet consiste di macchine singole (host e router) e dell'infrastruttura di comunicazione che le connette. All'interno di un singolo edificio sono largamente usate per l'interconnessione le LAN, ma la maggior parte dell'infrastruttura delle WAN è costruita su linee dedicate punto a punto. Nel capitolo 4 considereremo le LAN; qui esamineremo i protocolli data link usati per le linee punto-a-punto nella rete Internet.

In pratica, la comunicazione punto-a-punto è usata soprattutto in due tipi di situazioni. Primo, migliaia di organizzazioni hanno una o più LAN, ognuna con diversi host (personal computer, workstation, server e così via) e un router (o un bridge, che è sostanzialmente simile). Spesso i router sono interconnessi da una dorsale LAN (backbone). Tipicamente tutte le connessioni con il mondo esterno passano attraverso uno o due router da cui linee dedicate punto-a-punto sono connesse con router più distanti. Sono questi router e le loro linee che costituiscono le sottoreti di comunicazione sulle quali è costruita la rete Internet.

La seconda situazione in cui le linee punto-a-punto giocano il ruolo primario in Internet è costituito da milioni di individui che utilizzano da casa connessioni a Internet usando modem e linee telefoniche. Generalmente quello che succede è che il PC dell'utente a casa chiama un **Internet provider** (fornitore di Internet) ad esempio una società commerciale come America Online, CompuServe o Microsoft Network; oppure un'università o un'azienda che offre la connessione a Internet agli studenti o agli impiegati. Alcune volte il PC di casa funziona come un terminale orientato al carattere connesso a un sistema a condivisione di tempo fornitore di servizi Internet. In questo modo l'utente può digitare comandi e far girare programmi ma non servizi grafici come il World Wide Web. Questo modo di connessione viene detto **shell account**.

In modo alternativo, il PC di casa può chiamare un router fornitore di servizi Internet e agire come un qualsiasi computer in Internet. Questa tecnica non è diversa da avere una linea dedicata tra il PC e il router eccetto per il fatto che la connessione viene chiusa quando l'utente chiude la sessione. Con questo tipo di approccio diventano disponibili tutti i servizi Internet, compresi quelli grafici. Un PC di casa che chiama un Internet provider viene illustrato in figura 3-26.

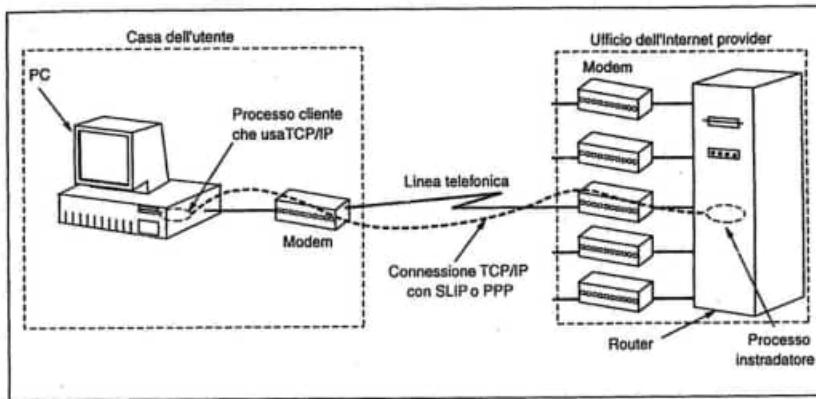


Fig. 3-26 Un PC di casa che funziona come host Internet.

Per entrambe le connessioni router-router e host-router, sono indispensabili alcuni protocolli data link punto-a-punto per la gestione della suddivisione in frame, controllo degli errori e le altre funzioni del livello data link che abbiamo studiato in questo capitolo. Due di questi protocolli sono largamente usati in Internet: SLIP e PPP. Esaminiamo di seguito entrambi.

SLIP – linea seriale IP

SLIP è il più vecchio dei due protocolli. Esso fu ideato da Rick Adams nel 1984 per connettere le workstation Sun a Internet via linea telefonica mediante un modem. Il protocollo, che viene descritto in RFC 1055, è molto semplice. La workstation trasmette sulla linea pacchetti IP con uno speciale byte indicatore (0xC0) alla fine per la suddivisione. Se tale byte si trova all'interno del pacchetto, viene utilizzata una forma speciale di rimpiazzamento di caratteri e viene spedita al suo posto la sequenza di due byte (0xDB, 0xDC). Se 0xDB è presente nel pacchetto IP viene anch'essa rimpiazzata. Alcune implementazioni di SLIP appendono un byte indicatore in testa e in coda ad ogni pacchetto IP spedito.

Alcune versioni più recenti di SLIP eseguono compressioni delle intestazioni TCP e IP. Sfruttano il fatto che pacchetti consecutivi hanno spesso molti campi di intestazione comuni. Questi vengono compressi omettendo quei campi che sono uguali nei pacchetti IP precedenti. Inoltre i campi che differiscono non vengono spediti interamente ma come incrementi dei valori precedenti. Queste ottimizzazioni sono descritte nell'RFC 1144. Sebbene sia ancora ampiamente usato, SLIP presenta alcuni gravi problemi. Primo, non fornisce alcuna gestione degli errori o loro correzione, quindi è compito dei livelli più alti quello di recuperare pacchetti danneggiati, persi o combinati.

Secondo, SLIP supporta solo IP. Con il crescere di Internet fino a comprendere reti che non usano IP come linguaggio nativo (come le LAN Novell), questa restrizione sta diventando troppo pesante.

Terzo, ogni parte deve sapere in anticipo l'indirizzo IP della controparte; nessun indirizzo può essere assegnato dinamicamente durante la connessione. Data l'attuale limitatezza degli indirizzi IP, non è possibile assegnare ad ogni utente Internet da casa un indirizzo IP unico.

Quarto, SLIP non fornisce alcuna forma di autenticazione, quindi nessuna delle due parti sa esattamente con chi sta parlando. Con linee dedicate, questo non è un problema, ma con linee telefoniche lo è.

Quinto, SLIP non è uno standard Internet approvato, per cui esistono molte versioni differenti (e incompatibili). Questa situazione non rende molto semplice l'intercomunicazione.

PPP – protocollo punto-a-punto

Per migliorare la situazione, l'IETF ha costituito un gruppo per ideare un protocollo data link per linee punto-a-punto che risolvesse tutti questi problemi e che potesse divenire uno standard ufficiale Internet. Questo lavoro è culminato nel **PPP (protocollo punto-a-punto)**, che viene definito in RFC 1661 e quindi elaborato in molti altri RFC (RFC 1662 e 1663). PPP gestisce il riconoscimento degli errori, supporta molti protocolli, permette che gli indirizzi IP siano stabiliti al momento della connessione, permette l'autenticazione e fornisce molte altre migliorie rispetto a SLIP. Sebbene molti fornitori Internet offrano tuttora sia SLIP che PPP, il futuro chiaramente va verso PPP, non solo per le linee telefoniche ma anche per le linee dedicate router-router.

PPP fornisce tre vantaggi:

1. Un metodo di suddivisione di frame che definisce in modo non ambiguo la fine di un frame e l'inizio del successivo. Il formato del frame permette anche il rilevamento degli errori.
2. Un protocollo di controllo di collegamento per agganciare linee, testarle, negoziare opzioni e ricederle educatamente quando non siano più necessarie. Questo protocollo è chiamato **LCP (Link Control Protocol** – protocollo di controllo del collegamento).
3. Un modo di negoziare opzioni del livello rete in modo che sia indipendente dal protocollo da utilizzarsi per il livello rete. Il metodo scelto consiste nell'avere un diverso **NCP (Network Control Protocol** – protocollo di controllo di rete) per ogni livello rete supportato.

Per vedere come queste componenti si amalgamano, consideriamo lo scenario tipico di un utente a casa che chiama un fornitore Internet per fare del suo PC un host Internet temporaneo. Prima il PC chiama il router del fornitore via modem. Quindi, dopo che il modem del router ha risposto 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 relative risposte selezionano i parametri PPP da utilizzare.

Una volta accordatisi, viene inviata una serie di pacchetti NCP per configurare il livello rete. Tipicamente il PC vuole fare girare un insieme di protocolli TCP/IP, quindi necessita di un indirizzo IP. Non esistono abbastanza indirizzi IP da assegnare, così ogni fornitore

Internet se ne accaparra un gruppo e ne assegna dinamicamente uno, non appena un PC si connette e per tutta la durata della sua sessione di login. Se un fornitore possiede n indirizzi IP, potrà avere fino a n macchine simultaneamente connesse, ma il numero base dei suoi clienti potrà essere molto più alto. L'NCP per IP viene utilizzato per assegnare indirizzi IP.

A questo punto, il PC è un host Internet e può inviare e ricevere pacchetti IP proprio come se fosse un host connesso fisicamente. Quando un utente termina NCP viene utilizzato per sganciare la connessione del livello rete e liberare l'indirizzo IP. Quindi LCP sgancia la connessione del livello data link. Infine, il computer ordina al modem di liberare la linea telefonica rilasciando la connessione del livello fisico.

Il formato dei frame PPP è stato scelto per essere simile al formato dei frame HDLC, poiché non c'erano ragioni di "reinventare la ruota". La maggiore differenza tra PPP e HDLC è che il primo è orientato al carattere piuttosto che al bit. In particolare PPP, come SLIP, usa rimpiazzamento di caratteri su linee telefoniche via modem, quindi tutti i frame sono composti da un numero intero di byte. Non è possibile spedire un frame composto di 30,25 byte come in HDLC. I frame PPP possono essere spediti su linee telefoniche ma possono anche essere inviati su SONET o linee HDLC orientate al bit (ad es. per una connessione router-router). Il formato dei frame PPP è mostrato in figura 3-27.

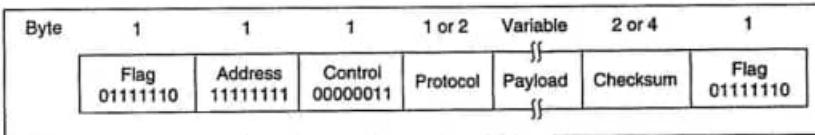


Fig. 3-27 Il formato completo di frame PPP per operazioni in modo non numerato.

Tutti i frame PPP iniziano con il byte standard HDLC (01111110), che viene rimpiazzato se si trova nel campo *Payload*. Il campo successivo è *Address*, che è sempre 11111111 a indicare che tutte le stazioni devono accettare il frame. Usando questo valore si evita il problema di dover assegnare indirizzi data link.

Il campo *Address* è seguito dal campo *Control*, il cui valore base è 00000011. Questo valore indica un frame non numerato. In altre parole, di solito PPP non fornisce trasmissione affidabile usando numeri progressivi e ack. In ambienti rumorosi, come le reti senza filo, può essere utilizzata una trasmissione affidabile che usa i numeri. Tutti i dettagli sono definiti in RFC 1663.

Poiché i campi *Address* e *Control* sono sempre costanti nella configurazione di base, LCP fornisce il meccanismo necessario perché le due parti possano negoziare un'opzione per omettere del tutto entrambi, risparmiando 2 byte per frame.

Il quarto campo PPP è il campo *Protocol*. Il suo compito è quello di indicare il tipo di pacchetto che si trova nel campo *Payload*. Sono definiti codici per LCP, NCP, IP, IPX, AppleTalk e altri protocolli. I protocolli che iniziano con un bit 0 sono protocolli di livello rete come IP, IPX, OSI, CLNP, XNS. quelli che iniziano con un bit 1 vengono usati per negoziare altri protocolli. Questi includono LCP e un NCP diverso per ogni protocollo di

livello rete supportato. La dimensione base del campo *Protocol* è di 2 byte ma può essere negoziato che sia di un 1 byte usando LCP.

Il campo *Payload* è di lunghezza variabile fino a una massima dimensione negoziata. Se la lunghezza non viene negoziata usando LCP durante l'agganciamento della linea, viene utilizzata una lunghezza base di 1500 byte. Se necessario, un riempimento può seguire il carico utile (payload).

Dopo il campo *Payload* c'è il campo *CHECKSUM* che di solito è di 2 byte, ma può essere negoziata una checksum di 4 byte.

Concludendo, PPP è un meccanismo di suddivisione di frame multiprotocollo adatto per essere usato con modem, linee seriali di bit HDLC, SONET e altri livelli fisici. Supporta rilevamento di errori, negoziazione di opzioni, compressione di header e, cosa opzionale, trasmissione affidabile, usando la suddivisione di frame HDLC.

Passiamo ora dal formato dei frame PPP, al modo il cui le linee vengono agganciate e rilasciate. Il diagramma (semplificato) della figura 3-28 mostra le fasi dell'agganciamento della linea, del suo utilizzo e del suo rilascio. Questa sequenza si applica sia alle connessioni via modem che a quelle router-router.

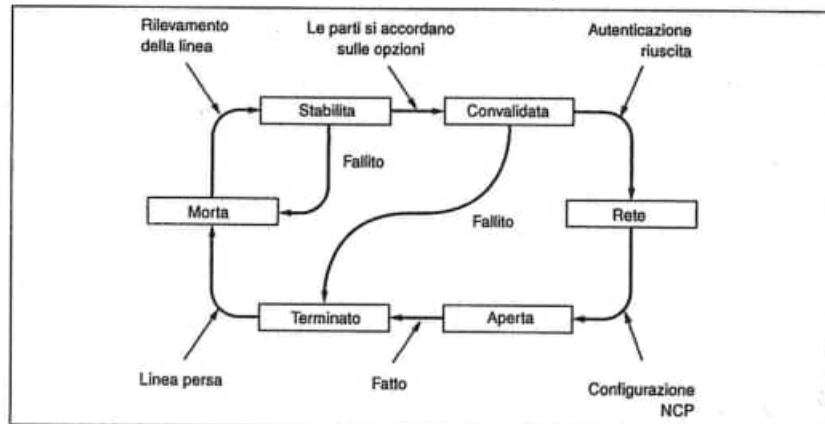


Fig. 3-28 Un diagramma semplificato della fase di agganciamento e rilascio della linea.

Quando la linea è morta (*DEAD*), non c'è collegamento fisico e non esiste una connessione al livello fisico. Dopo che si è stabilita la connessione fisica, la linea diventa *ESTABLISHED* (stabilità). A questo punto inizia la negoziazione delle opzioni LCP, e se ha successo la linea è convalidata (*AUTHENTICATE*). Ora le due parti possono, se vogliono, controllare le relative identità. Quando inizia la fase *NETWORK* (rete), viene invocato l'appropriato protocollo NCP per configurare il livello rete. Se la configurazione ha successo, si arriva allo stato *OPEN* (aperto) e la trasmissione dati può avere luogo. Quando la trasmissione è terminata lo stato diventa *TERMINATE* (terminato) e da qui si passa ancora a *DEAD* se la connessione fisica viene rilasciata.

anche gli errori di bit multipli. Se assumiamo che la probabilità di un errore di bit singolo sia di 10^{-8} , allora la probabilità che una cella contenga un errore di bit multiplo dell'intestazione rilevabile è di 10^{-13} . La probabilità che una cella passi con un errore non rilevato nell'intestazione è di circa 10^{-20} , il che significa che a una velocità OC-3, una cella errata riesce a passare non rilevata una volta ogni 90.000 anni. Sebbene questo sembri un margine molto ampio, quando sulla Terra ci saranno, diciamo, 1.000.000.000 di telefoni ATM, ognuno usato per il 10% del tempo, ogni anno passeranno non scoperte più di 1000 celle di intestazione errate.

Per le applicazioni a cui occorre una trasmissione affidabile al livello data link, Shacham e McKenna (1990) hanno sviluppato una tecnica con la quale su una sequenza di celle consecutive viene calcolata la disgiunzione esclusiva (EXCLUSIVE OR) e il risultato viene appeso alla sequenza come una intera cella. Se una cella viene persa o danneggiata potrà essere ricostruita mediante l'informazione aggiunta.

Una volta generato l'HEC e inserito nell'intestazione della cella, questa è pronta per la trasmissione. La trasmissione può avvenire in modo sincrono o asincrono. Quando viene utilizzato un mezzo asincrono una cella può essere spedita appena è pronta; non ci sono restrizioni di tempo.

Con un mezzo sincrono le celle devono essere trasmesse in accordo con i criteri temporali prestabiliti. Se non ci sono celle di dati disponibili quando servirebbero, il sottostato TC deve inventarne. Queste sono dette **celle oziose (idlecells)**.

Altre celle senza dati sono le **OAM (Operation And Maintenance** – di operazione e manutenzione). Le celle OAM sono utilizzate anche dai commutatori ATM per scambiarsi controlli e altre informazioni necessarie per mantenere il sistema in funzione. Le celle OAM hanno anche altre funzioni speciali. Per esempio, la velocità di 155,52 Mbps OC-3 combacia con l'abbondante frequenza di dati di SONET, ma un frame STM-1 ha un totale di 10 colonne complessive su 270, quindi il carico di SONET è solo di $260/270 \times 155,52$ Mbps o 149,76 Mbps. Per evitare di travolgere SONET, una sorgente ATM che usi SONET dovrebbe emettere una cella OAM ogni 27 celle, per diminuire la frequenza di dati fino a $26/27$ di 155,52 Mbps, combaciando così perfettamente con SONET. Il far combaciare la frequenza di output ATM con la frequenza del sistema di trasmissione sottostante è un importante compito del sottostato TC.

Dalla parte del ricevente, le celle non usate (idle) sono elaborate nel sottostato TC, le celle OAM sono invece passate al livello ATM. Le celle OAM si distinguono dalle celle di dati perché hanno i primi tre byte dell'intestazione a 0, cosa non permessa nelle celle di dati. Il quarto byte descrive la natura della cella OAM.

Un altro compito importante del sottostato TC è quello di generare l'informazione sulla suddivisione dei frame per il sistema di trasmissione sottostante. Per esempio, una videocamera ATM può produrre una sequenza di celle sul cavo, ma può anche produrre frame SONET con celle ATM inglobate nel carico (payload) dei frame SONET. In quest'ultimo caso, il sottostato TC genererà la suddivisione in frame SONET e ingloberà nei frame le celle ATM; questo non è un compito semplice infatti un carico SONET non può contenere un numero intero di celle di 53 byte.

Anche se le società telefoniche intendono chiaramente usare SONET come infrastruttura di trasmissione per ATM, sono state definite alcune associazioni tra ATM e i campi

Payload di altri tipi di sistemi, e si sta lavorando su alcune altre. In particolare, esistono anche associazioni con T1, T3 e FDDI. -

Ricezione di una cella

Il compito del sottostato TC in termini di output è quello di prendere una sequenza di celle, aggiungere a ognuna un HEC, convertire il risultato in un flusso di bit e adattare il flusso di bit alla velocità del sistema di trasmissione fisico sottostante inserendo alcune celle OAM di riempimento. In termini di input il sottostato TC esegue esattamente le azioni inverse. Prende in input un flusso di bit, determina gli estremi delle celle, verifica gli header (scartando le celle con header non valido), elabora le celle OAM e passa le celle di dati al livello ATM soprastante.

La parte più complicata è quella di definire gli estremi delle celle sul flusso entrante di bit. A livello di bit, una cella è una sequenza di $53 \times 8 = 424$ bit. Non è presente nessun byte di flag del tipo 0111110 per marcare l'inizio e la fine di una cella, come in HDLC. Infatti non c'è alcun demarcatore. Come si possono riconoscere gli estremi di una cella in queste condizioni?

In alcuni casi viene in aiuto il livello fisico sottostante. Con SONET, per esempio, le celle sono allineate in modo sincrono con il guscio payload, quindi il puntatore SPE nell'intestazione SONET indica l'inizio della prima cella piena. Tuttavia, alcune volte il livello fisico non offre alcun aiuto.

In questi casi il trucco consiste nell'utilizzare HEC. Quando arrivano i bit, il sottostato TC mantiene un registro di shift di 40 bit, con i bit che entrano a sinistra ed escono a destra. TC analizza quindi i 40 bit per vedere se c'è un'intestazione valida di cella. Se la trova, gli 8 bit più a destra saranno un HEC valido per i 32 bit più a sinistra. Se questa condizione non è valida, il buffer non conterrà una cella valida e i bit del buffer verranno spostati (shifted) a destra di 1 bit, causando la scomparsa del primo bit a destra e l'apparizione di un nuovo bit nella posizione più a sinistra. Questa tecnica è ripetuta finché non viene individuato un HEC valido. A questo punto, gli estremi della cella vengono riconosciuti poiché il registro di shift contiene un header valido.

Il problema di questa euristica è che l'HEC è di soli 8 bit. Per ogni possibile registro di shift, anche uno contenente bit casuali, la probabilità di trovare un HEC valido è di 1/256, un valore piuttosto alto. Usata da sola, questa procedura individuerebbe header di cella troppo spesso.

Per migliorare la precisione dell'algoritmo di riconoscimento, viene utilizzata una macchina a stati finiti come quella in figura 3-30. Essa consiste di tre stati: **HUNT** (caccia), **PRESYNCH** e **SYNCH**. Allo stato **HUNT**, il sottostato TC sta spostando i bit all'interno del registro di shift, uno alla volta, cercando un header valido. Quando ne trova uno, la macchina a stati finiti cambia stato e va in **PRESYNCH**, indicando gli ipotetici estremi di una cella. Quindi sposta i successivi 424 bit (53 byte) senza esaminarli. Se la sua ipotesi sugli estremi fosse valida, il registro di shift dovrebbe contenere ora un altro header di cella valido, nel qual caso l'algoritmo di HEC dovrebbe iniziare. Se l'HEC è scorretto, TC ritorna allo stato **HUNT** e continua la ricerca bit per bit di un header con HEC valido. D'altra parte anche se il secondo HEC è corretto, TC sposta di altri 424 bit e riprova ancora. Continua quindi esaminando gli header in questo modo finché non trova 8 header

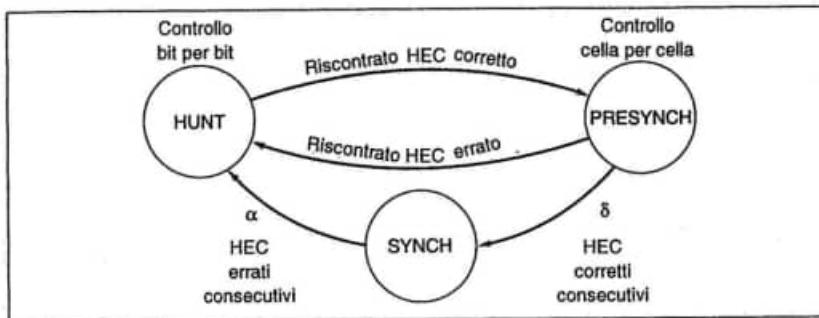


Fig. 3-30 L'euristica di demarcazione della cella.

corretti su una riga, dopodiché assume di essere sincronizzato e passa allo stato *SYNCH* per iniziare le normali operazioni. Si noti che la probabilità di entrare in uno stato *SYNCH* per caso con una stringa di bit è di 2^{-85} , che può essere resa molto piccola se si sceglie un δ abbastanza grande. Il prezzo pagato per un δ grande è tuttavia un lungo tempo di sincronizzazione.

Inoltre per risincronizzarsi dopo avere perso la sincronizzazione (o all'inizio), il sottostato TC necessita di una euristica per determinare quando ha perso la sincronizzazione, per esempio dopo che un bit è stato inserito o cancellato da un flusso di bit. Non sarebbe saggio abbandonare se si trova un HEC scorretto, poiché la maggior parte degli errori consistono in inversione di bit, cancellazione o inserimento. Il comportamento più saggio è quello di scartare la cella con un header errato e sperare che la prossima sia corretta. Se per caso α HEC sono errati, il sottostato TC deve concludere di avere perso la sincronizzazione e deve ritornare nello stato *HUNT*.

Nonostante sia difficile ammetterlo, può darsi che un utente malizioso possa tentare di ingannare il sottostato TC inserendo una sequenza di dati che imita l'algoritmo di HEC nel campo *Payload* di diverse celle consecutive. In questo caso, se si perde la sincronizzazione, può darsi che la si riprenda nel punto sbagliato. Per evitare questo, i bit di payload sono rimescolati prima della trasmissione e riordinati dopo.

Prima di abbandonare il sottostato TC occorre fare un commento. Il meccanismo scelto per delineare i limiti delle celle ha bisogno che il sottostato TC comprenda e utilizzi l'intestazione del livello ATM sovrastante. Un livello che capisce e interpreta un'intestazione di un livello sovrastante è una violazione dei principi di base dell'ingegneria dei protocolli. L'idea di avere diversi protocolli stratificati è quella di rendere ogni livello indipendente da quelli sovrastanti. Dovrebbe essere possibile, per esempio, cambiare il formato dell'intestazione del livello ATM senza ritoccare il sottostato TC. Tuttavia, a causa del modo in cui le celle vengono delimitate, questo cambiamento non è possibile.

3.7 Riassunto

Il compito del livello data link è quello di convertire un flusso di bit offerto dal livello

fisico in un flusso di pacchetti per il livello rete. Vengono usati vari metodi di suddivisione dei pacchetti, inclusi il conteggio e il rimpiazzamento di caratteri e il rimpiazzamento di bit. I protocolli data link possono fornire metodi di controllo degli errori per ritrasmettere pacchetti danneggiati o persi. Per impedire che un mittente troppo rapido travolga un ricevente più lento, il protocollo di livello data link può prevedere controllo di flusso. Il meccanismo di sliding window è ampiamente usato per integrare in modo opportuno il controllo degli errori e il controllo di flusso.

I protocolli sliding window possono essere classificati mediante la dimensione della finestra del mittente e del ricevente. Quando entrambe hanno misura 1, il protocollo è detto stop-and-wait. Quando la finestra del mittente è maggiore di 1, per esempio per impedire che il mittente si blochi su un circuito con un ritardo di propagazione ampio, il ricevente può essere programmato per scartare tutti i frame diversi dal successivo nella sequenza progressiva (protocollo 5) o per "bufferizzare" tutti i frame non in ordine finché non saranno richiesti (protocollo 6).

I protocolli possono essere modellati mediante diverse tecniche per dimostrare la loro correttezza (o la loro scorrettezza). Per tale scopo sono generalmente usati i modelli a macchina a stati finiti o a reti di Petri.

Molte reti usano protocolli orientati al bit a livello data link: SDLC, HDLC, ADCCP o LAPB. Tutti questi protocolli usano byte per delimitare i pacchetti, e rimpiazzamento di bit per evitare che tali byte di separazione compaiano nei dati. Tutti utilizzano inoltre sliding window per controllo di flusso. La rete Internet utilizza SLIP e PPP come protocolli data link. I sistemi ATM hanno protocolli semplici, che effettuano un mediocre controllo di errore e non hanno controllo di flusso.

Esercizi

1. Un messaggio di livello più alto viene suddiviso in 10 pacchetti, ognuno dei quali ha una probabilità dell'80% di arrivare correttamente a destinazione. Se non si ha controllo di errore da parte del livello data link, quante volte dovrà essere spedito il messaggio in media perché sia ricevuto correttamente?
2. Il seguente frammento di dati si presenta in mezzo a un flusso di dati per cui viene utilizzato il metodo di riempimento di caratteri descritto nel testo: DLE, STX, A, DLE, B, DLE, ETX. Qual è l'output dopo il riempimento?
3. Se la stringa di bit 0111101111101111110 è sottoposta a riempimento di bit, qual è l'output?
4. Quando si utilizza riempimento di bit è possibile che la perdita, l'inserimento o la modifica di un unico bit causi un errore non rilevato dalla checksum? Se no, perché? Se si, come? La lunghezza della checksum è importante in questo caso?
5. Si indichi una circostanza in cui un protocollo a ciclo aperto (ad es. un codice di Hamming) sia preferibile ai protocolli di tipo feedback (riscontro) trattati in questo capitolo.
6. Per aumentare l'affidabilità rispetto a quel che può dare un singolo bit di parità, viene

- utilizzata una tecnica di rivelazione di errore con un bit di parità per controllare tutti i bit di numero dispari, e un altro bit di parità per controllare tutti i bit di numero pari. Qual è la distanza di Hamming di tale codice?
7. Un modo per rilevare errori è quello di trasmettere i dati come blocchi di n righe di k bit e di aggiungere bit di parità per ogni riga e ogni colonna. Questa tecnica rileva tutti gli errori di bit singolo? E gli errori doppi? E quelli tripli?
 8. Un blocco di bit con n righe e k colonne usa bit di parità orizzontali e verticali per rilevare errori. Si supponga che siano stati invertiti esattamente 4 bit a causa di errori di trasmissione. Si derivi un'espressione della probabilità che l'errore non sia rilevato.
 9. Qual è il resto ottenuto dividendo $x^7 + x^5 + 1$ per il generatore polinomiale $x^3 + 1$?
 10. I protocolli data link inseriscono quasi sempre CRC nel trailer (coda), piuttosto che nell'intestazione. Perché?
 11. Un canale ha una velocità di 4 kbps e un ritardo di propagazione di 20 ms. Per quale intervallo di dimensione di frame il protocollo stop-and-wait ha un'efficienza minima del 50%?
 12. Un tronco T1 di 3000 km viene utilizzato per trasmettere frame di 64 byte usando il protocollo 5. Se la velocità di propagazione è di $6 \mu\text{s}/\text{km}$, di quanti bit dovrebbero essere i numeri progressivi?
 13. Si immagini un protocollo sliding window che utilizzi così tanti numeri progressivi da rendere impossibile la sovrapposizione delle estremità. Che relazioni devono sussistere tra i quattro lati della finestra e le sue dimensioni?
 14. Se la procedura *between* del protocollo 5 contenesse la condizione $a \leq b \leq c$ invece di $a \leq b < c$, si avrebbe qualche effetto sull'efficienza o la correttezza del protocollo? Motivare la risposta.
 15. Nel protocollo 6, quando arrivano i frame di dati, viene fatto un controllo per vedere se i numeri progressivi siano diversi da quelli attesi e se *no-nak* sia vero. Se entrambe le condizioni sono valide, viene spedito un NAK. Altrimenti viene inizializzato un timer ausiliario. Si supponga che la clausola *else* sia omessa. Avrebbe qualche influenza sulla correttezza del protocollo?
 16. Si supponga di cancellare dal codice il ciclo while di tre righe alla fine del protocollo 6. Si avrebbe qualche riscontro a livello di correttezza del protocollo o solo a livello di prestazioni? Motivare la risposta.
 17. Si supponga che il caso riguardante gli errori di checksum sia rimosso dal blocco switch del protocollo 6. In che modo questa modifica influirebbe sul comportamento del protocollo?
 18. Nel protocollo 6 il codice di *frame-arrival* ha una sezione usata per i NAK. Questa

- sezione viene invocata se il frame arrivato è un NAK e se vale un'altra condizione. Si dia uno scenario in cui la presenza di quest'altra condizione è fondamentale.
19. Immaginate di scrivere il software del livello data link per una linea usata per inviare dati a voi ma non da voi. L'altra parte utilizza HDLC con numeri progressivi di 3 bit e una finestra di 7 frame di dimensione. Voi vorreste "bufferizzare" quanti più frame possibile per aumentare l'efficienza, ma non potete modificare il software dalla parte del mittente. È possibile avere una finestra ricevente più grande di 1 e garantire comunque che il protocollo non fallisca? Se sì, qual è la più grande dimensione di finestra utilizzabile senza che il protocollo fallisca?
 20. Si consideri il comportamento del protocollo 6 su di una linea di 1 Mbps esente da errore. La dimensione massima del frame è di 1000 bit. Nuovi pacchetti sono prodotti ogni secondo. L'intervallo di timeout è di 10 ms. Se venisse eliminato il timer speciale di ack, si avrebbero dei timeout non necessari. Quante volte verrebbe ritrasmesso in media un messaggio?
 21. Nel protocollo 6 $\text{MAX-SEQ} = 2^n - 1$. Sebbene questa condizione sia ovviamente desiderabile per rendere efficiente l'utilizzo dei bit dell'intestazione, non si è dimostrato che sia essenziale. Per esempio, il protocollo funziona correttamente per $\text{MAX-SEQ} = 4$?
 22. Frame di 1000 bit vengono spediti su un canale satellitare di 1 Mbps. Gli ack vengono sempre agganciati con piggybacking ai frame di dati. Gli header sono molto corti. Si utilizzano numeri progressivi di 3 bit. Qual è il massimo utilizzo del canale raggiungibile con:
 - a) Stop-and-wait.
 - b) Protocollo 5.
 - c) Protocollo 6.
 23. Calcolare la frazione di larghezza di banda che viene sprecata per l'appesantimento (header e ritrasmissioni) per il protocollo 6 su un canale satellitare molto trafficato di 50 kbps, con frame di dati composti da 40 bit di intestazione e 3960 bit di dati. Non si hanno mai frame di ack. I frame NAK sono di 40 bit. Il tasso di errore per i frame di dati è dell'1% e il tasso di errore per i frame NAK è trascurabile. I numeri progressivi sono di 8 bit.
 24. Si consideri un canale satellitare esente da errore di 64 kbps utilizzato per spedire frame di dati di 512 byte in una direzione, con riscontri corti che viaggiano in direzione opposta. Qual è la produttività nel caso di una finestra di dimensioni 1, 7, 15 e 127?
 25. Un cavo di 100 km ha una banda T1. La velocità di propagazione del cavo è $2/3$ della velocità della luce. Quanti bit possono stare nel cavo?
 26. Si ridisegni la figura 3-21 per un canale full-duplex esente da perdita di frame. È ancora possibile il fallimento del protocollo?

27. Si calcoli la sequenza di firing della rete di Petri in figura 3-23 corrispondente alla sequenza di stati (000), (01A), (01-), (010), (01A) in figura 3-20. Si spieghi cosa rappresenta la sequenza.
28. Date le regole di transizione $AC \rightarrow B$, $B \rightarrow AC$, $CD \rightarrow E$, ed $E \rightarrow CD$, si disegni la rete di Petri descritta. Dalla rete di Petri, si disegni poi il grafo a stati finiti raggiungibile dallo stato iniziale ACD . Quale concetto molto noto della scienza dei computer modella queste transizioni?
29. PPP è basato in parte su HDLC, che utilizza riempimento di bit per evitare che eventuali byte indicatori compaiano nel campo *Payload* causando confusione. Si dia almeno una motivazione del perché invece PPP utilizza riempimento di caratteri.
30. Qual è l'appesantimento minimo della spedizione di un pacchetto IP con PPP? Si consideri solo l'appesantimento introdotto da PPP, non quello dell'intestazione IP.
31. Si consideri l'euristica per la demarcazione dei limiti delle celle ATM con $\alpha = 5$, $\delta = 6$ e tasso di errore per bit di 10^{-5} . Una volta che il sistema è sincronizzato, quanto riesce a rimanere tale, nonostante gli eventuali errori di bit nell'intestazione? Si assuma che la linea sia OC-3.
32. Si scriva un programma per simulare stocasticamente il comportamento di una rete di Petri. Il programma dovrebbe leggere in input le regole di transizione e una lista di stati corrispondente al livello rete che spedisca un nuovo pacchetto o che accetti un nuovo pacchetto. Dallo stato iniziale, preso anch'esso in input, il programma dovrebbe in modo casuale scegliere transizioni abilitate e farle scattare, controllando se un host accetti due messaggi senza che un altro host ne emetta uno nuovo nel frattempo.

Capitolo 4

IL SOTTOLIVELLO DI ACCESSO AL MEZZO

Come abbiamo visto nel capitolo 1, le reti possono essere suddivise in due categorie: quelle che utilizzano connessioni punto-a-punto e quelle che utilizzano canali di broadcast. Questo capitolo illustrerà le reti broadcast e i loro protocolli.

In ogni rete broadcast il problema principale è quello di individuare chi deve usare il canale quando c'è un conflitto per utilizzarlo. Per rendere questo punto più chiaro, si consideri una conferenza telefonica in cui sei persone, su sei differenti telefoni, siano interconnesse in modo che ciascuna possa sentire e parlare con tutte le altre. Facilmente, quando una smettesse di parlare, due o più degli altri provrebbero a parlare contemporaneamente generando caos. In un incontro faccia faccia, la confusione viene evitata tramite mezzi esterni: per esempio, in una assemblea, le persone alzano la mano per richiedere il permesso di parlare. Quando è disponibile un solo canale, è molto più arduo determinare a chi si deve dare la precedenza. Sono noti molti protocolli per risolvere questo problema ed essi sono materia di questo capitolo. Nella letteratura, i canali di broadcast vengono definiti **canali multiaccesso** oppure **canali ad accesso casuale**.

I protocolli utilizzati per determinare a chi spetta la precedenza su un canale multiaccesso appartengono a un sottolivello del livello data link chiamato sottolivello **MAC** (**M**edium **A**ccess **C**ontrol-controllo di accesso al mezzo). Il sottolivello MAC è particolarmente importante nelle LAN, che usano spesso un canale multiaccesso come base per la loro comunicazione. Le WAN al contrario utilizzano collegamenti punto a punto, a eccezione delle reti satellitari. Poiché i canali multiaccesso e le LAN sono in così stretta relazione, in questo capitolo discuteremo delle LAN in generale, come delle reti satellitari e di altre reti di broadcast.

Tecnicamente, il sottolivello MAC è la parte inferiore del livello data link, quindi logicamente avremmo dovuto studiarlo prima di esaminare tutti i protocolli punto-a-punto del capitolo 3. Tuttavia, per molte persone è più facile comprendere protocolli che coinvolgono parti multiple dopo aver studiato protocolli che coinvolgono due sole parti. Per questo motivo si è preferito deviare lievemente dall'ordinamento strettamente dal basso all'alto della presentazione.

4.1 I problemi di allocazione del canale

Il tema centrale di questo capitolo è quello di capire come allocare un singolo canale di broadcast tra diversi utenti in competizione. Per prima cosa considereremo le tecniche di allocazione statica e dinamica in generale. Poi esamineremo alcuni algoritmi specifici.

4.1.1 Allocazione statica di canali nelle LAN e nella WAN

Il metodo tradizionale per allocare un singolo canale, come una dorsale telefonica, tra più utenti in competizione è FDM (Frequency Division Multiplexing-multiplexing basato sulla suddivisione di frequenza). Se si hanno N utenti, la larghezza di banda viene suddivisa in N parti di dimensione uguale (si veda la figura 2-24), assegnando ad ogni utente una parte. Poiché ogni utente ha una diversa banda di frequenza, non c'è interferenza. Quando si ha un numero piccolo e fisso di utenti, ciascuno con un carico di traffico pesante ("bufferizzato"), per esempio gli uffici di commutazione della società dei telefoni, FDM è un meccanismo semplice ed efficiente.

Tuttavia, quando il numero dei mittenti è grande e varia continuamente, o il traffico è irregolare, FDM inizia a presentare alcuni problemi. Se lo spettro viene suddiviso in N zone e meno di N utenti sono interessati a comunicare, verrà sprecata un'ampia porzione dello spettro. Se più di N utenti vogliono comunicare, alcuni di essi si vedranno negare il permesso di farlo per mancanza di banda, anche se alcuni degli utenti a cui è stata assegnata una frequenza trasmettono o ricevono raramente.

Anche assumendo che il numero degli utenti possa in qualche modo essere tenuto costantemente a N , dividere l'unico canale disponibile in sottocanali statici è una cosa fondamentalmente inefficiente. Il problema principale è che quando alcuni utenti sono inattivi, la loro larghezza di banda viene sprecata. Essi non la utilizzano e nessun altro può utilizzarla al loro posto. Inoltre, nella maggior parte dei sistemi di computer, il traffico dati è molto irregolare (le proporzioni tra il traffico massimo e quello medio sono spesso di 1000:1). Di conseguenza la maggior parte dei canali resterà inattiva per la maggior parte del tempo.

Le basse prestazioni di FDM statico possono essere facilmente osservate basandosi su un semplice calcolo di teoria delle code. Iniziamo assumendo un ritardo medio, T ; per un canale di capacità C bps, con una frequenza di arrivo di λ frame/s; ogni pacchetto ha una lunghezza ricavata da una funzione di densità di probabilità esponenziale con una media di $1/\mu$ bit/frame:

$$T = \frac{1}{\mu C - \lambda}$$

Suddividiamo ora il canale singolo in N sottocanali indipendenti, ognuno con capacità C/N bps. Il grado di input medio su ogni sottocanale sarà di λ/N . Ricalcolando T abbiamo:

$$T_{FDM} = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{n}{\mu C - \lambda} = NT \quad (4-1)$$

Il ritardo medio usando FDM è N volte peggiore che nel caso in cui i frame fossero magicamente ordinati in una grossa coda centrale.

Precisamente gli stessi argomenti che si applicano a FDM si applicano anche al multiplexing a divisione di tempo (TDM). Ad ogni utente viene allocato staticamente un intervallo temporale N -esimo. Se l'utente non utilizza l'intervallo allocato, esso rimane

4.1 I problemi di allocazione del canale

inutilizzato. Poiché nessuno dei metodi tradizionali di allocazione statica dei canali sembra funzionare con traffico irregolare, considereremo ora i metodi dinamici.

4.1.2 Allocazione dinamica di canali nelle LAN e nelle WAN

Prima di introdurre il primo dei diversi metodi di allocazione dinamica discussi in questo capitolo, è importante formulare precisamente il problema dell'allocazione. Cinque assunzioni sono alla base di tutto:

- Modello a stazioni.** Il modello consiste di N stazioni indipendenti (computer, telefoni, strumenti di comunicazione personale ecc.), ognuna dipendente da un programma o da un utente che generi frame da trasmettere. La probabilità che un pacchetto sia generato in un intervallo di lunghezza Δt è di $\lambda \Delta t$, dove λ è una costante (la frequenza di arrivo di un nuovo pacchetto). Una volta generato il pacchetto, la stazione è bloccata e resta inattiva finché il pacchetto non è stato completamente trasmesso.
- Assunzione di canale singolo.** È disponibile un singolo canale per la comunicazione. Tutte le stazioni possono trasmettere su di esso e tutte possono ricevere da esso. Per quanto riguarda l'aspetto hardware tutte le stazioni sono equivalenti, sebbene il software del protocollo possa assegnare diverse priorità.
- Assunzione di collisione.** Se due frame vengono trasmessi simultaneamente, essi si sovrapporranno temporalmente e il segnale risultante sarà confuso. Questo evento viene detto **collisione**. Tutte le stazioni possono rilevare le collisioni. Un pacchetto su cui si è avuta collisione dovrà essere ritrasmesso successivamente. Non ci sono altri errori oltre quelli generati dalle collisioni.
- a) **Tempo continuo.** La trasmissione di frame può iniziare in qualsiasi istante; non c'è un orologio globale che suddivide il tempo in intervalli discreti.
b) **Tempo discreto.** Il tempo è suddiviso in intervalli discreti (slot). Le trasmissioni dei frame cominciano sempre all'inizio di uno slot. Uno slot può contenere 0, 1 o più frame, cioè rispettivamente uno slot ozioso, una trasmissione con successo o una collisione.
- a) **Rilevamento della portante.** Le stazioni possono sapere se il canale è utilizzato prima di tentare di utilizzarlo. Se il canale è impegnato, nessuna stazione proverà a utilizzarlo finché non si rivelerà inattivo.
b) **Senza rilevamento della portante.** Le stazioni non sono in grado di "testare" il canale prima di tentare di usarlo; esse semplicemente trasmettono. Solo dopo possono determinare se la trasmissione è andata a buon fine.

È necessario discutere un po' queste assunzioni. La prima asserisce che le stazioni sono indipendenti e che il lavoro è prodotto con una frequenza costante. Assume anche implicitamente che ogni stazione dipenda da un singolo utente o programma, quindi, quando la stazione è bloccata non viene svolto altro lavoro. Alcuni modelli più sofisticati prevedono stazioni multiprogrammabili che possono generare lavoro anche quando una stazione è bloccata ma l'analisi di queste stazioni è molto più complessa.

L'assunzione di canale singolo è un'ipotesi fondamentale. Non ci sono modi esterni di comunicazione. Le stazioni non possono "alzare la mano per richiedere che l'insegnante le faccia parlare".

L'assunzione di collisione è altrettanto basilare, sebbene in alcuni sistemi (principalmente in quelli a spettro diffuso), questa assunzione venga trascurata, con risultati sorprendenti. Anche alcune LAN, come le token ring, usano meccanismi di eliminazione dei conflitti che evitano le collisioni.

Ci sono due assunzioni alternative riguardo al tempo. Esso può essere continuo o discreto. Alcuni sistemi lo considerano in un modo, altri nell'altro, quindi li discuteremo e analizzeremo entrambi. Ovviamente, per un dato sistema, vale solo una delle due possibili assunzioni.

In modo analogo, una rete può rilevare la portante oppure no. Le LAN solitamente rilevano la portante, le reti satellitari no (a causa del lungo ritardo di propagazione). Le stazioni in una rete con rilevazione di portante possono terminare le trasmissioni prematuramente se scoprono di collidere con un'altra trasmissione. Si noti che la parola portante in questo contesto si riferisce a un segnale elettrico sul cavo.

4.2 Protocolli di accesso multiplo

Sono noti molti algoritmi per l'allocazione di un canale ad accesso multiplo. Nel seguito studieremo un campione rappresentativo tra quelli più interessanti e daremo alcuni esempi del loro utilizzo.

4.2.1 ALOHA

Negli anni settanta, Norman Abramson e i suoi colleghi dell'Università delle Hawaii idearono un metodo nuovo ed elegante per risolvere il problema dell'allocazione di un canale. Da allora il loro lavoro è stato ampliato da molti ricercatori (Abramson, 1985). Sebbene il lavoro di Abramson, chiamato sistema ALOHA, usasse broadcasting via radio tra stazioni a terra, l'idea base è applicabile ad ogni sistema in cui utenti indipendenti competano per l'uso di un unico canale condiviso.

Discuteremo qui due versioni di ALOHA: puro e a slot. Esse differiscono nella gestione del tempo che non viene suddiviso (ALOHA puro), oppure in intervalli discreti in cui devono stare i frame (ALOHA a slot). ALOHA puro infatti non richiede sincronizzazione sul tempo globale, ALOHA a slot sì.

ALOHA puro

L'idea base di un sistema ALOHA è semplice: gli utenti possono trasmettere ogni volta che hanno dati da spedire. Ovviamente ci saranno collisioni e pacchetti in collisione che verranno distrutti. Tuttavia, per la proprietà di retroazione del broadcasting, un mittente potrà sempre determinare se il suo pacchetto sia stato distrutto ascoltando il canale, come faranno tutti gli altri utenti. Nel caso di una LAN, la retroazione sarà immediata: con un satellite, ci sarà un ritardo di 270 ms prima che il mittente possa sapere se la trasmissione abbia avuto successo. Se il pacchetto è stato distrutto, il mittente aspetta una quantità di tempo casuale e lo rispedisce. I tempi di attesa devono essere casuali altrimenti si avreb-

4.2 Protocolli di accesso multiplo

berò sempre le stesse collisioni e si avrebbe una situazione di stallo. I sistemi in cui si possono avere collisioni perché più utenti condividono lo stesso canale, sono detti sistemi di contesa.

Un esempio di generazione di pacchetti in un sistema ALOHA è mostrato in figura 4-1. I frame sono definiti tutti della stessa lunghezza poiché la produttività del sistema ALOHA è massima quando i pacchetti sono di dimensione uniforme piuttosto che di lunghezza variabile.

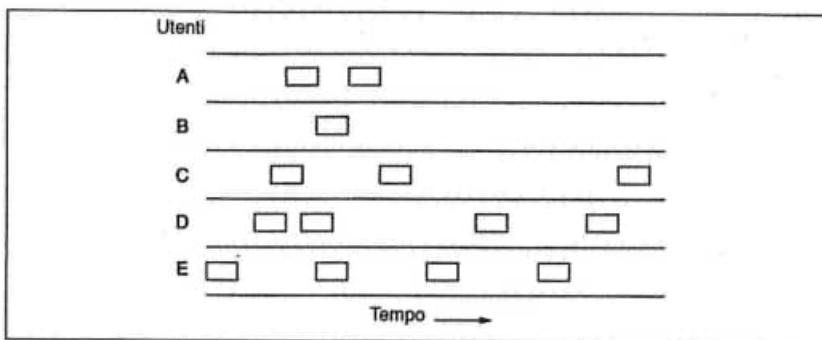


Fig. 4-1 In ALOHA puro i pacchetti vengono trasmessi in momenti arbitrari.

Quando due pacchetti tentano di occupare il canale nello stesso istante, si ha una collisione ed entrambi vanno persi. Se il primo bit di un nuovo pacchetto si sovrappone anche solo con l'ultimo del pacchetto precedente, entrambi saranno totalmente distrutti e dovranno essere ritrasmessi successivamente. La checksum non può (e non deve) distinguere una perdita totale da una quasi riuscita: quel che è corrotto è corrotto.

Una domanda più interessante: qual è l'efficienza di un canale ALOHA? Cioè, quale frazione di tutti i pacchetti spediti evita le collisioni in queste circostanze caotiche? Consideriamo un insieme infinito di utenti interattivi seduti davanti ai loro computer (stazioni). Un utente è sempre in uno di questi due stati: o sta scrivendo o sta aspettando. Inizialmente tutti gli utenti stanno scrivendo. Quando finiscono una riga essi smettono di scrivere e aspettano una risposta. Quindi la stazione spedisce un pacchetto contenente la riga e controlla il canale per vedere se la trasmissione abbia avuto successo. Se sì, l'utente leggerà la risposta e continuerà a scrivere. Se no, l'utente continuerà ad attendere e il pacchetto verrà ritrasmesso ancora e ancora finché la trasmissione non avrà successo.

Sia "il tempo di frame" la quantità di tempo necessaria per trasmettere un pacchetto standard di lunghezza fissata (cioè la lunghezza del pacchetto divisa per la velocità dei bit). A questo punto assumiamo che l'insieme infinito di utenti generi nuovi frame in accordo con la distribuzione di Poisson con media di S frame per tempo di frame. (L'assunzione di insieme infinito di utenti è necessaria per assicurare che S non diminuisca se alcuni utenti si bloccano). Se $S > 1$, la comunità di utenti sta generando frame a una velocità maggiore di quella che il canale è in grado di gestire e quasi tutti i frame avranno una collisione. Per una produttività ragionevole si dovrebbe avere $0 < S < 1$.

Oltre ai nuovi frame, le stazioni generano le ritrasmissioni dei frame che avevano subito collisione precedentemente. Assumiamo che la probabilità di k tentativi di trasmissione per tempo di frame, vecchi e nuovi, sia anch'essa di Poisson con media G per tempo di frame. Chiaramente $G \geq S$. Se si ha basso carico ($S = 0$) si avranno poche collisioni, quindi poche ritrasmissioni, quindi $G = S$. Se invece si ha alto carico ci saranno molte collisioni, quindi $G > S$. In ogni caso la produttività sarà uguale al carico, G , per la probabilità che la trasmissione abbia successo: cioè $S = GP_0$, dove P_0 è la probabilità che a un pacchetto non capiti una collisione.

A un pacchetto non capiterà una collisione se nessun altro pacchetto sarà spedito per un tempo di frame dalla sua partenza, come mostrato in figura 4-2. In queste condizioni il pacchetto ombreggiato arriverà indenne? Sia t il tempo richiesto per inviare un pacchetto. Se qualche altro utente ha generato un pacchetto tra il tempo t_0 e $t_0 + t$, l'estremità del pacchetto colliderà con l'inizio del pacchetto ombreggiato. Infatti, il destino del pacchetto ombreggiato era già deciso prima che ne fosse inviato il primo bit, ma poiché in ALOHA puro una stazione non ascolta il canale prima di spedire, non c'era modo di sapere che un altro pacchetto fosse già in viaggio. Analogamente, qualsiasi altro pacchetto spedito fra il tempo $t_0 + t$ e $t_0 + 2t$ colliderà con la fine del pacchetto ombreggiato.

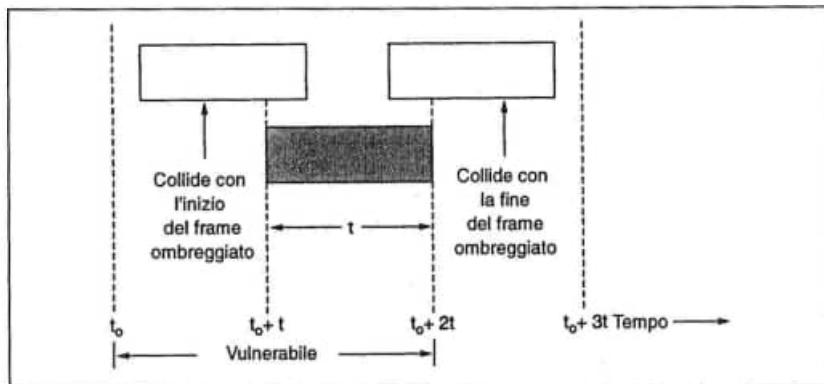


Fig. 4-2 Periodo critico per il pacchetto ombreggiato.

La probabilità che k frame siano generati in un dato tempo di frame è data dalla distribuzione di Poisson:

$$\Pr[k] = \frac{G^k e^{-G}}{k!} \quad (4-2)$$

quindi la probabilità di zero frame è e^{-G} . In un intervallo di due tempi di frame il numero medio di frame generati è $2G$. La probabilità che non inizi altro traffico per tutto il periodo critico è quindi data da $P_0 = e^{-2G}$. Usando $S = GP_0$ abbiamo:

$$S = G e^{-2G}$$

La relazione tra il traffico offerto e il throughput (produttività) è mostrato in figura 4-3. La massima produttività si ha quando $G = 0,5$, con $S = 1/e$, che è circa 0,184. In altre parole, il meglio che si può sperare è un utilizzo del canale del 18%. Un risultato non molto incoraggiante ma dato che ognuno può trasmettere quando vuole, difficilmente potevamo aspettarci il 100%.

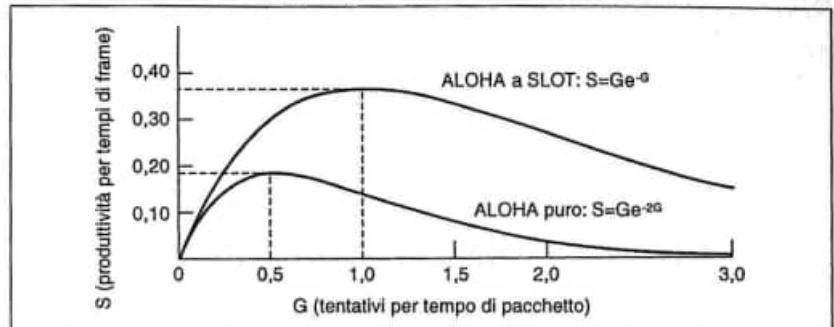


Fig. 4-3 Throughput per traffico offerto in sistemi ALOHA.

ALOHA a slot

Nel 1972, Roberts pubblicò un metodo per duplicare la capacità di un sistema ALOHA (Roberts, 1972). La sua proposta consisteva nel dividere il tempo in intervalli discreti, ognuno corrispondente a un pacchetto. Questo approccio richiede che gli utenti concordino sugli estremi degli slot. Un modo per ottenere la sincronizzazione sarebbe quello di avere una stazione speciale che emettesse un segnale all'inizio di ogni intervallo, come un orologio.

Diversamente da ALOHA puro di Abramson, nel metodo di Roberts, comunemente noto come ALOHA a slot, un computer non può spedire ogni volta che viene battuto un ritorno carrello. Invece viene richiesto che il computer aspetti l'inizio di un nuovo slot. Quindi ALOHA puro che era continuo viene mutato in un ALOHA discreto. Poiché il periodo critico ora è la metà, la probabilità di non avere altro traffico nello stesso slot del nostro pacchetto di test sarà e^{-G} per cui:

$$S = Ge^{-G} \quad (4-3)$$

Come mostrato in figura 4-3, la punta massima di ALOHA a slot è $G = 1$, con un throughput di $S = 1/e$, o di circa 0,368, il doppio di quello di ALOHA puro. Se il sistema sta operando con $G = 1$, la probabilità che uno slot sia non utilizzato è di 0,368 (dall'equazione (4-2)). Il meglio che possiamo sperare usando ALOHA a slot è di ottenere un 37% di slot inutilizzati, un 37% di successi e un 26% di collisioni. Operare con valori di G più elevati, riduce il numero di slot inutilizzati ma aumenta il numero

delle collisioni in modo esponenziale. Per notare come la crescita di G influenzi la crescita rapida del numero di collisioni, si consideri la ritrasmissione di un pacchetto di test. La probabilità che esso eviti una collisione è di e^{-G} , cioè la probabilità che tutti gli altri utenti siano inattivi in quello slot. La probabilità di una collisione è quindi solo $1-e^{-G}$. La probabilità che una ritrasmissione richieda esattamente k tentativi (cioè $k-1$ collisioni seguite da un successo) è

$$P_k = e^{-G} (1-e^{-G})^{k-1}$$

Il numero atteso di ritrasmissioni, E , per ogni ritorno carrello battuto sarà quindi

$$E = \sum_{k=1}^{\infty} k P_k = \sum_{k=1}^{\infty} k e^{-G} (1-e^{-G})^{k-1} = e^G$$

A causa della dipendenza esponenziale di E da G , piccoli aumenti del carico sul canale possono ridurre drasticamente le sue prestazioni.

4.2.2 Protocolli multiaccesso con rilevamento della portante

Con ALOHA a slot il miglior utilizzo di canale che si può ottenere è di $1/e$. Questo non può sorprendere, infatti le stazioni trasmettono quando vogliono, senza prestare attenzione a quello che stanno facendo le altre stazioni: questo comporta un numero abbastanza alto di collisioni. Nelle LAN è comunque possibile per le stazioni rilevare cosa stanno facendo le altre e adattare il proprio comportamento. Queste reti possono raggiungere un utilizzo migliore di $1/e$. In questo paragrafo discuteremo alcuni protocolli per migliorare le prestazioni. I protocolli in cui le stazioni controllano la portante (cioè la trasmissione) e agiscono di conseguenza sono detti **protocolli con rilevamenti della portante**. Ne sono stati proposti molti. Kleinrock e Tobagi (1975) hanno analizzato molti di questi protocolli in dettaglio. Di seguito menzioneremo diverse versioni di protocolli con rilevamento della portante.

CSMA persistente e non persistente

Il primo protocollo con rilevamento della portante che studieremo è detto **CSMA (Carrier Sense Multiple Access – multiaccesso con rilevamento della portante) 1-persistente**. Quando una stazione deve inviare dati, essa prima ascolta il canale per sapere se qualcun altro sta trasmettendo. Se il canale è occupato la stazione aspetta finché non si libera. Quando una stazione trova il canale libero trasmette un pacchetto. Se avviene una collisione la stazione aspetta un intervallo di tempo arbitrario e quindi riinizializza il ciclo. Il protocollo viene detto 1-persistent perché una stazione trasmette con probabilità 1 quando trova il canale libero.

Il ritardo di propagazione ha un effetto importante sulle prestazioni del protocollo. C'è una piccola probabilità che, non appena una stazione abbia iniziato a spedire, un'altra stazione diventi pronta per spedire e ascolti il canale. Se il segnale della prima stazione non è ancora stato ricevuto dalla seconda, questa troverà il canale inattivo e invierà

4.2 Protocolli di accesso multiplo

anch'essa il pacchetto causando una collisione. Più grande è il ritardo di propagazione più questo effetto si accentua e peggiori saranno le prestazioni del protocollo.

Anche se il ritardo di propagazione è zero avremo collisione. Se due stazioni sono pronte a spedire quando una terza sta trasmettendo, entrambe aspetteranno educatamente che la trasmissione termini e quindi inizieranno a trasmettere simultaneamente causando una collisione. Se non fossero così impazienti ci sarebbero molte meno collisioni. Anche con questi difetti il protocollo rimane sempre migliore di ALOHA puro, poiché entrambe le stazioni hanno il buon senso di non interferire con la trasmissione del pacchetto della terza stazione. Intuitivamente, questo implica delle prestazioni migliori di ALOHA puro. Le stesse considerazioni valgono anche per l'ALOHA a slot.

Un secondo protocollo con rilevamento della portante è **CSMA non persistente**. In questo protocollo, viene attuato un apposito tentativo di non ingerenza, rispetto al precedente. Prima di spedire, una stazione controlla il canale. Se nessuno sta inviando, la stazione inizia a trasmettere. Se invece il canale è occupato la stazione non continuerà a controllare il canale per impadronirsi non appena la trasmissione termina, ma aspetterà un tempo casuale e quindi ripeterà l'algoritmo. Intuitivamente questa tecnica porterà a un miglior utilizzo del canale e ritardi più lunghi rispetto a CSMA 1-persistente.

L'ultimo protocollo è **CSMA p-persistente**. Si applica a canali discreti e funziona come segue. Quando una stazione è pronta a trasmettere ascolta il canale. Se è libero essa trasmetterà con probabilità p . Con una probabilità di $q = p-1$ aspetterà il prossimo slot. Se anche questo slot è libero trasmetterà oppure aspetterà ancora, con probabilità p e q . Questo processo sarà ripetuto finché il pacchetto non verrà spedito oppure un'altra stazione non inizierà a trasmettere. In quest'ultimo caso la stazione si comporterà come se ci fosse stata una collisione (cioè aspetterà una quantità di tempo casuale e riinizierà). Se inizialmente la stazione trova il canale occupato aspetterà il prossimo slot e riapplicherà l'algoritmo. La figura 4-4 mostra la produttività in termini di traffico offerto per i tre protocolli, per l'ALOHA puro e a slot.

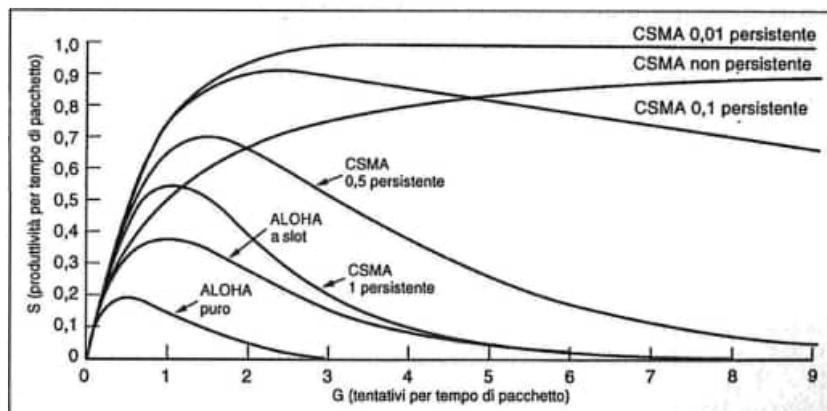


Fig. 4-4 Confronti di utilizzo del canale per carico in diversi protocolli ad accesso casuale.

CSMA con rilevazione di collisione

I protocolli CSMA persistenti e non persistenti costituiscono chiaramente un miglioramento rispetto ad ALOHA. Infatti, essi assicurano che nessuna stazione inizi a trasmettere dopo aver trovato il canale occupato. Un'altra miglioria consiste nel fare in modo che le stazioni blocchino la trasmissione se si accorgono di una collisione. In altre parole, se due stazioni trovano il canale libero e iniziano a trasmettere simultaneamente si accorgeranno altrettanto simultaneamente della collisione. Piuttosto di terminare le trasmissioni, che risulterebbero comunque invalide, esse dovrebbero interrompere la trasmissione appena si accorgono della collisione. Interrompere prima del tempo trasmisso danneggiate risparmia larghezza di banda e tempo. Questo protocollo, noto come **CSMA/CD** (**C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **D**etection – multiaccess con rilevazione della portante e rilevazione di collisione), viene largamente utilizzato nel sottolivello MAC delle LAN.

CSMA/CD, come molti altri protocolli LAN, utilizza il modello concettuale in figura 4-5. Al tempo t_0 , una stazione ha finito di trasmettere il suo pacchetto. Qualsiasi altra stazione che deve spedire un pacchetto potrebbe adesso tentare di inviarlo. Se due o più stazioni decidono simultaneamente di trasmettere si avrà una collisione. Le collisioni possono essere rilevate osservando la potenza o la larghezza dell'impulso del segnale ricevuto e confrontandolo con quello trasmesso.

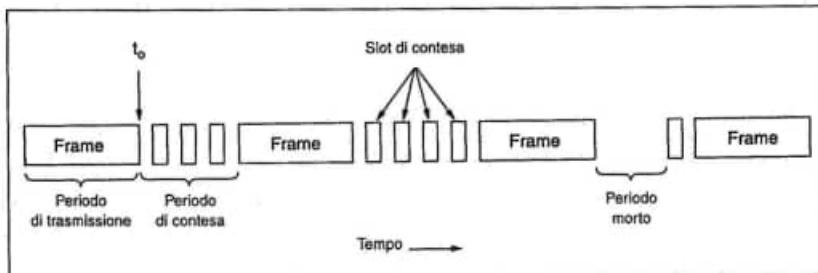


Fig. 4-5 CSMA/CD può essere in uno dei tre stati: contesa, trasmissione o inattività.

Dopo che una stazione ha rilevato una collisione, essa interrompe la trasmissione, aspetta una quantità di tempo casuale e quindi riprova assumendo che nessuna altra stazione abbia iniziato a trasmettere nello stesso istante. Il nostro modello per CSMA/CD considererà quindi in periodi di contesa e altri di trasmissione, con periodi di quiete che si avranno quando tutte le stazioni saranno inattive (ad es. per mancanza di lavoro).

Consideriamo adesso più attentamente i dettagli dell'algoritmo di contesa: si supponga che due stazioni inizino entrambe a trasmettere al tempo t_0 . Quanto tempo impiegheranno per accorgersi della collisione? La risposta a questa domanda è di vitale importanza per determinare la lunghezza del periodo di contesa e quindi il ritardo e la produttività. Il tempo minimo per determinare la collisione è il tempo impiegato per propagare un segnale da una stazione all'altra.

Sulla base di questo ragionamento si potrebbe pensare che una stazione che non avesse

notato una collisione per un tempo uguale al tempo di propagazione dell'intero cavo dopo aver iniziato la sua trasmissione, possa avere la certezza di possedere il cavo. Con il termine "possedere" intendiamo dire che tutte le altre stazioni sarebbero a conoscenza che essa sta trasmettendo e non interferirebbero. Questa deduzione è errata. Infatti si consideri il seguente scenario (il caso peggiore). Sia il tempo di propagazione tra le due stazioni più lontane τ . Al tempo t_0 una stazione inizia a trasmettere. Al tempo $\tau - \epsilon$, un istante prima che il segnale arrivi alla stazione più lontana, quella stazione inizia anch'essa a trasmettere. Ovvamente si accorgerà della collisione quasi immediatamente e si arresterà, ma la piccola raffica di rumore causata dalla collisione non raggiungerà la prima stazione fino al tempo $2\tau - \epsilon$. In altre parole, nel caso peggiore una stazione non potrà essere sicura di possedere il canale finché non avrà trasmesso per un tempo 2τ senza rilevare collisione. Per questo motivo modelleremo l'intervallo di contesa come un sistema ALOHA a slot con slot di larghezza 2τ . Su un cavo coassiale lungo un 1 km $\tau = 5\mu s$. Per semplicità assumeremo che ogni slot contenga un solo bit. Una volta che possiede il canale, una stazione può trasmettere a qualsiasi velocità e non dovrà limitarsi a 1 bit per 2τ s.

È importante capire che la determinazione di una collisione è un processo *analogico*. L'hardware della stazione deve "ascoltare" il cavo mentre sta trasmettendo. Se ciò che torna indietro è diverso da ciò che spedisce, si accorge di una collisione. L'implicazione è che la codifica del segnale deve permettere la rilevazione delle collisioni (ad es. una collisione di due segnali di 0 volt potrebbe benissimo non essere rilevabile). Per questa ragione, si utilizza spesso una codifica speciale.

CSMA/CD è un protocollo importante. Più avanti in questo stesso capitolo studieremo una sua versione, IEEE 802.3 (Ethernet), che è uno standard internazionale.

Per evitare fraintendimenti, occorre sottolineare che nessun protocollo del sottolivello MAC garantisce la consegna affidabile. Anche in assenza di collisioni, il ricevente potrebbe non aver ricoppiato correttamente il pacchetto per diverse ragioni (ad es. mancanza di spazio di buffer oppure un'interruzione non raccolta).

4.2.3 Protocolli esenti da collisione

Sebbene con CSMA/CD non avvengano collisioni dopo che la stazione si è impadronita del canale, esse possono ancora accadere durante il periodo di contesa. Queste collisioni incidono negativamente sulle prestazioni, specialmente quando il cavo è lungo (cioè si ha un grande τ) e il pacchetto è corto. Da quando sono entrate in uso lunghe reti in fibra ottica e ampia larghezza di banda, la combinazione di un grande τ con un pacchetto corto è divenuta sempre più un problema serio. In questo paragrafo esamineremo alcuni protocolli che risolvono i conflitti su un canale senza avere collisioni, nemmeno durante la fase di contesa.

Nei protocolli descritti di seguito assumeremo che ci siano N stazioni, ciascuna con un indirizzo unico da 0 a $N-1$ legato a esse. Non importa se qualche stazione resta inattiva parte del tempo. La questione base resta: quale stazione ottiene il canale dopo una trasmissione riuscita? Continueremo a utilizzare il modello in figura 4-5 con slot di contesa discreti.

Un protocollo a mappa di bit

Nel nostro primo protocollo esente da collisione, il **metodo di base a mappa di bit**, ogni periodo di contesa consiste di esattamente N slot. Se la stazione 0 deve spedire un pacchetto, essa trasmette un bit 1 nello slot 0. A nessuna altra stazione è permesso di trasmettere in quello slot. Indipendentemente da quello che farà la stazione 0, la stazione 1 avrà l'opportunità di trasmettere durante lo slot 1, ovviamente solo se ha un pacchetto in attesa. In generale, la stazione j potrà annunciare di avere un pacchetto in attesa di spedizione emettendo un bit 1 durante lo slot j . Trascorsi gli N slot, ogni stazione sa esattamente quali stazioni vogliono trasmettere. A questo punto, esse iniziano a trasmettere in ordine di numero (si veda la figura 4-6).



Fig. 4-6 Il protocollo di base a mappa di bit.

Poiché tutti sono d'accordo sull'ordine, non si avranno mai collisioni. Dopo che l'ultima stazione pronta ha trasmesso il suo pacchetto, un evento che tutte le stazioni sono in grado di osservare, ha inizio un altro periodo di contesa di N bit. Se una stazione diventa pronta dopo che è trascorso il suo slot, non avrà la possibilità di spedire e dovrà rimanere inattiva finché ogni stazione non avrà avuto la sua possibilità e la mappa di bit non avrà compiuto l'intero giro. Protocolli come questo, in cui il desiderio di trasmettere viene comunicato con broadcast prima della reale trasmissione, sono detti **protocolli di prenotazione**. Analizziamo brevemente le prestazioni di questo protocollo. Per convenienza misureremo il tempo in unità di slot di bit di contesa, con pacchetto di dati di d unità di tempo. In condizioni di carico ridotto, la mappa di bit sarà ripetuta di continuo, per mancanza di pacchetti.

Si consideri la situazione dal punto di vista di una stazione con un numero basso, come 0 o 1. Tipicamente, quando sarà pronta a inviare, lo slot corrente sarà da qualche parte nella mappa di bit. Nel caso medio la stazione dovrà aspettare $N/2$ slot perché il giro in corso finisca, e un altro giro di N slot per il completamento della scansione seguente prima di poter trasmettere.

La situazione per le stazioni con numeri alti è migliore. Generalmente dovranno aspettare solo mezzo giro ($N/2$ slot di bit) prima di trasmettere. Difficilmente dovranno aspettare il giro successivo. Poiché le stazioni con numero basso devono aspettare in media $1.5N$ slot e quelle con numero alto $0.5N$ slot, il risultato medio è di N slot. L'efficienza del canale a carico ridotto è facilmente calcolabile. L'appesantimento per pacchetto è di N bit e la quantità di dati di d bit, per un efficienza di $d/(N+d)$.

Sotto carico elevato, quando tutte le stazioni devono sempre spedire qualcosa, il periodo di contesa di N bit è distribuito su N frame, comportando un appesantimento di un solo

bit per pacchetto, un efficienza di $d/(d+1)$. Il ritardo medio per un pacchetto è uguale alla somma del tempo che esso attende in coda all'interno della stazione più un aggiuntivo $N(d+1)/2$ una volta arrivato all'inizio della coda interna.

Conteggio binario a ritroso

Un problema con il protocollo a mappa di bit è che l'appesantimento è di un bit per stazione. Possiamo ottenere qualcosa di meglio usando indirizzi binari per le stazioni. Una stazione che vuole utilizzare il canale adesso trasmetterà con un broadcast il suo indirizzo come stringa binaria di bit, partendo da quello di ordine più alto. Si assume che tutti gli indirizzi abbiano la stessa lunghezza. I bit in ogni posizione dell'indirizzo per le diverse stazioni vengono tra loro "disgiunti" (con l'OR booleano). Chiameremo questo protocollo a **conteggio binario a ritroso**. Esso viene utilizzato in Datakit (Fraser, 1987). Per evitare conflitti, deve essere applicata una regola di arbitraggio: non appena una stazione nota che un bit di posizione più elevata nel suo indirizzo è stato riscritto da un 1, si interrompe. Per esempio, se le stazioni 0010, 0100, 1001 e 1010 stanno tutte cercando di ottenere il canale, nel primo bit di tempo le stazioni trasmettono rispettivamente 0, 0, 1 e 1. Questi vengono disgiunti (con OR) dando come risultato 1. Le stazioni 0010 e 0100 notano l'1 e capiscono che una stazione con numero alto vuole ottenere il canale, quindi si autoescludono dal corrente giro. Le stazioni 1001 e 1010 continuano.

Il bit successivo è 0 ed entrambe le stazioni continuano. Il bit seguente è 1, quindi la stazione 1001 termina. La vincente è la stazione 1010, poiché ha l'indirizzo più alto. Dopo aver conquistato il canale può trasmettere il pacchetto, dopodiché inizierà un nuovo giro. Il protocollo è illustrato in figura 4-7.

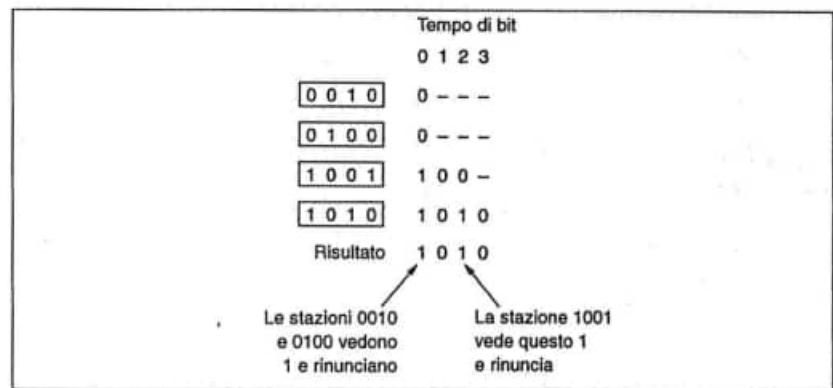


Fig. 4-7 Il protocollo di conteggio binario a ritroso. La lineetta indica stazione silente.

L'efficienza del canale con questo metodo è di $d/(d + \log_2 N)$. Se inoltre il formato del pacchetto viene scelto in modo che l'indirizzo del mittente sia il primo campo, anche gli N bit non andranno persi e l'efficienza sarà del 100%.

Mok e Ward (1979) hanno descritto una variante del conteggio binario a ritroso usando

un'interfaccia parallela anziché seriale. Essi suggeriscono anche di usare numeri virtuali di stazioni, da 0 fino alla stazione che ha avuto successo permutandola dopo ogni trasmissione, per dare priorità più alta alle stazioni rimaste inattive per troppo tempo. Per esempio, se le stazioni *C, H, D, A, G, B, E, F* hanno priorità rispettivamente 7, 6, 5, 4, 3, 2, 1 e 0, allora una trasmissione di successo per *D* la porterà ad essere ultima della lista, con ordine di priorità risultante: *C, H, A, G, B, E, F, D*. Quindi *C* rimarrà la stazione virtuale numero 7 ma *A* da 4 diventerà 5 e *D* da 5 sarà 0. La stazione *D* sarà in grado di occupare il canale solo se nessun'altra stazione lo vorrà.

4.2.4 Protocolli a contesa limitata

Abbiamo considerato due strategie di base per l'acquisizione del canale in una rete via cavo: la contesa, come in CSMA, e i metodi esenti da collisione. Ogni strategia può essere valutata in base a due importanti misure delle prestazioni, il ritardo a basso carico e l'efficienza del canale a carico elevato. In condizioni di carico ridotto la contesa (ALOHA puro e a slot) è preferibile a causa del basso ritardo che comporta. Se invece il carico aumenta, la contesa diviene sempre meno attraente, poiché l'appesantimento associato con l'arbitraggio del canale diventa alto. Per i protocolli esenti da collisione è vero esattamente il contrario. A basso carico essi hanno un ritardo elevato, ma se il carico aumenta, l'efficienza del canale aumenta, invece di peggiorare come nel caso dei protocolli di contesa.

Ovviamente sarebbe utile poter combinare le proprietà dei protocolli di contesa con quelli esenti da collisioni, ottenendo un nuovo protocollo che usi contesa a basso carico per avere ritardi piccoli e tecniche esenti da collisione quando si ha alto carico per fornire un'elevata efficienza del canale. Questo tipo di protocolli che chiameremo **protocolli a contesa limitata**, esistono effettivamente e concluderanno il nostro studio sulle reti a rilevamento di carico.

Fino ad ora gli unici protocolli di contesa considerati erano simmetrici, cioè ogni stazione tentava di ottenere il canale con qualche probabilità *p* dove *p* era uguale per tutte le stazioni. È interessante notare che le prestazioni possono talvolta essere migliorate usando un protocollo che assegna probabilità diverse alle diverse stazioni.

Prima di addentrarci nei protocolli asimmetrici rivediamo rapidamente le prestazioni di quelli simmetrici. Supponiamo che *k* stazioni si stiano contendendo l'accesso al canale. Ognuna ha probabilità *p* di trasmettere in ogni slot. La probabilità che alcune stazioni acquisiscano il canale durante un dato slot è $k p (1-p)^{k-1}$. Per determinare il valore ottimale di *p* deriviamo rispetto a *p* ed egualiamo a 0 risolvendo rispetto a *p*. Facendo questo troviamo che il miglior valore per *p* è $1/k$. Sostituendo *p* = $1/k$ otteniamo:

$$\text{Pr} [\text{successo con } p \text{ ottimo}] = \left\{ \frac{k-1}{k} \right\}^{k-1} \quad (4-4)$$

Questa probabilità viene rappresentata in figura 4-8. Per un numero piccolo di stazioni, la possibilità di successo è alta ma a mano a mano che il numero di stazioni arriva a 5, la probabilità scende fino al suo valore asintotico $1/e$.

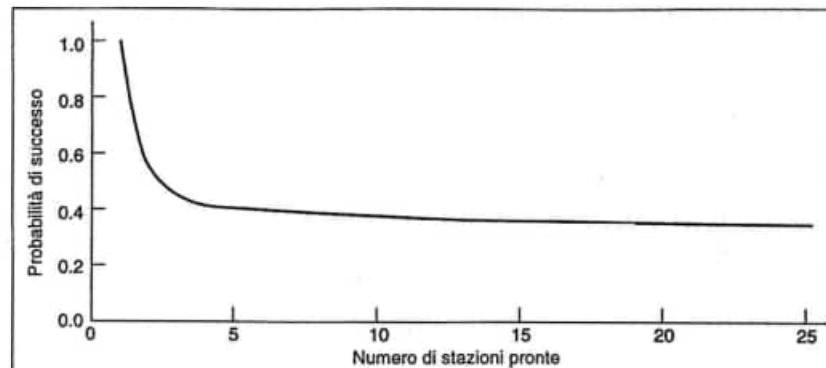


Fig. 4-8 Probabilità di acquisizione per un canale di contesa simmetrico.

Dalla figura 4-8 è abbastanza ovvio che la probabilità di acquisire un canale da parte di una stazione può essere aumentata diminuendo il grado di competizione. I protocolli a contesa limitata fanno esattamente questo. Essi prima dividono le stazioni in gruppi (non necessariamente disgiunti). Solo i componenti del gruppo 0 potranno competere per lo slot 0. Se uno di essi ha successo acquisisce il canale e trasmette il suo pacchetto. Se lo slot resta inutilizzato oppure avviene una collisione, i componenti del gruppo 1 si contendono lo slot 1 e così via. Attuando una divisione appropriata delle stazioni in gruppi, il grado di contesa per ogni slot può essere ridotto, in modo che ogni slot abbia sempre un grafico simile all'estremità sinistra della figura 4-8.

Il problema è come assegnare le stazioni agli slot. Prima di considerare il caso generale consideriamo alcuni casi speciali. Nel caso un gruppo abbia un solo componente, non si avranno mai collisioni poiché al massimo una stazione sarà in competizione per lo slot. Abbiamo già visto questo tipo di protocollo (cioè il protocollo a conteggio binario a ritroso). L'altro caso speciale è quello dell'assegnamento di due stazioni a ogni gruppo. La probabilità che entrambe tentino di trasmettere durante uno slot è p^2 , che per *p* piccoli è trascurabile. Aumentando il numero di stazioni assegnate a uno stesso slot, la probabilità di una collisione aumenta ma la lunghezza della scansione della mappa di bit necessaria per dare ad ognuno una possibilità diminuisce. Il caso limite si ha quando un unico gruppo contiene tutte le stazioni (come in ALOHA). Quello che serve è un metodo dinamico di assegnamento delle stazioni agli slot, con molte stazioni per slot quando il carico è ridotto e poche (anche solo una) per slot quando il carico è elevato.

Il protocollo di attraversamento adattivo dell'albero

Un modo particolarmente semplice per ottenere l'assegnamento necessario è quello di utilizzare l'algoritmo ideato dall'esercito americano per individuare i soldati colpiti da sifilide durante la seconda guerra mondiale (Dorfman, 1943). In breve, l'esercito prelevava campioni di sangue da *N* soldati, una parte di ogni campione veniva versata in un unico campione di test. Questo campione misto veniva poi controllato per individuare

eventuali anticorpi e se non se ne rilevava la presenza tutti i soldati del gruppo erano considerati sani. Se venivano ritrovati anticorpi, venivano preparati altri due campioni misti, uno dai soldati da 1 a $N/2$, e l'altro dai restanti soldati. Il processo era ripetuto ricorsivamente finché non si identificavano i soldati infetti.

Per la versione informatica di tale algoritmo (Capetanakis, 1979) conviene immaginare le stazioni come foglie di un albero binario, come mostrato in figura 4-9. Nel primo slot di contesa che seguia la trasmissione riuscita di un pacchetto, lo slot 0, tutte le stazioni potranno tentare di ottenere il canale. Se una di esse riesce, bene; se si ha una collisione, durante lo slot 1 solo le stazioni che cadono sotto il nodo 2 potranno competere. Se una di esse acquisisce il canale, lo slot che seguirà il pacchetto sarà riservato alle stazioni sotto il nodo 3. Se invece due o più stazioni sotto il nodo 2 vorranno trasmettere, si avrà una collisione durante lo slot 1 e sarà il turno delle stazioni sotto il nodo 4 per lo slot 2.

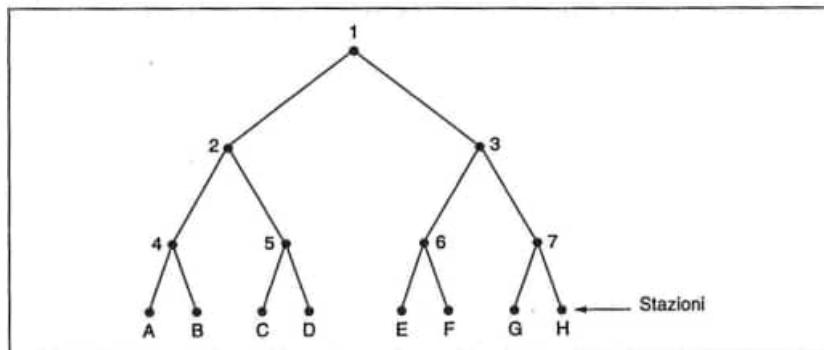


Fig. 4-9 Un albero con otto stazioni.

In sostanza, se si ha una collisione durante lo slot 0, verrà visitato l'intero albero in profondità, per individuare tutte le stazioni pronte. Ogni slot di bit è associato con un particolare nodo dell'albero. Se si ha una collisione la ricerca continua ricorsivamente sui figli destro e sinistro. Se lo slot di bit è inattivo o se si ha solo una stazione che vuole trasmettere, la ricerca del suo nodo verrà arrestata poiché tutte le stazioni pronte sono state individuate. (Se ci fosse più di una stazione avverrebbe una collisione).

Quando il carico del sistema è elevato non vale la pena di assegnare lo slot 0 al nodo 1 poiché ha senso solo nel caso raro in cui esattamente una stazione ha un pacchetto da spedire. In modo simile, si può supporre che i nodi 2 e 3 potrebbero essere saltati per lo stesso motivo. In termini più generali a che livello dell'albero dovrebbe iniziare la ricerca? Chiaramente più elevato è il carico più in basso nell'albero occorre iniziare la ricerca. Assumeremo che ogni stazione abbia una buona stima del numero delle stazioni pronte, q , per esempio controllando il traffico recente.

Procediamo numerando i livelli dell'albero dall'alto con il nodo 1 in figura 4-9 al livello 0, i nodi 2 e 3 al livello 1 e così via. Si noti che ogni nodo al livello i ha una frazione 2^i di stazioni sotto di lui. Se le q stazioni pronte sono distribuite in modo uniforme, il numero

atteso di esse sotto un certo nodo al livello i sarà $2^i q$. Intuitivamente, ci aspetteremmo che il livello ottimale a cui iniziare la ricerca nell'albero sia dove il numero medio di stazioni in competizione per ogni slot sia 1, cioè il livello in cui $2^i q = 1$. Risolvendo questa equazione troviamo $i = \log_2 q$.

Numerosi miglioramenti sono stati scoperti rispetto all'algoritmo di base e vengono descritti nei dettagli da Bertsekas e Gallager (1992). Per esempio, si consideri il caso in cui le stazioni G e H siano le sole a voler trasmettere. Sul nodo 1 si avrà una collisione quindi si proverà il nodo 2 e verrà trovato inattivo. È inutile provare il nodo 3 poiché si avrà sicuramente una collisione (sappiamo che due o più stazioni sotto 1 sono pronte e nessuna di esse è sotto 2, allora saranno sotto 3). Si può evitare la prova di 3 e passare a 6. Quando anche questa prova non avrà portato a niente, 7 sarà saltato e si proverà il nodo G .

4.2.5 Protocolli multiaccesso a suddivisione di lunghezza d'onda

Un approccio diverso per quanto riguarda l'allocazione del canale è quello di dividere il canale in sottocanali usando FDM, TDM o entrambi e allocarli dinamicamente secondo le necessità. Tecniche come questa sono generalmente utilizzate sulle LAN a fibra ottica al fine di permettere che diverse conversazioni utilizzino lunghezze d'onda (frequenze) diverse nello stesso momento. In questo paragrafo esamineremo uno di questi protocolli (Humblet et al. 1992).

Un modo semplice per costruire una LAN completamente ottica è quello di usare un accoppiatore a stella passivo (si veda la figura 2-10). Due fibre di ogni stazione vengono saldate a un cilindro di vetro. Una fibra è per l'output sul cilindro e l'altra per l'input dal cilindro. L'output luminoso da una stazione illumina il cilindro e può essere rilevato da tutte le altre stazioni. Le stelle passive possono gestire centinaia di stazioni.

Per permettere trasmissioni multiple nello stesso istante, lo spettro viene suddiviso in diversi canali (bande di lunghezza d'onda), come mostrato in figura 2-24. In questo protocollo, WDMA (Wavelength Division Multiple Access – multiaccesso a suddivisione di lunghezza d'onda), ad ogni stazione vengono assegnati due canali. Un canale stretto viene fornito come canale di controllo per i segnali delle stazioni e un canale largo per l'output dei frame di dati delle stazioni.

Ogni canale è suddiviso in gruppi di intervalli di tempo (slot), come in figura 4-10. Sia m il numero di slot del canale di controllo ed $n+1$ il numero di slot nel canale di dati, dove n sono per i dati e l'ultimo viene usato dalla stazione per riportare il suo stato (cioè quali slot di entrambi i canali sono liberi). Su entrambi i canali la sequenza di slot si ripete ininterrottamente, con lo slot 0 indicato in modo speciale per essere individuabile. Tutti i canali sono sincronizzati da un orologio globale unico.

Il protocollo supporta tre classi di traffico: (1) il traffico orientato alla connessione a frequenza costante di dati, come dati video non compressi, (2) il traffico orientato alla connessione con frequenza di traffico variabile, come i trasferimenti di file, (3) il traffico di datagrammi, come i pacchetti UDP. Per i due protocolli orientati alla connessione, quando A vuole comunicare con B deve prima inserire un pacchetto di RICHIESTA DI CONNESSIONE in uno slot libero sul canale di controllo di B . Se B accetta, la comunicazione potrà avere luogo sul canale dati di A .

Ogni stazione ha due trasmettitori e due riceventi organizzati come segue:

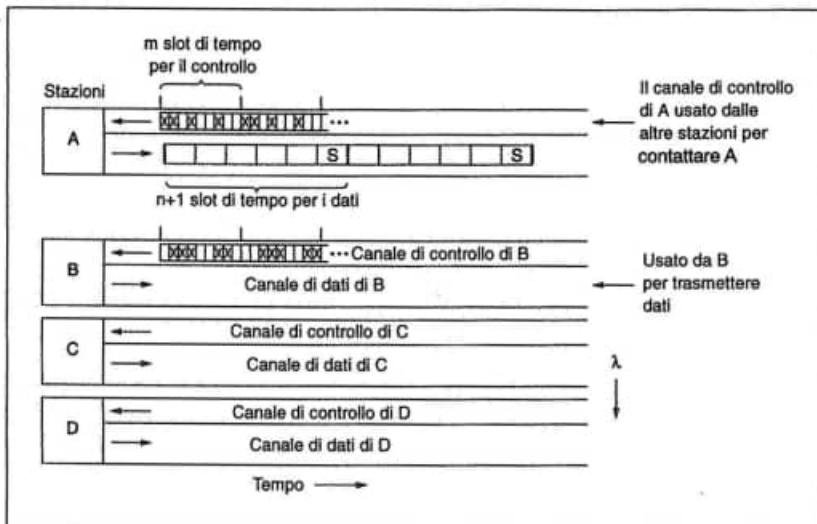


Fig. 4-10 Multiaccesso a suddivisione di lunghezza d'onda.

1. Un ricevente a lunghezza d'onda fissa per ascoltare il suo canale di controllo.
2. Un trasmettitore a frequenza regolabile per spedire sul canale di controllo delle altre stazioni.
3. Un trasmettitore a lunghezza d'onda fissa per mandare in output frame di dati.
4. Un ricevente a frequenza regolabile per selezionare il trasmettitore da ascoltare.

In altre parole, ogni stazione ascolta il proprio canale di controllo attendendo richieste, ma dovrà regolare la sua frequenza su quella del trasmettitore per ottenere i dati. La regolazione della frequenza viene fatta con l'interferometro Fabry-Perot oppure Mach-Zehnder, che scarta tutte le lunghezze d'onda eccetto la banda di lunghezza d'onda desiderata.

Consideriamo ora come una stazione *A* installi un canale di comunicazione di classe 2 con la stazione *B* per, diciamo, trasferimento di file. Prima *A* regola la frequenza del suo ricevente di dati sul canale dati di *B* e attende lo slot di stato. Questo slot indica quali slot di controllo sono assegnati e quali liberi. Nella figura 4-10, ad esempio, si nota che di otto slot di controllo di *B*, solo lo 0, il 4 e il 5 sono liberi. Gli altri sono occupati (indicati con delle croci).

A sceglie uno degli slot liberi, diciamo il 4, e vi inserisce il suo messaggio di RICHIESTA DI CONNESSIONE. Poiché *B* controlla continuamente il suo canale di controllo, esso nota la richiesta e la accorda assegnando lo slot 4 ad *A*. L'assegnamento è annunciato nello slot di stato del canale di controllo. Quando *A* nota la risposta, sa di avere una

4.2 Protocolli di accesso multiplo

connessione unidirezionale. Se *A* avesse chiesto una connessione bidirezionale, *B* avrebbe ripetuto lo stesso algoritmo nei confronti di *A*.

È possibile che nello stesso momento in cui *A* cerca di occupare lo slot di controllo 4 di *B*, *C* faccia altrettanto. Nessuna delle due stazioni lo otterrà ed entrambe noteranno il fallimento controllando lo slot di stato nel canale di controllo di *B*. Esse aspetteranno una quantità casuale di tempo e ritenteranno ancora.

A questo punto ogni parte sa come spedire in modo esente da collisioni un corto messaggio di controllo all'altra. Per compiere il trasferimento di file, *A* spedisce ora un messaggio di controllo a *B* che ad esempio dice "Per favore osserva il mio prossimo slot di dati in output 3. C'è un pacchetto di dati per te". Quando *B* riceve il messaggio di controllo, regola la frequenza del suo ricevente sul canale di output di *A* per leggere il pacchetto di dati. A seconda del protocollo di livello più alto, *B* potrà usare lo stesso meccanismo per inviare indietro, se vuole, un ack.

Si noti che sorge un problema se *A* e *C* hanno entrambi una connessione con *B* e a un certo punto dicono a *B* di osservare lo slot 3. *B* sceglierà in modo casuale la richiesta da gestire e l'altra trasmissione andrà persa.

Se la frequenza di traffico è costante viene utilizzata una variante di questo protocollo. Quando *A* richiede una connessione esso simultaneamente dice anche una cosa del tipo "Va bene se ti mando un pacchetto in ogni slot 3?". Se *B* è in grado di accettare (non ha preso altri impegni per lo slot 3), viene garantita una connessione di una frequenza stabilita. Altrimenti *A* potrà provare con una proposta diversa a seconda di quali slot di output disponga.

Il traffico di classe 3 (datagrammi) utilizza ancora un'altra variante. Invece di inviare un messaggio RICHIESTA DI CONNESSIONE nello slot di controllo che trova (4), invia un messaggio DATI PER TE NELLO SLOT 3. Se *B* è disponibile per il successivo slot 3 la trasmissione avrà successo. Altrimenti il pacchetto di dati sarà perso. In questo modo non sono mai necessarie specifiche connessioni.

Sono possibili molte varianti dell'intero protocollo. Per esempio, invece di avere un canale di controllo per ogni stazione, un singolo canale può essere utilizzato da tutte le stazioni. A ogni stazione viene assegnato un blocco di slot in ogni gruppo, raggruppando diversi canali virtuali su di un unico canale fisico.

È anche possibile avere un unico trasmettitore e un unico ricevente con frequenza variabile per stazione, dividendo ogni canale della stazione in *m* slot di controllo seguiti da *n* + 1 slot di dati. Lo svantaggio in questo caso è che il mittente deve attendere più a lungo per catturare uno slot di controllo e i frame di dati consecutivi sono piuttosto distanziati poiché in mezzo viaggiano informazioni di controllo.

Sono stati proposti molti altri protocolli WDMA, che differiscono nei dettagli. Alcuni hanno un canale di controllo, altri più canali di controllo. Alcuni tengono in considerazione il ritardo di propagazione, altri no; diversi considerano il tempo di variazione di frequenza come parte del modello, altri lo ignorano. I protocolli differiscono anche in complessità di elaborazione, produttività e scalabilità. Per ulteriori dettagli si vedano Bogineni et al. (1993); Chen (1994); Chen, Yum (1991); Jia, Mukherjee (1993); Levine, Akyildiz (1995); Williams et al. (1993).

4.2.6 Protocollo per LAN senza filo

Al crescere del numero di computer portatili e degli strumenti di comunicazione, cresce anche la richiesta di connessione di questi con il mondo esterno. Anche il primo telefono portatile aveva già la capacità di connettersi con gli altri telefoni. Il primo computer portatile non aveva questa capacità, ma dopo poco sono diventati di uso comune i modem. Per entrare nella rete questi computer dovevano essere collegati con una spina al muro. La richiesta di connessione via cavo a una rete fissa significava che i computer erano sì portatili ma non mobili.

Per ottenere la mobilità effettiva i computer portatili devono utilizzare per comunicare segnali radio (o infrarossi). In questo modo gli utenti possono leggere o inviare posta elettronica mentre stanno guidando o sono in barca. Un sistema di computer portatili che comunicano via radio può essere visto come una LAN senza fili. Questo tipo di LAN ha alcune proprietà diverse rispetto alle LAN convenzionali e richiede speciali protocolli per il sottolivello MAC. In questo paragrafo esamineremo alcuni di questi protocolli. Per ulteriori dettagli riguardo alle LAN senza filo si vedano Davis, McGuffin (1995); Nemzow (1995).

Una configurazione comune per una LAN senza filo è un palazzo per uffici con stazioni base allocate intorno all'edificio. Tutte le stazioni base sono collegate tra loro usando rame e fibra. Se la potenza di trasmissione delle stazioni base e dei portatili è regolata in modo da avere un raggio di 3 o 4 m, allora ogni stanza diviene un'unica cella e l'intero edificio un grande sistema cellulare, come nei tradizionali sistemi di telefonia cellulare studiati nel capitolo 2. Diversamente dai sistemi di telefonia cellulare, ogni cella ha solo un canale, che copre l'intera larghezza di banda disponibile. Tipicamente la sua larghezza di banda è di 1 o 2 Mbps.

Nella discussione che seguirà, faremo l'assunzione semplificativa che tutti i trasmettitori radio abbiano un campo fisso. Quando un ricevente è nel campo di due trasmettitori attivi, il segnale risultante sarà illeggibile e inutile (con alcune eccezioni discusse di seguito). È importante notare che in alcune LAN senza filo non tutte le stazioni sono nel range di un'altra: si ha quindi una serie di complicazioni. Inoltre, per LAN senza filo all'interno, la presenza di muri tra le stazioni può avere un impatto maggiore sul campo effettivo di ogni stazione.

Un approccio semplicistico di utilizzo di una LAN senza filo può essere quello di usare CSMA: ascoltare le trasmissioni degli altri e trasmettere solo se non lo sta facendo nessun altro. Il problema è che questo protocollo non è assolutamente appropriato poiché ciò che importa è l'interferenza sul ricevente e non sul mittente. Per vedere la natura del problema, si consideri la figura 4-11, dove sono illustrate quattro stazioni senza filo. Per i nostri scopi non importa quali siano le stazioni base e quali quelle portatili. Il campo radio è tale che A e B sono nel campo l'una dell'altra e possono potenzialmente interferire tra loro. Anche C potrebbe potenzialmente interferire con B e D ma non con A.

Consideriamo per prima cosa, cosa succede quando A trasmette a B, come mostrato nella figura 4-11 (a). Se C sta ascoltando non sentirà A perché è fuori campo e quindi concluderà, sbagliando, di poter trasmettere. Se C inizia a trasmettere interferirà su B danneggiando i frame di A. Il problema di una stazione incapace di individuare i potenziali competitori per la trasmissione poiché sono fuori area, viene detto **problema della stazione nascosta**.

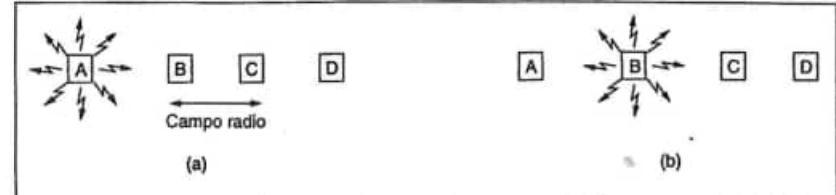


Fig. 4-11 Una LAN senza filo (a) mentre A trasmette (b) mentre B trasmette.

Consideriamo ora il problema opposto: B che cerca di trasmettere ad A, come in figura 4-11 (b). Se C è in ascolto, si accorgerà di una trasmissione uscente e concluderà, sbagliando, di non poter trasmettere verso D, quando invece la trasmissione verrebbe distorta solo se fosse nella zona tra B e C. Questa situazione viene detta **problema della stazione esposta**.

La questione è che prima di iniziare una trasmissione, una stazione vuole conoscere con certezza se c'è attività nella zona del ricevente. Il CSMA dice solo se c'è attività vicino alla stazione in ascolto sul canale. Quando si ha un cavo, tutti i segnali vengono propagati verso tutte le stazioni quindi solo una trasmissione alla volta può avere luogo in qualsiasi punto del sistema. In un sistema basato su onde radio a basso raggio le trasmissioni multiple possono avvenire simultaneamente se hanno tutte diversa destinazione e queste destinazioni sono fuori campo l'una dall'altra.

Un altro modo di raffigurare questo problema è quello di immaginare un edificio di uffici dove ogni impiegato abbia un computer portatile. Si supponga che Linda voglia spedire un messaggio a Mauro. Il computer di Linda ascolterà l'ambiente locale e, in caso non rilevi attività inizierà a spedire. Si può comunque avere una collisione nell'ufficio di Mauro poiché una terza persona può avere iniziato una trasmissione verso di lui da un luogo lontano da Linda senza che il suo computer se ne sia accorto.

MACA e MACAW

Uno dei primi protocolli progettati per LAN senza filo è **MACA** (**M**ultiple **A**ccess with **C**ollision **A**voidance – multiaccess esente da collisioni) (Karn, 1990). Esso veniva usato come base nello standard IEEE 802.11 per le LAN senza filo. L'idea alla base è che il mittente stimolasse il ricevente nell'emettere un piccolo pacchetto: quindi le stazioni vicine potevano rilevare questa trasmissione ed evitare di trasmettere per la durata del pacchetto di dati (più grosso) in procinto di arrivare. MACA viene illustrato in figura 4-12.

Consideriamo il modo in cui A spedisce un pacchetto a B. A inizia spedendo un pacchetto RTS (Request To Send: richiesta di spedizione) a B, come mostrato in figura 4-12 (a). Questo piccolo pacchetto (30 byte) contiene la lunghezza del pacchetto di dati che seguirà. Quindi B risponde con un pacchetto CTS (Clear To Send – permesso di spedire) come in figura 4-12 (b). Il pacchetto CTS contiene la lunghezza del pacchetto di dati (copiata dal pacchetto RTS). Dopo aver ricevuto il pacchetto CTS, A inizia la trasmissione.

Osserviamo ora come reagiscono le stazioni ricevendo i due pacchetti. Qualsiasi stazione che riceva il pacchetto RTS è ovviamente vicina ad A e dovrà rimanere in silenzio finché

il pacchetto CTS non sia trasmesso ad A senza conflitti. Qualsiasi stazione che riceva il pacchetto CTS è ovviamente vicina a B e dovrà rimanere inattiva durante la seguente trasmissione dei dati, la cui lunghezza è comunicata nel pacchetto CTS.

In figura 4-12, C è nel campo di A ma non in quello di B. Per cui riceve l'RTS da A ma non il CTS da B. Poiché esso non interferisce con il CTS, potrà trasmettere durante la spedizione del pacchetto di dati. Al contrario, D è nel campo di B ma non in quello di A; non riceverà l'RTS ma riceverà il CTS. Si accorgerà quindi di essere vicino a una stazione che sta per ricevere un pacchetto di dati, quindi rinuncerà a spedire finché non sia finita la trasmissione. La stazione E riceve entrambi i frame di controllo e, come D, deve restare inattiva finché la trasmissione del frame di dati non sia completa.

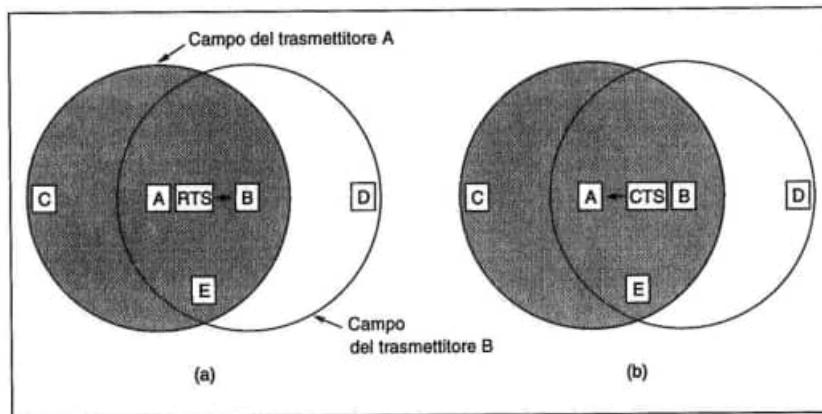


Fig. 4-12 Il protocollo MACA. (a) A spedisce un RTS a B. (b) B risponde con un CTS ad A.

Malgrado tali precauzioni, possono ancora avvenire delle collisioni. Ad esempio, B e C possono spedire simultaneamente un RTS ad A. Essi collideranno e andranno persi. Nel caso di una collisione, un trasmettitore che non ha avuto successo (cioè che non ha ricevuto un CTS entro un certo tempo) aspetterà una quantità casuale di tempo e riproverà. L'algoritmo usato è di regressione binaria esponenziale, che studieremo parlando della LAN IEEE 802.3.

Basandosi su studi di simulazione di MACA, Bharghavan e altri (1994) migliorarono le prestazioni di MACA e ridenominarono il protocollo risultante come MACAW. Per prima cosa essi notarono che senza gli ack del livello data link i frame persi non venivano ritrasmessi finché, molto dopo, il livello trasporto non avesse notato la loro assenza. Essi risolsero questo problema introducendo un frame ack dopo ogni trasmissione riuscita di un frame di dati. Essi osservarono anche che CSMA aveva diversi vantaggi, ad esempio quello di impedire a una stazione di trasmettere un RTS nello stesso momento in cui una stazione vicina lo stesse trasmettendo alla stessa destinazione, aggiunsero quindi il rilevamento del carico. Inoltre essi decisero di utilizzare l'algoritmo di regressione separata-

mente per ogni serie di pacchetti (coppie sorgente-destinazione), piuttosto che per ogni stazione. Questo cambiamento migliorò l'equità del protocollo. Infine essi aggiunsero un meccanismo perché le stazioni si scambiassero informazioni sugli ingorghi e un modo per cui l'algoritmo di regressione reagisse meno violentemente ai problemi temporanei al fine di migliorare le prestazioni del sistema.

4.2.7 Radio cellulari digitali

Una seconda categoria di reti senza filo è composta dalle reti radio cellulari digitali, successori del sistema AMPS studiato nel capitolo 2. Le radio cellulari digitali presentano un ambiente in qualche modo diverso rispetto a quello delle LAN senza filo e usano protocolli differenti. In particolare sono orientate alla telefonia, che richiede che la connessione sia stabile per alcuni minuti e non millisecondi: è quindi più efficiente allocare il canale per ogni chiamata piuttosto che per ogni frame. Tuttavia le tecniche sono ugualmente valide per il traffico di dati. In questo paragrafo considereremo tre approcci radicalmente diversi tra loro di allocazione del canale per sistemi senza filo basati su radio digitale: GSM, CDPD e CDMA.

GSM – Global System for Mobile Communication

La prima generazione di telefoni cellulari era analogica, come detto nel capitolo 2, ma l'attuale generazione è digitale e utilizza radiopacchetti. La trasmissione digitale fornisce diversi vantaggi rispetto a quella analogica, quando si parla di comunicazione mobile. Primo, voce, dati e fax possono essere integrati in un singolo sistema. Secondo, essendo stati scoperti migliori algoritmi di compressione vocale, è necessaria minor larghezza di banda per canale. Terzo, per migliorare la qualità della trasmissione possono essere usati codici di correzione di errore. Infine, i segnali digitali possono essere criptati per problemi di sicurezza.

Anche se sarebbe stato bello che tutto il mondo avesse adottato gli stessi standard digitali, questo non è accaduto; il sistema americano, IS-54 e quello giapponese, JDC, sono stati progettati per essere compatibili con qualsiasi sistema analogico esistente di qualsiasi paese, quindi ogni canale AMPS potrebbe essere usato sia per trasmissioni analogiche che digitali.

Al contrario, il sistema digitale europeo, **GSM (Global System for Mobile Communication)** – sistema globale per comunicazione mobile, è stato progettato da zero come sistema completamente digitale, senza nessun compromesso di compatibilità con le tecnologie passate (ad es. usando gli slot di frequenza esistenti). Poiché GSM è piuttosto progredito rispetto al sistema americano ed è attualmente in uso in più di cinquanta paesi, all'interno e all'esterno dell'Europa, lo considereremo come esempio di radio cellulare digitale. GSM fu originariamente progettato per un uso su una banda di 900 MHz. In seguito, le frequenze vennero allocate a 1800 MHz e venne costruito un secondo sistema, molto simile a GSM: il suo nome è **DCS 1800** ma è essenzialmente GSM.

Lo standard GSM è lungo più di 5000 pagine. Una buona parte di questo materiale tratta aspetti ingegneristici del sistema, specialmente la struttura dei riceventi che devono gestire la propagazione dei segnali multicammino e della sincronizzazione dei mittenti e dei riceventi.

Un sistema GSM ha un massimo di 200 canali full-duplex per cella. Ogni canale consiste di una frequenza di discesa (downlink) (dalla stazione base alle stazioni mobili) e di una frequenza di salita (uplink) (dalle stazioni mobili alla base). Ogni banda di frequenza è di 200 MHz come mostrato in figura 4-13.

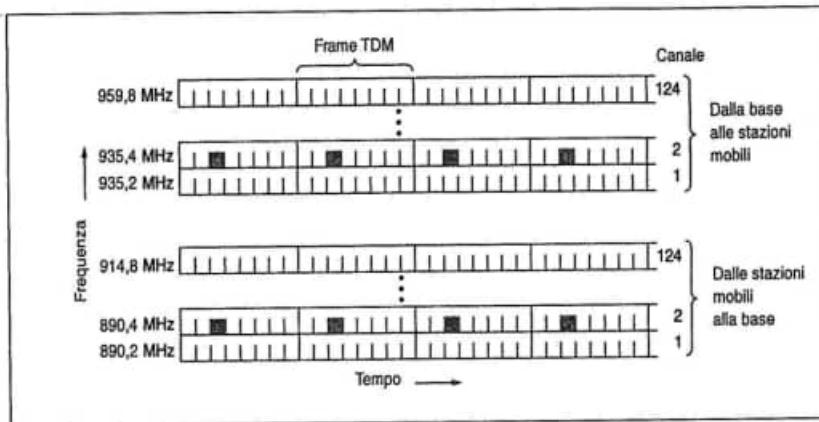


Fig. 4-13 GSM usa 124 canali di comunicazione, ognuno dei quali usa un sistema TDM a 8 slot.

Ognuno dei 124 canali di frequenza regge otto diverse connessioni usando multiplexing a divisione di tempo. Ad ogni stazione attiva viene assegnato uno slot di tempo in un canale. Teoricamente ogni cella può reggere 992 canali ma molti di essi non sono disponibili al fine di evitare conflitti di frequenze con le celle vicine. Nella figura 4-13 gli otto slot di tempo ombreggiati appartengono tutti allo stesso canale, quattro in ogni direzione. Se la stazione mobile assegnata a 890,4/935,4 MHz e allo slot 2 volesse trasmettere alla stazione base, dovrebbe usare i quattro slot ombreggiati più bassi (e quelli che li seguono temporalmente), mettendo alcuni dati in ogni slot finché tutti i dati non siano stati spediti. Gli slot TDM mostrati in figura 4-13 fanno parte della complessa gerarchia di suddivisione di frame. Ogni slot TDM ha una struttura specifica e gruppi di slot TDM formano dei multiframe, anch'essi con una struttura specifica. Una versione semplificata di questa gerarchia è in figura 4-14. Qui possiamo notare che ogni slot TDM consiste di frame di dati di 148 bit. Ogni frame di dati inizia e termina con 3 bit 0, al fine di delineare il frame. Esso contiene anche campi *Information* (informazione) di 57 bit, ognuno con un bit di controllo che indica se il successivo campo *Information* contiene voce o dati. Tra i campi *Information* si trova un campo *Sync* di 26 bit utilizzato dal ricevente per sincronizzarsi con le estremità del frame del mittente. Un frame di dati viene trasmesso in 547 ms, ma un trasmettitore non può spedire un dato ogni 4,615 ms poiché sta condividendo il canale con altre sette stazioni. La velocità complessiva di ogni canale è di 270.833 bps suddivisa per otto utenti. Trascurando tutti gli appesantimenti, ogni connessione può inviare un segnale vocale compresso oppure 9600 bps di dati.

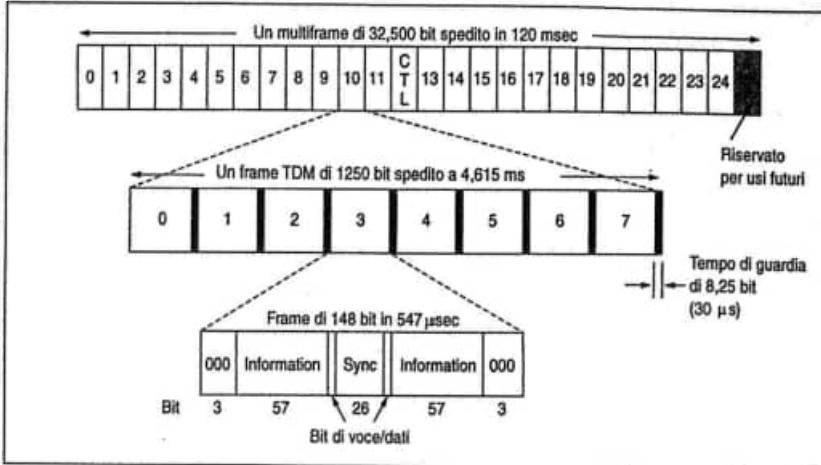


Fig. 4-14 Una porzione della struttura di frammentazione GSM.

Come si vede in figura 4-14, otto frame di dati costituiscono un frame TDM e 26 frame TDM costituiscono un multiframe di 120 ms. Dei 26 frame TDM di un multiframe, lo slot 12 viene usato per il controllo e lo slot 25 per usi futuri, quindi solo 24 slot sono disponibili per il traffico dei dati.

Tuttavia, oltre ai multiframe di 26 slot mostrati in figura 4-14, viene utilizzato un multiframe di 51 slot (non raffigurato). Alcuni di questi slot sono utilizzati per canali di controllo per la gestione del sistema. Il **canale di controllo broadcast** consiste in un flusso continuo di output dalla stazione base che contiene la sua identificazione e lo stato del canale. Tutte le stazioni mobili controllano la potenza del suo segnale per capire quando si sono spostati in una cella diversa.

Il **canale di controllo dedicato** viene usato per aggiornare le locazioni, le registrazioni e la configurazione delle chiamate. In particolare, ogni stazione base mantiene un database delle stazioni mobili attualmente sotto la sua giurisdizione. Le informazioni necessarie per mantenere questo database vengono spedite sul canale di controllo dedicato.

Infine, si ha il **canale di controllo comune**, che è suddiviso in tre sottocanali logici. Il primo di questi sottocanali è il **canale di paginazione**, usato dalla stazione base per annunciare le chiamate in arrivo. Ogni stazione mobile lo tiene sotto controllo continuamente per controllare le chiamate a cui dovrebbe rispondere. Il secondo è il **canale ad accesso casuale**, costituito da un sistema ALOHA a slot per permettere a una stazione mobile di richiedere uno slot sul canale di controllo dedicato. Usando questo slot la stazione può configurare una chiamata. Lo slot assegnato viene annunciato su di un terzo sottocanale, il **canale di garanzia di accesso**.

In generale GSM è un sistema piuttosto complesso. Gestisce gli accessi ai canali usando una combinazione di ALOHA a slot, FDM e TDM. Per ulteriori dettagli su GSM, inclusi

gli aspetti del sistema che non sono stati discussi, ad esempio l'architettura dei livelli di protocolli, si veda Rahnema (1993).

CDPD – Cellular Digital Packet Data

GSM è fondamentalmente basato su commutatori di circuito. Un computer mobile con un modem speciale può effettuare una chiamata su un telefono GSM come su un telefono normale. Tuttavia l'utilizzo di questo metodo presenta dei problemi. Per prima cosa i passaggi di consegna (handoff) tra le stazioni base sono frequenti, alcune volte anche con utenti stazionari (le stazioni base possono rimescolare gli utenti per bilanciare il carico) e ogni handoff comporta una perdita di circa 300 ms di dati. Inoltre GSM può comportare un alto grado di errore. Digitare una "a" e avere sempre un'eco di una "m" può diventare molto fastidioso. Infine le chiamate senza fili sono costose e i costi aumentano rapidamente in quanto la tariffa si basa sulla connessione e non sui byte spediti.

Un approccio alla soluzione di questi problemi è il servizio di datagrammi digitali basati su commutazione di pacchetto, anche detto **CDPD** (Cellular Digital Packet Data – dati in pacchetti cellulari digitali). È costruito su di AMPS (si veda il capitolo 2) ed è interamente compatibile con AMPS. Ogni canale inattivo di 30 kHz può essere occupato per la spedizione di pacchetti alla velocità di 19,2 kbps. Poiché CDPD implica un overhead notevole, la velocità di trasmissione netta è circa 9600 bps. E tuttavia, un sistema senza connessione e senza filo di datagrammi, per inviare, ad esempio, pacchetti IP usando il sistema di telefonia cellulare esistente, è una idea interessante per molti utenti e il suo uso sta crescendo rapidamente.

CDPC segue fondamentalmente il modello OSI. Il livello fisico ha a che fare con i dettagli della modulazione e della trasmissione radio, che non ci riguarda in questo contesto. I protocolli di livello data link, rete e trasporto esistono ma non ci interessano a questo punto. Per ulteriori dettagli riguardo al sistema CDPD completo si vedano Quick, Balachandran (1993).

Un sistema CDPD consiste di tre tipi di stazioni: gli host mobili, le stazioni base e le stazioni di interfaccia di base (in gergo CDPD, rispettivamente: sistemi punto mobili, sistemi di base di dati e sistemi mobili intermedi). Queste stazioni interagiscono con gli host stazionari e router standard, che troviamo in ogni WAN. Gli host mobili sono i computer portatili degli utenti. Le stazioni di base sono i trasmettitori che comunicano con gli host mobili. Le stazioni di interfaccia di base sono nodi speciali che interfacciano tutte le stazioni base in una area di un fornitore CDPD a un router standard (fisso) per ulteriori trasmissioni a Internet o ad altre WAN. Questo modello è mostrato in figura 4.15. In CDPD vengono definiti tre tipi di interfaccia. L'**interfaccia E** (esterna al fornitore CDPD) connette un'area CDPD a una rete fissa. Questa interfaccia deve essere ben definita per permettere a CDPD di connettersi a diversi tipi di rete. L'**interfaccia I** (interna al provider CDPD) connette due aree CDPD. Deve essere standardizzata per permettere agli utenti di passare da un'area all'altra. La terza è l'**interfaccia A** (interfaccia aerea) tra la stazione base e gli host mobili. Questa è la più interessante, quindi la esamineremo in dettaglio.

I dati sull'interfaccia area sono spediti usando la compressione, la codifica e la criptazione degli errori. Le unità di 274 bit di dati compressi e crittati sono compattati in blocchi di

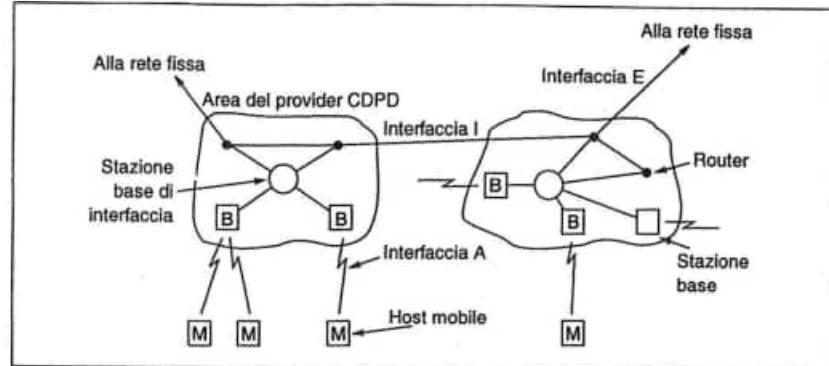


Fig. 4-15 Un esempio di sistema CDPD.

378 bit usando il codice di correzione di errore Reed-Solomon. Ad ogni blocco RS vengono aggiunte 7 parole di controllo di 6 bit, per un totale di 420 bit per blocco. Ogni blocco di 420 bit viene suddiviso in sette microblocchi di 60 bit, che vengono inviati in sequenza. Ogni microblocco ha una sua parola di controllo di 6 bit usata per indicare lo stato del canale. Questi microblocchi viaggiano su di un canale di discesa di 19,2 kbps (proveniente dalla base) e su di un canale di salita di 19,2 kbps (verso la base), in modo full-duplex. I due canali sono entrambi suddivisi in slot, cioè sequenze di microblocchi di 60 bit. Ogni microblocco dura 3,125 ms.

Ogni cella CDPD ha a disposizione solo una coppia di canali (discesa/salita) per i dati. Il canale di discesa è di facile gestione poiché c'è solo un mittente per cella: la stazione base. Tutti i frame spediti su di esso vengono diffusi via broadcast e ogni host mobile seleziona quelli diretti a lui e quelli diretti a tutti.

La parte più complicata corrisponde alla gestione del canale di salita, che viene conteso tra tutti gli host che vogliono trasmettere. Quando un host mobile deve spedire un frame osserva il canale di discesa cercando un bit di controllo che gli indichi se l'attuale slot del canale di salita è impegnato o vuoto. Se è impegnato, invece di aspettare il prossimo slot, aspetterà un numero casuale di slot prima di riprovare. Se lo troverà ancora occupato aspetterà un tempo casuale ancora più lungo e ripeterà la procedura. Il tempo di attesa statistico medio raddoppia per ogni tentativo non riuscito. Quando finalmente trova il canale libero trasmetterà il suo microblocco.

Lo scopo di questo algoritmo, detto **DSMA** (Digital Sense Multiple Access – multiaccesso a rilevamento digitale), è quello di impedire agli host mobili di occupare il canale non appena questo si libera. In qualche modo esso assomiglia al protocollo CSMA persistente a slot, descritto in precedenza, in quanto anch'esso utilizza slot discreti di tempo su entrambi i canali.

Il problema è che, nonostante DSMA, è ancora possibile la collisione tra due host mobili, infatti due o più di essi possono occupare lo stesso slot di tempo e iniziare a trasmettere. Per permettere agli host mobili di capire se è avvenuta una collisione, un bit in ogni

microblocco indica se un precedente microblocco sul canale di salita è stato ricevuto correttamente. Sfortunatamente la stazione base non può determinare istantaneamente la cosa dopo la terminazione di un microblocco, quindi la ricezione corretta/scorretta del microblocco n viene ritardata fino al microblocco $n + 2$.

Poiché non può dire se la sua trasmissione abbia avuto successo, un mittente che abbia più microblockchi da spedire andrà avanti nella trasmissione senza dover riconquistare il canale. Se nello slot di tempo successivo nota che la sua trasmissione precedente è andata persa si blocca. Altrimenti continua a trasmettere fino a un numero massimo di blocchi Reed-Solomon o finché la stazione base non pone un bit di controllo sul canale di discesa a indicare di aver ricevuto abbastanza, per il momento, da quel particolare mittente.

Una proprietà aggiuntiva di CDPD è che gli utenti dei dati sono cittadini "di seconda classe". Quando una nuova chiamata vocale sta per essere assegnata a un canale usato da CDPD, la stazione base spedisce un segnale speciale sul canale di discesa, chiudendolo. Se la stazione base conosce già il numero del nuovo canale CDPD, lo comunica. Altrimenti gli host mobili devono cercarlo tra i potenziali canali CDPD. In questo modo CDPD può sfruttare la potenziale inattività di ogni cella senza interferire con la fonte dei soldi veri, cioè il traffico vocale.

Dovrebbe essere chiaro da questa descrizione che CDPD fu aggiunto al sistema vocale dopo che questo era già attivo e che il suo progetto era limitato a non introdurre cambiamenti al sistema vocale esistente. Di conseguenza quando si ha una selezione di canale per chiamate vocali, l'algoritmo non sa dell'esistenza di CDPD. Questo è il motivo per cui il canale CDPD viene a volte improvvisamente rilasciato. Tuttavia niente nel progetto impedisce di avere canali CDPD dedicati. La sua crescita di popolarità spinge i fornitori a riservare canali esclusivamente per CDPD.

CDMA – Code Division Multiple Access

GSM può essere descritto come una soluzione brutale al problema dell'allocazione dei canali. È una combinazione di tutte le tecniche conosciute (ALOHA, TIM, FDM) correlate in modi complessi. CDPD per trasmissione di frame singoli è praticamente CSMA non persistente. Ora esamineremo un altro metodo di allocazione di canali senza filo, **CDMA (Code Division Multiple Access – multiaccesso a suddivisione di codice)**.

CDMA è completamente diverso da tutte le altre tecniche studiate fino ad ora. Alcune di queste erano basate sull'idea di suddividere il canale in bande di frequenza da assegnare staticamente (FDM) o su richiesta (multiplexing a suddivisione di lunghezza d'onda), dopodiché l'utente utilizzava la banda finché voleva. Altre allocavano il canale a intervalli, assegnando alle stazioni l'intero canale staticamente (TDM con slot di tempo fissati) o dinamicamente (ALOHA). CDMA permette a ogni stazione di spedire su tutto lo spettro di frequenza per tutto il tempo. Le trasmissioni multiple simultanee vengono separate usando la teoria della codifica. CDMA rilascia anche l'assunzione che i frame che collidono siano totalmente danneggiati: si assume invece, che i segnali multipli si sommino linearmente.

Prima di inoltrarci nello studio dell'algoritmo consideriamo la teoria del "ricevimento da cocktail" di accesso al canale. In un'ampia sala molte coppie di persone stanno conversando. Usando TDM tutte le persone sono in mezzo alla sala e parlano a turno. Con FDM

le persone si dividono in grossi gruppi separati, ogni gruppo porta avanti contemporaneamente una propria conversazione distinta. Con CDMA invece tutti sono nella sala e stanno parlando contemporaneamente a coppie, ogni coppia però in lingua diversa. La coppia che parla in francese ascolta solo ciò che è in francese considerando qualsiasi altra lingua come rumore. Quindi il concetto chiave di CDMA è quello di essere in grado di estrarre il segnale desiderato rifiutando tutto il resto come rumore casuale.

In CDMA ogni tempo di bit è suddiviso in m piccoli intervalli detti **chip** (frammenti). Tipicamente ci sono 64 o 128 chip per bit ma nell'esempio considerato di seguito consideriamo per semplicità 8 chip per bit.

Ad ogni stazione viene assegnato un codice unico di m bit (**sequenza di chip**). Per trasmettere un bit 1 una stazione invia la sua sequenza di chip. Per trasmettere uno 0 invia il complemento della sua sequenza di bit. Nessun'altra configurazione è permessa. Quindi per $m = 8$, se la stazione A ha come sequenza di chip 00011011, essa invierà un bit 1 inviando 00011011 e un bit 0 inviando 11100100.

È possibile aumentare della quantità di informazione da trasmettere da b bit/s a mb bit/s, solo se la larghezza di banda disponibile viene aumentata di un fattore m , rendendo CDMA una forma di comunicazione ad ampio spettro (assumendo che non ci siano costi per le tecniche di modulazione e di codifica). Se consideriamo una banda di 1 MHz disponibile per 100 stazioni, con FDM, ognuna avrebbe 10 MHz e potrebbe spedire a 10 bps (assumendo 1 bit per Hz). Con CDMA ogni stazione usa l'intero 1 MHz quindi la velocità del chip è un megachip per secondo. Con meno di 100 chip per bit, la larghezza di banda effettiva per stazione è maggiore per CDMA che per FDM e il problema dell'allocazione del canale è risolto, come si può vedere di seguito.

Per motivi didattici usiamo una notazione bipolare con lo 0 binario codificato come -1 e l'1 binario come +1. Mostreremo le sequenze di chip tra parentesi, quindi un 1 binario per la stazione A diventa ora (-1-1-1+1+1-1+1). In figura 4-16 (a) mostriamo la sequenza di chip binaria assegnata a quattro stazioni esempio. In figura 4.16 (b) mostriamo gli stessi numeri in notazione bipolare.

Ogni stazione ha la sua sequenza di chip. Usiamo il simbolo S per indicare il vettore di m chip per la stazione S ed \bar{S} per la sua negazione. Tutte le sequenze di chip sono ortogonali a coppie, cioè il prodotto interno normalizzato di due qualsiasi sequenze di chip, $S \bullet T$ (scritto come $S \bullet T$) è 0. In termini matematici,

$$S \bullet T = \frac{1}{m} \sum_{i=1}^m S_i T_i = 0 \quad (4-5)$$

A parole, quante coppie uguali ci sono, tante se ne hanno di diverse. Questa proprietà di ortogonalità si rivelerà cruciale in seguito.

Si noti che $S \bullet T = 0$ implica $S \bullet \bar{T} = 0$. Il prodotto interno normalizzato di ogni sequenza di chip per se stessa è 1:

A: 00011011	A: (-1 -1 -1 +1 +1 -1 +1 +1)
B: 00101110	B: (-1 -1 +1 -1 +1 +1 +1 -1)
C: 01011100	C: (-1 +1 -1 +1 +1 +1 -1 -1)
D: 01000010	D: (-1 +1 -1 -1 -1 +1 -1)

(a)

(b)

Sei esempi:

--1- C	$S_1 = (-1 +1 -1 +1 +1 +1 -1 -1)$
-11- B + C	$S_2 = (-2 \ 0 \ 0 \ 0 +2 +2 \ 0 -2)$
10-- A + B	$S_3 = (\ 0 \ 0 -2 +2 \ 0 -2 \ 0 +2)$
101- A + B + C	$S_4 = (-1 +1 -3 +3 -1 -1 -1 +1)$
1111 A + B + C + D	$S_5 = (-4 \ 0 -2 \ 0 +2 \ 0 +2 -2)$
1101 A + B + C + D	$S_6 = (-2 -2 \ 0 -2 \ 0 -2 +4 \ 0)$

(c)

$$\begin{aligned} S_1 \cdot C &= (1 +1 +1 +1 +1 +1 +1)/8 = 1 \\ S_2 \cdot C &= (2 +0 +0 +0 +2 +2 +0 +2)/8 = 1 \\ S_3 \cdot C &= (0 +0 +2 +2 +0 -2 +0 -2)/8 = 0 \\ S_4 \cdot C &= (1 +1 +3 +3 +1 -1 +1 -1)/8 = 1 \\ S_5 \cdot C &= (4 +0 +2 +0 +2 +0 -2 +2)/8 = 1 \\ S_6 \cdot C &= (2 -2 +0 -2 +0 -2 -4 +0)/8 = -1 \end{aligned}$$

(d)

Fig. 4-16 (a) Sequenze di chip binari per 4 stazioni. (b) Sequenze di chip bipolar. (c) Sei esempi di trasmissione. (d) Ricostruzione del segnale della stazione C.

$$S \cdot S = \frac{1}{m} \sum_{i=1}^m S_i S_i = \frac{1}{m} \sum_{i=1}^m S_i^2 = \frac{1}{m} \sum_{i=1}^m (\pm 1)^2 = 1$$

Si ottiene questo risultato poiché ognuno degli m termini nel prodotto interno è 1, quindi la somma è m . Si noti anche che $S \cdot \bar{S} = -1$.

Durante ogni tempo di bit una stazione può trasmettere un 1 inviando la sua sequenza di chip e uno 0 inviando il complemento della sequenza, oppure può rimanere inattiva e non trasmettere nulla. Per il momento assumiamo che tutte le stazioni siano sincronizzate cioè che tutte le sequenze di chip inizino nello stesso istante.

Quando due o più stazioni trasmettono simultaneamente i loro segnali bipolar si sommano linearmente. Ad esempio se in un intervallo di un chip tre stazioni emettono un +1 e una stazione emette -1 il risultato sarà +2. Si può pensare a questo come a una somma di voltaggio: tre stazioni che emettono +1 volt e una che emette -1 volt portano a 2 volt. In figura 4-16 (c) sono illustrati sei esempi di una o più stazioni che trasmettono nello stesso istante. Nel primo esempio C trasmette un bit 1 quindi otteniamo la sequenza di chip di C. Nel secondo esempio entrambi B e C trasmettono un bit 1 e otteniamo la somma delle loro sequenze di chip bipolar, cioè:

$$(-1 -1 +1 -1 +1 +1 +1 -1) + (-1 +1 -1 +1 +1 +1 -1) = (-2 \ 0 \ 0 \ 0 +2 +2 \ 0 -2)$$

Nel terzo esempio la stazione A invia un 1 e la stazione B uno 0. Le altre stazioni sono inattive. Nel quarto esempio A e C inviano un 1 mentre B invia uno 0. Nel quinto esempio tutte e quattro le stazioni inviano un bit 1. Infine nell'ultimo esempio, A, B e D inviano un 1 mentre C invia un bit 0. Si noti che ognuna delle cinque sequenze S_1 fino a S_6 date in figura 4.16 (c) rappresenta un unico tempo di bit.

Per ritrovare un flusso di bit di una stazione il ricevente deve sapere in anticipo la sequenza di chip di quella stazione. Esso la ritrova calcolando il prodotto interno normalizzato della sequenza di chip ricevuta (la somma lineare di tutte le stazioni che hanno trasmesso) e la sequenza di chip della stazione di cui sta cercando di recuperare il flusso di bit. Se la sequenza di chip ricevuta è S e il ricevente sta tentando di ascoltare una stazione la cui sequenza di bit è C , esso calcola il prodotto interno normalizzato $S \cdot C$. Per capire come mai questa tecnica funzioni si considerino due stazioni A e C che trasmettono entrambe un bit 1 nello stesso istante in cui B trasmette un bit 0. Il ricevente ne osserverà la somma: $S = A + B + C$ e calcolerà:

$$S \cdot C = (A + B + C) \cdot C = A \cdot C + B \cdot C + C \cdot C = 0 + 0 + 1 = 1$$

I primi due termini spariscono in quanto tutte le coppie di sequenze di chip sono state scelte ortogonali come mostrato nell'equazione (4-5). Ora dovrebbe essere chiaro perché questa proprietà deve essere imposta sulla sequenza di chip.

Un modo alternativo di pensare questa situazione è quello di immaginare che le tre sequenze di chip arrivino separatamente piuttosto che sommate. Il ricevente quindi calcolerebbe il prodotto interno di ognuna separatamente e sommerebbe i risultati. Per la proprietà di ortogonalità, tutti i prodotti interni eccetto $C \cdot C$ sarebbero 0. Sommarli e calcolare il prodotto interno è esattamente la stessa cosa di calcolare i prodotti interni e sommarli.

Per rendere più concreto il processo di decodifica consideriamo ancora i sei esempi della figura 4-16 (c). Supponiamo che il ricevente sia interessato esattamente al bit spedito dalla stazione C da una delle sei somme S_1, \dots, S_6 . Esso calcola il bit sommando i prodotti delle S ricevute e del vettore C della figura 4-16 (b) e considerando 1/8 del risultato (poiché $m = 8$). Come mostrato, ogni volta viene determinato il bit esatto. È proprio come parlare francese.

In un sistema CDSM ideale, esente da rumore, la capacità (cioè il numero di stazioni) può essere grande a piacere proprio come la capacità di un canale esente da rumore Nyquist può essere grande a piacere usando più e più bit per volta. In pratica le limitazioni fisiche riducono considerabilmente questa capacità. Primo, abbiamo assunto che le sequenze di chip siano sincronizzate, in realtà questo è impossibile. Quello che si può ottenere è che il mittente e il ricevente si sincronizzino facendo in modo che il mittente trasmetta una sequenza di chip nota e sufficientemente lunga in modo che il ricevente possa agganciarla. Tutte le altre trasmissioni non sincronizzate vengono viste come rumori casuali. Se non ce ne sono troppe l'algoritmo di decodifica di base continua a funzionare bene.

Esiste un'ampia quantità di teoria in relazione alla sovrapposizione delle sequenze di chip rispetto al livello di rumore (Pickholtz et al. 1982). Come ci si potrebbe aspettare, più lunga è la sequenza di chip, più alta è la probabilità di determinarla correttamente in presenza di rumore. Per aumentare la sicurezza, la sequenza di bit può utilizzare un codice di correzione di errore. Le sequenze di chip non usano mai codici di correzione di errore. Un'assunzione implicita nella precedente discussione è che i livelli di potenza di tutte le stazioni sono gli stessi di quelli ricevuti dal ricevente. CDMA tipicamente viene utilizzato per sistemi senza filo con una stazione base fissa e più stazioni mobili a distanza diversa da essa. I livelli di potenza ricevuti dalla stazione base dipendono da quanto lontani sono i trasmettitori. Una buona euristica è quella che ogni stazione mobile trasmetta alla stazione base a un livello di potenza inverso rispetto a quello con cui riceve dalla stazione base, in modo che una stazione che riceva un segnale debole dalla base ne usi uno più forte di quello che userebbe per spedire dopo aver ricevuto un segnale forte. La stazione base può dare comandi esplicativi alle stazioni mobili per aumentare o diminuire la loro potenza di trasmissione.

Abbiamo anche assunto che il ricevente sappia chi è il mittente. In linea di principio, data una sufficiente capacità di calcolo, il ricevente può ascoltare tutti i mittenti insieme usando in parallelo l'algoritmo di decodifica per ciascuno. In realtà è più facile a dirsi che a farsi. CDMA ha molti altri fattori che aumentano la complessità che sono stati omessi nella precedente descrizione. Tuttavia, CDMA è una tecnica brillante che viene sempre più introdotta quando si ha a che fare con comunicazione senza filo.

I lettori con solide basi di ingegneria elettronica che vogliono ottenere una comprensione più approfondita di CDMA dovrebbero leggere Viterbi (1995). Una tecnica alternativa di suddivisione in termini di tempo e non di frequenza, viene descritta in Crespo et al. (1995).

4.3 Standard IEEE 802 per LAN e MAN

Abbiamo terminato la trattazione generale dei protocolli astratti di allocazione del canale ed è giunto il momento di vedere come questi concetti si applichino ai sistemi reali, in particolare alle LAN. Come discusso nel paragrafo 1.7.2, IEEE ha prodotto diversi standard per le LAN. Questi standard, globalmente noti come **IEEE 802**, includono CSMA/CD, token bus e token ring. I vari standard differiscono nei livelli fisico e MAC ma sono compatibili al livello data link. Gli standard IEEE 802 sono stati adottati dall'ANSI come standard nazionali americani, dal NIST come standard governativi e dall'ISO come standard internazionali (noti come ISO 8802). Essi sono sorprendentemente leggibili (come dovrebbero essere gli standard).

Gli standard sono suddivisi in parti, ognuna pubblicata in un libro separato. Lo standard 802.1 dà un'introduzione dell'insieme degli standard e definisce le primitive di interfaccia. Lo standard 802.2 descrive la parte più alta del livello data link, che usa il protocollo **LLC (Logical Link Control – controllo di collegamento logico)**. Le parti da 802.3 a 802.5 descrivono le tre LAN standard, CSMA/CD, token bus e token ring rispettivamente. Ogni standard comprende il protocollo di livello fisico e di sottolivello MAC. I prossimi paragrafi trattano questi tre sistemi. Ulteriori dettagli si trovano in Stallings (1993b).

4.3 Standard IEEE 802 per LAN e MAN

4.3.1 Lo standard IEEE 802.3 ed Ethernet

Lo standard IEEE 802.3 è per una LAN CSMA/CD 1-persistent. Per richiamare l'idea, quando una stazione vuole trasmettere ascolta il cavo. Se il cavo è impegnato la stazione aspetta finché non diventa inattivo; altrimenti trasmette immediatamente. Se due o più stazioni iniziano simultaneamente a trasmettere su di un cavo libero, ci sarà collisione. Tutte le stazioni coinvolte nella collisione termineranno la loro trasmissione, attenderanno una quantità di tempo casuale e ripeteranno l'intero procedimento.

Lo standard 802.3 ha una storia interessante. Il primo sistema era un ALOHA costruito per permettere la comunicazione radio tra macchine distribuite sulle isole Hawaii. Quindi venne aggiunto il rilevamento del carico e la Xerox PARC costruì un sistema CSMA/CD a 2,94 Mbps per connettere più di 100 workstation su di un cavo di 1 km (Metcalfe, Boggs, 1976). Questo sistema venne chiamato **Ethernet** dal *luminiferous ether*, attraverso il quale si pensava che le onde elettromagnetiche si propagassero. (Quando il fisico britannico del XIX secolo James Clerk Maxwell scoprì che le radiazioni elettromagnetiche potevano essere descritte da una equazione d'onda, gli scienziati assunsero che lo spazio doveva essere costituito da una sostanza eterea in cui le radiazioni si propagavano. Solo dopo il famoso esperimento Michelson-Morley del 1887 i fisici scoprirono che le radiazioni elettromagnetiche possono propagarsi nel vuoto).

La Ethernet della Xerox ebbe così tanto successo che la Xerox, la DEC e la Intel decisero uno standard per una Ethernet a 10 Mbps. Questo standard è alla base dell'802.3. Lo standard 802.3 pubblicato differisce dalla specifica di Ethernet nel fatto che descrive un'intera famiglia di sistemi CSMA/CD 1-persistenti, a una velocità che varia da 1 a 10 Mbps su diversi mezzi. Inoltre, il campo intestazione di uno differisce dall'altro (in Ethernet il campo lunghezza di 802.3 è usato per il tipo del pacchetto). Lo standard iniziale dà anche i parametri per un sistema basato su di una banda di 10 Mbps che usi un cavo coassiale di 50Ω (ohm). La definizione degli altri parametri e delle velocità venne aggiunta in seguito.

Molte persone, erroneamente, usano la parola "Ethernet" in senso generico per riferirsi a tutti i protocolli CSMA/CD anche se in realtà essa si riferisce a un prodotto specifico che più o meno implementa l'802.3. Nei prossimi paragrafi useremo i termini "802.3" e "CSMA/CD" eccetto quando ci riferiamo specificamente al prodotto Ethernet.

Cablaggio 802.3

Poiché il nome "Ethernet" si riferisce a un mezzo (l'etere), inizieremo la nostra dissertazione da questo. Sono utilizzati cinque tipi di collegamento come mostrato in figura 4-17. Storicamente il primo è stato il cablaggio **10Base5**, comunemente detto **thick Ethernet** (Ethernet spessa). Essa assomiglia ai tubi gialli da giardino, con tacche ogni 2,5 m che indicano dove si agganciano le spine (lo standard 802.3 non richiede che il cavo sia proprio giallo ma lo suggerisce). Le connessioni vengono fatte usando **spine a vampiro** (vampire taps), con uno spillo che viene inserito nel cuore del tubo coassiale. La notazione 10Base5 indica che esso opera a 10 Mbps e usa segnalazione di banda di base e può supportare segmenti lunghi fino a 500 m.

Nome	Cavo	Max segmento	Nodi/segmento	Vantaggi
10Base5	Coax grosso	500m	100	Buono per dorsali
10Base2	Coax sottile	200m	30	Sistema economico
10Base-T	Doppino	100m	1024	Manutenzione semplice
10Base-F	Fibre ottiche	2000m	1024	Collegare edifici

Fig. 4-17 I tipi più comuni di banda di base delle LAN 802.3.

Storicamente il secondo tipo di collegamento è stato quello **10Base2 o thin Ethernet** (Ethernet sottile), che diversamente da thick Ethernet si piega facilmente. Le connessioni sono fatte usando i connettori industriali standard BNC che formano giunzioni a T invece che con le spine a vampiro. Essi sono più facili da usare e più affidabili. La thin Ethernet è meno costosa e più facile da installare ma collega fino a 200 m e può gestire solo 30 macchine per segmento di cavo.

Individuare rotture del cavo, spine rotte e connettori staccati diventa il problema più serio per entrambi i collegamenti. Per questo motivo sono state sviluppate tecniche speciali. Fondamentalmente un impulso di tipo noto viene inviato sul cavo: se l'impulso incontra un ostacolo o la fine del cavo, sarà generata un'eco che tornerà indietro. Cronometrando attentamente l'intervallo tra la spedizione dell'impulso e la ricezione dell'eco è possibile localizzare l'origine dell'eco. Questa tecnica è nota come **riflessometria a dominio di tempo**.

Il problema associato con la determinazione delle rotture del cavo ha portato alla costruzione di nuovi modelli di collegamento in cui tutte le stazioni sono connesse con un fulcro centrale (**hub**). Di solito questi collegamenti sono costituiti da doppini telefonici, poiché molti edifici sono già connessi in questo modo e questi doppini esistono in abbondanza. Questa tecnica è detta **10Base-T**.

Questi tre metodi di collegamento sono illustrati in figura 4-18. Per il metodo 10Base5 un **transceiver** (ricetransmettitore) viene collegato con il cavo in modo che la spina sia effettivamente in contatto con il cuore del cavo. Il transceiver è composto da elementi elettronici che gestiscono il rilevamento del carico e delle collisioni. Quando viene individuata una collisione il transceiver invia un segnale non valido sul cavo per assicurarsi che anche tutti gli altri transceiver rilevino la collisione.

Con il metodo 10Base5 un cavo ricetrasmettitore (**transceiver cable**) collega il transceiver a una scheda di interfaccia nel computer. Il cavo può essere lungo fino a 50 m e connettere cinque doppini schermati. Due di questi sono dedicati ai dati in ingresso e in uscita. Altri due per l'input e l'output di segnali di controllo. Il quinto, che non sempre viene utilizzato, permette al computer di alimentare i componenti elettronici del transceiver. Alcuni transceiver possono collegare fino a otto computer, riducendo il numero di transceiver necessari.

Il cavo transceiver termina sulla scheda di interfaccia nel computer. La scheda di interfaccia contiene un processore controllore che trasmette (e riceve) pacchetti al (e dal) transceiver. Il controllore è responsabile del compattamento dei dati nell'esatto formato

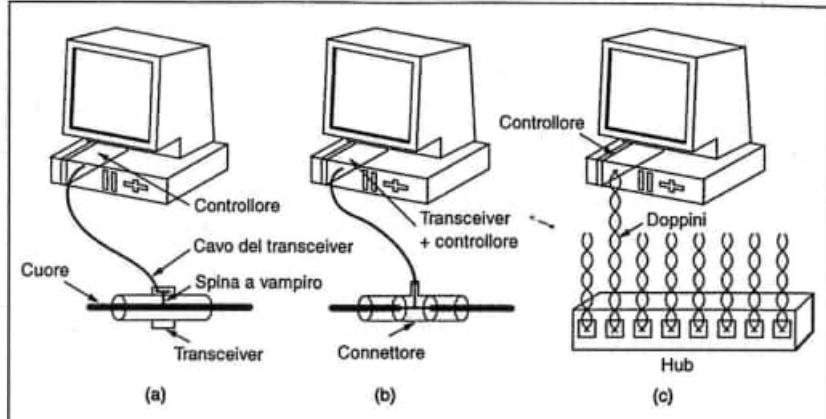


Fig. 4-18 Tre tipi di cablaggio. (a) 10Base-5. (b) 10Base-2. (c) 10Base-T.

di pacchetto, del calcolo della checksum sui pacchetti uscenti e della sua verifica sui pacchetti in ingresso. Alcuni processori controlleri gestiscono anche diversi buffer per i pacchetti in ingresso, le code di buffer dei pacchetti in uscita, i trasferimenti DMA con gli altri host e altri aspetti di gestione di rete.

Con il metodo 10Base2 la connessione con il cavo è composta da un connettore BNC a giunzione a T. I componenti elettronici del transceiver sono sulla scheda del controllore e ogni stazione ha il suo transceiver.

Con il metodo 10BaseT non si ha un cavo di collegamento ma piuttosto un fulcro centrale (hub), una scatola piena di componenti elettronici. In questa configurazione è più facile aggiungere o togliere stazioni, e le rotture del cavo possono essere rilevate con facilità. Lo svantaggio del metodo 10Base-T è che la lunghezza massima del cavo che parte dal centro è di soli 100 m, forse 150 se si utilizzano doppini di alta qualità. Inoltre, un grosso hub può costare migliaia di dollari. Nonostante questo il 10Base-T sta divenendo sempre più popolare per la sua facilità di mantenimento. Una versione più veloce di 10Base-T (100Base-T) verrà illustrata più avanti nel capitolo.

Un altro possibile metodo di collegamento per l'802.3 è il 10BaseF, che usa fibra ottica. Questa alternativa è piuttosto costosa per i prezzi dei connettori e dei terminatori ma offre livelli di rumore molto bassi ed è il metodo scelto quando si tratta di collegare edifici o hub molto lontani.

La figura 4-19 mostra diversi modi di collegamento di un edificio. In figura 4-19 (a) un singolo cavo viene passato tra una stanza e l'altra e ogni stazione collegata al cavo nel punto più vicino. In figura 4-19 (b) una dorsale collega il terreno al tetto e cavi orizzontali a ogni piano sono connessi ad essa con speciali amplificatori (ripetitori). In alcuni edifici i cavi orizzontali sono sottili e la dorsale spessa. La topologia più generale è un albero, come quello in figura 4-19 (c), poiché una rete con due cammini tra alcune coppie di stazioni darebbe la possibilità di interferenza tra i due segnali.

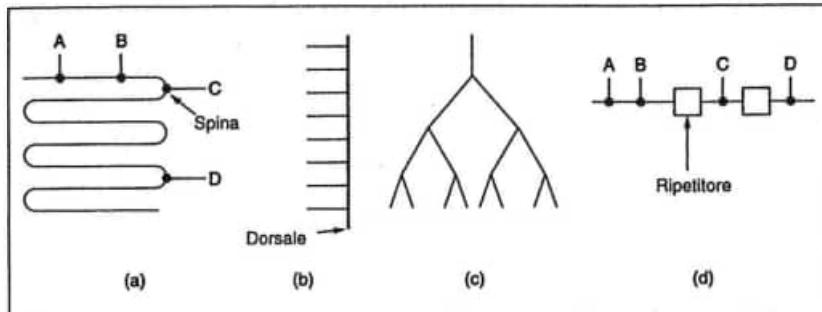


Fig. 4-19 Topologie di cablaggio. (a) Lineare. (b) Dorsale. (c) Ad albero. (d) Segmentata.

Ogni versione dell'802.3 ha una lunghezza massima per segmento. Per permettere la costruzione di reti più ampie diversi cavi possono essere connessi da ripetitori come in figura 4-19 (d). Un ripetitore è un dispositivo di livello fisico. Esso riceve, amplifica e ritrasmette segnali in entrambe le direzioni. Per quanto riguarda il software, una serie di segmenti connessi da un ripetitore non è diversa da un cavo singolo (eccetto per il ritardo introdotto dal ripetitore). Un sistema può contenere diversi segmenti di cavo e diversi ripetitori ma due transceiver non possono essere più distanti di 2,5 km e un cammino tra due transceiver non può contenere più di quattro ripetitori.

Codifica Manchester

Nessuna delle versioni di 802.3 utilizza codifica binaria diretta, dove 0 V indica un bit 0, e 5 V un bit 1, perché questo porta ad ambiguità. Se una stazione spedisce la stringa di bit 0001000, le altre potrebbero interpretarla erroneamente come 10000000 oppure 01000000 poiché non distinguono un mittente inattivo da un bit 0 (0 V).

È necessario un sistema per cui il ricevente determini in modo non ambiguo l'inizio, la metà e la fine di ogni bit senza riferimento a un orologio esterno. Due di questi approcci sono la **codifica Manchester** e la **codifica Manchester differenziale**. Con la codifica Manchester ogni periodo di bit viene suddiviso in due intervalli uguali. Un bit binario 1 viene spedito con voltaggio alto durante il primo intervallo e basso nel secondo. Uno 0 binario viene spedito esattamente nel modo opposto: prima voltaggio basso poi alto. Questa tecnica assicura che ogni periodo di bit abbia una transizione nel mezzo, rendendo più facile al ricevente la sincronizzazione col mittente. Uno svantaggio della codifica Manchester è che richiede il doppio della larghezza di banda della codifica binaria diretta, poiché gli impulsi sono metà della larghezza. La codifica Manchester è mostrata in figura 4-20 (b).

La codifica Manchester differenziale, mostrata in figura 4-20 (c), è una variante della codifica Manchester di base. Un bit 1 è indicato dall'assenza di transizione all'inizio dell'intervalle. Un bit 0 è indicato dalla presenza di una transizione all'inizio dell'intervalle. In entrambi i casi si ha una transizione nel mezzo. La tecnica differenziale richiede un'apparecchiatura più complessa ma comporta una migliore immunità al rumore. Tutti

i sistemi 802.3 a banda di base usano la codifica Manchester per la sua semplicità. Il segnale alto è di +0,85 V e il segnale basso di -0,85 V, con un valore DC di 0 V.

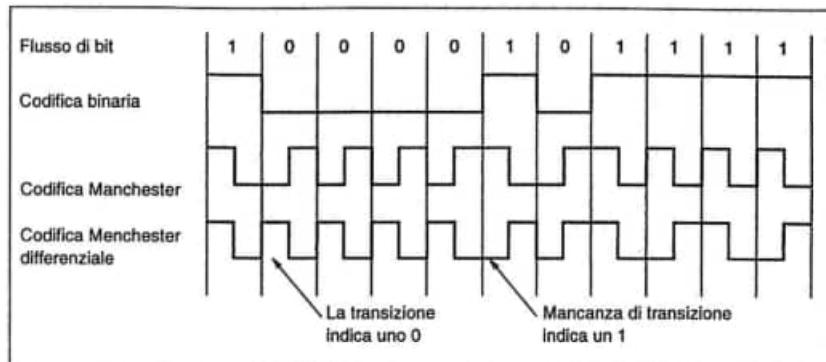


Fig. 4-20 (a) Codifica binaria. (b) Codifica Manchester. (c) Codifica Manchester differenziale.

Il protocollo del sottolivello MAC 802.3

La struttura del pacchetto in 802.3 (IEEE 1985a) è mostrata in figura 4-21. Ogni pacchetto inizia con un *Preamble* di 7 byte contenente la sequenza di bit 10101010. La codifica Manchester di questa sequenza produce un'onda quadra di 10 MHz per 5,6 µs per permettere all'orologio del ricevente di sincronizzarsi con quello del mittente. Quindi si ha un byte *Startofframe* contenente 10101011 che indica l'inizio del pacchetto stesso.

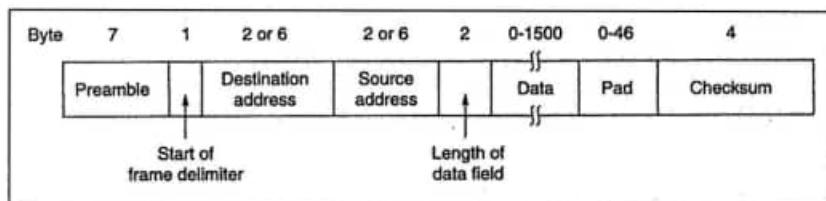


Fig. 4-21 Il formato di pacchetto 802.3.

Il pacchetto contiene due indirizzi, uno per la destinazione e uno per la sorgente. Lo standard permette indirizzi di 2 o 6 byte ma i parametri definiti per la banda base di 10 Mbps usano solo indirizzi di 6 byte. Il bit di livello più alto dell'indirizzo destinazione è 0 per gli indirizzi ordinari e 1 per indirizzi di gruppo. Gli indirizzi di gruppo permettono a più stazioni di ascoltare da un singolo indirizzo. Quando un frame viene inviato a un indirizzo di gruppo, tutte le stazioni del gruppo lo ricevono. La spedizione a un gruppo di stazioni è detta **multicast**. Un indirizzo composto di tutti 1 è riservato per il **broadcast**. Un pacchetto contenente tutti 1 nel campo destinazione viene consegnato a tutte le stazioni sulla rete.

Un'altra caratteristica interessante dell'indirizzamento è quella di usare il bit 46 (vicino al bit di ordine più alto) per distinguere gli indirizzi locali da quelli globali. Gli indirizzi locali sono assegnati dall'amministratore di rete e non hanno significato fuori dalla rete locale. Invece gli indirizzi globali sono assegnati dall'IEEE per assicurare che nessuna stazione nel mondo abbia un indirizzo uguale a un'altra. Con $48-2 = 46$ bit disponibili si possono ottenere 7×10^{13} indirizzi globali. L'idea è che una stazione possa indirizzare qualsiasi altra stazione usando l'esatto numero di 48 bit. È compito del livello rete definire il metodo di localizzazione della destinazione.

Il campo *Lenght* indica quanti byte ci sono nel campo dei dati, da un minimo di 0 a un massimo di 1500. Anche se un campo di 0 byte è legale, esso causa un problema: quando un transceiver individua una collisione tronca il frame corrente, ciò significa che bit spediti e pezzi di frame possono apparire sul cavo in qualsiasi istante. Per rendere più semplice la distinzione di un frame valido da uno corrotto, 802.3 indica che un frame valido deve essere di almeno 64 byte, compresi l'indirizzo del destinatario e la checksum. Se la porzione di dati di un frame è meno di 46 byte, un campo "cuscinetto" (Pad) viene aggiunto per raggiungere la dimensione minima di frame.

Un'altra (e più importante) ragione per avere una lunghezza minima di pacchetto è quella di evitare che una stazione possa terminare la spedizione di un pacchetto corto prima che il primo bit abbia raggiunto l'estremità più lontana del cavo, dove potrebbe collidere con un altro pacchetto. Questa situazione è illustrata in figura 4-22. Al tempo 0 la stazione A, situata a un'estremità della rete, invia un frame. Sia τ il tempo di propagazione del frame fino all'estremità opposta. Appena prima che il pacchetto abbia raggiunto l'altra estremità (cioè al tempo $\tau - \epsilon$) la stazione più distante, B, inizia a trasmettere. Quando B si accorge di ricevere più potenza di quanto sta spedendo capisce che si è avuta una collisione quindi tronca la sua trasmissione e genera un rumore di 48 bit per avvisare le altre stazioni. Al tempo 2τ circa, il mittente si accorge del rumore e anch'esso interrompe la sua trasmissione. Aspetta dunque una quantità di tempo casuale prima di riprovare.

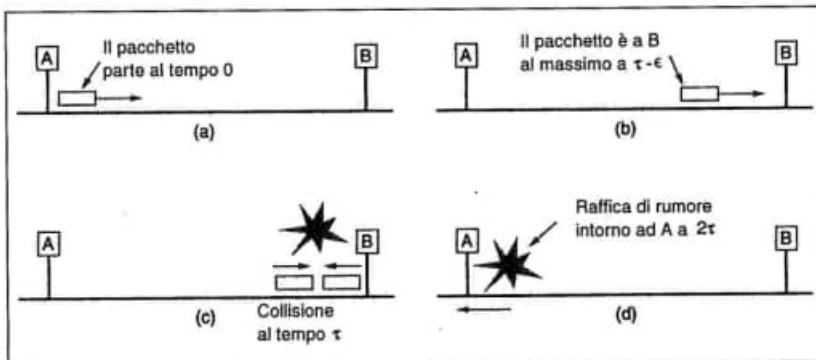


Fig. 4-22 Il rilevamento di una collisione può impiegare fino a un tempo di 2τ .

Se una stazione prova a trasmettere un pacchetto molto piccolo, è possibile che si abbia una collisione ma la trasmissione terminerà prima che il rumore sia ritornato indietro, cioè al tempo 2τ . Il mittente concluderà, erroneamente, che il frame sia stato spedito correttamente. Per prevenire questa situazione, tutti i pacchetti devono impiegare più di 2τ per essere spediti. In una LAN a 10 Mbps con una lunghezza massima di 2500 m e quattro ripetitori (dalla specifica di 208.3), un pacchetto deve impiegare almeno 51,2 μ s. Questo tempo corrisponde a 64 byte. I pacchetti piccoli vengono "imbottiti" fino a raggiungere i 64 byte.

Più aumenta la velocità della rete, più elevata deve essere la lunghezza minima del frame oppure più bassa la lunghezza massima del cavo. Per una LAN di 2500 m che opera a 1 Gbps, la lunghezza minima di frame potrebbe essere di 640 byte e la distanza massima tra due stazioni di 250 m. Queste restrizioni diventano pesanti a mano a mano che si va verso reti di gigabit.

L'ultimo campo di 802.3 è *Checksum*. Esso è un codice hash di 32 bit per i dati. Se alcuni bit di dati vengono ricevuti corrotti (per i rumori sul cavo), la checksum sarà quasi certamente errata e l'errore verrà individuato. L'algoritmo di checksum è un controllo ciclico ridondante del tipo di quelli trattati nel capitolo 3.

L'algoritmo di regressione binaria esponenziale

Consideriamo ora come viene calcolato il tempo casuale dopo una collisione. Il modello è quello della figura 4-5. Dopo una collisione il tempo viene suddiviso in intervalli discreti la cui lunghezza è uguale al caso peggiore del tempo propagazione nell'etero (e ritorno): 2τ . Per sistemare il più lungo cammino possibile in 802.3 (2,5 km e quattro ripetitori), il tempo di slot viene portato a 512 tempi di bit oppure 51,2 μ s.

Dopo la prima collisione ogni stazione aspetta 0 oppure 1 intervallo di tempo prima di riprovare. Se due stazioni collidono e ognuna aspetta lo stesso tempo casuale esse colideranno di nuovo. Dopo una seconda collisione ogni stazione sceglierà un numero a caso 0, 1, 2 o 3 e aspetterà tale numero di intervalli di tempo. Se avviene una terza collisione (la probabilità che questo accada è 0,25) la volta successiva il numero di intervalli da attendere sarà scelto casualmente fra i numeri tra 0 e 2^3-1 .

In generale dopo i collisioni, si sceglie un numero tra 0 e 2^i-1 e si aspetta quel numero di intervalli. Tuttavia, dopo dieci collisioni l'intervalllo di randomizzazione viene fissato a 1023 intervalli di tempo. Dopo 16 collisioni il controllore "getta la spugna" e notifica il fallimento al computer. Ulteriori provvedimenti sono compito dei livelli più alti.

Questo algoritmo, detto di **regressione binaria esponenziale**, venne scelto per l'adattamento dinamico al numero di stazioni che vogliono trasmettere. Se l'intervalllo di randomizzazione fosse in ogni caso 1023, la probabilità che due stazioni collidano una seconda volta sarebbe trascurabile, tuttavia il tempo di attesa medio dopo una collisione sarebbe di centinaia di tempi di slot e introdurrebbe ritardi significativi. D'altra parte se ogni stazione aspettasse sempre 0 o 1 intervallo, se 100 stazioni tentassero di trasmettere nello stesso istante colliderebbero continuamente finché 99 di esse non decidessero per 0 e una di esse per 1 o viceversa. Questo potrebbe richiedere anni. Facendo in modo che l'intervalllo di randomizzazione cresca a mano a mano che avvengono collisioni consecutive, l'algoritmo assicura un basso ritardo quando poche stazioni collidono, ma assicura anche

che le collisioni siano risolte in un intervallo di tempo ragionevole se molte stazioni collidono.

Come descritto finora, CSMA/CD non prevede ack. Poiché l'assenza di collisioni da sola non garantisce che i bit non siano corrotti da punte di rumore sul cavo, per ottenere comunicazione affidabile il destinatario dovrà verificare la checksum e, se corretta, inviare un frame di ack alla sorgente. In genere questo ack è un ulteriore pacchetto, per quanto riguarda il protocollo, e dovrà competere per il canale come gli altri pacchetti. Tuttavia, una semplice modifica all'algoritmo di contesa permetterebbe conferma rapida alla spedizione di frame (Tokoro, Tamaru, 1977). Tutto quello che sarebbe necessario è di riservare il primo slot di contesa seguente una trasmissione con successo, alla stazione destinazione.

Prestazioni di 802.3

Esaminiamo ora brevemente le prestazioni di 802.3 in condizioni di traffico pesante e carico costante, cioè k stazioni sempre pronte a trasmettere. Un'analisi rigorosa dell'algoritmo di regressione binaria esponenziale sarebbe complicata. Seguiremo invece l'approccio di Metcalfe e Boggs (1976) e assumeremo una probabilità costante di ritrasmissione per ogni slot: se ogni stazione trasmette durante uno slot di contesa con probabilità p , la probabilità A che una stazione acquisisca il controllo del canale in quello slot sarà:

$$A = kp(1-p)^{k-1} \quad (4-6)$$

A è massimo quando $p = 1/k$, con $A \rightarrow 1/e$ se $k \rightarrow \infty$. La probabilità che l'intervallo di contesa abbia esattamente j slot è $A(1-A)^{j-1}$, quindi il numero medio di slot per contesa è dato da:

$$\sum_{j=0}^{\infty} j A(1-A)^{j-1} = \frac{1}{A}$$

Poiché ogni slot ha una durata di 2τ , l'intervallo di contesa medio, w , sarà $2\tau/A$. Assumendo p ottimale, il numero medio di slot di contesa non sarà mai maggiore di e , e w sarà al massimo $2\tau e \approx 5,4\tau$.

Se un frame medio impiega P secondi per la trasmissione, quando molte stazioni devono inviare frame,

$$\text{Efficienza del canale} = \frac{P}{P + 2\tau/A} \quad (4-7)$$

Qui notiamo dove la massima distanza tra due stazioni influenza le prestazioni, dando origine a topologie diverse da quelle in figura 4-19 (a). Più lungo è il cavo, più lungo è l'intervallo di contesa. Non permettendo cavi più lunghi di 2,5 km e non più di quattro

ripetitori tra due transceiver, il tempo per percorrere tutto il giro può essere limitato a 51,2 μsec, che a 10 Mbps corrisponde a 512 bit o 64 byte, la lunghezza minima del frame. È istruttivo formulare l'equazione (4-7) in termini di lunghezza di frame, F , di larghezza di banda, B , di lunghezza del cavo, L e di velocità di propagazione del segnale, c , per il caso ottimale di e slot di contesa per frame. Con $P = F/B$, l'equazione (4-7) diventa:

$$\text{Efficienza del canale} = \frac{1}{1 + 2BLc/eF} \quad (4-8)$$

Quando il secondo termine del denominatore è grande, l'efficienza della rete sarà bassa. Più precisamente, aumentando la larghezza di banda della rete o la distanza (il prodotto BL) si riduce l'efficienza per una data dimensione di frame. Sfortunatamente, molta ricerca sull'hardware di rete ha lo scopo di aumentare questo prodotto. Si vuole una ampia larghezza di banda su distanze lunghe (ad es. MAN in fibra ottica), ciò suggerisce che forse 802.3 può non essere il modello migliore per queste applicazioni.

In figura 4-23, l'efficienza del canale viene calcolata rispetto al numero di stazioni pronte per $2\tau = 51,2 \mu\text{s}$ e una velocità di dati di 10 Mbps, usando l'equazione (4-8). Con un intervallo di tempo di 64 byte non sorprende che frame di 64 byte non siano efficienti. D'altra parte, con frame di 1024 byte e valore asintotico di e slot di 64 byte per intervallo di contesa, il periodo di contesa è di 174 byte e l'efficienza è 0,85.

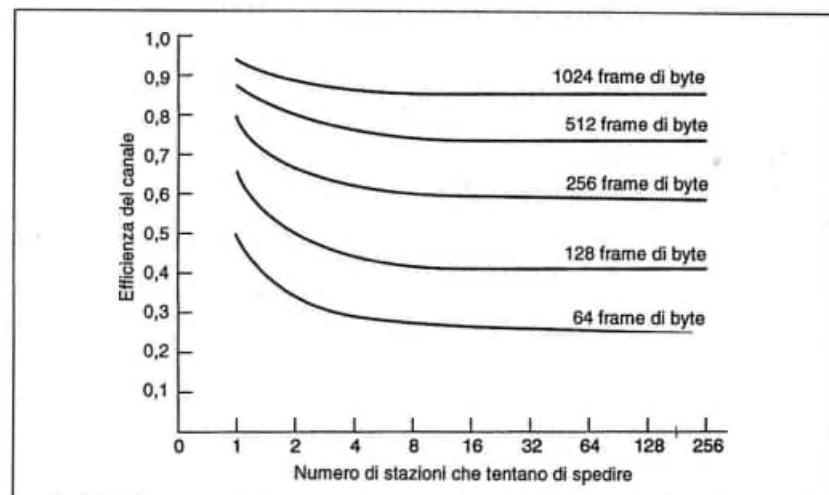


Fig. 4-23 Efficienza di 802.3 a 10 Mbps con tempi di slot di 512 bit.

Per determinare il numero medio di stazioni pronte a trasmettere in condizioni di carico pesante, possiamo usare la seguente constatazione. Ogni frame occupa il canale per un intervallo di contesa e uno di trasmissione, per un totale di $P + w$ secondi. Il numero di

frame per secondo è quindi $1/(P+w)$. Se ogni stazione genera frame alla velocità media di λ frame/s, quando il sistema è nello stato k , la velocità totale di trasmissione di tutte le stazioni attive è $k\lambda$ frame/s. Poiché all'equilibrio la velocità di input e di output devono essere identiche, possiamo egualare queste due espressioni e risolvere rispetto a k . (Si noti che w è una funzione di k). In Bertsekas e Gallager (1992) si può trovare un'analisi più dettagliata.

Vale forse la pena di dire che esistono molti studi teorici di prestazioni dell'802.3 (e di altre reti). Tutti questi lavori assumono teoricamente che il traffico di rete sia del tipo di Poisson. Da quando i ricercatori hanno iniziato a osservare i dati reali, ci si è accorti che il traffico di rete raramente ha forma di Poisson; è piuttosto autosimilare (Paxson, Floyd (1994); Willinger et al., 1995). Questo significa che fare la media su periodi di tempo molto lunghi non rende il traffico omogeneo. Il numero medio di pacchetti in ogni minuto di un'ora ha uguale variazione del numero medio di pacchetti di ogni secondo di un minuto. La conseguenza di questa scoperta è che la maggior parte dei modelli di traffico di rete non sono applicabili al mondo reale e dovrebbero essere considerati con la dovuta cautela.

LAN 802.3 commutate

Più stazioni vengono aggiunte a una LAN 802.3 più il traffico aumenta; e prima o poi la LAN sarà satura. Una soluzione è quella di aumentare la velocità, per esempio da 10 Mbps a 100 Mbps. Questa soluzione è piuttosto costosa in quanto comporta l'eliminazione di tutte le schede adattatrici e l'acquisto di nuove schede. Se i processori dell'802.3 sono sulle schede di circuito principale delle macchine può anche non essere possibile sostituirle.

Fortunatamente è possibile una soluzione diversa e meno drastica: una LAN 802.3 commutata, come quella in figura 4-24. Il cuore del sistema è un commutatore contenente un backplane ad alta velocità e spazio per 4 fino a 32 schede di linea, ognuna contenente da uno a otto connettori. Più spesso ogni connettore ha un doppino di connessione 10Base-T a un singolo host.

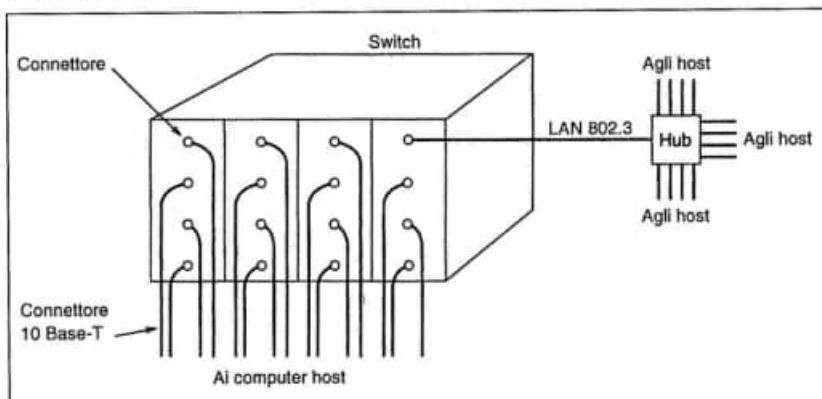


Fig. 4-24 Una LAN 802.3 commutata.

Quando una stazione vuole trasmettere un frame 802.3 emette un pacchetto standard verso il commutatore. La scheda riceve il pacchetto e controlla se è destinato a una delle altre stazioni connesse alla stessa scheda. Se sì, il pacchetto viene ricoppiato là. Altrimenti, viene spedito sul backplane ad alta velocità alla scheda della stazione di destinazione. Il backplane in genere ha una velocità di 1 Gbps o più, e usa un protocollo proprietario.

Che cosa succede se due macchine connesse alla stessa scheda trasmettono un frame nello stesso istante? Dipende da come è stata costruita la scheda. Una possibilità è che tutte le porte siano collegate in modo da formare un'unica LAN locale in linea. Le collisioni su questa LAN in linea saranno gestite e riconosciute come in una rete CSMA/DC: con ritrasmissione mediante l'algoritmo di regressione binaria. Con questo tipo di scheda è possibile una sola ritrasmissione per scheda ad ogni istante, ma tutte le schede possono trasmettere in parallelo. Con questa struttura ogni scheda forma il proprio **dominio di collisione** indipendentemente dalle altre.

Con gli altri tipi di schede ogni porta di input viene "bufferizzata", quindi i pacchetti entranti sono memorizzati, quando arrivano, sulla RAM a bordo della scheda. Questa struttura permette che tutte le porte di input ricevano (e trasmettano) frame nello stesso istante per operazioni parallele e full-duplex. Una volta che il frame è stato ricevuto completamente la scheda controlla se è destinato a un'altra porta sulla stessa scheda oppure a una porta su un'altra scheda. Nel primo caso esso può essere trasmesso direttamente a destinazione. Nell'altro caso deve essere ritrasmesso sul backplane alla scheda appropriata. Ogni porta ha un dominio di collisione separato quindi non avvengono collisioni. La produttività totale del sistema può spesso essere migliorata di un ordine di grandezza rispetto al 10Base-5, che ha un dominio di collisione unico per l'intero sistema. Poiché il commutatore aspetta i pacchetti standard 802.3 su ogni porta di input, è possibile usare alcune delle porte come concentratori. In figura 4-24 la porta nell'angolo in alto a destra non è connessa a una singola stazione ma a un concentratore hub a 12 porte. Quando i pacchetti arrivano allo hub, si contendono la LAN 802.3 nel solito modo, comprese le collisioni e la regressione binaria. I pacchetti trasmessi con successo raggiungono il commutatore e sono trattati come gli altri pacchetti in input: sono direzionati correttamente sul backplane ad alta velocità. Se tutte le porte di input sono connesse a hub piuttosto che a stazioni individuali, il commutatore diventa un bridge tra reti 802.3. Studieremo i bridge più avanti in questo capitolo.

4.3.2 Lo standard IEEE 802.4: token bus

Sebbene 802.3 sia largamente usato negli uffici, durante lo sviluppo dello standard 802 la General Motors e altre società interessate all'automazione di aziende avevano diverse riserve su di esso. Da un lato, a causa del protocollo MAC probabilistico: con un po' di sfortuna una stazione avrebbe dovuto aspettare un tempo arbitrariamente lungo prima di spedire un frame (cioè il caso peggiore non era limitato). Dall'altro, i frame 802.3 non avevano priorità e ciò li rendeva non adatti ai sistemi reali dove i frame più importanti non dovrebbero attendere il passaggio di quelli meno significativi.

Un semplice sistema con caso peggiore conosciuto è un anello (ring) dove le stazioni spediscono frame a turno. Se ci sono n stazioni che impiegano tempo T a spedire un frame, nessun frame attenderà più di nT secondi prima di essere spedito. Le persone che studia-

vano l'automazione dell'azienda nella commissione 802 apprezzarono l'idea concettuale di un anello ma non la sua implementazione fisica poiché un'interruzione sul cavo ad anello avrebbe fatto cadere l'intera rete. Inoltre, essi notarono che un anello non si adattava bene alla topologia lineare di molte linee di produzione. Come risultato, venne sviluppato un nuovo standard avente la robustezza del cavo di broadcast 802.3, ma il comportamento del caso peggiore del conosciuto caso dell'anello.

Questo standard, 802.4 (Dirvin, Miller, 1986; IEEE, 1985b), descrive una LAN chiamata **token bus**. Fisicamente è un cavo lineare o ad albero a cui le stazioni vengono attaccate. Logicamente, le stazioni sono disposte ad anello (si veda la figura 4-25), ogni stazione conosce gli indirizzi delle stazioni alla sua "destra" e alla sua "sinistra". Quando l'anello logico viene inizializzato la stazione con il numero più alto può spedire il primo frame. Dopotutto essa cede il permesso di trasmettere alla sua vicina più prossima spedendole un frame di controllo speciale detto **token** (gettone). Il token viene propagato su tutto l'anello logico e solo il possessore del token ha il permesso di trasmettere frame. Poiché solo una stazione per volta possiede il token, non si avranno collisioni.

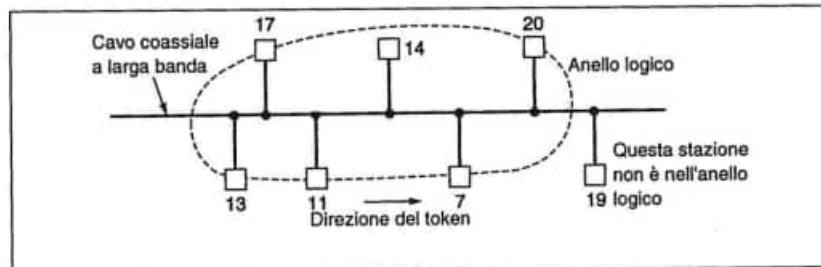


Fig. 4-25 Una LAN token bus.

Una caratteristica importante è che l'ordinamento fisico in cui le stazioni sono connesse al cavo è irrilevante. Dato che il cavo è fondamentalmente un tramite di broadcast, ogni stazione riceverà tutti i frame e scarterà quelli che non sono diretti a lei. Quando una stazione trasmette il token, spedisce un frame indirizzato alla sua vicina nell'anello logico, senza tenere presente la sua locazione fisica. Vale anche la pena notare che quando le stazioni vengono attivate per la prima volta, esse non saranno nell'anello (come le stazioni 14 e 19 in figura 4-25), quindi il protocollo MAC deve avere la capacità di aggiungere ed eliminare stazioni dall'anello.

Il protocollo MAC 802.4 è molto complesso: ogni stazione deve mantenere dieci diversi timer e più di due dozzine di variabili di stato interno. Lo standard 802.4 è molto più lungo dell'802.3 e riempie più di 200 pagine. I due standard sono anche molto diversi per quanto riguarda lo stile, in 802.3 i protocolli vengono descritti come procedure Pascal, mentre in 802.4 essi sono definiti come macchine a stati finiti con azioni descritte in Ada. Per il livello fisico il token bus usa il cavo coassiale larga banda di 75Ω usato per la TV via cavo. Sono previsti sia sistemi con cavo singolo che duale, con o senza terminatore. Si hanno tre possibili metodi di modulazione analogica:

- ♦ Phase continuous frequency shift keying.
- ♦ Phase coherent frequency shift keying.
- ♦ Multilevel duobinary amplitude modulated phase shift keying.

Le velocità possono essere di 1, 5 o 10 Mbps. Inoltre i metodi di modulazione non prevedono solo il modo di rappresentare 0, 1 oppure il canale vuoto, ma anche altri tre simboli usati per il controllo della rete. In generale quindi il livello fisico è totalmente incompatibile con quello 802.3 e molto più complicato.

Il protocollo del sottolivello MAC di token bus

Quando l'anello viene inizializzato, le stazioni vengono inserite in ordine di indirizzo, dal più alto al più basso. Anche il passaggio del token avviene secondo questo ordinamento. Ogni volta che una stazione ottiene il token, essa può trasmettere frame per un certo periodo di tempo; quindi deve passare il token ad altri. Se i frame sono abbastanza corti, ne possono essere spediti diversi. Se una stazione non ha dati da spedire passa immediatamente il token alla successiva.

Il token bus definisce quattro classi di priorità per il traffico, 0, 2, 4 e 6, con 0 che è la priorità più bassa e 6 la più alta. È più semplice pensare ogni stazione come divisa internamente in quattro sottostazioni, ognuna a un livello di priorità. Mentre i dati arrivano al sottolivello MAC dai livelli superiori, viene controllata la loro priorità ed essi vengono direzionati verso una delle quattro sottostazioni. Quindi ogni sottostazione mantiene la propria coda dei frame da trasmettere.

Quando il token arriva alla stazione dal cavo, viene passato internamente alla sottostazione di priorità 6, che potrà iniziare a trasmettere frame se ne possiede. Quando ha finito (o il suo timer è scaduto), il token viene passato internamente alla sottostazione di priorità 4 che potrà trasmettere i suoi frame finché il suo timer non sarà scaduto e non dovrà passare il token internamente alla sottostazione di priorità 2. Questo processo viene ripetuto finché o la stazione di priorità 0 non avrà spedito tutti i suoi frame, o il suo timer non scadrà. In ogni caso a questo punto il token viene trasmesso alla stazione successiva nell'anello.

Senza entrare nei dettagli di come vengono gestiti i vari timer, dovrebbe essere chiaro che impostare i timer in modo esatto può essere utile per garantire che una certa frazione di tempo di possesso di token possa essere allocata per il traffico con priorità 6. Le priorità più basse dovranno accontentarsi del tempo restante. Se le sottostazioni di priorità più alta non hanno bisogno di tutto il tempo allocato per loro, le sottostazioni di priorità più bassa potranno ottenere la porzione inutilizzata in modo che non vada sprecata.

Questo modello di priorità, che garantisce al traffico di priorità 6 una frazione nota di larghezza di banda può essere utilizzato per implementare il traffico in tempo reale. Ad esempio, si supponga che i parametri di una rete di 50 stazioni con velocità di 10 Mbps siano stati fissati per garantire al traffico di priorità 6 $\frac{1}{3}$ della larghezza di banda. Ogni stazione avrà 67 kbps riservati per il traffico di priorità 6. Questa larghezza di banda potrebbe essere utilizzata per sincronizzare robot su una catena di assemblaggio oppure per gestire un canale vocale per stazione, riservandone una piccola frazione per le informazioni di controllo.

Il formato di frame del token bus viene mostrato in figura 4-26. Sfortunatamente, è differente dal formato di frame di 802.3. Il preamble viene usato per sincronizzare l'orologio del ricevente, come in 802.3, ma qui esso può essere anche di un solo byte. I campi *Starting Delimiter* ed *Ending Delimiter* sono utilizzati per individuare le estremità del frame. Entrambi questi campi contengono una codifica analogica di simboli diversi da 0 e 1, quindi non possono presentarsi accidentalmente nei dati utente. Come risultato non è necessario un campo per la lunghezza.



Fig. 4-26 Il formato di frame 802.4.

Il campo *Frame control* viene usato per distinguere i frame di dati da quelli di controllo. Per i frame di dati esso contiene la priorità del frame. Può anche contenere un indicatore per fare in modo che la stazione destinazione mandi un ack per il corretto (o scorretto) ricevimento del frame. Senza tale indicatore la destinazione non avrebbe il permesso di spedire nulla poiché non possiede il token. Questo indicatore fa assomigliare il token bus al modello di ack di Tokoro e Tamaru.

Per quanto riguarda i frame di controllo il campo *Frame control* viene utilizzato per specificarne il tipo. I tipi ammessi includono il passaggio del token e vari frame di mantenimento dell'anello, tra cui il meccanismo per introdurre nuove stazioni nell'anello, per eliminarle e così via. Si noti che il protocollo 802.3 non prevede frame di controllo. Tutto quello che il sottolivello MAC fa in quel caso è di fornire un modo per porre i frame sul cavo, senza preoccuparsi del loro contenuto.

I campi *Destination address* e *Source address* sono uguali a quelli di 802.3 (in effetti i due gruppi si potrebbero "parlare", non andrebbero però molto d'accordo). Come in 802.3, la rete dovrebbe usare indirizzi tutti di 2 byte oppure tutti di 6 byte, non entrambi i tipi sullo stesso cavo. Lo standard iniziale per 802.4 permetteva entrambe le misure. L'indirizzamento individuale o di gruppo e l'assegnamento di indirizzi globali e locali sono identici a quelli di 802.3.

Il campo *Data* può arrivare fino a 8182 byte se sono utilizzati indirizzi di 2 byte e fino a 8174 byte se si usano indirizzi di 6 byte. Questo è più lungo di cinque volte la lunghezza massima di un frame 802.3, che è corto per evitare che una stazione occupi il canale troppo a lungo. Con token bus i timer possono essere utilizzati a questo scopo quando

necessario, ma è utile poter spedire frame lunghi quando non si ha traffico di dati real time. Il campo *Checksum* è usato per rilevare errori di trasmissione. Viene usato lo stesso algoritmo e polinomio di 802.3.

I frame di controllo di token bus sono mostrati in figura 4-27. Essi verranno discussi in seguito. L'unico che abbiamo già incontrato è il frame *token*, usato per passare il token da stazione a stazione. Gli altri hanno a che fare con l'inserimento o la cancellazione di stazioni dall'anello logico.

Campo frame di controllo	Nome	Significato
00000000	Claim_token	Richiede token durante l'inizializzazione dell'anello
00000001	Solicit_successor_1	Permette alla stazione di entrare nell'anello
00000010	Solicit_successor_2	Permette alla stazione di entrare nell'anello
00000011	Who_follows	Recupera la perdita del token
00000100	Resolve_contention	Usato quando più stazioni vogliono entrare
00001000	Token	Passa il token
00001100	Set_successor	Permette alla stazione di lasciare l'anello

Fig. 4-27 I frame di controllo di token bus.

Manutenzione dell'anello logico

Di tanto in tanto vengono attivate nuove stazioni che devono essere aggiunte all'anello. Altre devono essere disattivate ed eliminate dall'anello. Il protocollo di sottolivello MAC fornisce una specifica dettagliata di come questo viene fatto esattamente, mantenendo il limite del caso peggiore sulla rotazione del token. Di seguito illustriamo brevemente i meccanismi utilizzati.

Una volta definito l'anello, ogni interfaccia di stazione mantiene internamente gli indirizzi della stazione che la precede e di quella che la segue. Ogni tanto il possessore di token invia uno dei frame SOLICIT_SUCCESSOR mostrati in figura 4-27, per sollecitare le proposte delle stazioni che si vogliono aggiungere all'anello. Il frame contiene l'indirizzo del mittente e del successore. Le stazioni con indirizzo in quell'intervallo possono proporsi per l'inserimento (per mantenere l'anello ordinato in ordine decrescente di indirizzo di stazione). Se nessuna stazione si proporrà in un intervallo di tempo (2τ , come in 802.3), la finestra di risposta verrà chiusa e il possessore del token continuerà il suo lavoro.

Se si proporrà esattamente una stazione, essa verrà inserita nell'anello e diverrà la successiva di quella che possiede il token. Se due o più stazioni si proporranno per l'inserimento i loro frame colideranno e saranno corrotti come in 802.3. Il possessore del token quindi utilizzerà un algoritmo di arbitraggio, iniziando un broadcast di un frame RESOLVE_CONTENTION. L'algoritmo sarà una variante della regressione binaria, con l'uso di due bit per volta.

Tutte le interfacce delle stazioni mantengono internamente due bit casuali. Questi bit vengono usati per ritardare tutte le proposte di 0, 1, 2, o 3 intervalli di tempo per ridurre

ulteriormente i conflitti. In altre parole, le proposte di due stazioni collidono solo se gli attuali due bit di indirizzo usati sono uguali e contengono gli stessi due bit casuali. Per evitare che le stazioni che devono aspettare tre intervalli per trasmettere si trovino in permanente svantaggio, i bit casuali vengono rigenerati ogni volta che vengono usati oppure periodicamente ogni 50 ms.

La sollecitazione delle nuove stazioni non può interferire con il caso peggiore garantito per la rotazione del token. Ogni stazione ha un timer che viene riinizializzato ogni volta che essa acquisisce il token. Quando il token arriva, il vecchio valore del timer (cioè il precedente tempo di rotazione del token) viene controllato appena prima che il timer sia riinizializzato. Se esso supera un certo valore prestabilito significa che recentemente si è avuto troppo traffico e non potrà essere sollecitata alcuna nuova proposta per l'intervallo successivo. In ogni caso solo una stazione potrà entrare ad ogni sollecitazione, in modo da definire un limite al tempo consumato per il mantenimento dell'anello. Non viene fornita nessuna garanzia sul tempo che una stazione deve attendere prima di unirsi all'anello nel caso il traffico sia abbondante, ma in pratica non attenderà più di pochi secondi. Questa incertezza è un peccato, infatti rende 802.4 meno adatto ai sistemi in tempo reale di quanto proclamino i suoi sostenitori.

Uscire dall'anello è facile. Una stazione, X , con successore S e predecessore P lascia l'anello inviando a P un frame SET_SUCCESSOR indicando che d'ora in avanti il suo successore sarà S e non più X . Quindi X smette di trasmettere.

L'inizializzazione dell'anello è un caso particolare di inserimento di nuove stazioni. Si consideri un sistema inattivo con tutte le stazioni spente. Quando la prima stazione viene attivata nota che non si ha traffico. Quindi spedisce un frame CLAIM_TOKEN. Non individuando alcun contendente per il token, crea un token e definisce un anello che contenga solo lei. Periodicamente essa sollecita proposte per le nuove stazioni che si vogliono aggiungere. Quando vengono attivate nuove stazioni esse risponderanno alle sollecitazioni e si aggiungeranno all'anello usando l'algoritmo di contesa descritto sopra. Prima o poi tutte le stazioni che si vogliono aggiungere all'anello saranno in grado di farlo. Se le prime due stazioni vengono attivate simultaneamente, il protocollo risolverà il problema facendo contendere il token usando l'algoritmo standard modificato di conteggio binario a ritroso e i due bit casuali.

A causa di errori di trasmissione o guasti hardware, possono sorgere problemi con l'anello logico o con il token. Ad esempio, se una stazione tenta di passare il token a una stazione guasta, cosa accade? La soluzione è immediata: dopo avere passato il token una stazione ascolta se il suo successore trasmette un frame o passa il token. Se non fa nulla il token viene passato una seconda volta.

Se fallisce anche la seconda volta essa invia un frame WHO_FOLLOWS (chi segue), specificando l'indirizzo del suo successore. Quando il successore della stazione guasta nota il frame WHO_FOLLOWS indicante il suo predecessore, risponde inviando un frame SET_SUCCESSOR alla stazione con il successore guasto, indicando se stessa come nuovo successore. In questo modo la stazione guasta viene rimossa dall'anello.

Supponiamo ora che una stazione non riesca a passare il token al suo successore e non riesca neppure a individuare il successore del successore, che potrebbe anch'esso essere guasto. Essa adotterà una diversa strategia e invierà un frame SOLICIT_SUCCESSOR_2

per vedere se ne esiste ancora qualcuno "vivo". Ancora una volta viene attivato il protocollo standard di contesa, a cui partecipano tutte le stazioni che vogliono essere nell'anello. L'anello finalmente verrà ristabilito.

Un altro tipo di problema si ha se il possessore del token si guasta. Questa situazione viene risolta usando l'algoritmo di inizializzazione. Ogni stazione avrà un timer che sarà inizializzato quando un frame apparirà sulla rete. Quando questo timer raggiungerà un particolare valore, la stazione trasmetterà un frame CLAIM_TOKEN e l'algoritmo di regressione binaria determinerà a chi spetta il token.

Un ultimo problema riguarda i token multipli. Se una stazione che possiede il token nota una trasmissione da parte di un'altra stazione essa scarta il proprio token. Se ce ne erano due adesso ce ne sarà solo uno. Se ce ne erano di più questo processo sarà ripetuto finché prima o poi tutti i token eccetto uno siano stati scartati. Se, per errore, vengono scartati tutti i token, la mancanza di attività farà in modo che una o più stazioni tentino di generare il token.

4.3.3 Lo Standard IEEE 802.5: token ring

Le reti ad anello sono state utilizzate per molti anni (Pierce, 1972) per reti locali e geografiche. Tra le varie caratteristiche interessanti c'è il fatto che esse non usano un mezzo di broadcast ma una collezione di collegamenti individuali punto-a-punto che formano un cerchio. I collegamenti punto-a-punto implicano una tecnologia ben nota e sperimentata e possono basarsi su doppini intrecciati, cavi coassiali o fibra ottica. La struttura dell'anello è quasi interamente digitale, mentre in 802.3, ad esempio, si hanno diversi componenti analogici indispensabili per la rilevazione di collisioni. Inoltre un anello è equo e ha un limite superiore noto per l'accesso al canale. Per queste ragioni l'IBM ha scelto un anello come modello per le LAN e l'IEEE ha accettato lo standard **token ring** come 802.5 (IEEE, 1985c; Latif et al., 1992).

Una delle caratteristiche principali nella progettazione e nell'analisi di ogni tipo di rete ad anello è la lunghezza fisica di un bit. Se la velocità dei dati sull'anello è R Mbps, viene emesso un bit ogni $1/R \mu\text{s}$. Con una velocità del propagazione di segnale tipica di $200 \text{ m}/\mu\text{s}$, ogni bit occupa $200/R \text{ m}$ sull'anello. Questo significa, ad esempio, che un anello a 1 Mbps, la cui circonferenza sia di 1000 m potrà contenere solo 5 bit alla volta. Le implicazioni del numero di bit sull'anello diverranno chiare in seguito.

Come detto poc'anzi, un anello in realtà consiste in una collezione di interfacce di anello connesse da linee punto a punto. Ogni bit che raggiunge un'interfaccia viene copiato in un buffer di capacità 1 bit e quindi ricopiatato sull'anello di nuovo. Mentre il bit è nel buffer, esso può essere controllato ed eventualmente modificato prima di essere riscritto sull'anello. Questo passo di copiatura introduce un ritardo di un bit per ogni interfaccia. Un anello e la sua interfaccia sono mostrati in figura 4-28.

In un token ring, una sequenza speciale di bit, detta **token**, circola sull'anello mentre tutte le stazioni sono inattive. Quando una stazione vuole trasmettere un frame, si impossessa del token e lo rimuove dall'anello prima di trasmettere. Questa azione viene attuata invertendo un bit nei 3 byte del token, che immediatamente diventano i primi 3 byte di un normale frame di dati. Poiché esiste solo un token, solo una stazione potrà trasmettere

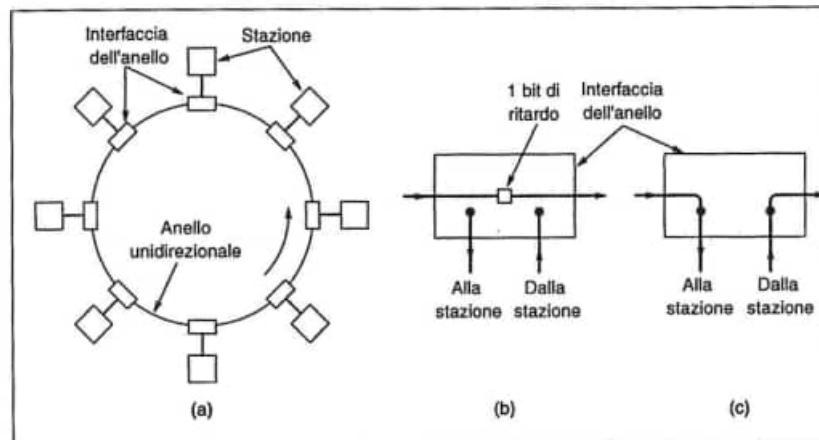


Fig. 4-28 (a) Una rete ad anello. (b) In modalità ascolto. (c) In modalità trasmissione.

in ogni istante, risolvendo quindi il problema di accesso al canale nello stesso modo in cui è risolto nel modello token bus.

Un'implicazione del progetto token ring è che l'anello stesso deve avere un ritardo sufficiente per contenere un token completo circolante quando tutte le stazioni sono inattive. Il ritardo ha due componenti: il ritardo di un bit introdotto da ogni stazione e il ritardo di propagazione del segnale. In quasi tutti gli anelli i progettisti devono assumere che le stazioni possano essere spente in più momenti, specialmente di notte. Se le interfacce sono alimentate dall'anello, la disconnessione di una stazione non ha effetto sull'interfaccia, ma se le interfacce vengono alimentate esternamente esse devono essere progettate in modo da connettere l'input con l'output ogni volta che vengono spente, eliminando il ritardo di un bit. Il punto è che su un anello corto potrebbe dover essere inserito un ritardo "artificiale" nell'anello di notte per assicurare che un token possa esservi contenuto.

Le interfacce dell'anello hanno due modalità operative: ascolto e trasmissione. In modalità di ascolto, i bit di input vengono direttamente copiati in output, con un ritardo di un tempo di bit, come si vede in figura 4-28 (b). In modalità di trasmissione, che si può avere solo dopo che si è ottenuto il token, l'interfaccia interrompe la connessione tra input e output immettendo i suoi dati sull'anello. Per essere in grado di cambiare modalità tra ascolto e trasmissione in un tempo di bit, l'interfaccia ha bisogno di memorizzare in un buffer privato uno o più pacchetti invece di prelevarli dalla stazione con un preavviso così breve. Quando i bit propagati sull'anello tornano indietro, vengono rimossi dall'anello dal mittente. La stazione mittente può o memorizzarli, per confrontarli con gli originali e verificare l'affidabilità dell'anello, oppure scartarli. Poiché l'intero frame non apparirà mai sull'anello interamente, l'architettura dell'anello non pone limiti alla dimensione del frame. Dopo che una stazione ha finito di trasmettere l'ultimo bit del suo ultimo frame, dovrà rigenerare il token. Quando l'ultimo bit del frame ha percorso tutto l'anello ed è tornato indietro, dovrà essere rimosso e l'interfaccia ritornerà immediatamente in modalità di

ascolto per evitare di rimuovere nuovamente il token se nessun'altra stazione lo ha rimosso.

La gestione dei riscontri in token ring è piuttosto semplice. Il formato del frame deve includere solo un campo di un bit per l'ack, inizialmente zero. Quando la stazione destinazione ha ricevuto il frame, inverte il bit. Ovviamente se l'ack significa che è stata verificata la checksum, il bit deve seguire la checksum e l'interfaccia dell'anello deve essere in grado di verificare la checksum non appena l'ultimo bit sarà arrivato. Quando viene fatto broadcast di un frame a più stazioni deve essere usato un meccanismo di ack più complicato (ammettendo che se ne utilizzi uno).

Quando il traffico è scarso il token passerà la maggior parte del suo tempo a circolare sull'anello. Occasionalmente una stazione lo preleverà, trasmetterà un frame e quindi riemetterà un nuovo token. Diversamente, quando il traffico è pesante e si hanno code su ogni stazione, non appena una stazione termina la trasmissione e rigenera il token, la stazione successiva nell'anello vedrà e preleverà a sua volta il token. In questo modo il permesso di trasmissione circola correttamente lungo l'anello seguendo il modello round-robin. L'efficienza della rete può aggirarsi intorno al 100% in condizioni di carico pesante. Passiamo ora a considerare in dettaglio, dopo token ring, lo standard 802.5. A livello fisico 802.5 richiede doppini intrecciati schermati da 1 fino a 4 Mbps sebbene IBM abbia in seguito introdotto una versione a 16 Mbps. I segnali vengono codificati secondo la codifica Manchester differenziale (si veda la figura 4-20 (c)), in cui l'alto e il basso indicano segnali positivi e negativi di valore assoluto da 3,0 a 4,5 V. In genere la codifica Manchester differenziale utilizza alto-basso o basso-alto per ogni bit, ma 802.5 usa anche alto-alto e basso-basso in alcuni byte di controllo (ad es. per indicare l'inizio e la fine di un frame). Questi segnali si susseguono sempre in coppie consecutive in modo da non introdurre una componente di corrente continua nel voltaggio dell'anello.

Un problema con le reti ad anello è che se il cavo si interrompe in qualche punto tutto l'anello risulta guasto. Questo problema può essere aggirato elegantemente utilizzando un concentratore di fili, come in figura 4-29. Anche se logicamente si ha un anello, fisicamente ogni stazione è connessa al centro da un cavo contenente (almeno) due doppini intrecciati, uno per i dati verso la stazione e uno per i dati provenienti da questa. All'interno del concentratore ci sono dei relè di bypass alimentati dalle stazioni. Se l'anello si interrompe oppure una stazione si guasta, la perdita di corrente farà scattare il relè e aggirerà la stazione guasta. I relè possono anche essere gestiti a software per far sì che programmi di diagnostica possano rimuovere le stazioni una alla volta per individuare stazioni guaste e segmenti dell'anello difettosi. L'anello potrà quindi continuare le sue operazioni aggirando il segmento difettoso. Sebbene lo standard 802.5 non richieda formalmente questo tipo di anello, spesso detto **anello stellato** (Saltzer et al., 1983), la maggior parte delle LAN 802.5 usa concentratori per migliorare l'affidabilità e la capacità di manutenzione.

Quando una rete è composta da molti gruppi di stazioni distanti, può essere utilizzata una topologia con più concentratori. Si immagini che il cavo che raggiunge una delle stazioni in figura 4-29 sia rimpiazzato da un cavo collegato a un concentratore distante. Sebbene logicamente le stazioni siano tutte sullo stesso anello, i requisiti di cablaggio sono note-

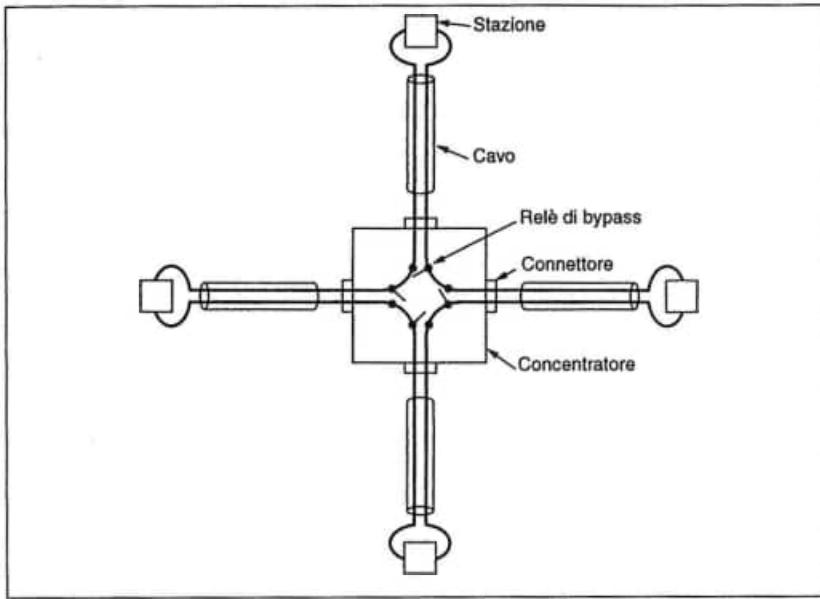


Fig. 4-29 Quattro stazioni connesse attraverso un concentratore.

volmente ridotti. Un anello 802.5 che utilizzi un concentratore avrà una topologia simile a una rete 802.3 10Base-T basata su hub, ma i formati e i protocolli saranno differenti.

Il protocollo del sottolivello MAC di token ring

Le operazioni fondamentali del protocollo MAC sono molto semplici. Quando non c'è traffico sull'anello, un token di 3 byte gira senza sosta attendendo che una stazione lo acquisisca invertendo un bit particolare da 0 a 1, convertendolo in una sequenza di inizio frame. La stazione emette quindi la normale parte dati del frame, come in figura 4-30. In condizioni normali, il primo bit del frame circolerà sull'anello e ritornerà al mittente prima che l'intero frame sia stato trasmesso. Solo un anello molto lungo potrebbe essere in grado di contenere interamente il più piccolo frame. Di conseguenza la stazione trasmettente deve ripulire l'anello mentre continua a trasmettere. Come mostrato in figura 4-28 (c), questo significa che i bit che hanno completato il giro sull'anello ritorneranno al mittente e verranno rimossi.

Una stazione può possedere il token per il **tempo di possesso di token**, che è di 10 ms a meno che non sia stato definito diversamente nell'installazione. Se rimane abbastanza tempo dopo la trasmissione del primo frame per inviare nuovi frame, anch'essi saranno trasmessi. Dopo che tutti i frame in coda sono stati trasmessi oppure che la trasmissione di un altro frame abbia superato il tempo di possesso di token, la stazione rigenera il token di 3 byte e lo pone sull'anello.

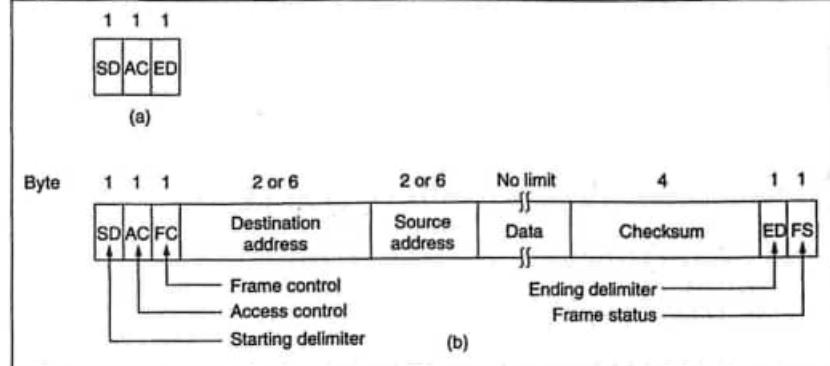


Fig. 4-30 (a) Formato del token. (b) Formato del frame di dati.

I campi *Starting delimiter* ed *Ending delimiter* della figura 4-30 (b) indicano l'inizio e la fine del frame. Ognuno contiene sequenze non valide differenziali Manchester (HH o LL) per distinguere dai byte di dati. Il byte di *Access control* contiene il bit *Sel token*, il bit *Monitor*, i bit *Priority* e *Reservation* (descritti di seguito). Il byte *Frame control* distingue i frame di dati dai vari possibili frame di controllo.

Quindi si hanno i campi *Destination address* e *Source address* che sono uguali a quelli dell'802.3 e 802.4. Questi sono seguiti dai dati, che saranno lunghi quanto necessario, a patto che il frame possa essere trasmesso in un tempo di possesso di frame. Il campo *Checksum*, come l'indirizzo destinazione e sorgente è identico a quello in 802.3 e 802.4. Un byte interessante non presente negli altri due protocolli è il byte *Frame status*. Contiene i bit *A* e *C*. Quando un frame arriva all'interfaccia di una stazione con l'indirizzo destinazione, l'interfaccia attiva il bit *A* mentre questo passa. Se l'interfaccia copia il frame nella stazione questa attiva anche il bit *C*. Una stazione può fallire nel copiare un frame, per ragioni di mancanza di spazio nel buffer oppure per altri motivi.

Quando la stazione mittente preleva il frame dall'anello, essa esamina i bit *A* e *C*. Sono possibili tre combinazioni:

1. *A* = 0 e *C* = 0: destinazione non presente o non attiva.
2. *A* = 1 e *C* = 0: destinazione presente ma frame non accettato.
3. *A* = 1 e *C* = 1: destinazione presente e frame copiato.

Questa tecnica fornisce un riscontro automatico per ogni frame. Se un frame viene scartato ma la stazione è presente, il mittente penserà di riprovare in un tempo breve. I bit *A* e *C* sono presenti due volte nel *Frame status* per aumentare l'affidabilità in quanto essi non sono coperti dalla checksum.

Ending delimiter contiene un bit *E* che è attivo se qualche interfaccia rileva un errore (ad es., una sequenza non Manchester dove non è permessa). Esso contiene anche un bit che

può essere usato per indicare l'ultimo frame in una sequenza logica, una specie di bit di fine file.

Il protocollo 802.5 adotta un tecnica elaborata per gestire frame con priorità multipla. Il token di 3 byte contiene un campo nel byte di mezzo che segnala la priorità del token. Quando una stazione vuole trasmettere un frame di priorità n , deve aspettare finché non riesce a catturare un token con priorità minore o uguale a n . Inoltre, dopo aver trasmesso un frame di dati, la stazione può tentare di riservare il token successivo scrivendo la priorità del token che vuole trasmettere nel bit *Reservation* del frame. Tuttavia, se è già stata riservata su quel bit una priorità più alta la stazione non può fare la prenotazione. Quando il frame corrente termina il token successivo viene generato a una priorità uguale a quella prenotata.

Una breve riflessione porterà a capire che questo meccanismo è simile a una ruota dentata, che aumenta la priorità sempre più. Per eliminare questo problema il protocollo contiene alcune regole complesse. Il concetto base è che la stazione che aumenta la priorità dovrà abbassare la priorità dopo aver terminato.

Questo modello di priorità è sostanzialmente differente dal metodo token bus, dove ogni stazione ottiene sempre la sua larghezza di banda, indipendentemente da quello che fanno le altre stazioni. Nel modello token ring, una stazione con solo frame di priorità bassa può attendere all'infinito l'apparizione di un token di priorità bassa. Ovviamente i due comitati avevano diversi punti di vista nel trattare il compromesso tra miglior servizio al traffico di priorità alta ed equità per tutte le stazioni.

Manutenzione dell'anello

Il protocollo token bus diventa particolarmente lungo a causa della gestione dell'anello che avviene in modo completamente decentralizzato. Il protocollo token ring gestisce il mantenimento in maniera differente. Ogni token ring ha una **stazione monitor** che sorveglia l'anello. Se il monitor si guasta, un protocollo di contesa assicura che un'altra stazione venga immediatamente eletta come monitor (ogni stazione ha le capacità di diventare un monitor). Quando il monitor funziona correttamente, esso solo è responsabile del fatto che l'anello funzioni correttamente.

Quando l'anello viene attivato, oppure qualche stazione si accorge che non c'è monitor, essa può trasmettere un frame di controllo CLAIM TOKEN. Se il frame circola per tutto l'anello prima che altri frame CLAIM TOKEN siano spediti, il mittente diventa il nuovo monitor (ogni stazione ha le capacità di monitor incorporate). I frame di controllo di token ring sono mostrati in figura 4-31.

Queste sono alcune delle responsabilità del monitor: controllare che il token non venga perso, intraprendere particolari azioni quando l'anello si spezza, ripulire l'anello quando appaiono frame corrotti e stare attento ai frame "orfani". Si ottiene un frame orfano quando una stazione trasmette un frame corto interamente su un anello lungo e quindi si guasta o viene disattivata prima che il frame possa essere ritirato. Se non si provvede in qualche modo, il frame circolerà sull'anello per sempre.

Per individuare i token persi, il monitor possiede un timer che è inizializzato al più lungo possibile intervallo senza token: una stazione che trasmetta per tutto il tempo di possesso di token. Se il timer scade il monitor ripulisce l'anello e genera un nuovo token.

Campo di controllo	Nome	Significato
00000000	Duplicate address test	Controllare se due stazioni hanno lo stesso indirizzo
00000010	Beacon	Usato per inserire interruzioni nell'anello
00000011	Claim token	Tentativo di diventare monitor
00000100	Purge	Reinizializzazione dell'anello
00000101	Active monitor present	Stabilito periodicamente dal monitor
00000110	Standby monitor present	Annuncia la presenza di potenziali monitor

Fig. 4-31 I frame di controllo di token ring.

Quando compare un frame corrotto, il monitor è in grado di rilevarlo attraverso il suo formato non valido o la sua checksum, quindi apre l'anello e rigenera un nuovo token dopo avere intrapreso azione di pulizia. Infine, il monitor rileva i frame orfani attivando il bit *Monitor* nel byte *Access control* ogni volta che questo lo attraversa. Se un frame entrante ha il suo bit attivato significa che qualcosa è andato storto, poiché lo stesso frame ha attraversato il monitor due volte senza essere ritirato: il monitor quindi lo ritira.

Un'ultima funzione riguarda la lunghezza dell'anello. Il token è di 24 bit, il che significa che l'anello dovrà essere abbastanza grande da contenere 24 bit. Se il ritardo di un bit nelle stazioni più la lunghezza del cavo danno un numero minore di 24 bit, il monitor inserisce bit di ritardo extra in modo che il token possa circolare.

Una funzione di mantenimento che non può essere gestita dal monitor è quella di individuare le interruzioni sull'anello. Quando una stazione nota che una delle sue vicine sembra morta, essa trasmette un frame BEACON (segnaletico) dando l'indirizzo della stazione presumibilmente morta. Quando questo frame viene propagato raggiungendo il punto più lontano ancora raggiungibile, sarà possibile determinare quante stazioni sono guaste ed eliminarle dall'anello usando i relè di bypass nel concentratore, il tutto senza alcun intervento umano.

È utile paragonare gli approcci utilizzati per controllare token bus e token ring. Il comitato 802.4 era sconvolto all'idea di imporre una componente centralizzata che potesse fallire in qualche modo inatteso e mettere in ginocchio l'intero sistema. Per questo progettarono un sistema in cui il possessore di token avrebbe avuto poteri speciali (ad es. sollecitare proposte per aggiungersi all'anello), ma nessuna stazione sarebbe stata per il resto diversa dalle altre (ad es. per quanto riguarda la responsabilità amministrativa correntemente assegnata per la manutenzione).

Il comitato 802.5, d'altra parte, notò che un monitor centralizzato permetteva di gestire più semplicemente token persi, pacchetti orfani e così via. Inoltre, in un sistema normale, le stazioni si guastano raramente, quindi dover gestire una contesa ogni tanto per il monitor non risultava un sovraccarico eccessivo. Il prezzo da pagare era che se in qualche modo il monitor "impazziva" ma continuava a inviare periodicamente un frame di con-

trollo MONITOR ATTIVO PRESENTE, nessuna stazione lo avrebbe sfidato. I monitor non possono essere messi in dubbio.

Questa diversità di approccio deriva dalle diverse aree di applicazione che i due comitati avevano in mente. Il comitato 802.4 pensava a fabbriche con grandi quantità di metallo in movimento sotto controllo di computer. I guasti di rete potevano essere molto dannosi e dovevano essere prevenuti a ogni costo. Il comitato 802.5 era interessato all'automazione d'ufficio, dove guasti saltuari potevano essere tollerati a patto di avere un sistema più semplice. Che 802.4 sia in effetti più affidabile è comunque una questione piuttosto discussa.

4.3.4 Confronti tra 802.3, 802.4 e 802.5

Con tre tipi incompatibili di LAN disponibili, ognuna con diverse proprietà, molte organizzazioni debbono affrontare la questione: "Quale è meglio installare?" In questo paragrafo analizzeremo le tre LAN standard 802, sottolineando i loro punti di forza e le loro debolezze, mettendole a confronto.

Tanto per iniziare, vale la pena di notare che le tre LAN standard usano più o meno le stesse tecnologie e ottengono più o meno le stesse prestazioni. Mentre scienziati e ingegneri possono discutere per ore i meriti del cavo coassiale rispetto ai doppini intrecciati, al personale del marketing o della contabilità probabilmente questa differenza non importa.

Cominciamo parlando dei vantaggi di 802.3. È attualmente il tipo più utilizzato, con una notevole quantità di installazioni e un'ampia esperienza operativa. Il protocollo è semplice. Le stazioni possono essere installate rapidamente, senza disattivare la rete. Viene utilizzato un cavo passivo e i modem non sono necessari. Inoltre, i ritardi a traffico ridotto sono praticamente nulli (le stazioni non devono aspettare il token, possono trasmettere all'istante).

D'altra parte 802.3 ha una componente analogica fondamentale. Ogni stazione deve essere in grado di distinguere anche i segnali delle stazioni più deboli, perfino mentre essa stessa sta trasmettendo e tutti i circuiti per la rilevazione di collisioni contenuti nel transceiver sono analogici. A causa della possibilità di ottenere frame corrotti da collisioni, il formato minimo di un frame valido è di 64 byte, che rappresenta un appesantimento notevole quando i dati sono composti da un solo carattere inviato da un terminale.

Inoltre, 802.3 è non deterministico e quindi spesso inadatto a trasmissioni in tempo reale (sebbene traffico in tempo reale sia possibile simulando via software token ring (Venkatramani, Chiueh, 1995)). Non prevede nemmeno priorità. La lunghezza del cavo è limitata a 2,5 km (a 10 Mbps) poiché la lunghezza del percorso completo sul cavo determina il tempo di slot e quindi le prestazioni. Quando la velocità aumenta, l'efficienza diminuisce poiché i tempi di trasmissione del frame diminuiscono ma l'intervallo di contesa no (lo spessore dello slot è 2 (indipendentemente dalla velocità dei dati). In alternativa si deve imporre un cavo più corto. In più, a carico elevato, la presenza di collisioni diventa un grave problema e può seriamente condizionare la produttività.

Consideriamo ora 802.4, token bus. Usa la tecnologia affidabile dei cavi televisivi, che è disponibile in commercio presso diversi punti vendita. È più deterministico di 802.3, sebbene perdite ripetute del token in momenti critici possano introdurre più incertezza

di quanto i suoi sostenitori vogliono ammettere. Può gestire frame di lunghezza minima.

Token bus prevede anche priorità e può essere configurato per garantire una frazione della larghezza di banda al traffico di priorità alta, come la voce digitalizzata. Ha anche un'eccellente produttività ed efficienza in condizioni di carico elevato, risultando in pratica un sistema TDM. Infine, il cavo a larga banda può prevedere più canali, non solo per i dati ma anche per voce e televisione.

D'altra parte, i sistemi a larga banda utilizzano molti meccanismi analogici e includono modemi e amplificatori a larga banda. Il protocollo è estremamente complesso e riporta ritardi ingenti a basso carico (le stazioni devono sempre attendere il token, anche in un sistema che altrimenti resta inattivo). È infine poco indicato per implementazioni in fibra ottica e non ci sono state massicce installazioni.

Consideriamo adesso token ring. Esso usa connessioni punto-a-punto, il che significa che la tecnologia è semplice e può essere completamente digitale. Gli anelli possono essere costruiti usando idealmente qualsiasi mezzo di trasmissione, dai "piccioni viaggiatori" alla fibra ottica. Il doppino standard è economico e semplice da installare. L'uso di concentratore rende token ring l'unica LAN che possa rilevare ed eliminare automaticamente interruzioni sul cavo.

Come token bus, è possibile avere priorità, sebbene la tecnica non sia così equa. Sempre come token bus, si possono avere frame corti ma diversamente da token bus è possibile anche averne di arbitrariamente lunghi, unico limite è il tempo di possesso del token. Infine, la produttività e l'efficienza in condizioni di carico elevato sono eccellenti, come per token bus e a differenza di 802.3.

Il maggior difetto è la presenza di una funzione di monitor centralizzata che introduce una componente critica. Anche se un monitor guasto può essere rimpiazzato, uno malfunzionante può causare seri problemi. Inoltre, come tutte le tecniche di passaggio di token, si hanno sempre ritardi a basso carico poiché il mittente deve attendere il token. Vale anche la pena notare che ci sono stati diversi studi delle tre LAN. La conclusione principale che possiamo trarre è che non si possono trarre conclusioni. Si possono sempre trovare insiemi di parametri che fanno sembrare una LAN migliore delle altre. In molte circostanze tutte e tre si comportano bene, per cui fattori diversi dalle prestazioni dovrebbero essere considerati decisivi per fare una scelta.

4.3.5 Lo standard IEEE 802.6: Distributed Queue Dual Bus

Nessuna delle LAN 802 che abbiamo studiato fino ad ora è adatta come MAN. Le limitazioni sulla lunghezza del cavo e i problemi di prestazioni quando si hanno migliaia di stazioni connesse, limita le loro dimensioni ad aree delle dimensioni di un campus. Per reti che coprono un intera città l'IEEE ha definito una MAN, detta **DQDB (Distributed Queue Dual Bus – bus duale a coda distribuita)**, come standard 802.6. In questo paragrafo esamineremo il suo funzionamento. Per ulteriori informazioni si vedano Kessler, Train (1992). In Sadiku, Arvind (1994) viene data una bibliografia di 171 articoli riguardanti DQDB.

La struttura di base di 802.6 viene illustrata in figura 1-4. Due bus paralleli e unidirezionali percorrono tutta la città, le stazioni sono attaccate a entrambi i bus in parallelo. Ogni

bus ha una estremità di testa che genera un continuo flusso di celle di 53 byte. Ogni cella viaggia scendendo dalla estremità di testa, quando raggiunge l'altra estremità esce dal bus.

Ogni cella trasporta un campo contenente il carico di 44 byte, che la rendono compatibile con alcune modalità AAL. Ogni cella contiene anche due bit di protocollo, *Busy*, per indicare che una cella è occupata, e *Request*, che può essere attivato quando una stazione vuole fare una richiesta.

Per trasmettere una cella una stazione deve sapere se la destinazione rimane alla sua destra o alla sua sinistra. Se la destinazione è a destra, il mittente utilizza il bus A, altrimenti il bus B. I dati vengono inseriti su entrambi i bus usando un circuito OR, quindi i guasti di una stazione non disattivano l'intera rete.

Diversamente dagli altri protocolli 802 per le LAN, questo non è "avidio". In tutti gli altri, se una stazione riesce a ottenere il permesso di spedire, essa spedirà. Qui, le stazioni si accodano ordinatamente quando sono pronte e trasmettono in ordine FIFO. La parte interessante del protocollo è come esso ottenga l'ordinamento FIFO senza usare una coda centralizzata.

La regola base è che le stazioni sono "educate": esse lasciano passare sempre le stazioni più in basso. Questa educazione è necessaria per prevenire la situazione in cui la stazione più vicina alla testa del bus catturi tutte le celle libere che arrivano e le riempia, lasciando in attesa infinita tutte le altre stazioni più in basso. Per semplicità esamineremo solo la trasmissione sul bus A, ma lo stesso discorso vale per il bus B.

Per simulare una coda FIFO, ogni stazione mantiene due contatori, *RC* e *CD*. *RC* (*Request Counter* – contatore delle richieste) conta il numero di richieste pendenti di stazioni più in basso finché la stazione stessa non ha finito di spedire i suoi frame. A questo punto *RC* viene copiato in *CD* e viene azzerato per iniziare a contare le richieste inoltrate dopo che la stazione è divenuta pronta. Ad esempio, se *CD* = 3 ed *RC* = 2 per la stazione *k*, le successive tre celle libere vengono riservate per le stazioni più in basso. Per semplicità assumiamo che una stazione sia pronta a trasmettere una sola cella alla volta.

Per inviare una cella una stazione deve prima fare richiesta, attivando il bit *Request* in qualche cella del bus in direzione opposta (cioè nel bus B per la trasmissione che avrà luogo sul bus A). Quando questa cella è stata propagata sul bus opposto, ogni stazione sul percorso noterà la cella e incrementerà il suo *RC*. Per illustrare questo concetto useremo un esempio. Inizialmente tutti i contatori *RC* saranno 0 e nessuna cella sarà in coda come mostrato in figura 4-32 (a). Quindi la stazione D inoltra una richiesta: questo fa in modo che le stazioni C, B e A incrementino i loro contatori *RC*, come in figura 4-32 (b).

Dopodiché B inoltra una richiesta copiando *RC* su *CD*, ottenendo una situazione come quella in figura 4-32 (c).

A questo punto la testa del bus A genera una cella libera. Quando questa passa da B, la stazione nota che il suo *CD* è maggiore di 0 e non utilizza la cella libera. (Quando una stazione ha una cella in coda, *CD* rappresenta la sua posizione nella coda, con 0 che indica la testa della coda). Essa invece decrementa *CD*. Quando la cella ancora libera raggiunge B essa nota che il suo *CD* è 0, cioè che nessuno è prima di lei nella coda, quindi unisce in OR i suoi dati nella cella e attiva il bit *Busy*. Dopo la trasmissione si ha una situazione come quella in figura 4-32 (d).

Quando la successiva cella libera viene generata, la stazione D nota che è in testa alla coda e si impossessa della cella (attivando il bit a 1) come in figura 4-32 (e). In questo modo le stazioni si accodano in attesa del turno, senza una gestione centralizzata della coda.

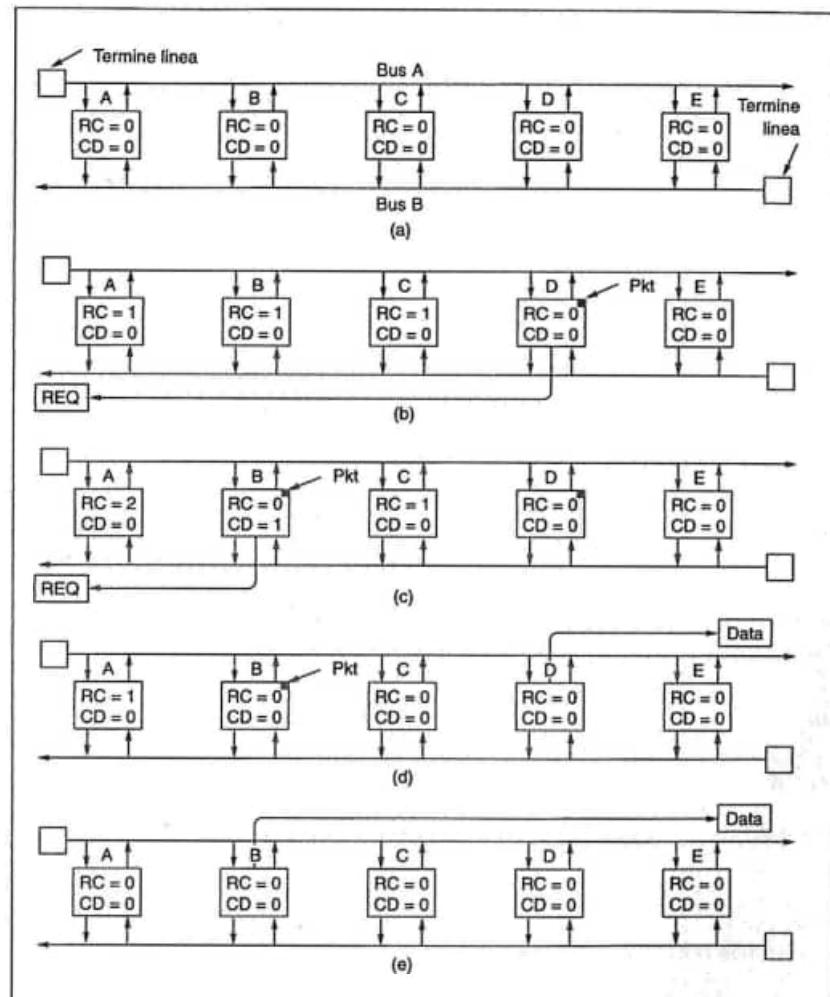


Fig. 4-32 (a) Inizialmente la MAN è inattiva. (b) Dopo che D ha fatto una richiesta. (c) Dopo che B ha fatto una richiesta. (d) Dopo che D ha trasmesso. (e) Dopo che B ha trasmesso.

I sistemi DQDB vengono ora installati da molte società su di intere città. Tipicamente si estendono fino a 160 km alla velocità di 44,736 Mbps (T3).

4.3.6 Lo standard 802.2: Logical Link Control

È ora giunto il momento di fare un passo indietro e paragonare quello che si è imparato in questo capitolo con ciò che si era studiato nel precedente. Nel capitolo 3 avevamo visto come due macchine potessero comunicare in modo affidabile su di una linea inaffidabile usando diversi protocolli data link. Questi protocolli fornivano controllo degli errori (usando gli ack) e controllo di flusso (usando sliding window).

Al contrario, in questo capitolo non si è detto nulla riguardo alla comunicazione affidabile. Tutto quello che le LAN e MAN 802 offrono è un servizio di datagrammi di tipo best-effort. Qualche volta questo servizio è adeguato. Ad esempio, per trasportare pacchetti IP non sono richieste e nemmeno attese garanzie. Un pacchetto IP può essere inserito in un campo 802 e spedito per la sua strada. Se va perso non importa.

Tuttavia, ci sono anche sistemi in cui è auspicabile un protocollo data link con controllo di flusso e di errore. IEEE ne ha definito uno che può girare sopra tutti i protocolli LAN e MAN 802. Inoltre, questo protocollo, detto LLC (Logical Link Control – controllo di collegamento logico), nasconde le differenze tra i vari tipi di reti 802 fornendo un unico formato e un'unica interfaccia al livello rete. Tali formato, interfaccia e protocollo sono strettamente basati su OSL. LLC costituisce la metà più alta del livello data link, avendo il livello MAC sotto di sé, come mostrato in figura 4-33.

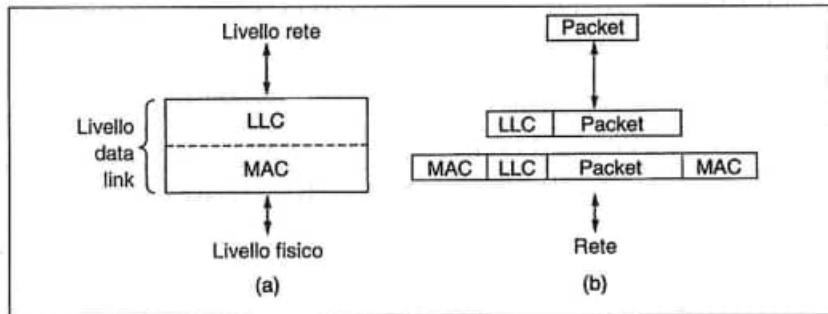


Fig. 4-33 (a) Posizione di LLC. (b) Formati del protocollo.

I tipici utilizzi di LLC sono i seguenti. Il livello rete della macchina sorgente passa un pacchetto a LLC usando le primitive di accesso di LLC. Il sottolivello LLC aggiunge l'intestazione LLC, contenente un numero progressivo e un numero di ack. La struttura risultante viene inserita nel campo di carico di un frame 802.x e trasmesso. Sulla macchina ricevente avviene il processo inverso.

LLC fornisce tre opzioni di servizio: servizio di datagrammi inaffidabile, servizio di datagrammi con ack e servizio orientato alla connessione affidabile. L'intestazione LLC si basa sul più vecchio protocollo HDLC. Si usano parecchi formati diversi per i dati e i

controlli. Per i datagrammi con ack e servizi orientati alla connessione, i frame di dati contengono un indirizzo sorgente, un indirizzo destinazione, un numero progressivo, un numero di ack e alcuni altri bit. Per il servizio di datagrammi non affidabile vengono omessi i numeri progressivi e i numeri di ack.

4.4 I bridge

Molte organizzazioni hanno più di una LAN e desiderano che siano collegate tra loro. Le LAN possono essere connesse mediante dispositivi detti **bridge** (ponti), che operano a livello data link. Questo significa che i bridge non esaminano l'intestazione del livello rete e possono quindi ricopiare pacchetti IP, IPX od OSI con uguale facilità. Al contrario, un router (intradattatore) IP puro, IPX o OSI può gestire solo pacchetti del tipo corrispondente.

Nei paragrafi seguenti considereremo il progetto dei bridge, specialmente per la connessione di LAN 802.3, 802.4 e 802.5. Per una trattazione completa dei bridge e argomenti correlati si veda Perlman, 1992. Prima di inoltrarci nella tecnologia dei bridge vale la pena considerare alcune comuni situazioni in cui sono utilizzati i bridge.

Indichiamo sei buone ragioni per cui una singola organizzazione può avere più LAN. Primo, molte università e dipartimenti hanno le loro LAN, fondamentalmente per connettere i propri computer, workstation e server. Poiché gli scopi dei vari dipartimenti differiscono, questi sceglieranno LAN differenti, indipendentemente l'uno dall'altro. Prima o poi si avrà bisogno di interagire ed è a questo punto che sono necessari i bridge. In questo esempio, diverse LAN vengono costruite distintamente a causa dell'autonomia dei loro proprietari.

Secondo, un'organizzazione può essere geograficamente estesa su più edifici separati da distanze considerevoli. Potrebbe essere più economico avere LAN separate per ogni edificio e connesse tramite bridge e collegamenti infrarossi su un singolo cavo coassiale che corra sull'intera zona.

Terzo, potrebbe essere necessario suddividere quella che logicamente è una singola LAN in diverse LAN per problemi di carico. In molte università, ad esempio, migliaia di workstation sono disponibili agli studenti e al personale. I file vengono generalmente tenuti su macchine file server e scaricati sulle macchine degli utenti quando richiesti. La dimensione enorme di questo sistema impedisce di collegare tutte le workstation a una singola LAN, la larghezza di banda necessaria è troppo elevata. Vengono connesse invece più LAN mediante bridge, come in figura 4-34. Ogni LAN contiene un gruppo di elaboratori con il proprio file server, in modo che la maggior parte del traffico sia ristretto a una singola LAN e non aggiunga carico sulla dorsale.

Quarto, in alcune situazioni un'unica LAN sarebbe adatta in termini di carico ma la distanza fisica tra le macchine più lontane è troppo elevata (cioè più di 2,5 km per 802.3). Anche se stendere il cavo è semplice, la rete può non funzionare per gli eccessivi ritardi sul percorso totale. L'unica soluzione è quella di frazionare la LAN e installare bridge tra i segmenti. Usando i bridge può essere aumentata la distanza fisica totale coperta.

Quinto, una questione di affidabilità: su di una singola LAN, un nodo difettoso che

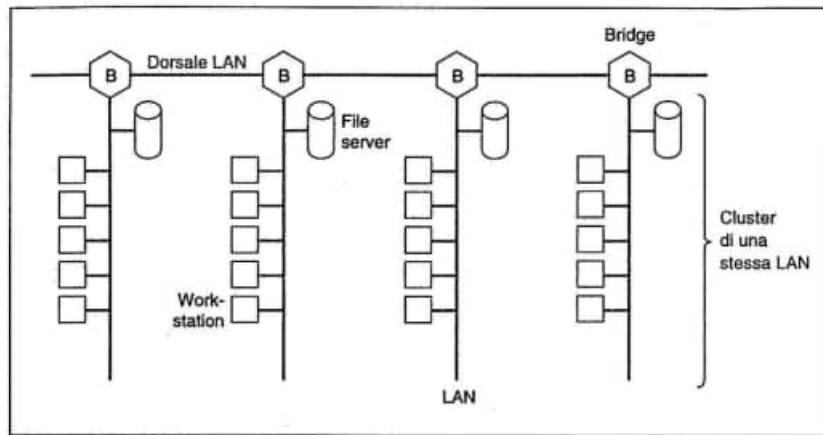


Fig. 4-34 Più LAN connesse da una dorsale per gestire un carico più elevato di quello gestibile da una singola LAN.

continui a emettere un flusso continuo di "spazzatura" paralizzerà la rete. I bridge possono essere inseriti come punti critici, come le porte tagliafuoco in un edificio, per impedire che un unico nodo impazzito metta in ginocchio l'intera rete. Diversamente dai ripetitori, che sanno solo copiare quello che gli viene passato, un bridge può essere programmato per discriminare cosa lasciare passare e cosa no.

Sesto e ultimo, i bridge possono contribuire alla sicurezza dell'organizzazione. La maggior parte delle interfacce LAN hanno **modalità promiscue** in cui *tutti* i frame vengono consegnati al computer, non solo quelli indirizzati ad esso. Spie e ficanasano adorano questa caratteristica: inserendo bridge in vari punti e prestando attenzione a non lasciar passare traffico importante è possibile isolare parti della rete in modo che il loro traffico non possa uscire e cadere in cattive mani.

Avendo visto perché i bridge sono utili, passiamo ora al loro funzionamento. La figura 4-35 illustra le operazioni di un semplice bridge a due porte. L'host A deve spedire un pacchetto. Il pacchetto scende nel livello LLC e gli viene agganciato un header LLC. Quindi viene passato al sottolivello MAC e gli viene appeso un header 802.3 (anche un trailer che per semplicità non è raffigurato). L'unità viene poi spedita sul cavo e prima o poi raggiungerà il sottolivello MAC nel bridge dove verrà eliminato l'header 802.3. Il pacchetto (con l'header LLC) è quindi passato al sottolivello LLC nel bridge. In questo esempio il pacchetto è destinato a una sottorete 802.4 connessa al bridge, quindi viene fatto passare attraverso il lato 802.4 del bridge e ritrasmesso sul cavo. Si noti che un bridge che connette *k* diverse LAN avrà *k* diversi sottolivelli MAC e *k* diversi livelli fisico, uno per ogni tipo.

4.4.1 Bridge da 802.x a 802.y

Si potrebbe pensare che un bridge tra due LAN 802 sia molto semplice. Invece ciò non

è del tutto vero. Nel resto di questo paragrafo indicheremo alcune delle difficoltà che si incontrano quando si tenta di costruire un bridge tra le varie LAN.

Ognuna delle nove combinazioni da LAN 802.x a LAN 802.y ha il proprio gruppo di problemi. Tuttavia, prima di considerarli uno per uno, ragioniamo su alcuni problemi generali comuni a tutti i bridge. Per iniziare, ogni LAN usa un diverso formato di **frame** (si veda la figura 4-36). Non ci sono ragioni tecniche valide alla base di questa incompatibilità, è solo che nessuna delle società che supportavano i tre standard (Xerox, GM e IBM) volle modificare il suo. Di conseguenza, ogni copia tra le diverse LAN richiede riformattazione, e quindi tempo di CPU, un nuovo calcolo di checksum e introduce la possibilità di errori non rilevati a causa dei bit sporchi nella memoria del bridge. Niente di tutto ciò sarebbe necessario se i tre comitati avessero concordato di avere un unico formato.

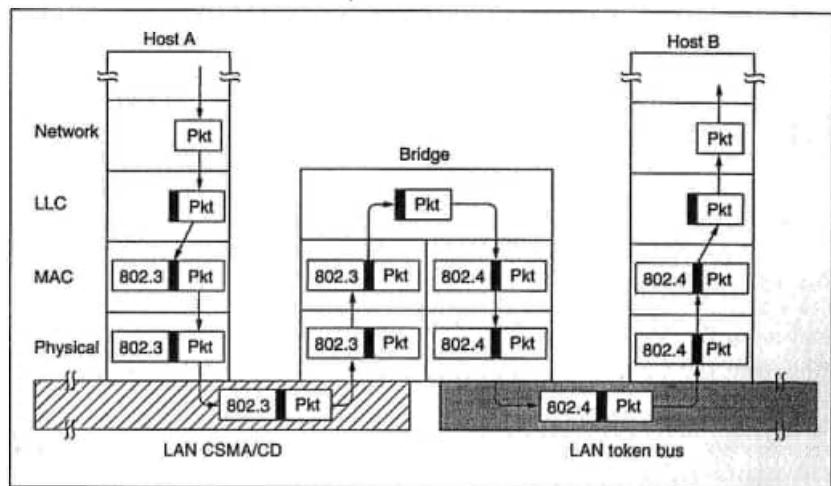


Fig. 4-35 Le operazioni di un bridge tra LAN 802.3 e 802.4.

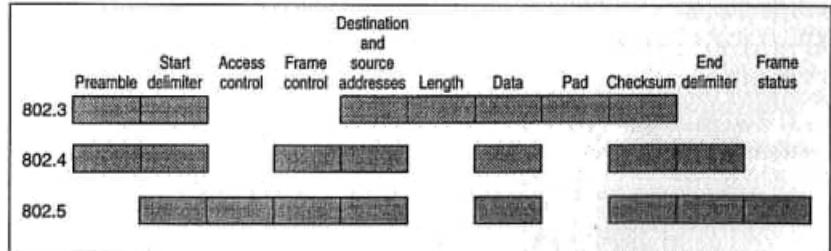


Fig. 4-36 I formati di pacchetto IEEE 802.

Un altro problema è che le LAN interconnesse non sempre hanno la stessa velocità di dati. Quando si stanno trasmettendo molti frame da una LAN veloce a una più lenta, il bridge può non essere in grado di gestire i frame alla velocità a cui arrivano. Dovrà "bufferizzarli", sperando di non esaurire la memoria. Il problema esiste anche nella trasmissione da una 802.4 a una 802.3 a 10 Mbps, poiché una parte della larghezza di banda della 802.3 viene sprecata a causa delle collisioni: essa non ha in realtà una velocità di 10 Mbps, mentre la 802.4 sì (più o meno). I bridge che connettono tre o più LAN hanno problemi simili quando diverse LAN stanno cercando di trasmettere verso la stessa LAN nello stesso istante.

Un problema ugualmente importante correlato alla questione "bridge = collo di bottiglia", riguarda il valore dei timer ai livelli più alti. Si supponga che il livello rete di una LAN 802.4 stia cercando di spedire un messaggio molto lungo come sequenza di frame. Dopo avere spedito l'ultimo inizializza un timer attendendo l'ack. Se il messaggio deve passare attraverso un bridge e raggiungere una LAN 802.5, c'è il pericolo che il timer scada prima che l'ultimo frame sia trasmesso sulla LAN più lenta. Il livello rete assumerà che il problema sia dovuto alla perdita di un frame e ritrasmetterà ancora l'intera sequenza. Dopo n tentativi falliti forse si arrenderà e dirà al livello trasporto che la destinazione è guasta.

Un'altra questione, potenzialmente la più seria di tutte, è che tutte le tre LAN 802 hanno un diverso massimo per la lunghezza del frame. Per la 802.3 essa dipende dai parametri della configurazione, ma per il sistema standard a 10 Mbps, il carico può essere al massimo 1500 byte. Per una 802.4 è fissato a 8191 byte. Per una 802.5 non c'è limite superiore, solo, una stazione non può trasmettere oltre il tempo di possesso del token. Con il valore solito di 10 ms, la lunghezza massima del frame è di 5000 byte.

Un problema ovvio sorge quando un frame lungo deve essere trasmesso a una LAN che non può accettarlo. Suddividere il frame in pezzi è fuori questione, in questo livello. Tutti i protocolli assumono che i frame o arrivano o non arrivano. Non ci sono meccanismi di riassemblaggio di un frame da unità più piccole. Non stiamo dicendo che un tale protocollo non potrebbe essere inventato, ma solo che 802 non offre questa possibilità. In sostanza, non c'è soluzione: i pacchetti che sono troppo larghi dovranno essere scartati. E questo è quanto per la trasparenza.

Ora, consideriamo brevemente ognuno dei nove casi di bridge da 802.x a 802.y per vedere quali altri problemi si nascondano. Da 802.3 a 802.3 è facile. Il solo caso che può non funzionare si ha quando la LAN destinazione è così carica che i frame continuano ad arrivare al bridge ma questo non riesce a liberarsene. Se questa situazione persiste abbastanza a lungo, il bridge riempirà il suo spazio di buffer e perderà dei frame. Poiché il problema è sempre potenzialmente presente quando si trasmette a una LAN 802.3, non lo menzioneremo più. Con le altre due LAN, ogni stazione, incluso il bridge, ha la garanzia di ottenere il token periodicamente e non potrà essere messa da parte per lunghi intervalli.

Da 802.4 a 802.3 si hanno due problemi. Primo, i frame 802.4 hanno bit di priorità che i frame 802.3 non hanno. Di conseguenza, se due LAN 802.4 comunicano attraverso una LAN 802.3, la priorità verrà persa nella LAN intermedia.

Il secondo problema è causato da una caratteristica specifica della 802.4: la cessione

temporanea del token. È possibile che un frame 802.4 abbia un bit 1 nell'intestazione per passare temporaneamente il token alla destinazione, permettendole di spedire un frame di ack. Tuttavia, se questo frame viene trasmesso a un bridge, esso cosa ne farà? Se trasmette esso stesso un ack sta mentendo, infatti il frame non è ancora stato consegnato a destinazione e quest'ultima, per quel che ne sa, potrebbe anche essere morta. D'altra parte, se non genera un ack il mittente concluderà assurdamente che la destinazione è guasta e trasmetterà una comunicazione di fallimento ai livelli superiori. Sembra che non ci sia modo di risolvere il problema.

Da 802.5 a 802.3 si hanno problemi simili. Il formato di un frame 802.5 contiene i bit A e C nel byte di stato del frame. Questi bit sono attivati dalla destinazione per comunicare al mittente se la stazione indirizzata ha visto il frame e se lo ha copiato. Anche qui il bridge può mentire e dire che il frame è stato copiato, ma se più tardi si scopre che la destinazione è guasta sorgeranno parecchi problemi. In sostanza, l'inserimento di un bridge sulla rete ha cambiato la semantica dei bit. È difficile immaginare una soluzione adatta a questa situazione.

Da 802.3 a 802.4 si ha il problema di cosa inserire nei bit di priorità. Un buon modo può essere quello di far ritrasmettere al bridge i frame tutti a priorità massima, poiché essi hanno probabilmente già accumulato abbastanza ritardo.

Da 802.4 a 802.4 il solo problema consiste nel come gestire la cessione temporanea del token. Almeno qui il bridge ha la possibilità di spedire i frame abbastanza velocemente in modo da avere la risposta prima che il timer scada. È comunque ancora un rischio. Spedendo i frame alla massima priorità, il bridge sta dicendo una piccola bugia ma questo serve a incrementare la probabilità di ottenere una risposta in tempo.

Da 802.5 a 802.4 si hanno ancora problemi con i bit A e C. Inoltre, la definizione dei bit di priorità è diversa per le due LAN, anche se il numero di bit è lo stesso. Tutto quello che i bridge possono fare è di trasmettere i bit di priorità e di sperare per il meglio.

Da 802.3 a 802.5 il bridge deve essere in grado di generare bit di priorità, ma non si hanno altri problemi particolari. Da 802.4 a 802.5 esiste un problema potenziale con i frame troppo lunghi e uno con la cessione temporanea del token. Infine, da 802.5 a 802.5 il problema è ancora come gestire i bit A e C. La figura 4-37 riassume i vari problemi appena discussi.

Quando il comitato IEEE 802 decise di definire uno standard per le LAN, non fu in grado di accordarsi su uno in particolare e produsse *tre* standard incompatibili, come abbiamo già detto: per questo venne ampiamente criticato. Quando più tardi gli venne assegnato il compito di progettare uno standard per i bridge per interconnettere le sue tre LAN decise di fare meglio. E così fu. Ne derivarono *due* progetti incompatibili. Da allora nessuno gli ha più chiesto di progettare un gateway standard per connettere i suoi due bridge incompatibili, ma almeno la tendenza (numerica) è in direzione giusta (sta decrescendo).

Questo paragrafo ha trattato i problemi che si incontrano nel connettere due LAN IEEE 802 attraverso un singolo bridge. I prossimi due paragrafi tratteranno il problema dell'interconnessione di larghe reti composte contenenti più LAN e più bridge e i due approcci IEEE per la progettazione di questi bridge.

		LAN destinazione		
		802.3 (CSMA/CD)	802.4 (token bus)	802.4 (token ring)
LAN sorgente	802.3	1, 4	1, 2, 4, 8	
	802.4	1, 5, 8, 9, 10	9	1, 2, 3, 8, 9, 10
	802.5	1, 2, 5, 6, 7, 10	1, 2, 3, 6, 7	6, 7

Azioni:

1. Riformattare il frame e calcolare una nuova checksum.
2. Voltare l'ordine dei bit.
3. Ricopiare la priorità, che abbia significato o meno.
4. Generare una priorità fittizia.
5. Scartare la priorità.
6. Scartare l'anello (in qualche modo).
7. Attivare i bit A e C.
8. Preoccuparsi della conversione (LAN veloci verso LAN lenti).
9. Preoccuparsi della impossibilità o del ritardo della cessione token per l'ACK.
10. Problemi se il frame è troppo lungo per la LAN destinazione.

Parametri assunti:

802.3	frame di 1500 bytes	10 Mbps (meno le collisioni)
802.4	frame di 8191 bytes	10 Mbps
802.5	frame di 5000 bytes	4 Mbps

Fig. 4-37 I problemi che si incontrano nel costruire bridge da 802.x a 802.y.

4.4.2 Bridge trasparenti

I primi bridge 802 sono il **bridge trasparente** e il **bridge ad albero di attraversamento** (Perlman, 1992). Lo scopo delle persone che supportarono il progetto era quello di avere completa trasparenza. Nella loro visione, chi avesse dovuto gestire un sito con più LAN avrebbe dovuto poter comprare un bridge progettato come standard IEEE, connetterlo e istantaneamente tutto avrebbe dovuto funzionare alla perfezione. Non avrebbero dovuto essere necessari cambiamenti hardware o software, nessun adattamento di indirizzi di commutazione, nessuno scaricamento di tabelle o parametri di routing. Sarebbe dovuto bastare connettere il bridge al cavo. Inoltre, le operazioni delle LAN esistenti non avrebbero dovuto essere condizionate dai bridge. Sorprendentemente, questo è stato il risultato.

Un bridge trasparente opera in modo adattivo accettando ogni pacchetto trasmesso da tutte le LAN cui è connesso. Come esempio si consideri la configurazione in figura 4-38. Il bridge B1 è connesso alle LAN 1 e 2 e il bridge B2 è connesso alle LAN 2, 3 e 4. Un frame sulla LAN 1 che arriva al bridge B1 destinato ad A potrà essere scartato immediatamente, poiché esso è già sulla LAN giusta, ma un frame che arrivi dalla LAN 1 con destinazione C o F dovrà essere trasmesso dal bridge.

Quando arriva un frame, un bridge deve decidere se scartarlo o trasmetterlo, e nel secondo caso a quale LAN trasmetterlo. Questa decisione viene presa cercando l'indirizzo destinazione in una grossa tabella (hash) contenuta nel bridge. La tabella può elencare tutte le possibili destinazioni e dire a quale linea di output (LAN) esso appartiene. Ad esempio, la tabella di B2 elencherà A tra le appartenenti alla LAN 2, poiché tutto quello che B2

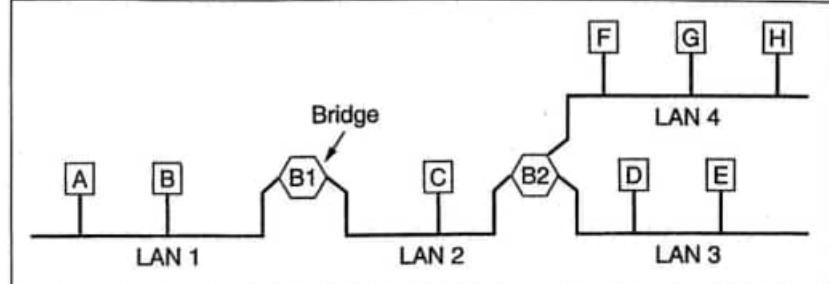


Fig. 4-38 Una configurazione con quattro LAN e due bridge.

deve sapere è su quale linea emettere il frame per A. Se si avranno poi altre trasmissioni più tardi, questo non interessa al bridge.

Quando i bridge vengono collegati per la prima volta tutte le tabelle hash sono vuote. Nessuno dei bridge sa dove sono le destinazioni e utilizzano l'algoritmo di flooding: ogni pacchetto in arrivo per una destinazione sconosciuta viene emesso su tutte le LAN a cui il bridge è connesso eccetto quella da cui proviene. Con il tempo i bridge vengono a conoscenza delle locazioni delle destinazioni, come descritto di seguito. Una volta che la destinazione è nota, i pacchetti destinati ad essa vengono emessi solo sulla LAN corretta. L'algoritmo utilizzato dai bridge trasparenti è l'apprendimento all'indietro (**backward learning**). Come detto in precedenza, i bridge operano in modo adattivo, quindi essi catturano ogni frame inviato da una delle LAN a cui sono connessi. Osservando l'indirizzo di provenienza essi possono dire quale macchina è accessibile e su quale LAN. Ad esempio se il bridge B1 in figura 4-38 vede un frame sulla LAN 2 proveniente da C, esso sa che C sarà accessibile attraverso la LAN 2, allora inserirà un nuovo dato nella sua tabella hash annotando che i frame diretti a C devono essere emessi sulla LAN 2. Ogni successivo frame per C proveniente dalla LAN 1 sarà trasmesso, ma un frame per C proveniente dalla LAN 2 sarà scartato.

La topologia può cambiare quando le macchine e i bridge vengono attivati, disattivati o spostati. Per gestire topologie dinamiche, quando viene inserito un ingresso per la tabella hash, viene memorizzato nell'ingresso anche il momento di arrivo del frame. Ogni volta che arriva un frame la cui destinazione è già nella tabella, la sua casella nella tabella viene aggiornata con il tempo corrente. Così il tempo associato con ogni casella indica l'ultima volta in cui si è ricevuto un frame da quella macchina.

Periodicamente, un processo nel bridge scandisce la tabella e ripulisce tutte le caselle con tempi più vecchi di pochi minuti. In questo modo se un computer viene staccato dalla sua LAN, spostato e riattivato su di un'altra LAN dell'edificio, in qualche minuto sarà di nuovo in grado di trasmettere perfettamente, senza nessun intervento manuale. Questo algoritmo comporta anche che se una macchina resta silente per pochi minuti, ogni frame diretto ad essa dovrà essere diffuso su tutte le LAN, finché essa non invierà un frame.

La procedura di instradamento di un frame in arrivo dipende dalla LAN di provenienza e dalla LAN a cui è destinato, come segue:

1. Se le LAN di destinazione e provenienza sono la stessa LAN, si scarti il frame.
2. Se le LAN di destinazione e provenienza sono diverse si trasmetta il frame.
3. Se la LAN di destinazione è sconosciuta si usi la procedura di flooding.

Ogni volta che arriva un frame occorre applicare la procedura. Esistono processori VLSI il cui compito specifico è quello di attuare la ricerca e l'aggiornamento delle celle della tabella, tutto in pochi microsecondi.

Per aumentare l'affidabilità alcune volte vengono utilizzati due o più bridge in parallelo tra coppie di LAN, come in figura 4-39. Questa tecnica però introduce anche altri problemi, poiché crea cicli nella topologia.

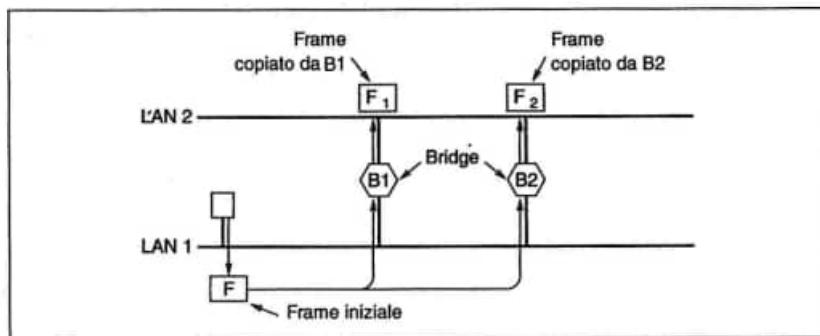


Fig. 4-39 Due bridge trasparenti paralleli.

Un esempio semplice di questo problema è riportato in figura 4-39, che descrive come viene gestito un pacchetto F con destinazione sconosciuta. Ogni bridge, seguendo le solite regole per la gestione di destinazioni ignote usa flooding, che nel caso dell'esempio significa solo copiare il pacchetto sulla LAN 2. Poco dopo, il bridge 1 vede F_2 , un pacchetto con destinazione ignota, e lo copia sulla LAN 1, generando F_3 (non mostrato). In modo analogo, il bridge 2 copia F_1 nella LAN 1 e genera F_4 (anch'esso non mostrato). Il bridge 1 quindi trasmette F_4 e il bridge 2 copia F_3 . Il ciclo continua all'infinito.

Bridge ad albero di attraversamento

La soluzione a questa difficoltà è quella di permettere ai bridge di comunicare tra di loro e di ricoprire la reale topologia con un albero di attraversamento che raggiunga ogni LAN. In effetti, alcune connessioni potenziali tra le LAN vengono ignorate ai fini di costruire una topologia fittizia senza cicli. Ad esempio, in figura 4-40 (a), si vedono nove LAN interconnesse mediante dieci bridge. Questa configurazione può essere astratta per ottenere un grafo con le LAN come nodi. Un arco connette una qualsiasi coppia di LAN che sia connessa mediante un bridge. Il grafo può essere ridotto a un albero di attraversamento cancellando gli archi tratteggiati in figura 4-40 (b). Utilizzando questo albero di attraversamento, si ha esattamente un percorso da una qualsiasi LAN a una qualsiasi altra. Una volta che i bridge si siano accordati sull'albero di attraversamento, tutte le trasmissioni tra le LAN seguiranno l'albero di attraversamento. Poiché si ha un percorso unico tra ogni coppia sorgente e destinazione, non ci saranno cicli.

samento, si ha esattamente un percorso da una qualsiasi LAN a una qualsiasi altra. Una volta che i bridge si siano accordati sull'albero di attraversamento, tutte le trasmissioni tra le LAN seguiranno l'albero di attraversamento. Poiché si ha un percorso unico tra ogni coppia sorgente e destinazione, non ci saranno cicli.

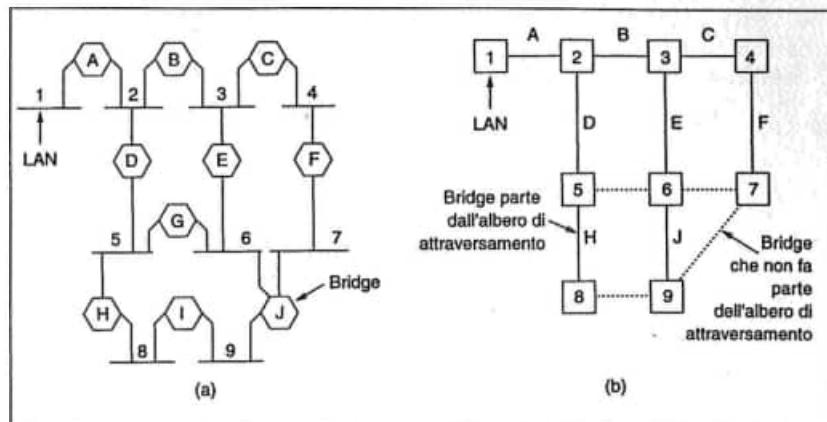


Fig. 4-40 (a) LAN interconnesse. (b) Un albero di attraversamento che copre le LAN. Le linee tratteggiate non sono parte dell'albero di instradamento.

Per costruire l'albero di attraversamento, prima i bridge devono scegliere uno di loro come radice dell'albero. Essi attuano questa scelta facendo tutti broadcast del proprio numero di serie, installato dal produttore e unico a livello mondiale. Il bridge con il numero più basso diventa radice. Quindi, viene costruito l'albero di cammino minimo tra la radice e ogni altro bridge e ogni altra LAN. Questo albero sarà l'albero di instradamento. Se un bridge o una LAN si rompono, ne sarà costruito un'altro.

Il risultato di questo algoritmo è che viene stabilito un percorso unico da ogni LAN alla radice e quindi ad ogni altra LAN. Sebbene l'albero attraversi tutte le LAN, non tutti i bridge sono necessariamente presenti nell'albero (per impedire cicli). Anche dopo la costruzione dell'albero, l'algoritmo continua a girare per rilevare automaticamente cambiamenti nella topologia e aggiornare l'albero. L'algoritmo distribuito utilizzato per costruire l'albero di instradamento fu inventato da Perlman ed è descritto in dettaglio in (Perlman, 1992).

I bridge possono anche essere utilizzati per connettere LAN molto distanti. In questo modello, ogni sito consiste di una collezione di LAN e bridge, di cui uno connesso con una WAN. I frame per le LAN remote viaggiano sulla WAN. Può essere utilizzato l'algoritmo di base dell'albero di attraversamento, preferibilmente con certe ottimizzazioni per scegliere un albero che minimizzi la quantità di traffico sulla WAN.

4.4.3 Bridge di instradamento da sorgente

I bridge trasparenti hanno il vantaggio di essere di facile installazione. Basta connetterli e funzionano. D'altra parte però essi non fanno un uso ottimale della larghezza di banda, infatti utilizzano solo un sottoinsieme della topologia (l'albero di instradamento). L'importanza relativa di questi due fattori (e di altri) portò a una scissione nel comitato 802 (Pitt, 1988). I sostenitori di CSMA/CD e di token bus scelsero i bridge trasparenti. I sostenitori di token ring (con l'incoraggiamento dell'IBM) preferirono una tecnica detta **intradamento da sorgente** che adesso descriveremo. Per ulteriori dettagli si veda Dixon (1987).

In linee essenziali, l'intradamento da sorgente assume che il mittente di ogni frame sappia se la destinazione del frame appartiene alla sua LAN. Quando spedisce un frame per una LAN diversa, la macchina sorgente attiva il bit di ordine più alto dell'indirizzo sorgente e include nell'header l'esatto cammino che il frame dovrà percorrere.

Questo cammino viene costruito nel modo seguente. Ogni LAN possiede un numero unico di 12 bit e ogni bridge possiede un numero di 4 bit che lo identifica univocamente nel contesto delle sue LAN. Così due bridge lontani possono avere entrambi il numero 3, ma due bridge tra due stesse LAN dovranno avere numeri diversi. Un intradamento (route) sarà quindi una sequenza di numeri di bridge, LAN, bridge, LAN e così via. Con riferimento alla figura 4-38, l'intradamento da A a D sarà (L1, B1, L2, B2, L3).

Un bridge di intradamento da sorgente è interessato solo a quei frame che hanno attivato il bit di ordine più alto nell'indirizzo destinazione attivato. Per ogni frame di questo tipo esso scandisce l'intradamento cercando il numero della LAN da cui proviene, se questo numero di LAN è seguito dal suo numero di bridge, esso trasmette il frame sulla LAN il cui numero segue il suo numero di bridge nell'intradamento. Se la LAN da cui il frame arriva è seguito dal numero di qualche altro bridge esso non trasmette il frame.

Questo algoritmo si presta a tre diverse implementazioni:

1. Software: il bridge funziona in modo adattivo, copiando tutti i frame nella sua memoria per controllare il bit più alto nell'indirizzo destinazione. Se è attivo il frame viene elaborato, altrimenti no.
2. Irido: l'interfaccia della LAN del bridge ispeziona il bit di ordine più alto dell'indirizzo destinazione e accetta il frame solo se il bit è attivato. Questa interfaccia è facile da costruire in hardware e riduce di molto il numero dei frame che il bridge deve ispezionare.
3. Hardware: l'interfaccia della LAN del bridge non controlla solo il bit di ordine più alto dell'indirizzo destinazione ma scandisce anche l'intradamento per vedere se questo bridge dovrà attuare la trasmissione o meno. Solo i frame che dovranno essere trasmessi vengono passati al bridge. Questa implementazione richiede un hardware complesso ma non perde cicli di CPU del bridge poiché tutti i frame non rilevanti vengono schermati dall'interfaccia.

Queste tre implementazioni hanno diversi costi e prestazioni. Il primo non introduce costi addizionali per l'hardware dell'interfaccia ma può richiedere CPU molto veloci per la

4.4 I bridge

gestione di tutti i frame. L'ultima richiede un chip speciale VLSI ma solleva il bridge da buona parte dell'elaborazione, quindi è possibile usare una CPU più lenta o, in alternativa, il bridge può gestire più LAN.

È implicito nel progetto dell'intradamento da sorgente che ogni macchina del sistema conosca, o possa determinare il cammino migliore per raggiungere ogni altra macchina. Come questi intradamenti vengano scoperti è una caratteristica importante dell'algoritmo. L'idea base è che se la destinazione è sconosciuta, la sorgente spedisce un frame di broadcast chiedendo dove si trova. Questo **frame di scoperta** viene trasmesso da ogni bridge e raggiunge quindi ogni LAN del sistema. Quando si riceve una risposta, i bridge memorizzano il loro identificativo in essa, così il mittente di origine possa osservare l'esatto intradamento seguito e scegliere quello migliore.

Sebbene questo algoritmo trovi l'intradamento migliore (li trova tutti), si ottiene un'esplosione di frame. Si consideri la configurazione della figura 4.41, con N LAN connesse linearmente mediante tre bridge. Ogni frame di scoperta spedito dalla stazione 1 viene copiato da ognuno dei tre bridge sulla LAN 1, portando tre frame di scoperta sulla LAN 2. Ognuno di questi viene copiato da ogni bridge sulla LAN 2, portando nove frame sulla LAN 3. Quando si raggiunge la LAN N , sono in circolazione 3^{N-1} frame. Se viene attraversata una dozzina di bridge, nell'ultima LAN saranno immessi più di un milione di pacchetti di scoperta, causando una congestione pesante.

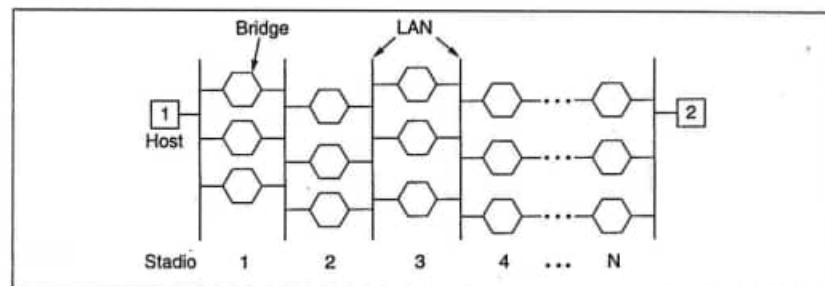


Fig. 4-41 Una serie di LAN connesse da bridge tripli.

Un processo analogo si ha anche con bridge trasparenti, solo che in questo caso non è così pericoloso. Quando arriva un pacchetto sconosciuto esso viene trasmesso, ma solo sull'albero di attraversamento, quindi la quantità totale di pacchetti spediti rimane lineare nella dimensione della rete, non esponenziale.

Una volta che un host ha scoperto un intradamento per una certa destinazione, lo memorizza nella cache, in modo che il processo di scoperta non debba essere rieseguito. Sebbene questo approccio limiti piuttosto bene l'impatto dell'esplosione del numero di pacchetti, assegna anche un pesante incarico di amministrazione agli host; inoltre l'intero algoritmo è decisamente non trasparente, sebbene questo fosse uno degli scopi inizialmente prefissi, come si è accennato in precedenza.

4.4.4 Confronti tra i bridge 802

I bridge trasparenti e a instradamento da sorgente hanno speciali vantaggi e svantaggi. In questo paragrafo ne discuteremo alcuni. La figura 4-42 fornisce un riassunto e ulteriori dettagli si ritrovano in Soha, Perlman (1988); Zhang (1988). Si sottolinea tuttavia che ciascuno di questi punti è ampiamente dibattuto.

Caratteristica	Bridge trasparente	Bridge a instradamento da sorgente
Orientamento	Senza connessione	Orientato alla connessione
Trasparenza	Completamente trasparente	Non trasparente
Configurazione	Automatica	Manuale
Intradamento	Quasi ottimale	Ottimale
Localizzazione	Backward learning	Frame di scoperta
Guasti	Gestito dai bridge	Gestiti dagli host
Complessità	Nel bridge	Negli host

Fig. 4-42 Confronti tra bridge trasparenti e a instradamento da sorgente.

La differenza principale tra i due tipi di bridge è la distinzione tra operatività senza connessione oppure orientata alla connessione. I bridge trasparenti non prevedono circuiti virtuali e instradano ogni frame indipendentemente da ogni altro. I bridge di instradamento da sorgente invece, ne determinano uno utilizzando i frame di scoperta e lo riutilizzano sempre, una volta che lo scoprano.

I bridge trasparenti sono completamente invisibili agli host e pienamente compatibili con tutti i prodotti 802 esistenti. I bridge a instradamento da sorgente non sono né trasparenti né compatibili. Per utilizzarli gli host devono conoscerne perfettamente il posizionamento e devono partecipare attivamente. La suddivisione di una LAN in due LAN connesse da un bridge di questo tipo comporta cambiamenti anche a livello del software degli host.

Quando si usano bridge trasparenti non è necessario alcun tipo di manutenzione. I bridge si autoconfigurano in relazione alla topologia. Con i bridge a instradamento da sorgente, l'amministratore della rete deve manualmente installare i numeri delle LAN e dei bridge. Errori come la duplicazione dei numeri di LAN o bridge possono risultare di difficile correzione, infatti possono causare cicli nell'intradamento dei frame oppure la mancata trasmissione di alcuni su qualche instradamento. Inoltre, quando si connettono due reti prima sconnesse, se si utilizzano bridge trasparenti non si deve far altro che collegarli, mentre con gli altri può essere necessario modificare manualmente diversi numeri di LAN per renderli unici sulla rete interconnessa.

Uno dei pochi vantaggi dell'intradamento da sorgente è che, in teoria, può utilizzare la via ottimale, mentre i bridge trasparenti si limitano a considerare gli alberi di attraversamento. Inoltre, l'intradamento da sorgente può anche fare buon uso di bridge paralleli tra due LAN al fine di ridurre il carico. Se i bridge reali siano però così astuti da sfruttare questi vantaggi teorici è una questione aperta.

4.4 I bridge

Con i bridge trasparenti la localizzazione di una destinazione avviene mediante apprendimento all'indietro mentre con gli altri mediante i frame di scoperta. Lo svantaggio dell'apprendimento all'indietro è che i bridge devono attendere di ricevere un frame da quella particolare macchina prima di sapere dove questa si trovi. Lo svantaggio dei frame di scoperta è invece l'esplosione del numero di frame se si connettono ampie reti con bridge in parallelo.

La gestione dei guasti è alquanto diversa nelle due tecniche. I bridge trasparenti vengono a conoscenza dei guasti alle LAN o ai bridge o dei cambiamenti nella topologia piuttosto velocemente e in modo automatico, osservando i frame di controllo degli altri bridge; gli host non notano per nulla questi cambiamenti.

Con l'intradamento da sorgente la situazione è diversa: quando un bridge si guasta, le macchine notano subito di non ricevere ack dei frame che spediscono, quindi il loro timer scade ed esse ritentano la spedizione più e più volte. Infine, concludono che c'è qualcosa che non va, ma non capiscono ancora quale sia il problema, se con la destinazione stessa oppure con l'intradamento. Solo spedendo un altro frame di scoperta esse si accorgono se la destinazione è attiva. Sfortunatamente, quando un bridge importante si guasta i timer di molti host scadono ed essi dovranno spedire frame di scoperta prima che il problema sia risolto, anche se è disponibile una via alternativa. Questa grande vulnerabilità ai guasti è una delle più gravi debolezze di tutti i sistemi orientati alla connessione.

Parliamo infine di complessità e di costi, punti molto dibattuti. Se i bridge a instradamento da sorgente posseggono chip VLSI che accettano solo i frame che devono essere trasmessi, avranno un carico minore da elaborare e avranno prestazioni migliori dato un certo investimento hardware. Senza questo chip si comporteranno in modo peggiorio poiché l'elaborazione necessaria per ogni frame (ricerca dell'intradamento nell'intestazione del frame) è sostanzialmente maggiore.

In più l'intradamento da sorgente rende gli host più complessi: essi infatti devono memorizzare i percorsi, spedire frame di scoperta e copiare le informazioni sui percorsi in ogni frame. Tutte queste cose richiedono memoria e cicli di CPU. Poiché ci sono di solito uno o due ordini di grandezza tra il numero di host e il numero di bridge, potrebbe convenire avere costo e complessità maggiori nei pochi bridge piuttosto che in tutti gli host.

4.4.5 Bridge remoti

Un uso comune dei bridge è quello di connettere due o più LAN distanti. Ad esempio, un'azienda può avere filiali in diverse città, ciascuna con una propria LAN. Idealmente tutte le LAN dovrebbero essere interconnesse, in modo che l'intero sistema si comporti come una singola grande LAN.

Questo scopo può essere raggiunto mettendo un bridge su di ogni LAN e connettendo i bridge a coppie con linee punto-a-punto (ad es. con linee dedicate). Un semplice sistema con tre LAN viene mostrato in figura 4.43. Viene applicato il solito algoritmo di instradamento. Il modo più semplice di considerare questo esempio è di vedere le tre linee punto-a-punto come LAN senza host. Otteniamo quindi un normale sistema con sei LAN interconnesse da quattro bridge. Niente di ciò che abbiamo studiato impone che una LAN abbia degli host.

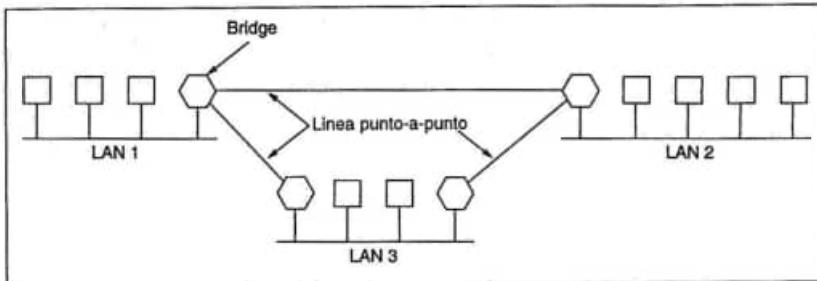


Fig. 4-43 I bridge remoti possono essere usati per interconnettere LAN distanti.

Possono essere utilizzati diversi protocolli su queste linee punto-a-punto. Una possibilità è quella di scegliere alcuni protocolli standard data link punto-a-punto, mettendo gli interi frame MAC nel campo di carico. Questo metodo funziona meglio se tutte le LAN sono identiche e l'unico problema è consegnare i frame alle LAN giuste. Un'altra possibilità è di eliminare l'header e il trailer MAC nel bridge sorgente e di inserire ciò che resta nel campo di carico del protocollo punto-a-punto. Sul bridge di destinazione possono quindi essere generate una nuova intestazione e una nuova coda. Uno svantaggio di questo approccio è che la somma di controllo che raggiunge la destinazione non è quella calcolata dalla sorgente, quindi non si possono rilevare errori dovuti a bit sporchi nella memoria del bridge.

4.5 LAN ad alta velocità

Le LAN e MAN 802 appena studiate sono tutte costruite con un cavo di rame (due cavi nel caso della 802.6). Per basse velocità e distanze piccole, questo può bastare ma per alte velocità e distanze grandi le LAN devono essere costruite con fibra ottica o cavi di rame altamente paralleli. La fibra ha una notevole larghezza di banda, è sottile e leggera, non è soggetta a interferenza elettromagnetica da parte di macchinari pesanti (importante quando i cavi corrono tra i pozzi degli ascensori), variazioni di potenza, fulmini ed è molto sicura poiché è quasi impossibile collegarsi al cavo senza accorgersene. Di conseguenza le LAN veloci solitamente sono in fibra. Nei paragrafi seguenti considereremo alcune reti locali che utilizzano fibra ottica, e una LAN ad alta velocità che utilizza il vecchio modello di cavi di rame.

4.5.1 FDDI

FDDI (Fiber Distributed Data Interface – interfaccia di dati distribuita in fibra) è una LAN token ring in fibra ottica con alte prestazioni con velocità di 100 Mbps su distanze fino a 200 km e fino a 1000 stazioni connesse (Black, 1994; Jain, 1994; Mirchandani, Khanna, 1993; Ross, Hamstra, 1993; Shah, Ramakrishnan, 1994; Wolter, 1990). Può essere usata nello stesso modo in cui vengono usate le altre LAN 802, ma, siccome possiede parecchia banda, un altro possibile uso è come dorsale delle altre LAN in rame,

come in figura 4-44. FDDI-II è la versione successiva di FDDI, modificata per gestire dati PCM sincroni a commutazione di circuito per la voce o per il traffico ISDN, oltre che per i normali dati. Le chiameremo entrambe FDDI. In questo paragrafo avremo a che fare con il livello fisico e il sottolivello MAC per FDDI.

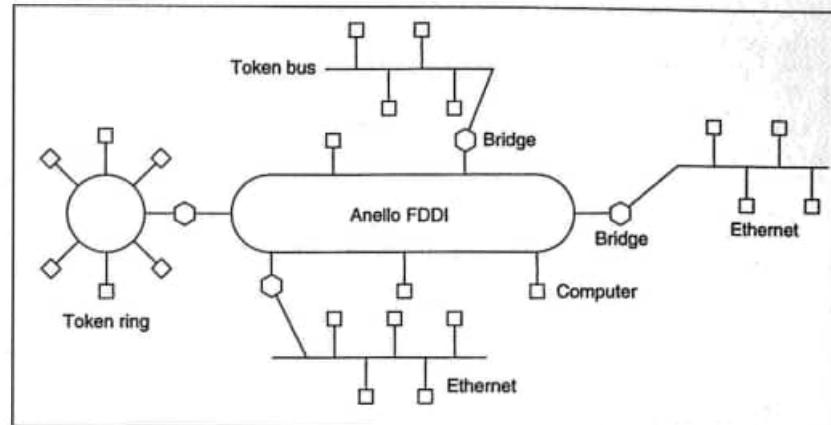


Fig. 4-44 Un anello FDDI usato come dorsale per connettere LAN e computer.

FDDI utilizza fibre multimodali poiché la spesa aggiuntiva di fibre unimodali non è giustificata per reti a velocità di 100 Mbps. Utilizza anche LED invece di laser, non solo per il loro basso costo, ma anche perché FDDI può essere occasionalmente utilizzato per connettere direttamente le workstation utenti, e può succedere che gli utenti più curiosi possano casualmente staccare il connettore della fibra e osservare i bit che viaggiano a 100 Mbps sulla fibra, mettendo in serio pericolo i propri occhi. I LED sono troppo deboli per causare danni alla vista ma sono abbastanza potenti per trasferire dati a 100 Mbps. La specifica del progetto di FDDI indica che non si ha più di un errore su $2,5 \times 10^{10}$ bit. Molte implementazioni fanno anche meglio.

Il cablaggio FDDI consiste di due anelli in fibra, uno che trasmette in senso orario e l'altro in senso inverso, come in figura 4-45 (a). Se se ne rompe uno, l'altro può essere usato di riserva. Se si rompono entrambi nello stesso punto, ad esempio a causa di un incendio o di un'altra catastrofe nel condotto del cavo, i due anelli possono essere ricongiunti a formare un singolo anello più o meno lungo il doppio, come in figura 4-45 (b). Ogni stazione contiene dei relè che possono essere usati per congiungere i due anelli o per saltare la stazione in caso di guasti. Possono essere usati anche concentratori come in 802.5.

FDDI definisce due classi di stazioni, A e B. Le stazioni della classe A sono connesse a entrambi gli anelli. Le stazioni più economiche della classe B sono connesse solo a uno degli anelli. A seconda di quanta importanza viene data alla tolleranza ai guasti, una installazione può decidere se avere stazioni della classe A o B, o un po' di ognuna delle due.

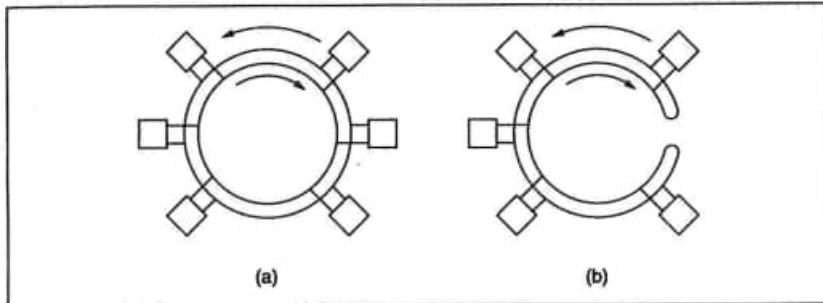


Fig. 4-45 (a) FDDI consiste di due anelli in senso opposto. (b) Nel caso si guastino entrambi gli anelli in un punto, i due anelli possono essere congiunti a formare un lungo anello singolo.

Il livello fisico non utilizza la codifica Manchester poiché a 100 Mbps richiede 200 MBd (megabaud), che è troppo costoso. Viene invece utilizzata un'altra tecnica, detta **4 su 5**. Ogni gruppo di quattro simboli MAC (0, 1 e certi simboli non di dati come inizio-frame) vengono codificati come un gruppo di 5 bit sul mezzo. 16 delle 32 combinazioni sono per i dati, 3 per i limitatori, 2 per il controllo, 3 per segnali hardware e 8 sono inutilizzate (cioè riservate per versioni future del protocollo).

Il vantaggio di questa tecnica è che risparmia larghezza di banda, ma lo svantaggio è la perdita della proprietà di autosincronizzazione della codifica Manchester. Per compensare questa perdita viene usato un lungo preambolo per sincronizzare l'orologio del ricevente con quello del mittente. Inoltre, viene richiesto che tutti gli orologi siano stabili almeno allo 0,005%. Con questa stabilità i frame fino a 4500 byte possono essere trasmessi senza pericolo che l'orologio del ricevente perda la sincronizzazione con il flusso di dati.

I protocolli di base FDDI sono stati modellati in modo simile a quelli 802.5. Per trasmettere dati una stazione deve prima catturare il token. Quindi essa trasmette un frame e lo rimuove quando ritorna. Una differenza tra FDDI e 802.5 è che in 802.5 una stazione non può generare un nuovo token finché il suo frame non ha percorso tutto il giro ed è ritornato. In FDDI invece, dove si possono avere 1000 stazioni e 200 km di fibra, la quantità di tempo persa per attendere che il frame finisca il giro potrebbe essere notevole, per questo si è deciso di permettere a una stazione di generare un nuovo token appena essa abbia finito di trasmettere i suoi frame. In un anello lungo molti frame possono essere sull'anello nello stesso istante.

I frame FDDI sono molto simili a quelli 802.5. Il formato FDDI è mostrato in figura 4-46. I campi *Start delimiter* e *Ending delimiter* indicano le estremità del frame. Il campo *Frame control* indica il tipo del frame (dati, controllo ecc.). Il byte di *Frame status* contiene i bit di ack, simili a quelli in 802.5. Gli altri campi sono simili a quelli in 802.5.

Oltre ai frame normali (asincroni), FDDI prevede anche frame sincroni speciali per PCM a commutazione di circuito oppure dati ISDN. I frame sincroni sono generati ogni 125 µs da una stazione master per fornire gli 8000 campioni al secondo necessari ai sistemi PCM.

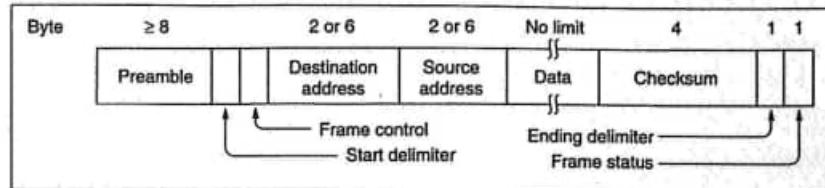


Fig. 4-46 Il formato di frame FDDI.

Ognuno di questi frame ha un header, 16 byte di dati non a commutazione di circuito e fino a 96 byte di dati a commutazione di circuito (cioè fino a 96 canali PCM per frame).

Il numero 96 venne scelto poiché permette di avere in un singolo frame quattro canali T1 (4×24) a 1,544 Mbps oppure tre canali E1 CCITT (3×32) a 2,048 Mbps, rendendo la stazione adatta ad essere usata ovunque nel mondo. Un frame sincrono ogni 125 µs consuma 6,144 Mbps di larghezza di banda per i 96 canali a commutazione di circuito. Un massimo di 16 frame sincroni ogni 125 µs permette fino a 1536 canali PCM e consuma fino a 98,3 Mbps.

Una volta che una stazione abbia acquisito uno o più slot di tempo in un frame sincrono, questi slot saranno riservati a lui finché non vengano esplicitamente rilasciati. La larghezza di banda totale non utilizzata dal frame sincrono viene allocata su richiesta. Una maschera di bit è presente in ognuno di questi frame a indicare gli slot liberi per le richieste di assegnamento. Il traffico non sincrono viene suddiviso in classi di priorità, con le priorità più alte che ottengono la precedenza sulla larghezza di banda lasciata libera. Il protocollo MAC FDDI utilizza tre timer. Il **timer di possesso del token** determina per quanto tempo ancora una stazione può continuare a trasmettere, una volta acquisito il token. Questo timer evita che una stazione occupi l'anello all'infinito. Il **timer di rotazione del token** viene azzerato ogni volta che passa il token. Se il timer scade significa che il token non è passato per un intervallo di tempo troppo lungo. Probabilmente è andato perso e occorre iniziare la procedura di recupero del token. Infine, il **timer di trasmissione valida** viene usato per ristabilizzare alcune situazioni di errori momentanei sull'anello.

FDDI possiede anche un algoritmo di priorità simile a quello di 802.4. Esso determina quali classi di priorità possono trasmettere a un dato passaggio di token. Se il token è in anticipo, tutte le priorità possono trasmettere, ma se è in ritardo, possono spedire solo quelle più alte.

4.5.2 Fast Ethernet

FDDI doveva essere la LAN della prossima generazione, invece non è andata molto oltre il mercato delle dorsali (dove continua ad andare piuttosto bene). La gestione delle stazioni è troppo complicata e implica chip complessi e costi elevati. Il costo notevole dei processori FDDI ha portato i costruttori di elaboratori a non volere FDDI come rete standard, quindi non si è mai avuta una produzione ingente e FDDI non è mai stata introdotta nel mercato di massa. Quello che si è imparato è che "semplice è bello".

In ogni caso il fallimento di FDDI lasciò spazio per LAN a velocità oltre 10 Mbps. Molte installazioni avevano bisogno di una maggiore larghezza di banda e aumentava il numero di LAN interconnesse attraverso ripetitori, bridge, router e gateway, sebbene ogni tanto agli amministratori sembrasse che questi fossero collegati fra loro per miracolo.

Fu in quel momento che IEEE riconvocò nel 1992 il comitato 802.3, con le istruzioni di creare una LAN più veloce. Una proposta fu di considerare 802.3 com'era e renderla più veloce. Un'altra di ridisegnarla completamente per introdurre molte nuove caratteristiche come quelle per il traffico a tempo reale e voce digitalizzata, mantenendo il vecchio nome (per ragioni di marketing). Dopo diverse discussioni il comitato decise di mantenere 802.3 com'era e di renderla più veloce. I sostenitori dell'altra proposta fecero quello che persone dell'industria dei computer avrebbero fatto in quel caso, formarono il loro comitato e standardizzarono la loro LAN (802.12).

Le tre ragioni primarie per la decisione del comitato 802.3 furono:

1. La necessità di essere compatibili con migliaia di LAN già esistenti.
2. La paura che un nuovo protocollo presentasse problemi non previsti.
3. Il desiderio di finire il lavoro prima che la tecnologia fosse mutata.

Il lavoro fu terminato velocemente (rispetto ai tempi di altri comitati) e il risultato, **802.3u**, venne ufficialmente approvato dall'IEEE nel giugno del 1995. Tecnicamente 802.3u non è un nuovo standard, ma un'appendice di 802.3 (per enfatizzare la compatibilità col passato). Poiché tutti la chiamano **fast Ethernet** (Ethernet veloce) piuttosto che 802.3u, adotteremo questo nome anche noi.

L'idea di base di fast Ethernet è semplice: mantenere tutti i vecchi formati di pacchetto, le interfacce, e le regole procedurali, riducendo però il tempo di bit da 100 ns a 10 ns. Tecnicamente sarebbe stato possibile copiare 10Base-5 o 10Base-2 e rilevare ancora le collisioni in tempo riducendo la lunghezza massima del cavo di un fattore 10. Tuttavia i vantaggi dei collegamenti 10Base-T erano troppi, quindi fast Ethernet è basata totalmente su questo progetto. Tutti i sistemi fast Ethernet utilizzano hub; non sono permessi cavi a più salti con prese a vampiro o connettori BCN.

Tuttavia è stato necessario operare alcune scelte, la più importante delle quali è stata decidere che tipo di cavo utilizzare. Una delle alternative è stata il doppino intrecciato di categoria 3. I suoi vantaggi stanno nel fatto che praticamente ogni ufficio del mondo occidentale possiede almeno quattro doppini di categoria 3 (o superiore) che raggiungono un telefono distante fino a 100 m. In alcuni casi di questi collegamenti ce ne sono due. Quindi, utilizzando doppini di categoria 3 è possibile collegare computer usando fast Ethernet senza dover rifare i collegamenti nell'edificio, un enorme vantaggio per molte organizzazioni.

Il più grave svantaggio dei doppini di categoria 3 consiste nella loro incapacità di trasportare segnali di 200 MBd (100 Mbps con codifica Manchester) per 100 m, la distanza massima concessa tra un hub e il computer secondo 10Base-T (vedere la figura 4-17). Al contrario, i doppini di categoria 5 possono gestire i 100 m più agevolmente e la fibra può andare anche oltre. Il compromesso scelto fu quello di permettere tutte e tre le possibilità,

come mostrato in figura 4-47, migliorando però la soluzione della categoria 3 al fine di permettere la necessaria capacità di trasporto.

Nome	Cavo	Distanza massima	Vantaggi
100Base-T4	Doppino	100 m	Usa UTP di categoria 3
100Base-TX	Doppino	100 m	Full duplex a 100 Mbps
100Base-F	Fibra ottica	2000 m	Full duplex a 100 Mbps; lunghi percorsi

Fig. 4-47 Cablaggio di fast Ethernet.

La tecnica di categoria 3 UTP, detta **100Base-T4**, utilizza una velocità di segnale di 25 MHz, solo il 25% più veloce dei 20 MHz di 802.3 (si ricordi che la codifica Manchester, come in figura 4-20, richiede due periodi di orologio per ognuno dei 10.000.000 di bit ogni secondo). Per raggiungere la necessaria larghezza di banda, 100Base-T4 richiede quattro doppiini. Poiché i collegamenti telefonici standard hanno avuto per anni quattro doppiini per cavo, la maggior parte degli uffici era già predisposta. Ovvivamente ciò significava eliminare il telefono, ma questo è sicuramente un prezzo basso da pagare per avere posta elettronica più veloce.

Dei quattro doppiini, uno è sempre diretto allo hub, uno proviene dallo hub e gli altri due sono orientabili a seconda della direzione di trasmissione. Per ottenere la necessaria larghezza di banda, non si utilizza la codifica Manchester ma d'altra parte con orologi moderni e distanze corte essa non è necessaria. Inoltre, vengono inviati segnali ternari, quindi durante un singolo periodo di orologio il cavo può contenere uno 0, un 1 o un 2. Con tre doppiini che viaggiano in avanti e segnali ternari, ognuno dei 27 possibili simboli può essere trasmesso, rendendo possibile la spedizione di quattro bit con un minimo di ridondanza. Trasmettendo 4 bit in ognuno dei 25.000.000 di cicli di clock per secondo si ottengono 100 Mbps. Inoltre, si ha sempre un canale in senso contrario di 33,3 Mbps sul rimanente doppino. Questa tecnica, nota come **8B6T**, (8 bit su 6 trit) non è certo delle più eleganti ma funziona sul sistema di collegamento esistente, e non è poco!

Il progetto dei collegamenti di categoria 5, **100Base-TX**, è più semplice poiché i fili possono sostenere una velocità di orologio fino a 125 MHz e oltre. Si usano solo due doppiini per stazione, uno verso lo hub e l'altro proveniente da esso. Piuttosto che utilizzare direttamente la codifica binaria, viene utilizzata una tecnica chiamata **4B5B** a 125 MHz. Ogni gruppo di cinque periodi di clock viene usato per spedire 4 bit al fine di avere un po' di ridondanza, fornire abbastanza transizioni per permettere la sincronizzazione degli orologi, creare standard unici per i delimitatori di frame ed essere compatibili con FDDI a livello fisico. Di conseguenza 100-BaseTX è un sistema full duplex; le stazioni possono trasmettere a 100 Mbps e ricevere a 100 Mbps nello stesso momento. Inoltre, si possono avere due telefoni nello stesso ufficio per comunicazione vocale nel caso che il computer sia totalmente impegnato per la navigazione sul Web.

L'ultima tecnica, **100Base-FX**, utilizza due cavi di fibra multimodale, uno per direzione:

è quindi anch'esso full duplex con 100 Mbps per direzione. Inoltre, la distanza tra una stazione e lo hub può essere fino a 2 km.

Con 100Base-T4 e 100Base-TX, universalmente noti come **100Base-T**, sono possibili due tipi di hub. In uno hub condiviso tutte le linee in entrata (o almeno tutte le linee provenienti da una scheda connessa) sono logicamente collegate e formano un unico dominio di collisione. Possono essere applicate tutte le regole standard, incluso l'algoritmo di regressione binaria, quindi il sistema funziona come la vecchia 802.3. In particolare può trasmettere solamente una stazione per volta.

In uno hub commutato, ogni pacchetto in ingresso viene memorizzato su una scheda. Sebbene questa caratteristica renda lo hub e le schede più care, fa anche in modo che tutte le stazioni possano trasmettere (e ricevere) nello stesso istante, migliorando notevolmente la larghezza di banda del sistema, spesso di un intero ordine di grandezza o di più. I frame "bufferizzati" vengono passati su un backplane ad alta velocità, dalla scheda di origine a quella di destinazione. Il backplane non è stato standardizzato e non sembra sia necessario che lo sia, poiché è completamente nascosto all'interno del commutatore. Se l'esperienza è di qualche utilità, i venditori di commutatori saranno in competizione per la produzione di backplane più veloci al fine di migliorare il rendimento del sistema. Poiché i cavi 100Base-FX sono troppo lunghi per il solito algoritmo Ethernet di collisione, devono essere connessi a hub "bufferizzati" e commutati, in modo che ognuno formi un dominio di collisione singolo.

Infine, notiamo che virtualmente tutti i commutatori possono gestire un insieme di stazioni a 10 Mbps messe a stazioni a 100 Mbps, al fine di rendere più semplici i miglioramenti. A mano a mano che su un sito aumentano le stazioni a 100 Mbps, tutto quello che si deve fare è acquistare il numero necessario di schede e inserirle opportunamente nel commutatore.

Ulteriori informazioni riguardo a fast Ethernet si ritrovano in Johnson (1996). Per un confronto di reti locali ad alta velocità, in particolare di FDDI, fast Ethernet, ATM e VG-AnyLAN, si vedano Cronin *et al.* (1994).

4.5.3 HIPPI – interfaccia parallela ad alte prestazioni

Durante la guerra fredda, il laboratorio nazionale di Los Alamos, centro di progettazione di armi nucleari del governo americano, comprava abitualmente un esemplare di tutti i supercomputer che venivano presentati sul mercato. A Los Alamos si collezionarono anche periferiche singolari, come dispositivi di memoria di massa e workstation grafiche speciali per la visualizzazione scientifica. In quel periodo ogni ditta aveva una sua interfaccia di connessione delle periferiche ai suoi supercomputer, quindi non era possibile per più macchine condividere periferiche o connettere tra loro due supercomputer.

Nel 1987 i ricercatori di Los Alamos iniziarono a lavorare su un'interfaccia standard per supercomputer, con l'intenzione di ottenere una standardizzazione da consigliare ai venditori. (Dato il notevole budget impiegato a Los Alamos per l'acquisto di computer, i venditori erano inclini ad ascoltare i loro consigli). L'obiettivo era una interfaccia che ognuno potesse implementare velocemente ed efficientemente. Il principio guida fu KISS (Keep It Simple, Stupid – rendilo semplice, stupido). L'interfaccia non doveva avere

opzioni, non doveva richiedere la progettazione aggiuntiva di alcun chip e doveva avere le prestazioni di un idrante antincendio.

La specifica iniziale richiedeva una velocità di trasmissione dati di 800 Mbps, poiché ad esempio, osservare una bomba che esplode richiede 1024×1024 pixel con 24 bit per pixel e 30 frame/s, per una dimensione totale di velocità di 750 Mbps. Più tardi si introdusse un'opzione, una seconda velocità di 1600 Mbps. Quando questa proposta, detta **HIPPI** (High Performance Parallel Interface – interfaccia parallela ad alte prestazioni) fu offerta all'ANSI per la standardizzazione, coloro che l'avevano proposta vennero guardati come marziani, infatti in quel momento parlare di LAN significava parlare di Ethernet a 10 Mbps. HIPPI venne inizialmente progettato per essere un canale dati piuttosto che una LAN. I canali di dati operano punto-a-punto, da un padrone (un computer) a un servo (una periferica), con linee dedicate e nessun commutatore. Nessun caso di contesa e situazioni completamente prevedibili. Più tardi, si fece strada la necessità di utilizzare una periferica anche da altri computer, e un commutatore a crossbar venne aggiunto al progetto di HIPPI, come mostrato in figura 4-48.

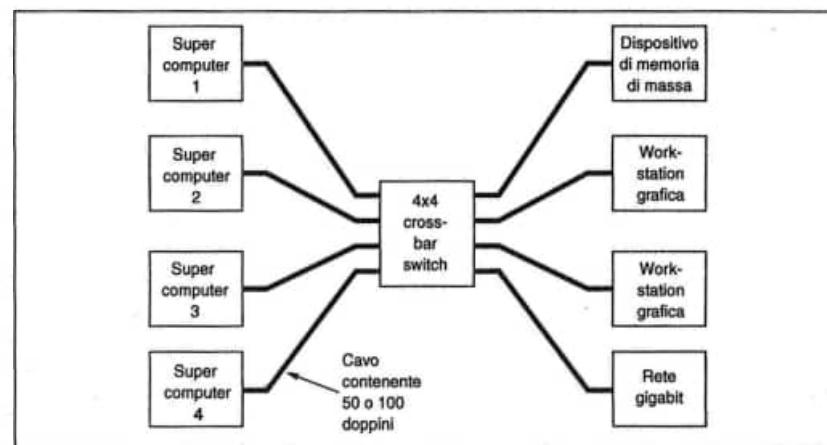


Fig. 4-48 HIPPI usa un commutatore a crossbar.

Al fine di ottenere altissime prestazioni, utilizzando solamente i chip già disponibili sul mercato, l'interfaccia di base è di 50 bit, 32 bit per i dati e 18 di controllo, quindi il cavo HIPPI contiene 50 doppini. Ogni 40 ns, viene trasferita una parola in parallelo sull'interfaccia. Per raggiungere i 1600 Mbps, vengono usati due cavi e vengono trasferite due parole per ciclo. Tutti i trasferimenti sono in un senso (simplex). Per ottenere comunicazione duplex, sono necessari due (o quattro) cavi. A queste velocità la lunghezza massima del cavo è di 25 m.

Dopo lo shock iniziale, il comitato ANSI X3T9.3 elaborò uno standard HIPPI secondo le indicazioni provenienti da Los Alamos. Lo standard copre i livelli fisico e data link. Tutto il resto è a discrezione dell'utente. Il protocollo di base è quello per comunicare, prima

un host chiede al commutatore crossbar di definire una connessione, quindi (solitamente) invia un solo messaggio e rilascia la connessione.

I messaggi sono composti da una parola di controllo, un'intestazione che può essere lunga fino a 1016 byte e una sezione dati fino a $2^{32}-2$ byte. A causa del controllo del flusso, i messaggi vengono suddivisi in frame di 256 parole. Quando il ricevente è in grado di accettare un frame lo segnala al mittente che invia quindi il messaggio. I riceventi possono anche richiedere più frame alla volta. Il controllo degli errori consiste in un bit di parità orizzontale per parola e un bit di parità verticale alla fine di ogni frame. Le tradizionali checksum vennero considerati inutili e troppo lente.

HIPPI venne velocemente implementato da dozzine di vendori e rimase lo standard di interconnessione dei supercomputer per anni. Per ulteriori informazioni si vedano Hughes, Franta (1994); Tolmie (1992); Tolmie, Renwick (1993).

4.5.4 Canali a fibra

Quando HIPPI fu progettato, la fibra ottica era troppo cara e considerata non abbastanza affidabile, quindi le LAN più veloci erano comunque costruite su cavi telefonici. Con l'andare del tempo però la fibra divenne più economica e affidabile e sembrò naturale tentare di ridefinire HIPPI usando una singola fibra invece di 50 o 100 doppini. Sfortunatamente la costanza che ebbe Los Alamos nel respingere le proposte di nuove caratteristiche ogni volta che si presentavano, fu eccessiva. Il successore di HIPPI, detto **canale a fibra**, è molto più complicato e più costoso da realizzare. Se sarà capace di riscuotere il successo commerciale di HIPPI è ancora da verificare.

I canali a fibra gestiscono sia i canali di dati che le connessioni di rete. In particolare possono essere usati per canali di dati di HIPPI, SCSI e il canale multiplexor utilizzato sui mainframe IBM. Può anche trasportare pacchetti di rete, compresi IEEE 802, IP e ATM. Come per HIPPI, la struttura di base dei canali a fibra è un crossbar che connette gli input agli output. Le connessioni possono essere stabilite per pacchetti singoli o per intervalli di tempo più lunghi.

I canali a fibra prevedono tre diverse classi di servizi; la prima classe è a commutazione di circuito semplice con garanzia di consegna ordinata. Le modalità a canali di dati utilizzano questa classe di servizio. La seconda classe è commutazione di pacchetto con garanzia di consegna. La terza è commutazione di servizio senza garanzia di consegna.

Il canale a fibra possiede una struttura di protocollo molto elaborata, come mostrato in figura 4-49. Si possono vedere cinque livelli, che insieme coprono i livelli fisico e data link. Il livello più basso ha a che fare con il mezzo fisico. Finora prevede velocità dati di 100, 200, 400 a 800 Mbps. Il secondo livello gestisce la codifica dei bit. La strategia usata è simile a quella di FDDI, ma invece di usare 5 bit per codificare 16 simboli validi, usa 10 bit per codificarne 256, fornendo un po' di ridondanza. Questi due livelli insieme sono funzionalmente equivalenti al livello fisico OSI.

Il livello intermedio definisce la struttura dei frame e il formato degli header. I dati vengono trasmessi in frame il cui carico può arrivare a 2048 byte. Il livello successivo permette che in futuro possano essere forniti i comuni servizi al livello più alto. Infine, il livello più alto fornisce le interfacce ai diversi tipi di computer e periferiche supportati. Stranamente, sebbene il canale a fibra sia stato progettato negli Stati Uniti, il nome venne

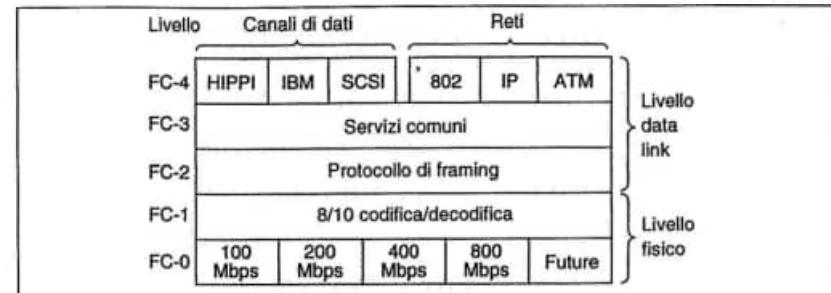


Fig. 4-49 I livelli di protocollo del canale a fibra.

scelto dall'autore dello standard che era inglese. Ulteriori informazioni sul canale a fibra si ritrovano in Tolmie (1992). Un confronto di esso con HIPPI e ATM si trova in Tolmie (1995).

4.6 Reti satellitari

Sebbene la maggior parte dei canali multiaccesso si ritrovino nelle LAN, anche un tipo di WAN utilizza canali multiaccesso: le WAN basate su comunicazione via satellite. In seguito studieremo brevemente alcuni problemi che si riscontrano nelle WAN satellitari. Approfondiremo anche lo studio di alcuni dei protocolli che sono stati ideati per gestire queste reti.

In genere i satelliti di comunicazione hanno circa una dozzina di trasponditori. Ogni trasponditore ha un fascio che copre una parte della Terra sotto di lui, a partire da una porzione molto ampia di 10.000 km fino a una piccola di 250 km. Le stazioni che ricadono nel fascio possono inviare frame al satellite alla frequenza di salita. Il satellite poi li ridifonde alla frequenza di discesa. Vengono utilizzate diverse frequenze per la salita e la discesa per evitare che il trasponditore vada in oscillazione. I satelliti che non fanno alcuna elaborazione, ma che ripetono a una diversa frequenza ciò che sentono (o la maggior parte di ciò che sentono) sono detti satelliti **bent pipe** (canali piegati).

Ogni antenna può rivolgersi verso una certa area, trasmettere qualche pacchetto e quindi rivolgersi verso un'altra area. Il puntamento verso un'area viene fatto elettronicamente ma richiede comunque un certo numero di microsecondi. La quantità di tempo in cui un'antenna è rivolta verso una certa area è detta **dwell time** (tempo di sosta). Per avere la massima efficienza non dovrebbe essere troppo corto, altrimenti la maggior parte del tempo sarebbe spesa nello spostare l'antenna.

Proprio come nelle LAN un punto importante della progettazione è come allocare i canali dei trasponditori. Tuttavia, diversamente dalle LAN, l'ascolto sul cavo è impossibile a causa del ritardo di propagazione di 270 ms. Quando una stazione ascolta il canale di discesa ascolta cosa è successo 270 ms prima. Ascoltare il canale di salita è praticamente impossibile. Quindi, i protocolli CSMA/CD (che assumono che la stazione trasmittente possa rilevare le collisioni entro i primi secondi e quindi ritirarsi nel-

caso ce ne siano state) non può essere utilizzato con i satelliti. Occorrono quindi altri protocolli.

Vengono usate cinque classi di protocolli sul canale di salita multiaccesso: polling, ALOHA, FDM, TDM e CDMA. Sebbene abbiamo già studiato ognuno di questi protocolli, le operazioni dei satelliti aggiungono talvolta alcune complicazioni. I problemi principali si hanno con il canale di salita, poiché il canale di discesa ha solo un mittente (il satellite) e quindi non si hanno problemi di allocazione del canale.

4.6.1 Polling

Il modo tradizionale in cui si alloca un canale singolo avendo diversi utenti che competono per esso è quello di avere qualcuno che li interroghi ciclicamente (polling). Fare in modo che il satellite interroghi ogni stazione a turno per sapere se ha un pacchetto è assolutamente troppo costoso a causa dei 270 ms richiesti per ogni sequenza interrogazione/risposta.

Tuttavia se tutte le stazioni a terra sono collegate a una rete a commutazione di pacchetto (tipicamente a bassa larghezza di banda), una variazione di questa soluzione diventa accettabile. L'idea è quella di sistemare le stazioni in un anello logico, in modo che ognuna conosca il suo successore nel cerchio. Su questo anello circolerà un token. Il satellite non vedrà mai il token, una stazione potrà trasmettere sul canale di salita solo quando sarà in possesso del token. Se il numero di stazioni è piccolo costante, il tempo di trasmissione del token sarà breve e la raffica inviata sul canale di salita sarà molto più lunga del tempo di rotazione del token: lo schema quindi sarà piuttosto efficiente.

4.6.2 ALOHA

L'implementazione di ALOHA puro è piuttosto semplice: ogni stazione trasmette quando vuole. Il problema è che l'efficienza del canale è solo del 18%. In genere un così basso fattore di utilizzazione è inaccettabile nel caso dei satelliti che costano decine di milioni di dollari.

Utilizzando ALOHA a slot l'efficienza raddoppia ma si introduce il problema di come sincronizzare tutte le stazioni in modo che tutte sappiano quando inizia ogni slot di tempo. Fortunatamente il satellite stesso è in grado di gestire la cosa, essendo un mezzo di diffusione. Una delle stazioni a terra, la **stazione di riferimento**, trasmette periodicamente un segnale speciale la cui ridiffusione viene usata da tutte le stazioni a terra come origine di tempo. Se gli slot di tempo hanno tutti una lunghezza di ΔT , ogni stazione ora sa che lo slot di tempo k inizierà al tempo $k\Delta T$ dopo l'origine del tempo. Poiché gli orologi hanno velocità lievemente diverse, è necessaria una risincronizzazione periodica per mantenere tutti in fase. Una complicazione aggiuntiva è che il tempo di propagazione dal satellite è diverso per ogni stazione a terra, ma questo effetto può essere corretto.

Per aumentare l'utilizzo del canale di salita oltre $1/e$, si può passare dal canale di salita singolo della figura 4.50 (a) alla struttura di salita doppia della figura 4.50 (b). Una stazione che ha un pacchetto da trasmettere sceglie uno dei due canali di salita in modo casuale e spedisce un pacchetto nello slot successivo. Ogni canale di salita funziona come un canale ALOHA a slot indipendente.

Se uno dei canali di salita contiene un unico pacchetto questo sarà trasmesso successiva-

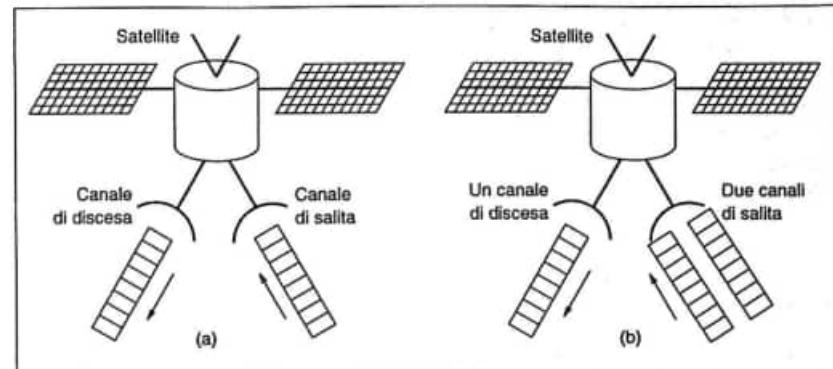


Fig. 4-50 (a) Un sistema ALOHA standard. (b) Viene aggiunto un secondo canale di salita.

mente nello slot di discesa corrispondente. Se entrambi i canali hanno successo, il satellite può memorizzare uno dei pacchetti e trasmetterlo in seguito durante un intervallo libero. Calcolando le probabilità si può mostrare che, data una quantità infinita di spazio di buffer, l'utilizzazione del canale di discesa può essere aumentata fino allo 0,736 a patto di incrementare le richieste di larghezza di banda di 1/2.

4.6.3 FDM

Il multiplexing a suddivisione di frequenza è il più vecchio e probabilmente ancora il più utilizzato metodo di allocazione del canale. Un comune trasponditore di 36 Mbps può essere staticamente suddiviso in circa 500 canali PCM a 64.000 bps, ognuno operante alla propria frequenza (unica) senza interferire con gli altri.

Sebbene sia semplice, FDM ha alcuni svantaggi. Primo, sono necessarie delle guardie di banda tra i canali per tenere separate le stazioni. Questa caratteristica è necessaria poiché non è possibile costruire trasmettitori che emanino tutta la loro energia sulla banda principale e nulla sulle bande vicine. La quantità di larghezza di banda persa per le guardie di banda potrebbe essere una frazione notevole del totale.

Secondo, le stazioni devono essere di potenza controllata. Se una stazione emana troppa potenza nella banda principale, automaticamente emetterà anche troppa potenza nelle bande laterali, debordando nei canali adiacenti e causando interferenza. Infine, FDM è interamente analogico e non si presta a implementazioni software.

Se il numero di stazioni è piccolo e costante, i canali di frequenza possono essere precedentemente allocati in modo statico. Tuttavia, se il numero di stazioni, o il carico su ognuna possono variare rapidamente, sono necessarie diverse forme di allocazione dinamica delle bande di frequenza. Uno di questi meccanismi è il sistema SPADE, utilizzato in alcuni dei primi satelliti Intelsat. Ogni trasponditore SPADE era suddiviso in 794 canali vocali simplex PCM (di 64 kbps), più un canale di segnalazione a 128 kbps, comune. I canali PCM erano utilizzati a coppie per ottenere un servizio full duplex. La larghezza di

banda totale del trasponditore utilizzata era di 50 Mbps per la parte di salita e di 50 Mbps per la parte di discesa.

Il canale di segnalazione comune era suddiviso in unità di 50 ms. Un'unità conteneva 50 intervalli di 1 ms (128 bit). Ogni intervallo veniva acquisito da una (e non di più) delle 50 stazioni a terra. Quando una stazione a terra doveva inviare dati, sceglieva in modo casuale uno dei canali al momento non usati e scriveva il numero di quel canale nel suo successivo slot di 128 bit. Se il canale selezionato era ancora inutilizzato quando la richiesta passava nel canale di discesa, veniva considerato allocato e le altre stazioni non potevano più richiederlo. Se due o più stazioni tentavano di allocare lo stesso canale, si aveva collisione ed esse avrebbero dovuto ritentare in seguito. Quando una stazione terminava di usare il canale trasmetteva un messaggio di deallocazione nel suo slot sul canale comune.

4.6.4 TDM

Come FDM, TDM è ben conosciuto e largamente utilizzato. Esso richiede sincronizzazione per gli slot, ma questo si ottiene usando una stazione di riferimento, come descritto sopra nel caso di ALOHA a slot. Come per FDM, per un numero di stazioni piccolo e costante, l'assegnamento dello slot può essere fatto in anticipo e non cambia mai, ma per un numero di stazioni variabile o per un numero costante di stazioni con carico variabile, gli slot di tempo devono essere assegnati dinamicamente.

L'assegnamento dello slot può essere fatto in modo centralizzato o decentralizzato. Come esempio di assegnamento centralizzato consideriamo l'**ACTS sperimental**e (Advanced Communication Technology Satellite – satellite a tecnologia di comunicazione avanzata), che fu progettato per poche dozzine di stazioni Palmer, White (1990). L'ACTS venne lanciato nel 1992. Ha quattro canali TDM indipendenti di 100 Mbps, due di salita e due di discesa. Ogni canale è organizzato come una sequenza di frame di 1 ms, ognuno contenente 1728 slot di tempo. Ogni slot di tempo ha una capacità di carico di 64 bit e permette a ciascuno di usufruire di un canale vocale di 64 kbps.

L'antenna può essere riposizionata da un'area geografica all'altra, ma poiché spostare l'antenna richiede diversi slot di tempo, i canali che iniziano e terminano nella stessa area geografica sono assegnati a tempi di slot contigui in modo da aumentare il tempo di sosta (dwell time) e minimizzare il tempo perso con lo spostamento dell'antenna. Quindi la gestione dei tempi di slot richiede una conoscenza totale della disposizione delle stazioni per minimizzare il numero di slot di tempo persi. Per questa e altre ragioni, la gestione degli slot di tempo viene attuata da una delle stazioni a terra, la **MCS** (Master Control Station – la stazione principale di controllo).

L'operazione di base di ACTS consiste in un processo continuo di tre passi, dove ogni passo dura 1 ms. Al primo passo il satellite riceve un pacchetto e lo memorizza in una RAM locale con 1728 ingressi. Al passo 2 un computer locale copia ogni ingresso in una uscita corrispondente (possibilmente per l'altra antenna). Al passo 3 il pacchetto in output viene trasmesso sul canale di discesa.

Inizialmente ad ogni stazione viene assegnato almeno uno slot di tempo. Per ottenere canali aggiuntivi (per nuove chiamate vocali), una stazione invia un breve messaggio di richiesta alla MCS. In modo similare essa può rilasciare un canale con un messaggio alla MCS. Questi messaggi fanno uso di un piccolo numero di bit aggiuntivi e forniscono un

canale speciale di controllo all'MCS con una capacità di circa 13 messaggi al secondo per stazione. I canali sono dedicati; cioè non si ha contesa per essi.

È anche possibile l'allocazione dinamica degli slot TDM. Di seguito discuteremo tre metodi. In ognuno di questi i frame TDM vengono suddivisi in slot di tempo, dove ogni slot ha un possessore (temporaneo). Solo il possessore può usare uno slot di tempo.

Il primo metodo assume che ci siano più slot di stazioni, in modo che ogni stazione possa avere uno slot di base (Binder, 1975). Se ci sono più slot di stazioni gli slot extra non sono assegnati ad alcuna stazione. Se il possessore di uno slot non lo vuole utilizzare in quel momento lo slot rimane libero. Uno slot vuoto è un segnale per le altre stazioni e indica che il possessore non ha nulla da trasmettere. Durante il frame successivo lo slot diventa disponibile per chiunque lo voglia, in base a una contesa tipo ALOHA.

Se il possessore vuole recuperare il suo slot di base, trasmette un pacchetto forzando una collisione (se c'era altro traffico). Dopo una collisione ognuno eccetto il possessore dovrà desistere dall'utilizzare lo slot per il pacchetto successivo. In questo modo il possessore può sempre iniziare a trasmettere entro il tempo di due frame nel caso peggiore. In caso di basso utilizzo del canale il sistema non si comporta esattamente come ALOHA a slot, poiché dopo ogni collisione gli interessati devono astenersi per un frame per vedere se il possessore vuole indietro lo slot. La figura 4-51 (a) mostra un frame con 8 slot, 7 dei quali appartengono rispettivamente a G, A, F, E, B, C e D. L'ottavo slot non appartiene a nessuno e può essere conteso.

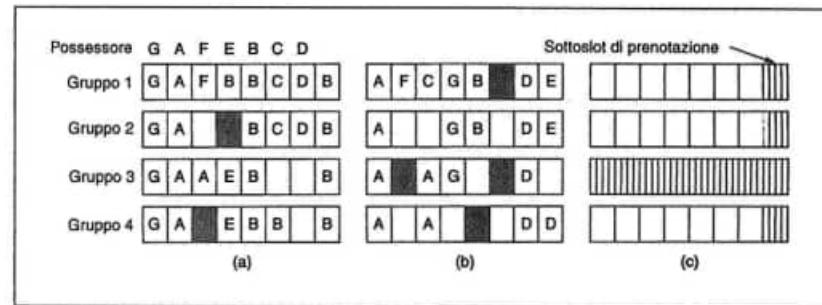


Fig. 4-51 Metodi di prenotazione. (a) Binder. (b) Crowther. (c) Roberts. I blocchi grigi indicano collisione. Per ognuno dei tre metodi sono mostrati quattro gruppi di slot consecutivi.

Un secondo metodo è applicabile anche quando non è noto il numero di stazioni, oppure è variabile (Crowther et al., 1973). In questo metodo gli slot non hanno un possessore permanente come nel metodo di Binder. Invece le stazioni competono per gli slot usando ALOHA a slot. Quando una trasmissione ha successo, la stazione che l'ha attuata viene assegnata a quello slot anche nel pacchetto successivo. Così, finché una stazione ha dati da inviare può continuare a farlo indefinitamente (pur essendo soggetta a regole di cortesia). In sostanza, il metodo propone un mix dinamico di ALOHA a slot e di TDM, con la possibilità di variare il numero degli slot dedicato ad ognuna. La figura 4-51 (b) mostra anch'essa un pacchetto con 8 slot. Inizialmente E sta usando l'ultimo slot, ma dopo due

frame non ne ha più bisogno. Resta inattiva per un frame e quindi D si impossessa dello slot e lo tiene finché non ha terminato.

Un terzo metodo, ideato da Roberts (1973), richiede che le stazioni facciano in anticipo le richieste prima di trasmettere. Ogni pacchetto contiene uno slot speciale (l'ultimo in figura 4-51 (c)) suddiviso in V sottoslot utilizzati per riservare gli slot. Quando una stazione vuole trasmettere diffonde un breve pacchetto di richiesta in un sottoslot per le prenotazioni in modo casuale. Se la richiesta ha successo (cioè non ci sono collisioni) il successivo slot o i successivi slot vengono riservati. Tutte le volte le stazioni devono tenere conto della lunghezza della coda (il numero di slot riservati), in modo che quando una stazione riserva con successo saprà quanti slot di dati dovrà saltare prima di trasmettere. Le stazioni non hanno bisogno di tenere traccia di chi è in coda; devono solo sapere quanto attendere. Quando la lunghezza della coda è zero, tutti gli slot diventano slot di prenotazione per velocizzare il processo.

Sebbene TDM sia ampiamente utilizzato, sia con che senza prenotazione, ha anch'esso degli svantaggi. Primo, è necessario che tutte le stazioni si sincronizzino, cosa non semplice in pratica poiché i satelliti tendono a muoversi in orbita modificando il tempo di propagazione di ogni stazione a terra. Esso richiede anche che ogni stazione a terra sia capace di un'altissima velocità di trasmissione a raffica. Ad esempio, anche se una stazione ACTS può avere solo un canale di 64 kbps, dev'essere capace di emettere una raffica di 64 bit in uno slot di tempo di 578 ns. In altre parole deve trasmettere a 110 Mbps. Al contrario una stazione FDM a 64 kbps trasmette davvero a 64 kbps.

4.6.5 CDMA

L'ultimo metodo è il CDMA, che non ha il problema di sincronizzazione e nemmeno quello di allocazione del canale. È completamente decentralizzato e totalmente dinamico. Tuttavia ha tre grossi svantaggi. Primo, la capacità di un canale CDMA in presenza di rumore e stazioni non coordinate è generalmente più bassa del livello che TDM può raggiungere. Secondo, con 128 chip/bit (un valore comune), anche se il tasso di bit non è elevato, il tasso di chip lo sarà, essendo necessario un trasmettitore veloce (leggi: costoso). Terzo, pochi ingegneri esperti conoscono realmente CDMA, cosa che in genere non aumenta le possibilità che lo utilizzi anche se è il metodo migliore per un'applicazione particolare. Comunque CDMA è stato usato dai militari per decenni e sta ora diventando più comune anche nelle applicazioni commerciali.

4.7 Riassunto

Alcune reti hanno un unico canale che viene usato per tutte le comunicazioni. Il punto focale del progetto di queste reti è l'allocazione del canale tra le stazioni che competono per usarlo. Sono stati inventati numerosi algoritmi di allocazione del canale. Un riassunto di alcuni dei più importanti metodi di allocazione si trova in figura 4-52.

I più semplici sono FDM e TDM. Questi metodi sono efficienti quando il numero di stazioni è piccolo e il traffico è continuo. Entrambi sono ampiamente utilizzati in questi casi, ad esempio per suddividere la larghezza di banda nel caso di collegamenti via satellite che fungono da dorsali telefoniche.

Metodo	Descrizione
FDM	Dedica una banda di frequenza a ogni stazione
TDM	Dedica uno slot di tempo ad ogni stazione
ALOHA puro	Trasmissione non sincronizzata in ogni momento
ALOHA a slot	Trasmissione casuale in slot di tempo ben definiti
CSMA 1 persistente	Accesso multiplo con rilevamento della portante standard
CSMA non persistente	Ritardo casuale quando si trova il canale impegnato
CSMA p persistente	CSMA ma con una probabilità p di persistenza
CSMA/CD	CSMA ma sospende quando rileva una collisione
Mappa di bit	Schedulazione round robin usando una mappa di bit
Conto alla rovescia binario	La stazione pronta con n° più alto è la prossima
Attraversamento di albero	Riduzione della contesa con abitazione selettiva
Divisione a lunghezza d'onda	Uno schema FDM dinamico per fibra
MACA, MACAW	Protocolli LAN wireless
GSM	FDM+TDM per radio cellulare
CDPD	Radio e pacchetti con un canale AMPS
CDMA	Tutti parlano contemporaneamente ma diversi linguaggi
Ethernet	CSMA/CD con backoff binario esponenziale
Token bus	Anello logico sul bus fisico
Token ring	Acquisisci il token per spedire un frame
DQDB	Code distributive su una MAN a 2 bus
FDDI	Token ring in fibra ottica
HIPPI	Crossbar con 50-100 doppine
Canale a fibre	Crossbar con fibra ottica
SPADE	FDM con allocazione dinamica del canale
ACTS	TDM con allocazione centralizzata di slot
Binder	TDM con ALOHA quando il possessore non è interessato
Crowther	ALOHA con il possessore di slot che deve preservare gli slot
Roberts	Tempo di canale prenotato in anticipo da ALOHA

Fig. 4-52 Metodi di allocazione del canale per sistemi con canale comune.

Quando il numero delle stazioni è grande e variabile oppure il traffico è a raffiche, FDM e TDM si rivelano buone scelte. Il protocollo ALOHA, che sia a slot oppure no, che sia con o senza controllo, è stato proposto come alternativa. ALOHA e le sue molte varianti sono stati discussi a lungo, analizzati e utilizzati per sistemi reali.

Quando lo stato del canale può essere ascoltato, le stazioni possono evitare di iniziare una trasmissione mentre un'altra stazione sta trasmettendo. Questo metodo, di rilevamento di carico, ha portato a una varietà di protocolli che possono essere usati su LAN e MAN. È nota una classe di protocolli che eliminano del tutto la contesa, o almeno la riducono considerabilmente. Il conteggio binario a ritroso elimina completamente la contesa. Il protocollo di attraversamento dell'albero la riduce suddividendo dinamicamente le stazioni in due gruppi disgiunti, uno dei quali può trasmettere mentre l'altro non può. Esso prova a fare una divisione in modo tale che solo una stazione che sia pronta a trasmettere possa veramente farlo.

Anche le LAN senza filo hanno i loro problemi e soluzioni. Il problema più grande è dovuto a stazioni nascoste, quindi CSMA non può funzionare. Una classe di soluzioni, rappresentate da MACA, tenta di stimolare le trasmissioni intorno alla destinazione, per fare in modo che CSMA possa funzionare meglio.

Nel caso di computer e telefoni mobili, la tecnologia radio cellulare è il futuro. GSM, CDPD e CDMA iniziano ad essere ampiamente utilizzati.

Le LAN IEEE 802 sono: CSMA/CD, token bus e token ring. Ognuna di queste ha i suoi vantaggi e svantaggi e ognuna ha trovato la propria comunità sostenitrice e continuerà probabilmente ad essere usata in quella comunità per gli anni a venire. La convergenza verso una singola LAN standard è un evento improbabile. Una nuova appartenente a questa famiglia è DQDB, che è venduta come MAN in molte città.

Un'organizzazione con più LAN le interconnette spesso mediante bridge. Quando un bridge connette due o più tipi diversi di LAN sorgono nuovi problemi, alcuni dei quali sono irrisolubili.

Mentre le LAN 802 sono i cavalli da tiro dei nostri giorni, i cavalli da corsa sono FDDI, fast Ethernet, HIPPI e il canale a fibra. Tutti questi offrono una larghezza di banda superiore a 100 Mbps.

Infine, anche le reti via satellite usano canali multiaccesso (in salita). Vengono usati vari metodi di allocazione del canale in questo caso, inclusi: ALOHA, FDM, TDM e CDMA.

Esercizi

1. Un gruppo di N stazioni condivide un canale ALOHA puro di 56 kbps. Ogni stazione trasmette in output un pacchetto di 1000 bit circa ogni 100 s, anche nel caso in cui quello precedente non sia ancora stato trasmesso (cioè le stazioni sono "bufferizzate"). Qual è il massimo valore di N ?
2. Si consideri il ritardo di ALOHA puro rispetto a quello di ALOHA a slot a basso carico. Qual è il minore? Motivare la risposta.
3. 10.000 stazioni di prenotazione di una linea aerea stanno competendo per l'uso di

un singolo canale ALOHA a slot. In media una stazione effettua 18 richieste all'ora. La durata di uno slot è di 125 μs. Qual è il carico approssimativo totale del canale?

4. Una grande popolazione di utenti di ALOHA riesce a generare 50 richieste al secondo, incluse quelle originali e le ritrasmissioni. Il tempo è suddiviso in slot di 40 ms.
 - a) Qual è la probabilità di successo al primo tentativo?
 - b) Qual è la probabilità di esattamente k collisioni prima di un successo?
 - c) Qual è il numero atteso di tentativi di trasmissione necessari?
5. Le misurazioni effettuate su un canale infinito di utenti di ALOHA a slot mostrano che il 10% degli slot è inutilizzato.
 - a) Qual è il carico G del canale?
 - b) Qual è la produttività?
 - c) Il canale è sottocarico o sovraccarico?
6. In un sistema ALOHA a slot di una popolazione infinita, il numero medio di slot che una stazione deve attendere tra una collisione e la sua ritrasmissione è 4. Si tracci la curva del ritardo in funzione della produttività per questo sistema.
7. Una LAN usa la versione Mok e Ward del conteggio binario alla rovescia. In un certo istante, le dieci stazioni hanno i numeri di stazioni virtuali 8, 2, 4, 5, 1, 7, 3, 6, 9 e 0. Le tre stazioni successive a inviare sono 4, 3, 9 in quest'ordine. Quali sono i nuovi numeri di stazioni virtuali dopo che tutte e tre hanno finito le loro trasmissioni?
8. Sedici stazioni sono in contesa per l'uso di un canale condiviso utilizzando il protocollo di attraversamento adattivo dell'albero. Se tutte le stazioni i cui indirizzi sono numeri primi divengono improvvisamente pronte in una sola volta, quanti slot di bit sono necessari per risolvere la contesa?
9. Un insieme di 2^n stazioni utilizza il protocollo di attraversamento adattivo dell'albero per gestire l'accesso a un cavo condiviso. A un certo istante due di esse divengono pronte. Quali sono il minimo, il massimo e il numero medio di slot per attraversare l'albero se $2^n > 1$?
10. Le LAN senza filo studiate utilizzano protocolli come MACA invece di CSMA/CD. Sotto quali condizioni sarebbe possibile usare invece CSMA/CD?
11. Che proprietà hanno in comune i protocolli di accesso al canale WDMA e GSM?
12. Utilizzando la struttura di suddivisione in frame GSM data in figura 4-14, si determini quanto spesso un dato utente può trasmettere un pacchetto di dati.
13. Si supponga che A , B e C stiano trasmettendo simultaneamente bit 0 usando un sistema CDMA con la sequenza di chip data in figura 4-16 (b). Qual è la sequenza di chip risultante?
14. Nella discussione riguardante l'ortogonalità delle sequenze di chip CDMA, si diceva che se $S \cdot T = 0$ allora $S \cdot \bar{T}$ è anch'esso 0. Dimostrare tale affermazione.

15. Si consideri un modo diverso di interpretare la proprietà di ortogonalità delle sequenze di processore CDMA. Ogni bit della coppia nella sequenza può fare match o meno. Si esprima la proprietà di ortogonalità in termini di matching.
16. Un ricevente CDMA riceve i seguenti chip: $(-1, +1, -3, +1, -1, +3, +1, +1)$. Assumendo la sequenza di chip definita in figura 4-16 (b), quali stazioni hanno trasmesso e quali bit ha spedito ognuna di esse?
17. Un edificio aziendale ha 7 piani e 15 uffici adiacenti per piano. Ciascun ufficio possiede una presa a muro per un terminale nella parete frontale, per cui le prese formano una griglia rettangolare nel piano verticale, con una separazione di 4 m tra le prese, sia in orizzontale che in verticale. Supponendo che sia possibile stendere un cavo diretto tra qualsiasi coppia di prese, in orizzontale, in verticale o in diagonale, quanti metri di cavo sono necessari per connettere tutte le prese mediante:
 - a) una configurazione a stella con un singolo router nel mezzo?
 - b) una LAN 802.3?
 - c) una rete ad anello (senza centro di cablaggio, concentratore)?
18. Qual è la velocità in baud dello standard 802.3 a 10 Mbps?
19. Una LAN CSMA/CD (non 802.3) a 10 Mbps, lunga 1 km, ha una velocità di propagazione di 200 m/ μ s. I pacchetti sono di 256 bit, compresi i 32 bit dell'intestazione, la checksum e altri appesantimenti. Il primo slot di bit dopo una trasmissione con successo è riservato al ricevente perché si impossessi del canale e invii un frame di ack di 32 bit. Qual è la velocità dati effettiva, escludendo l'appesantimento, assumendo che non ci siano collisioni?
20. Due stazioni CSMA/CD stanno tentando di trasmettere file molto lunghi (multiframe). Dopo la spedizione di ogni pacchetto, esse si contendono il canale utilizzando l'algoritmo di regressione binaria esponenziale. Qual è la probabilità che la contesa termini al tentativo k e qual è il numero medio di tentativi per ciascun periodo di contesa?
21. Si consideri di dover costruire una rete CSMA/CD con una velocità di 1 Gbps su di un cavo di 1 km senza ripetitori. La velocità del segnale sul cavo è di 200.000 km/s. Qual è la dimensione minima del pacchetto?
22. Si dia la codifica Manchester della stringa di bit: 0001110101.
23. Si dia la codifica Manchester differenziale della stringa di bit dell'esercizio precedente. Si assume che la linea sia inizialmente in stato basso.
24. Un sistema token bus funziona nel seguente modo. Quando il token raggiunge una stazione un timer viene inizializzato a 0. La stazione quindi inizia a trasmettere i frame di priorità 6 finché il timer non raggiunge T_6 . A questo punto passa ai frame di priorità 4 finché il timer non arriva a T_4 . Questo algoritmo viene ripetuto poi con priorità 2 e 0. Se tutte le stazioni hanno valori di timer di 40, 80, 90 e 100 ms per

- T_6 fino a T_0 rispettivamente, quale frazione della larghezza di banda totale è riservata per ciascuna classe di priorità?
25. Che cosa succede in un sistema token bus se una stazione accetta il token e quindi si guasta? Come gestisce questo caso il protocollo illustrato nel testo?
 26. A una velocità di trasmissione di 5 Mbps e con una velocità di propagazione di 200 m/ μ s, a quanti metri di cavo equivale il ritardo di 1 bit in un'interfaccia di token ring?
 27. Il ritardo totale in un token ring deve essere tale da contenere l'intero token. Se il cavo non è abbastanza lungo devono essere introdotti dei ritardi artificiali. Si spieghi perché questo ritardo extra è necessario nel contesto di una rete con un token di 24 bit e un anello con un ritardo di soli 16 bit.
 28. Una rete token ring sotto condizioni di carico pesante, lunga 1 km, a 10 Mbps, ha una velocità di propagazione di 200 m/ μ s. Cinquanta stazioni sono uniformemente distribuite sull'anello. I pacchetti sono lunghi 256 bit, compresi i 32 bit di overhead. Gli ack sono agganciati ai pacchetti di dati e sono quindi inclusi come bit sparsi nei pacchetti, sono quindi effettivamente gratis. Il token è di 8 bit. La velocità effettiva di questo anello è più alta o più bassa della velocità effettiva di una rete CSMA/CD a 10 Mbps?
 29. In una rete token ring è il trasmettitore che rimuove il pacchetto. Quali modifiche al sistema sarebbero necessarie per fare in modo che fosse il ricevente a rimuovere il pacchetto e quali sarebbero le conseguenze?
 30. Una rete token ring a 4 Mbps ha un timer di possesso del token con valore di 10 ms. Qual è il pacchetto più lungo che può essere inviato su questo anello?
 31. L'impiego di un concentratore ha influenza sulle prestazioni di token ring?
 32. Una token ring in fibra ottica utilizzata come MAN è lunga 200 km e opera a 100 Mbps. Dopo aver inviato un pacchetto, una stazione elimina il pacchetto dall'anello prima di rigenerare il token. La velocità di propagazione del segnale sulla fibra è di 200.000 km/s e la massima dimensione del pacchetto è di 1 Kbyte. Qual è la massima efficienza dell'anello (si trascurino tutte le fonti di appesantimento)?
 33. Nella figura 4-32 la stazione D vuole inviare una cella. A quale stazione vuole inviarla?
 34. Il sistema della figura 4-32 è inattivo. Poco dopo le stazioni C , A e B divengono pronte a trasmettere, in questo ordine e in rapida successione. Assumendo che non siano trasmessi frame di dati finché tutte e tre non abbiano inviato una richiesta, mostrare i valori di RC e CD dopo ogni richiesta e dopo i tre frame di dati.
 35. Si dice spesso che Ethernet non sia adatta per trasmissioni real-time poiché non si ha limite per l'intervallo di ritrasmissione nel caso peggiore. Sotto quali condizioni possono valere queste stesse argomentazioni per token ring? Sotto quali condizioni

- si ha un limite noto del caso peggiore per token ring? Si assuma che il numero di stazioni sull'anello sia fisso e noto.
36. I frame di Ethernet devono essere lunghi almeno 64 byte per assicurare che il mittente sia ancora attivo nel caso si abbia una collisione dal lato opposto del cavo. Fast Ethernet ha la stessa lunghezza minima di pacchetto di 64 byte ma può emettere bit dieci volte più velocemente. Com'è possibile mantenere la stessa dimensione minima di pacchetto?
 37. Si immaginino due bridge di LAN, che connettono entrambi una coppia di reti 802.4. Il primo bridge sopporta un carico di trasmissione di 1000 frame da 512 byte al secondo. Il secondo sopporta un carico di 200 frame da 4096 byte al secondo. Quale bridge avrà bisogno della CPU più potente? Si motivi la risposta.
 38. Si supponga che i due bridge dell'esercizio precedente connettano una LAN 802.4 a una 802.5. Queste modifiche hanno influenza sulla risposta precedente?
 39. Un bridge tra una LAN 802.3 e una 802.4 ha un problema con gli errori di memoria intermittenti. Può questo problema causare errori non rilevati nei frame trasmessi, oppure saranno tutti rilevati mediante la checksum?
 40. Un dipartimento di informatica di un'università è composto da tre segmenti Ethernet, connessi da due bridge trasparenti a formare una rete lineare. Un giorno l'amministratore di sistema si licenzia e viene frettolosamente rimpiazzato da qualcun altro del centro di calcolo, che è un centro IBM token ring. Il nuovo sistemista, notando che le due estremità della rete non sono collegate, ordina subito un bridge trasparente e le connette, creando un anello. Cosa succederà?
 41. Un anello FDDI molto esteso è formato da 100 stazioni e ha un tempo di rotazione del token di 40 ms. Il tempo di possesso del token è di 10 ms. Qual è la massima efficienza raggiungibile dell'anello?
 42. Si pensi di dover costruire un supercomputer interconnesso utilizzando l'approccio HIPPI e una tecnologia moderna. Lo spazio per i dati è di 64 bit e può essere inviata una parola ogni 10 ns. Quale sarà la banda passante nel canale?
 43. Nel testo viene detto che un satellite con due canali di salita e uno di discesa ALOHA a slot può raggiungere un utilizzo sul canale di discesa dello 0,736, dato uno spazio infinito di buffer. Si mostri come si ottiene questo risultato.

Capitolo 5

IL LIVELLO RETE

Il livello rete (network) si occupa di trasmettere pacchetti dalla sorgente alla destinazione. Per raggiungere la destinazione può essere necessario attraversare lungo il percorso diversi router intermedi. Questa funzione è chiaramente diversa da quella del livello data link, il quale ha il compito più modesto di trasportare pacchetti da un estremo all'altro di un cavo. Quindi il livello rete è quello più basso che si occupa di trasmissione punto-a-punto. Per ulteriori informazioni, si vedano Huitema (1995); Perlman (1992). Per raggiungere i suoi obiettivi, il livello rete deve conoscere qualcosa sulla topologia della rete di comunicazione (cioè l'insieme di tutti i router) e deve scegliere percorsi appropriati attraverso di essa. Deve inoltre fare attenzione a scegliere i percorsi in modo tale da non sovraccaricare parte delle linee di comunicazione e dei router, lasciandone altri inattivi. Infine, qualora il mittente e il destinatario appartengano a reti differenti, è compito del livello rete tener conto di queste differenze e risolvere i problemi causati da esse. In questo capitolo studieremo tutte queste problematiche, illustrandole con due esempi attuali, Internet e ATM.

5.1 Caratteristiche di progetto del livello rete

Nel seguito introdurremo alcune delle problematiche che devono essere tenute in considerazione dal progettista del livello rete. Fra queste problematiche, ricordiamo i servizi offerti al livello trasporto e la progettazione interna della rete.

5.1.1 Servizi offerti al livello trasporto

I servizi offerti dal livello rete al livello trasporto sono collocati nell'interfaccia rete/trasporto. Questa interfaccia è spesso molto importante per un altro motivo: in molti casi costituisce l'interfaccia tra il fornitore e il cliente, in altre parole il confine della rete. Spesso il fornitore mantiene il controllo sui protocolli e sulle interfacce fino al livello rete incluso. Il suo compito è consegnare pacchetti ricevuti dai suoi clienti. Per questo motivo, l'interfaccia deve essere definita particolarmente bene.

I servizi del livello rete sono stati progettati con i seguenti obiettivi.

1. I servizi dovrebbero essere indipendenti dalla tecnologia della rete.
2. Il livello trasporto dovrebbe essere indipendente dal numero, dal tipo e dalla topologia delle reti presenti.

3. Gli indirizzi di rete disponibili al livello trasporto dovrebbero utilizzare uno schema di numerazione uniforme, anche attraverso LAN e WAN diverse.

Dati questi obiettivi, i progettisti del livello rete hanno molta libertà nello scrivere le specifiche dettagliate dei servizi offerti al livello trasporto. Questa libertà spesso degenera in una battaglia furiosa tra due opposte fazioni. La discussione è incentrata sulla questione di fornire un servizio orientato alla connessione (connection-oriented) oppure senza connessione (connectionless).

Una fazione (rappresentata dalla comunità Internet) sostiene che il compito di una rete è quello di trasportare bit e nient'altro. Nella loro visione (basata su circa 30 anni di esperienza effettiva con una rete di computer reale e funzionante), le reti sono inerentemente inaffidabili, non importa come siano state progettate. Quindi, gli host dovrebbero accettare questo fatto ed eseguire loro stessi controlli di errore (cioè ricerca e correzione degli errori) e controllo di flusso.

Questo punto di vista porta rapidamente alla conclusione che un servizio di rete dovrebbe essere senza connessioni, con primitive SEND PACKET, RECEIVE PACKET e poco altro. In particolare, non si dovrebbe eseguire nessuna operazione di ordinamento pacchetti o controllo di flusso, in quanto verranno eseguite in ogni caso dagli host e probabilmente si guadagna ben poco a farlo due volte. Inoltre, ogni pacchetto deve contenere l'indirizzo di destinazione completo, in quanto ogni pacchetto viene trasportato indipendentemente dai suoi predecessori.

L'altra fazione (rappresentata dalle società telefoniche) sostiene che le reti dovrebbero fornire un servizio orientato alla connessione (ragionevolmente) affidabile e afferma che 100 anni di esperienza pieni di successi con il sistema telefonico mondiale sono una buona guida. In questa visione, le connessioni dovrebbero avere le seguenti proprietà:

1. Prima di spedire dati, un processo del livello rete nel lato mittente dovrebbe stabilire una connessione con il suo pari nel lato ricevente. Questa connessione, alla quale viene dato un identificatore speciale, viene quindi utilizzata finché tutti i dati non sono stati spediti; a questo punto viene rilasciata esplicitamente.
2. Quando viene creata una connessione, i due processi possono iniziare una negoziazione sui parametri, sulla qualità e sui costi del servizio che deve essere fornito.
3. La comunicazione avviene in entrambe le direzioni, e i pacchetti vengono consegnati in sequenza.
4. Il controllo di flusso viene eseguito automaticamente per evitare che un mittente veloce inserisca pacchetti nel canale di comunicazione più velocemente di quanto il ricevente possa estrarne, causando un sovraccarico.

Altri servizi, come ad esempio il recapito garantito, la conferma esplicita dell'avvenuto recapito e la possibilità di spedire pacchetti ad alta priorità, sono opzionali. Come sottolineato nel capitolo 1, un servizio senza connessione è simile al sistema postale, mentre un servizio orientato alla connessione è simile al sistema telefonico.

5.1 Caratteristiche di progetto del livello rete

Il dibattito tra servizi orientati alla connessione e servizi senza connessione è in realtà collegato al problema di dove collocare la complessità. Nei servizi orientati alla connessione, la complessità è collocata nel livello rete; nei servizi senza connessione è posta nel livello trasporto (host). I sostenitori dei servizi senza connessione affermano che la potenza di calcolo a livello utente è diventata economica e quindi non c'è motivo di non collocare la complessità negli host. Inoltre sostengono che una rete è un investimento (inter) nazionale di primaria importanza che durerà per decenni; per questo motivo, non dovrebbe essere piena zeppa di caratteristiche che potrebbero diventare obsolete in fretta, ma che dovranno essere calcolate nel costo strutturale per molti anni. Inoltre alcune applicazioni, come la voce digitalizzata e la raccolta di dati in tempo reale, possono ritenere la consegna *rapida* molto più importante della consegna *accurata*.

Al contrario, i sostenitori dei servizi orientati alla connessione affermano che molti utenti non desiderano eseguire nelle loro macchine complessi protocolli del livello trasporto. Quello che vogliono è un servizio affidabile ed esente da problemi, e questo servizio può essere fornito più agevolmente attraverso connessioni nel livello rete. Inoltre alcuni servizi, come ad esempio audio e video in tempo reale possono essere ottenuti più facilmente da una rete orientata alla connessione che da una rete senza connessione.

Sebbene si sia discusso raramente in questi termini, in questo caso le problematiche coinvolte sono due. Primo, la rete può essere orientata alla connessione (creazione delle connessioni obbligatoria) oppure senza connessione (nessun obbligo di creare connessioni). Secondo, la rete può essere affidabile (nessun pacchetto perso, duplicato o modificato) o inaffidabile (i pacchetti possono essere persi, duplicati o modificati). In teoria, esistono tutte e quattro le combinazioni, ma quelle dominanti sono i servizi affidabili e orientati alla connessione, e i servizi inaffidabili e senza connessione. Le altre due tendono a perdere nel rumore.

Queste due fazioni sono rappresentate dai nostri due esempi ricorrenti. Internet ha un livello rete senza connessione, mentre le reti ATM hanno un livello rete orientato alla connessione. Una domanda nasce spontanea: come funziona Internet quando viene eseguita sopra una rete basata su ATM? La risposta è la seguente: l'host sorgente stabilisce innanzitutto una connessione nel livello rete ATM con l'host destinazione e quindi spedisce pacchetti IP indipendenti attraverso di essa, come mostrato in figura 5-1. Sebbene questo approccio funzioni, è inefficiente, poiché certe funzionalità sono presenti in entrambi i livelli. Ad esempio, il livello rete ATM garantisce che i pacchetti siano sempre consegnati in ordine, mentre il protocollo TCP contiene l'intero meccanismo per gestire e riordinare pacchetti fuori ordine. Per ulteriori informazioni su come implementare IP sopra ATM, si veda l'RFC 1557, e Armitage, Adams (1995).

5.1.2 Organizzazione interna del livello rete

Dopo aver introdotto le due classi di servizi che il livello rete può fornire ai propri utenti, è tempo di vedere come funziona internamente. Esistono fondamentalmente due filosofie di organizzazione della rete, una che utilizza le connessioni e l'altra che funziona senza connessione. Nel contesto delle operazioni *interne* della rete, una connessione viene normalmente chiamata *circuito virtuale*, in analogia con i circuiti fisici stabiliti dal sistema

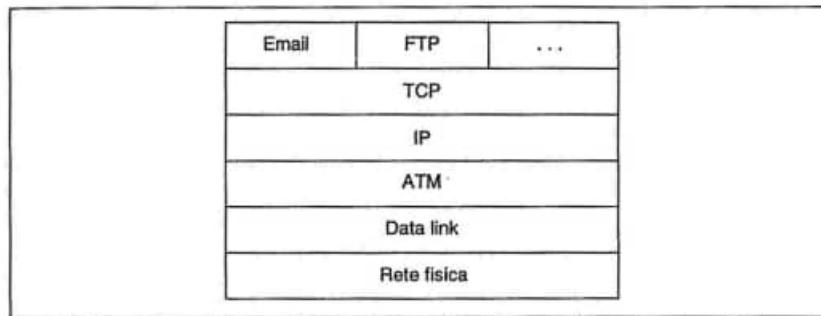


Fig. 5-1 Esecuzione di TCP/IP sopra una rete ATM.

telefonico. I pacchetti indipendenti dell'organizzazione senza connessione vengono chiamati **datagram**, in analogia con i telegrammi. I circuiti virtuali sono utilizzati generalmente nelle reti il cui servizio principale è orientato alla connessione e quindi li descriveremo in quel contesto. L'idea che sta dietro i circuiti virtuali consiste nell'evitare di dover scegliere un nuovo percorso per ogni pacchetto o cella spediti. Invece, quando viene stabilita una connessione, il procedimento di creazione prevede che venga scelto e memorizzato un percorso dalla sorgente alla destinazione. Questo percorso viene utilizzato per tutto il traffico che scorre lungo la connessione, esattamente nello stesso modo in cui funziona il sistema telefonico. Quando la connessione viene rilasciata, anche il circuito virtuale viene chiuso.

In contrasto, in una rete basata su datagram non viene calcolato anticipatamente nessun percorso, anche nel caso in cui il servizio sia orientato alla connessione. Pacchetti successivi possono seguire percorsi differenti. Sebbene le reti basate su datagram debbano lavorare di più, sono però generalmente più robuste e si adattano ai guasti e alle congestioni più facilmente delle reti basate su circuito virtuale. Discuteremo i pro e i contro dei due approcci in seguito.

Se i pacchetti che circolano lungo un dato circuito virtuale utilizzano sempre lo stesso percorso attraverso la rete, ogni router deve ricordare dove inoltrare i pacchetti di ogni circuito virtuale aperto passante attraverso di esso, mantenendo una tabella con una registrazione per ogni circuito virtuale che lo attraversa. Il preamble di ogni pacchetto che viaggia lungo la rete deve contenere un campo di identificazione del circuito virtuale, in aggiunta ai numeri di sequenza, alla checksum e dati simili. Quando un pacchetto arriva a un router, quest'ultimo può identificare la linea da cui è arrivato e il numero di circuito virtuale. Basandosi solo su questa informazione, il pacchetto deve essere inoltrato sulla linea di output corretta.

Quando viene stabilita una connessione di rete, viene scelto come identificatore un numero di circuito virtuale non ancora utilizzato in quella macchina. Poiché ogni macchina sceglie indipendentemente gli identificatori dei circuiti virtuali, questi numeri hanno solo significato locale. Se avessero significato globale, sarebbe possibile

5.1 Caratteristiche di progetto del livello rete

che due circuiti virtuali recanti lo stesso numero attraversassero lo stesso router intermedio, causando ambiguità.

Poiché i circuiti virtuali possono essere aperti da entrambi i lati, un problema si presenta nel caso in cui le procedure di creazione si stiano propagando in entrambe le direzioni contemporaneamente lungo una catena di router. A un certo istante giungono a router adiacenti. Ogni router deve scegliere un numero di circuito virtuale per il circuito (full-duplex) che si sta cercando di stabilire. Se sono stati programmati per scegliere il numero più piccolo non ancora utilizzato lungo il canale sceglieranno lo stesso numero, ottenendo che due circuiti virtuali non correlati sulla stessa linea fisica abbiano lo stesso numero. Quando più tardi arriva un pacchetto, il router ricevente non ha modo di stabilire se è un pacchetto da inoltrare in un circuito o un pacchetto da rispedire indietro sull'altro. Se i circuiti sono di tipo simplex, non ci sono ambiguità.

Si noti che ogni processo è vincolato ad avvisare quando smette di utilizzare un circuito virtuale, in modo che esso possa essere rimosso dalle tabelle dei router per recuperare spazio. Nelle reti pubbliche, la motivazione è fornita più dal bastone che dalla carota: gli utenti devono pagare immancabilmente per il tempo di connessione così come per i dati trasferiti. Inoltre, è necessario prendere provvedimenti contro gli host che terminano i propri circuiti virtuali interrompendoli all'improvviso piuttosto che chiuderli educatamente dopo l'uso.

Questo per quanto riguarda l'utilizzazione interna dei circuiti virtuali. L'altra possibilità è quella di utilizzare internamente i datagram, nel qual caso i router non mantengono una tabella con una registrazione per ogni circuito virtuale aperto. Invece, hanno una tabella che indica quale linea di uscita utilizzare per ogni possibile router di destinazione. Queste tabelle sono necessarie anche nel caso in cui vengano utilizzati i circuiti virtuali, per determinare il percorso di un pacchetto di creazione/connessione.

Ogni datagram deve contenere l'indirizzo completo della destinazione. Per reti di grandi dimensioni, questi indirizzi possono essere abbastanza lunghi (ad es., una dozzina di byte o più). Quando arriva un pacchetto, il router cerca quale linea di uscita utilizzare e spedisce il pacchetto. Inoltre, la creazione e il rilascio di connessioni nei livelli rete o trasporto non richiedono particolare impegno da parte dei router.

5.1.3 Confronto fra reti basate su circuito virtuale e reti basate su datagram

Sia le reti basate su circuito virtuale che le reti basate su datagram hanno i propri sostenitori e detrattori. Cercheremo ora di riassumere le argomentazioni di entrambe le parti. Le caratteristiche principali sono riassunte in figura 5-2, sebbene i puristi potrebbero trovare un esempio opposto per ogni elemento della figura.

All'interno della rete esistono molti compromessi tra i circuiti virtuali e i datagram. Un compromesso riguarda lo spazio di memoria dei router e la larghezza di banda. I circuiti virtuali permettono ai pacchetti di contenere un identificatore di circuito al posto di un indirizzo di destinazione completo. Se i pacchetti tendono a essere abbastanza brevi, un indirizzo completo in ogni pacchetto può rappresentare una quantità significativa di sovraccarico, e quindi di banda sprecata. Il prezzo da pagare per utilizzare internamente i circuiti virtuali è lo spazio occupato dalle tabelle all'interno

Caratteristica	Reti basate su datagrammi	Reti basate su circuito virtuale
Creazione circuito	Non richiesto	Richiesto
Indirizzamento	Ogni pacchetto contiene gli indirizzi sorgente e destinazione completi	Ogni pacchetto contiene un piccolo numero VC
Informazioni di stato	La sottorete non conserva informazioni di stato	Ogni circuito virtuale richiede spazio di tabella nella sottorete
Instradamento	Ogni pacchetto è instradato indipendentemente	Percorso scelto alla creazione del circuito virtuale; tutti i pacchetti seguono questo percorso
Effetti dei guasti nei router	Nessuno, a parte i pacchetti persi durante il guasto	Tutti i circuiti virtuali che passano attraverso il router guasto vengono terminali
Controllo di congestione	Complesso	Semplice se può essere allocato spazio sufficiente in anticipo per ogni circuito virtuale

Fig. 5-2 Confronto fra reti basate su circuito virtuale e reti basate su datagrammi.

dei router. A seconda del rapporto fra i costi dei circuiti di comunicazione e i costi della memoria nei router, l'una o l'altra soluzione può essere la più economica.

Un altro compromesso riguarda il tempo di creazione di una connessione e il tempo di scansione dell'indirizzo. Utilizzare i circuiti virtuali richiede una fase di creazione del circuito, che consuma tempo e risorse. Ciò nonostante, capire cosa fare di un pacchetto dati in una rete basata su circuito virtuale è facile: il router usa il numero di circuito per indicizzare una tabella al fine di scoprire dove mandare il pacchetto. In una rete basata su datagram, è richiesta invece una procedura più complicata per determinare dove mandare il pacchetto.

I circuiti virtuali sono in parte avvantaggiati quando cercano di evitare congestioni all'interno di una rete, in quanto le risorse possono essere riservate in anticipo al momento di creare una connessione. Una volta che i pacchetti iniziano ad arrivare, la banda necessaria e la capacità dei router sono sicuramente presenti. Con una rete basata su datagram evitare le congestioni è più difficile.

Per sistemi di elaborazione di transazioni (ad es., negozi che richiedono di verificare acquisti tramite carta di credito), il carico aggiuntivo richiesto per creare e terminare un circuito virtuale può facilmente scoraggiarne l'uso. Se si prevede che la maggior parte del traffico sia di questo tipo, l'uso di circuiti virtuali commutati all'interno delle reti ha poco senso. D'altra parte, in questo caso possono essere utili i circuiti virtuali permanenti, che sono creati manualmente e durano mesi o anni.

I circuiti virtuali presentano inoltre un problema di vulnerabilità. Se un router si guasta e

perde la sua memoria, anche se riparte un secondo più tardi tutti i circuiti virtuali passanti attraverso di esso devono essere abortiti. Al contrario, se si guasta un router basato su datagram, ne soffriranno solo gli utenti i cui pacchetti erano in coda nel router al momento del guasto, e forse nemmeno tutti, a seconda se erano già stati confermati oppure no. La perdita di una linea di comunicazione è fatale per i circuiti virtuali che la utilizzano, mentre può essere facilmente compensata nel caso vengano usati i datagram. Realizzare connessioni utilizzando datagram ha senso anche quando la rete sta cercando di fornire un servizio fortemente robusto.

La quarta possibilità, un servizio senza connessione sopra una rete basata su circuito virtuale, sembra strana ma è certamente possibile. L'esempio più ovvio è eseguire IP su una rete ATM. In questo caso si desidera eseguire un protocollo preesistente senza connessione sopra un nuovo livello di rete orientato alla connessione. Come si è detto in precedenza, questa è una soluzione ad hoc di un problema piuttosto che una buona progettazione. In un nuovo sistema progettato per funzionare sopra una rete ATM, normalmente non si dovrebbe collocare un protocollo senza connessione come IP sopra un livello rete orientato alla connessione come ATM e quindi collocare un ulteriore protocollo di trasporto orientato alla connessione sopra il protocollo senza connessione. Esempi dei quattro casi sono mostrati in figura 5-3.

5.2 Algoritmi di routing

La funzione principale del livello di rete è quella di instradare pacchetti dall'elaboratore sorgente all'elaboratore destinazione. In gran parte delle reti, sarà necessario più di un salto affinché i pacchetti completino il percorso. L'unica eccezione notevole sono le reti broadcast, ma anche in questo caso il routing è un problema se la sorgente e la destinazione non appartengono alla stessa rete. Gli algoritmi che scelgono i percorsi e le loro strutture dati rappresentano un'area importante della progettazione del livello rete.

Livello soprastante	Tipo di sottorete	
	Datagram	Circuito virtuale
Senza connessioni	UDP sopra IP	UDP sopra IP sopra ATM
	TCP sopra IP	ATM AAL1 sopra ATM

Fig. 5-3 Esempi di combinazioni differenti di servizi e strutture di rete.

Un algoritmo di routing è quella parte del software del livello rete che ha la responsabilità di decidere su quale linea di output trasmettere un pacchetto in arrivo. Se la rete utilizza internamente i datagram, questa decisione deve essere eseguita nuovamente per ogni pacchetto di dati, in quanto il percorso migliore può essere cambiato dall'ultima volta. Se

la rete utilizza internamente i circuiti virtuali, le decisioni di routing vengono prese unicamente all'atto di stabilire un nuovo circuito virtuale, dopo di che i pacchetti di dati seguono il percorso stabilito in precedenza. Il secondo caso è qualche volta chiamato **routing di sessione**, in quanto un percorso rimane in uso per un'intera sessione utente (ad es., una sessione di login in un terminale o un trasferimento file).

Sia che i percorsi vengano scelti indipendentemente per ogni pacchetto, sia che vengano scelti quando viene creata una nuova connessione, esistono alcune proprietà che sono desiderabili in un algoritmo di routing: correttezza, semplicità, robustezza, stabilità, imparzialità e ottimalità. Correttezza e semplicità difficilmente richiedono commenti, mentre il bisogno di robustezza può essere meno evidente di quanto sembri. Quando viene realizzata una rete importante, ci si può attendere che continui a funzionare per anni senza guasti globali. Durante questo periodo ci saranno guasti hardware e software di ogni tipo. Host, router e canali si guasteranno e riprenderanno a funzionare ripetutamente, e la topologia cambierà diverse volte. L'algoritmo di routing dovrebbe essere in grado di affrontare cambiamenti nella topologia e nel traffico senza richiedere di abortire tutti i processi in tutti gli host e di far ripartire la rete tutte le volte che qualche router si guasta. La stabilità è un altro obiettivo importante per un algoritmo di routing. Esistono algoritmi di routing che non convergono mai a un equilibrio, per quanto a lungo possano essere eseguiti. Imparzialità e ottimalità possono suonare scontati – sicuramente nessuno si opporrebbe a essi – ma come risulta, sono spesso obiettivi contraddittori. Come semplice esempio di questo conflitto, si osservi la figura 5-4. Si supponga che ci sia traffico sufficiente tra A e A', tra B e B' e tra C e C' per saturare i link orizzontali. Per massimizzare il flusso totale, il traffico da X a X' dovrebbe essere tagliato fuori completamente. Purtroppo, X e X' possono non essere d'accordo. Evidentemente, è necessario qualche compromesso tra efficienza globale e imparzialità verso connessioni individuali.

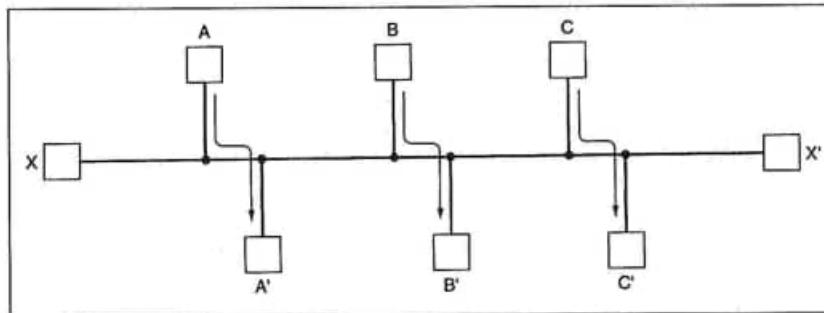


Fig. 5-4 Conflitto tra imparzialità e ottimalità.

Prima di tentare di trovare un compromesso tra imparzialità e ottimalità, dobbiamo decidere cosa vogliamo ottimizzare. La minimizzazione del ritardo medio dei pacchetti è un candidato scontato, ma lo è anche la massimizzazione del volume di dati trasmessi dall'intera rete. Inoltre, questi due goal sono in conflitto, in quanto gestire qualsiasi sistema

a coda al limite della capacità implica lunghi ritardi nella formazione delle code. Come compromesso, molte reti cercano di minimizzare il numero di salti che un pacchetto deve attraversare, in quanto ridurre il numero di salti tende a far diminuire i ritardi e quindi a ridurre la quantità di banda consumata, che porta a migliorare anche la capacità totale della rete.

Gli algoritmi di routing possono essere raggruppati in due classi principali: non adattivi e adattivi. Gli **algoritmi non adattivi** non basano le loro decisioni di routing su misurazioni o stime del traffico corrente e della topologia. Al contrario, la scelta del percorso da usare per andare da *I* a *J* (per tutti gli *I* e *J*) è calcolata in anticipo, off-line, e copiata nei router quando la rete viene fatta partire. Questa procedura viene a volte chiamata **routing statico**.

Gli **algoritmi adattivi**, invece, modificano le loro decisioni di routing a seconda dei cambiamenti della topologia e normalmente anche del traffico. È possibile distinguere gli algoritmi adattivi a seconda di dove prendono le informazioni (ad. es. localmente, da router adiacenti o da tutti i router), di quando cambiano i percorsi (ad. es. ogni *T* secondi, quando il carico cambia oppure quando cambia la topologia) e di metrica utilizzata nell'ottimizzazione (ad es. distanza, numero di salti, oppure tempo atteso di transito). Nei paragrafi seguenti vedremo un gran numero di algoritmi di routing, sia statici che dinamici.

5.2.1 Il principio di ottimalità

Prima di iniziare a vedere algoritmi specifici, può essere d'aiuto notare che è possibile fare un'affermazione generale sui percorsi ottimali, indipendentemente dalla topologia o dal traffico della rete. L'affermazione prende il nome di **principio di ottimalità** e afferma che se un router *J* è sul cammino ottimale dal router *I* al router *K*, allora anche il cammino ottimale tra *J* e *K* cade lungo lo stesso percorso. Per dimostrarlo, chiamiamo r_1 la parte del percorso da *I* a *J* e il resto del percorso r_2 . Se esiste un percorso da *J* a *K* migliore di r_2 , può essere concatenato a r_1 per migliorare il percorso da *I* a *K*; questo contraddice la nostra affermazione che $r_1 r_2$ sia ottimale.

Come diretta conseguenza del principio di ottimalità, possiamo vedere che l'insieme dei cammini ottimali da tutte le sorgenti a una data destinazione forma un albero con radice nella destinazione. Questo albero viene chiamato **sink tree** (albero a pozzo) ed è illustrato in figura 5-5 (la metrica utilizzata è data dal numero di salti). Si noti che un sink tree non è necessariamente unico; possono esistere altri alberi con le stesse lunghezze dei cammini. L'obiettivo di tutti gli algoritmi di routing è quello di scoprire e utilizzare i sink tree di tutti i router.

Poiché un sink tree è comunque un albero e non contiene nessun ciclo, ogni pacchetto verrà consegnato entro un numero finito e limitato di salti. In pratica, la realtà non è così semplice. I canali e i router possono guastarsi e tornare a lavorare durante il funzionamento e quindi router differenti possono avere idee diverse sulla topologia corrente della rete. Inoltre, abbiamo silenziosamente introdotto la questione se i router debbano acquisire individualmente le informazioni sulle quali basare la computazione del loro sink tree, oppure se queste informazioni possono essere raccolte in qualche altro modo. Torneremo presto su questo problema. Ciò nonostante, il principio di ottimalità e i sink tree forniscono un confronto in base al quale misurare gli altri algoritmi di routing.

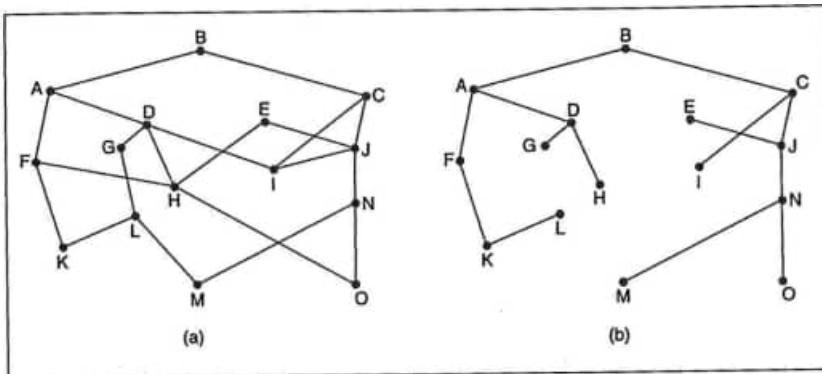


Fig. 5-5 (a) Una rete. (b) Un sink tree per il router B.

Nei paragrafi seguenti vedremo tre differenti algoritmi di routing statico. Dopo di che passeremo agli algoritmi adattivi.

5.2.2 Routing lungo il cammino minimo

Iniziamo il nostro studio degli algoritmi di routing con una tecnica che è largamente utilizzata in molte forme differenti, in quanto è semplice e facile da capire. L'idea è quella di costruire un grafo della rete, dove ogni nodo del grafo rappresenta un router e ogni arco del grafo rappresenta una linea di comunicazione (spesso chiamata *canale*). Per scegliere un percorso tra due router, l'algoritmo cerca nel grafo il cammino più breve tra di essi.

Il concetto di **cammino minimo** merita alcune spiegazioni. Un modo per misurare la lunghezza dei cammini è il numero degli salti. Usando questa metrica, i cammini ABC e ABE in figura 5-6 sono equivalenti. Un'altra metrica è la distanza geografica in chilometri, nel qual caso ABC è chiaramente molto più lungo di ABE (assumendo che la figura sia disegnata in scala).

Tuttavia, oltre al numero di salti e alla distanza fisica possono essere utilizzate molte altre metriche. Ad esempio, ogni arco può essere etichettato con il ritardo medio di formazione della coda e trasmissione per qualche pacchetto di test spedito periodicamente ogni ora. Con questo etichettamento del grafo, il cammino minimo risultante è quello più veloce, piuttosto che quello con meno archi o chilometri.

Nel caso più generale, le etichette sugli archi potrebbero essere calcolate come una funzione della distanza, della larghezza di banda, del traffico medio, del costo di comunicazione, della lunghezza media delle code, dei ritardi misurati e di altri fattori. Modificando la funzione dei pesi, l'algoritmo di routing calcola il cammino "minimo" misurato secondo qualsiasi criterio o combinazione di criteri.

Sono noti molti algoritmi per calcolare il cammino minimo tra due nodi; quello che vediamo è dovuto a Dijkstra (1959). Ogni nodo è etichettato (in parentesi) con la sua distanza dal nodo sorgente lungo il miglior cammino conosciuto. Inizialmente non si

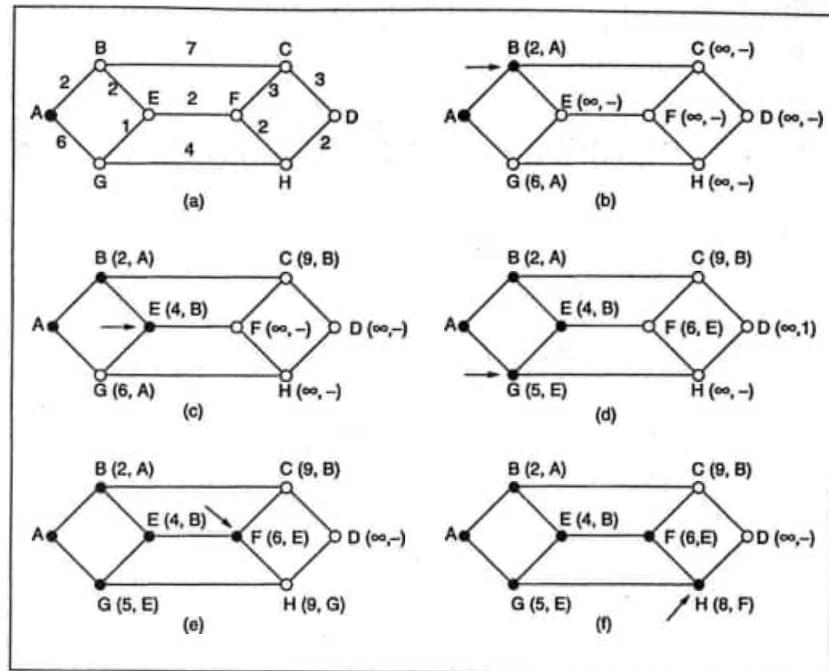


Fig. 5-6 I primi cinque passi utilizzati nella computazione del cammino minimo tra A e D. Le frecce indicano il nodo su cui si sta lavorando.

conosce nessun cammino e quindi tutti i nodi sono etichettati con infinito. Mentre l'algoritmo procede e vengono trovati nuovi cammini, le etichette possono cambiare riflettendo cammini migliori. Un'etichetta può essere o un tentativo o permanente. Inizialmente, tutte le etichette sono tentativi. Quando si scopre che un'etichetta rappresenta il cammino più breve possibile dalla sorgente a quel nodo, viene resa permanente e non viene più cambiata.

Per illustrare come lavora l'algoritmo di etichettamento, si osservi il grado indiretto e pesato di figura 5-6 (a), dove i pesi rappresentano, ad esempio, le distanze. Vogliamo trovare il cammino più breve da A a D. Iniziamo marcando il nodo A come permanente, indicato da un punto nero. Quindi esaminiamo, a turno, ogni nodo adiacente ad A (il nodo attivo), etichettando ognuno con la sua distanza da A. Ogni volta che un nodo è rietichettato, all'etichetta si aggiunge anche il nodo a partire dal quale è stata fatta l'indagine, in modo da poter ricostruire il cammino finale più tardi. Dopo aver esaminato ogni nodo adiacente ad A, esaminiamo tutti i nodi etichettati come tentativi nell'intero grafo e rendiamo permanente quello con l'etichetta più piccola, come mostrato in figura 5-6 (b). Questo diventa il nuovo nodo attivo.

A questo punto partiamo da B ed esaminiamo tutti i nodi adiacenti ad esso. Se la somma dell'etichetta di B e della distanza di B dal nodo che si sta considerando è minore dell'etichetta associata al nodo, il cammino ispezionato è più breve e quindi il nodo viene rietichettato.

Dopo aver ispezionato tutti i nodi adiacenti al nodo attivo e possibilmente dopo aver cambiato le etichette di tentativo, si cerca nell'intero grafo il nodo con il valore minore. Questo nodo è reso permanente e diventa il nodo attivo per il prossimo round. La figura 5-6 mostra i primi 5 passi dell'algoritmo.

Per capire perché l'algoritmo funziona, si osservi figura 5-6 (c), in cui E è stato appena reso permanente. Si supponga che esista un cammino più corto di ABE , diciamo $AXYZE$. Ci sono due possibilità: o il nodo Z è già stato reso permanente, oppure no. Se lo è stato, allora E è già stato ispezionato (nel round successivo a quello in cui Z è stato reso permanente) e quindi il cammino $AXYZE$ non è sfuggito alla nostra attenzione.

Ora si consideri il caso in cui Z sia ancora etichettato come tentativo. O l'etichetta di Z è più grande o uguale a quella di E , nel qual caso $AXYZE$ non può essere un cammino più breve di ABE , oppure è minore di quella di E , nel qual caso Z e non E diventerà permanente per primo, consentendo a E di essere indagato da Z .

L'algoritmo completo si trova in figura 5-7. L'unica differenza tra il programma e l'algoritmo appena descritto è che in figura 5-7 calcoliamo il cammino minimo a partire dal nodo terminale, t , piuttosto che dal nodo sorgente, s . Poiché il cammino minimo da t a s in un grafo indiretto è uguale al cammino minimo da s a t , non ha importanza da dove si comincia (a meno che non vi siano diversi cammini minimi, nel qual caso invertire la ricerca può portare a scoprirne uno diverso). La ragione per cui si cerca all'indietro è che ogni nodo è etichettato con il suo predecessore piuttosto che con il successore. Quando si copia il cammino finale nella variabile di output, $path$, il cammino è quindi invertito. Invertendo la ricerca, i due effetti si cancellano, e la risposta è prodotta nell'ordine corretto.

5.2.3 Flooding

Un altro algoritmo statico è **flooding** (inondazione), nel quale ogni pacchetto in arrivo viene inoltrato su ogni linea di uscita eccetto quella da cui è arrivato. Questo algoritmo genera ovviamente un vasto numero di pacchetti duplicati; in effetti, un numero infinito, a meno di non prendere qualche misura per fermare il processo. Una di queste misure è quella di avere un contatore di salti all'interno di ogni pacchetto, che viene decrementato a ogni salto, scartando il pacchetto quando il contatore raggiunge lo zero. Idealmente, il contatore di salto dovrebbe essere inizializzato alla lunghezza del cammino dalla sorgente alla destinazione. Se il mittente non conosce la lunghezza del cammino, può inizializzare il contatore con il caso peggiore: in pratica, il diametro della rete.

Una tecnica alternativa per arginare l'inondazione consiste nel tener traccia di quali pacchetti sono stati già inoltrati, in modo da evitare di rispedirli una seconda volta. È possibile raggiungere questo obiettivo richiedendo che il router sorgente inserisca un numero di sequenza in ogni pacchetto che riceve dai suoi host. Ogni router ha bisogno quindi di un elenco per ogni router sorgente, contenente i numeri di sequenza originati da quella sorgente, che sono già stati visti. Se un pacchetto in arrivo è nella lista, non viene inoltrato.

```

#define MAX_NODES 1024           /* maximum number of nodes */
#define INFINITY 1000000000      /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES]; /* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{
    struct state {                /* the path being worked on */
        int predecessor;          /* previous node */
        int length;                /* length from source to this node */
        enum {permanent, tentative} label; /* label state */
    } state[MAX_NODES];

    int i, k, min;
    struct state *p;
    for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
        p->predecessor = -1;
        p->length = INFINITY;
        p->label = tentative;
    }
    state[t].length = 0; state[t].label = permanent;
    k = t;                                /* k is the initial working node */
    do {
        for (i = 0; i < n; i++)             /* Is there a better path from k? */
            if (dist[k][i] != 0 && state[i].label == tentative) { /* this graph has n nodes */
                if (state[k].length + dist[k][i] < state[i].length) {
                    state[i].predecessor = k;
                    state[i].length = state[k].length + dist[k][i];
                }
            }
        /* Find the tentatively labeled node with the smallest label. */
        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);

    /* Copy the path into the output array. */
    i = 0; k = s;
    do {path[i++] = k; k = state[k].predecessor;} while (k >= 0);
}

```

Fig. 5-7 Algoritmo di Dijkstra per calcolare il cammino minimo in un grafo.

Per evitare che gli elenchi crescano senza limitazioni, bisogna aggiungere a ogni elenco un contatore k , con il significato che tutti i numeri di sequenza fino a k sono già stati visti. Quando arriva un pacchetto, è semplice verificare se il pacchetto è un duplicato (nel qual caso viene scartato). Inoltre, l'elenco precedente a k non è necessario, in quanto k lo riassume completamente.

Una variazione di flooding leggermente più pratica è il **flooding selettivo**. In questo algoritmo i router non spediscono ogni pacchetto in arrivo in ogni linea di uscita, ma solo su quelle linee che vanno approssimativamente nella giusta direzione. Ha poco senso spedire un pacchetto diretto a ovest in una linea diretta a est, a meno che la topologia sia estremamente particolare.

Il flooding non è pratico in gran parte delle applicazioni, ma possiede alcune specifiche utilizzazioni. Per esempio, nelle applicazioni militari, dove un gran numero di router può essere distrutto in un istante, l'estrema robustezza del flooding è auspicabile. In applicazioni di database distribuiti, in qualche caso è necessario aggiornare tutti i database correntemente, nel qual caso il flooding è utile. Un terzo possibile utilizzo di flooding è quello di metrica di confronto per gli altri algoritmi di routing. Il flooding sceglie sempre il cammino più breve, in quanto sceglie ogni possibile cammino in parallelo. Conseguentemente, nessun altro algoritmo può produrre un ritardo minore (se ignoriamo il carico aggiuntivo generato dal processo di flooding stesso).

5.2.4 Routing basato su flusso

Gli algoritmi studiati fino a ora prendono in considerazione soltanto la topologia e non considerano il carico. Se, ad esempio, c'è sempre una grossa quantità di traffico da A a B in figura 5-6, allora può essere meglio instradare il traffico da A a C attraverso $AGEFC$, anche se questo cammino è molto più lungo di ABC . In questo paragrafo studieremo un algoritmo statico di routing che utilizza sia la topologia che il carico, chiamato **flow-based routing** (routing basato su flusso).

In alcune reti, il flusso medio di dati attraverso ogni coppia di nodi è relativamente stabile e predicibile. Ad esempio, nella rete privata di una catena di negozi al dettaglio, ogni negozio può spedire ordini, rapporti di vendita, aggiornamenti di inventario e altri tipi di messaggi ben definiti, a siti noti e in una sequenza predefinita, in modo che il volume totale di traffico cambi molto poco da un giorno all'altro. Alla condizione che il traffico medio tra i e j sia conosciuto in anticipo e che sia, sotto una ragionevole approssimazione, costante nel tempo, è possibile analizzare il flusso matematicamente per ottimizzare il routing.

L'idea fondamentale dietro questa analisi è che per una data linea, se la capacità e il flusso medio sono noti, è possibile calcolare il ritardo medio di un pacchetto usando la teoria delle code. Partendo dal ritardo medio di tutte le linee è banale calcolare una media pesata sul flusso, per ottenere il ritardo medio dei pacchetti per l'intera rete.

Per utilizzare questa tecnica, alcune informazioni devono essere conosciute in anticipo. Innanzitutto la topologia della rete deve essere nota. Secondo, deve essere conosciuta la matrice di traffico F_{ij} . Terzo, deve essere disponibile la matrice C_{ij} della capacità delle linee, che specifica la capacità di ogni linea in bps. Infine, deve essere scelto un algoritmo di routing (possibilmente di prova).

Come esempio di questo metodo, si consideri la rete full duplex di figura 5-8 (a). I pesi sugli archi rappresentano le capacità C_{ij} in entrambe le direzioni, misurate in kbps. La matrice di figura 5-8 (b) conserva una registrazione per ogni coppia sorgente-destinazione. La registrazione per la sorgente i e la destinazione j mostra la strada da seguire per il traffico $i-j$, e anche il numero di pacchetti al secondo che devono essere spediti dalla sorgente i alla destinazione j . Ad esempio, 3 pacchetti/s vanno da B a D , e utilizzano il percorso BFD per arrivarci. Si noti che è già stato applicato qualche algoritmo di routing per ottenere i percorsi mostrati nella matrice.

Date queste informazioni, è banale calcolare il totale sulla linea i , λ_i . Ad esempio, il traffico $B-D$ contribuisce per 3 pacchetti/s nella linea BF e per 3 pacchetti/s anche nella linea FD . Allo stesso modo, il traffico $A-D$ contribuisce per 1 pacchetto/s per ognuna delle tre linee. Il traffico totale per ogni linea diretta a est è indicato nella colonna λ_i di figura 5-9. In questo esempio, tutto il traffico è simmetrico, cioè il traffico XY è uguale al traffico, per ogni X e Y . Nelle reti reali questa condizione non è sempre valida. La figura indica anche il numero medio di pacchetti al secondo in ogni linea μC_i assumendo che la dimensione media di un pacchetto sia $1/\mu = 800$ bit.

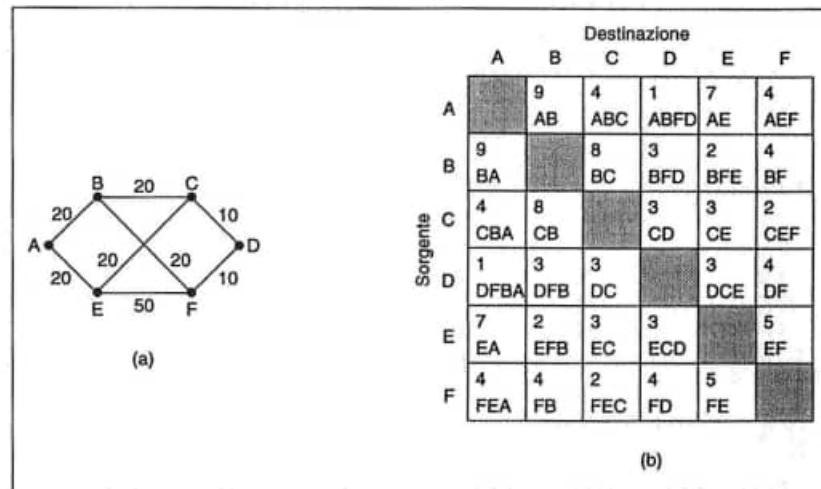


Fig. 5-8 (a) Una rete con le capacità delle linee in kbps. (b) Il traffico in pacchetti al secondo e la matrice di routing.

La penultima colonna di figura 5-9 riporta il ritardo medio di ogni linea ottenuto dalla formula della teoria delle code

$$T = \frac{1}{\mu C - \lambda}$$

dove $1/\mu$ è la dimensione media di un pacchetto in bit, C è la capacità in bps e λ è il flusso medio in pacchetti al secondo. Ad esempio, con una capacità $\mu C = 25$ pacchetti/s e un flusso reale di $\lambda = 14$ pacchetti/s, il ritardo medio è 91 ms. Si noti che con $\lambda = 0$, il ritardo medio è ancora di 40 ms, in quanto la capacità è di 25 pacchetti/s. In altre parole, il "ritardo" include sia il tempo di coda che di servizio.

Per calcolare il ritardo medio per l'intera rete, prendiamo la somma pesata di ognuna delle otto linee, utilizzando come pesi la frazione di traffico totale che utilizza quella linea. In questo esempio, la media è 86 ms.

Per valutare un algoritmo di routing differente, possiamo ripetere l'intero processo utilizzando flussi diversi per ottenere un nuovo ritardo medio. Se ci limitiamo solo ad algoritmi basati su singolo percorso, come abbiamo fatto fino a ora, esiste solo un numero finito di modi per instradare pacchetti da ogni sorgente a ogni destinazione. È sempre possibile scrivere un programma che li provi tutti uno dopo l'altro, e trovi quello con il più piccolo ritardo medio. Poiché questo calcolo può essere fatto anticipatamente e fuori linea, il fatto che possa consumare tempo non è necessariamente un problema serio. Questo è quindi il miglior algoritmo di routing. Bertsekas e Gallager (1992) discutono nei dettagli il routing basato su flusso.

i	Linea	λ_i (pacc./s)	C_i (kbps)	μC_i (pacc./s)	C_i (ms)	Peso
1	AB	14	20	25	91	0,171
2	BC	12	20	25	77	0,146
3	CD	6	10	12.5	154	0,073
4	AE	11	20	25	71	0,134
5	EF	13	50	62.5	20	0,159
6	FD	8	10	12.5	222	0,098
7	BF	10	20	25	67	0,122
8	EC	8	20	25	59	0,098

Fig. 5-9 Analisi della rete di figura 5-8 utilizzando una dimensione media dei pacchetti di 800 bit. Il traffico inverso (BA, CB ecc.) è uguale al traffico diretto.

5.2.5 Routing basato su vettori di distanza

Le reti di calcolatori odierne utilizzano generalmente algoritmi di routing dinamici al posto di quelli statici descritti in precedenza. Due algoritmi dinamici in particolare, basati rispettivamente su vettori di distanza e sullo stato dei canali, sono i più popolari. In questo paragrafo vedremo il primo algoritmo. Nel seguente studieremo il secondo.

Gli algoritmi di routing basati su vettori di distanza (**distance vector routing**) funzionano facendo in modo che ogni router mantenga una tabella (cioè un vettore) contenente la migliore distanza conosciuta per ogni destinazione e quale canale utilizzare per raggiungerla. Queste tabelle sono aggiornate scambiando informazioni con i vicini.

In alcune occasioni, l'algoritmo di distance vector routing viene chiamato con altri nomi, come "algoritmo di routing distribuito Bellman-Ford" o "algoritmo di Ford-Fulkerson", dal nome dei ricercatori che l'hanno sviluppato (Bellman, 1957; Ford, Fulkerson, 1962). Era l'algoritmo di routing originale di ARPANET ed è stato usato anche in Internet sotto il nome di RIP e in versioni primitive di DECnet e IPX di Novell. I router AppleTalk e Cisco utilizzano protocolli migliorati basati su vettori di distanza.

Nel distance vector routing, ogni router mantiene una tabella di routing indicizzata da tutti i router nella rete, e contenente una registrazione per ognuno di essi. Questa registrazione è suddivisa in due parti: la linea di uscita prescelta da utilizzare per quella destinazione, e una stima del tempo o della distanza per quella destinazione. La metrica usata può essere il numero di salti, il ritardo in millisecondi, il numero totale di pacchetti in coda lungo un cammino o qualcosa di simile.

Si assume che il router conosca la "distanza" di tutti i suoi vicini. Se la metrica è data dai salti, la distanza è un solo salto. Se la metrica è la lunghezza delle code, il router esamina ogni coda. Se la metrica è il ritardo, il router può misurarlo direttamente con pacchetti speciali ECHO su cui il ricevente mette un timbro temporale e li rispedisce indietro più velocemente possibile.

Ad esempio, si assuma di utilizzare il ritardo come metrica e che il router conosca il ritardo per ognuno dei suoi vicini. Una volta ogni T ms ogni router spedisce ai suoi vicini una lista dei ritardi stimati per ogni destinazione, e riceve inoltre una lista simile da tutti i vicini. Si immagini che una di queste tabelle arrivi dal vicino X , dove X_i è la stima di X di quanto tempo è necessario per raggiungere il router i . Se il router sa che il ritardo per X è di m ms, sa anche di poter raggiungere il router i attraverso X in $X_i + m$ ms. Eseguendo questo calcolo per ogni vicino, un router può scoprire quale stima sembra la migliore, e utilizzare questa stima e la linea corrispondente nella sua nuova tabella di routing. Si noti che la vecchia tabella di routing non viene utilizzata nel calcolo.

Questo processo di aggiornamento è illustrato in figura 5-10. La parte (a) mostra una rete. Le prima quattro colonne della parte (b) mostrano i vettori di ritardo ricevuti dai vicini del router J . A afferma di avere un ritardo di 12 ms da B , un ritardo di 25 ms da C , un ritardo di 40 ms da D ecc. Si supponga che J abbia misurato o stimato il suo ritardo dai suoi vicini A , I , H e K rispettivamente come 8, 10, 12 e 6 ms.

Si osservi come J calcola il suo nuovo itinerario verso il router G . Sa di poter raggiungere A in 8 ms, e A afferma di essere in grado di raggiungere G in 18 ms. Allo stesso modo calcola il ritardo verso G attraverso I , H e K , ottenendo rispettivamente 41 (31+10), 18 (6+12) e 37 (31+6) ms. Il migliore valore fra questi è 18, quindi registra nella sua tabella di routing che il ritardo per G è 18 ms e che l'itinerario da utilizzare passa attraverso H . Lo stesso calcolo viene effettuato per tutte le altre destinazioni, ottenendo la nuova tabella di routing presente nell'ultima colonna della figura.

Il problema del conteggio all'infinito

Il distance vector routing funziona in teoria, ma in pratica ha seri inconvenienti: sebbene converga alla risposta corretta, lo può fare molto lentamente. In particolare, reagisce rapidamente alle buone notizie e placidamente alle cattive. Si consideri un router il cui miglior itinerario per la destinazione X sia molto lento. Se al prossimo scambio il vicino

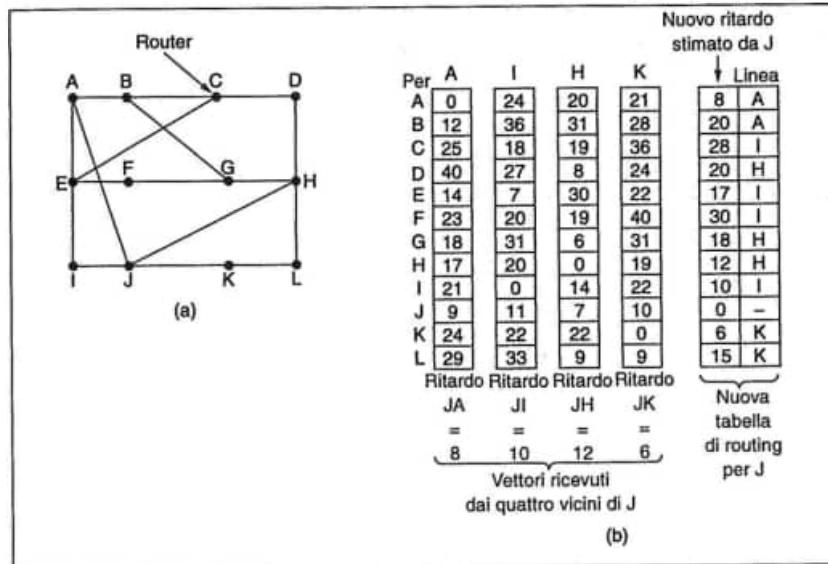


Figura 5-11 (a) Una rete. (b) Input da A, I, H, K e la nuova tabella di routing per J.

A riporta all'improvviso un ritardo breve per X, il router passa a utilizzare la linea A per spedire il traffico a X. Le buone notizie vengono elaborate in uno scambio di vettori. Per vedere a quale velocità vengono propagate le buone notizie, si osservi la rete lineare a cinque nodi di figura 5-11, dove la metrica utilizzata è il numero di salti. Si supponga che A sia inizialmente guasto e che tutti gli altri router lo sappiano. In altre parole, tutti registrano che il ritardo per A è infinito.

A	B	C	D	E	A	B	C	D	E
∞	∞	∞	∞	∞	Inizialmente	1	2	3	4
1	∞	∞	∞	∞	Dopo 1 scambio	3	2	3	4
1	2	∞	∞	∞	Dopo 2 scambi	3	4	3	4
1	2	3	∞	∞	Dopo 3 scambi	5	4	5	4
1	2	3	4	∞	Dopo 4 scambi	5	6	5	6
						7	6	7	6
						7	8	7	8
						⋮			
						∞	∞	∞	∞

Figura 5-11 Il problema del conteggio all'infinito.

5.2 Algoritmi di routing

Quando A viene riparato, gli altri router lo vengono a sapere attraverso lo scambio dei vettori. Per semplicità assumeremo che ci sia da qualche parte un gigantesco gong che viene suonato periodicamente per far iniziare uno scambio di vettori contemporaneamente da tutti i router. Al momento del primo scambio, B viene a sapere che il suo vicino di sinistra ha un ritardo 0 per A. B a questo punto registra nella tabella di routing che A dista un salto verso sinistra. Tutti gli altri router pensano che A sia ancora guasto. A questo punto, la tabella di routing per A è uguale alla seconda riga di figura 5-11 (a). Allo scambio successivo, C viene a sapere che B ha un cammino di lunghezza 1 per A e così aggiorna la sua tabella di routing per indicare un cammino di lunghezza 2, mentre D ed E verranno raggiunti dalla buona notizia solo più tardi. Chiaramente, la buona notizia si diffonde al ritmo di un salto per scambio di vettori. In una rete il cui cammino più lungo è di N salti, tutti i router verranno a sapere entro N scambi dei canali e dei router tornati a funzionare. A questo punto si consideri la situazione di figura 5-11 (b), in cui tutte le linee e i router sono inizialmente attivi. I router B, C, D ed E hanno una distanza da A rispettivamente di 1, 2, 3 e 4. All'improvviso A si guasta, oppure viene tagliata la linea tra A e B, che dal punto di vista di B è la stessa cosa.

Al primo scambio di pacchetti, B non riceve nulla da A. Fortunatamente, C dice "Non ti preoccupare. Io ho un cammino da A di lunghezza 2". Niente fa sospettare a B che il cammino di C passi attraverso B. Per quanto ne sa B, C può avere dieci linee uscenti tutte con cammini indipendenti per A di lunghezza 2. Come risultato, B sa a questo punto di poter raggiungere A attraverso C, con un cammino di lunghezza 3. Al primo scambio, D ed E non aggiornano le loro registrazioni per A.

Al secondo scambio, C nota che entrambi i suoi vicini affermano di avere un cammino per A di lunghezza 3. Ne sceglie uno a caso e porta a 4 la nuova distanza da A, come mostrato nella terza riga di figura 5-11 (b). Scambi successivi producono il resto della sequenza mostrata nel resto di figura 5-11 (b).

Da questa figura, dovrebbe essere chiaro perché le cattive notizie viaggiano piano: nessun router ha un valore più alto di 1 del minimo di tutti i suoi vicini. Gradualmente, tutti i router lavorano per raggiungere l'infinito, ma il numero di scambi dipende dal valore utilizzato al posto di infinito. Per questa ragione, è saggio utilizzare una valle corrispondente al cammino più lungo più 1. Se la metrica è il ritardo di tempo, non esiste nessun limite superiore ben definito e così è necessario un valore alto per evitare che un cammino con un lungo ritardo sia trattato come guasto. Non ci sorprende che questo problema sia conosciuto come problema del conteggio all'infinito.

L'algoritmo split horizon

In letteratura sono state proposte molte soluzioni ad hoc per il problema del conteggio all'infinito, ognuna delle quali è più complicata e meno utile di quelle che la precedevano. Ne descriveremo solo una e mostreremo perché non funziona. L'algoritmo dello **split horizon** (orizzonte spezzato) funziona nello stesso modo del distance vector routing, a parte il fatto che la distanza per X non viene riportata sulla linea sulla quale vengono spediti i pacchetti per X (in realtà viene riportato infinito). Nello stato iniziale di figura 5-11 (b), ad esempio, C dice a D la verità per quanto riguarda la distanza per A, mentre C dice a B che la distanza per A è infinito. In modo simile, D dice la verità a E e mente a C.

A questo punto vediamo cosa succede quando *A* si guasta. Nel primo scambio, *B* scopre che la sua linea diretta è inutilizzabile, e anche *C* riporta una distanza infinita da *A*. Poiché nessuno dei vicini può raggiungere *A*, anche *B* mette a infinito la sua distanza da *A*. Nel prossimo scambio, *C* viene a sapere che *A* è irraggiungibile da entrambi i suoi vicini e mette a infinito la sua distanza da *A*. Utilizzando split horizon, le cattive notizie si propagano di un salto per scambio di vettori. Questa velocità è più alta di quanto non accada senza split horizon.

La cattiva notizia è che lo split horizon, sebbene largamente utilizzato, qualche volta non funziona. Si consideri, per esempio, la rete di quattro nodi di figura 5-12. Inizialmente, sia *A* che *B* hanno una distanza 2 da *D*, mentre *C* ha una distanza 1.

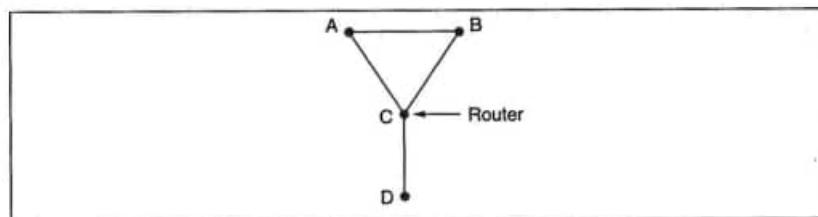


Fig. 5-12 Un esempio dove split horizon fallisce.

A questo punto supponiamo che la linea *CD* si guasti. Utilizzando lo split horizon, sia *A* che *B* dicono a *C* di non poter raggiungere *D*. Quindi *C* conclude immediatamente che *D* è irraggiungibile e informa di questo fatto sia *A* che *B*. Purtroppo *A* viene a sapere che *B* ha un cammino di lunghezza 2 per *D*, e così assume di poter raggiungere *D* in 3 salti passando per *B*. In modo simile, *B* conclude di poter raggiungere *D* in 3 salti passando per *A*. Al prossimo scambio, ognuno di essi registra che la loro distanza da *D* è 4. Entrambi raggiungono gradualmente l'infinito, precisamente il comportamento che stavamo cercando di evitare.

5.2.6 Routing basato sullo stato dei canali

Il distance vector routing è stato usato in ARPANET fino al 1979, quando fu sostituito dal routing basato sullo stato dei canali. Due furono i problemi principali che causarono il suo decesso. Primo, poiché la metrica di ritardo era la lunghezza delle code, non teneva in considerazione la capacità delle linee al momento di scegliere i percorsi. Inizialmente, tutte le linee trasportavano 56 kbps e la capacità non era un problema; ma quando alcune linee furono portate a 230 kbps e altre a 1,544 Mbps, non tenere in considerazione la banda fu un grosso problema. Naturalmente, sarebbe stato possibile cambiare la metrica di ritardo per tener conto della larghezza di banda delle linee, ma esisteva un secondo problema: l'algoritmo spesso impiegava troppo tempo per convergere, anche con trucchi come split horizon. Per queste ragioni, fu rimpiazzato da un algoritmo completamente nuovo chiamato routing basato sullo stato dei canali (**link state routing**). Oggi vengono utilizzate molte varianti del link state routing.

L'idea dietro al link state routing è semplice e può essere enunciata in cinque parti. Ogni router deve:

1. Scoprire i propri vicini e i loro indirizzi di rete.
2. Misurare il ritardo o il costo per ognuno dei suoi vicini.
3. Costruire un pacchetto contenente tutto quello che ha appena scoperto.
4. Spedire questo pacchetto a tutti i router.
5. Calcolare il cammino minimo per ogni altro router.

In effetti, la topologia completa e tutti i ritardi vengono misurati sperimentalmente e distribuiti a ogni router. Quindi l'algoritmo di Dijkstra può essere utilizzato per trovare il cammino minimo verso ogni altro router. Nel seguito descriveremo ognuno dei cinque passi più dettagliatamente.

Indagare sui vicini

Quando un router viene fatto partire, il suo primo compito è capire quali sono i suoi vicini. Porta a termine questo compito spedendo un pacchetto speciale HELLO su ogni linea punto-a-punto. Il router all'altro capo deve spedire una risposta dicendo chi è. Questi nomi devono essere globalmente unici, poiché quando un router lontano viene a sapere che tre router sono connessi a *F*, è essenziale che possa determinare se tutti e tre intendono lo stesso *F* oppure no.

Quando due o più router vengono connessi a una LAN, la situazione è leggermente più complicata. La figura 5-13 (a) illustra una LAN alla quale sono direttamente connessi tre router *A*, *C* ed *F*. Ognuno di questi router è connesso a uno o più router, come mostrato.

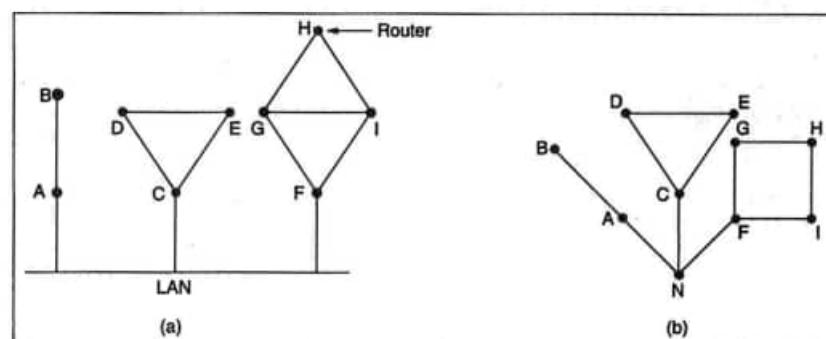


Fig. 5-13 (a) Nove router in una LAN. (b) Un grafo che modella (a).

Un modo per modellare la LAN è considerare essa stessa un nodo, come illustrato in figura 5-13 (b), dove è stato introdotto un nodo artificiale *N* al quale sono connessi *A*, *C*

e F. Il fatto che sia possibile andare da A a C attraverso la LAN è quindi rappresentato dal cammino ANC.

Misurare il costo dei canali

L'algoritmo di link state routing richiede che ogni router conosca il ritardo verso ognuno dei suoi vicini, o almeno ne abbia una stima ragionevole. Il metodo più diretto per misurare il ritardo è quello di spedire un pacchetto speciale ECHO su tutte le linee; all'altro lato si richiede di rispondere immediatamente. Misurando il tempo di andata e ritorno e dividendo per 2, il router mittente può ottenere una stima ragionevole del ritardo. Per ottenere risultati migliori è possibile ripetere il test molte volte e utilizzare la media.

Un problema interessante è se tenere in considerazione il carico mentre si misura il ritardo oppure no. Per considerare anche il carico, la misurazione del tempo di andata e ritorno deve iniziare quando il pacchetto ECHO viene messo in coda. Per ignorare il carico, la misurazione deve iniziare quando il pacchetto ECHO raggiunge il fronte della coda.

Vi sono valide argomentazioni a favore di entrambi gli approcci. Includere nella misurazione i ritardi indotti dal traffico significa che quando un router può scegliere tra due linee con la stessa banda, una delle quali è pesantemente utilizzata tutto il tempo e l'altra no, sceglierà il percorso attraverso la linea meno carica come cammino più breve. Questa scelta produrrà prestazioni migliori.

Purtroppo, esiste anche un'argomentazione contro l'inclusione del traffico nel calcolo del ritardo. Si consideri la rete di figura 5-14, che è suddivisa in due parti, Est e Ovest, connesse da due linee, *CF* ed *EI*. Si supponga che gran parte del traffico tra Est e Ovest utilizzi la linea *CF*, e come risultato la linea sia pesantemente sovraccarica con lunghi ritardi. L'inclusione dei ritardi di coda nel calcolo del cammino minimo renderà *EI* più attraente. Dopo aver calcolato le nuove tabelle di routing, gran parte del traffico Est-Ovest utilizzerà ora la linea *EI*, sovraccaricandola. Conseguentemente, nel prossimo aggiornamento *CF* sembrerà essere il cammino più breve. Come risultato, le tabelle di routing possono oscillare selvaggiamente, portando a un routing incostante e a molti problemi potenziali. Se il carico viene ignorato e viene considerata solo la larghezza di banda, questo problema non si presenta. Alternativamente, il carico può essere suddiviso su entrambe le linee, ma questa soluzione non utilizza completamente il cammino migliore.

Costruire pacchetti link state

Una volta raccolta l'informazione necessaria per lo scambio, il passo successivo richiede che ogni router costruisca un pacchetto contenente tutti i dati. Il pacchetto inizia con l'identità del mittente, seguito da un numero di sequenza, da un'età (che verrà descritta in seguito), e da una lista di vicini. Per ogni vicino viene fornito il ritardo per raggiungerlo. Una rete di esempio è quella di figura 5-15 (a), dove i ritardi sono mostrati sulle linee. I pacchetti link state corrispondenti a ognuno dei 6 router sono mostrati in figura 5-15 (b).

Costruire i pacchetti link state è facile. La parte difficile è stabilire quando costruirli. Una possibilità è di farlo periodicamente, cioè a intervalli regolari. Un'altra possibilità è di costruirli quando avviene qualche evento significativo, come una linea o un vicino che si

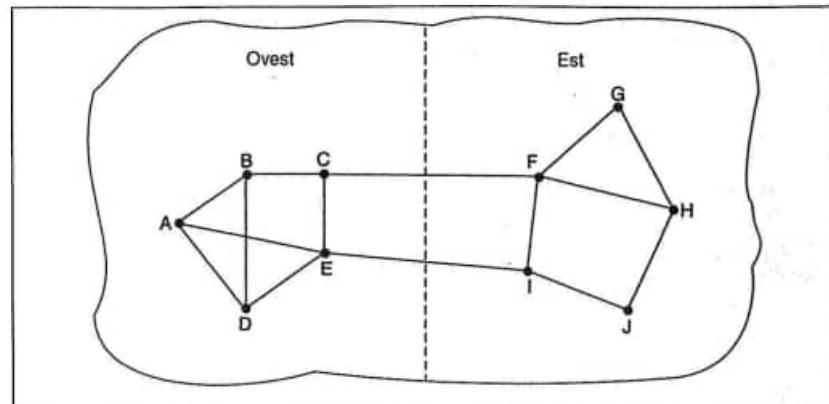


Fig. 5-14 Una rete nella quale le parti Est e Ovest sono connesse da due linee.

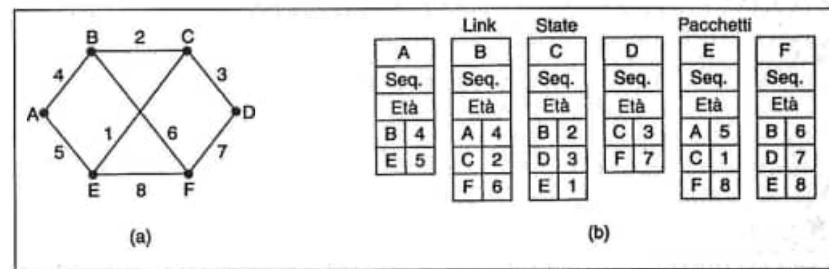


Fig. 5-15 (a) Una rete. (b) I pacchetti link state per la rete (a).

guastano oppure vengono riparati, oppure cambiano apprezzabilmente le loro caratteristiche.

Distribuzione dei pacchetti link state

La parte più difficile dell'algoritmo è distribuire i pacchetti link state affidabilmente. Non appena i pacchetti vengono distribuiti e installati, i router che li ricevono per primi cambieranno i loro itinerari. Conseguentemente, router differenti possono utilizzare versioni differenti della topologia, e ciò può portare a inconsistenze, cicli, nodi irraggiungibili e altri problemi.

Innanzitutto, descriveremo l'algoritmo principale di distribuzione. Più tardi vedremo alcuni raffinamenti. L'idea fondamentale è di utilizzare il flooding per distribuire i pacchetti link state. Per tenere sotto controllo l'inondazione, ogni pacchetto contiene un numero di sequenza che viene incrementato per ogni nuovo pacchetto spedito. I router tengono traccia di tutte le coppie (router sorgente, numero di sequenza) che vedono. Quando arriva un nuovo pacchetto link state, si controlla la lista dei pacchetti già visti. Se è nuovo, viene

inoltrato su tutte le linee eccetto quella da cui è stato ricevuto. Se è un duplicato, viene scartato. Se arriva un pacchetto con un numero di sequenza minore del più alto visto finora, viene scartato in quanto obsoleto.

Questo algoritmo ha qualche problema, ma è maneggevole. Innanzitutto, se i numeri di sequenza venissero riutilizzati, regnerebbe la confusione. La soluzione è di utilizzare numeri di sequenza a 32 bit. Spedendo un pacchetto link state ogni secondo, ci vorrebbero 137 anni per tornare a utilizzare gli stessi numeri e quindi questa possibilità può essere ignorata.

Secondo, se un router si rompesse, perderebbe traccia del suo numero di sequenza. Se partisse di nuovo da 0, il prossimo pacchetto potrebbe essere scartato come duplicato.

Terzo, se un numero di sequenza venisse modificato a 65.540 venisse ricevuto al posto di 4 (un errore di 1 bit), i pacchetti da 5 a 65.540 verrebbero scartati come obsoleti, in quanto si riterrebbe 65.540 il numero di sequenza corretto.

La soluzione a tutti questi problemi è di includere l'età del pacchetto dopo il numero di sequenza e di decrementarla una volta al secondo. Quando l'età raggiunge lo 0, il pacchetto viene scartato. Supponiamo che un nuovo pacchetto arrivi normalmente ogni 10 min; le informazioni di routing perderebbero la loro validità solamente quando un router si guasta (oppure sono stati persi 6 pacchetti link state consecutivi, un evento improbabile). Il campo età viene decrementato da ogni router anche durante il processo iniziale di flooding, per essere sicuri che nessun pacchetto venga perso e sopravviva per un periodo indefinito di tempo (un pacchetto la cui età è 0 viene scartato).

Alcune migliorie a questo algoritmo possono renderlo più robusto. Quando un pacchetto link state arriva a un router per il flooding, non viene messo nella coda di trasmissione immediatamente. Viene invece tenuto in un'area di appoggio per attendere prima un istante. Se arriva un altro pacchetto link state dalla stessa sorgente prima che il precedente sia trasmesso, i loro numeri di sequenza vengono confrontati. Se sono uguali, il duplicato viene scartato. Se sono differenti, viene scartato quello più vecchio. Per prevenire contro gli errori lungo le linee router-router, viene spedita una conferma per tutti i pacchetti link state. Quando una linea diventa inutilizzata, l'area di appoggio viene scandita in ordine round robin per scegliere un pacchetto o una conferma da spedire.

La struttura dati utilizzata dal router *B* per la rete mostrata in figura 5-15 (a) è illustrata in figura 5-16. Ogni riga corrisponde a un pacchetto link state arrivato recentemente, ma non ancora completamente elaborato. La tabella memorizza l'origine del pacchetto, il suo numero di sequenza e l'età e infine i dati. Sono inoltre presenti flag di spedizione e di conferma per ognuna delle tre linee di *B* (per *A*, *C* ed *F*, rispettivamente). I flag di spedizione indicano che il pacchetto deve essere spedito sulla linea indicata. I flag di conferma indicano che deve essere spedita una conferma.

In figura 5-16, il pacchetto link state da *A* arriva direttamente e quindi deve essere spedito a *C* ed *F* e confermato ad *A*, come indicato dai bit di flag. In modo simile, il pacchetto da *F* deve essere inoltrato ad *A* e *C* e confermato a *F*.

Tuttavia, la situazione del terzo pacchetto, arrivato da *E*, è differente. È arrivato due volte, una volta attraverso *EAB* e una volta attraverso *EFB*. Conseguentemente, deve essere spedito solo a *C*, ma confermato sia ad *A* che a *F*, come indicato dai bit.

Se arriva un duplicato mentre l'originale è ancora nel buffer, i bit devono essere cambiati.

Ad esempio, se una copia dello stato di *C* arriva da *F* prima che la quarta registrazione nella tabella sia inoltrata, i 6 bit devono essere cambiati a 100011 per indicare che il pacchetto deve essere confermato a *F* e non rispedito.

Sorgente	Seq.	Età	Flag di spedizione			Flag di conferma			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	

Fig. 5-16 Il buffer dei pacchetti per il router *B* in figura 5-15.

Il calcolo dei nuovi percorsi

Una volta che un router ha accumulato un insieme completo di pacchetti link state, può ricostruire l'intero grafo della rete in quanto ha informazioni su ogni canale. In effetti, ogni canale è rappresentato due volte, una per ogni direzione. Si può calcolare una media dei due valori, oppure possono essere usati separatamente.

A questo punto l'algoritmo di Dijkstra può essere eseguito localmente per costruire il cammino minimo per tutte le possibili destinazioni. Il risultato di questo algoritmo può essere installato nelle tabelle di routing e può essere ripreso il funzionamento normale. Per una rete di n router, ognuno dei quali ha k vicini, la memoria richiesta per memorizzare i dati di input è proporzionale a kn . Per reti di grandi dimensioni, questo può essere un problema. Anche il tempo di computazione è una caratteristica da tenere in considerazione. Ciò nonostante, in molte situazioni pratiche il link state routing funziona bene.

Tuttavia, problemi hardware e software possono devastare questo algoritmo (e non solo questo). Ad esempio, se un router affermasse di avere una linea che non ha oppure dimenticasse di avere una linea, il grafo della rete sarebbe errato. Se un router non riuscisse a spedire pacchetti, o li modificasse mentre passano attraverso di esso, sorgerebbero delle difficoltà. Infine, se un router esaurisse la memoria oppure eseguisse male i calcoli di routing, ci sarebbero non pochi guai. Nel caso in cui la rete crescesse di dimensioni fino a raggiungere decine o centinaia di migliaia di nodi, la probabilità che occasionalmente qualche router si possa guastare non è trascurabile. Il trucco è cercare di limitare i danni quando accade l'inevitabile. Perlman (1988) discute dettagliatamente questi problemi e le loro soluzioni.

Poiché il link state routing viene largamente usato nelle reti reali, è opportuno spendere poche parole su alcuni esempi di protocollo che lo utilizzano. Il protocollo OSPF, che

viene sempre più usato in Internet, utilizza un algoritmo link state. Descriveremo OSPF nel paragrafo 5.5.5.

Un altro importante protocollo link-state è IS-IS (Intermediate System-Intermediate System-, Sistema Intermedio-Sistema Intermedio), che fu progettato per DECnet e in seguito adottato da ISO per utilizzarlo con il suo protocollo senza connessione del livello rete, CLNP. Da allora è stato modificato per gestire anche altri protocolli, in particolar modo IP. IS-IS è utilizzato in numerosi dorsali Internet (inclusa la vecchia dorsale NSFNET) e in alcuni sistemi cellulari digitali come CDPD. Novell Netware utilizza una variante minore di IS-IS (NLSP) per instradare i pacchetti IPX.

Fondamentalmente IS-IS distribuisce un'immagine della topologia dei router, sulla quale viene calcolato il cammino minimo. Ogni router dichiara, nelle sue informazioni link state, quali indirizzi del livello rete può raggiungere direttamente. Questi indirizzi possono essere IP, IPX, AppleTalk o qualsiasi altro. L'indirizzamento IS-IS può anche supportare contemporaneamente più protocolli di rete.

Molte delle innovazioni introdotte con IS-IS furono adottate da OSPF (OSPF fu progettato molti anni più tardi di IS-IS). Fra queste ricordiamo un metodo autostabilizzante di diffusione degli aggiornamenti link state, il concetto di router designato in una LAN e il metodo per calcolare e supportare la suddivisione dei cammini e metriche multiple. Come conseguenza, ci sono poche differenze tra IS-IS e OSPF. La differenza più importante è che IS-IS è stato progettato in modo che sia facile e naturale trasportare informazioni su protocolli multipli di rete, una caratteristica che OSPF non ha. Questo vantaggio è particolarmente apprezzabile in ambienti di grandi dimensioni e multiprotocollo.

5.2.7 Routing gerarchico

Con la crescita delle dimensioni della reti, le tabelle di routing crescono proporzionalmente. Non solo viene consumata dalle tabelle una quantità di memoria crescente, ma sono anche necessarie una quantità maggiore di tempo di CPU per esaminarle e una maggiore larghezza di banda per spedire le informazioni di routing. A un certo punto la rete può crescere oltre il punto in cui non è più possibile che ogni router abbia una registrazione per ogni altro router e quindi il routing deve essere gestito gerarchicamente, come nelle reti telefoniche.

Quando si utilizza il routing gerarchico, i router vengono suddivisi in quelle che chiamiamo **regioni**. Ogni router conosce tutti i dettagli su come instradare i pacchetti verso destinazioni all'interno della propria regione, ma non sa nulla sulla struttura interna di altre regioni. Quando reti differenti sono connesse insieme, è naturale considerarle ognuna come una regione separata, al fine di evitare che i router di una rete debbano conoscere la struttura topologica delle altre.

Per reti gigantesche, una gerarchia a due livelli può non essere sufficiente; può essere necessario raggruppare le regioni in cluster, i cluster in zone, le zone in gruppi, e così via fino a quando terminano i nomi per le aggregazioni. Come esempio di una gerarchia multilivello, si consideri come un pacchetto da Berkeley (California) potrebbe essere instradato fino a Malindi (Kenya). Il router di Berkeley dovrebbe conoscere la topologia dettagliata all'interno della California, mentre spedirebbe tutto il traffico extrastatale al router di Los Angeles. Il router di Los Angeles dovrebbe essere in grado di instradare il

traffico verso altri router nazionali, mentre spedirebbe tutto il traffico estero a New York. Il router di New York dovrebbe essere programmato in modo da dirigere tutto il traffico al router nel paese destinazione responsabile di gestire il traffico estero, ad esempio a Nairobi. Infine, il pacchetto dovrebbe trovare la sua strada lungo l'albero in Kenia fino a raggiungere Malindi.

La figura 5-17 mostra un esempio quantitativo di routing in una gerarchia a due livelli con 5 regioni. La tabella completa per il router 1A avrebbe 17 registrazioni, come mostrato in figura 5-17 (b). Quando il routing è gerarchico, come in figura 5-17 (c), sono presenti registrazioni per tutti i router locali, mentre tutte le altre regioni vengono considerate come un unico router. Ad esempio, il traffico per la regione 2 attraversa la linea 1B-2A, mentre il resto del traffico remoto attraversa la linea 1C-3B. Il routing gerarchico ha ridotto la tabella da 17 a 7 entrate. Crescendo il rapporto tra numero di regioni e numero di router, si incrementa anche lo spazio risparmiato nelle tabelle di routing.

Tabella completa per 1A			Tabella gerarchica per 1A		
Dest.	Linea	Salti	Dest.	Linea	Salti
1A	-	-	1A	-	-
1B	1B	1	1B	1B	1
1C	1C	1	1C	1C	1
2A	1B	2	2	1B	2
2B	1B	3	3	1C	2
2C	1B	3	4	1C	3
2D	1B	4	5	1C	4
3A	1C	3			
3B	1C	2			
4A	1C	3			
4B	1C	4			
4C	1C	4			
5A	1C	4			
5B	1C	5			
5C	1B	5			
5D	1C	6			
5E	1C	5			

(a)

(b)

(c)

Fig. 5-17 Routing gerarchico.

Purtroppo, questo guadagno in spazio non è gratuito. La penalità da pagare è rappresentata dalla maggior lunghezza dei cammini. Ad esempio, il percorso migliore da 1A a 5C passa per la regione 2, mentre con il routing gerarchico tutto il traffico dalla regione 5 passa per la regione 3, in quanto questa via è migliore per gran parte delle destinazioni nella regione 5.

Quando una rete singola diventa veramente grande, una questione interessante è la se-

guente: quanti livelli dovrebbe avere la gerarchia? Ad esempio, si consideri una rete con 720 router. Se non esiste gerarchia, ogni router necessita di 720 registrazioni nella tabella di routing. Se la rete è partizionata in 24 regioni di 30 router ciascuno, ogni router necessita di 30 registrazioni locali e 23 registrazioni remote, per un totale di 53 registrazioni. Se viene scelta una gerarchia a tre livelli, con 8 cluster ognuno contenente 9 regioni di 10 router, ogni router ha bisogno di 10 registrazioni per i router locali, 8 registrazioni per il routing in altre regioni entro il proprio cluster e 7 registrazioni per i cluster distanti, per un totale di 25 entrate. Kamoun e Kleinrock (1979) hanno scoperto che il numero ottimale di livelli per una rete di N router è $\ln N$, richiedendo un totale di $e \ln N$ registrazioni per router. Hanno inoltre provato che l'incremento medio della lunghezza dei campioni causato dal routing gerarchico è sufficientemente piccolo da essere generalmente accettabile.

5.2.8 Routing per host mobili

Ai giorni nostri milioni di persone hanno computer portatili e in generale vorrebbero leggere la loro posta elettronica e accedere al loro file system ovunque siano nel mondo. Questi host mobili introducono una nuova complicazione: prima di instradare un pacchetto a un host mobile, la rete deve localizzarlo. Il problema di incorporare host mobili in una rete è molto recente, ma in questo paragrafo abbozzeremo alcune delle sue caratteristiche e vedremo una possibile soluzione.

Il modello del mondo tipicamente usato dai progettisti di rete è mostrato in figura 5-18. Qui abbiamo una WAN consistente di router e host. Connesse alla WAN ci sono LAN, MAN e celle senza filo del tipo studiato nel capitolo 2.

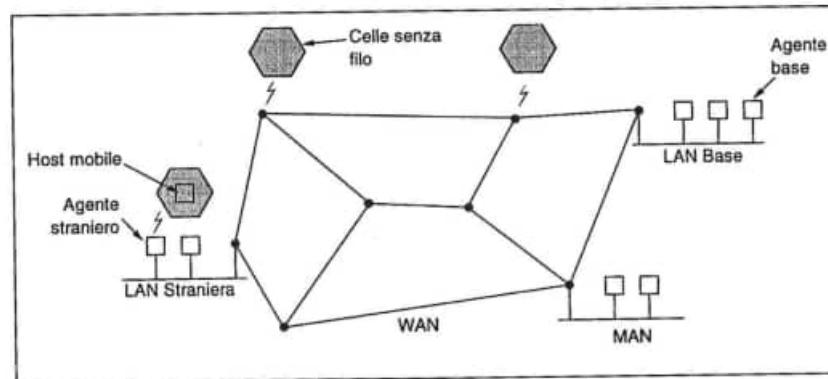


Fig. 5-18 Una WAN alla quale sono attaccate LAN, MAN e celle senza filo.

Gli utenti che non si muovono mai sono detti stazionari. Sono connessi alla rete con cavi di rame o fibre ottiche. Viceversa, possiamo distinguere due altri tipi di utente. Gli utenti "migratori" sono utenti fondamentalmente stazionari che si muovono da un sito fisso a un altro e utilizzano la rete solo quando sono fisicamente connessi a essa. Gli utenti "raminghi" lavorano in movimento e vogliono mantenere la loro connessione mentre si

muovono. Utilizzeremo il termine **utenti mobili** per indicare le ultime due categorie, cioè tutti gli utenti che sono lontani da casa.

Si assume che tutti gli utenti abbiano una **locazione base** permanente che non cambia mai e che abbiano anche un indirizzo base che possa essere utilizzato per determinare la loro locazione base, allo stesso modo in cui il numero di telefono 1-212-5551212 indica gli Stati Uniti (codice nazionale 1) e Manhattan (212). L'obiettivo che si vuole raggiungere nei sistemi con utenti mobili è rendere possibile la spedizione di pacchetti a utenti mobili usando il loro indirizzo base, e far in modo che i pacchetti li raggiungano efficientemente dovunque siano. Il trucco, naturalmente, è trovarli.

Nel modello di figura 5-18, il mondo è (geograficamente) suddiviso in piccole unità che chiameremo aree, dove un'area è tipicamente una LAN o una cella di un sistema telefonico cellulare. Ogni area possiede uno o più **agenti stranieri**, che tengono traccia di tutti gli utenti mobili che visitano l'area. Inoltre, ogni area ha un **agente base** che tiene traccia degli utenti la cui base è nell'area, ma che ne stanno al momento visitando un'altra.

Quando un nuovo utente entra in un'area, o connettendosi alla LAN o semplicemente muovendosi dentro la cella, il suo computer deve registrarsi presso l'agente straniero locale. La procedura di registrazione funziona tipicamente in questo modo:

1. Periodicamente, ogni agente straniero spedisce in broadcast un pacchetto che annuncia la sua esistenza e il suo indirizzo. Un host mobile appena arrivato aspetta uno di questi pacchetti, ma nel caso non ne arrivi nessuno abbastanza in fretta, può spedire in broadcast un pacchetto che dice "C'è qualche agente straniero qui intorno?".
2. L'host mobile si registra presso l'agente straniero, fornendo il proprio indirizzo base, il proprio indirizzo corrente del livello data link e alcune informazioni di sicurezza.
3. L'agente straniero contatta l'agente base dell'host mobile e dice: "Uno dei tuoi host è qui". Il messaggio dall'agente straniero all'agente base contiene l'indirizzo di rete dell'agente straniero. Include anche le informazioni di sicurezza, per convincere l'agente base che l'host mobile è realmente lì.
4. L'agente base esamina le informazioni di sicurezza, che contengono un timestamp per dimostrare che è stato generato pochi secondi prima. Se è soddisfatto, dice all'agente straniero di procedere.
5. Quando l'agente straniero riceve una conferma dall'agente base, crea una registrazione nelle sue tabelle e informa l'host mobile che è finalmente registrato.

Idealmente, l'uscita di un utente da un'area dovrebbe essere annunciata per permettere la deregistrazione; molti utenti invece spengono barbaramente il loro computer quando hanno finito.

Quando un pacchetto è spedito a un utente mobile, viene instradato verso la LAN base dell'utente mobile, in quanto questo è quello che si dovrebbe fare seguendo l'indirizzo, come illustrato nel passo 1 della figura 5-19. I pacchetti spediti a utenti mobili nella loro LAN sono intercettati dall'agente base. L'agente base quindi cerca la nuova locazione (temporanea) dell'utente mobile e trova l'indirizzo dell'agente straniero che gestisce

l'utente mobile. A questo punto, l'agente base esegue due operazioni. Primo, incapsula il pacchetto nel campo contenuto di un pacchetto uscente e spedisce quest'ultimo all'agente straniero (passo 2 di figura 5-19). Questo meccanismo è chiamato tunneling. Dopo aver ricevuto il pacchetto incapsulato, l'agente straniero estraе il pacchetto originale dal campo contenuto e lo spedisce all'utente mobile come un data link frame.

Secondo, l'agente base informa il mittente che d'ora in poi dovrà spedire i pacchetti destinati all'host mobile incapsulandoli nel contenuto di pacchetti esplicitamente indirizzati all'agente straniero, invece di spedirli all'indirizzo base dell'utente mobile (passo 3). Pacchetti successivi possono ora essere instradati direttamente all'utente attraverso l'agente straniero (passo 4), saltando completamente la sua locazione base.

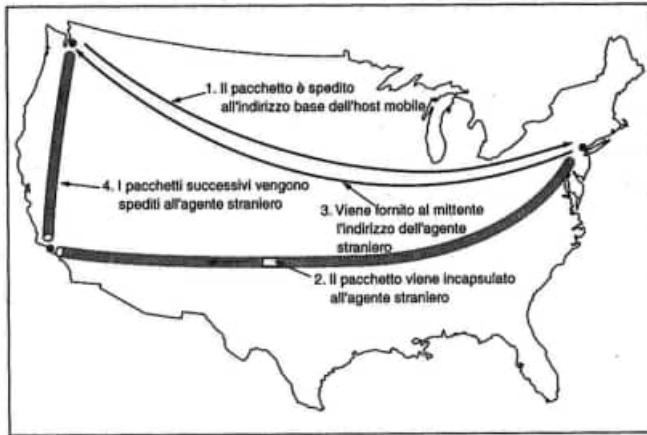


Fig. 5-19 Routing di pacchetti per utenti mobili.

I vari schemi che sono stati proposti differiscono in molti modi. Innanzitutto, ci si domanda quanta parte di questo protocollo debba essere realizzata dai router e quanta dagli host mobili, e nel secondo caso, da quale livello nell'host. Secondo, in pochi schemi i router lungo la strada memorizzano gli indirizzi mappati in modo da poter intercettare il traffico anche prima che raggiunga la locazione base. Terzo, in alcuni schemi viene dato a ogni visitatore un indirizzo temporaneo; in altri, l'indirizzo temporaneo si riferisce a un agente che gestisce il traffico per tutti i visitatori.

Quarto, gli schemi differiscono per come riescono a consegnare a una destinazione i pacchetti indirizzati a un'altra. Una possibilità è quella di cambiare l'indirizzo di destinazione e ritrasmettere il pacchetto modificato. Alternativamente, l'intero pacchetto (indirizzo base e tutto il resto) può essere incapsulato nel contenuto di un altro pacchetto spedito a indirizzo temporaneo. Infine, gli schemi differiscono negli aspetti di sicurezza. In generale, quando un host o un router riceve un messaggio della forma "A partire da ora, per favore spedire tutta la posta di Cayla a me", potrebbe farsi un paio di domande su chi sta parlando e se assecondare la richiesta è una buona idea oppure no. Molti protocolli per host mobili sono descritti e confrontati in Ioannidis, Maguire (1993); Myles, Skellern (1993); Parkins (1993); Taraoka *et al.* (1993); Wada *et al.* (1993).

5.2.9 Broadcast routing

Per alcune applicazioni, gli host hanno necessità di spedire messaggi a molti o addirittura a tutti gli altri host. Ad esempio, un servizio che distribuisce bollettini meteorologici, aggiornamenti sul mercato delle merci, o programmi radio dal vivo può funzionare meglio spedendo in broadcast a tutti gli host e lasciando che quelli che sono interessati possano leggere i dati. La spedizione di un pacchetto a tutte le destinazioni simultaneamente è chiamato **broadcasting**; sono stati proposti molti metodi per realizzarlo.

Un metodo di broadcasting che non richiede particolari caratteristiche alla rete prevede che la sorgente spedisca un pacchetto a tutte le destinazioni. Questo metodo non solo spreca enormemente la banda disponibile, ma richiede anche che la sorgente abbia una lista completa di tutte le destinazioni. In pratica questa può essere l'unica possibilità, ma è il metodo meno desiderabile.

Un altro candidato ovvio è il **flooding**. Sebbene il flooding sia poco adatto per le comunicazioni ordinarie punto-a-punto, nel caso del broadcasting ci sono serie considerazioni a suo favore, in modo particolare se nessuno degli altri metodi descritti in seguito è applicabile. Il problema del flooding come tecnica di broadcast è lo stesso che si presenta nel caso di comunicazioni punto-a-punto: genera troppi pacchetti e consuma troppa banda. Un terzo algoritmo è il **routing multidestinazione**. Se viene usato questo metodo, ogni pacchetto contiene o una lista di destinazioni, oppure un mappa di bit che indica le destinazioni desiderate. Quando un pacchetto arriva a un router, il router controlla tutte le destinazioni per determinare l'insieme delle linee di output che devono essere usate (una linea di output è necessaria se appartiene al percorso migliore per almeno una delle destinazioni). Il router genera una nuova copia del pacchetto per ogni linea di output e include in ogni pacchetto solo le destinazioni che devono usare quella linea. In effetti, l'insieme delle destinazioni viene frazionato tra le linee di output. Dopo un numero sufficiente di salti, ogni pacchetto conterrà solo una destinazione e potrà essere trattato come un pacchetto normale. Il routing multidestinazione è simile alla spedizione di pacchetti indirizzati separatamente, a parte il fatto che quando molti pacchetti devono seguire lo stesso itinerario, uno di essi paga il biglietto intero e gli altri seguono gratuitamente.

Un quarto algoritmo di broadcast rende esplicito l'uso del sink tree relativo al router che inizia il broadcast, o qualsiasi altro spanning tree conveniente. Uno **spanning tree** (albero di ricoprimento) è un sottoinsieme della rete che include tutti i router ma non contiene cicli. Se ogni router sa quale delle sue linee appartengono allo spanning tree, può copiare un pacchetto di broadcast in arrivo su tutte le linee appartenenti allo spanning tree eccetto quella di arrivo. Questo metodo utilizza molto bene la banda, generando in assoluto il numero più piccolo di pacchetti necessari a portare a termine il lavoro. L'unico problema è che ogni router deve conoscere qualche spanning tree affinché sia applicabile. Qualche volta questa informazione è disponibile (ad es. con il link state routing), ma qualche volta no (ad es. con distance vector routing).

Il nostro ultimo algoritmo di broadcast è un tentativo di approssimare il comportamento di quello precedente, anche quando i router non sanno nulla sugli spanning tree. Quando a un router arriva un pacchetto di broadcast, il router controlla se il pacchetto è arrivato nella linea normalmente utilizzata per spedire pacchetti verso la sorgente del broadcast. Se è così,

esiste una buona probabilità che lo stesso pacchetto di broadcast abbia seguito il percorso migliore dal router e sia quindi la prima copia che arriva al router. Se questo è il caso, il router inoltra copie di esso su tutte le linee eccetto quella di arrivo. Se, viceversa, il pacchetto di broadcast è arrivato da un linea diversa da quella migliore per raggiungere la sorgente, il pacchetto viene scartato come probabile duplicato.

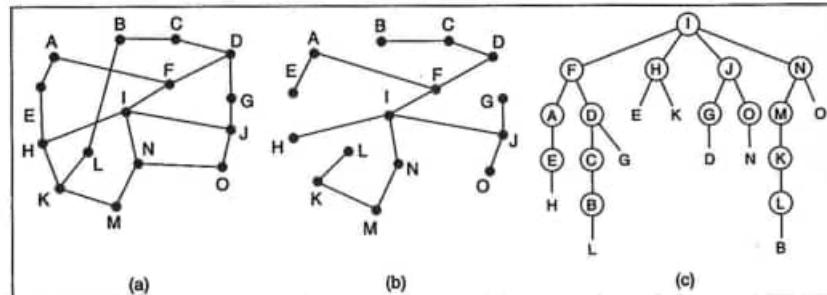


Fig. 5-20 Reverse path forwarding. (a) Una sottorete. (b) Uno spanning tree. (c) L'albero costruito tramite reverse path forwarding.

Un esempio dell'algoritmo, chiamato **reverse path forwarding** (rispedizione sul cammino inverso) è mostrato in figura 5-20. La parte (a) mostra una rete, la parte (b) mostra un sink tree per il router *I* della rete e la parte (c) mostra come funziona l'algoritmo di rispedizione sul cammino inverso. Al primo salto, *I* spedisce i pacchetti a *F*, *H*, *J* ed *N*, come indicato dalla seconda riga dell'albero. Ognuno di questi pacchetti arriva dalla via prescelta per *I* (assumendo che le vie prescelte cadano lungo il sink tree) che è quindi indicata da un cerchio attorno alla lettera. Al secondo passo, vengono generati 8 pacchetti, 2 per ogni router che ha ricevuto 1 pacchetto durante il primo salto. Come si può osservare, tutti e 8 arrivano a router precedentemente non visitati, e tutti tranne 1 arrivano lungo la linea prescelta. Dei 9 pacchetti generati nel terzo passo, solo due arrivano dal percorso prescelto, e quindi solo questi generano ulteriori pacchetti. Dopo 5 salti e 24 pacchetti, il broadcast termina, a confronto dei 4 salti e dei 14 pacchetti nel caso il sink tree sia stato seguito correttamente.

Il vantaggio principale della rispedizione sul cammino inverso consiste nell'essere ragionevolmente efficiente e facile da implementare. Non richiede né che i router conoscano gli spanning tree, né il carico aggiuntivo della lista di destinazioni o della mappa di bit in ogni pacchetto di broadcast come nell'indirizzamento multidestinazione. Non richiede nemmeno un meccanismo speciale per fermare il processo, come il flooding (o un contatore di salti in ogni pacchetto e una conoscenza a priori del diametro della rete, oppure una lista di pacchetti già visti per ogni sorgente).

5.2.10 Multicast routing

In alcune applicazioni, processi molto distanti lavorano insieme in gruppi, come per esempio un gruppo di processi che implementa un database distribuito. È spesso necessario che un processo spedisca un messaggio a tutti i membri del gruppo. Se il gruppo è pic-

colo, può spedire un messaggio punto-a-punto ad ogni altro membro. Se il gruppo è di grandi dimensioni, questa strategia può essere dispendiosa. In alcuni casi si possono utilizzare messaggi di broadcast, ma utilizzare messaggi di broadcast per informare 1000 macchine in una rete di un 1.000.000 di nodi è inefficiente in quanto la gran parte dei riceventi non è interessata al messaggio (o peggio ancora, sono senz'altro interessati ma non dovrebbero leggerlo). Quindi abbiamo bisogno di un modo per spedire messaggi a gruppi ben definiti che sono estesi numericamente, ma piccoli se confrontati con l'intera rete. Spedire un messaggio a tale gruppo viene detto **multicasting**, e il suo algoritmo di routing è chiamato **multicast routing**. In questo paragrafo descriveremo un modo per fare multicast routing. Per informazioni aggiuntive, si vedano Deering, Cheriton (1990); Deering *et al.* (1994); Rajagopalan (1992).

Se occorre effettuare comunicazioni multicast è richiesta la gestione dei gruppi. Sono necessari dei metodi per creare e distruggere gruppi, e nel caso dei processi dei metodi per unirsi ai gruppi e uscirne. Come questi compiti vengano eseguiti non riguarda il multicast routing. Ciò che conta è che quando un processo si unisce a un gruppo, informi il suo host di questo fatto. È importante che i router conoscano a quali gruppi appartengono i loro host. O gli host devono informare i loro router dei cambiamenti nell'appartenenza ai gruppi, oppure i router devono interrogare i propri host periodicamente. In entrambi i casi, i router vengono a sapere a quali gruppi appartengono i loro host. I router informano i loro vicini, in modo che le informazioni si propaghino lungo la rete.

Per realizzare il multicast routing, ogni router calcola uno spanning tree che copre tutti gli altri router nella rete. Ad esempio, in figura 5-21 (a) è presente una rete con due gruppi, 1 e 2. Alcuni router sono attaccati a host che appartengono a uno o a entrambi questi gruppi, come indicato in figura. Uno spanning tree per il router più a sinistra è indicato in figura 5-21 (b).

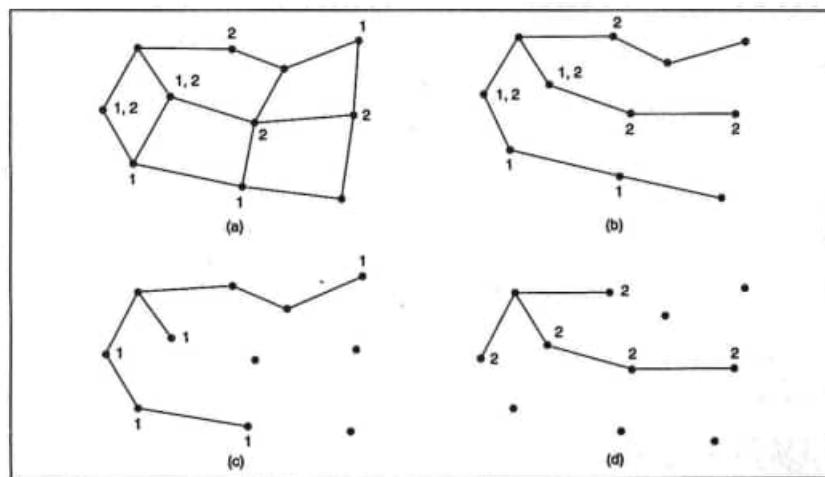


Fig. 5-21 (a) Una rete. (b) Uno spanning tree per il router più a sinistra. (c) Spanning tree potato per il gruppo 1. (d) Spanning tree potato per il gruppo 2.

Quando un processo spedisce un pacchetto a un gruppo, il primo router esamina il suo spanning tree e lo "pota", rimuovendo tutte le linee che non portano a host membri del gruppo. Nel nostro esempio, figura 5-21 (c) mostra lo spanning tree potato per il gruppo 1, mentre figura 5-21 (d) mostra lo spanning tree potato per il gruppo 2. I pacchetti di multicast sono inoltrati solo lungo gli spanning tree appropriati.

Sono possibili vari metodi per potare lo spanning tree. Il metodo più semplice può essere utilizzato se viene usato il link state routing, e ogni router conosce la topologia completa della rete, incluso a quali gruppi appartengono gli host. Quindi lo spanning tree può essere potato partendo dalla fine di ogni ramo e lavorando all'indietro verso la radice, rimuovendo tutti i router che non appartengono al gruppo in questione.

Con il distance vector routing può essere seguita una strategia di potatura diversa. L'algoritmo base è quello di reverse path forwarding. Ogni qual volta un router con nessun host interessato in un gruppo particolare e con nessuna connessione con altri router riceve un messaggio di multicast per quel gruppo, risponde con un messaggio PRUNE (potatura), dicendo al mittente di non spedire ulteriori multicast per quel gruppo. Quando un router con nessun membro del gruppo fra i propri host riceve tale messaggio da tutte le sue linee, può rispondere anch'esso con un messaggio PRUNE. In questo modo, la rete viene potata ricorsivamente. Uno svantaggio potenziale di questo algoritmo è che è poco adatto per reti di grandi dimensioni. Si supponga che una rete abbia n gruppi, ognuno dei quali con una media di m membri. Per ogni gruppo, devono essere memorizzati m spanning tree potati, per un totale di mn alberi. Qualora esistano molti gruppi di grandi dimensioni, è necessario una memoria considerevole per memorizzare tutti gli alberi.

Uno schema alternativo utilizza i **core-base tree** (alberi a nucleo base) (Ballardie et al, 1993), in cui viene calcolato un singolo spanning tree per ogni gruppo, con la radice (il nucleo) vicina al centro del gruppo. Per spedire un messaggio di multicast, un host lo spedisce al nucleo, il quale poi spedisce il multicast lungo lo spanning tree. Sebbene questo albero non sia ottimale per tutte le sorgenti, la riduzione nei costi di memorizzazione da m alberi a uno è considerevole.

5.3 Algoritmi di controllo della congestione

Quando nella rete (o in una sua parte) sono presenti troppi pacchetti, le prestazioni degradano. Questa situazione viene chiamata congestione. La figura 5-22 ne illustra i sintomi. Quando il numero di pacchetti immagazzinati dagli host nella rete è minore della sua capacità di trasporto, tutti i pacchetti vengono consegnati (a parte i pochi affetti da errori di trasmissione) e il numero dei pacchetti consegnati è proporzionale al numero di quelli spediti. Ciò nonostante, quando il traffico cresce troppo, i router non sono più in grado di sopportare il carico, e iniziano a perdere pacchetti. Questo tende a peggiorare la situazione. In caso di traffico veramente elevato, le prestazioni collassano completamente e non viene consegnato quasi nessun pacchetto. La congestione può essere causata da molti fattori. Se tre o quattro flussi di pacchetti di input iniziano ad arrivare all'improvviso e tutti richiedono la stessa linea di output, verrà creata una coda. Se non esiste memoria sufficiente per contenere tutti, alcuni pacchetti verranno persi. Aggiungere più memoria può certamente aiutare, ma Nagle (1987) ha scoperto che se i router avessero memoria infinita, la congestione sarebbe peggiore, in quanto al momento di raggiungere l'inizio della coda sarebbero già stati ritenuti persi (anche più di una volta) e sarebbero stati spediti dei duplicati. Tutti questi pacchetti verrebbero obbedientemente inoltrati al prossimo router, incrementando il carico sul cammino verso la destinazione.

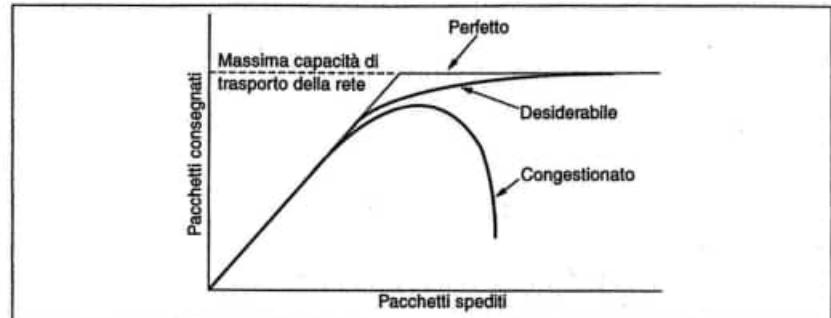


Fig. 5-22 Quando viene offerto troppo traffico alla rete, si ha congestione e le prestazioni degradano visibilmente.

Anche i processori lenti possono essere motivo di congestione. Se le CPU dei router sono lente nell'eseguire i compiti di contabilità loro richiesti (gestione dei buffer, aggiornamento delle tabelle ecc.), le code possono crescere, anche nel caso in cui ci sia un eccesso di capacità delle linee. In modo simile, anche linee a bassa banda possono causare congestioni. Aggiornare le linee senza cambiare i processori (o viceversa) può essere utile, ma spesso non fa altro che spostare il collo di bottiglia. Inoltre, aggiornare soltanto una parte di un sistema, spesso non fa altro che trasferire il collo di bottiglia in qualche altra zona. Il problema reale è un cattivo assortimento tra le diverse parti del sistema. Il problema persistrà fino a quando tutti i componenti non saranno bilanciati.

La congestione tende a nutrirsi di se stessa e a peggiorare. Se un router non ha buffer liberi, deve ignorare i pacchetti in arrivo. Quando un pacchetto viene scartato, il router mittente (un vicino) può andare in timeout e ritrasmetterlo, forse addirittura molte volte. Poiché non si può scartare il pacchetto fino a quando non sia giunta conferma della ricezione, la congestione dalla parte del ricevente forza il mittente a evitare di rilasciare buffer che diversamente avrebbe liberato. In questo modo, la congestione risale all'indietro, come auto che si avvicinano a un casello.

È importante sottolineare la differenza tra controllo della congestione e controllo di flusso, poiché la relazione è sottile. Il controllo della congestione si preoccupa di fare in modo che la rete sia in grado di trasportare il traffico richiesto. È una problematica globale, che coinvolge il comportamento di tutti gli host e di tutti i router, il processo di memorizzazione e rispedizione all'interno dei router e tutti gli altri fattori che tendono a diminuire la capacità di trasporto della rete.

Il controllo di flusso, al contrario, è collegato al traffico punto-a-punto tra un dato mittente e un dato ricevente. Il suo compito è assicurarsi che un mittente veloce non possa continuamente trasmettere dati più velocemente di quanto il ricevente possa assorbirne. Il controllo di flusso quasi sempre coinvolge qualche feed-back diretto dal ricevente al mittente, necessario per informare il mittente della situazione all'altro capo.

Per vedere la differenza tra questi due concetti, si consideri una rete basata su fibra ottica con una capacità di 1000 gigabit/s, sulla quale un supercomputer sta cercando di trasferire un file a un personal computer alla velocità di un 1 Gbps. Sebbene non ci sia congestione

(la rete non presenta problemi), è necessario forzare il supercomputer a interruzioni frequenti, per agevolare il funzionamento del personal computer.

All'altro estremo, si consideri una rete store-and-forward con linee a un 1 Mbps e 1000 computer potenti, metà dei quali sta cercando di trasferire file all'altra metà alla velocità di 100 kbps. In questo caso il problema non è dato da mittenti veloci che sovraccaricano ricevitori lenti, ma solo dal fatto che il traffico totale richiesto è più grande di quello che può essere gestito dalla rete.

Il motivo per cui il controllo di congestione e quello di flusso sono spesso confusi è il fatto che qualche algoritmo di controllo di congestione funziona spedendo messaggi ai vari mittenti chiedendo di rallentare quando la rete presenta problemi. Quindi un host può ricevere un messaggio "vai piano" o perché il ricevente non è in grado di gestire il carico, o perché non può essere gestito dalla rete. Torneremo su questo punto più tardi.

Inizieremo il nostro studio del controllo di congestione descrivendo un modello generale per la sua gestione. Vedremo quindi numerosi approcci per eliminarlo fin dalla nascita. Dopo di ciò, vedremo vari algoritmi dinamici per affrontarlo una volta che si è presentato.

5.3.1 Principi generali del controllo di congestione

Molti problemi in sistemi complessi, come le reti di computer, possono essere osservati da un punto di vista di teoria del controllo. Questo approccio porta a dividere tutte le soluzioni in due categorie: a ciclo aperto e a ciclo chiuso. Le soluzioni a ciclo aperto cercano di risolvere il problema con una buona progettazione, per evitare che si presenti. Una volta che il sistema è funzionante, non vengono eseguite correzioni "a metà strada". Per implementare il controllo a ciclo aperto, è possibile decidere quando accettare nuovi pacchetti, quando scartare i pacchetti (e quali) e prendere decisioni di scheduling in vari punti della rete. Tutti questi metodi hanno in comune il fatto che prendono decisioni indipendentemente dallo stato corrente della rete.

Al contrario, le soluzioni a ciclo chiuso sono basate sul concetto di ciclo di feed-back. Quando viene applicato al controllo di congestione, questo approccio è suddiviso in tre parti.

1. Monitorare il sistema per rilevare quando e dove avvengono congestioni.
2. Passare queste informazioni a chi può prendere provvedimenti.
3. Modificare il funzionamento del sistema per correggere il problema.

Per monitorare le congestioni nella rete possono essere utilizzate diverse metriche. Fra queste, le principali sono la percentuale di pacchetti scartati per mancanza di spazio nei buffer, la lunghezza media delle code, il numero di pacchetti che vanno in timeout e vengono ritrasmessi, il ritardo medio dei pacchetti e la deviazione standard del ritardo dei pacchetti. In tutti i casi, numeri crescenti indicano una congestione in aumento.

Il secondo passo nel ciclo di feed-back è quello di trasferire le informazioni sulla congestione dal punto in cui vengono rilevate a quello in cui si può prendere qualche provvedimento in proposito. Il metodo più ovvio richiede che il router che rileva la congestione spedisca un pacchetto alla sorgente o alle sorgenti del traffico, annunciando il problema.

Naturalmente, questi pacchetti extra aumentano il carico proprio nel momento in cui non è richiesto traffico ulteriore, e cioè quando la rete è congestionata.

Ciò nonostante, esistono altre possibilità. Ad esempio, si può riservare per i router un bit o un campo in ogni pacchetto, da riempire ogni volta la congestione superi un certo livello soglia. Quando un router rileva uno stato di congestione, riempie il campo di tutti i pacchetti per avvisare i vicini.

Un ulteriore approccio richiede che gli host o i router spediscano periodicamente pacchetti sonda per informarsi esplicitamente della congestione. Questa informazione può essere utilizzata poi per instradare il traffico aggirando le aree con problemi. Alcune stazioni radio hanno degli elicotteri che volano sulle città per informare delle congestioni del traffico, nella speranza che i loro ascoltatori instradino i loro pacchetti (auto) evitando i punti caldi.

In tutti gli schemi basati su feed-back, la speranza è che le conoscenze sulla congestione permetteranno agli host di prendere decisioni appropriate per ridurre il fenomeno. Per funzionare correttamente, la scala temporale deve essere regolata attentamente. Se un router ordinasse STOP ogni volta che riceve due pacchetti di seguito, e ordinasse GO ogni volta che resta inattivo per 20 µs, il sistema oscillerebbe selvaggiamente e non convergerebbe mai. Al contrario, se attendesse di essere sicuro per 30 min prima di dire qualcosa, il meccanismo di controllo della congestione reagirebbe troppo pigramente per essere realmente utilizzabile. Per funzionare correttamente, è necessario qualche tipo di mediazione, ma trovare la giusta costante temporale non è un problema banale.

Esistono molti algoritmi per il controllo della congestione. Al fine di fornire un metodo per organizzarli in modo assennato, Yang e Reddy (1995) hanno sviluppato una tassonomia per gli algoritmi di controllo della congestione. Inizialmente dividono tutti gli algoritmi in base a ciclo aperto o chiuso, come descritto in precedenza. Suddividono inoltre gli algoritmi a ciclo aperto che agiscono dal lato della sorgente da quelli che agiscono dal lato della destinazione. Anche gli algoritmi a ciclo chiuso sono divisi in due categorie: feed-back esplicito in contrapposizione con il feed-back implicito. Negli algoritmi a feed-back esplicito, i pacchetti vengono spediti indietro dal punto di congestione per avvisare la sorgente. Negli algoritmi a feed-back implicito, la sorgente deduce l'esistenza della congestione facendo osservazioni locali, ad esempio sul tempo necessario per avere una conferma dell'avvenuta ricezione.

La presenza di una congestione significa che il carico è (temporaneamente) maggiore di quello che le risorse possono sopportare (in una parte del sistema). È possibile pensare a due soluzioni: incrementare le risorse o decrementare il carico. Ad esempio, la rete può iniziare a utilizzare linee telefoniche per incrementare temporaneamente la larghezza di banda tra due punti. In sistemi come SDMS (vedi capitolo 1), la rete può chiedere al fornitore banda addizionale per un certo tempo. Nei sistemi satellitari, incrementare la potenza di trasmissione spesso produce un aumento di larghezza di banda. Anche suddividere il traffico in percorsi multipli invece di utilizzare sempre quello migliore può aumentare effettivamente la banda. Infine, router che normalmente vengono utilizzati di riserva (per rendere il sistema in grado di tollerare guasti) possono essere attivati per ottenere una maggior capacità nel caso di congestioni serie.

Ciò nonostante, qualche volta non è possibile aumentare la capacità, oppure questa è già

stata aumentata fino al limite. L'unico modo quindi per abbattere la congestione è decrementare il carico. Esistono molti modi di ridurre il carico, fra cui rifiutare il servizio ad alcuni utenti, peggiorare il servizio ad alcuni o a tutti gli utenti, e fare in modo che gli utenti programmino le loro richieste in modo più predicable. Alcuni di questi metodi, che studieremo brevemente, sono più adatti ai circuiti virtuali. Per reti che utilizzano internamente i circuiti virtuali, questi metodi possono essere utilizzati nel livello rete. Per reti datagram, possono essere utilizzati in connessioni del livello trasporto. In questo capitolo, focalizzeremo l'attenzione sul loro uso nel livello rete. Nel prossimo, vedremo come si può gestire la congestione nel livello trasporto.

5.3.2 Politiche di prevenzione della congestione

Iniziamo il nostro studio dei metodi di controllo della congestione descrivendo i sistemi a ciclo aperto. Questi sistemi sono progettati per minimizzare in primo luogo la congestione, piuttosto che lasciare che si presenti e reagire dopo. Si cerca di raggiungere l'obiettivo utilizzando politiche appropriate a vari livelli. In figura 5-23 vediamo differenti politiche per i livelli data link, rete e trasporto che possono influire sulla congestione (Jain, 1990).

Livello	Politiche
Trasporto	<ul style="list-style-type: none"> • Politica di ritrasmissione • Politica di gestione della cache • Politica di ack • Politica di controllo di flusso • Calcolo del timeout
Rete	<ul style="list-style-type: none"> • Circuiti virtuali vs datagram nella sottorete • Politica di servizio e incodamento dei pacchetti • Politica di eliminazione dei pacchetti • Algoritmo di routing • Gestione del tempo di vita dei pacchetti
Data link	<ul style="list-style-type: none"> • Politica di ritrasmissione • Politica di gestione della cache • Politica di ack • Politica di controllo di flusso

Fig. 5-23 Politiche che influiscono sulla congestione.

Partiamo dal livello data link e proseguiamo salendo fra i livelli. La politica di ritrasmissione si occupa della velocità con cui un mittente genera un timeout e cosa deve trasmettere in tal caso. Un mittente "nervoso" che va in timeout velocemente e ritrasmette tutti i pacchetti in sospeso utilizzando go back n caricherà il sistema più pesantemente di un mittente tranquillo che utilizza ripetizioni selettive. Strettamente collegata a questo pro-

blema è la politica di gestione della cache. Se il ricevente scarta abitualmente tutti i pacchetti fuori ordine, questi pacchetti dovranno essere trasmessi di nuovo più tardi, creando carico extra.

Anche la politica di ack influenza la congestione. Se ogni pacchetto viene confermato immediatamente, il pacchetto di conferma genera traffico extra. Al contrario, se gli ack vengono ritardati per inserirli nel traffico di senso inverso, è possibile che avvenga un maggior numero di timeout e di ritrasmissioni. Uno schema rigido di controllo di flusso (ad es. una finestra piccola) riduce la velocità dei dati e aiuta a combattere la congestione.

Nel livello rete la scelta tra circuiti virtuali e datagram incide sulla congestione, in quanto molti algoritmi di controllo della congestione funzionano solo con reti basate su circuito virtuale. A seconda che i router abbiano una coda per ogni linea di input, una coda per ogni linea di output, oppure entrambe, la politica di gestione delle code è differente. È inoltre collegata all'ordine in cui vengono elaborati i pacchetti (ad es. round robin, oppure in base a priorità). La politica di eliminazione è la regola che individua i pacchetti da scartare quando non c'è più spazio. Una buona politica aiuta ad alleviare la congestione, mentre una cattiva la può rendere peggiore.

L'algoritmo di routing può aiutare a evitare la congestione distribuendo il traffico su tutte le linee, mentre un algoritmo mal concegnato può spedire troppo traffico su una linea già congestionata. Infine, la gestione del tempo di vita dei pacchetti si occupa di quanto tempo può vivere un pacchetto prima di essere scartato. Se il tempo di vita è troppo lungo, i pacchetti persi possono intasare i canali per molto tempo; se è troppo piccolo, i pacchetti possono andare in timeout prima di raggiungere la loro destinazione, causando quindi frequenti ritrasmissioni.

Nel livello di trasporto vi sono gli stessi problemi del livello data link; inoltre, la determinazione dell'intervallo di timeout è più difficile, in quanto il tempo di transito attraverso una rete è meno predicable del tempo di transito su un cavo che connette due router. Se è troppo piccolo, possono essere spediti senza motivo pacchetti extra. Se è troppo grande, la congestione viene ridotta, ma il tempo di risposta ne soffrirà ogni volta che viene perso un pacchetto.

5.3.3 Normalizzazione del traffico

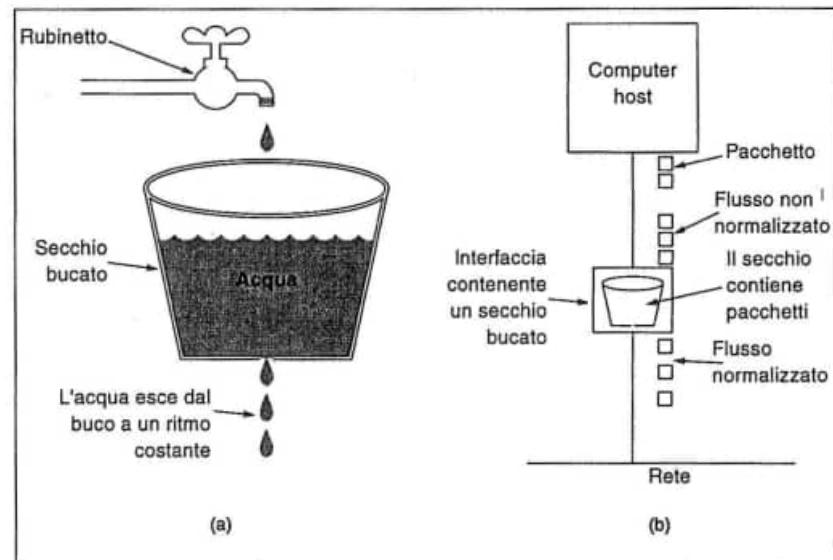
Una delle cause principali delle congestioni è il fatto che spesso il traffico presenta dei picchi improvvisi. Se si potesse fare in modo che gli host trasmettessero a un ritmo costante, le congestioni sarebbero meno comuni. Un altro metodo a ciclo aperto che aiuta a gestire le congestioni consiste nel forzare la trasmissione dei pacchetti a un ritmo più predicable. Questo approccio alla gestione delle congestioni è molto usato nelle reti ATM ed è chiamato **traffic shaping** (normalizzazione del traffico). La normalizzazione del traffico si occupa di regolarizzare la velocità media della trasmissione dei dati (e di diminuire la presenza di picchi). In contrasto, i protocolli di sliding window che abbiamo studiato in precedenza limitano la quantità di dati che possono essere spediti in una volta, non la velocità a cui vengono spediti. Quando viene creato un circuito virtuale, l'utente e la rete (cioè il cliente e il fornitore) si accordano su un certo schema di traffico per esso. Finché il cliente mantiene la sua parte nel patto e spedisce unicamente pacchetti che

rispettano il contratto, il fornitore mantiene la promessa di consegnarli tempestivamente. Tali accordi non sono molto importanti nel caso di trasferimento file, mentre hanno grande importanza nel caso di traffico in tempo reale (ad es. connessioni audio e video), che non tollerano bene le congestioni.

Quando si sta utilizzando la normalizzazione del traffico, in effetti il cliente chiede al fornitore: "Il mio schema di trasmissione sarà come questo. Sei in grado di gestirlo?". Nel caso il fornitore accetti, sorge il problema di come possa controllare se il cliente sta osservando il contratto, e di cosa fare nel caso contrario. Il monitoraggio di un flusso di traffico viene chiamato "pattugliamento del traffico" (**traffic policing**). Accordarsi su un certo modello di traffico e monitorarlo successivamente è più semplice quando si utilizzano reti basate su circuito virtuale invece di reti basate su datagram. Ciò nonostante, le stesse idee possono essere applicate nel livello trasporto delle reti basate su datagram.

L'algoritmo del secchio bucatto

Si immagini un secchio con un piccolo buco sul fondo, come illustrato in figura 5-24 (a). Indipendentemente dalla velocità con cui l'acqua entra nel secchio, il flusso di uscita ha un ritmo costante ρ , quando è presente acqua nel secchio, e 0, quando il secchio è vuoto. Inoltre, una volta riempito il secchio, eventuale acqua aggiuntiva fuoriesce dai lati e va persa (cioè non appare nel flusso di uscita sotto il secchio).



La stessa idea può essere applicata ai pacchetti, come in figura 5-24 (b). Concettualmente, ogni host è connesso a una rete tramite un'interfaccia contenente un secchio bucatto, cioè una

coda interna finita. Se un pacchetto arriva quando la coda è piena, viene scartato. In altre parole, se uno o più processi all'interno dell'host cercano di spedire un pacchetto quando è già stato inserito in coda il numero massimo di pacchetti, il nuovo pacchetto viene scartato senza cerimonie. Questo expediente può essere inserito nell'interfaccia hardware, oppure simulato dal sistema operativo dell'host. Fu proposto per la prima volta da Turner (1986) ed è chiamato **algoritmo del secchio bucatto** (leaky bucket algorithm). In effetti, non è altro che un sistema a coda per singolo server con tempo di servizio costante.

L'host può inserire un pacchetto nella rete a ogni battito di orologio. Ancora una volta, ciò può essere messo in pratica dalla scheda di interfaccia o dal sistema operativo. Questo meccanismo trasforma un flusso diseguale di pacchetti provenienti dai processi utente di un host in un flusso costante, livellando i picchi e riducendo enormemente le possibilità di congestione.

Quando i pacchetti sono tutti della stessa taglia (ad es. celle ATM), questo algoritmo può essere utilizzato come descritto. Viceversa, quando si usano pacchetti di dimensioni variabili, spesso è più indicato consentire la trasmissione di un numero fisso di byte per battito, invece di un pacchetto. Quindi se la regola è 1024 byte per battito, in ogni battito può essere ammesso un pacchetto di 1024 byte, due pacchetti da 512 byte, 4 da 256 byte e così via. Se il numero di byte residui è troppo basso, il prossimo pacchetto deve attendere fino al prossimo battito.

È facile implementare l'algoritmo del secchio bucatto originale. Il secchio bucatto è una coda finita. Quando arriva un pacchetto, se c'è spazio viene aggiunto alla coda; altrimenti viene scartato. A ogni battito dell'orologio, viene trasmesso un pacchetto (a meno che la coda non sia vuota).

Il secchio bucatto con conteggio dei byte si realizza quasi allo stesso modo. A ogni battito, un contatore viene inizializzato a n . Se il primo pacchetto nella coda ha un numero di byte inferiore al valore del contatore, allora viene trasmesso e il contatore viene decrementato del numero di byte corrispondente. Si possono anche spedire altri pacchetti fino a quando il contatore è sufficientemente alto. Quando il contatore scende sotto la lunghezza del prossimo pacchetto nella coda, la trasmissione si ferma fino al prossimo battito di orologio, dopodiché il numero di byte restanti viene sovrascritto e quindi perso.

Per fare un esempio, si immagini che un computer possa produrre dati alla velocità di 25.000.000 di byte/s (200 Mbps) e che anche la rete viaggi a questa velocità. Tuttavia, i router possono reggere questo ritmo di dati solo per brevi intervalli. Durante intervalli lunghi, il loro funzionamento è migliore nel caso la velocità non superi i 2.000.000 di byte/s. Supponiamo ora che i dati vengano prodotti in insiemi compatti di 1.000.000 di byte, lunghi 40 ms, ogni secondo. Per ridurre la velocità media a 2 MB/s, potremmo utilizzare un secchio bucatto con $\rho = 2 \text{ MB/s}$ e una capacità C di 1 MB. Questo significa che scariche di 1 MB possono essere gestite senza perdita di dati, e che tali scariche vengono distribuite su 500 ms, indipendentemente dalla velocità a cui arrivano.

In figura 5-25 (a) possiamo vedere l'input all'algoritmo del secchio bucatto a 25 MB/s per 40 ms. In figura 5-25 (b) vediamo l'output che fuoriesce alla velocità costante di 2MB/s per 500 ms.

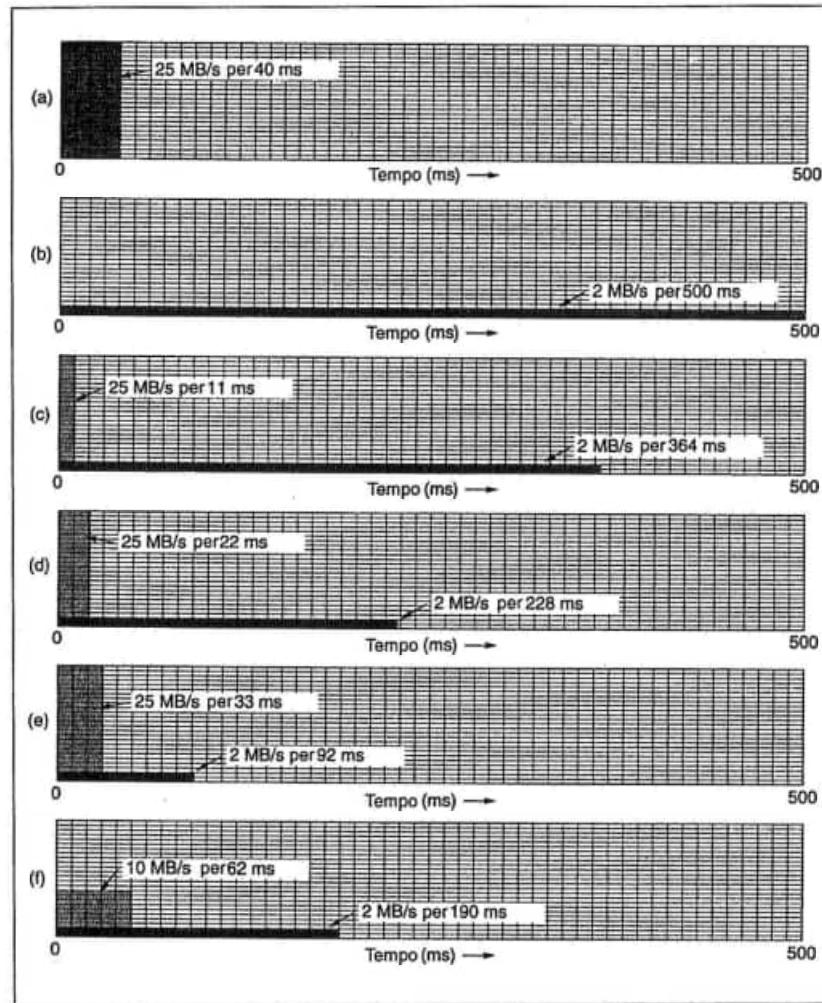


Fig. 5-25 (a) Input per un secchio bucato. (b) Output da un secchio bucato. (c) – (e) Output da un token bucket con capacità di 250 KB, 500 KB e 750 KB. (f) Output da un token bucket di 500 KB che rifornisce un secchio bucato da 10 MB/s.

L'algoritmo token bucket

L'algoritmo del secchio bucato impone un rigido schema di output dove la velocità di trasmissione è attestata a un certo valore medio, indipendentemente dal numero di picchi

presenti nel traffico. Per molte applicazioni, è preferibile consentire che l'output sia leggermente più veloce nel caso arrivino grandi quantità di dati; è quindi necessario un algoritmo più flessibile, se possibile senza perdita di dati. L'**algoritmo token bucket** (algoritmo del secchio di gettoni) possiede queste caratteristiche. In esso, il secchio bucato contiene gettoni, generati da un orologio al ritmo di 1 ogni ΔT s. In figura 5-26 (a) vediamo un secchio contenente 3 gettoni, con 5 pacchetti che aspettano di essere trasmessi. Affinché 1 pacchetto possa essere trasmesso, deve prima catturare e distruggere 1 gettone. In figura 5-26 (b) vediamo che tre dei cinque pacchetti sono passati, mentre gli altri 2 stanno aspettando che vengano generati ulteriori gettoni.

L'algoritmo token bucket fornisce un tipo di regolarizzazione del traffico differente da quella offerta dall'algoritmo del secchio bucato. L'algoritmo non consente che gli host inattivi mettano da parte i permessi per spedire grandi quantità di dati nel seguito; l'algoritmo token bucket consente invece di mettere da parte gettoni fino alla dimensione massima del secchio, n . Questa proprietà significa che è possibile spedire tutti insieme getti lunghi fino a n pacchetti, consentendo qualche picco nel flusso di output e dando una risposta più veloce ai flussi improvvisi di input.

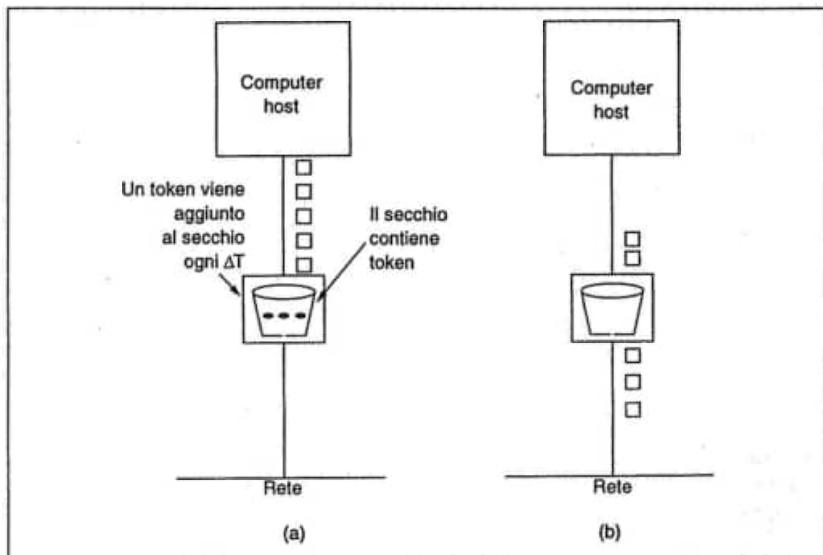


Fig. 5-26 L'algoritmo token bucket. (a) Prima. (b) Dopo.

Un'ulteriore differenza tra questi due algoritmi è la seguente: l'algoritmo token bucket butta via i gettoni quando il secchio è pieno, ma non scarta mai i pacchetti; al contrario, l'algoritmo del secchio bucato scarta i pacchetti quando il secchio si riempie.

Anche in questo caso è possibile immaginare una variante minore, nella quale ogni gettone non rappresenta il diritto di spedire un pacchetto, ma k byte. Un pacchetto può essere

trasmesso se e solo se sono disponibili abbastanza gettoni da coprire la sua lunghezza in byte. I gettoni frazionati vengono conservati per usi futuri.

Gli algoritmi leaky bucket e token bucket possono anche essere usati per regolarizzare il traffico tra router, nello stesso modo in cui vengono utilizzati per regolarizzare l'output degli host nei nostri esempi. Tuttavia, esiste una differenza importante. Se l'algoritmo di token bucket normalizza il traffico di un host, può imporre all'host di smettere di spedire quando le regole glielo impongono. Imporre a un router di smettere di spedire mentre il suo input sta trabocando può portare alla perdita di pacchetti.

Per implementare l'algoritmo token bucket di base è sufficiente una variabile che conti i gettoni. Il contatore viene incrementato di 1 ogni ΔT secondi e viene decrementato di 1 ogni qualvolta viene spedito un pacchetto. Quando il contatore raggiunge lo 0, nessun pacchetto può essere spedito. Nella variante basata sui byte, il contatore viene incrementato di k byte ogni secchio bucato ΔT secondi e decrementato dalla lunghezza totale dei pacchetti spediti. In pratica, con token bucket sono permessi picchi occasionali, ma solo fino a una certa lunghezza. Si osservi ad esempio la figura 5-25 (c), dove è presente un token bucket di capacità di 250 KB. I gettoni arrivano a una velocità che consente un output di 2 MB/s. Assumendo che il token bucket sia pieno quando arriva un picco lungo 1 MB, il bucket può mandare in output al massimo 25 MB/s per circa 11 ms. Quindi deve tornare a 2 MB/s fino a quando l'intero getto di input è stato spedito.

Calcolare la lunghezza massima di una trasmissione accelerata è abbastanza difficile. Non si ottiene semplicemente dividendo 1 MB per 25 MB/s, in quanto mentre il getto viene mandato in output vengono prodotti ulteriori gettoni. Se indichiamo con S la lunghezza in secondi del getto, con C la capacità in byte, con p la velocità di arrivo dei token in byte/s e con M la velocità massima di output in byte/s, possiamo vedere che un getto di output può contenere al massimo $C + pS$ byte. Sappiamo inoltre che il numero di byte trasmessi durante un picco alla massima velocità lungo S secondi è MS . Otteniamo quindi

$$C + pS = MS$$

Possiamo risolvere questa equazione ottenendo $S = C/(M - p)$. Per i nostri parametri di $C = 250$ KB, $M = 25$ MB/s e $p = 2$ MB/s, otteniamo un tempo di scarica di 11 ms circa. Le figura 5-25 (d) e figura 5-25 (e) mostrano l'algoritmo token bucket con capacità rispettivamente di 500 KB e 750 KB.

Un problema potenziale dell'algoritmo token bucket è il fatto che continua a consentire la spedizione di getti di lunga durata, sebbene la lunghezza massima di un getto possa essere regolata da un scelta accurata di p ed M . Spesso è desiderabile ridurre la velocità di picco, ma senza tornare al basso valore dell'algoritmo del secchio bucato originale.

Un modo per ottenere traffico omogeneo è quello di collocare un leaky bucket dopo il token bucket. Il tasso di uscita del leaky buket dovrebbe essere maggiore del tasso di uscita p del token bucket, ma più basso della velocità massima della rete. La figura 5-25 (f) mostra l'output di un token bucket di 500 KB seguito da un leaky bucket da 10 MB/s. Sorvegliare tutti questi schemi può essere abbastanza difficile. Fondamentalmente, la rete deve simulare l'algoritmo e fare in modo che non vengano spediti più pacchetti o byte di

quelli consentiti. I pacchetti in eccesso vengono scartati o declassati, come discusso in seguito.

5.3.4 Specifiche di flusso

La normalizzazione del traffico è più efficace quando il mittente, il ricevente e la rete raggiungono un accordo su di essa. Per raggiungere l'accordo, è necessario specificare lo schema di traffico in modo preciso. Tale accordo viene chiamato **specifiche di flusso** e consiste in una struttura dati che descrive sia il pattern del traffico immesso, sia la qualità del servizio richiesta dalle applicazioni. Una specifica di flusso può essere applicata ai pacchetti spediti lungo un circuito virtuale, oppure a una sequenza di datagram spediti tra la sorgente e la destinazione (o anche a destinazioni multiple).

In questo paragrafo descriveremo una specifica di flusso progettata da Partridge (1992) e illustrata in figura 5-27. L'idea consiste nel fornire la specifica di flusso alla rete per l'approvazione prima di stabilire una connessione o di spedire una sequenza di datagram. La rete può accettare, rifiutare, oppure rispondere con una controproposta ("Non posso darti 100 ms di ritardo medio; puoi vivere con 150 ms?"). Una volta che il mittente e la rete hanno concluso un accordo, il primo può chiedere al ricevente se concorda anch'esso.

Caratteristiche dell'input	Servizio desiderato
Dimensione massima dei pacchetti (byte)	Sensibilità alle perdite (byte)
Velocità dei token bucket (byte/s)	Intervallo fra le perdite (μs)
Dimensione del token bucket (byte)	Sensibilità ai picchi di perdite (pacchetti)
Velocità massima di trasmissione (byte/s)	Minimo ritardo notificato (μs)
	Massima variazione di ritardo (μs)
	Qualità della garanzia

Fig. 5-27 Un esempio di specifica di flusso.

Esaminiamo ora i parametri della nostra specifica di flusso partendo dalla specifica di traffico. La *Dimensione massima dei pacchetti* indica quanto grandi possono essere i pacchetti. I due parametri successivi assumono implicitamente che il traffico venga normalizzato dall'algoritmo token bucket (lavorando in byte) e indicano quanti byte/s vengono immessi nel token bucket, e quanto grande sia il bucket. Se la velocità di immissione è r byte/s e la dimensione del bucket è di b byte, allora in qualsiasi intervallo arbitrario di tempo Δt , il numero massimo di byte che possono essere spediti è $b+r\Delta t$. Il primo termine rappresenta la dimensione massima del bucket all'inizio dell'intervallo, mentre il secondo rappresenta i crediti inseriti durante l'intervallo di tempo. La *Velocità massima di trasmissione* è la velocità più alta che l'host è in grado di produrre sotto qualsiasi condizione e implicitamente specifica l'intervallo di tempo più corto nel quale il token bucket può essere svuotato.

La seconda colonna specifica cosa vuole l'applicazione dalla rete. Il primo e secondo parametro rappresentano il numeratore e il denominatore di una frazione che fornisce il più alto

tasso di perdite ritenuto accettabile (ad es., 1 byte/h). Alternativamente, possono indicare che il flusso è insensibile ai pacchetti persi. La *Sensibilità ai picchi di perdita* indica il numero massimo di pacchetti persi consecutivamente che possono essere tollerati.

I due parametri di servizio seguenti gestiscono i ritardi. Il *Minimo ritardo notificato* indica quanto tempo un pacchetto può essere in ritardo senza che l'applicazione ne sia informata. Per un trasferimento file, un secondo può essere accettabile, mentre per un canale audio 3 ms può essere il limite. La *Massima variazione di ritardo* cerca di quantificare il fatto che alcune applicazioni non sono sensibili al ritardo effettivo, mentre sono fortemente sensibili al **jitter** (tremolio), cioè alle variazioni nel tempo di transito dei pacchetti. È uguale a due volte il numero di microsecondi per cui il ritardo di un pacchetto può scostarsi dalla media. Quindi un valore di 2000 significa che un pacchetto può essere fino a 1 ms in anticipo o in ritardo, ma non di più.

Infine, la *Qualità della garanzia* indica se l'applicazione si basa su di essa oppure no. Da un lato, le caratteristiche riguardanti perdite di dati e ritardi potrebbero essere obiettivi ideali, ma non ci sono problemi se non sono soddisfatti. D'altra parte, essi potrebbero essere così importanti che se non venissero soddisfatti, l'applicazione potrebbe solo terminare. Sono possibili anche posizioni intermedie.

Sebbene finora abbiamo studiato la specifica di flusso come una richiesta dall'applicazione alla rete, essa può anche essere vista come una risposta contenente ciò che la rete è in grado di fare. Quindi può essere potenzialmente utilizzata per una negoziazione estesa sul livello di servizio.

Un problema inerente a qualsiasi specifica di flusso è il fatto che le applicazioni potrebbero non sapere cosa vogliono realmente. Ad esempio, un programma applicativo eseguito a New York potrebbe accontentarsi di un ritardo di 200 ms con Sidney, ma non sopportare lo stesso ritardo con Boston. Qui il "servizio minimo" è chiaramente una funzione di ciò che si pensa sia possibile.

5.3.5 Controllo delle congestioni in reti basate su circuito virtuale

I metodi di controllo della congestione descritti in precedenza sono principalmente a ciclo aperto: in primo luogo cercano di prevenire la congestione, piuttosto di occuparsene dopo che è comparsa. In questo paragrafo descriveremo alcuni approcci al controllo dinamico della congestione nelle reti basate su circuito virtuale. Poi daremo un'occhiata a tecniche che possono essere utilizzate con qualsiasi tipo di rete.

Una tecnica che viene largamente utilizzata per controllare un fenomeno di congestione che ha già iniziato a peggiorare è il **controllo delle ammissioni**. L'idea è semplice: una volta che viene segnalata una congestione, non vengono creati nuovi circuiti virtuali fino a quando il problema non è scomparso. Quindi i tentativi di creare nuove connessioni nel livello trasporto falliscono. Lasciare entrare altri utenti peggiorerebbe la situazione. Sebbene questo approccio sia rozzo, è facile da implementare. Nei sistemi telefonici, quando un commutatore si sovraccarica mette in funzione il controllo di ammissione, non trasmettendo il segnale di libero.

Un'altra soluzione è quella di consentire nuovi circuiti virtuali, ma instradarli attentamente aggirando le aree congestionate. Ad esempio, si consideri la rete di figura 5-28 (a), nella quale due router sono congestionati.

Si supponga che un host collegato al router A voglia creare una connessione con un host collegato al router B. Normalmente, questa connessione passerebbe attraverso uno dei router congestionati. Per evitare questa situazione, possiamo ridisegnare la rete come mostrato in figura 5-28 (b), omettendo i router congestionati e tutte le loro linee. La linea tratteggiata mostra un possibile percorso per il circuito virtuale che evita i router congestionati.

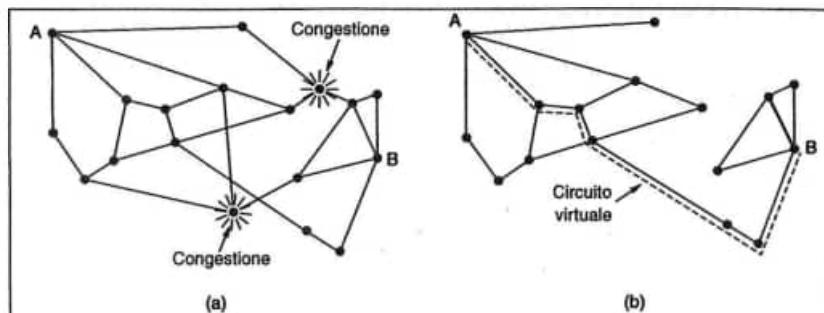


Fig. 5-28 (a) Una rete congestionata; (b) una rete ridisegnata che elimina la congestione e un circuito virtuale tra A e B.

Un'altra strategia collegata ai circuiti virtuali è quella di negoziare un accordo tra gli host e la rete all'atto di creazione di un circuito. Questo accordo specifica normalmente la quantità e il tipo di traffico, la qualità di servizio richiesta e altri parametri. Per mantenere la sua parte di accordo, in genere la rete riserverà delle risorse lungo il percorso al momento di stabilire il circuito. Queste risorse possono includere spazio per tabelle e buffer nei router e larghezza di banda lungo i canali.

In questo modo, nei nuovi circuiti virtuali la congestione è meno probabile, in quanto è garantita la presenza di tutte le risorse necessarie.

Questo tipo di prenotazione può essere eseguita sempre come procedura di funzionamento standard, oppure solo quando la rete è congestionata. Uno svantaggio nell'eseguirla sempre è che tende a sprecare risorse. Se sei circuiti virtuali che possono utilizzare 1 Mbps passano tutti attraverso la stessa linea da 6 Mbps, la linea deve essere marcata come piena, sebbene accada raramente che tutti e sei i circuiti virtuali stiano trasmettendo contemporaneamente. Conseguentemente, il prezzo da pagare per il controllo della congestione è lo spreco di banda.

5.3.6 Pacchetti regolatori

Passiamo ora a un approccio che può essere utilizzato sia nei circuiti virtuali che nelle reti basate su datagram. Ogni router può facilmente monitorare l'utilizzazione delle sue linee di output e delle altre risorse. Ad esempio, può associare a ogni linea una variabile reale u , il cui valore, compreso tra 0,0 e 1,0, riflette l'utilizzazione recente della linea. Per mantenere una buona stima di u si può eseguire un campionamento periodico f dell'utilizzazione istantanea del canale (0 oppure 1) e può venire aggiornato in accordo a

$$u_{\text{new}} = au_{\text{old}} + (1 - a)f$$

dove la costante a determina la velocità con cui il router dimentica la storia recente. Ogni volta che u supera una certa soglia, il canale di output entra in uno stato di "allarme". Per ogni pacchetto in arrivo, si verifica se il corrispondente canale di output è in uno stato di allarme, nel qual caso il router spedisce indietro un **choke packet** (pacchetto regolatore) all'host sorgente, indicandogli la destinazione trovata nel pacchetto. Il pacchetto originale viene marcato (viene acceso un bit del preamble), in modo che non generi ulteriori pacchetti regolatori lungo il percorso, e viene inoltrato nel solito modo.

Quando l'host sorgente riceve il pacchetto regolatore, è vincolato a ridurre dell' $X\%$ il traffico spedito alla destinazione specificata. Poiché gli altri pacchetti diretti alla stessa destinazione sono probabilmente già per la strada e genereranno ulteriori pacchetti regolatori, l'host dovrebbe ignorare per un intervallo fissato i pacchetti regolatori che si riferiscono alla stessa destinazione. Al termine di questo periodo, l'host resta in attesa di nuovi pacchetti regolatori per un ulteriore intervallo. Se ne arriva un altro, il canale è ancora congestionato e l'host riduce ulteriormente il flusso e inizia a ignorare nuovamente i pacchetti regolatori. Se non arrivano pacchetti regolatori durante il periodo di ascolto, l'host può tornare a incrementare il flusso. Il feed-back implicito in questo protocollo può aiutare a prevenire la congestione a meno che non si presentino problemi.

Gli host possono ridurre il traffico regolando i parametri della loro politica, ad esempio la dimensione delle finestre e il tasso di output del leaky bucket. Generalmente, il primo pacchetto regolatore fa in modo che la velocità di trasmissione dati sia ridotta al 50% di quella iniziale, quello successivo causa una riduzione al 25% e così via. Gli incrementi vengono eseguiti con valori più piccoli per evitare che la congestione si ripresenti in fretta. Sono state proposte numerose varianti di questo algoritmo per il controllo della congestione. In una di esse, i router possono mantenere più di una soglia. A seconda di quale soglia sia stata oltrepassata, i pacchetti regolatori possono contenere un avvertimento leggero, un avvertimento severo oppure un ultimatum.

Un'ulteriore variante consiste nell'utilizzare come segnale di attivazione la lunghezza delle code o l'utilizzazione dei buffer, al posto dell'utilizzazione dei canali. Naturalmente, con questa metrica è possibile utilizzare gli stessi pesi esponenziali utilizzati per u .

Gestione pesata e onesta delle code

Un problema che si presenta utilizzando i pacchetti regolatori è il fatto che i provvedimenti che devono essere presi dalla sorgente sono volontari. Supponiamo che un router sia sommerso da pacchetti provenienti da quattro sorgenti e spedisca pacchetti regolatori a tutte. Una di esse riduce il traffico come previsto, mentre le altre tre continuano a spedire a pieno volume. Come risultato, l'host "onesto" ottiene una percentuale di banda addirittura minore di quella che aveva in precedenza.

Per aggirare questo problema, e quindi rendere più attraente l'essere conformi, Nagle (1987) propose un algoritmo di gestione "onesta" delle code. L'essenza dell'algoritmo è la seguente: i router devono possedere code multiple per ognuna delle linee di output, una per ogni sorgente; quando una linea diventa inattiva, il router scandisce le code in modo round robin, prendendo il primo pacchetto dalla coda successiva. In questo modo, con n host che competono per un dato canale di output, ogni host può spedire un pacchetto ogni

n spediti. Spedire ulteriori pacchetti non aumenta questa frazione. Alcuni commutatori ATM utilizzano questo algoritmo.

Sebbene questo algoritmo sia un punto di partenza, purtroppo presenta un problema: assegna una banda maggiore agli host che utilizzano pacchetti grandi rispetto a host che utilizzano pacchetti più piccoli. Demers et al. (1990) hanno suggerito un miglioramento nel quale il round robin viene eseguito in modo da simulare un round robin byte-per-byte, al posto di un round robin pacchetto-per-pacchetto.

In effetti, l'algoritmo scandisce le code ripetutamente, byte dopo byte, finché la scansione di tutti i pacchetti non viene terminata. I pacchetti vengono quindi ordinati in base all'istante in cui la loro scansione è terminata e spediti in tal ordine. L'algoritmo è illustrato in figura 5-29.

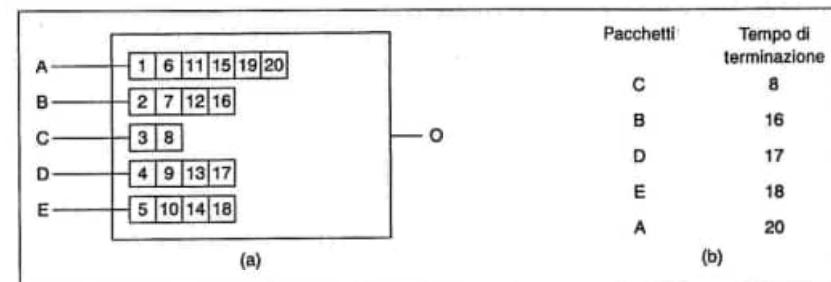


Fig. 5-29 (a) Un router con cinque pacchetti in coda sul canale O . (b) Tempi di terminazione per i cinque pacchetti.

In figura 5-29 (a) è possibile vedere pacchetti lunghi da 2 a 6 byte. Al tempo (virtuale) 1, viene spedito il primo byte del pacchetto sulla linea A . Segue quindi il primo byte del pacchetto della linea B , e così via. Il primo pacchetto a finire è C , al tempo 8. L'ordine finale è visibile in figura 5-29 (b). In assenza di nuovi arrivi, i pacchetti vengono spediti nell'ordine elencato, da C ad A .

Uno dei problemi di questo algoritmo è il fatto che assegna a tutti gli host la stessa priorità. In molte situazioni, è desiderabile dare ai file server e ad altri servizi maggiore banda rispetto ai clienti; per questo motivo, possono ricevere più di un byte per battito di orologio. Questo algoritmo modificato viene chiamato **gestione pesata e onesta delle code** ed è molto utilizzato. In qualche caso il peso è uguale al numero di circuiti virtuali o flussi che fuoriescono da un elaboratore, e quindi ogni processo ottiene banda uguale. Un'implementazione efficiente dell'algoritmo viene discussa in Shreedhar, Varghese (1995).

Pacchetti regolatori hop-per-hop

Nel caso di elevate velocità, oppure di lunghe distanze, la spedizione di un pacchetto regolatore all'host sorgente può non essere efficace a causa della reazione lenta. Si consideri, ad esempio, un host di San Francisco (router A in figura 5-30) che sta spedendo messaggi a un host di New York (router D in figura 5-30) alla velocità di 155 Mbps. Se l'host di New York inizia a esaurire lo spazio, saranno necessari circa 30 ms affinché un

pacchetto regolatore arrivi a San Francisco per chiedere di rallentare. La propagazione del pacchetto regolatore è rappresentata come il secondo, il terzo e il quarto passo in figura 5-30 (a). In questi 30 ms, verranno spediti altri 4,6 megabit di dati (cioè oltre 10.000 celle ATM). Anche nel caso in cui l'host di San Francisco smettesse immediatamente, i 4,6 megabit presenti nei canali continuerebbero ad affluire e dovrebbero venire gestiti. Solo nel settimo diagramma di figura 5-30 (a) il router di New York ridurrà il flusso.

Un approccio alternativo richiede che il pacchetto regolatore abbia effetto su ognuno dei salti che effettua, come nella sequenza di figura 5-30 (b). Qui, non appena il pacchetto regolatore raggiunge *F*, questi deve ridurre il flusso verso *D*. Questa operazione richiederà che *F* dedichi più buffer al flusso, in quanto la sorgente continua a spedire a pieno ritmo, ma fornisce un sollievo immediato a *D* (come un rimedio per il mal di testa in una televisione commerciale). Nel passo successivo, il pacchetto regolatore raggiunge *E*, imponendogli di ridurre il flusso verso *F*. Questa azione aumenta la richiesta di buffer in *E*, ma fornisce un sollievo immediato a *F*. Infine, il pacchetto regolatore raggiunge *A* e il flusso diminuisce effettivamente.

Il risultato finale di questo schema passo-per-passo è quello di fornire un rapido sollievo nel punto di congestione, al prezzo di utilizzare più buffer a monte. In questo modo, la congestione può essere stroncata sul nascere senza perdere alcun pacchetto. In Mishra, Kanakia (1992) l'idea viene discussa più dettagliatamente e vengono presentati risultati di simulazione.

5.3.7 Caduta di carico

Quando nessuno dei metodi precedenti elimina le congestioni, i router possono tirare fuori l'artiglieria pesante: la caduta di carico (**load shedding**). È un modo elaborato per dire che quando i router sono subissati da pacchetti che non possono gestire, li gettano via. Il termine proviene dall'ambiente delle grandi società elettriche e si riferisce alla pratica di mandare intenzionalmente in blackout certe aree per evitare il collasso dell'intera griglia nei caldi giorni estivi, quando la domanda di elettricità eccede largamente l'offerta.

Un router che è sommerso dai pacchetti può selezionare quelli da gettare in modo casuale, anche se in genere può fare di meglio. L'insieme dei pacchetti da scartare dipende dall'applicazione in esecuzione. Per il trasferimento file, un pacchetto vecchio vale di più di uno nuovo, in quanto scartare il pacchetto 6 e tenere i pacchetti 7-10 provocherebbe un buco presso il ricevente, che potrebbe forzare la ritrasmissione dei pacchetti 6-10 (nel caso il ricevente scarti abitualmente i pacchetti fuori ordine). In un file di 12 pacchetti, saltare il 6 può richiedere la rispedizione dei pacchetti 7-12, mentre scartare il 10 può forzare la rispedizione dei soli pacchetti 10-12. Al contrario, per i dati multimediali, un pacchetto nuovo è più importante di un pacchetto vecchio. La prima politica (vecchio è meglio di nuovo) è chiamata spesso **vino**, mentre la seconda (nuovo è meglio di vecchio) viene spesso chiamata **latte**.

Una versione più intelligente di questo algoritmo richiede la cooperazione del mittente. Per certe applicazioni, alcuni pacchetti sono più importanti di altri. Ad esempio, certi algoritmi di compressione video trasmettono periodicamente un quadro intero e quindi spediscono i quadri successivi come differenze rispetto all'ultimo quadro completo. In questo caso, è preferibile scartare un pacchetto che fa parte di un quadro differenza, piuttosto di scartare uno che fa parte del quadro intero. Come ulteriore esempio, si consideri la trasmissione di un documento contenente testo ASCII e immagini. Perdere una linea di pixel in qualche immagine è meno dannoso della perdita di una linea di testo leggibile.

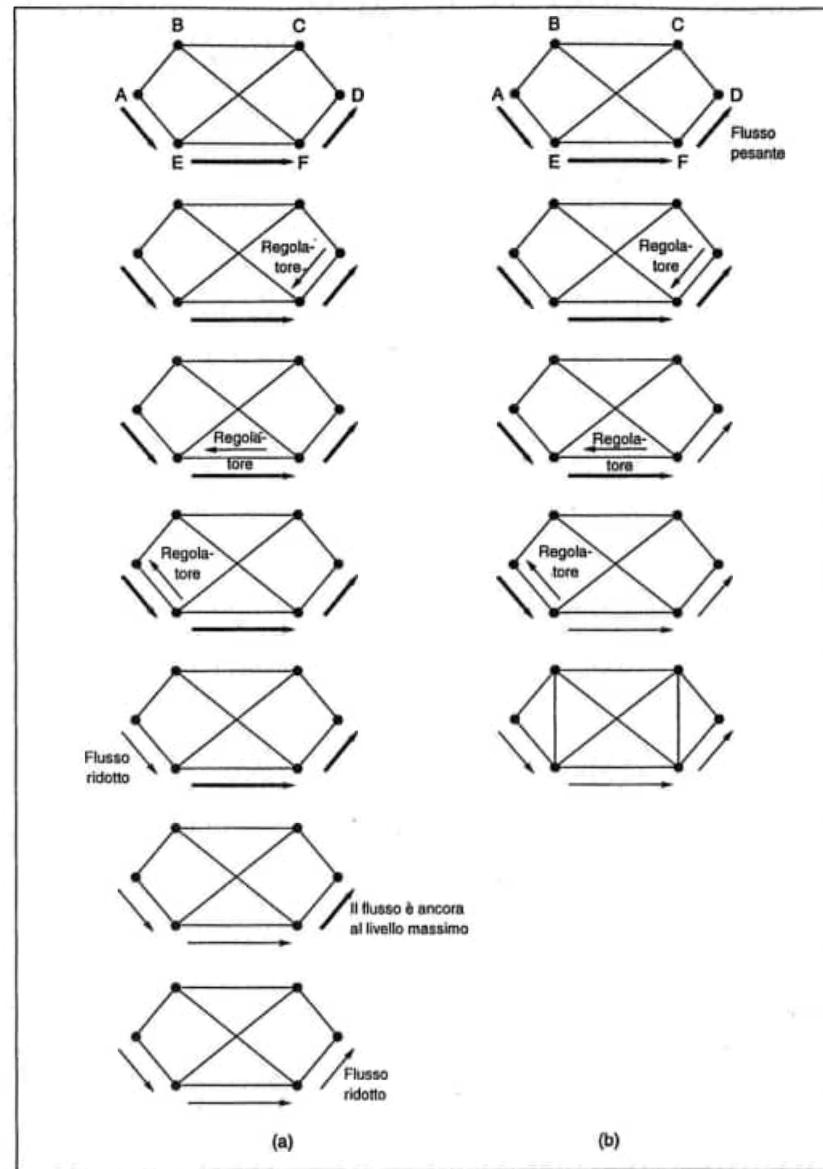


Fig. 5-30 (a) Un pacchetto regolatore che influisce solo sulla sorgente. (b) Un pacchetto regolatore che influisce su ogni hop che attraversa.

Per implementare una politica di scarto intelligente, le applicazioni dovrebbero marcare i loro pacchetti tramite classi di priorità per indicare quanto sono importanti. Se questo accade, ogni qual volta sia necessario scartare un pacchetto, i router possono scartare per primi quelli della classe più bassa, poi di quella successiva e così via. Naturalmente, a meno che non ci sia qualche incentivo significativo a marcare i pacchetti diversamente da molto importante – non scartare mai, nessuno lo farebbe.

L'incentivo potrebbe essere di tipo economico, nel qual caso spedire pacchetti a bassa priorità potrebbe costare meno di spedire pacchetti ad alta priorità. Alternativamente, le classi di priorità potrebbero essere accoppiate alla normalizzazione del traffico. Ad esempio, potrebbe esserci una regola che dice che quando viene usato un algoritmo token bucket e un pacchetto arriva in un momento in cui non ci sono token disponibili, potrebbe essere spedito ugualmente nel caso sia marcato con la priorità più bassa possibile, e quindi soggetto a essere scartato nell'istante in cui si presentano i problemi. Nel caso di carico leggero, gli utenti potrebbero essere felici di operare in questo modo; non appena il carico cresce e i pacchetti iniziano effettivamente a essere scartati, possono farsi da parte e spedire pacchetti solo quando ci sono token disponibili.

Un'altra opzione è quella di permettere agli host di superare il limite specificato nell'accordo negoziato all'atto di creazione del circuito virtuale (cioè utilizzare una banda maggiore di quella consentita), ma alla condizione che tutto il traffico in eccesso debba essere marcato a bassa priorità. Tale strategia non è effettivamente una cattiva idea, in quanto utilizza più efficientemente le risorse inattive, permettendo agli utenti di utilizzarle finché nessun altro è interessato, ma senza stabilire un diritto quando la situazione diventa più pesante.

Marcare i pacchetti con la classe richiede 1 o più bit del preambolo nei quali scrivere la priorità. Le celle ATM hanno 1 bit riservato nel preambolo per questo scopo; quindi ogni cella ATM può essere marcata a bassa priorità oppure ad alta priorità. I commutatori ATM utilizzano proprio questo bit per decidere quali pacchetti scartare.

In alcune reti, i pacchetti sono raggruppati insieme in unità più grandi che vengono utilizzate per scopi di ritrasmissione. Nelle reti ATM ad esempio, ciò che abbiamo chiamato "pacchetti" sono celle a lunghezza fissa. Queste celle sono semplicemente frammenti di "messaggi". Quando una cella viene scartata, alla fine verrà ritrasmesso l'intero "messaggio", non solo la cella mancante. In questo caso, un router che scarta una cella di un messaggio è libero anche di tutte le altre, in quanto trasmetterle consuma larghezza di banda e non serve a nulla – anche se arrivassero, verrebbero ritrasmesse in seguito. Risultati di simulazione mostrano che se un router intravede guai all'orizzonte, è preferibile iniziare a scartare pacchetti in anticipo, piuttosto che esserne sommersi (Floyd, Jacobson (1993); Romanow, Floyd (1994). In questo modo si può evitare che la congestione prenda piede.

5.3.8 Controllo del jitter

Per applicazioni quali trasmissioni audio e video, non è importante che un pacchetto impieghi 20 ms o 30 ms a essere consegnato, finché il tempo di transito è costante. Il fatto che alcuni pacchetti siano consegnati in 20 ms e altri in 30 ms può produrre una cattiva qualità del suono o dell'immagine. Quindi l'accordo potrebbe essere che il 99% dei pacchetti vengano consegnati con un ritardo compreso fra 24,5 ms e 25,5 ms. Il valore

medio scelto deve essere ovviamente realizzabile. In altre parole, nel calcolo deve essere compreso anche un valore medio di congestione.

Il jitter può essere limitato calcolando il tempo atteso di transito per ogni salto lungo il percorso. Quando un pacchetto arriva a un router, quest'ultimo verifica quanto esso sia in ritardo oppure in anticipo rispetto alla sua tabella oraria. Questa informazione viene registrata nel pacchetto e aggiornata a ogni salto. Se il pacchetto è in anticipo, viene trattenuto quel tanto che basta a riportarlo in orario. Se è in ritardo, il router cerca di spedirlo fuori in fretta. In effetti, l'algoritmo che determina quale pacchetto dovrebbe essere il prossimo a uscire fra quelli che competono per una linea di output potrebbe scegliere sempre quello più in ritardo sulla sua tabella oraria. In questo modo, i pacchetti in anticipo vengono rallentati, mentre pacchetti in ritardo sono accelerati, in entrambi i casi riducendo il jitter.

5.3.9 Controllo di congestione per il multicasting

Tutti gli algoritmi per il controllo di congestione discussi fino a ora si occupano di messaggi spediti da una singola sorgente a una singola destinazione. In questo paragrafo descriveremo un metodo per gestire flussi di multicast da sorgenti multiple a destinazioni multiple. Ad esempio, si immagini un gran numero di stazioni televisive a circuito chiuso che trasmettono audio e video a un gruppo di riceventi, ognuno dei quali può vedere una o più stazioni contemporaneamente ed è libero di cambiare stazione a piacere. Un'applicazione di questa tecnologia potrebbe essere la videoconferenza, nella quale ogni partecipante potrebbe concentrare la propria attenzione sull'oratore corrente oppure sull'espressione del direttore, a piacere.

In molte applicazioni di multicast, i gruppi possono cambiare in modo dinamico, ad esempio quando una persona si unisce a una videoconferenza oppure si annoia e cerca una telenovela. In questo caso, l'approccio in cui il mittente riserva banda in anticipo non è soddisfacente, in quanto richiederebbe che ogni mittente tenesse traccia di tutti gli ingressi e le uscite fra i suoi ascoltatori e rigenerasse lo spanning tree a ogni modifica. In un sistema progettato per trasmettere televisione via cavo, con milioni di sottoscrittori, non funziona per niente.

RSVP – Resource reSerVation Protocol (protocollo di prenotazione risorse)

Una soluzione interessante in grado di gestire questo ambiente è il protocollo RSVP (Zhang et al., 1993). Tale soluzione permette a mittenti multipli di trasmettere a gruppi multipli di riceventi, consente ai riceventi individuali di cambiare canale liberamente e ottimizza la banda eliminando allo stesso tempo le congestioni.

Nella sua forma più semplice, il protocollo utilizza il multicast routing basato su spanning tree, come discusso in precedenza. A ogni gruppo viene assegnato un indirizzo di gruppo. Per spedire a un gruppo, un mittente inserisce l'indirizzo del gruppo nei suoi pacchetti. L'algoritmo di multicast routing standard costruisce quindi lo spanning tree che raggiunge tutti i membri del gruppo. L'algoritmo di routing non fa parte di RSVP. L'unica differenza con il multicasting normale è una piccola informazione extra che viene spedita al gruppo periodicamente per ordinare ai router di mantenere certe strutture dati nella loro memoria. Come esempio, si consideri la rete di figura 5-31 (a). Gli host 1 e 2 sono mittenti multicast, e gli host 3, 4 e 5 sono riceventi multicast. In questo esempio, i mittenti e i riceventi sono

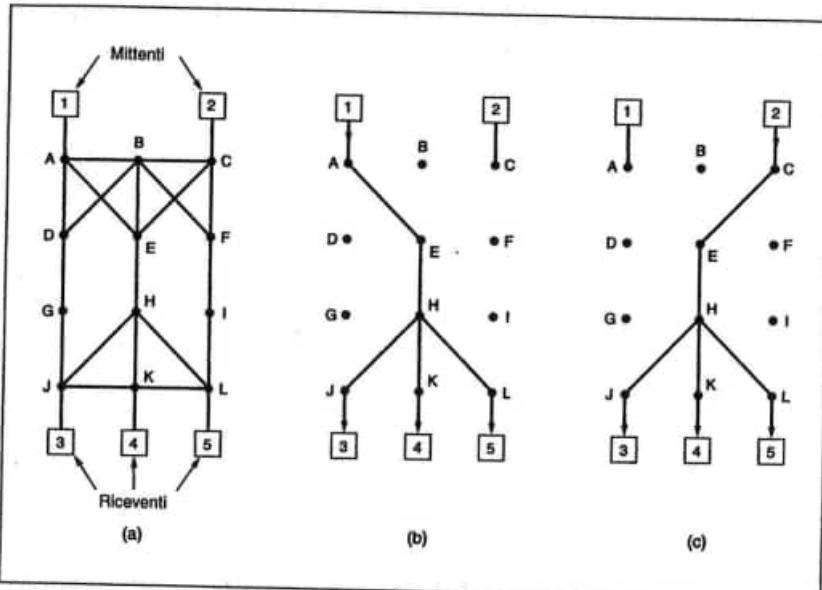


Fig. 5-31 (a) Una rete. (b) Lo spanning tree di multicast per l'host 1. (c) Lo spanning tree di multicast per l'host 2.

disgiunti, ma in generale i due insiemi possono intersecarsi. Gli alberi di multicast per gli host 1 e 2 sono rispettivamente mostrati in figura 5-31 (b) e in figura 5-31 (c). Per ottenere una migliore ricezione ed eliminare la congestione, qualsiasi ricevente in un gruppo può spedire un messaggio di prenotazione che risale lungo l'albero fino al mittente. Il messaggio viene propagato utilizzando l'algoritmo reverse path forwarding discussso in precedenza. A ogni salto, il router registra la prenotazione e riserva la banda necessaria; se non c'è banda sufficiente, restituisce una risposta di fallimento. Quando il messaggio arriva alla sorgente, la banda è stata riservata lungo il percorso dal mittente al ricevente tramite le richieste di prenotazione lungo lo spanning tree.

Un esempio di questa prenotazione è mostrato in figura 5-32 (a). Qui l'host 3 ha richiesto un canale per l'host 1. Una volta preparato, i pacchetti possono andare da 1 a 3 senza congestione. A questo punto si consideri cosa succede se l'host 3 prenota successivamente un canale di un altro mittente, host 2, in modo che l'utente possa guardare due programmi televisivi contemporaneamente. Viene riservato un secondo canale, come illustrato in figura 5-32 (b). Si noti che sono necessari due canali separati dall'host 3 al router E, in quanto vengono trasmessi due flussi separati.

Infine, in figura 5-32 (c), l'host 5 decide di guardare il programma trasmesso dall'host 1 e quindi fa una prenotazione. Innanzitutto, viene riservata banda dedicata fino al router H. Tuttavia, questo router si accorge che sta già ricevendo informazioni dall'host 1; la banda necessaria è quindi già stata riservata e non occorre prenotarne ancora. Si noti che gli host

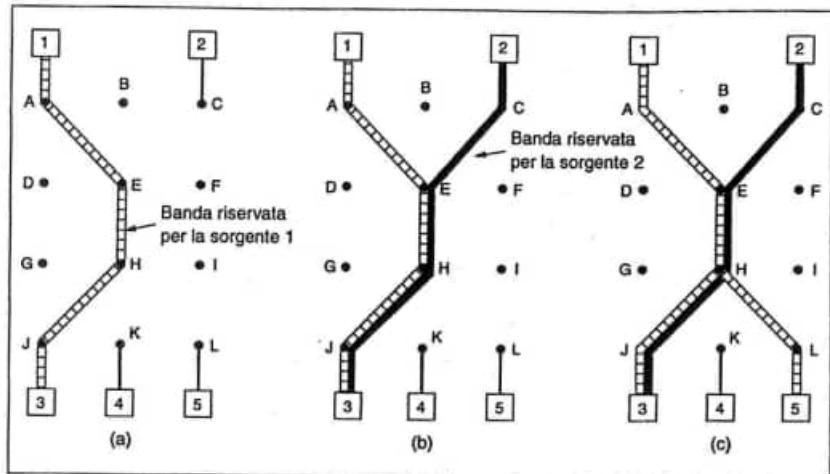


Fig. 5-32 (a) L'host 3 richiede un canale all'host 1. (b) L'host 3 richiede quindi un secondo canale all'host 2. (c) L'host 5 richiede un canale all'host 1.

3 e 5 potrebbero avere richiesto quantità di banda differenti (ad es., 3 ha una televisione in bianco e nero, e quindi non richiederebbe informazioni sui colori), per cui la capacità riservata dovrebbe essere grande abbastanza per soddisfare il ricevente più pigro.

Al momento di prenotare, un ricevente può (opzionalmente) specificare uno o più sorgenti da cui vuole ricevere. Può anche specificare se queste scelte sono fisse per la durata della prenotazione, oppure se il ricevente vuole tenere aperta la possibilità di cambiare sorgenti più tardi. I router utilizzano queste informazioni per ottimizzare la pianificazione dell'uso della banda. In particolare si fa in modo che due riceventi condividano lo stesso percorso se entrambi concordano di non cambiare sorgenti in seguito.

Nel caso completamente dinamico, questa strategia è motivata dal fatto che la prenotazione della banda non è accoppiata alla scelta della sorgente. Dopo che il ricevente ha prenotato una certa banda, può cambiare sorgente e conservare la porzione del cammino esistente che resta valida per la nuova sorgente. Ad esempio, se l'host 2 sta trasmettendo molti canali video, l'host 3 può scegliere tra essi senza cambiare la sua prenotazione: i router non si preoccupano di quali programmi il ricevente stia guardando.

5.4 Internetworking

Fino a ora, abbiamo assunto implicitamente l'esistenza di una singola rete omogenea, dove ogni macchina utilizza, in ognuno dei livelli, lo stesso protocollo.

Purtroppo, questa assunzione è estremamente ottimista. Esistono molti tipi diversi di rete, fra cui LAN, MAN e WAN. In ognuno dei livelli vengono utilizzati innumerevoli protocolli. In seguito descriveremo attentamente le problematiche che sorgono quando due o più reti vengono unite per formare un'internet.

Esiste un considerevole numero di controversie sulla seguente questione: l'attuale abbondanza di tipologie di rete è una condizione temporanea, che sparirà quando tutti comprendranno quanto magnifica sia (inserire la vostra rete preferita), oppure è una caratteristica inevitabile e permanente, destinata a durare? Avere reti differenti significa invariabilmente avere protocolli differenti.

A nostro parere esisterà sempre una grande varietà di reti distinte (e quindi di protocolli), per le seguenti ragioni. Innanzitutto, la base installata di reti distinte è vasta e in continua crescita. Quasi tutti i sistemi UNIX utilizzano TCP/IP. Molte società di grandi dimensioni hanno mainframe che utilizzano SNA. DEC sviluppa ancora DECnet. Le LAN di personal computer utilizzano spesso Novell NCP/IPX e AppleTalk. I sistemi ATM iniziano a diffondersi. Infine, spesso vengono utilizzati protocolli specializzati per reti di tipo satellitare, cellulare e a infrarossi. Questa tendenza continuerà per anni, grazie al grande numero di reti esistenti e poiché non tutti i vendoriti ritengono che sia nel loro interesse il fatto che i clienti possano migrare facilmente.

Secondo, con la diminuzione del prezzo di computer e reti, il livello in cui si prendono le decisioni si muove verso il basso. Molte società private hanno una politica per cui gli acquisti sopra un 1.000.000 di dollari devono essere approvati dai dirigenti di alto livello, gli acquisti sopra 100.000 dollari devono essere approvati dai dirigenti di medio livello, mentre gli acquisti sotto 100.000 dollari possono essere decisi dai capi dipartimento senza approvazioni dall'alto. Questo può facilmente portare a una situazione in cui l'ufficio contabilità installa una Ethernet, il dipartimento di ingegneria installa un token bus, mentre il dipartimento del personale installa un token ring.

Terzo, poiché reti differenti (ad es., ATM e senza filo) hanno tecnologie radicalmente differenti, non dovrebbe essere una sorpresa che in caso di nuovi sviluppi hardware, venga creato nuovo software per adattarsi a essi. Ad esempio, la casa media odierna è simile all'ufficio medio dieci anni fa: è piena di computer che non comunicano fra loro. Nel futuro, è probabile che il telefono, la televisione e altre apparecchiature siano tutte collegate in rete, in modo da poter essere controllate in remoto. Questa nuova tecnologia richiederà senza dubbio nuovi protocolli.

Si consideri il seguente esempio di come possano interagire reti differenti. In molte università, i dipartimenti di informatica e di ingegneria elettronica hanno le loro LAN, spesso differenti. Inoltre, il centro informatico dell'università possiede un mainframe e un supercomputer, il primo per i membri di facoltà umanistiche che non vogliono occuparsi della gestione dei loro computer, il secondo per i fisici che vogliono "macinare numeri". Come conseguenza di questa varietà di servizi e reti, è facile immaginare i seguenti scenari:

1. LAN-LAN: un informatico che scarica un file da ingegneria.
2. LAN-WAN: un informatico che spedisce posta a un fisico distante.
3. WAN-WAN: due poeti che si scambiano sonetti.
4. LAN-WAN-LAN: ingegneri presso università differenti che comunicano.

La figura 5-33 illustra questi quattro tipi di connessioni come linee tratteggiate. In ognuno

di questi casi, è necessario inserire una "scatola nera" come giunzione tra due reti, per gestire le conversioni necessarie quando i pacchetti si muovono da una rete a un'altra.

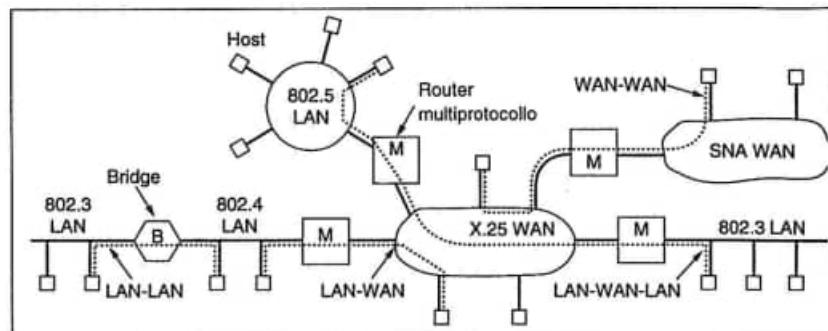


Fig. 5-33 Interconnessione di reti.

Il nome utilizzato per la scatola nera che connette due reti dipende dal livello al quale lavora. Alcuni dei nomi più comuni sono i seguenti (sebbene non ci sia molto accordo in quest'area).

Livello 1: i ripetitori copiano i bit individuali tra due segmenti di cavi.

Livello 2: i bridge memorizzano e inoltrano i frame data link tra LAN.

Livello 3: i router multiprotocollo inoltrano pacchetti fra reti differenti.

Livello 4: i gateway di trasporto connettono flussi di dati nel livello trasporto.

Oltre il 4: i gateway di applicazione permettono l'interconnessione oltre il livello 4.

Per comodità, a volte utilizzeremo il termine gateway per indicare qualsiasi dispositivo che connette due o più reti distinte.

I ripetitori (**repeater**) sono dispositivi di basso livello che amplificano o rigenerano segnali deboli. Sono necessari per fornire corrente a cavi lunghi. In 802.3, ad esempio, le proprietà temporali del protocollo MAC (il valore scelto di τ) permettono cavi lunghi fino a 2,5 km, mentre i chip ricetrasmettitori possono fornire potenza solo fino a 500 m. Una soluzione è quella di utilizzare ripetitori che estendono la lunghezza dei cavi dove si desidera.

A differenza dei ripetitori, che copiano i bit non appena arrivano, i **bridge** (ponti) sono dispositivi store-and-forward. Un bridge accetta un intero frame e lo passa al livello data link dove viene verificata la checksum. Quindi il frame viene spedito di nuovo al livello fisico per essere inoltrato su una rete differente. I bridge possono eseguire piccoli cambiamenti al frame prima di inoltrarlo, come aggiungere o cancellare alcuni campi del suo preamble. Poiché sono dispositivi del livello data link, non si occupano dei preamboli del livello 3 e superiori, e non possono fare cambiamenti o prendere decisioni che influiscono su di essi.

I **router multiprotocollo** sono concettualmente simili ai bridge, a parte il fatto che si

trovano nel livello rete. Essi prendono i pacchetti in arrivo da una linea e li rispediscono lungo un'altra, come fanno tutti i router; in questo caso però le linee possono appartenere a reti differenti e utilizzare protocolli differenti (ad es., IP, IPX o il protocollo senza connessione di OSI, CLNP). Come tutti i router, i router multiprotocollo operano a livello rete.

I **gateway di trasporto** realizzano una connessione tra due reti nel livello trasporto. Discuteremo di questa possibilità in seguito, quando parleremo dei circuiti virtuali concatenati.

Infine, i **gateway di applicazione** connettono due parti di un'applicazione nel livello application. Ad esempio, per spedire posta da una macchina Internet che utilizza il formato Internet per la posta a una casella postale ISO MOTIS, si potrebbe inviare il messaggio a un gateway postale. Il gateway postale scomporrebbe il messaggio, lo convertirebbe nel formato MOTIS e quindi lo inoltrerebbe lungo la seconda rete utilizzando i protocolli di rete e di trasporto utilizzati in essa.

Quando un gateway è collocato tra due WAN di due organizzazioni distinte, al limite di due nazioni distinte, la gestione congiunta di una workstation può dare origine a un sacco di guai. Per evitarli è possibile utilizzare un approccio leggermente differente. Il gateway viene effettivamente spezzato a metà e le due parti vengono connesse a un cavo. Ogni metà viene chiamata **half-gateway** e viene posseduta e gestita da uno degli amministratori di rete. L'intero problema di realizzare un gateway si riduce ad accordarsi su un protocollo comune da utilizzare nel cavo, un protocollo che sia neutrale e che non favorisca nessuno dei due lati. La figura 5-34 mostra sia gateway normali che half-gateway. Entrambi i tipi possono essere utilizzati in ogni livello (ad es., esistono anche half-bridge).

Detto questo, la situazione è più complessa in pratica di quanto non sia in teoria. Molti dispositivi nel mercato combinano funzionalità di bridge e di router. La caratteristica principale di un bridge puro consiste nell'esaminare i preamboli dei frame data link e nel non guardare o modificare i pacchetti del livello rete all'interno dei frame. Un bridge non può sapere e non deve preoccuparsi se il frame che sta inoltrando da una LAN 802.x a una 802.y contiene un pacchetto IP, IPX o CLNP.

Un router, al contrario, sa bene di essere un router IP, IPX, CLNP o tutti e tre insieme; esamina i rispettivi preamboli e prende decisioni in base agli indirizzi trovati in essi. D'altra parte, quando un router puro restituisce un pacchetto al livello data link, non sa e non vuole sapere se verrà trasportato in un frame Ethernet o in un frame token ring. Questa è una responsabilità del livello data-link.

Due sono le sorgenti della confusione presente in campo industriale. Innanzitutto, bridge e router non sono completamente differenti dal punto di vista funzionale. Entrambi accettano PDU in arrivo (Protocol Data Unit- Unità di dati del protocollo), esaminano qualche campo del preambolo, e prendono decisioni su dove mandare i PDU in base alle informazioni del preambolo e alle tabelle interne.

Inoltre, molti prodotti commerciali vengono venduti con l'etichetta sbagliata oppure combinano sia le funzionalità di bridge che di router. Ad esempio, i bridge con routing basato su sorgente non sono in realtà semplici bridge, poiché prevedono un livello di protocollo superiore al livello data link per assolvere ai loro compiti. Per una discussione illuminante su bridge e router, si veda il capitolo 12 di Perlman (1992).

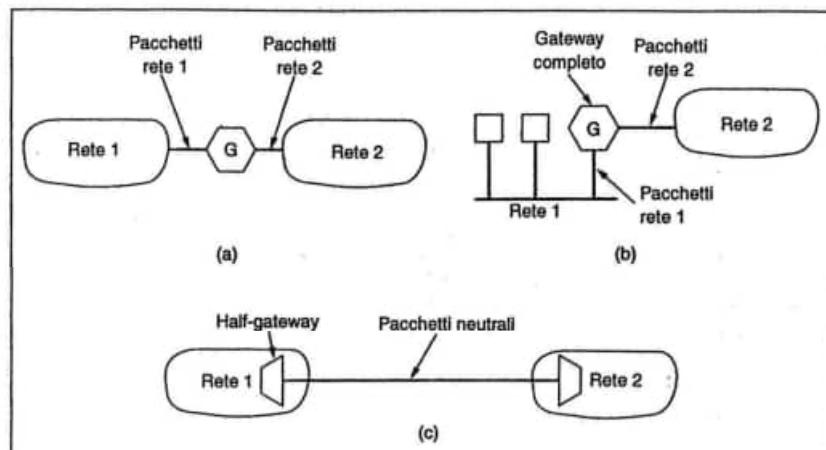


Fig. 5-34 (a) Un gateway normale tra due WAN. (b) Un gateway normale tra una LAN e una WAN. (c) Due half-gateway.

5.4.1 In cosa differiscono le reti

Le reti possono differire in molti aspetti. In figura 5-35 elenchiamo alcune delle differenze che si possono incontrare nel livello rete. È il tentativo di appianare queste differenze che rende l'internetworking più difficile dell'utilizzazione di una singola rete.

Caratteristica	Alcune possibilità
Servizio offerto	Orientamento alle connessioni vs senza connessioni
Protocolli	IP, IPX, CLNP, AppleTalk, DECnet, ecc.
Indirizzamento	Piatto (802) vs gerarchico (IP)
Multicasting	Presente o assente (broadcasting)
Dimensione dei pacchetti	Ogni rete ha il proprio massimo
Qualità del servizio	Presente o assente; varie tipologie
Gestione degli errori	Consegna affidabile, ordinata o non ordinata
Flusso di controllo	Sliding window, controllo di velocità, altro, o nessuno
Controllo di congestione	Secchio bucato, pacchetti regolatori, ecc.
Sicurezza	Regole di privacy, crittogramma ecc.
Parametri	Timeout differenti, specifiche di flusso, ecc.
Tariffazione	Per tempo di connessione, per pacchetto, per byte, nessuna

Fig. 5-35 Alcuni dei modi in cui le reti possono differire.

Quando pacchetti spediti da una sorgente in una rete devono transitare lungo una o più reti straniere prima di raggiungere la rete destinazione (che può essere anch'essa differente dalla rete sorgente), possono sorgere molti problemi nelle interfacce fra le reti. Ad esempio, quando i pacchetti di una rete orientata alla connessione devono transitare attraverso una rete senza connessioni, possono essere riordinati, un evento inatteso per il mittente e che trova impreparato il ricevente. In molti casi potrebbero essere necessarie conversioni di protocollo, che possono essere molto complesse nel caso la funzionalità richiesta non possa essere espressa. Potrebbero essere necessarie anche conversioni di indirizzo, e ciò può richiedere qualche tipo di elenco. Trasportare pacchetti di tipo multicast lungo una rete che non supporta il multicasting richiede la generazione di pacchetti separati per ogni destinazione.

Uno dei problemi principali è la differenza nelle dimensioni massime dei pacchetti che possono essere trasportati da reti distinte. Come si può trasportare un pacchetto di 8000 byte in una rete la cui dimensione massima è 1500 byte? Un ulteriore problema è rappresentato dalle diverse qualità di servizio; si consideri ad esempio un pacchetto con requisiti di consegna real time che attraversa una rete che non offre garanzie in tempo reale.

I controlli di errore, di flusso e di congestione differiscono spesso tra reti distinte. Molte applicazioni potrebbero guastarsi nel caso in cui sia la sorgente che la destinazione richiedessero la consegna dei pacchetti in sequenza e senza errori (comunque una delle reti intermedie scarta sempre i pacchetti non appena intravede la congestione all'orizzonte), oppure esiste la possibilità che i pacchetti vadano in giro per la rete per un sacco di tempo e quindi emergano all'improvviso per essere consegnati. Ulteriori problemi possono essere causati da meccanismi di sicurezza, scelte di parametri, regole di conteggio e leggi nazionali distinte.

5.4.2 Circuiti virtuali concatenati

Esistono fondamentalmente due stili di interconnessione: una concatenazione orientata alla connessione di reti basate su circuiti virtuali, oppure un'internet basata su datagram. In questo paragrafo e nel seguente verranno analizzati più in dettaglio. Nel modello basato su circuiti virtuali concatenati, mostrato in figura 5-36, una connessione con un host di una rete distante viene creata in modo simile al modo in cui normalmente vengono create le connessioni. La rete si accorge che la destinazione non è locale e stabilisce un circuito virtuale con il router più vicino alla rete di destinazione. Quindi costruisce un circuito virtuale da quel router a un "gateway" esterno (router multiprotocollo). Questo gateway memorizza l'esistenza del circuito virtuale nelle sue tabelle e inizia a stabilire un ulteriore circuito virtuale con un router della prossima rete. Questo processo continua fino a quando la destinazione non viene raggiunta.

Una volta che i pacchetti iniziano a viaggiare lungo il percorso, ogni gateway ritrasmette i pacchetti ricevuti, convertendo i formati dei pacchetti e i numeri di circuito virtuale a seconda delle necessità. Chiaramente, tutti i pacchetti di dati devono attraversare la stessa sequenza di gateway e quindi arrivano in ordine.

La caratteristica principale di questo approccio è la creazione di una sequenza di circuiti virtuali dalla sorgente attraverso uno o più gateway fino alla destinazione. Ogni gateway

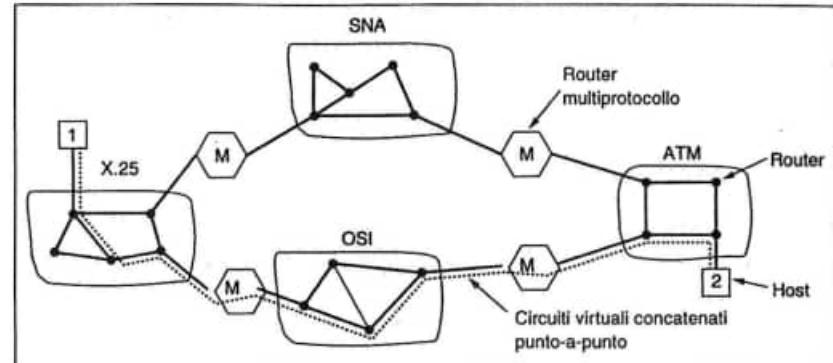


Fig. 5-36 Interconnessione di reti basata su circuiti virtuali concatenati.

mantiene alcune tabelle in cui registra quali circuiti passano attraverso di esso, dove devono essere instradati e qual è il nuovo numero di circuito virtuale.

Sebbene la figura 5-36 mostri una connessione realizzata attraverso un gateway normale, potrebbe essere realizzata ugualmente tramite half-gateway.

Il funzionamento di questo schema è migliore quando tutte le reti hanno approssimativamente le stesse proprietà. Ad esempio, se ogni rete garantisce la consegna affidabile dei pacchetti del livello rete (escludendo quindi crash lungo il percorso), il flusso dalla sorgente alla destinazione sarebbe anch'esso affidabile. In modo simile, se nessuna di esse garantisce la consegna affidabile, nemmeno la concatenazione dei circuiti virtuali sarebbe affidabile. D'altra parte, se la macchina sorgente è in una rete che garantisce la consegna affidabile, mentre una delle reti intermedie può perdere pacchetti, la concatenazione cambia fondamentalmente la natura del servizio.

I circuiti virtuali concatenati sono comuni anche nel livello trasporto. In particolare, è possibile costruire una connessione utilizzando, ad esempio, OSI, il quale termina presso un gateway, e avere una connessione TCP da quel gateway a quello successivo. In questo modo è possibile costruire un circuito virtuale end-to-end che attraversa reti e protocolli differenti.

5.4.3 Internetworking senza connessioni

Un modello di internet alternativo è quello basato su datagram, mostrato in figura 5-37. In questo modello, l'unico servizio che il livello rete offre al livello trasporto è la capacità di inserire datagram in una rete e sperare per il meglio. Non esiste alcuna nozione di circuito virtuale nel livello rete, figuriamoci poi il concetto di concatenazione di circuiti. Questo modello non richiede che tutti i pacchetti appartenenti a una connessione attraversino la stessa sequenza di gateway. In figura 5-37 si mostra come datagram dell'host 1 diretti all'host 2 possano prendere percorsi differenti lungo la rete. Le decisioni di routing vengono prese separatamente per ogni pacchetto, e possono dipendere dal traffico presente nel momento in cui il pacchetto viene spedito. Questa strategia può utilizzare più di un percorso e ottenere quindi una banda maggiore di quella del modello basato su

circuiti virtuali concatenati. D'altra parte, non esistono garanzie che i pacchetti arrivino alla destinazione in ordine, già assumendo che arrivino.

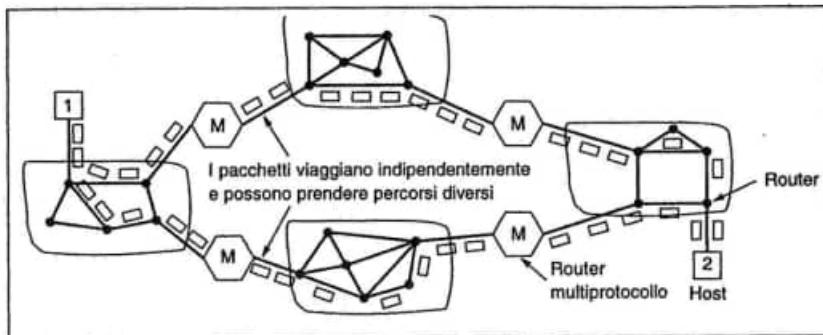


Fig. 5-37 Un'internet senza connessioni.

Il modello di figura 5-37 non è così semplice come sembra. Innanzitutto, se ogni rete ha il proprio protocollo di livello rete, non è possibile far transitare un pacchetto di una rete sopra un'altra. Si potrebbe immaginare che i router multiprotocollo cerchino effettivamente di tradurre da un formato a un altro, ma a meno che i due formati siano parenti stretti con gli stessi campi di informazione, tale conversione sarà sempre incompleta e spesso votata al fallimento. Per questo motivo si tenta raramente la conversione.

Un secondo e più serio problema è l'indirizzamento. Si immagini un caso semplice: un host in Internet sta cercando di spedire un pacchetto IP a un host appartenente a una rete OSI contigua. Il protocollo OSI basato su datagram, CLNP, è basato su IP e gli somiglia a tal punto che una conversione potrebbe funzionare. Il problema è il fatto che il preambolo dei pacchetti IP contiene l'indirizzo Internet della destinazione lungo 32 bit, mentre gli host OSI non hanno indirizzi Internet a 32 bit, ma utilizzano indirizzi decimali simili ai numeri telefonici.

Per consentire a un router multiprotocollo di convertire i formati, qualcuno dovrebbe assegnare un indirizzo Internet a 32 bit a ciascuno degli host OSI. Portato all'estremo, questo approccio significherebbe assegnare un indirizzo Internet a ogni macchina nel mondo con cui un nodo Internet potrebbe desiderare di parlare. Significherebbe anche l'assegnazione di un indirizzo OSI a ognuna delle macchine nel mondo con cui un host OSI vorrebbe poter comunicare. Lo stesso problema si presenta con ogni altro spazio di indirizzamento (SNA, Apple talk ecc.). Questi problemi sono insormontabili; inoltre, qualcuno dovrebbe mantenere un database che "mappi" qualsiasi cosa con qualsiasi altra. Un'altra idea è quella di progettare un pacchetto "internet" universale e fare in modo che tutti i router siano in grado di riconoscerlo. Questo approccio coincide con l'idea di IP – un pacchetto progettato per essere trasportato attraverso molte reti. L'unico problema è dato dall'esistenza di altri formati universali come IPX e CLNP, che quindi sono tutto tranne che universali. Ottenere che tutti concordino su un singolo formato è semplicemente impossibile.

A questo punto, ricapitoliamo sommariamente i due modi per affrontare il problema di internetworking. Il modello basato su circuiti virtuali concatenati ha essenzialmente gli stessi vantaggi di utilizzare i circuiti virtuali all'interno di una rete singola: i buffer possono essere prenotati in anticipo, è possibile garantire l'ordinamento, possono essere utilizzati preamboli più brevi, e i problemi causati da pacchetti duplicati in ritardo possono essere evitati.

Ha anche alcuni svantaggi, come ad esempio la necessità di spazio nelle tabelle dei router per ognuna delle connessioni aperte, la mancanza di percorsi alternativi per evitare le aree congestionate e la vulnerabilità verso i guasti dei router lungo il cammino. Ha anche lo svantaggio che è difficile (se non impossibile) da implementare se una delle reti coinvolte è una rete a datagram non affidabile.

Le proprietà dell'approccio basato su datagram sono le stesse di quelle delle reti datagram: maggiore potenzialità di congestione, ma anche maggiore potenzialità di adattarsi a essa, robustezza nel caso di guasti ai router, e preamboli più lunghi. In un'internet, è possibile utilizzare diversi algoritmi di routing adattivo, proprio come nel caso di una singola rete basata su datagram.

Uno dei vantaggi principali dell'approccio basato su datagram all'internetworking è il fatto di poter essere utilizzato anche su reti che non usano i circuiti virtuali al loro interno. Molte LAN, reti mobili (ad es. flotte aeree e navali), e anche alcune WAN cadono in questa categoria. Quando un'internet include una di queste reti, possono sorgere seri problemi se la strategia di internetworking è basata sui circuiti virtuali.

5.4.4 Tunneling

Gestire il caso generale di interconnessione di due reti differenti è eccessivamente complesso. Tuttavia, esiste un caso speciale che è trattabile, e cioè quando gli host sorgente e destinazione appartengono allo stesso tipo di rete, ma esistono reti differenti lungo il cammino. Come esempio, si pensi a un banca internazionale con una Ethernet basata su TCP/IP a Parigi, una Ethernet basata su TCP/IP a Londra e una WAN PTT nel mezzo, come mostrato in figura 5-38.

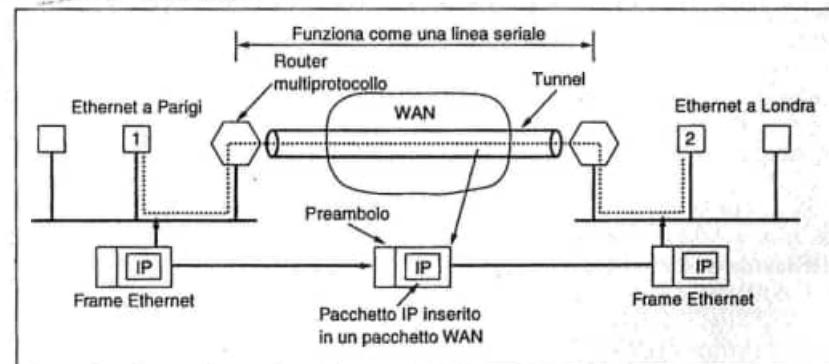


Fig. 5-38 Tunneling di un pacchetto da Parigi a Londra.

La soluzione a questo problema è una tecnica chiamata **tunneling** (incapsulamento). Per spedire un pacchetto IP all'host 2, l'host 1 costruisce un pacchetto contenente l'indirizzo IP dell'host 2, lo inserisce in un frame Ethernet indirizzato al router multiprotocollo di Parigi, e lo inserisce nella Ethernet. Quando il router multiprotocollo riceve il frame, rimuove il pacchetto IP, lo inserisce nel campo contenuto del pacchetto della rete WAN, e invia quest'ultimo all'indirizzo WAN del router multiprotocollo di Londra. Quando arriva, il router di Londra rimuove il pacchetto IP e lo spedisce all'host 2 all'interno di 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 estremo a un altro della rete, sicuro nella sua bella scatola. Non deve preoccuparsi di sapere nulla della WAN, né lo devono fare gli host in entrambe le Ethernet. Solo il router multiprotocollo deve essere in grado di comprendere i pacchetti IP e WAN. In effetti, l'intera tratta dal centro di un router multiprotocollo al centro dell'altro router agisce come una linea seriale.

Un'analogia può chiarire il concetto di tunneling. Si consideri una persona che sta guidando la sua auto da Parigi a Londra. In Francia, la macchina si muove autonomamente; quando raggiunge la Manica, viene caricata su un treno ad alta velocità e trasportata in Inghilterra attraverso il tunnel sotto la Manica (che non può essere percorso direttamente dalle auto). In effetti, l'auto viene trasportata come merce, come illustrato in figura 5-39. Alla fine, l'auto viene lasciata sulle strade inglesi e riprende a muoversi autonomamente. Il tunneling dei pacchetti attraverso una rete straniera lavora allo stesso modo.

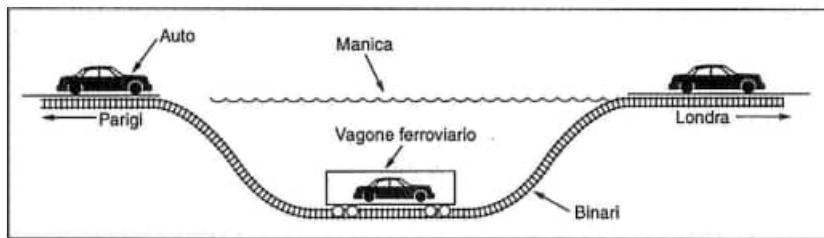


Fig. 5-39 Incanalamento di un'auto dalla Francia all'Inghilterra.

5.4.5 Routing all'interno di Internet

Il routing all'interno di Internet è simile al routing all'interno di una singola rete, ma con alcune complicazioni aggiuntive. Si consideri ad esempio l'internet di figura 5-40 (a) nella quale cinque reti sono connesse da sei router multiprotocollo. Disegnare un grafico di questa situazione è reso più complicato dal fatto che ogni router multiprotocollo può accedere direttamente (cioè spedire pacchetti) a ogni altro router compreso in qualsiasi rete alla quale è connesso. Ad esempio, B in figura 5-40 (a) può accedere direttamente ad A e C attraverso la rete 2 e a D attraverso la rete 3. Questo porta al grafo di figura 5-40 (b).

Una volta che il grafo è stato costruito, all'insieme dei router multiprotocollo possono essere applicati gli algoritmi di routing conosciuti, come il distance vector routing e il

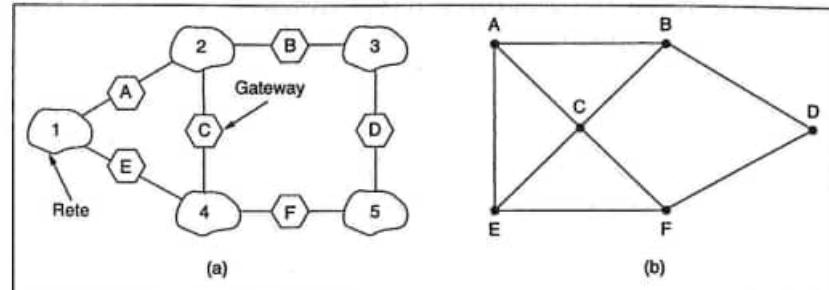


Fig. 5-40 (a) Un'internet. (b) Un grafo dell'internet.

link state routing. Questo dà origine a un algoritmo di routing a due livelli: all'interno di ogni rete viene utilizzato un **interior gateway protocol** (protocollo di routing tra gateway interni), mentre fra le reti viene utilizzato un **exterior gateway protocol** (protocollo di routing tra gateway esterni), dove "gateway" è un termine obsoleto per "router". In effetti, poiché ogni rete è indipendente, ognuna di esse potrebbe utilizzare un algoritmo differente. Poiché ogni rete in un'internet è indipendente dalle altre, spesso ci si riferisce a essa come a un **sistema autonomo** (Autonomous System, o AS).

Un tipico pacchetto internet parte dalla propria LAN indirizzato al router multiprotocollo locale (nel preambolo del livello MAC). Dopo esservi giunto, il protocollo del livello rete decide a quale router multiprotocollo inoltrare il pacchetto, usando le proprie tabelle di routing. Se quel router può essere raggiunto utilizzando il protocollo di rete nativo per il pacchetto, questo viene inoltrato direttamente. Altrimenti viene incapsulato nel protocollo richiesto dalla rete che viene utilizzata. Questo processo viene ripetuto finché il pacchetto non raggiunge la rete di destinazione.

Una delle differenze tra il routing di tipo internet e il routing di tipo intranet è il fatto che spesso il routing di tipo internet richiede di attraversare confini nazionali. Entrano in gioco all'improvviso svariate leggi, come ad esempio la severa legge svedese sulla privacy (sull'esportazione di dati personali di cittadini svedesi). Un altro esempio è la legge canadese che impone che il traffico originato in Canada e destinato in Canada non possa lasciare il paese. Questa legge significa che il traffico da Windsor (Ontario) diretto a Vancouver non può essere instradato attraverso la vicina Detroit.

Un'altra differenza tra routing interno e routing esterno sono i costi. All'interno di una singola rete viene normalmente applicato un unico algoritmo di tariffazione. Ciò nonostante, reti differenti possono appartenere a diverse amministrazioni, e un percorso può essere meno costoso di un altro. In modo simile, la qualità del servizio offerto da reti distinte può essere differente, e questa potrebbe essere una ragione per scegliere una via invece di un'altra.

In una rete di grandi dimensioni, scegliere la strada migliore può essere un'operazione molto costosa in termini di tempo. Estrin et al. (1992) hanno proposto di gestire questo problema precalcolando i percorsi per le coppie (sorgente, destinazione) più popolari, registrandoli in un database da consultare al momento di selezionare il percorso.

5.4.6 Frammentazione

Ogni rete impone qualche dimensione massima per i suoi pacchetti. Esistono molte cause per questa limitazione, fra le quali:

1. Hardware (ad es., l'ampiezza di uno slot di trasmissione TDM).
2. Sistema operativo (ad es., tutti i buffer sono di 512 byte).
3. Protocolli (ad es., il numero di bit nel campo lunghezza del pacchetto).
4. Conformità a qualche standard (inter) nazionale.
5. Desiderio di ridurre a un certo livello le ritrasmissioni indotte dagli errori.
6. Desiderio di evitare che un pacchetto occupi troppo a lungo un canale.

La conseguenza di tutti questi fattori è che i progettisti delle reti non sono liberi di scegliere come vogliono la dimensione massima dei pacchetti. La dimensione massima spazia dai 48 byte delle celle ATM ai 65.515 byte dei pacchetti IP, sebbene nei livelli superiori la dimensione massima sia spesso maggiore.

Ovviamente sorge un problema quando un pacchetto grande vuole viaggiare attraverso una rete in cui la dimensione massima prevista dei pacchetti è inferiore alla sua. Una soluzione è assicurarsi che il problema non si presenti mai. In altre parole, l'internet dovrebbe usare un algoritmo di routing che eviti di spedire pacchetti attraverso reti che non sono in grado di gestirli. Tuttavia, questa soluzione non è sempre applicabile. Cosa succede se il pacchetto originato dalla sorgente è troppo grande per essere gestito dalla rete destinazione? È difficile che l'algoritmo di routing possa evitare di passare per la destinazione.

Fondamentalmente, l'unica soluzione al problema è consentire ai gateway di spezzare i pacchetti in **frammenti**, spedendo ogni frammento in un pacchetto internet separato. Tuttavia convertire un oggetto grande in frammenti piccoli è considerevolmente più facile del processo inverso (i fisici hanno anche dato un nome a questo effetto: la seconda legge della termodinamica). Anche le reti a commutazione di pacchetto hanno difficoltà a rimettere insieme i frammenti.

Esistono due strategie opposte per ricostruire i pacchetti originali a partire dai frammenti. La prima strategia consiste nel rendere la frammentazione causata da una rete "a pacchetti piccoli" trasparente a qualsiasi rete successiva attraverso la quale deve passare il pacchetto nella sua strada verso la destinazione finale. Questa possibilità viene mostrata in figura 5-41 (a). Quando un pacchetto troppo grande giunge a un gateway, questo lo spezza in frammenti. Ogni frammento viene indirizzato a un certo gateway di uscita, dove i pezzi vengono ricomposti. In questo modo, il passaggio attraverso la rete a pacchetti piccoli viene reso trasparente. Le reti successive non si rendono conto che è avvenuta una frammentazione. Ad esempio, le reti ATM posseggono un hardware speciale per frammentare in modo trasparente i pacchetti in celle e quindi riassemblare le celle in pacchetti. Nel mondo ATM, la frammentazione è chiamata segmentazione: il concetto è lo stesso, sebbene alcuni dettagli siano differenti.

La frammentazione trasparente è semplice, ma presenta alcuni problemi. Innanzitutto, il

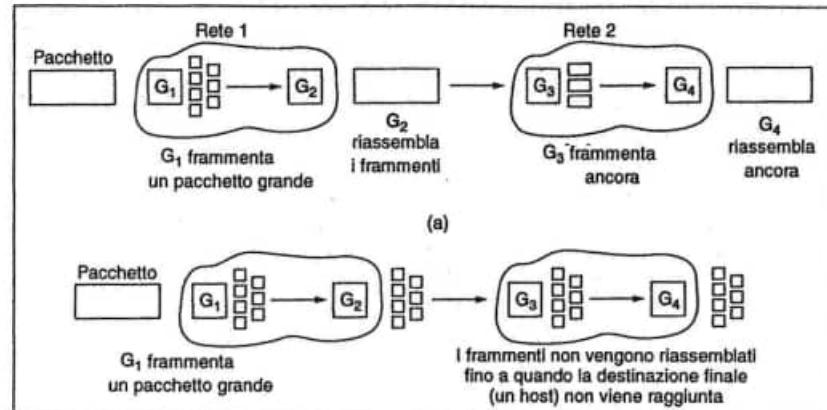


Fig. 5-41 (a) Frammentazione trasparente. (b) Frammentazione non trasparente.

gateway di uscita deve sapere quando ha ricevuto tutti i pezzi; quindi in ogni pacchetto deve essere incluso un campo contatore o un bit "fine del pacchetto". D'altro canto, tutti i pacchetti devono uscire attraverso lo stesso gateway. Non consentire che i frammenti seguano percorsi distinti verso la destinazione finale può causare perdite di prestazioni. Un ultimo problema riguarda il carico aggiuntivo richiesto per riassemblare e frammentare ripetutamente un pacchetto grande che passa attraverso una serie di reti a pacchetti piccoli.

L'altra strategia di frammentazione consiste nell'evitare di ricombinare i frammenti a ogni gateway intermedio. Una volta che il pacchetto è stato frammentato, ogni frammento viene trattato come se fosse un pacchetto originale. Tutti i frammenti vengono fatti passare attraverso il gateway di uscita (o i gateway), come mostrato in figura 5-41 (b). La ricombinazione avviene solo all'host destinazione.

Anche la frammentazione non trasparente non è esente da problemi. Ad esempio, richiede che *ogni* host sia in grado di eseguire il riassemblaggio. Un ulteriore problema è il fatto che quando un pacchetto grande viene frammentato, l'overhead totale aumenta, in quanto ogni frammento deve avere un preambolo. Mentre nel primo metodo questo overhead scompare non appena si esce da una rete a pacchetti piccoli, in questo metodo l'overhead rimane per il resto del viaggio. Un vantaggio di questo sistema, tuttavia, è che ora possono essere utilizzati più gateway di uscita e si possono ottenere prestazioni migliori. Naturalmente, se viene usato il modello basato su circuiti virtuali concatenati, questo vantaggio scompare.

Quando un pacchetto viene frammentato, i frammenti devono essere numerati in modo che il flusso di dati originali possa essere ricostruito. Un metodo per numerare i frammenti consiste nell'utilizzare un albero. Se il pacchetto 0 deve essere suddiviso, i pezzi vengono chiamati 0.0, 0.1, 0.2 ecc. Se in seguito gli stessi frammenti devono essere frammentati, i pezzi vengono numerati 0.0.0, 0.0.1, 0.0.2..., 0.1.0, 0.1.1, 0.1.2 ecc. Se nel preambolo viene riservato abbastanza spazio per il caso peggiore e non vengono generati duplicati,

questo schema è sufficiente per assicurare che tutti i pezzi possano essere riassemblati correttamente dalla destinazione, non importa l'ordine in cui arrivano. Ciò nonostante, se anche una sola rete perde o scarta pacchetti, è necessaria la ritrasmissione dell'intero pacchetto, con effetti sgradevoli per il sistema di numerazione. Si supponga che un pacchetto di 1024 bit venga inizialmente frammentato in quattro frammenti di dimensione uguali, 0,0, 0,1, 0,2 e 0,3. Il frammento 0,1 viene perso, mentre le altre parti arrivano alla destinazione. Prima o poi la sorgente andrà in timeout e ritrasmetterà di nuovo il pacchetto originale. Solo che questa volta il percorso passa attraverso una rete con limite di 512 bit, e quindi vengono generati due frammenti. Quando il nuovo frammento 0,1 arriva alla destinazione, il ricevente penserà che nessuno dei quattro pezzi manchi all'appello e ricostruirà il pacchetto in modo non corretto.

Un sistema di numerazione completamente diverso richiede che il protocollo di internetworking definisca una dimensione elementare dei frammenti abbastanza piccola da poter passare attraverso qualsiasi rete. Quando un pacchetto viene frammentato, tutti i pezzi sono uguali alla dimensione elementare del frammento eccetto l'ultimo, che può essere più piccolo. Per ragioni di efficienza, un pacchetto internet può contenere molti frammenti. Il preambolo internet deve contenere il numero originale del pacchetto e il numero del primo frammento elementare contenuto nel pacchetto. Come al solito, deve esserci un bit che indichi che l'ultimo frammento elementare contenuto nel pacchetto internet è l'ultimo del pacchetto originario.

Questo approccio richiede due campi di sequenza nel preambolo internet: il numero originario del pacchetto e il numero di frammento. Esiste chiaramente un compromesso tra la dimensione elementare dei frammenti e il numero di bit nell'identificatore di frammento. Poiché si assume che la dimensione elementare dei frammenti possa essere accettata da ogni rete, frammentazioni successive di un pacchetto internet contenente molti pacchetti non danno origine a problemi. Il caso limite consiste nell'utilizzare un frammento elementare grande quanto un singolo bit o byte, con l'identificatore di frammento uguale alla posizione del bit o del byte all'interno del pacchetto originario, come mostrato in figura 5-42.

Alcuni protocolli internet portano ancora più in là questo metodo e considerano l'intera trasmissione su un circuito virtuale come un gigantesco pacchetto, in modo che ogni frammento contenga la posizione assoluta nel pacchetto del primo byte del frammento. Altre problematiche legate alla frammentazione vengono discusse in Kent, Mogul (1987).

5.4.7 Firewall

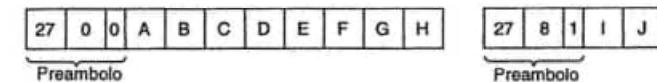
La capacità di connettere qualsiasi computer nel mondo con qualsiasi altro è sicuramente un fatto positivo, ma con alcuni risvolti negativi. Per persone a casa, vagabondare per Internet è molto divertente. Per gli esperti di sicurezza delle società private è un incubo. Gran parte delle società posseggono grandi quantità di informazioni confidenziali on-line (segreti commerciali, piani di sviluppo dei prodotti, strategie di marketing, analisi finanziarie). La divulgazione di queste informazioni ai rivali potrebbe avere conseguenze disastrose.

Oltre al pericolo che le informazioni escano, c'è anche il pericolo che le informazioni

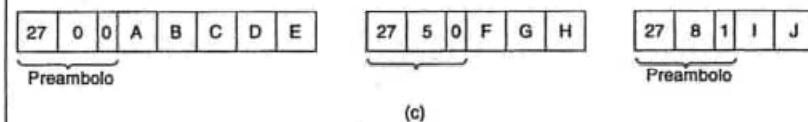
Numero del primo frammento elementare in questo pacchetto



(a)



(b)



(c)

Fig. 5-42 Frammentazione quando la dimensione elementare dei dati è di 1 byte. (a) Pacchetto originale, contenente 10 byte. (b) I frammenti dopo essere passati attraverso una rete con dimensione massima dei pacchetti di 8 byte. (c) I frammenti dopo essere passati attraverso un gateway con dimensione 5.

entrino. In particolare, virus e altri agenti nocivi digitali (Kaufman et al., 1995) possono infrangere la sicurezza, distruggere dati preziosi e far sprecare un sacco di tempo agli amministratori che cercano di ripulire il caos lasciato. Spesso sono importati da impiegati distratti che vogliono giocare a qualche nuovo, ingegnoso gioco.

Conseguentemente, sono necessari dei meccanismi per conservare i bit "buoni" all'interno e lasciare i bit "cattivi" all'esterno. Un metodo è la codifica critografica. Questo approccio protegge i dati scambiati tra siti sicuri. Verrà analizzato nel capitolo 7. Tuttavia, la codifica critografica non può impedire l'ingresso agli agenti nocivi. Per realizzare questo obiettivo, abbiamo bisogno di dare un'occhiata ai firewall (Chapman, Zwicky, 1995; Cheswick, Belowin, 1994).

I **firewall** (muro di fuoco) sono un moderno adattamento del vecchio principio di sicurezza medioevale: costruire un profondo fossato attorno al castello. Questa progettazione fa in modo che chiunque entri o esca dal castello passi attraverso un singolo ponte levatoio, dove può essere ispezionato dalla polizia di I/O. È possibile utilizzare lo stesso accorgimento con le reti: una società può avere diverse LAN connesse in qualsiasi modo, ma tutto il traffico da o per la società deve passare attraverso un ponte levatoio elettronico (un firewall), come mostrato in figura 5-43.

Il firewall in questa configurazione ha due componenti: due router che eseguono il filtraggio dei pacchetti e un gateway di applicazione. Esistono anche configurazioni più semplici, ma il vantaggio di questa progettazione è il fatto che ogni pacchetto deve

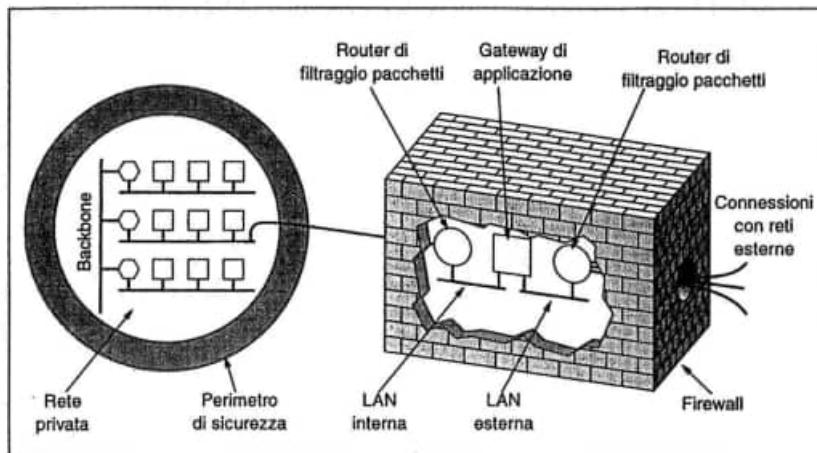


Fig. 5-43 Un firewall che consiste di due filtri per pacchetti e di un gateway a livello applicativo.

transitare attraverso due filtri e un gateway a livello applicativo per entrare o uscire. Non esistono altre vie. I lettori che pensano che un singolo controllo di sicurezza sia sufficiente, chiaramente non hanno volato di recente con una compagnia di linea internazionale.

Ogni filtro per pacchetti è un router standard equipaggiato con qualche funzionalità extra, che fa in modo che ogni pacchetto in entrata o in uscita venga ispezionato. I pacchetti che soddisfano certi criteri vengono lasciati passare normalmente. Quelli che falliscono il test vengono scartati.

In figura 5-43, è probabile che il filtro per pacchetti sulla LAN interna verifichi i pacchetti in uscita, mentre quello nella LAN esterna verifichi i pacchetti in entrata. I pacchetti che attraversano il primo ostacolo passano al gateway per ulteriori esami. Il motivo per cui vengono messi due filtri per pacchetti su LAN differenti è garantire che nessun pacchetto entri o esca senza essere passato attraverso il gateway di applicazione; non esiste alcun percorso che lo eviti.

Generalmente, i filtri per pacchetti sono basati su tabelle configurate dall'amministratore di sistema. Queste tabelle elencano le sorgenti e le destinazioni che sono accettabili, quelle che sono vietate e le regole standard su cosa fare dei pacchetti che provengono o sono diretti ad altre macchine.

Nel caso diffuso di un ambiente UNIX, le sorgenti e le destinazioni consistono in un indirizzo IP e in una porta. Le porte indicano quali sono i servizi desiderati. Ad esempio, la porta 23 è per Telnet, la porta 79 è per Finger, la porta 119 è per le news USENET. Una società potrebbe bloccare i pacchetti in entrata da tutti gli indirizzi IP combinati con una di queste porte. In questo modo, nessuno da fuori potrebbe entrare via Telnet, oppure cercare informazioni sugli utenti utilizzando Finger. Inoltre, questo impedirebbe agli impiegati di passare tutto il giorno a leggere le news USENET.

Bloccare i pacchetti in uscita è più complesso, in quanto sebbene gran parte dei siti si attengano alle convenzioni standard di identificazione delle porte, non sono forzati a farlo. Inoltre, per alcuni servizi importanti quali FTP (File Transfer Protocol – protocollo di trasferimento file), i numeri delle porte vengono assegnati dinamicamente. Infine, se bloccare le connessioni TCP è difficile, bloccare i pacchetti UDP è anche peggio, in quanto si conosce a priori molto poco di ciò che vogliono fare. Molti filtri per pacchetti bloccano tutto il traffico UDP.

La seconda parte del meccanismo di firewall è il **gateway di applicazione**. Piuttosto che guardare i pacchetti nudi e crudi, questo gateway funziona a livello applicazione. Ad esempio, si potrebbe creare un gateway per la posta, che esamina ogni messaggio in entrata o in uscita. Per ognuno di essi si decide se trasmetterlo o scartarlo in base ai campi del preambolo, alla dimensione del messaggio, o anche del contenuto (ad es., in una installazione militare, la presenza di parole come "nucleare" o "bomba" potrebbe causare qualche azione speciale).

Le installazioni sono libere di creare uno o più gateway di applicazione per applicazioni specifiche, ma non è raro che organizzazioni sospette permettano la posta elettronica in entrata e in uscita, e forse anche l'uso del World Wide Web, ma escludano qualsiasi altro tipo di traffico perché troppo rischioso. Combinato con la codifica critografica e il filtraggio dei pacchetti, questo meccanismo offre una certa quantità di sicurezza al prezzo di qualche inconveniente.

Una nota finale concerne le comunicazioni senza filo e i firewall. È facile concepire un sistema che sia in teoria completamente sicuro, ma che in pratica faccia acqua da tutte le parti. Questa situazione può presentarsi se alcune delle macchine sono senza filo e utilizzano comunicazioni radio, che aggirano i firewall in entrambi sensi.

5.5 Il livello rete in Internet

Dal punto di vista del livello rete, Internet può essere vista come una collezione di sottoreti o **sistemi autonomi** (Autonomous System, o AS) connessi insieme. Non esiste alcuna struttura reale, anche se possono esistere molte backbone (dorsali), basate su canali ad alta capacità e router veloci. A questi backbone vengono connesse le reti regionali (di medio livello); collegate alle reti regionali vi sono le LAN di centinaia di università, società e fornitori di servizi Internet. Uno schema di questa organizzazione quasi gerarchica è mostrata in figura 5-44.

La colla che tiene unita Internet è il protocollo del livello rete, IP (Internet Protocol). A differenza di gran parte dei protocolli del livello rete più vecchi, è stato progettato fin dall'inizio con l'internetworking in mente. Un buon modo per pensare il livello rete è il seguente. Il suo compito è fornire un modo best-efforts per trasportare datagram dalla sorgente alla destinazione, indipendentemente dall'esistenza di reti intermedie lungo il percorso e dal fatto che queste macchine appartengono oppure no alla stessa destinazione. Il sistema di comunicazione di Internet funziona nel modo seguente. Il livello trasporto riceve un flusso di dati e lo frammenta in datagram. In teoria, i datagram potrebbero arrivare fino a 64 Kbyte, ma in pratica sono lunghi circa 1500 byte. Ogni datagram viene trasmesso attraverso Internet, possibilmente frammentato in segmenti più piccoli durante

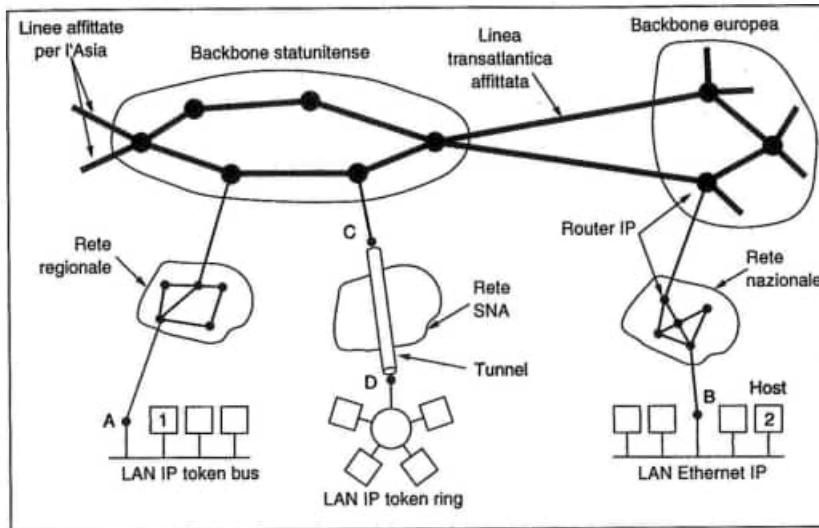


Fig. 5-44 Internet come una collezione interconnessa di reti.

il percorso. Quando finalmente tutti i pezzi raggiungono la destinazione, vengono riassemblati dal livello rete nel datagram originale. Questo datagram viene poi passato al livello trasporto, che lo inserisce nel flusso di input del processo ricevente.

5.5.1 Il protocollo IP

Il luogo appropriato per iniziare il nostro studio del livello rete di Internet è il formato dei datagram IP. Un datagram IP consiste in un preambolo e di una parte testo. Il preambolo ha una parte fissa di 20 byte e una parte opzionale di lunghezza variabile. Il formato del preambolo è mostrato in figura 5-45. Viene trasmesso in ordine big endian: da sinistra a destra, con il bit più significativo del campo *Version* come primo bit. (Lo SPARC è di tipo big endian; il Pentium è di tipo little endian). Sulle macchine little endian, è necessaria una conversione software sia in trasmissione che in ricezione.

Il campo *Version* (versione) descrive la versione del protocollo utilizzato dal datagram. Includendo la versione in ognuno dei datagram, diviene possibile fare in modo che la transizione fra diverse versioni duri dei mesi, o anche degli anni, con alcuni nodi che utilizzano la vecchia versione e altri che si servono della nuova.

Poiché la lunghezza del preambolo non è costante, in esso è presente un campo chiamato *IHL* che contiene la lunghezza del preambolo stesso, in parole di 32 bit. Il valore minimo è 5, che si applica quando non è presente nessuna opzione. Il valore massimo di questo campo a 4 bit è 15, il che limita il preambolo a 60 byte, e quindi il campo opzionale a 40 byte. Per qualche opzione, per esempio quella che memorizza il cammino percorso da un pacchetto, 40 byte sono troppo pochi, e ciò rende inutile il campo opzionale.

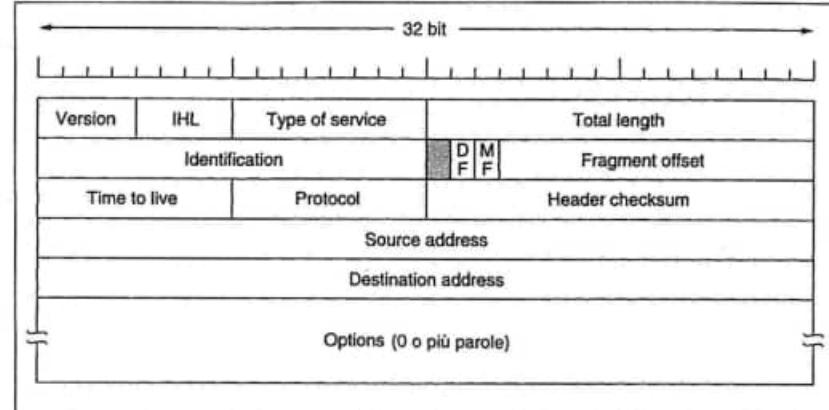


Fig. 5-45 Il preambolo IP (Internet Protocol).

Il campo *Type of service* (tipo di servizio) consente agli host di comunicare alla rete il tipo di servizio desiderato. Esistono varie combinazioni di affidabilità e velocità. Per la voce digitalizzata, la ricezione rapida è preferibile alla ricezione accurata. Per il trasferimento file, la trasmissione priva di errori è più importante della trasmissione rapida. Il campo contiene (da sinistra a destra) il campo a 3 bit *Precedence* (precedenza), 3 flag (*D*, *T* ed *R*), e 2 bit inutilizzati. Il campo *Precedence* è una priorità, da 0 (normale) a 7 (pacchetto di gestione della rete). I 3 bit di flag permettono all'host di specificare a cosa esso sia più interessato dell'insieme { ritardo, capacità di trasmissione, affidabilità }. In teoria, questi campi permettono ai router di fare scelte, ad esempio tra un canale satellitare ad alta capacità ma con molto ritardo, e una linea affittata a bassa capacità ma con basso ritardo. In pratica, i router attuali ignorano completamente il campo *Type of Service*.

Il campo *Total Length* (lunghezza totale) riguarda l'intero datagram (sia il preambolo che i dati). La lunghezza massima è 65.535 byte. Al momento, questo limite superiore è tollerabile, ma con le future reti ad alta velocità saranno necessari datagram più grandi. Il campo *Identification* (identificazione) è necessario per permettere all'host destinazione di determinare a quale datagram appartiene un frammento appena arrivato. Tutti i frammenti di un datagram contengono lo stesso valore *Identification*.

Seguono quindi 1 bit inutilizzato e due campi da 1 bit. *DF* sta per "Don't Fragment" (non frammentare), e ordina ai router di non frammentare i datagram poiché la destinazione non è capace di rimettere insieme i pezzi. Ad esempio, quando un computer viene acceso, la sua ROM potrebbe chiedere la spedizione di un'immagine di memoria come singolo datagram. Marcando il datagram con il bit *DF*, il mittente sa che verrà consegnato tutto intero, anche se questo significa evitare una rete a pacchetti piccoli sul cammino migliore e scegliere un percorso subottimale. Tutte le macchine devono essere in grado di accettare frammenti di 576 byte o meno.

MF sta per "More Fragments" (ulteriori frammenti). A parte l'ultimo, tutti i frammenti hanno questo bit uguale a 1. È necessario per sapere quando sono arrivati tutti i frammenti di un datagram.

Il *Fragment offset* (scostamento del frammento) indica in quale posizione del datagram corrente si trova questo frammento. Tutti i frammenti eccetto l'ultimo devono essere multipli di 8 byte, l'unità elementare di frammentazione. Poiché sono presenti 13 bit, esiste un massimo di 8192 frammenti per datagram; ciò implica una lunghezza massima del datagram pari a 65.536 byte, uno in più del campo *Total length*. Il campo *Time to live* (tempo da vivere) è un contatore utilizzato per limitare la durata della vita dei pacchetti. Si suppone che il tempo venga calcolato in secondi, consentendo una vita massima di 255 s. Deve essere decrementato a ogni salto e si suppone che ciò avvenga più volte quando resta nella coda di un router per molto tempo. In pratica, conta i salti. Quando il contatore raggiunge lo 0, il pacchetto viene scartato e viene restituito alla sorgente un pacchetto di avvertimento. Questo espediente evita che i datagram restino nella rete per sempre (un evento possibile nel caso in cui le tabelle di routing siano corrotte).

Quando il livello rete ha assemblato un datagram completo, ha bisogno di sapere cosa deve farne. Il campo *Protocol* (protocollo) indica a quale processo del trasporto deve essere consegnato. Fra le possibilità ci sono TCP, UDP e alcuni altri. La numerazione dei protocolli è globale in tutta Internet ed è definita nell'RFC 1700.

L'*Header checksum* (checksum del preambolo) verifica solamente il preambolo. Tale checksum viene utilizzata per scoprire errori generati da errori di memoria all'interno dei router. L'algoritmo consiste nel sommare tutte le parole di 16 bit che arrivano, usando l'aritmetica di complemento a 1 e quindi prendendo il complemento a 1 del risultato. Per gli scopi di questo algoritmo si assume che l'*Header checksum* sia uguale a 0 fino all'arrivo. Quest'algoritmo è più robusto di una normale somma. Si noti che l'*Header checksum* deve essere ricalcolata a ogni salto, in quanto cambia sempre almeno un campo (*Time to live*); esistono tuttavia dei trucchi per velocizzare la computazione.

Il *Source address* (indirizzo sorgente) e *Destination address* (indirizzo destinazione) contengono l'identificatore di rete e l'identificatore di host. Discuteremo gli indirizzi Internet nel prossimo paragrafo. Il campo *Options* fu progettato per fornire la possibilità alle versioni successive del protocollo di includere informazioni non presenti nel progetto iniziale, per permettere agli sperimentatori di provare nuove idee, e di evitare di allocare bit del preambolo per informazioni che sono raramente richieste. Le opzioni sono di lunghezza variabile. Ognuna di esse inizia con un codice di 1 byte che identifica l'opzione. Alcune opzioni sono seguite da un campo di 1 byte contenente la lunghezza dell'opzione, e quindi da uno o più byte di dati. Il campo *Options* viene completato a multipli di 4 byte. Attualmente sono definite cinque opzioni, elencate in figura 5-46, ma qualche router non le supporta tutte.

L'opzione *Security* descrive il grado di segretezza delle informazioni. In teoria, un router militare dovrebbe utilizzare questo campo per richiedere di non attraversare paesi considerati "a rischio" dai militari. In pratica, tutti i router lo ignorano e quindi la sua unica funzione pratica è aiutare le spie a trovare più facilmente del buon materiale.

L'opzione *Strict source routing* (routing rigido dalla sorgente) fornisce l'intero percorso

Opzione	Descrizione
Security	Specifica il livello di segretezza
Strict source routing	Fornisce il percorso completo da seguire
Loose source routing	Fornisce una lista di router che non devono essere saltati
Record route	Forza ogni router ad aggiungere il proprio indirizzo IP
Timestamp	Forza ogni router ad aggiungere il proprio indirizzo IP e timestamp

Fig. 5-46 Opzioni IP.

dalla sorgente alla destinazione come sequenza di indirizzi IP. Il datagram deve seguire esattamente quel percorso. Viene utilizzato per lo più dagli amministratori di sistema per spedire pacchetti di emergenza quando le tabelle di routing sono corrotte, o per fare misurazioni di tempo.

L'opzione *Loose source routing* (routing flessibile dalla sorgente) richiede che il pacchetto attraversi una lista di router specificati nell'ordine specificato, ma è possibile che passi attraverso altri router lungo il percorso. Normalmente, questa opzione dovrebbe elencare pochi router, per forzare un determinato cammino. Ad esempio, per fare in modo che un pacchetto da Londra a Sidney vada a ovest piuttosto che a est, questa opzione dovrebbe specificare i router di New York, Los Angeles e Honolulu. Viene per lo più utilizzata quando considerazioni politiche ed economiche impongono di evitare o di passare attraverso certi paesi.

L'opzione *Record route* (registrazione percorso) forza i router lungo il cammino ad aggiungere il loro indirizzo IP al campo opzione. Questo permette ai manager di sistema di scoprire errori negli algoritmi di routing. ("Perché tutti i pacchetti da Houston a Dallas visitano prima Tokyo?"). Quando ARPANET fu creata, nessun pacchetto passava per più di nove router, e quindi 40 byte di opzione erano sufficienti. Come menzionato in precedenza, oggi sono troppo pochi.

Infine, l'opzione *Timestamp* (timbro temporale) è simile all'opzione *Record route*, a parte il fatto che oltre a registrare l'indirizzo IP di 32 bit, ogni router registra anche un timestamp di 32 bit. Anche questa opzione viene per lo più utilizzata per il debug degli algoritmi di routing.

5.5.2 Indirizzi IP

Ogni host e router in Internet ha un indirizzo IP, che codifica l'identificatore di rete e l'identificatore di host. La combinazione è unica: non esistono due macchine con lo stesso indirizzo IP. Tutti gli indirizzi IP sono lunghi 32 bit e vengono utilizzati nei campi *Source address* e *Destination address* dei pacchetti IP. Il formato utilizzato per gli indirizzi IP è mostrato in figura 5-47. Le macchine connesse a reti multiple hanno un indirizzo IP differente per ognuna delle reti.

Le classi di formato A, B, C e D danno origine a 126 reti con 16.000.000 di host ognuna, 16.382 reti con 64K host, 2.000.000 di reti (ad esempio, LAN) con 254 host, e a un

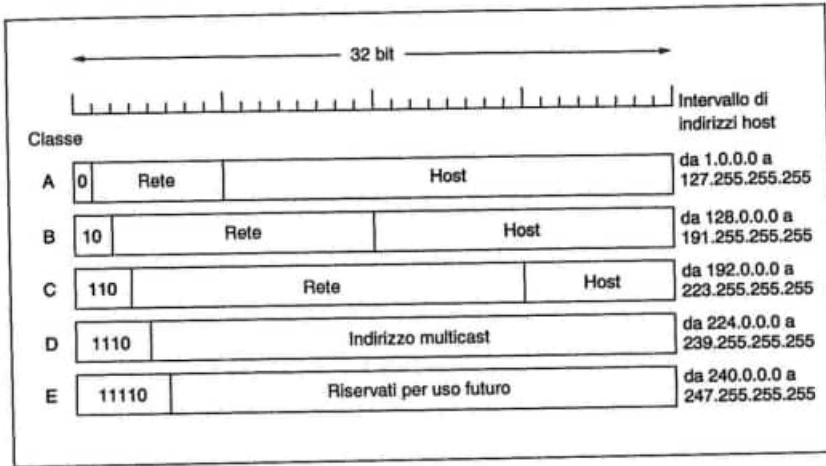


Fig. 5-47 Il formato degli indirizzi IP.

insieme di indirizzi di multicast, nei quali un datagram viene indirizzato a più di un host. Gli indirizzi che iniziano con 11110 sono riservati per uso futuro. Decine di migliaia di reti sono oggi connesse a Internet, e il numero raddoppia ogni anno. Gli indirizzi di rete sono assegnati dal NIC (Network Information Center – centro informativo di rete) per evitare conflitti.

Gli indirizzi di rete, che sono numeri di 32 bit, sono scritti normalmente in **notazione decimale a punti**. In questo formato, ognuno dei 4 byte viene scritto in decimale, da 0 a 255. Ad esempio, l'indirizzo esadecimale C0280614 viene scritto come 192.41.6.20. L'indirizzo IP più basso è 0.0.0.0 e il più alto è 255.255.255.255.

I valori 0 e -1 hanno significati speciali, come mostrato in figura 5-48. Il valore 0 significa questa rete o questo host. Il valore -1 viene usato come indirizzo di broadcast per indicare tutti gli host di una certa rete.

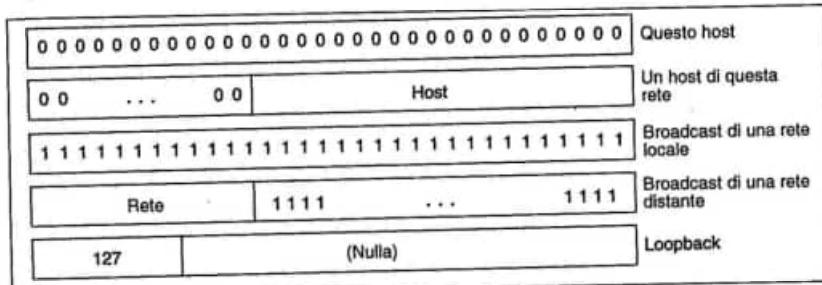


Fig. 5-48 Indirizzi speciali IP.

L'indirizzo IP 0.0.0.0 viene utilizzato dagli host al momento dell'accensione, mentre non viene più usato successivamente. Gli indirizzi IP con 0 come identificatore di rete si riferiscono alla rete corrente. Questi indirizzi consentono alle macchine di riferirsi alla propria rete senza conoscere il proprio identificatore. (ma devono conoscere la classe per sapere quanti 0 includere). L'indirizzo consistente di tutti 1 consente il broadcast sulla rete locale, generalmente una LAN. Gli indirizzi con un identificatore di rete corretto e tutti 1 nel campo host permettono di spedire pacchetti broadcast a LAN distanti ovunque in Internet. Infine, tutti gli indirizzi della forma 127.xx.yz sono riservati per test di loopback. I pacchetti spediti a tale indirizzo non vengono spediti esternamente; vengono elaborati localmente e trattati come pacchetti in arrivo. Queste consente che i pacchetti vengano spediti alla rete locale senza che il mittente conosca il proprio identificatore. Questa caratteristica viene anche usata per il debug di software di rete.

5.5.3 Sottoreti

Come abbiamo visto, tutti gli host in una rete devono avere lo stesso identificatore di rete. Questa proprietà dell'indirizzamento IP può causare problemi con la crescita della rete. Ad esempio, si consideri una società che colleghi in Internet la sua LAN di classe C. Col passare del tempo, potrebbe acquistare più di 254 computer, e quindi aver bisogno di un secondo indirizzo di classe C. Alternativamente, potrebbe acquistare una seconda LAN di tipo differente e volere un indirizzo IP diverso per essa (le LAN potrebbero essere unite tramite bridge per formare una singola rete IP, ma i bridge presentano qualche problema). Prima o poi potrebbe finire ad avere molte LAN, ognuna con il suo router e ognuna con il suo identificatore di rete di classe C.

Con la crescita del numero di reti locali distinte, la loro gestione potrebbe diventare un serio problema. Ogni volta che una nuova rete viene installata, l'amministratore di sistema deve contattare il NIC per ottenere un nuovo identificatore di rete, che deve quindi essere annunciato a tutto il mondo. Inoltre, muovere una macchina da una LAN a un'altra impone di cambiare il suo indirizzo IP, che a sua volta significa cambiare i file di configurazione e annunciare il nuovo indirizzo al mondo. Se a qualche altra macchina venisse dato l'indirizzo IP appena rilasciato, questa macchina riceverebbe email e altri dati destinati alla macchina originale fino a quando l'indirizzo non si fosse propagato ovunque nel mondo.

La soluzione di questi problemi è consentire la suddivisione di una rete in più parti per motivi interni, continuando ad agire come una singola rete per il mondo esterno. Nella letteratura Internet, queste parti vengono chiamate **sottoreti**. Come abbiamo menzionato nel capitolo 1, questo uso è in conflitto con il significato di sottorete inteso come l'insieme di tutti i router e le linee di comunicazione in una rete. Dovrebbe essere chiaro dal contesto quale sia il significato. In questo paragrafo verrà utilizzata la nuova definizione. Se la nostra società in crescita inizia con un indirizzo di classe B al posto di uno di classe C, potrebbe iniziare a numerare gli host da 1 a 254. Quando arriva la seconda LAN, potrebbe decidere, per esempio di suddividere l'identificatore di host a 16 bit in identificatore di sottorete di 6 bit e in identificatore di host a 10 bit, come mostrato in figura 5-49. Questo suddivisione permette 62 LAN (0 e -1 sono riservati), ognuna con al massimo 1022 host. Fuori dalla rete, la suddivisione non è visibile e quindi allocare una nuova sottorete non richiede di contattare il NIC o di cambiare qualsiasi database esterno. In questo esempio,

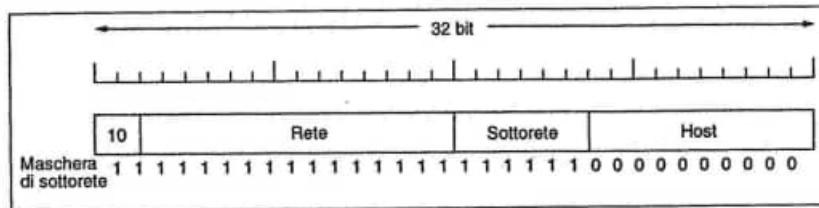


Fig. 5-49 Uno dei modi di suddividere una rete di classe B.

la prima sottorete potrebbe utilizzare indirizzi IP che iniziano a 130.50.4.1, la seconda sottorete potrebbe iniziare a 130.50.8.1 e così via.

Per vedere come funziona una sottorete, è necessario spiegare come i pacchetti vengono elaborati da un router. Ogni router ha una tabella che elenca alcuni numeri di indirizzi IP (rete, 0) e alcuni numeri di indirizzi IP (questa-rete, host). Il primo tipo indica come raggiungere una rete distante. Il secondo tipo indica come raggiungere un host locale. A ogni tabella sono associate l'interfaccia di rete da usare per raggiungere la destinazione e alcune altre informazioni.

Quando arriva un pacchetto IP, il suo indirizzo di destinazione viene cercato nella tabella di routing. Se il pacchetto è per una rete remota, viene inoltrato al prossimo router sull'interfaccia registrata nella tabella. Se è un host locale (ad es., sulla LAN del router), viene spedito direttamente alla destinazione. Se la rete non è presente, il pacchetto viene inoltrato a un router di default con tabelle più estese. Questo algoritmo comporta che ogni router deve solo tenere traccia delle reti e degli host locali, e non delle coppie (rete, host), riducendo enormemente la dimensione delle tabelle di routing.

Quando viene introdotta la suddivisione della rete, le tabelle di routing vengono modificate, aggiungendo registrazioni della forma (questa-rete, sottorete, 0) e (questa-rete, questa-sottorete, host). Quindi un router sulla sottorete k sa come raggiungere tutte le altre sottoreti e quindi come raggiungere tutti gli host nella sottorete k . Non deve conoscere i dettagli riguardanti gli host delle altre sottoreti. In effetti, l'unica cosa da cambiare è fare in modo che ogni router esegua un AND booleano dell'indirizzo con la **maschera di sottorete** della rete (vedi fig. 5-49) per sbarazzarsi dell'identificatore di host e cerchi l'indirizzo risultante nelle sue tabelle (dopo aver determinato la classe di rete). Ad esempio, un pacchetto indirizzato a 130.50.15.6 che arriva a un router della sottorete 5 viene messo in AND con la maschera di sottorete di figura 5-49 per ottenere l'indirizzo 130.50.12.0. Questo indirizzo viene cercato nelle tabelle di routing per individuare come raggiungere gli host della sottorete 3. Il router della sottorete 5 si risparmia il lavoro di tenere traccia degli indirizzi data link degli host diversi da quelli della rete 5. La suddivisione delle reti riduce lo spazio delle tabelle di routing creando una gerarchia a tre livelli.

5.5.4 Protocolli di controllo di Internet

Oltre a IP, che viene usato per il trasferimento di dati, Internet possiede molti protocolli di controllo utilizzati nel livello rete, fra cui ICMP, ARP, RARP e BOOTP. In questo paragrafo daremo un'occhiata a ognuno di essi.

Il protocollo dei messaggi di controllo di Internet

Il funzionamento di Internet viene sorvegliato accuratamente dai router. Quando avviene qualcosa di inatteso, l'evento viene riportato tramite **ICMP** (**I**nternet **C**ontrol **M**essage **P**rotocol – protocollo dei messaggi di controllo di Internet), che viene anche usato per "testare" Internet. È stata definita circa una dozzina di messaggi ICMP. I più importanti sono elencati in figura 5-50. Ogni messaggio ICMP viene encapsulato in un pacchetto IP. Il messaggio DESTINATION UNREACHABLE (destinazione irraggiungibile) viene utilizzato quando la sottorete o un router non sono in grado di localizzare la destinazione, oppure un pacchetto con il bit *DF* accesso non può essere consegnato in quanto una rete "a pacchetti piccoli" si trova lungo la strada.

Il messaggio TIME EXCEEDED (tempo scaduto) viene spedito quando un pacchetto viene scartato a causa del suo contatore che ha raggiunto lo 0. Questo evento è un sintomo che i pacchetti sono in loop, che c'è una congestione enorme, o che il valore iniziale del contatore era troppo basso.

Tipo di messaggio	Descrizione
Destination unreachable	Il pacchetto non può essere consegnato
Time exceeded	Il campo Time to live ha raggiunto 0
Parameter problem	Campo del preamble non valido
Source quench	Pacchetto regolatore
Redirect	Insegna la geografia a un router
Echo request	Chiede a una macchina se è vivo
Echo reply	Sì, sono vivo
Timestamp request	Ugualmente a Echo request, ma con timestamp
Timestamp reply	Ugualmente a Echo reply, ma con timestamp

Fig. 5.50 I principali messaggi tipo ICMP.

Il messaggio PARAMETER PROBLEM (problema nei parametri) indica che è stato trovato un valore illegale in qualche campo del preamble. Questo problema indica un errore nel software IP dell'host mittente, o al limite nel software di un router in transito.

Il messaggio SOURCE QUENCH (rimozione sorgente) era usato tempo fa per zittire gli host che stavano spedendo troppi pacchetti. Quando un host riceveva questo messaggio, era vincolato a rallentare. Oggi viene usato raramente, in quanto in caso di congestione questi pacchetti non fanno altro che aggiungere benzina al fuoco. Attualmente il controllo della congestione in Internet viene realizzato per la maggior parte nel livello trasporto e verrà analizzato in dettaglio nel capitolo 6.

Il messaggio REDIRECT (redirezione) viene usato quando un router nota che un pacchetto sembra inistradato male. Viene usato dai router per informare l'host mittente del probabile errore.

I messaggi ECHO REQUEST (richiesta di eco) e ECHO REPLY (risposta all'eco) sono usati per controllare se una data destinazione è raggiungibile e attiva. Alla ricezione di un messaggio ECHO, la destinazione è vincolata a rispondere con un messaggio ECHO REPLY. I messaggi TIMESTAMP REQUEST (richiesta di timestamp) e TIMESTAMP REPLY (risposta al timestamp) sono simili ai precedenti, a parte il fatto che il tempo di arrivo della richiesta e il tempo di partenza della risposta vengono registrati nella risposta stessa. Questo servizio viene utilizzato per misurare le prestazioni della rete.

Oltre a questi messaggi, ne esistono altri quattro tipi che si occupano di indirizzamento Internet, per consentire agli host di scoprire il proprio identificatore di rete e di gestire il caso di LAN multiple che condividono un singolo indirizzo IP. ICMP è definito in RFC 792.

Il protocollo di risoluzione degli indirizzi

Sebbene ogni macchina in Internet abbia uno o più indirizzi IP, questi non possono essere utilizzati effettivamente per spedire pacchetti, in quanto l'hardware del livello data link non è in grado di comprenderli. Attualmente, gran parte degli host è collegata a una LAN attraverso una scheda di interfaccia che capisce unicamente gli indirizzi LAN. Ad esempio, tutte le schede Ethernet mai costruite sono equipaggiate con un indirizzo Ethernet di 48 byte. I costruttori di schede Ethernet richiedono un blocco di indirizzi da un'autorità centrale per assicurarsi che nessuna coppia di schede abbia lo stesso indirizzo (per evitare conflitti nel caso le due schede dovessero apparire sulla stessa LAN). Le schede spediscono e ricevono frame basati su indirizzi Ethernet a 48 bit. Non capiscono nulla degli indirizzi IP a 32 bit.

Nasce spontanea una domanda: come vengono mappati gli indirizzi IP sugli indirizzi del livello data link, come ad esempio Ethernet? Per spiegare come funziona, utilizziamo l'esempio di figura 5-51, nel quale viene mostrata una piccola università con diverse reti di classe C. Ci sono due Ethernet, una nel dipartimento di Informatica (con indirizzo IP 192.31.65.0) e l'altra a Ingegneria elettronica (con indirizzo IP 192.31.63.0). Queste sono connesse da un anello FDDI con indirizzo IP 192.31.60.0. Ogni macchina in una Ethernet ha un unico indirizzo Ethernet, etichettato da E1 a E6, e ogni macchina nell'anello FDDI ha un indirizzo FDDI, etichettato da F1 a F3.

Iniziamo vedendo come un utente dell'host 1 spedisce un pacchetto a un utente dell'host 2. Assumiamo che il mittente conosca il nome del ricevente, possibilmente qualcosa come mary@eagle.cs.uni.edu. Il primo passo è trovare l'indirizzo IP per l'host 2, conosciuto come eagle.cs.uni.edu. Questa ricerca viene eseguita dal Domain Name System, che studieremo nel capitolo 7. Per ora, assumiamo solo che il DNS restituiscia l'indirizzo IP per l'host 2 (192.31.65.5).

A questo punto, il livello software superiore dell'host 1 costruisce un pacchetto con 192.31.65.5 nel campo *Destination address* e lo passa al software IP per la trasmissione. Analizzando l'indirizzo, il software IP vede che la destinazione è sulla propria rete, ma ha bisogno di un modo per individuare l'indirizzo Ethernet della destinazione. Una soluzione è conservare da qualche parte un file di configurazione che trasformi gli indirizzi IP in indirizzi Ethernet. Questa soluzione è certamente possibile, ma per organizzazioni con migliaia di macchine, tenere aggiornati questi file è un compito che va soggetto a errori e molto dispendioso in termini di tempo.

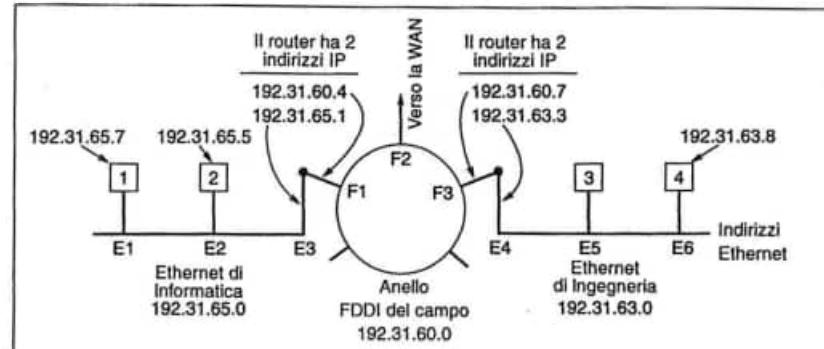


Fig. 5-51 Tre reti di classe C interconnesse: due Ethernet e un anello FDDI.

Una soluzione migliore richiede che l'host 1 mandi un pacchetto in broadcast sulla Ethernet chiedendo: "Chi possiede l'indirizzo IP 192.31.65.5?" Il broadcast verrà ricevuto dalle macchine sulla Ethernet 192.31.65.0, e ogni macchina verificherà il proprio indirizzo IP. Solo l'host 2 risponderà con il proprio indirizzo Ethernet (E2). In questo modo l'host 1 viene a sapere che l'indirizzo IP 192.31.65.5 corrisponde all'indirizzo Ethernet E2. Il protocollo per fare questo tipo di domande e ricevere una risposta viene chiamato **ARP (Address Resolution Protocol – protocollo di risoluzione degli indirizzi)**. Viene eseguito da quasi tutte le macchine in Internet. È definito nell'RFC 826.

Il vantaggio di usare ARP al posto dei file di configurazione è la semplicità. L'amministratore di sistema non ha molto da fare, a parte assegnare a ogni macchina un indirizzo IP e decidere le maschere di sottorete. ARP fa il resto.

A questo punto, il software IP dell'host 1 costruisce un frame Ethernet indirizzato a E2, inserisce il pacchetto IP (indirizzato a 192.31.65.5) nel campo contenuto, e lo copia sulla Ethernet. La scheda Ethernet dell'host 2 rileva questo frame, lo riconosce come indirizzato a se stessa, lo prende e fa partire un'interruzione. Il driver Ethernet estrae il pacchetto IP dal campo contenuto e lo passa al software IP, il quale controlla che sia stato indirizzato correttamente, e lo elabora.

Esistono varie ottimizzazioni per rendere più efficiente ARP. Innanzitutto, una volta che una macchina ha eseguito ARP, mette da parte il risultato nel caso abbia bisogno di contattare la stessa macchina nel seguito. La prossima volta troverà il mapping nella sua cache, eliminando quindi la necessità di un secondo broadcast. In molti casi l'host 2 avrà bisogno di spedire una risposta, eseguendo quindi ARP per determinare l'indirizzo IP del mittente. Questo broadcast ARP può essere evitato facendo in modo che l'host 1 includa la sua coppia (Indirizzo IP, Indirizzo Ethernet) nel proprio pacchetto ARP. Quando un broadcast ARP arriva all'host 2, la coppia (192.31.65.7, E1) viene inserita nella cache ARP dell'host 2 per usi futuri. In effetti, tutte le macchine sulla Ethernet possono inserire la stessa coppia nella loro cache ARP.

Un'altra ottimizzazione richiede che ogni macchina mandi la propria coppia in broadcast all'inizializzazione. Questo broadcast viene eseguito nella forma di un pacchetto ARP che

ricerca il proprio indirizzo IP. Non dovrebbe esserci risposta, ma un effetto collaterale del broadcast è quello di creare una registrazione nella cache ARP di tutti. Se arriva una risposta, lo stesso indirizzo IP è stato assegnato a due macchine diverse. Quella inizializzata dopo dovrebbe informare l'amministratore di sistema e bloccarsi.

Per permettere alle cache di cambiare, ad esempio quando una scheda Ethernet si guasta e viene rimpiazzata da una nuova (con un nuovo indirizzo Ethernet), le registrazioni nella cache ARP dovrebbero scadere dopo pochi minuti.

A questo punto guardiamo nuovamente la figura 5-51, solo che questa volta l'host 1 vuole spedire un pacchetto all'host 4 (192.31.63.8). Utilizzare ARP non funziona, in quanto l'host 4 non riceverà il broadcast (i router non inoltrano broadcast a livello di Ethernet). Ci sono due soluzioni. Primo, il router CS potrebbe essere configurato per rispondere alle richieste ARP per la rete 192.31.63.0 (e probabilmente anche per altre reti locali). In questo caso, l'host 1 creerà una registrazione ARP per (192.31.63.8, E3) e spedirà tranquillamente tutto il traffico per l'host 4 al router locale. Questa soluzione viene chiamata **proxy ARP** (ARP per procura). La seconda soluzione consiste nel fare in modo che l'host 1 capisca immediatamente che la destinazione appartiene a una rete remota e spedisca tutto il traffico remoto all'indirizzo Ethernet di default che gestisce tutto il traffico remoto, in questo caso E3. Questa soluzione non richiede che il router CS conosca quali reti remote sta servendo.

In entrambi i casi l'host 1 inserisce il pacchetto IP nel campo contenuto di un frame Ethernet indirizzato a E3. Quando il router CS riceve il frame Ethernet, rimuove il pacchetto IP dal contenuto del frame e cerca l'indirizzo IP nelle sue tabelle di routing, scoprendo che si suppone che i pacchetti per la rete 192.32.63.0 vadano al router 192.31.60.7. Se non conosce già l'indirizzo FDDI di 192.31.60.7, manda in broadcast un pacchetto ARP sull'anello e viene a sapere che il suo indirizzo nell'anello è F3. Inserisce quindi il pacchetto nel campo contenuto di un frame FDDI indirizzato a F3 e lo inserisce nell'anello.

Nel router EE, il driver FDDI rimuove il pacchetto dal campo contenuto e lo passa al software IP, che capisce che deve spedire il pacchetto a 192.31.63.8. Se questo indirizzo IP non è nella cache ARP, manda in broadcast una richiesta ARP sulla Ethernet EE e viene a sapere che l'indirizzo di destinazione è E6. Costruisce quindi un frame Ethernet indirizzato a E6, vi inserisce il pacchetto e lo spedisce sulla Ethernet. Quando il frame Ethernet arriva all'host 4, il pacchetto viene estratto dal frame e passato al software IP per essere elaborato.

Per far comunicare l'host 1 con una rete distante appartenente a una WAN si lavora essenzialmente allo stesso modo, a parte il fatto che questa volta le tabelle del router CS vincolano il router stesso a utilizzare il router WAN il cui indirizzo FDDI è F2.

Il Protocollo di risoluzione inversa degli indirizzi

ARP risolve il problema di individuare, dato un indirizzo IP, l'indirizzo Ethernet corrispondente. Qualche volta deve essere risolto il problema inverso: dato un indirizzo Ethernet, qual è l'indirizzo IP corrispondente? In particolare, questo problema si presenta all'accensione di una workstation senza disco rigido. Generalmente, questo tipo di macchina riceverà l'immagine binaria del suo sistema operativo da un file server remoto. Ma come viene a conoscenza del proprio indirizzo IP?

5.5 Il livello rete in Internet

La soluzione consiste nell'utilizzare **RARP** (Reverse Address Resolution Protocol – protocollo di risoluzione inversa degli indirizzi) (definito in RFC 903). Questo protocollo permette a una workstation appena inizializzata di mandare in broadcast il proprio indirizzo Ethernet e dire "Il mio indirizzo Ethernet da 48 bit è 14.04.05.18.01.25. Qualcuno lì fuori conosce il mio indirizzo IP?". Il server RARP si accorge di questa richiesta, cerca l'indirizzo Ethernet nei file di configurazione e risponde con l'indirizzo IP corrispondente. Utilizzare RARP è preferibile all'inserimento dell'indirizzo IP nell'immagine di memoria, poiché consente di usufruire della stessa immagine su tutte le macchine. Se l'indirizzo IP fosse seppellito all'interno dell'immagine, ogni workstation avrebbe bisogno di un'immagine diversa.

Uno svantaggio di RARP è il fatto che utilizza un indirizzo di destinazione composto da tutti 1 (broadcasting limitato) per raggiungere il server RARP. Tuttavia tali broadcast non sono inoltrati dai router, e quindi è necessario un server RARP per ogni rete. Per aggirare questo problema, è stato inventato un protocollo alternativo di bootstrap chiamato **BOOTP** (si vedano RFC 951, 1048 e 1084). A differenza di RARP, BOOTP utilizza messaggi UDP, che vengono inoltrati dai router. Fornisce inoltre a una workstation senza dischi informazioni aggiuntive, incluso l'indirizzo IP del file server che mantiene l'immagine di memoria, l'indirizzo IP del router di default e la maschera di sottorete da utilizzare. BOOTP è descritto in RFC 951.

5.5.5 Il protocollo di routing tra gateway interni: OSPF

Come abbiamo menzionato in precedenza, Internet è costituita da un gran numero di sistemi autonomi. Ogni AS viene gestito da organizzazioni differenti e può utilizzare al suo interno i propri algoritmi di routing. Ad esempio, se tutte e tre le reti interne delle società X, Y e Z fossero in Internet, sarebbero viste come tre AS distinti, ognuno dei quali potrebbe utilizzare un algoritmo di routing differente. Ciò nonostante, la presenza di standard anche per il routing interno semplifica l'implementazione dei confini fra gli AS e permette di riutilizzare il codice. In questo paragrafo analizzeremo il routing interno degli AS. Nel prossimo vedremo il routing tra gli AS. Un algoritmo di routing all'interno di un AS è detto **interior gateway protocol** (protocollo di routing tra gateway interni); un algoritmo di routing tra gli AS viene detto **exterior gateway protocol** (protocollo di routing tra gateway esterni).

Il protocollo di routing tra gateway interni usato originariamente in Internet era un protocollo distance vector (RIP), basato su un algoritmo Bellman-Ford. Funzionava bene in sistemi piccoli, ma meno bene con la crescita degli AS. Soffriva inoltre del problema del conteggio all'infinito e in generale di una convergenza lenta; per questo motivo fu rimpiazzato nel maggio 1979 da un protocollo link state. Nel 1988, l'Internet Engineering Task Force iniziò a lavorare a un successore, chiamato **OSPF** (Open Shortest Path First – (protocollo) aperto di cammino più breve per primo), che divenne uno standard nel 1990. Attualmente molti vendori di router lo supportano, e diventerà il principale protocollo di routing tra gateway interni nel prossimo futuro. Di seguito daremo un'occhiata a come funziona OSPF. Per la storia completa, si veda RFC 1247.

Data la lunga esperienza con altri protocolli di routing, il gruppo che progettava il nuovo protocollo aveva una lista di esigenze che andavano soddisfatte. Innanzitutto, l'algoritmo

doveva essere diffuso attraverso il suo inserimento nella letteratura di dominio pubblico, da cui la "O" di Open in OSPF. Una soluzione proprietaria posseduta da una specifica società non sarebbe andata bene. Secondo, il nuovo protocollo doveva supportare una gran varietà di metriche di distanza, inclusa la distanza fisica, il ritardo e così via. Terzo, doveva essere un algoritmo dinamico, capace di adattarsi automaticamente e velocemente ai cambiamenti della topologia.

Quarto (una novità di OSPF), doveva supportare il routing basato sul tipo di servizio. Il nuovo protocollo doveva essere in grado di instradare traffico in tempo reale in un certo modo e gli altri tipi di traffico in maniera differente. Il protocollo IP possiede il campo *Type of service*, ma nessun protocollo esistente lo utilizzava.

Quinto (collegato al punto precedente), il nuovo protocollo doveva eseguire il bilanciamento di carico, suddividendolo su linee multiple. Gran parte dei protocolli precedenti spedivano tutti i pacchetti lungo il percorso migliore. Il percorso in seconda posizione non veniva usato per nulla. In molti casi, suddividere il carico su linee multiple fornisce prestazioni migliori.

Sesto, era necessario un supporto per i sistemi gerarchici. Dal 1988, Internet era cresciuta a tal punto che nessuno si aspettava che un router potesse conoscere l'intera topologia. Il nuovo protocollo di routing doveva essere progettato in modo che nessun router dovesse conoscerla. Settimo, era necessario un minimo di sicurezza per evitare che studenti amanti del divertimento prendessero in giro i router spedendo false informazioni di routing. Infine, erano necessari dei provvedimenti per gestire i router connessi a Internet tramite tunnel. I protocolli precedenti non gestivano perfettamente queste connessioni.

OSPF supporta tre tipi di connessioni e reti:

1. Linee punto-a-punto tra due router.
2. Reti multiaccesso basate su broadcast (ad es., gran parte delle LAN).
3. Reti multiaccesso senza broadcast (ad es., gran parte delle LAN a commutazione di pacchetto).

Una rete **multiaccesso** può connettere molti router, ognuno dei quali può comunicare direttamente con tutti gli altri. Tutte le LAN e le WAN hanno questa proprietà. La figura 5-52 (a) mostra un AS contenente tutti e tre i tipi di rete. Si noti che generalmente gli host non hanno nessun ruolo in OSPF.

OSPF funziona astraendo l'insieme delle reti, dei router e dei canali effettivi tramite un grafo diretto nel quale a ogni arco viene assegnato un costo (distanza, ritardo ecc.). Viene poi calcolato il cammino più breve in base ai pesi degli archi. Una connessione seriale tra due router è rappresentata da una coppia di archi, uno per ogni direzione, i cui pesi possono essere differenti. Una rete multiaccesso è rappresentata da un nodo per la rete stessa, più un nodo per ogni router. Gli archi dal nodo della rete ai router hanno peso 0 e non sono compresi nel grafo.

La figura 5-52 (b) mostra la rappresentazione su grafo della rete di figura 5-52 (a). Fondamentalmente, OSPF non fa altro che rappresentare la rete effettiva con un grafo simile a questo e quindi calcolare il cammino più breve fra ogni coppia di router.

Poiché molti degli AS in Internet sono comunque molto grandi e non facili da gestire, OSPF consente di dividerli in **aree numerate**, ognuna delle quali è costituita da una rete oppure da un insieme di reti contigue. Le aree non devono intersecarsi, ma non è necessario che siano esaustive, ovvero qualche router può non appartenere a nessuna area. All'esterno di un'area, la topologia interna e i dettagli non sono visibili.

Ogni AS ha un'area **backbone** (spina dorsale), chiamata area 0. Tutte le aree sono connesse alla backbone, al limite tramite tunnel, in modo che sia possibile connettere qualsiasi coppia di aree nell'AS passando per il backbone. Un tunnel viene rappresentato graficamente come un arco pesato. Ogni router che è connesso a due o più aree fa parte della backbone. Come per le aree, la topologia della backbone non è visibile all'esterno.

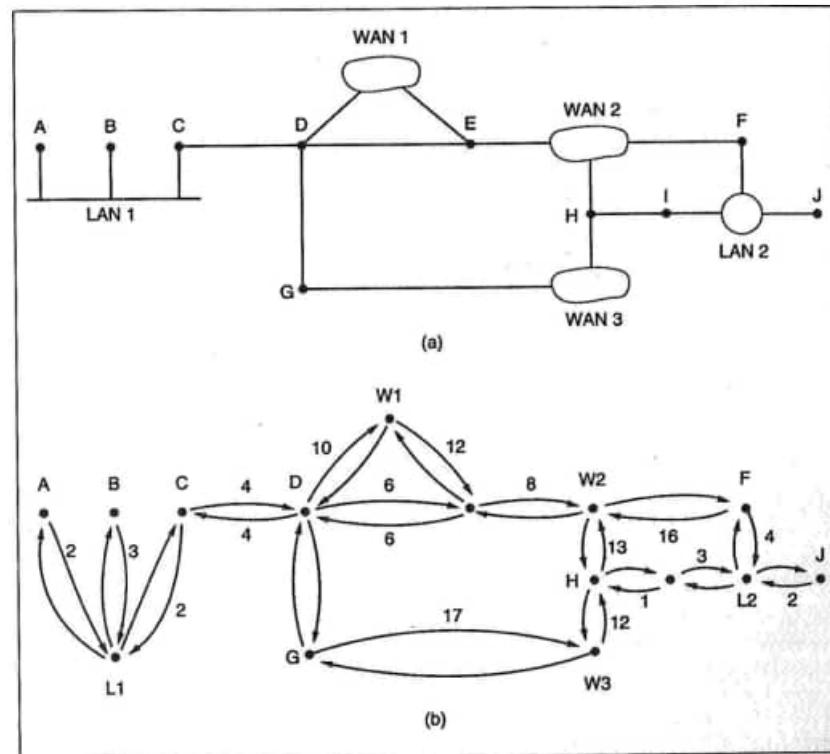


Fig. 5-52 (a) Un sistema autonomo (AS). (b) Un grafo che rappresenta (a).

All'interno di un'area, ogni router ha lo stesso database link state ed esegue lo stesso algoritmo di cammino minimo. Il suo compito principale è calcolare il cammino minimo da se stesso a ogni altro router nell'area, incluso il router che è connesso alla backbone (per definizione, ne esiste almeno uno). Un router che è connesso a due aree ha bisogno

dei database di entrambe e deve eseguire l'algoritmo di cammino minimo separatamente per ciascuna.

Il modo in cui OSPF gestisce il routing basato sul tipo di servizio è quello di avere grafi multipli, uno etichettato dai costi in cui la metrica è il ritardo, uno etichettato dai costi in cui la metrica è la capacità totale e infine uno etichettato dai costi in cui la metrica è l'affidabilità. Sebbene questo triplichi i calcoli necessari, consente di ottenere percorsi separati per ottimizzare i ritardi, la capacità totale e l'affidabilità.

Durante il funzionamento normale, possono essere richiesti tre tipi di percorsi: intra-area, inter-area e inter-AS. I percorsi intra-area sono i più facili, in quanto il router sorgente conosce già il cammino minimo per il router destinazione. Il routing inter-area procede sempre in tre passi: andare dalla sorgente alla backbone; andare attraverso la backbone all'area di destinazione; andare alla destinazione. Questo algoritmo impone una configurazione a stella di OSPF dove la backbone rappresenta il centro, mentre le altre aree sono i raggi. I pacchetti sono instradati dalla sorgente alla destinazione "così come sono". Non vengono incapsulati o incanalati, a meno di attraversare un'area la cui unica connessione alla backbone sia un tunnel. La figura 5-53 mostra sezioni di Internet con AS e aree.

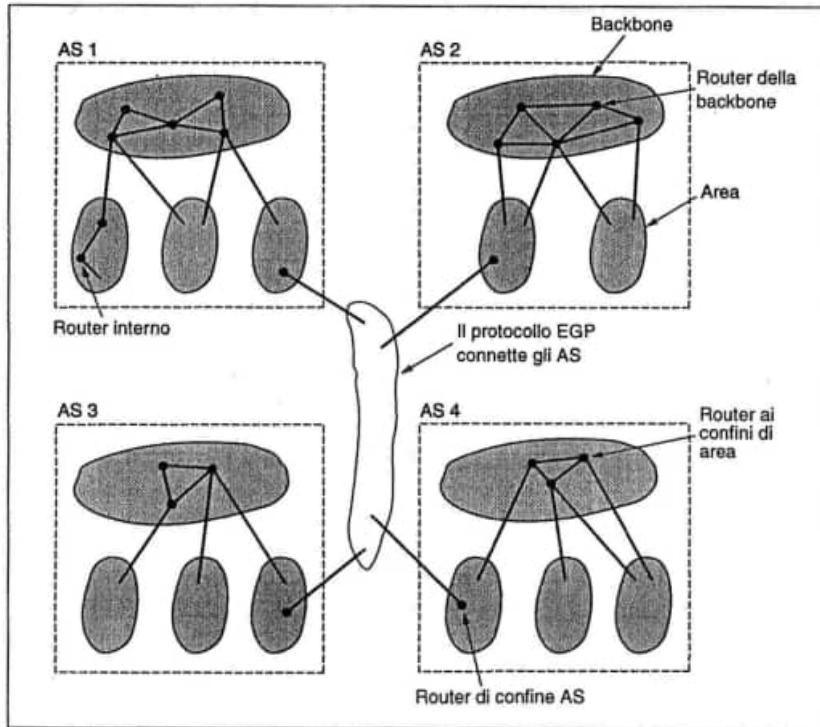


Fig. 5-53 La relazione fra AS, backbone e aree in OSPF.

OSPF distingue quattro classi di router:

1. I router interni sono completamente all'interno di un'area.
2. I router ai confini di area connettono due o più aree.
3. I router del backbone sono sulla backbone.
4. I router ai confini degli AS sono connessi a router di altri AS.

Queste classi possono intersecarsi. Ad esempio, tutti i router ai confini di area fanno automaticamente parte della backbone. Inoltre, un router che fa parte della backbone, ma che non è parte di nessun'altra area, è anche un router interno. Esempi di tutte e quattro le classi di router sono illustrati in figura 5-53.

Quando un router viene inizializzato, spedisce un messaggio HELLO su tutte le sue linee punto-a-punto e lo manda in multicast sulla LAN al gruppo costituito da tutti gli altri router. Sulle WAN ha bisogno di alcune informazioni di configurazione per sapere chi contattare. Ogni router viene a conoscere l'insieme dei vicini a partire dalle risposte.

OSPF funziona scambiando informazioni tra router **adiacenti**, che non coincidono con i 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 viene eletto **router designato**; tale router viene detto adiacente a tutti gli altri router e scambia informazioni con essi. Router vicini che non sono adiacenti non scambiano informazioni tra loro. Un router designato di backup viene sempre tenuto aggiornato per facilitare la transizione nel caso in cui il router designato primario si guasti.

Durante il funzionamento normale, ogni router spedisce un messaggio **LINK STATE UPDATE** (aggiornamento link state) a tutti i router adiacenti tramite flooding. Questo messaggio contiene lo stato e tutti i costi utilizzati nel database topologico. I messaggi di flooding vengono confermati, al fine di renderli affidabili. Ogni messaggio possiede un numero di sequenza, in modo che un router possa capire se un messaggio **LINK STATE UPDATE** appena ricevuto sia più vecchio o più nuovo di quello attualmente posseduto. I router spediscono questi messaggi anche quando una linea si guasta o viene riparata, oppure quando i costi subiscono un cambiamento.

I messaggi **DATABASE DESCRIPTION** (descrizione del database) contengono i numeri di sequenza di tutte le registrazioni link state possedute dal mittente. Confrontando i propri valori con quelli del mittente, il ricevente può determinare chi possiede i valori più recenti. Questi messaggi vengono utilizzati quando una linea viene riparata.

Un partner può richiedere all'altro informazioni link state utilizzando un messaggio **LINK STATE REQUEST** (richiesta link state). Il risultato finale di questo algoritmo è il fatto che ogni coppia di router adiacenti verifica chi possiede i dati più recenti, e le nuove informazioni vengono diffuse nell'area in questo modo. Tutti questi messaggi vengono spediti come semplici pacchetti IP. I cinque tipi di messaggi sono elencati in figura 5-54.

Infine, possiamo unire tutti questi pezzi. Utilizzando il flooding, ogni router comunica l'insieme dei propri vicini e i relativi costi a tutti gli altri router nella propria area. Queste informazioni consentono a ogni router di costruire il grafo per la propria area e calcolare

Tipo di messaggio	Descrizione
Hello	Utilizzato per scoprire chi sono i vicini
Link state update	Fornisce i costi del mittente ai suoi vicini
Link state ack	Conferma link state update
Database description	Annuncia gli aggiornamenti conosciuti dal mittente
Link state request	Richiede informazioni dal partner

Fig. 5-54 I cinque tipi di messaggi OSPF.

il cammino minimo. Anche l'area backbone esegue queste operazioni. Inoltre, i router della backbone accettano informazioni dai router ai confini di area per calcolare il cammino migliore da ogni router della backbone a ogni altro router. Queste informazioni vengono spedite indietro fino ai router ai confini di area, che le pubblicizzano all'interno delle loro aree. Utilizzando queste informazioni, un router che sta spedendo un pacchetto inter-area può selezionare il miglior router di uscita per la backbone.

5.5.6 Il protocollo di routing tra gateway esterni: BGP

All'interno di un singolo AS, il protocollo raccomandato per Internet è OSPF (sebbene non sia certamente l'unico utilizzato). Tra gli AS viene utilizzato un protocollo diverso, **BGP** (Border Gateway Protocol – protocollo per gateway di confine). È necessario un protocollo differente per gli AS, in quanto gli obiettivi di un protocollo di routing tra gateway interni e un protocollo di routing tra gateway esterni non sono gli stessi. Tutto ciò che un protocollo di routing tra gateway interni deve fare è trasportare i pacchetti dalla sorgente alla destinazione nel modo più efficiente possibile. Non ha altri problemi.

Invece i protocolli di routing tra gateway esterni devono preoccuparsi molto della gestione delle relazioni fra i clienti. Ad esempio, un AS di una società privata vorrebbe avere la possibilità di spedire e ricevere pacchetti da qualsiasi sito Internet. Ciò nonostante, potrebbe non desiderare di trasportare pacchetti originati in AS stranieri e destinati in AS differenti ("Questo è il loro problema, non il nostro"). D'altra parte, potrebbe desiderare di trasportare il traffico per i propri vicini, oppure per altri AS specifici che hanno pagato per questo servizio. Le società telefoniche, ad esempio, potrebbero essere felici di trasportare pacchetti per i loro clienti, ma non per altri. I protocolli di routing tra gateway esterni in generale, e BGP in particolare, devono essere progettati per consentire l'utilizzazione di diversi tipi di politiche gestione di routing nel traffico inter-AS, gestione che deve affrontare situazioni politiche, di sicurezza ed economiche. Alcuni esempi di questi vincoli di routing sono:

1. Nessun traffico in transito attraverso certi AS.
2. Mai mettere l'Iraq in un percorso che parte dal Pentagono.

3. Non utilizzare gli Stati Uniti per andare dal British Columbia all'Ontario.
4. Transitare per l'Albania solo se non ci sono alternative per la destinazione.
5. Il traffico che parte o arriva all'IBM non deve passare per Microsoft.

I vincoli vengono configurati a mano in ogni router BGP e non fanno parte del protocollo stesso.

Dal punto di vista dei router BGP, il mondo consiste di altri router BGP e dei canali che li uniscono. Due router BGP sono considerati connessi se condividono una rete comune. Dato l'interesse speciale di BGP nel traffico in transito, le reti sono raggruppate in tre categorie. La prima categoria è quella delle **reti stub**, che hanno una sola connessione al grafo BGP. Queste non possono essere utilizzate per il traffico in transito in quanto non c'è nessuno dall'altra parte. Poi vengono le **reti multiconnesse**. Queste potrebbero essere utilizzate per il traffico in transito, a meno che non rifiutino. Infine, ci sono le **reti di transito**, quali le backbone, che sono disponibili a trattare pacchetti di terze parti, al limite con qualche restrizione.

Le coppie di router BGP comunicano stabilendo connessioni TCP. Lavorando in questo modo, si ottengono comunicazioni affidabili e si nascondono tutti i dettagli delle reti attraversate.

BGP è fondamentalmente un protocollo distance vector, sebbene sia abbastanza differente da gran parte di essi (ad es., RIP). Invece di registrare i costi corrispondenti a ognuna delle destinazioni, ogni router BGP tiene traccia dell'intero cammino utilizzato. Allo stesso modo, invece di comunicare ai propri vicini i costi stimati per ogni possibile destinazione, ogni router BGP li tiene informati dell'intero percorso che sta utilizzando.

Si considerino per esempio i router BGP mostrati in figura 5-55 (a). In particolare, si consideri la tabella di routing di F e si supponga che utilizzi il cammino FGCD per raggiungere D. Quando i vicini spediscono a F informazioni di routing, spediscono i percorsi completi, come mostrato in figura 5-55 (b) (per semplicità, viene mostrata solo la destinazione D).

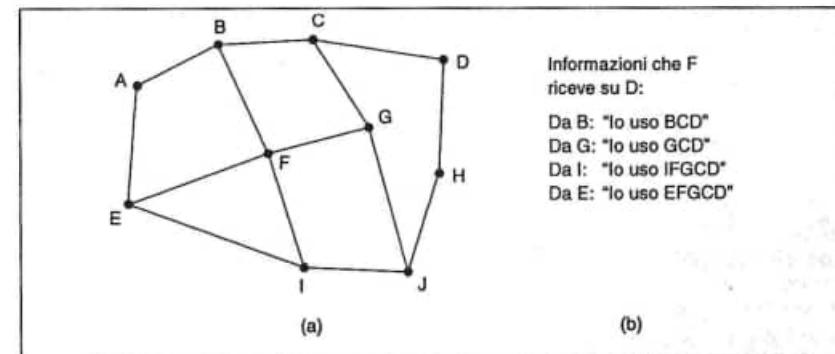


Fig. 5-55 (a) Un insieme di router BGP. (b) Informazioni spedite a F.

Dopo aver ricevuto i cammini da tutti i vicini, F li esamina per vedere qual è il migliore. Scarta velocemente i cammini attraverso I ed E , poiché passano attraverso F stesso. La scelta si riduce a B e G . Ogni router BGP contiene un modulo che esamina i percorsi per una data destinazione e assegna loro un punteggio, restituendo per ognuno di essi un valore rappresentante la "distanza". Qualsiasi percorso che non soddisfi uno dei vincoli assume automaticamente un punteggio infinito. I router scelgono quindi il percorso con la distanza più breve. La funzione di punteggio non fa parte del protocollo BGP e viene scelta dall'amministratore di sistema.

BGP risolve facilmente il problema del conteggio all'infinito che disturba gli altri algoritmi basati su vettori di distanza. Ad esempio, si supponga che G si guasti oppure che la linea FG si interrompa. F riceve quindi i percorsi BCD , $IFGCD$ e $EFGCD$ dai suoi tre vicini rimanenti. Può accorgersi immediatamente che gli ultimi due percorsi sono inutili, in quanto passano attraverso F ; quindi F sceglie $FBCD$ come nuovo percorso. Altri algoritmi basati su vettori di distanza spesso fanno la scelta sbagliata in quanto non sanno dire quale dei loro vicini possieda percorsi indipendenti per la destinazione, e quali non ne possiedano. La definizione attuale di BGP è in RFC 1654. È possibile trovare utili informazioni aggiuntive in RFC 1268.

5.5.7 Internet multicasting

Normalmente, le comunicazioni IP coinvolgono un mittente e un ricevente. Ciò nonostante, per alcune applicazioni è utile che i processi possano comunicare con un grande numero di riceventi contemporaneamente. Fra gli esempi ricordiamo la gestione di aggiornamenti replicati, i database distribuiti, la trasmissione di quotazioni di mercato a più di un broker e la gestione di chiamate telefoniche in teleconferenza (cioè con molti utenti). IP supporta il multicasting utilizzando gli indirizzi di classe D. Ogni indirizzo di classe D identifica un gruppo di host. Sono disponibili 28 bit per identificare i gruppi, in modo che possano esistere oltre 250.000.000 di gruppi contemporaneamente. Quando un processo spedisce un pacchetto a un indirizzo di classe D, viene fatto un tentativo best-efforts di consegnarlo a tutti i membri del gruppo, ma non vengono fornite garanzie. Alcuni membri potrebbero non ricevere il pacchetto.

Sono supportati due tipi di indirizzi di gruppo: permanenti e temporanei. Un gruppo permanente esiste sempre e non deve essere creato. Ogni gruppo permanente ha un indirizzo permanente. Alcuni esempi di tale tipo di indirizzo possono essere:

- 224.0.0.1 tutti i sistemi in una LAN
- 224.0.0.2 tutti i router in una LAN
- 224.0.0.5 tutti i router OSPF in una LAN
- 224.0.0.6 tutti i router OSPF designati in una LAN

I gruppi temporanei devono essere creati prima di poter essere usati. Un processo può domandare al proprio host di unirsi a un gruppo specifico. Può anche domandare al proprio host di lasciare il gruppo. Quando l'ultimo processo di un host lascia il gruppo, quest'ultimo non è più presente nell'host. Ogni host tiene traccia dei gruppi a cui appartengono i propri processi.

5.5 Il livello rete in Internet

Il multicasting è implementato attraverso speciali router multicast, che possono coincidere con i router standard oppure no. Approssimativamente una volta al minuto, ogni multicast router spedisce un multicast hardware (cioè del livello data link) agli host nella sua LAN (indirizzo 224.0.0.1) chiedendo di rispondere con l'elenco dei gruppi a cui sono interessati i loro processi. Ogni host spedisce una risposta contenente tutti gli indirizzi di classe D a cui è interessato.

Questi pacchetti di domanda e risposta utilizzano un protocollo chiamato IGMP (**Internet Group Management Protocol** – protocollo Internet di gestione dei gruppi), che è vagamente analogo a ICMP. Ci sono solo due tipi di pacchetti, di richiesta e di risposta, ognuno con un semplice formato fisso contenente alcuni informazioni di controllo nella prima parola del contenuto del pacchetto e un indirizzo di classe D nella seconda parola. Viene descritto in RFC 1112.

Il multicast routing viene realizzato basandosi sugli spanning tree. Ogni router di multicast scambia informazioni con i propri vicini utilizzando un protocollo distance vector modificato, al fine di costruire per ognuno dei gruppi uno spanning tree che copra tutti i membri del gruppo. Vengono utilizzate varie ottimizzazioni per potare l'albero ed eliminare i router e le reti che non sono interessati in gruppi particolari. Il protocollo fa un uso pesante del tunneling per evitare di infastidire nodi che non fanno parte dello spanning tree.

5.5.8 IP mobile

Molti utenti di Internet hanno computer portatili e vogliono restare connessi alla rete quando visitano un sito Internet remoto e addirittura lungo la strada per raggiungerlo. Sfortunatamente, l'indirizzamento IP rende il lavoro remoto più facile a dirsi che a farsi. In questo paragrafo esamineremo sia il problema che la soluzione. Una descrizione più dettagliata è compresa in Johnson (1995).

Il vero problema è lo schema di indirizzamento stesso. Ogni indirizzo IP contiene tre campi: la classe, l'identificatore di rete e l'identificatore di host. Ad esempio, si consideri la macchina con indirizzo IP 160.80.40.20. Il 160.80 descrive la classe (B) e l'identificatore di rete (8272); il 40.20 è l'identificatore di host (10260). I router di tutto il mondo possiedono tabelle di routing che indicano quale linea usare per raggiungere la rete 160.80. Ogni qual volta arriva un pacchetto con indirizzo IP di destinazione della forma 160.80.xxx.yyy, esce attraverso quella linea.

Se all'improvviso la macchina con quell'indirizzo venisse trascinata in qualche posto remoto, i pacchetti per essa continuerebbero a essere instradati verso la sua LAN (o router). Il proprietario non riceverebbe più posta e altri servizi. Dare alla macchina un nuovo indirizzo IP corrispondente alla nuova locazione non sarebbe consigliabile, in quanto un gran numero di persone, programmi e database dovrebbero essere informati del cambiamento.

Un altro approccio è fare in modo che i router utilizzino per il routing l'intero indirizzo IP, invece di utilizzare unicamente classe e rete. Tuttavia, questa strategia richiederebbe che ogni router avesse milioni di registrazioni nelle tabelle, un costo astronomico per Internet.

Quando gli utenti iniziarono a richiedere la possibilità di utilizzare host mobili, l'IETF creò un Gruppo di Lavoro per trovare la soluzione. Il Gruppo di Lavoro formulò velocemente un numero di obiettivi considerati desiderabili in ogni soluzione. I principali erano:

1. Ogni stazione mobile deve essere in grado di utilizzare ovunque il proprio indirizzo IP.
2. Non sono consentiti cambiamenti software alle stazioni fisse.
3. Non sono consentiti cambiamenti al software e alle tabelle dei router.
4. La gran parte dei pacchetti delle stazioni mobili non dovrebbero fare deviazioni lungo la strada.
5. Non dovrebbe esserci nessun costo aggiuntivo quando la stazione mobile è a casa.

La soluzione scelta è quella descritta nel paragrafo 5.2.8. Per riassumerla brevemente, ogni sito che vuole permettere ai propri utenti di viaggiare deve creare un agente base. Ogni sito che vuole consentire la presenza di visitatori deve creare un agente straniero. Quando un host mobile si collega a un sito straniero, contatta l'host straniero e si registra. L'host straniero contatta quindi l'agente base e gli spedisce un **care-of address**, normalmente l'indirizzo IP dell'agente straniero.

Quando un pacchetto arriva alla LAN base dell'utente, giunge a qualche router connesso alla LAN. Il router cerca di localizzare l'host nel modo solito, mandando in broadcast un pacchetto ARP che chiede, ad esempio: "Qual è l'indirizzo Ethernet di 160.80.40.20?". L'agente base risponde a questa interrogazione dando il proprio indirizzo Ethernet. Il router spedisce quindi i pacchetti per 160.80.40.20 all'agente base, il quale, a sua volta, li trasmette tramite tunnel all'indirizzo care-of incapsulandoli nel contenuto di un pacchetto IP indirizzato all'agente straniero. Questo quindi li estrae e li consegna all'indirizzo data-link della stazione mobile. Inoltre, l'agente base fornisce l'indirizzo care-of al mittente, in modo che i pacchetti futuri possano essere spediti direttamente all'agente straniero. Questa soluzione soddisfa tutte le richieste illustrate in precedenza.

Esiste un piccolo dettaglio di cui probabilmente non è neanche il caso di parlare. Nel momento in cui l'host mobile si muove, il router probabilmente conserva nella cache il corrispondente indirizzo Ethernet (troppo presto affinché sia invalidato). Per rimpiazzare questo indirizzo Ethernet con quello dell'agente base, viene utilizzato un trucco chiamato **ARP gratuito**. Questo è un messaggio speciale e non richiesto destinato al router, che causa il cambiamento di una registrazione nella cache; in questo caso, quella dell'host mobile che se ne sta andando. Quando l'host mobile ritorna più tardi, lo stesso trucco viene utilizzato per aggiornare nuovamente la cache.

Niente nel progetto impone che un host mobile non possa essere il proprio agente straniero, ma questo approccio funziona solamente se l'host mobile (nel suo ruolo di agente straniero) è connesso logicamente a Internet presso il sito in cui si trova. Inoltre, deve essere in grado di acquisire un indirizzo IP care-of (temporaneo) da utilizzare. Questo indirizzo IP deve appartenere alla LAN alla quale è attualmente connesso.

La soluzione IETF per gli host mobili risolve un certo numero di altri problemi non citati finora. Ad esempio, come vengono trovati gli agenti? La soluzione è che ognuno di essi trasmetta periodicamente in broadcast il proprio indirizzo e il tipo di servizi che è disposto a offrire (ad es., agente base, straniero o entrambi). Quando un host mobile arriva da qualche parte, aspetta uno di questi broadcast, chiamati **pubblicità**. Alternativamente, può

trasmettere in broadcast un pacchetto che avvisi del proprio arrivo e sperare che l'agente straniero locale risponda.

Un altro problema che deve essere risolto è cosa fare con host mobili "maleducati" che se ne vanno senza dire arrivederci. La soluzione è rendere valida un'iscrizione solo per un intervallo fisso di tempo. Se non viene rinfrescata periodicamente, scade, in modo che l'agente straniero possa ripulire le proprie tabelle.

Un'altra problematica riguarda la sicurezza. Quando un agente base riceve un messaggio che richiede di "inoltrare verso qualche indirizzo IP tutti i pacchetti per Nora", sarebbe meglio non accettarlo a meno di non essere convinti che Nora sia la mittente della richiesta, e che nessun altro stia cercando di impersonarla. Per questi scopi vengono utilizzati protocolli di autenticazione crittografica. Studieremo questi protocolli nel capitolo 7.

L'ultimo punto discusso dal Gruppo di Lavoro è collegato ai livelli di mobilità. Si immagini un aeroplano con una Ethernet bordo utilizzata dai computer di navigazione. Su questa Ethernet è presente un router standard che comunica con il resto di Internet a terra per mezzo di un canale radio. Un bel giorno, a qualche geniale dirigente di marketing viene l'idea di installare connettori Ethernet in tutti i braccioli dell'aereo, in modo che possano connettersi i passeggeri con computer portatili.

A questo punto esistono due livelli di mobilità: i computer dell'apparecchio, che sono stazionari rispetto alla Ethernet, e i computer dei passeggeri, che invece sono mobili. Inoltre, il router a bordo è mobile rispetto ai router sulla terraferma. La mobilità rispetto a un sistema che è esso stesso mobile può essere gestita utilizzando tunneling ricorsivo.

5.5.9 CIDR – routing interdominio senza classe

IP è stato usato tantissimo per oltre un decennio. Ha funzionato molto bene, come ha dimostrato la crescita esponenziale di Internet. Sfortunatamente, IP sta rapidamente diventando vittima della propria popolarità: sta esaurendo gli indirizzi. L'incombente disastro ha suscitato un gran numero di discussioni e controversie su cosa fare. In questo paragrafo descriveremo sia il problema che molte delle soluzioni proposte. Una descrizione più completa è data in Huitema (1996).

Già nel 1987, alcuni lungimiranti predissero che un giorno Internet sarebbe potuta crescere fino a 100.000 reti. Gran parte degli esperti si fecero beffe di questa previsione, credendola lontana di decenni. Le 100.000 reti sono state connesse nel 1996. Il problema, esposto semplicemente, è che Internet sta rapidamente esaurendo gli indirizzi. Teoricamente esistono oltre 2.000.000.000 di indirizzi, ma la pratica di organizzare lo spazio di indirizzamento in classi (vedi fig. 5-47) ne spreca milioni. In particolare, il vero problema sono gli indirizzi di classe B. Per molte organizzazioni, una rete di classe A, con 16.000.000 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 è quella giusta. Nel folklore Internet, questa situazione è conosciuta come il **problema dei tre orsi** (dalla favola "Chiomadore e i tre orsi").

In realtà, un indirizzo di classe B è ancora troppo grande per gran parte delle organizzazioni. Alcuni studi hanno mostrato come più della metà delle reti di classe B abbiano meno di 50 host. Una rete di classe C sarebbe andata meglio, ma non c'è dubbio che ogni organizzazione che ha chiesto un indirizzo di classe B pensava di crescere oltre gli 8 bit del campo host. In retrospettiva, sarebbe stato meglio fare in modo che le reti di classe

C utilizzassero 10 bit al posto di 8, permettendo 1022 host per rete. In questo caso, gran parte delle organizzazioni avrebbe scelto probabilmente una rete di classe C, di cui ce ne sarebbero state 500.000 (al posto delle 16.384 reti di classe B).

Tuttavia, un ulteriore problema è emerso più velocemente: l'esplosione delle tabelle di routing. Dal punto di vista dei router, lo spazio di indirizzamento IP è una gerarchia a due livelli, con l'identificatore di rete e l'identificatore di host. I router non devono conoscere tutti gli host, ma devono conoscere tutte le reti. Se esistessero 500.000 reti di classe C, ogni router in Internet avrebbe bisogno di una tabella con 500.000 registrazioni, una per rete, contenente la linea da utilizzare per raggiungere tale rete e alcune altre informazioni.

La memorizzazione effettiva di tabelle da 500.000 registrazioni è probabilmente fattibile, sebbene costosa per router critici che mantengono le loro tabelle in RAM statiche sulle schede di I/O. Un problema più serio è il fatto che la complessità di molti algoritmi collegati alla gestione delle tabelle cresce più che linearmente. Peggio ancora, gran parte del software e del firmware dei router esistenti è stata progettata quando Internet aveva 1.000 reti connesse, e 10.000 reti sembravano lontane decenni. Le scelte di progetto elaborate allora sono oggi lontane dall'essere ottimali.

Inoltre, molti algoritmi di routing richiedono la trasmissione periodica delle tabelle. Più grandi le tabelle, più facilmente qualche parte di esse verrà persa lungo la strada, causando incompletezza dei dati presso la destinazione, e al limite instabilità nel routing.

I problemi delle tabelle di routing avrebbero potuto essere risolti scegliendo una gerarchia più profonda. Ad esempio, potrebbe essere utile avere un campo per il paese, lo stato, la città, la rete e l'host. Quindi ogni router dovrebbe sapere come raggiungere ogni paese, ogni stato (o provincia) nel suo paese, ogni città nel suo stato (o provincia), e ogni rete nella sua città. Sfortunatamente, questa soluzione richiederebbe più di 32 bit per gli indirizzi IP e utilizzerebbe inefficientemente gli indirizzi (il Liechtenstein avrebbe lo stesso numero di bit degli Stati Uniti).

In breve, molte soluzioni risolvono un problema, ma ne creano un altro. Una soluzione che sta per essere implementata e darà a Internet un po' di fiato è **CIDR (Classless InterDomain Routing – routing interdominio senza classi)**. L'idea base di CIDR, che è descritto in RFC 1519, è quella di allocare le reti di classe C rimaste (non più di 2.000.000) in blocchi di dimensioni variabili. Se un sito necessita di 2000 indirizzi, viene fornito di un blocco di 2048 indirizzi (8 reti di classe C contigue), e non un indirizzo completo di classe B. In modo simile, un sito che necessita di 8000 indirizzi ne riceve 8192 (32 reti di classe C contigue).

Oltre a usare blocchi contigui di reti di classe C come unità, in RFC 1519 vennero cambiate anche le regole di allocazione per gli indirizzi di classe C, nel modo seguente:

Indirizzi da 194.0.0.0 a 195.255.255 sono per l'Europa

Indirizzi da 198.0.0.0 a 199.255.255 sono per il Nord America

Indirizzi da 200.0.0.0 a 201.255.255 sono per il Sud e il Centro America

Indirizzi da 202.0.0.0 a 203.255.255 sono per l'Asia e il Pacifico

In questo modo, ogni regione viene fornita di 32.000.000 di indirizzi allocabili, con altri

320.000.000 di indirizzi di classe C da 204.0.0.0 fino a 223.255.255.255 tenuti da parte per il futuro. Il vantaggio di questa allocazione è il fatto che ora ogni router esterno all'Europa che riceve un pacchetto indirizzato a 194.xx.yy.zz o 195.xx.yy.zz può spedirlo al suo gateway europeo standard. In effetti 32.000.000 di indirizzi sono stati compressi in una singola registrazione della tabella di routing. Lo stesso avviene per le altre regioni. Naturalmente, una volta che un pacchetto 194.xx.yy.zz raggiunge l'Europa, sono necessarie tabelle di routing più dettagliate. Una possibilità è quella di avere 131.072 registrazioni per le reti da 194.0.0.xx a 195.255.255.xx, ma ciò è precisamente quell'esplosione delle tabelle di routing che stiamo cercando di evitare. Al suo posto, ogni tabella di routing viene estesa con un maschera di 32 bit. Quando arriva un pacchetto, il suo indirizzo di destinazione viene innanzitutto estratto. Quindi (concettualmente) la tabella di routing viene scindita registrazione per registrazione, mascherando l'indirizzo di destinazione e comparandolo con le registrazioni in tabella cercando una corrispondenza.

Per rendere più chiaro questo processo di confronto, consideriamo un esempio. Si supponga che l'Università di Cambridge abbia bisogno di 2048 indirizzi e che le vengano assegnati gli indirizzi da 194.24.0.0 a 194.24.7.255, assieme alla maschera 255.255.248.0. Successivamente l'Università di Oxford richiede 4096 indirizzi. Poiché un blocco di 4096 indirizzi deve trovarsi a un confine di 4096 byte, non può ricevere gli indirizzi che partono a 194.8.0.0. Al loro posto riceve gli indirizzi da 194.24.16.0 fino a 194.24.31.255 assieme alla maschera 255.255.240.0. A questo punto l'Università di Edimburgo richiede 1024 indirizzi e riceve gli indirizzi da 194.24.8.0 a 194.24.11.255 e la maschera 255.255.252.0. Le tabelle di routing di tutta Europa vengono aggiornate con tre registrazioni, ognuna delle quali contiene un indirizzo di base e un maschera. Queste registrazioni (in binario) sono:

Indirizzi	Maschere
11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
11000010 00011000 00010000 00000000	11111111 11111111 11110000 00000000
11000010 00011000 00001000 00000000	11111111 11111111 11111100 00000000

A questo punto si consideri cosa succede quando arriva un pacchetto indirizzato a 194.24.17.4, che in binario corrisponde a:

11000010 00011000 00010001 00000100

Innanzitutto viene messo in AND booleano con la maschera di Cambridge ottenendo

11000010 00011000 00010000 00000000

Questo valore non corrisponde all'indirizzo base di Cambridge, e quindi l'indirizzo originale viene messo in AND con la maschera di Oxford ottenendo

11000010 00011000 00010000 00000000

Questo valore coincide con l'indirizzo base di Oxford, e così il pacchetto viene spedito al router corrispondente. In pratica, le registrazioni dei router non vengono provate sequenzialmente, ma vengono utilizzati dei trucchi di indicizzazione per velocizzare la ricerca. Inoltre, è possibile che due registrazioni coincidano, nel qual caso quella che possiede il maggior numero di bit 1 vince. Infine, la stessa idea può essere applicata a tutti gli indirizzi, non solo a quelli di classe C; in questo modo con CIDR le vecchie classi A, B e C non vengono più utilizzate per il routing. Questo è il motivo per cui viene detto routing senza classi. CIDR viene descritto più dettagliatamente in Ford *et al.* (1993); Huitema (1995).

5.5.10 IPv6

Sebbene CIDR probabilmente potrebbe avere ancora alcuni anni di vita, chiunque può capire che IP nella sua versione attuale (IPv4) ha i giorni contati. In aggiunta a questi problemi tecnici, ne esiste un altro che incombe. Fino a poco tempo fa, Internet è stata utilizzata in larga parte dalle università, dall'industria ad alta tecnologia e dal governo americano (in particolare il Dipartimento della difesa). Con l'esplosione dell'interesse per Internet alla metà degli anni novanta, è probabile che nel prossimo millennio verrà utilizzata da un insieme di persone molto più grande, e in particolare da persone con esigenze differenti. Tanto per cominciare, milioni di persone potrebbero utilizzare portatili senza filo per tenersi in contatto con la loro postazione base. Inoltre, con l'incombente convergenza fra le industrie di computer, delle telecomunicazioni e dell'intrattenimento, può non essere lontano il giorno in cui ogni televisore nel mondo sarà un nodo Internet, rendendo possibile che miliardi di macchine vengano utilizzate per il video a richiesta. Date le circostanze, diviene chiaro che IP deve evolversi e diventare più flessibile.

Nel 1990, intuendo la nascita di questi problemi, l'IETF iniziò a lavorare a una nuova versione di IP, che non esaurisse mai lo spazio di indirizzamento, che risolvesse una varietà di altri problemi e che fosse anche più flessibile ed efficiente. I suoi obiettivi principali erano:

1. Supportare miliardi di host, anche in caso di allocazione inefficiente dello spazio.
2. Ridurre la dimensione delle tabelle di routing.
3. Semplificare il protocollo, per permettere ai router di smistare i pacchetti più velocemente.
4. Fornire una maggiore sicurezza (autenticazione e privacy) di quella offerta dall'IP corrente.
5. Prestare più attenzione al tipo di servizio, in particolare per dati in tempo reale.
6. Semplificare il multicasting permettendo di specificare il campo di azione.
7. Rendere possibile il trasferimento di un host senza modificare il suo indirizzo.
8. Permettere al protocollo di evolversi in futuro.
9. Permettere ai protocolli vecchio e nuovo di coesistere per anni.

Per trovare un protocollo che soddisfacesse tutte queste esigenze, l'IETF fece una richiesta di proposte e discussione in RFC 1550. Le risposte furono 21, anche se non tutte furono proposte complete. Entro il dicembre 1992 venivano discusse 7 proposte valide, che andavano dal fare piccole correzioni a IP al gettare via tutto e rimpiazzarlo con un protocollo completamente differente.

Una proposta fu quella di eseguire TCP sopra CLNP, il quale, con i suoi 160 bit per gli indirizzi avrebbe fornito uno spazio di indirizzamento sufficiente per sempre e avrebbe unificato due dei principali protocolli del livello rete. Tuttavia, molti sentivano che ciò sarebbe stata un'ammissione del fatto che qualcosa nell'ambiente OSI era stata effettivamente realizzato bene, un'affermazione considerata politicamente scorretta nell'ambiente Internet. Poiché CLNP fu modellato a partire da IP, essi non sono in realtà molto differenti. In effetti, il protocollo scelto alla fine differisce da IP più di quanto faccia CLNP.

Un ulteriore punto a sfavore di CLNP fu il suo supporto insufficiente per la gestione di tipi di servizio differenti, necessario per trasmettere efficientemente dati multimediali. Tre delle migliori proposte furono pubblicate in *IEEE Network* (Deering, 1993; Francis, 1993; Katz, Ford, 1993). Dopo molte discussioni, revisioni e cambiamenti di fronte, fu scelta una versione modificata e combinata delle proposte di Deering e Francis, fino ad allora chiamata SIPP (**S**imple **I**nternet **P**rotocol **P**lus) e le fu dato il nome di IPv6 (IPv5 era già in uso per un protocollo sperimentale in tempo reale).

IPv6 soddisfa discretamente bene gli obiettivi. Mantiene le caratteristiche positive di IP, scarta o nasconde quelle negative e ne inserisce di nuove quando necessario. In generale, IPv6 non è compatibile con IPv4, ma lo è con tutti gli altri protocolli di Internet, inclusi TCP, UDP, ICMP, IGMP, OSPF, BGP e DNS, in qualche caso con alcune piccole modifiche (in gran parte per gestire gli indirizzi più lunghi). Le caratteristiche principali di IPv6 sono discusse in seguito. Ulteriori informazioni possono essere trovate negli RFC 1883-1887.

Innanzitutto, IPv6 ha indirizzi più lunghi di IPv4. Sono lunghi 16 byte e ciò risolve il problema per cui è stato introdotto IPv6: fornire un riserva effettivamente illimitata di indirizzi Internet. Parleremo più diffusamente degli indirizzi fra poco.

Un altro vantaggio di IPv6 è la semplificazione del preambolo, che contiene solo 7 campi (contro i 13 di IPv4). Questa modifica permette ai router di elaborare i pacchetti più velocemente e di aumentare la capacità di trasmissione. Discuteremo fra poco anche del preambolo.

Un altro fattore di miglioramento è il supporto per le opzioni. Questo cambiamento è essenziale con il nuovo preambolo, in quanto campi che prima erano obbligatori ora sono opzionali. Inoltre, il modo in cui vengono rappresentate le opzioni è differente e permette ai router di saltare più velocemente le opzioni che non li riguardano. Questa caratteristica velocizza l'elaborazione dei pacchetti.

Un'altra area in cui IPv6 rappresenta un grande miglioramento è quella della sicurezza. L'IETF ne aveva abbastanza di storie giornalistiche su precoci dodicenni che usano i loro personal computer per forzare banche e basi militari in Internet. Era opinione diffusa che bisognasse prendere qualche provvedimento per migliorare la sicurezza. Autenticazione e privacy sono caratteristiche chiave del nuovo IP.

Infine, è stata prestata ai tipi di servizio un'attenzione maggiore rispetto al passato. IPv4

effettivamente possiede un campo a 8 bit dedicato a questo problema, ma con la crescita che ci si attende per il futuro nel traffico multimediale è necessario più spazio.

Il preambolo IPv6 principale

Il preambolo IPv6 è mostrato in figura 5-56. Il campo *Version* (versione) è sempre uguale a 6 per IPv6 (a 4 per IPv4). Durante il periodo di transizione da IPv4, che durerà probabilmente un decennio, i router dovranno essere in grado di esaminare questo campo per decidere quale tipo di pacchetto stanno esaminando. Fra parentesi, si noti che eseguire questo test spreca qualche istruzione in un momento critico, e quindi molte implementazioni probabilmente cercheranno di evitarlo utilizzando qualche campo del preambolo data link per distinguere pacchetti IPv4 da pacchetti IPv6. In questo modo, i pacchetti possono essere passati direttamente al gestore del livello di rete corretto. Ciò nonostante, il fatto che il livello data link sia a conoscenza dei tipi di pacchetti del livello rete viola completamente il principio progettuale che ogni livello dovrebbe ignorare il significato dei bit che riceve da quello superiore. La discussione tra le fazioni "Lo fa correttamente" e "Lo fa velocemente" sarà senza dubbio lunga e cruenta.

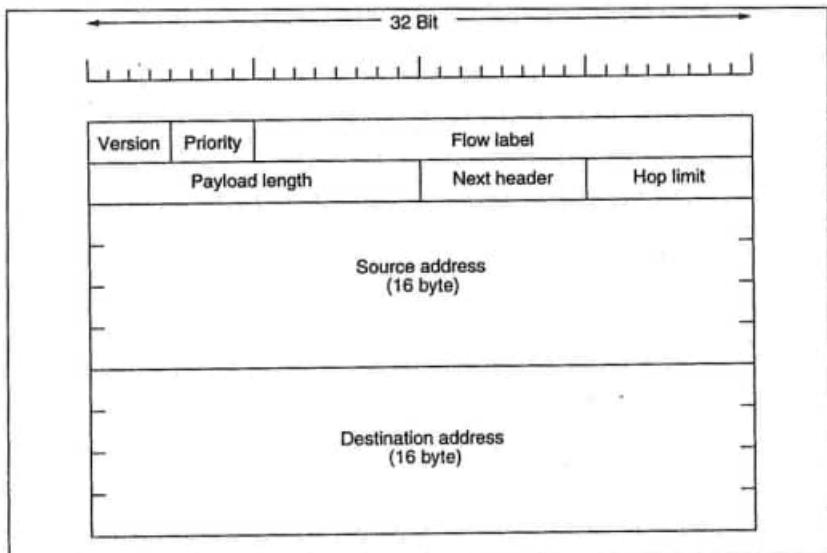


Fig. 5-56 Il preambolo IPv6 standard (richiesto).

Il campo *Priority* (priorità) è utilizzato per distinguere tra pacchetti le cui sorgenti possono essere sotto controllo di flusso e quelle che non possono esserlo. Valori da 0 a 7 sono riservati a trasmissioni che possono rallentare nel caso di congestione. Valori da 8 a 15 sono riservati a traffico in tempo reale il cui tasso di spedizione è costante, anche se tutti i pacchetti vengono persi. Audio e video cadono nella seconda categoria. Questa distin-

zione permette ai router di gestire meglio i pacchetti in caso di congestione. All'interno di ogni gruppo, pacchetti con numeri più bassi sono meno importanti di pacchetti con numeri più alti. Lo standard IPv6 suggerisce, ad esempio, di utilizzare 1 per le news, 4 per FTP e 6 per Telnet, in quanto ritardare un pacchetto di news per pochi secondi non provoca problemi, mentre ritardare un pacchetto Telnet certamente sì.

Il campo *Flow label* (etichetta di flusso) è ancora sperimentale e verrà utilizzato per permettere a una sorgente e a una destinazione di creare una pseudoconnessione con particolari proprietà ed esigenze. Ad esempio, un flusso di pacchetti da un processo di un certo host sorgente a un processo di un certo host destinazione potrebbe avere esigenze rigide per quanto riguarda i ritardi e aver bisogno quindi di riservare una certa larghezza di banda. È possibile creare anticipatamente un flusso e assegnare a esso un identificatore. Ogni qual volta arrivi un pacchetto con il campo *Flow label* diverso da 0, tutti i router possono guardare nelle tabelle interne per vedere quale tipo di trattamento speciale è richiesto. In effetti, i flussi sono un tentativo per avere contemporaneamente la flessibilità delle reti datagram e le garanzie di una rete a circuito virtuale.

Ogni flusso è identificato dall'indirizzo della sorgente, dall'indirizzo della destinazione e dall'identificatore di flusso; quindi, possono essere attivi più di un flusso contemporaneamente tra una data coppia di indirizzi IP. In questo modo, se due flussi provenienti da host differenti ma con lo stesso identificatore attraversassero lo stesso router, quest'ultimo sarebbe in grado di distinguere i flussi utilizzando gli indirizzi di destinazione e sorgente. Gli identificatori di flusso saranno scelti casualmente, piuttosto che assegnati sequenzialmente a partire da 1, in modo da semplificare ai router la procedura di hashing.

Il campo *Payload lenght* (lunghezza del carico utile) indica quanti byte seguono il preambolo di 40 byte di figura 5-56. Il nome è differente da quello del campo IPv4 *Total lenght*, poiché il significato è cambiato leggermente: il preambolo di 40 byte non viene più contato come parte della lunghezza come in precedenza.

Il campo *Next header* (preambolo successivo) svela l'inganno. Il motivo per cui il preambolo può essere semplificato è il fatto che possono esistere preamboli addizionali di estensione (opzionali). Questo campo indica quale dei sei preamboli di estensione definiti fino a ora seguia il preambolo principale. Se questo preambolo è l'ultimo preambolo IP, il campo *Next header* indica a quale gestore di protocollo di trasporto (ad es. TCP, UDP) debba essere passato il pacchetto.

Il campo *Hop limit* (numero massimo di salti) è usato per evitare che un pacchetto sopravviva in eterno. È, in pratica, uguale al campo *Time to live* di IPv4, cioè un campo che viene decrementato a ogni salto. In teoria, in IPv4 corrispondeva a un tempo in secondi, ma nessun router lo utilizzava in questo modo; per questo motivo, il nome è stato cambiato per riflettere il modo in cui viene effettivamente usato.

Vengono quindi i campi *Source address* (indirizzo sorgente) e *Destination address* (indirizzo destinatario). La proposta originale di Deering, SIP, utilizzava indirizzi di 8 byte, ma durante il processo di revisione alcune persone ritenevano che IPv6 con indirizzi di 8 byte avrebbe probabilmente esaurito lo spazio di indirizzamento in pochi decenni, mentre indirizzi di 16 byte non si sarebbero mai esauriti. Altre persone sostenevano che 16 byte sarebbero stati eccessivi, mentre altri ancora avrebbero preferito utilizzare indirizzi di 20 byte per compatibilità con il protocollo datagram OSI. Un'altra fazione avrebbe voluto

indirizzi di lunghezza variabile. Dopo molte discussioni, si decise che indirizzi a lunghezza fissa di 16 byte sarebbero stati il miglior compromesso.

Lo spazio di indirizzamento di IPv6 è suddiviso come mostrato in figura 5-57. Indirizzi che iniziano con 80 zeri sono riservati a indirizzi IPv4. Sono supportate due varianti, distinte dai successivi 16 bit. Queste varianti riguardano il metodo di tunneling di pacchetti IPv6 all'interno di una infrastruttura IPv4 esistente.

Prefisso (binario)	Utilizzazione	Frazione
0000 0000	Riservati (includono IPv4)	1/256
0000 0001	Non assegnato	1/256
0000 001	Indirizzi OSI NSAP	1/128
0000 010	Indirizzi Novell NetWare IPX	1/128
0000 011	Non assegnato	1/128
0000 1	Non assegnato	1/32
0001	Non assegnato	1/16
001	Non assegnato	1/8
010	Indirizzi basati sui fornitori di servizi	1/8
011	Non assegnato	1/8
100	Indirizzi geografici	1/8
101	Non assegnato	1/8
110	Non assegnato	1/8
1110	Non assegnato	1/16
1111 0	Non assegnato	1/32
1111 10	Non assegnato	1/64
1111 110	Non assegnato	1/128
1111 1110 0	Non assegnato	1/512
1111 1110 10	Indirizzi locali di canale	1/1024
1111 1110 11	Indirizzi locali di sito	1/1024
1111 1111	Multicast	1/256

Fig. 5-57 Indirizzi IPv6.

L'uso di prefissi separati per indirizzi basati sui fornitori di servizi e indirizzi geografici è un compromesso tra due differenti visioni del futuro di Internet. Indirizzi basati sul provider hanno senso se si pensa che nel futuro ci sarà un certo numero di società che forniranno servizi Internet ai clienti, analogamente alle attuali società fornitrice di servizi

telefonici come AT&T, MCI, Sprint, British Telecom. A ognuna di queste società verrà fornita qualche frazione dello spazio di indirizzamento. I primi 5 bit che seguono il prefisso 010 sono usati per indicare in quale registro cercare il provider. Attualmente sono operativi 3 registri, per il Nord America, l'Europa e l'Asia. Fino a 29 registri se ne potranno aggiungere in seguito.

Ogni registro è libero di dividere i successivi 15 byte come crede più opportuno. È probabile che molti di essi utilizzeranno un numero di provider di 3 byte, per un totale di circa 16.000.000 di provider, al fine di consentire alle grandi società commerciali di agire come proprio provider. Un'altra possibilità è quella di utilizzare 1 byte per indicare provider nazionali e lasciare che essi allochino ulteriormente lo spazio. In questo modo, livelli addizionali di gerarchia potranno essere introdotti quando necessario.

Il modello geografico è invece uguale a quello attuale di Internet, nel quale i provider non hanno un ruolo importante. In questo modo, IPv6 può gestire entrambi i tipi di indirizzamento.

Gli indirizzi locali di canale e di sito hanno solo significato locale. Possono essere riutilizzati da ogni organizzazione senza conflitti e non possono essere propagati all'esterno di un'organizzazione, rendendoli adatti per essere utilizzati da organizzazioni che attualmente usano un firewall per separarsi dal resto di Internet.

Gli indirizzi di multicast hanno un campo di 4 bit di flag e un campo di 4 bit di ambito subito dopo il prefisso, e quindi un identificatore di gruppo di 112 bit. Uno dei bit di flag distingue i gruppi permanenti dai gruppi transitori. Il campo di ambito permette di limitare un multicast al link, sito, organizzazione o pianeta corrente. Questi 4 ambiti sono sparsi nei 16 valori per permettere di aggiungere nuovi ambiti in seguito. Ad esempio, l'ambito planetario è 14 e quindi il codice 15 è disponibile per permettere future espansioni di Internet ad altri pianeti, sistemi solari e galassie.

Oltre a supportare i normali indirizzi punto-a-punto e gli indirizzi di multicast, IPv6 supporta inoltre un nuovo tipo di indirizzamento: anycast. Un **indirizzo di anycast** è simile a un indirizzo di multicast in quanto la destinazione è un gruppo di indirizzi, ma invece di consegnare il pacchetto a tutti, cerca di consegnare il pacchetto a uno solo, normalmente il più vicino. Ad esempio, per contattare un gruppo di file server cooperanti, un cliente può utilizzare un indirizzo di anycast per raggiungere quello più vicino, senza dover sapere quale sia. Gli indirizzi di anycast utilizzano gli indirizzi regolari punto-a-punto. È compito del sistema di routing scegliere l'host fortunato che riceve il pacchetto. È stata concepita una nuova notazione per scrivere gli indirizzi di 16 byte. Vengono scritti come 8 gruppi di 4 cifre esadecimale separati dal carattere due punti, come in questo esempio:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Poiché in molti indirizzi vi è un gran numero di zeri, sono state autorizzate tre ottimizzazioni. Primo, gli zeri iniziali all'interno di un gruppo possono essere omessi: 0123 viene scritto come 123. Secondo, uno o più gruppi di 16 zeri possono essere rimpiazzati da una coppia di caratteri due punti. Quindi l'indirizzo precedente diventa

8000::123:4567:89AB:CDEF

Infine, gli indirizzi IPv4 possono essere scritti come una coppia di caratteri due punti seguiti dalla vecchia notazione decimale a punti, ad esempio:

::192.31.20.46

Forse non è necessario essere così esplicativi al riguardo, ma è importante notare che esistono moltissimi indirizzi a 16 byte. Per la precisione, ne esistono 2^{128} , che è circa 3×10^{38} . Se la Terra intera, compresi gli oceani, fosse ricoperta di computer, IPv6 permetterebbe 7×10^{23} indirizzi per metro quadrato. Gli studenti di chimica noteranno che questo numero è più grande del numero di Avogadro. Sebbene non vi sia l'intenzione di assegnare a ogni molecola sulla faccia della Terra il suo indirizzo IP, poco ci manca.

In pratica, lo spazio di indirizzamento non verrà utilizzato efficientemente, proprio come lo spazio di indirizzamento dei numeri di telefono (il prefisso di Manhattan, 212, è quasi pieno, mentre il prefisso del Wyoming, 307, è quasi vuoto). In RFC 1715, Huitema calcolò che utilizzando l'allocazione dei numeri di telefono come guida, anche nello scenario più pessimistico ci sarebbero oltre 1000 indirizzi IP per metro quadrato della superficie della Terra (continenti e oceani). In qualsiasi scenario probabile, ci sarebbero trilioni di indirizzi per metro quadrato. Val la pena di notare che solo il 28% dello spazio di indirizzamento è stato allocato fino a ora. Il restante 72% è disponibile per utilizzi futuri non ancora immaginati.

È istruttivo confrontare il preambolo IPv4 (figura 5-45) con il preambolo IPv6 (figura 5-56) per vedere cosa è stato escluso da IPv6. Il campo *IHL* è stato scartato poiché il preambolo IPv6 ha lunghezza fissa. Il campo *Protocol* è stato tralasciato in quanto il campo *Next header* descrive il tipo di dati che segue l'ultimo preambolo IP (ad es., un segmento UDP o TCP).

Tutti i campi collegati alla frammentazione sono stati rimossi, poiché IPv6 assume un approccio differente da essa. Innanzitutto, tutti gli host e i router conformi a IPv6 devono supportare pacchetti di 576 byte. Questa regola rende la frammentazione meno probabile. Inoltre, quando un host invia un pacchetto IPv6 troppo grande, il router che non è in grado di farlo proseguire risponde con un messaggio di errore invece di frammentarlo. Il fatto che l'host invii pacchetti della giusta dimensione è in ultima analisi cosa più efficiente della frammentazione "al volo".

Infine, il campo *Csum* non è più presente, in quanto calcolarlo riduceva enormemente le prestazioni. Data l'affidabilità delle reti oggi in uso, combinata con il fatto che i livelli data link e trasporto hanno normalmente la loro *checksum*, il valore di un ulteriore *checksum* non vale il prezzo delle prestazioni perse. La rimozione di tutte queste caratteristiche ha portato a un protocollo del livello di rete efficiente ed energico. Quindi l'obiettivo di IPv6 – un protocollo veloce e affidabile con uno spazio di indirizzamento enorme – è stato raggiunto da questa progettazione.

Preamboli di estensione

Ciò nonostante, alcuni dei campi mancanti sono ancora occasionalmente necessari e così IPv6 ha introdotto il concetto di preamboli di estensione opzionali. Questi preamboli possono essere utilizzati per fornire informazioni extra, codificate in modo efficiente. Al-

momento sono stati definiti 6 tipi di estensione, elencati in figura 5-58. Sono tutti opzionali, ma se presenti, devono apparire subito dopo il preambolo fisso e preferibilmente nell'ordine della lista.

Preambolo di estensione	Descrizione
Hop-by-hop options	Informazioni varie per i router
Routing	Percorso completo o parziale da seguire
Fragmentation	Gestione dei datagrammi frammentati
Authentication	Verifica dell'identità del mittente
Encrypted security payload	Informazioni sul contenuto codificato
Destination options	Informazioni addizionali sulla destinazione

Fig. 5-58 Preamboli di estensione di IPv6.

Alcuni degli preamboli hanno un formato fisso; altri contengono un numero variabile di campi di lunghezza variabile. Per questo motivo, ogni preambolo è codificato come una tripla (*Type*, *Length*, *Value*) (tipo, lunghezza, valore). *Type* è un campo di 1 byte che indica l'opzione di cui si sta parlando. I valori di *Type* sono stati scelti in modo che i primi 2 bit indichino cosa devono fare i router che non conoscono l'opzione. Le scelte sono: salta l'opzione; scarta il pacchetto, scarta il pacchetto e spedisci indietro un pacchetto ICMP; scarta il pacchetto e spedisci indietro un pacchetto ICMP a esclusione degli indirizzi di multicast (per evitare che un pacchetto di multicast malformato generi milioni di risposte ICMP).

Anche il campo *Length* è lungo 1 byte e contiene la lunghezza del campo *Value* (da 0 a 255 byte). Il campo *Value* contiene le informazioni richieste, fino a 255 byte.

Il preambolo hop-by-hop è utilizzato per informazioni che devono essere esaminate da tutti i router lungo il cammino. Fino a ora, è stata definita un'unica opzione: il supporto per pacchetti più lunghi di 64K. Il formato del preambolo è mostrato in figura 5-59.

Prossimo preambolo	0	194	0
Lunghezza del contenuto del jumbogram			

Fig. 5-59 Il preambolo di estensione hop-by-hop per pacchetti grandi (jumbogram).

Come tutti i preamboli di estensione, anche questo inizia con 1 byte che indica quale tipo di preambolo viene dopo. Segue quindi 1 byte contenente la lunghezza in byte del preambolo hop-by-hop, esclusi i primi 8 byte che sono obbligatori. I successivi 2 byte indicano che questa opzione definisce la grandezza di un pacchetto (codice 194) come un numero a 4 byte. Gli ultimi 4 byte restituiscono la dimensione del pacchetto. Dimensioni

minori di 65.536 non sono permesse e fanno sì che il primo router scarti il pacchetto e spedisca indietro un messaggio di errore ICMP. I pacchetti che utilizzano questo preambolo di estensione vengono chiamati **jumbogram**. L'uso di jumbogram è importante per applicazioni di supercalcolo che devono trasferire gigabyte di dati efficientemente attraverso Internet.

Il preambolo di routing elenca uno o più router che devono essere visitati sul cammino verso la destinazione. Sono disponibili sia il routing "rigido" (viene fornito l'intero cammino) che il routing "flessibile" (sono forniti solo router selezionati), che però sono combinati. Il formato del preambolo di routing è mostrato in figura 5-60.

Prossimo preambolo	0	Numero di indirizzi	Prossimo indirizzo
Mappa di bit			
1 - 24 Indirizzi			

Fig. 5-60 Il preambolo di estensione per il routing.

I primi 4 byte del preambolo di estensione per il routing contengono quattro interi di 1 byte: il tipo del prossimo preambolo, il tipo di routing (normalmente 0), il numero di indirizzi presenti nel preambolo (da 1 a 24) e l'indice del prossimo indirizzo da visitare. L'ultimo campo inizia a 0 e viene incrementato a ogni indirizzo che viene visitato.

Viene poi un byte riservato seguito da una mappa di bit, con un bit per ognuno dei 24 indirizzi IPv6 potenziali che seguono. Dato un indirizzo, il bit corrispondente indica se deve essere raggiunto direttamente da quello precedente (routing rigido dalla sorgente), oppure possono esserci altri router (routing flessibile dalla sorgente).

Il preambolo di frammentazione si occupa della frammentazione esattamente allo stesso modo di IPv4. Il preambolo contiene l'identificatore di pacchetto, il numero di frammento e un bit che dice se seguiranno ulteriori frammenti. In IPv6, diversamente da IPv4, solo l'host sorgente può frammentare un pacchetto. I router lungo il cammino non possono farlo. Sebbene questo cambiamento sia una grande rottura filosofica con il passato, semplifica notevolmente il lavoro dei router e velocizza il routing. Come detto in precedenza, se un router incontra un pacchetto troppo grande, lo scarta e spedisce un pacchetto ICMP alla sorgente. Questa informazione consente all'host sorgente di frammentare il pacchetto in pezzi più piccoli utilizzando questo preambolo e provare ancora.

Il preambolo di autenticazione fornisce un meccanismo con il quale chi riceve un pacchetto può essere sicuro dell'identità del mittente. IPv4 non fornisce questa garanzia. Il preambolo di estensione per la codifica del contenuto rende possibile crittografare il contenuto di un pacchetto in modo che solo il destinatario designato possa leggerlo. Questi preamboli utilizzano tecniche crittografiche per portare a termine la loro missione, che in questo paragrafo descriveremo brevemente. I lettori non abituati alla crittografia moderna potrebbero non comprendere l'intera descrizione. Sarebbe

opportuno rivederla dopo aver letto il capitolo 7 (in particolare il par. 7.1), che descrive i protocolli crittografici.

Quando un mittente e un ricevente desiderano comunicare in sicurezza, devono innanzitutto accordarsi su una o più chiavi segrete che solo loro conoscono. Come questo sia possibile non è di competenza di IPv6. A ognuna di queste chiavi è assegnato un unico numero chiave di 32 bit. I numeri chiave sono globali, per cui se Alice sta usando la chiave 4 per parlare con Bob, non può avere una chiave 4 per comunicare anche con Carol. Associati a ogni numero chiave esistono altri parametri, come ad esempio il tempo di vita.

Per spedire un messaggio autenticato, il mittente prima costruisce un pacchetto consistente di tutti i preamboli IP e del contenuto e quindi rimpiazza i campi che cambiano lungo il cammino (ad es., *Hop limit*) con degli zeri. Il pacchetto è "imbottito" di zeri fino a raggiungere un multiplo di 16 byte. In modo simile, anche la chiave segreta utilizzata è imbottita con zeri fino a raggiungere un multiplo di 16 byte. A questo punto viene calcolata una checksum crittografica della concatenazione della chiave segreta, del pacchetto e ancora della chiave segreta, ognuno dei quali completati a multipli di 16 byte. Gli utenti possono definire il proprio algoritmo di checksum crittografico, sebbene utenti crittograficamente non sofisticati dovrebbero utilizzare l'algoritmo di default, MD5.

A questo punto vediamo il ruolo del preambolo di autenticazione. Sostanzialmente contiene tre parti. La prima parte consiste di 4 byte che contengono l'identificatore del prossimo preambolo, la lunghezza del preambolo di autenticazione e 16 bit zero. Poi viene un numero di chiave di 32 bit. Infine, viene inserita la checksum di MD5 (o altro algoritmo).

Il ricevente deve utilizzare il numero chiave per trovare la chiave segreta. Innanzitutto aggiunge byte di valore 0 al numero chiave e al pacchetto, in entrambi i casi fino a raggiungere un multiplo di 16 byte. Quindi inserisce il numero chiave prima e dopo il pacchetto, nel quale sono stati azzerati i campi variabili. A questo punto calcola la checksum. Se questo coincide con quello incluso nel preambolo di autenticazione, il ricevente può essere sicuro che il pacchetto provenga dal mittente con cui condivide la chiave segreta ed è inoltre sicuro che il pacchetto non sia stato modificato lungo il cammino. Le proprietà di MD5 rendono computazionalmente improbabile che si possa fingere di essere il mittente o modificare il pacchetto senza essere scoperti.

È importante notare che il contenuto dei pacchetti autenticati è spedito in chiaro. Qualsiasi router lungo la strada può leggere cosa contiene. Per molte applicazioni, la segretezza non è molto importante rispetto all'autenticazione. Ad esempio, se un utente ordina alla propria banca di pagare la bolletta del telefono, non c'è probabilmente nessuna necessità di segretezza, ma c'è piuttosto un bisogno reale che la banca sia assolutamente sicura di chi abbia spedito il pacchetto contenente l'ordine di pagamento.

Per pacchetti che devono essere spediti segretamente, viene utilizzato il preambolo di estensione per la codifica del contenuto. Inizia con un numero di chiave a 32 bit, seguito dal contenuto crittografato. L'algoritmo crittografico è lasciato al mittente e al ricevente, ma il default è il DES con cipher block chaining mode (DES-CBC). Quando viene utilizzato DES-CBC, il campo del contenuto inizia con il vettore di inizializzazione (un multiplo di 4 byte); segue quindi il contenuto, infine viene completato a un multiplo di

8 byte. Se si vuole utilizzare sia la segretezza che l'autenticazione, sono necessari entrambi i preamboli.

Il preambolo per opzioni della destinazione è stato concepito per campi che devono essere interpretati unicamente alla destinazione. Nella versione iniziale di IPv6, le uniche opzioni definite sono opzioni nulle per completare questo preambolo a multipli di 8 byte, quindi inizialmente non sono usate. Sono state incluse per essere sicuri che possano essere gestite da nuovi software per i router e gli host, nel caso in cui qualcuno un giorno introduca opzioni della destinazione.

Controversie

Dato il processo di progettazione aperto e la difesa tenace delle opinioni di molte delle persone coinvolte, non dovrebbe essere una sorpresa che gran parte delle scelte riguardanti IPv6 siano state altamente dibattute. In seguito riassumeremo alcune di esse. Per tutti i dettagli, si veda Huitema (1996).

Abbiamo già visto le argomentazioni sulla lunghezza degli indirizzi. Il risultato è stato un compromesso: indirizzi di 16 byte a lunghezza fissa.

Un altro scontro si è sviluppato sulla lunghezza del campo *Hop limit*. Una fazione riteneva che limitare il massimo numero di salti a 255 (derivante dall'uso di 1 byte) fosse un grosso errore. Dopo tutto, cammini di 32 salti sono oggi comuni, e fra 10 anni potrebbero essere comuni cammini molto più lunghi. Costoro sostenevano che utilizzare una dimensione gigantesca degli indirizzi fosse lungimirante, mentre fosse miope usare un piccolo contatore di salti. Nella loro visione, il più grande errore che un informatico avrebbe potuto fare, sarebbe stato quello di prevedere troppi pochi bit da qualche parte.

La risposta fu che queste motivazioni avrebbero potuto essere utilizzate per aumentare la dimensione di qualsiasi campo, generando un preambolo eccessivamente lungo. Inoltre, la funzione del campo *Hop limit* è quella di evitare che un pacchetto resti in circolazione per troppo tempo e 65.536 salti è sicuramente troppo. Infine, con la crescita di Internet, verranno costruiti sempre più canali a lunga distanza, rendendo possibile andare da un paese a qualsiasi altro in mezza dozzina di salti al massimo. Se sono necessari più di 125 salti per andare dalla sorgente e dalla destinazione ai rispettivi gateway internazionali, qualcosa è andato storto nelle backbone nazionali. I fautori degli 8 bit vinsero questa battaglia.

Un altro problema da superare fu la dimensione massima del pacchetto. La comunità che utilizza supercomputer voleva pacchetti più grandi di 64 KB. Quando un supercomputer inizia a trasferire dati, fa sul serio e non intende essere interrotto ogni 64 KB. L'argomentazione contro pacchetti veramente grandi è la seguente: se un pacchetto di 1 MB capita su una linea T1 a 1.5 Mbps, essa viene occupata per 5 s, producendo un ritardo raggardevole per gli utenti interattivi che condividono la linea. Qui è stato raggiunto un compromesso: i pacchetti normali sono limitati a 64 KB, mentre il preambolo di estensione hop-by-hop permette di spedire jumbogram.

Un altro importante argomento riguardava la rimozione della checksum IPv4. Alcune persone paragonavano questa mossa alla rimozione dei freni da un'auto: leggerezza e velocità contro l'imprevisto.

L'argomentazione contro la checksum era la seguente: qualsiasi applicazione che realmente si preoccupasse dell'integrità dei dati avrebbe dovuto in ogni caso avere una checksum a livello di trasporto e quindi un'altra nell'IP (oltre alla checksum del livello data link) sarebbe stata eccessiva. Inoltre, l'esperienza dimostrava come calcolare la checksum di IP fosse stato uno dei costi maggiori in IPv4. La fazione anti-checksum vinse, e IPv6 non esegue checksum. Gli host mobili furono un ulteriore punto di disputa. Se un computer portatile viaggia per mezzo mondo, può continuare a funzionare presso la destinazione con lo stesso indirizzo IPv6 o deve utilizzare qualche schema con agenti base e agenti stranieri? Gli host mobili introducono anche asimmetrie nel sistema di routing. È sempre possibile che un piccolo computer portatile possa sentire il segnale emesso da un grande router fisso e che il router fisso non possa sentire il flebile segnale emesso dall'host mobile. Conseguentemente, alcune persone volevano inserire supporto esplicito per gli host mobili in IPv6. Questo sforzo fallì quando non si poté trovare nessun accordo su qualche proposta specifica.

Probabilmente la battaglia più grande fu a proposito della sicurezza. Tutti erano d'accordo sulla sua necessità. La battaglia era sul dove e sul come. Innanzitutto, dove. Fra le argomentazioni a favore dell'inserimento nel livello rete, che sarebbe diventato un servizio standard utilizzabile da tutte le applicazioni senza progettazione anticipata. Di contro, invece si soteneva che in realtà le applicazioni sicure generalmente non richiedevano nient'altro che crittografia punto-a-punto, con l'applicazione sorgente che codifica e l'applicazione destinazione che decodifica. Al contrario, in questo inserimento l'utente sarebbe stato alla mercè di implementazioni del livello di rete potenzialmente scorrette, sulle quali non avrebbe avuto controllo. La risposta a questa argomentazione fu la seguente: queste applicazioni avrebbero potuto comunque evitare di usare la sicurezza di IP e fare da sole. La replica fu che le persone che non si fossero fidate del comportamento della rete non avrebbero voluto dover pagare il prezzo di implementazioni IP lente e ingombranti con questa capacità, anche se disabilitata.

Un altro problema dibattuto concernente la sicurezza, fu il fatto che molti stati (ma non tutti) hanno rigide leggi concernenti l'esportazione di crittografia. Alcuni, fra cui Francia e Iraq, restringono fortemente l'uso domestico, in modo che la gente non possa avere segreti per la polizia. Come risultato, qualsiasi implementazione IP che utilizzi sistemi crittografici sufficientemente forti non può essere esportata dagli Stati Uniti (e da molti altri paesi) a clienti sparsi nel mondo. Avere due insiemi di software, uno per l'uso domestico e l'altro per l'esportazione è qualcosa a cui si oppongono molti produttori di computer.

Una possibile soluzione è quella che tutti i venditori spostino i propri punti di vendita di crittografia in paesi che non la regolano, come la Finlandia e la Svizzera. Il software crittografico può essere progettato e realizzato in questi paesi e poi venduto legalmente a tutti i paesi, esclusi la Francia e l'Iraq. Il problema di questo approccio è il fatto che progettare parte del software di routing in un paese e parte in un altro può portare a problemi di integrazione.

L'ultima controversia riguardante la sicurezza era collegata alla scelta dell'algoritmo di default che tutte le implementazioni avrebbero dovuto supportare. Sebbene MD5 fosse ritenuto sufficientemente sicuro, i continui progressi nella crittografia avrebbero potuto indebolirlo. Nessun esperto di crittografia crede che DES sia sicuro contro gli attacchi

dei governi più importanti, ma è probabilmente abbastanza buono da sgominare i più precoci dodicenni di oggi. Il compromesso fu quindi quello di affidare la sicurezza a IPv6, utilizzare un algoritmo di checksum allo stato dell'arte per una buona autenticazione e utilizzare un algoritmo deboluccio per la segretezza, lasciando all'utente la possibilità di cambiare questi algoritmi con i propri.

Un punto su cui non ci fu alcuna controversia è il seguente: nessuno si attende che Internet basata su IPv4 sia spenta una domenica mattina e sia riaccessa il lunedì mattina come Internet basata su IPv6. Invece, verranno convertiti settori isolati di IPv6, che inizialmente comunicheranno tramite tunnel. Crescendo, settori distinti di IPv6 convergeranno in settori più grandi. Prima o poi, tutti i settori si riuniranno, e Internet sarà convertita completamente. Dati i grandi investimenti attualmente spiegati nel campo dei router IPv4, il processo di conversione richiederà probabilmente un decennio. Per questa ragione, è stato fatto un grande sforzo affinché la transizione sia il più possibile indolore.

5.6 Il livello rete nelle reti ATM

I livelli del modello ATM (vedi fig. 1.30) non trovano una precisa corrispondenza nei livelli OSI, e ciò causa alcune ambiguità. Il livello data link di OSI si occupa della gestione dei protocolli di frammentazione e di trasferimento tra due macchine collegate dallo stesso cavo fisico (o fibra). I protocolli del livello data link sono basati su singolo salto. Non si occupano di connessione end-to-end, in quanto l'instradamento non avviene nel livello data link. Su questo punto non ci sono dubbi.

Il livello più basso che collega sorgente e destinazione, e quindi richiede instradamento (vale a dire è multi-hop), è il livello rete. Il livello ATM gestisce il trasferimento di celle dalla sorgente alla destinazione e richiede senza dubbio algoritmi e protocolli di instradamento all'interno dei commutatori ATM. Gestisce inoltre l'indirizzamento globale. Quindi il livello ATM assolve funzionalmente i compiti del livello rete. Il livello ATM non è affidabile al 100%; d'altronde, questo non è richiesto da un protocollo del livello rete.

Inoltre, il livello ATM assomiglia al livello 3 di X.25, su cui tutti concordano che sia un protocollo del livello rete. In funzione della disposizione dei bit, il protocollo rete di X.25 può essere affidabile oppure no, sebbene la gran parte delle implementazioni lo consideri inaffidabile. Poiché il livello ATM possiede le funzionalità attese in un livello rete, non possiede le funzionalità attese in un livello data link ed è molto simile ai protocolli del livello rete esistenti, discuteremo del livello ATM in questo capitolo.

Molta confusione nasce dal fatto che molte persone nella comunità ATM considerano il livello ATM come un livello data link, o addirittura un livello fisico quando emula una LAN. Molta gente nella comunità Internet lo considera un livello data link, in quanto vogliono collocare il protocollo IP sopra di esso, e considerare il livello ATM come un livello data link si adatta bene a questa idea. (Sebbene, seguendo questa linea di ragionamento, per la comunità Internet *tutte* le reti operino come strati data link, indipendentemente dalle loro caratteristiche).

L'unico problema è il fatto che il livello ATM non ha le caratteristiche di un protocollo data link, ovvero di un protocollo basato su singolo salto usato da computer agli estremi

5.6 Il livello rete nelle reti ATM

opposti di un cavo, quali i protocolli 1-6 del capitolo 3. Ha invece le caratteristiche di un protocollo del livello rete: utilizzazione di circuiti virtuali end-to-end e gestione di commutatori e router.

L'autore si ricorda di un vecchio indovinello:

D: Quante zampe ha un mulo se si chiama zampa anche la coda?

R: Quattro. *Chiamare* zampa anche la coda non la *rende* una zampa.

Basti dire che il lettore è avvertito della controversia, alquanto irrazionale. Il livello ATM è orientato alla connessione, sia in termini di servizi offerti sia nel modo in cui opera internamente. L'elemento base del livello ATM è il circuito virtuale (chiamato ufficialmente **virtual channel**, o canale virtuale). Generalmente, un circuito virtuale è una connessione da una sorgente a una destinazione, sebbene siano consentite anche connessioni di tipo multicast. I circuiti virtuali sono unidirezionali, ma una coppia di circuiti può essere creata nello stesso istante. Entrambe le parti della coppia sono individuate dallo stesso identificatore e quindi i circuiti virtuali sono in pratica full duplex. Tuttavia, sia la capacità del canale che altre proprietà possono essere differenti nelle due direzioni e possono anche essere 0 per una di esse.

Il livello ATM si diversifica da altri protocolli orientati alla connessione in quanto non fornisce alcuna conferma dell'avvenuta ricezione. Questa scelta è motivata dal fatto che ATM fu progettato per essere utilizzato in reti basate su fibre ottiche, le quali sono altamente affidabili. Si credette opportuno lasciare il controllo dell'errore ai livelli più alti. Dopo tutto, spedire i messaggi di conferma nei livelli data link e rete è in pratica solo un'ottimizzazione. Per il livello di trasporto, è sufficiente spedire il messaggio e quindi rispedirlo nuovamente se la ricezione non viene confermata in tempo.

Inoltre, le reti ATM sono spesso usate per traffico in tempo reale, come connessioni audio/video. Per questo tipo di traffico, ritrasmettere occasionalmente una cella persa o corrotta è peggio che ignorarla del tutto.

Nonostante la mancanza di conferme esplicite, il livello ATM fornisce una garanzia molto forte: le celle spedite lungo un circuito virtuale non arrivano mai fuori ordine. La rete ATM può scartare celle in caso di congestione, ma in nessun caso può riordinare le celle spedite lungo un singolo circuito virtuale. Tuttavia, non viene fornita nessuna garanzia per quanto riguarda celle spedite lungo circuiti virtuali *differenti*. Ad esempio, se un host spedisce alla stessa destinazione una cella nel circuito virtuale 10 e successivamente una cella nel circuito virtuale 20, la seconda cella può arrivare prima. Se le due celle fossero state spedite lungo lo stesso circuito virtuale, la prima spedita sarebbe sempre la prima a essere ricevuta.

Il livello ATM supporta una gerarchia di connessione a due livelli che è visibile al livello di trasporto. Lungo qualunque cammino di trasmissione da una certa sorgente a una certa destinazione, un insieme di circuiti virtuali può essere raggruppato in quello che viene detto **virtual path** (percorso virtuale), come illustrato in figura 5-61. Concettualmente, un percorso virtuale è simile a un fascio di doppini telefonici attorcigliati: quando viene reinstradato, tutti i doppini (circuiti virtuali) vengono reinstradati assieme. Le implicazioni di questa progettazione a due livelli verranno più avanti illustrate in dettaglio.

5.6.1 Formati delle celle

Nel livello ATM vengono distinte due interfacce: l'interfaccia UNI (User-Network Interface – interfaccia utente-rete) e l'interfaccia NNI (Network-Network Interface – interfaccia rete-rete). La prima definisce il confine tra un host e una rete ATM (in molti casi, tra il cliente e il fornitore). La seconda si applica alla linea tra due commutatori ATM (il termine ATM sta per router).

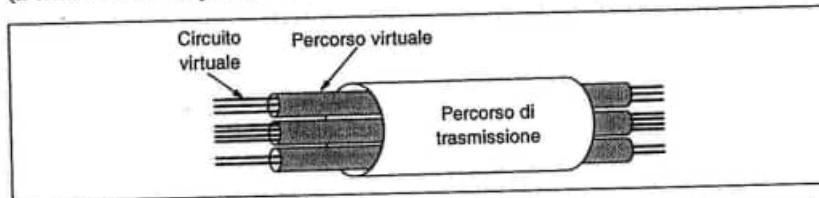


Fig. 5-61 Un percorso di trasmissione può contenere percorsi virtuali multipli, ognuno dei quali può essere composto da circuiti virtuali multipli.

In entrambi i casi le celle sono formate da un header di 5 byte seguiti da 48 byte di contenuto, sebbene i due header siano leggermente differenti. I preamboli definiti dal Forum ATM sono illustrati in figura 5-62. Le celle sono trasmesse con il byte più a sinistra per primo; all'interno di un byte, con il bit più a sinistra per primo.

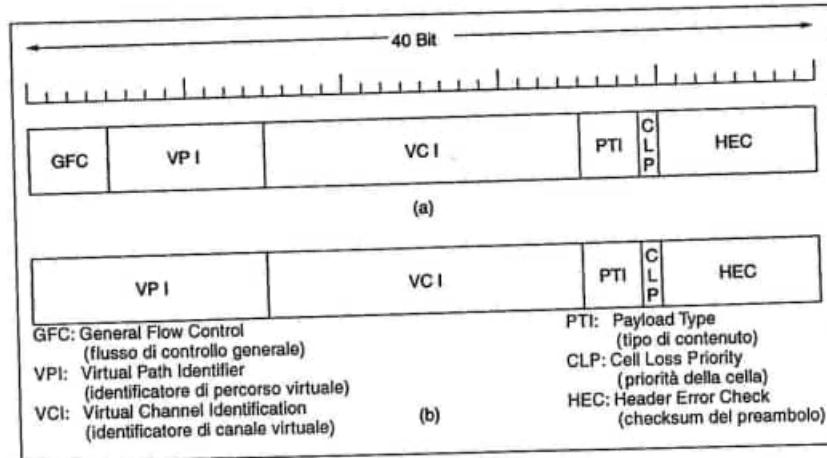


Fig. 5-62 (a) Il preambolo ATM presso la UNI. (b) Il preambolo ATM presso la NNI.

Il campo *GFC* è presente solo nelle celle scambiate tra un host e la rete. Viene sovrascritto dal primo commutatore che raggiunge, quindi non ha significato e non viene consegnato alla destinazione. Venne concepito originariamente forse per essere utilizzato nel controllo di flusso o di priorità tra gli host e la rete, ma nessun valore è

5.6 Il livello rete nelle reti ATM

stato definito per esso e la rete lo ignora. La cosa migliore da fare è pensarlo come un errore nello standard.

Il campo *VPI* è un piccolo valore intero che seleziona un particolare percorso virtuale (vedi figura 5-61). In modo simile, il campo *VCI* seleziona un particolare circuito virtuale all'interno del percorso virtuale scelto. Poiché il campo *VPI* è composto da 8 bit (presso la UNI) e il campo *VCI* è composto da 16 bit, un host potrebbe teoricamente possedere fino a 256 percorsi virtuali, ognuno dei quali potrebbe contenere fino a 65.536 circuiti virtuali. In realtà ne è disponibile un numero leggermente inferiore in quanto alcuni *VCI* sono riservati a funzioni di controllo, quali la creazione di circuiti virtuali.

Il campo *PTI* definisce il tipo di contenuto nelle celle, in accordo con i valori di figura 5-63. I tipi di cella sono forniti dall'utente, le informazioni di congestione, dalla rete. In altre parole, una cella spedita con *PTI* 000 potrebbe arrivare con *PTI* 010 per informare la destinazione di problemi lungo lo strada.

Tipo del contenuto	Significato
000	Cella dati utente, niente congestione, tipo 0
001	Cella dati utente, niente congestione, tipo 1
010	Cella dati utente, presenza di congestione, tipo 0
011	Cella dati utente, presenza di congestione, tipo 1
100	Informazioni scambiate tra switch adiacenti
101	Informazioni scambiate tra switch sorgente e destinazione
110	Cella di gestione risorse (utilizzata per il controllo di congestione in ABR)
111	Riservato a funzioni future

Fig. 5-63 Valori del campo *PTI*.

Il bit *CLP* può essere modificato da un host per differenziare tra traffico ad alta o bassa priorità. Se avviene una congestione e alcune celle devono essere scartate, i commutatori tentano di scartare le celle con *CLP* uguale a 1 prima di quelle con *CLP* uguale a 0.

Infine, il campo *HEC* è una checksum del preambolo. Il contenuto del pacchetto non viene controllato. Un codice Hamming su un numero di 40 bit richiede solamente 5 bit, quindi con 8 bit può essere utilizzato un codice più sofisticato. Quello scelto può correggere tutti gli errori su bit singolo e il 90% degli errori su più bit. Numerosi studi hanno evidenziato come la grande maggioranza degli errori sui collegamenti ottici riguardino un singolo bit.

Dietro il preambolo vengono collocati i 48 byte del contenuto. Non tutti i 48 byte sono disponibili per l'utente, in quanto alcuni protocolli AAL inseriscono i loro preamboli e ulteriori informazioni nel contenuto.

Il formato NNI è uguale al formato UNI, a parte l'assenza del campo *GFC*, i cui 4 bit sono utilizzati per rendere il campo *VPI* di 12 bit anziché di 8.

5.6.2 Creazione delle connessioni

ATM supporta sia i circuiti virtuali permanenti, sia i circuiti virtuali commutati. I primi sono sempre presenti e possono essere utilizzati a piacimento, come le linee affittate. I secondi devono essere stabiliti ogni volta che vengono usati, come nel caso delle chiamate telefoniche. In questo paragrafo descriveremo come vengono stabiliti i circuiti virtuali commutati. Tecnicamente, la creazione delle connessione non fa parte del livello ATM, ma è gestita dal control plane (vedi figura 1-30) utilizzando un complesso protocollo ITU chiamato Q. 2931 (Stiller, 1995). Ciò nonostante, la posizione logica dove gestire la creazione di una connessione è il livello rete, e protocolli rete simili gestiscono la creazione delle connessioni, per cui discuteremo di essa in questo capitolo.

Esistono molti modi per creare una connessione. Il modo normale consiste nell'acquisire un circuito virtuale per i segnali e utilizzarlo. Per stabilire tale circuito, quelle contenenti la richiesta vengono spedite nel percorso virtuale 0, circuito virtuale 5. In caso di successo viene aperto un nuovo circuito virtuale nel quale spedire le richieste di connessione e ricevere le relative risposte.

Questa procedura di creazione suddivisa in due parti è motivata dal fatto che in questo modo è minima la banda riservata al circuito virtuale 5 (il quale viene usato pochissimo). Inoltre, esiste un modo alternativo per stabilire circuiti virtuali. Alcuni fornitori consentono agli utenti di possedere percorsi virtuali permanenti con destinazioni predefinite, oppure di crearli dinamicamente. Una volta che un host possiede un percorso virtuale con qualche altro host, può autonomamente allocare circuiti virtuali su di esso, senza coinvolgere i commutatori.

La creazione di circuiti virtuali utilizza i sei tipi di messaggi elencati in figura 5-64. Ogni messaggio occupa una o più celle e contiene il tipo di messaggio, la sua lunghezza e alcuni parametri. I messaggi possono essere spediti da un host alla rete o dalla rete a un host (generalmente in risposta a un messaggio di un altro host). Esistono altri messaggi di stato o di errore, che però non sono elencati qui.

Messaggio	Significato quando è spedito da un host	Significato quando è spedito dalla rete
SETUP	Richiesta di stabilire un circuito	Chiamata in arrivo
CALL PROCEEDING	Ho visto la chiamata in arrivo	La tua richiesta verrà inoltrata
CONNECT	Accetto la chiamata in arrivo	La tua richiesta è stata accettata
CONNECT ACK	Grazie per aver accettato	Grazie per aver chiamato
RELEASE	Richiesta di chiusura del circuito	L'altro lato vuole chiudere
RELEASE COMPLETE	Conferma per RELEASE	Conferma per RELEASE

Fig. 5-64 Messaggi utilizzati per stabilire e interrompere connessioni.

La procedura normale per iniziare una chiamata da parte di un host consiste nello spedire un messaggio SETUP (creazione) su uno speciale circuito virtuale. La rete risponde con un messaggio CALL PROCEEDING (chiamata in corso) per confermare la ricezione della

5.6 Il livello rete nelle reti ATM

richiesta. Mentre il messaggio SETUP si propaga fino alla destinazione, ogni salto viene confermato da un messaggio CALL PROCEEDING.

Quando finalmente il messaggio SETUP arriva, la destinazione può rispondere con CONNECT (connessione) per accettare la chiamata. La rete quindi spedisce un messaggio CONNECT ACK (conferma di connessione) per indicare che ha ricevuto il messaggio CONNECT. Mentre un messaggio CONNECT si propaga all'indietro fino all'origine, ogni commutatore che lo riceve spedisce un messaggio CONNECT ACK come risposta. Questa sequenza di eventi è mostrata in figura 5-65 (a).

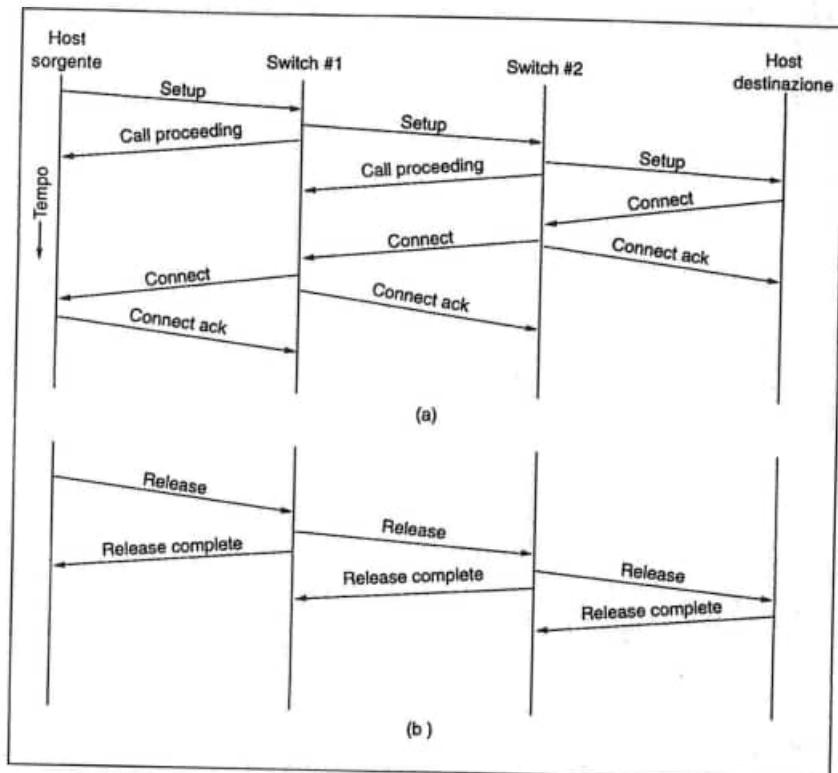


Fig. 5-65 (a) Creazione di una connessione in una rete ATM. (b) Interruzione di una connessione.

La sequenza di messaggi per terminare un circuito virtuale è semplice. L'host che desidera smettere spedisce un messaggio RELEASE (chiusura), che si propaga fino all'altro estremo e causa l'interruzione del circuito. La ricezione del messaggio viene confermata a ogni salto, come mostrato in figura 5-65 (b).

Le reti ATM consentono di creare canali di tipo multicast. Un canale multicast ha un solo mittente e più di un ricevente. Questi canali vengono costruiti creando una connessione con una delle destinazioni nel solito modo. Quindi viene spedito un messaggio ADD PARTY (aggiunta interlocutore) per aggiungere una seconda destinazione al circuito virtuale restituito dalla precedente chiamata. Messaggi ADD PARTY addizionali possono essere spediti in seguito per aumentare la dimensione del gruppo di multicast.

Al fine di creare una connessione con una destinazione, è necessario specificare quest'ultima includendo il suo indirizzo nel messaggio SETUP. Esistono tre tipi di indirizzi ATM. Il primo è lungo 20 byte ed è basato sugli indirizzi OSI. Il primo byte indica in quale dei tre formati è l'indirizzo. Nel primo formato, i byte 2 e 3 indicano un paese, mentre il byte 4 definisce il formato del resto dell'indirizzo, il quale contiene 3 byte per l'autorità, 2 byte per il dominio, 2 byte per l'area e 6 byte per l'indirizzo, più altre informazioni. Nel secondo formato, i byte 2 e 3 designano un'organizzazione internazionale al posto di uno stato. Il resto dell'indirizzo è uguale al formato 1. Alternativamente, viene permesso un vecchio formato di indirizzamento (CCITT E.164) che utilizza numeri telefonici ISDN a 15 cifre.

5.6.3 Instradamento e commutazione

Al momento di creare un circuito virtuale, il messaggio SETUP percorre la rete dalla sorgente alla destinazione. L'algoritmo di routing determina il percorso preso da questo messaggio, e quindi dal circuito virtuale. Lo standard ATM non specifica nessun algoritmo di instradamento particolare, quindi il fornitore è libero di scegliere fra gli algoritmi discussi in precedenza in questo capitolo, o di usarne uno differente.

L'esperienza con reti orientate alla connessione quali X.25 ha mostrato che una quantità considerevole di potenza di calcolo nei commutatori può essere sprecata determinando come convertire nella scelta di una linea di output le informazioni riguardanti il circuito virtuale presenti in ogni cella. I progettisti ATM volevano evitare questo destino, e così il livello ATM è stato progettato per rendere possibile un routing efficiente. In particolare l'idea è stata quella di scegliere il percorso sulla base del campo *VPI* e non su *VCI*, a parte nell'ultimo salto, nel quale le celle vengono spedite tra un commutatore e un host. Tra due commutatori deve essere utilizzato solo il percorso virtuale.

Utilizzare unicamente i *VPI* tra commutatori interni ha molti vantaggi. Primo, una volta che sia stato stabilito un percorso virtuale da una sorgente a una destinazione, ogni circuito virtuale aggiuntivo lungo questo percorso può seguire il cammino esistente. Non devono essere compiute nuove scelte di routing. È come se un fascio di doppini telefonici fosse già stato collocato tra la sorgente e la destinazione. Creare una nuova connessione richiede unicamente l'allocazione di uno dei doppini inutilizzati. Secondo, il routing di celle individuali è più semplice quando tutti i circuiti virtuali per un dato percorso sono sempre nello stesso fascio. Le decisioni di routing si basano unicamente su 12 bit e non su 12+16 bit: dovrebbe essere chiaro che indicizzare una tabella di 2^{12} elementi è fattibile, mentre indicizzare una tabella di 2^{28} elementi non lo è. Terzo, basare l'intero routing sui percorsi virtuali rende più facile commutare un intero gruppo di circuiti virtuali. Si consideri, ad esempio, l'ipotetica rete ATM degli USA illustrata in figura 5-66. Normalmente, i circuiti virtuali tra NY e SF passano attraverso Omaha e Denver. Tuttavia, supponiamo che si presenti un problema sulla linea Omaha-Denver. Reinstradando il percorso virtuale Omaha-Denver verso LA e poi SF, tutti i circuiti virtuali (potenzialmente fino a 65.535) possono essere commutati in un'unica operazione al posto di migliaia di operazioni potenziali.

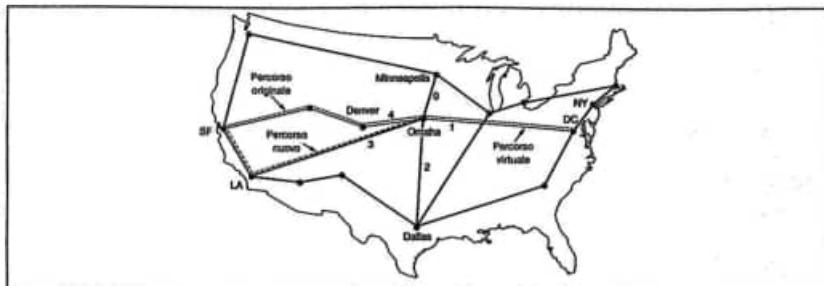


Fig. 5-66 Reinstradare un percorso virtuale modifica il percorso di tutti i suoi circuiti virtuali.

Infine, i percorsi virtuali rendono più facile fornire a grandi società gruppi chiusi di utenti (reti private). Una società può creare una rete di percorsi virtuali permanenti tra i propri uffici, e quindi allocare su richiesta circuiti virtuali all'interno di questi percorsi. Nessuna chiamata esterna può entrare nella rete privata, né uscirne, a parte attraverso gateway speciali. Resta da vedere se i commutatori utilizzeranno effettivamente il campo *VPI* per il routing come progettato, oppure se utilizzeranno la combinazione dei campi *VPI* e *VCI* (rinunciando quindi a tutti i vantaggi appena discussi).

Vediamo ora come le celle possono essere instradate in un commutatore interno (connesso cioè solo ad altri commutatori e non a host). Ad esempio, si consideri il commutatore di Omaha in figura 5-66. Tale commutatore possiede una tabella detta *vpi_table* per ognuna delle 5 linee entranti; questa tabella è indicizzata dai *VPI* entranti, e indica quale *VPI* e quale delle 5 linee utilizzare per le celle in uscita. Assumiamo che le cinque linee siano numerate da 0 a 4 in senso orario partendo da Minneapolis. Per ogni linea in uscita, il commutatore mantiene una mappa di bit con lo scopo di indicare quali *VPI* sono attualmente utilizzati lungo quella linea. Quando il commutatore viene inizializzato, tutti gli elementi nell'intera tabella *vpi_table* vengono marcati come non utilizzati. In modo simile, tutte le mappe di bit sono marcate in modo da indicare che tutti i *VPI* sono disponibili (a parte quelli riservati). Ora supponiamo che arrivino delle chiamate come mostrato in figura 5-67. Alla creazione di ogni percorso virtuale (e circuito virtuale),

Sorgente	Linea entrante	VPI entrante	Destinazione	Linea uscente	VPI uscente	Cammino
NY	1	1	SF	4	1	New
NY	1	2	Denver	4	2	New
LA	3	1	Minneapolis	0	1	New
DC	1	3	LA	3	2	New
NY	1	1	SF	4	1	Old
SF	4	3	DC	1	4	New
DC	1	5	SF	4	4	New
NY	1	2	Denver	4	2	Old
SF	4	5	Minneapolis	0	2	New
NY	1	1	SF	4	1	Old

Fig. 5-67 Alcuni percorsi attraverso il commutatore di Omaha di fig. 5-66.

vengono create le corrispondenti registrazioni. Assumeremo che i circuiti virtuali siano full duplex, in modo che a ogni connessione corrispondano due registrazioni, una per il traffico in andata dalla sorgente e una per il traffico di ritorno dalla destinazione.

Le tabelle corrispondenti ai percorsi di figura 5-67 sono mostrate in figura 5-68. Ad esempio, la prima chiamata genera la registrazione (4, 1) per *VPI* 1 nella tabella di DC, in quanto si riferisce a celle che arrivano dalla linea 1 con *VPI* 1 destinate a SF. Inoltre, viene creata una registrazione nella tabella di Denver per *VPI* 1, indicante che le celle in arrivo a Denver con *VPI* 1 devono uscire dalla linea 1 con *VPI* 1. Queste sono le celle che hanno direzione opposta (da SF a NY) su questo percorso virtuale. Si noti che in qualche caso due o tre circuiti virtuali condividono un percorso comune. Non sono richieste nuove registrazioni nella tabella per circuiti virtuali addizionali che connettano una sorgente e una destinazione a cui è già stato assegnato un percorso.

A questo punto possiamo spiegare come vengono elaborate le celle all'interno di un commutatore. Supponiamo che una cella arrivi lungo la linea 1 (DC) con *VPI* 3. Il commutatore (hardware oppure software) utilizza 3 come indice nella tabella associata alla linea 1 e osserva che le celle devono uscire lungo la linea 3 (LA) con *VPI* 2. Quindi sovrascrive il campo *VPI* con un 2 e inserisce il numero di linea di uscita (3) da qualche parte nella cella, ad esempio nel campo *HEC*, poiché deve essere ricalcolato in ogni caso più tardi. Ora il problema è come spostare la cella dal suo input buffer corrente alla linea 3 (cfr. cap. 2, in particolare pp. 145-49).

A questo punto è intuitivo capire come può essere reinstradato un intero fascio di circuiti virtuali, come in figura 5-66. Cambiando la registrazione per *VPI* 1 nella tabella di DC da (4, 1) a (3, 3), le celle di NY destinate a SF verranno deviate per LA. Naturalmente, il commutatore di LA deve essere informato di questo evento; quindi il commutatore deve generare e spedire un messaggio *SETUP* a LA per stabilire un nuovo percorso con *VPI* 3. Una volta creato questo percorso, tutti i circuiti virtuali da NY a SF sono deviati attraverso LA, anche nel caso in cui siano migliaia. In assenza di percorsi virtuali, ogni circuito virtuale avrebbe una registrazione nella tabella e dovrebbe essere reinstradato separatamente. È importante notare che la discussione precedente riguarda ATM nelle WAN. In una LAN il discorso è più semplice. Ad esempio, può essere utilizzato un singolo percorso virtuale per tutti i circuiti virtuali.

VPI_table for Minn.		VPI_table for DC		VPI_table for Dallas		VPI_table for LA		VPI_table for Denver	
Entrante VPI	Uscente Linea VPI	Entrante VPI	Uscente Linea VPI	Entrante VPI	Uscente Linea VPI	Entrante VPI	Uscente Linea VPI	Entrante VPI	Uscente Linea VPI
0						0	1	1	1
1	3 1	4 1				1	2		
2	4 5	4 2				1	4		
3		3 2				1	5		
4		4 3				0	2		
5		4 4							
6									
7									
8									
4095									
	Line 0	Line 1	Line 2	Line 3	Line 4				

Fig. 5-68 Le tabelle relative ai percorsi della fig. 5-67.

5.6.4 Categorie di servizi

Dopo un legittimo numero di tentativi ed errori, dalla versione 4.0 della specifica ATM si è compreso quale tipo di traffico le reti ATM stessero supportando e quali tipi di servizi fossero richiesti dai clienti. Di conseguenza, lo standard fu modificato per elencare in maniera esplicita le categorie di servizio comunemente in uso, al fine di consentire ai produttori di hardware di ottimizzare le loro schede di interfaccia e i loro commutatori per alcune o tutte queste categorie (fig. 5-69).

Classe	Descrizione	Esempio
CBR	Constant bit rate	Circuito T1
RT-VBR	Variabile bit rate: a tempo reale	Videoconferenza in tempo reale
NRT-VBR	Variabile bit rate: a tempo non reale	e-mail multimediale
ABR	Available bit rate	Navigare per il Web
UBR	Unspecified bit rate	Trasferimento file in background

Fig. 5-69 Le categorie del servizio ATM.

La classe **CBR** (Constant Bit Rate – velocità costante di trasmissione bit) è stata concepita per emulare un cavo di rame o una fibra ottica (solamente a un costo molto minore). I bit sono inseriti a un estremo ed escono dall'altro. Non vengono effettuati nessuna verifica di errore, nessun controllo di flusso o altri tipi di elaborazione. Tuttavia, questa classe è essenziale per realizzare una transizione morbida dagli attuali sistemi telefonici ai futuri sistemi B-ISDN, in quanto canali PCM voice-grade, circuiti T1 e gran parte dei restanti sistemi telefonici utilizzano trasmissioni sincrone a velocità costante. Con la classe CBR, tutto questo traffico può essere trasportato direttamente da un sistema ATM. CBR è inoltre adatto per tutti gli altri flussi di dati audio/video interattivi (cioè in tempo reale).

La classe successiva, **VBR** (Variable Bit Rate – velocità variabile di bit) è suddivisa in due sottoclassi, rispettivamente per tempo reale e non. **RT-VBR** è destinato a servizi che abbiano una frequenza di bit variabile combinata con stringenti esigenze in tempo reale, come video interattivo compresso (cioè videoconferenze). A causa del modo in cui lavorano MPEG e altri schemi di compressione, con un'immagine base completa seguita da una serie di differenze tra l'immagine corrente e l'immagine base, la frequenza di trasmissione varia fortemente nel tempo (Pancha, El Zarki, 1994). Nonostante queste variazioni, è essenziale che la rete ATM non introduca nessun jitter nella sequenza di arrivo delle celle, in quanto questo renderebbe il video a scatti. In altre parole, sia il ritardo medio delle celle che la varianza nel ritardo delle celle devono essere rigidamente controllati. D'altra parte, la perdita di una cella di tanto in tanto è in questo caso tollerabile ed è quindi meglio che sia ignorata. L'altra sottoclasse **VBR** riguarda il tipo di traffico in cui la consegna tempestiva è importante, ma un certo ammontare di jitter può essere tollerato dall'applicazione. Ad esempio, la posta elettronica multimediale viene registrata nel disco locale del ricevente prima di essere visualizzata; quindi, ogni variazione nei tempi di consegna delle celle viene eliminata prima che il messaggio sia letto.

La categoria di servizi ABR (Available Bit Rate – frequenza di bit disponibile) è progettata per traffico caratterizzato da picchi, in cui la larghezza di banda richiesta è conosciuta solo approssimativamente. Un esempio tipico di utilizzazione potrebbe essere una società che connette i propri uffici attraverso un insieme di linee affittate. Generalmente, la società potrebbe scegliere di dedicare una capacità sufficiente per gestire il carico di picco (e ciò significa che alcune linee sono inutilizzate per parte della giornata), oppure dedicare una capacità sufficiente per il carico minimo, che può causare congestioni durante la parte più utilizzata della giornata.

Utilizzare servizi ABR evita di dover impegnare a lungo termine una larghezza di banda fissa. Tramite ABR è possibile dire, ad esempio, che la capacità tra due punti deve essere sempre di 5 Mbps, ma può avere picchi fino a 10 Mbps. Il sistema garantirà sempre quindi 5 Mbps, e farà del suo meglio per fornire 10 Mbps quando necessari, ma senza garanzie.

ABR è l'unica categoria di servizio nella quale la rete fornisca un feed-back sulla velocità al mittente, chiedendogli di rallentare in caso di congestione. Assumendo che il mittente soddisfi tali richieste, si presuppone che la perdita di celle per il traffico ABR sia bassa. Viaggiare tramite ABR è come volare come passeggero di riserva: se sono rimasti posti (capacità in eccesso), i passeggeri in attesa vengono trasportati senza ritardi. Se esiste capacità insufficiente, devono aspettare (a meno che non sia disponibile una parte della banda minima).

Infine, arriviamo a UBR (Unspecified Bit Rate – frequenza di bit non specificata), che non fa alcuna promessa e non fornisce feed-back sulla congestione. Questa categoria è adatta per la spedizione di pacchetti IP, in quanto anche IP non fa promesse sulla consegna. Tutte le celle UBR vengono accettate, e se è rimasta capacità sufficiente, vengono anche consegnate. Se si presenta una congestione, le celle UBR verranno scartate, senza alcun feed-back al mittente e senza attendersi che il mittente rallenti.

Per continuare la nostra analogia sul trasporto aereo, con UBR i passeggeri in attesa vengono presi a bordo, ma se a metà strada per la destinazione il pilota si accorge che il carburante sta per finire, i passeggeri in eccesso vengono buttati fuori dall'uscita di emergenza senza ceremonie. Per rendere UBR attraente, è probabile che i fornitori lo rendano meno costoso di altre classi. Per applicazioni che non hanno vincoli sulla consegna e vogliono in ogni caso fare i propri controlli di errore e di flusso, UBR è una scelta perfettamente ragionevole. Trasferimento file, email e news USENET sono tutti candidati potenziali per i servizi UBR, in quanto nessuna di queste applicazioni ha caratteristiche di tempo reale.

Le proprietà delle varie categorie di servizio sono riassunte in figura 5-70.

Caratteristiche del servizio	CBR	RT-VBR	NRT-VBR	ABR	UBR
Banda garantita	Si	Si	Si	Opzionale	No
Adatto per traffico real time	Si	Si	No	No	No
Adatto per traffico "con picchi"	No	No	Si	Si	Si
Feedback sulle congestioni	No	No	No	Si	No

Fig. 5-70 Caratteristiche delle categorie di servizio di ATM.

5.6.5 Qualità del servizio

La qualità del servizio è una caratteristica importante delle reti ATM, anche a causa della loro utilizzazione per traffico in tempo reale (ad es., connessioni audio/video). Quando viene creato un circuito virtuale, sia il livello trasporto (tipicamente un processo nella macchina host, il "cliente") che il livello rete ATM (ad es. un operatore di rete, il "fornitore") devono concordare su un contratto che definisce il servizio. Nel caso di reti pubbliche, questo contratto può avere implicazioni legali. Ad esempio, se il fornitore avesse concordato di non perdere più di una cella per miliardo e invece ne perdesse due, il cliente potrebbe rompere il contratto.

Il contratto tra il cliente e la rete è diviso in tre parti: il traffico presentato; il servizio su cui si è raggiunto un accordo; i requisiti di conformità.

È importante notare che il contratto potrebbe avere caratteristiche differenti nelle due direzioni. Per un'applicazione di video a richiesta, la banda necessaria dal controllo remoto dell'utente al server video potrebbe essere 1200 bps. Nell'altra direzione potrebbe essere 5 Mbps. È importante notare che se il cliente e il fornitore non sono in grado di accordarsi sui termini, oppure il fornitore non è in grado di soddisfare le richieste, il circuito virtuale non verrà creato. La prima parte del contratto è il **descrittore di traffico**, che caratterizza il carico che verrà presentato alla rete. La seconda parte specifica la qualità del servizio desiderata dal cliente e accettata dal fornitore. Sia il carico che il servizio devono essere formulati in termini di quantità misurabili, in modo che la conformità possa essere determinata oggettivamente. Dire solo "carico moderato" o "buon servizio" non è sufficiente.

Per fare in modo che sia possibile stringere contratti di traffico concreti, lo standard ATM definisce un certo numero di parametri **QoS** (Quality of Service – qualità del servizio) i cui valori possono essere negoziati dal cliente e dal fornitore. Per ogni parametro di qualità del servizio, il fornitore deve soddisfare o superare le richieste. In qualche caso, il parametro è un minimo; in altri è un massimo. Ancora una volta, la qualità del servizio è specificata separatamente per ogni direzione. Alcuni dei parametri più importanti sono elencati in figura 5-71, ma non tutti sono applicabili a tutte le categorie di servizio.

I primi tre parametri specificano la velocità a cui il mittente intende spedire. **PCR** (Peak Cell Rate – velocità di picco di trasmissione celle) è la massima velocità alla quale il

Parametro	Acronimo	Significato
Peak cell rate	PCR	Massima velocità a cui verranno spedite le celle
Sustained cell rate	SCR	Velocità media di spedizione sul lungo periodo
Minimum cell rate	MCR	Velocità minima accettabile
Cell delay variation tolerance	CDVT	Jitter accettabile massimo
Cell loss ratio	CLR	Percentuale di celle perse o consegnate troppo tardi
Cell transfer delay	CTD	Tempo necessario per la consegna (medio e massimo)
Cell delay variation	CDV	Varianza nei tempi di consegna delle celle
Cell error rate	CER	Percentuale di celle consegnate senza errori
Severely-errorred cell block ratio	SECBR	Percentuale di blocchi danneggiati
Cell misinsertion rate	CMR	Percentuale di celle consegnate alla destinazione sbagliata

Fig. 5-71 Alcune delle qualità dei parametri di servizio.

mittente ha intenzione di spedire celle. Questo parametro può essere inferiore a quello consentito dalle linee. Se il mittente ha intenzione di mandare una cella ogni $4 \mu s$, il suo *PCR* è 250.000 celle/s, anche se il tempo effettivo di trasmissione potrebbe essere di $2,7 \mu s$.

SCR (Sustained Cell Rate – velocità sostenuta di trasmissione celle) è il tasso di trasmissione atteso o richiesto in media durante un lungo intervallo di tempo. Per il traffico CBR, *SCR* sarà uguale a *PCR*, ma per tutte le altre categorie di servizio sarà sensibilmente inferiore. Il rapporto *PCR/SCR* è una misura della quantità di picchi presenti nel traffico. **MCR (Minimum Cell Rate – velocità minima di trasmissione celle)** è il minimo numero di celle/s che il cliente considera accettabile. Se il fornitore non è in grado di garantire questa velocità, deve rifiutare la connessione. Quando viene richiesto un servizio ABR, la banda effettivamente utilizzata dovrebbe situarsi fra *MCR* e *PCR*, ma può variare dinamicamente durante la vita della connessione. Se il cliente e il fornitore concordano di uguagliare *MCR* a 0, allora il servizio ABR diviene simile a un servizio UBR.

CVDT (Cell Variation Delay Tolerance – tolleranza alle variazioni nei ritardi dei messaggi) indica l'ammontare delle variazioni nei tempi di trasmissione delle celle. Viene specificato indipendentemente da *PCR* ed *SCR*. Per una sorgente perfetta che funzioni alla velocità *PCR*, ogni cella arriverà *esattamente* $1/\text{PCR}$ s dopo quella precedente. Nessuna cella sarà mai in anticipo e nessuna cella sarà mai in ritardo, nemmeno di un picosecondo. Per una sorgente reale che funzioni alla velocità *PCR*, ci saranno alcune variazioni nei tempi di trasmissione delle celle. La questione è: quant'è la massima variazione accettabile? Può una cella arrivare 1 ns in anticipo? E 30 s? *CVDT* controlla la variabilità accettabile utilizzando un algoritmo del secchio bucato che verrà descritto brevemente.

I tre parametri successivi descrivono le caratteristiche della rete e sono misurati presso il ricevente. Tutti e tre sono negoziabili. Il significato di **CLR (Cell Loss Ratio – Percentuale di celle perse)** è intuitivo. Misura la frazione di celle trasmesse che non vengono consegnate o vengono consegnate così in ritardo da essere inutilizzabili (ad es., per il traffico in tempo reale). **CTD (Cell Transit Delay – ritardo di transito delle celle)** è il tempo medio di transito dalla sorgente alla destinazione. **CDV (Cell Delay Variation – variazione di ritardo nelle celle)** misura quanto uniformemente vengano consegnate le celle.

Il modello per *CDT* e *CDV* è illustrato in figura 5-72, dove possiamo vedere la probabilità che una cella impieghi un tempo t per arrivare, come funzione di t . Per una data sorgente, una data destinazione e un dato percorso attraverso commutatori intermedi, esiste sempre qualche ritardo minimo dovuto al tempo di propagazione e di commutazione. Tuttavia, non tutte le celle hanno lo stesso ritardo minimo; la funzione di probabilità ha in generale una coda lunga. Scegliendo un valore di *CDT*, il cliente e il fornitore si accordano, in effetti, sul limite di ritardo di una cella e sulla misura in cui possa essere ancora conteggiata come una cella corretta. Generalmente, *CDV* verrà scelto in modo che α (la frazione di celle che vengono scartate perché arrivate troppo tardi), sia dell'ordine di 10^{-10} o meno. *CDV* misura la distribuzione dei tempi di arrivo. Per il traffico in tempo reale, questo parametro è spesso più importante di *CDT*.

Gli ultimi tre parametri QoS specificano le caratteristiche della rete. Non sono generalmente negoziabili. **CER (Cell Error Ratio – Percentuale di celle errate)** è la frazione di

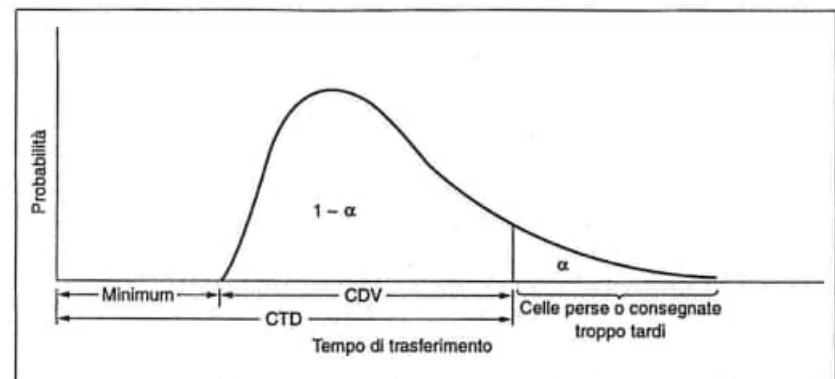


Fig. 5-72 La funzione di densità di probabilità per i tempi di arrivo delle celle.

celle che vengono consegnate con uno o più bit errati. **SECBR (Severely-Errored Cell Block Ratio – percentuale di blocchi di celle gravemente errati)** è la frazione di blocchi di N celle delle quali M o più celle contengono un errore. Infine, **CMR (Cell Misinsertion Rate – tasso di celle con destinazione errata)** è il numero di celle al secondo che vengono consegnate alla destinazione sbagliata in seguito a un errore non rilevato nel pREAMBOLo. La terza parte del contratto di traffico descrive cosa significa rispettare le regole. Se il cliente spedisce una cella troppo presto, ciò annulla il contratto? Se il fornitore non riesce a rispettare uno degli obiettivi di qualità per 1 ms, il cliente può intentare una causa? In effetti, questa parte del contratto viene negoziata tra il cliente e il fornitore e dice quanto strettamente debbano essere rispettate le prime due parti.

I modelli di qualità del servizio di ATM e Internet sono piuttosto differenti, e ciò ha un impatto sulle rispettive implementazioni. Il modello ATM è strettamente basato sulle connessioni, mentre il modello Internet usa i datagram combinati con i flussi (ad es., RSVP). Un confronto fra questi due modelli è dato in Crowcroft *et al.* (1995).

5.6.6 Normalizzazione e pattugliamento del traffico

Il meccanismo utilizzato per implementare e rispettare i parametri di qualità del servizio è basato (in parte) su un algoritmo specifico, **GCRA (Generic Cell Rate Algorithm – algoritmo a tasso generico di celle)**, che è illustrato in figura 5-73. Funziona verificando che ogni cella sia conforme ai parametri del suo circuito virtuale.

GCRA possiede due parametri, che specificano la velocità massima di trasmissione (*PCR*) e la varianza di ritardo considerata tollerabile (*CVDT*). Il reciproco di *PCR*, $T = 1/\text{PCR}$, corrisponde all'intervallo minimo tra i tempi di arrivo di due celle consecutive, come mostrato in figura 5-73 (a). Se il cliente concorda di non spedire più di 100.000 celle/s, allora $T = 10 \mu s$. Nel caso migliore, ogni cella arriva esattamente ogni 10 μs . Per evitare numeri troppo piccoli, lavoreremo in microsecondi, ma poiché tutti i parametri sono numeri reali, l'unità di tempo non ha importanza.

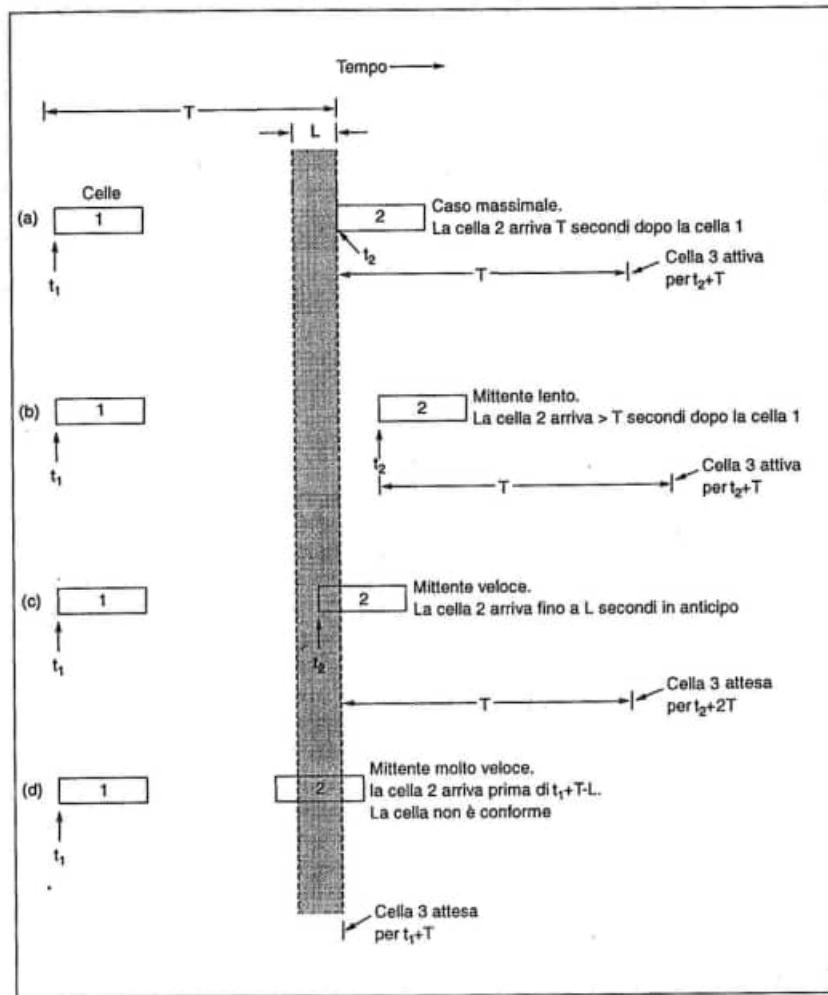


Fig. 5-73 Generic Cell Rate Algorithm.

Un mittente può sempre distanziare due celle consecutive più di T μs, come mostrato in figura 5-73 (b). Qualsiasi cella che arrivi più di T μs dopo quella precedente va bene. Il problema si presenta con i mittenti che tendono a essere troppo precipitosi, come in figura 5-73 (c) e (d). Se una cella arriva troppo presto (all'istante $t_1 + T-L$) va bene, ma la cella successiva è attesa per l'istante $t_1 + 2T$ (non per $t_2 + T$). In questo modo si evita che il mittente trasmetta ogni cella L μs prima, e quindi aumenti la velocità di trasmissione di picco.

Se una cella arriva più di L μs prima, è detta non conforme. Il trattamento delle celle non conformi è responsabilità del fornitore. Alcuni di essi possono scartare; altri possono conservarle, "settando" però il bit *CLP* in modo da ridurre la priorità e consentire ai commutatori di scartare le celle non conformi in caso di congestione. L'uso del bit *CLP* può variare a seconda delle diverse categorie di servizio di figura 5-69.

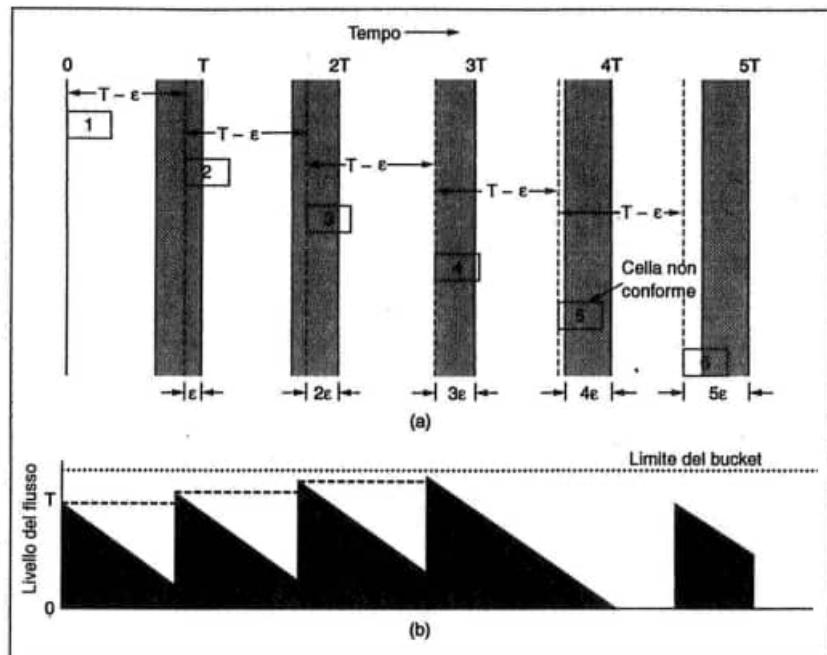


Fig. 5-74 (a) Un mittente che cerca di imbrogliare. (b) Lo stesso schema di arrivo delle celle, visto dal punto di vista di un secchio bucato.

A questo punto consideriamo cosa succede se un mittente cerca di imbrogliare, come mostrato in figura 5-74 (a). Invece di aspettare fino al tempo T prima di spedire la cella 2, il mittente la trasmette un attimo prima, al tempo $T - \epsilon$, dove $\epsilon = 0,3L$. Questa cella viene accettata senza problemi. A questo punto il mittente trasmette la cella 3, sempre dopo $T - \epsilon$ dalla cella precedente, cioè a $T - 2\epsilon$. Ancora una volta viene accettata. Tuttavia, ogni cella successiva si avvicina sempre di più verso il confine fatale $T - L$. In questo caso la cella 5 arriva al tempo $T - 4\epsilon = T - 1,2L$ (cioè troppo presto) e quindi la cella 5 viene dichiarata non conforme e scartata dall'interfaccia di rete.

Quando viene visto in questi termini, l'algoritmo GCRA viene detto **algoritmo di schedulazione virtuale**. Tuttavia, visto in modo differente, è equivalente a un algoritmo del secchio bucato (come illustrato in figura 5-74 (b)). Si immagini che ogni cella conforme

in arrivo versi T unità di liquido nel secchio. Poiché il secchio perde liquido alla velocità di $1 \text{ unità}/\mu\text{s}$, dopo $T \mu\text{s}$ tutto il liquido sarà uscito. Se le celle arrivano esattamente ogni $T \mu\text{s}$, ognuna di esse troverà il secchio (appena) svuotato e lo riempirà nuovamente con T unità di liquido. Quindi il livello del liquido viene riportato a T quando arriva una cella, per essere poi ridotto linearmente fino a quando raggiunge lo 0. La situazione è illustrata in figura 5-74 (b).

Poiché il liquido fuoriesce linearmente nel tempo, t istanti dopo l'arrivo di una cella la quantità di liquido rimasto è $T - t$. All'arrivo della cella 2 (nell'istante $T - \varepsilon$) ci sono ancora ε unità di liquido nel secchio. L'arrivo della nuova cella innalza questo valore a $T + \varepsilon$. In modo simile, all'arrivo della cella 3 rimarranno nel secchio 2ε unità di liquido e quindi la nuova cella innalzerà il livello del liquido a $T + 2\varepsilon$. All'arrivo della cella 4, verrà innalzato fino a $T + 3\varepsilon$.

Continuando in questo modo, prima o poi una cella innalzerà il livello oltre la capacità del secchio e quindi verrà scartata. Per capire quale delle celle sarà scartata, calcoliamo il valore della capacità. Vogliamo che l'algoritmo del secchio bucati dia gli stessi risultati di figura 5-74 (a), cioè vogliamo che il secchio trabocchi quando una cella arriva $L \mu\text{s}$ troppo presto. Poiché il fluido rimasto richiede $L \mu\text{s}$ per fuoriuscire, la quantità di liquido deve essere L , in quanto la velocità di uscita è $1 \text{ unità}/\mu\text{s}$. Vogliamo quindi che la capacità del secchio sia $L + T$, in modo che qualsiasi cella che arrivi più di $L \mu\text{s}$ in anticipo venga rifiutata a causa dello straripamento del secchio. In figura 5-74 (b), all'arrivo della 5 l'aggiunta di T unità alle 4 ε unità già presenti innalza il livello del secchio a $T + 4\varepsilon$. Poiché in questo esempio stiamo utilizzando $\varepsilon = 0,3L$, il secchio raggiunge il livello $T + 1,2L$ e quindi la cella viene rifiutata, non si inserisce nuovo liquido e il secchio prima o poi si svuota.

Dato un certo T , se il valore utilizzato per L fosse molto piccolo, la capacità del secchio supererebbe difficilmente il valore T ; quindi tutte le celle sarebbero distanziate in modo molto uniforme. Viceversa, se L fosse molto superiore a T , il secchio potrebbe contenere molte celle in quanto $T + L > T$. Questo significa che il mittente potrebbe trasmettere una sequenza di celle consecutive alla velocità massima e vederle comunque accettate.

Possiamo calcolare facilmente il numero di celle conformi, N , che possono essere trasmesse consecutivamente alla velocità di picco ($PCR = 1/T$). Durante un getto di N celle, la quantità totale di liquido aggiunto al secchio è NT in quanto ogni cella aggiunge T . Tuttavia, durante il getto massimo, il liquido fuoriesce dal secchio alla velocità di un'unità per intervallo di tempo. Chiamiamo il tempo di trasmissione delle celle δ unità di tempo. Si noti che $\delta \leq T$ in quanto è assolutamente possibile che un mittente su una linea a 155.52 Mbps concordi di spedire non più di 100.000 celle/sec, nel qual caso $\delta = 2,73 \mu\text{s}$ e $T = 10 \mu\text{s}$. Durante un getto di 100.000 celle, la quantità di liquido uscito è uguale a $(N-1)\delta$, in quanto il liquido non inizia a uscire fino a quando la prima cella non è stata completamente trasmessa.

Da queste osservazioni, otteniamo che l'incremento netto di liquido nel secchio durante il getto massimo è $NT - (N-1)\delta$. La capacità del secchio è $T + L$. Egualando queste due quantità otteniamo

$$NT - (N-1)\delta = T + L.$$

Risolvendo questa equazione per N otteniamo

$$N = 1 + \frac{L}{(T - \delta)}$$

Tuttavia, se questo numero non è intero, deve essere arrotondato all'intero inferiore per evitare lo straripamento del secchio. Ad esempio, con $PCR = 100.000$ celle/s, $\delta = 2,73 \mu\text{s}$ ed $L = 50 \mu\text{s}$, possono essere spedite 7 celle consecutive alla velocità di 155,52 Mbps senza riempire il secchio. Un'ottava cella sarebbe non conforme.

L'algoritmo GCRA viene normalmente specificato fornendo i parametri T ed L . T è il reciproco di PCR ; L è uguale a $CDVT$. GCRA viene anche utilizzato per assicurarsi che la velocità media di spedizione non superi SCR per lunghi periodi di tempo.

In questo esempio abbiamo assunto che le celle giungano in modo uniforme. In realtà non è così. Ciò nonostante, l'algoritmo del secchio bucati può essere utilizzato anche in questo caso. A ogni nuova cella si controlla se nel secchio sia presente spazio sufficiente per T unità addizionali di liquido. In caso affermativo, la cella è conforme; altrimenti è non conforme. Oltre a fornire una regola per stabilire quali celle siano conformi e quali no, GCRA normalizza il traffico rimuovendo alcuni dei picchi. Più piccolo $CDVT$, più grande l'effetto di normalizzazione; è però anche maggiore la possibilità che le celle vengano scartate come non conformi. Alcune implementazioni combinano il leaky bucket di GCRA con un token bucket, per fornire normalizzazione aggiuntiva.

5.6.7 Controllo della congestione

Anche nel caso di normalizzazione del traffico, le reti ATM non soddisfano automaticamente le prestazioni richieste nel contratto di traffico. Ad esempio, la congestione nei commutatori intermedi è sempre un problema potenziale, specialmente quando più di 350.000 celle/sec stanno affluendo da una singola linea, e un commutatore può avere più di 100 linee. Conseguentemente, sono stati dedicati molti sforzi per risolvere il problema delle prestazioni e della congestione nelle reti ATM. In questo paragrafo discuteremo alcuni degli approcci utilizzati. Per informazioni aggiuntive, si vedano Eckberg (1992; Eckberg et al. (1991); Hong, Suda (1991); Jain (1995); Newman (1994)).

Le reti ATM devono gestire sia congestioni a lungo termine, causate da un traffico in entrata superiore a quello che la rete può gestire, sia congestioni a breve termine, causate da picchi nel traffico. Come risultato, molte strategie differenti vengono utilizzate contemporaneamente. Le più importanti di queste cadono in tre categorie:

1. Controllo di ammissione.
2. Prenotazione di risorse.
3. Controllo di congestione basato sulla velocità.

Controllo di ammissione

In reti a bassa velocità, è in generale sufficiente aspettare che le congestioni si presentino e quindi reagire chiedendo di rallentare alla sorgente dei pacchetti. In reti ad alta velocità,

questo approccio spesso non funziona, in quanto nell'intervallo tra la spedizione di una notifica e il suo arrivo alla sorgente possono arrivare migliaia di pacchetti addizionali. Inoltre, molte reti ATM possiedono sorgenti di traffico in tempo reale che producono dati a un ritmo prestabilito. Richiedere che la sorgente rallenti può non funzionare (si immagini un nuovo telefono digitale dotato di una spia rossa; in caso di congestione, la spia si accende e si richiede che l'utente parli più lentamente del 25%).

Conseguentemente, le reti ATM preferiscono prevenire le congestioni piuttosto che curarle. Tuttavia, per il traffico di tipo CBR, VBR e UBR, non è presente nessun controllo dinamico delle congestioni, e quindi in questo caso un'onda di prevenzione è meglio di una libbra (in effetti, meglio dire una tonnellata) di cura. Uno degli strumenti principali per prevenire la congestione è il controllo di ammissione. Quando un host desidera un nuovo circuito, deve descrivere il traffico che verrà offerto e il servizio atteso. La rete quindi verifica se è possibile gestire questa connessione senza creare problemi alle connessioni esistenti. È possibile che debba esaminare molti percorsi potenziali prima di trovare quello in grado di compiere il lavoro. Se non viene trovato nessun percorso, la chiamata viene rifiutata.

I rifiuti di ammissione devono essere fatti equamente. È equo che una persona sprofondata nel divano che fa zapping tra dozzine di programmi televisivi possa escludere 100 lavoratori indefesi che cercano di leggere la loro posta? Se non venisse applicato nessun controllo, un piccolo numero di utenti "a larga banda" potrebbe disturbare pesantemente un gran numero di utenti "a bassa banda". Per evitare che ciò accada, gli utenti dovrebbero essere divisi in classi basate sul tipo di utilizzazione. La probabilità di sentirsi rifiutato il servizio dovrebbe essere all'incirca la stessa per tutte le classi (possibilmente dando a ogni classe il proprio sottoinsieme di risorse).

Prenotazione delle risorse

La tecnica per prenotare le risorse in anticipo (generalmente alla creazione della chiamata) è strettamente collegata al controllo di ammissione. Poiché il descrittore di traffico contiene la velocità di picco di trasmissione, la rete ha la possibilità di prenotare banda sufficiente lungo il cammino per gestire tale velocità. La banda può essere riservata facendo in modo che il messaggio setup prenoti una certa quantità di capacità trasmittiva lungo ogni linea che attraversa, assicurandosi, ovviamente, che la banda totale messa da parte lungo una linea sia minore della capacità della linea stessa. Se il messaggio SETUP arriva su una linea piena, deve tornare sui suoi passi e cercare un percorso alternativo. Il descrittore di traffico può contenere non solo la banda di picco, ma anche la banda media. Se un host desiderasse, ad esempio, una banda di picco di 100.000 celle/s e una banda media di sole 20.000 celle/s, in principio cinque di questi circuiti potrebbero essere sovrapposti sullo stesso canale fisico. Purtroppo, potrebbe verificarsi il caso in cui tutte e cinque le connessioni restino inattive per un'ora, per poi iniziare a spedire pacchetti alla massima velocità, causando grandi perdite di celle. Poiché il traffico VBR può essere statisticamente sovrapposto, i problemi possono avvenire con questa categoria di servizio. Sono allo studio possibili soluzioni.

Controllo della congestione basato sulla velocità

Nel caso di traffico CBR o VBR, non è in generale possibile che il mittente rallenti, anche in presenza di congestioni, a causa della natura inherentemente in tempo reale (anche parzialmente) della sorgente delle informazioni. Nel caso di traffico UBR, nessuno si preoccupa; se ci sono troppe celle, quelle extra vengono scartate.

Tuttavia, con il traffico ABR, è possibile e ragionevole che la rete mandi un segnale a uno o più mittenti chiedendo di rallentare temporaneamente fino a quando la rete non potrà recuperare. È interesse del mittente assecondare la rete, che può sempre punirlo scartando le celle in eccesso.

I problemi relativi al rilevamento, alla segnalazione e al controllo delle congestioni per il traffico ABR generarono notevoli discussioni durante lo sviluppo dello standard ATM, con argomentazioni vigorose a favore di ognuna delle soluzioni proposte. Prima di esaminare la soluzione vincente, diamo una breve occhiata ad alcune di quelle che furono velocemente rifiutate.

In una delle proposte, ogni volta che il mittente avesse voluto spedire un getto di dati, avrebbe dovuto prima spedire una cella speciale per riservare la banda necessaria. Alla ricezione della conferma, il getto avrebbe potuto essere spedito. Questa soluzione sarebbe stata vantaggiosa per il fatto che avrebbe eliminato completamente la congestione, in quanto la banda richiesta è sempre presente quando necessario. Il forum ATM la rifiutò a causa dei lunghi ritardi con cui l'host avrebbe potuto iniziare a spedire.

Una seconda proposta prevedeva l'utilizzazione di celle regolatrici da parte dei commutatori ogni volta che la congestione avesse iniziato a presentarsi. Alla ricezione di tale cella, si presumeva che il mittente riducesse della metà la sua velocità di trasmissione attuale. Furono proposti molti schemi per tornare alla velocità normale, alla scomparsa della congestione. Questo schema fu rifiutato poiché le celle regolatrici potevano venir perse nella congestione, e poiché lo schema sembrava poco equo per i piccoli utenti. Come esempio, si considerò un commutatore che ricevesse un flusso da 100 Mbps da cinque utenti, e un flusso di 100 kbps da un altro utente. Molti membri del comitato trovarono inappropriato richiedere all'utente da 100 kbps di ridurre la velocità a 50 kbps perché stava causando troppa congestione.

Una terza proposta utilizzava il fatto che i confini dei pacchetti sono marcati da un bit nell'ultima cella. L'idea era di scartare celle per alleviare la congestione, ma in modo altamente selettivo. Il commutatore doveva osservare il flusso di celle in arrivo e quindi gettare via tutte le celle del prossimo pacchetto. Naturalmente, questo pacchetto sarebbe stato ritrasmesso successivamente, ma scartare tutte le k celle di un pacchetto avrebbe causato la ritrasmessione al massimo di un pacchetto, preferibile allo scarto di k celle casuali, che avrebbe potuto causare la ritrasmessione di k pacchetti. Questo schema fu rifiutato per motivi di equità, in quanto il prossimo segnale di fine pacchetto può non appartenere al mittente che ha sovraccaricato il commutatore. Inoltre, questo schema non avrebbe avuto bisogno di essere standardizzato. Qualunque venditore di commutatori sarebbe stato libero di scegliere quali celle scartare nel caso di congestione.

Dopo molte discussioni, la battaglia si focalizzò su due contendenti, un soluzione basata sui crediti (Kung, Morris (1995) e una soluzione basata sulla velocità (Bonomi, Fendick (1995). La soluzione basata sui crediti era essenzialmente un protocollo sliding window

dinamico. Richiedeva che ogni commutatore mantenesse un credito per ogni circuito – essenzialmente il numero di buffer riservati per quel circuito: fino a quando ogni cella trasmessa possiede un buffer riservato, la congestione non si presenta mai.

Gli argomenti contro questa soluzione vennero dai produttori di commutatori, che non volevano gestire tutta la contabilità riguardante i crediti e non volevano riservare così tanti buffer in anticipo. Si riteneva che il carico aggiuntivo e lo spreco richiesti fossero eccessivi, e così alla fine fu adottato lo schema di controllo della congestione basato sulla velocità.

Il modello base richiede che dopo k celle dati, ogni mittente spedisca una cella speciale **RM (Resource Management – Gestione risorse)**. Questa cella viaggia lungo lo stesso percorso delle celle dati, ma viene trattata in modo speciale dai commutatori lungo il percorso. Quando raggiunge la destinazione, viene esaminata, aggiornata e rispedita al mittente. Il percorso completo delle celle RM è illustrato in figura 5-75.



Fig. 5-75 Il percorso preso dalle celle RM.

Inoltre, vengono forniti due ulteriori meccanismi di controllo della congestione. Primo, i commutatori sovraccarichi possono generare spontaneamente delle celle RM e spedirle al mittente. Secondo, i commutatori sovraccarichi possono “settare” il bit intermedio *PTI* delle celle dati che viaggiano dal mittente al ricevente. Nessuno di questi metodi è completamente affidabile, in quanto queste celle possono essere perse nella congestione senza che nessuno lo noti. Al contrario, una cella RM persa verrà notata dal mittente nel caso in cui non riceva una risposta entro l’intervallo atteso. Inoltre, il bit *CLP* non viene usato per il controllo della congestione ABR.

Il controllo di congestione ABR è basato sull’idea che ogni mittente abbia una velocità attuale di spedizione *ACR* (**A**ctual **C**ell **R**ate) che cade tra *MCR* e *PCR*. Quando avviene una congestione, *ACR* viene ridotto (ma non sotto *MCR*). In assenza di congestione, *ACR* viene aumentato (ma non oltre *PCR*). Ogni cella RM spedita contiene la velocità alla quale il mittente vorrebbe spedire attualmente, al limite *PCR*, probabilmente inferiore.

Questo valore è chiamato **ER** (**E**xplicit **R**ate – velocità esplicita). Poiché le celle RM passano attraverso diversi commutatori lungo il percorso verso il ricevente, quelli congestionati possono ridurre ER. Nessun commutatore lo può aumentare. La riduzione può avvenire in una delle due direzioni. Quando il mittente riottiene la cella RM, può calcolare la velocità minima accettabile da ognuno dei commutatori lungo il cammino. Può quindi migliorare *ACR*, se necessario, per portarlo in linea con la velocità del commutatore più lento.

Il meccanismo di congestione che utilizza il bit *PTI* centrale può essere integrato nelle

celle RM facendo in modo che il ricevente includa questo bit (preso dall’ultima cella di dati) in ognuna delle celle RM spedite indietro. Il bit non può essere copiato dalla cella RM stessa, in quanto esso è sempre uguale a 1 in qualsiasi cella RM, come mostrato in figura 5-63.

Il livello ATM è abbastanza complicato. In questo capitolo, abbiamo sottolineato solo alcune delle sue caratteristiche. Per informazioni aggiuntive, si vedano De Prycker (1993); McDysan, Spohn (1995); Minoli, Vitella (1994); La Porta (1994). Tuttavia, si noti che tutta questa bibliografia è relativa allo standard ATM 3 e non allo standard ATM 4, che nel 1996 non era ancora stato definito completamente.

5.6.8 LAN basate su ATM

Poiché è sempre più ovvio che l’obiettivo originale di ITU di rimpiazzare la rete telefonica commutata pubblica tramite una rete ATM richiederà molto tempo, l’attenzione si sta spostando sull’utilizzazione della tecnologia ATM per collegare LAN esistenti. In questo approccio, una rete ATM può funzionare o come una LAN, connettendo host individuali, oppure come bridge, connettendo LAN multiple. Sebbene entrambi i concetti siano molto interessanti, hanno sollevato alcune questioni impegnative che discuteremo in seguito. Informazioni aggiuntive sulle LAN basate su ATM possono essere trovate in Chao *et al.* (1994); Newman (1994); Truong *et al.* (1995).

Il problema principale che deve essere risolto riguarda l’implementazione di un servizio LAN senza connessione sopra una rete ATM orientata alla connessione. Una soluzione possibile è introdurre un server senza connessione nella rete. Ogni host crea inizialmente una connessione con questo server, e gli spedisce i pacchetti da inoltrare. Sebbene sia semplice, questa soluzione non utilizza l’intera banda di una rete ATM, e il server può diventare facilmente un collo di bottiglia.

Un approccio alternativo, proposto dal Forum ATM, è mostrato in figura 5-76. Qui ogni host ha un circuito ATM virtuale (potenziale) con ogni altro host. Questi circuiti possono essere creati e chiusi dinamicamente a seconda delle necessità, o possono essere circuiti virtuali permanenti. Per spedire un frame, l’host sorgente prima incapsula il pacchetto nel campo contenuto di un messaggio AAL di ATM e lo spedisce alla destinazione, allo stesso modo in cui i frame vengono spediti sopra Ethernet, token ring e altre LAN.

Il problema principale introdotto da questo schema riguarda l’associazione degli indirizzi IP (o qualunque altro indirizzo del livello rete) ai circuiti virtuali. In una LAN 802, questo problema è risolto dal protocollo ARP, nel quale un host può spedire una richiesta del tipo “Chi ha indirizzo IP 192.31.20.47?”. L’host che utilizza questo indirizzo spedisce una risposta punto-a-punto, che viene salvata per uso futuro.

Con una LAN ATM, questa soluzione non funziona in quanto le LAN ATM non supportano il broadcasting. Questo problema è risolto introducendo un nuovo server, chiamato **LES (LAN Emulation Server – server di emulazione LAN)**. Per cercare un indirizzo del livello rete (ad es., un indirizzo IP), gli host spediscono un pacchetto (ad es., una richiesta ARP) al LES, il quale cerca l’indirizzo ATM corrispondente e lo restituisce alla macchina che lo richiede. Questo indirizzo può essere usato quindi per spedire pacchetti alla destinazione.

Tuttavia, questa soluzione risolve solo il problema di trovare gli host. Alcuni programmi

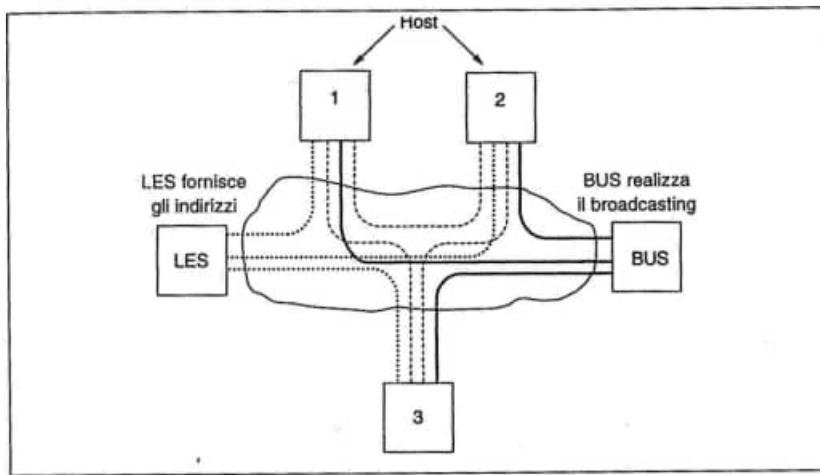


Fig. 5-76 Emulazione LAN basata su ATM.

utilizzano broadcast e multicast come parti essenziali dell'applicazione. Per queste applicazioni, è stato introdotto il **BUS** (Broadcast/Unknown Server – server di broadcast/destinatario sconosciuto), che possiede connessioni con tutti gli host e può simulare il broadcast spedendo un pacchetto su ognuna di esse, una alla volta. Gli host possono velocizzare la consegna di un pacchetto a un host sconosciuto spedendo il pacchetto al BUS per il broadcast e quindi (in parallelo) cercando l'indirizzo (per usi futuri) usando il LES.

Un modello simile a questo è stato adottato dall'IETF come modo ufficiale di Internet per usare una rete ATM per trasportare pacchetti IP. In questo modello il server LES viene chiamato server **ATMARP**, ma la funzionalità è essenzialmente la stessa. Il broadcast e il multicast non sono supportati dalla proposta IETF. Il modello è descritto nell'RFC 1483 e nell'RFC 1577. Un'altra buona sorgente di informazione è Comer (1995).

Nel metodo IETF, un insieme di computer ATM possono essere raggruppati per formare una **LIS** (Logical IP Subnet – sottorete IP logica). Ogni LIS ha il proprio server ATMARP. In effetti, una LIS agisce come una LAN virtuale. Gli host nella stessa LIS possono scambiare pacchetti IP direttamente, mentre gli host su LIS differenti devono passare attraverso un router. L'utilizzazione di più LIS è motivata dal fatto che ogni host in un LIS deve avere (potenzialmente) un circuito virtuale aperto per ogni altro host nello stesso LIS. Restringendo il numero degli host per LIS, il numero dei circuiti virtuali aperti può essere ridotto a un numero ragionevole.

Un'altra utilizzazione delle reti ATM è quella di bridge per connettere LAN esistenti. In questa configurazione, solo una macchina in ogni LAN ha bisogno di una connessione ATM. Come tutti i bridge trasparenti, il bridge LAN deve stare in ascolto da tutte le LAN a cui è connesso, inoltrando i frame dove necessario. Poiché i bridge utilizzano solo gli

5.7 Riassunto

indirizzi MAC (non indirizzi IP), i bridge ATM devono costruire uno spanning tree, proprio come i bridge 802.

In breve, sebbene l'emulazione LAN sia un'idea interessante, esistono seri dubbi sulle sue prestazioni e sul suo prezzo, ed esiste sicuramente una pesante competizione con le LAN esistenti e i bridge, che sono ben stabilizzati e altamente ottimizzati. Se mai le LAN e i bridge ATM rimpiazzeranno mai le LAN e i bridge 802 resta da vedere.

5.7 Riassunto

Il livello rete fornisce servizi al livello trasporto. Può essere basato su circuiti virtuali, oppure su datagram. In entrambi i casi il suo compito principale è quello di instradare i pacchetti dalla sorgente alla destinazione. Nelle reti basate su circuito virtuale le decisioni di routing vengono prese alla creazione del circuito virtuale; nelle reti basate su datagram si prendono decisioni diverse all'arrivo di ogni pacchetto.

Nelle reti di computer vengono utilizzati diversi algoritmi di instradamento. Fra gli algoritmi statici ricordiamo l'instradamento lungo il cammino minimo, il flooding e l'instradamento basato su flusso; fra gli algoritmi ricordiamo il distance vector routing e il link state routing. Gran parte delle reti attuali utilizzano uno di questi algoritmi. Altre importanti problematiche riguardanti l'instradamento sono il routing gerarchico, il routing per host mobili, i routing broadcast e multicast.

Le reti possono presentare congestioni, aumentando i ritardi e abbassando la velocità di trasmissione dei pacchetti. I progettisti di rete cercano di evitare le congestioni con una progettazione appropriata. Vengono utilizzate tecniche quali normalizzazione del traffico, specifiche di flusso e prenotazione della banda. Nel caso avvenga una congestione, è necessario occuparsene. I pacchetti regolatori possono essere spediti indietro, il carico può essere distribuito e possono essere applicati altri metodi.

Poiché le reti differiscono in molti particolari, quando reti multiple vengono connesse possono sorgere dei problemi. In qualche caso questi problemi possono essere aggirati incapsulando i pacchetti attraverso reti aliene, ma se le reti sorgente e destinazione sono differenti questo approccio non funziona. Quando reti differenti utilizzano diverse dimensioni massime dei pacchetti, può rendersi necessaria la frammentazione.

Internet possiede una grande varietà di protocolli collegati al livello rete. Oltre al protocollo di trasmissione dati IP, esistono i protocolli di controllo ICMP, ARP e RARP e i protocolli di routing OSPF e BGP. Poiché Internet sta rapidamente esaurendo gli indirizzi IP, è stata sviluppata una nuova versione di IP (IPv6).

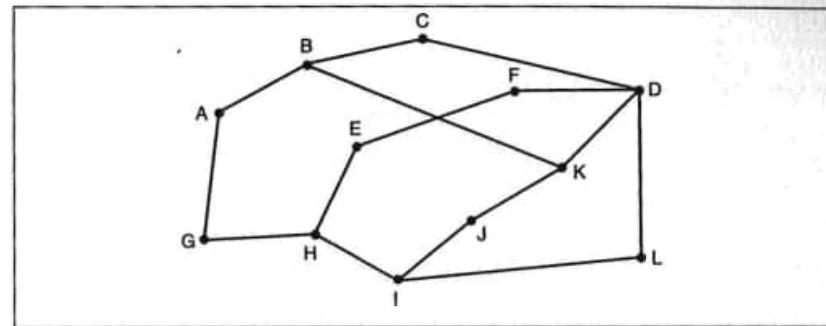
A differenza di Internet, basata su datagram, le reti ATM utilizzano internamente i circuiti virtuali, che devono essere creati prima di poter trasferire dati e devono essere chiusi dopo aver completato la trasmissione. La qualità del servizio e il controllo della congestione sono fra le problematiche principali riguardanti le reti ATM.

Esercizi

1. Descrivere due applicazioni a piacere per le quali siano adatti i servizi orientati alla connessione.
2. Esiste la possibilità che un circuito virtuale consegna i pacchetti fuori ordine? Spiegare.
3. Le reti basate su datagram instradano ogni pacchetto come un'unità separata, indipendente da tutte le altre. Le reti basate su circuito virtuale non devono comportarsi in questo modo, in quanto ogni pacchetto segue un percorso prefissato. Questa osservazione implica che le reti basate su circuito virtuale non hanno bisogno della capacità di instradare pacchetti isolati da una sorgente arbitraria a una destinazione arbitraria? Spiegare la risposta.
4. Dare tre esempi di parametri di protocollo che possono essere negoziati durante la creazione di una connessione.
5. Si consideri il seguente problema di progettazione concernente l'implementazione di servizi basati su circuito virtuale. Se all'interno di una rete vengono utilizzati i circuiti virtuali, ogni pacchetto deve avere un preambolo di 3 byte, e ogni router deve utilizzare 8 byte di memoria per l'identificazione dei circuiti. Se vengono utilizzati i datagram, sono necessari 15 byte di preambolo, ma non è richiesto alcuno spazio nelle tabelle dei router. La capacità di trasmissione costa 1 centesimo per 10^6 byte, per ogni salto. La memoria dei router può essere acquistata per 1 centesimo al byte e si deprezza in due anni (solo ore di esercizio). La sessione media dura 1000 s, durante la quale vengono trasmessi 200 pacchetti. Il pacchetto medio richiede 4 salti. Quale implementazione è più economica, e di quanto?
6. Assumendo che tutti i router e tutti gli host stiano funzionando correttamente e che tutto il software sia privo di errori, esiste una qualsiasi possibilità, comunque piccola, che un pacchetto sia consegnato a una destinazione errata?
7. Fornire una semplice euristica per trovare due percorsi da un data sorgente a una data destinazione che possano sopravvivere alla perdita di qualsiasi linea di comunicazione (assumendo che tali percorsi esistano). I router sono considerati abbastanza affidabili, quindi non è necessario preoccuparsi della possibilità che si guastino.
8. Si consideri la rete della figura 5-15 (a). Viene utilizzato il distance vector routing, e i seguenti vettori sono appena arrivati al router C: da B: (5, 0, 8, 12, 6, 2); da D: (16, 12, 6, 0, 9, 10); e da E: (7, 6, 3, 9, 0, 4). I ritardi misurati per B, D ed E sono rispettivamente 6, 3 e 5. Qual è la nuova tabella di routing di C? (Contenente sia la linea di output da utilizzare che il ritardo atteso).
9. Se i ritardi vengono memorizzati come numeri a 8 bit in una rete di 50 router, e i vettori di ritardo vengono scambiati due volte al secondo, quanta capacità per linea full duplex viene occupata dall'algoritmo distribuito di routing? Si assuma che ogni router comunichi con altri tre router.

5.7 Riassunto

10. In figura 5-16 l'OR booleano dei due insiemi di bit ACF è uguale a 111 in ogni riga. È solo un caso, oppure ciò accade in tutte le reti e in qualsiasi circostanza?
11. Nel caso di un routing gerarchico di 4800 router, quali dimensioni di regione e cluster devono essere scelte per minimizzare la dimensione delle tabelle di routing in una gerarchia a tre livelli?
12. Nel testo si è affermato che quando un host mobile non è nella sua locazione base, i pacchetti spediti alla sua LAN base sono intercettati dal suo agente base. Nel caso di una rete IP su una LAN 802.3, in che modo l'agente base esegue questa intercettazione?
13. Osservando la rete di figura 5-5, quanti pacchetti sono generati da un broadcast a partire da B, utilizzando:
 - (a) Reverse path forwarding.
 - (b) Sink tree.
14. Nella rete seguente, calcolare uno spanning tree di multicast a partire dal router C per un gruppo composto dai router A, B, C, D, E, F, I, K.



15. Come possibile meccanismo di controllo della congestione in una rete internamente basata su circuiti virtuali, un router può astenersi dal confermare l'avvenuta ricezione di un pacchetto finché (1) non viene a sapere che la sua ultima trasmissione lungo il circuito virtuale è stata ricevuta correttamente e (2) non dispone di un buffer libero. Per semplicità, si assuma che i router utilizzino un protocollo stop-and-wait e che ogni circuito virtuale possieda un buffer dedicato per ogni direzione di traffico. Se la trasmissione di un pacchetto (di dati o di conferma) dura T secondi ed esistono n router lungo il percorso, quale è la frequenza con cui i pacchetti vengono consegnati all'host destinazione? Si assuma che gli errori di trasmissione siano rari, e che le connessioni host-router siano infinitamente veloci.
16. Una rete basata su datagram consente ai router di scartare pacchetti in caso di bisogno. La probabilità che un router scarti un pacchetto è p. Si consideri il caso di un

host sorgente connesso al router sorgente, a sua volta connesso al router destinazione, e quindi all'host destinazione. Se uno dei due router scarta il pacchetto, l'host sorgente va prima o poi in timeout e prova ancora. Se entrambe le linee host-router e router-router vengono contate come salti, qual è il numero medio di:

- (a) Salti attraversati da un pacchetto per ogni trasmissione?
 - (b) Trasmissioni ripetute per ogni pacchetto?
 - (c) Salti richiesti per la ricezione di un pacchetto?
17. Fornire una motivazione per cui l'algoritmo del secchio bucato dovrebbe consentire il passaggio di un solo pacchetto per battito di orologio, indipendentemente dalla dimensione del pacchetto.
18. In un certo sistema viene usata la variante che conta i byte dell'algoritmo del secchio bucato. La regola consente la spedizione di un pacchetto di 1024 byte, due pacchetti di 512 byte ecc., che possono essere spediti a ogni battito di orologio. Descrivete una seria limitazione di questo sistema che non sia stata citata nel testo.
19. Una rete ATM utilizza uno schema token bucket per la normalizzazione del traffico. Un nuovo gettone viene inserito nel secchio ogni $5\ \mu\text{s}$. Qual è la massima velocità di trasmissione sostenibile (senza contare i bit del preambolo)?
20. Un computer in una rete a 6 Mbps è regolato da un token bucket, riempito al ritmo di 1 Mbps. La sua capacità iniziale è di 8 Mb. Per quanto tempo il computer può trasmettere alla massima velocità?
21. La figura 5-27 mostra quattro caratteristiche dell'input per una specifica di flusso di esempio. Si supponga che la dimensione massima del pacchetto sia 100 byte, la velocità del token bucket sia 10.000.000 di byte/s, la dimensione del token bucket sia 1.000.000 di byte, e la velocità massima di trasmissione sia 50.000.000 di byte/s. Per quanto tempo il computer può trasmettere alla massima velocità?
22. Un dispositivo accetta frame dalla Ethernet a cui è connesso. Rimuove il pacchetto all'interno di ogni frame, aggiunge altre informazioni di framing e lo trasmette lungo una linea telefonica noleggiata (la sua unica connessione con il mondo esterno) a un dispositivo identico dall'altra parte. Questo dispositivo rimuove il framing, inserisce il pacchetto in un frame token ring e lo trasmette a un host locale su una LAN token ring. Come chiamereste il dispositivo?
23. La frammentazione è necessaria anche nelle internet a circuito virtuale concatenato, oppure solo nei sistemi a datagram?
24. Il tunneling attraverso una rete basata su circuiti virtuali concatenati è banale: il router multiprotocollo a un'estremità crea un circuito virtuale con l'altro estremo e trasmette i pacchetti attraverso di esso. Le tecniche di tunneling possono essere utilizzate nelle reti basate su datagram? Se sì, come?
25. Un datagram IP che utilizza l'opzione *Strict source routing* deve essere frammentato.

5.7 Riassunto

- Pensate che l'opzione venga copiata in tutti i frammenti, oppure è sufficiente inserirla nel primo frammento? Spiegare la risposta.
26. Si supponga che invece di utilizzare 16 bit per la sezione rete di un indirizzo di classe B, vengano utilizzati 20 bit. Quante reti di classe B ci sarebbero?
 27. Convertire l'indirizzo IP la cui rappresentazione esadecimale è C22F11582 nella notazione decimale a punti.
 28. Una rete di classe B ha come maschera di sottorete 255.255.240.0. Qual è il massimo numero di host per sottorete?
 29. Avete appena spiegato il protocollo ARP a un amico. Quando terminate, vi dice: "Ho capito. ARP fornisce un servizio al livello rete, quindi fa parte del livello data link". Cosa gli dite?
 30. Sia ARP che RARP mappano indirizzi da uno spazio a un altro. Sotto questo punto di vista sono simili. Ciò nonostante, le loro implementazioni sono sostanzialmente differenti. Qual è la differenza principale?
 31. Descrivere un modo per riassemblare i frammenti IP alla destinazione.
 32. Gran parte degli algoritmi di riassemblamento dei datagram IP hanno un timer per evitare che un frammento perso occupi i buffer di riassemblamento per sempre. Si supponga che un datagram sia suddiviso in quattro frammenti. I primi tre frammenti arrivano, mentre l'ultimo è in ritardo. Prima o poi il timer scadrà e i tre frammenti nella memoria del ricevente verranno scartati. Un attimo dopo, arriva l'ultimo frammento. Cosa si dovrebbe farne?
 33. Gran parte dei protocolli di routing IP utilizzano il numero di salti come valore da minimizzare al momento di ricalcolare le tabelle di routing. Per le reti ATM, il numero di HOP non è molto importante. Perché no? Suggerimento: leggere il capitolo 2 per vedere come funzionano i commutatori ATM. Sono basati su store-and-forward?
 34. Sia in IP che in ATM, la somma di controllo copre solo il preambolo e non i dati. Quale credete sia il motivo di questa scelta?
 35. Una persona che vive a Boston viaggia fino a Minneapolis, con il suo computer portatile. Con sua sorpresa, la LAN presso la sua destinazione a Minneapolis è una LAN IP senza filo; non deve quindi fare collegamenti. È comunque necessario utilizzare l'intero sistema di agenti base e agenti stranieri affinché posta elettronica e altro traffico arrivino correttamente?
 36. IPv6 utilizza indirizzi di 16 byte. Se si allocasse un blocco di 1.000.000 di indirizzi ogni picosecondo, per quanto tempo durerebbero gli indirizzi?
 37. Il campo *Protocol* utilizzato nel preambolo IPv4 non è presente nel preambolo fisso di IPv6. Perché?

38. L'introduzione del protocollo IPv6 implica cambiamenti nel protocollo ARP? Se sì, i cambiamenti sono concettuali o tecnici?
39. Nel capitolo 1, abbiamo classificato le interazioni tra reti e host utilizzando quattro classi di primitive: *request*, *indication*, *response* e *confirm*. Si classifichino i messaggi SETUP e CONNECT di figura 5-65 in queste categorie.
40. Si sta creando un nuovo circuito virtuale in una rete ATM. Fra gli host sorgente e destinazione vi sono tre commutatori ATM. Quanti messaggi (inclusi gli ack) verranno spediti per stabilire il circuito?
41. La logica utilizzata per costruire la tabella di figura 5-67 è semplice: il *VPI* inutilizzato più basso viene assegnato alla connessione. Se viene richiesto un nuovo circuito virtuale fra NY e Denver, quale *VPI* gli verrà assegnato?
42. In figura 5-73 (c), se una cella arriva in anticipo, la cella successiva è attesa comunque all'istante $t_1 + 2T$. Si supponga che la regola sia differente, ovvero che la cella successiva sia attesa all'istante $t_2 + 2T$, e che il mittente faccia un uso massiccio di questa regola. Qual è la massima velocità di trasmissione che può essere raggiunta? Per $T = 10 \mu\text{s}$ e $L = 2 \mu\text{s}$, dare la velocità di picco originale e quella nuova.
43. Qual è la massima lunghezza di un getto in una connessione ABR in una rete ATM da 155,52 Mbps il cui valore *PCR* sia 200.000 e il cui valore *L* sia 25 μs ?
44. Scrivere un programma che simuli il routing basato su flooding. Ogni pacchetto dovrebbe contenere un contatore da decrementare a ogni hop. Quando il contatore raggiunge lo 0, il pacchetto viene scartato. Il tempo è discreto, ovvero ogni canale trasporta un pacchetto a ogni istante temporale. Scrivere tre versioni del programma: (1) vengono utilizzate tutte le linee; (2) vengono utilizzate tutte le linee eccetto quella di input; (3) vengono utilizzate le migliori k linee (scelte statisticamente). Si confronti il flooding con il routing deterministico ($k = 1$) in termini di ritardi e di capacità utilizzata.
45. Scrivere un programma che simuli una rete di computer basati su tempo discreto. Il primo pacchetto di ogni coda dei router viene trasportato per un hop a ogni istante temporale. Ogni router ha un numero finito di buffer. Se arriva un pacchetto e non c'è spazio per esso, viene scartato e non inoltrato. Al contrario, esiste un protocollo end-to-end, completo di timeout e di pacchetti di ack, che prima o poi fa ripartire il pacchetto dal router sorgente. Si tracci il grafico della capacità della rete come funzione dell'intervallo di timeout end-to-end, parametrizzato dal numero di errori.

Capitolo 6

IL LIVELLO TRASPORTO

Il livello trasporto non è solo un altro strato. È il cuore dell'intera gerarchia di protocolli. Il suo compito è quello di fornire comunicazioni affidabili e convenienti dal computer sorgente al computer destinatario, indipendentemente dalla rete fisica utilizzata. Senza il livello trasporto, l'intera nozione di gerarchia di protocolli avrebbe poco senso. In questo capitolo lo studieremo dettagliatamente, analizzando i suoi servizi, la sua progettazione, i suoi protocolli e le sue prestazioni.

6.1 Il servizio di trasporto

In questo paragrafo introdurremo il servizio di trasporto. Vedremo quale tipo di servizio viene offerto al livello delle applicazioni (o a quello sessione, se presente), e in particolare come caratterizzarne la qualità. Analizzeremo quindi il modo in cui le applicazioni accedono al servizio di trasporto; in altre parole, quali sono le caratteristiche dell'interfaccia.

6.1.1 I servizi forniti ai livelli superiori

Il vero scopo del livello trasporto è quello di fornire un servizio efficiente, affidabile e conveniente ai propri utenti, in genere processi del livello applicazione. Per raggiungere questo obiettivo, il livello trasporto utilizza i servizi forniti dal livello rete. Lo hardware e/o il software che esegue il lavoro (internamente al livello trasporto) è chiamato **entità di trasporto**. L'entità di trasporto può essere situata nel nucleo del sistema operativo, in un processo utente separato, in un pacchetto di libreria collegato alle applicazioni di rete, oppure nella scheda di rete. In alcuni casi può essere lo stesso fornitore a offrire un servizio di trasporto affidabile, nel qual caso l'entità di trasporto è situata in speciali sistemi di interfaccia al confine della rete a cui si connettono gli host. I collegamenti (logici) fra i livelli rete, trasporto e applicazione sono illustrati in figura 6-1.

Proprio come nel livello rete, dove esistono servizi orientati alla connessione e servizi senza connessione, nel livello trasporto esistono le stesse tipologie di servizio. Il servizio di trasporto orientato alla connessione è simile per molti aspetti all'analogo servizio di rete. In entrambi i casi, le connessioni sono caratterizzate da tre fasi: creazione, trasferimento dei dati e chiusura. Anche l'indirizzamento e il controllo di flusso sono molto simili in entrambi gli strati. Inoltre, anche il servizio di trasporto senza connessione è molto simile all'analogo servizio di rete.

Una domanda ovvia è la seguente: se i servizi del livello trasporto sono così simili a quelli del livello rete, perché esistono due strati distinti? Perché uno solo non è sufficiente? La

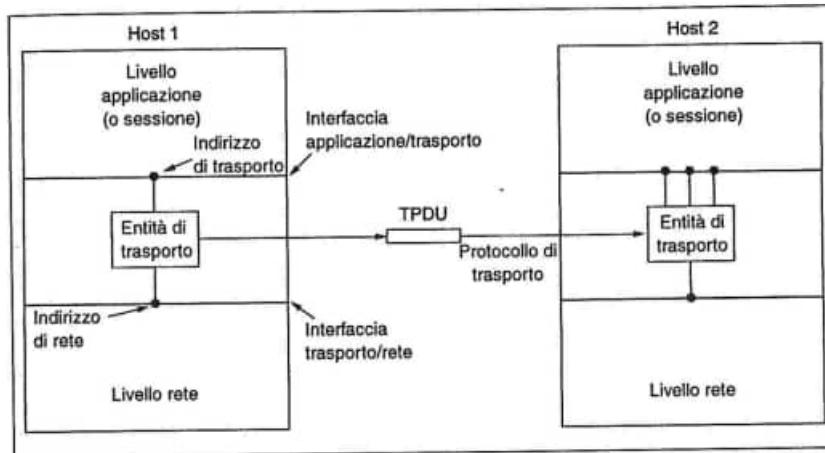


Fig. 6-1 Gli strati di rete, trasporto e applicazione.

risposta è cruciale, ed è collegata alla figura 1-16. In questo esempio possiamo vedere che il livello rete fa parte della sottorete di comunicazione ed è eseguito dal fornitore (almeno nelle WAN). Cosa succede se il livello rete offre servizi orientati alle connessioni, ma è inaffidabile? Cosa succede se perde frequentemente pacchetti? Cosa succede se i router si guastano di tanto in tanto?

I problemi esistono, questo è quanto. L'utente non ha controllo sulla rete; non può risolvere i problemi di un servizio inadeguato utilizzando router migliori, oppure aumentando la gestione degli errori nel livello data link. L'unica possibilità è quella di collocare un ulteriore strato sopra il livello rete che migliori la qualità del servizio. Se durante una lunga trasmissione un'entità di trasporto venisse informata che la sua connessione di rete è stata interrotta improvvisamente, senza nessuna indicazione su cosa sia successo ai dati attualmente in transito, potrebbe creare una nuova connessione di rete con l'entità di trasporto remota. Utilizzando questa nuova connessione di rete, potrebbe spedire una richiesta al suo corrispondente domandando quali dati sono arrivati e quali no, e quindi ripartire da dove ha smesso.

In pratica, l'esistenza del livello trasporto fa in modo che il servizio di comunicazione sia più affidabile di quello offerto dal livello rete sottostante. Eventuali problemi relativi a pacchetti persi o a dati corrotti possono essere rilevati e risolti dal livello trasporto. Inoltre, le primitive del livello trasporto possono essere progettate in modo da essere indipendenti da quelle del livello rete, le quali possono variare considerabilmente da rete a rete (ad es. un servizio senza connessione su LAN può essere molto differente da un servizio orientato alla connessione su WAN).

Grazie al livello trasporto, è possibile scrivere applicazioni basandosi su un insieme standard di primitive, che possono essere utilizzate in una grande varietà di reti, senza preoccuparsi di interfacce di rete differenti e di trasmissioni inaffidabili. Se tutte le reti reali

6.1 Il servizio di trasporto

fossero senza difetti e condividessero le stesse primitive, il livello trasporto probabilmente non sarebbe necessario. Tuttavia, nel mondo reale esso realizza l'importante funzione di isolare gli strati più alti dalla tecnologia, dalla progettazione e dalle imperfezioni delle reti.

Per questa ragione, molti hanno distinto gli strati da 1 a 4 da un lato, e il livello (gli strati) oltre il 4 dall'altro. I quattro strati inferiori possono essere visti come **fornitore di servizio di trasporto**, mentre gli strati superiori rappresentano l'**utilizzatore dei servizi di trasporto**. Questa distinzione fra fornitore e utilizzatore ha un impatto considerevole sulla progettazione degli strati e colloca il livello trasporto in una posizione chiave, in quanto costituisce la più importante frontiera tra il fornitore e l'utilizzatore dei servizi di trasmissione dati affidabile.

6.1.2 Qualità del servizio

Un altro modo di esaminare il livello trasporto riguarda la sua funzione principale di miglioramento della qualità del servizio (**QoS, Quality of Service**) fornita dal livello rete. Se il servizio di rete è esente da difetti, il livello trasporto non incontra difficoltà. Se, d'altro canto, il servizio di rete è scadente, il livello trasporto deve colmare il divario tra ciò che desidera l'utente del servizio di trasporto e ciò che fornisce il livello rete.

Sebbene a prima vista il concetto di qualità del servizio possa sembrare vago (fare in modo che tutti concordino su ciò che costituisce un "buon" servizio è un esercizio non banale), QoS può essere caratterizzato da un insieme di parametri specifici, come abbiamo visto nel capitolo 5. Al momento di creare la connessione, il servizio di trasporto può consentire all'utente di specificare sia i valori richiesti che quelli accettabili e minimi per i vari parametri del servizio. Alcuni dei parametri si applicano anche ai servizi di trasporto senza connessione. È compito del livello trasporto esaminare questi parametri e, a seconda del servizio di rete disponibile, determinare se è in grado di fornire il servizio richiesto. Nel seguito di questo paragrafo discuteremo alcuni esempi di parametri QoS, elencati in figura 6-2. Si noti che poche reti o protocolli forniscono tutti questi parametri. Alcuni fanno del loro meglio per ridurre il tasso di errori residuo e si limitano a questo. Altri hanno invece elaborate architetture QoS (Campbell et al., 1994).

Ritardo di creazione della connessione
Probabilità di fallimento nella creazione di una connessione
Capacità
Ritardo di trasmissione
Tasso di errori
Protezione
Priorità
Elasticità

Fig. 6-2 Tipici parametri QoS del livello trasporto.

Il parametro *Connection establishment delay* (ritardo di creazione della connessione) rappresenta la quantità di tempo che passa tra la richiesta di connessione del trasporto e la ricezione della conferma da parte dell'utente del servizio di trasporto. Include il ritardo indotto dalla computazione nell'entità di trasporto remota. Come tutti i parametri che misurano un ritardo, a ritardo minore corrisponde un servizio migliore.

Il parametro *Connection establishment failure probability* (probabilità di fallimento nella creazione di una connessione) rappresenta la probabilità che una connessione non venga creata entro il ritardo di creazione massimo; ciò può essere dovuto ad esempio a congestione nella rete, a mancanza di spazio in qualche tabella, oppure ad altri problemi interni. Il parametro *Throughput* (capacità) misura il numero di byte di dati utente trasferiti in un secondo, calcolato durante un certo intervallo di tempo. Il suo valore viene misurato separatamente per ogni direzione.

Il parametro *Transit delay* (ritardo di trasmissione) misura il tempo che intercorre fra la spedizione di un messaggio da parte dell'utente del trasporto del computer mittente e la ricezione da parte dell'utente del trasporto del computer destinatario. Come per il parametro *Throughput*, ogni direzione è gestita separatamente.

Il parametro *Residual error ratio* (tasso di errori residui) misura il numero di messaggi persi o corrotti come frazione del numero totale di messaggi spediti. In teoria, il tasso di errori residui dovrebbe essere 0, in quanto è compito del livello trasporto mascherare tutti gli errori del livello rete. In pratica può avere un (piccolo) valore finito.

Il parametro *Protection* (protezione) permette all'utente del trasporto di richiedere al livello trasporto protezione contro la possibilità che terze parti non autorizzate (tramite intercettazioni) leggano o modifichino i dati trasmessi.

Il parametro *Priority* (priorità) consente a un utente del livello trasporto di indicare che alcune delle sue connessioni sono più importanti di altre, e di assicurarsi che, in caso di congestione, le connessioni ad alta priorità siano servite prima delle connessioni a bassa priorità.

Infine, il parametro *Resilience* (resilienza) indica la probabilità che lo stesso strato trasporto interrompa spontaneamente una connessione a causa di problemi interni o di congestione.

I parametri QoS vengono specificati dall'utente con la richiesta di connessione. Possono essere indicati sia i valori accettabili che quelli minimi. In alcuni casi, dopo aver visto i parametri QoS, il livello trasporto può comprendere immediatamente che alcuni di essi sono irrealizzabili, nel qual caso risponde al cliente che il tentativo di connessione è fallito, senza disturbare la destinazione. Il rapporto sul fallimento specifica le ragioni dell'insuccesso.

In altri casi, il livello trasporto sa di non poter raggiungere l'obiettivo richiesto (ad es. capacità di 600 Mbps), ma è in grado di fornire un servizio minore, ma ancora accettabile (ad es. 150 Mbps). Spedisce quindi il valore inferiore e il valore minimo accettabile alla macchina remota, chiedendo di stabilire una connessione. Se la macchina remota non è in grado di gestire il valore proposto, ma può gestire un valore superiore a quello minimo, può fare un contropatto. Se non è in grado di gestire un valore superiore a quello minimo, rifiuta il tentativo di connessione. Infine, l'utente del trasporto viene informato sull'esito

della richiesta di connessione e, in caso di creazione, sul valore dei parametri su cui si è raggiunto l'accordo.

Questo processo è chiamato **negoziare delle opzioni**. Una volta che le opzioni sono state negoziate, non vengono più modificate durante la vita della connessione. Per evitare che i clienti siano troppo ingordi, molti fornitori tendono a richiedere più denaro per servizi migliori.

6.1.3 Primitive del servizio di trasporto

Le primitive del servizio di trasporto permettono ai loro utenti (ad es., programmi applicativi) di accedere al servizio stesso. Ogni servizio di trasporto ha le proprie primitive di accesso. Esamineremo innanzitutto un semplice servizio di trasporto (ipotetico) e quindi daremo un'occhiata a un esempio reale.

I servizi di trasporto sono simili ai servizi di rete, ma ci sono anche alcune importanti differenze. La differenza principale è che un servizio di rete intende modellare i servizi delle reti reali, con le loro imperfezioni. Le reti reali possono perdere pacchetti e quindi i servizi di rete sono generalmente inaffidabili.

Al contrario, il servizio di trasporto (orientato alla connessione) è in genere affidabile. Naturalmente, le reti reali non sono esenti da errori, ma questo è proprio lo scopo del livello trasporto – fornire un servizio affidabile sopra una rete inaffidabile.

Ad esempio, si considerino due processi connessi tramite pipe in UNIX, che assumono che la loro connessione sia perfetta. Non vogliono sapere nulla di ack, pacchetti persi, congestioni, e così via. Tutto ciò che vogliono è una connessione affidabile al 100%. Il processo A inserisce i dati da un lato del canale, e il processo B li estrae dall'altro. Questo è il compito di un servizio di trasporto orientato alla connessione: nascondere le imperfezioni del servizio di rete in modo che i processi possano assumere l'esistenza di un flusso esente da errori.

D'altra parte, il livello trasporto può anche fornire un servizio inaffidabile basato su datagram, ma data la sua scarsa complessità, in questo capitolo ci concentreremo sui servizi di trasporto orientati alla connessione.

Una seconda differenza tra i servizi di rete e quelli di trasporto riguarda il destinatario per cui è inteso il servizio. I servizi di rete sono utilizzati solo dalle entità di trasporto. Pochi utenti scrivono le loro entità di trasporto, e quindi pochi utenti o programmi vedono mai il servizio di rete vero e proprio. Al contrario, molti programmi (e quindi programmati) utilizzano le primitive di trasporto. Conseguentemente, il servizio di trasporto deve essere conveniente e facile da usare.

Per capire a cosa dovrebbe assomigliare un servizio di trasporto, si considerino le cinque primitive elencate in figura 6-3. Questa interfaccia di trasporto è veramente essenziale, ma rappresenta il succo di ciò che dovrebbe essere una interfaccia di trasporto orientata alla connessione. Consente ai programmi applicativi di stabilire, usare e chiudere le connessioni; ciò è sufficiente per molte applicazioni.

Per vedere come queste primitive possano essere utilizzate, si consideri un'applicazione basata su un server e un certo numero di clienti remoti. Innanzitutto, il server esegue una primitiva LISTEN, generalmente chiamando una procedura di libreria che implementa una chiamata di sistema che blocca il server fino a quando un cliente non si fa sentire. Quando

Primitiva	TPDU spedito	Significato
LISTEN	(nessuna)	Si blocca fino a quando un processo cerca di connettersi
CONNECT	CONNECTION REQ.	Cerca di creare attualmente una connessione
SEND	DATA	Spedizione di informazioni
RECEIVE	(nessuna)	Si blocca fino all'arrivo di un TPDU DATA
DISCONNECT	DISCONNECTION REQ.	Questo lato vuole chiudere la connessione

Fig. 6-3 Le primitive di un semplice servizio di trasporto.

un cliente vuole parlare con il server, esegue una primitiva CONNECT. L'entità di trasporto realizza questa primitiva bloccando il chiamante e spedendo un pacchetto al server. Nel campo contenuto di questo pacchetto viene incapsulato un messaggio del livello trasporto destinato all'entità di trasporto del server.

A questo punto sono necessarie alcune note sulla terminologia. Per mancanza di un termine migliore, utilizzeremo (anche se riluttanti) il termine **TPDU** (**T**ransport **P**rotocol **D**ata **U**nit – unità di dati del protocollo di trasporto) per caratterizzare i messaggi spediti da un'entità di trasporto a un'altra. Quindi i TPDU (scambiati dal livello trasporto) sono contenuti in pacchetti (scambiati dal livello rete). A loro volta i pacchetti sono contenuti in frame (scambiati dal livello data link). Quando arriva un frame, il livello data link elabora il preamble del frame e passa il contenuto del frame all'entità di rete. L'entità di rete elabora il preamble del pacchetto e passa il suo contenuto all'entità di trasporto. Questo annidamento è illustrato in figura 6.4.

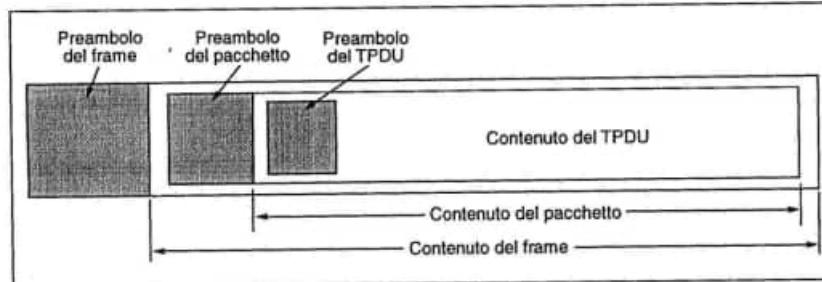


Fig. 6-4 Annidamento di TPDU, pacchetti e frame.

Tornando al nostro esempio client-server, la chiamata CONNECT del cliente fa in modo che venga spedito un TPDU CONNECTION REQUEST al server. Quando questo arriva, l'entità di trasporto verifica che il server sia bloccato su un LISTEN (ovvero sia interessato a gestire richieste). Quindi sblocca il server e spedisce indietro un TPDU CONNECTION ACCEPTED al cliente. Quando questo TPDU arriva, il cliente viene sbloccato e la connessione viene stabilita.

6.1 Il servizio di trasporto

I dati possono ora essere scambiati utilizzando le primitive SEND e RECEIVE. Nella forma più semplice, entrambi gli interlocutori possono eseguire una RECEIVE (bloccante) per aspettare che l'altro interlocutore esegua un SEND. Quando il TPDU arriva, il ricevente viene sbloccato, può elaborare il TPDU e spedire una risposta. Fino a quando entrambi i lati possono tener traccia del turno di spedizione, questo schema lavora molto bene. Si noti che nel livello rete, anche un semplice scambio unidirezionale di dati è più complicato di quello corrispondente nel livello trasporto. Per ogni pacchetto di dati verrà (prima o poi) spedito un ack. Anche per i pacchetti che trasportano TPDU di controllo verrà fornito ack, implicitamente o esplicitamente. Questi ack vengono gestiti dalle entità di trasporto utilizzando il protocollo del livello rete e non sono visibili agli utenti del livello trasporto. Allo stesso modo, le entità di trasporto dovranno occuparsi di timer e ritrasmissioni. Nessuno di questi meccanismi è visto dagli utenti del livello trasporto. Per gli utenti del trasporto, una connessione è un canale di bit affidabile: un'utente inserisce dentro i bit e questi appaiono magicamente all'altro estremo. Questa abilità di nascondere la complessità è la ragione per cui i protocolli stratificati sono strumenti così potenti. Quando una connessione non è più necessaria, deve essere chiusa per liberare lo spazio nelle tabelle all'interno delle due entità di trasporto. La disconnessione ha due varianti: asimmetrica e simmetrica. Nella variante asimmetrica, uno degli utenti del trasporto può chiamare una primitiva DISCONNECT, che produce un TPDU DISCONNECT che viene spedito all'entità di trasporto remota. All'arrivo, la connessione viene chiusa. Nella variante simmetrica, ogni direzione viene chiusa separatamente, indipendentemente dall'altra. Quando un lato esegue un DISCONNECT, questo significa che non ha più dati da spedire, mentre è ancora disposto ad accettare dati dal suo partner. In questo modello, una connessione viene chiusa quando entrambi i lati hanno eseguito un DISCONNECT. Un diagramma degli stati per la creazione e la chiusura delle connessioni per queste semplici primitive è mostrato in figura 6-5. Ogni transizione è attivata da qualche evento, da una primitiva eseguita dall'utente locale del trasporto oppure da un pacchetto in arrivo. Per semplicità, assumiamo che ogni TPDU venga confermato separatamente. Assumiamo inoltre che venga utilizzato un modello di disconnessione simmetrico a partire dal cliente. Si prega di notare che questo modello è estremamente semplificato. Daremo un'occhiata a un modello più realistico in seguito.

I Socket di Berkeley

Esaminiamo ora brevemente un altro insieme di primitive di trasporto, ovvero le primitive SOCKET per TCP utilizzate nello UNIX di Berkeley (elencate in figura 6-6). A grandi linee, le primitive SOCKET seguono il modello del nostro primo esempio, sebbene offrano più possibilità e flessibilità. Non analizzeremo i TPDU corrispondenti; questa discussione dovrà attendere fino a quando non studieremo TCP nel seguito del capitolo.

Le prime quattro primitive nella lista sono eseguite dal server nell'ordine corrispondente. La primitiva SOCKET crea un nuovo punto di accesso e alloca spazio per esso nelle tabelle dell'entità di trasporto. I parametri della chiamata specificano il formato di indirizzamento che dovrà essere usato, il tipo di servizio desiderato (ad es. flusso di byte affidabile) e il protocollo. Una chiamata SOCKET che ha successo restituisce un descrittore di file ordinario da utilizzare nelle chiamate successive, allo stesso modo di una chiamata OPEN.

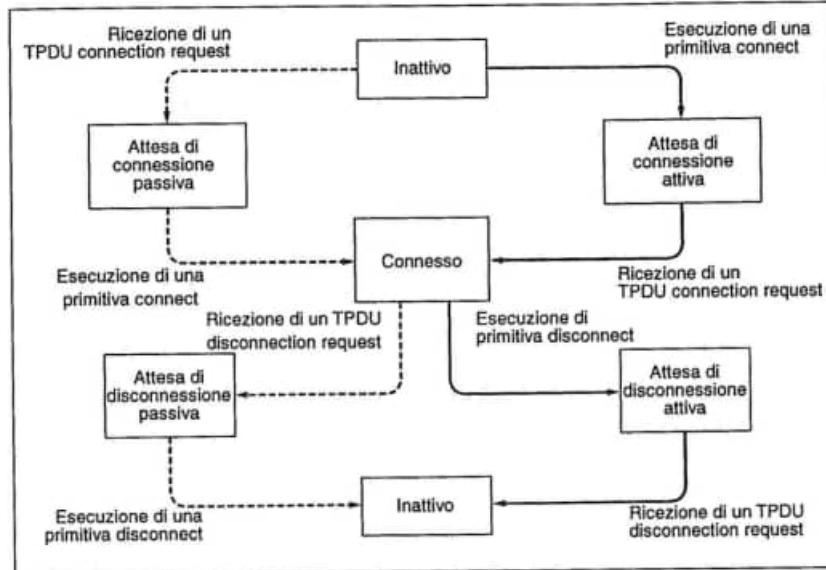


Fig. 6-5 Un diagramma degli stati per un semplice schema di gestione delle connessioni. Transizioni etichettate in corsivo sono causate da pacchetti in arrivo. Le linee continue mostrano la sequenza degli stati del cliente. Le linee tratteggiate mostrano la sequenza degli stati del server.

Primitiva	Significato
SOCKET	Crea un nuovo punto finale di comunicazione
BIND	Associa un indirizzo locale ad un socket
LISTEN	Annuncia la volontà di accettare connessioni; da dimensione alla coda
ACCEPT	Arresta il visitatore finché non arriva la connessione
CONNECT	Tenta attivamente di stabilire la connessione
SEND	Spedisce dati sulla connessione
RECEIVE	Riceve dati dalla connessione
CLOSE	Chiude la connessione

Fig. 6-6 Le primitive SOCKET per TCP.

I socket appena creati non hanno indirizzi. Questi sono assegnati utilizzando la primitiva BIND. Una volta che il server ha legato un indirizzo a un socket, i clienti remoti possono connettersi a esso. Il motivo per cui la chiamata SOCKET non crea direttamente un indirizzo è il fatto che qualche processo si interessa dei propri indirizzi (ad es. ha utilizzato lo stesso indirizzo, conosciuto da tutti, per anni), mentre altri non se ne curano.

Segue la chiamata LISTEN, che alloca spazio per la coda delle chiamate in arrivo, nel caso che molti clienti cerchino di connettersi contemporaneamente. Al contrario di listen del nostro primo esempio, la chiamata LISTEN del modello basato su socket non è bloccante. Per bloccarsi in attesa di una connessione in arrivo, il server esegue una primitiva ACCEPT. Quando arriva un TPDU che richiede una connessione, l'entità di trasporto crea un nuovo socket con le stesse proprietà dell'originale e restituisce per esso un descrittore di file. Il server può creare un processo o un thread per gestire la connessione del nuovo socket, e tornare ad aspettare la prossima connessione sul socket originario.

A questo punto analizziamo il lato del cliente. Anche qui i socket devono essere prima creati con la primitiva SOCKET, mentre BIND non è richiesto in quanto l'indirizzo utilizzato non ha importanza per il cliente. La primitiva CONNECT blocca il chiamante e attiva la procedura di connessione. Quando questa è completata (cioè quando il TPDU appropriato viene ricevuto dal server), il processo cliente viene sbloccato e la connessione viene stabilita. A questo punto, entrambi i lati possono utilizzare SEND e RECEIVE per trasmettere e ricevere dati sulla connessione full duplex.

Nei socket, la chiusura delle connessioni è simmetrica. Quando entrambi gli interlocutori hanno eseguito un primitiva CLOSE, la connessione viene chiusa.

6.2 Elementi del protocollo di trasporto

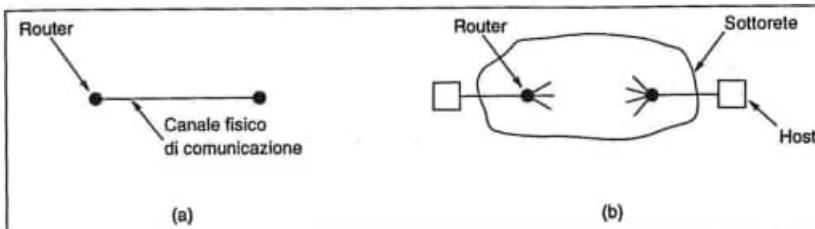
Il servizio di trasporto viene realizzato da un **protocollo di trasporto** utilizzato tra le due entità di trasporto. Da un certo punto di vista i protocolli di trasporto somigliano ai protocolli data link che abbiamo studiato dettagliatamente nel capitolo 3. Entrambi devono occuparsi, tra l'altro, di controllo degli errori, di gestione dei numeri di sequenza e di controllo di flusso.

Tuttavia, tra i due strati esistono anche significative differenze, dovute alle differenti caratteristiche degli ambienti in cui i due protocolli operano, come si può vedere in figura 6-7. Nel livello data link due router comunicano direttamente attraverso un canale fisico, mentre nel livello trasporto questo canale fisico è sostituito dall'intera rete. Questa differenza ha molte implicazioni importanti per quanto riguarda i protocolli.

Da un lato, nel livello data link non è necessario che un router specifichi con quale router vuole parlare – ogni linea in uscita identifica univocamente un particolare router. Nel livello trasporto, è invece necessario un indirizzamento esplicito delle destinazioni.

D'altra parte, il processo per stabilire una connessione lungo il cavo di figura 6-7 (a) è semplice: l'altro capo è sempre presente (a meno che non sia guasto, nel qual caso non è presente). In entrambi i casi, non vi sono molte difficoltà. Come vedremo, nel livello trasporto la creazione iniziale delle connessioni è più complicata.

Un'altra differenza estremamente fastidiosa tra il livello data link e il livello trasporto è la capacità potenziale di memorizzazione della rete. Quando un router spedisce un frame,



questo può arrivare oppure essere perso, ma non può vagare in giro per la rete, nascondendosi in un lontano angolo del mondo, e quindi apparire all'improvviso 30 s dopo in un momento inopportuno. Se la rete è internamente basata su datagram e su un routing adattivo, c'è una probabilità non trascurabile che un pacchetto possa essere memorizzato per un certo numero di secondi e quindi consegnato successivamente. In alcuni casi, le conseguenze di questa abilità della rete di memorizzare pacchetti possono essere disastrose e richiedono l'uso di protocolli speciali.

L'ultima differenza tra gli strati data link e trasporto è di tipo quantitativo piuttosto che qualitativo. La bufferizzazione e il controllo di flusso sono necessari in entrambi gli strati, ma la presenza di una gran quantità di connessioni (il cui numero può variare dinamicamente) del livello trasporto può richiedere un approccio differente a quello utilizzato nel livello data link. Nel capitolo 3, alcuni dei protocolli allocavano un numero fisso di buffer per ogni linea, in modo che all'arrivo di un frame esistesse sempre un buffer disponibile. Il gran numero di connessioni che devono essere gestite nel livello trasporto rende l'idea di dedicare molti buffer ad ognuna di esse meno attraente. Nel seguito esamineremo tutte queste importanti problematiche e altre ancora.

6.2.1 Indirizzamento

Quando un'applicazione desidera creare una connessione con un processo remoto, deve specificare con chi vuole connettersi. (Il trasporto senza connessione ha lo stesso problema: a chi deve essere spedito ogni messaggio?) Il metodo normalmente utilizzato consiste nel definire indirizzi di trasporto presso i quali i processi possono attendere le richieste di connessione. In Internet, questi punti di accesso sono coppie (Indirizzo IP-porta locale). Nelle reti ATM sono AAL-SAP. Utilizzeremo il termine neutrale **TSAP** (**T**ransport **S**ervice **A**ccess **P**oint – punto di accesso del servizio di trasporto). Gli analoghi punti di accesso nel livello rete sono chiamati NSAP. Gli indirizzi IP sono esempi di NSAP.

La figura 6.8 illustra la relazione tra NSAP, TSAP, connessioni di rete e connessioni di trasporto per una rete orientata alla connessione (ad es. ATM). Si noti che un'entità di trasporto gestisce generalmente più di un TSAP. In qualche rete esistono anche NSAP multipli, mentre in altre ogni macchina ha un solo NSAP (ad es. un solo indirizzo IP). Un possibile scenario di creazione di una connessione di trasporto su uno strato rete orientato alla connessione è il seguente.

1. Un server orario sull'host 2 si mette in ascolto al TSAP 122 aspettando una chiamata in arrivo. Il metodo con cui il processo si mette in ascolto a un TSAP non riguarda il modello di rete e dipende interamente dal sistema operativo locale. Ad esempio, può essere utilizzata una chiamata come la nostra LISTEN.
2. Un'applicazione eseguita sull'host 1 vuole conoscere l'orario e quindi emette una richiesta CONNECT specificando TSAP 6 come sorgente e TSAP 122 come destinazione.
3. L'entità di trasporto dell'host 1 seleziona uno degli indirizzi di rete della sua macchina (se ne ha più di uno) e crea una connessione di rete (con una rete senza connessione, questo passo non sarebbe necessario). Usando questa connessione di rete, l'entità di trasporto dell'host 1 può comunicare con l'entità di trasporto dell'host 2.
4. La prima cosa che l'entità di trasporto di 1 dice alla sua pari presso 2 è: "Buon giorno. Vorrei stabilire una connessione di trasporto tra la mia TSAP 6 e la tua TSAP 122. Che ne dici?".
5. L'entità di trasporto in 2 chiede quindi al server orario presso TSAP 122 se desidera accettare una nuova connessione. Se questo è d'accordo, la connessione di trasporto viene creata.

Si noti che la connessione di trasporto va da TSAP a TSAP, mentre quella di rete attraversa solo parte del percorso, da NSAP a NSAP.

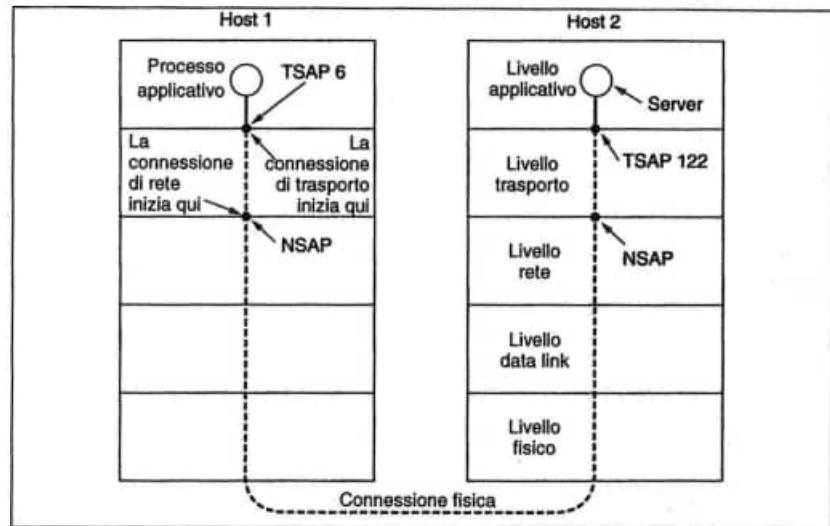


Fig. 6-8 TSAP, NSAP e connessioni.

L'illustrazione appena presentata è elegante, ma vi è un piccolo problema nascosto: come

fa il processo utente dell'host 1 a sapere che il server orario è in ascolto al TSAP 122? Ad esempio, è possibile che il server orario sia stato in ascolto al TSAP 122 per anni, e gradualmente tutti gli utenti della rete lo siano venuti a sapere. In questo modello, i servizi hanno indirizzi TSAP stabili che possono essere stampati su carta e dati ai nuovi utenti quando si collegano per la prima volta.

Mentre gli indirizzi TSAP stabili possono andare bene nel caso di un piccolo numero di servizi chiave che non cambiano mai, in generale i processi utente vogliono comunicare con altri processi utente che esistono solo per poco tempo e non hanno un indirizzo TSAP noto in anticipo. Inoltre, se esistono molti processi server, gran parte dei quali vengono utilizzati raramente, sarebbe uno spreco fare in modo che ognuno di essi fosse attivo e in ascolto tutto il giorno presso un indirizzo TSAP stabile. In pratica, è necessario uno schema migliore.

Uno di questi schemi, utilizzato dagli host UNIX in Internet, è mostrato in figura 6-9 in una forma semplificata. È conosciuto come **protocollo della connessione iniziale**. Invece di imporre che ogni server resti in ascolto su un TSAP noto a tutti, una macchina che vuole offrire servizi a utenti remoti esegue un **server generico** speciale che agisce per conto dei server utilizzati poco. Questo server ascolta contemporaneamente un insieme di porte, aspettando una richiesta di connessione TCP. Gli utenti potenziali di un servizio iniziano facendo una richiesta CONNECT, specificando l'indirizzo TSAP (porta TCP) del servizio desiderato. Se nessun server è in attesa della richiesta, ottengono una connessione al server generico, come mostrato in figura 6-9 (a).

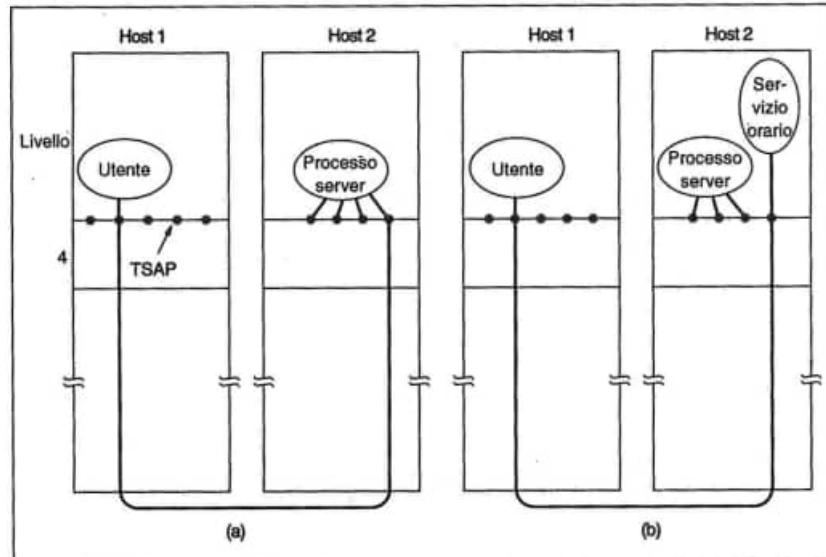


Fig. 6-9 Come un processo nel computer 1 crea una connessione con il servizio orario del computer 2.

6.2 Elementi del protocollo di trasporto

Dopo aver ricevuto la richiesta in arrivo, il server generico crea il server specifico, permettendogli di ereditare la connessione con l'utente. Il nuovo server completa il lavoro desiderato, mentre il server generico torna ad aspettare nuove richieste, come mostrato in figura 6-9 (b).

Mentre il protocollo di connessione iniziale lavora bene con i server che possono essere creati quando sono necessari, ci sono molte situazioni in cui i servizi esistono indipendentemente dal server generico. Un file server, ad esempio, ha bisogno di essere eseguito su hardware speciale (una macchina con un disco) e non può essere creato al momento quando qualcuno vuole comunicare.

Per gestire questa situazione viene spesso utilizzato uno schema alternativo. In questo modello esiste un processo speciale chiamato **name server**, o in qualche caso **directory server**. Per trovare l'indirizzo TSAP corrispondente a un certo nome di servizio, come "time-of-day", un utente crea una connessione con il name server (che ascolta un TSAP noto a tutti). L'utente spedisce quindi un messaggio specificando il nome del servizio, e il name server risponde con l'indirizzo TSAP. Quindi l'utente rilascia la connessione con il name server e ne crea una nuova con il servizio desiderato.

In questo modello, tutti i nuovi servizi devono registrarsi presso il name server, fornendo sia l'identificatore di servizio (tipicamente una stringa ASCII) sia l'indirizzo del TSAP corrispondente. Il name server memorizza queste informazioni nel suo database interno, in modo da poter rispondere quando arriveranno le richieste.

La funzione del name server è analoga al servizio informazioni nei sistemi telefonici – fornisce una corrispondenza tra nomi e numeri. Proprio come nei sistemi telefonici, è essenziale che l'indirizzo del TSAP utilizzato dal name server (o dal server di processo utilizzato nel protocollo di connessione iniziale) sia veramente noto a tutti. Se non si conosce il numero dell'operatore addetto alle informazioni, non si può chiamare l'operatore per chiederlo. Se pensate che il numero del servizio informazioni sia ovvio, provatelo una volta o l'altra in uno stato estero.

Supponiamo a questo punto che l'utente abbia trovato con successo l'indirizzo TSAP a cui deve connettersi. Un'altra questione interessante è la seguente: in che modo l'entità di trasporto locale viene a conoscere presso quale macchina si trova il TSAP? Più specificatamente, come fa l'entità di trasporto a conoscere quale indirizzo di rete utilizzare per creare una connessione di rete con l'entità di trasporto remota che gestisce il TSAP richiesto?

La risposta dipende dalla struttura degli indirizzi TSAP. Una possibilità prevede che i TSAP corrispondano a **indirizzi gerarchici**, nel qual caso ogni indirizzo consiste in una sequenza di campi utilizzati per frazionare lo spazio di indirizzamento. Ad esempio, un indirizzo TSAP veramente universale potrebbe avere la seguente struttura:

indirizzo = <galassia> <stella> <pianeta> <stato> <rete> <host> <porta>

Con questo schema, è banale individuare un qualunque TSAP nell'universo conosciuto. In modo equivalente, se un indirizzo TSAP è una concatenazione di un indirizzo NSAP e una porta (un identificatore locale che specifica uno degli TSAP locali), allora quando un indirizzo TSAP viene dato a un'entità di trasporto, questa usa l'indirizzo NSAP contenuto nell'indirizzo TSAP per raggiungere l'entità di trasporto remota appropriata.

Come semplice esempio di un indirizzo gerarchico, si consideri il numero telefonico 19076543210. Questo numero può essere scandito come 1-907-654-3210, dove 1 è il codice dello stato (Stati Uniti + Canada), 907 è il codice di area (Alaska), 654 è un terminale in Alaska, e 3210 è una delle porte (linee degli abbonati) in quel terminale. L'alternativa a uno spazio di indirizzamento gerarchico è uno **spazio di indirizzamento piatto**. Se gli indirizzi TSAP non sono gerarchici, per trovare la macchina appropriata è necessario un secondo livello di mappatura. Dovrebbe esistere un name server che prende indirizzi di trasporto come input e restituisce indirizzi di rete come output. Alternativamente, in alcune situazioni (ad es. in una LAN) è possibile emettere una richiesta in broadcast chiedendo che la macchina di destinazione si identifichi gentilmente da sola tramite la spedizione di un pacchetto.

6.2.2 Creare una connessione

Creare una connessione può sembrare facile, ma in realtà è sorprendentemente complesso. A prima vista, potrebbe sembrare sufficiente che un'entità di trasporto spedisca un TPDU CONNECTION REQUEST alla destinazione e attenda una risposta CONNECTION ACCEPTED. I problemi si presentano nel caso in cui la rete possa perdere, memorizzare o duplicare i pacchetti.

Si immagini una rete congestionata a tal punto che difficilmente gli ack tornano indietro in tempo e ogni pacchetto viene rispedito due o tre volte. Si supponga che la rete sia basata su datagram, e che ogni pacchetto segua un percorso differente. Alcuni pacchetti possono bloccarsi da qualche parte nella rete e richiedere molto tempo prima di arrivare; in altre parole, possono essere memorizzati nella rete e consegnati molto più tardi.

Una delle possibilità più spaventose è la seguente. Un utente crea una connessione con una banca, spedisce un messaggio chiedendo alla banca di trasferire un grande quantità di denaro nel conto di un persona "non del tutto fidata", e quindi chiude la connessione. Sfortunatamente, ogni pacchetto nello scenario viene duplicato e memorizzato nella rete. Dopo la chiusura della connessione, tutti i pacchetti escono nuovamente dalla rete e arrivano alla destinazione in ordine, richiedendo alla banca di creare una nuova connessione, trasferire il denaro (di nuovo) e quindi chiudere la connessione. La banca non ha modo di affermare che questi siano duplicati. Deve assumere che questa è una seconda transazione indipendente e trasferire nuovamente il denaro. Nel seguito di questo paragrafo studieremo il problema dei duplicati in ritardo, con un'enfasi speciale sugli algoritmi affidabili per la creazione di connessioni, in modo che scenari come questo non si possano realizzare.

Il nucleo del problema è l'esistenza dei duplicati in ritardo. Può essere affrontato in vari modi, nessuno dei quali è veramente soddisfacente. Una possibilità è quella di utilizzare indirizzi di trasporto "usa e getta". In questo approccio, ogni volta che è necessario un indirizzo di trasporto ne viene creato uno nuovo. Quando una connessione viene chiusa, l'indirizzo viene scartato. Questa strategia rende impossibile il modello del server generico di figura 6-9.

Un'ulteriore possibilità è quella di dare ad ogni connessione un'identificatore di connessione (cioè un numero di sequenza incrementato ad ogni creazione di connessione), scelto dal chiamante e copiato in ogni TPDU, compreso quello di richiesta della connessione. Dopo la chiusura di ogni connessione, l'entità di trasporto può aggiornare una tabella che elenchi le connessioni obsolete come coppie (entità di trasporto remota, identificatore di

connessione). Ogni volta che arriva una richiesta di connessione, può essere cercata nella tabella, per assicurarsi che non appartenga a una connessione chiusa in precedenza. Sfortunatamente, questo schema ha un problema fondamentale: richiede che ogni entità di trasporto mantenga indefinitamente una certa quantità di informazioni storiche sulle connessioni. Se una macchina si guasta e perde la propria memoria, perde anche il ricordo degli identificatori di connessione utilizzati.

Abbiamo quindi bisogno di una tattica diversa. Invece di consentire che i pacchetti all'interno della rete vivano per sempre, dobbiamo escogitare un meccanismo per eliminare i pacchetti anziani che vagabondano nella rete. Se possiamo fare in modo che nessun pacchetto viva più di un certo intervallo di tempo finito, il problema diventa in qualche modo più gestibile.

Il tempo di vita di un pacchetto può essere ristretto a un massimo conosciuto utilizzando una delle seguenti tecniche:

1. Progettazione ristretta delle reti.
2. Inserimento di un contatore di hop in ogni pacchetto.
3. Inserimento di un orario in ogni pacchetto.

La prima tecnica include qualsiasi metodo che eviti che i pacchetti vadano in ciclo, combinandolo con qualche metodo per limitare i ritardi di congestione sul percorso più lungo possibile (ignoto a priori). La seconda tecnica consiste nell'incrementare un contatore ogni volta che un pacchetto viene inoltrato. Il livello data link scarta ogni pacchetto il cui contatore abbia superato un certo limite. La terza tecnica richiede che ogni pacchetto contenga l'istante della sua creazione e che i router abbiano concordato di scartare qualsiasi pacchetto più vecchio di una certa quantità di tempo. Quest'ultima tecnica richiede che gli orologi dei router siano sincronizzati, un compito non facile a meno che la sincronizzazione sia ottenuta esternamente alla rete, ad esempio ascoltando una stazione radio che emetta periodicamente un segnale orario.

In pratica,abbiamo bisogno di garantire non solo che un pacchetto sia defunto, ma che lo siano anche tutti i suoi ack; per questo motivo introduciamo T , un multiplo non troppo elevato del massimo tempo di vita dei pacchetti. Il valore di moltiplicazione dipende dal protocollo e ha il semplice effetto di allungare T . Se aspettiamo un tempo T dopo la spedizione di un pacchetto, possiamo essere sicuri che tutte le sue tracce saranno scomparse e che né il pacchetto, né i suoi ack salteranno fuori dal nulla all'improvviso per creare complicazioni.

Dopo aver limitato la vita dei pacchetti, è possibile escogitare un metodo infallibile per stabilire una connessione in modo sicuro. Il metodo descritto in seguito si deve a Tomlinson (1975). Risolve il problema, ma introduce alcune particolarità. Il metodo fu raffinato ulteriormente da Sunshine e Dalal (1978). Alcune sue varianti sono largamente utilizzate in pratica.

Per aggirare il problema di un macchina che dopo un guasto perde tutta la memoria, Tomlinson propose di equipaggiare ogni host con un orologio. Non serve che gli orologi di host differenti siano sincronizzati. Si assume che ogni orologio abbia la forma di un

contatore binario incrementato a intervalli uniformi. Inoltre, il numero di bit nel contatore deve essere uguale o superiore al numero di bit nei numeri di sequenza. Ultima caratteristica (molto importante), si assume che l'orologio continui a funzionare anche se l'host si guasta.

L'idea di base consiste nel fare in modo che non possano esistere contemporaneamente due TPDU con lo stesso numero di sequenza. Quando viene creata una connessione, i k bit dell'orologio di ordine più basso vengono utilizzati come numero di sequenza iniziale (anch'esso di k bit). Quindi, a differenza del nostro protocollo del capitolo 3, ogni connessione inizia a numerare i propri TPDU con un numero di sequenza differente. L'insieme dei numeri di sequenza dovrebbe essere così grande da ottenere, che prima uno di essi venga riutilizzato, i vecchi TPDU con lo stesso numero di sequenza siano ormai scomparsi da tempo. Questa relazione lineare tra intervalli di tempo e numeri di sequenza iniziali è mostrata in figura 6-10.

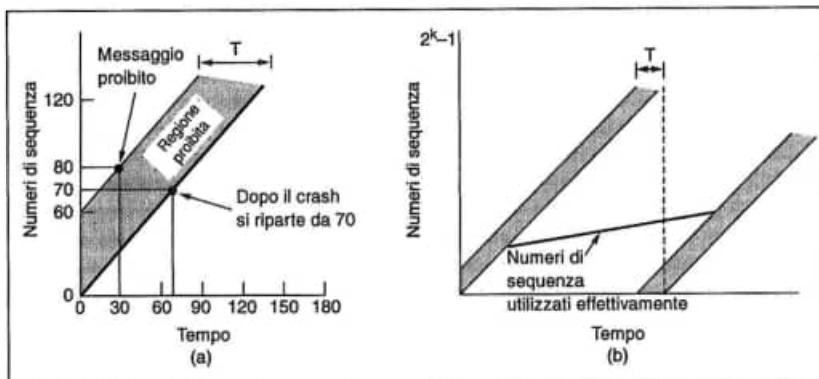


Fig. 6-10 (a) I TPDU non possono entrare nella regione proibita. (b) Il problema della risincronizzazione.

Dopo che entrambe le entità di trasporto hanno concordato un numero di sequenza iniziale, per controllare il flusso di dati si può utilizzare qualsiasi protocollo sliding window. In realtà, la curva dei numeri di sequenza iniziali (la linea in grassetto) non è davvero lineare, ma una scala, in quanto l'orologio avanza a passi discreti. Per semplicità ignoremo questo dettaglio.

La caduta di un host rappresenta un problema. Quando l'host torna a funzionare, la sua entità di trasporto non ricorda la sua posizione nell'insieme dei numeri sequenza. Una soluzione è richiedere che le entità di trasporto restino inattive per T secondi dopo un recupero, per consentire l'eliminazione di tutti i TPDU vecchi. Tuttavia, in una rete interconnessa molto complessa, T può essere molto grande, e quindi questa strategia non è attraente.

Per evitare di richiedere T secondi di inattività dopo un guasto, è necessario introdurre una nuova restrizione sull'uso dei numeri di sequenza. Possiamo comprenderne meglio

il bisogno tramite un esempio. Sia T , il tempo massimo di vita di un pacchetto, uguale a 60 s e assumiamo che gli orologi avanzino ogni secondo. Come mostrato in figura 6-10, il numero di sequenza iniziale per una connessione aperta al tempo x sarà x . Si immagini che al tempo $t = 30$ s sia assegnato un numero di sequenza 80 a un TPDU contenente dati ordinari che viene spedito sulla connessione 5 (aperta in precedenza). Chiamiamo questo TPDU X. Immediatamente dopo aver spedito il TPDU X, l'host si guasta e torna subito a funzionare. Al tempo $t = 60$, inizia a riaprire le connessioni da 0 a 4. Al tempo $t = 70$, riapre la connessione 5, utilizzando il numero di sequenza iniziale 70 come richiesto. Entro i 15 s successivi spedisce i TPDU contenenti dati compresi tra 70 e 80. Quindi al tempo $t = 85$, un nuovo TPDU con numero di sequenza 80 e numero di connessione 5 viene inserito nella rete. Sfortunatamente, il TPDU X esiste ancora. Se dovesse arrivare al ricevente prima del nuovo TPDU 80, TPDU X verrebbe accettato mentre il TPDU 80 corretto verrebbe rifiutato come duplicato.

Per prevenire tali problemi, dobbiamo evitare che i numeri di sequenza vengano usati (cioè assegnati a nuovi TPDU) nei T istanti di tempo precedenti al loro uso potenziale come numeri iniziali. Le combinazioni illegali di tempo e di numeri di sequenza sono racchiuse nella **regione proibita** di figura 6-10 (a). Prima di spedire un TPDU su qualsiasi connessione, l'entità di trasporto deve leggere l'orologio e verificare di non essere nella regione proibita.

Il protocollo può presentare due tipologie di problemi. Se un host trasmette troppo velocemente su una connessione appena aperta, la curva dei numeri di sequenza effettivi può crescere più drasticamente della curva dei numeri di sequenza iniziali. Questo significa che la velocità massima di qualsiasi connessione è di un TPDU ogni battito di orologio. Significa inoltre che l'entità di trasporto deve aspettare fino al prossimo battito di orologio prima di aprire una connessione dopo la ripartenza causata da un guasto, in modo che lo stesso numero non venga utilizzato due volte. Entrambi questi argomenti spingono verso un battito di orologio molto breve (pochi millisecondi).

Sfortunatamente, entrare nella regione proibita dal basso trasmettendo troppo velocemente non è l'unico problema. Dalla figura 6-10 (b), dovrebbe essere chiaro che, a qualsiasi velocità di spedizione inferiore alla velocità dell'orologio, la curva dei numeri di sequenza effettivi entrerà prima o poi nella regione da sinistra. Maggiore è la pendenza della curva effettiva dei numeri di sequenza, più tardi questo evento verrà rimandato. Come abbiamo affermato in precedenza, prima di spedire ogni TPDU l'entità di trasporto deve verificare se sta per entrare nella regione proibita, nel qual caso deve ritardare il TPDU per T secondi oppure risincronizzare i numeri di sequenza.

Il metodo basato sugli orologi risolve il problema dei duplicati in ritardo per TPDU contenenti dati, ma è necessario creare una connessione prima che questo metodo sia utilizzabile. Poiché anche i TPDU di controllo possono essere in ritardo, lo stesso problema si può presentare cercando di mettere d'accordo entrambi gli interlocutori sul numero di sequenza iniziale. Supponiamo, ad esempio, che le connessioni siano create facendo in modo che l'host 1 spedisca all'host 2 un TPDU CONNECTION REQUEST contenente il numero di sequenza iniziale proposto e il numero della porta di destinazione. L'host 2 risponde a questa richiesta spedendo un TPDU CONNECTION ACCEPTED. Se il TPDU CONNECTION REQUEST viene perso, mentre un duplicato in ritardo CONNECTION

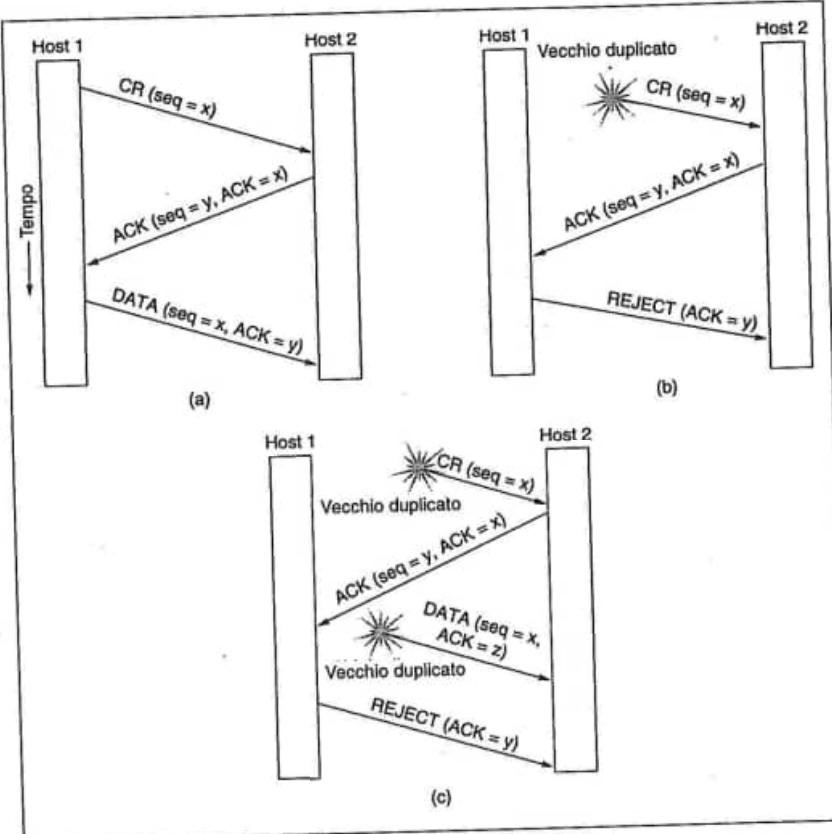


Fig. 6-11 Tre scenari di protocolli di creazione connessioni che utilizzano un protocollo three-way handshake. CR e ACC denotano rispettivamente CONNECTION REQUEST e CONNECTION ACCEPTED. (a) Funzionamento normale. (b) Vecchi duplicati di CONNECTION REQUEST saltano fuori dal nulla. (c) CONNECTION REQUEST duplicati e ACK duplicati.

REQUEST appare all'improvviso presso l'host 2, la connessione viene creata in modo scorretto.

Per risolvere questo problema, Tomlinson (1975) introdusse il protocollo **three-way handshake** (accordo a tre mani). Questo protocollo di creazione non richiede che entrambi i lati comincino a spedire con lo stesso numero di sequenza, e quindi può essere utilizzato con i metodi di sincronizzazione diversi da quello basato su orologio globale. La procedura di creazione normale nel caso l'host 1 sia il chiamante è mostrata in figura 6-11 (a). L'host 1 sceglie un numero di sequenza, x , e invia un TPDU CONNECTION REQUEST che lo contiene all'host 2. L'host 2 replica con un TPDU CONNECTION ACCEPTED confermando x e

6.2 Elementi del protocollo di trasporto

annunciando il suo nuovo numero di sequenza, y . Infine, l'host 1 conferma il numero di sequenza iniziale scelto dall'host 2 nel primo TPDU di dati che spedisce.

Vediamo ora il funzionamento del protocollo three-way handshake in presenza di TPDU di controllo duplicati e in ritardo. In figura 6-11 (b), il primo TPDU è un CONNECTION REQUEST duplicato e in ritardo di una vecchia connessione. Questo TPDU arriva all'host 2 senza che l'host 1 lo sappia. L'host 2 reagisce a questo TPDU spedendo all'host 1 un TPDU CONNECTION ACCEPTED, in effetti chiedendo una conferma che l'host 1 stia ancora tentando di creare una nuova connessione. Quando l'host 1 respinge il tentativo di creazione dell'host 2, quest'ultimo comprende che è stato ingannato da un duplicato in ritardo e abbandona la connessione. In questo modo, un duplicato in ritardo non crea problemi. Il caso peggiore avviene quando in giro per la rete vi sono ancora sia un CONNECTION REQUEST in ritardo, sia un ACK a un CONNECTION ACCEPTED. Questo scenario è mostrato in figura 6-11 (c). Come nell'esempio precedente, l'host 2 riceve un CONNECTION REQUEST in ritardo a cui risponde. A questo punto è importante comprendere che l'host 2 ha proposto di utilizzare y come numero di sequenza iniziale per il traffico host 2-host 1, essendo certo che non esiste più nessun TPDU contenente un numero di sequenza y o un ack per y . Quando il secondo TPDU in ritardo arriva all'host 2, il fatto che sia stato confermato z invece di y fa capire all'host 2 che anche questo è un vecchio duplicato. È importante rendersi conto che non esiste combinazione di CONNECTION REQUEST, CONNECTION ACCEPTED o altri TPDU obsoleti che possono far fallire il protocollo, spingendolo a creare per errore una connessione non desiderata.

Uno schema alternativo per creare connessioni in modo affidabile a fronte di duplicati in ritardo è descritto in Watson (1981) ed è basato su timer multipli per escludere eventi indesiderati.

6.2.3 Chiudere una connessione

Chiudere una connessione è più facile che crearne una. Tuttavia, ci sono più trabocchetti di quanto uno si possa aspettare. Come abbiamo detto in precedenza, ci sono due metodi per chiudere una connessione: la chiusura asimmetrica e quella simmetrica. La chiusura asimmetrica è quella utilizzata nei sistemi telefonici: quando uno degli interlocutori appende la cornetta, la connessione viene terminata. La chiusura simmetrica tratta una connessione come due connessioni separate unidirezionali e richiede che ognuna di esse venga chiusa separatamente.

La disconnessione simmetrica è improvvisa e può portare a perdite di dati. Si consideri lo scenario di figura 6-12. Dopo aver stabilito la connessione, l'host 1 spedisce un TPDU che arriva propriamente all'host 2. Quindi l'host 1 spedisce un altro TPDU. Sfortunatamente, l'host 2 emette un **DISCONNECT** prima dell'arrivo del secondo TPDU. Come risultato, la connessione viene chiusa e i dati vengono persi.

Chiaramente, è necessario un protocollo di chiusura più complicato per evitare perdite di dati. Un metodo consiste nell'utilizzare la chiusura asimmetrica, nella quale ogni direzione è chiusa indipendentemente dall'altra. In questo caso, un host può continuare a spedire dati anche se l'altro ha spedito un TPDU DISCONNECT.

La chiusura asimmetrica funziona quando ogni processo ha una quantità fissa di dati da spedire e chiaramente sa quando li ha spediti tutti. In altre situazioni, determinare che

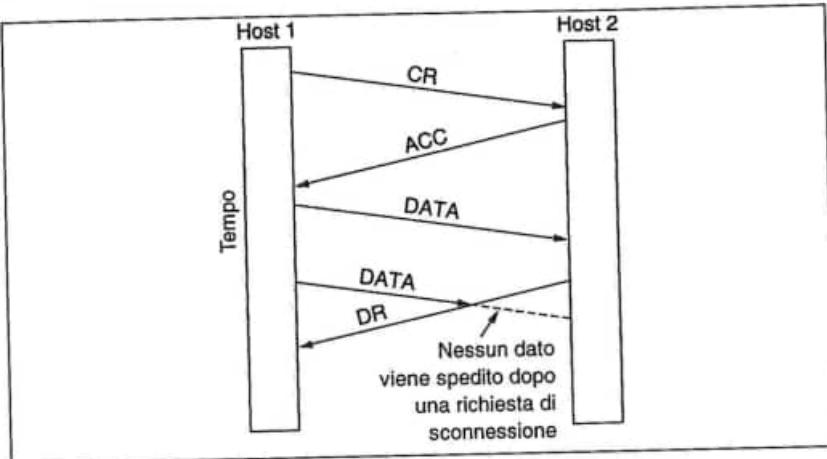


Fig. 6-12 Disconnessione improvvisa con perdita di dati.

tutto il lavoro è stato fatto e che la connessione dovrebbe essere terminata non è così ovvio. Uno si può immaginare un protocollo nel quale l'host 1 dice "Ho finito. Hai finito anche tu?". Se l'host 2 risponde: "Ho finito anch'io. Arrivederci" la connessione può essere chiusa in modo sicuro.

Sfortunatamente, questo protocollo non funziona sempre. Esiste un famoso problema, che si occupa di questa problematica. Viene chiamato **problema dei due eserciti**. Si immagini che un esercito bianco sia accampato in una valle, come mostrato in figura 6-13. Su entrambe le colline attorno ci sono gli eserciti blu. L'esercito bianco è più grande di entrambi gli eserciti blu, ma insieme questi sono più grandi dell'esercito bianco. Se uno dei due eserciti blu attacca da solo, verrà sconfitto, ma se i due eserciti attaccano simultaneamente, ne usciranno vittoriosi.

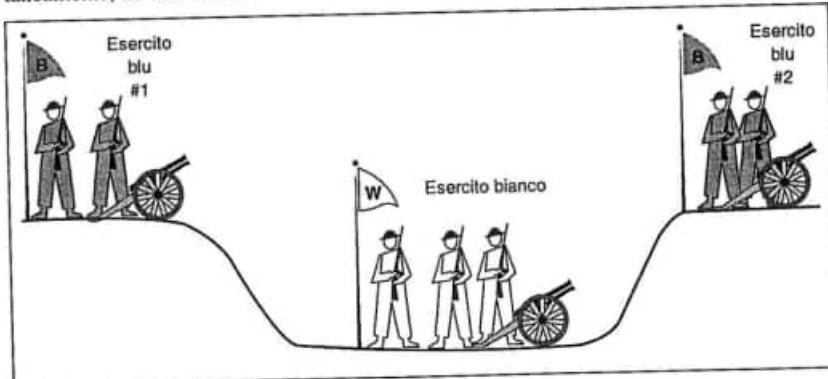


Fig. 6-13 Il problema dei due eserciti.

Gli eserciti blu vogliono sincronizzare i loro attacchi. Tuttavia, l'unico mezzo di comunicazione è spedire messaggeri a piedi nella valle, dove potrebbero essere catturati e il messaggio potrebbe essere perso (cioè devono usare un mezzo di comunicazione inaffidabile). La questione è la seguente: esiste un protocollo che permetta agli eserciti blu di vincere?

Si supponga che il comandante dell'esercito blu #1 spedisca un messaggio che dice "Propongo di attaccare all'alba di marzo 29. Che ne dite?". Si supponga ora che il messaggio arrivi, e il comandante dell'esercito blu #2 sia d'accordo, e che la sua risposta raggiunga senza danno l'esercito blu #1. L'attacco avrà successo? Probabilmente no, in quanto il comandante #2 non sa che la sua risposta è passata, e quindi non rischia, nel dubbio di doversi muovere da solo.

A questo punto, miglioriamo il protocollo trasformandolo in un three-way handshake. L'iniziatore della proposta originale deve confermare la ricezione della risposta. Assumendo che nessun messaggio venga perso, l'esercito blu #2 riceverà la conferma, ma a questo punto è il comandante dell'esercito blu #1 a esitare. Dopo tutto non sa se la sua conferma sia passata, e anche lui non vuole rischiare di attaccare da solo. Potremmo realizzare un protocollo 4-way handshake, ma questo non sarebbe ancora sufficiente.

In effetti, si può provare che non esiste nessun protocollo corretto. Supponiamo per assurdo che tale protocollo esista. O l'ultimo messaggio è essenziale, oppure non lo è. Se non lo è, lo rimuoviamo (e con esso ogni altro messaggio non essenziale) fino a quando rimaniamo con protocollo in cui ogni messaggio è essenziale. Cosa succede se l'ultimo messaggio non passa? Abbiamo appena detto che è essenziale, quindi se viene perso l'attacco non avrà luogo. Poiché il mittente dell'ultimo messaggio non può mai essere sicuro della sua ricezione, non correrà mai il rischio di attaccare da solo. Peggio ancora, l'altro esercito blu lo sa, e quindi non attaccherà in ogni caso.

Per capire la rilevanza del problema dei due eserciti nella chiusura delle connessioni, si sostituisca "attacco" con "disconnessione". Se nessuno degli interlocutori è pronto a disconnettersi fino a quando non è convinto che anche l'altro è pronto, la disconnessione non avverrà mai.

In pratica, in genere si è più disposti ad accettare dei rischi quando si chiudono le connessioni piuttosto che quando si attaccano gli eserciti bianchi, e quindi la situazione non è completamente senza speranza. La figura 6-14 illustra quattro scenari di chiusura che utilizzano un protocollo three-way handshake. Sebbene questo protocollo non sia infallibile, è in genere sufficientemente preciso.

In figura 6-14 (a) vediamo il caso normale in cui uno degli utenti spedisce un TPDU DR (DISCONNECTION REQUEST) per iniziare la chiusura della connessione. Quando questo arriva, anche il ricevente spedisce indietro un TPDU DR e fa partire un timer, nel caso che il suo DR venga perso. Quando questo DR arriva, il mittente originale risponde con un TPDU ACK e chiude la connessione. Infine, quando il TPDU ACK arriva, anche il ricevente chiude la connessione. Chiudere la connessione significa che l'entità di trasporto elimina le informazioni sulla connessione dalla sua tabella delle connessioni aperte e manda in qualche modo un segnale al suo possessore (l'utente del trasporto). Questa azione è differente da un utente del trasporto che utilizza una primitiva DISCONNECT.

Se il TPDU ACK finale viene perso, come mostrato in figura 6-14 (b), la situazione è

salvata dal timer. Quando questo timer finisce, la connessione viene rilasciata in ogni caso.

Consideriamo ora il caso in cui si perda il secondo DR. L'utente che inizia la disconnessione non riceverà la risposta attesa, andrà in timeout e ripartirà da capo. Questo scenario è mostrato in figura 6-14 (c), assumendo che la seconda volta non venga perso nessuno dei TPDU, che vengono invece consegnati tutti correttamente e puntualmente.

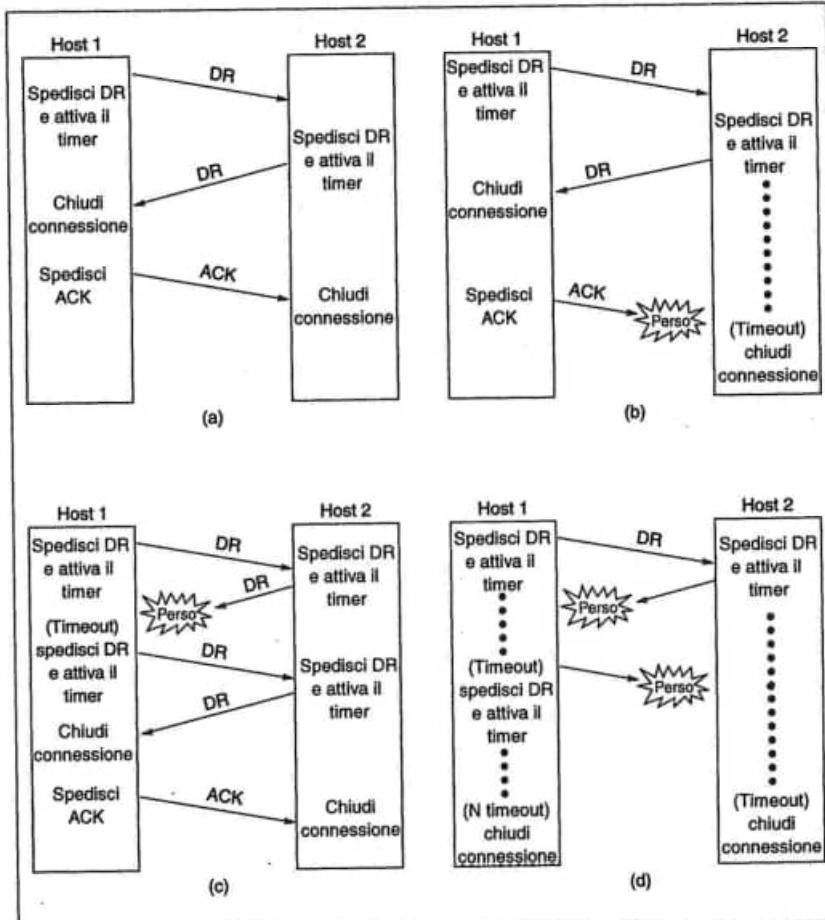


Fig. 6-14 Quattro scenari del protocollo di chiusura di una connessione. (a) Three-way handshake normale. (b) Perdita dell'ACK finale. (c) Perdita della risposta. (d) Perdita della risposta e dei RR successivi.

Il nostro ultimo scenario (figura 6-14 (d)) è simile a figura 6-14 (c), a parte il fatto che ora assumiamo che tutti i tentativi ripetuti di ritrasmettere il DR falliscano a causa di TPDU persi. Dopo N tentativi, il mittente smette e chiude la connessione. Nel frattempo, la connessione viene chiusa anche dal ricevente in seguito a un timeout.

Sebbene questo protocollo sia in generale sufficiente, in teoria potrebbe fallire se il DR iniziale e le N ritrasmissioni venissero persi. Il mittente chiude la connessione, mentre l'altro lato non sa nulla dei tentativi di disconnessione ed è ancora completamente attivo. Questa situazione produce una connessione aperta a metà.

Avremmo potuto evitare questo problema non permettendo al mittente di smettere dopo N prove, ma forzandolo ad andare avanti fino a quando non avesse ricevuto una risposta. Tuttavia, se l'altro lato potesse andare in timeout, allora il mittente andrebbe avanti per sempre, in quanto non arriverebbe mai nessuna risposta. Se non consentiremo al lato ricevente di andare in timeout, allora il protocollo si bloccherà nello scenario di figura 6-14 (b). Per eliminare le connessioni lasciate aperte a metà, si può realizzare una regola che forzi la chiusura automatica di una connessione nel caso in cui non arrivino nuovi TPDU per un certo numero di secondi. In questo modo, se un lato si disconnettesse, l'altro lato si accorgerebbe della mancanza di attività e chiuderebbe anch'esso la connessione. Naturalmente, se venisse introdotta questa regola, sarebbe necessario che ogni entità di trasporto avesse un timer che venisse fermato e fatto ripartire ad ogni spedizione di TPDU. Se questo timer scadesse, verrebbe trasmesso un TPDU nullo, solo per evitare la disconnessione dell'altro lato. D'altra parte, se viene usata la regola di disconnessione automatica e vengono persi troppi TPDU nulli consecutivi in una connessione altrimenti inattiva, prima un lato, poi l'altro si disconnetteranno automaticamente.

Non insisteremo ulteriormente su questo punto, ma dovrebbe essere chiaro che chiudere una concessione non è così semplice come sembra a prima vista.

6.2.4 Controllo di flusso e gestione dei buffer

Dopo aver esaminato dettagliatamente la creazione e la chiusura delle connessioni, analizziamo a questo punto la gestione delle connessioni durante la loro utilizzazione. Uno dei problemi principali si è già presentato in precedenza: il controllo di flusso. Da un certo punto di vista, il problema del controllo di flusso nel livello trasporto è uguale a quello del livello data link, sebbene esistano alcune differenze. Per quanto riguarda le somiglianze, in entrambi gli strati è necessario un protocollo di sliding window o qualche altro schema per ognuna delle connessioni, al fine di evitare che un trasmettitore veloce sovraccarichi un ricevente lento. La differenza principale riguarda il numero generalmente basso di connessioni possedute da un router, in confronto al gran numero di connessioni utilizzabili da un singolo host. Questa differenza rende poco pratico realizzare nel livello trasporto la strategia di gestione dei buffer del livello data link.

Nel protocollo data link del capitolo 3, i frame venivano memorizzati sia nel router mittente che in quello ricevente. Nel protocollo 6, sia il mittente che il ricevente devono dedicare $\text{MaxSeq}+1$ buffer ad ogni linea, metà per l'input e metà per l'output. Per un host con massimo di 64 connessioni e un numero di sequenza di 4 bit, questo protocollo richiederebbe 1024 buffer.

Nel livello data link, il lato mittente deve memorizzare i frame in uscita perché potrebbe

essere necessario ritrasmetterli. Se la rete fornisce un servizio datagram, anche l'entità di trasporto mittente deve utilizzare dei buffer, e per gli stessi motivi. Se il ricevente sa che il mittente memorizza tutti i TPDU fino a quando non ne viene confermata la ricezione, può dedicare (o non dedicare) buffer specifici per connessioni specifiche, come gli sembra opportuno. Il ricevente può, ad esempio, mantenere un unico insieme di buffer condiviso da tutte le connessioni. All'arrivo di un TPDU, viene fatto un tentativo di acquisire dinamicamente un nuovo buffer. Se ne esiste uno disponibile, il TPDU viene accettato; altrimenti, viene scartato. Poiché il mittente è preparato a ritrasmettere TPDU persi dalla rete, non vi è alcun problema se il ricevente scarta qualche TPDU, a parte lo spreco di risorse. Il mittente continua a provare fino a quando non riceve una conferma.

Per riassumere, se il servizio di rete è inaffidabile, il mittente deve memorizzare in buffer tutti i TPDU spediti, proprio come nel livello data link. Tuttavia, con servizi di rete affidabili, diventano possibili altri compromessi. In particolare, se il mittente sa che il ricevente ha sempre spazio disponibile, non deve trattenere delle copie dei TPDU che spedisce. Al contrario, se il ricevente non può garantire che ogni TPDU in arrivo verrà accettato, il mittente deve comunque utilizzare dei buffer. Nel secondo caso, il mittente non può basarsi sull'ack del livello rete, in quanto l'ack significa solo che il TPDU è arrivato, non che è stato accettato. Torneremo a discutere di questo importante punto nel seguito.

Anche se il ricevente ha concordato di utilizzare dei buffer, rimane ancora la questione della loro dimensione. Se gran parte dei TPDU sono all'incirca della stessa dimensione, è naturale organizzare il buffer come un insieme di buffer identici, con un TPDU per ogni buffer, come in figura 6-15 (a). Al contrario, se esiste una grande variabilità nella dimensione dei TPDU, da pochi caratteri battuti a un terminale fino a migliaia di caratteri nel caso di un trasferimento file, un insieme di buffer di dimensione fissa non è soddisfacente. Se la dimensione del buffer venisse scelta uguale a quella del TPDU più grande, verrebbe sprecato dello spazio tutte le volte che arriva un TPDU breve. Se la dimensione del buffer scelta fosse più piccola del TPDU di dimensione massima, sarebbero necessari buffer multipli per TPDU lunghi, con la conseguente complessità aggiuntiva.

Un altro approccio al problema della dimensione del buffer consiste nell'utilizzare buffer di dimensione variabile, come in figura 6-15 (b). Il vantaggio in questo caso è una migliore utilizzazione della memoria, al prezzo di una più complessa gestione dei buffer. Una terza possibilità è dedicare ad ogni connessione un singolo buffer circolare di grandi dimensioni, come in figura 6-15 (c). Questo sistema utilizza efficientemente la memoria se tutte le connessioni sono pesantemente utilizzate, mentre è inefficiente se qualche connessione è poco utilizzata.

Il compromesso ottimale fra utilizzazione dei buffer alla sorgente o alla destinazione dipende dal tipo di traffico trasportato dalla connessione. Per traffico a banda stretta caratterizzato da picchi, come quello prodotto dai terminali interattivi, è preferibile non dedicare alcun buffer, ma piuttosto acquisirli dinamicamente in entrambi i lati. Poiché il mittente non può essere sicuro che il ricevente sia in grado di acquisire un buffer, deve mantenere una copia del TPDU fino a quando la ricezione non è confermata. Al contrario, per trasferimenti di file e altro traffico a banda larga, è preferibile che il ricevente dedichi un intero insieme di buffer, per consentire ai dati di arrivare alla velocità massima. Quindi

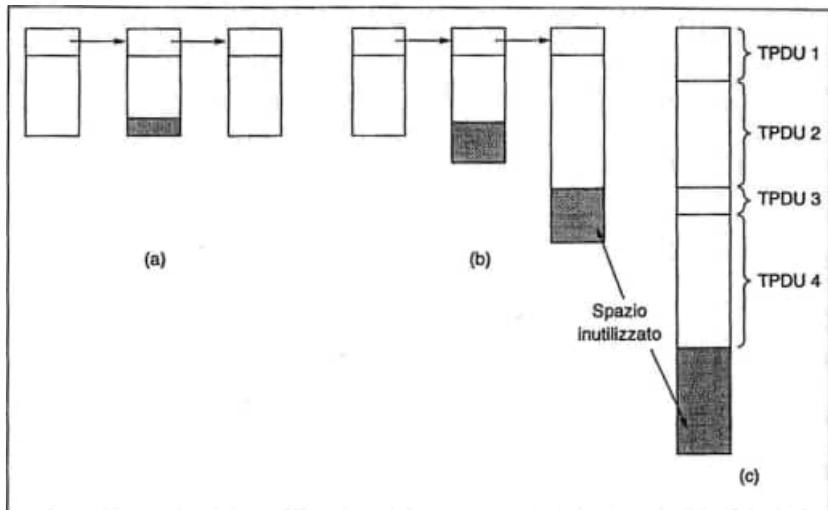


Fig. 6-15 (a) Buffer di dimensione fissa concatenati. (b) Buffer di dimensioni variabili concatenati. (c) Un buffer circolare di grandi dimensioni per ogni connessione.

per il traffico a banda stretta e caratterizzato da picchi è preferibile utilizzare i buffer presso il mittente, mentre per il traffico a banda larga e omogeneo è preferibile utilizzare i buffer presso il ricevente.

Quando le connessioni vengono aperte e chiuse, o il traffico subisce un cambiamento, il mittente e il ricevente devono modificare dinamicamente l'allocazione dei loro buffer. Conseguentemente, il protocollo di trasporto dovrebbe permettere all'host mittente di richiedere spazio di buffer all'altro capo. I buffer potrebbero essere allocati per ogni connessione, oppure collettivamente per tutte le connessioni tra i due host. Come alternativa, il ricevente, conoscendo la propria situazione di buffer (ma non conoscendo il traffico offerto) potrebbe comunicare al mittente "Ho riservato X buffer per te". Se il numero di connessioni aperte dovesse crescere, potrebbe essere necessario ridurre l'allocatione, e quindi il protocollo deve tenere conto di questa possibilità.

Un metodo ragionevolmente generico per gestire l'allocazione dinamica dei buffer consiste nel separare la gestione dei buffer da quella degli ack, a differenza del protocollo sliding window del capitolo 3. Gestione dinamica dei buffer significa, in effetti, finestre di dimensioni variabili. Inizialmente, il mittente richiede un certo numero di buffer, in base ai propri bisogni presunti. Il ricevente assegna tutti i buffer di cui può disporre. Ogni volta che il mittente spedisce un TPDU, deve diminuire la propria allocatione, smettendo completamente quando quest'ultima raggiunge lo 0. Il ricevente inserisce separatamente nel traffico di ritorno sia gli ack che l'allocazione dei buffer.

La figura 6-16 mostra un esempio del funzionamento della gestione dinamica delle finestre in una rete basata su datagram con numeri di sequenza di 4 bit. Si assuma che le

informazioni sull'allocazione dei buffer viaggino su TPDU separati, come mostrato, e non siano inseriti nel traffico di ritorno. All'inizio, A desidera 8 buffer, ma ne riceve solo 4. Spedisce quindi 3 TPDU, il terzo dei quali viene perso. Il TPDU 6 conferma la ricezione di tutti i TPDU fino al numero 1 incluso, consentendo quindi ad A di rilasciare quei buffer, e inoltre accorda ad A il permesso di spedire altri 3 TPDU successivi al numero 1 (cioè i TPDU 2, 3, 4). A sa di aver già spedito il TPDU 2, e quindi spedisce i TPDU 3 e 4. A questo punto si blocca e deve attendere ulteriori allocazioni di buffer. Tuttavia, durante l'attesa lo scadere di un timeout può indurre a una ritrasmissione (linea 9), in quanto sono stati utilizzati buffer già allocati. In linea 10, B conferma la ricezione di tutti i TPDU fino al 4 incluso, ma non consente ad A di continuare. Questa situazione non si può presentare nel caso del protocollo a finestra fissa del capitolo 3.

Il prossimo TPDU da B ad A alloca un ulteriore buffer e consente ad A di continuare.

A	Messaggio	B	Commenti
1	→ <request 8 buffers>	→	A vuole 8 buffer
2	→ <ack = 15, buf = 4>	→	B consente solo i messaggi 0-3
3	→ <seq = 0, data = m0>	→	A rimane con 3 buffer
4	→ <seq = 1, data = m1>	→	A rimane con 2 buffer
5	→ <seq = 2, data = m2>	...	Messaggio perso, ma A pensa di rimanere con 1
6	→ <ack = 1, buf = 3>	→	B conferma 0, 1, permette 2-4
7	→ <seq = 3, data = m3>	→	A rimane con 1 buffer
8	→ <seq = 4, data = m4>	→	A finisce i buffer, deve bloccarsi
9	→ <seq = 2, data = m2>	→	A va in timeout e ritrasmette
10	→ <ack = 4, buf = 0>	→	Tutto confermato, ma A è ancora bloccato
11	→ <ack = 4, buf = 1>	→	A può spedire 5
12	→ <ack = 4, buf = 2>	→	B trova un nuovo buffer da qualche parte
13	→ <seq = 5, data = m5>	→	A rimane con 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>	Deadlock potenziale

Fig. 6-16 Allocazione dinamica dei buffer. Le frecce mostrano la direzione di trasmissione. I tre punti (...) indicano un TPDU perso.

Queste tipologie di schema di allocazione dei buffer danno origine ad alcuni problemi nel caso di reti basate su datagram in cui i TPDU di controllo possano essere persi. Si osservi la linea 16. B ha allocato un numero di buffer maggiore di A, ma il TPDU di allocazione è stato perso. Poiché non viene utilizzato nessun timeout per i TPDU di controllo, a questo punto A è in stallo. Per evitare questa situazione, ogni host deve spedire periodicamente TPDU di controllo contenenti gli ack e lo stato dei buffer per ognuna delle connessioni. In questo modo lo stallo viene prima o poi interrotto.

Fino a questo punto abbiamo tacitamente assunto che l'unico limite imposto alla velocità di trasmissione sia il numero di buffer di cui dispone il ricevente. Con il calo

6.2 Elementi del protocollo di trasporto

drammatico del prezzo dei chip di memoria, potrebbe diventare ragionevole equipaggiare gli host con una quantità di memoria tale che la mancanza di buffer sia raramente un problema.

Quando la quantità di buffer non limiterà più il flusso massimo, il collo di bottiglia sarà un altro: la capacità di trasporto della rete. Se router adiacenti possono scambiare al massimo x frame/s ed esistono k cammini disgiunti tra una coppia di host, è impossibile che gli host possano scambiarsi più di kx TPDU/s, indipendentemente dalla quantità di buffer ad ogni estremo. Se il mittente spedisce troppo in fretta (cioè spedisce più di kx TPDU/s), la rete presenterà una congestione, in quanto non è in grado di consegnare i TPDU alla stessa velocità a cui arrivano.

È quindi necessario un meccanismo basato sulla capacità di trasporto delle reti piuttosto che sul numero di buffer del ricevente. Chiaramente, il meccanismo di controllo di flusso deve essere applicato al mittente per evitare che possieda un numero eccessivo di TPDU non confermati. Belsnes (1975) propose di utilizzare uno schema di controllo di flusso sliding window nel quale il mittente modifica dinamicamente la dimensione della finestra per adattarsi alla capacità di trasporto della rete. Se la rete può gestire c TPDU/s, e il tempo di andata e ritorno (che include i tempi di trasmissione, di propagazione, di coda, di elaborazione da parte del ricevente e di ritorno degli ack) è r , allora la finestra del mittente dovrebbe essere cr . Con una finestra di questo tipo il mittente lavora al limite della capacità. Ogni piccolo calo nelle prestazioni della rete può portare a un blocco. Per adattare periodicamente la dimensione della finestra, il mittente potrebbe monitorare entrambi i parametri e calcolare quindi la dimensione desiderata. La capacità di trasporto potrebbe essere misurata contando il numero di TPDU confermati durante un certo periodo di tempo e quindi dividendo per la durata del periodo. Durante la misurazione, il mittente dovrebbe spedire alla massima velocità, per accertarsi che il fattore che limita la velocità di arrivo degli ack sia la capacità trasmisiva della rete e non una bassa velocità di spedizione. Il tempo richiesto affinché un TPDU trasmesso sia confermato può essere misurato esattamente, e contemporaneamente può essere calcolata la media. Poiché la capacità della rete dipende dalla quantità di traffico presente, la dimensione della finestra dovrebbe essere adattata frequentemente, in modo da tener conto dei cambiamenti nella capacità di trasporto. Come vedremo in seguito, Internet utilizza uno schema simile.

6.2.5 Multiplexing

La sovrapposizione di più di una conversazione su connessioni, circuiti virtuali e canali fisici assume un ruolo importante in molti strati dell'architettura di rete. Nel livello trasporto la necessità di sovrapposizioni può presentarsi in molte occasioni. Ad esempio, nelle reti internamente basate su circuiti virtuali, ogni connessione aperta consuma spazio nelle tabelle dei router per l'intera durata della connessione. Se in ogni router venisse dedicato ai circuiti virtuali anche un certo numero di buffer, un utente che lascia un terminale aperto in una macchina remota durante la pausa del caffè consumerebbe risorse preziose. Sebbene questa realizzazione della commutazione di pacchetto vanifichi uno dei motivi principali che spingono alla sua utilizzazione (far pagare l'utente in base alla quantità di dati spediti e non al tempo di connessione), molti fornitori hanno scelto questo

approccio in quanto assomiglia molto al modello della commutazione di circuito a cui sono abituati da decenni.

Come conseguenza di una struttura dei prezzi che penalizza le installazioni che mantengono aperti un gran numero di circuiti per lunghi periodi di tempo, la sovrapposizione di differenti connessioni del trasporto sullo stesso circuito virtuale (multiplexing) diviene alllettante. Questa forma di multiplexing, detta **upward multiplexing** (multiplexing verso l'alto), è illustrata in figura 6-17 (a). In questa figura, quattro distinte connessioni di trasporto utilizzano la stessa connessione di rete (ad es. circuiti virtuali ATM) con un host remoto. Quando il tempo di connessione costituisce una componente principale della bolletta del fornitore, è compito del livello trasporto raggruppare connessioni di trasporto in base alla loro destinazione e mappare ogni gruppo su un numero minimo di connessioni di rete. Nel caso in cui troppe connessioni di trasporto venissero mappate su una singola connessione, le prestazioni sarebbero misere, in quanto la finestra sarebbe generalmente piena e l'utente dovrebbe attendere il proprio turno per spedire un messaggio. Nel caso in cui venisse mappato un numero troppo basso di connessioni di trasporto su una singola connessione di rete, il servizio sarebbe costoso. Quando il multiplexing verso l'alto viene utilizzato con ATM, siamo nell'ironica (tragica?) situazione di dover identificare la connessione con un campo del preambolo di trasporto, sebbene ATM fornisca più di 4000 identificatori di circuiti virtuali per ogni percorso virtuale, espressamente per questo motivo.

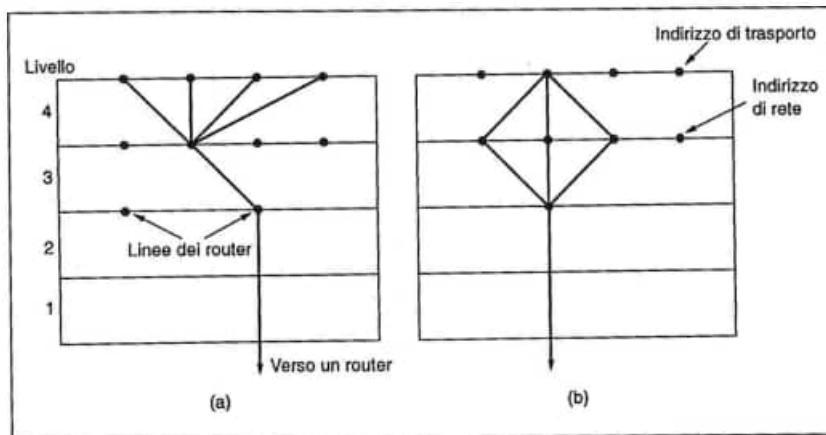


Fig. 6-17 (a) Multiplexing verso l'alto. (b) Multiplexing verso il basso.

Nel livello trasporto, la tecnica del multiplexing può essere utile anche per un altro motivo, collegato a considerazioni tecniche del fornitore, piuttosto che a decisioni di tariffamento. Si supponga, ad esempio, che qualche utente importante abbia bisogno di tanto in tanto di una connessione ad alta capacità. Se la rete realizza un controllo di flusso sliding window con un numero di sequenza a n bit, l'utente deve smettere di spedire non

appena vi sono $2^n - 1$ pacchetti in sospeso e deve aspettare che i pacchetti si propaghino fino all'host remoto e vengano confermati. Se la connessione fisica attraversa un satellite, l'utente è limitato effettivamente a $2^n - 1$ pacchetti ogni 540 m. Per esempio, con $n = 8$ e pacchetti di 128 byte, la banda utilizzabile è di 484 kbps, anche nel caso in cui la banda del canale fisico sia 100 volte più grande.

Una soluzione possibile richiede che il livello trasporto apra diverse connessioni di rete e distribuisca il traffico sopra di esse con metodo round-robin, come indicato in figura 6-17 (b). Questa modalità viene chiamata **downward multiplexing** (multiplexing verso il basso). Con k connessioni aperte, la banda effettiva cresce di un fattore k . Con 4095 circuiti virtuali, pacchetti di 128 byte e numeri di sequenza di 8 bit, è teoricamente possibile ottenere una velocità di trasmissione di oltre 1,6 Gbps. Naturalmente, queste prestazioni possono essere ottenute solo se la linea di output è in grado di supportare 1,6 Gbps, in quanto tutti i 4095 circuiti virtuali continuano ad essere trasportati da una singola linea fisica, almeno in figura 6-17 (b). Se sono disponibili più linee di output, il multiplexing verso il basso può essere utilizzato per incrementare ancora di più la banda.

6.2.6 Ripristino dai guasti

Se gli host e i router sono soggetti a guasti, abbiamo il problema del ripristino. Se l'entità di trasporto è completamente contenuta negli host, il ripristino in seguito a guasti della rete o dei router è banale. Se il livello rete fornisce un servizio datagram, l'entità di trasporto si attende perdite di TPDU in qualunque istante, e sa come affrontare il problema. Se il livello rete fornisce un servizio orientato alla connessione, la perdita di un circuito virtuale viene gestita creandone uno nuovo e quindi chiedendo all'entità di trasporto remota quali TPDU ha ricevuto; i TPDU non ricevuti possono essere ritrasmessi. Un problema più fastidioso riguarda il ripristino di host guasti. In particolare, i clienti possono desiderare di continuare a lavorare nel caso in cui i server si guastassero e ripartissero immediatamente. Per illustrare queste difficoltà, assumiamo che un host, il cliente, stia spedendo un file molto lungo a un file server, utilizzando un semplice protocollo stop-and-wait. Il livello trasporto del server non fa altro che passare i TPDU in entrata all'utente del trasporto, uno per uno. Durante la trasmissione, il server si guasta. Quando torna a funzionare, le sue tabelle vengono reinizializzate e quindi non ricorda più dove si trovasse precisamente.

Nel tentativo di ricostruire il suo stato precedente, il server potrebbe spedire un TPDU di broadcast a tutti gli altri host, annunciando il fatto che si è appena guastato e richiedendo ai suoi clienti di informarlo sullo stato delle connessioni aperte. Ogni cliente può essere in due stati: un TPDU in sospeso, *SJ*, oppure nessun TPDU in sospeso, *SO*. In base a questa semplice informazione di stato, il cliente deve decidere se ritrasmettere oppure no il TPDU più recente.

A prima vista la soluzione sembrerebbe ovvia: il cliente dovrebbe ritrasmettere solo se possiede un TPDU non confermato in sospeso (cioè è nello stato *SJ*) al momento di sapere del guasto. Tuttavia, un'ispezione più attenta rivela alcuni problemi di questo ingenuo approccio. Ad esempio, si consideri la situazione in cui l'entità di trasporto del server prima spedisce una conferma, e poi esegue la scrittura nel processo applicativo. La scrittura di un TPDU nel flusso di output e la spedizione di una conferma sono due eventi

indivisibili che non possono essere eseguiti simultaneamente. Se avviene un guasto dopo la spedizione della conferma, ma prima che la scrittura sia stata effettuata, il cliente riceverà un ack e quindi si troverà nello stato *S0* quando arriverà l'annuncio del guasto. Il cliente eviterà quindi di ritrasmettere, pensando (correttamente) che il TPDU sia già arrivato. La decisione presa dal cliente genera un TPDU mancante.

A questo punto potreste pensare: "Questo problema può essere risolto facilmente. Tutto quello che bisogna fare è riprogrammare l'entità di trasporto in modo che prima venga eseguita la scrittura e poi venga spedita la conferma". Proviamo ancora. Immaginiamo che la scrittura sia stata eseguita, ma che il guasto avvenga prima che la conferma possa essere spedita. Il cliente si troverà nello stato *S1* e quindi ritrasmetterà, causando un TPDU duplicato non rilevato nel flusso in uscita verso l'applicazione del server.

Indipendentemente da come il mittente o il ricevente siano programmati, esistono sempre delle situazioni in cui il protocollo non esegue un ripristino corretto. Il server può essere programmato in due modi: eseguendo prima la conferma oppure prima la scrittura. Il cliente può essere programmato in quattro modi: ritrasmettendo sempre l'ultimo TPDU, non ritrasmettendo mai l'ultimo TPDU, ritrasmettendo solo nello stato *S0*, o ritrasmettendo solo nello stato *S1*. Questo porta a otto combinazioni, ma come vedremo, per ogni combinazione esistono alcuni insiemi di eventi che fanno fallire il protocollo.

Presso il server sono possibili tre eventi: spedizione di una conferma (*A*), scrittura verso il processo (*W*) e guasto (*C*). I tre eventi possono avvenire in sei ordinamenti diversi: *AC* (*W*), *AWC*, *C* (*AW*), *C* (*WA*), *WAC*, e *WC* (*A*), dove le parentesi vengono utilizzate per indicare che né *A* né *W* possono seguire *C* (cioè, una volta che si è guastato non esegue più azioni). La figura 6-18 mostra tutte e otto le combinazioni delle strategie del cliente e del server e le sequenze valide di eventi per ognuna di esse. Si noti che per ogni strategia esiste qualche sequenza di eventi che causa il fallimento del processo. Ad esempio, se il cliente ritrasmetterà sempre, la sequenza di eventi *AWC* genererà un duplicato non rilevato, nonostante le altre sequenze di eventi funzionino correttamente.

		Strategia utilizzata dall'host ricevente					
		Prima ACK, poi scrivi			Prima scrivi, poi ACK		
		AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Ritrasmetti sempre	OK	DUP	OK		OK	DUP	DUP
Non ritrasmette 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

Fig. 6-18 Combinazioni differenti di strategie del cliente e del server.

Rendere il protocollo più elaborato non serve a nulla. Anche nel caso il cliente e il server si scambino diversi TPDU prima che il server tenti di scrivere, in modo che il cliente conosca esattamente cosa sta per succedere, il cliente non ha modo di sapere se il guasto è avvenuto un attimo prima o un attimo dopo la scrittura. La conclusione è inevitabile: sotto la nostra condizione fondamentale di non poter eseguire eventi simultanei, il guasto degli host e il successivo ripristino non possono essere resi trasparenti ai livelli più alti. Messo in termini più generali, questo risultato può essere riformulato dicendo che il guasto di uno strato *N* può essere eseguito solo dal livello *N+1*, e quindi solo se il livello più alto mantiene sufficienti informazioni. Come è stato detto in precedenza, il livello trasporto può funzionare correttamente nel caso di guasti nel livello rete, a condizione che entrambi i lati della connessione tengano traccia di dove sono.

Questo problema ci porta a considerare cosa significhi in realtà riscontro (ack) punto a punto. In teoria, il livello trasporto è di tipo punto-a-punto, non concatenato come i livelli sottostanti. Si consideri il caso di un utente che effettua una richiesta di aggiornamento nei confronti di un database remoto. Si supponga che l'entità di trasporto remota sia programmata in modo che prima passi il TPDU al livello superiore e poi spedisca l'ack. Anche in questo caso, la ricezione dell'ack da parte della macchina dell'utente non significa che l'host remoto sia rimasto attivo un tempo sufficiente per aggiornare effettivamente il database. Un ack davvero punto-a-punto, la cui ricezione implica che il lavoro è stato effettivamente portato a termine e la cui mancanza implica il contrario, è probabilmente impossibile da ottenere. Questo aspetto viene discusso più dettagliatamente da Saltzer et al. (1984).

6.3 Un semplice protocollo di trasporto

Per concretizzare le idee discusse fino ad ora, in questo paragrafo analizzeremo dettagliatamente un livello trasporto di esempio.

6.3.1 Le primitive del servizio di esempio

Il nostro primo problema è come esprimere in pratica queste primitive di trasporto. CONNECT non è complessa: scriveremo una procedura di libreria *connect* (connetti) che può essere chiamata con gli appropriati parametri necessari per stabilire una connessione. I parametri sono il TSAP locale e quello remoto. Durante la chiamata, il chiamante viene bloccato (sospeso) mentre l'entità di trasporto cerca di stabilire la connessione. Se quest'ultima ha successo, il chiamante viene sbloccato e può iniziare a trasmettere dati.

Quando un processo desidera accettare chiamate in arrivo, chiama la procedura *listen* (ascolta), specificando un TSAP particolare da cui stare in ascolto. Il processo quindi si blocca finché un processo remoto non tenta di stabilire una connessione con il suo TSAP. Si noti che questo modello è estremamente asimmetrico. Un interlocutore è passivo: esegue *listen* e resta in attesa fino a quando non succede qualcosa; l'altro è attivo e dà inizio alla connessione. Una questione interessante è la seguente: cosa si deve fare se il lato attivo inizia per primo? Una strategia consiste nel rifiutare il tentativo di connessione nel caso in cui presso il TSAP remoto non esista nessuno in ascolto. Un'altra strategia consiste nel bloccare il chiamante (al limite per sempre) fino a quando non si presenta un processo in ascolto.

Il compromesso che verrà utilizzato nel nostro esempio è mantenere valida per un certo intervallo di tempo la richiesta di connessione presso il ricevente. Se un processo dell'host chiamato esegue *listen* prima che il tempo termini, la connessione viene stabilita; altrimenti, viene rifiutata e il chiamante viene sbloccato e informato tramite un messaggio di errore. Per chiudere una connessione, utilizzeremo una procedura *disconnect* (*disconnetti*). Quando entrambi gli estremi si sono scollegati, la connessione viene chiusa. In altre parole, stiamo utilizzando un modello simmetrico di disconnessione.

La trasmissione di dati presenta lo stesso problema relativo alla creazione delle connessioni: la spedizione è attiva, mentre la ricezione è passiva. Utilizzeremo anche per la trasmissione dati la soluzione utilizzata per la creazione delle connessioni: una chiamata attiva *send* (*spedisci*) per trasmettere dati, un chiamata passiva *receive* (*ricevi*) che blocca il chiamante fino all'arrivo di un TPDU.

La definizione del nostro 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 stessa. I parametri per le primitive di servizio e per le procedure di libreria sono i seguenti:

```
connum = LISTEN(local)
connum = CONNECT(local, remote)
status = SEND(connum, buffer, nbytes)
status = RECEIVE(connum, buffer, nbytes)
status = DISCONNECT(connum)
```

La primitiva LISTEN annuncia la volontà del chiamante di accettare richieste di connessione dirette al TSAP indicato. L'utente della primitiva rimane bloccato fino a quando non viene fatto un tentativo di connessione con esso. Non esiste tempo massimo.

La primitiva CONNECT richiede due parametri, un TSAP locale denominato *local* (cioè un indirizzo di trasporto) e un TSAP remoto denominato *remote*, e cerca di stabilire un connessione di trasporto fra i due. In caso di successo, restituisce un numero non negativo *connum* utilizzato per identificare la connessione nelle chiamate successive. Se la chiamata non ha successo, la ragione del fallimento viene inserita in *connum* come numero negativo. Nel nostro semplice modello, ogni TSAP può partecipare al massimo a una connessione di trasporto; quindi una possibile causa di fallimento potrebbe essere il fatto che uno degli indirizzi di trasporto sia già in uso. Fra le altre possibili cause ricordiamo un guasto nell'host remoto oppure la scorretta definizione dell'indirizzo locale o di quello remoto.

La primitiva SEND trasmette il contenuto del buffer come un messaggio spedito sulla connessione di trasporto indicata, al limite diviso in più parti se è troppo grande. Fra i possibili errori, restituiti in *status*, ricordiamo l'assenza di connessione, un indirizzo di buffer illegale oppure una lunghezza negativa del messaggio.

La primitiva RECEIVE indica che il chiamante desidera ricevere dati. La dimensione del messaggio in arrivo viene copiata in *nbytes*. Se il processo remoto ha chiuso la connessione oppure l'indirizzo del buffer è illegale (ad es. fuori dall'area utente), un codice di errore indicante la natura del problema viene copiato in *status*.

La primitiva DISCONNECT termina una connessione di trasporto. Il parametro *connum*

indica quale connessione venga chiusa. Se *connum* appartiene a un altro processo o non è un identificatore valido, un codice di errore viene restituito in *status*; altrimenti, viene restituito 0 per indicare il successo.

6.3.2 L'entità di trasporto di esempio

Prima di esaminare il codice relativo all'entità di trasporto di esempio, è importante accertarsi di aver capito che questo esempio è analogo a quelli presentati in precedenza nel capitolo 3: più che essere una proposta seria, il codice proposto ha fini pedagogici. Per motivi di semplicità, sono stati omessi molti dei dettagli tecnici necessari in un sistema commerciale (come ad esempio un più esteso controllo degli errori).

Il livello trasporto utilizza le primitive del livello rete per spedire e ricevere TPDU. Per questo esempio, dobbiamo scegliere quali primitive di rete utilizzare. Una scelta potrebbe essere un servizio inaffidabile basato su datagram. Per non complicare inutilmente l'esempio, abbiamo deciso diversamente. Con un servizio inaffidabile basato su datagram, il codice del livello trasporto sarebbe molto grande e complesso, in gran parte dedicato ai pacchetti persi o in ritardo. Inoltre, molte di queste idee sono già state discusse a lungo nel capitolo 3.

Al suo posto abbiamo scelto di utilizzare un servizio di rete affidabile e orientato alla connessione. In questo modo possiamo concentrarci sulle problematiche del trasporto che non si presentano negli strati inferiori. Fra queste ricordiamo la creazione e la chiusura delle connessioni e la tariffazione. Un semplice protocollo di trasporto realizzato sopra ATM potrebbe assomigliare molto a quello da noi proposto.

In generale, l'entità di trasporto può far parte del sistema operativo dell'host oppure può essere una libreria di procedure eseguite nello spazio di indirizzamento dell'utente. Potrebbe anche essere contenuta in un chip coprocessore o in una scheda di rete inserita all'interno di un host. Per semplicità, il nostro esempio è stato programmato come se fosse una libreria di procedure; i cambiamenti necessari per trasformarlo in una parte del sistema operativo sono minimi (ad es. nel sistema di accesso ai buffer).

Vale la pena notare, tuttavia, che in questo esempio l'"entità di trasporto" non è veramente un'entità separata, ma un parte del processo utente. In particolare, quando l'utente esegue una primitiva bloccante (come LISTEN), si blocca anche l'intera entità di trasporto. Mentre questa progettazione è adatta per host con un singolo processo utente, nel caso di host con utenti multipli sarebbe più naturale che l'entità di trasporto fosse un processo separato, distinto da tutti i processi utenti.

L'interfaccia con il livello rete è data dalle procedure *to_net* e *from_net* (non mostrate). Ognuna di esse ha sei parametri. Per primo viene l'identificatore di connessione, a cui corrisponde un circuito virtuale del livello rete. Quindi vengono i bit *A* ed *M*, che quando sono uguali a 1 caratterizzano, rispettivamente, i messaggi di controllo e la presenza di ulteriori dati relativi a questo messaggio nel pacchetto successivo. Seguono il tipo di pacchetto, scelto fra i sei tipi elencati in figura 6-19. Infine abbiamo un puntatore ai dati stessi e un intero contenente il numero di byte di dati.

Alla chiamata di *to_net*, l'entità di trasporto inserisce tutti i parametri che verranno letti dal livello rete; alla chiamata di *from_net*, il livello rete estrae un pacchetto incompleto per l'entità di trasporto. Grazie al fatto che le informazioni vengono passate come parametri di procedura invece di essere passate come pacchetti effettivi, i dettagli del proto-

Pacchetto di rete	Significato
CALL REQUEST	Spedito per stabilire una connessione
CALL ACCEPTED	Risposta a CALL REQUEST
CLEAR REQUEST	Spedito per chiudere una connessione
CLEAR CONFIRMATION	Risposta a CLEAR REQUEST
DATA	Utilizzato per trasportare dati
CREDIT	Pacchetto di controllo per gestire la finestra

Fig. 6-19 I pacchetti del livello rete utilizzati nel nostro esempio.

collo del livello rete sono nascosti al livello trasporto. Se l'entità di trasporto dovesse tentare di spedire un pacchetto mentre la finestra scorrevole del circuito virtuale sottostante fosse piena, verrebbe sospeso all'interno di *to_net* fino a quando non si fosse liberato spazio nella finestra scorrevole. Questo meccanismo è trasparente per l'entità di trasporto ed è controllato dal livello rete utilizzando comandi come *enable_transport_layer* (abilitazione strato trasporto) e *disable_transport_layer* (disabilitazione strato trasporto), analoghi a quelli descritti nei protocolli del capitolo 3. Anche la gestione della finestra del packet layer è realizzata dal livello rete.

Oltre a questi meccanismi trasparenti di sospensione, l'entità di trasporto può anche chiamare esplicitamente le procedure *sleep* (dormi) e *wakeup* (svegliati) (non mostrate). La procedura *sleep* viene chiamata quando l'entità di trasporto è bloccata logicamente in attesa che avvenga un evento esterno, in genere l'arrivo di un pacchetto. Dopo la chiamata *sleep* l'entità di trasporto (e naturalmente il processo utente) sospende la propria esecuzione.

Il codice effettivo dell'entità di trasporto è mostrato in figura 6-20. Ogni connessione è sempre in uno dei seguenti sette stati:

1. IDLE – Connessione non ancora stabilita.
2. WAITING – È stato eseguito CONNECT ed è stato spedito CALL REQUEST.
3. QUEUED – È arrivato un CALL REQUEST; ancora nessun LISTEN.
4. ESTABLISHED – La connessione è stata creata.
5. SENDING – L'utente sta aspettando il permesso di spedire un pacchetto.
6. RECEIVING – È stata eseguita una RECEIVE.
7. DISCONNECTING – È stata eseguita localmente una DISCONNECT.

Le transizioni fra gli stati possono avvenire in seguito a uno dei seguenti eventi: viene eseguita una primitiva, arriva un pacchetto oppure scade un timer.

Le procedure mostrate in figura 6-20 sono di due tipi. Gran parte di esse sono invocabili direttamente dai programmi utente; tuttavia, *packet_arrival* e *clock* sono differenti. Sono

```

#define MAX_CONN 32           /* maximum number of simultaneous connections */
#define MAX_MSG_SIZE 8192     /* largest message in bytes */
#define MAX_PKT_SIZE 512      /* largest packet in bytes */
#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;

/* Global variables. */
transport_address listen_address;          /* local address being listened to */
int listen_conn;                           /* connection identifier for listen */
unsigned char data[MAX_PKT_SIZE];          /* scratch area for packet data */

struct conn {
    transport_address local_address, remote_address; /* state of this connection */
    cstate state;                                         /* pointer to receive buffer */
    int byte_count;                                      /* send/receive count */
    int cir_req_received;                                /* set when CLEAR_REQ packet received */
    int timer;                                           /* used to time out CALL_REQ packets */
    int credits;                                         /* number of messages that may be sent */
} conn[MAX_CONN];

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)
{ /* User wants to listen for a connection. See if CALL_REQ has already arrived. */
  int i = 1, found = 0;

  for (i = 1; i <= MAX_CONN; i++)           /* search the table for CALL_REQ */
    if (conn[i].state == QUEUED && conn[i].local_address == t) {
      found = i;
      break;
    }

  if (found == 0) {                         /* No CALL_REQ is waiting. Go to sleep until arrival or timeout. */
    listen_address = t; sleep(); i = listen_conn;
  }
  conn[i].state = ESTABLISHED;             /* connection is ESTABLISHED */
  conn[i].timer = 0;                       /* timer is not used */
}

```

```

listen_conn = 0;           /* 0 is assumed to be an invalid address */
to_net(i, 0, 0, CALL_ACC, data, 0);    /* tell net to accept connection */
return(i);                /* return connection identifier */
}

int connect(transport_address l, transport_address r)
/* User wants to connect to a remote process; send CALL_REQ packet. */
int i;
struct conn *cptr;

data[0] = r;   data[1] = l;           /* CALL_REQ packet needs these */
i = MAX_CONN;          /* search table backward */
while (conn[i].state != IDLE && i > 1) i = i - 1;
if (conn[i].state == IDLE) {
    /* Make a table entry that CALL_REQ has been sent. */
    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();
    if (cptr->state == ESTABLISHED) return(i);
    if (cptr->clr_req_received) {
        /* Other side refused call. */
        cptr->state = IDLE;           /* back to IDLE state */
        to_net(i, 0, 0, CLEAR_CONF, data, 0);
        return(ERR_REJECT);
    }
} else return(ERR_FULL);           /* reject CONNECT: no table space */
}

int send(int cid, unsigned char bufptr[], int bytes)
/* User wants to send a message. */
int i, count, m;
struct conn *cptr = &conn[cid];

/* Enter SENDING state. */
cptr->state = SENDING;           /* # bytes sent so far this message */
cptr->byte_count = 0;
if (cptr->clr_req_received == 0 && cptr->credits == 0) sleep();
if (cptr->clr_req_received == 0) {
    /* Credit available; split message into packets if need be. */
    do {
        if (bytes - cptr->byte_count > MAX_PKT_SIZE) /* multipacket message */
            count = MAX_PKT_SIZE; m = 1; /* more packets later */
        else {                                /* single packet message */
            count = bytes - cptr->byte_count; m = 0; /* last pkt of this message */
        }
        for (i = 0; i < count; i++) data[i] = bufptr[cptr->byte_count + i];
        to_net(cid, 0, m, DATA_PKT, data, count); /* send 1 packet */
        cptr->byte_count = cptr->byte_count + count; /* increment bytes sent so far */
    } while (cptr->byte_count < bytes); /* loop until whole message sent */
}

```

```

cptr->credits--;           /* each message uses up one credit */
cptr->state = ESTABLISHED;
return(OK);

} else {
    cptr->state = ESTABLISHED;
    return(ERR_CLOSED);           /* send failed: peer wants to disconnect */
}

int receive(int cid, unsigned char bufptr[], int *bytes)
/* User is prepared to receive a message. */
struct conn *cptr = &conn[cid];

if (cptr->clr_req_received == 0) {
    /* Connection still established; try to receive. */
    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); /* send credit */
    sleep();                         /* block awaiting data */
    *bytes = cptr->byte_count;
}
cptr->state = ESTABLISHED;
return(cptr->clr_req_received ? ERR_CLOSED : OK);

}

int disconnect(int cid)
/* User wants to release a connection. */
struct conn *cptr = &conn[cid];

if (cptr->clr_req_received) {           /* other side initiated termination */
    cptr->state = IDLE;                 /* connection is now released */
    to_net(cid, 0, 0, CLEAR_CONF, data, 0);
} else {                                /* we initiated termination */
    cptr->state = DISCONN;              /* not released until other side agrees */
    to_net(cid, 0, 0, CLEAR_REQ, data, 0);
}
return(OK);

}

void packet_arrival(void)
/* A packet has arrived, get and process it. */
int cid;                               /* connection on which packet arrived */
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]; /* data portion of the incoming packet */
struct conn *cptr;

from_net(&cid, &q, &m, &ptype, data, &count); /* go get it */
cptr = &conn[cid];

```

```

switch (ptype) {
    case CALL_REQ: /* remote user wants to establish connection */
        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;

    case CALL_ACC: /* remote user has accepted our CALL_REQ */
        cptr->state = ESTABLISHED;
        wakeup();

    case CLEAR_REQ: /* remote user wants to disconnect or reject call */
        cptr->clr_req_received = 1;
        if (cptr->state == DISCONN) cptr->state = IDLE; /* clear collision */
        if (cptr->state == WAITING || cptr->state == RECEIVING || cptr->state == SENDING) wakeup();
        if (cptr->state == CLEAR_CONF):
            cptr->state = IDLE;

    case CREDIT: /* remote user is waiting for data */
        cptr->credits += data[1];
        if (cptr->state == SENDING) wakeup();

    case DATA_PKT: /* remote user has sent data */
        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)
    { /* The clock has ticked, check for timeouts of queued connect requests. */
        int i;
        struct conn *cptr;

        for (i = 1; i <= MAX_CONN; i++) {
            cptr = &conn[i];
            if (cptr->timer > 0) /* timer was running */
                cptr->timer--;
            if (cptr->timer == 0) /* timer has now expired */
                if (cptr->state == IDLE);
                    to_net(i, 0, 0, CLEAR_REQ, data, 0);
        }
    }
}

```

Fig. 6-20 Un esempio di entità di trasporto.

	Stato						
	Idle	Waiting	Queued	Established	Sending	Receiving	Disconnecting
LISTEN	P1: ~idle P2: A1/Estab P3: A2/idle			~Estab			
CONNECT	P1: ~idle P1: A3/Wait						
DISCONNECT				P4: A5/idle P4: A6/Disconnect			
SEND				P5: A7/Estab P5: A8/Send			
RECEIVE				A9/Receiving			
Call_req	P3: A1/Estab P3: A4/Queued						
Call_acc		~Estab					
Clear_req		~idle		A10/Estab	A10/Estab	A10/Estab	~idle
Clear_conf							~idle
DataPkt							A12/Estab
Credit					A11/Estab	A7/Estab	
Timeout				~idle			

Predicati

P1: Tabella connessioni piena	A1: Spedisci Call_acc	A7: Spedisci messaggio
P2: In attesa di Call_req	A2: Attendi Call_req	A8: Attendi nuovi crediti
P3: In attesa di LISTEN	A3: Spedisci Call_req	A9: Spedisci crediti
P4: In attesa di Clear_req	A4: Inizializza timer	A10: Accendi il flag Clr_req_received
P5: Credito disponibile	A5: Spedisci Clear_conf	A11: Memorizza credito
	A6: Spedisci Clear_req	A12: Accetta messaggio

Azioni

Fig. 6-21 Il protocollo di esempio come una macchina a stati finiti. Ogni casella possiede un predicato opzionale, un'azione opzionale, e un nuovo stato. La tilde indica che non viene presa nessuna azione principale. I predicati barrati indicano la negazione del predicato. Caselle vuote corrispondono a eventi impossibili o non validi.

attivate spontaneamente da eventi esterni, rispettivamente l'arrivo di un pacchetto e il battito di orologio. In effetti, sono routine di interruzione. Assumeremo che non vengano mai invocate finché l'entità di trasporto è in esecuzione. Possono essere chiamate solo quando il processo utente è sospeso, oppure sta eseguendo codice esterno all'entità di

trasporto. Questa proprietà è fondamentale per il corretto funzionamento dell'entità di trasporto. L'esistenza del bit *A* (attributo) nel preambolo del pacchetto permette di evitare di sprecare spazio per un preambolo del livello trasporto. I messaggi ordinari vengono spediti con *A* = 0. I messaggi di controllo del protocollo di trasporto (nel nostro esempio ne esiste solo uno, **CREDIT**) vengono spediti come pacchetti normali con *A* = 1. Questi messaggi di controllo sono rilevati ed elaborati dall'entità di trasporto ricevente.

La struttura dati principale usata dall'entità di trasporto è l'array *conn*, che possiede un record per ogni connessione potenziale. Il record mantiene lo stato della connessione, includendo gli indirizzi di trasporto all'altro capo, il numero di messaggi spediti e ricevuti sulla connessione, lo stato corrente, il puntatore al buffer utente, il numero di byte del messaggio attuale spediti o ricevuti finora, un bit che indica che l'utente remoto ha eseguito un **DISCONNECT**, un timer e un contatore di permessi utilizzato per abilitare la spedizione di messaggi. Non tutti questi campi sono utilizzati nel nostro semplice esempio, ma un'entità di trasporto completa dovrebbe includerli tutti. Si assume che ogni registrazione di *conn* sia inizializzata allo stato **IDLE**.

Quando l'utente chiama **CONNECT**, viene ordinato al livello rete di spedire un pacchetto **CALL REQUEST** alla macchina remota e l'utente si sospende. Quando il pacchetto **CALL REQUEST** arriva all'altro lato, l'entità di trasporto viene interrotta per eseguire *packet_arrival*, in modo da verificare se l'utente locale è in ascolto all'indirizzo specificato. In caso affermativo, viene spedito indietro un pacchetto **CALL ACCEPTED** e l'utente remoto viene svegliato; altrimenti, il pacchetto **CALL REQUEST** viene messo in coda per **TIMEOUT** battiti di orologio. Se viene eseguita una procedura **LISTEN** entro questo periodo, la connessione viene creata; altrimenti, il tempo scade e la connessione viene rifiutata tramite pacchetto **CALL REQUEST**. Questo meccanismo è necessario per evitare che il chiamante si blocchi per sempre nel caso che il processo remoto non voglia connettersi.

Sebbene il preambolo del protocollo sia stato eliminato, esiste sempre la necessità di individuare a quale connessione appartengano i pacchetti, in quanto possono esistere molte connessioni contemporaneamente. L'approccio più semplice è quello di utilizzare l'identificatore di circuito virtuale del livello trasporto anche come identificatore della connessione del trasporto. Inoltre, il numero di circuito virtuale può essere anche utilizzato come indice all'array *conn*. Quando un pacchetto arriva dal circuito virtuale *k* del livello rete, appartiene alla connessione di trasporto *k*, il cui stato è contenuto nel record *conn* [*k*]. Nel caso di connessioni iniziate da un host, l'identificatore di connessione è scelto dall'entità di trasporto originaria. Nel caso di chiamate in arrivo, è il livello rete a scegliere, decidendo per qualsiasi identificatore di circuito virtuale non utilizzato.

Per evitare la gestione dei buffer all'interno dell'entità di trasporto, viene utilizzato un meccanismo di controllo di flusso differente dallo sliding window originale. Al suo posto, quando un utente chiama **RECEIVE**, uno speciale **messaggio di credito** viene spedito all'entità di trasporto della macchina mittente e registrato nell'array *conn*. Quando viene chiamata la primitiva **SEND**, l'entità di trasporto verifica la presenza di un credito relativo alla connessione specificata. In caso affermativo, il messaggio viene spedito (in più di un pacchetto se necessario) e il credito viene decrementato; altrimenti, l'entità di trasporto si sospende fino all'arrivo di un credito. Questo meccanismo garantisce che nessun messaggio venga mai spedito, a meno che l'altro lato non abbia già eseguito un **RECEIVE**.

Come risultato, ogni volta che un messaggio arriva al ricevente esiste sicuramente un buffer disponibile in cui può essere collocato. Lo schema può essere facilmente generalizzato in modo che i riceventi abbiano più di un buffer e richiedano più di un messaggio. Ancora una volta, è importante tenere in considerazione la semplicità di figura 6-20. Generalmente, un'entità di trasporto realistica dovrebbe verificare la validità di tutti i parametri offerti, gestire il recovery di guasti del livello rete, occuparsi di collisioni di chiamate e supportare un servizio di trasporto più generale che includa servizi quali interruzioni, datagram e versioni non bloccanti delle primitive **SEND** e **RECEIVE**.

6.3.3 L'esempio visto come una macchina a stati finiti

Scrivere un'entità di trasporto è un lavoro difficile e impegnativo, specialmente nel caso di protocolli più realistici. Per ridurre la probabilità di fare un errore, è spesso utile rappresentare lo stato del protocollo come una macchina a stati finiti.

Abbiamo già visto che il nostro protocollo di esempio ha sette stati per ogni connessione. È anche possibile isolare dodici eventi che possono portare una connessione da uno stato a un altro. Cinque di questi eventi sono le cinque primitive del servizio. Altri sei corrispondono all'arrivo dei sei tipi di pacchetti legali. L'ultimo è causato dal timer. La figura 6-21 mostra le principali azioni del protocollo in forma matriciale. Le colonne corrispondono agli stati, mentre le righe corrispondono ai dodici eventi.

Ogni registrazione della matrice (cioè della macchina a stati finiti) di figura 6-21 ha al massimo tre campi: un predicato, un'azione e un nuovo stato. Il predicato indica sotto quali condizioni l'azione debba essere eseguita. Per esempio, nella casella in alto a sinistra, se viene eseguita una primitiva **LISTEN** e non esiste più spazio nelle tabelle (predicato *P1*), **LISTEN** fallisce e lo stato non cambia. Altrimenti, se è già arrivato un pacchetto **CALL REQUEST** per l'indirizzo di trasporto su cui si era in ascolto (predicato *P2*), la connessione viene creata immediatamente. Un'altra possibilità è il fatto che *P2* sia falso, cioè che non sia arrivato nessun pacchetto **CALL REQUEST**, nel qual caso la connessione rimane nello stato **IDLE** in attesa di un pacchetto **CALL REQUEST**.

È importante sottolineare che la scelta degli stati da utilizzare nella matrice non è interamente fissata dal protocollo stesso. In questo esempio, non esiste nessun stato **LISTENING**, che potrebbe ragionevolmente seguire la primitiva **LISTEN**. Tale mancanza è dovuta al fatto che ogni stato è associato a un record di connessione, e nessun record di connessione è creato da **LISTEN**. Perché no? Poiché si è deciso di utilizzare l'identificatore del circuito virtuale del livello rete come identificatore di connessione, e nel caso di un **LISTEN** l'identificatore di circuito virtuale viene scelto dal livello rete all'arrivo di un pacchetto **CALL REQUEST**.

Le azioni *A1-A12* rappresentano le azioni principali, come spedire un pacchetto e azionare un timer. Non sono elencate tutte le azioni minori, quali per esempio inizializzare i campi di un record di connessione. Se un'azione richiede l'attivazione di un processo sospeso, vanno contate anche le azioni che seguono la riattivazione. Ad esempio, se un pacchetto **CALL REQUEST** arriva a un processo sospeso in sua attesa, la trasmissione del pacchetto **CALL ACCEPT** che segue la riattivazione fa parte delle azioni di **CALL REQUEST**. Dopo l'esecuzione di ogni azione la connessione può spostarsi in un nuovo stato, come mostrato in figura 6-21.

Il vantaggio di rappresentare il protocollo come una matrice è triplice. Innanzitutto, in

questa forma è molto più facile per un programmatore controllare l'azione richiesta per ognuna delle combinazioni di stato ed evento. Nelle realizzazioni commerciali, alcune delle combinazioni verrebbero usate per la gestione degli errori. In figura 6-21 non esiste alcuna distinzione fra le situazioni impossibili e quelle illegali. Ad esempio, se una connessione è in stato *waiting*, l'evento *DISCONNECT* è impossibile, poiché l'utente è bloccato e non può eseguire nessuna primitiva. D'altro canto, nello stato *sending*, non sono attesi pacchetti dati in quanto non è stato creato nessun credito. L'arrivo di un pacchetto dati è un errore del protocollo.

Il secondo vantaggio della rappresentazione matriciale del protocollo si incontra durante la sua realizzazione. Si potrebbe pensare a un array bidimensionale nel quale l'elemento $a[i][j]$ è un puntatore o un indice di una procedura che gestisce l'occorrenza dell'evento i quando lo stato è j . Una possibile realizzazione consiste nello scrivere l'entità di trasporto come un piccolo ciclo, all'inizio del quale si attende un evento. Quando questo si verifica, la connessione corrispondente viene individuata e si seleziona il suo stato. Una volta noti l'evento e lo stato, l'entità di trasporto non deve far altro che utilizzare l'array a e chiamare la procedura appropriata. Questo approccio è caratterizzato da un design più regolare e sistematico rispetto alla nostra entità di trasporto.

Il terzo vantaggio dell'approccio basato sulle macchine a stati finiti riguarda la descrizione del protocollo. In alcuni documenti standard, i protocolli sono descritti tramite una macchina a stati finiti come quella di figura 6-21. Passare da questo tipo di descrizione a un'entità di trasporto funzionante è molto più facile se quest'ultima è anch'essa guidata da una macchina a stati finiti basata su quella dello standard.

Lo svantaggio principale dell'approccio basato sulle macchine a stati finiti consiste nella maggiore difficoltà di comprensione rispetto all'esempio di programmazione intuitiva che abbiamo utilizzato inizialmente. Tuttavia, questo problema può essere parzialmente risolto disegnando la macchina a stati finiti come un grafo, come in figura 6-22.

6.4 I protocolli di trasporto di Internet (TCP e UDP)

Internet possiede due protocolli principali del livello trasporto, uno orientato alla connessione e uno senza connessione. Nei prossimi paragrafi li studieremo entrambi. Il protocollo orientato alla connessione si chiama TCP, mentre quello senza connessione è detto UDP. Poiché UDP è in pratica IP con un piccolo preambolo aggiunto, concentreremo la nostra attenzione su TCP.

TCP (Transmission Control Protocol-protocollo di controllo della trasmissione) fu progettato esplicitamente per fornire un flusso affidabile ed end-to-end a partire da un'internet inaffidabile. Un'internet si distingue da una singola rete in quanto sezioni distinte possono essere completamente differenti per quanto riguarda topologia, capacità, ritardi, dimensione dei pacchetti e altri parametri. TCP fu progettato per adattarsi dinamicamente alle proprietà dell'internet e per essere robusto nei confronti di molte tipologie di guasto. TCP fu definito formalmente in RFC 793. Con il passare del tempo furono scoperti vari errori e inconsistenze, e i requisiti di alcune sezioni furono modificati. Questi chiarimenti sono illustrati dettagliatamente in RFC 1122, insieme alla correzione di alcuni errori. Ulteriori estensioni sono contenute in RFC 1323.

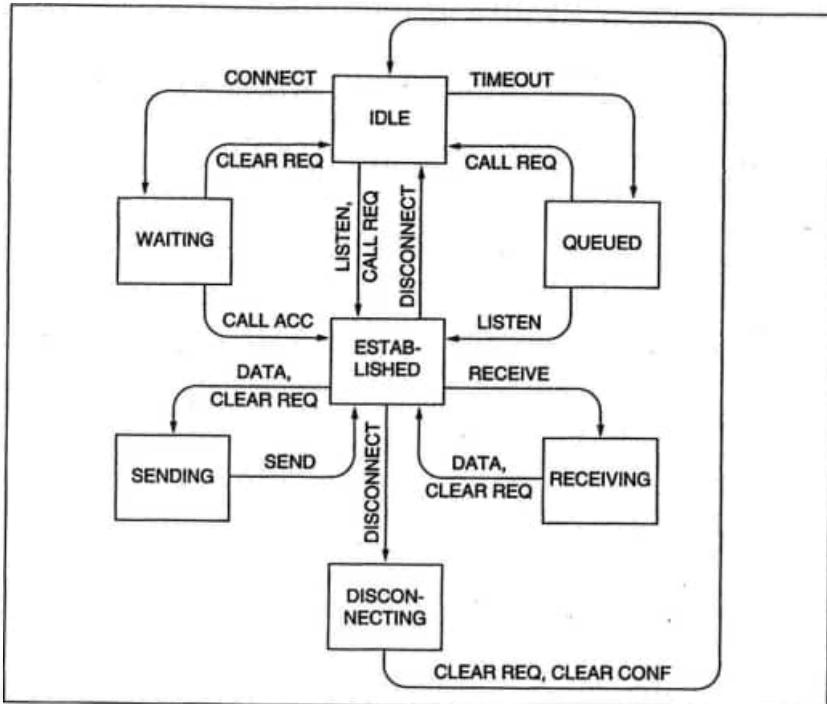


Fig. 6-22 Il protocollo di esempio in forma grafica. Le transizioni che non modificano lo stato di una connessione sono omesse per semplicità.

Ogni macchina che supporta TCP possiede un'entità di trasporto TCP, che gestisce i flussi di dati TCP e si interfaccia con il livello IP; può essere un processo utente, oppure far parte del nucleo. Un'entità TCP riceve flussi di dati dai processi locali, li spezza in unità lunghe al più 64 KB (ma generalmente di circa 1500 byte) e spedisce queste unità come datagram IP separati. Quando un datagram IP contenente dati TCP giunge a una macchina, i dati vengono passati all'entità TCP, la quale ricostruisce il flusso originale di byte. Per semplicità, in alcune occasioni utilizzeremo il termine "TCP" per indicare l'entità di trasporto TCP (un elemento software) oppure il protocollo TCP (un insieme di regole). Il significato sarà chiaro dal contesto. Ad esempio, nella frase "L'utente passa i dati a TCP" si intende ovviamente l'entità di trasporto TCP.

Il livello IP non fornisce alcuna garanzia sulla consegna corretta dei datagram; quindi, è compito di TCP ritrasmetterli quando necessario. I datagram in arrivo possono essere nell'ordine sbagliato; è ancora compito di TCP riassemblare i messaggi nella sequenza corretta. In breve, TCP deve fornire l'affidabilità che molti utenti desiderano e che IP non possiede.

6.4.1 Il modello di servizio TCP

Il servizio di TCP si ottiene tramite la creazione, da parte dell'utente e del ricevente, di punti di accesso chiamati socket, discussi nel paragrafo 6.1.3. Ogni socket è caratterizzato da un identificatore (indirizzo), consistente dell'indirizzo IP dell'host e di un numero di 16 bit locale all'host, detto **porta**. Una porta non è altro che il nome TCP dei TSAP. Per ottenere un servizio TCP, si deve creare esplicitamente una connessione fra un socket della macchina mittente e un socket della macchina ricevente. Le chiamate socket sono elencate in figura 6-6.

Un socket può essere utilizzato da più di una connessione contemporaneamente. In altre parole, due o più connessioni possono terminare nello stesso socket. Le connessioni sono identificate dall'identificatore di socket di entrambi i lati, cioè (*socket1, socket2*). Non vengono utilizzati numeri di circuiti virtuali o altri identificatori.

Le porte inferiori alla 256 sono chiamate **porte ben note** (well-known ports) e sono riservate per servizi standard. Ad esempio, qualsiasi processo che desideri stabilire una connessione con un host per trasferire file utilizzando TCP può connettersi alla porta 21 dell'host destinazione per contattare il demone FTP. In modo analogo, per stabilire una sessione di login remoto utilizzando TELNET, si adopera la porta 23. L'elenco delle porte standard è contenuto in RFC 1700.

Tutte le connessioni TCP sono full duplex e punto-a-punto. Full duplex significa che il traffico può scorrere in entrambe le direzioni contemporaneamente; punto-a-punto significa che ogni connessione ha esattamente due estremi. TCP non supporta il multicasting e il broadcasting.

Una connessione TCP consiste di un flusso di dati e non di messaggi. I confini dei messaggi non sono preservati da un capo all'altro. Ad esempio, se il processo mittente esegue quattro scritture di 512 byte su un canale TCP, queste possono essere consegnate al processo ricevente come quattro segmenti di 512 byte, due segmenti di 1024 byte oppure un segmento di 2048 byte (vedi figura 6-23), o in qualsiasi altra maniera. Non c'è modo che il ricevente possa individuare le unità di scrittura dei dati. Anche i file di UNIX hanno questa proprietà. Il lettore di un file non può sapere se il file è stato scritto un blocco o 1 byte alla volta, oppure tutto insieme. Come per i file UNIX, il software TCP non ha idea del significato dei byte e nessun interesse nel conoscerlo. Un byte è solo un byte.

Quando un'applicazione passa dati a TCP, quest'ultimo potrebbe spedirli immediatamente oppure salvarli in un buffer (per spedire insieme una grande quantità di dati), a sua discrezione.

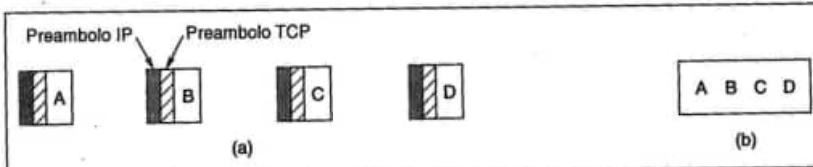


Fig. 6-23 (a) Quattro segmenti di 512 byte spediti come datagram IP separati. (b) I 2048 byte di dati consegnati in una singola chiamata READ.

Tuttavia, in alcune occasioni l'applicazione stessa desidera effettivamente che i dati vengano spediti immediatamente. Ad esempio, si supponga che un utente sia connesso a un macchina remota. Dopo la fine di una linea di comando e dopo l'inserimento del carattere "a capo", è essenziale che la linea sia inviata immediatamente alla macchina remota e non conservata in attesa delle prossime linee. Per forzare l'output dei dati, le applicazioni possono utilizzare il flag PUSH, per indicare a TCP di non ritardare la trasmissione. Alcune delle applicazioni più vecchie utilizzavano il flag PUSH per delimitare i confini dei messaggi. Sebbene in qualche occasione questo trucco funzioni, qualche volta fallisce in quanto non tutte le realizzazioni IP passano il flag PUSH all'applicazione del lato ricevente. Inoltre, se un altro PUSH arriva prima che quello precedente sia stato trasmesso (ad es. perché la linea di output è occupata), TCP è libero di raggruppare tutti i dati in un singolo datagram IP, senza alcuna separazione fra le varie sezioni.

L'ultima caratteristica del servizio di TCP che vale la pena menzionare riguarda i **dati urgenti**. Quando un utente interattivo preme i tasti DEL o CTRL-C per interrompere una computazione remota che è già iniziata, l'applicazione mittente inserisce alcune informazioni di controllo nel flusso di dati e le passa a TCP assieme al flag URGENT. Questo evento forza TCP a smettere di accumulare dati e a trasmettere immediatamente tutto quello che riguarda quella connessione.

Quando i dati urgenti vengono ricevuti dalla destinazione, l'applicazione ricevente viene interrotta (per esempio tramite un segnale di UNIX), in modo che possa sospendere tutto quello che stava facendo e leggere il flusso di input per trovare i dati urgenti, la cui fine viene marcata, e quindi l'applicazione sa riconoscere la loro terminazione. Al contrario, è compito dell'applicazione individuare l'inizio dei dati urgenti, in quanto non è segnalato. È compito dell'applicazione individuarlo. Questo schema fornisce un rozzo meccanismo di segnalazione e lascia all'applicazione tutto il resto.

6.4.2 Il protocollo TCP

In questo paragrafo vedremo una panoramica generale del protocollo TCP. Nel prossimo analizzeremo campo per campo il preambolo corrispondente.

Ogni byte di una connessione TCP possiede il proprio numero di sequenza a 32 bit. Nel caso di un host che spedisce alla piena velocità di 10 Mbps, teoricamente i numeri di sequenza potrebbero essere riutilizzati dopo un'ora, ma in pratica è necessario molto più tempo. I numeri di sequenza vengono utilizzati sia per il meccanismo degli ack, sia per quello delle finestre, inseriti in campi separati del preambolo da 32 bit.

Le entità TCP mittente e ricevente si scambiano dati sotto forma di segmenti. Un **segmento** consiste di un preambolo fisso di 20 byte (più alcune parti opzionali), seguito da zero o più byte di dati. Il software TCP decide la dimensione dei segmenti. Può accumulare in un segmento dati ottenuti da parecchie operazioni di scrittura, oppure dividere in più segmenti dati ottenuti da una singola operazione di scrittura.

Due limiti pongono delle restrizioni sulla dimensione dei segmenti. Primo, ogni segmento, incluso il preambolo TCP, deve entrare in un pacchetto IP di 65.536 byte. Secondo, ogni rete possiede un **MTU** (Maximum Transfer Unit – unità massima di trasferimento), e ogni segmento deve entrare in un MTU. In pratica, l'MTU è lungo generalmente poche migliaia di byte e quindi definisce il limite superiore della dimensione del segmento. Se

un segmento attraversa una serie di reti senza essere frammentato e quindi arriva a una in cui MTU è inferiore alla dimensione del segmento, il router al confine frammenta il segmento in due o più segmenti più piccoli.

Dunque, un segmento che è troppo grande per una delle reti che deve attraversare può essere spezzato da un router in più segmenti. Ogni nuovo segmento ottiene il proprio preambolo IP e TCP, e quindi la frammentazione aumenta lo spazio totale sprecato (in quanto ogni segmento aggiunge 40 byte di preamboli extra).

Il principale protocollo utilizzato dalle entità TCP è di tipo sliding window. Al momento di trasmettere un segmento, il mittente inizializza un timer. Quando il segmento arriva a destinazione, l'entità TCP ricevente spedisce indietro un segmento (contenente anche dati, se presenti) che trasporta un numero di ack uguale al successivo numero di sequenza che attende di ricevere. Se il timer del mittente scade prima che l'ack sia ricevuto, il segmento viene ritrasmesso.

Sebbene questo protocollo sembri semplice, esistono molti dettagli che illustreremo in seguito. Ad esempio, poiché i segmenti possono essere frammentati, è possibile che parte dei segmenti trasmessi arrivino e siano confermati dall'entità TCP ricevente, ma che il resto sia perso. I segmenti possono anche arrivare fuori ordine, e quindi è possibile che i byte 3072-4095 siano già stati ricevuti, ma che non possano essere confermati in quanto i byte 2048-3071 non sono ancora arrivati. Durante la trasmissione, i segmenti possono anche essere ritardati così a lungo da essere ritrasmessi dal mittente. Se un segmento ritrasmesso prende un percorso differente dall'originale e viene frammentato diversamente, possono arrivare sporadicamente sezioni sia dell'originale che del duplicato, richiedendo un'amministrazione attenta per ottenere un flusso di byte affidabile. Infine, dato il numero di reti che compongono Internet, è possibile che un segmento attraversi una rete congestionata (oppure rotta) lungo il cammino.

TCP dove essere in grado di affrontare questi problemi e risolverli in modo efficiente. Sono stati fatti un gran numero di sforzi per ottimizzare le prestazioni dei flussi TCP, anche in caso di problemi di rete. In seguito discuteremo numerosi algoritmi utilizzati da svariate realizzazioni TCP.

6.4.3 Il preambolo del segmento TCP

La figura 6-24 mostra la struttura di un segmento TCP. Ogni segmento inizia con un preambolo a formato fisso di 20 byte. Il preambolo fisso può essere seguito da opzioni di preambolo. Dopo le opzioni (se presenti) possono seguire fino a $65.535 - 20 - 20 = 65.495$ byte di dati, dove i primi 20 si riferiscono al preambolo IP e i secondi al preambolo TCP. I segmenti senza dati sono legali e vengono comunemente utilizzati per gli ack e per i messaggi di controllo.

Analizziamo il preambolo TCP campo per campo. I campi *Source port* (porta sorgente) e *Destination port* (porta destinazione) identificano gli estremi locali della connessione. Ogni host deve decidere come allocare le proprie porte successive alla 256. La coppia (Porta, Indirizzo IP dell'host) forma un TSAP unico di 48 bit. I numeri di socket sorgente e destinazione identificano la connessione.

I campi *Sequence number* (numero di sequenza) e *Acknowledgement number* (numero di ack) hanno il loro normale significato. Si noti che il secondo specifica il prossimo byte

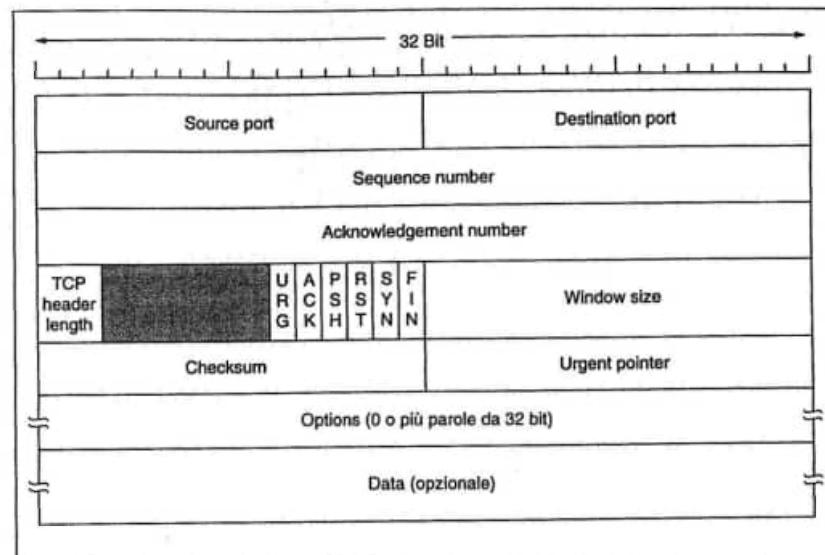


Fig. 6-24 Il preambolo TCP.

atteso e non l'ultimo byte correttamente spedito. Entrambi sono lunghi 32 bit, in quanto in un flusso TCP ogni byte di dati viene numerato.

Il campo *TCP header length* (lunghezza del preambolo TCP) misura il contenuto del preambolo TCP in parole di 32 bit. Questa informazione è necessaria in quanto il campo *Options* è di lunghezza variabile (e quindi lo è anche il preambolo). Tecnicamente, questo campo indica in realtà il punto di partenza dei dati all'interno del segmento, misurato in parole di 32 bit, ma questo numero coincide con la lunghezza del preambolo e quindi l'effetto è lo stesso.

Segue quindi un campo di 6 bit non utilizzato. Il fatto che questo campo sia sopravvissuto intatto per oltre un decennio indica quanto sia ben progettato TCP. Altri protocolli lo avrebbero utilizzato per correggere errori della progettazione originale.

A questo punto vengono sei flag da 1 bit. *URG* è uguale a 1 se il campo *Urgent pointer* (puntatore urgente) è valido. L'*Urgent pointer* viene utilizzato per indicare uno scostamento in byte a partire dal numero di sequenza attuale dove possono essere trovati i dati urgenti. Questo possibilità permette di utilizzare messaggi di interruzione. Come menzionato in precedenza, questo servizio è essenziale per consentire al mittente di mandare un segnale al ricevente senza coinvolgere TCP nel motivo dell'interruzione.

Il bit *ACK* è uguale a 1 per indicare che il campo *Acknowledgement number* è valido. Se *ACK* è 0, il segmento non contiene un ack e quindi il campo *Acknowledgement number* viene ignorato.

Il bit *PSH* indica dati di tipo PUSH. In questo modo si richiede gentilmente al mittente

di consegnare i dati all'applicazione al momento dell'arrivo, evitando che siano salvati in un buffer in attesa che quest'ultimo venga riempito (operazione altrimenti probabile per motivi di efficienza).

Il bit *RST* viene utilizzato per far reinizializzare una connessione che è diventata instabile a causa del guasto di un host o per qualche altro motivo. Viene anche utilizzato per rigettare un segmento non valido o per rifiutare l'apertura di una connessione. In generale, se si riceve un segmento con il bit *RST* acceso, ciò significa che da qualche parte c'è un problema.

Il bit *SYN* viene utilizzato per creare connessioni. La richiesta di connessione è caratterizzata da *SYN* = 1 e *ACK* = 0 per indicare che il campo acknowledgement non è valido. La risposta alla connessione contiene un ack, e quindi è caratterizzata da *SYN* = 1 e *ACK* = 1. In pratica, il bit *SYN* viene utilizzato per denotare messaggi di tipo CONNECTION REQUEST e CONNECTION ACCEPTED, mentre il bit *ACK* viene utilizzato per distinguere tra queste due possibilità.

Il bit *FIN* viene utilizzato per chiudere una connessione. Specifica che il mittente non ha ulteriori dati da spedire. Tuttavia, dopo aver chiuso una connessione, un processo può continuare a ricevere dati indefinitamente. Sia i segmenti *SYN* che quelli *FIN* possiedono numeri di sequenza e quindi vengono elaborati nell'ordine corretto.

Il controllo di flusso in TCP è gestito utilizzando un protocollo sliding window a dimensione variabile. Il campo *Window* (finestra) indica quanti byte possono essere spediti a partire dal byte confermato. Un campo *Window* uguale a 0 è legale e dice che i byte fino al numero *Acknowledgement number* - 1 inclusi sono stati ricevuti, ma che il ricevente ha bisogno di una pausa e non desidera ulteriori dati per il momento. Permessi di spedire potranno essere assegnati in seguito spedendo un segmento con lo stesso *Acknowledgement number* e con un campo *Window* diverso da 0.

Per garantire l'affidabilità, è presente anche un campo *Checksum*, che verifica il preambolo, i dati e lo pseudopreambolo mostrato in figura 6-25. Quando viene eseguito il calcolo, il campo *Checksum* viene posto uguale a 0, e il campo dati viene completato con uno 0 addizionale se la sua lunghezza è un numero dispari. L'algoritmo di checksum somma tutte le parole di 16 bit in complemento a 1 e quindi prende il complemento a 1 della somma. Come conseguenza, quando il ricevente esegue il calcolo sull'intero segmento (compreso il campo *Checksum*) il risultato deve essere 0.



Fig. 6-25 Lo psuedopreambolo incluso nella checksum di TCP.

Lo pseudopreambolo contiene l'indirizzo IP a 32 bit delle macchine sorgente e destinazione, il numero di protocollo per TCP (6), e il numero di byte del segmento TCP (compreso il preambolo). L'inclusione dello pseudopreambolo nel calcolo della checksum TCP aiuta a individuare i pacchetti consegnati alla destinazione sbagliata, ma questo è contrario al principio di gerarchia di protocolli, in quanto gli indirizzi IP appartengono al livello IP e non al livello TCP.

Il campo *Option* (opzione) fu progettato per fornire un metodo per aggiungere servizi extra non compresi nel preambolo regolare. L'opzione più importante è quella che consente a un host di specificare il più grande segmento TCP che è in grado di accettare. Utilizzare segmenti grandi è più efficiente che utilizzare segmenti piccoli, in quanto il preambolo di 20 byte può essere ammortizzato su una quantità superiore di dati, ma gli host più piccoli possono non essere in grado di gestire segmenti veramente grandi. Durante la creazione della connessione, ogni estremo può annunciare il suo massimo e vedere quello del partner. Il più piccolo dei due numeri vince. Se un host non utilizza questa opzione, il default è 536 byte di contenuto. Tutti gli host Internet devono accettare segmenti TCP di $536 + 20 = 556$ byte.

Per linee con grande capacità, grande ritardo, o entrambi, la finestra di 64 KB è spesso un problema. Su una linea T3 (44,736 Mbps), ci vogliono 12 ms per spedire una finestra di 64 KB. Se il tempo di propagazione di andata e ritorno è uguale a 50 ms (tipico per una fibra transcontinentale), il mittente resterà inattivo per 3/4 del tempo aspettando l'ack. In una comunicazione satellitare, la situazione è anche peggiore. Una dimensione maggiore della finestra permetterebbe al mittente di continuare a spedire dati, ma il campo *Window size* di 16 bit non consente di esprimere dimensioni superiori a 64 KB. In RFC 1323 è stata proposta un'opzione *Window scale* (scala della finestra), che consenta al mittente e al ricevente di negoziare un fattore di scala della finestra. Questo numero permette a entrambi i lati di spostare il campo *Window size* fino a 16 bit sulla sinistra, consentendo quindi fino a 2^{32} byte. Molte delle attuali realizzazioni di TCP supportano questa opzione.

Un'ulteriore opzione proposta in RFC 1106 e ora ampiamente implementata consiste nell'utilizzare un protocollo selective repeat invece di go back n. Se il ricevente ricevesse un segmento corrotto e quindi un gran numero di segmenti buoni, il protocollo TCP normale andrebbe normalmente in timeout e ritrasmetterebbe tutti i segmenti non confermati, inclusi quelli che sono stati ricevuti correttamente. RFC 1106 introduce i NAK (ack negativi), per consentire al ricevente di chiedere un segmento specifico (o più segmenti). Dopo averli ricevuti, può confermare la ricezione di tutti i dati nel buffer, riducendo in questo modo il numero di dati trasmessi.

6.4.4 Gestione delle connessioni TCP

In TCP, le connessioni vengono create utilizzando il protocollo three-way handshake discusso nel paragrafo 6.2.2. Per stabilire una connessione, un lato (il server) aspetta passivamente una connessione in arrivo eseguendo le primitive LISTEN e ACCEPT, specificando una sorgente specifica oppure nessuna in particolare.

L'altro lato (il cliente), esegue una primitiva CONNECT specificando l'indirizzo IP e la porta a cui vuole connettersi, la dimensione accettabile massima dei segmenti TCP e

opzionalmente alcuni dati utente (ad es. una password). La primitiva CONNECT spedisce un segmento TCP con il bit *SYN* acceso e il bit *ACK* spento e aspetta una risposta. Quando questo segmento arriva alla destinazione, l'entità TCP verifica l'esistenza di un processo che ha eseguito LISTEN sulla porta contenuta nel campo *Destination port*. Altrimenti, spedisce una risposta con il bit *RST* acceso per rifiutare la connessione.

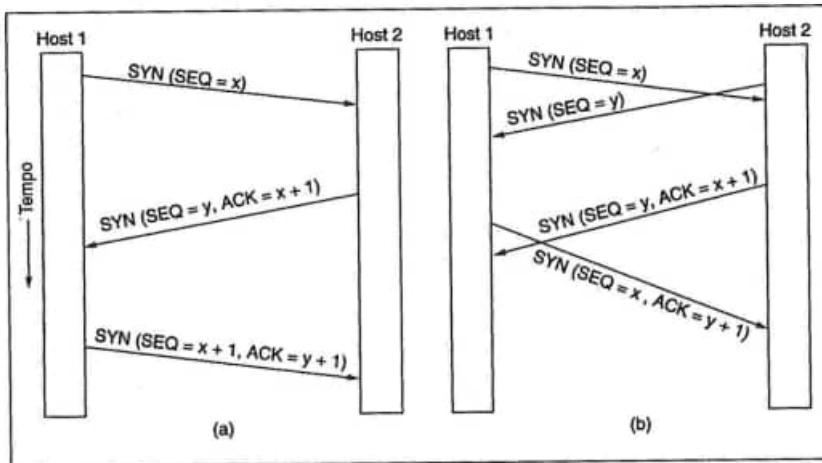


Fig. 6-26 (a) Creazione di una connessione TCP nel caso normale. (b) Collisione di chiamate.

Se qualche processo è in ascolto alla porta, gli viene consegnato il pacchetto TCP appena arrivato. A questo punto, può accettare o rifiutare la connessione. Se accetta, viene spedito un segmento di ack. La sequenza di segmenti TCP nel caso normale è mostrata in figura 6-26 (a). Si noti che un segmento *SYN* consuma 1 byte di spazio di sequenza in modo che possa essere confermato senza ambiguità.

Nel caso che i due host tentino contemporaneamente di creare una connessione tra gli stessi socket, la sequenza di eventi è quella mostrata in figura 6-26 (b). Il risultato di questi eventi è la creazione di una sola connessione, poiché le connessioni vengono identificate dai loro end point. Se sia la prima che la seconda connessione vengono identificate come (x, y) , viene creata una sola registrazione nelle tabelle.

Il numero di sequenza iniziale su una connessione è diverso da 0 per i motivi che abbiamo discusso in precedenza. Viene utilizzato un schema basato su orologio, dove quest'ultimo produce un battito ogni 4 μ s. Per ulteriore sicurezza, quando un host si guasta, non può essere reinizializzato per il tempo massimo di vita di un pacchetto (120 s), per essere sicuri che nessun pacchetto delle connessioni precedenti stia ancora vagando per Internet. Sebbene le connessioni TCP siano full duplex, per comprendere come avviene la chiusura è preferibile pensarle come una coppia di connessioni simplex. Ogni connessione simplex viene chiusa indipendentemente dalla sua gemella. Per chiudere una connessione, entram-

be le parti possono spedire un segmento TCP con il bit *FIN* acceso, a indicare che non vi sono più dati da trasmettere. Quando viene confermata la ricezione del bit *FIN*, quella direzione viene spenta. Tuttavia, i dati possono continuare a fluire indefinitamente nell'altra direzione. Quando entrambe le direzioni vengono chiuse, la connessione viene rilasciata. Generalmente, sono necessari 4 segmenti TCP per chiudere una connessione. Tuttavia, è possibile che il primo *ACK* e il secondo *FIN* siano contenuti nello stesso segmento, riducendo il totale a 3.

Proprio come nelle chiamate telefoniche, in cui gli interlocutori dicono arrivederci e appendono il ricevitore contemporaneamente, entrambi gli estremi di una connessione TCP possono spedire segmenti *FIN* contemporaneamente. Questi vengono confermati nel solito modo, e la connessione viene chiusa. In effetti, non esiste alcuna differenza tra chiudere una connessione fra due host in sequenza oppure simultaneamente.

Per evitare il problema dei due eserciti, vengono utilizzati dei timer. Se una risposta a un segmento *FIN* non arriva entro il doppio del tempo massimo di vita di un pacchetto, il mittente del segmento *FIN* chiude la connessione. Prima o poi, l'altro lato si accorgerebbe che non c'è più nessuno in ascolto e andrà anch'esso in timeout. Sebbene questa soluzione non sia perfetta, dovrà essere sufficiente, dato che una soluzione perfetta è impossibile. In pratica raramente si presentano problemi.

I passi richiesti per creare e chiudere le connessioni possono essere rappresentati in una macchina a stati finiti con gli 11 stati elencati in figura 6-27. In ogni stato, alcuni eventi sono legali. Quando avviene un evento legale, è possibile intraprendere alcune azioni. Se avvengono altri tipi di eventi, viene riportato un errore.

Ogni connessione inizia nello stato *CLOSED* (chiusa). Lascia questo stato quando esegue o una apertura passiva (*LISTEN*) o un'apertura attiva (*CONNECT*). Se l'altro lato esegue l'operazione inversa, viene stabilita una connessione e lo stato diventa *ESTABLISHED* (stabilita). La chiusura di una connessione può essere iniziata da entrambi gli interlocutori. Quando viene completata, lo stato ritorna a *CLOSED*.

La macchina a stati finiti è mostrata in figura 6-28. Il caso comune di un cliente che si connette attivamente a un server passivo viene mostrato con linee più marcate (tratteggiate per il server, continue per il cliente). Le linee meno marcate rappresentano sequenze di eventi meno usuali. Ogni linea in figura 6-28 è etichettata da una coppia *evento/azione*. L'evento può essere o una chiamata di sistema iniziata dall'utente (*CONNECT*, *LISTEN*, *SEND* o *CLOSE*), l'arrivo di un segmento (*SYN*, *FIN*, *ACK* o *RST*), o il sopraggiungere di un timeout. L'azione corrisponde alla spedizione di un segmento di controllo (*SYN*, *FIN* o *RST*) oppure all'azione nulla, indicata da $-$. I commenti sono mostrati in parentesi.

Il diagramma può essere compreso più facilmente seguendo prima il percorso di un cliente (la linea continua più marcata) e quindi il percorso del server (la linea tratteggiata più marcata). Quando un'applicazione della macchina cliente esegue una richiesta *CONNECT*, l'entità TCP locale crea un record di connessione, gli assegna lo stato *SYN SENT* (*SYN* spedito) e spedisce un segmento *SYN*. Si noti che possono essere aperte molte connessioni simultaneamente su richiesta di diverse applicazioni; per questo motivo, ogni connessione ha il suo stato registrato nel record di connessione. Quando arriva il segmento *SYN+ACK*, TCP spedisce l'*ACK* finale del protocollo three-way handshake e si sposta nello stato *ESTABLISHED*. A questo punto è possibile spedire e ricevere dati.

Stato	Descrizione
CLOSED	Nessuna connessione attiva o in attesa
LISTEN	Il server attende una chiamata
SYN RCV	È arrivata una richiesta di connessione, attendi l'ACK
SYN SENT	L'applicazione ha iniziato ad aprire una connessione
ESTABLISHED	Stato normale di trasferimento dati
FIN WAIT 1	L'applicazione fa detto di aver finito
FIN WAIT 2	L'altro lato è d'accordo di chiudere
TIMED WAIT	Attende che spariscano tutti i pacchetti
CLOSING	Entrambi i lati hanno tentato di chiudere contemporaneamente
CLOSE WAIT	L'altro lato ha iniziato a chiudere
LAST ACK	Attende che spariscano tutti i pacchetti

Fig. 6-27 Gli stati utilizzati dalla macchina a stati finiti che gestisce le connessioni TCP.

Quando un'applicazione ha finito, esegue una primitiva *CLOSE*, che forza l'entità TCP a spedire un segmento *FIN* e ad aspettare l'*ACK* corrispondente (rettangolo tratteggiato denotato chiusura attiva). Quando arriva un ack, si passa allo stato *FIN WAIT 2* (attesa di *FIN 2*) e una direzione della connessione viene chiusa. A questo punto entrambi i lati vengono chiusi, ma TCP aspetta un tempo uguale al tempo massimo di vita dei pacchetti per essere sicuro che tutti i pacchetti della connessione siano scomparsi, nel caso in cui l'*ack* sia andato perso. Quando il tempo scade, TCP cancella il record di connessione.

A questo punto esaminiamo la gestione delle connessioni dal punto di vista del server. Il server esegue una primitiva *LISTEN* e si mette comodo ad aspettare. All'arrivo di un segmento *SYN*, spedisce una conferma e passa allo stato *SYN RCV* (*SYN* ricevuto). Quando il segmento *SYN* del server viene confermato, il protocollo three-way handshake è completato e il server passa nello stato *ESTABLISHED*. A questo punto è possibile trasferire dati.

Quando un cliente ha spedito abbastanza, esegue una primitiva *CLOSE*, che causa l'arrivo di un pacchetto *FIN* al server (il rettangolo tratteggiato denotato come chiusura passiva). Il server viene quindi informato. Quando anch'esso esegue una primitiva *CLOSE*, viene spedito un segmento *FIN* al cliente. Quando arriva la conferma del cliente, il server chiude la connessione e scarta il record di registrazione.

6.4.5 Politica di trasmissione di TCP

La gestione delle finestre in TCP non è direttamente collegata agli ack come in gran parte dei protocollti data link. Ad esempio, si supponga che il ricevente abbia un buffer di 4096 byte come mostrato in figura 6-29. Se il mittente trasmetterà un segmento di 2048 byte che verrà correttamente ricevuto, il ricevente confermerà la ricezione del segmento. Tuttavia, poiché a questo punto il buffer disporrà solamente di 2048 byte (fino a quando

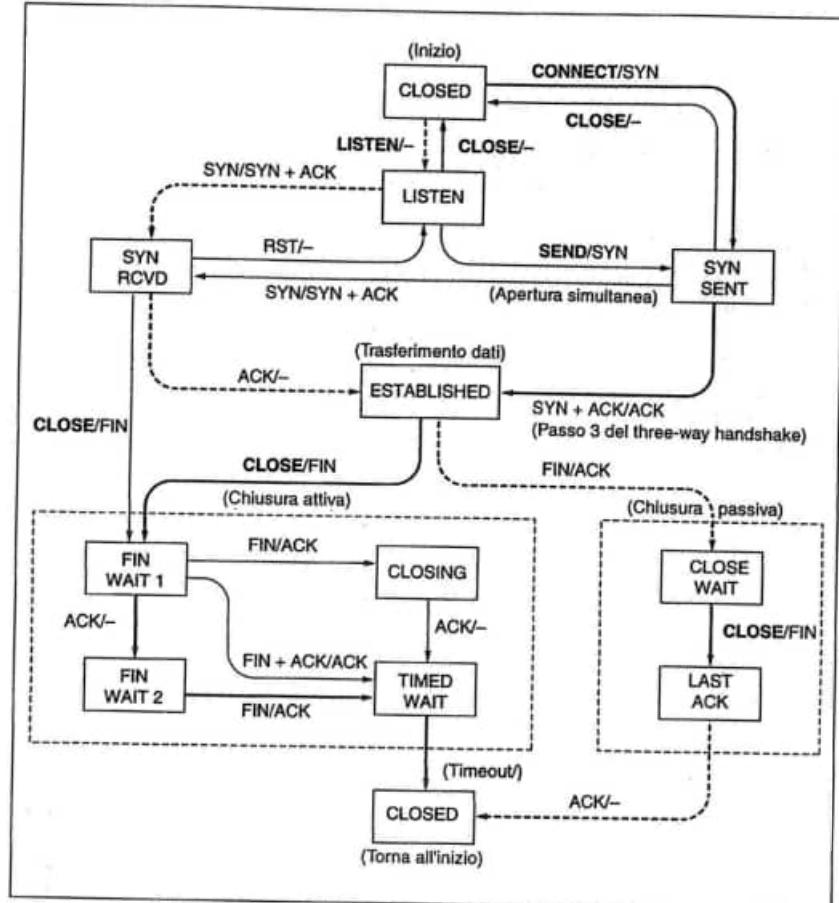


Fig. 6-28 La macchina a stati finiti che gestisce le connessioni TCP. La linea continua più marcata rappresenta il percorso normale per un cliente. La linea tratteggiata più marcata rappresenta il percorso normale per un server. Le linee più leggere rappresentano eventi meno usuali.

l'applicazione non rimuoverà i dati dal buffer), annuncerà una finestra di 2048 byte a partire dal prossimo byte atteso.

A questo punto il mittente trasmetterà altri 2048 byte, che verranno consegnati; la finestra annunciata sarà ora di 0 byte. Il mittente dovrà fermarsi fino a quando l'applicazione sul lato ricevente non rimuoverà qualche dato dal buffer; a quel punto TCP potrà annunciare una finestra più grande.

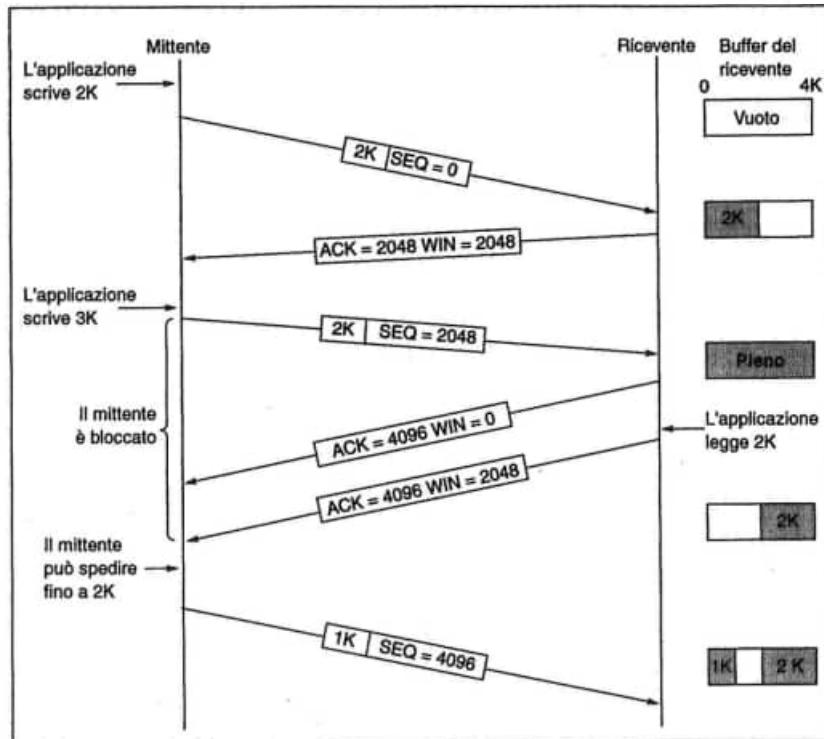


Fig. 6-29 Gestione delle finestre in TCP.

Quando la finestra è 0, generalmente il mittente non può spedire segmenti, con due eccezioni. Primo, possono essere spediti i dati urgenti, ad esempio per consentire all'utente di terminare i processi in esecuzione sulla macchina remota. Secondo, il mittente può spedire un segmento di 1 byte per fare in modo che il ricevente annuncii nuovamente il prossimo byte atteso e la dimensione della finestra. Lo standard TCP fornisce esplicitamente questa opzione per prevenire lo stallo nel caso in cui un segmento contenente la dimensione della finestra sia andato perso.

Non è richiesto che i mittenti trasmettano dati non appena questi giungono dall'applicazione. Né si richiede che i riceventi spediscono gli ack il più presto possibile. Ad esempio (figura 6-29), all'arrivo dei primi 2 KB di dati, TCP (sapendo di disporre di un buffer di 4 KB) avrebbe potuto conservare i dati fino all'arrivo di ulteriori 2 KB, per poter trasmettere un segmento di 4 KB. Questa libertà può essere utilizzata per migliorare le prestazioni.

Si consideri una connessione TELNET con un editor interattivo che reagisce ad ogni tasto premuto. Nel caso peggiore, quando un carattere arriva all'entità TCP, quest'ultima crea un segmento TCP di 21 byte e lo consegna a IP che spedisce un datagram di 41 byte. Dal

lato del ricevente, TCP spedisce immediatamente un ack di 40 byte (20 byte per il preambolo TCP e 20 per il preambolo IP). In seguito, quando l'editor ha letto il byte, TCP spedisce un aggiornamento alla finestra, spostandola avanti di un byte. Anche questo pacchetto è lungo 40 byte. Infine, quando l'editor ha elaborato il carattere, spedisce un pacchetto di 41 byte come eco. In tutto, vengono utilizzati 162 byte e 4 segmenti per ogni carattere battuto. Quando la banda disponibile è scarsa, questo metodo di gestione è svantaggioso.

Un approccio utilizzato da molte implementazioni TCP per ottimizzare questo scenario consiste nel ritardare gli ack e gli aggiornamenti delle finestre di 500 ms nella speranza di ricevere ulteriori dati. Assumendo che l'editor produca l'eco entro 500 ms, solo un pacchetto di 41 byte deve essere spedito indietro all'utente remoto, riducendo della metà la quantità di pacchetti e di banda utilizzata.

Sebbene questa regola riduca il carico di rete creato dal ricevente, il mittente sta ancora operando in modo inefficiente spedendo pacchetti di 41 byte contenenti 1 byte di dati. Un metodo per ridurre questo uso è conosciuto come **algoritmo di Nagle** (Nagle, 1984). Il suggerimento di Nagle è molto semplice: quando i dati arrivano al mittente 1 byte alla volta, si spedisce solo il primo byte e si salva nel buffer il resto fino a quando il primo byte non viene confermato. Quindi si spediscono tutti i caratteri salvati in un unico segmento TCP e si inizia a "bufferizzare" ancora fino a quando non vengono tutti confermati. Se l'utente sta scrivendo velocemente e la rete è lenta, un numero sostanziale di caratteri può andare in ogni segmento, riducendo di molto la banda utilizzata. L'algoritmo consente inoltre di spedire un nuovo pacchetto se arrivano dati sufficienti per riempire metà della finestra o l'intero pacchetto massimo.

L'algoritmo di Nagle viene utilizzato da molte implementazioni TCP, ma esistono occasioni in cui è meglio disabilitarlo. In particolare, quando un'applicazione X-Windows viene eseguita su Internet, i movimenti del mouse devono essere spediti al computer remoto. Metterli da parte per spedirli come un unico pacchetto rende irregolare il movimento del cursore del mouse.

Un ulteriore problema che può rovinare le prestazioni di TCP è conosciuto come **silly window syndrome** (sindrome della finestra futile) (Clark, 1982). Questo problema si presenta quando i dati vengono passati all'entità TCP mittente in blocchi di grandi dimensioni, mentre un'applicazione interattiva dal lato ricevente legge i dati 1 byte alla volta. Per comprendere il problema, si osservi la figura 6-30. Inizialmente, il buffer TCP sul lato ricevente è pieno e il mittente ne è a conoscenza (ha una finestra di dimensione 0). Quindi l'applicazione interattiva legge un carattere dal flusso TCP. Questa azione rende felice il TCP ricevente, che spedisce un aggiornamento della finestra consentendo la spedizione di 1 byte. Il mittente si rende servizievole e spedisce 1 byte. Il buffer è di nuovo pieno e quindi il ricevente manda la conferma per il segmento di 1 byte, ma setta la finestra a 0. Questo comportamento può continuare per sempre.

La soluzione di Clark consiste nell'evitare che il ricevente spedisca un aggiornamento delle finestre di 1 byte. Al contrario, è forzato ad aspettare finché non possiede una quantità decente di spazio disponibile. Più specificatamente, il ricevente non dovrebbe spedire un aggiornamento della finestra finché non è in grado di gestire la dimensione

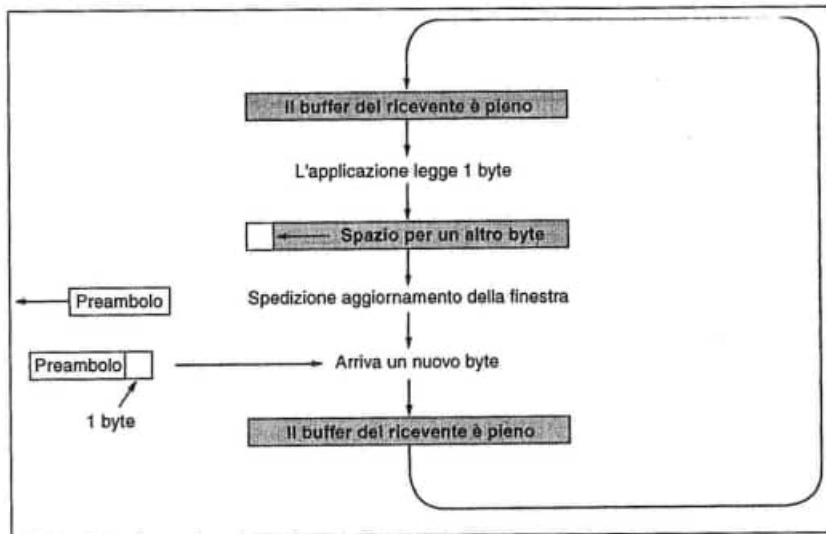


Fig. 6-30 Sindrome della finestra futile.

massima di segmento negoziata al momento di stabilire la connessione, oppure il suo buffer non è mezzo vuoto.

Inoltre, anche il mittente può dare una mano evitando di spedire segmenti minuscoli. Al contrario, dovrebbe aspettare fino a quando non si accumula nella finestra uno spazio sufficiente per spedire un segmento intero o almeno uno lungo la metà della dimensione del buffer del ricevente (che deve stimare dalla sequenza degli aggiornamenti di finestra che ha ricevuto nel passato).

L'algoritmo di Nagle e la soluzione di Clark alla sindrome della finestra futile sono complementari. Nagle stava cercando di risolvere il problema causato dalle applicazioni mittenti che consegnano i dati a TCP 1 byte alla volta. Clark stava cercando di risolvere il problema delle applicazioni riceventi che prendono i dati da TCP al ritmo di 1 byte alla volta. Entrambe le soluzioni sono valide e possono lavorare insieme. L'obiettivo è quello di evitare che il mittente spedisca segmenti piccoli e che il ricevente li richieda.

Per migliorare le prestazioni, il TCP ricevente può fare di meglio che aggiornare le finestre tramite grandi unità. Come il TCP mittente, anch'esso ha la possibilità di salvare nei buffer i dati; può quindi bloccare una richiesta READ dall'applicazione fino a quando non ha ricevuto un certo numero di dati da fornire. In questo modo si riducono il numero di chiamate a TCP e l'overhead corrispondente. Naturalmente, questo fa aumentare il tempo di risposta, ma per applicazioni non interattive quale il trasferimento file, l'efficienza può essere più importante del tempo di risposta alle richieste individuali.

Un'altra problematica relativa al ricevente riguarda cosa fare dei segmenti fuori ordine. Possono essere conservati oppure scartati, a discrezione del ricevente. Naturalmente, gli

ack possono essere spediti solo dopo che tutti i dati fino al byte confermato sono stati ricevuti. Se i segmenti 0, 1, 2, 4, 5, 6, 7 arrivano al ricevente, quest'ultimo può confermare tutti i dati fino all'ultimo byte del segmento 2 incluso. Quando il timer del mittente scade, ritrasmette il segmento 3. Se il ricevente ha "bufferizzato" i segmenti da 4 a 7, alla ricezione del segmento 3 può mandare la conferma di tutti i byte fino all'ultimo del segmento 7.

6.4.6 Controllo di congestione di TCP

Quando il carico offerto a una rete è maggiore di quello gestibile, si presenta il problema della congestione. Internet non è un'eccezione. In questo paragrafo presenteremo gli algoritmi di gestione delle congestioni che sono stati sviluppati nell'ultimo decennio. Sebbene anche il livello rete si preoccupi delle congestioni, il grosso del lavoro è eseguito da TCP, in quanto l'unica soluzione per una congestione è ridurre il flusso di dati. In teoria, la congestione potrebbe essere affrontata applicando un principio derivato dalla fisica: la legge della conservazione dei pacchetti. L'idea è quella di non inserire un nuovo pacchetto nella rete fino a quando uno vecchio non ne esce (cioè viene consegnato). TCP cerca di raggiungere questo obiettivo manipolando dinamicamente la dimensione delle finestre.

Il primo passo nella gestione delle congestioni consiste nel rilevarle. Tempo fa, rilevare una congestione era difficile. Un timeout causato da un pacchetto perso poteva essere causato da rumore su una linea di trasmissione, oppure dalla rimozione di un pacchetto da parte di un router congestionato. Individuare la differenza era difficile.

Oggi, la perdita di pacchetti dovuta a errori di trasmissione è relativamente rara in quanto gran parte dei canali a lunga distanza sono fibre (sebbene le reti senza filo siano un discorso a parte). Conseguentemente, gran parte dei timeout di trasmissione su Internet sono dovuti alle congestioni. Tutti gli algoritmi TCP di Internet assumono che i timeout siano causati dalle congestioni e sorvegliano i tempi di trasmissione nello stesso modo in cui i minatori sorvegliano i loro canarini.

Prima di discutere della modalità con cui TCP reagisce alle congestioni, descriveremo innanzitutto i tentativi per evitare che si presentino. Quando una connessione viene creata, si sceglie una dimensione di finestra accettabile. Il ricevente può specificare una finestra basata sulla dimensione del proprio buffer. Se il mittente accetta questa dimensione di finestra, non vi saranno problemi causati da esaurimento del buffer del ricevente; i problemi potrebbero però essere causati dalla congestione interna della rete.

In figura 6-31, vediamo questo problema illustrato tramite l'idraulica. In figura 6-31 (a), possiamo vedere un tubo largo che porta a un ricevente con scarsa capacità. Finché il mittente non versa più acqua di quella che il secchio può contenere, non verrà persa acqua. In figura 6-31 (b), il fattore limitante non è la capacità del secchio, ma la capacità di trasporto interna della rete. Se l'acqua arriva troppo in fretta, se ne perderà una parte (in questo caso traboccano dall'imbuto).

La soluzione di Internet consiste nel tener conto dell'esistenza di due problemi potenziali – capacità del ricevente e capacità della rete – e nell'affrontarli separatamente. A tale scopo, ogni mittente mantiene due finestre: una relativa al ricevente e l'altra relativa alla congestione (la **finestra di congestione**). Entrambe riflettono il numero di byte che il

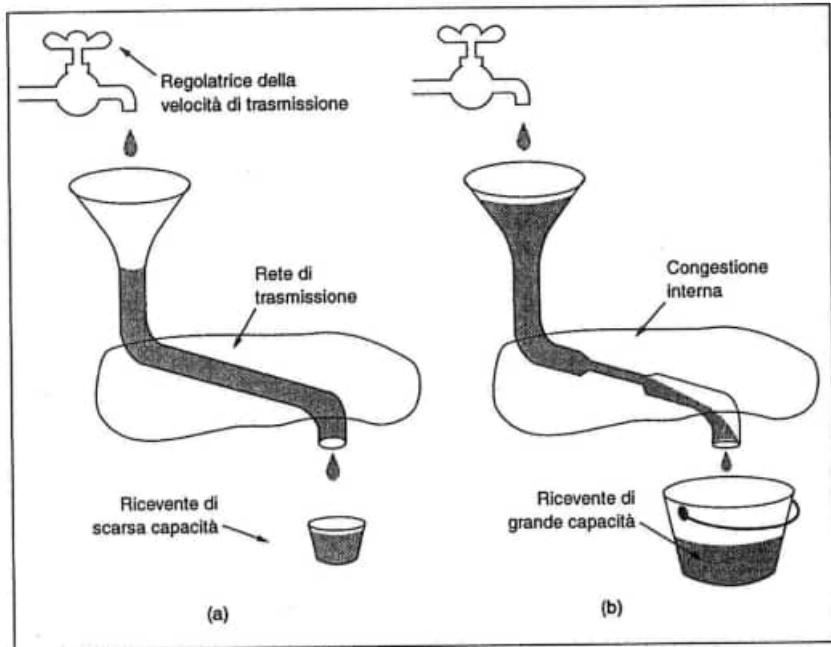


Fig. 6-31 (a) Un rete veloce che alimenta un ricevente a bassa capacità. (b) Una rete lenta che alimenta un ricevente ad alta capacità.

mittente è abilitato a trasmettere. Il numero di byte che può essere spedito è il minimo fra le due finestre. Quindi la finestra effettiva corrisponde al minimo fra ciò che è ritenuto adatto dal mittente e ciò che è ritenuto adatto dal ricevente. Se il ricevente dice "Spedisci 8K", ma il mittente sa che un insieme di dati maggiore di 4K intaserebbe la rete, spedisce 4K. D'altra parte, se il ricevente dice "Spedisci 8K" e il mittente sa che insiemi di dati inferiori a 32K possono passare senza problemi, spedisce gli 8K richiesti. Quando viene creata una connessione, il mittente pone la finestra di congestione uguale alla dimensione del segmento massimo che può essere spedito sulla connessione. Spedisce quindi un segmento di dimensione massima. Se questo segmento viene consegnato prima che il timer scada, il valore della finestra di congestione viene incrementato della dimensione massima di un segmento e il mittente spedisce 2 segmenti. Ogni qual volta uno di questi segmenti viene confermato, la dimensione della finestra viene aumentata della dimensione massima di 1 segmento. Quando la finestra di congestione è uguale a n segmenti, se tutti ed n vengono confermati in tempo, la finestra di congestione viene portata a $2n$ segmenti di dimensione massima. In effetti, ogni getto il cui arrivo viene confermato con successo raddoppia la finestra di congestione.

La finestra di congestione continua a crescere esponenzialmente fino a quando scade un

timeout oppure si raggiunge la finestra del ricevente. L'idea è la seguente: se getti di 1024, 2048 e 4096 byte funzionano perfettamente, mentre un getto di 8192 byte riceve un timeout, per evitare congestioni la finestra di congestione dovrebbe essere uguale a 4096. Fino a quando la finestra del ricevente rimane di 4096 byte, nessun getto più lungo verrà mai spedito, indipendentemente dalla finestra del ricevente. Questo algoritmo viene detto *slow start* (partenza lenta), ma non è per niente lento, in quanto esponenziale (Jacobson, 1988). Tutte le implementazioni TCP devono supportarlo.

Analizziamo ora l'algoritmo di controllo della congestione di Internet, che utilizza un terzo parametro, detto *soglia*, inizialmente di 64K, in aggiunta alle finestre del ricevente e di congestione. Quando scade un timeout, la soglia è posta a metà della finestra di congestione corrente, e la finestra di congestione viene riportata alla dimensione massima di un segmento. Viene utilizzato quindi *slow start* per determinare la capacità della rete, a parte il fatto che la crescita esponenziale si ferma quando la soglia viene raggiunta. Da quel punto in poi, trasmissioni con successo fanno crescere linearmente la finestra di congestione (della dimensione di un segmento massimo alla volta) invece di raddoppiarla. In effetti, questo algoritmo presuppone che ridurre della metà la finestra di congestione sia probabilmente accettabile, e quindi lavora gradualmente da lì in poi.

Come esempio del funzionamento dell'algoritmo di congestione, si veda la figura 6-32. In questo caso, la dimensione massima di segmento è 1024 byte. Inizialmente la finestra di congestione era di 64K, ma è avvenuto un timeout e quindi la soglia viene posta a 32K e la finestra di congestione a 1K corrispondenti alla trasmissione 0. La finestra di congestione cresce quindi esponenzialmente fino a raggiungere la soglia (32K). Da quel punto in poi cresce linearmente.

La trasmissione 13 è sfortunata (ovviamente) e produce un timeout. La soglia viene posta a metà della finestra corrente (fino ad ora 40K, e quindi la metà è 20K) e l'algoritmo *slow start* riparte da capo. Quando iniziano ad arrivare gli ack per la trasmissione 18, i primi quattro incrementi della finestra di congestione producono un raddoppiamento, mentre successivamente la crescita torna ad essere lineare.

Se non avvengono ulteriori timeout, la finestra di congestione continuerà a crescere fino alla dimensione della finestra del ricevente. A quel punto, rimarrà costante finché non avverranno ulteriori timeout e la finestra del ricevente non cambierà dimensione. Fra l'altro, se arriva un pacchetto ICMP SOURCE QUENCH e viene passato a TCP, questo evento viene trattato alla stessa maniera di un timeout.

Il tentativo di migliorare gli algoritmi di controllo della congestione è tuttora in corso. Per esempio, Brakmo et al. (1994) hanno ottenuto un miglioramento della capacità trasmissiva totale di TCP fra il 40% e il 70%, mediante una gestione più accurata dell'orologio, la predizione delle congestioni prima dello scadere dei timeout e l'utilizzazione di sistemi di allarme rapido per migliorare l'algoritmo *slow start*.

6.4.7 Gestione dei timer di TCP

TCP per eseguire i propri compiti utilizza più di un timer (almeno concettualmente). I più importanti sono i *timer di ritrasmissione*, che vengono fatti partire al momento di spedire un pacchetto. Se la ricezione del segmento viene confermata prima che il tempo finisca, il timer viene fermato. Se, d'altra parte, il timer scade prima dell'arrivo della conferma,

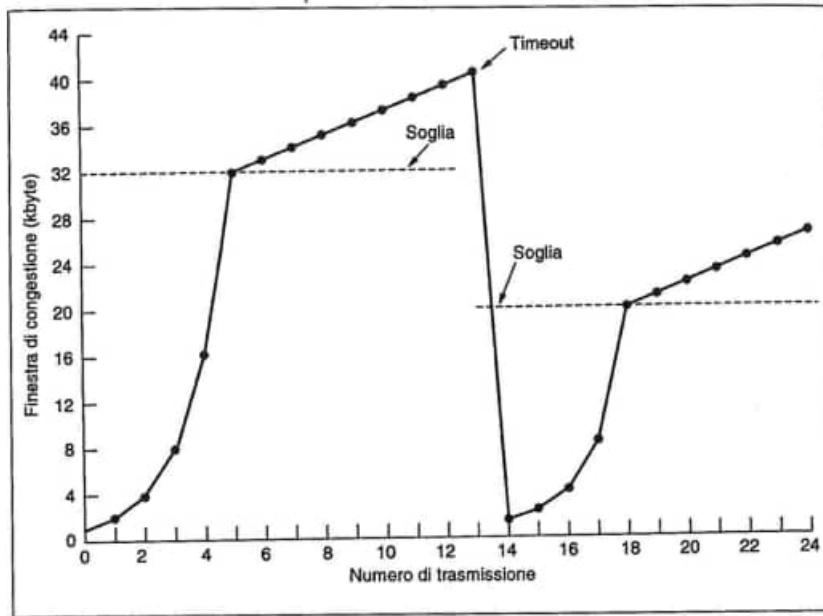


Fig. 6-32 Un esempio del funzionamento dell'algoritmo di controllo della congestione di Internet.

il segmento viene ritrasmesso (e il timer viene fatto ripartire). La domanda che ci poniamo è la seguente: quanto deve essere lungo questo intervallo di tempo?

Questo problema è molto più complesso nel livello trasporto di Internet che nei protocolli data link generici del capitolo 3. Nel secondo caso, il ritardo atteso è largamente predicable (cioè ha una bassa variabilità) e quindi si può fare in modo che il timer scada un attimo dopo il tempo presunto di arrivo della conferma, come illustrato in figura 6-33 (a). Poiché nel livello data link gli ack vengono raramente ritardati, l'assenza di notizie al tempo atteso significa generalmente che il frame o il suo ack sono andati persi.

TCP affronta un ambiente completamente diverso. La funzione di densità di probabilità del tempo richiesto da un ack TCP per tornare indietro somiglia più alla figura 6-33 (b) che alla figura 6-33 (a). Determinare il tempo di andata e ritorno dalla destinazione è difficile. Anche quando questo è conosciuto, scegliere un intervallo di timeout è sempre complicato. Se il timeout scelto è troppo piccolo (ad es. T_1 in figura 6-33 (b)), avverranno ritrasmissioni non necessarie, che intaseranno la rete di pacchetti inutili. Se viene scelto troppo grande (T_2), le prestazioni ne risentiranno a causa dei lunghi ritardi di trasmissione ogni qual volta viene perso un pacchetto. Inoltre, la media e la varianza della distribuzione degli arrivi degli ack possono cambiare velocemente in pochi secondi con l'apparire e lo scomparire delle congestioni.

La soluzione consiste nell'utilizzare un algoritmo altamente dinamico che regoli costan-

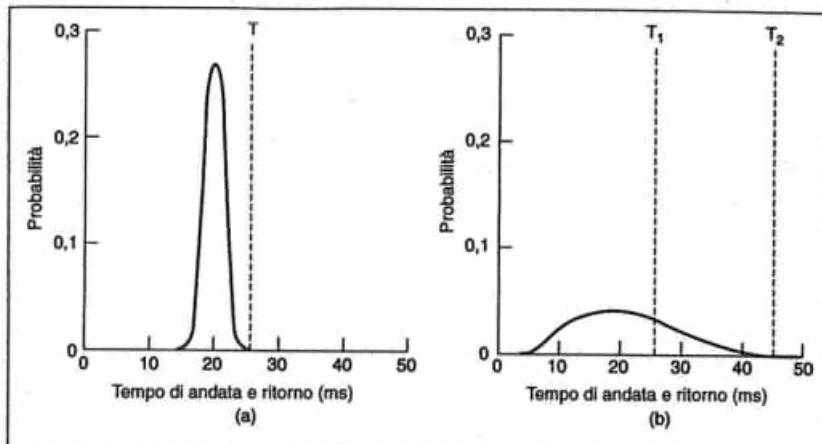


Fig. 6-33 (a) Densità di probabilità dei tempi di arrivo degli ack nel livello data link. (b) Densità di probabilità dei tempi di arrivo degli ack per TCP.

temente l'intervallo di timeout, basandosi sulle misurazioni continue delle prestazioni della rete. L'algoritmo generalmente usato per TCP è dovuto a Jacobson (1988) e funziona come segue. Per ogni connessione, TCP mantiene una variabile, RTT , che è la migliore stima attuale del tempo di andata e ritorno (round-trip time) dalla destinazione in questione. Quando viene spedito un segmento, si inizializza un timer, sia per misurare il tempo impiegato da un ack, sia per attivare una ritrasmissione nel caso impieghi troppo tempo. Se l'ack torna indietro prima della scadenza del timer, TCP misura il tempo M di andata e ritorno. Aggiorna quindi RTT tramite la seguente formula:

$$RTT = \alpha RTT + (1 - \alpha) M$$

dove α è un fattore di aggiustamento che determina il peso dato ai valori vecchi. Tipicamente $\alpha = 7/8$.

Anche in presenza di un buon valore di RTT , scegliere un timeout di ritrasmissione accettabile non è banale. Generalmente, TCP utilizza βRTT , ma il problema è scegliere β . Nelle realizzazioni iniziali, β era sempre uguale a 2, ma l'esperienza ha mostrato che un valore costante non è flessibile, in quanto non risponde al crescere della varianza. Nel 1988, Jacobson propose di rendere β all'incirca proporzionale alla deviazione standard della funzione di densità di probabilità dei tempi di arrivo degli ack; quindi, un valore grande della varianza implica un valore grande di β e viceversa. In particolare, suggerì di utilizzare la *deviazione media* come stima economica della *deviazione standard*. Il suo algoritmo richiede di tener traccia di un'ulteriore variabile di aggiustamento, la deviazione D . Ogni qual volta arriva un ack, viene calcolata la differenza tra il valore atteso e quello osservato $|RTT - M|$. Un valore arrotondato di questa viene mantenuto in D con la formula

$$D = \alpha D + (1 - \alpha) |RTT - M|$$

dove α può essere uguale al valore utilizzato per *RTT* oppure no. Sebbene *D* non sia esattamente uguale alla deviazione standard, è sufficientemente buono; Jacobson mostrò come questo valore può essere calcolato utilizzando solo somme, sottrazioni e shift interi. Gran parte delle implementazioni TCP utilizzano oggi questo algoritmo e scelgono come intervallo di timeout

$$\text{Timeout} = RTT + 4*D$$

La scelta del fattore 4 è in qualche modo arbitraria, ma ha due vantaggi. Innanzitutto, la moltiplicazione per 4 può essere eseguita con uno shift singolo. Secondo, minimizza i timeout non necessari e le relative ritrasmissioni in quanto meno dell'1% di tutti i pacchetti arriva più tardi della deviazione standard moltiplicata per 4. (Effettivamente, Jacobson disse inizialmente di utilizzare 2, ma in lavori successivi dimostrò che 4 dava prestazioni migliori).

Un problema che si presenta utilizzando la stima dinamica di *RTT* riguarda cosa fare quando un segmento va in timeout e viene ristretto. Quando arriva l'ack, non è chiaro se si riferisce alla prima trasmissione o a quella successiva. Una scelta sbagliata può contaminare seriamente la stima di *RTT*. Phil Karn scoprì questo problema con la pratica. Egli, un entusiasta radioamatore interessato alla trasmissione di pacchetti TCP/IP attraverso canali radio, un mezzo di trasmissione notoriamente inaffidabile (nelle giornate buone, vengono persi metà dei pacchetti), propose di non aggiornare *RTT* per qualsiasi pacchetto che fosse stato ritrasmesso. Al contrario, il timeout sarebbe stato raddoppiato ad ogni fallimento fino a quando il segmento non fosse passato. Questa correzione è nota come **algoritmo di Karn**, ed è utilizzata da molte implementazioni TCP.

Il time di ritrasmissione non è l'unico utilizzato da TCP. Il secondo è il **timer di persistenza**. Viene utilizzato per prevenire il seguente stallo. Il ricevente spedisce un ack con una dimensione di finestra uguale a 0, chiedendo al mittente di aspettare. Successivamente, il ricevente aggiorna la finestra, ma il pacchetto con l'aggiornamento viene perso. A questo punto sia il mittente che il ricevente stanno aspettando che l'altro faccia qualcosa. Quando il timer di persistenza scade, il mittente trasmette una sonda al ricevente. La risposta alla sonda contiene la dimensione della finestra. Se è ancora 0, il timer di persistenza viene fatto ripartire e il ciclo ricomincia. Se è diverso da 0, è possibile spedire dati.

Un terzo timer utilizzato da alcune implementazioni è il **keepalive timer**. Quando una connessione è rimasta inattiva per molto tempo, il keepalive timer può scadere per verificare che l'altro lato sia ancora presente. Se non vi è risposta, la connessione viene terminata. Questa caratteristica è controversa, in quanto aggiunge overhead e può terminare una connessione altrimenti sana a causa di una partizione temporanea della rete.

L'ultimo timer di ogni connessione TCP è quello utilizzato nello stato **TIMED_WAIT** durante la chiusura. La sua durata è due volte il tempo massimo di vita di un pacchetto; in questo modo si è sicuri che quando la connessione viene chiusa, non esiste più alcun pacchetto creato da essa.

6.4.8 UDP

Internet possiede anche un protocollo di trasporto senza connessione chiamato **UDP** (User Data Protocol – protocollo dati utenti). UDP fornisce un metodo per spedire datagram IP senza dover stabilire una connessione. Molte applicazioni client-server che prevedono una richiesta e una risposta utilizzano UDP piuttosto che perdere tempo a stabilire e in seguito a chiudere una connessione. UDP viene descritto in RFC 768.

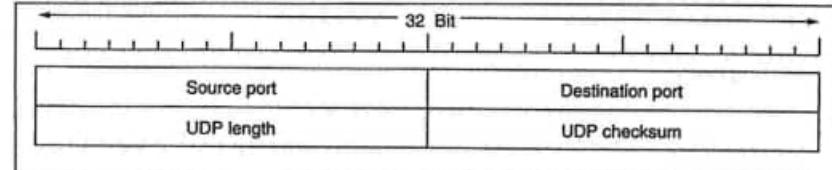


Fig. 6-34 Il preambolo UDP.

Un segmento UDP consiste in un preambolo di 8 byte seguito dai dati. Il preambolo è mostrato in figura 6-34. Le due porte hanno lo stesso significato che avevano in TCP: identificano i punti di accesso nelle macchine sorgente e destinazione. Il campo *UDP Length* (lunghezza UDP) include gli 8 byte del preambolo e i dati. Il campo *UDP checksum* comprende lo stesso pseudopreambolo mostrato in figura 6-25, il preambolo UDP e i dati UDP, il tutto completato a un numero di byte pari se necessario. È opzionale e viene memorizzato come 0 se non viene calcolato (se il calcolo desse valore 0, verrebbe registrato come un numero composto solo da 1). Disabilitare la checksum è stupido, a meno che la qualità dei dati non sia importante (ad es. voce digitalizzata).

6.4.9 TCP e UDP senza filo

In teoria, i protocolli di trasporto dovrebbero essere indipendenti dalla tecnologia del livello rete sottostante. In particolare, TCP non dovrebbe sapere se IP viene eseguito su una fibra o su un canale radio. In pratica, questo non è vero in quanto gran parte delle implementazioni TCP sono state ottimizzate in base ad assunzioni vere nel caso di reti cablate, che falliscono però nel caso di reti senza filo. Ignorare le proprietà di una trasmissione senza filo può portare a un'implementazione TCP logicamente corretta, ma con prestazioni orrende.

Il problema principale è l'algoritmo di controllo della congestione. Quasi tutte le implementazioni TCP attuali assumono che i timeout siano causati dalla congestione e non dai pacchetti persi. Di conseguenza, quando un timer scade, TCP rallenta e spedisce più tranquillamente (ad es. l'algoritmo slow start di Jacobson). L'idea dietro a questo approccio è quella di ridurre il carico della rete e quindi alleggerire la congestione.

Sfortunatamente, i canali di trasmissione senza filo sono altamente inaffidabili. Continuano a perdere pacchetti. L'approccio migliore nella gestione dei pacchetti persi è rispedirli e il più presto possibile. Rallentare fa peggiorare le cose. Se, per esempio, viene perso il 20% dei pacchetti, allora quando il mittente spedisce 100 pacchetti/s la velocità di trasmissione è di 80 pacchetti/s. Se il mittente rallenta a 50 pacchetti/s, la velocità di trasmissione si riduce a 40 pacchetti/s.

In effetti, nel caso di perdita di un pacchetto in una rete cablata, il mittente dovrebbe rallentare; nel caso di perdita di un pacchetto in una rete senza filo, il mittente dovrebbe aumentare il ritmo. Quando il mittente non conosce il tipo di rete, è difficile prendere la decisione giusta.

Frequentemente, il cammino dal mittente al ricevente non è omogeneo. I primi 1000 km potrebbero correre lungo una rete cablata, mentre l'ultimo chilometro potrebbe essere senza filo. A questo punto prendere la decisione corretta è ancora più difficile, in quanto dipende dal punto in cui si presenta il problema. Una soluzione proposta da Bakne e Badrinath (1995), che prende il nome di **TCP indiretto**, consiste nel suddividere la connessione TCP in due connessioni separate, come mostrato in figura 6-35. La prima connessione va dal mittente alla stazione base. La seconda va dalla stazione base al ricevente. La stazione base non fa altro che copiare i pacchetti tra le due connessioni in entrambe le direzioni.

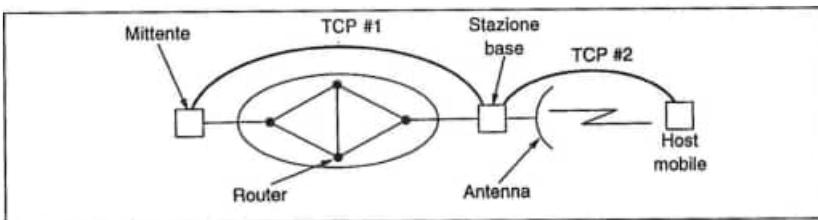


Fig. 6-35 Suddivisione di una connessione TCP in due connessioni.

Il vantaggio di questo schema è che entrambe le connessioni sono a questo punto omogenee. I timeout nella prima connessione possono indurre il mittente a rallentare, mentre i timeout nella seconda possono indurlo ad aumentare. Altri parametri possono essere scelti indipendentemente per ognuna delle due connessioni. Purtroppo, questo è contrario alla semantica di TCP. Poiché ognuna delle connessioni è di tipo TCP, la stazione base conferma ogni pacchetto TCP nel solito modo. Solo che ora se il mittente riceve un ack, questo non significa che il ricevente abbia ricevuto il segmento, ma solo che è stato ricevuto dalla stazione base.

Una soluzione differente, dovuta a Balakrishnan et al. (1995), non è contraria alla semantica TCP. Funziona facendo parecchie modifiche minori al codice del livello rete della stazione base. Uno dei cambiamenti è l'inserimento di un agente spia che osservi e memorizzi i segmenti TCP spediti all'host mobile, e confermi quelli di ritorno da esso. Quando un agente spia vede un segmento TCP indirizzato all'host mobile, ma non vede un ack prima che il tempo di timeout (relativamente breve) finisca, ritrasmette il segmento, senza informare la sorgente di quanto stia facendo. Un'ulteriore ritrasmissione viene generata quando vede passare un ack duplicato spedito dall'host mobile, in quanto ciò significa invariabilmente che l'utente mobile ha perso qualcosa. Gli ack duplicati vengono scartati sul posto, per evitare che la sorgente li interpreti come segnali di congestione. Lo svantaggio di questa trasparenza, tuttavia, è il fatto che se il canale senza filo fosse estremamente inaffidabile, la sorgente potrebbe andare in timeout aspettando un ack e

invocare l'algoritmo di controllo della congestione. Con TCP indiretto, tale algoritmo non viene mai invocato a meno che non ci sia effettivamente una congestione nella parte cablata della rete.

L'articolo di Balakrishnan et al. prevede anche una soluzione al problema dei segmenti persi originati dall'host mobile. Quando la stazione base nota un salto nei numeri di sequenza, genera una richiesta per una rispedizione selettiva dei byte mancanti utilizzando un'opzione TCP. Grazie a queste due correzioni, il canale senza filo viene reso più affidabile in entrambe le direzioni, senza che la sorgente debba sapere della sua esistenza e senza cambiare la semantica di TCP.

Sebbene UDP non soffra degli stessi problemi di TCP, le comunicazioni senza filo introducono delle difficoltà anche per esso. Il guaio principale è il fatto che i programmi che utilizzano UDP si aspettano che sia altamente affidabile. Sanno che non vi sono garanzie, ma si aspettano comunque che sia quasi perfetto. In un ambiente senza filo UDP è molto lontano dall'essere perfetto. Nel caso di programmi in grado di prendere provvedimenti riguardo ai messaggi UDP persi, ma a un prezzo considerevole, passare all'improvviso da un ambiente in cui messaggi possono essere persi raramente a uno in cui vengono persi costantemente può portare a un disastro nelle prestazioni.

Le comunicazioni senza filo influiscono non solo sulle prestazioni, ma anche in altre aree. Ad esempio, in che modo un host mobile individua la stampante locale a cui connettersi, invece di utilizzare la propria stampante base? Collegato a questo è il problema di prendere la pagina WWW della cella locale, anche se il suo nome è ignoto. Inoltre, i progettisti di pagine WWW tendono ad assumere che siano sempre disponibili grandi quantità di banda. Inserire un logo di grandi dimensioni in ogni pagina diventerebbe controproducente se fossero necessari 30 s di trasmissione alla velocità di 9600 bps ogni volta che la pagina venisse scaricata, irritando a non finire l'utente.

6.5 I protocolli del livello AAL di ATM

Non è completamente chiaro se ATM possiede o no uno strato trasporto. Da un lato, il livello ATM possiede le funzionalità di uno strato rete, ed esiste un ulteriore strato sopra di esso (AAL); ciò rende AAL uno strato trasporto. Alcuni esperti condividono questa visione (ad es. De Prycker, 1993, p. 112). Uno dei protocolli utilizzati (AAL 5) possiede funzionalità simili a UDP, che è senza dubbio un protocollo di trasporto.

D'altra parte, nessuno dei protocolli AAL fornisce una connessione affidabile end-to-end come TCP (anche se con pochissime modifiche potrebbero farlo). Inoltre, in gran parte delle applicazioni viene utilizzato uno strato trasporto sopra AAL. Piuttosto di tentare di spaccare un capello in quattro, discuteremo il livello AAL e i suoi protocolli in questo capitolo senza però affermare che sia effettivamente uno strato trasporto.

Il livello AAL nelle reti ATM è radicalmente differente da TCP, in gran parte perché i progettisti erano interessati principalmente alla trasmissione di voce e video, per la quale la consegna rapida è più importante della consegna accurata. È importante ricordare che il livello ATM non fa altro che mandare in output celle di 53 byte una dopo l'altra. Non ha controllo di errore, controllo di flusso e nessun altro tipo di controllo. Conseguentemente, non è adatto ai requisiti di gran parte delle applicazioni.

Per colmare questo divario, nella raccomandazione I.363, ITU definì uno strato end-to-end sopra il livello ATM. Questo strato, chiamato **AAL** (ATM Adaptation Layer – strato ATM di adattamento) ha una storia complessa, piena di errori, revisioni e intrighi infiniti. Nei prossimi paragrafi analizzeremo la sua progettazione.

L'obiettivo di AAL è quello di fornire servizi utili alle applicazioni e di schermarle dal meccanismo di suddivisione dei dati in celle alla sorgente e da quello di riassemblamento alla destinazione. Quando ITU iniziò a definire AAL, comprese che applicazioni differenti hanno requisiti distinti; per questo motivo organizzò lo spazio dei servizi lungo tre assi:

1. Servizi in tempo reale contro servizi non in tempo reale.
2. Servizi con velocità di trasmissione costante contro servizi a velocità variabile.
3. Servizi orientati alla connessione contro servizi senza connessione.

In linea di principio, dati tre assi e due valori per ogni asse, potrebbero essere definiti otto distinti servizi, come mostrato in figura 6-36. ITU pensò che solo quattro di essi avessero qualche utilità, e li chiamò rispettivamente A, B, C e D. Gli altri non sono supportati. A partire da ATM 4.0, la figura 6-36 è diventata obsoleta; è stata presentata qui per background culturale, in modo da aiutare a comprendere perché i protocolli AAL siano stati progettati in questa maniera. Al posto di queste classi di servizio, attualmente la distinzione principale si basa sulle classi di traffico che abbiamo studiato nel capitolo 5 (ABR, CBR, NRT-VBR, RT-VBR e UBR).

	A		B		C		D	
Temporizzazione	Real time	Nessuna	Real time	Nessuna	Real time	Nessuna	Real time	Nessuna
Velocità dei bit	Costante		Variabile		Costante		Variabile	
Modalità	Orientato alla connessione				Senza connessione			

Fig. 6-36 Le classi di servizio originali supportati da AAL (ora obsolete).

Per gestire queste quattro classi di servizio, ITU definì quattro protocolli, denominati rispettivamente AAL 1, AAL 2, AAL 3, AAL 4. Tuttavia, in seguito si scoprì che i requisiti tecnici per le classi C e D erano così simili che AAL 3 e AAL 4 furono combinati in AAL 3/4. A questo punto l'industria dei calcolatori, che non se ne era occupata molto, comprese che nessuno di essi era veramente soddisfacente. Risolse il problema definendo un ulteriore protocollo, AAL 5. Analizzeremo brevemente tutti questi protocolli. Daremo anche un'occhiata a un interessante protocollo di controllo utilizzato nei sistemi ATM.

6.5.1 Struttura del livello AAL

Il livello AAL è suddiviso in due sezioni principali, una delle quali è ulteriormente suddivisa, come illustrato in figura 6-37.

La parte superiore del livello AAL viene chiamata **convergence sublayer** (sottostrato di convergenza), detto anche CS. Il suo compito è fornire l'interfaccia per le applicazioni. Consiste in una sottoparte comune a tutte le applicazioni (per un dato protocollo AAL) e in una sottoparte specifica dell'applicazione. Le funzioni di ognuna di queste parti dipendono dal protocollo, ma possono includere suddivisione dei messaggi e rilevamento degli errori.

Inoltre, il sottostrato di convergenza alla sorgente è responsabile dell'accettazione dei flussi di bit o di messaggi di lunghezza arbitraria da parte dell'applicazione e della suddivisione in unità da 44 a 48 byte per la trasmissione. La quantità esatta dipende dai protocolli, in quanto alcuni di essi utilizzano parte dei 48 byte del contenuto di una cella ATM per i propri preamboli. Alla destinazione, questo sottostrato riassembra le celle nei messaggi originali. In generale, i confini dei messaggi (se presenti) vengono preservati. In altre parole, se la sorgente spedisce quattro messaggi di 512 byte, questi verranno ricevuti come quattro messaggi di 512 byte e non come uno di 2048. Per i flussi di dati, non esistono confini fra i messaggi, che quindi non vengono preservati.

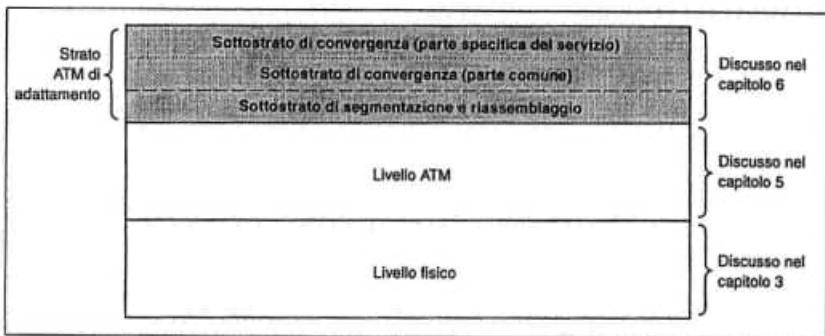


Fig. 6-37 Il modello ATM che illustra il livello AAL e i suoi sottostrati.

La parte inferiore di AAL viene detta sottostrato **SAR** (Segmentation and Reassembly – segmentazione e riassemblamento). Può aggiungere preamboli e code alle unità di dati ricevute dal sottostrato di convergenza per formare il contenuto delle celle. Questo contenuto viene quindi passato al livello ATM per la trasmissione. Alla destinazione, il sottostrato SAR riassembra le celle in messaggi. Il sottostrato SAR si occupa principalmente di celle, mentre il sottostrato di convergenza si occupa di messaggi.

Il funzionamento generico dei sottostrati CS e SAR è illustrato in figura 6-38. Quando l'applicazione consegna ad AAL un messaggio, il livello di convergenza può aggiungere un preambolo e/o una coda. Il messaggio viene quindi spezzato in unità lunghe da 44 a 48 byte, che vengono passate al sottostrato SAR. Il livello SAR può aggiungere il proprio

preambolo e la propria coda ad ognuno dei pezzi e passarli al livello ATM sottostante per la trasmissione in celle indipendenti. Si noti che la figura mostra il caso più generale, in quanto alcuni protocolli AAL non hanno preambolo e/o coda.

Per alcune classi di servizio (ma non tutte), il sottostrato SAR possiede anche alcune funzioni addizionali. In particolare, in qualche caso gestisce il rilevamento degli errori e il multiplexing. Il sottostrato SAR è presente per tutte le classi di servizio, ma è più o meno importante a seconda del protocollo specifico.

Le comunicazioni tra l'applicazione e il livello ATM utilizzano le primitive OSI standard *request* (richiesta) e *indication* (indicazione) che abbiamo discusso nel capitolo 1. Le comunicazioni tra sottostrati utilizzano primitive diverse.

6.5.2 AAL 1

AAL 1 è il protocollo utilizzato per trasmettere traffico di classe A, ovvero in tempo reale, a velocità costante e orientato alla connessione, come ad esempio audio e video non compressi.

I bit vengono introdotti dall'applicazione a una velocità costante e devono essere consegnati all'altro estremo alla stessa velocità costante, con un ritardo, un jitter e un overhead minimo. L'input è un flusso di bit, senza confini fra messaggi. Per questo tipo di traffico, protocolli che rilevano gli errori come stop-and-wait non sono utilizzabili, in quanto i ritardi introdotti da timeout e ritrasmissioni sono inaccettabili. Tuttavia, le celle mancanti vengono notificate all'applicazione, che può prendere gli opportuni provvedimenti per risolvere il problema.

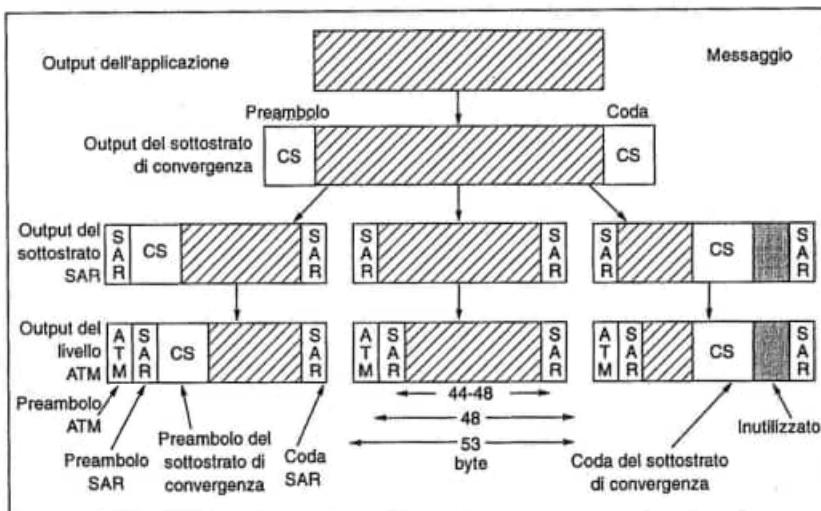


Fig. 6-38 I preamboli e le code che possono essere aggiunti a un messaggio nelle reti ATM.

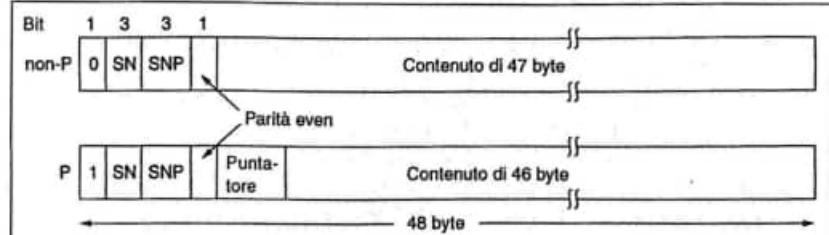


Fig. 6-39 Il formato delle celle AAL 1.

AAL 1 utilizza un sottostrato di convergenza e un sottostrato SAR. Il sottostrato di convergenza rileva celle perse o mal inserite (una cella mal inserita è consegnata alla destinazione sbagliata in seguito a un errore non rilevato nell'identificatore del circuito o percorso virtuale). Regolarizza inoltre il traffico in entrata per garantire la consegna delle celle a una velocità costante. Infine, il sottostrato di convergenza spezza i messaggi o i flussi di input in unità di 46 o 47 byte che vengono consegnate al sottostrato SAR per la trasmissione. All'altro capo queste celle vengono estratte e si ricostruisce l'input originale. Il sottostrato di convergenza non possiede nessun preambolo di protocollo.

Al contrario, il sottostrato SAR di AAL1 possiede un preambolo. I formati delle sue celle sono mostrati in figura 6-39. Entrambi i formati iniziano con un preambolo di 1 byte contenente un numero di sequenza di celle a 3 bit, SN (per rilevare celle mancanti o mal inserite). Questo campo è seguito da una protezione del numero di sequenza a 3 bit (ad es. una checksum) sul numero di sequenza, al fine di consentire la correzione di un singolo errore e il rilevamento di errori doppi nel campo di sequenza. Utilizza un controllo a ridondanza ciclica basata sul polinomio $x^3 + x + 1$. Un bit di parità che copre il byte di preambolo riduce ulteriormente la probabilità di un numero di sequenza errato non rilevato. Le celle AAL 1 non hanno bisogno di essere lunghe esattamente 47 byte. Per esempio, per trasmettere voce digitalizzata alla velocità di 1 byte ogni 125 μs, riempire una cella di 47 byte significa raggruppare campionamenti per 5,875 ms. Se questo ritardo prima della trasmissione è inaccettabile, possono essere spedite celle parziali. In questo caso, il numero di byte effettivi di dati per cella è lo stesso in tutte le celle e viene accordato in anticipo.

Le celle P vengono utilizzate quando i confini dei messaggi devono essere preservati. Il campo Pointer (puntatore) viene utilizzato per dare lo scostamento dell'inizio del prossimo messaggio. Solo celle con un numero di sequenza pari possono essere celle P, e quindi il puntatore è compreso tra 0 e 92, in modo che sia compreso nel contenuto della propria cella o di quella successiva. Si noti che questo schema consente di utilizzare messaggi di lunghezza arbitraria, che possono essere spediti senza interruzioni e non hanno bisogno di essere allineati ai confini delle celle. Il bit di ordine più alto del campo Pointer è riservato per usi futuri. Il bit iniziale del preambolo di tutte le celle dispari forma un flusso di dati utilizzato per la sincronizzazione degli orologi.

6.5.3 AAL 2

AAL 1 è stato progettato per flussi di dati semplici, orientati alla connessione, senza controllo degli errori, eccetto il caso di celle mancanti o mal inserite. Per canali audio e video non compressi, o qualsiasi altro flusso di dati in cui qualche bit errato di tanto in tanto non costituisce un problema, AAL 1 è adeguato.

Per canali audio o video compressi, la quantità di dati spediti può variare molto nel tempo. Ad esempio, molti schemi di compressione trasmettono periodicamente un quadro video completo, dopo di che spediscono per molti quadri solo le differenze tra i quadri successivi e l'ultimo quadro intero. Quando la telecamera è stabile e non c'è niente in movimento, i quadri differenza sono piccoli, ma quando la cinepresa sta carrellando rapidamente possono diventare molto grandi. Inoltre, è necessario preservare i confini dei messaggi in modo che l'inizio del prossimo quadro intero possa essere riconosciuto, anche in presenza di celle perse o di dati non corretti. Per questa ragione è necessario un protocollo più elaborato. AAL è stato progettato per questo scopo.

Come in AAL 1, il sottostrato CS non possiede un protocollo, a differenza del sottostrato SAR. Il formato delle celle SAR è mostrato in figura 6-40. È costituito da un preambolo di 1 byte e da una coda di 2 byte, lasciando spazio per 45 byte di dati per cella.

Il campo *SN* (*Sequence Number* – numero di sequenza) viene utilizzato per numerare le celle al fine di individuare celle mancanti o mal inserite. Il campo *IT* (*Information Type* – tipo di informazione) viene utilizzato per indicare che la cella costituisce l'inizio, il corpo o la fine di un messaggio. Il campo *LI* (*Length Indicator* – indicatore di lunghezza) contiene la lunghezza del contenuto della cella in byte (potrebbe essere minore di 45 byte). Infine, il campo *CRC* è una checksum sull'intera cella, per individuare gli errori.

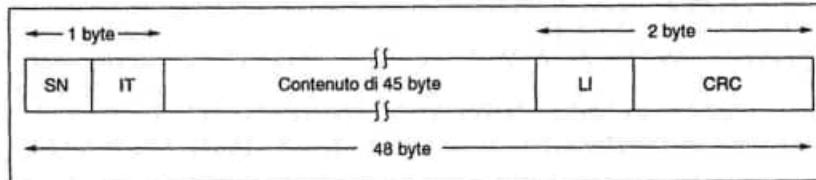


Fig. 6-40 Il formato delle celle AAL 2.

Per quanto strano possa sembrare, le dimensioni dei campi non sono inclusi nello standard. Secondo una delle persone coinvolte, al termine del processo di standardizzazione il comitato comprese che AAL 2 aveva così tanti problemi da essere inutilizzabile. Sfortunatamente, era troppo tardi per fermare il processo di standardizzazione. Avevano una scadenza da rispettare. In un ultimo tentativo disperato, il comitato rimosse tutte le dimensioni dei campi in modo che lo standard formale potesse essere pubblicato in tempo, ma in modo tale che nessuno potesse utilizzarlo effettivamente. Questa è la vita nel mondo della standardizzazione.

6.5.4 AAL 3/4

Originariamente, ITU possedeva protocolli differenti per le classi C e D, rispettivamente

un servizio di trasporto orientato alla connessione e un servizio di trasporto senza connessione, sensibili alla perdita di dati o agli errori, ma indipendenti dal tempo. Successivamente ITU scoprì che non c'era un bisogno reale di due protocolli, che quindi vennero combinati in un unico protocollo, AAL 3/4.

AAL 3/4 può operare in due modalità: flusso o messaggi. In modalità messaggi, ogni chiamata a AAL 3/4 da parte dell'applicazione inserisce un messaggio nella rete. Il messaggio viene consegnato senza modifiche, preservando i confini dei messaggi. In modalità flusso i confini non vengono rispettati. La discussione sottostante si concentrerà sulla modalità messaggi. In entrambe le modalità sono disponibili un trasporto affidabile e uno non affidabile (cioè senza garanzie).

Una caratteristica di AAL 3/4 non presente in altri protocolli è il multiplexing. Questo aspetto di AAL 3/4 consente di far viaggiare più sessioni (ad es. login remoti) da una singola sorgente lungo lo stesso circuito virtuale, per separarle alla destinazione, come illustrato in figura 6-41.

Il motivo per cui questo servizio è desiderabile deve essere cercato nel sistema di tariffazione: spesso i fornitori fanno pagare per ogni creazione di connessione e per ogni secondo in cui la connessione resta aperta. Se una coppia di host avesse molte connessioni aperte simultaneamente, assegnare ad ognuna il proprio circuito virtuale sarebbe più costoso che spartire fra di esse lo stesso circuito virtuale. Se un circuito virtuale ha banda sufficiente per gestire il carico, non è necessario averne più di uno. Tutte le sessioni che utilizzano lo stesso circuito virtuale ottengono la stessa qualità di servizio, in quanto questa viene negoziata alla creazione del circuito virtuale.

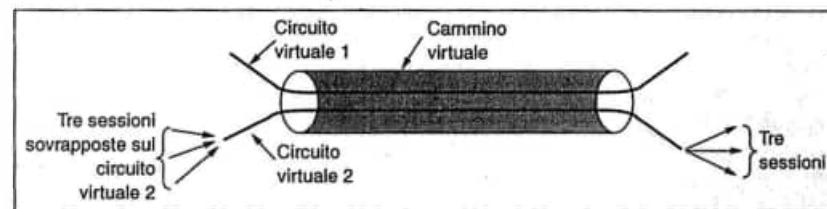


Fig. 6-41 Multiplexing di un certo numero di sessioni su un circuito virtuale.

Questa caratteristica è il motivo reale per cui inizialmente c'erano due formati separati AAL 3 e AAL 4: a differenza degli europei, gli americani volevano il multiplexing. I gruppi si separarono e crearono i loro standard. Alla fine, gli europei decisero che risparmiare 10 bit del preambolo non fosse un motivo valido per rinunciare alla compatibilità fra Stati Uniti ed Europa. Per lo stesso motivo, avrebbero potuto rimanere fermi sulle loro posizioni, e avremmo quattro standard AAL incompatibili (uno dei quali non funziona) al posto di tre.

A differenza di AAL 1 e AAL 2, AAL 3/4 possiede sia un protocollo del sottostrato di convergenza che un protocollo del sottostrato SAR. Messaggi grandi fino a 65.535 byte vengono consegnati dall'applicazione al sottostrato di convergenza. Innanzitutto si aggiunge un certo numero di byte fino a raggiungere un multiplo di 4 byte. Quindi vengono aggiunti un preambolo e una coda, come mostrato in figura 6-42.

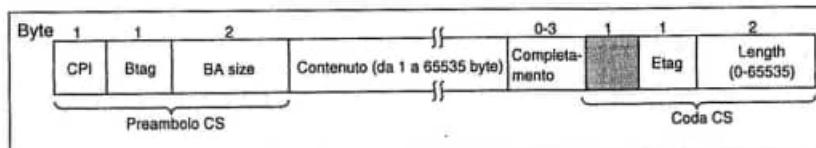


Fig. 6-42 Il formato di messaggio del sottostato di convergenza di AAL 3/4.

Il campo *CPI* (Common Part Indicator – indicatore della parte comune) contiene il tipo di messaggio e le unità utilizzate per i campi *BA size* (dimensione dell'allocazione dei buffer) e *Length* (lunghezza). I campi *Btag* e *Etag* vengono utilizzati per frammentare i messaggi. I 2 byte devono essere uguali e vengono incrementati di 1 ad ogni nuovo messaggio spedito. Questo meccanismo verifica che le celle non vengano perse o mal inserite. Il campo *BA size* viene utilizzato per l'allocazione dei buffer. Indica al ricevente quanto spazio allocare per il messaggio prima del suo arrivo. Il campo *Length* fornisce ancora una volta la lunghezza del messaggio. In modalità messaggio, *Length* e *BA size* devono essere uguali, mentre in modalità flusso possono essere differenti. La coda contiene anche 1 byte inutilizzato.

Dopo che il sottostato di convergenza ha costruito e aggiunto un preambolo e una coda al messaggio, come mostrato in figura 6-42, quest'ultimo viene passato al sottostato SAR, il quale lo spezza in sezioni di 44 byte. Si noti che per supportare il multiplexing, il sottostato di convergenza potrebbe aver costruito internamente molti messaggi in una sola volta, ma potrebbe passare sezioni di 44 byte al sottostato SAR prima da un messaggio, poi da un altro, in qualsiasi ordine.

Il sottostato SAR inserisce ogni sezione di 44 byte nelle celle il cui formato è visualizzato in figura 6-43. Queste celle vengono quindi trasmesse alla destinazione per il riassemblaggio, dopo di che viene controllata la checksum e in caso di necessità vengono presi gli opportuni provvedimenti.

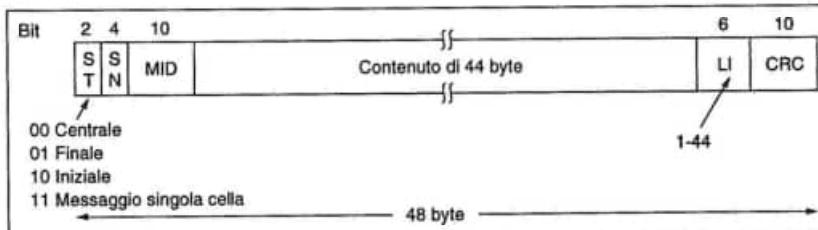


Fig. 6-43 Il formato delle celle AAL 3/4.

Le celle AAL 3/4 contengono i seguenti campi. Il campo *ST* (Segment Type – tipo di segmento) viene utilizzato per la frammentazione dei messaggi. Indica se la cella è all'inizio del messaggio, al centro, alla fine, oppure se il messaggio è piccolo (cioè singola cella). Vengono quindi 4 bit di numero di sequenza, *SN*, per l'individuazione di celle mancanti o mal inserite. Il campo *MID* (Multiplexing ID) viene utilizzato per identificare

6.5 I protocolli del livello AAL di ATM

a quale sessione appartenga la cella. Si ricordi che il sottostato di convergenza può conservare nei buffer molti messaggi contemporaneamente, appartenenti a sessioni differenti, e può spedire sezioni di questi messaggi nell'ordine che preferisce. Tutte le sezioni dei messaggi appartenenti alla sessione *i* trasportano il valore *i* nel campo *MID*, in modo che i messaggi possano essere riassemblati correttamente alla destinazione. La coda contiene la lunghezza del contenuto e la checksum della cella.

Si noti che AAL 3/4 ha due livelli di overhead di protocollo: vengono aggiunti 8 byte per ogni messaggio e 4 per ogni cella. Alla fine è un meccanismo pesante, specialmente per messaggi brevi.

6.5.5 AAL 5

I protocolli AAL 1-AAL 3/4 furono progettati dall'industria delle telecomunicazioni e standardizzati dall'ITU senza vera partecipazione da parte dell'industria informatica. Quando questa industria si svegliò e iniziò a capire le implicazioni di figura 6-43, si diffuse un senso di panico. La complessità e l'inefficienza generate dai due livelli di protocollo, combinate con la checksum sorprendentemente corta (solo 10 bit), obbligò alcuni ricercatori a inventare un nuovo protocollo di adattamento. Fu chiamato **SEAL** (Simple Efficient Adaptation Layer – protocollo di adattamento semplice ed efficiente), e questo dimostra cosa pensavano i progettisti dei protocolli precedenti. Dopo alcune discussioni, il Forum ATM accettò SEAL e gli assegnò il nome AAL 5. Per maggiori informazioni su AAL 5 e sulle sue differenze con AAL 3/4, si veda Suzuki (1994).

AAL 5 offre alle applicazioni un gran numero di tipologie di servizio diverse. Una delle possibilità è data da un servizio affidabile (cioè consegna garantita con controllo di flusso per evitare straripamenti).

Un'altra possibilità è data da un servizio inaffidabile (cioè senza consegna garantita), con la possibilità di scegliere se le celle con errori di checksum debbano essere scartate oppure passate all'applicazione in ogni caso (sebbene marcate come errate). Sono supportate sia le trasmissioni punto-a-punto che quelle multicast, sebbene queste ultime non assicurino la consegna garantita.

Come AAL 3/4, AAL 5 supporta sia la modalità messaggi che la modalità flusso. In modalità messaggi, un'applicazione può passare al livello AAL un datagram di lunghezza compresa fra 1 e 65.535; questo verrà consegnato alla destinazione, in base a un servizio best effort oppure con consegna garantita. All'arrivo nel sottostato di convergenza, un messaggio viene completato e gli viene aggiunta una coda, come mostrato in figura 6-44. Il numero di byte di completamento (da 0 a 47) viene scelto in modo che l'intero messaggio (coda compresa) sia un multiplo di 48 byte. AAL 5 non possiede un preambolo del sottostato di convergenza, ma solo una coda di 8 byte.

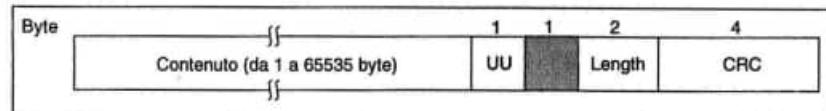


Fig. 6-44 Formato dei messaggi del sottostato di convergenza di AAL 5.

Il campo *UU* (*User to User* – da utente a utente) non viene utilizzato dal sottostrato AAL stesso. È invece disponibile per gli strati superiori per i propri scopi, come ad esempio l'introduzione di numeri di sequenza o per il multiplexing. Il livello superiore in questione potrebbe essere la parte del sottostrato di convergenza specifica per il tipo di servizio. Il campo *Length* contiene la lunghezza del contenuto in byte, senza contare il completamento. Un valore 0 viene utilizzato per abortire il messaggio corrente a metà strada. Il campo *CRC* è la checksum a 32 bit dell'intero contenuto del messaggio, incluso il completamento e la coda (con il campo *CRC* posto uguale a 0). Un campo di 8 bit della coda è riservato per usi futuri.

Il messaggio viene trasmesso passandolo al sottostrato SAR, che non aggiunge nessun preambolo o coda. Spezza invece il messaggio in unità di 48 byte e le passa al livello ATM per la trasmissione. Informa inoltre il livello ATM di accendere 1 bit nel campo *PTI* dell'ultima cella, in modo da conservare i confini dei messaggi. Si potrebbe sollevare un caso su queste interazioni scorrette fra i protocolli di strati diversi, in quanto il livello AAL non dovrebbe utilizzare bit del preambolo del livello ATM. Questo è contro gran parte dei principi di ingegneria dei protocolli, e suggerisce che forse la struttura degli strati sarebbe dovuta essere diversa.

Il vantaggio principale di AAL 5 rispetto ad AAL 3/4 è la sua efficienza. Sebbene AAL 3/4 aggiunga solo 4 byte per messaggio, aggiunge anche 4 byte per ogni cella, riducendo la capacità delle celle a 44 byte, una perdita dell'8% sui messaggi lunghi. AAL 5 possiede una coda leggermente più lunga (8 byte), ma non appesantisce le celle. La mancanza di numeri di sequenza nelle celle è compensata dalla lunghezza della checksum che può individuare messaggi persi, mal inseriti o celle mancanti senza utilizzare numeri di sequenza.

La comunità Internet si aspetta che la modalità normale per interfacciarsi alle reti ATM sia quella di trasportare i pacchetti IP nel contenuto dei messaggi AAL 5. Varie problematiche relative a questo approccio vengono discusse in RFC 1483 ed RFC 1577.

6.5.6 Confronto dei protocolli AAL

Non bisogna preoccuparsi se si è dell'opinione che i vari protocolli AAL sembrano assurdamente simili l'uno all'altro e mal congegnati. Anche il vantaggio di avere due sottostrati distinti è discutibile, specialmente perché il sottostrato SAR di AAL 5 è vuoto. Un preambolo ATM leggermente migliorato avrebbe potuto fornire un supporto adeguato per i numeri di sequenza, il multiplexing e la frammentazione.

In figura 6-45 sono elencate alcune differenze tra i vari protocolli AAL, relativamente all'efficienza, alla gestione degli errori, al multiplexing e alla relazione tra i sottostrati AAL.

L'impressione generale che si riceve da AAL è quella di un lavoro compiuto a metà, con troppe varianti e differenze di minor importanza. Le quattro classi di servizio originali, A, B, C, D sono state effettivamente abbandonate. Probabilmente, AAL 1 non è realmente necessario; AAL 2 non funziona; AAL 3 e AAL 4 non hanno mai visto la luce del giorno; AAL 3/4 è inefficiente e ha una checksum troppo breve.

AAL 5 è il futuro, ma anche in questo caso c'è spazio per miglioramenti. Per poter essere utilizzato come protocollo di trasporto affidabile, i messaggi AAL 5 dovrebbero contenere

Caratteristica	AAL 1	AAL 2	AAL 3/4	AAL 5
Classe del servizio	A	B	C/D	C/D
Multiplexing	No	No	Si	No
Delimitazione dei messaggi	Nessuno	Nessuno	Btag/Etag	Bit in PTI
Allocazione anticipata dei buffer	No	No	Si	No
Byte utente disponibili	0	0	0	1
Completamento CS	0	0	Parola di 32 bit	0-47 byte
Overhead del protocollo CS (byte)	0	0	8	8
Checksum CS	Nessuno	Nessuno	Nessuno	32 bit
Byte di contenuto SAR	46-47	45	44	48
Overhead del protocollo SAR (byte)	1-2	3	4	0
Checksum CS	Nessuno	Nessuno	10 Bit	Nessuno

Fig. 6-45 Alcune differenze tra i vari protocolli AAL.

un numero di sequenza e 1 bit per distinguere i messaggi normali da quelli di controllo. È ancora disponibile spazio inutilizzato nella coda. Attualmente, affinché sia utilizzato come trasporto affidabile è necessario l'overhead aggiuntivo di uno strato trasporto sopra di esso, che altrimenti potrebbe essere evitato. Se l'intero comitato AAL avesse presentato il proprio lavoro come progetto d'esame, il professore lo avrebbe probabilmente restituito con il suggerimento di correggerlo e di ripresentarlo nuovamente quando fosse finito. Per ulteriori critiche, si vedano Sternbenz et al. (1995).

6.5.7 SSCOP

Nonostante questo gran numero di protocolli differenti, nessuno di essi fornisce connessioni di trasporto affidabili e punto-a-punto. Per applicazioni con queste esigenze, esiste un altro protocollo AAL: **SSCOP** (**S**ervice **S**pecific **C**onnection-**O**riented **P**rotocol – protocollo di servizio specifico orientato alla connessione), che però viene utilizzato solo per le trasmissioni di controllo e non per le trasmissioni dati.

Gli utenti di SSCOP spediscono messaggi, ad ognuno dei quali viene assegnato un numero di sequenza di 24 bit. I messaggi possono arrivare fino a 64K e non vengono frammentati. Devono essere consegnati nell'ordine. A differenza di altri protocolli di trasporto affidabili, i messaggi mancanti vengono ritrasmessi utilizzando sempre selective repeat invece di go back n.

SSCOP è fondamentalmente un protocollo sliding window dinamico. Per ognuna delle connessioni, il ricevente mantiene una finestra di numeri di sequenza che è pronto a ricevere, e una mappa di bit per individuare quelli che ha già ricevuto. La finestra può cambiare dimensione durante il funzionamento del protocollo.

Ciò che rende SSCOP diverso da altri protocolli è il modo in cui gli ack vengono gestiti:

non vengono inseriti nel traffico di ritorno. Al contrario, il mittente interroga periodicamente il ricevente e richiede l'invio della mappa di bit contenente lo stato della finestra. In base a questo risultato, il mittente scarta i messaggi che sono stati accettati e aggiorna la sua finestra. SSCOP viene descritto dettagliatamente in Henderson (1995).

6.6 Problematiche relative alle prestazioni

I problemi relativi alle prestazioni sono veramente importanti nelle reti di computer. Quando centinaia o migliaia di calcolatori sono connessi fra loro, è possibile che si producano interazioni complesse con conseguenze imprevedibili. Spesso questa complessità conduce a basse prestazioni, senza che nessuno ne conosca il motivo. Nei paragrafi seguenti esamineremo molte delle problematiche relative alle prestazioni, per vedere quali sono le tipologie di problemi esistenti e come possono essere affrontate.

Sfortunatamente, comprendere le prestazioni di una rete è più un'arte che una scienza. Esiste poca teoria che sia effettivamente utilizzabile in pratica.

Il meglio che possiamo fare è dare alcune regole pratiche ottenute dall'esperienza e presentare esempi presi dal mondo reale. Abbiamo posticipato questa discussione successivamente allo studio del livello trasporto in Internet e nelle reti ATM, per poter sottolineare in questi sistemi le scelte corrette e quelle sbagliate.

I problemi relativi alle prestazioni non si presentano solo nel livello trasporto. Ne abbiamo visti alcuni nella descrizione del livello rete nel capitolo precedente. Ciò nonostante, il livello rete tende a occuparsi per lo più di routing e di controllo della congestione. Problematiche più vaste e orientate al sistema tendono ad essere collegate al trasporto, e quindi questo capitolo è il luogo adatto per esaminarle.

Nei prossimi cinque paragrafi analizzeremo cinque aspetti delle prestazioni di rete:

1. Problemi di prestazioni.
2. Misurazione delle prestazioni di rete.
3. Progettazione di sistema per migliorare le prestazioni.
4. Elaborazione veloce dei TPDU.
5. Protocolli per le future reti ad alta velocità.

Fra le altre cose, abbiamo bisogno di un nome per le unità scambiate dalle entità di trasporto. Il termine segmento genera confusione e non viene usato in questo contesto esternamente al mondo TCP. I termini propri di ATM, CS-PDU, SAR-PDU e CPCS-PDU sono troppo specifici. Il termine pacchetto si riferisce chiaramente al livello rete, mentre i messaggi appartengono al livello applicazione. Per mancanza di un termine standard, torneremo indietro e chiameremo le unità scambiate dalle entità di trasporto TPDU. Quando parleremo contemporaneamente di TPDU e pacchetti, utilizzeremo il termine collettivo pacchetto (ad es. in "La CPU deve essere sufficientemente veloce da elaborare i pacchetti in tempo reale", intendiamo sia i pacchetti del livello rete che i TPDU encapsulati in essi).

6.6.1 Problemi di prestazione nelle reti di computer

Alcuni dei problemi relativi alle prestazioni, come ad esempio le congestioni, sono causati da un temporaneo sovraccarico delle risorse. Se a un router arrivasse improvvisamente più traffico di quanto ne possa gestire, si presenterebbe una congestione e le prestazioni ne soffrirebbero. Abbiamo studiato dettagliatamente la congestione nel capitolo precedente. Le prestazioni degradano anche in presenza di uno sbilanciamento strutturale delle risorse. Ad esempio, se una linea di comunicazione da 1 Gb è attaccata a un PC di basso profilo, la povera CPU non sarà in grado di elaborare i pacchetti in arrivo con velocità sufficiente, e alcuni di essi verranno persi. Questi pacchetti verranno prima o poi ritrasmessi, aumentando i ritardi, sprecando risorse e in generale riducendo le prestazioni.

I sovraccarichi possono essere attivati anche in modo sincrono. Ad esempio, se un TPDU contiene un parametro errato (ad es. la porta del processo destinazione), in molti casi il ricevente risponderà gentilmente con una notifica di errore. A questo punto, si consideri cosa succede se un TPDU errato viene mandato in broadcast a 10.000 macchine: ognuna di esse risponderà con un messaggio di errore. La tempesta di broadcast risultante potrebbe bloccare la rete. UDP soffre di questo problema finché il protocollo è stato cambiato in modo che gli host non rispondano agli errori trovati nei segmenti TCP spediti a indirizzi di broadcast.

Un secondo esempio di sovraccarico sincrono è collegato ai black-out elettrici. Quando l'elettricità ritorna, tutte le macchine saltano simultaneamente alle loro ROM per iniziare a eseguire il reboot. Una tipica sequenza di reboot potrebbe prevedere innanzitutto la richiesta della propria identità a un server RARP, e quindi il collegamento a qualche file server per ottenere una copia del proprio sistema operativo. Se tutto ciò venisse eseguito da centinaia di macchine contemporaneamente, il server probabilmente collasserebbe sotto il carico.

Anche in assenza di sovraccarichi sincroni e in presenza di risorse sufficienti, si possono ottenere prestazioni scadenti a causa della mancanza di sintonia fra le varie parti del sistema. Ad esempio, se una macchina dispone di grandi capacità di memoria e di CPU, ma ai buffer è stata dedicata poca memoria, i sovraccarichi si presenteranno comunque e qualche TPDU verrà perso. Un altro problema di sintonia è la scelta dei timeout corretti. Quando viene spedito un TPDU, generalmente viene inizializzato un timer per premunirsi in caso di perdita. Se il timeout scelto è troppo breve, verranno effettuate ritrasmissioni non necessarie, intasando i collegamenti. Se il timeout scelto è troppo grande, ci saranno ritardi eccessivamente lunghi dopo la perdita di un TPDU. Altri parametri regolabili sono il tempo di attesa prima di spedire un ack separato e il numero di ritrasmissioni prima di smettere.

Le reti ad alta velocità danno origine a nuovi problemi di prestazione. Si consideri, ad esempio, la spedizione di dati da San Diego a Boston nel caso che il buffer del ricevente sia di 64 KB. Si supponga che la velocità del collegamento sia di 1 Gbps e che il ritardo indotto dalla fibra sia di 20 ms (solo andata). Inizialmente, al tempo $t = 0$, il canale è vuoto, come illustrato in figura 6-46 (a). Dopo soli 500 μ s (figura 6-46 (b)), tutti i TPDU sono spediti fuori sulla fibra. Il TPDU di testa dovrebbe essere da qualche parte nelle vicinanze di Brawley, nella California del Sud. Tuttavia, il mittente deve smettere in attesa di un aggiornamento della finestra.

Dopo 20 ms, il TPDU di testa raggiunge Boston, come mostrato in figura 6-46 (c) e viene confermato. Infine, 40 ms dopo la partenza, il primo ack raggiunge il mittente e il secondo getto può essere trasmesso. Poiché la linea di trasmissione è stata utilizzata per 0,5 ms su 40, l'efficienza è circa dell'1,25%. Questa situazione è tipica di quando si eseguono protocolli vecchi su linee ad alta capacità.

Una quantità utile da tenere in considerazione quando si analizzano le prestazioni di una rete è il **prodotto banda-ritardo**. Si ottiene moltiplicando la banda (in bit al secondo) per il tempo di andata e ritorno (in secondi). Il prodotto corrisponde alla capacità del canale dal mittente al ricevente e ritorno (in bit).

Nell'esempio di figura 6-46 il prodotto banda-ritardo è di 40.000.000 di bit. In altre parole, il mittente dovrebbe trasmettere un getto di 40.000.000 di bit per poter continuare a spedire alla massima velocità prima che il primo ack torni indietro. Servono tutti questi bit per riempire il canale (in entrambe le direzioni). Questo è il motivo per cui un getto di 500.000 bit ottiene un'efficienza dell'1,25%: corrisponde al 1,25% della capacità del canale.

A questo punto si può quindi concludere che per ottenere delle buone prestazioni, la finestra del ricevente deve essere almeno uguale al prodotto banda-ritardo, se possibile leggermente più grande perché il ricevente non risponde istantaneamente. Per una linea transcontinentale da 1 Gb, sono necessari almeno 5 MB per ogni connessione.

Se l'efficienza è terribile nella spedizione di 1 Mb, immaginatevi il suo valore nella spedizione di poche centinaia di byte per una chiamata di procedura remota. A meno di trovare qualche altra utilizzazione per la linea mentre il primo cliente sta aspettando una risposta, una linea da 1 Gb non è migliore di una da 1 Mb, ma solo più costosa.

Un altro problema di prestazione che si presenta con applicazioni dipendenti dal tempo (come le trasmissioni audio-video) riguarda il jitter. Avere un tempo di trasmissione medio non è sufficiente. È richiesta anche una piccola deviazione standard. Raggiungere entrambi questi obiettivi richiede un grande sforzo ingegneristico.

6.6.2 Misurare le prestazioni di rete

Quando le prestazioni di una rete sono scadenti, spesso gli utenti si lamentano con il fornitore, richiedendo miglioramenti. Per aumentare le prestazioni, gli operatori devono innanzitutto determinare esattamente cosa sta succedendo, ovvero eseguire delle misurazioni. La discussione che segue è basata sul lavoro di Mogul (1993). Per una discussione più precisa del processo di misurazione, si vedano Jain (1991); Villamizar, Song (1995). Il ciclo fondamentale utilizzato per migliorare le prestazioni di una rete contiene i seguenti passi:

1. Misurare i parametri più importanti della rete e le sue prestazioni.
2. Cercare di capire cosa sta succedendo.
3. Modificare un parametro.

Questi passi vengono ripetuti finché le prestazioni non sono sufficienti, oppure è chiaro che l'ultimo giro di miglioramento ha spremuto l'ultima goccia.

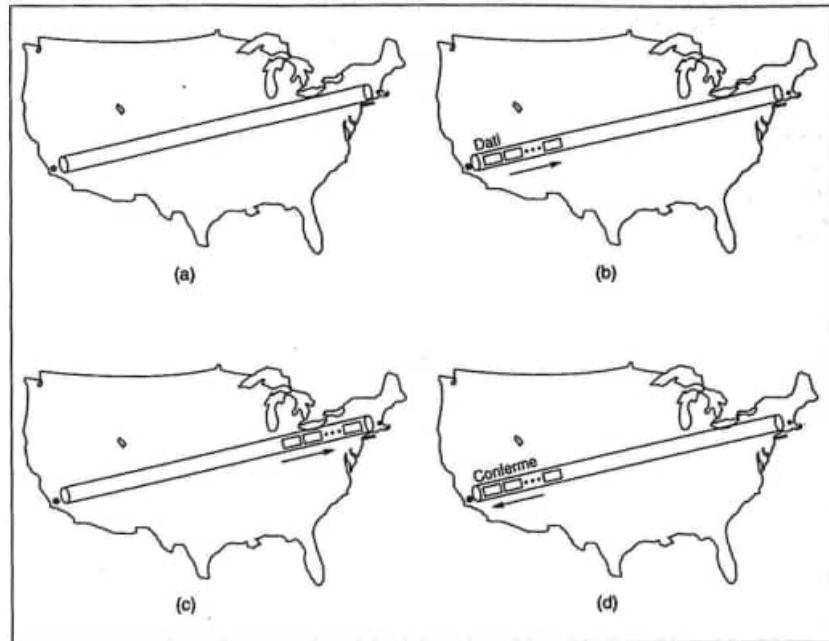


Fig. 6-46 Lo stato di trasmissione di 1 Mb da San Diego a Boston. (a) $t = 0$. (b) Dopo 500 μ s. (c) Dopo 20 ms. (d) Dopo 40 ms.

Le misure possono essere prese in molti modi e in molti luoghi diversi (sia fisicamente, che nello stack dei protocolli). Il principale tipo di misurazione consiste nel far partire un timer all'inizio di una certa attività e di utilizzarlo per vedere il tempo impiegato da quest'ultima. Ad esempio, il tempo di conferma di un TPDU è una misura chiave. Altre misure vengono effettuate con contatori che memorizzano quante volte si presenta un evento (ad es. il numero di TPDU persi). Infine, si è spesso interessati nel misurare quantità, ad esempio il numero di byte spediti durante un certo intervallo di tempo. La misurazione delle prestazioni della rete e dei suoi parametri presenta molti trabocchetti potenziali. In seguito elencheremo alcuni di essi. Qualsiasi tentativo sistematico di misurare le prestazioni di una rete dovrebbe evitarli accuratamente.

Assicuratevi che il numero di campioni sia abbastanza grande

Non misurate il tempo di spedizione di un singolo TPDU, ma ripetete la misura (ad es. 1.000.000 di volte) e calcolate la media. Un campionamento di grandi dimensioni ridurrà l'incertezza nella media e nella varianza misurate. Questa incertezza può essere misurata utilizzando le formule statistiche standard.

Assicuratevi che i campioni siano rappresentativi

Idealmente, l'intera sequenza di 1.000.000 di misurazioni dovrebbe essere ripetuta in differenti periodi del giorno e della settimana, per vedere gli effetti dei differenti carichi di lavoro sulle quantità misurate. Le misurazioni della congestione, ad esempio, sono di scarsa utilità se vengono eseguite in momenti in cui non c'è congestione. In qualche caso i risultati possono essere inizialmente controintuitivi, come ad esempio una congestione pesante alle 10, una alle 11, una all'una e una alle 2, ma nessuna congestione a mezzogiorno (quando tutti gli utenti sono a pranzo).

Attenzione nell'utilizzare orologi poco precisi

Gli orologi dei computer funzionano aggiungendo 1 a qualche contatore a intervalli regolari. Per esempio, un timer può aggiungere 1 a un contatore ogni 1 ms. Utilizzare tale timer per misurare un evento che dura meno di 1 ms non è impossibile, ma richiede qualche attenzione.

Ad esempio, per misurare il tempo necessario per spedire un TPDU, l'orologio di sistema (ad es. in millisecondi) dovrebbe essere letto sia all'entrata che all'uscita del codice dell'entità di trasporto. Se il tempo esatto di spedizione di un TPDU è di 300 µs, la differenza tra le due letture sarà 0 oppure 1, valori entrambi sbagliati. Tuttavia, se la misura viene ripetuta 1.000.000 di volte e si divide la somma totale delle letture per 1.000.000, il tempo medio risulta accurato con una precisione di 1 µs.

Assicuratevi che non succeda niente di inatteso durante i test

Realizzare delle misurazioni su un sistema universitario nel giorno di consegna di qualche importante progetto di laboratorio può dare risultati differenti da quelli ottenuti il giorno successivo. Allo stesso modo, se qualche ricercatore ha deciso di lanciare una videoconferenza sulla rete durante il test, potreste ottenere un risultato falso. È preferibile eseguire i test su un sistema inattivo e crearsi in proprio l'intero carico di lavoro. Anche questo approccio presenta dei trabocchetti. Sebbene sia ragionevole pensare che nessuno stia utilizzando la rete alle 3 di notte, potrebbe essere l'istante preciso in cui il programma automatico di backup inizia a copiare tutti i dischi su un nastro. Inoltre, potrebbero esservi molte richieste relative alle vostre magnifiche pagine WWW originate da fusi orari diversi.

I sistemi di cache possono falsare le misure

Per misurare il tempo di trasferimento file, il modo più ovvio è aprire un file di grandi dimensioni, leggerlo completamente, chiuderlo e quindi misurare il tempo richiesto. Quindi di ripetere la misura molte volte per ottenere una buona media. Vi è però un problema: il sistema potrebbe copiare in memoria cache il file e quindi solo la prima misurazione coinvolgerebbe effettivamente traffico di rete. Il resto verrebbe letto dalla memoria locale. Il risultato di tale misurazione sarebbe essenzialmente inutile (a meno che non vogliate misurare le prestazioni della cache).

Un metodo per aggirare questo problema consiste nel sovraccaricare la cache. Per esempio, se la cache è di 10 MB, il ciclo di test potrebbe aprire, leggere e chiudere due file da 10 MB ad ogni passo, nel tentativo di forzare il numero di hit a 0. Ancora una volta si

consiglia cautela, a meno che non siate assolutamente sicuri di aver compreso l'algoritmo di gestione della cache.

La gestione dei buffer può avere un effetto simile. È noto che un programma di misurazione delle prestazioni TCP/IP molto popolare riportava prestazioni per UDP sostanzialmente superiori a quelle concesse dalle linee fisiche. Qual era il motivo? Generalmente, una chiamata UDP restituiva il controllo non appena il messaggio è stato accettato dal nucleo e aggiunto alla coda di trasmissione. Se i buffer sono sufficientemente grandi, misurare il tempo necessario per eseguire 1000 chiamate UDP non significa che tutti i dati siano stati spediti. Gran parte di essi potrebbero essere ancora nel nucleo, mentre il programma di misurazione delle prestazioni pensa che siano già stati trasmessi tutti.

Capire cosa si sta misurando

Quando state misurando il tempo necessario per leggere un file remoto, le vostre misure dipendono dalla rete, dal sistema operativo sia del cliente che del server, dalle particolari schede di interfaccia utilizzate, dai loro driver e da molti altri fattori. Se lavorate con attenzione, alla fine scoprirete il tempo di trasferimento file per la configurazione che state utilizzando. Se il vostro obiettivo è regolare questa particolare configurazione, queste misure saranno perfette.

Tuttavia, se state eseguendo misure simili su tre sistemi differenti per scegliere quale interfaccia di rete acquistare, i vostri risultati potrebbero essere sballati per il fatto che uno dei driver di rete è veramente pessimo e sta ottenendo solo il 10% delle prestazioni della scheda.

Attenti a estrapolare i risultati

Si supponga di misurare qualcosa con carico di rete simulato che va da 0 (inattiva) a 0,4 (40% della capacità), come mostrato in figura 6-47. Si potrebbe essere tentati di estrarre linearmente i risultati, come nella linea punteggiata. Tuttavia, molte espressioni relative ai tempi di coda comprendono un fattore del tipo $1/(1-p)$, dove p è il carico e quindi il valore esatto può essere molto più simile alla linea tratteggiata.

6.6.3 Progettazione di sistema per migliorare le prestazioni

La misurazione e gli aggiustamenti possono migliorare considerevolmente le prestazioni, ma non possono sostituire una buona progettazione di base. Una rete mal progettata può essere migliorata solo fino a un certo punto, oltre il quale deve essere riprogettata da 0. In questo paragrafo presenteremo alcune regole pratiche basate sull'esperienza con molte reti. Queste regole sono legate alla progettazione del sistema e non solamente della rete, in quanto il software e i sistemi operativi sono spesso più importanti di router e schede di interfaccia. Gran parte di queste idee fanno parte da anni del bagaglio culturale dei progettisti di rete e sono state passate di generazione in generazione per via orale. Sono state enunciate esplicitamente per la prima volta da Mogul (1993); il nostro trattamento è in gran parte simile al suo. Un'altra importante fonte è Metcalfe (1993).

Regola #1: la velocità della CPU è più importante della velocità della rete

L'esperienza ha mostrato che in quasi tutte le reti, l'overhead dovuto al sistema operativo

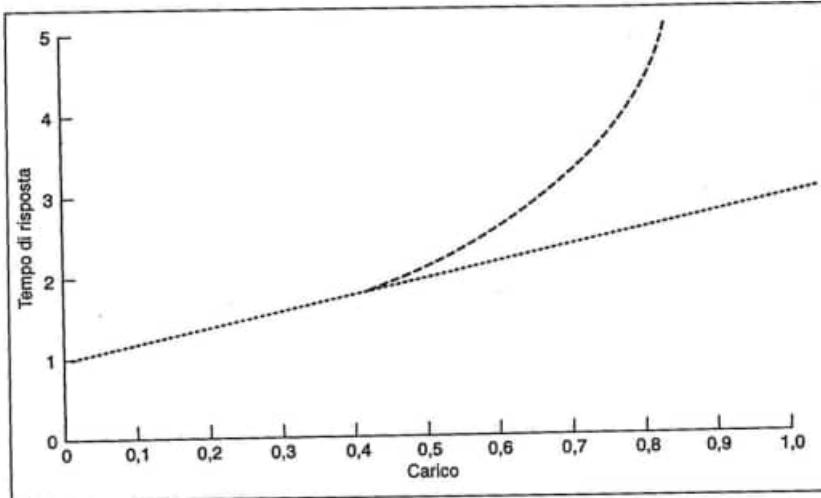


Fig. 6-47 La risposta come funzione del carico.

e ai protocolli è maggiore del tempo effettivo passato sul cavo. Ad esempio, in teoria, il tempo minimo di RPC su una Ethernet dovrebbe essere di 102 µs, corrispondente al tempo necessario per una richiesta minima (64 byte) seguita da una risposta minima (64 byte). In pratica, ottenere un tempo di RPC inferiore a 1500 µs è un risultato considerevole (Van Renesse et al., 1988). Si noti che 1500 µs è 15 volte peggiore del minimo teorico. Quasi tutto l'overhead è nel software.

Analogamente, il principale problema nell'utilizzazione di un canale da 1 Gbps è fare in modo di copiare i bit dal buffer utente alla fibra a una velocità sufficiente e di copiare i bit dalla fibra al buffer del ricevente alla stessa velocità. In breve, se raddoppierete la velocità della CPU, spesso otterrete un raddoppio delle prestazioni. Raddoppiare la capacità della rete non ha effetto in quanto spesso i colli di bottiglia sono gli host.

Regola #2: ridurre il conteggio di pacchetti per ridurre l'overhead software

L'elaborazione di un TPDU comporta un certo overhead per ogni TPDU (ad es. l'elaborazione del preambolo) e un certo overhead per ogni byte (ad es. calcolare la somma di controllo). Quando si sta spedendo 1.000.000 di byte, l'overhead relativo ai byte è indipendente dalla dimensione del TPDU. Tuttavia, utilizzare TPDU di 128 byte piuttosto che TPDU di 4K presenta un overhead molto maggiore, che cresce velocemente.

Oltre a questo overhead bisogna considerare anche quello negli strati inferiori. Ogni pacchetto in arrivo causa una interruzione. Sui processori RISC moderni, ogni interrupt interrompe il pipeline della CPU, interferisce con la cache, richiede un cambiamento di contesto di memoria e forza il salvataggio di un gran numero di registri. Ridurre di n volte il numero di TPDU spediti riduce l'overhead relativo a interrupt e pacchetti di n volte. Questa osservazione spinge ad accumulare una sostanziale quantità di dati prima della

trasmissione per ridurre il numero di interrupt all'altro lato. L'algoritmo di Nagle e la soluzione di Clark alla sindrome della finestra futile sono tentativi in questo senso.

Regola #3: minimizzare il numero di cambiamenti di contesto

I cambiamenti di contesto (ad es. da modalità kernel a modalità utente) sono micidiali. Hanno le stesse cattive proprietà delle interruzioni, fra cui un gran numero di errori di cache. Il numero di cambiamenti di contesto può essere minimizzato richiedendo che la procedura di libreria per la spedizione utilizzi alcuni buffer interni finché non dispone di una certa quantità di dati. In modo simile, il ricevente dovrebbe accumulare i TPDU in arrivo e passarli all'utente in un singolo colpo invece che individualmente, per minimizzare il numero di cambiamenti di contesto.

Nel caso migliore, un pacchetto in arrivo forza un cambiamento di contesto dall'utente corrente al kernel, e quindi un cambiamento al processo ricevente per passargli i dati appena arrivati. Ad esempio, se il gestore di rete corrisponde a un processo speciale nello spazio utente, l'arrivo di un pacchetto può facilmente causare un cambiamento di contesto dall'utente corrente al kernel, e infine dal kernel al processo ricevente. Questa sequenza è mostrata in figura 6-48. Tutti questi cambiamenti di contesto sprecano molto tempo di CPU e hanno un effetto devastante sulle prestazioni della rete.

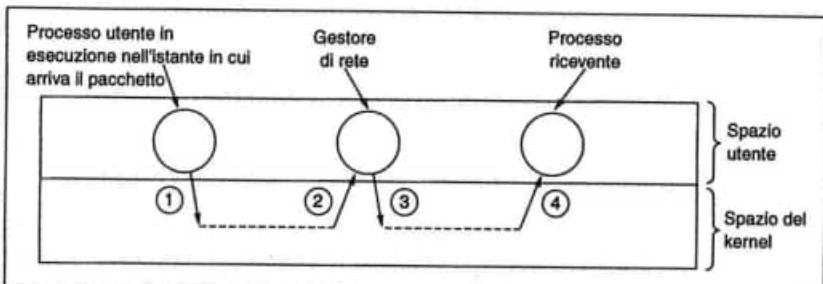


Fig. 6-48 Quattro cambiamenti di contesto per gestire un pacchetto con un gestore di rete nello spazio utente.

Regola #4: minimizzare il numero di copie

Le copie multiple sono anche peggio dei cambiamenti multipli di contesto. Non è insolito che un pacchetto in arrivo sia copiato tre o quattro volte prima che il TPDU racchiuso all'interno venga consegnato. Dopo che il pacchetto è stato ricevuto dall'interfaccia di rete in uno speciale buffer hardware sulla scheda, esso viene generalmente copiato in un buffer del kernel. Da qui viene copiato in un buffer del livello rete, quindi in un buffer del livello trasporto e infine in un buffer dell'applicazione.

Un sistema operativo intelligente copierà una parola alla volta, ma non è insolito che siano necessarie 5 istruzioni per parola (lettura, scrittura, incremento di un registro indice, un test di fine copia e un salto condizionato). Su una macchina da 50 MIPS, eseguire tre copie di ogni pacchetto con 5 istruzioni per ogni parola da 32 bit copiata richiede 75 ns per ogni byte.

Tale macchina può accettare dati alla velocità massima di 107 Mbps. Calcolando anche l'overhead per l'elaborazione del preambolo, la gestione delle interruzioni e il cambiamento di contesto, 50 Mbps potrebbero essere accettabili, e non abbiamo ancora calcolato l'elaborazione effettiva dei dati. Chiaramente, gestire una linea da 1 Gbps è impossibile.

In effetti, anche la gestione di una linea da 50 Mbps è fuori questione. Nel calcolo precedente, abbiamo assunto che una macchina da 50 MIPS possa eseguire sempre 50.000.000 istruzioni/s. In realtà, i processori possono andare a questa velocità solo se non stanno referenziando la memoria. Le operazioni in memoria sono spesso più lente di un fattore 3 delle istruzioni registro-registro, e quindi ottenere 16 Mbps dalla linea da 1 Gbps potrebbe essere considerato un buon risultato. Si noti che l'hardware non è di aiuto in questo caso. Il problema è dato dal numero eccessivo di copie effettuate nel sistema operativo.

Regola #5: potete comprare una capacità maggiore, ma non un ritardo minore

Le tre regole seguenti sono relative alla comunicazione, invece che all'elaborazione dei protocolli. La prima regola afferma che se volete una capacità maggiore, potete acquistarla facilmente. Inserire una seconda fibra vicino alla prima raddoppia la banda, ma non riduce il ritardo. Diminuire i ritardi richiede il miglioramento del software dei protocolli, del sistema operativo o dell'interfaccia di rete. Quand'anche tutti questi fattori fossero stati considerati, il ritardo non verrebbe ridotto se il collo di bottiglia fosse il tempo di trasmissione.

Regola #6: prevenire è meglio che curare (le congestioni)

La vecchia massima che dice che prevenire è meglio che curare vale certamente anche per le congestioni di rete. Quando una rete è congestionata, i pacchetti vengono persi, la banda viene sprecata, vengono introdotti ritardi inutili e altro ancora. Correggere questa situazione richiede tempo e pazienza. Evitare a priori che si presenti è sicuramente meglio. Evitare le congestioni è come ricevere una vaccinazione: leggermente dolorosa quando la ricevi, ma previene qualcosa che farebbe molto più male.

Regola #7: evitare i timeout

Nelle reti i timer sono necessari, ma dovrebbero essere utilizzati in modo spartano e il numero dei timeout dovrebbe essere minimizzato. Quando un timer scade, generalmente viene ripetuta qualche azione. Se è veramente necessario ripeterla non ci sono problemi, ma farlo senza motivo è uno spreco.

Il metodo per evitare lavoro extra consiste nel fare in modo che i timer vengano inizializzati in modo conservativo. Un timer che dura troppo a lungo aggiunge un piccolo ritardo a una connessione nell'eventualità (improbabile) che un TPDU venga perso. Un timer che scade quando non è necessario utilizza lo scarso tempo di CPU, spreca banda, e aggiunge carico extra a una dozzina di router senza un buon motivo.

6.6.4 Elaborazione veloce dei TPDU

La morale di quanto abbiamo detto finora è la seguente: il principale ostacolo alle reti veloci è il software dei protocolli. In questo paragrafo analizzeremo alcuni metodi per velocizzare tale software. Per ulteriori informazioni, si vedano Clark et al. (1989); Edwards, Muir (1995); Chandranmenon, Varghese (1995).

L'overhead relativo all'elaborazione dei TPDU possiede due componenti: quella associata ad ogni TPDU e quella associata ad ogni byte. Devono essere ridotte entrambe. La chiave per velocizzare il trattamento dei TPDU consiste nell'estrapolare il caso normale (trasferimento di dati in una singola direzione) gestendolo in modo speciale. Sebbene sia necessaria una sequenza di TPDU per raggiungere lo stato *ESTABLISHED*, in seguito l'elaborazione dei TPDU sarà semplice fino a quando uno dei due lati non inizierà a chiudere la connessione.

Iniziamo esaminando il lato mittente nello stato *ESTABLISHED* quando ci sono dati da spedire. Per chiarezza, in questo caso assumiamo che l'entità di trasporto sia contenuta nel nucleo, sebbene la stessa idea si applichi a un processo dello spazio utente oppure a una libreria all'interno del processo mittente. In figura 6-49, il processo mittente salta nel nucleo per eseguire *SEND*. La prima azione intrapresa dall'entità di trasporto è quella di eseguire un test per vedere se si trova nel caso normale: lo stato è *ESTABLISHED*, nessuno dei due lati sta cercando di chiudere la connessione, sta per essere spedito un TPDU regolare (cioè non fuori banda), ed esiste spazio di finestra sufficiente dal lato del ricevente. Se tutte queste condizioni sono soddisfatte, non sono più necessari ulteriori test e si può prendere il cammino più veloce attraverso l'entità di trasporto.

Nel caso normale, i preamboli di TPDU consecutivi sono quasi uguali. Per approfittare di questo fatto, un preambolo prototipo viene memorizzato nell'entità di trasporto. All'inizio del cammino veloce, viene copiato il più velocemente possibile in un buffer improvvisato, parola per parola. I campi che cambiano da TPDU a TPDU vengono quindi sovrascritti nel buffer. Spesso, questi campi sono facilmente derivabili da qualche variabile di stato, come ad esempio il numero di sequenza successivo. Un puntatore al preambolo di TPDU intero più un puntatore ai dati utenti vengono quindi passati al livello rete. Anche in questo caso può essere seguita la stessa strategia (non mostrata in figura 6-49). Infine, il livello rete passa il pacchetto risultante al livello data link per la trasmissione.

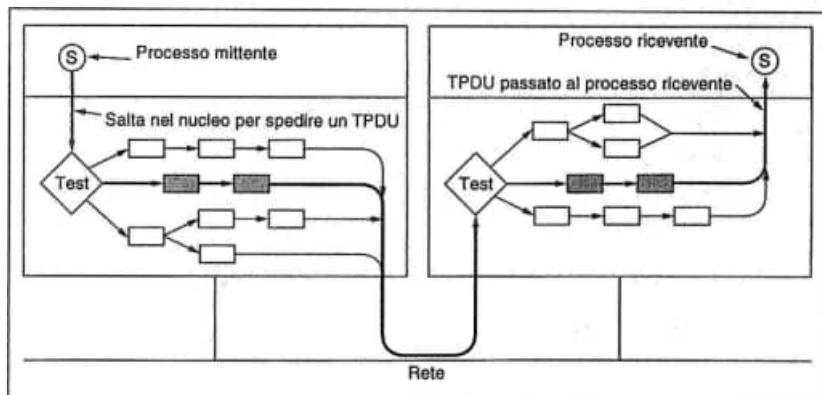


Fig. 6-49 Il cammino rapido dal mittente al ricevente viene mostrato con una linea marcata. I passi di elaborazione di questo cammino sono ombreggiati.

Come esempio pratico del funzionamento di questo principio, consideriamo TCP/IP. La figura 6-50 mostra il preambolo di TCP. I campi identici tra TPDU consecutivi in un flusso unidirezionale sono ombreggiati. Tutte le entità di trasporto devono copiare le cinque parole del preambolo prototipo nel buffer di output, inserire il prossimo numero di sequenza (copiandolo da una parola di memoria), calcolare la checksum, e incrementare il numero di sequenza in memoria. A questo punto possono passare il preambolo e i dati a una speciale procedura IP per spedire un TPDU regolare e massimale. IP copia il suo preambolo prototipo lungo cinque parole (vedi figura 6-50 (b)) nel buffer, inserisce il campo *Identification*, e calcola la checksum. Il pacchetto è quindi pronto per la trasmissione.

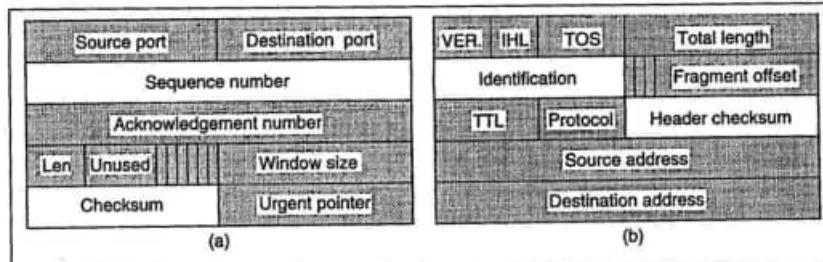


Fig. 6-50 (a) Preambolo TCP. (b) Preambolo IP. In entrambi i casi, i campi ombreggiati sono presi dal prototipo senza cambiamenti.

A questo punto, analizziamo un'elaborazione su cammino rapido dal lato ricevente di figura 6-49. Il primo passo consiste nel localizzare il record di connessione del TPDU in arrivo. Per ATM, trovare il record di connessione è facile: si può utilizzare il campo *VPI* come indice della tabella dei cammini per individuare la tabella dei circuiti virtuali per quel cammino, mentre *VCI* può essere utilizzato come indice per trovare il record di connessione. Per TCP, il record di connessione può essere memorizzato in una tabella hash la cui chiave consiste in qualche semplice funzione dei due indirizzi IP e delle due porte. Una volta individuato il record di connessione, bisogna confrontare entrambi gli indirizzi ed entrambe le porte per essere sicuri di aver trovato il record corretto.

Un'ottimizzazione che spesso velocizza ancora di più la ricerca del record di connessione consiste nel mantenere un puntatore all'ultimo record utilizzato e consultarlo per primo. Clark et al. (1989) hanno sperimentato questo sistema e hanno osservato che la percentuale di successi è superiore al 90%. Altre euristiche di lookup sono descritte in McKinney, Dove (1992).

A questo punto, si controlla se il TDPU è normale: lo stato è *ESTABLISHED*, nessuno dei due lati sta cercando di chiudere la connessione, il TPDU è completo, non sono presenti flag speciali e il numero di sequenza è quello atteso. Questi test richiedono solo pochissime istruzioni. Se tutte le condizioni sono soddisfatte, viene chiamata una procedura TCP speciale per il cammino rapido.

Il cammino rapido aggiorna il record di connessione e copia i dati all'utente. Mentre sta

copiando, calcola anche la checksum, eliminando una passata ulteriore sui dati. Se la checksum è corretta, viene aggiornato il record di connessione e viene spedito un ack. Questo schema generale, consistente nell'eseguire subito un rapido controllo per vedere se il preambolo è quello atteso e nell'utilizzare una procedura speciale per gestire questo caso, è chiamato **predizione del preambolo**, ed è realizzato da molte implementazioni TCP. Quando questa ottimizzazione e tutte le altre discusse in questo capitolo vengono utilizzate contemporaneamente, è possibile ottenere una velocità di TCP pari al 90% di una copia memoria locale-memoria locale, assumendo che la rete sia sufficientemente veloce.

Altre due aree in cui si possono ottenere grandi miglioramenti di prestazioni sono la gestione dei buffer e la gestione dei timer.

Il problema della gestione dei buffer consiste nell'evitare copie non necessarie, come citato in precedenza. La gestione dei timer è importante perché quasi tutti i timer utilizzati non fanno in tempo a scadere. Sono utilizzati per premunirsi contro la perdita di TPDU, ma gran parte di essi arrivano correttamente e così i loro ack. Quindi è importante ottimizzare la gestione dei timer per il caso in cui scadano raramente.

Uno schema comune è quello di utilizzare una lista di eventi di timer ordinati per tempo di scadenza. La prima registrazione contiene un contatore che indica quanti battiti mancano al tempo di scadenza. Ogni entrata successiva contiene un contatore che indica la differenza con il contatore precedente. Quindi, se i tempi di scadenza corrispondono agli istanti 3, 10 e 12, i tre contatori saranno rispettivamente uguali a 3, 7 e 2.

A ogni battito di orologio, il contatore nella prima registrazione viene decrementato. Quando raggiunge lo 0, l'evento corrispondente viene elaborato e l'elemento successivo della lista diviene il primo. In questo schema, inserire e cancellare timer sono operazioni costose, con un tempo di esecuzione proporzionale alla lunghezza della lista.

Un approccio più efficiente può essere utilizzato se l'intervallo di tempo massimo è limitato e noto in anticipo. In questo caso si può utilizzare un array, detto **ruota dei tempi**, come mostrato in figura 6-51. Ogni slot corrisponde a un battito di orologio. Il tempo corrente mostrato è $T = 4$. I timer sono programmati per scadere fra 3, 10 e 12 battiti. Se viene inizializzato un nuovo timer che deve scadere tra 7 battiti, viene creata una registrazione nello slot 11. In modo simile, se il timer corrispondente a $T + 10$ deve essere cancellato, la lista che inizia nello slot 14 deve essere scandita per rimuovere la registrazione richiesta. Si noti che l'array di figura 6-51 non può servire timer oltre $T + 15$.

Quando l'orologio batte, il puntatore del tempo corrente viene portato avanti di una casella (in modo circolare). Se la registrazione corrente è diversa da 0, vengono elaborati tutti i timer. Molte variazioni di questa idea di base vengono discusse in Varghese, Lauck (1987).

6.6.5 Protocolli per reti ad alta velocità

All'inizio del 1990 iniziarono ad apparire reti ad alta velocità. La prima reazione fu di utilizzarle con i vecchi protocolli, ma apparvero subito molti problemi. In questa sezione discuteremo alcuni di questi problemi e le direzioni prese dai nuovi protocolli per risolverli. Altre informazioni possono essere trovate in Baransel et al. (1995); Partridge (1994).

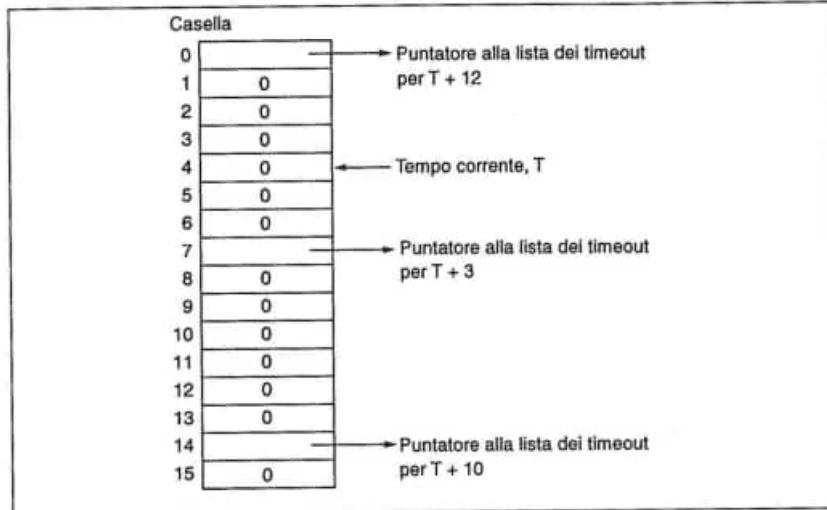


Fig. 6-51 Una ruota dei tempi.

Il primo problema è il fatto che molti protocolli utilizzano numeri di sequenza di 16 o 32 bit. Tempo fa, 2^{32} era una buona approssimazione per l'infinito. Ora non più. Alla velocità di 1 Gbps, sono necessari 32 s per trasmettere 2^{32} byte. Se i numeri di sequenza si riferiscono ai byte, come in TCP, un mittente può iniziare a trasmettere il byte 0, continuare a spedire alla massima velocità, e 32 s dopo tornare nuovamente al byte 0. Anche assumendo che tutti i byte siano stati confermati, il mittente non può trasmettere in modo sicuro nuovi dati etichettati a partire da 0, in quanto i vecchi pacchetti possono essere ancora in circolazione. In Internet, ad esempio, i pacchetti possono vivere per 120 s. Se vengono numerati i pacchetti al posto dei byte, il problema è meno grave, a meno che i numeri di sequenza siano di 16 bit, nel qual caso è anche peggiore.

Il problema è dato dal fatto che molti progettisti di protocolli assumevano solo, senza provarlo, che il tempo necessario per esaurire l'intero spazio dei numeri di sequenza fosse molto più grande del tempo massimo di vita di un pacchetto. Conseguentemente, non c'è necessità di preoccuparsi del problema dei vecchi duplicati ancora in circolazione quando i numeri di sequenza hanno eseguito un giro completo. Alle velocità dell'ordine di 1 Gb questa assunzione non provata fallisce.

Un secondo problema è il fatto che le velocità di comunicazione sono aumentate molto più velocemente di quelle di calcolo. Negli anni settanta, ARPANET aveva una velocità di 56 kbps ed era basata su computer da 1 MIPS. I pacchetti erano lunghi 1008 bit e quindi ARPANET era in grado di consegnare 56 pacchetti/s. Avendo a disposizione circa 18 ms per pacchetto, un host poteva dedicare 18.000 istruzioni ad ognuno di essi. Ovviamenete, in questo modo l'intera CPU viene occupata; in ogni caso, si potrebbero dedicare 9.000 istruzioni per pacchetto e avere ancora metà CPU disponibile per eseguire lavoro

utile. Si confrontino questi numeri con i moderni computer da 100 MIPS che si scambiano pacchetti da 4KB su una rete da 1 Gb. I pacchetti possono arrivare a una velocità superiore a 30.000 pacchetti/s, e l'elaborazione di ognuno di essi dovrebbe essere completata in 15 μ s se volessimo riservare metà della CPU alle applicazioni. In 15 μ s, un computer da 100 MIPS può eseguire 1500 istruzioni, solamente 1/6 di quelle a disposizione degli host ARPANET. Inoltre, le moderne istruzioni RISC sono meno potenti delle vecchie istruzioni CISC, e quindi il problema è anche peggiore di come appare. La conclusione è la seguente: c'è meno tempo disponibile per l'elaborazione dei protocolli, e quindi i protocolli stessi devono diventare più semplici.

Un terzo problema riguarda il protocollo go back n, che ha cattive prestazioni nel caso di linee con un elevato prodotto banda-ritardo. Si consideri, ad esempio, una linea da 4000 km che operi alla velocità di 1 Gbps. Il tempo di trasmissione andata ritorno è di 40 ms, durante il quale un mittente può spedire 5 Mb. Se viene rilevato un errore, sono necessari 40 ms prima che il mittente ne venga a conoscenza. Se venisse utilizzato go back n, il mittente dovrebbe rispedire non solo il pacchetto incriminato, ma anche tutti i 5 MB che seguono. Chiaramente, questo è uno spreco enorme di risorse.

Un quarto problema è dato dal fatto che le linee da 1 Gb sono fondamentalmente differenti da quelle da 1 Mb, in quanto quelle più lunghe sono limitate dal ritardo e non dalla banda. In figura 6-52 viene mostrato il tempo necessario per trasferire un file da 1 Mb per 4000 km a varie velocità di trasmissione. A velocità inferiori a 1 Mbps, il tempo di trasmissione è dominato dalla velocità di spedizione dei bit. Oltre 1 Gbps, il ritardo di andata e ritorno di 40 ms domina il tempo di 1 ms necessario per trasmettere i bit. Ulteriori incrementi di banda difficilmente producono miglioramenti.

La figura 6-52 possiede alcune implicazioni sgradevoli per i protocolli di rete. Si dice che i protocolli stop-and-wait, come ad esempio RPC, hanno un inherente limite superiore alle loro prestazioni. Il limite è dettato dalla velocità della luce. Nessun progresso tecnologico nell'ottica produrrà miglioramenti (al contrario, nuove leggi della fisica sarebbero d'aiuto).

Un quinto problema che vale la pena di menzionare non dipende dalla tecnologia o dai protocolli come i precedenti, ma è un risultato delle nuove applicazioni. In poche parole, per molte applicazioni ad alta velocità come quelle multimediali, la varianza nei tempi di arrivo è importante quanto il ritardo medio stesso. Una velocità di trasmissione "lenta ma uniforme" è spesso preferibile a un comportamento "veloce, ma a scatti".

Passiamo ora dal problema ai modi di affrontarlo. Faremo innanzitutto alcuni commenti generali, quindi analizzeremo i meccanismi dei protocolli, la struttura dei pacchetti e il software dei protocolli.

Il principio base che tutti i progettisti di reti ad alte prestazioni dovrebbero sempre tenere in mente è:

Progettare per ottimizzare la velocità e non la banda.

I vecchi protocolli venivano spesso progettati per minimizzare il numero di bit spediti, spesso utilizzando campi piccoli e impacchettandoli insieme in byte e parole. Ai nostri

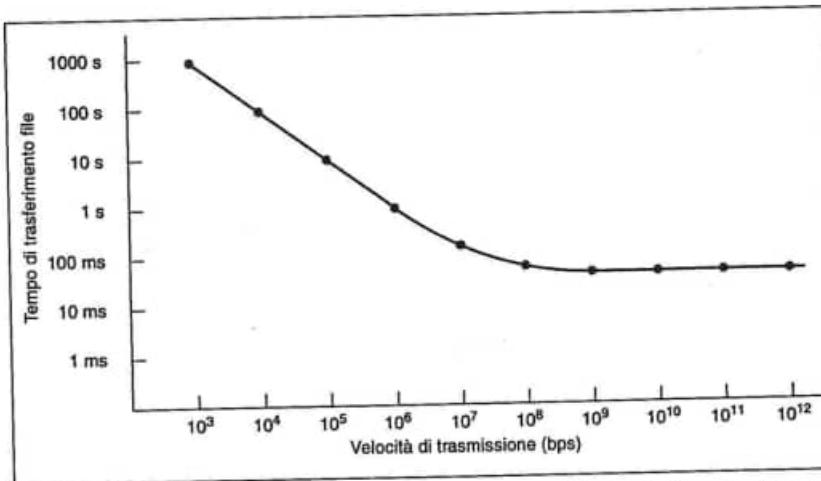


Fig. 6-52 Tempo necessario per trasferire e confermare un file da 1 Mb su una linea di 4000 km.

giorni, abbiamo a disposizione grandi quantità di banda. Il problema è l'elaborazione dei protocolli, che quindi devono essere progettati in modo da ridurli al minimo. Un tentativo per andare più veloce consiste nel costruire interfacce di rete molto veloci in hardware. A meno che il protocollo non sia eccezionalmente semplice, la difficoltà di questa strategia fa in modo che hardware non significhi altro che una scheda con una seconda CPU e il proprio programma. Per evitare che il coprocessore di rete sia costoso quanto la CPU principale, spesso è un chip più lento. Come conseguenza di questa progettazione, la CPU principale passa gran parte del tempo in attesa che la seconda CPU esegua il lavoro critico. Pensare che la CPU principale abbia altro lavoro da portare avanti mentre è in attesa non è altro che un mito. Inoltre, quando due CPU di tipo general-purpose comunicano, possono presentarsi dei conflitti, e quindi sono necessari elaborati protocolli tra i due processori per sincronizzarli correttamente. In generale, l'approccio migliore è rendere semplici i protocolli e utilizzare la CPU principale per essi. A questo punto analizziamo il problema del feed-back nei protocolli ad alta velocità. A causa del ciclo di ritardo (relativamente) lungo, il feed-back dovrebbe essere evitato: è necessario troppo tempo affinché il mittente riceva un segnale dal ricevente. Un esempio di feed-back consiste nel governare la velocità di trasmissione utilizzando un protocollo di sliding window. Per evitare i lunghi ritardi insiti nella spedizione di aggiornamenti della finestra dal ricevente al mittente, è preferibile utilizzare un protocollo basato sulla velocità. In tale protocollo, il mittente può spedire tutto quello che vuole, a condizione che non trasmetta più velocemente di quanto concordato in precedenza con il mittente. Un secondo esempio di feed-back è l'algoritmo slow start di Jacobson. Questo algoritmo effettua molti tentativi per vedere quanto sia in grado di gestire la rete. Con reti ad alta velocità, fare un mezza dozzina di piccoli tentativi per vedere come risponde la rete spreca

un sacco di banda. Uno schema più efficiente consiste nel fare in modo che il mittente, il ricevente e la rete riservino le risorse necessarie al momento di creare la connessione. Prenotare le risorse in anticipo ha anche il vantaggio di rendere più facile ridurre il jitter. In breve, andare verso le alte velocità spinge inesorabilmente la progettazione verso un funzionamento orientato alla connessione, o verso qualcosa di molto simile.

La struttura dei pacchetti è un'altra importante caratteristica delle reti ad alta velocità. Il preambolo deve contenere il minor numero di campi possibili, per ridurre il tempo di elaborazione, e questi campi devono essere abbastanza grandi per la loro funzione e devono essere allineati alle parole per semplificare l'elaborazione. In questo contesto, "abbastanza grande" significa che non si presentano problemi quali l'utilizzazione multipla dei numeri di sequenza, l'impossibilità di pubblicizzare spazio di finestra perché il campo corrispondente è troppo piccolo e così via.

Le checksum del preambolo e dei dati devono essere calcolate separatamente, per due motivi. Primo, per poter controllare il preambolo senza controllare i dati. Secondo, per verificare che il preambolo sia corretto senza iniziare a copiare i dati nello spazio utente. È desiderabile che i dati vengano controllati mentre vengono copiati nello spazio utente, ma se il preambolo non è corretto, la copia può avvenire nel processo sbagliato. Per evitare di copiare i dati nel posto sbagliato e consentire il calcolo della checksum durante la copia, è essenziale che le due checksum siano separate.

La dimensione massima dei dati deve essere grande, in modo da consentire un funzionamento efficiente anche in caso di lunghi ritardi. Inoltre, più grandi sono i blocchi dati, più piccola è la frazione di banda dedicata ai preamboli.

Un'altra caratteristica importante è l'abilità di spedire una quantità normale di dati insieme alla richiesta di connessione. In questo modo, un intervallo di andata e ritorno può essere evitato.

Infine, sono necessarie alcune parole sul software dei protocolli. Un concetto chiave è concentrarsi sul caso di successo. Molti vecchi protocolli tendevano a enfatizzare cosa fare nel caso che qualcosa andasse storto (ad es. un pacchetto perso). Per velocizzare i protocolli, i progettisti dovrebbero dedicarsi a minimizzare il tempo di elaborazione quando tutto va bene. Minimizzare il tempo di elaborazione quando si presenta un errore è secondario.

Una seconda problematica relativa al software riguarda la minimizzazione del tempo di copia. Come abbiamo detto in precedenza, copiare i dati è spesso la principale sorgente di perdite di tempo. Idealmente, l'hardware dovrebbe duplicare ogni pacchetto in arrivo nella memoria come un blocco contiguo di dati. Il software dovrebbe quindi copiare questo pacchetto nel buffer dell'utente in una singola copia. A seconda di come funziona la cache, può anche essere preferibile evitare un ciclo di copia. In altre parole, per copiare 1024 parole, il modo migliore è avere 1024 istruzione MOVE consecutive (o 1024 coppie LOAD-STORe). La routine di copia è così critica che dovrebbe essere realizzata a mano in codice assembly, a meno che il compilatore non sia in grado di produrre precisamente il codice ottimo.

Alla fine degli anni ottanta, ci fu una breve folata di interesse in protocolli veloci di tipo special-purpose come NETBLT (Clark et al., 1987), VTMP (Cheriton, Williamson, 1989) e XTP (Chesson, 1989). Una panoramica è presente in Doeringer et al. (1990). Tuttavia,

la tendenza attuale è semplificare i protocolli general-purpose per renderli anch'essi veloci. ATM presenta molte delle caratteristiche discusse in questa sezione, così come IPv6.

6.7 Riassunto

Il livello trasporto è fondamentale per comprendere la gerarchia dei protocolli. Fornisce vari servizi, il più importante dei quali è un flusso di dati punto-a-punto, affidabile e orientato alla connessione dal mittente al ricevente. Vi si accede tramite primitive di servizio che consentono la creazione, l'utilizzazione e la chiusura delle connessioni.

I protocolli di trasporto devono essere in grado di gestire le connessioni su reti non affidabili. La creazione delle connessioni è complicata dall'esistenza di pacchetti duplicati in ritardo che possono apparire in momenti inopportuni. Per gestirli, sono necessari protocolli three-way handshake per creare connessioni. Chiudere una connessione è più semplice che crearne una, ma non è del tutto banale a causa del problema dei due eserciti. Anche quando il livello rete è completamente affidabile, il livello trasporto deve eseguire molto lavoro, come abbiamo visto nel nostro esempio. Deve gestire tutte le primitive di servizio, gestire le connessioni e i timer, e allocare e utilizzare i crediti.

Il principale protocollo di trasporto di Internet è TCP. Utilizza un preambolo di 20 byte su tutti i segmenti. I segmenti possono essere frammentati dai router interni a Internet, e quindi gli host devono essere in grado di riassembrarli. Grandi sforzi sono stati dedicati all'ottimizzazione delle prestazioni di TCP, utilizzando algoritmi di Nagle, Clark, Jacobson, Karn et al.

ATM possiede quattro protocolli del livello AAL. Ognuno di essi spezza i messaggi in celle alla sorgente e riassembra le celle in messaggi alla destinazione. Gli strati CS e SAR aggiungono i loro preamboli e code in vari modi, lasciando da 44 a 48 byte di contenuto per cella.

Le prestazioni della rete sono generalmente dominate dall'overhead del protocollo e dell'elaborazione dei TPDU, e questa situazione peggiora con le alte velocità. I protocolli dovrebbero essere progettati per minimizzare il numero di TPDU, il numero di cambiamenti di contesto e il numero di volte che una TPDU viene copiata. Nel caso di reti ad altissima velocità sono richiesti semplici protocolli di controllo di flusso basati sulla velocità piuttosto che sui crediti.

Esercizi

- Nelle primitive di trasporto dell'esempio di figura 6-3, LISTEN è una chiamata bloccante. Ciò è davvero necessario? Se non lo è, spiegare come potrebbe essere utilizzata una primitiva non bloccante. Quali sarebbero i vantaggi di questo schema rispetto a quello descritto nel testo?
- Nel modello sottostante alla figura 6-5, si assume che i pacchetti possano essere persi dal livello rete e che debbano essere confermati individualmente. Si supponga che il livello rete sia affidabile al 100% e non perda mai pacchetti. Quali cambiamenti sono necessari in figura 6-5?

6.7 Riassunto

- Si immagini un problema generalizzato degli n eserciti, nel quale l'accordo di almeno due eserciti è sufficiente per la vittoria. Esiste un protocollo che consenta all'esercito blu di vincere?
- Si supponga che lo schema di generazione dei numeri di sequenza iniziali basato su orologio venga utilizzato con un contatore di 15 bit. L'orologio emette 1 battito ogni 100 ms, e il tempo massimo di vita di 1 pacchetto è di 60 s. Ogni quanto è necessario eseguire una risincronizzazione
 - nel caso peggiore?
 - quando i dati consumano 240 numeri di sequenza al minuto?
- Per quale motivo il tempo massimo di vita di un pacchetto, T , deve essere grande a sufficienza da assicurare che non solo il pacchetto, ma anche i suoi ack siano scomparsi?
- Immaginare di utilizzare un protocollo two-way handshake al posto di three-way handshake per creare le connessioni. In altre parole, il terzo messaggio non è richiesto. In questo caso è possibile che si presenti uno stallo? Fare un esempio oppure dimostrare che non ne esiste nessuno.
- Si consideri il problema del ripristino dopo il guasto di un host (figura 6-18). Se l'intervallo tra la scrittura e la spedizione dell'ack (o viceversa) può essere reso relativamente piccolo, quali sono le due migliori strategie mittente-ricevente per minimizzare la probabilità di un fallimento del protocollo?
- Esiste la possibilità di stallo dell'entità di trasporto descritta nel testo?
- Per curiosità, l'implementatore dell'entità di trasporto di figura 6-20 ha deciso di inserire dei contatori nella procedura di *sleep* per collezionare statistiche sull'array *conn*. Fra queste vi è il numero di connessioni in ognuno dei sette stati possibili, n_i ($i = 1, \dots, 7$). Dopo aver scritto un massiccio programma FORTRAN per analizzare i dati, il nostro implementatore ha scoperto che la relazione $\sum n_i = MAX_CONN$ sembra essere sempre vera. Esistono anche altri invarianti che coinvolgono solamente queste sette variabili?
- Cosa succede se l'utente dell'entità di trasporto di figura 6-20 spedisce un messaggio di lunghezza 0? Discutere il significato della risposta.
- Per ciascuno degli eventi che possono avvenire nell'entità di trasporto di figura 6-20, descrivere la sua legalità quando l'utente è sospeso nello stato *sending*.
- Discutere i vantaggi e gli svantaggi dei crediti nei confronti dei protocolli sliding window.
- La frammentazione e la ricomposizione dei datagram vengono gestite da IP e sono invisibili a TCP. Questo significa che TCP non deve preoccuparsi di dati che arrivano nell'ordine sbagliato?
- A un processo dell'host 1 è stata assegnata la porta p , mentre a un processo dell'host

- 2 è stata assegnata la porta q . È possibile che esistano contemporaneamente due o più connessioni TCP tra queste due porte?
15. Il massimo contenuto di un segmento TCP è 65.495 byte. Perché è stato scelto questo strano numero?
 16. Descrivere due modi per entrare nel livello *SYN RCVD* di figura 6-28.
 17. Trovare uno svantaggio potenziale dell'algoritmo di Nagle quando viene utilizzato in una rete fortemente congestionata.
 18. Si considerino gli effetti dell'utilizzo di slow start in una linea con un tempo di andata e ritorno di 10 ms e nessuna congestione. La finestra del ricevente è di 24 KB e la dimensione massima di un segmento è 2 KB. Quanto tempo è necessario per spedire la finestra intera?
 19. Si supponga che la finestra di congestione di TCP sia posta a 18 KB e avvenga un timeout. Quanto sarà grande la finestra se le quattro spedizioni successive hanno tutte successo? Si assume che il segmento di dimensione massima sia di 1 KB.
 20. Se il tempo di andata e ritorno di TCP, RTT , è attualmente di 30 ms e gli ack successivi arrivano rispettivamente dopo 26, 32 e 24 ms dalla spedizione, qual è la nuova stima di RTT quando $\alpha = 0,9$?
 21. Una macchina TCP sta spedendo finestre di 65.535 byte su un canale di 1 Gbps con un ritardo di sola andata di 10 ms. Qual è la velocità di trasmissione massima? Qual è l'efficienza della linea?
 22. Se una rete ha una dimensione massima di TPDU di 128 byte, un tempo di vita massimo di TPDU di 30 s, e un numero di sequenza di 8 bit, qual è la velocità di trasmissione massima per ogni connessione?
 23. Perché esiste UDP? Non sarebbe stato sufficiente lasciare che i processi utenti spedissero pacchetti IP normali?
 24. Un gruppo di N utenti nello stesso edificio sta utilizzando lo stesso computer remoto attraverso una rete ATM. L'utente medio genera L linee di traffico (input + output) per ora, in media, con una lunghezza media delle linee di P byte, escludendo i preamboli ATM. Il fornitore fa pagare C centesimi per ogni byte di dati utente trasportati, più X centesimi per ora per ognuno dei circuiti virtuali aperti. A quali condizioni è più economico eseguire un multiplexing delle N connessioni di trasporto sullo stesso circuito virtuale ATM, se tale multiplexing aggiunge 2 byte di dati ad ogni pacchetto? Si assume che un singolo circuito virtuale ATM possiede banda sufficiente per tutti gli utenti.
 25. AAL 1 è in grado di gestire messaggi più corti di 40 byte utilizzando lo schema basato sul campo *Pointer*? Spiegare la risposta.
 26. Cercare di indovinare le dimensioni dei campi di AAL 2 prima che fossero rimossi dallo standard.

27. AAL 3/4 consente di eseguire il multiplexing di connessioni su di un singolo circuito virtuale. Si dia una situazione di esempio in cui questo non ha valore. Si assuma che un singolo circuito virtuale possiede banda sufficiente per tutto il traffico. Suggerimento: si pensi ai circuiti virtuali.
28. Quant'è il contenuto di un messaggio di dimensione massima che entra in una singola cella AAL 3/4?
29. Quando un messaggio di 1024 byte viene spedito tramite AAL 3/4, qual è l'efficienza ottenuta? In altre parole, qual è la frazione utilizzabile dei bit trasmessi? Si ripeta il calcolo con AAL 5.
30. Un dispositivo ATM sta trasmettendo messaggi a singola cella alla velocità di 600 Mbps. 1 cella ogni 100 viene persa a causa di rumore casuale. Quanti errori non rilevati per settimana ci si può attendere utilizzando la checksum a 32 bit di AAL 5?
31. Un cliente invia una richiesta di 128 byte a un server distante circa 100 km collegato con una fibra ottica di 1 Gb. Qual è l'efficienza della linea durante la chiamata di procedura remota?
32. Si consideri nuovamente la situazione del problema precedente. Si calcoli il minor tempo possibile di risposta sia per una linea da 1 Gbps, sia per una linea da 1 Mbps. Quali conclusioni si possano trarre?
33. Supporre di misurare il tempo di ricezione di un TPDU. Quando si presenta un'interruzione, leggere l'orologio di sistema in millisecondi. Quando il TPDU è elaborato completamente, leggere nuovamente l'orologio. Misurare 0 ms 270.000 volte e 1 ms 730.000 volte. Quanto tempo è necessario per ricevere un TPDU?
34. Una CPU esegue istruzione alla velocità di 100 MIPS. I dati possono essere copiati 64 bit alla volta, dove ogni parola copiata costa sei istruzioni. Se un pacchetto deve essere copiato due volte, questo modello è in grado di gestire una linea da 1 Gbps? Per semplicità, si assume che tutte le istruzioni, anche quelle di scrittura e lettura della memoria, vengano eseguite alla velocità massima di 100 MIPS.
35. Per aggirare il problema dei numeri di sequenza che si sovrappongono quando esistono ancora vecchi pacchetti, si potrebbe utilizzare un numero di sequenza di 64 bit. Tuttavia, in linea teorica una fibra ottica potrebbe trasportare 75 Tbps. Qual è il valore necessario del tempo massimo di vita di un pacchetto, per essere sicuri che nelle future reti da 75 Tbps non si presentino problemi di sovrapposizioni con i numeri di sequenza da 64 bit? Si assume che ogni byte abbia il proprio numero di sequenza, proprio come TCP.
36. Nel testo abbiamo calcolato che una linea da 1 Gb scarica in un host 30.000 celle/s, lasciando solo 1500 istruzioni per elaborare e metà del tempo di CPU per le applicazioni. Questo calcolo assume l'esistenza di un pacchetto di 4 KB. Rifare il calcolo per un pacchetto di dimensioni ARPANET (128 byte).
37. Per una rete da 1 Gbps che funziona su 4000 km, il fattore limitante è il ritardo e

- non la banda. Si consideri una MAN con una distanza media tra sorgente e destinazione di 20 km. Quanto è lungo il tempo di andata e ritorno nel caso in cui il tempo di ritardo dipenda dalla velocità della luce per pacchetti di 1 KB?
38. Si modifichi il programma di figura 6-20 in modo che gestisca gli errori. Si aggiunga un nuovo tipo di pacchetto, *reset*, che possa arrivare dopo che una connessione sia stata aperta da entrambi i lati, ma non sia stata chiusa da nessuno. (Questo evento, che avviene simultaneamente su entrambi i lati della connessione, significa che qualsiasi pacchetto fosse in transito sarebbe stato consegnato oppure distrutto, ma in entrambi i casi non esisterebbe più nella rete).
39. Scrivere un programma che simuli la gestione dei buffer in un'entità di trasporto che utilizzi un protocollo sliding window per il controllo di flusso al posto del sistema dei crediti di figura 6-20. Per mantenerlo semplice, supporre che tutti i dati viaggino dalla macchina A alla macchina B, e nessuno nel senso opposto. Sperimentare differenti strategie di allocazione dei buffer nella macchina B, come ad esempio dedicare buffer a una connessione specifica oppure utilizzare un insieme comune di buffer, e misurare la capacità di trasmissione totale ottenuta in ognuna delle due macchine.

Capitolo 7

IL LIVELLO DELLE APPLICAZIONI

Esaminata tutta la gerarchia sottostante, arriviamo finalmente al livello delle applicazioni (application), dove troviamo tutte le applicazioni interessanti. I livelli inferiori a questo forniscono un supporto di comunicazione affidabile, ma non effettuano nessun compito utile per l'utente. In questo capitolo analizzeremo alcune applicazioni reali.

In ogni caso, anche a livello applicazioni vi è la necessità di protocolli di supporto per permettere alle applicazioni reali di funzionare. Di conseguenza, analizzeremo tre classi di protocolli prima di iniziare con le applicazioni stesse. La prima classe riguarda la sicurezza, che non si basa su un protocollo singolo, bensì su un grande numero di concetti e protocolli che possono essere utilizzati per assicurare la privacy dove necessario. La seconda classe è il DNS, che si occupa della gestione dei nomi in Internet. Il terzo protocollo di supporto riguarda la gestione della rete. Dopo questa analisi, esamineremo quattro applicazioni reali: la posta elettronica, USENET (news via rete), il World Wide Web e infine il multimediale.

7.1 La sicurezza in rete

Durante i primi decenni della loro esistenza, le reti di computer sono state usate principalmente dai ricercatori universitari per inviare messaggi elettronici, e dai dipendenti di industrie per condividere stampanti. In questi ambiti, la segretezza non aveva ragione di esistere. Invece adesso, con milioni di comuni cittadini che utilizzano le reti per operazioni bancarie, commerciali e fiscali questo problema è apparso all'orizzonte come potenzialmente collettivo. Nei prossimi paragrafi studieremo la sicurezza di rete da differenti punti di vista, evidenziando i numerosi problemi e discutendo diversi algoritmi e protocolli per rendere le reti più sicure.

Questo argomento è vasto e riguarda una moltitudine di crimini. Nella sua forma più semplice, si occupa di assicurare che nessuno possa leggere o, peggio ancora, modificare i messaggi destinati ad altri. Si occupa di persone che cercano di accedere a servizi remoti che non sono autorizzate a usare. Inoltre si preoccupa di capire se quel messaggio con la pretesa di provenire dall'ufficio delle tasse che dice: "Pagare Venerdì oppure..." sia veramente dell'ufficio delle tasse oppure sia una minaccia di estorsione. La sicurezza si occupa anche di problemi come l'intercettazione di messaggi autografi e la loro riproduzione, e di persone che cercano di negare di aver spedito tali messaggi.

La figura 7-1 elenca alcuni dei nemici più comuni. Da questa lista appare chiaro che rendere sicura una rete vuol dire molto di più che mantenerla solo libera da errori di

programmazione. La sicurezza implica una lotta contro avversari spesso intelligenti, che vi si dedicano completamente e che spesso sono tecnologicamente ben attrezzati. Inoltre deve essere chiaro che le misure per fermare gli avversari occasionali avranno poco successo con quelli che fanno sul serio.

Avversario	Scopo
Studente	Divertirsi curiosando nella posta altrui
Hacker	Verificare i sistemi di sicurezza di qualcun altro; rubare dati
Venditore	Pretendere di rappresentare tutta Europa
Uomo d'affari	Scoprire il piano strategico di mercato di un concorrente
Ex dipendente	Vendicarsi per essere stato licenziato
Cassiere	Appropriarsi del denaro di una società
Agente di cambio	Negare una promessa fatta a un acquirente per email
Truffatore	Rubare numeri di carte di credito
Spia	Scoprire la forza militare di un nemico
Terrorista	Rubare segreti per una guerra batteriologica

Fig. 7-1 Alcune delle persone che causano problemi di sicurezza e perché.

È possibile suddividere i problemi di sicurezza di rete in quattro aree tra loro intersecate: segretezza, autenticazione, non disconoscimento e controllo di integrità.

La segretezza ha a che fare con la riservatezza delle informazioni nei confronti degli utenti non autorizzati. Questo è ciò che di solito viene in mente quanto si pensa alla sicurezza di rete.

L'autenticazione riguarda il determinare con chi si sta parlando prima di rivelare informazioni particolari o di iniziare una trattativa d'affari.

Il non disconoscimento riguarda le firme: come è possibile provare che il vostro cliente abbia realmente inviato un ordine elettronico per 10.000.000 di pezzi a 89 centesimi l'uno quando egli afferma che il prezzo fosse di 69 centesimi?

In ultimo, com'è possibile essere sicuri che un messaggio ricevuto sia davvero quello spedito e non qualcosa che un avversario ha modificato mentre transitava oppure che ha inventato?

Tutti questi argomenti (segretezza, autenticazione, non disconoscimento e controllo di integrità) si ritrovano anche nei sistemi tradizionali, ma con alcune differenze significative. La segretezza e l'integrità vengono conseguite utilizzando la registrazione della posta e chiudendo i documenti in busta chiusa.

Anche perché non è sempre così facile accorgersi della differenza tra un documento originale e una fotocopia (ad es. con gli assegni elettronici, originali e copie sono indistinguibili).

Le persone riconoscono le altre persone attraverso i loro visi, le voci e la calligrafia. La prova di autenticità si ottiene mediante firme su lettere intestate, sigilli, e così via. Le falsificazioni di solito possono essere scoperte da esperti calligrafici, di carte e inchiostri. Nessuna di queste opzioni è disponibile elettronicamente. Chiaramente, sono necessarie altre soluzioni.

Prima di analizzare le soluzioni stesse, è bene spendere alcuni minuti considerando dove si trova la sicurezza di rete nell'insieme stratificato dei protocolli. Probabilmente non esiste un posto unico: ogni livello contribuisce in qualche modo. Nel livello fisico, l'intercettazione di messaggi può essere contrastata racchiudendo le linee di trasmissione in tubi sigillati contenenti del gas argo ad alta pressione. Qualsiasi tentativo per perforare il tubo provocherà una fuga di gas, riducendo la pressione e facendo scattare un allarme. Alcuni sistemi militari utilizzano questa tecnica.

A livello data link, i pacchetti su una linea punto-a-punto possono venire codificati appena lasciano una macchina e decodificati appena arrivano all'altra. Tutti i dettagli possono essere gestiti dal livello data link, lasciando i livelli superiori ignari di ciò che sta succedendo. Questa soluzione, tuttavia, fallisce quando i pacchetti devono attraversare più router, in quanto i pacchetti devono essere decodificati su ciascuno, rendendoli vulnerabili agli attacchi dall'interno di un qualsiasi router. Inoltre, non permette di proteggere alcune sessioni (ad es., quelle che coinvolgono pagamenti tramite carte di credito) e altre no. Ciò nonostante, la codifica del link (**link encryption**), come viene chiamato questo metodo, è facile da aggiungere a ogni rete e viene utilizzata di frequente.

Al livello rete è possibile installare firewall per accettare o non accettare pacchetti. Abbiamo analizzato i firewall nel capitolo 5. Al livello trasporto, è possibile codificare l'intera connessione da un capo all'altro, che significa da processo a processo. Sebbene queste soluzioni risolvano alcuni problemi di segretezza, e sebbene molte persone stiano lavorando seriamente per migliorarle, nessuna di esse risolve il problema dell'autenticazione o del non disconoscimento in un modo sufficientemente generale. Per affrontare questi problemi, le soluzioni devono essere al livello delle applicazioni, e questa è la ragione per la quale vengono studiate in questo capitolo.

7.1.1 La crittografia tradizionale

La crittografia ha una storia lunga e pittoresca. In questo paragrafo vedremo solamente gli aspetti più significativi, utili per capire per ciò che descriveremo in seguito. Per una storia completa, il libro di Kahn (1967) è ancora la lettura raccomandata. Per una trattazione completa dell'attuale stato dell'arte, si vedano Kaufman *et al.* (1996); Schneier (1996); Stinson (1995).

Storicamente, sono stati quattro i gruppi di persone che hanno dato il loro contributo all'arte della crittografia: i militari, il corpo diplomatico, gli scrittori di diari e gli amanti. Tra questi, i militari hanno avuto il ruolo più importante e hanno dato un'impronta al settore. All'interno delle organizzazioni militari, i messaggi da crittografare per tradizione venivano affidati a dipendenti mal pagati per essere codificati e trasmessi. Il basso volume dei messaggi impediva che questo lavoro venisse affidato a pochi specialisti scelti.

Fino all'avvento dei computer, uno dei principali limiti della crittografia era il grado di abilità del codificatore nell'eseguire le trasformazioni necessarie, spesso su un campo di battaglia con il minimo equipaggiamento. Un ulteriore vincolo era la difficoltà di passare velocemente da un metodo crittografico all'altro, poiché questo implicava di dover istruire nuovamente un gran numero di persone. Ciò nonostante, il pericolo che chi si occupava della codificazione fosse catturato dal nemico rese essenziale poter cambiare il metodo crittografico all'istante, se fosse sorta la necessità. Questi requisiti tra loro in conflitto hanno portato al modello in figura 7-2.

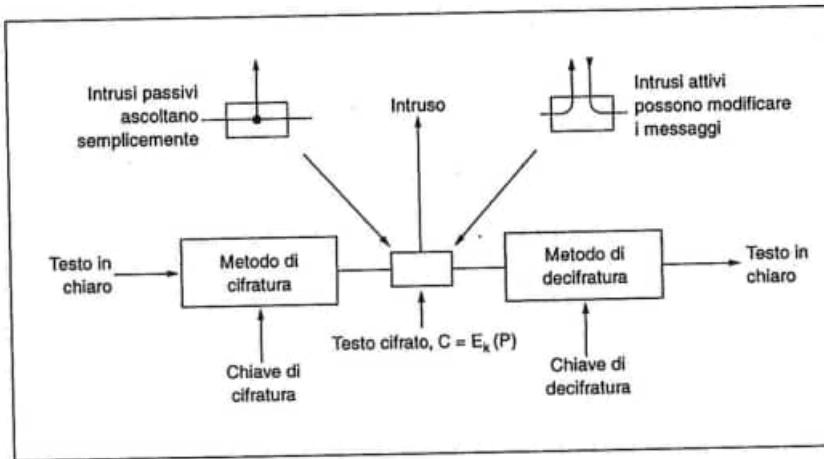


Fig. 7-2 Il modello crittografico.

I messaggi da codificare, detti **testo in chiaro**, vengono trasformati da una funzione che è parametrizzata da una **chiave**. L'uscita del processo di codifica, detto **testo cifrato**, viene quindi trasmessa, tramite un messaggero o via radio. Si assume che il nemico, o l'**intruso**, ascolti e trascriva il testo cifrato completo. In ogni caso, a differenza del corretto destinatario, egli non sa qual è la chiave per decodificare e quindi non può facilmente decodificare il messaggio cifrato. A volte l'intruso può non soltanto ascoltare il canale di comunicazione (intruso passivo) ma può anche registrare i messaggi e riascoltarli in seguito, inviare un proprio messaggio oppure modificare i messaggi originali prima che questi vengano ricevuti dal destinatario (intruso attivo). L'arte di violare cifrari viene chiamata **crittanalisi**. L'arte di progettare cifrari (crittografia) e di violarli (crittanalisi) è genericamente conosciuta come **crittologia**.

Spesso è utile avere una notazione per riferire il testo in chiaro, il testo cifrato e le chiavi. Useremo $C = E_k(P)$ per significare che la codifica del testo in chiaro P utilizzando la chiave K produce il testo cifrato C .

In modo simile, $P = D_k(C)$ rappresenta la decodifica di C per produrre nuovamente il testo in chiaro. Segue quindi che:

$$D_k(E_k(P)) = P$$

Questa notazione suggerisce che E e D sono funzioni matematiche, come in effetti sono. La sola parte discutibile è che entrambe sono funzioni di due parametri, e noi abbiamo scritto uno dei due parametri (la chiave) come un indice, piuttosto che come un argomento, per distinguerlo dal messaggio.

Una regola fondamentale della crittografia è che si deve assumere che il crittanalista conosca il metodo generale di codifica usato. In altre parole, il crittanalista sa come funziona il metodo di codifica, E , di figura 7-2. L'ammontare dello sforzo necessario a inventare, verificare e installare un nuovo metodo ogni volta che il metodo vecchio è compromesso o si pensa che sia compromesso ha sempre reso impraticabile il fatto di tenerlo segreto, e pensare che esso sia segreto mentre non lo è può essere molto pericoloso. È questo il punto in cui assume importanza la chiave. Questa è una stringa corta (relativamente) che seleziona una tra molte codifiche potenziali. Al contrario del metodo generale, che raramente può essere modificato, si può modificare la chiave ogni volta che è necessario. In questo modo il nostro modello base è un metodo generale stabile e conosciuto pubblicamente parametrizzato da una chiave segreta e facilmente modificabile.

La mancanza di segretezza dell'algoritmo non deve essere troppo enfatizzata. Pubblicizzando l'algoritmo, il codificatore attiva un grande numero di crittologisti accademici avidi di forzare il sistema in modo da pubblicare articoli che dimostrino quanto sono furbi. Se molti esperti hanno cercato di violare l'algoritmo per cinque anni dopo la sua pubblicazione e nessuno ha avuto successo, probabilmente è abbastanza solido.

La vera segretezza è nella chiave, e la sua lunghezza uno dei punti più importanti. Si consideri una semplice serratura a combinazione. Il principio generale è quello di inserire le cifre in sequenza. Tutti lo sanno, ma la chiave è segreta. Una chiave lunga 3 cifre significa che ci sono 100 possibilità e una chiave di 6 cifre significa che ce n'è 1.000.000. Più lunga è la chiave, più alto è il grado di complicazione con cui il crittanalista ha a che fare. Se si volesse violare il sistema con una ricerca esaustiva dello spazio della chiave, tale grado sarebbe esponenziale rispetto alla lunghezza della chiave. La sicurezza deriva dall'avere un algoritmo forte (ma pubblico) e una chiave lunga. Per evitare che qualcuno vi legga la posta è sufficiente una chiave di 64 bit. Per tenere in scacco le maggiori potenze mondiali sono necessari almeno 256 bit.

Dal punto di vista del crittanalista, il problema della crittanalisi ha tre varianti principali. Quando egli ha a disposizione una certa quantità di testo cifrato e nessun testo in chiaro, si confronta con il problema del **testo cifrato semplice**. I crittogrammi che appaiono nella sezione giochi dei giornali pongono questo tipo di problema. Quando egli ha a disposizione del testo cifrato e il corrispondente testo in chiaro, il problema è noto come il problema del **testo in chiaro conosciuto**. Infine, quando il crittanalista è in grado di codificare parti di testo in chiaro di propria scelta, si ha il problema del **testo in chiaro selezionato**. I crittogrammi dei giornali si possono facilmente risolvere se il crittanalista ha la possibilità di porre una domanda del tipo: "Qual è la codifica di ABCDE?".

I novizi nel campo della crittografia spesso assumono che se un cifrario può sopportare un attacco del testo cifrato, esso è sicuro. Questa assunzione è veramente ingenua. In molti casi il crittanalista può indovinare facilmente alcune parti di testo in chiaro. Ad esempio, la prima

cosa che dicono molti sistemi a condivisione di tempo quando ci si connette è "PLEASE LOGIN". Se potesse usare qualche coppia corrispondente testo in chiaro-testo cifrato, il lavoro del crittanalista diventerebbe molto più semplice. Per raggiungere la sicurezza, il crittografo dovrebbe essere prudente e assicurarsi che il sistema sia inviolabile anche se il suo avversario può decodificare un'arbitraria quantità di testo in chiaro scelto.

I metodi di codifica sono storicamente suddivisi in due categorie: cifrari a sostituzione e cifrari a trasposizione. Tratteremo adesso ciascuno di questi argomenti in una breve storia della crittografia moderna.

Cifrari a sostituzione

In un **cifrario a sostituzione** ogni lettera o gruppo di lettere vengono rimpiazzati da un'altra lettera o gruppo di lettere in modo da mascherarli. Uno dei più vecchi e noti cifrari 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, *attack* diventa *DWWDFN*. Negli esempi, il testo in chiaro è rappresentato in lettere minuscole e il testo cifrato in lettere maiuscole.

Una semplice generalizzazione del cifrario di Cesare permette di trasporre l'alfabeto del testo cifrato di *k* lettere, invece delle solite 3. In questo caso *k* diventa una chiave del metodo generale degli alfabeti trasposti a rotazione. Il cifrario di Cesare può aver fatto impazzire i Cartaginesi, ma da allora non ha più fatto impazzire nessuno.

Il primo miglioramento è di avere ogni simbolo del testo in chiaro, diciamo per semplicità le 26 lettere, mappato in qualche altra lettera. Ad 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

Questo generico sistema è detto di **sostituzione monoalfabetica**, essendo la chiave la stringa di 26 lettere corrispondente all'intero alfabeto. Per la chiave precedente, il testo in chiaro *attack* verrebbe trasformato nel testo cifrato *QZZQEA*.

Alla prima occhiata questo può sembrare un sistema sicuro perché anche se il crittanalista conosce il sistema generale (sostituzione lettera per lettera), non conosce quale delle possibili $26! \equiv 4 \times 10^{26}$ chiavi sia stata utilizzata. Contrariamente al cifrario di Cesare, provarle tutte non è un approccio promettente. Anche con $1\text{ }\mu\text{s}$ a soluzione, un computer potrebbe impiegare 10^{13} anni per provare tutte le chiavi.

Tuttavia, data una quantità sorprendentemente piccola di testo cifrato, è possibile violare facilmente il cifrario. L'attacco base sfrutta le 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 più comuni, o **bigrammi**, sono *th*, *in*, *er*, *re* e *an*. Le combinazioni di tre lettere più comuni, o **trigrammi**, sono *the*, *ing*, *and* e *ion*.

Un crittanalista che tenti di violare un cifrario monoalfabetico potrebbe iniziare contando le frequenze relative di tutte le lettere nel testo cifrato. Quindi potrebbe cercare di assegnare quella più comune alla *e* e quella più comune successiva alla *t*. Quindi potrebbe cercare i trigrammi per trovarne uno dei più comuni del tipo *tXe*, che indica probabilmente

7.1 La sicurezza in rete

che *X* sia *h*. Analogamente, se è frequente il tipo *thYt*, la *Y* probabilmente sta per *a*. Con questa informazione, può cercare un trigramma che occorre frequentemente del tipo *aZW*, che assomiglia molto a *and*. Cercando di indovinare lettere, bigrammi e trigrammi comuni, e riconoscendo i tipi simili di vocali e consonanti, il crittanalista ricostruisce un ipotetico testo in chiaro, lettera per lettera.

Un altro approccio è indovinare una probabile parola o frase. Ad esempio, si consideri il seguente testo cifrato di una società finanziaria (suddiviso 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 probabile in un messaggio da parte di una società finanziaria è *financial*. Sapendo che *financial* ha una lettera ripetuta (*i*), e che ci sono altre quattro lettere tra le loro occorrenze, cerchiamo nel testo cifrato lettere ripetute con questa distanza. Troviamo 12 possibilità, in posizione 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 e 82. Comunque, solo due di queste, 31 e 42, hanno la lettera successiva (corrispondente a *n* nel testo in chiaro) ripetuta nel posto giusto. Di queste due, solamente la 31 ha anche la *a* posizionata correttamente, quindi assumiamo che *financial* inizi in posizione 30. Da questo punto in avanti, è semplice dedurre la chiave utilizzando le statistiche di frequenza per i testi inglesi.

Cifrari a trasposizione

I cifrari a sostituzione conservano l'ordine dei simboli del testo in chiaro ma li trasformano. I **cifrari a trasposizione**, al contrario, riordinano le lettere ma non le trasformano. La figura 7-3 descrive un comune cifrario a trasposizione, quello per colonne. La chiave del cifrario è una parola o frase che non contiene alcuna lettera ripetuta. In questo esempio, la chiave è *MEGABUCK*. Lo scopo della chiave è numerare le colonne, con la colonna 1 sotto la lettera della chiave più vicina all'inizio dell'alfabeto, e così via. Il testo in chiaro è scritto orizzontalmente, in righe. Il testo cifrato viene letto per colonne, iniziando con la colonna la cui lettera della chiave è la più bassa.

M E G A B U C K		
7 4 5 1 2 8 3 6		Testo in chiaro
p l e a s e t r	please	transferonemilliondollarsto
a n s f e r o n	ansferon	myswissbankaccountsixtwotwo
e m i l l i o n	million	
d o l l a r s t	dollar	
o m y s w i s s	myswiss	
b a n k a c c o	bankacc	
u n t s i x t w	untsixtw	
o t w o a b c d	twoabcd	
	AFLLSKSOSELAWAIAUTOSSCTLNMOMANT ESILYNTWRNNNTSOWDPAEDOBUOERIRICXB	

Fig. 7-3 Un cifrario a trasposizione.

Per violare un cifrario a trasposizione, il crittanalista deve per prima cosa essere sicuro di avere a che fare proprio con un cifrario a trasposizione. Controllando la frequenza di *E, T, A, O, I, N* ecc., è facile vedere se queste lettere corrispondono al valore normale per il testo in chiaro. Se è così, il cifrario è chiaramente un cifrario a trasposizione, in quanto i cifrari di questo tipo ogni lettera rappresenta se stessa.

Il passo successivo consiste nell'ipotizzare il numero di colonne. In molti casi è possibile ipotizzare una parola o frase probabile dal contesto del messaggio. Per esempio, supponiamo che il nostro crittanalista sospetti che la frase di testo in chiaro *milliondollars* appaia da qualche parte nel messaggio. Si osservi che i digrammi *MO*, *IL*, *LL*, *LA*, *IR* e *OS* appaiono nel testo cifrato come risultato del fatto che questa frase viene avvolta su se stessa. La lettera *O* del testo cifrato segue la lettera *M* (cioè, sono verticalmente adiacenti in colonna 4) in quanto sono separate nell'ipotetica frase da una distanza equivalente alla lunghezza della chiave. Se fosse stata utilizzata una chiave di lunghezza 7, sarebbero apparsi i digrammi *MD*, *IO*, *LL*, *LL*, *IA*, *OR* e *NS*. Infatti, per ogni lunghezza di chiave, nel testo cifrato viene prodotto un diverso insieme di digrammi. Verificando le varie possibilità, il crittanalista riesce spesso a determinare con facilità la lunghezza della chiave.

Il passo che rimane da fare è ordinare le colonne. Quando il numero delle colonne, *k*, è piccolo, è possibile esaminare ognuna delle *k* (*k* – 1) coppie di colonne per vedere se le frequenze dei digrammi corrispondono a quelle per il testo in chiaro inglese. Si assume che la coppia con la miglior corrispondenza sia posizionata correttamente. A questo punto si prova ognuna delle colonne restanti come successore di questa coppia. Per tentativi si assume che sia giusta la colonna le cui frequenze di digrammi e trigrammi hanno la miglior corrispondenza. La colonna che precede viene trovata con lo stesso criterio. L'intero processo continua fino a che non si sia raggiunto un ordine potenziale. Ci sono buone probabilità che a questo punto il testo in chiaro sia riconoscibile (ad esempio, se appare *milloin*, è ovvio quale sia l'errore).

Alcuni cifrari a trasposizione accettano in ingresso un blocco di lunghezza fissa e producono in uscita un blocco di lunghezza fissa. È possibile descrivere completamente questi cifrari dando una lista che dica in quale ordine i caratteri debbano essere prodotti. Ad esempio, è possibile vedere il cifrario di figura 7-3 come un cifrario a blocchi di 64 caratteri. Il suo risultato è 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, ..., 62. In altre parole, il quarto carattere in ingresso, *a*, è il primo in uscita, seguito dal dodicesimo, *f*, e così via.

Metodo one-time pad

Costruire un cifrario non violabile è in verità molto semplice; la tecnica è nota da decenni. Per prima cosa, si scelga una stringa di bit casuale come chiave. Quindi si converta il testo in chiaro in una stringa di bit, ad esempio utilizzando la rappresentazione ASCII. Infine, si esegua un OR ESCLUSIVO di queste due stringhe, bit per bit. Il testo cifrato che risulta non può essere violato, in quanto qualsiasi testo in chiaro probabile è ugualmente un candidato probabile. Il testo cifrato non offre al crittanalista alcuna informazione. In un esempio di testo cifrato sufficientemente grande, ogni lettera ha la stessa frequenza, così come ogni bigramma e trigramma.

Questo metodo, conosciuto come **one-time pad**, ha sfortunatamente un insieme di svan-

7.1 La sicurezza in rete

taggi di natura pratica. Per iniziare, non è possibile memorizzare la chiave, quindi sia il mittente che il destinatario devono averne una copia scritta. Se uno dei due è esposto a pericoli di cattura, è chiaro che non si desidera dover gestire chiavi scritte. Inoltre, la quantità totale di dati che può essere trasmessa è limitata dalla quantità disponibile di chiave. Se la spia è fortunata e scopre una grande quantità di dati, si può trovare nell'impossibilità di trasmetterli al quartier generale perché ha esaurito la chiave. Un altro problema è la sensibilità del metodo alla perdita o inserzione di caratteri. Se il mittente e il destinatario perdono la sincronizzazione, tutti i dati da quel punto in avanti appariranno troncati.

Con l'avvento dei computer, il metodo one-time pad potrebbe trovare applicazione pratica in certi casi. La sorgente della chiave potrebbe essere un CD speciale contenente diversi gigabit di dati preceduti dalla registrazione di alcune canzoni, trasportato in un contenitore per CD musicali.

Naturalmente, se la rete macina molti gigabit, dover inserire un nuovo CD ogni 5 secondi potrebbe diventare noioso. Per questa ragione, studieremo più avanti gli algoritmi di codifica moderni che possono elaborare quantità arbitrariamente grandi di testo in chiaro.

7.1.2 Due principi fondamentali di crittografia

Anche se nelle pagine che seguono studieremo molti sistemi critografici diversi, esistono principi che li ispirano tutti e che è importante capire. Il primo principio è che tutti i messaggi in codice devono contenere della ridondanza, cioè informazioni non necessarie alla comprensione del messaggio. Un esempio può rendere chiaro il perché questo sia necessario. Consideriamo una società di vendite per corrispondenza, The Couch Potato (TCP), con 60.000 prodotti. Supponiamo che gli efficienti programmati della TCP decidano che i messaggi con gli ordinativi debbano includere il nome del cliente su 16 byte seguiti da un campo dati lungo 3 byte (1 byte per la quantità e 2 byte per il numero del prodotto). Gli ultimi 3 byte dovranno essere codificati utilizzando una chiave molto lunga nota solamente al cliente e alla TCP.

In un primo momento questo può sembrare sicuro, e in certo senso lo è in quanto gli intrusi passivi non possono codificare i messaggi. Sfortunatamente, c'è anche un inconveniente fatale che rende questo sistema inutilizzabile. Supponiamo che una dipendente voglia punire la TCP per averla licenziata. Proprio prima di andarsene, prende con sé la lista (o parte di essa) dei clienti. Lavora di notte per scrivere un programma che genera finti ordini utilizzando nomi di clienti reali. Dato che non ha la lista delle chiavi, essa inserisce dei numeri casuali negli ultimi 3 byte, e invia centinaia di ordinazioni alla TCP. Quando questi messaggi arrivano, il computer della TCP utilizza il nome del cliente per ottenere la chiave e decifra il messaggio. Sfortunatamente per la TCP, quasi tutti i messaggi di 3 byte sono corretti, così il computer inizia a stampare le istruzioni per le spedizioni. Sebbene possa sembrare strano che un cliente ordini 137 altalene per bambini o 240 contenitori per sabbia, per quel che ne sa il computer il cliente può aver deciso di aprire una catena di campi-gioco in franchising. In questo modo un intruso attivo (la ex dipendente) può provocare grossi guai, anche se non capisce i messaggi che il proprio computer sta generando.

Questo problema può essere risolto aggiungendo ridondanza in tutti i messaggi. Ad esem-

pio, se i messaggi d'ordinativo sono estesi a 12 byte, i primi 9 dei quali devono essere degli zeri, allora l'attacco non funziona perché la ex dipendente non può più generare una lunga sequenza di messaggi validi. La morale della storia è che tutti i messaggi devono contenere una ridondanza considerevole in modo che gli intrusi attivi non possano inviare informazioni a caso e ottenere che queste siano interpretate come messaggi validi. D'altra parte, aggiungere ridondanza rende anche molto più facile ai crittanalisti violare i messaggi. Supponiamo che il campo delle vendite per corrispondenza sia molto competitivo e che il maggior concorrente della TCP, The Sofa Tuber (TST), desideri ardente-mente sapere quanti contenitori per sabbia stia vendendo la TCP. A questo scopo la TST intercetta la linea telefonica della TCP. Con lo schema originario con messaggi di 3 byte, la crittanalisi sarebbe praticamente impossibile, perché dopo aver ipotizzato una chiave il crittanalista non ha nessun mezzo per dire se la supposizione sia corretta. Dopo tutto, quasi tutti i messaggi sono tecnicamente corretti. Con il nuovo schema a 12 byte, è facile per il crittanalista distinguere tra un messaggio valido e uno non valido.

Quindi il principio numero 1 della crittografia è che tutti i messaggi devono contenere ridondanza per impedire agli intrusi attivi di ingannare il ricevente portandolo ad agire mediante falsi messaggi. D'altra parte, questa stessa ridondanza rende molto più semplice agli intrusi passivi violare il sistema, perciò su questo punto vi è un po' di incertezza. Inoltre, la ridondanza non dovrebbe mai essere del tipo n zeri all'inizio o alla fine del messaggio, dato che facendo passare questi messaggi attraverso alcuni algoritmi crittografici si ottengono risultati predibibili, rendendo più semplice il compito del crittanalista. Una stringa casuale di parole inglesi potrebbe essere una scelta migliore per la ridondanza. Il secondo principio crittografico è che è necessario prendere qualche misura per impedire agli intrusi attivi di rispedire i vecchi messaggi. Se tali misure non vengono adottate, la nostra ex dipendente potrebbe intercettare la linea telefonica della TCP e ripetere i messaggi validi inviati in precedenza. Un metodo di questo tipo si ottiene includendo in ogni messaggio una marca temporale valida solo per, diciamo, 5 min. Il ricevente può quindi mantenere i messaggi per 5 min, confrontare quelli appena arrivati con quelli precedenti per filtrare i duplicati. I messaggi più vecchi di 5 min possono essere scartati, in quanto qualsiasi duplicato inviato più di 5 min dopo verrà scartato come troppo vecchio. In seguito discuteremo metodi diversi dalla marca temporale.

7.1.3 Algoritmi a chiave segreta

La crittografia moderna utilizza gli stessi principi di base della crittografia tradizionale: trasposizione e sostituzione, ma con enfasi diversa. Tradizionalmente i crittografi utilizzavano algoritmi semplici e si affidavano a chiavi molto lunghe per la loro sicurezza. Ai nostri giorni vale il contrario: l'obiettivo è rendere l'algoritmo di codifica così complesso e involuto che anche se il crittanalista entrasse in possesso di una grande mole di testo cifrato di sua propria scelta, non sarebbe assolutamente capace di dare a esso alcun senso.

È possibile implementare le trasposizioni e le sostituzioni mediante semplici circuiti. La figura 7-4 (a) mostra un dispositivo, noto come **blocco-P** (P sta per permutazione), usato per effettuare una trasposizione su un ingresso di 8 bit. Se gli 8 bit sono dall'alto in basso 01234567, l'uscita di questo particolare blocco-P è 36071245. Mediante una cablatura

interna opportuna, è possibile creare un blocco-P che esegua qualsiasi trasposizione, e farlo praticamente alla velocità della luce.

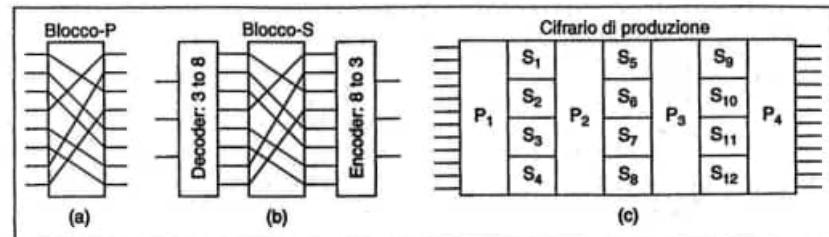


Fig. 7-4 Elementi base di cifrari composti. (a) Blocco-P. (b) Blocco-S. (c) In cascata.

Le sostituzioni vengono eseguite da un **blocco-S**, come mostrato in figura 7-4 (b). In questo esempio, l'ingresso è un testo in chiaro di 3 bit e l'uscita è un testo cifrato di 3 bit. L'ingresso di 3 bit seleziona una delle 8 linee uscenti dal primo componente e la pone a 1; tutte le altre linee sono poste a 0. Il secondo componente è un blocco-P. Il terzo componente codifica la linea selezionata in ingresso nuovamente in binario. Con la cablatura mostrata, se gli 8 numeri ottali 01234567 sono immessi uno dopo l'altro, la sequenza in uscita dovrebbe essere 24506713. In altre parole, 0 viene sostituito dal 2, 1 dal 4 ecc. Di nuovo, è possibile ottenere qualsiasi sostituzione mediante un'opportuna cablatura del blocco-P interno al blocco-S.

Il potere reale di questi elementi di base diventa chiaro solamente quando mettiamo in cascata un'intera serie di blocchi per ottenere un cifrario composto, come è mostrato in figura 7-4 (c). In questo esempio, il primo blocco traspone 12 linee in ingresso. Teoricamente, sarebbe possibile avere come secondo componente un blocco-S che trasformi un numero di 12 bit in un altro numero di 12 bit. D'altra parte, un dispositivo di questo tipo necessita di $2^{12} = 4096$ connessioni incrociate nel suo elemento centrale. Invece, l'ingresso viene suddiviso in 4 gruppi di 3 bit, ciascuno dei quali viene sostituito indipendentemente dagli altri. Anche se questo metodo è meno generale, è comunque molto efficace. Includendo nel cifrario composto un numero sufficientemente grande di componenti, l'uscita può essere il risultato di una funzione dell'ingresso estremamente complicata.

II DES

Nel gennaio 1977 il governo statunitense adottò un cifrario composto sviluppato dall'IBM come il proprio standard ufficiale per le informazioni non classificabili. Questo cifrario, il **DES (Data Encryption Standard)**, fu adottato in maniera estesa dall'industria per prodotti per la sicurezza. Nella sua forma originale non è più sicuro (Wayner, 1995), ma è ancora utilizzato in una sua forma modificata. Vediamo adesso di chiarire come lavora il DES.

La figura 7-5 (a) mostra uno schema del DES. Il testo in chiaro viene codificato in blocchi di 64 bit, generando 64 bit di testo cifrato. L'algoritmo, parametrizzato da una

chiave di 56 bit, ha 19 passi distinti. Il primo passo è una trasposizione indipendente dalla chiave sui 64 bit del testo in chiaro. L'ultimo passo è l'esatto contrario di questa trasposizione. Il passo che precede l'ultimo scambia i 32 bit più a sinistra con i 32 bit più a destra. I rimanenti 16 passi sono funzionalmente identici, ma sono parametrizzati da differenti funzioni della chiave. L'algoritmo è stato progettato per permettere di decodificare con la stessa chiave della codifica. I passi vengono eseguiti in senso inverso.

In figura 7-5 (b) viene illustrata l'operazione di uno di questi passi intermedi. Ogni passo richiede due ingressi di 32 bit e produce due uscite di 32 bit. L'uscita a sinistra è una copia dell'ingresso a destra. L'uscita a destra è l'OR ESCLUSIVO bit per bit dell'ingresso a sinistra, una funzione dell'ingresso a destra, e la chiave per questo passo, K_i . Tutta la complessità risiede in questa funzione.

La funzione è composta da 4 passi, eseguiti in sequenza. Per primo viene costruito un numero di 48 bit, E , espandendo i 32 bit di R_{i-1} secondo una trasposizione e una regola di duplicazione fissate. Secondo, viene eseguito l'OR ESCLUSIVO tra E e K_i . Questa uscita viene quindi suddivisa in 8 gruppi di 6 bit ciascuno, ognuno dei quali viene inviato a un differente blocco-S. Ognuno dei possibili 64 ingressi a un blocco-S viene trasformato in un'uscita di 4 bit. Infine, questi 8×4 bit passano attraverso un blocco-P.

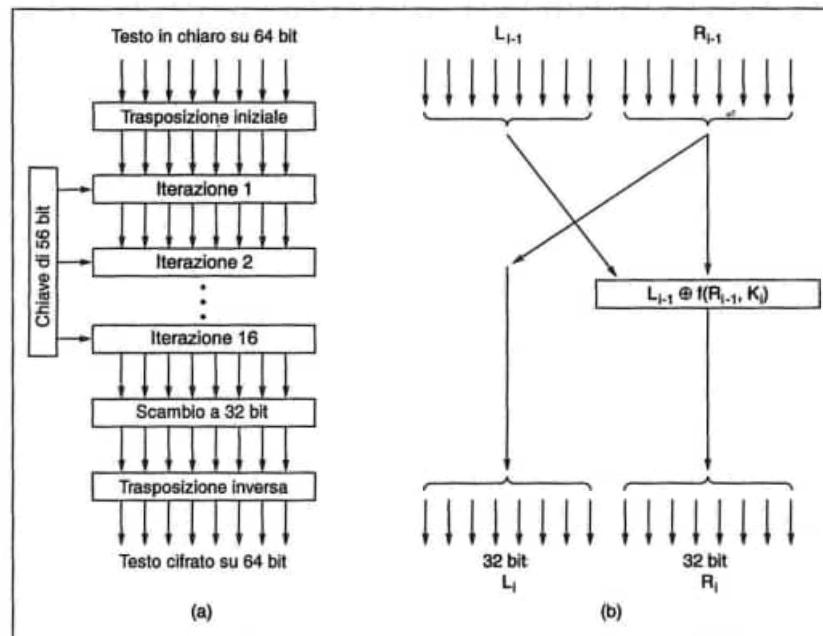


Fig. 7-5 L'algoritmo DES. (a) Schema generale. (b) Dettaglio di un'iterazione.

In ognuna delle 16 iterazioni, viene usata una chiave diversa. Prima che l'algoritmo inizi, alla chiave viene applicata una trasposizione di 56 bit. Subito prima di ogni iterazione, la chiave viene frzionata in 2 unità di 28 bit, ognuna delle quali viene ruotata a sinistra di un numero di bit che dipende dal numero di iterazione. K_i deriva da questa chiave ruotata applicando a essa nuovamente una trasposizione di 56 bit. Ogni volta viene estratto e permutato un diverso sottoinsieme di 48 bit.

DES concatenato

A dispetto di tutta questa complessità, il DES è essenzialmente un cifrario a sostituzione monoalfabetico che utilizza un carattere di 64 bit. Ogni volta che viene dato in ingresso lo stesso testo in chiaro di 64 bit, in uscita si ha lo stesso testo cifrato di 64 bit. Un crittanalista può utilizzare questa proprietà per violare il DES.

Per vedere come si può usare questo cifrario a sostituzione monoalfabetico per forzare il DES, consideriamo come si codifica un messaggio lungo nella maniera ovvia: suddividendolo in blocchi consecutivi di 8 byte (64 bit) e codificandoli uno dopo l'altro con la stessa chiave. Se necessario, l'ultimo blocco è completato a 64 bit. Questa tecnica è conosciuta come **modalità libro codice elettronico** (**electronic code book mode**).

In figura 7-6 troviamo l'inizio di un file di un computer che elenca i premi di produzione annuali che una società ha deciso di assegnare ai propri dipendenti. Questo file è composto da record di 32 byte, 1 per dipendente, nel formato illustrato: 16 byte per il nome, 8 byte per la posizione e 8 byte per il premio di produzione. Ognuno dei 16 blocchi di 8 byte (numerati da 0 a 15) viene codificato tramite il DES.

	Nome	Posizione	Bonus
A_d_a_m_i_s_.	L_e_s_l_i_l_e_	C_l_e_r_k_	\$1111110
B_l_a_c_k_.	R_o_b_i_n_	B_o_s_s_	\$50000000
C_o_l_l_i_l_i_n_s_.	K_i_m_	M_a_j_n_a_g_e_r_i	\$10000000
D_a_v_l_i_s_.	B_o_b_b_l_e_	J_a_n_i_t_o_r_i	\$11111115
	Byte	16	8

Fig. 7-6 Il testo in chiaro di un file codificato come 16 blocchi DES.

Leslie ha appena litigato col suo superiore e non si aspetta un premio di produzione alto. Kim, al contrario è la favorita del boss, e tutti lo sanno. Leslie può accedere al file dopo che è stato codificato, ma prima che sia spedito alla banca. È in grado Leslie di rettificare la sua spiacevole situazione, dato solamente il file codificato?

Nessun problema. Tutto quello che Leslie deve fare è una copia del blocco 11 del testo codificato (quello che contiene il premio di produzione di Kim) e utilizzarlo in sostituzione del blocco 3 del testo cifrato (quello che contiene il premio di produzione di Leslie). Anche senza conoscere il contenuto del blocco 11, Leslie può aspettarsi il risultato atteso. (Un'altra possibilità è quella di copiare il blocco 7 del testo cifrato, ma ciò può essere scoperto più facilmente).

Per prevenire questo tipo di attacchi, il DES (e così tutti i cifrari a blocchi) può venire concatenato in vari modi così che sostituire un blocco nella maniera usata da Leslie possa provocare la cancellazione del testo in chiaro a partire dal blocco rimpiazzato. Un metodo per la concatenazione è la **concatenazione a blocchi del cifrario**. In questo metodo, illustrato in figura 7-7, ogni blocco di testo in chiaro viene messo in OR ESCLUSIVO (#) con il precedente blocco di testo cifrato prima di essere codificato. Di conseguenza, lo stesso blocco di testo in chiaro non corrisponde allo stesso blocco di testo cifrato, e la codifica non è più un grosso cifrario per sostituzione monoalfabetico. Il primo blocco viene messo in OR ESCLUSIVO con un **vettore di inizializzazione**, IV, scelto a caso e che viene trasmesso insieme al testo cifrato.

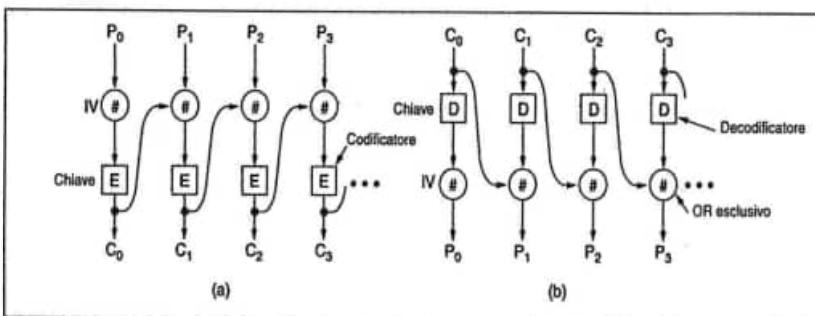


Fig. 7-7 Concatenario e blocchi del cifrario.

Possiamo vedere come lavora la concatenazione a blocchi del cifrario considerando l'esempio in figura 7-7. Iniziamo calcolando $C_0 = E(P_0 \text{ XOR } IV)$. Quindi calcoliamo $C_1 = E(P_1 \text{ XOR } C_0)$ e così via. La decriptazione funziona nell'altro modo, con $P_0 = IV \text{ XOR } D(C_0)$ e così via. Si noti che la cifratura del blocco i è una funzione di tutto il testo in chiaro nei blocchi da 0 a $i - 1$, quindi lo stesso testo in chiaro genera differenti testi cifrati a seconda della sua posizione. Una trasformazione tipo quella fatta da Leslie risulterebbe in una mancanza di significato per i due blocchi a cominciare dal campo del premio di produzione di Leslie. Per un addetto alla sicurezza astuto, questa particolarità potrebbe suggerire il punto da cui iniziare le investigazioni.

La concatenazione a blocchi del cifrario inoltre ha il vantaggio che lo stesso blocco di testo in chiaro non corrisponderà nello stesso blocco di testo cifrato, rendendo la crittanalisi più difficile. In effetti, questa è la ragione principale per cui viene usata.

In ogni modo, la concatenazione a blocchi del cifrario ha lo svantaggio di richiedere che un intero blocco di 64 bit arrivi prima di iniziare la decriptazione. Questo metodo non è adatto a essere utilizzato con terminali interattivi, con i quali le persone possono digitare linee più brevi di otto caratteri e quindi si fermano, aspettando una risposta. Per le codifiche byte per byte, possiamo utilizzare la **modalità feedback del cifrario** mostrata in figura 7-8. In questa figura, vediamo lo stato della macchina di cifra dopo che i byte da 0 a 9 sono stati codificati e spediti. Quando arriva il byte 10 del testo in chiaro, come si

vede in figura 7-8 (a), l'algoritmo DES opera sul registro a scorrimento di 64 bit per generare un testo cifrato di 64 bit. Viene estratto il byte più a sinistra di quel testo cifrato e messo in OR ESCLUSIVO con P_{10} . Questo byte viene trasmesso sulla linea di trasmissione. In aggiunta, il registro a scorrimento viene fatto scorrere a sinistra di 8 bit, in modo che C_2 esca dal margine sinistro e C_{10} entri nella posizione appena liberata al margine destro da C_9 . Notiamo che il contenuto del registro a scorrimento dipende dall'intera storia precedente del testo in chiaro, perciò una configurazione che si ripete più volte nel testo in chiaro verrà codificata in testo cifrato ogni volta in modo differente. Come nel caso della concatenazione a blocchi del cifrario, è necessario un vettore di inizializzazione per iniziare il tutto.

La decriptazione con la modalità feedback del cifrario avviene allo stesso modo della cifratura. In particolare, il contenuto del registro a scorrimento viene *codificato*, non *decodificato*, così il byte selezionato messo in OR ESCLUSIVO con C_{10} per ottenere P_{10} è lo stesso che viene messo in OR ESCLUSIVO con P_{10} per generare C_{10} nel primo caso. Fino a che i due registri a scorrimento rimangono identici, la decriptazione avviene correttamente.

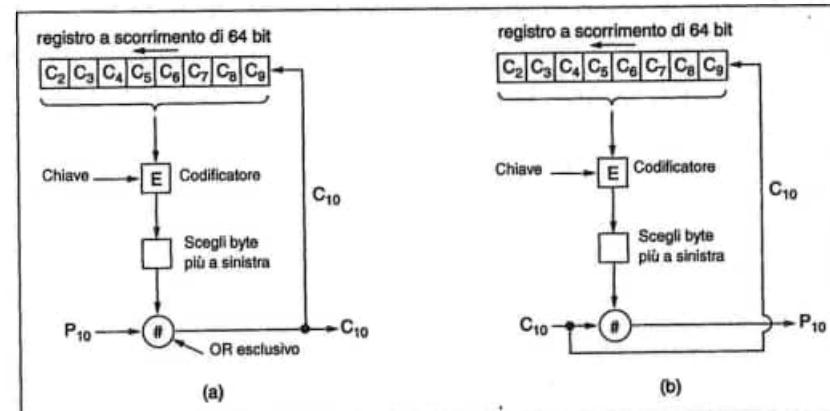


Fig. 7-8 Modalità feedback del cifrario.

Per inciso, si noti che se 1 bit del testo cifrato viene invertito accidentalmente durante la trasmissione, gli 8 byte che vengono decifrati mentre il byte corrotto si trova nel registro a scorrimento saranno anch'essi corrotti. Non appena il byte corrotto esce dal registro a scorrimento, verrà nuovamente generato del testo in chiaro corretto. In questo modo, l'effetto di un singolo bit invertito rimane relativamente localizzato e non distrugge il resto del messaggio.

Ciò nonostante, esistono delle applicazioni per le quali un errore di trasmissione di 1 bit che distrugge 64 bit di testo in chiaro è un effetto inaccettabile. Per queste applicazioni esiste una quarta opzione, la **modalità output feedback**. Questa è identica alla modalità

feedback del cifrario, eccetto che il byte inserito al margine destro del registro a scorrimento è preso subito prima del blocco OR ESCLUSIVO, non subito dopo di esso.

La modalità output feedback gode della proprietà che un errore di 1 bit nel testo cifrato causi solamente un errore di 1 bit nel testo in chiaro risultante. D'altra parte, è meno sicuro degli altri metodi, e dovrebbe essere evitato per l'uso generico. Anche la modalità electronic code book dovrebbe essere evitata, eccetto che in particolari circostanze (ad es., per codificare un singolo numero casuale, come può essere una chiave di sessione). Per le operazioni normali, quando l'input arriva in unità di 8 byte andrebbe utilizzata la concatenazione a blocchi del cifrario (ad es., per codificare i file su disco), mentre per gli input irregolari andrebbe utilizzata la modalità feedback del cifrario, come per gli input da tastiera.

Rompere DES

Il DES è stato al centro di controversie fin dal giorno in cui è stato pubblicato. Era basato su un cifrario sviluppato dalla IBM, chiamato Lucifer, a parte il fatto che la IBM utilizzava una chiave di 128 bit invece di una chiave a 56 bit. Quando il governo federale degli Stati Uniti decise di adottare un cifrario standard per informazioni non classificate, "invitò" l'IBM a "discutere" la questione con la NSA, il corpo governativo per la violazione dei codici, che è il più importante datore di lavoro per matematici e crittografi. La NSA è così segreta che nell'industria è nota la battuta:

D: Cosa sta per NSA?

R: No Such Agency (nessuna simile agenzia).

In effetti, NSA sta per National Security Agency, cioè Agenzia nazionale per la sicurezza. In seguito a tali discussioni, l'IBM ridusse la chiave da 128 a 56 bit e decise di mantenere segreto il processo col quale il DES era stato progettato. Molta gente ha sospettato che la lunghezza della chiave fosse stata ridotta per essere sicuri che solo la NSA potesse forzare il DES, mentre nessuna organizzazione con un bilancio più piccolo potesse esserne capace. Probabilmente il punto focale del progetto segreto è stato nascondere un trabocchetto che potesse rendere ancora più facile alla NSA forzare il DES. Quando un dipendente della NSA chiese discretamente all'IEEE di cancellare una conferenza programmata sulla crittologia, questo non rese più tranquilli gli studiosi.

Nel 1977, due ricercatori in crittografia di Stanford, Diffie e Hellman (1977), progettarono una macchina per violare il DES e stimarono che potesse essere costruita con 20.000.000 di dollari. Tale macchina, fornita di una piccola parte di testo in chiaro e del corrispondente testo cifrato, può trovare la chiave mediante una ricerca esaustiva dello spazio delle 2^{56} chiavi in meno di un giorno. Ai nostri giorni, una macchina di questo tipo potrebbe forse costare 1.000.000 di dollari. In Wiener (1994) viene presentato un progetto dettagliato di una macchina che può forzare il DES mediante una ricerca esaustiva in circa quattro ore. Ecco un'altra strategia. Sebbene la codifica via software sia 1000 volte più lenta di quella via hardware, un computer di casa ad alte prestazioni può comunque eseguire 250.000 codifiche/s a software mentre è probabilmente inattivo per 2.000.000 di secondi al mese. Questo tempo di inattività potrebbe venire usato per forzare il DES. Se qualcuno invia

7.1 La sicurezza in rete

un messaggio a uno dei newsgroup di Internet più famosi, non dovrebbe essere difficile trovare le 140.000 persone che servono a controllare tutte le 7×10^{16} chiavi in un mese. Probabilmente l'idea più nuova per forzare il DES è la **Lotteria Cinese** (Quisquater, Girault, 1991). In questo progetto, ogni radio e televisione dovrebbe essere fornita di un economico processore DES capace di eseguire 1.000.000 di codifiche/s via hardware. Assumendo che 1.200.000.000 persone in Cina possieda una radio o una televisione, ogni volta che il governo cinese volesse decifrare un messaggio cifrato mediante il DES, non fa altro che inviare a tutti la coppia testo in chiaro-testo cifrato, e il 1.200.000.000 di processori inizierebbe a esplorare la propria preassegnata sezione dello spazio delle chiavi. Entro 60 s, verrebbero trovate una o più soluzioni. Per essere sicuri che queste siano comunicate, i processori potrebbero essere programmati in modo da visualizzare o annunciare il messaggio:

CONGRATULAZIONI! HAI APPENA VINTO ALLA LOTTERIA CINESE.
PER RISCUOTERE LA VINCITA, TELEFONA AL NUMERO 1-800-BIG-PRIZE.

La conclusione che si può trarre da queste argomentazioni è che il DES non dovrebbe più essere usato per funzioni importanti. In ogni caso, se 2^{56} è un irrisorio 7×10^{16} , 2^{112} è un magnifico 5×10^{33} . Anche con 1.000.000.000 di processori DES che eseguono 1.000.000.000 di operazioni al secondo, servirebbero 100.000.000 di anni per esplorare esaustivamente lo spazio delle chiavi a 112 bit. Così è nata l'idea di eseguire due volte il DES, con due diverse chiavi di 56 bit.

Sfortunatamente, Merkle e Hellman (1981) hanno sviluppato un metodo che raddoppia i sospetti. È chiamato l'attacco "incontriamoci a metà" (**meet-in-the-middle**) e funziona come segue (Hellman, 1980). Supponiamo che qualcuno abbia codificato due volte una serie di blocchi di testo in chiaro, attraverso la modalità electronic code book. Per alcuni valori di i , il crittanalista ha le coppie corrispondenti (P_i, C_i) , dove

$$C_i = E_{k2}(E_{k1}(P_i))$$

Se adesso applichiamo la funzione di decifrazione D_{k2} a ogni lato di questa equazione, otteniamo

$$D_{k2}(C_i) = E_{k1}(P_i) \quad (7-1)$$

perché codificare x e quindi decodificarlo con la stessa chiave significa ottenere nuovamente x .

L'attacco "incontriamoci a metà" utilizza questa equazione per trovare le chiavi DES, K_1 e K_2 , nel modo seguente:

- Calcolare $R_i = E_i(P_i)$ per tutti i 2^{56} valori di i , dove E è la funzione di codifica del DES. Ordinare questa tabella in ordine ascendente secondo R_i .

2. Calcolare $S_j = D_j(C_1)$ per tutti i 2⁵⁶ valori di j , dove D è la funzione di codifica del DES. Ordinare questa tabella in ordine ascendente secondo S_j .
3. Esaminare sequenzialmente la prima tabella ricercando un R_i equivalente a qualche S_j della seconda tabella. Quando viene trovata un'equivalenza, abbiamo una coppia di chiavi (i, j) tale che $D_j(C_1) = E_i(P_i)$. Potenzialmente, i è K_1 e j è K_2 .
4. Controllare se $E_j(E_i(P_2))$ è equivalente a C_2 . Se lo è, provare tutte le altre coppie (testo in chiaro, testo cifrato). Se non lo è, continuare a ricercare nelle due tabelle per le equivalenze.

Certamente si avranno molti falsi allarmi prima di localizzare le vere chiavi, ma alla fine saranno trovate. Questo attacco richiede solamente 2⁵⁷ operazioni di codifica o decodifica (per costruire le due tabelle), molto meno di 2¹¹². Comunque richiede anche un totale di 2⁶⁰ byte di memoria per le due tabelle, perciò non è attualmente realizzabile in questa forma di base, ma Merkle e Hellman hanno illustrato varie ottimizzazioni e modifiche che richiedono minor memoria a scapito di maggior tempo di computazione. È più importante rilevare che la doppia codifica utilizzando il DES è probabilmente non molto più sicura della singola codifica.

La tripla codifica è un'altra cosa. Fin dal 1979 l'IBM capì che la lunghezza della chiave del DES era troppo breve ed escogitò un modo efficace per aumentarla utilizzando la tripla codifica (Tuchman, 1979).

Il metodo scelto, che è stato allora incorporato nello Standard Internazionale 8732, è illustrato in figura 7-9. In questo esempio vengono usate 2 chiavi e 3 passi. Nel primo passo, il testo in chiaro viene codificato con la chiave K_1 . Nel secondo passo il DES lavora in modalità decifrazione, utilizzando K_2 come chiave. Infine, viene eseguita un'altra codifica con K_1 .

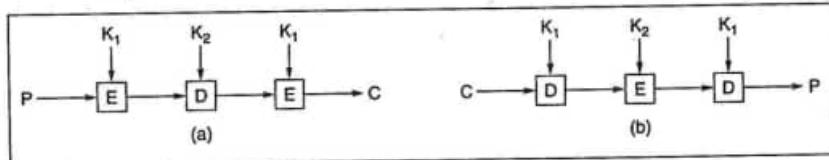


Fig. 7-9 La codifica tripla mediante il DES.

Con questo schema sorgono immediatamente due domande. Primo, perché vengono usate solamente 2 chiavi, invece di 3? Secondo, perché si usa EDE, invece di EEE? La ragione per la quale si utilizzano 2 chiavi invece di 3 è che anche i crittografi più pignoli ammettono che 112 bit sono sufficienti per le applicazioni commerciali coi tempi che corrono. Arrivare a 168 bit aumenterebbe inutilmente i tempi di gestione e trasporto di un'altra chiave.

La ragione per cui si codifica, decodifica e quindi si codifica di nuovo è dovuta al tentativo di mantenere la compatibilità con i sistemi esistenti DES a singola chiave. Entrambe le funzioni di codifica e decodifica sono trasformazioni tra insiemi di numeri a 64 bit. Da

un punto di vista crittografico, le due trasformazioni sono ugualmente forti. Utilizzando l'EDE, comunque, invece dell'EEE, un computer che utilizza la codifica tripla può parlare con quei computer che utilizzano la codifica singola semplicemente impostando $K_1 = K_2$. Questa proprietà permette di inserire gradualmente la codifica tripla, il che non interessa ai crittografi accademici, ma è di considerevole importanza per l'IBM e i suoi clienti. Non si conosce alcun metodo capace di forzare il triplo DES in modalità EDE. Van Oorschot e Wiener (1988) hanno presentato un metodo per velocizzare la ricerca di un EDE di un fattore 16, ma anche con il loro attacco EDE è estremamente sicuro. Per chi voglia solo il meglio, raccomandiamo EEE con 3 chiavi distinte di 56 bit (168 bit in tutto).

Prima di lasciare l'argomento DES, è bene menzionare due recenti sviluppi nella crittanalisi. Il primo sviluppo è la **crittanalisi differenziale** (Biham, Shamir, 1993). Questa tecnica può essere usata per forzare qualsiasi cifrario a blocchi. Lavora iniziando con una coppia di blocchi di testo in chiaro che differiscono solamente per un piccolo numero di bit e controllando attentamente ciò che accade in ogni iterazione interna via via che la codifica procede. In molti casi, alcuni schemi sono molto più comuni di altri, e questa osservazione porta a un attacco probabilistico.

L'altro sviluppo che vale la pena di ricordare è la **crittanalisi lineare** (Matsui, 1994). Questa è in grado di violare il DES conoscendo solamente 2⁴³ testi in chiaro. Lavora ponendo in OR ESCLUSIVO alcuni bit di testo in chiaro e testo cifrato insieme per generare 1 bit. Se eseguito ripetutamente, metà dei bit dovrebbero essere 0 e metà 1. Spesso, comunque, i cifrari introducono uno sbilanciamento in una direzione o in un'altra, e questo sbilanciamento, comunque piccolo, può essere utilizzato per ridurre il lavoro. Per i dettagli, si veda l'articolo di Matsui.

IDEA

Probabilmente insistere a migliorare la sicurezza del DES equivale a incitare un cavallo morto, tuttavia il DES a codifica singola è ancora largamente utilizzato per applicazioni di sicurezza, come quelle bancarie tipo bancomat. Sebbene questa scelta fosse probabilmente adeguata nel momento in cui venne fatta, una decina o più di anni fa, adesso non lo è più.

A questo punto il lettore può legittimamente chiedersi: "Se il DES è così debole, perché nessun altro ha inventato un cifrario a blocchi migliore?". Il fatto è che sono stati proposti molti altri cifrari a blocchi, incluso il BLOWFISH (Schneier, 1994), il Crab (Kaliski, Robshaw, 1994), il FEAL (Shimizu, Miyaguchi, 1988), il KHAFRE (Merkle, 1991), il LOKI91 (Brown *et al.*, 1991), il NEWDES (Scott, 1985), il REDOC-II (Cusick, Wood, 1991), e il SAFER K64 (Massey, 1994). Schneier (1996) discute di tutti questi e di molti altri. Probabilmente il più interessante e importante dei cifrari a blocchi del dopo-DES è **IDEA** (**I**nternational **D**ata **E**ncryption **A**lgorithm) (Lai, Massey, 1990; Lai, 1992). Analizziamo IDEA in maggior dettaglio.

IDEA è stato progettato da due ricercatori in Svizzera, quindi è probabilmente sceso da ogni "consiglio" NSA che possa aver introdotto un trabocchetto segreto. Utilizza una chiave di 128 bit, che lo rende immune ad attacchi brutali, alla Lotteria Cinese e a "incontriamoci a metà" per i decenni a venire. Inoltre è stato progettato per resistere alla

crittanalisi differenziale. Oggi si pensa che nessuna tecnica conosciuta o macchina esistente sia in grado di forzare IDEA.

La struttura base dell'algoritmo assomiglia al DES in quanto i blocchi in ingresso di 64 bit di testo in chiaro vengono macinati in una sequenza di iterazioni parametrizzate per produrre in uscita blocchi di 64 bit di testo cifrato, come mostra la figura 7-10 (a). Dando in pasto il bit di estensione (a ogni iterazione, ogni bit in uscita dipende da ogni bit in ingresso), sono sufficienti 4 iterazioni. Come per tutti i cifrari a blocchi, è possibile utilizzare IDEA in modalità feedback del cifrario e nelle altre modalità del DES.

I dettagli di una iterazione sono illustrati in figura 7-10 (b). Sono usate 3 operazioni, tutte su numeri di 16 bit senza segno. Queste operazioni sono l'OR ESCLUSIVO, l'addizione modulo 2^{16} e la moltiplicazione modulo $2^{16} + 1$. Tutte e 3 possono venire eseguite da un microcomputer a 16 bit ignorando la parte di ordine superiore del risultato. Le operazioni hanno la proprietà che non ci sono due coppie che obbediscono alla proprietà associativa o alla distributiva, rendendo la crittanalisi più difficile. La chiave di 128 bit viene usata per generare 52 sottochiavi di 16 bit ciascuna, 6 per ognuna delle 8 iterazioni e 4 per la trasformazione finale. La decifratura utilizza lo stesso algoritmo della cifratura, solo con sottochiavi differenti.

Di IDEA sono state costruite implementazioni sia software che hardware. La prima implementazione software fu su un 386 a 33 MHz e raggiunse una velocità di codifica di 0,88 Mbps. Su una macchina moderna che gira dieci volte più veloce, si raggiungono i 9 Mbps a software. Al Politecnico di Zurigo (ETH) venne costruito un processore sperimentale VLSI a 25 MHz con una velocità di cifratura di 177 Mbps.

7.1.4 Algoritmi a chiave pubblica

Storicamente il problema della distribuzione delle chiavi è sempre stato il punto debole della maggior parte dei crittosistemi. Non importa quanto sicuro fosse un crittosistema, se un intruso riusciva a entrare in possesso di una chiave, il sistema perdeva valore. Dato che tutti i crittologi accettavano sempre per vero che la chiave di cifratura e quella di decifratura fossero la stessa (o derivabile in maniera semplice l'una dall'altra) e che la chiave dovesse essere distribuita a tutti gli utenti del sistema, sembrava che il problema fosse intrinseco: le chiavi dovevano essere protette dai ladri, ma dovevano anche essere distribuite, perciò non era possibile chiuderle semplicemente in una cassetta di sicurezza di una banca.

Nel 1976 due ricercatori della Stanford University, Diffie e Hellman (1976) proposero un tipo di crittosistema radicalmente nuovo, nel quale le chiavi di cifratura e decifratura fossero differenti, e la chiave di decifratura non potesse essere derivata da quella di cifratura. Nella loro proposta, l'algoritmo di cifratura (con chiave), E , e l'algoritmo di decifratura (con chiave), D , devono rispettare insieme i seguenti tre requisiti. Questi requisiti possono enunciarsi come segue:

1. $D(E(P)) = P$.
2. È estremamente difficile dedurre D da E .
3. E non può venir violato da un attacco con testo in chiaro selezionato.

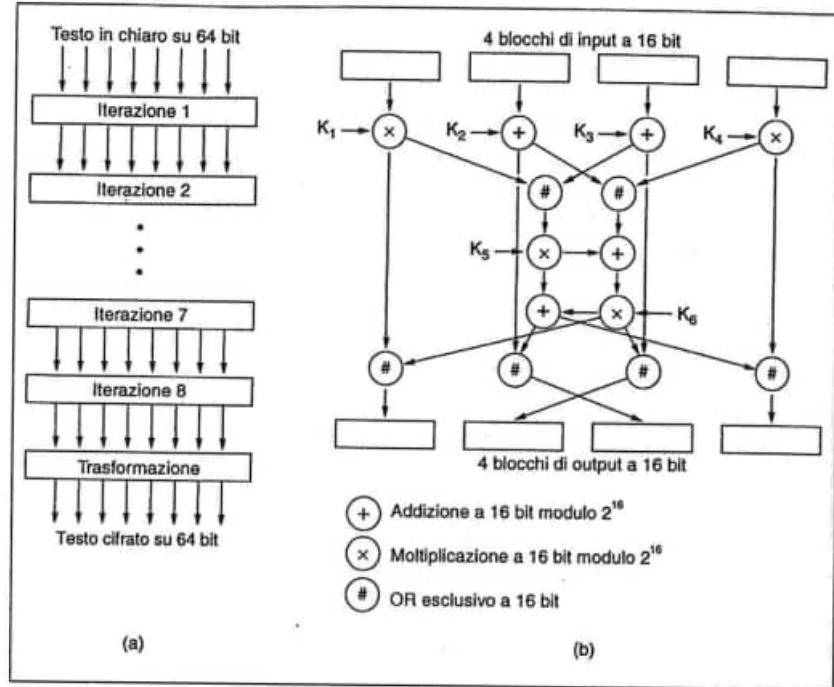


Fig. 7-10 (a) IDEA. (b) Il dettaglio di una iterazione.

Il primo requisito significa che se applichiamo D a un messaggio cifrato, $E(P)$, otteniamo il messaggio in chiaro originale, P . Il secondo requisito parla da solo. Il terzo requisito è necessario perché, come vedremo tra un momento, gli intrusi potrebbero applicare l'algoritmo a un messaggio arbitrario. Sotto queste condizioni, non c'è nessuna ragione per cui la chiave di cifratura non possa essere pubblica.

Il metodo funziona così. Una persona, diciamo Alice, che vuole ricevere dei messaggi segreti, dapprima inventa due algoritmi, E_A e D_A , che rispettino i requisiti suddetti. L'algoritmo (e chiave di cifratura), E_A , è quindi reso pubblico, da qui il nome **crittografia a chiave pubblica** (in contrasto con la crittografia tradizionale a chiave segreta). Questo può essere realizzato inserendo l'algoritmo in un file leggibile da chiunque. Alice pubblica l'algoritmo di decifratura (perché sia liberamente consultabile), ma mantiene segreta la chiave di decifratura. In questo modo, E_A è pubblico, mentre D_A è privato.

Vediamo adesso se riusciamo a risolvere il problema di stabilire un canale sicuro tra Alice e Bob, che non hanno mai avuto alcun precedente contatto. Assumiamo che sia la chiave di cifratura di Alice, E_A , che la chiave di cifratura di Bob, E_B , siano in un file pubblicamente leggibile. (Fondamentalmente, si suppone che tutti gli utenti della rete pubblichino le loro chiavi di cifratura non appena diventano utenti della rete.) Adesso Alice prende il

suo primo messaggio, P , calcola $E_B(P)$ e lo invia a Bob. Bob quindi lo decifra applicando la sua chiave segreta D_B [vale a dire, calcola $D_B(E_B(P)) = P$]. Nessun altro può leggere il messaggio cifrato, $E_B(P)$, perché il sistema di cifratura si suppone sia forte e perché è troppo difficile derivare D_B da E_B nota pubblicamente. Alice e Bob possono adesso comunicare in modo sicuro.

Forse a questo punto torna utile una nota terminologica. La crittografia a chiave pubblica richiede che ogni utente possieda due chiavi: una chiave pubblica, usata da tutto il mondo per cifrare i messaggi da inviare a quell'utente, e una chiave privata, che l'utente utilizza per decifrare i messaggi. Ci riferiremo d'ora in avanti a queste chiavi come chiavi *pubbliche e private*, rispettivamente, e le distingueremo dalle chiavi *segrete* usate sia per la cifratura che per la decifratura nella crittografia tradizionale (chiamate anche *chiavi simmetriche*).

L'algoritmo RSA

Vogliamo trovare degli algoritmi che alla fine soddisfino tutti e tre i requisiti. Grazie ai vantaggi potenziali della crittografia a chiave pubblica, molti ricercatori hanno lavorato assiduamente e sono già stati pubblicati alcuni algoritmi.

Un gruppo del MIT (Rivest *et al.*, 1978) ha trovato un buon metodo. È noto con le iniziali dei suoi tre inventori (Rivest, Shamir, Adleman): **RSA**. Il loro metodo si basa su alcuni principi che derivano da un certo numero di teorie. Riassumeremo adesso come utilizzare il metodo; per i dettagli, si veda l'articolo.

1. Scegliere due grandi numeri primi, p e q , (tipicamente più grandi di 10^{100}).
2. Calcolare $n = p \times q$ e $z = (p - 1) \times (q - 1)$.
3. Scegliere un numero che sia primo rispetto a z e chiamarlo d .
4. Trovare e tale che $e \times d = 1 \text{ mod } z$.

Con questi parametri calcolati in anticipo, siamo pronti a iniziare la cifratura. Dividiamo il testo in chiaro (visto come una stringa di bit) in blocchi, in modo che ogni messaggio di testo in chiaro, P , ricada nell'intervallo $0 \leq P < n$. Questo può essere fatto raggruppando il testo in chiaro in blocchi di k bit, dove k è il più grande intero per cui è vero che $2^k < n$. Per cifrare un messaggio, P , calcoliamo $C = P^e \text{ (mod } n\text{)}$. Per decifrare C , calcoliamo $P = C^d \text{ (mod } n\text{)}$. È possibile provare che per tutti i P nel campo specificato, le funzioni di cifratura e decifratura sono inverse. Per eseguire la cifratura, devi conoscere e ed n . Per eseguire la decifratura, devi conoscere d ed n . Quindi, la chiave pubblica consiste in una coppia (e, n) e la chiave privata consiste di (d, n) .

La sicurezza del metodo si basa sulla difficoltà di fattorizzare i numeri grandi. Se il crittanalista riuscisse a fattorizzare n (pubblicamente nota), potrebbe quindi trovare p e q , e da questi z . Attraverso la conoscenza di z ed e , d può essere trovato usando l'algoritmo di Euclide. Fortunatamente, i matematici hanno provato a fattorizzare i grandi numeri per almeno 300 anni, e l'esperienza accumulata suggerisce che è un problema molto difficile.

7.1 La sicurezza in rete

Secondo Rivest e colleghi, fattorizzare un numero di 200 cifre richiede 4.000.000.000 di anni di tempo macchina; fattorizzare un numero di 500 cifre richiede 10^{25} anni. In entrambi i casi, si utilizzano il miglior algoritmo noto e un computer con un tempo di 1 μs per istruzione. Anche se i computer diventano sempre più veloci, di un ordine di grandezza per decennio, serviranno secoli prima che fattorizzare un numero di 500 cifre diventi possibile, e in quel momento i nostri discendenti potranno scegliere p e q più grandi. Un semplice esempio a scopo pedagogico dell'algoritmo RSA è mostrato in figura 7-11. Per questo esempio abbiamo scelto $p = 3$ e $q = 11$, con $n = 33$ e $z = 20$. Un buon valore per d è $d = 7$, dato che 7 e 20 non hanno fattori in comune. Con queste scelte, e può essere trovato risolvendo l'equazione $7e = 1 \text{ (mod } 20\text{)}$, che restituisce $e = 3$. Il testo cifrato, C , per un messaggio di testo in chiaro, P , è dato da $C = P^3 \text{ (mod } 33\text{)}$. Il testo cifrato è decifrato dal ricevente secondo la regola $P = C^7 \text{ (mod } 33\text{)}$. La figura 7-11 mostra come esempio la cifratura del testo in chiaro "SUZANNE".

Dato che i numeri primi scelti per questo esempio sono davvero piccoli, P deve essere minore di 33, così ogni blocco di testo in chiaro può contenere solamente un singolo carattere. Il risultato è un cifrario per sostituzione monoalfabetico, che non è gran che. Se invece scegliessimo p e $q \sim 10^{100}$, avremmo $n \sim 10^{200}$, quindi ogni blocco potrebbe contenere fino a 664 bit ($2^{664} - 10^{200}$) oppure 83 caratteri di 8 bit, contro gli 8 caratteri del DES.

Testo chiaro		Testo cifrato		Dopo la decrittazione	
Simbolico	Numerico	P^3	$P^3 \text{ (mod } 33\text{)}$	C^7	$C^7 \text{ (mod } 33\text{)}$ Simbolico
S	19	6859	28	13492928512	19 S
U	21	9261	21	1801088541	21 U
Z	26	17576	20	1280000000	26 Z
A	01	1	1	1	1 A
N	14	2744	5	78125	14 N
N	14	2744	5	78125	14 N
E	05	125	26	8031810176	5 E

Calcoli del mittente
Calcoli del ricevente

Fig. 7-11 Un esempio di algoritmo RSA.

Si deve puntualizzare che usare l'RSA nella maniera da noi descritta è come usare il DES in modalità ECB – lo stesso blocco in ingresso restituisce lo stesso blocco in uscita. Dunque è necessario utilizzare qualche forma di concatenamento per la cifratura dei dati. In ogni modo, nella pratica, la maggior parte dei sistemi basati sull'RSA utilizzano la crittografia a chiave pubblica principalmente per distribuire in una volta sola le chiavi di sessione da utilizzare col DES, IDEA o algoritmi simili. Attualmente RSA è troppo lento per cifrare grosse quantità di dati.

Altri algoritmi a chiave pubblica

Anche se RSA è molto usato, non significa che sia il solo algoritmo a chiave pubblica

noto. Il primo algoritmo a chiave pubblica fu l'algoritmo knapsack, o dello zaino (Merkle e Hellman, 1978). L'idea è che qualcuno possiede un grande numero di oggetti, ciascuno con un differente peso. Il proprietario codifica il messaggio scegliendo segretamente un sottoinsieme degli oggetti e mette questi nello zaino. Il peso totale degli oggetti nello zaino è reso pubblico, come la lista di tutti i possibili oggetti. La lista degli oggetti nello zaino è mantenuta segreta. Con determinate restrizioni aggiuntive, si pensava che il problema di immaginare una possibile lista di oggetti con il dato peso fosse computazionalmente impossibile, e costituì la base dell'algoritmo a chiave pubblica.

L'inventore dell'algoritmo, Ralph Merkle, era abbastanza sicuro che questo algoritmo non potesse essere forzato, perciò, per verificare se avesse ragione, offrì una ricompensa di 100 dollari a chiunque fosse riuscito a violarlo. Adi Shamir (la "S" di RSA) lo forzò immediatamente e intascò la ricompensa. Non convinto, Merkle rafforzò l'algoritmo e offrì 1000 dollari di ricompensa a chiunque fosse in grado di violare il nuovo algoritmo. Ron Rivest (la "R" di RSA) lo forzò immediatamente e intascò la ricompensa. Anche se è stato nuovamente ridefinito, è evidente perché l'algoritmo knapsack non sia considerato sicuro e venga usato raramente.

Altri schemi di algoritmi pubblici sono basati sulla difficoltà di calcolare algoritmi discreti (Rabin, 1979). El Gamal (1985) e Schnorr (1991) hanno inventato degli algoritmi che utilizzano questo principio.

Esistono alcuni altri schemi, come quelli basati sulle curve ellittiche (Menezes, Vanstone, 1993), ma le tre categorie principali sono quelle basate sulla difficoltà di fattorizzare grandi numeri, calcolare logaritmi discreti e determinare il contenuto di uno zaino dal suo peso. Si pensa che questi problemi siano veramente difficili da risolvere perché i matematici hanno lavorato senza successo su di essi per molti anni.

7.1.5 Protocolli di autenticazione

L'autenticazione è una tecnica per mezzo della quale un processo verifica che il suo partner nella comunicazione sia colui che si suppone, e non un impostore. Verificando l'identità di un processo remoto a dispetto di uno malintenzionato, l'intrusione attiva è sorprendentemente difficile e richiede protocolli complessi basati sulla crittografia. In questo paragrafo, studieremo alcuni dei molti protocolli di autenticazione usati sulle insicure reti di computer.

Per inciso, alcune persone confondono autorizzazione con autenticazione. L'autenticazione riguarda il problema di capire se attualmente tu stai comunicando o no con uno specifico processo. L'autorizzazione riguarda ciò che a quel processo è permesso. Ad esempio, un processo cliente contatta un file server e dice: "Sono il processo di Scott e voglio cancellare il file *cookbook.old*". Dal punto di vista del file server, ci sono due domande a cui rispondere:

1. Questo è davvero il processo di Scott (autenticazione)?
2. Può Scott cancellare *cookbook.old* (autorizzazione)?

7.1 La sicurezza in rete

L'azione richiesta può aver luogo solo dopo che entrambe le domande hanno ricevuto risposta affermativa senza alcuna ambiguità. La prima domanda è di fatto quella più importante. Una volta che il file server capisce con chi sta parlando, controllare l'autorizzazione è solo questione di ricercare degli elementi in alcune tabelle locali. Per questa ragione, in questo paragrafo ci concentreremo sull'autenticazione.

Il modello generale che tutti i protocolli di autenticazione utilizzano è il seguente. Un primo utente (nella realtà un processo), diciamo Alice, vuole stabilire una connessione sicura con un secondo utente, Bob. Alice e Bob sono talvolta chiamati **principal**i, i protagonisti della nostra storia. Bob è un banchiere col quale Alice vuole fare degli affari. Alice inizia inviando un messaggio a Bob oppure a un fidato **centro di distribuzione delle chiavi** (Key distribution center o KDC), che è sempre onesto. Seguono molti altri scambi di messaggi nelle vari direzioni. Mentre questi messaggi viaggiano, un pericoloso intruso, Trudy, può intercettare, modificare o rispondere a essi con lo scopo di ingannare Alice e Bob o solo per bloccarli.

Ciò nonostante, quando il protocollo è terminato, Alice è sicura di aver parlato con Bob e Bob è sicuro di aver parlato con Alice. Inoltre, in molti protocolli loro due hanno anche stabilito una **chiave di sessione** segreta da utilizzare nelle future conversazioni. In pratica, per ragioni di efficienza, tutto il traffico dei dati viene cifrato usando la crittografia a chiave segreta, anche se la crittografia a chiave pubblica è molto usata per gli stessi protocolli di autenticazione e per stabilire la chiave di sessione.

Il fatto di usare una chiave di sessione nuova, scelta in maniera casuale per ogni nuova connessione, serve a minimizzare il traffico totale che avviene con le chiavi segrete o pubbliche dell'utente, per ridurre l'ammontare di testo cifrato che un intruso può ottenere e per minimizzare il danno fatto se un processo termina e il suo dump finisce nelle mani sbagliate. Fortunatamente, la sola chiave presente sarà poi quella di sessione. Tutte le chiavi permanenti saranno annullate dopo che è stata stabilita la chiave di sessione.

Autenticazione basata su una chiave segreta condivisa

Per il nostro primo protocollo di autenticazione, assumiamo che Alice e Bob condividano già una chiave segreta, K_{AB} . (Nei protocolli formali, abbrevieremo Alice con A e Bob con B , rispettivamente.) Questa chiave condivisa potrebbe essere stata concordata per telefono, o di persona, ma, in ogni caso, non attraverso la rete (insicura).

Questo protocollo si basa su un principio che si ritrova in molti protocolli di autenticazione: una parte invia un numero casuale all'altra, la quale lo trasforma in un modo speciale e quindi restituisce il risultato. Tali protocolli sono chiamati protocolli **sfida-risposta**. In questo e nei prossimi protocolli di autenticazione, sarà usata la notazione seguente:

A, B sono gli identificativi di Alice e Bob

R_i sono le sfide, dove l'indice i indica lo sfidante

K_i sono le chiavi, dove i indica il proprietario; K_s è la chiave di sessione.

La sequenza dei messaggi per il nostro primo protocollo di autenticazione a chiave condivisa è mostrata in figura 7-12. Nel messaggio 1, Alice invia la propria identità, A , a Bob, in un modo che Bob, chiaramente, non ha modo di sapere se questo mes-

saggio venga da Alice o da Trudy, così sceglie una *sfida*, un numero casuale molto grande, R_B , e lo invia indietro a "Alice" come messaggio 2, col testo in chiaro. Alice allora cifra il messaggio con la chiave che condivide con Bob e restituisce il testo cifrato, $K_{AB}(R_B)$ nel messaggio 3. Quando Bob vede questo messaggio, immediatamente capisce che viene da Alice perché Trudy non conosce K_{AB} e quindi non può averlo generato. Inoltre, dato che R_B era stato scelto casualmente in un grande spazio (diciamo, numeri casuali di 128 bit), è molto difficile che Trudy abbia visto R_B e la sua risposta da una precedente sessione.

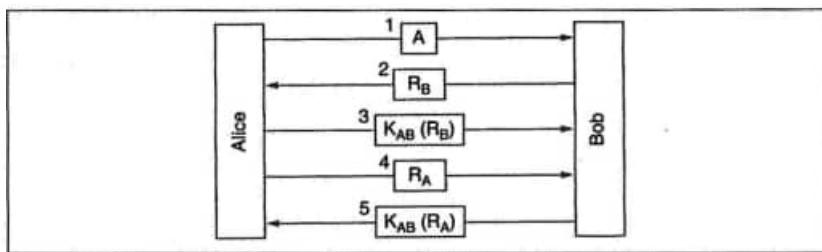


Fig. 7-12 Autenticazione a doppia via usando il protocollo sfida-risposta.

A questo punto, Bob è sicuro di stare parlando con Alice, mentre Alice non è sicura di niente. Per tutto quello che Alice sa, Trudy potrebbe aver intercettato il messaggio 1 e inviato R_B in risposta. Forse Bob è morto la notte scorsa. Per scoprire con chi sta parlando, Alice prende un numero a caso, R_A , e lo invia a Bob come testo in chiaro, col messaggio 4. Quando Bob risponde con $K_{AB}(R_A)$, Alice sa che sta parlando con Bob. Se a questo punto vogliono stabilire una connessione, Alice prende un numero, K_S , e lo invia cifrato con K_{AB} a Bob.

Anche se il protocollo di figura 7-12 funziona, contiene alcuni messaggi extra. Questi possono essere eliminati combinando le informazioni, come mostrato in figura 7-13. Qui è Alice che avvia il protocollo sfida-risposta invece di attendere che lo faccia Bob. Analogamente, mentre lui risponde alla sfida di Alice, Bob invia il proprio protocollo. L'intero protocollo può essere ridotto a tre messaggi invece di cinque.

Questo protocollo è una miglioria rispetto all'originale? In un certo senso, è più breve.

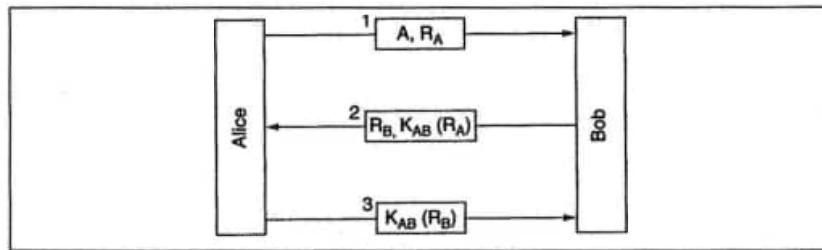


Fig. 7-13 Un protocollo a doppia via più breve.

Sfortunatamente, è anche sbagliato. In certe circostanze, Trudy può battere questo protocollo utilizzando quello che è noto come *attacco riflesso*. In particolare, Trudy può farlo se è possibile aprire sessioni multiple con Bob nello stesso momento. Questa situazione potrebbe verificarsi, ad esempio, se Bob fosse una banca e fosse pronto ad accettare più connessioni simultanee dagli sportelli bancomat.

La figura 7-14 mostra l'attacco riflesso di Trudy. Questo inizia con Trudy che afferma di essere Alice inviando R_T . Bob risponde, come al solito, con la sua sfida, R_B . A questo punto Trudy è bloccata. Cosa può fare? Non conosce $K_{AB}(R_B)$.

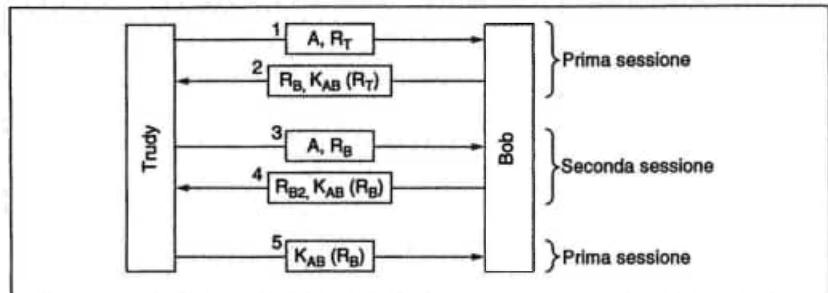


Fig. 7-14 L'attacco riflesso.

Può aprire una seconda sessione col messaggio 3, fornendo R_B preso dal messaggio 2 come la sua sfida. Bob tranquillamente lo decifra e restituisce $K_{AB}(R_B)$ nel messaggio 4. Adesso Trudy ha l'informazione mancante, così può completare la prima sessione e abortire la seconda. Bob è adesso convinto che Trudy sia Alice, così quando lei chiede il saldo del suo conto bancario, lui glielo restituisce senza domande. Perciò quando lei gli chiede di trasferire tutto su un conto segreto in una banca svizzera, lui lo fa senza un attimo di esitazione.

La morale di questa storia:

Progettare un protocollo di autenticazione corretto è più difficile di quel che sembra.

Spesso aiutano le seguenti tre regole fondamentali:

- Chi inizia deve provare chi è, prima che lo faccia chi risponde. In questo caso, Bob aveva dato delle informazioni preziose prima ancora che Trudy desse una qualsiasi prova di chi fosse.
- Chi inizia e chi risponde devono usare chiavi differenti per la prova, anche se questo significa avere due chiavi condivise, K_{AB} e K'_{AB} .
- Chi inizia e chi risponde sceglie la propria sfida da insiemi differenti. Ad esempio, chi inizia deve usare numeri pari e chi risponde deve usare numeri dispari.

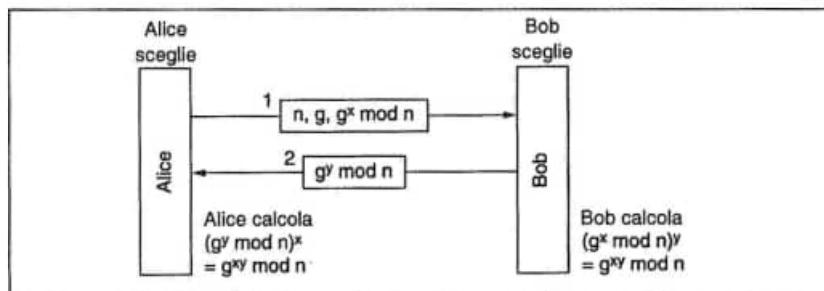
Qui sono state violate tutte e tre le regole, con risultati disastrosi. Notare che il nostro primo protocollo di autenticazione (a 5 messaggi) richiede che Alice provi prima la sua identità, in questo modo quel protocollo non sarà soggetto all'attacco riflesso.

Stabilire una chiave condivisa: lo scambio di chiavi di Diffie-Hellman

Fino a qui abbiamo supposto che Alice e Bob condividano una chiave segreta. E se supponiamo di no? Come possono stabilirne una? Un modo sarebbe che Alice chiamasse Bob e gli desse la sua chiave per telefono, ma lui probabilmente potrebbe dire: "Come faccio a sapere che sei Alice e non Trudy?". Possono provare a darsi un appuntamento, dove ognuno di loro porta il passaporto, una patente di guida e tre carte di credito, ma essendo delle persone molto occupate, è probabile che non riescano a trovare una data accettabile da entrambi, per mesi. Fortunatamente, incredibile a dirsi, c'è un modo per stabilire una chiave segreta alla luce del giorno anche per persone completamente estranee, anche se Trudy registra accuratamente tutti i messaggi.

Il protocollo che permette a entrambi di condividere una chiave segreta si chiama **scambio di chiavi Diffie-Hellman** (Diffie, Hellman, 1976) e funziona nel modo seguente. Alice e Bob devono trovarsi d'accordo su due grandi numeri primi, n e g , con $(n - 1)/2$ ancora primo e certe condizioni da applicare a g . Questi numeri possono essere pubblici, così ciascuno di loro può prendere n e g e dirlo all'altro apertamente. A questo punto Alice sceglie un numero grande (diciamo, di 512 bit), x , e lo tiene segreto. In maniera analoga, Bob sceglie un numero segreto grande, y .

Alice inizia il protocollo per lo scambio delle chiavi inviando a Bob un messaggio contenente $(n, g, g^x \bmod n)$, come mostrato in figura 7-15. Bob risponde inviando ad Alice un messaggio contenente $g^y \bmod n$. Adesso Alice prende il numero che Bob le ha inviato e lo eleva alla x -esima potenza per ottenere $(g^y \bmod n)^x$. Bob esegue un'operazione simile per ottenere $(g^x \bmod n)^y$. Per le leggi dell'aritmetica modulare, entrambi i calcoli producono $g^{xy} \bmod n$. Provare per credere, Alice e Bob adesso condividono una chiave segreta, $g^{xy} \bmod n$.



Trudy, chiaramente, ha visto entrambi i messaggi. Conosce g ed n dal messaggio 1. Se potesse calcolare x e y , potrebbe indovinare la chiave segreta. Il problema è che, dato

soltanente $g^x \bmod n$, non può trovare x . Non si conosce nessun algoritmo pratico per calcolare i logaritmi (discreti modulo un numero primo molto grande).

Per rendere più concreto l'esempio precedente, usiamo i valori (completamente non realistici) $n = 47$ e $g = 3$. Alice sceglie $x = 8$ e Bob sceglie $y = 10$. Entrambi vengono tenuti segreti. Il messaggio di Alice a Bob è $(47, 3, 28)$ perché $3^8 \bmod 47$ è 28. Il messaggio di Bob ad Alice è (17) . Alice calcola $17^8 \bmod 47$, che è 4. Bob calcola $28^{10} \bmod 47$, che è 4. Alice e Bob hanno determinato in maniera indipendente che la chiave segreta adesso è 4. Trudy deve risolvere l'equazione $3^x \bmod 47 = 28$, che può essere svolta mediante ricerca esaustiva per dei numeri piccoli come questi, ma non quando tutti i numeri sono lunghi centinaia di bit. Tutti gli algoritmi attualmente noti impiegano troppo tempo, anche se si utilizza un supercomputer massicciamente parallelo.

A dispetto dell'eleganza dell'algoritmo di Diffie-Hellman, c'è un problema: quando Bob riceve la tripla $(47, 3, 28)$, come fa a sapere che arriva da Alice e non da Trudy? Non c'è modo di saperlo. Sfortunatamente, Trudy può sfruttare questo fatto per ingannare sia Alice che Bob, come illustrato in figura 7-16. Qui, mentre Alice e Bob stanno scegliendo x e y , rispettivamente, Trudy lo intercetta e invia il messaggio 2 a Bob, utilizzando g e n corretti (che sono comunque pubblici) ma con il proprio z al posto di x . Inoltre restituisce il messaggio 3 ad Alice. Più tardi Bob invia il messaggio 4 ad Alice, che di nuovo viene intercettato e preso da Trudy.

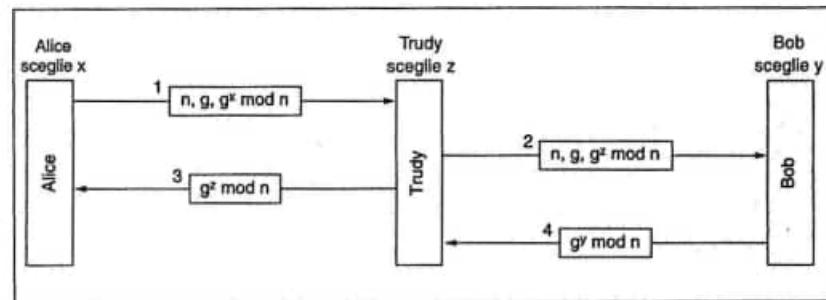


Fig. 7-16 L'attacco della banda del secchio.

A questo punto ognuno usa l'aritmetica modulare. Alice calcola la chiave segreta con $g^{xz} \bmod n$, e lo stesso fa Trudy (per i messaggi ad Alice). Bob calcola $g^{yz} \bmod n$ e così fa Trudy (per i messaggi a Bob). Alice pensa di parlare con Bob così stabilisce una chiave di sessione (con Trudy). Lo stesso fa Bob. Ogni messaggio che Alice invia attraverso la sessione cifrata viene catturato da Trudy, memorizzato, modificato come voluto e quindi (eventualmente) passato a Bob. Analogamente nella direzione opposta. Trudy vede tutto e può modificare tutti i messaggi a sua volontà, mentre sia Alice che Bob si illudono di avere un canale sicuro l'uno verso l'altro. Questo attacco è noto come **attacco della banda del secchio**, perché ricorda vagamente i pompieri di una volta che si passavano secchi lungo la linea, dalla cisterna dei pompieri all'incendio. Viene anche chiamato l'**attacco dell'uomo nel mezzo**, da non confondersi con l'attac-

co meet-in-the-middle sui cifrari a blocchi. Fortunatamente, ci sono algoritmi più complessi che possono battere questo attacco.

L'autenticazione mediante un centro di distribuzione delle chiavi

Stabilire una chiave segreta con un estraneo funziona quasi, ma non abbastanza bene. D'altra parte, probabilmente non vale la pena di farlo. Per parlare con n persone in questo modo, sarebbero necessarie n chiavi. Per la maggior parte delle persone, gestire le chiavi diventerebbe un problema enorme, specialmente se ogni chiave dovesse essere registrata su una carta intelligente.

Un approccio diverso è quello di introdurre un centro di distribuzione delle chiavi affidabile (KDC). In questo modello ogni utente ha una singola chiave condivisa con il KDC. L'autenticazione e la gestione delle chiavi di sessione adesso passa attraverso il KDC. Il protocollo di autenticazione KDC più semplice che si conosce è quello della **rana dalla bocca larga** (wide-mouth frog) (Burrows *et al.*, 1990) (chiamato così per via di un soprannome dato al suo inventore (Burrows) durante gli anni di università). Questo protocollo è descritto in figura 7-17.

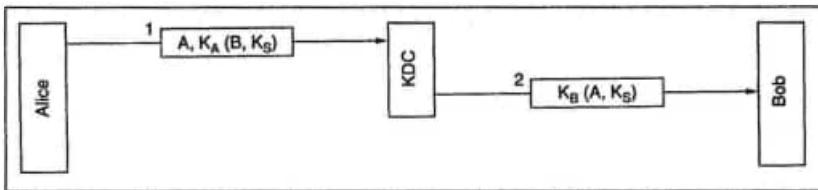


Fig. 7-17. Il protocollo di autenticazione della rana dalla bocca larga.

L'idea dietro al protocollo della rana dalla bocca larga è semplice: Alice prende una chiave di sessione, K_S , e informa il KDC di voler parlare con Bob usando K_S . Questo messaggio è cifrato con una chiave segreta che Alice condivide (solamente) con il KDC, K_A . Il KDC decifra questo messaggio, estraendo identità di Bob e la chiave di sessione. Quindi costruisce un nuovo messaggio contenente identità di Alice e la chiave di sessione, e invia questo messaggio a Bob. Questa codifica viene fatta con K_B , la chiave segreta che Bob condivide con il KDC. Quando Bob decifra il messaggio, viene a sapere che Alice vuol parlare con lui e qual è la chiave che lei vuole usare.

Qui l'autenticazione avviene gratuitamente. Il KDC sa che il messaggio 1 deve arrivare da Alice, dato che nessun altro sarebbe stato capace di cifrarlo con la chiave segreta di Alice. Analogamente, Bob sa che il messaggio 2 deve venire dal KDC, in cui ha fiducia, dato che nessun altro conosce la sua chiave segreta.

Sfortunatamente, questo protocollo ha un serio inconveniente. Trudy ha bisogno di denaro, così si rende conto di quali servizi leciti può fare per Alice, le fa un'offerta attraente e ottiene il lavoro. Dopo aver svolto il lavoro, Trudy chiede cortesemente ad Alice di essere pagata con un accredito bancario. Alice allora stabilisce una chiave di sessione col proprio banchiere, Bob. Quindi manda a Bob un messaggio richiedendo che il denaro sia trasferito sul conto di Trudy.

Nel frattempo, Trudy è ritornata al suo vecchio comportamento, cercando di origliare sulla rete. Copia sia il messaggio 2 in figura 7-17 che la richiesta di trasferimento del denaro che segue. Più tardi, li rispedisce entrambi a Bob. Bob li riceve e pensa: "Alice deve aver assunto nuovamente Trudy. Chiaramente deve aver fatto un buon lavoro". Bob allora trasferisce un'uguale somma di denaro dal conto di Alice a quello di Trudy. Qualche tempo dopo la 50-esima coppia di messaggi, Bob esce dal suo ufficio per trovare Trudy e offrirle un grosso prestito in modo che possa espandere il suo giro di affari di successo. Questo problema è conosciuto come **attacco di risposta** (reply attack).

Esistono diverse soluzioni all'attacco di risposta. La prima è quella di inserire una marca di tempo in ogni messaggio. In questo modo se qualcuno riceve un messaggio obsoleto, può scartarlo. Il problema con questo approccio è che gli orologi sulla rete non sono mai sincronizzati esattamente tra loro, così ci deve essere un certo intervallo di validità per la marca di tempo. Trudy può rispondere al messaggio durante quest'intervallo e appropriarsene.

La seconda soluzione è inserire in ogni messaggio un numero di messaggio unico, una volta sola, chiamato di solito **nonce**. In seguito ogni parte deve ricordarsi tutti i nonce precedenti e rifiutare qualsiasi messaggio che ne contenga uno già usato. Ma i nonce sono da ricordarsi per sempre, per paura che Trudy provi a rispondere con un messaggio vecchio di cinque anni. Inoltre, se qualche macchina si guasta e perde la sua lista di nonce, è nuovamente vulnerabile a un attacco di risposta. È possibile combinare marche di tempo e nonce in modo da limitare il periodo in cui si devono ricordare i nonce, ma chiaramente il protocollo inizia a diventare un po' troppo complicato.

Un approccio più sofisticato all'autenticazione è quello di usare un protocollo sfida-risposta a più vie. Un esempio molto noto di un protocollo di quel tipo è il **protocollo di autenticazione Needham-Schroeder** (Needham, Schroeder, 1978), una variante del quale è illustrata in figura 7-18.

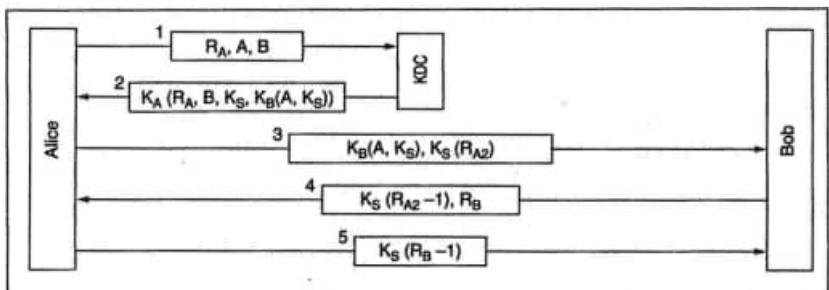


Fig. 7-18 Il protocollo di autenticazione Needham-Schroeder.

Il protocollo inizia con Alice che informa il KDC di voler parlare con Bob. Questo messaggio contiene un numero casuale grande, R_A , come un nonce. Il KDC restituisce il messaggio 2 che contiene il numero casuale di Alice, una chiave di sessione e un biglietto che lei può inviare a Bob. Lo scopo del numero casuale R_A è di assicurare ad

Alice che il messaggio 2 sia nuovo, e non una replica. Viene anche inserita l'identità di Bob nel caso in cui Trudy abbia la divertente idea di rimpiazzare B nel messaggio 1 con la propria identità così che il KDC codifichi il biglietto alla fine del messaggio 2 con K_T invece di K_B . Il biglietto cifrato con K_B viene incluso all'interno del messaggio cifrato per evitare che Trudy lo rimpiazzi con qualcosa' altro sulla via del ritorno ad Alice.

Alice adesso invia il biglietto a Bob, insieme a un nuovo numero casuale, R_{A2} , cifrato con la chiave di sessione K_S . Nel messaggio 4, Bob restituisce $K_S(R_{A2} - 1)$ per provare ad Alice che sta parlando con il vero Bob. Restituire $K_S(R_{A2})$ non avrebbe funzionato, perché Trudy lo avrebbe potuto rubare dal messaggio 3.

Dopo aver ricevuto il messaggio 4, Alice è ormai convinta di parlare con Bob, e che finora non sia stata riusata alcuna risposta. Dopo tutto, ha generato R_{A2} solo qualche millisecondo fa. Lo scopo del messaggio 5 è di convincere Bob che sia davvero Alice quella con cui sta parlando, e che non sia stata riutilizzata nessuna risposta. Dato che ciascuna parte ha generato una sfida e ha inviato una risposta, è eliminata la possibilità di qualunque tipo di attacco di risposta.

Anche se questo protocollo sembra abbastanza solido, ha comunque una debolezza. Se Trudy fa di tutto per ottenere una vecchia chiave di sessione col testo in chiaro, può iniziare una nuova sessione con Bob rispondendo al messaggio 3 corrispondente alla chiave compromessa e convincerlo di essere Alice (Denning, Sacco, 1981). Questa volta può saccheggiare il conto bancario di Alice senza dover eseguire il servizio di legittimazione neppure una volta.

Successivamente Needham e Schroeder pubblicarono un protocollo che corregeva questo problema (Needham, Schroeder, 1987). Nello stesso numero della stessa rivista, anche Otway e Rees (1987) ne pubblicarono uno che risolveva il problema in maniera più breve. La figura 7-19 mostra il protocollo Otway-Rees leggermente modificato.

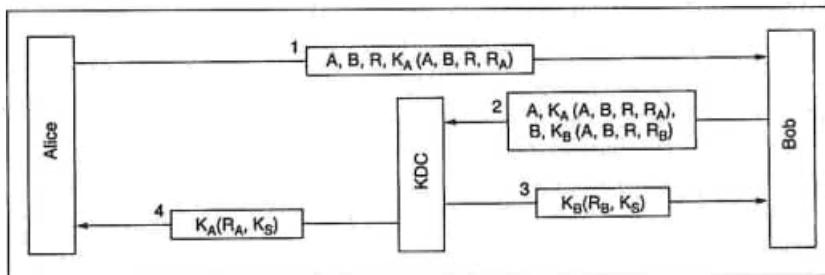


Fig. 7-19 Il protocollo di autenticazione Otway-Rees (leggermente semplificato).

Nel protocollo Otway-Rees Alice inizia generando un paio di numeri casuali: R , che verrà usato come un identificatore comune, e R_A che Alice userà per sfidare Bob. Quando Bob riceve questo messaggio, costruisce un nuovo messaggio dalla parte cifrata del messaggio di Alice, e un analogo di sua iniziativa. Entrambe le parti cifrate con K_A e K_B identificano Alice e Bob, contengono l'identificatore comune e una sfida.

Il KDC controlla per vedere se R è lo stesso in entrambe le parti. Potrebbe non esserlo se

Trudy avesse manomesso il messaggio 1 con R o rimpiazzato parte del messaggio 2. Se i due R_S coincidono, il KDC si convince che il messaggio di richiesta da Bob sia valido. Quindi genera una chiave di sessione e la codifica due volte, una volta per Alice e una volta per Bob. Ogni messaggio contiene il numero casuale del destinatario, come prova che è il KDC, e non Trudy, ad aver generato il messaggio. A questo punto Alice e Bob sono in possesso della stessa chiave di sessione e possono iniziare a comunicare. La prima volta che si scambiano messaggi di dati, ognuno di loro può vedere che l'altro ha una copia di K_S identica, così l'autenticazione è completa.

Autenticazione mediante Kerberos

Un protocollo utilizzato in molti sistemi reali è **Kerberos**, basato su una variante di Needham-Schroeder. Il suo nome deriva da un cane con più teste della mitologia greca che stava a guardia dell'ingresso dell'inferno. Kerberos è stato progettato al MIT per permettere agli utenti di accedere alle risorse di rete in maniera sicura. La più grossa differenza col protocollo Needham-Schroeder è il suo presupposto che tutti gli orologi siano abbastanza ben sincronizzati. Il protocollo passa attraverso diverse iterazioni. V4 è la versione più utilizzata nell'industria, perciò descrivremo quella. Diremo qualcosa anche sul suo successore, V5. Per ulteriori informazioni, si vedano Neuman, Ts'o (1994); Steiner *et al.* (1988).

Kerberos coinvolge tre server oltre ad Alice (un calcolatore cliente):

- il server di autenticazione (SA): verifica gli utenti al momento del login
- il server di gestione di convalide (SGC): distribuisce "convalide per provare identità"
- il server Bob: in realtà svolge il lavoro che serve ad Alice.

SA è simile a KDC in quanto condivide una password segreta con ogni utente. Il lavoro di SGC è inviare convalide che possano convincere il server corrente che colui che esibisce la convalida sia effettivamente chi pretende di essere.

Per iniziare una sessione, Alice si siede a una workstation pubblica arbitraria e digita il suo nome. La workstation invia il suo nome all'SA come testo in chiaro, come mostra la figura 7-20. Ciò che torna indietro è una chiave di sessione e una convalida, $K_{SGC}(A, K_S)$, per l'SGC. Questi elementi sono messi insieme e cifrati mediante la chiave segreta di Alice, in modo che solo Alice possa decifrarli. Solamente quando arriva il messaggio 2, la workstation richiede la password ad Alice. La password quindi viene usata per generare K_A , con lo scopo di decifrare il messaggio 2 e ottenere la chiave di sessione e la convalida SGC in esso contenute. A questo punto, la workstation sovrascrive la password di Alice, per essere sicuri che rimanga nella workstation solo per pochi millisecondi. Se Trudy cercasse di connettersi come Alice, la password da lei immessa sarebbe errata e la workstation lo rivelerebbe perché la parte standard del messaggio 2 sarebbe scorretta.

Dopo essersi connessa, Alice può dire alla workstation di voler contattare il file server Bob. La workstation allora invia il messaggio 3 all'SGC richiedendo una convalida da usare con Bob. L'elemento chiave di questa richiesta è $K_{SGC}(A, K_B)$, che viene cifrato con la chiave segreta dell'SGC e usato per provare che il mittente è davvero Alice. L'SGC risponde creando una chiave di sessione, K_{AB} , perché Alice la usi con Bob. Vengono

restituite due versioni di tale chiave. La prima viene codificata con K_A , in modo che Alice sia in grado di leggerla. La seconda viene codificata con K_B , in modo che Bob sia in grado di leggerla.

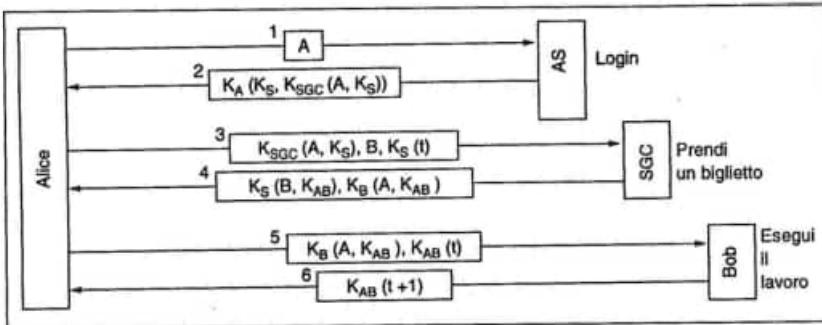


Fig. 7-20 L'operazione di Kerberos V4.

Trudy può copiare il messaggio 3 e tentare di riusarlo, ma sarà smascherata dalla marca di tempo cifrata, t , inviata col messaggio. Trudy non può sostituire la marca di tempo con una più recente, in quanto non conosce K_S , la chiave di sessione che Alice usa per comunicare con l'SGC. Anche se Trudy rispondesse velocemente col messaggio 3, tutto ciò che otterrebbe sarebbe un'altra copia del messaggio 4, che lei non ha saputo decifrare la prima volta e non può decifrare neppure la seconda.

Adesso Alice può inviare a Bob il K_{AB} per stabilire una sessione con lui. Anche questo scambio contiene una marca di tempo. La risposta è la prova per Alice che adesso lei sta parlando con Bob, e non con Trudy.

Dopo questa serie di scambi, Alice può comunicare con Bob sotto la protezione di K_{AB} . Se più tardi decidesse di voler parlare con un altro server, Carol, ripete il messaggio 3 all'SGC, solo che questa volta specifica C al posto di B . L'SGC risponderà subito con una convalida cifrata con K_C che Alice può inviare a Carol e che a Carol proverà che arriva da Alice.

Lo scopo di questa operazione è che adesso Alice può accedere a tutti i server in rete in maniera sicura, e che la sua password non transiterà mai su rete. Infatti, rimarrà nella sua workstation solo per pochi millisecondi. In ogni caso, si noti che ogni server effettua la sua autorizzazione. Quando Alice presenta la sua convalida a Bob, questo prova a Bob solo chi è che l'ha inviata. Più precisamente, ciò che Alice può fare dipende solamente da Bob.

Dato che i progettisti di Kerberos non si aspettavano che tutto il mondo si affidasse a un singolo server di autenticazione, previdero la possibilità di avere più **domini**, ognuno con i suoi SA e SGC. Per ottenere una convalida per un server in un dominio diverso, Alice dovrebbe richiedere al proprio SGC una convalida accettata dall'SGC dell'altro dominio. Se il dominio remoto è registrato con l'SGC locale (allo stesso modo in cui lo sono i server locali), l'SGC locale restituirà ad Alice una convalida corretta per l'SGC remoto.

Quindi essa può lavorare con quello, ad esempio può ottenere convalide per i server in quel dominio. Si noti, comunque, che lavorare tra parti che si trovano in due domini diversi implica che ognuna debba aver fiducia nello SGC dell'altra.

Kerberos V5 è più raffinato del V4 e ha un overhead maggiore. Inoltre utilizza l'OSI ASN.1 (Abstract Syntax Notation 1) per descrivere i tipi di dati e ha delle piccole differenze nel protocollo. Inoltre, le convalide hanno una vita più lunga, permette il rinnovo delle convalide e l'invio di convalide postdate. In aggiunta, almeno in teoria, non è DES-dipendente, come è V4, e ammette domini multipli.

Autenticazione mediante crittografia a chiave pubblica

L'autenticazione multipla può essere fatta mediante la crittografia a chiave pubblica. Per cominciare, assumiamo che Alice e Bob conoscano già le reciproche chiavi pubbliche (un aspetto non banale). Vogliono stabilire una sessione e quindi per quella sessione usano la crittografia a chiave segreta, in quanto è tipicamente da 100 a 1000 volte più veloce della crittografia a chiave pubblica. Lo scopo dello scambio iniziale è quindi quello di autenticarsi reciprocamente e accordarsi su una chiave di sessione segreta e condivisa.

Questo accordo può essere fatto in più modi. Un modo tipico è mostrato in figura 7-21. Qui Alice inizia cifrando la sua identità e un numero casuale, R_A , usando la chiave pubblica (o di codifica), E_B . Quando Bob riceve questo messaggio, non ha idea se questo arrivi da Alice o da Trudy, ma sta al gioco e restituisce ad Alice un messaggio contenente lo R_A di Alice, un proprio numero casuale, R_B , e propone una chiave di sessione K_S .

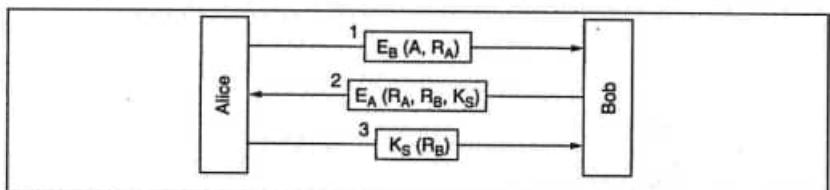


Fig. 7-21 Autenticazione multipla mediante la crittografia a chiave pubblica.

Quando Alice riceve il messaggio 2, lo decifra usando la sua chiave privata. Trova in esso R_A , che la rassicura. Il messaggio dovrebbe venire da Bob, dato che Trudy non ha modo di determinare R_A . Inoltre, deve essere nuovo e non una replica, dato che lei ha appena spedito R_A a Bob. Alice accetta la sessione restituendo il messaggio 3. Quando Bob vede R_B cifrato con la chiave di sessione da lui appena generata, sa che Alice ha ricevuto il messaggio 2 e verificato R_A .

Cosa può fare Trudy per cercare di sconvolgere questo protocollo? Può costruire il messaggio 1 e ingannare Bob fingendosi Alice, ma Alice vedrà un R_A che non ha inviato e non andrà oltre. Trudy non può falsificare il messaggio 3 in maniera convincente perché non conosce R_B o K_S e non li può determinare senza la chiave privata di Alice. È sfortunata.

Comunque, il protocollo ha una debolezza: assume che Alice e Bob conoscano già le reciproche chiavi pubbliche. Supponiamo che questo non sia vero. Alice potrebbe inviare

a Bob la sua chiave pubblica nel primo messaggio e chiedere a Bob di restituirla la sua nel prossimo messaggio. Il problema con questo approccio è che è soggetto a un attacco della banda del secchio. Trudy può catturare il messaggio di Alice verso Bob e restituire la propria chiave pubblica ad Alice. Alice a questo punto crede di avere una chiave per comunicare con Bob, quando invece ha una chiave per comunicare con Trudy. Adesso Trudy può leggere tutti i messaggi cifrati con quello che Alice pensa sia la chiave pubblica di Bob.

È possibile evitare lo scambio di chiavi pubbliche iniziale se tutte sono registrate in una base di dati pubblica. In questo modo Alice e Bob possono prendere le reciproche chiavi pubbliche dalla base di dati. Sfortunatamente Trudy può ancora sferrare l'attacco della banda del secchio intercettando le richieste alla base di dati e inviando delle richieste simulate contenenti la propria chiave pubblica. Dopo tutto, come fanno Alice e Bob a sapere che le risposte arrivino davvero dalla base di dati e non da Trudy?

Rivest e Shamir (1984) hanno escogitato un protocollo che contrasta l'attacco della banda del secchio di Trudy. Nel loro **protocollo interlock**, dopo lo scambio delle chiavi pubbliche, Alice invia solamente metà del suo messaggio a Bob, diciamo solo i bit pari (dopo la cifratura). Bob allora risponde con i propri bit pari. Dopo aver ricevuto i bit pari di Bob, Alice invia i suoi bit dispari, e lo stesso fa Bob.

Il trucco qui è che quando Trudy prende i bit pari di Alice, non è in grado di decifrare il messaggio, anche se Trudy ha la chiave privata. Di conseguenza, non è in grado di cifrare nuovamente i bit pari usando la chiave pubblica di Bob. Se invia un falso messaggio a Bob, il protocollo proseguirà, ma Bob in breve scoprirà che l'intero messaggio assemblato non ha senso e capisce che è stato imbrogliato.

7.1.6 Firme digitali

L'autenticità di molti documenti legali, finanziari e altri ancora è determinata dalla presenza o dall'assenza di una firma autografa. Le fotocopie non contano. È necessario trovare una soluzione a questi problemi per i sistemi telematici che eliminano lo scambio fisico di documenti fatti di carta e inchiostro.

Il problema di inventare un'alternativa alle firme autografe è difficile. Essenzialmente, ciò che serve è un sistema col quale ogni corrispondente possa inviare un messaggio "firmato" a un altro corrispondente in modo che:

1. Il destinatario possa verificare l'identità rivendicata dal mittente.
2. Il mittente non possa in seguito disconoscere il contenuto del messaggio firmato.
3. Il destinatario non abbia la possibilità di inventarsi da solo il messaggio firmato.

Il primo requisito è necessario, ad esempio, nei sistemi finanziari. Quando il computer di un cliente ordina al computer della banca di comperare una tonnellata d'oro, il computer della banca deve essere sicuro che il computer che ha inviato l'ordine appartenga veramente alla società sul cui conto va addebitato l'acquisto.

Il secondo requisito è necessario per proteggere la banca contro le frodi. Supponiamo che la banca compri la tonnellata d'oro, e subito dopo il prezzo dell'oro crolli vertiginosa-

mente. Un cliente disonesto potrebbe far causa alla banca, pretendendo di non aver mai inviato alcun ordine per l'acquisto dell'oro: quando la banca porta il messaggio in tribunale, il cliente nega di averlo spedito.

Il terzo requisito è necessario per proteggere il cliente nel caso in cui il prezzo dell'oro salga e la banca cerchi di creare un messaggio firmato nel quale il cliente chieda un lingotto d'oro invece di una tonnellata.

Firme a chiave segreta

Un approccio per le firme digitali è quello di avere un'autorità centrale che conosce tutto e di cui tutti si fidano, detta il Grande Amico (*GA*). Ciascun utente quindi sceglie una chiave segreta e la porta a mano all'ufficio di *GA*. In questo modo solamente Alice e *GA* conoscono il segreto di Alice, K_A , e così via.

Quando Alice vuole inviare un messaggio firmato di testo in chiaro, P , al suo banchiere, Bob, genera $K_A(B, R_A, t, P)$ e lo invia come in figura 7-22. *GA* vede che il messaggio viene da Alice, lo decifra e invia un messaggio a Bob. Il messaggio per Bob contiene il testo in chiaro del messaggio di Alice e inoltre il messaggio firmato $K_{GA}(A, t, P)$, dove t è una marca di tempo. A questo punto Bob esegue la richiesta di Alice.

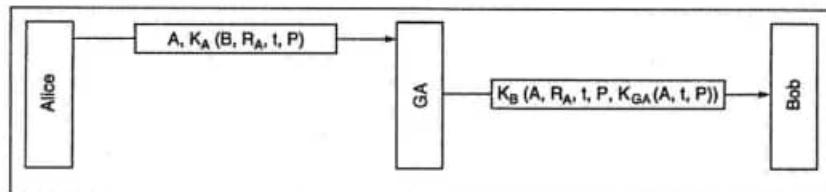


Fig. 7-22 Firme digitali con il Grande Amico.

Cosa accadrà se in seguito Alice negherà di aver spedito il messaggio? Il punto 1 è che chiunque fa causa a chiunque altro (almeno negli Stati Uniti). Alla fine, quando il caso arriverà in tribunale e Alice negherà decisamente di aver spedito a Bob il messaggio in discussione, il giudice chiederà a Bob come faccia a essere sicuro che il messaggio in discussione venga da Alice e non da Trudy. Per prima cosa Bob puntualizzerà che *GA* non accetterebbe mai un messaggio da Alice che non sia cifrato con K_A , perciò non c'è alcuna possibilità che Trudy abbia inviato a *GA* un messaggio falso da parte di Alice.

Quindi Bob fornirà drammaticamente la Prova A, $K_{GA}(A, t, P)$. Bob dirà che questo è un messaggio firmato da *GA* che proverà che Alice ha inviato P a Bob. Il giudice allora chiederà a *GA* (in cui tutti credono) di decifrare la Prova A. Quando *GA* testimonierà che Bob sta dicendo la verità, il giudice deciderà in favore di Bob. Il caso sarà chiuso.

Un potenziale problema con il protocollo di firma di figura 7-22 è che Trudy potrebbe rispondere a entrambi i messaggi. Per minimizzare questo problema, si usano le marche di tempo in maniera sistematica. Inoltre, Bob potrebbe controllare tutti i messaggi recenti per vedere se R_A sia stata usata in qualcuno di essi. Se è così, il messaggio verrà scartato come ripetizione. Si noti che Bob rifiuterà qualsiasi messaggio molto vecchio in base alla

marca di tempo. Per proteggersi contro gli attacchi con risposte istantanee, Bob controllerà R_A in ogni messaggio in arrivo per vedere se un messaggio simile sia stato ricevuto da Alice nelle ultime ore. Se questo non si verifica, Bob potrà assumere sicuramente che questa sia una richiesta nuova.

Firme a chiave pubblica

Un problema strutturale nell'utilizzo della crittografia a chiave segreta per le firme digitali è che ognuno deve essere d'accordo nel dare fiducia al Grande Amico. Inoltre, il Grande Amico legge tutti i messaggi firmati. I candidati più logici per fungere da server del Grande Amico sono il governo, le banche o gli avvocati. Queste organizzazioni non ispirano una totale fiducia nei cittadini. Di qui, sarebbe meglio che per firmare i documenti non fosse necessaria alcuna autorità di fiducia.

Fortunatamente, la crittografia a chiave pubblica può dare in questo caso un contributo importante. Assumiamo che gli algoritmi a chiave pubblica di codifica e decodifica godano della proprietà che $E(D(P)) = P$ oltre alla proprietà solita che $D(E(P)) = P$. (L'RSA gode di questa proprietà, perciò quest'assunzione non è irragionevole). Assumendo che questo sia il caso, Alice può inviare un messaggio di testo in chiaro firmato, P , a Bob trasmettendo $E_B(D_A(P))$. Si noti bene che Alice conosce la propria (privata) chiave di decifrazione, D_A , così come la chiave pubblica di Bob, E_B , quindi costruire questo messaggio è qualcosa che Alice può fare.

Quando Bob riceve il messaggio, lo trasforma mediante la sua chiave privata, come sempre, ottenendo $D_A(P)$, come mostra la figura 7-23. Memorizza questo testo in un posto sicuro e quindi lo decifra usando E_A per ottenere il testo in chiaro originale.

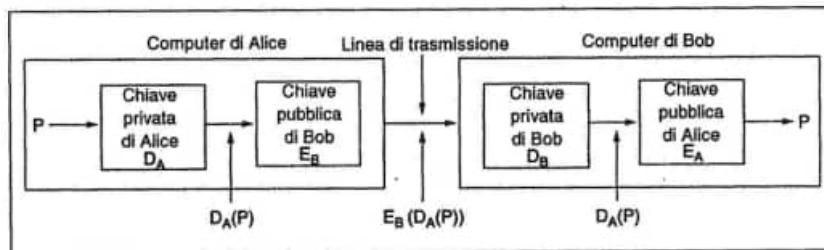


Fig. 7-23 Firme digitali mediante crittografia a chiave pubblica.

Per vedere come funziona la proprietà della firma, supponiamo che Alice in seguito neghi di aver inviato il messaggio P a Bob. Quando il caso arriva in tribunale, Bob può mostrare sia P che $D_A(P)$. Il giudice può facilmente verificare che Bob infatti ha un messaggio valido codificato con D_A , semplicemente applicando a esso E_A . Dato che Bob non conosce la chiave privata di Alice, la sola possibilità per Bob di aver acquisito un messaggio codificato con essa è che Alice lo abbia spedito davvero. Alice verrà condannata per frode. Sebbene la crittografia a chiave pubblica per le firme digitali sia uno schema elegante, ci sono dei problemi relativi all'ambiente in cui esse operano piuttosto che all'algoritmo di

7.1 La sicurezza in rete

base. Per prima cosa, Bob può provare che un messaggio era stato inviato da Alice solamente fino a che D_A rimane segreta. Se Alice rivela la sua chiave segreta, l'argomento non tiene più, perché chiunque potrebbe aver inviato il messaggio, compreso lo stesso Bob.

Il problema potrebbe sorgere, ad esempio, se Bob fosse l'agente di cambio di Alice. Alice chiede a Bob di comprare un certo titolo od obbligazione. Immediatamente dopo, il prezzo crolla rapidamente. Per ripudiare il suo messaggio a Bob, Alice corre dalla polizia dicendo che la sua casa è stata svaligiata e la sua chiave è stata rubata. A seconda della legge del suo stato, lei può non essere legalmente responsabile, specialmente se afferma di non aver scoperto il furto fino a che non è rientrata dal lavoro, parecchie ore più tardi.

Un altro problema con lo schema delle firme è relativo a ciò che accade se Alice decide di cambiare la sua chiave. Farlo è chiaramente legale, e il farlo regolarmente è probabilmente una buona cosa. Se più tardi interverrà una causa in tribunale, come abbiamo visto sopra, il giudice applicherà l'attuale E_A a $D_A(P)$ e scoprirà che essa non produce P . Bob a questo punto apparirà abbastanza ingenuo. Di conseguenza, sembra proprio necessario che ci sia una qualche autorità che registri tutti i cambiamenti di chiavi e la loro data.

In linea di principio qualsiasi algoritmo a chiave pubblica può venire usato per le firme digitali. Lo standard industriale di fatto è l'algoritmo RSA. Lo utilizzano molti prodotti di sicurezza. Tuttavia, nel 1991, l'NIST (National Institute of Standard and Technology) propose di utilizzare una variante dell'algoritmo El Gamal a chiave pubblica per il nuovo **Digital Signature Standard (DSS)**. El Gamal deve la sua sicurezza alla difficoltà di fattorizzare i grandi numeri.

Di solito quando il governo cerca di imporre uno standard crittografico avviene un tumulto. Il DSS fu criticato perché

1. Troppo segreto (l'NSA progettò il protocollo perché fosse utilizzato con El Gamal).
2. Troppo nuovo (El Gamal non è stato ancora pienamente analizzato).
3. Troppo lento (da 10 a 40 volte più lento dell'RSA nella verifica delle firme).
4. Troppo insicuro (chiave fissa a 512 bit).

In una revisione successiva, il quarto punto fu messo in dubbio quando furono permesse chiavi fino a 1024 bit. Non è ancora chiaro se il DSS verrà mantenuto. Per ulteriori dettagli, si vedano Kaufman *et al.* (1995); Schneier (1996); Stinson (1995).

Sintesi di messaggi

Una critica ai metodi di firma è che essi spesso accoppiano due funzioni distinte: autenticazione e segretezza. Di solito, l'autenticazione è necessaria, mentre non lo è la segretezza. Dato che la crittografia è lenta, di solito si vuole essere in grado di inviare dei documenti di testo in chiaro firmati. Qui sotto descriviamo uno schema di autenticazione che non richiede di cifrare l'intero messaggio (De Jonge, Chaum, 1987).

Questo schema si basa sull'idea di una funzione hash non invertibile che prende una parte arbitrariamente lunga di testo in chiaro e calcola da essa una stringa di bit di lunghezza

fissa. Questa funzione hash, spesso detta **sintesi di messaggi SM** (message digest), ha tre proprietà importanti:

1. Dato P , è facile calcolare $SM(P)$.
2. Dato $SM(P)$, è effettivamente impossibile trovare P .
3. Nessuno può generare due messaggi che abbiano la stessa sintesi di messaggi.

Per rispettare il criterio 3, la hash dovrebbe essere almeno di 128 bit, preferibilmente maggiore.

Calcolare una sintesi di messaggi da una parte di testo in chiaro è molto più veloce che codificare quel testo in chiaro con un algoritmo a chiave pubblica, così la sintesi di messaggi può venire usata per velocizzare gli algoritmi di firma digitale. Per vedere come questo funziona, consideriamo nuovamente il protocollo di firma di figura 7-22. GA adesso, invece di firmare P con $K_{GA}(A, t, P)$, calcola la sintesi di messaggi applicando SM a P , ottenendo $SM(P)$. GA quindi inserisce $K_{GA}(A, t, SM(P))$ come il quinto elemento della lista codificata con K_B che viene inviata a Bob, invece di $K_{GA}(A, t, P)$.

Se sorge una disputa, Bob può fornire sia P che $K_{GA}(A, t, SM(P))$. Dopo che il Grande Amico lo ha decifrato per il giudice, Bob ha $SM(P)$, che è sicuramente genuino, e l'allegato P . In ogni caso, dato che è veramente impossibile per Bob trovare qualsiasi altro messaggio che dia questa hash, il giudice si convincerà facilmente che Bob sta dicendo la verità. Utilizzare la sintesi di messaggi in questo modo significa risparmiare sia il tempo di codifica che i costi di trasporto e memorizzazione del messaggio.

Le sintesi di messaggi funzionano anche in crittosistemi a chiave pubblica, come si vede in figura 7-24. Qui, Alice dapprima calcola la sintesi di messaggi del suo testo in chiaro. Quindi firma la sintesi di messaggi e invia a Bob sia la sintesi firmata che il testo in chiaro. Se Trudy rimpiazzasse di nascosto P , Bob lo vedrebbe al momento di calcolare lo stesso $SM(P)$.

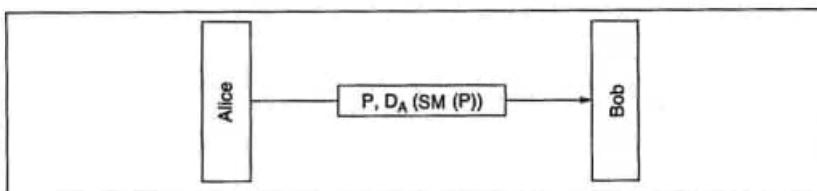


Fig. 7-24 Firme digitali mediante la raccolta di messaggi.

Sono state proposte diverse funzioni di sintesi di messaggi. Quelle maggiormente usate sono MD5 (Rivest, 1992) e SHA (NIST, 1993). MD5 è la quinta di una serie di funzioni hash progettate da Ron Rivest. Opera macinando bit in una maniera sufficientemente complicata in modo che ogni bit in uscita dipenda da ogni bit in ingresso. Molto brevemente, inizia portando la lunghezza del messaggio a 448 bit (modulo 512) mediante bit

aggiuntivi. Quindi la lunghezza originale del messaggio viene inserita alla fine come un intero di 64 bit in modo che da avere un ingresso totale la cui lunghezza sia un multiplo di 512 bit. L'ultimo passo di precalcolo è quello di inizializzare un buffer di 128 bit a un valore fissato.

A questo punto inizia la computazione. Ogni passo prende un blocco in ingresso di 512 bit e lo mescola completamente con il buffer di 128 bit. Per precauzione, è aggiunta anche una tabella costruita dalla funzione seno. Lo scopo di usare una funzione nota come il seno non è perché essa è più casuale di un generatore di numeri casuali, ma per evitare qualsiasi sospetto che il progettista abbia costruito una trappola intelligente attraverso il quale solo lui può entrare. Il rifiuto della IBM di rivelare i principi dietro al progetto dei blocchi S nel DES ha portato a una grossa speculazione sulle trappole. Per ogni blocco in ingresso vengono eseguiti quattro passi. Questo processo continua fino a che tutti i blocchi in ingresso sono stati consumati. I contenuti del buffer di 128 bit formano la raccolta di messaggi. L'algoritmo è stato ottimizzato per una implementazione software su una macchina a 32 bit. Di conseguenza, non può essere sufficientemente veloce per le future reti ad alta velocità (Touch, 1995).

L'altra importante funzione di sintesi di messaggi è la **SHA** (Secure Hash Algorithm), sviluppata dalla NSA e benedetta dall'NIST. Anch'essa elabora i dati in ingresso in blocchi di 512 bit, solo che a differenza di MD5, genera una raccolta di messaggi di 160 bit. Inizia completando il messaggio con bit aggiuntivi, quindi aggiunge una lunghezza di 64 bit per arrivare a un multiplo di 512 bit. Dopo inizializza il suo buffer di uscita di 160 bit.

Per ogni blocco in ingresso, viene aggiornato il buffer in uscita usando il blocco in ingresso di 512 bit. Non viene usata alcuna tavola di numeri casuali (o valori della funzione seno), ma per ogni blocco vengono eseguiti 80 passi, e il risultato è un completo rimescolamento. Ogni gruppo di 20 passi utilizza delle funzioni di mescolamento diverse. Dato che il codice hash di SHA è di 32 bit più lungo di quello di MD5, mentre tutto il resto è uguale, è di un fattore 2^{32} più sicuro di MD5. Comunque, è anche più lento di MD5, e un codice hash che non è una potenza di 2 può talvolta essere un inconveniente. In ogni modo, le due funzioni sono nella sostanza tecnicamente simili. Da un punto di vista politico, MD5 è definita in un RFC e viene usata moltissimo in Internet. SHA è uno standard del governo, ed è usata da società che la devono usare perché il governo dice loro di farlo, oppure da quelle che vogliono sicurezza aggiuntiva. Una versione riveduta, SHA-1, è stata approvata come standard dall'NIST.

L'attacco del compleanno

Nel mondo della crittografia, niente è mai come appare. Si potrebbe pensare che siano necessarie circa 2^m operazioni per sovvertire una raccolta di messaggi di m bit. In realtà, usando l'**attacco del compleanno**, un approccio pubblicato da Yuval (1979) nel suo articolo diventato un classico "How to Swindle Rabin", serviranno $2^{m/2}$ operazioni.

L'idea di questo attacco deriva da una tecnica che i professori di matematica usano spesso nei loro corsi di probabilità facendo questa domanda: "Quanti studenti sono necessari in una classe prima che la probabilità di avere due persone con lo stesso compleanno superi 1/2?". Molti studenti si aspettano che la risposta sia oltre 100. In effetti, la teoria delle

probabilità dice che è appena 23. Senza entrare in un'analisi rigorosa, intuitivamente, con 23 persone possiamo formare $(23 \times 22)/2 = 253$ coppie differenti, ognuna delle quali ha probabilità 1/365 di avere successo. Sotto questa luce, non sorprende più alcunché. Più genericamente, se c'è qualche corrispondenza tra gli ingressi e le uscite con n ingressi (persone, messaggi ecc.) e k possibili uscite (compleanni, selezione di messaggi ecc.), ci sono $n(n-1)/2$ coppie in ingresso. Se $n(n-1)/2 > k$, la possibilità di avere almeno una corrispondenza è abbastanza buona. Così, approssimativamente, una corrispondenza è circa $n > \sqrt{k}$. Questo risultato significa che una raccolta di messaggi di 64 bit può probabilmente essere violata generando circa 2^{32} messaggi e cercandone due con la stessa raccolta di messaggi.

Vediamo un esempio pratico. Il dipartimento di Informatica dell'Università Statale ha un posto libero per un professore e due candidati, Tom e Dick. Tom è stato assunto due anni prima di Dick, così egli viene esaminato prima. Se la prova risultasse soddisfacente, le probabilità per Dick di ottenere il posto sarebbero esigue. Tom sa che il capo del dipartimento, Marilyn, ha una grande considerazione del suo lavoro, così le chiede di scrivere una lettera di raccomandazione al Rettore, che deciderà sul caso di Tom. Una volta inviate, tutte le lettere diventano confidenziali.

Marilyn dice alla sua segretaria, Ellen, di scrivere al Rettore una lettera, dandole l'idea di ciò che vuole venga scritto. Quando sarà pronta, Marilyn rivedrà la lettera, calcolerà e firmerà la raccolta di messaggi di 64 bit, e invierà questa al Rettore. Ellen può in seguito inviare la lettera per posta elettronica.

Sfortunatamente per Tom, Ellen è coinvolta sentimentalmente con Dick, e così scrive la lettera seguente con le 32 opzioni tra parentesi.

Caro Rettore Smith,

Questa [lettera | messaggio] contiene la mia [onesta | franca] opinione sul professor Tom Wilson, che è [un candidato | in lizza] per il posto di professore [di adesso | di quest'anno]. Ho [conosciuto | lavorato con] il professor Wilson per [circa | quasi] sei anni. Egli è un [superbo | eccellente] ricercatore di grande [talento | abilità] conosciuto [in tutto il mondo | a livello internazionale] per le sue [brillanti | creative] capacità in [molti | una grande varietà di] [impegnativi | stimolanti] problemi.

È anche un [insegnante | educatore] [altamente | grandemente] [rispettato | ammirato]. I suoi allievi danno alle sue [classi | lezioni] un punteggio [con lodi smisurate | spettacolare]. Egli è il [più popolare | più amato] [insegnante | istruttore] [tra i nostri | del dipartimento].

[In aggiunta | Inoltre] il professor Wilson è un procacciatore di fondi [di talento | abile]. I suoi [studi | contratti] hanno portato una [grande | sostanziale] somma di denaro [al | al nostro] dipartimento. [Questo denaro ha | Questi fondi hanno] [messo in grado | permesso] di [continuare | realizzare] molti programmi [speciali | importanti], [come | ad esempio] il Suo programma Stato 2000. Senza questi fondi non saremmo stati [capaci | in grado] di continuare questo programma, che è così [importante | essenziale] per entrambi. Raccomando fortemente che egli sia nominato professore.

Sfortunatamente per Tom, appena Ellen termina di stendere e battere questa lettera, ne scrive una seconda:

7.1 La sicurezza in rete

Caro Rettore Smith,

Questa [lettera | messaggio] contiene la mia [onesta | franca] opinione sul professor Tom Wilson, che è [un candidato | in lizza] per il posto di professore [di adesso | di quest'anno]. Ho [conosciuto | lavorato con] il professor Wilson per [circa | quasi] sei anni. Egli è un [semplice | debole] ricercatore non molto conosciuto nella sua [disciplina | area]. La sua ricerca [difficilmente | raramente] dimostra [capacità | comprensione] dei problemi [chiave | maggiori] [dei | dei nostri] giorni.

Inoltre, non è un [insegnante | educatore] [rispettato | ammirato]. I suoi allievi danno alle sue [classi | lezioni] un punteggio [misero | pessimo]. È il meno popolare [insegnante | istruttore] [tra i nostri | del dipartimento], noto [soprattutto | principalmente] all'interno [del | del nostro] dipartimento per la sua [tendenza | propensione] a [ridicolizzare | mettere in imbarazzo] gli studenti abbastanza [sciocchi | imprudenti] da fare domande nelle sue classi.

[In aggiunta | Inoltre] il professor Wilson è un procacciatore di fondi [influente | marginale]. I suoi [studi | contratti] hanno portato solamente una [scarsa | insignificante] somma di denaro [al | al nostro] dipartimento. A meno che [del nuovo denaro sia trovato | dei nuovi fondi siano trovati] velocemente, dovremo cancellare alcuni programmi essenziali, come il Suo programma Stato 2000. Sfortunatamente, sotto queste [condizioni | circostanze] non posso in buona [coscienza | fede] raccomandarlo a Lei per [il posto di professore | una posizione permanente].

A questo punto Ellen imposta il proprio computer per calcolare durante la notte 2^{32} sintesi di messaggi per entrambe le lettere. Ci sono probabilità che una sintesi della prima lettera corrisponda a una della seconda lettera. Se no, può aggiungere alcune altre opzioni e provare di nuovo durante il fine settimana. Supponiamo che trovi una corrispondenza. Chiamiamo la lettera "buona" A e la lettera "cattiva" B.

Ellen adesso spedisce per posta elettronica la lettera A a Marilyn per la sua approvazione. Marilyn, ovviamente, approva, calcola la sua raccolta di messaggi di 64 bit, la firma e la spedisce per posta elettronica al Rettore Smith. In maniera indipendente, Ellen invia per posta elettronica la lettera B al Rettore.

Dopo aver ricevuto la lettera e la raccolta firmata, il Rettore esegue l'algoritmo della selezione del messaggio sulla lettera B, verificando che concorda con quello che Marilyn ha inviato, quindi licenzia Tom.

Con MD5 l'attacco del compleanno non è fattibile perché anche con 1.000.000.000 di selezioni al secondo, ci vorrebbero più di 500 anni per calcolare tutte le 2^{64} raccolte di due lettere con 64 varianti ciascuna, e anche dopo questo non è garantita una corrispondenza.

7.1.7 Aspetti sociali

Le implicazioni della sicurezza di rete per la segretezza individuale e della società in genere sono difficili da analizzare. In seguito menzioneremo solamente alcuni degli aspetti salienti.

Ai governi non piacciono i cittadini che hanno dei segreti nei loro confronti. In alcuni paesi (ad esempio, Francia) tutta la crittografia non governativa è semplicemente proibita

a meno che non sia il governo a fornire tutte le chiavi utilizzate. Come puntualizzano Kahn (1980), Selfridge, Schwartz (1980), le spie governative hanno fatto pratica in una scala più ampia di quello che la maggior parte della gente può pensare, e si pretende dai loro sforzi qualcosa di più di un mucchio di bit indecifrabili.

Il governo degli Stati Uniti ha proposto uno schema di cifratura per la futura telefonia digitale che comprende una speciale caratteristica che permette alla polizia di intercettare e decifrare tutte le chiamate telefoniche fatte negli Stati Uniti. Il governo ha promesso di non utilizzare questa possibilità senza un ordine del tribunale, ma molte persone ricordano ancora come l'ex direttore della FBI J. Edgar Hoover intercettò illegalmente i telefoni di Martin Luther King, Jr. e di altre persone nel tentativo di neutralizzarli. Il dibattito su questo argomento è molto acceso. In Kaufman *et al.* (1995) troviamo una discussione sulla tecnologia coinvolta (Clipper). In Blaze (1994); Schneier (1996) è descritto un modo per raggiungere questa tecnologia e inviare messaggi che il governo non può leggere. In Hoffman (1995) vengono analizzate le posizioni di entrambe le parti.

Gli Stati Uniti hanno una legge (22 U.S.C. 2778) che proibisce ai cittadini di esportare armi (materiale bellico), come carri armati e cacciabombardieri, senza l'autorizzazione del Ministero della Difesa (DoD). Secondo gli scopi di questa legge, il software crittografico è classificato come un'arma. Phil Zimmermann, che ha scritto il PGP (Pretty Good Privacy), un programma di protezione della posta elettronica, è stato accusato di violare questa legge, anche se il governo ammette che non lo abbia esportato (ma essendo arrivato su Internet può essere preso da stranieri). Molti persone considerano questo incidente largamente pubblicizzato come una grossa violazione ai diritti di un cittadino americano per garantire la segretezza alle persone.

Non essere americano non aiuta. Il 9 luglio 1986, tre ricercatori israeliani dell'Istituto Weizmann in Israele presentarono la richiesta per un brevetto statunitense per un nuovo schema di firma digitale che avevano inventato. Impiegarono i successivi 6 mesi a discutere le loro ricerche in conferenze in tutto il mondo. Il 6 gennaio 1987, l'ufficio brevetti degli Stati Uniti rispose loro di informare tutti gli americani a conoscenza dei loro risultati, che rivelare qualcosa sulla ricerca poteva significare due anni di prigione, una multa di 10.000 dollari o entrambe le cose. L'ufficio brevetti inoltre volle una lista di tutte le nazioni estere che sapevano della ricerca. Per capire come questa storia è andata a finire, si veda Landau (1988).

I brevetti sono un altro argomento caldo. Quasi tutti gli algoritmi a chiave pubblica sono brevettati. La protezione dei brevetti dura 17 anni. Il brevetto RSA, ad esempio, scade il 20 settembre 2000.

La sicurezza della rete è politicizzata come pochi altri campi tecnici, e giustamente, poiché mette in relazione la differenza tra una democrazia e uno stato di polizia nell'era digitale. I numeri di marzo 1993 e novembre 1994 di *Communications of the ACM* avevano dei lunghi capitoli riguardanti rispettivamente la sicurezza telefonica e di rete, con argomenti vigorosamente spiegati e difesi da diversi punti di vista. Il capitolo 25 del libro sulla sicurezza di Schneier tratta le politiche della crittografia (Schneier, 1996). Lo stesso per il capitolo 8 del suo libro sulla posta elettronica (Schneier, 1995). Privacy e computer sono discussi anche in Adam (1995). Questi riferimenti sono

fortemente raccomandati ai lettori che desiderino proseguire i loro studi su questo argomento.

7.2 DNS – Domain Name System

È raro che i programmi si riferiscono a host, caselle di posta elettronica e altre risorse di rete mediante il loro indirizzo assoluto in binario. Invece di usare numeri binari si usano stringhe ASCII, come ad esempio *tana@art.ucsbd.edu*. D'altra parte, la rete è capace di usare solamente gli indirizzi in binario e quindi è necessario un meccanismo di conversione delle stringhe ASCII in indirizzi di rete. Nei paragrafi che seguono studieremo come si ottiene tale associazione nel caso di Internet.

Ai tempi di ARPANET esisteva un unico file, *hosts.txt*, che elencava tutti gli host e i loro indirizzi IP. Ogni notte tutti gli host della rete di allora lo copiavano scaricandolo dal sito in cui era mantenuto. Quando la rete comprendeva solo qualche centinaio di grosse macchine a suddivisione di tempo questo approccio funzionava ragionevolmente bene.

Tuttavia, quando la rete crebbe fino a comprendere migliaia di macchine, tutti compresero che tale soluzione non poteva continuare a funzionare per sempre. Per prima cosa il file sarebbe diventato troppo grosso. Più importante ancora, ci sarebbero stati sempre più conflitti di nome degli host, a meno che tali nomi non fossero gestiti in maniera centralizzata, il che è impensabile per una grande rete internazionale. Per risolvere questi problemi venne inventato il **DNS (Domain Name System)**.

Alla base del DNS si trova l'idea di uno schema di denominazione gerarchico e ripartito in domini, implementato mediante una base di dati distribuita. Il sistema viene usato principalmente per associare nomi di host e destinazioni di posta elettronica a indirizzi IP, ma può essere usato anche a scopo alternativo. Il sistema DNS è definito nei documenti RFC 1034 e 1035.

Vediamo in breve come si usa il DNS. Quando un programma deve trasformare un nome in un indirizzo IP chiama una procedura di libreria detta risolutrice (resolver), passandole il nome come parametro di ingresso. Il resolver manda un pacchetto UDP a un server DNS locale, che cerca il nome e restituisce l'indirizzo UDP al resolver, che a sua volta lo trasmette al programma chiamante. Usando tale indirizzo IP il programma può aprire una connessione TCP con la destinazione, oppure mandarle pacchetti UDP.

7.2.1 Lo spazio dei nomi del DNS

La gestione di un insieme di nomi vasto e costantemente modificabile è un problema non banale. Nel normale sistema postale la gestione dei nomi viene fatta richiedendo che le lettere specifichino in modo esplicito o implicito la nazione, la provincia, la città e la via del destinatario. Grazie a questo indirizzamento gerarchico non si fa confusione tra Mario Rossi abitante in viale Risorgimento a Pinerolo (Mi), e Mario Rossi abitante in viale Risorgimento a Nola (Na). Il DNS funziona allo stesso modo.

Concettualmente, Internet è suddivisa in centinaia di **domini radice**; ciascun dominio include moltissimi host. Ogni dominio si divide in sottodomini, che a loro volta possono essere frazionati. Tutto questo si rappresenta graficamente con un albero come in figura 7.25.

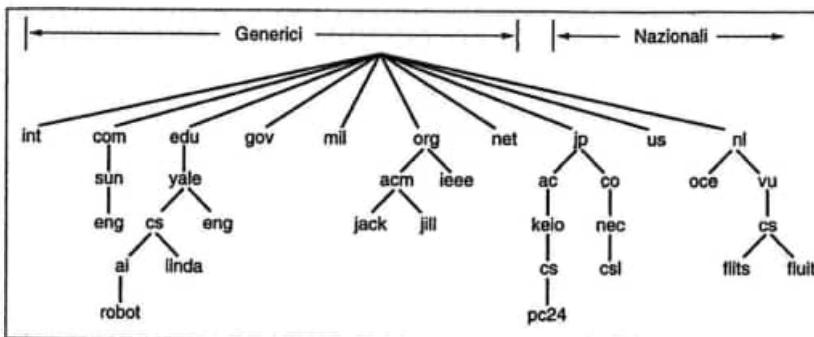


Fig. 7.25. Una parte dello spazio dei nomi dei domini di Internet.

Le foglie dell'albero rappresentano domini che non hanno sottodomini (ma naturalmente contengono macchine). Un dominio foglia può contenere un singolo host oppure migliaia, e può rappresentare un'intera società commerciale.

I domini radice possono essere generici o nazionali. I domini generici sono *com* (commerciale), *edu* (istituzioni educative), *gov* (il governo federale USA), *int* (alcune organizzazioni internazionali), *mil* (le forze armate USA), *net* (fornitori di servizi di rete), e *org* (organizzazioni senza scopo di lucro). I domini nazionali sono uno per nazione, e sono definiti nel documento ISO 3166.

Il nome di un dominio è composto dal cammino inverso dalla foglia fino alla radice (anonima). I componenti del cammino sono separati da punti. Per esempio, il dominio del dipartimento ingegneristico della Sun Microsystems potrebbe essere denotato dal cammino *eng.sun.com*, invece che da un cammino stile UNIX come */com/sun/eng*. Si noti che questo sistema di denominazione gerarchico implica che l'uso di *eng* in *eng.sun.com* non contrasta con un analogo possibile uso in *eng.yale.edu*, che potrebbe denotare il dipartimento di letteratura inglese dell'Università di Yale.

I nomi dei domini possono essere assoluti o relativi. Un nome assoluto finisce con un punto (*eng.sun.com.*), mentre un nome relativo finisce senza punto. I nomi relativi debbono essere interpretati in qualche contesto per determinare univocamente il loro significato. In entrambi i casi tuttavia il nome di un dominio denota sia un nodo specifico che tutti i suoi sottodonodi nell'albero globale.

I nomi dei domini sono insensibili alle maiuscole/minuscole, quindi *edu* e *EDU* denotano lo stesso dominio. I nomi dei componenti possono essere lunghi non più di 63 caratteri, mentre un cammino non può superare complessivamente i 255 caratteri.

In linea di principio i domini possono trovare posto nell'albero in più modi. Ad esempio, *cs.yale.edu* potrebbe essere inserito in alternativa nel dominio nazionale *us*, come *cs.yale.ct.us*. Tuttavia, quasi tutte le organizzazioni statunitensi in pratica usano un dominio generico, mentre tutte quelle fuori degli Stati Uniti usano il proprio dominio nazionale. Nessuna regola proibisce di registrarsi sotto due domini radice, ma, poiché la cosa potrebbe generare confusione, poche organizzazioni si avvalgono di questa possibilità.

Ciascun dominio controlla l'allocazione dei suoi sottodomini. Ad esempio, il Giappone ha i domini *ac.jp* e *co.jp* che ricalcano *edu* e *com*. L'Olanda non fa distinzione e mette tutte le organizzazioni direttamente sotto *nl*. Quelli che seguono sono tre dipartimenti di Informatica:

1. *cs.yale.edu* (Yale University, negli USA)
2. *cs.vu.nl* (Vrije Universiteit, in Olanda)
3. *cs.keio.ac.jp* (Keio University, in Giappone).

Per creare un nuovo sottodominio occorre il permesso del dominio in cui verrà incluso. Ad esempio, se a Yale nascesse un gruppo di ricerca su VLSI e volesse essere denominato *vlsi.cs.yale.edu* occorrerebbe il permesso del gestore di *cs.yale.edu*. Analogamente, se venisse fondata una nuova università, diciamo la University of Northern South Dakota, dovrebbe chiedere al gestore del dominio *edu* di assegnarle il nome *unsd.edu*. In questo modo si evitano i conflitti di nome e ciascun dominio può tenere traccia in modo indipendente di tutti i suoi sottodomini. Infatti, non appena un nuovo dominio è stato creato e registrato, esso può creare propri sottodomini, come ad esempio *cs.unsd.edu*, senza dover chiedere alcun ulteriore permesso ai livelli superiori dell'albero.

Il sistema di denominazione si conforma alle necessità dell'organizzazione, non ai limiti delle reti fisiche. Ad esempio, se i dipartimenti di Informatica e di Ingegneria elettronica risiedono nello stesso edificio e usano la stessa rete locale, possono avere comunque domini separati. Allo stesso modo, anche se il dipartimento di Informatica è diviso su due edifici tutti gli host di entrambi normalmente apparterranno allo stesso dominio.

7.2.2. Descrittori di risorsa

Ciascun dominio, che include un solo host, oppure che sia radice, può essere associato a un insieme di **descrittori di risorse** (resource records). Il descrittore più comune di un singolo host è il suo indirizzo IP, ma ne esistono anche altri tipi. Quando un resolver trasmette un nome di dominio a un DNS, ciò che ottiene in risposta sono i descrittori di risorsa associati a quel nome. Quindi la vera funzione del DNS è di mappare i nomi di domini sui descrittori di risorsa.

Un descrittore di risorsa è una quintupla. Sebbene siano efficientemente codificati in binario, quando vengono descritti vengono presentati in forma di testo ASCII, un record per linea. Noi useremo il seguente formato:

Nome_dominio	Tempo_di_vita	Tipo	Classe	Valore
--------------	---------------	------	--------	--------

Il *Nome_dominio* denota il dominio a cui questo descrittore appartiene. Normalmente esistono più descrittori per dominio e ogni copia del database conserva dati su più domini. Questo campo quindi costituisce la chiave primaria di ricerca usata per rispondere alle richieste. L'ordine dei descrittori nel database non è significativo. Quando occorre rispondere a qualche richiesta su un dominio, vengono restituiti tutti i descrittori della classe richiesta. Il campo *Tempo_di_vita* indica quanto sia stabile il descrittore. Se il descrittore è forte-

mente stabile questo campo ha un valore numerico alto, come ad esempio 86400 (il numero di secondi in un giorno). Se invece il descrittore è volatile, tale valore è molto basso: ad esempio 60 (un minuto). Torneremo su questo punto quando discuteremo il caching.

Il campo *Tipo* dice di che specie di descrittore si tratti. I tipi più importanti sono elencati in figura 7.26.

Tipo	Significato	Valore
SOA	inizio di autorità	parametri per questa zona
A	indirizzo IP di un host	intero a 32 bit
MX	scambio posta	priorità, dominio che accetta posta
NS	name server	nome di un server per questo dominio
CNAME	nome canonico	nome dominio
PTR	puntatore	alias per indirizzo IP
HINFO	descrizione di host	CPU e sistema operativo in ASCII
TXT	testo	testo ASCII non interpretato

Fig. 7.26 I tipi principali di descrittori di risorsa DNS.

Un descrittore di tipo *SOA* fornisce il nome della fonte principale di informazioni sulla zona del name server (vedi sotto), l'indirizzo di posta elettronica dell'amministratore, un numero di serie univoco, e alcuni flag e temporizzazioni.

Il tipo più importante è *A* (indirizzo). Contiene l'indirizzo IP su 32 bit di un host. Ogni host in Internet deve possedere almeno un indirizzo IP, in modo che gli altri membri della rete possano comunicare con esso. Alcune macchine possiedono due o più connessioni di rete, e in questo caso devono possedere un descrittore di tipo *A* per ciascuna connessione (e quindi per ciascun indirizzo IP).

Viene poi per ordine di importanza il tipo *MX*, che specifica il nome del dominio preparato ad accettare la posta elettronica del dominio specificato. Questo descrittore tipicamente si usa per permettere a una macchina al di fuori di Internet di ricevere posta da siti Internet. La trasmissione si ottiene facendo in modo che il sito non-Internet si accordi con qualche sito Internet in modo che quest'ultimo accetti posta in sua vece e la ritrasmetta usando qualsiasi protocollo venga concordato.

Per esempio, supponiamo che Catia sia una studentessa di Informatica a Bologna. Dopo la laurea fonda una società, la Electrobrain Corporation, per commercializzare le sue idee. Non si può permettere una connessione Internet, così si accorda con l'Università di Bologna in modo che la sua posta elettronica le venga ritrasmessa. Lei userà un telefono per chiamare una volta al giorno e leggere la posta.

Catia registra la sua società nel dominio com: ottiene l'indirizzo *electrobrain.com*. Po-

trebbe quindi chiedere all'amministratore del dominio *com* di aggiungere il seguente descrittore *MX* al database *com*:

electrobrain.com 86400 IN MX 1 mailserver.cs.unibo.it

In questo modo la sua posta viene trasmessa al Dipartimento di Informatica di Bologna da dove lei la legge. Oppure, Bologna potrebbe chiamarla e trasferire la posta mediante qualche protocollo concordato.

Il descrittore *NS* specifica un name server. Ad esempio, ciascun database DNS normalmente ha un record *NS* per ciascun dominio radice, in modo che la posta elettronica possa essere trasmessa a punti distanti dell'albero dei nomi. Torneremo su questo punto più avanti.

I descrittori di tipo *CNAME* permettono di creare alias. Ad esempio, una persona che crede di conoscere le regole di denominazione di Internet potrebbe tentare di scrivere al MIT qualcuno il cui nome login sia *paul*, usando un indirizzo come *paul@cs.mit.edu*. Il dipartimento di Informatica del MIT si chiama in realtà *lcs.mit.edu*. Allo scopo di fornire un servizio a tutti coloro che usano l'altro indirizzo, i sistemisti del MIT potrebbero creare un record *CNAME* per aiutare l'instradamento della posta:

cs.mit.edu 86400 IN CNAME lcs.mit.edu

Anche i descrittori *PTR*, come i *CNAME*, puntano a un altro nome. Tuttavia, mentre *CNAME* è una definizione di macro, *PTR* è un normale tipo DNS la cui interpretazione dipende dal contesto in cui si incontra. In pratica viene quasi sempre usato per associare un nome a un indirizzo IP per permettere ricerche dell'indirizzo IP e restituire il nome della macchina corrispondente.

I descrittori *HINFO* dicono a quale tipo di macchina e sistema operativo corrisponda un dominio. Infine, i descrittori *TXT* permettono a un dominio di identificarsi in modo arbitrario. Entrambi questi ultimi due tipi sono a disposizione degli utenti. Nessuno dei due è necessario, quindi i programmi non possono assumerne l'esistenza (e forse non potrebbero gestirli se li ottenessero).

Torniamo alla struttura generale dei descrittori: il quarto campo di un descrittore denota sempre la *Classe*. Il codice che denota la classe Internet è *IN*; altri codici denotano informazioni non-Internet.

Infine, spieghiamo il campo *Valore*. Questo campo può contenere un numero, un nome di dominio, o una stringa ASCII. La semantica dipende dal tipo di descrittore. La figura 7-26 descrive brevemente i campi *Valore* dei principali tipi di descrittore.

La figura 7-27 contiene un esempio dell'informazione che potremmo trovare in un database DNS. Questa figura descrive parte di un ipotetico database per il dominio *cs.vu.nl*, mostrato in figura 7-25. Il database contiene sette tipi di descrittori di risorsa.

La prima linea non commentata della figura 7-27 offre alcune informazioni di base sul dominio, di cui non ci preoccupiamo. Le successive due linee includono informazioni testuali sulla locazione del dominio. I due elementi successivi indicano il primo e secondo posto in cui provare a spedire il messaggio di posta inviato da *person@cs.vu.nl*. La *zephyr*

; Authoritative data for cs.vu.nl				
cs.vu.nl.	86400	IN	SOA	star boss (952771,7200,7200,2419200,86400)
cs.vu.nl.	86400	IN	TXT	"Faculteit 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

Fig. 7-27 Una parte di un ipotetico database DNS per il dominio cs.vu.nl.

(una macchina specifica) è la prima da provare. Se la ricerca fallisce, subito dopo va provata *top*.

Dopo la linea bianca, aggiunta per migliore leggibilità, troviamo le linee che indicano che *flits* è una workstation Sun con UNIX e che danno entrambi i suoi indirizzi. Quindi si hanno tre scelte per gestire la posta elettronica inviata a *flits.cs.vu.nl*. La prima scelta è naturalmente *flits* stesso, ma se questo è già, *zephyr* e *top* sono la seconda e terza scelta. Quindi troviamo un alias, *www.cs.vu.nl*, in modo che questo indirizzo possa essere usato senza indicare una macchina specifica. La creazione di questo alias permette a *cs.vu.nl* di cambiare il suo server World Wide Web senza modifiche all'indirizzo che la gente usa. La stessa cosa vale per *ftp.cs.vu.nl*.

Le quattro linee successive contengono un elemento tipico per una workstation, in questo caso *rowboat.cs.vu.nl*. L'informazione fornita contiene l'indirizzo IP, la destinazione primaria e secondaria per la posta, e informazioni sulla macchina. Quindi segue un elemento

per un sistema non UNIX che non è in grado di ricevere da solo la posta, e un elemento per una stampante laser.

Ciò che non compare (e non si trova in questo file) sono gli indirizzi IP da usare per cercare nei domini a livello superiore. Questi servono per cercare host distanti, ma dato che non fanno parte del dominio *cs.vu.nl*, non si trovano in questo file. Vengono forniti dai server radice, i cui indirizzi IP si trovano in un file di configurazione e vengono caricati nella cache del DNS quando viene avviato il server DNS. Hanno timeout molto lunghi, in modo che una volta caricati non vengono mai eliminati dalla cache.

7.2.3 Name server

Almeno in teoria un solo name server può contenere l'intero database DNS e rispondere a tutte le interrogazioni che lo riguardano. In pratica, questo server sarebbe così sovraccarico da essere inservibile. Inoltre, se mai si guastasse, l'intera Internet sarebbe bloccata. Per evitare i problemi associati a un'unica fonte di informazioni, lo spazio dei nomi DNS è suddiviso in **zone** non sovrapposte. Una possibilità di dividere lo spazio dei nomi di figura 7-25 è quella di figura 7-28. Ogni zona contiene alcune parti dell'albero e inoltre contiene dei name server che mantengono le informazioni di autorità relativamente a quella zona. Normalmente una zona avrà un name server principale, che prende le sue informazioni da un file sul proprio disco, e uno o più name server secondari, che prendono le loro informazioni dal name server principale. Per migliorare l'affidabilità, è possibile che alcuni server di zona si trovino al di fuori della zona stessa.

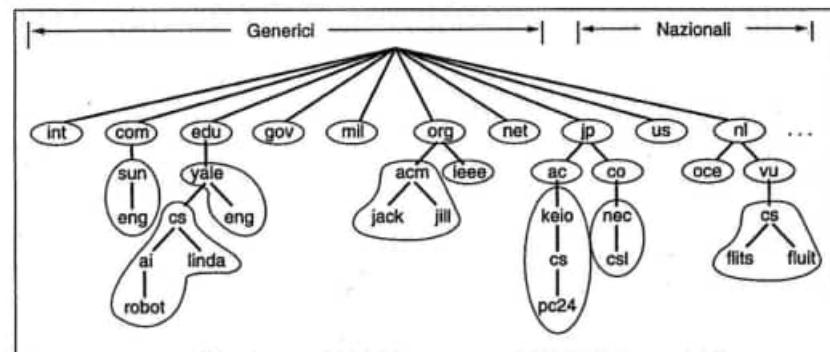


Fig. 7-28 Parte dello spazio dei nomi DNS con la suddivisione in zone.

Dove siano posti i confini di una zona è affare dell'amministratore della zona. Questa decisione è in gran parte basata su quanti name server si vogliono e dove vanno collocati. Ad esempio, nella figura 7-28, Yale ha un server per *yale.edu* che gestisce *eng.yale.edu*, che si trova in una zona separata da quella del proprio name server. Una decisione di questo tipo potrebbe essere presa se un dipartimento come quello di Inglese non volesse

mantenere il proprio name server, mentre lo vuole il dipartimento di Informatica. Di conseguenza, *cs.yale.edu* fa zona a sé, invece *eng.yale.edu* no.

Quando un resolver riceve un'interrogazione su un nome di dominio, la passa a uno dei name server locali. Se il dominio in esame si trova sotto la giurisdizione del name server, come *ai.cs.yale.edu* è sotto *cs.yale.edu*, esso restituisce i record di autorità della risorsa. Un **record di autorità** proviene dall'autorità che gestisce il record, ed è quindi sempre corretto. I record di autorità sono più importanti dei record in cache, che possono non essere aggiornati.

In ogni modo, se il dominio è remoto e localmente non è disponibile alcuna informazione relativa al dominio richiesto, il name server invia un'interrogazione per il dominio richiesto al name server al livello più alto. Per chiarire questo processo, consideriamo l'esempio di figura 7-29. Qui, un resolver su *flits.cs.vu.nl* richiede l'indirizzo IP dell'host *linda.cs.yale.edu*. Al passo 1, invia una interrogazione al name server locale, *cs.vu.nl*. Questa interrogazione contiene il nome del dominio analizzato, il tipo (*A*) e la classe (*IN*).

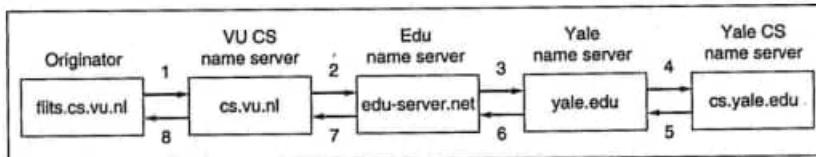


Fig. 7-29 Come un resolver ricerca un nome remoto in 8 passi.

Supponiamo che un name server locale non abbia mai ricevuto prima un'interrogazione per questo dominio e non ne sappia niente. Può chiedere ad alcuni name server vicini, ma se nessuno di questi sa niente, invia un pacchetto UDP al server di *.edu* che trova nel suo database (vedi la figura 7-29), *edu-server.net*. Probabilmente questo server non conosce l'indirizzo di *linda.cs.yale.edu* e neppure quello di *cs.yale.edu*, ma deve conoscere i propri figli, quindi invia la richiesta al name server per *yale.edu* (passo 3). A sua volta questo invia la richiesta a *cs.yale.edu* (passo 4), che deve avere i record di autorità della risorsa. Dato che ogni richiesta va da un cliente a un server, il record della risorsa richiesto torna indietro nei passi dal 5 all'8.

Una volta che questi record sono tornati al name server di *cs.vu.nl*, qui verranno inseriti nella cache, nel caso servano più tardi. Comunque, questa informazione non è di autorità, in quanto le modifiche fatte a *cs.yale.edu* non si propagheranno a tutte le cache del mondo che hanno loro notizie. Per questa ragione, gli elementi della cache non dovrebbero avere lunga durata. Infatti in ogni record della risorsa è inserito il campo *Time_to_live*. Indica ai name server remoti quanto tempo debbono conservare i record in cache. Se una certa macchina conserva lo stesso indirizzo IP da anni, può essere sicuro mantenere quell'informazione in cache per un giorno. Per informazioni più volatili, potrebbe essere consigliabile eliminare i record dopo alcuni secondi o minuti.

Ricordiamo che il metodo delle interrogazioni che abbiamo descritto è conosciuto come **interrogazione ricorsiva**, dato che ogni server che non ha le informazioni richieste le cerca da qualche parte, e quindi le invia a destinazione. Esiste anche una forma alternativa.

In questo caso, quando un'interrogazione non può essere soddisfatta localmente, l'interrogazione fallisce, ma viene restituito il nome del prossimo server da interrogare lungo il cammino. Questa procedura offre maggior controllo al cliente nel processo di ricerca. Alcuni server non implementano le interrogazioni ricorsive e restituiscono sempre il nome del prossimo server con cui provare.

È anche bene puntualizzare che quando un cliente DNS non riesce ad avere una risposta prima dello scadere del timer, di solito la volta successiva proverà con un altro server. In questo caso si assume che il server sia spento e non che sia andata persa la richiesta o la risposta.

7.3 SNMP – Simple Network Management Protocol

Nel primo periodo di ARPANET, se il ritardo di risposta da qualche host diveniva inspiegabilmente grande, chi scopriva il problema semplicemente eseguiva il programma Ping per inviare un pacchetto a destinazione. Osservando le marche temporali nell'intestazione del pacchetto restituito, il problema di solito veniva localizzato e si poteva intraprendere qualche azione appropriata. Inoltre, il numero dei router era così piccolo che era possibile fare ping su ognuno per vedere se era guasto.

Quando ARPANET si trasformò nella rete internazionale Internet, con dorsali multiple e molti operatori, questa soluzione non fu più adeguata, perciò vennero costruiti nuovi strumenti per la gestione della rete. Ci furono due tentativi definiti in RFC 1028 e RFC 1067, ma ebbero vita breve. Nel maggio 1990, fu pubblicato l'RFC 1157, che definiva la prima versione di **SNMP (Simple Network Management Protocol)**. Insieme a un manuale (RFC 1155) sulle informazioni di gestione, SNMP definiva una maniera sistematica di controllare e gestire una rete di computer. Questo metodo e questo protocollo vennero realizzati in moltissimi prodotti commerciali e divennero lo standard di fatto per la gestione di rete.

Col tempo vennero alla luce alcune imperfezioni nell'SNMP, per cui fu definita e applicata una nuova versione, SNMPv2 (negli RFC da 1441 a 1452), che divenne uno standard di Internet. Nei paragrafi seguenti daremo una breve analisi del modello e del protocollo SNMP (intendendo SNMPv2).

Sebbene SNMP sia stato progettato con l'obiettivo della semplicità, esiste un autore che ha scritto al riguardo un libro di 600 pagine (Stallings, 1993a). Per delle descrizioni più brevi, si vedano i libri di Rose (1994) e di Rose e McCloghrie (1995), entrambi tra i progettisti dell'SNMP. Altri riferimenti sono Feit (1995); Hein, Griffiths (1995).

7.3.1 Il Modello SNMP

Il modello SNMP di una rete consiste di quattro componenti:

1. Nodi gestiti.
2. Stazioni di gestione.
3. Informazioni di gestione.
4. Un protocollo di gestione.

Queste componenti sono illustrate in figura 7-30 e analizzate in seguito.

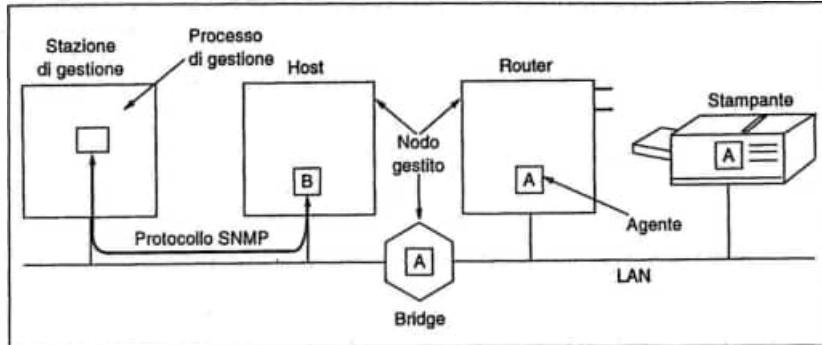


Fig. 7-30 I componenti del modello di gestione SNMP.

I nodi gestiti possono essere host, router, bridge, stampanti o qualsiasi altro dispositivo capace di comunicare informazioni di stato verso il mondo esterno. Un nodo può essere gestito direttamente dall'SNMP se è in grado di eseguire un processo di gestione SNMP, detto **agente SNMP**. Tutti i computer seguono questo requisito, così come molti bridge, router e periferiche progettati per la rete. Ogni agente mantiene un database di variabili locali che descrive il suo stato e la sua storia e che influenza le sue operazioni.

La gestione della rete è fatta dalle **stazioni di gestione**, che sono in effetti computer normali che eseguono un software speciale di gestione. Le stazioni di gestione contengono uno o più processi che comunicano con gli agenti sulla rete, inviando comandi e ricevendo risposte. In questo progetto, tutta l'intelligenza si trova nelle stazioni di gestione, in modo da mantenere gli agenti il più possibile semplici e minimizzare l'effetto sui dispositivi su cui girano. Molte stazioni di gestione hanno un'interfaccia utente/grafica per permettere al gestore della rete di controllare lo stato della rete e intervenire quando necessario.

La maggior parte delle reti nella realtà han componenti di provenienza diversa, con host di uno o più costruttori, bridge e router di altri produttori e stampanti di altri ancora. Per permettere a una stazione di gestione (potenzialmente ancora di un altro produttore) di colloquiare con tutti questi componenti diversi, bisogna che l'informazione mantenuta da essi sia rigidamente specificata. È inutile che una stazione di gestione chieda a un router qual è il suo tasso di perdita di pacchetti, se il router non ne serba memoria. Perciò, SNMP descrive (in esasperante dettaglio) l'informazione precisa che ogni tipo di agente deve mantenere e il formato che gli deve essere fornito. La parte maggiore del modello SNMP è la definizione di chi deve tener traccia di cosa e come questa informazione deve essere comunicata.

In breve, ogni dispositivo conserva una o più variabili che descrivono il suo stato. Nella letteratura SNMP queste variabili sono dette **oggetti**, ma il termine è impreciso in quanto non sono oggetti nel senso di un sistema orientato agli oggetti dato che hanno solamente lo stato e nessun metodo (eccetto che per leggerne e scriverne il valore). Ciò nonostante,

il termine è così sfruttato (cioè usato in diverse parole riservate nel linguaggio di specifica utilizzato) che lo useremo anche noi. La collezione di tutti i possibili oggetti in una rete si trova in una struttura dati detta **MIB (Management Information Base)**.

La stazione di gestione interagisce con gli agenti mediante il protocollo SNMP. Questo protocollo permette alla stazione di gestione di richiedere lo stato degli oggetti locali di un agente, e di modificarli se necessario. Essenzialmente SNMP consiste in queste comunicazioni tipo domanda-risposta.

In ogni modo, a volte capitano eventi imprevisti. I nodi gestiti possono guastarsi e poi ripartire, le linee possono andare giù e ritornare su, si può verificare una congestione, e così via. Ogni evento significativo è descritto in un modulo MIB. Quando un agente scopre che si è verificato un evento significativo, lo notifica immediatamente a tutte le stazioni di gestione nella propria lista di configurazione. Questa notifica viene detta eccezione SNMP (**trap**, per ragioni storiche). Di solito la notifica indica solo il verificarsi di qualche evento. Sta alla stazione di gestione richiedere dettagli sull'accaduto. Dato che la comunicazione tra i nodi gestiti e la stazione di gestione non è affidabile (nel senso che non è prevista conferma), è bene che la stazione di gestione ogni tanto contatti ciascun nodo gestito per verificare se è avvenuto qualche evento strano. Chiamiamo **polling orientato ai trap** il modello nel quale si eseguono verifiche a intervalli lunghi con accelerazioni al momento della rilevazione di un trap.

Questo modello assume che ogni nodo gestito possa eseguire al proprio interno un agente SNMP. Ci sono alcuni dispositivi vecchi o che in origine non erano stati pensati per un utilizzo in rete che non hanno questa possibilità. Per gestirli, l'SNMP definisce ciò che viene chiamato un **agente proxy**, di fatto un agente che controlla uno o più dispositivi non SNMP e che comunica con la stazione di gestione riguardo la loro affidabilità, e con i dispositivi stessi mediante dei protocolli non standard.

Per concludere, la sicurezza e l'autenticazione giocano un ruolo importante nell'SNMP. Una stazione di gestione è in grado di assumere una grande quantità di informazioni su ogni nodo che controlla e anche di fare uno shut down di tutti. È quindi estremamente importante che gli agenti siano sicuri che le richieste che presumibilmente arrivano dalla stazione di gestione siano effettivamente inviate dalla stazione di gestione. Nell'SNMPv1 è proprio la stazione di gestione che la sua identità inserendo una password (in chiaro) in ogni messaggio. Nell'SNMPv2 la sicurezza è stata migliorata in maniera considerevole mediante tecniche moderne di crittografia del tipo che abbiamo visto. In ogni modo, questa aggiunta ha reso un protocollo già pesante ancora più pesante, e in seguito è stato accantonato.

7.3.2. ASN.1 – Abstract Syntax Notation I

Il nucleo del modello SNMP è l'insieme di oggetti gestiti dagli agenti e che vengono letti e scritti dalla stazione di gestione. Per rendere possibile la comunicazione tra piattaforme di produttori diversi, è essenziale uno standard indipendente dal produttore per definire gli oggetti. Inoltre, sarà uno standard il modo per codificarli e trasferirli sulla rete. Le definizioni in C soddisfano il primo requisito, ma non definiscono una codifica a bit per uno scambio di informazioni senza ambiguità tra una piccola stazione di gestione a 32 bit in complemento a 2 e una grossa CPU a 16 bit in complemento a 1.

Per questa ragione, è necessario un linguaggio standard di definizione degli oggetti accompagnato da regole di codifica. Lo standard usato per SNMP è OSI ed è chiamato **ASN.1 (Abstract Syntax Notation 1)**. Come la maggior parte degli OSI, è grosso, complesso e non particolarmente efficiente. (Sembrerebbe che chiamandolo ASN.1 invece di semplicemente ASN, i progettisti ammettessero implicitamente che sarebbe stato ben presto rimpiazzato da un ASN.2.)

La principale presunta forza dell'ASN.1, cioè l'esistenza di regole non ambigue per la codifica binaria, è adesso in realtà un punto debole, in quanto per minimizzare il numero di bit sulla linea le regole di codifica sono ottimizzate, al costo di una perdita di tempo di CPU da entrambi i lati per codifica e decodifica. Sarebbe stato migliore uno schema più semplice, che utilizzasse interi di 32 bit allineati su 4 byte. In ogni modo, bene o male, l'SNMP è modellato su ASN.1 (o meglio su un suo sottoinsieme semplificato), perciò chiunque voglia veramente capire SNMP deve studiare ASN.1. Da qui la spiegazione che segue.

Iniziamo con il linguaggio per la descrizione dei dati, definito nello Standard Internazionale 8824. Dopo analizzeremo le regole di codifica, descritte nello Standard Internazionale 8825. La sintassi astratta di ASN.1 è essenzialmente un linguaggio dichiarativo di dati primitivi. Permette agli utenti di definire oggetti primitivi e quindi combinarli in oggetti più complessi. Una serie di dichiarazioni in ASN.1 è funzionalmente simile alle dichiarazioni trovate nei file di intestazione associati a molti programmi C.

SNMP definisce alcune convenzioni lessicali che seguiremo. Queste comunque non sono proprio uguali a quelle usate dall'ASN.1 puro. I tipi di dato predefiniti sono scritti in maiuscolo (ad es. *INTEGER*). I tipi definiti dall'utente iniziano con una lettera maiuscola ma devono contenere almeno un carattere diverso da una lettera maiuscola. Gli identificatori possono contenere lettere maiuscole e minuscole, cifre e trattini, ma devono iniziare con una lettera minuscola (ad es. *contatore*). Lo spazio bianco (tabulatori, ritorno carrello ecc.) non è significativo. Infine, i commenti iniziano con il segno - - e continuano fino al termine della linea o alla prossima occorrenza di segni - -.

La figura 7-31 mostra i tipi di dati base di ASN.1 permessi da SNMP. (In genere ignorremo le particolarità di ASN.1, come i tipi *BOOLEAN* e *REAL*, non permessi in SNMP.) L'utilizzo dei codici verrà descritto più tardi.

Tipi Primitivi	Significato	Codice
INTEGER	Intero di lunghezza arbitraria	2
BIT STRING	Una stringa di 0 o più bit	3
OCTET STRING	Una stringa di 0 o più byte senza segno	4
NULL	Una posizione vuota	5
OBJECT IDENTIFIER	Un tipo di dato definito ufficialmente	6

Fig. 7-31 I tipi di dato primitivi ASN. 1 permessi in SNMP.

Una variabile di tipo *INTEGER* può, in teoria, assumere un qualsiasi valore intero, ma esistono delle altre regole SNMP che limitano l'intervallo di validità. Per dare un esempio

dell'uso dei tipi, vediamo com'è possibile in ASN.1 dichiarare una variabile, *contatore*, di tipo *INTEGER* ed eventualmente inizializzarla a 100:

contatore INTEGER ::= 100

Spesso è necessario un sottotipo le cui variabili siano limitate a specifici valori o a uno specifico intervallo. Questo si può dichiarare nella maniera seguente:

Stato ::= INTEGER { su (1), giù (2), sconosciuto (3) }

DimensionePacchetto ::= INTEGER (0..1023)

Le variabili di tipo *BIT STRING* e *OCTET STRING* contengono zero o più bit e byte rispettivamente. Un bit è 0 o 1. Un byte varia nell'intervallo da 0 a 255, inclusi. Per entrambi i tipi, è possibile dare una lunghezza di stringa e un valore iniziale.

Gli *OBJECT IDENTIFIER* forniscono un modo per identificare gli oggetti. In linea di principio, qualsiasi oggetto definito in un qualsiasi standard ufficiale può venire identificato in maniera univoca. Il meccanismo utilizzato è quello di definire un albero di standard, e porre ciascun oggetto in ciascuno standard a una locazione unica dell'albero. La porzione di albero che comprende il MIB di SNMP è mostrata in figura 7-32.

Il livello più alto dell'albero elenca tutte le organizzazioni importanti di standard nel mondo (dal punto di vista ISO), che sono ISO e CCITT (adesso ITU), più la combinazione delle due. Dal nodo *iso*, escono 4 archi, uno dei quali è per l'*organizzazione-identificata*, la quale è una concessione dell'ISO che probabilmente indica qualcun'altro coinvolto con gli standard. Il Dipartimento della Difesa statunitense (DoD) ha un posto in questo sottoalbero e ha assegnato il numero 1 di Internet nella sua gerarchia. Sotto la gerarchia Internet, il MIB SNMP ha il codice 1.

Ogni arco in figura 7-32 ha un'etichetta e un numero. In questo modo è possibile identificare i nodi mediante una lista di archi, usando etichetta (numero) o numeri. Tutti gli oggetti MIB SNMP vengono identificati da un'etichetta del tipo

{iso organizzazione-identificata (3) dod (6) internet (1) mgmt (2) mib-2 (1) ... }

o alternativamente {1 3 6 1 2 1...}. Sono permesse anche le forme miste. Ad esempio, l'identificatore precedente può anche essere scritto come

{internet (1) 2 1...}

In questo modo, qualsiasi oggetto in qualsiasi standard può essere rappresentato come un *OBJECT IDENTIFIER*.

ASN.1 definisce cinque modi per costruire nuovi tipi da quelli base. *SEQUENCE* è una lista ordinata di tipi, simile a una struttura in C e un record in Pascal. *SEQUENCE OF* è un array a una dimensione di un singolo tipo. *SET* e *SET OF* sono simili, ma non ordinati.

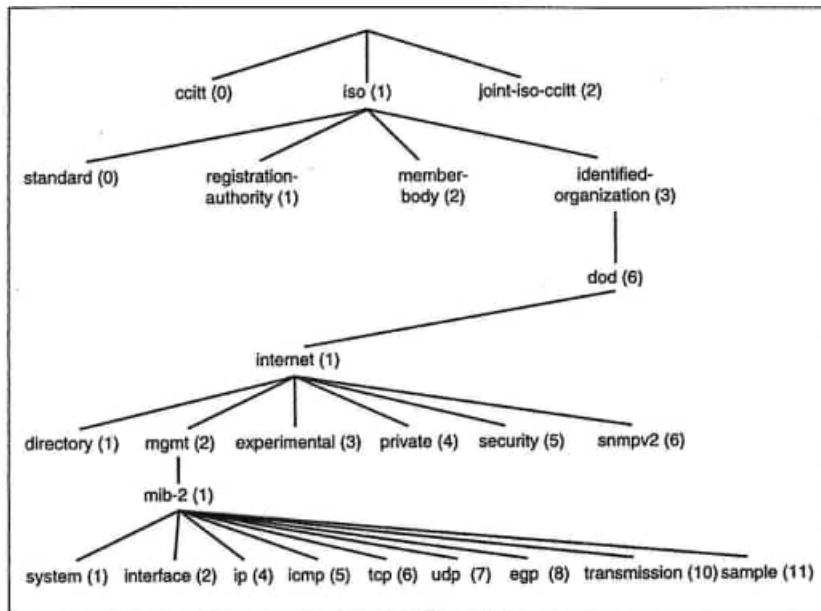


Fig. 7-32 Parte dell'albero dei nomi degli oggetti ASN.1.

CHOICE crea una unione da una data lista di tipi. I due costruttori di insiemi non sono usati in alcun documento SNMP.

Un altro modo per creare nuovi tipi è di rietichettare quelli vecchi. Rietichettare un tipo somiglia alla pratica in C per definire nuovi tipi, diciamo *time_t* e *size_t*, entrambi interi lunghi, ma utilizzati in contesti diversi. Le estensioni sono di quattro categorie: universali, di applicazione, dipendenti dal contesto e private. Ogni rietichettamento consiste in un'etichetta e in un intero che identifica l'etichetta. Ad esempio,

Contatore32 ::= [APPLICATION 1] INTEGER (0..4294967295)

Gauge32 ::= [APPLICATION 2] INTEGER (0..4294967295)

definiscono due tipi validi per l'applicazione, entrambi rappresentati con interi a 32 bit senza segno, ma che sono concettualmente differenti. Il primo potrebbe, ad esempio, ricominciare da capo quando arriva al valore massimo, mentre il secondo potrebbe semplicemente continuare a restituire il valore massimo fino a che questo non venga decrementato o resettato.

Un tipo rietichettato può avere la parola chiave *IMPLICIT* dopo la parentesi quadra chiusa quando il tipo seguente sia desumibile dal contesto (non vero in un *CHOICE*, ad esempio). Questo modo permette una codifica più efficiente dato che l'estensione non deve essere

trasmessa. In un tipo che coinvolge un *CHOICE* tra due tipi diversi, è necessario trasmettere un'estensione per informare il destinatario su quale sia il tipo presente.

ASN.1 definisce un complesso meccanismo di macro, che viene usato moltissimo in SNMP. È possibile usare una macro come un prototipo per generare un insieme di tipi e valori nuovi, ognuno con la propria sintassi. Ogni macro definisce alcune parole chiave (possibilmente opzionali), che vengono usate nella chiamata per identificare i parametri (infatti, i parametri di macro vengono identificati per parola chiave, non per posizione). I particolari di come funzionano le macro ASN.1 vanno al di là dello scopo di questo libro. È sufficiente dire che una macro viene invocata attraverso il suo nome e l'elenco di alcune sue parole chiave e loro valore relativo a questa invocazione. Le macro vengono espanso a tempo di compilazione, non a tempo di esecuzione. In seguito daremo alcuni esempi di macro.

Sintassi di trasferimento ASN.1

Una sintassi di trasferimento ASN.1 definisce il modo in cui i valori dei tipi ASN.1 vengono convertiti senza ambiguità in una sequenza di byte per la trasmissione (e decodificati senza ambiguità dalla parte opposta). La sintassi di trasferimento usata dall'ASN.1 viene detta **BER** (**B**asic **E**ncoding **R**ules – regole base di codifica). L'ASN.1 ha altre sintassi di trasferimento che l'SNMP non utilizza. Le regole sono ricorsive, così la codifica di un oggetto strutturato è semplicemente la concatenazione delle codifiche degli oggetti componenti. In questo modo, tutte le codifiche degli oggetti possono venire ricodificate a una sequenza ben definita di oggetti codificati primitivi. La codifica di questi oggetti, a turno, è definita dal BER.

Il principio guida dietro le regole base di codifica è che qualsiasi valore trasmesso, sia primitivo che costruito, consiste in non più di quattro campi:

1. L'identificatore (tipo o estensione).
2. La lunghezza di un campo dati, in byte.
3. Il campo dati.
4. Il flag fine-del-contenuto, se la lunghezza del dato è sconosciuta.

L'ultimo è permesso dall'ASN.1, ma espressamente proibito dall'SNMP, e quindi assumeremo che la lunghezza dei dati sia sempre nota.

Il primo campo identifica l'elemento che segue. Esso, di per sé, ha tre sottocampi come mostra la figura 7-33. I due bit di ordine più alto identificano il tipo dell'estensione. Il bit successivo indica se il valore è primitivo (0) oppure no (1). I bit dell'estensione sono 00, 01, 10 e 11, rispettivamente per *UNIVERSAL*, *APPLICATION*, dipendente dal contesto e *PRIVATE*. I rimanenti 5 bit possono essere usati per codificare il valore dell'estensione se si trova nell'intervallo da 0 a 30. Se l'estensione è 31 o maggiore di 31, i 5 bit di ordine più basso contengono 11111, con il valore vero nel prossimo o nei prossimi byte.

La regola usata per codificare le estensioni maggiori di 30 è stata pensata per gestire

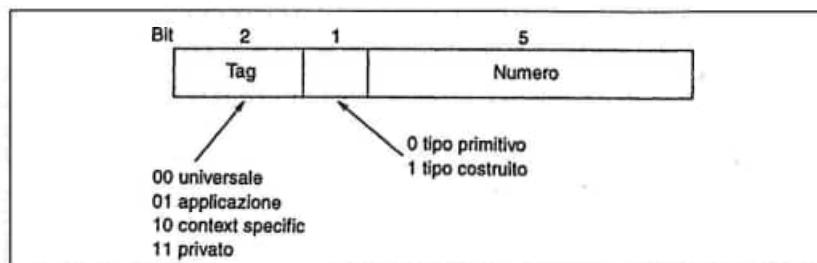


Fig. 7-33 Il primo byte di ogni elemento dati inviato nella sintassi di trasferimento ASN.1.

numeri arbitrariamente grandi. Ogni byte identificatore dopo il primo contiene 7 bit dati. Il bit di ordine massimo è posto a 0 in tutti i byte meno che l'ultimo. In questo modo i valori dell'estensione fino a $2^7 - 1$ possono stare in 2 byte, e fino a $2^{14} - 1$ in 3 byte.

La codifica dei tipi *UNIVERSAL* è immediata. Ogni tipo primitivo ha un codice, come si vede nella terza colonna di figura 7-31. *SEQUENCE* e *SEQUENCE OF* condividono il codice 16. *CHOICE* non ha codice, in quanto qualsiasi valore attuale ha sempre un tipo specifico. Gli altri codici sono per tipi non utilizzati dall'SNMP.

Dopo il campo identificatore di 1 byte segue un campo che indica quanti byte sono occupati dal dato. Le lunghezze minori di 128 byte sono codificate direttamente in 1 byte che ha il bit più a sinistra a 0. Quelli più lunghi utilizzano più byte, con il primo byte che ha 1 nel bit di ordine più alto e il campo lunghezza (fino a 127 byte) nei 7 bit di ordine più basso. Ad esempio, se la lunghezza del dato è di 1000 byte, il primo byte contiene 130 per indicare che segue un campo lunghezza di 2 byte. Quindi seguono 2 byte il cui valore è 1000, per primo il byte di ordine più alto.

La codifica del campo dato dipende dal tipo del dato presente. Gli interi sono codificati in complemento a 2. Un intero positivo minore di 128 richiede 1 byte, un intero positivo minore di 32.768 richiede 2 byte e così via. Il byte più significativo viene trasmesso per primo.

Le stringhe di bit vengono codificate come sono. Il solo problema è come indicare la lunghezza. Il campo lunghezza indica quanti siano i byte del valore, non quanti i bit. La soluzione scelta è quella di trasmettere 1 byte prima dell'attuale stringa di bit per indicare quanti bit (da 0 a 7) dell'ultimo byte siano inutilizzati. In questo modo la codifica della stringa di 9 bit '010011111' potrebbe essere 07, 4F, 80 (in esadecimale). Le stringhe otetto sono semplici. I byte della stringa sono trasmessi nello stile standard big endian, da sinistra a destra. Il valore nullo è indicato impostando il campo lunghezza a 0. Nessun valore viene trasmesso.

Un *OBJECT IDENTIFIER* viene codificato come la sequenza di interi che rappresenta. Per esempio, Internet è {1, 3, 6, 1}. Comunque, dato che il primo numero è sempre 0, 1 o 2, e il secondo è minore di 40 (per definizione-ISO, il sistema semplicemente non riconoscerà la 41-esima categoria che si presenta a bussare alla sua porta), i primi due numeri, *a* e *b*, vengono codificati con un byte di valore $40a + b$. Per Internet, questo numero è 43. Come al solito, i numeri che superano 127 vengono codificati in più byte, il primo dei quali contiene il bit di ordine più alto posto a 1 e un contatore di byte negli

altri 7 bit. Entrambi i tipi sequenza vengono trasmessi inviando dapprima il tipo o estensione, quindi la lunghezza totale della codifica per tutti i campi, seguita dagli stessi campi. I campi vengono inviati in ordine. La codifica di un valore *CHOICE* è la stessa della codifica della struttura dati attuale trasferita. La figura 7-34 contiene un esempio che mostra la codifica di alcuni valori. I valori codificati sono 49 per *INTEGER*, '110' per *OCTET STRING*, "xy", l'unico valore possibile per *NULL*, Internet {1, 3, 6, 1} per *OBJECT IDENTIFIER*, e 14 per un valore *Gauge32*.

	Tag type	Tag Number	Lunghezza	Valore
Integer 49	00000010	00000001	00110001	
Bit String '110'	00000011	00000010	00000101	11000000
Octet String "xy"	00000100	00000010	01111000	011111001
NULL	00000101	00000000		
Internet object	00000110	00000011	00101011	00000110 00000001
Gauge32 14	01000010	00000001	00001110	

Fig. 7-34 Codifica ASN.1 di alcuni valori di esempio.

7.3.3 SMI – Struttura delle informazioni di gestione

Nel paragrafo precedente, abbiamo discusso solamente quelle parti di ASN.1 che sono usate in SNMP. Nella realtà, i documenti SNMP sono organizzati in maniera diversa. Per prima cosa il documento RFC 1442 dice che ASN.1 verrà utilizzato per descrivere le strutture dati SNMP, quindi prosegue per 57 pagine eliminando le parti di ASN.1 che non interessano e aggiungendo delle nuove definizioni (in ASN.1) che sono necessarie. In particolare, l'RFC 1442 definisce 4 macro chiave e 8 nuovi tipi di dati che sono molto usati in SNMP. È questo sotto-super-insieme dell'ASN.1, noto come **SMI** (**struttura delle informazioni di gestione**), che viene in realtà usato per definire le strutture dei dati di SNMP. Al livello più basso, le variabili SNMP vengono definite come oggetti individuali. Gli oggetti collegati vengono collezionati insieme in gruppi, e i gruppi vengono riuniti in moduli. Ad esempio, esistono gruppi per oggetti IP e oggetti TCP. Un router può supportare il gruppo IP, in quanto il suo gestore si occupa di quanti pacchetti vengono persi. D'altra parte, un router a basso livello non può supportare il gruppo TCP, dato che non ha necessità di usare il TCP per eseguire le sue funzioni di routing. Generalmente i produttori che consentono l'utilizzo di un gruppo permettono di usare anche tutti gli oggetti che appartengono a quel gruppo. In ogni modo, un produttore che consente l'utilizzo di

un modulo non deve consentirlo per tutti i suoi gruppi, dato che non tutti possono essere applicabili a quel dispositivo.

Tutti i moduli MIB iniziano con l'invocazione della macro **MODULE-IDENTITY**. I suoi parametri forniscono il nome e l'indirizzo di chi ha effettuato l'implementazione, la storia delle revisioni e altre informazioni di carattere amministrativo. Tipicamente, questa chiamata è seguita da una invocazione della macro **OBJECT-IDENTITY**, che indica dove si trova il modulo nell'albero dei nomi di figura 7-32.

Seguono due o più invocazioni della macro **OBJECT-TYPE**, che nomina le variabili attuali da gestire e specifica le loro proprietà. Le variabili vengono raggruppate per convenzione; non esistono istruzioni **BEGIN-GROUP** e **END-GROUP** in ASN.1 o in SMI. La macro **OBJECT-TYPE** richiede quattro parametri e (a volte) altri 4 opzionali. Il primo parametro richiesto è **SYNTAX** che definisce il tipo di dato della variabile tra i tipi visti in figura 7-35. Per la maggior parte, questi tipi dovrebbero essere autoesplicativi, con i commenti che seguono. Il suffisso 32 viene usato quando chi implementa vuole veramente un numero a 32 bit, anche se tutte le macchine possibili hanno una CPU a 64 bit. I gauge differiscono dai contatori per il fatto che non sono ciclici nel momento in cui raggiungono i propri limiti. Rimangono dove sono. Se un router ha perso esattamente 2^{32} pacchetti è più giusto riportare questo evento con $2^{32} - 1$ che con 0. Anche l'SMI consente l'utilizzo di array, ma non ce ne occuperemo qui. Per i dettagli, si veda Rose (1994).

Nome	Tipo	Byte	Significato
INTEGER	Numeric	4	Intero (a 32 bit nell'implementazione attuale)
Counter32	Numeric	4	Contatore senza segno ciclico a 32 bit
Gauge32	Numeric	4	Valore senza segno non ciclico
Integer32	Numeric	4	A 32 bit, anche su una CPU a 64 bit
UInteger32	Numeric	4	Come Integer32, ma senza segno
Counter64	Numeric	8	Contatore a 64 bit
TimeTicks	Numeric	4	Centesimi di secondo a partire da un dato momento
BIT STRING	String	4	Mappa di bit da 1 a 32 bit
OCTET STRING	String	≥ 0	Stringa di byte a lunghezza variabile
Opaque	String	≥ 0	Obsoleto; usato solo per ragioni di compatibilità
OBJECT IDENTIFIER	String	≥ 0	Una lista di interi dalla fig. 7-32
IpAddress	String	4	Un indirizzo Internet e decimali separati da punti
NsapAddress	String	< 22	Un indirizzo NSAP OSI

Fig. 7-35 Tipo di dato delle variabili SNMP.

Oltre a richiedere la specifica del tipo di dato usato dalla variabile da dichiarare, la macro **OBJECT-TYPE** richiede anche tre parametri opzionali. **MAX-ACCESS** contiene le informazioni sull'accesso alla variabile. I valori più comuni sono lettura-scrittura e sola lettura. Se la variabile è lettura-scrittura, la stazione di gestione la può impostare. Se è solo lettura, la stazione di gestione può leggerla ma non modificarla.

STATUS ha tre valori possibili. Una variabile corrente è conforme alla specifica corrente

SNMP. Una variabile obsoleta non è conforme, ma era conforme in una versione più vecchia. Una variabile sconsigliata (deprecated) è una via di mezzo. È davvero obsoleta, ma chi ha scritto lo standard non lo ha dichiarato pubblicamente per paura della reazione dei produttori che vendono prodotti che la utilizzano. Ciò nonostante, la cosa è nota. L'ultimo parametro richiesto è **DESCRIPTION**, che è una stringa ASCII che indica ciò che fa la variabile. Se il gestore acquista un nuovo dispositivo, ne fa richiesta dalla stazione di gestione, e scopre così che essa tiene traccia di *pktCnt*, controllando il campo **DESCRIPTION** che si suppone dia un'idea del tipo di pacchetti che si stanno contando. Questo campo è pensato esclusivamente per un utilizzo umano (contrapposto al computer). In figura 7-36 vediamo un esempio di dichiarazione di un **OBJECT-TYPE**. La variabile è chiamata *lostPackets* e può essere utile in un router o in un altro dispositivo che tratti pacchetti. Il valore dopo il segno ::= lo inserisce nell'albero.

lostPackets OBJECT TYPE	
SYNTAX Counter32	-- use a 32-bit counter
MAX-ACCESS read-only	-- the management station may not change it
STATUS current	-- this variable is not obsolete (yet)
DESCRIPTION	
"The number of packets lost since the last boot"	
::= (experimental 20)	

Fig. 7-36 Un esempio di variabile SNMP

7.3.4 Il MIB – Management Information Base

La collezione di oggetti gestita da SNMP viene definita nel MIB. Per convenienza, questi oggetti sono (al momento attuale) raggruppati in dieci categorie, che corrispondono a 10 nodi al di sotto di *mib-2* in figura 7-32. (Notare che *mib-2* corrisponde a SNMPv2 e che l'oggetto 9 non è più presente.) Le dieci categorie sono pensate per fornire una base a ciò che deve compreso da una stazione di gestione. In futuro verranno sicuramente aggiunte nuove categorie e oggetti, e i produttori saranno liberi di definire oggetti aggiuntivi per i loro prodotti. La figura 7-37 riassume le dieci categorie.

Gruppo	#Oggetti	Descrizione
Sistema	7	Nome, locazione e descrizione dell'attrezzatura
Interfacce	23	Interfacce di rete e misura del loro traffico
AT	3	Traduzione di indirizzo (sconsigliata)
IP	42	Statistiche di pacchetti IP
ICMP	26	Statistiche sui messaggi ICMP ricevuti
TCP	19	Algoritmi, parametri e statistiche TCP
UDP	6	Statistiche di traffico UDP
EGP	20	Statistiche sul traffico del protocollo del gateway esterno
Trasmissione	0	Riservato a MIB particolari
SNMP	29	Statistiche sul traffico SNMP

Fig. 7-37 I gruppi di oggetti del MIB-II di Internet.

Sebbene le limitazioni di spazio ci impediscono di entrare nei dettagli di tutti i 175 oggetti definiti nel MIB-II, sono necessari alcuni commenti. Il gruppo di sistema permette al gestore di capire quale sia il dispositivo chiamato, chi lo abbia chiamato, quale hardware e software contenga, dove è collocato, e quale sia la sua funzione. Sono fornite anche la data e ora dell'ultimo avvio di sistema e il nome e l'indirizzo della persona da contattare. Con queste informazioni un'azienda può appaltare la gestione del sistema a un'altra azienda in una città distante, mettendola in grado di capire quale sia la configurazione attualmente da gestire e chi dover contattare in caso di problemi con dei dispositivi.

Il gruppo delle interfacce tratta degli adattatori di rete. Tiene traccia del numero di pacchetti e byte inviati e ricevuti dalla rete, del numero di quelli rifiutati, del numero di quelli in broadcast e la dimensione corrente della coda in uscita.

Il gruppo AT era presente nel MIB-I e forniva informazioni riguardanti le corrispondenze degli indirizzi (ad es., dagli indirizzi Ethernet a Internet). Queste informazioni sono state spostate agli specifici protocolli MIB nell'SNMPv2.

Il gruppo IP tratta del traffico IP al nodo e dal nodo. È particolarmente ricco di contatori per tener traccia del numero di pacchetti scaricati per una qualsiasi varietà di ragioni (ad es., nessuna conoscenza dell'instradamento a una destinazione, oppure perdita di risorse). Sono inoltre disponibili delle statistiche riguardo la frammentazione dei dati e loro riassemblamento. Tutte queste informazioni sono di particolare importanza per i router gestionali.

Il gruppo ICMP riguarda i messaggi di errore IP. Essenzialmente, contiene un contatore per ogni messaggio ICMP che registra quanti messaggi di quel tipo sono stati spediti.

Il gruppo TCP tiene sotto controllo il numero totale e corrente di connessioni aperte, segmenti inviati e di quanti ricevuti e varie altre statistiche.

Il gruppo UDP tiene conto del numero di datagram UDP inviati e ricevuti, e di quanti fra questi fossero non spedibili a causa di una porta sconosciuta o per qualche altra ragione.

Il gruppo EGP viene usato per i router che supportano il protocollo di gateway esterno. Tiene traccia di quanti pacchetti e di quale tipo siano usciti, entrati e rinviati tramite forward correttamente, e di quanti entrati e scaricati.

Il gruppo di trasmissione è il luogo in cui risiedono i MIB specifici di dispositivo. Per esempio, qui vengono mantenute le statistiche specifiche Ethernet. Lo scopo di inserire un gruppo vuoto nel MIB-II è per riservare l'identificatore {internet 2 1 9} per questi scopi.

L'ultimo gruppo serve a collezionare statistiche sull'operazione dello stesso SNMP. Quanti messaggi siano stati inviati, che tipo di messaggi fossero, e così via.

Il MIB-II è definito formalmente nell'RFC 1213. La parte essenziale dell'RFC 1213 consiste in 175 chiamate di macro simili a quelle di figura 7-36, con dei commenti che delimitano i 10 gruppi. Per ognuno dei 175 oggetti definiti, viene fornito il tipo di dato insieme a una descrizione testuale in inglese sul perché venga usata la variabile. Per ulteriori informazioni sul MIB-II, il lettore consulti tale RFC.

7.3.5 Il protocollo SNMP

Abbiamo appena visto che il modello sottostante l'SNMP è una stazione di gestione che

invia le richieste ad agenti nei nodi gestiti, interrogando le 175 variabili sopra menzionate, e molte altre variabili dipendenti dal produttore. L'ultimo nostro obiettivo è l'attuale protocollo che stazione di gestione e agenti utilizzano. Il protocollo in sé è definito nell'RFC 1448.

Normalmente l'SNMP viene usato quando la stazione di gestione invia a un agente una richiesta d'informazioni o di uno specifico aggiornamento sul suo stato. Idealmente, l'agente replica con le informazioni richieste o conferma di aver aggiornato il suo stato come richiesto. I dati vengono spediti utilizzando la sintassi di trasferimento ASN.1. Comunque, possono essere riportati anche diversi errori, del tipo Variabile Non Esistente.

L'SNMP definisce sette possibili messaggi da inviare. I sei messaggi provenienti da chi inizia sono elencati in figura 7-38 (il settimo messaggio è un messaggio di risposta). I primi tre richiedono la restituzione di valori di variabili. Il primo formato specifica in maniera esplicita il nome della variabile. Il secondo fa richiesta della variabile successiva, permettendo al gestore di elencare l'intero MIB in ordine alfabetico (il default è la prima variabile). Il terzo serve per i grossi trasferimenti, ad esempio per tabelle.

Messaggio	Descrizione
Get-request	Richiede il valore di una o più variabili
Get-next-request	Richiede il valore della variabile successiva
Get-bulk-request	Richiede una tabella grande
Set-request	Aggiorna una o più variabili
Inform-request	Messaggio da gestore a gestore per descrivere il MIB locale
SnmpV2-trap	Segnalazione di trap da agente a gestore

Fig. 7-38 Tipi dei messaggi dell'SNMP.

Segue un messaggio che permette al gestore di aggiornare le variabili di un agente, nel caso in cui la specifica di un oggetto permetta tali modifiche, chiaramente. Il successivo è una richiesta di informazioni che consente a un gestore di specificare a un altro gestore quali siano le variabili da lui gestite. Infine, troviamo il messaggio inviato da un agente a un gestore quando si verifica un trap.

Dopo aver visto alcuni dei protocolli di supporto usati dal livello delle applicazioni, arriviamo finalmente alle applicazioni reali. Quando si chiede: "Cosa stai per fare adesso?", pochi rispondono: "Sto per cercare alcuni nomi con il DNS". Di solito si risponde che si sta per leggere la posta o le news, navigare nel Web o guardare un film in rete. Nel seguito di questo capitolo spiegheremo con un certo grado di dettaglio come lavorano proprio queste quattro applicazioni.

La posta elettronica, o email, come è conosciuta dai suoi molti ammiratori, esiste da oltre due decenni. Il primo sistema di posta elettronica consisteva semplicemente di protocolli

di trasferimento file, con la convenzione che la prima linea di ogni messaggio (cioè, il file) contenesse l'indirizzo del destinatario. Con il tempo, le limitazioni di questo approccio divennero evidenti. Elenchiamo alcuni inconvenienti:

1. Inviare un messaggio a un gruppo di persone era scomodo. I dirigenti spesso avevano bisogno di questa possibilità per poter inviare dei memo ai loro dipendenti.
2. I messaggi non avevano una struttura interna, rendendo difficile l'elaborazione automatica. Ad esempio, se un messaggio reinviato veniva incluso nel corpo di un altro messaggio, era difficile estrarre la parte reinviata dal messaggio ricevuto.
3. Il mittente non sapeva mai se un messaggio fosse arrivato oppure no.
4. Se qualcuno pianificava di assentarsi per affari per alcune settimane e voleva che la sua posta venisse gestita dalla segretaria, non era una cosa facile da impostare.
5. L'interfaccia utente era integrata in maniera troppo primitiva con il sistema di trasmissione, richiedendo agli utenti dapprima di "editare" un file, quindi di lasciare l'editor per poi invocare il programma di trasferimento.
6. Non era possibile creare e inviare messaggi che contenessero contemporaneamente testo, disegni, fax e voce.

Con l'esperienza sono stati proposti dei sistemi di posta elettronica più elaborati. Nel 1982 le proposte di ARPANET furono pubblicate come RFC 821 (protocollo di trasmissione) ed RFC 822 (formato dei messaggi). Da allora sono divenute lo standard Internet de facto. Due anni più tardi, il CCITT definì la sua raccomandazione l'X.400, che in seguito fu adottato come la base per il MOTIS di OSI. Nel 1988, il CCITT modificò l'X.400 per allinearla a MOTIS. MOTIS doveva diventare l'applicazione bandiera per l'OSI, un sistema che doveva essere tutto per tutti.

Dopo un decennio di competizione, i sistemi di posta elettronica basati su RFC 822 sono ampiamente usati, mentre quelli basati su X.400 sono scomparsi dall'orizzonte. Il modo in cui un sistema messo insieme da un valido gruppo di studenti di informatica può battere un standard internazionale ufficiale fortemente sostenuto dalle PTT mondiali, da molti governi e da una parte sostanziale dell'industria informatica, ricorda la biblica storia di Davide e Golia. La ragione del successo dell'RFC 822 non dipende solo dal fatto che sia buono, ma anche dal fatto che l'X.400 sia stato progettato male e sia così complesso da non permettere a nessuno di implementarlo bene. Molte organizzazioni, dovendo scegliere tra un sistema di posta progettato semplicemente, ma funzionale basato sull'RFC 822, e un sistema di posta X.400 che si suppone veramente fantastico, ma non funzionale, sceglieranno il primo. Per una lunga discussione su cosa sia sbagliato nell'X.400, si veda l'appendice C di Rose (1993). Di conseguenza, la nostra discussione sulla posta elettronica si concentrerà sull'RFC 821 ed RFC 822, usati da Internet.

7.4.1 Architettura e servizi

In questo paragrafo discuteremo i sistemi di posta elettronica e come sono organizzati. Normalmente coesistono due sottosistemi: gli **agenti utente (useragent)**, che consentono di leggere e inviare posta, e gli **agenti di trasferimento messaggi**, che trasferiscono i messaggi dalla sorgente alla destinazione. Gli agenti utente sono programmi locali che forniscono un'interfaccia basata su comandi, su menu oppure grafica per interagire con il sistema di posta elettronica. Gli agenti di trasferimento messaggi sono tipicamente demoni di sistema che girano in background e trasferiscono la posta attraverso la rete. Tipicamente, i sistemi di posta elettronica supportano cinque funzioni base, come viene descritto nel seguito.

La **composizione** si riferisce al processo di creazione di messaggi e risposte. Anche se è possibile usare qualsiasi editor di testi per il corpo del messaggio, il sistema stesso può fornire aiuto per l'indirizzamento e i numerosi campi dell'intestazione inclusi in ogni messaggio. Ad esempio, nel rispondere a un messaggio, il sistema può estrarre l'indirizzo del mittente dal messaggio in arrivo e inserirlo automaticamente nella risposta al posto giusto.

Il **trasferimento** si riferisce al trasferimento dei messaggi dal mittente al destinatario. In molti casi, questo richiede di stabilire una connessione verso il destinatario o qualche macchina intermedia, spedire il messaggio e rilasciare la connessione. Il sistema di posta elettronica dovrebbe fare tutto questo in maniera automatica, senza coinvolgere l'utente. Bisogna inoltre **notificare** al mittente ciò che è accaduto al messaggio. È stato spedito? È stato rifiutato? È andato perso? Esistono diverse applicazioni per le quali è importante la "ricevuta di ritorno", che può avere anche un significato legale (ad es. un mandato di comparazione elettronico andato perso).

La **visualizzazione** dei messaggi in arrivo è necessaria perché le persone possano leggere i loro messaggi. A volte è necessaria una conversione oppure deve essere invocato un visualizzatore speciale, ad esempio, se il messaggio è un file PostScript oppure una voce digitalizzata. A volte sono comunque eseguite delle conversioni e delle semplici formattazioni.

L'**eliminazione** è il passo finale e riguarda quale recipiente usare per il messaggio dopo la sua ricezione. Le diverse possibilità includono di gettarlo via prima di leggerlo, gettarlo via dopo averlo letto, di salvarlo e così via. Inoltre è possibile recuperare e leggere di nuovo messaggi salvati, reindirizzarli o elaborarli in qualche modo.

Oltre a questi servizi base, molti sistemi di posta elettronica forniscono una grande varietà di funzioni avanzate. Citiamone brevemente alcune. Quando un utente si sposta, o quando è lontano per un certo periodo di tempo, può richiedere che la sua posta venga spedita da un'altra parte; e il sistema dovrebbe essere in grado di farlo automaticamente.

Molti sistemi permettono agli utenti di creare **caselle di posta elettronica (mailbox)**, per memorizzare la posta in arrivo. Sono necessari dei comandi per creare e distruggere le caselle, ispezionarne il contenuto, inserirvi e cancellarne messaggi e così via.

I dirigenti di una azienda spesso hanno necessità di inviare un messaggio a tutti i propri dipendenti, clienti o fornitori. Questo fa sorgere l'idea di una **mailing list**, o lista di **destinatari**, che è una lista di indirizzi di posta elettronica. Quando un messaggio viene inviato a una mailing list, a chiunque sia nella lista viene inviata una copia identica.

La posta elettronica registrata è un'altra idea importante, in modo da permettere al mit-

tente di sapere se il messaggio è arrivato. Alternativamente, è desiderabile avere la notifica automatica per i messaggi non spediti. In ogni caso, il mittente dovrebbe avere un certo controllo sulle informazioni di controllo di quanto è successo.

Altre funzioni avanzate sono le copie carbone, la posta elettronica ad alta priorità, la posta elettronica segreta (crittografata), recipienti alternativi se quello primario non è disponibile e la possibilità per le segretarie di gestire la posta elettronica dei loro capi.

La posta elettronica è attualmente molto utilizzata nelle industrie per le comunicazioni all'interno della società. Permette ai dipendenti in sedi lontane di cooperare tra loro su progetti complessi, anche se lavorano con fusi orari differenti. Eliminando molte differenze legate alla razza, all'età e al sesso, i dibattiti per posta elettronica tendono a mettere in evidenza le idee personali più che lo stato aziendale. Con la posta elettronica, un'idea brillante di una persona poco qualificata in merito a qualsiasi argomento, può avere più impatto di un'idea qualunque di una persona potenzialmente più idonea a trattarlo. Alcune aziende hanno stimato che la posta elettronica ha migliorato la loro produttività di almeno il 30% (Perry, Adam, 1992).

Un'idea chiave presente in tutti i sistemi di posta elettronica moderni è la distinzione tra la **busta** e il suo contenuto. La busta racchiude il messaggio. Contiene tutte le informazioni necessarie per trasportare il messaggio, come l'indirizzo del destinatario, la priorità e il livello di sicurezza, tutti quanti distinti dal messaggio stesso. Gli agenti di trasporto usano la busta per l'intradamento, proprio come fanno gli uffici postali.

Il messaggio all'interno della busta contiene due parti: l'**intestazione** e il **corpo**. L'intestazione contiene le informazioni di controllo per gli agenti utente. Il corpo è interamente per il destinatario umano. Buste e messaggi sono illustrati in figura 7-39.

7.4.2 L'agente utente (User Agent)

I sistemi di posta elettronica hanno due componenti principali, come abbiamo visto: gli agenti utente e gli agenti di trasferimento messaggi. In questo paragrafo vedremo gli agenti utente. Un agente utente è di solito un programma (a volte detto lettore di posta) che accetta una varietà di comandi per comporre, ricevere e rispondere ai messaggi, così come per manipolare le caselle postali. Alcuni agenti utente hanno un'interfaccia guidata da un menù o da icone che richiede l'uso di un mouse, mentre altre aspettano comandi di un carattere dalla tastiera. Funzionalmente sono la stessa cosa.

Inviare messaggi di posta elettronica

Per inviare un messaggio di posta elettronica, un utente deve fornire un messaggio, l'indirizzo del destinatario e possibilmente alcuni altri parametri (ad es., la priorità o il livello di sicurezza). Il messaggio può venire prodotto con un qualsiasi editore di testi disponibile, un programma di elaborazione testi o possibilmente con l'editore compreso nell'agente utente. L'indirizzo di destinazione deve essere in un formato che l'agente utente capisca.

Molti agenti utente si aspettano degli indirizzi DNS della forma *mailbox@locazione*. Dato che questi li abbiamo già analizzati in precedenza in questo capitolo, non ripeteremo qui il tutto. In ogni caso, è bene notare che esistono altre forme di indirizzamento. In particolare, gli indirizzi X.400 hanno un aspetto completamente diverso dagli indirizzi DNS. Sono composti da coppie *attributo = valore*, ad esempio

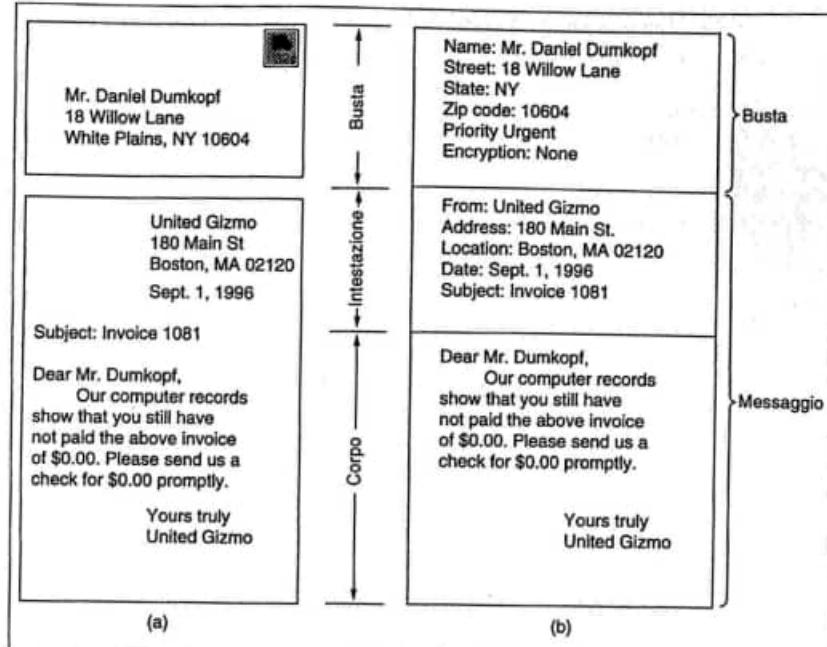


Fig. 7.39 Buste e messaggi. (a) Posta normale. (b) Posta elettronica.

/C=US/SP=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR.
/CN=KEN SMITH/

Questo indirizzo specifica un paese, uno stato, una località, un indirizzo personale e un nome comune (Ken Smith). Sono possibili molti altri attributi, in modo da inviare posta a qualcuno di cui non si conosca il nome, a patto di conoscere altri attributi sufficienti (ad es., il nome della società e la qualifica aziendale). Molte persone pensano che questa forma di indirizzamento sia assai meno conveniente dei nomi DNS.

In tutta onestà, comunque, i progettisti dell'X.400 assumevano che le persone utilizzassero gli **alias** (brevi stringhe assegnate dall'utente) per identificare i recipienti, in modo tale da non vedere mai gli indirizzi completi. In ogni modo, il software necessario non è mai stato pienamente disponibile, così le persone per inviare della posta a utenti con indirizzi X.400 avevano spesso bisogno di digitare stringhe del tipo precedente. Al contrario, la maggior parte dei sistemi di posta elettronica per Internet hanno sempre permesso agli utenti di usare dei file di alias.

La maggior parte dei sistemi di posta elettronica supporta le mailing list, per permettere a un utente di inviare lo stesso messaggio a una lista di persone con un solo comando. Se la mailing list viene mantenuta localmente, l'agente utente può semplicemente

inviare un messaggio separato a ognuno dei diversi destinatari. Al contrario, se la lista è mantenuta in maniera remota, allora il messaggio verrà espanso in quel sito. Ad esempio, se un gruppo di persone che pratica il bird watching avesse una lista di nome *birders* installata su *meadowlark.arizona.edu*, allora qualsiasi messaggio inviato a *birders@meadowlark.arizona.edu* verrà indirizzato all'Università dell'Arizona ed espanso là in messaggi individuali a tutti i componenti della mailing list, in qualsiasi parte del mondo essi si trovino. Gli utenti di questa mailing list non possono dire che sia una mailing list. Potrebbe semplicemente essere la casella postale personale del professor Gabriel O. Birders.

Leggere messaggi di posta elettronica

Di solito, quando un agente utente viene avviato, controlla la casella dell'utente per la posta in arrivo prima di visualizzare qualsiasi cosa sullo schermo. Quindi può annunciare il numero di messaggi nella casella o visualizzare un sommario di una linea per ciascuno di essi e attendere un comando.

Come esempio del modo in cui lavora un agente utente, vediamo un tipico scenario di posta. Dopo aver avviato l'agente utente, l'utente chiede il sommario dei suoi messaggi elettronici. Sullo schermo appare un elenco tipo quello di figura 7-40. Ogni linea si riferisce a un messaggio. In questo esempio, la casella postale contiene otto messaggi. Ogni linea visualizzata contiene diversi campi estratti dalla busta o dall'intestazione del messaggio corrispondente. In un sistema di posta elettronica semplice, la scelta dei campi visualizzati è fatta dal programma. In un sistema più sofisticato, l'utente può specificare quali siano i campi da visualizzare mediante un **profilo utente**, un file che descrive il formato di visualizzazione. In questo esempio, il primo campo è il numero del messaggio. Il secondo campo, *Flags*, può contenere un carattere *K*, che significa che il messaggio non è nuovo ma è già stato letto e mantenuto nella casella postale; un carattere *A*, che significa che al messaggio è già stato risposto; e/o un carattere *F*, che significa che il messaggio è stato rispedito a qualcun altro. Sono possibili anche altri indicatori.

#	Flags	Bytes	Mittente	Argomento
1	K	1030	asw	Changes to MINIX
2	KA	6348	radia	Comments on material you sent me
3	K F	4519	Amy N. Wong	Request for information
4		1236	bal	Deadline for grant proposal
5		103610	kaashoek	Text of DCS paper
6		1223	emily E.	Pointer to WWW page
7		3110	saniya	Referee reports for the paper
8		1204	dmr	Re: My student's visit

Fig. 7-40 Un esempio del contenuto di una casella postale.

Il terzo campo indica la lunghezza del messaggio e il quarto indica chi lo ha inviato. Dato che questo campo viene semplicemente estratto dal messaggio, può contenere i semplici nomi, i nomi completi (nome e cognome), le iniziali, i nomi di login o qualsiasi altra cosa il mittente scelga di inserirvi. Infine, il campo *Subject* dà un breve sommario del contenuto del messaggio. Le persone che non inseriscono un campo *Subject* spesso scoprono che le risposte ai propri messaggi non hanno la priorità massima.

Dopo aver visualizzato l'intestazione, l'utente può eseguire uno qualsiasi dei comandi disponibili. La figura 7-41 ne mostra una tipica collezione. Alcuni comandi richiedono un parametro. Il carattere *#* significa che è necessario il numero di un messaggio (o anche di più messaggi). Alternativamente, usando la lettera *a* si indicano tutti i messaggi.

Esistono innumerevoli programmi di posta elettronica. Il nostro esempio di programma di posta è modellato su quello usato dal sistema Mmdf di UNIX, che è abbastanza potente. Il comando *h* visualizza una o più intestazioni nel formato di figura 7-40. Il comando *c* stampa l'intestazione del messaggio corrente. Il comando *t* mostra (cioè, visualizza sullo schermo) il messaggio richiesto o i messaggi richiesti. Dei comandi possibili sono *t 3*, per vedere il messaggio 3, *t 4-6*, per vedere i messaggi dal 4 al 6, e *t a* per vederli tutti. Il successivo gruppo di tre comandi si occupa di inviare messaggi piuttosto che riceverli. Il comando *s* invia un messaggio richiamando un editore appropriato (ad es., specificato nel profilo utente) per permettere all'utente di comporre il messaggio. Correttori didattici, grammaticali e lessicali possono verificare che il messaggio sia sintatticamente corretto. Sfortunatamente, la generazione attuale di programmi di posta elettronica non ha correttori capaci di controllare se il mittente sa ciò di cui sta parlando. Quando il messaggio è terminato, viene preparato per la trasmissione all'agente di trasferimento messaggi.

Il comando *f* rispedisce un messaggio da una casella postale, richiedendo l'immissione di un indirizzo per l'invio.

Il comando *a* estrae l'indirizzo sorgente dal messaggio a cui si risponde e richiama l'editore per consentire all'utente di comporre la risposta.

Il successivo gruppo di comandi serve per manipolare le caselle di posta elettronica. Tipicamente gli utenti hanno una casella specifica per ogni persona con la quale hanno della corrispondenza, oltre alla casella generica per la posta in arrivo che abbiamo già visto.

Il comando *d* cancella un messaggio, mentre il comando *u* annulla la cancellazione. (Il messaggio non viene veramente cancellato fino a che non si esce dal programma di posta). Il comando *m* sposta un messaggio a un'altra casella. Questo è il modo in cui di solito si salvano i messaggi importanti dopo averli letti.

Il comando *k* conserva il messaggio indicato anche dopo averlo letto. Se un messaggio viene letto ma non mantenuto in maniera esplicita, all'uscita dal programma viene intrapresa una azione di default come spostare il messaggio in una casella speciale di default.

Infine, il comando *r* viene usato per lasciare la casella corrente e andare a leggerne un'altra.

I comandi *n*, *b* e *g* servono per spostarsi all'interno della casella corrente. Di solito un utente legge il messaggio 1, risponde, lo sposta o lo cancella, quindi digita *n* per vedere il successivo. Il valore di questo comando è che l'utente non deve ricordarsi di dov'è. È possibile tornare indietro usando *b* o andare a un certo messaggio con *g*.

Infine, il comando *e* termina il programma di posta elettronica ed esegue tutte le modifiche

richieste, come cancellare alcuni messaggi e marcarne altri come mantenuti. Questo comando riscrive la casella postale, rimpiazzandone il contenuto.

Nei sistemi di posta elettronica progettati per i principianti, ciascuno di questi comandi è associato a un'icona sullo schermo, in modo che l'utente non debba ricordare che a serve a *rispondere*. Deve invece ricordarsi che la figurina con una persona con la bocca aperta significa "rispondere" e non "visualizzare il messaggio".

Da questo esempio dovrebbe essere chiaro che la posta elettronica ha fatto tanta strada dai giorni in cui si trattava di un semplice trasferimento file. Gli agenti utenti sofisticati rendono possibile la gestione di grossi volumi di posta. Per persone che ricevono e inviano migliaia di messaggi in un anno, questi sono strumenti inestimabili.

Comando	Parametro	Descrizione
h	#	Visualizza l'intestazione (le intestazioni) sullo schermo
c		Visualizza solamente l'intestazione corrente
t	#	Mostra il/i messaggio/i sullo schermo
s	indirizzo	Invia un messaggio
f	#	Rispedisce il/i messaggio/i
a	#	Risponde al/ai messaggio/i
d	#	Cancella il/i messaggio/i
u	#	Elimina una precedente cancellazione del/dei messaggio/i
m	#	Sposta il/i messaggio/i a un'altra casella postale
k	#	Mantiene il/i messaggio/i dopo la chiusura
r	mailbox	Legge una nuova casella postale
n		Vai al prossimo messaggio e visualizzalo
b		Torna al messaggio precedente e visualizzalo
g	#	Vai al messaggio specificato ma non visualizzarlo
e		Esci dal sistema di posta e aggiorna la casella postale

Fig. 7-41 Tipici comandi per la gestione della posta.

7.4.3 Formato dei messaggi

Passiamo adesso dall'interfaccia utente al formato dei messaggi di posta elettronica in sé. Dapprima vedremo la posta elettronica ASCII base che utilizza l'RFC 822. Dopo questo, vedremo l'estensione multimediale dell'RFC 822.

RFC 822

I messaggi consistono in una busta primitiva (descritta nell'RFC 821), un certo numero di campi di intestazione, una linea bianca e quindi il corpo del messaggio. Ogni campo dell'intestazione (logicamente) consiste in una singola linea di testo ASCII contenente il nome del campo, un due punti e, per la maggior parte dei campi, un valore. L'RFC 822 è uno standard vecchio, e non distingue in maniera precisa la busta dai campi dell'intestazione, come farebbe

un nuovo standard. Nell'uso normale, l'utente costruisce un messaggio e lo passa all'agente di trasferimento messaggi, il quale usa alcuni campi dell'intestazione per costruire la busta attuale, qualcosa che è un mix di vecchio stile di messaggio e busta.

In figura 7-42 sono elencati i principali campi dell'intestazione relativi al trasferimento del messaggio. Il campo *To:* contiene l'indirizzo DNS del destinatario primario. È anche possibile avere più destinatari. Il campo *Cc:* contiene gli indirizzi di destinatari secondari. In termini di spedizione, non c'è alcuna distinzione tra i mittenti primari e secondari. È una differenza completamente psicologica che può essere importante per le persone coinvolte ma non è importante per il sistema di posta. Il termine *Cc:* (copia carbone) è un po' datato, in quanto i computer non utilizzano la carta carbone, come si sa. Il campo *Bcc:* (copia carbone trasparente) equivale al campo *Cc:*, eccetto che questa linea viene eliminata da tutte le copie inviate ai mittenti primari e secondari. Questa caratteristica permette alle persone di inviare copie a terze parti senza che i mittenti primari e secondari ne siano a conoscenza. I due campi successivi, *From:* e *Sender:* indicano rispettivamente chi ha scritto e inviato il messaggio. Questi possono non essere la stessa persona. Ad esempio, un uomo d'affari può scrivere un messaggio, mentre in realtà è la sua segretaria che lo trasmette. In questo caso, l'uomo d'affari apparirà nel campo *From:* e la segretaria nel campo *Sender:*. Il campo *From:* è necessario, mentre il campo *Sender:* può essere omesso se coincide con il campo *From:*. Questi campi sono necessari nel caso in cui non sia possibile spedire il messaggio, che deve quindi essere restituito al mittente.

Ogni agente di trasferimento messaggio lungo la via aggiunge una linea contenente *Received:*. La linea contiene l'identità dell'agente, la data e l'ora in cui il messaggio è stato ricevuto, e altre informazioni che possono venire utilizzate per trovare dei problemi nel sistema di instradamento.

Il campo *Return-Path:* viene aggiunto dall'ultimo agente di trasferimento messaggio e aveva lo scopo di indicare come tornare al mittente. In teoria, questa informazione potrebbe essere derivata da tutte le intestazioni *Received:* (eccetto che per il nome della casella del mittente), ma viene utilizzato di rado in questo modo, e di solito contiene solamente l'indirizzo del mittente.

Intestazione	Significato
To:	Indirizzo/i elettronici per destinatario/i primario/i
Cc:	Indirizzo/i elettronici per destinatario/i secondario/i
Bcc:	Indirizzo/i elettronici per copie carbone trasparenti
From:	Persona o persone che hanno creato il messaggio
Sender:	Indirizzo elettronico del mittente attuale
Received:	Linea aggiunta da ogni agente di trasferimento messaggi lungo la strada
Return-Path:	Può venire usato per individuare una strada all'indietro verso il mittente

Fig. 7-42 I campi intestazione relativi al trasferimento del messaggio in RFC 822.

Oltre ai campi di figura 7-42, i messaggi in RFC 822 possono contenere anche una varietà di campi intestazione usati dagli agenti utente o dai destinatari umani. Quelli più comuni sono elencati in figura 7-43. Molti di essi sono autoesplicativi, perciò non li vedremo in dettaglio.

Intestazione	Significato
Date:	Data e ora in cui il messaggio è stato inviato
Reply-To:	Indirizzo elettronico al quale inviare le risposte
Message-Id:	Numero unico per riferire questo messaggio in seguito
In-Reply-To:	Message-Id del messaggio al quale si sta rispondendo
References:	Altri Message-Id importanti
Keywords:	Parole chiave scelte dall'utente
Subject:	Breve sommario del messaggio per la visualizzazione su una linea

Fig. 7-43 Alcuni campi usati nell'intestazione del messaggio in RFC 822.

Il campo *Reply-To*: viene talvolta usato quando nè la persona che ha composto il messaggio nè la persona che invia il messaggio vogliono vedere la risposta. Ad esempio, un dirigente del marketing scrive un messaggio di posta elettronica per comunicare ai clienti un nuovo prodotto. Il messaggio viene inviato da una segretaria, ma il campo *Reply-To*: indica il capo del reparto vendite, che può rispondere a domande e prendere ordinazioni.

Il documento RFC 822 dice esplicitamente che gli utenti possono inventarsi nuove intestazioni per il loro uso privato, facendo iniziare queste intestazioni con la stringa *X-*. Viene garantito che nessun campo futuro utilizzerà nomi che iniziano con *X-*, per evitare conflitti tra intestazioni ufficiali e private. A volte gli studenti inseriscono campi del tipo *X-Frutto-del-Giorno*: o *X-Malattia-della-Settimana*: che sono legali, anche se non sempre illuminanti.

Dopo le intestazioni arriva il messaggio. Qui gli utenti possono mettere tutto ciò che vogliono. Alcune persone terminano il loro messaggio con firme elaborate, che comprendono semplici figure ASCII, citazioni famose o sconosciute, affermazioni politiche o avvertimenti di ogni tipo (ad es.: "La società ABS non è responsabile delle mie opinioni; non può neppure comprenderle").

MIME – Multipurpose Internet Mail Extensions

Ai primi tempi di ARPANET, la posta elettronica consisteva esclusivamente di messaggi testuali espressi in ASCII. In questo scenario, l'RFC 822 faceva tutto il necessario: specificava le intestazioni, ma lasciava completamente agli utenti la loro compilazione. Ai nostri giorni, nella rete Internet mondiale, questo approccio non è più adeguato. I problemi comprendono inviare e ricevere

1. Messaggi in lingue accentate (come francese e tedesco).
2. Messaggi in alfabeti non latini (come ebraico e russo).
3. Messaggi in lingue senza alfabeto (come cinese e giapponese).
4. Messaggi che non contengono assolutamente testo scritto (come audio e video).

Una soluzione fu proposta con l'RFC 1341 e aggiornata con l'RFC 1521. Questa soluzione, detta **MIME** (*Multipurpose Internet Mail Extensions*), attualmente è usata in maniera estesa. La descriveremo in seguito. Per ulteriori informazioni su MIME, si veda l'RFC 1521 oppure Rose (1993).

L'idea base di MIME è continuare a utilizzare il formato RFC 822, ma aggiungere una struttura al corpo del messaggio e di definire delle regole di codifica per i messaggi non ASCII. Con la scelta di non abbandonare l'RFC 822, è possibile inviare i messaggi MIME usando i programmi e i protocolli di posta elettronica esistenti. Tutto quello che va modificato sono i programmi per l'invio e la ricezione, cosa che gli utenti possono fare da soli.

MIME definisce cinque nuovi tipi di intestazioni del messaggio, mostrate in figura 7-44. La prima di queste dice semplicemente all'agente utente che riceve il messaggio che sta trattando un messaggio MIME, e quale versione di MIME è usata. Ogni messaggio che non contenga un'intestazione *MIME-Version*: si assume che sia un messaggio di testo in chiaro, e viene elaborato come tale.

L'intestazione *Content-Description*: è una stringa ASCII che indica ciò che il messaggio contiene. Questa intestazione è necessaria affinché il ricevente capisca se è il caso di decodificare e leggere il messaggio. Se la stringa dice: "Foto del gerbillo di Barbara" e la persona che riceve il messaggio non è un grande ammiratore dei gerbilli, probabilmente ignorerà il messaggio invece di decodificarlo in una grossa fotografia a colori ad alta risoluzione.

Intestazione	Significato
<i>MIME-Version</i> :	Identifica la versione MIME
<i>Content-Description</i> :	Stringa leggibile che descrive il contenuto del messaggio
<i>Content-Id</i> :	Identificatore unico
<i>Content-Transfer-Encoding</i> :	Modo in cui il corpo viene trattato per la trasmissione
<i>Content-Type</i> :	La natura del messaggio

Fig. 7-44 Le intestazioni RFC 822 aggiunte da MIME.

L'intestazione *Content-Id*: identifica il contenuto. Usa lo stesso formato dell'intestazione standard *Message-Id*.

L'intestazione *Content-Transfer-Encoding*: dice come è stato trattato il corpo per la trasmissione attraverso una rete che può sollevare obiezioni per caratteri diversi da lettere, numeri e punteggiatura. Ci sono cinque schemi (più un modo per definire nuovi schemi). Lo schema più semplice è il solo testo ASCII. I caratteri ASCII usano 7 bit e possono essere trasportati direttamente dal protocollo di posta elettronica se ciascuna linea non supera i 1000 caratteri.

Lo schema più semplice successivo è lo stesso, ma utilizza 8 bit, cioè tutti i valori da 0 fino a 255 compreso. Questo schema di codifica viola il protocollo (originale) di posta elettronica di Internet, ma viene usato da alcune parti di Internet che implementano alcune estensioni al protocollo originale. Mentre il dichiarare la codifica non la rende legale, esplicitarla può almeno motivare il caso in cui qualcosa vada male. I messaggi che usano la codifica a 8 bit devono comunque rispettare la lunghezza massima standard per una linea.

Va ancora peggio per i messaggi che utilizzano una codifica binaria. Questi sono file binari arbitrari che non solo utilizzano 8 bit ma che non rispettano neppure il limite di 1000 caratteri per linea. I programmi eseguibili ricadono in questa categoria. Non c'è nessuna garanzia che i messaggi in binario arrivino correttamente, ma molte persone li spediscono comunque.

Il modo corretto per codificare i messaggi in binario è di utilizzare la **codifica base64**, spesso chiamata **protezione ASCII**. In questo schema, i gruppi di 24 bit vengono suddivisi in 4 unità di 6 bit, e ogni unità viene spedita come un carattere ASCII legale. La codifica è "A" per 0, "B" per 1, e così via, seguite dalle 26 lettere in minuscolo, le 10 cifre e infine da + e - per 62 e 63, rispettivamente. Le sequenze == e = vengono rispettivamente usate per indicare che l'ultimo gruppo contiene solamente 8 o 16 bit. I ritorni carrello e gli "a capo" vengono ignorati, così possono venire inseriti a piacere per mantenere le linee abbastanza brevi. Con questo schema è possibile spedire in maniera sicura qualsiasi testo binario.

Per i messaggi che sono quasi interamente in ASCII, ma che contengono alcuni caratteri non ASCII, la codifica base64 è abbastanza inefficiente. Al suo posto, viene usata la **codifica stampabile (quoted-printable encoding)**. Questa è semplicemente ASCII a 7 bit, con tutti i caratteri superiori a 127 codificati con un segno uguale seguito dal valore del carattere espresso con 2 cifre esadecimali.

Riassumendo, i dati binari si dovrebbero spedire codificati nella forma base64 o stampabile. Quando esistono ragioni valide per non utilizzare uno di questi schemi, è possibile specificare una codifica definita dall'utente nell'intestazione *Content-Transfer-Encoding*:

L'ultima intestazione mostrata in figura 7-44 è davvero quella più interessante. Specifica la natura del corpo del messaggio. L'RFC 1521 definisce sette tipi, ognuno dei quali ha uno o più sottotipi. Il tipo e il sottotipo sono separati da un carattere /, come in

Content-Type: video/mpeg

Il sottotipo deve essere dato in maniera esplicita nell'intestazione; non viene fornito alcun default. La figura 7-45 dà la lista iniziale di tipi e sottotipi specificati nell'RFC 1521. Da allora ne sono stati aggiunti molti altri, e altri elementi vengono aggiunti nel momento in cui sorge la necessità.

Analizziamo a questo punto la lista dei tipi. Il tipo *text* è per il puro testo. La combinazione *text/plain* è per i messaggi comuni che si possono visualizzare appena si ricevono, senza alcuna codifica e ulteriore elaborazione. Questa opzione permette di trasportare i messaggi ordinari in MIME aggiungendo solamente delle intestazioni extra.

Il sottotipo *text/richtext* permette di includere nel testo un linguaggio di formattazione semplice. Questo linguaggio fornisce un modo indipendente dal sistema per esprimere grassetto, corsivo, dimensioni di punto minori e maggiori, rientri di margine, giustificazione, apici e pedici e una semplice impaginazione. Il linguaggio di formattazione si basa sull'*SGML*, lo Standard Generalized Markup Language, utilizzato anche come base per l'*HTML* del World Wide Web. Ad esempio, il messaggio

È venuto il momento disse il <i>tricheco</i> di parlar ...

verrebbe visualizzato come

È venuto il **momento** disse il *tricheco* di parlar ...

È compito del sistema ricevente scegliere l'interpretazione appropriata. Se sono disponibili grassetto e corsivo, verranno utilizzati; altrimenti, per evidenziare, possono venire usati i colori, il lampeggiamento, la sottolineatura, l'inversione di colore ecc. Sistemi differenti possono fare, e fanno, scelte differenti.

Il successivo tipo MIME è *image*, che viene usato per trasmettere immagini ferme. Attualmente ci sono diversi formati usati per la memorizzazione e la trasmissione di immagini, sia con che senza compressione. Due di questi, GIF e JPEG, sono sottotipi ufficiali, ma senza dubbio in futuro ne saranno aggiunti altri.

I tipi *audio* e *video* sono per il suono e le immagini in movimento, rispettivamente. Si noti che *video* comprende solamente le informazioni di visualizzazione, non la colonna sonora. Se deve essere trasmesso un filmato col suono, le porzioni audio e video devono essere trasmesse separatamente, a seconda del sistema di codifica usato. L'unico formato video definito fino a ora è quello inventato dal gruppo Moving Picture Experts Group (MPEG).

Tipo	Sottotipo	Descrizione
Text	Plain	Testo non formattato
	Richtext	Testo con semplici comandi di formattazione
Image	Gif	Immagine ferma in formato GIF
	Jpeg	Immagine ferma in formato JPEG
Audio	Basic	Suono ascoltabile
Video	Mpeg	Filmato in 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	Messaggio suddiviso per la trasmissione
	External-body	Il messaggio stesso deve essere recuperato tramite rete
Multipart	Mixed	Parti indipendenti nell'ordine specificato
	Alternative	Lo stesso messaggio in formati differenti
	Parallel	Le parti vanno visualizzate simultaneamente
	Digest	Ogni parte è un messaggio RFC 822 completo

Fig. 7-45 I tipi e sottotipi MIME definiti nell'RFC 1521.

Il tipo *application* è la scelta di rigore per tutti quei formati che richiedono dell'elaborazione esterna non prevista da uno degli altri tipi. Un *octet-stream* è una sequenza

di byte non interpretabile. Un agente utente, dopo aver ricevuto una stringa di questo tipo, dovrebbe probabilmente visualizzarla suggerendo all'utente di copiarla in un file e proponendogli un nome per il file. Elaborazioni successive dipendono a questo punto dall'utente.

L'altro sottotipo definito è *postscript*, che si riferisce al linguaggio PostScript prodotto dalla Adobe Systems e largamente usato per descrivere pagine da stampare. Molte stampanti includono interpreti PostScript. Anche se un agente utente può chiamare semplicemente un interprete PostScript per visualizzare i file PostScript in arrivo, questa azione non è priva di rischi. Il PostScript ha tutte le caratteristiche di un linguaggio di programmazione. Dandogli abbastanza tempo, una persona sufficientemente masochista potrebbe scrivere un compilatore C o un sistema di gestione di base di dati in PostScript. La visualizzazione di un messaggio PostScript in arrivo è fatta eseguendo il programma PostScript che esso contiene. Oltre a visualizzare del testo, questo programma può leggere, modificare o cancellare file dell'utente, oppure avere altri effetti nocivi.

Il tipo *message* permette di incapsulare totalmente un messaggio in un altro. Questo schema è utile per rispedire dei messaggi, ad esempio. Quando un messaggio RFC 822 completo viene incapsulato dentro un messaggio in uscita, si dovrebbe usare il sottotipo *rfc822*.

Il sottotipo *partial* rende possibile la suddivisione di messaggi incapsulati in più parti e il loro invio separato (ad es., se il messaggio incapsulato è troppo lungo). I parametri consentono al destinatario di riassemblare tutte le parti nell'ordine corretto.

Infine, per messaggi molto lunghi si utilizza il sottotipo *external-body* (ad es., per filmati video). Invece di includere il file MPEG nel messaggio, viene dato un indirizzo FTP e l'agente utente del ricevente può recuperarlo via rete nel momento in cui vuole. Questa caratteristica è particolarmente utile quando si vuole inviare un filmato a una lista di persone, delle quali ci aspettiamo che solamente alcune lo vedranno (si pensi a una lista elettronica che contiene video pubblicitari).

L'ultimo tipo è il *multipart*, che permette a un messaggio di contenere più di una parte, con l'inizio e la fine di ciascuna parte chiaramente delimitate. Il sottotipo *mixed* permette a ogni parte di essere differente, senza imporre alcuna struttura aggiuntiva. Al contrario, con il sottotipo *alternative* ogni parte deve contenere lo stesso messaggio espresso però con un mezzo o codifica differente. Ad esempio, un messaggio può essere spedito in puro testo, in formato richtext e in PostScript. Un agente utente opportunamente progettato per ricevere un tale messaggio potrebbe se possibile visualizzarlo in PostScript. La seconda scelta potrebbe essere il formato richtext. Se nessuno di questi è possibile, verrà visualizzato il testo ASCII. Le parti dovrebbero essere ordinate dalla più semplice alla più complessa per aiutare i destinatari che utilizzano agenti utente precedenti al MIME a dare un senso al messaggio (infatti, anche un agente utente precedente al MIME può leggere un testo in ASCII).

Il sottotipo *alternative* si può usare anche per più lingue. In questo contesto, si può pensare alla Stele di Rosetta come a un precoce messaggio *multipart/alternative*.

La figura 7-46 mostra un esempio multimediale. Qui viene trasmesso un augurio di buon compleanno sia come testo che come canzone. Se chi riceve ha le capacità di ascolto, allora

l'agente utente recupererà il file col suono, *birthday.snd*, e lo suonerà. Altrimenti, verranno visualizzate le parole sullo schermo in un silenzio di pietra. Le parti sono delimitate da due trattini seguiti dalla stringa (definita dall'utente) che specifica il parametro di *confine*.

Si noti che in questo esempio l'intestazione *Content-Type* è presente tre volte. Al livello più alto, indica che il messaggio ha più parti. All'interno di ciascuna parte, indica il tipo e sottotipo di ciascuna parte. Infine, all'interno del corpo della seconda parte, è necessario per indicare all'agente utente quale tipo di file esterno debba recuperare. Per indicare questa differenza di utilizzo, abbiamo usato le lettere minuscole, anche se tutte le intestazioni sono insensibili alla differenza tra maiuscole e minuscole. Allo stesso modo per qualsiasi corpo esterno che non sia codificato come ASCII a 7 bit è richiesto il *Content-Transfer-Encoding*. Tornando ai sottotipi per messaggi multipart, esistono altre due possibilità. Il sottotipo *parallel* viene usato quando tutte le parti devono essere "visualizzate" simultaneamente. Per esempio, i film spesso hanno un canale audio e un canale video. I film hanno miglior effetto se questi due canali vengono trasmessi in parallelo, invece che uno dopo l'altro. Infine, il sottotipo *digest* viene usato quando c'è un messaggio composito che raggruppa più messaggi. Ad esempio, alcuni gruppi di discussione su Internet collezionano i messaggi dai sottoscriventi e li spediscono come un singolo messaggio *multipart/digest*.

7.4.4 Trasferimento di messaggi

Il sistema di trasferimento di messaggi riguarda il passaggio dei messaggi dall'origine al ricevente. La maniera più semplice per fare questo è stabilire una connessione di trasporto dalla macchina sorgente alla macchina destinataria e quindi trasferire semplicemente il messaggio. Dopo aver esaminato il modo in cui questo viene normalmente fatto, analizzeremo alcune situazioni in cui questo non funziona e cosa bisogna fare.

SMTP – Simple Mail Transfer Protocol

All'interno di Internet, la posta elettronica viene spedita quando la macchina mittente ha stabilito una connessione TCP alla porta 25 della macchina destinataria. Su questa porta c'è in ascolto un demone della posta elettronica il quale parla il protocollo **SMTP (Simple Mail Transfer Protocol)**. Questo demone accetta le connessioni in arrivo e copia da queste i messaggi nelle appropriate caselle postali. Se non è possibile spedire un messaggio, al mittente viene restituita una notifica di errore contenente la prima parte del messaggio non spedito.

SMTP è un semplice protocollo ASCII. Dopo aver stabilito la connessione TCP con la porta 25, la macchina mittente, che opera come un cliente, attende che la macchina destinataria, che opera come un server, parli per prima. Il server inizia spedendo una linea di testo che lo identifica e dice se è pronto o no a ricevere la posta. Se non lo è, il cliente rilascierà la connessione e riproverà più tardi.

Se il server è intenzionato ad accettare la posta, il cliente annuncia da chi è inviato il messaggio e anche a chi dev'essere consegnato. Se tale destinatario esiste alla destinazione, il server dice al cliente di proseguire nell'invio del messaggio. Il cliente quindi invia il messaggio e il server ne dà conferma. Di solito non è necessario alcun controllo di validità perché il TCP fornisce un flusso di byte affidabile. Se c'è altra posta, viene inviata a questo punto. Quando tutta la posta viene scambiata in entrambe le direzioni, viene

```

From: elinor@abc.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@abc.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

```

This is the preamble. The user agent ignores it. Have a nice day.

```
--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/richtext
```

```

Happy birthday to you
Happy birthday to you
Happy birthday dear <b>Carolyn</b>
Happy birthday to you

```

```
--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
access-type="anon-ftp";
site="bicycle.abc.com";
directory="pub";
name="birthday.snd"
```

```

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

Fig. 7-46 Un messaggio *multipart* contenente le alternative richtext e audio.

rilasciata la connessione. La figura 7-47 mostra un esempio di dialogo per inviare il messaggio di figura 7-46, compresi i codici numerici usati dall' SMTP. Le linee inviate dal cliente iniziano con *C:*, quelle inviate dal server con *S:*.

Possono essere utili alcuni commenti sulla figura 7-47. Il primo comando dal cliente è proprio *HELO*. Delle due abbreviazioni a quattro caratteri di *HELLO* questa ha numerosi vantaggi rispetto all'altra. Il perché tutti i comandi debbano essere di quattro caratteri si perde nella notte dei tempi.

In figura 7-47 il messaggio viene inviato a un solo destinatario, perciò viene usato un solo comando *RCPT*. Per inviare lo stesso messaggio a più destinatari vengono usati più comandi di questo tipo. Ognuno di essi viene confermato o rifiutato individualmente. Anche se alcuni mittenti vengono rifiutati (perché non esistono alla destinazione), il messaggio si può spedire a tutti gli altri.

Infine, anche se la sintassi dei comandi a quattro caratteri dal cliente è specificata in

```

S: 220 xyz.com SMTP service ready
C: HELO abc.com
S: 250 xyz.com says hello to abc.com
C: MAIL FROM: <elinor@abc.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@abc.com
C: To: carolyn@xyz.com
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@abc.com>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: Earth orbits sun integral number of times
C:
C: This is the preamble. The user agent ignores it. Have a nice day.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/richtext
C:
C: Happy birthday to you
C: Happy birthday to you
C: Happy birthday dear <b>Carolyn</b>
C: Happy birthday to you
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C: access-type="anon-ftp";
C: site="bicycle.abc.com";
C: directory="pub";
C: name="birthday.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
S: 250 message accepted
C: QUIT
S: 221 xyz.com closing connection

```

Fig. 7-47 Trasferimento di un messaggio da *elinor@abc.com* a *carolyn@xyz.com*.

maniera rigida, la sintassi per le risposte è meno rigida. Ciò che veramente conta è il codice numerico. Ogni implementazione può inserire qualsiasi stringa dopo il codice.

Anche se il protocollo SMTP è ben definito (dall'RFC 821), possono insorgere alcuni problemi. Un problema è legato alla lunghezza del messaggio. Alcune implementazioni più vecchie non potevano gestire messaggi superiori a 64 KB. Un altro problema riguarda le scadenze (timeout). Se il cliente e il server hanno timeout differenti, uno di essi può scadere quando l'altro è ancora valido, terminando inaspettatamente la connessione. Infine, in situazioni rare, si può verificare un traffico di posta infinito. Ad esempio, se il sistema 1 mantiene la mailing list A e il sistema 2 mantiene la mailing list B e ciascuna lista contiene un riferimento all'altra, allora qualsiasi messaggio inviato all'altra lista genererà un traffico di posta senza fine.

Per ovviare ad alcuni di questi problemi, è stato definito un protocollo SMTP esteso (**ESMTP**) nell'RFC 1425. I clienti che lo vogliono utilizzare devono spedire un messaggio *EHLO* iniziale al posto di *HELO*. Se questo viene rifiutato, allora il server è un SMTP standard, e il cliente deve procedere nel solito modo. Se viene accettato *EHLO*, allora sono permessi comandi e parametri nuovi. La standardizzazione di questi comandi e parametri è un processo in fase di sviluppo.

Gateway di posta elettronica

La posta elettronica che utilizza SMTP funziona meglio se sia il mittente che il destinatario si trovano su Internet e supportano connessioni TCP tra mittente e destinatario. In ogni modo, molte macchine che non si trovano in Internet continuano a voler inviare e ricevere la posta dai nodi Internet. Ad esempio, molte aziende preferiscono non essere in Internet per problemi di sicurezza. Alcune di esse addirittura si estromettono da Internet interponendo dei firewall.

Un altro problema insorge quando il mittente parla solamente RFC 822 e il ricevente parla solamente X.400 o il linguaggio di qualche protocollo proprietario di posta elettronica. Date che tutti questi mondi differiscono nei formati dei messaggi e nei protocolli, la comunicazione diretta è impossibile.

Questi problemi vengono entrambi risolti utilizzando i **gateway di posta elettronica** del livello delle applicazioni. Nella figura 7-48 la macchina 1 parla solamente TCP/IP ed RFC 822, mentre la macchina 2 parla solamente OSI TP4 e X.400. Ciò nonostante, possono

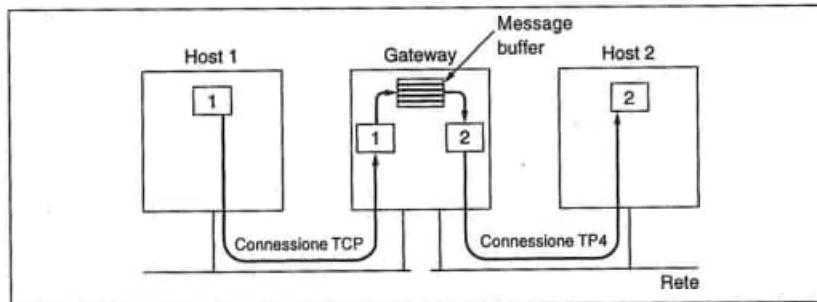


Fig. 7-48 Trasferimento di posta elettronica mediante un gateway a livello delle applicazioni.

scambiarsi la posta utilizzando un gateway di posta elettronica. Per la macchina 1 la procedura è quella di stabilire una connessione TCP col gateway e quindi usare l'SMTP per trasferirvi il messaggio (1). Quindi il demone nel gateway inserisce il messaggio in un buffer di messaggi destinati alla macchina 2. In seguito, viene stabilita una connessione TP4 (l'equivalente OSI per TCP) con la macchina 2 e il messaggio (2) viene trasferito utilizzando l'equivalente OSI dell'SMTP. Tutto quello che deve fare il processo del gateway è estrarre i messaggi in arrivo da una coda e depositarli in un'altra.

Sembra facile, ma non lo è. Il primo problema è che gli indirizzi Internet e gli indirizzi X.400 sono totalmente diversi. Tra loro è necessario un meccanismo elaborato per la corrispondenza. Il secondo problema è che la busta o i campi intestazione presenti in un sistema possono non essere presenti nell'altro. Ad esempio, se un sistema richiede le classi di priorità e l'altro non ha assolutamente questo concetto, in una direzione l'informazione di valutazione dev'essere eliminata e nell'altro va generata al volo.

Peggio ancora è ciò che si deve fare se le parti del corpo sono incompatibili. Cosa deve fare un gateway con un messaggio proveniente da Internet il cui corpo contenga un riferimento a un file audio da ottenere con un FTP nel caso in cui il sistema destinatario non supporti questa funzione? Cosa dovrebbe fare quando un sistema X.400 gli dice di spedire un messaggio a un certo indirizzo, ma se questo fallisce, di inviare il contenuto via fax? L'utilizzo del fax non fa parte del modello RFC 822. Chiaramente, in questo caso non ci sono soluzioni semplici. Per semplici messaggi testuali ASCII non strutturati la soluzione del gateway è ragionevole, ma per qualsiasi funzione più avanzata l'idea fallisce.

Trasmissione finale

Fino a questo punto abbiamo assunto che tutti gli utenti lavorino su macchine in grado di inviare e ricevere posta elettronica. Molto spesso questa situazione non si verifica. Ad esempio, in molte aziende gli utenti lavorano su PC che non sono collegati in Internet e non sono in grado di inviare o ricevere posta al di fuori dell'azienda. Al contrario, l'azienda ha uno o più server di posta elettronica in grado di inviare e ricevere messaggi. Per inviare o ricevere un messaggio un PC deve parlare con un server di posta mediante un certo tipo di protocollo di spedizione.

Un semplice protocollo usato per recuperare la posta da una casella postale remota è il **POP3 (Post Office Protocol)**, definito nell'RFC 1225. Ha comandi per permettere all'utente di connettersi, disconnettersi, recuperare messaggi e cancellarli. Il protocollo in sé è costituito da testo ASCII e ha qualcosa che ricorda l'SMTP. Lo scopo del POP3 è recuperare la posta dalla casella remota e memorizzarla nella macchina locale dell'utente per leggerla in seguito.

Un protocollo di trasmissione più sofisticato è l'**IMAP (Interactive Mail Access Protocol)**, definito nell'RFC 1064. Era stato progettato per aiutare l'utente che usasse diversi computer, come una workstation in ufficio, un PC a casa e un portatile in viaggio. L'idea base dell'IMAP è che il server della posta debba conservare un deposito centrale che sia accessibile da qualsiasi macchina. In questo modo, a differenza di POP3, IMAP non copia la posta sulla macchina personale dell'utente perché può averne parecchie.

IMAP ha molte altre caratteristiche, come la capacità di indirizzare la posta non a seconda del numero di arrivo come avviene nella figura 7-40, ma utilizzando degli attributi (ad

esempio, "Dammi il primo messaggio per Sandro"). In questo modo, una casella postale somiglia di più a una base di dati relazionale che a una sequenza lineare di messaggi. Ancora un altro protocollo è il **DMSP (Distributed Mail System Protocol)**, che fa parte del sistema PCMAIL ed è descritto nell'RFC 1056. Questo non assume che la posta si trovi tutta su un solo server, come fanno POP3 e IMAP. Invece, permette agli utenti di scaricare la posta dal server a un computer e quindi di disconnettersi. È possibile leggere la posta e rispondere mentre si è disconnessi. Quando più tardi ci sarà una nuova connessione, la posta verrà trasferita e il sistema verrà nuovamente sincronizzato.

Indipendentemente dal fatto che la posta sia spedita direttamente dal computer dell'utente o da un server remoto, molti sistemi forniscono dei modi per elaborare ulteriormente la posta in arrivo. Uno strumento particolarmente prezioso per molti utenti di posta elettronica è la capacità di impostare dei **filtri**. Sono regole che vengono verificate quando arriva la posta o quando viene avviato l'agente utente. Ogni regola specifica una condizione e un'azione. Ad esempio, una regola potrebbe dire che qualsiasi messaggio proveniente da Andrew S. Tanenbaum dovrebbe essere visualizzato in un corpo a 24 punti lampeggiante in grassetto di colore rosso (oppure essere eliminato automaticamente senza commenti). Un'altra funzione di trasmissione spesso presente è la capacità di rispedire (temporaneamente) la posta in arrivo a un indirizzo differente. Questo indirizzo può anche essere un computer che lavora per un servizio di notifica commerciale, che quindi contatta l'utente via radio o via satellite, visualizzando la linea *Subject:* sul suo cercapersonne.

Un'altra funzione abbastanza comune è il **demon di vacanza**. È un programma che esamina ogni messaggio in arrivo e invia al mittente una risposta del tipo

Salve. Sono in vacanza. Tornerò il 24 agosto. Buona giornata.

Queste risposte potrebbero anche specificare come gestire questioni urgenti, contattare altre persone per problemi specifici ecc. Molti demoni di vacanza tengono traccia di coloro ai quali sono stati spedite le risposte prestabilite e non rispediscono alla stessa persona una seconda risposta. Quelli migliori inoltre verificano se il messaggio in arrivo sia stato spedito da una mailing list, e se sì non mandano assolutamente la risposta prestabilita (le persone che spediscono messaggi a una lunga mailing list durante l'estate probabilmente non vogliono ricevere centinaia di risposte che informino sui piani delle vacanze di ciascuno).

L'autore di recente è incorso in una forma estrema del processo di trasferimento quando ha inviato un messaggio di posta a una persona che dice di ricevere 600 messaggi al giorno. La sua identità non verrà qui rivelata, per paura che anche metà dei lettori di questo libro gli invino dei messaggi. Chiamiamolo Giovanni.

Giovanni ha installato un robot per la posta elettronica che controlla ogni messaggio in arrivo per vedere se arriva da un nuovo corrispondente. Se sì, invia una risposta prestabilita in cui spiega che Giovanni non può più leggere personalmente tutta la sua posta. Ha invece prodotto un documento FAQ (Frequently Asked Questions – domande richieste più frequentemente) personale che risponde a molte domande che gli vengono di solito rivolte. Normalmente sono i gruppi di news ad avere documenti FAQ, non le singole persone.

Il documento FAQ di Giovanni fornisce il suo indirizzo, fax e numeri di telefono e spiega come contattare la sua azienda. Spiega come poter parlare con lui e descrive dove trovare i

suoi articoli e altri documenti. Inoltre fornisce dei puntatori al software da lui scritto, a una conferenza che sta organizzando, a uno standard di cui è curatore e così via. Questo atteggiamento sarà anche giustificato, ma certo un documento FAQ personale è lo status symbol più originale.

7.4.5 Posta elettronica privata

Quando un messaggio di posta elettronica viene trasmesso tra due nodi distanti, di solito transita lungo la strada per dozzine di macchine. Ciascuna di esse può leggere e memorizzare il messaggio per un uso futuro. La segretezza non esiste, al contrario di ciò che molti pensano (Weisband, Reinig, 1995). Ciò nonostante, a molte persone piacerebbe poter inviare messaggi che siano letti dal mittente desiderato e da nessun altro: nè il loro capo, nè spie, e neppure dal governo. Questo desiderio ha stimolato molte persone e gruppi ad applicare alla posta elettronica i principi di crittografia che abbiamo studiato precedentemente in modo da produrre una posta elettronica sicura. Nei paragrafi che seguono analizzeremo due sistemi di posta elettronica sicura che sono largamente usati, PGP e PEM. Per ulteriori informazioni, si vedano: Kaufman *et al.* (1995); Schneier (1995); Stallings (1995b, 1995c).

PGP – Pretty Good Privacy

Il nostro primo esempio, **PGP (Pretty Good Privacy)**, è essenzialmente un'idea di una sola persona, Phil Zimmermann (Zimmermann, 1995a, 1995b). È un pacchetto completo per la sicurezza della posta elettronica che garantisce segretezza, autenticazione, firma digitale e compressione, tutto in un formato semplice da utilizzare. Inoltre, l'intero pacchetto, compreso tutta la sorgente, viene distribuito gratuitamente via Internet, i bulletin boards e le reti commerciali. Grazie alla sua qualità, al prezzo (0) e alla semplice disponibilità su piattaforme MS-DOS/Windows, UNIX e Macintosh è attualmente molto utilizzato. È inoltre disponibile una versione commerciale per quelle aziende che necessitano di assistenza.

È stato anche coinvolto in varie controversie (Levy, 1993). Dato che è disponibile gratuitamente sull'Internet, il governo degli Stati Uniti ha dichiarato che la possibilità da parte degli stranieri di ottenerlo costituisce una violazione delle leggi riguardanti l'esportazione di armi. Le ultime versioni sono state prodotte fuori dagli Stati Uniti in modo da aggirare questa restrizione. Un altro problema riguarda una presunta trasgressione del brevetto dell'RSA, ma questo problema è stato risolto con le versioni a partire dalla 2.6.

Ciò nonostante, non a tutti piace l'idea che le persone siano in grado di mantenere nei loro confronti dei segreti, perciò i nemici del PGP sono sempre in agguato nell'ombra, pronti ad attaccare. Di conseguenza, il motto di Zimmermann è "Se la segretezza è fuorilegge, solamente i fuorilegge avranno segretezza".

Il PGP usa intenzionalmente gli algoritmi crittografici esistenti piuttosto che inventarne di nuovi. È in gran parte basato sugli algoritmi RSA, IDEA ed MD5, tutti algoritmi che hanno retto a profonde analisi e che non erano stati progettati o influenzati da alcuna agenzia governativa che tentasse di indebolirli. Per persone che tendenzialmente non hanno fiducia nel governo, questa è una proprietà positiva.

Il PGP supporta la compressione del testo, la segretezza e le firme digitali e inoltre fornisce degli strumenti flessibili per la gestione della chiave. Per capire come lavora il

PGP, consideriamo l'esempio di figura 7-49. Qui Alice vuole inviare un messaggio in chiaro firmato, P , a Bob in un modo sicuro. Sia Alice che Bob hanno due chiavi RSA, una privata (D_x) e viceversa una pubblica (E_x). Supponiamo che l'una conosca la chiave pubblica dell'altro; vedremo in seguito la gestione della chiave.

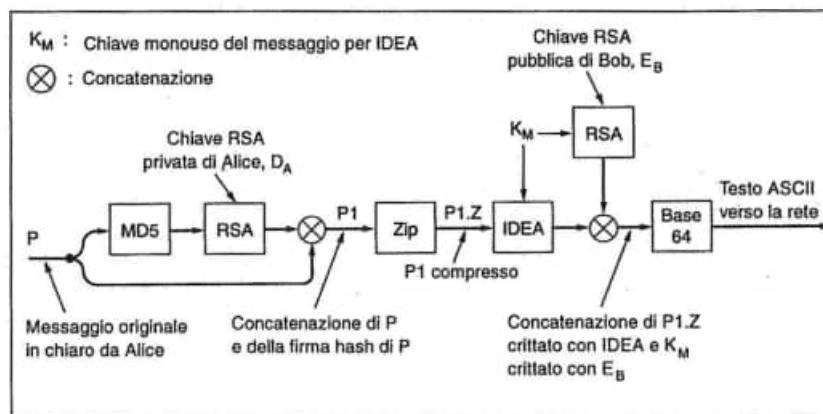


Fig. 7-49. Il PGP all'opera per inviare un messaggio.

Alice inizia invocando il programma PGP sul proprio computer. Il PGP prima sintetizza il suo messaggio, P , utilizzando l'MD5 e quindi cifra la sintesi risultante mediante la sua chiave RSA privata, D_A . Nel momento in cui Bob ottiene il messaggio, può decifrarlo con la chiave pubblica di Alice e verificare che la sintesi sia corretta. Anche se qualcun altro (ad es., Trudy) riuscisse a ottenere a questo punto la sintesi e a decifrarla mediante la nota chiave pubblica di Alice, la forza dell'MD5 garantirà che non sia computazionalmente possibile produrre un altro messaggio con la stessa sintesi MD5.

La sintesi cifrata e il messaggio originale sono adesso concatenati in un singolo messaggio, $P1$, e compressi mediante il programma ZIP, che utilizza l'algoritmo Ziv-Lempel (Ziv, Lempel, 1977). Chiamiamo $P1.Z$ l'uscita di questo passo.

A questo punto, il PGP richiede ad Alice di inserire un ingresso casuale. Sia il contenuto che la velocità di immissione vengono utilizzati per generare una chiave IDEA a 128 bit del messaggio, K_M (detta chiave di sessione nella letteratura PGP, ma in realtà è una definizione poco appropriata perché non c'è alcuna sessione). La chiave K_M viene adesso usata per cifrare $P1.Z$ con l'algoritmo IDEA in modalità feedback del cifrario. Inoltre, la K_M viene cifrata con la chiave pubblica di Bob, E_B . Queste due componenti vengono quindi concatenate e convertite in base64, come discusso nel paragrafo su MIME. Il messaggio risultante quindi contiene solo lettere, cifre e i simboli +, /, =, e questo significa che può essere inserito in un corpo RFC 822 e aspettarsi che arrivi integro.

Quando Bob riceve il messaggio, inverte la codifica base64 e decifra la chiave IDEA utilizzando la sua chiave RSA privata. Con questa chiave, decifra il messaggio per ottenere $P1.Z$. Dopo averlo decompresso, Bob separa il testo in chiaro dalla sintesi cifrata e decifra la sintesi

7.4 La posta elettronica

utilizzando la chiave pubblica di Alice. Se la sintesi in chiaro corrisponde alla sua computazione MD5, capisce che P è il messaggio corretto e che quindi proviene da Alice.

È bene notare che l'algoritmo RSA viene usato solamente in due punti: per cifrare la sintesi MD5 di 128 bit e per cifrare la chiave IDEA a 128 bit. Anche se l'RSA è lento, questo deve cifrare solo 256 bit, non un grosso volume di dati. Inoltre, tutti i 256 bit in chiaro sono estremamente casuali, così Trudy dovrà lavorare moltissimo per determinare se una supposta chiave sia corretta. Il vero lavoro di cifratura è fatto dall'IDEA, che è diversi ordini di grandezza più veloce dell'RSA. Così il PGP fornisce sicurezza, compressione e una firma digitale e lo fa in un modo molto più efficiente dello schema illustrato in figura 7-23.

Il PGP supporta tre lunghezze per la chiave RSA. La selezione di quella più appropriata dipende dall'utente. Le lunghezze sono

1. Casuale (384 bit): ai nostri giorni può essere violata da chiunque abbia un grosso budget.
2. Commerciale (512 bit): può essere violata da organizzazioni il cui nome sia composto da tre lettere (ad es. CIA, KGB).
3. Militare (1024): non è violabile da nessuno sulla terra.

Ci sono state alcune discussioni su una quarta categoria: aliena (2048 bit), che non può essere violata da nessuno sulla terra e da niente nell'universo, ma per il momento non è ancora stata adottata. Dato che l'RSA è usato solamente per due piccole computazioni, probabilmente tutti dovrebbero usare sempre chiavi con lunghezze militari, fatta eccezione probabilmente per chi possiede un vecchio personal XT.

La figura 7-50 mostra il formato di un messaggio PGP. Il messaggio ha tre parti, contenenti la chiave IDEA, la firma e il messaggio, rispettivamente. La parte della chiave contiene non solamente la chiave, ma anche un identificatore di chiave, dato che gli utenti possono avere più chiavi pubbliche.

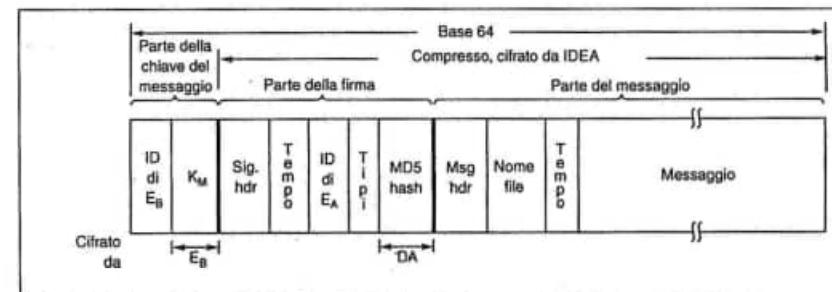


Fig. 7-50 Un messaggio PGP.

La parte della firma contiene un'intestazione, che qui non ci interessa. L'intestazione è seguita da una marca di tempo, l'identificatore per la chiave pubblica del mittente che può essere usato per decifrare la mistura della firma, un certo tipo di informazione che

identifichi l'algoritmo usato (per permettere di usare l'MD6 e l'RSA2 quando verranno inventati) e la stessa mistura cifrata.

La parte del messaggio inoltre contiene un'intestazione, il nome di default del file da usare se il destinatario memorizza il file sul disco, una marca di tempo di creazione del messaggio e, ultimo, il messaggio stesso.

La gestione della chiave ha richiesto una grande attenzione nel PGP in quanto è il tallone d'Achille di tutti i sistemi di sicurezza. Ciascun utente mantiene localmente due strutture dati: un anello di chiavi private e un anello di chiavi pubbliche. L'**anello di chiavi private** contiene una o più coppie personali di chiavi privata-pubblica. La ragione di consentire coppie multiple per utente è per permettere agli utenti di cambiare periodicamente la loro chiave pubblica o quando si teme che una sia compromessa, senza invalidare i messaggi in preparazione o in transito. Ogni coppia ha un identificatore associato, in modo che il mittente di un messaggio possa dire al destinatario quale chiave pubblica sia stata usata per la cifratura. Gli identificatori del messaggio corrispondono ai 64 bit di ordine più basso della chiave pubblica. Gli utenti hanno la responsabilità di evitare conflitti negli identificatori delle loro chiavi pubbliche. Le chiavi private su disco vengono cifrate mediante una speciale password (arbitrariamente lunga) per proteggerle da attacchi di spie. L'**anello di chiavi pubbliche** contiene le chiavi pubbliche di coloro che corrispondono con l'utente. Questi devono cifrare le chiavi del messaggio associate a ogni messaggio. Ogni elemento dell'anello delle chiavi pubbliche contiene non solamente la chiave pubblica, ma anche il suo identificatore di 64 bit e una indicazione di quanto fortemente l'utente crede nella chiave.

Il problema da affrontare qui è il seguente. Supponiamo che le chiavi pubbliche vengano mantenute su un bulletin board. Un modo col quale Trudy può leggere la posta segreta di Bob è attaccare il bulletin board e sostituire la chiave pubblica di Bob con una da lei scelta. Quando in seguito Alice recupererà la chiave che si dice appartenente a Bob, Trudy potrà effettuare un attacco della banda del buco a Bob.

Per prevenire un attacco di questo tipo, o almeno per minimizzare le sue conseguenze, Alice ha bisogno di sapere quanto fidarsi dell'elemento chiamato "chiave di Bob" appartenente al suo anello di chiavi pubbliche. Se lei sa che è stato lo stesso Bob a inviarle un dischetto contenente la chiave, è in grado di avere quel valore con la massima sicurezza. Al contrario, in pratica le persone di solito ricevono le chiavi pubbliche richiedendole a un server per le chiavi di fiducia, un certo numero dei quali è già operativo su Internet. Quando un server per le chiavi riceve una richiesta per una chiave pubblica, genera una risposta che contiene la chiave pubblica, una marca di tempo e la data di scadenza della chiave. Quindi tratta questa risposta con l'algoritmo MD5 e firma la risposta con la propria chiave privata in modo che la parte richiedente possa verificare chi le ha inviato tale risposta. Sta all'utente assegnare un valore di affidabilità alle chiavi mantenute dall'amministratore del sistema locale, dalla società dei telefoni, dall'ACM, dall'Associazione dei Bar, dal governo o da chiunque altro decida di entrare nel giro d'affari della gestione delle chiavi.

PEM – Privacy Enhanced Mail

Al contrario del PGP, che era inizialmente il lavoro di una sola persona, il nostro secondo esempio, il **PEM (Privacy Enhanced Mail)** è uno standard Internet ufficiale ed è de-

scritto in quattro documenti RFC: dall'RFC 1421 all'RFC 1424. Molto brevemente, il PEM copre la stessa area del PGP: segretezza e autenticazione per i sistemi di posta elettronica basati sullo RFC 822. Ciò nonostante, presenta alcune differenze rispetto al PGP per quanto riguarda sia l'approccio che la tecnologia.

In seguito descrivremo il PEM e quindi lo confronteremo con il PGP. Per ulteriori informazioni sul PEM, si veda Kent (1993).

I messaggi inviati mediante il PEM vengono dapprima convertiti in una forma canonica in modo che abbiano tutti le stesse convenzioni per quanto riguarda gli spazi bianchi (ad es., tabulazioni e interlinea) e l'uso del ritorno carrello e dell'"a capo". Questa trasformazione è eseguita eliminando gli effetti degli agenti di trasferimento messaggi i quali modificano i messaggi in modi differenti.

Senza una forma canonica, queste modifiche possono influenzare le procedure di mescolamento eseguite a partire dai messaggi alle loro destinazioni.

A questo punto, i messaggi vengono trattati con l'MD2 o l'MD5. La scelta non è opzionale, come nel PGP. Quindi viene cifrata la concatenazione della sintesi e del messaggio mediante il DES. Alla luce della nota debolezza della chiave a 56 bit, questa scelta è certamente sospetta. Il messaggio cifrato viene quindi codificato mediante la codifica base64 e trasmesso al destinatario. Le mailing list sono esplicitamente permesse.

Come nel PGP, ogni messaggio viene cifrato con una chiave valida una sola volta, che viene inclusa assieme al messaggio. È possibile proteggere la chiave con l'RSA oppure con un triplo DES mediante EDE. In pratica, tutti utilizzano l'RSA, per cui ci concentreremo su questo. Infatti il PEM non indica come effettuare la gestione della chiave mediante il DES. La gestione della chiave è più strutturata che nel PGP. Le chiavi sono certificate da **autorità di certificazione** in un formato di certificazione con un nome di utente, una chiave pubblica e una data di scadenza della chiave. Ogni certificazione ha un numero di serie unico per identificarla. È inclusa una sintesi MD5 firmata da una chiave privata autorità di certificazione. Queste certificazioni sono conformi alla raccomandazione ITU X.509 per i certificati delle chiavi pubbliche, e come tali, utilizzano i nomi X.400 come l'esempio di Tom Smith dato in precedenza.

Il PGP ha uno schema simile (senza fare uso di X.509), ma ha un problema: l'utente può credere alle autorità di certificazione? Il PEM risolve questo problema certificando le autorità di certificazione mediante quelle che vengono chiamate **PCA (Policy Certification Authority)**. Queste, a loro volta, sono certificate dall'IPRA (**Internet Policy Registration Authority**), l'arbitro più recente su ciò che è giusto, e su ciò che non lo è.

Ogni PCA deve definire una politica ufficiale di registrazione e registrarla con lo IPRA. Questa procedura viene quindi siglata dall'IPRA e resa pubblica. Ad esempio, un PCA può insistere nel volere che gli utenti sotto la propria giurisdizione si presentino di persona con un certificato di nascita, la patente di guida, il passaporto, due carte di credito, una assicurazione sulla vita e una chiave pubblica su dischetto. Un altro PCA accetta le registrazioni via posta elettronica da stranieri. Rendendo pubblica la politica di decisione, gli utenti hanno una certa base per decidere a quale autorità affidarsi. Non viene presa alcuna precauzione per controllare che le politiche siano effettivamente rispettate.

Sono previsti tre diversi tipi di autorità di certificazione. Una è di tipo organizzativo, che invia i certificati ai suoi impiegati. Molte aziende lo faranno in proprio. Una residente

che opererà per i privati cittadini, molto simile a fornitori di servizi Internet che forniranno il servizio a chiunque sia disposto a pagarlo. Infine, è prevista la modalità per le registrazioni anonime. Con tutte queste autorità di certificazione in giro, la necessità di guidare il gregge delle PCA dovrebbe essere chiara.

Anche se gerarchico e burocratico, questo schema ha il vantaggio rispetto al PGP di rendere potenzialmente praticabile la revoca della certificazione. La revoca è necessaria quando un utente desideri cambiare la propria chiave pubblica, ad esempio perché è stata compromessa o perché la propria autorità di certificazione è stata rapinata. La revoca viene effettuata dall'utente informando la sua autorità di certificazione che la sua chiave pubblica è stata compromessa (o anche viceversa). Allora l'autorità di certificazione aggiunge il numero seriale del certificato divenuto adesso non valido a una lista di certificati revocati, lo sigla e distribuisce la lista in lungo e in largo.

Chiunque voglia inviare un messaggio PEM a un utente deve dapprima controllare la lista delle revoche più recente per vedere se la chiave pubblica in suo possesso sia ancora valida. Questo processo è simile a quello che fa un commerciante controllando la lista delle carte di credito rubate prima di accettarne una. Chi critica il PEM asserisce che controllare tutte le volte sia troppo faticoso e che nessuno lo farebbe. I sostenitori asseriscono che i computer non si annoiano; se sono programmati per effettuare ogni volta la verifica, la effettueranno ogni volta.

La figura 7-51 elenca alcune delle somiglianze e delle differenze tra PGP e PEM. Molti di questi aspetti sono già stati visti, ma è bene commentarne altri. L'autenticazione sembra più importante nel PEM che nel PGP dato che è obbligatoria nel PEM e opzionale nel PGP. Inoltre il PEM porta l'informazione di autenticazione al di fuori del pacchetto cifrato, il che significa che la rete può verificare l'origine di ogni messaggio. Di conseguenza, chi origlia può sapere chi è che spedisce, anche se non è in grado di leggere i messaggi.

A parte tutte queste differenze tecniche, vi è una sorprendente differenza culturale. Il PGP, che non è uno standard Internet ufficiale, ha la cultura Internet. Il PEM, che è uno standard Internet ufficiale, non ce l'ha. Il PGP era basato su ciò che Dave Clark chiama "generico consenso e codice funzionante". Qualcuno (Zimmermann) ha pensato a una soluzione di un ben noto problema, implementandola bene, e lasciando libero chiunque di utilizzare il codice sorgente. Il PEM è iniziato come la quarta parte di uno standard ufficiale, utilizzando ASN.1 per definire i formati, X.400 per definire i nomi, e X.509 per definire i certificati. Utilizza una rigida gerarchia organizzativa a tre livelli per i diversi tipi di autorità di certificazione, completata da una procedura ufficiale per la politica di certificazione e il requisito che tutti si fidino dell'IPRA. Le implementazioni sono arrivate in seguito e sono molto indietro rispetto alla qualità, quantità e disponibilità su più piattaforme del PGP. In breve, il PGP sembra proprio un tipico pacchetto per Internet, mentre il PEM mostra parecchie delle caratteristiche di uno standard OSI, odiato dagli abitanti di Internet e amato dalle PTT. Immaginate voi.

7.5 Le news di USENET

Una delle applicazioni più popolari di Internet è il sistema mondiale di gruppi di interesse (newsgroup) chiamato **notizie di rete** (news). Spesso il servizio di notizie è anche chia-

mato **USENET**, un nome che ricorda la rete fisica di sistemi UNIX su cui una volta circolava il traffico di USENET utilizzando un programma chiamato **uucp**. Ai nostri giorni, la maggior parte del traffico circola su Internet, ma USENET e Internet non sono la stessa cosa. Alcuni nodi Internet non ricevono le news, mentre altri nodi ricevono le news senza essere in Internet.

Nei prossimi paragrafi descriveremo USENET. Analizzeremo il servizio dapprima dal punto di vista dell'utente. Quindi descriveremo come è realizzato.

Elemento	PGP	PEM
Ammette la cifratura?	Sì	Sì
Ammette l'autenticazione?	Sì	Sì
Impedisce il ripudio?	Sì	Sì
Ammette la compressione?	Sì	No
Ammette la forma canonica?	No	Sì
Ammette le mailing list?	No	Sì
Ammette la cifratura?	Sì	Sì
Utilizza la codifica base64?	Sì	Sì
Algoritmo di cifratura dei dati usato	IDEA	DES
Lunghezza della chiave per cifrare i dati (bit)	128	56
Attuale algoritmo per la gestione delle chiavi	RSA	RSA o DES
Lunghezza della chiave per la gestione delle chiavi (bit)	384/512/1024	Variabile
Spazio dei nomi d'utente	definito dall'utente	X.400
Conforme a X.509?	No	Sì
Devi fidarti di chiunque?	No	Sì (IPRA)
Certificazione della chiave	Ad hoc	gerarchia IPRA/PCA/CA
Revoca della chiave	Azzardata	Migliore
Chi origlia può leggere i messaggi?	No	No
Chi origlia può leggere le firme?	No	Sì
Standard Internet?	No	Sì
Progettato da un	Piccolo gruppo	Comitato di standard

Fig. 7-51 Un confronto tra PGP e PEM.

7.5.1 USENET per l'utente

Un gruppo di notizie è un luogo di discussione su un particolare argomento. Le persone interessate a un argomento possono "isciversi" a un gruppo. I sottoscrittori possono usare

un particolare tipo di interfaccia agente, un lettore di news, per leggere tutti gli articoli (messaggi) affissi al gruppo. Ogni utente oltre a leggere può anche scrivere articoli nel gruppo. Ogni articolo affisso a un gruppo viene automaticamente inviato a tutti i sottoscrittori, ovunque si trovino nel mondo. La spedizione tipicamente richiede da qualche secondo a qualche ora: dipende dalla distanza fra mittente e destinatario. In pratica, un gruppo è simile a una lista di indirizzi di posta, ma è realizzato in maniera differente. Possiamo anche pensarlo come una specie di comunicazione di gruppo (multicast) ad alto livello.

I gruppi sono così tanti (probabilmente più di 10.000) da dover essere organizzati in una gerarchia in modo che tutto sia più semplice. La figura 7-52 mostra il livello più alto della gerarchia "ufficiale". Esistono anche altre gerarchie, ma queste sono di solito intese per un utilizzo regionale oppure sono in lingue diverse dall'inglese. Tra le gerarchie alternative, *alt* è speciale, perché sta ai gruppi ufficiali come un mercatino delle pulci sta a un grande magazzino. È un'accozzaglia caotica e irregolare di gruppi di news su tutti gli argomenti, alcuni dei quali sono molto popolari, e molti dei quali sono di interesse mondiale.

Nome	Argomenti
Comp	Computer
Sci	Le scienze fisiche e ingegneristiche
Humanities	Letteratura e studi umanistici
News	Discussione su USENET stessa
Rec	Attività ricreative
Misc	Tutto ciò che non appartiene a qualche altro gruppo
Soc	Socializzazione e argomenti sociali
Talk	Polemiche
Alt	Gerarchia alternativa che interessa virtualmente tutto

Fig. 7-52 Le gerarchie USENET ordinate con un criterio decrescente da segnale a rumore.

I gruppi *comp* sono quelli originali di USENET. Questi gruppi sono frequentati da scienziati, professionisti informatici e persone che hanno l'informatica come hobby. Ognuno caratterizza una discussione tecnica su un argomento concernente l'hardware o il software dei sistemi di elaborazione informatici.

I gruppi *sci* e *humanities* sono frequentati da scienziati, studenti e persone interessate a un particolare argomento di cultura (fisica, chimica, biologia, letteratura e così via). Non sorprende che la gerarchia *sci* sia molto più vasta della gerarchia *humanities* perché il concetto di comunicazione elettronica istantanea tra colleghi di tutte le parti del mondo è qualcosa che piace alla maggior parte degli scienziati, mentre la maggior parte degli umanisti è alquanto scettica. C.P. Snow aveva ragione.

La gerarchia *news* viene usata per discutere e gestire il sistema stesso. I gestori di sistemi qui trovano aiuto e inoltre qui avvengono le discussioni sulla creazione di nuovi gruppi di news.

Le gerarchie viste fino a ora hanno un tono professionale, in qualche modo accademico.

Questo cambia con la *rec* che riguarda le attività ricreative e gli hobby. Ciò nonostante, molte persone che pubblicano qui sono molto ben informate sui loro interessi reciproci. Scendendo, arriviamo a *soc*, che ha diversi gruppi di news che riguardano politica, sesso, religione, culture di varie nazionalità e genealogia. *Talk* riguarda argomenti controversi ed è frequentata da persone forti nelle opinioni, deboli nei fatti. *Alt* è una gerarchia completamente alternativa che opera sotto le proprie regole.

Ognuna delle categorie elencate in figura 7-52 è suddivisa in sottocategorie, ricorsivamente. Ad esempio, *rec.sport* si occupa di sport, *rec.sport.basketball* si occupa di pallacanestro, e *rec.sport.basketball.women* si occupa di pallacanestro femminile. La figura 7-53 contiene un esempio di alcuni gruppi di ogni categoria. In molti casi, è possibile inferire con l'esistenza di altri gruppi di news semplicemente cambiando i parametri ovvi. Ad esempio, *comp.lang.C* riguarda il linguaggio di programmazione C, ma il *.C* può essere sostituito semplicemente da qualsiasi altro linguaggio di programmazione per generare il nome del gruppo di news corrispondente.

Esistono parecchi programmi lettori di news. Come i lettori di posta elettronica, alcuni si basano sull'uso della tastiera, altri sull'uso del mouse. In quasi tutti i casi, quando il lettore di news viene avviato, questo legge un file per vedere a quali gruppi di news l'utente è abbonato. In genere a questo punto visualizza un sommario a linea per ogni articolo non ancora letto nel primo gruppo di news e attende che l'utente ne selezioni uno o più di uno da leggere. Gli articoli selezionati vengono quindi visualizzati uno alla volta. Dopo essere stato letto, un articolo può venire scaricato, salvato, stampato e così via.

I lettori di news inoltre permettono agli utenti di sottoscriversi e cancellare le sottoscrizioni ai vari gruppi di news. Modificare una sottoscrizione significa semplicemente editare il file locale che elenca i gruppi di news ai quali l'utente è abbonato. Per fare un'analogia, abbonarsi a un gruppo di news è come guardare un programma alla televisione. Se vuoi guardare lo stesso programma tutte le settimane, semplicemente lo fai. Non devi prima registrarti presso una certa autorità locale.

I lettori di news inoltre si occupano della pubblicazione. L'utente compone un articolo e quindi invia un comando o fa click su un'icona per inviare l'articolo al suo posto. Entro un giorno, questo raggiungerà praticamente tutti gli abbonati del mondo di quel gruppo di news nel quale l'articolo è stato affisso. È possibile effettuare una pubblicazione multipla (crosspost) di un articolo, cioè inviarlo a più gruppi di news con un solo comando. È inoltre possibile restringere l'area geografica di distribuzione di una pubblicazione. L'annuncio di un seminario al martedì a Stanford probabilmente non sarà di grande interesse, diciamo, a Hong Kong, e quindi si può limitare la pubblicazione alla California.

Il fenomeno USENET non ha precedenti. Non sarebbe mai stato possibile prima di USENET per migliaia di persone sconosciute l'una all'altra discutere a livello mondiale su una tale quantità di argomenti. Ad esempio, adesso è possibile a chiunque abbia un problema di renderlo pubblico tramite la rete. Il giorno dopo, quella stessa persona potrà avere 18 soluzioni, e con un po' di fortuna, solo 17 di queste saranno sbagliate.

Sfortunatamente, alcune persone usano questo loro nuovo potere per comunicare in maniera irresponsabile. Quando qualcuno pubblica un messaggio del tipo: "Persone come te dovrebbero essere uccise" accende una miccia e scatena un torrente di messaggi indignati, detto **flamewar**, che di solito fa seguito.

Questa situazione può essere evitata grazie a due comportamenti, uno individuale e uno collettivo. I singoli utenti possono installare un **killfile**, il quale specifichi che gli articoli con un certo argomento o provenienti da una certa persona vanno cancellati al loro arrivo, prima di essere visualizzati. Molti programmi lettori di news inoltre permettono anche di cancellare automaticamente un'intera discussione fatta di più articoli. Questa caratteristica è utile quando una discussione sembra continuare all'infinito.

Nome	Argomenti
Comp.ai	Intelligenza artificiale
Comp.databases	Progetto e realizzazione di sistemi di basi di dati
Comp.lang.c	Il linguaggio di programmazione C
Comp.os.minix	Il sistema operativo MINIX di Tanenbaum
Comp.os.ms-windows.video	Hardware e software di video per Windows
Sci.bio.entomology.lepidoptera	Ricerche su farfalle e mosche
Sci.geo.earthquakes	Geologia
Sci.med.orthopedics	Chirurgia ortopedica
Humanities.it.authors.Shakespeare	Commedie e poesie di Shakespeare
News.groups	Nuovi potenziali gruppi
News.lists	Elenchi di gruppi USENET
Rec.arts.poems	Poesia libera
Rec.food.chocolate	Gnam gnam
Rec.humor.funny	Sapete la barzelletta sul contadino che...
Rec.music.folk	Gente che discute di musica folk
Misc.jobs.offered	Annunci per offerte di lavoro
Misc.health.diabetes	Il vivere quotidiano di un diabetico
Soc.culture.estonia	Vita e cultura in Estonia
Soc.singles	Persone sole e loro interessi
Soc.couples	Ex membri di soc.singles
Talk.abortion	Nessun segnale
Talk.rumors	Da qui vengono tutti i pettigolezzi
Alt.alien.visitors	Luogo in cui rendere note le incursioni di UFO
Alt.bermuda.triangle	Se lo leggete, sparirete misteriosamente
Alt.sex.voyeurism	Date un'occhiata e vedete da voi
Alt.tv.simpsons	Bart e gli altri

Fig. 7-53 Una piccola selezione di gruppi di news.

Se un numero sufficiente di iscritti a un gruppo è stanco dell'inquinamento che regna nel gruppo stesso, è possibile proporre che il gruppo diventi moderato. In un gruppo moderato

solo una persona può scrivere articoli. Tutto ciò che si vuole inserire in un gruppo moderato va spedito al moderatore, che pubblica ciò che è accettabile ed elimina ciò che non lo è. Alcuni argomenti hanno sia un gruppo di news moderato che uno non moderato. Dato che ogni giorno sono migliaia le persone che si iscrivono a USENET per la prima volta, ci sono delle domande tipiche da principiante che vengono formulate in continuazione. Per snellire il traffico di rete, molti gruppi hanno costruito un apposito documento **FAQ (Frequently Asked Questions – domande frequenti)** per rispondere a tutte le domande che fanno i principianti. Alcuni di tali documenti sono molto specializzati e sono costituiti da più di 100 pagine. Chi li gestisce di solito li aggiorna una o due volte al mese. USENET è ricca di espressioni in gergo come BTW (By The Way – a proposito), ROFL (Rolling On the Floor Laughing – rotolarsi dalle risate) e IMHO (In My Humble Opinion – a mio modesto avviso). Molte persone usano anche delle composizioni di simboli ASCII dette **smiley** o **emoticon** che viste dopo averle fatte ruotare di 90° in senso orario, appaiono propriamente come dei disegni con un particolare significato. Alcune tra le più interessanti sono riportate in figura 7-54. Sanderson e Dougherty (1993) hanno scritto un minilibro che descrive e spiega più di 650 smiley.

Smiley	Significato
:=)	Sono felice
:-(Sono triste/arrabbiato
: -	Sono apatico
:;-)	Sto facendo l'occhiolino
:-(O)	Sto gridando
:-(*)	Sto vomitando
= :-)	Abramo Lincoln
=):-)	Zio Sam
*<:-)	Babbo Natale
<:-)	Somaro
(:-	Australiano
:)-X	Uomo col fiocchino
:+)	Naso grosso
:)-))	Doppio mento
:)-()	Baffi
#:-)	capelli arruffati
8-)	Porta gli occhiali
C:-)	Gran cervello

Fig. 7-54 Alcuni smiley.

Molte persone firmano col proprio nome quando desiderano pubblicare un loro articolo, ma vi sono particolari settori dove si desidera mantenere l'anonimato. Questo desiderio ha portato alla creazione di **remailer anonimi**, cioè server che accettano messaggi di posta elettronica (che contengono ciò che va pubblicato) e modificano i campi *From:*, *Sender:* e *Reply-To:* in modo che puntino al remailer invece che al mittente. Alcuni remailer assegnano un numero a ogni utente e rispediscono i messaggi elettronici indirizzati a questi numeri, in modo che gli interessati possano inviare le risposte in forma anonima come "Donna nubile bianca età 25 cerca uomo bianco scapolo o divorziato età 20-30". È dubbio che tali remailer riescano a non rivelare i loro segreti quando la polizia locale s'incuriosisce sull'identità di un utente (Barlow, 1995).

Con l'aumento del numero di utenti che si iscrivono a USENET, c'è una costante richiesta di nuovi gruppi più specializzati. Di conseguenza, è stata stabilita una procedura per la creazione di nuovi gruppi. Supponiamo che a qualcuno piacciono gli scarafaggi e voglia parlare con altri ammiratori degli scarafaggi. Invia un messaggio a *news.group* specificando un nome per il gruppo proposto, diciamo *rec.animals.wildlife.cockroaches*, e descrivendo perché sia così importante (gli scarafaggi sono affascinanti; ne esistono 3500 specie; sono rossi, gialli, verdi, marroni e neri; apparvero sulla terra molto tempo prima dei primi dinosauri; furono probabilmente i primi animali che volarono, e così via). Inoltre specifica se vuole o no che sia moderato.

La discussione quindi è aperta. Quando il gruppo viene istituito, si apre una votazione via posta elettronica. I voti vengono pubblicati, dopo aver identificato chi ha votato e in quale modo (per evitare frodi). Se i sì superano i no per più di 2:1 e ci sono almeno 100 sì in più dei no, il moderatore di *news.group* pubblica un messaggio di accettazione ufficiale del nuovo gruppo. Questo messaggio è il segnale per tutti gli amministratori di sistema del mondo che il nuovo gruppo è stato benedetto "colà dove si puote" e adesso è ufficiale.

La creazione di nuovi gruppi è meno formale per la gerarchia *alt* e questa è in realtà la sua ragion d'essere. Alcuni dei suoi gruppi sono così vicini al confine di ciò che è legalmente e moralmente accettabile che non verrebbero mai ammessi a una votazione pubblica. In effetti, le persone che li sostengono scavalcano semplicemente la normale procedura e creano la propria gerarchia. Nonostante ciò, gran parte della gerarchia *alt* è abbastanza convenzionale.

7.5.2 Come è realizzata USENET

Alcuni dei gruppi più piccoli sono realizzati come mailing list. Per pubblicare un articolo lo si manda all'indirizzo della lista, che si occupa di inviare delle copie a ogni indirizzo in lista.

D'altra parte, se metà degli studenti di una grossa università si iscrivessero ad *alt.sex*, il server locale crollerebbe sotto il peso della posta in arrivo. Di conseguenza, USENET di solito non viene realizzata con le mailing list. Invece ogni sito (campus, azienda o fornitore di servizi Internet) memorizza la posta in arrivo in una singola directory, chiamiamola *news*, con la sottodirectory per *comp*, *sci* ecc. Queste a loro volta hanno delle sottodirectory tipo *news/comp/os/minix*. Tutta la posta che arriva viene depositata nel directory opportuna. I programmi lettori di news recuperano gli articoli quando ne hanno bisogno.

Con questa organizzazione ogni sito ha bisogno di una sola copia di ogni articolo, indipendentemente dal numero di persone iscritte al gruppo interessato. Dopo alcuni giorni, gli articoli scadono e vengono rimossi dal disco.

Per aderire a USENET, un sito deve ottenere un **newsfeed** (alimentatore di news) da un altro sito su USENET. Si può pensare all'insieme di tutti i siti che ricevono le notizie di rete come ai nodi di un grafo diretto. Le linee di trasmissione che connettono coppie di nodi formano gli archi del grafo. Questo grafo è USENET. Si noti che essere su Internet non è né necessario né sufficiente per essere in USENET.

Periodicamente, ogni sito che vuole le news può interrogare il proprio o i propri newsfeed per sapere se sono arrivate nuove notizie dal momento della precedente interrogazione. Se sì, tali notizie vengono raccolte e memorizzate nell'opportuna sottodirectory di *news*. In questo modo gli articoli si diffondono sulla rete. In modo equivalente può essere il newsfeed, invece del ricevente, a prendere l'iniziativa e stabilire il contatto quando ci sono abbastanza articoli nuovi. All'inizio molti siti interrogavano i loro newsfeed, ma adesso la maggior parte usa l'altro modo.

Non tutti i siti ricevono tutti i gruppi, per diversi motivi. Primo, l'ammontare di nuovi articoli supera i 500 MB al giorno e questo dato cresce rapidamente. Memorizzare tutto richiederebbe una enorme quantità di spazio disco. Secondo, il tempo di trasmissione e i costi sono notevoli. A 28,8 Kbps, ci vogliono 39 h e una linea telefonica dedicata per trasmettere le notizie di 24 h. Anche a 56 kbps, ricevere tutto richiederebbe una linea dedicata per almeno 20 h al giorno. Infatti, il volume complessivo è adesso diventato così grande che sono stati creati i newsfeed via satellite.

Terzo, non tutti i siti sono interessati a ogni argomento. Ad esempio, è inverosimile che i lavoratori delle aziende in Finlandia vogliano leggere *rec.arts.manga* (sui fumetti giapponesi). Infine, alcuni gruppi sono un po' troppo pericolosi per i gusti di molti amministratori di sistema, che quindi li bandiscono, nonostante un interesse locale considerevole. Nel dicembre 1995, la rete mondiale CompuServe smise (temporaneamente) di trasportare tutti i gruppi con la parola "sex" nel nome perché qualche autorità in Germania aveva pensato che questo potesse essere un buon modo per combattere la pornografia. Il chiasso che ne derivò era prevedibile, e fu immediato, esteso in tutto il mondo e molto forte.

Gli articoli hanno lo stesso formato RFC 822 dei messaggi di posta elettronica, ma con l'aggiunta di qualche campo. Questa proprietà li rende facili da trasportare e compatibili con la maggior parte del software esistente per la posta elettronica. Le intestazioni delle news sono definiti nello RFC 1036. La figura 7-55 mostra un articolo come esempio.

Probabilmente è necessario spendere alcune parole sui campi dei messaggi USENET. Il campo *Path*: contiene la lista dei nodi che il messaggio ha attraversato per arrivare dalla sorgente a destinazione. A ogni salto, la macchina che rispedisce inserisce il suo nome all'inizio della lista. Questa lista fornisce il cammino inverso alla sorgente. L'uso dei punti esclamativi (che si pronunciano bang) rimanda agli indirizzi USENET, precedenti al DNS. Il campo *Newsgroup*: indica a quale gruppo di news appartiene il messaggio. Può contenere più di un nome di gruppo. Ogni messaggio che va inserito in più gruppi conterrà tutti i loro nomi. Dato che sono ammessi più nomi, è necessario il campo *Followup-To*:

```

From: Vogel@nyu.edu
Message-Id: <54731@nyu.edu>
Subject: Bird Sighting
Path: cs.vu.nl!sun4nl!EU.net!news.sprintlink.net!in2.uu.net!pc144.nyu.edu!news
Newsgroups: rec.birds
Followup-To: rec.birds
Distribution: world
Nntp-Posting-Host: nuthatch.bio.nyu.edu
References:
Organization: New York University
Lines: 4
Summary: Guarda un po' cosa ho visto

Ho appena visto uno struzzo tra la 52esima e la quinta avenue a New York.
Questa è la loro stagione migratoria? Qualcun altro lo ha visto?

Jay Vogel

```

Fig. 7-55 Un articolo di esempio.

per dire alle persone dove inviare i commenti e le reazioni in modo la discussione che ne segue vada a finire in un unico gruppo di news.

Il campo *Distribution*: indica fino a dove venga propagato l'articolo. Può contenere uno o più codici di stato o regione, il nome di uno specifico sito o rete, oppure "world".

Il campo *Nntp-Posting-Host*: è analogo al campo *Sender*: dell'RFC 822. Indica la macchina che ha pubblicato l'articolo, anche se in realtà questo è stato composto su una macchina differente (NNTP è il protocollo delle news, come descritto sotto).

Il campo *References*: indica che questo articolo è una risposta a un articolo precedente e riporta l'ID di questo articolo. È richiesto su tutti gli articoli conseguenti e proibito quando si inizia una nuova discussione.

Il campo *Organization*: si può usare per specificare da quale azienda, università o agenzia dipenda il mittente. Gli articoli che hanno riempito questo campo, spesso hanno alla fine una nota di declino di responsabilità che dice che se l'articolo non è intelligente non è colpa dell'organizzazione.

Il campo *Lines*: riporta la lunghezza del corpo del messaggio. Le linee dell'intestazione e la linea bianca che separa l'intestazione dal corpo non contano.

Le linee del campo *Subject*: riportano l'argomento della discussione. Molti lettori di news hanno un comando che permette all'utente di passare direttamente all'articolo successivo del soggetto corrente, piuttosto che leggere l'articolo arrivato dopo in sequenza. Inoltre, i comandi killfiles e kill usano questo campo per sapere cosa cancellare automaticamente.

Infine, il campo *Summary*: viene di solito usato per riassumere l'articolo di risposta. Negli articoli di risposta il campo *Subject*: contiene "Re:" seguito dal soggetto originale.

NNTP – Network News Transfer Protocol

Vediamo adesso in che modo gli articoli vengano diffusi per il mondo. L'algoritmo iniziale distribuiva semplicemente gli articoli su ogni linea di USENET. Questo ha funzionato per un po', ma l'eventualità che il volume del traffico rendesse impraticabile questo schema ha portato a definire qualcosa di meglio.

Il sostituto è stato il protocollo **NNTP** (**Network News Transfer Protocol**), definito nell'RFC 977. L'NNTP è qualcosa di simile all'SNMP, con un cliente che invia comandi in ASCII e un server che invia risposte tipo numeri decimali codificati in ASCII. Molte macchine USENET adesso utilizzano NNTP.

NNTP era stato progettato con due scopi. Il primo obiettivo era permettere agli articoli di news di propagarsi da una macchina all'altra su una connessione affidabile (ad es., TCP). Il secondo obiettivo era permettere agli utenti con un piccolo computer che non può ricevere le news di leggerle in maniera remota. Queste funzioni sono entrambe molto usate, ma noi ci concentreremo su come gli articoli vengano diffusi sulla rete mediante NNTP.

Come detto prima, sono possibili due soluzioni. Con la prima, *a richiesta* (pull), il cliente chiama uno dei suoi newsfeed e richiede i nuovi articoli. Nella seconda, *a invio* (push), il newsfeed chiama il cliente e annuncia che ci sono nuovi articoli. I comandi NNTP permettono entrambi gli approcci, così come la lettura remota.

Per acquisire gli articoli recenti, un cliente deve dapprima stabilire una connessione TCP con la porta 119 di uno dei suoi newsfeed. Al di là di questa porta c'è il demone NNTP, che o sta lì sempre in attesa dei clienti o viene creato al momento del bisogno. Dopo che la connessione è stata stabilita, il cliente e il server comunicano usando una sequenza di comandi e di risposte. Questi comandi e risposte sono usati per assicurarsi che il cliente recuperi tutti gli articoli di cui ha bisogno, senza duplicazioni, indipendentemente da quanti newsfeed utilizzi. La figura 7-56 mostra i comandi principali usati per trasferire articoli tra demoni di news.

Comando	Significato
LIST	Dammi una lista di tutti i gruppi e articoli che hai
NEWSGROUP data ora	Dammi una lista dei gruppi creati dopo data/ora
GROUP grp	Dammi una lista di tutti gli articoli in grp
NEWNEWS grp data ora	Dammi una lista delle nuove news nei gruppi specificati
ARTICLE id	Dammi l'articolo specificato con id
POST	Ho un articolo per te che è stato pubblicato qui
IHAVE id	Ho l'articolo id. Lo vuoi?
QUIT	Termina la sessione

Fig. 7-56 I principali comandi NNTP per la diffusione degli articoli.

I comandi *LIST* e *NEWSGROUP* permettono al cliente di capire quali siano i gruppi del server. Il primo restituisce la lista completa. Il secondo restituisce solamente quei gruppi

creati dopo la data e l'ora specificati. Se il cliente sa che la lista è lunga, per lui è più efficiente mantenere la traccia di ciò che ha ciascuno dei suoi newsfeed e richiedere solamente gli aggiornamenti. Le risposte a questi comandi consistono in una lista in ASCII, un gruppo per linea, che riporta il nome del gruppo, il numero dell'ultimo articolo posseduto dal server, il numero del primo articolo posseduto dal server e un flag che indica se è permesso pubblicare in questo gruppo.

Una volta che il cliente conosce i gruppi del server, può iniziare a chiedere quali siano invece gli articoli che possiede (ad es., per i vecchi gruppi quando usa *NEWSGROUP*). I comandi *GROUP* e *NEWNEWS* sono usati per questo scopo. Di nuovo, il primo dà una lista completa e il secondo dà solamente gli aggiornamenti successivi alla data e all'ora indicata, di solito il momento dell'ultima connessione a questo newsfeed. Il primo parametro può contenere degli asterischi, per indicarli tutti. Ad esempio, *comp.os.** rappresenta tutti i gruppi che iniziano con la stringa *comp.os*.

Dopo che il cliente ha costruito una lista completa di quali articoli esistano in quali gruppi (o anche prima che abbia la lista completa), può iniziare a richiedere gli articoli che vuole mediante il comando *ARTICLE*. Una volta che ha ricevuto tutti gli articoli, il cliente può offrire gli articoli che ha acquisito da un altro newsfeed utilizzando il comando *IHAVE* e articoli che sono stati pubblicati localmente utilizzando il comando *POST*. Il server può accettarli o rifiutarli, come vuole. Quando il cliente ha finito può terminare la sessione usando *QUIT*. In questo modo, ogni macchina ha il controllo completo su quali articoli recuperare da quali newsfeed, eliminando tutti gli articoli duplicati.

Come esempio di come lavora NNTP, consideriamo ora un fornitore di informazioni, *wholesome.com* che vuole evitare controversie a tutti i costi, perciò i soli gruppi che offre sono *soc.couples* e *misc.kids*. Nonostante ciò, l'amministratore è di mente aperta e vuole portare altri gruppi, assicurandosi che non contengano materiale potenzialmente offensivo. Dunque vuole essere informato su tutti i gruppi creati di recente in modo da prendere una decisione oculata per i suoi clienti. In figura 7-57 abbiamo un possibile scenario tra *wholesome.com* come cliente e il suo newsfeed, *feeder.com*, come server. Questo scenario usa l'approccio a richiesta (il cliente avvia la connessione per richiedere i nuovi articoli). Le frasi tra parentesi sono commenti e non fanno parte del protocollo NNTP.

In questa sessione, *wholesome.com* dapprima chiede se ci sono nuovi articoli per *soc.couples*. Quando gli viene detto che ci sono due articoli, li recupera entrambi e li memorizza in *news/soc/couples* in file separati. Ogni file ha come nome il numero dell'articolo. Quindi *wholesome.com* chiede di *misc.kids* e gli viene risposto che c'è un articolo. Lo recupera e lo inserisce in *news/misc/kids*.

Dopo aver recuperato tutti gli articoli per i gruppi di news che mantiene, chiede se ci siano nuovi gruppi e gli viene risposto che sono apparsi due nuovi gruppi dall'ultima sessione. Uno di questi sembra promettente, così recupera i suoi articoli. L'altro sembra allarmante, per cui non lo prende (*wholesome.com* ha fatto un grosso investimento in software di intelligenza artificiale per essere in grado di capire cosa recuperare semplicemente guardando i nomi.)

Dopo aver acquisito tutti gli articoli voluti, *wholesome.com* offre a *feeder.com* un nuovo articolo pubblicato localmente da qualcuno del suo sito. L'offerta viene accettata e

l'articolo trasferito. Adesso *wholesome.com* offre un altro articolo, uno che viene da uno dei suoi newsfeed. Dato che *feeder.com* ce l'ha già, lo rifiuta.

Infine, *wholesome.com* termina la sessione e rilascia la connessione TCP. L'approccio che prevede l'invio è simile. Inizia con il newsfeed che chiama la macchina che deve ricevere i nuovi articoli. Il newsfeed di solito tiene traccia di quali siano i gruppi ai quali i suoi clienti sono abbonati e inizia annunciando il suo primo articolo nel primo di questi gruppi usando il comando *IHAVE*. Il destinatario potenziale allora controlla le sue tabelle per vedere se ha già l'articolo e se accettarlo o rifiutarlo. Se l'articolo viene accettato, viene trasmesso, seguito da una linea che contiene un punto. Quindi il newsfeed annuncia il secondo articolo, e così via, fino a che li trasferisce tutti.

Un problema con entrambi i metodi è il fatto che si fermano e attendono. Tipicamente vengono persi 100 ms nell'attesa di una risposta a una domanda. Con 100.000 o più nuovi articoli al giorno, questa perdita di tempo comporta uno spreco sostanziale.

7.6 Il World Wide Web

Il World Wide Web è un'architettura software per accedere a documenti tra loro collegati e distribuiti su migliaia di macchine nell'intera Internet. Cinque anni fa era un metodo per distribuire dati per la fisica delle alte energie; oggi è l'applicazione che milioni di persone credono sia "Internet". La sua enorme popolarità deriva dal fatto che ha un'interfaccia grafica colorata facile da utilizzare per i principianti, e costituisce un'enorme fonte di informazioni praticamente su qualsiasi argomento immaginabile, dagli aborigeni alla zoologia.

Il Web (conosciuto anche come WWW) è nato nel 1989 al CERN, il Centro europeo per la ricerca sulla fisica nucleare. Il CERN ha diversi acceleratori per mezzo dei quali i grossi gruppi di scienziati appartenenti ai paesi europei partecipanti portano avanti la ricerca sulla fisica delle particelle. Questi gruppi spesso includono membri provenienti da una mezza dozzina o più di paesi. Molti esperimenti hanno un'alta complessità, e richiedono anni per la pianificazione preventiva e per la costruzione delle apparecchiature. Il Web è nato dalla necessità che hanno questi grossi gruppi di ricercatori di collaborare utilizzando una collezione costantemente mutevole di rapporti, schemi, disegni, foto e altri documenti.

La proposta iniziale per una ragnatela (web) di documenti collegati è del fisico del CERN Tim Berners-Lee, datata marzo 1989. Il primo prototipo (basato su testo) divenne operativo 18 mesi più tardi. Nel dicembre 1991, alla conferenza Hypertext 1991 a San Antonio, Texas, ci fu una dimostrazione al pubblico. Lo sviluppo continuò nell'anno seguente, culminando nella realizzazione della prima interfaccia grafica, Mosaic, nel febbraio 1993 (Vetter et al., 1994).

Mosaic divenne così popolare che un anno più tardi il suo autore, Marc Andreessen, lasciò il National Center for Supercomputing Applications, dove era stato sviluppato Mosaic, per formare una società commerciale, la Netscape Communications Corp., il cui obiettivo era sviluppare programmi client, server e altro software per Web. Quando nel 1995 il programma Netscape divenne pubblico, molti investitori, ipotizzando che la società produttrice fosse la nuova Microsoft, pagaroni in tutto circa 1.500.000.000 di dollari per l'acquisto delle sue azioni. Questo record fu tanto più sorprendente in quanto la società

```

S: 200 feeder.com NNTP server at your service(response to new connection)
C: NEWNEWS soc.couples 960901 030000 (any new news in soc.couples?)
S: 230 List of 2 articles follows
S: <13281@psyc.berkeley.edu> (article 1 of 2 in soc.couples is from Berkeley)
S: <162721@aol.com> (article 2 of 2 in soc.couples is from AOL)
S: .
S: .
C: ARTICLE <13281@psyc.berkeley.edu> (please give me the Berkeley article)
S: 220 <13281@psyc.berkeley.edu> follows
S: (entire article <13281@psyc.berkeley.edu> is sent here)
S: .
S: .
C: ARTICLE <162721@aol.com> (please give me the AOL article)
S: 220 <162721@aol.com> follows
S: (entire article <162721@aol.com> is sent here)
S: .
S: .
C: NEWNEWS misc.kids 960901 030000 (any new news in misc.kids?)
S: 230 List of 1 article follows
S: <43222@bio.rice.edu> (1 article from Rice)
S: .
S: .
C: ARTICLE <43222@bio.rice.edu> (please give me the Rice article)
S: 220 <43222@bio.rice.edu> follows
S: (entire article <43222@bio.rice.edu> is sent here)
S: .
S: .
C: NEWSGROUPS 960901 030000
S: 231 2 new groups follow
S: rec.pets
S: rec.nude
S: .
C: NEWNEWS rec.pets 0 0 (list everything you have)
S: 230 List of 1 article follows
S: <124@fido.net> (1 article from fido.net)
S: .
S: .
C: ARTICLE <124@fido.net> (please give me the fido.net article)
S: 220 <124@fido.net> follows
S: (entire article is sent here)
S: .
C: POST
S: 340 (please send your posting)
C: (article posted on wholesome.com sent here)
S: 240 (article received)
C: IHAVE <5321@foo.com>
S: 435 (I already have it, please do not send it)
C: QUIT
S: 205 (Have a nice day)

```

Fig. 7-57 Il computer *wholesome.com* acquisisce nuovi articoli dal suo fornitore di news.

aveva un solo prodotto, stava operando in profondo rosso e aveva annunciato nel suo prospetto informativo che non si aspettava di avere un profitto nell'immediato futuro.

Nel 1994, CERN e MIT stipularono un accordo che costituiva il Consorzio World Wide Web, un'organizzazione dedicata allo sviluppo ulteriore del Web, alla standardizzazione dei suoi protocolli e a incoraggiare l'interoperatività tra siti. Berners-Lee ne divenne il direttore. Da allora, centinaia di università e società hanno aderito al consorzio. Il MIT si occupa della parte statunitense del consorzio mentre il centro di ricerche francesi, l'INRIA, si occupa della parte europea. Sebbene ci siano più libri sul Web di quanti si possano immaginare, il miglior posto per trovare le informazioni più aggiornate sul Web è naturalmente il Web stesso. La pagina principale (home page) si trova all'indirizzo <http://www.w3.org>. I lettori interessati si riferiscono a essa per le pagine che si occupano di tutti i documenti e le attività del consorzio.

Nei paragrafi che seguono descriveremo come appare il Web all'utente e, in particolar modo, come funziona al suo interno. Dato che il Web è fondamentalmente un sistema client-server, discuteremo sia la parte cliente (cioè l'utente) che la parte server. Quindi esamineremo il linguaggio col quale sono scritte le pagine Web (HTML e Java). Per ultimo, prenderemo in esame il modo in cui trovare le informazioni sul Web.

7.6.1 La parte client

Dal punto di vista dell'utente, il Web consiste in un'enorme collezione di documenti sparsi per il mondo, di solito chiamati brevemente **pagine**. Ogni pagina può contenere puntatori (link) ad altre pagine correlate, ovunque nel mondo. Gli utenti possono seguire un puntatore (ad es., facendo click su di esso), che li porta quindi alla pagina collegata. Questo processo può essere ripetuto all'infinito, con la possibilità in questo modo di attraversare centinaia di pagine correlate. Questa nozione di pagine che puntano ad altre pagine viene detta **ipertesto**. Le pagine vengono visualizzate mediante un programma cliente detto **browser**, tra i quali i più conosciuti sono Mosaic, Netscape e Microsoft Explorer. Il browser recupera la pagina richiesta, interpreta il testo e formatta i comandi che questo contiene, quindi visualizza la pagina, opportunamente formattata, sullo schermo.

La figura 7-58 (a) mostra un esempio. Come molte pagine Web, questa inizia con un titolo, contiene alcune informazioni e termina con l'indirizzo di posta elettronica del curatore della pagina. Le stringhe di testo che sono puntatori ad altre pagine, detti **iperpuntatori**, sono evidenziate o mediante sottolineatura, oppure visualizzandole in un colore speciale, o in entrambi i modi. Per seguire un puntatore l'utente sposta il cursore sulla zona evidenziata (utilizzando il mouse o i tasti freccia) e la seleziona (facendo click sul bottone del mouse o digitando il tasto ENTER). Anche se esistono browser non grafici, per esempio Lynx, questi non sono così popolari come i browser grafici, quindi ci concentreremo su questi ultimi. Sono in fase di sviluppo anche dei browser basati sulla voce. Gli utenti che sono interessati al dipartimento di Psicologia animale possono saperne di più facendo click sul suo nome (sottolineato). Il browser quindi recupera la pagina al quale il nome è collegato e la visualizza, come mostra la figura 7-58 (b). Anche qui è possibile fare click sugli elementi sottolineati per recuperare altre pagine, e così via. La nuova pagina può trovarsi sulla stessa macchina della prima, oppure su una macchina in giro per il mondo. L'utente non può dirlo. Il recupero delle pagine è fatto dal browser,

WELCOME TO THE UNIVERSITY OF EAST PODUNK'S WWW HOME PAGE

- Campus Information
 - [Admissions information](#)
 - [Campus map](#)
 - [Directions to campus](#)
 - [The UEP student body](#)

- Academic Departments
 - [Department of Animal Psychology](#)
 - [Department of Alternative Studies](#)
 - [Department of Microbiotic Cooking](#)
 - [Department of Nontraditional Studies](#)
 - [Department of Traditional Studies](#)

Webmaster@eastpodunk.edu

(a)

THE DEPARTMENT OF ANIMAL PSYCHOLOGY

- [Information for prospective majors](#)
- Personnel
 - [Faculty members](#)
 - [Graduate students](#)
 - [Nonacademic staff](#)
- [Research Projects](#)
- [Positions available](#)
- Our most popular courses
 - [Dealing with herbivores](#)
 - [Horse management](#)
 - [Negotiating with your pet](#)
 - [User-friendly doghouse construction](#)
- [Full list of courses](#)

Webmaster@animalpsyc.eastpodunk.edu

(b)

Fig. 7-58 (a) Una pagina Web. (b) La pagina raggiunta "cliccando" su *Department of Animal Psychology*.

senza alcun aiuto da parte dell'utente. Se l'utente ritorna alla pagina principale, i puntatori già seguiti appaiono sottolineati da una linea tratteggiata (e se possibile in un diverso colore) per distinguere dai puntatori che non sono ancora stati seguiti. Si noti che facendo click sulla linea *Campus Information* nella pagina principale non succede niente. Questa non è sottolineata, e ciò significa che è solo testo e che non è collegata a un'altra pagina. La maggior parte dei browser ha diversi bottoni e diversi modi per rendere più facile la navigazione sul Web. Molti hanno un bottone per tornare indietro alla pagina precedente, un bottone per andare avanti alla prossima pagina (funziona solamente dopo che l'utente è tornato indietro da essa) e un bottone per tornare direttamente indietro alla pagina principale dell'utente. La maggior parte dei browser ha un bottone o un elemento di menu per impostare dei segnalibro (bookmark) su una data pagina e un altro per visualizzare una lista di segnalibri, rendendo possibile visitare nuovamente qualsiasi di essi con un semplice click del mouse. È possibile anche salvare una pagina sul disco oppure stamparla. In genere sono disponibili molte opzioni per controllare la visualizzazione sullo schermo e impostare varie preferenze dell'utente. In Berghel (1996) troviamo un confronto tra nove browser. Oltre ad avere del testo normale (non sottolineato) e dell'ipertesto (sottolineato), le pagine Web possono contenere icone, linee, mappe e fotografie. Ognuna di queste può (optionalmente) venire collegata a un'altra pagina. Facendo click su uno di questi elementi si fa in modo che il browser ritrovi la pagina collegata e la visualizzi, come quando si fa click sul testo. Con immagini come foto o mappe, quale sia la successiva pagina recuperata può dipendere dalla parte dell'immagine su cui si è fatto click. Non tutte le pagine sono visualizzabili in modo convenzionale. Ad esempio, alcune pagine sono costituite da tracce sonore, filmati video o entrambe le cose. Quando le pagine di ipertesto sono meschiate con altre di diversa natura, il risultato è detto **ipermediale**. Alcuni browser possono visualizzare tutti i tipi di dati ipermediali, mentre altri no. Al loro posto questi cercano un file di configurazione per capire come gestire i dati ricevuti. Normalmente, il file di configurazione restituisce il nome di un programma, chiamato **visualizzatore esterno**, o **applicazione di aiuto**, che verrà eseguito avendo la nuova pagina in ingresso. Se non c'è nessun visualizzatore configurato, di solito il browser chiede all'utente di sceglierne uno. Se non esiste alcun visualizzatore, l'utente può ordinare al browser di salvare la nuova pagina in un file su disco, oppure di ignorarla. Le applicazioni ausiliarie che riproducono il linguaggio parlato rendono possibile l'accesso al Web anche per i non vedenti. Altre applicazioni contengono interpreti per particolari linguaggi del Web, rendendo possibile scaricare ed eseguire programmi dalle pagine Web. Questo meccanismo consente di estendere la funzionalità del Web stesso.

Molte pagine Web contengono grandi immagini, che richiedono parecchio tempo per il loro caricamento. Ad esempio, caricare un'immagine non compressa 640×480 (VGA) con 24 bit per pixel (922 KB) richiede circa 4 min con una linea modem a 28,8 Kbps. Alcuni browser aggirano il problema del caricamento lento delle immagini recuperando e visualizzando dapprima il testo, quindi prendendo le immagini. Questa strategia consente all'utente di avere qualcosa da leggere mentre arrivano le immagini ed eventualmente di fermare il caricamento se la pagina non è abbastanza interessante da giustificare l'attesa. Una strategia alternativa è quella di fornire un'opzione per disabilitare il recupero e la visualizzazione automatica delle immagini.

Alcuni scrittori di pagine cercano di attirare la curiosità dell'utente visualizzando immagini in maniera speciale. Dapprima l'immagine appare velocemente con una scarsa risoluzione. Quindi appaiono gradualmente i dettagli. Spesso per l'utente vedere l'intera immagine dopo qualche secondo, anche se a bassa risoluzione, è preferibile al vederla apparire lentamente dall'inizio, linea dopo linea.

Alcune pagine Web contengono dei "form", che sono una specie di moduli elettronici che richiedono all'utente di immettere delle informazioni. Le applicazioni tipiche per i form sono le interrogazioni a una base di dati, vuoi per ordinare un prodotto, vuoi per partecipare a un sondaggio di opinioni. Altre pagine Web contengono mappe attive che permettono all'utente di fare click su di esse per visualizzare o recuperare informazioni su una particolare zona geografica. La gestione di form e mappe attive richiede un'elaborazione più sofisticata rispetto al solo recupero di una pagina data. Vedremo in seguito come sono realizzate tali funzionalità.

Alcuni browser utilizzano il disco locale come memoria temporanea (cache) per le pagine recuperate. Prima di recuperare una pagina, viene fatta una verifica per vedere se si trova già nella cache locale. Se c'è, basta verificare che la pagina sia aggiornata. In caso affermativo, non è necessario caricare nuovamente la pagina. Per esempio, il click sul bottone BACK per vedere la pagina precedente è di solito molto veloce.

Per utilizzare un browser Web, una macchina deve necessariamente essere direttamente su Internet, o almeno avere una connessione SLIP o PPP a un router o a un'altra macchina che sia collegata direttamente in Internet. Questo requisito è necessario perché il modo in cui il browser recupera una pagina avviene stabilendo una connessione TCP alla macchina su cui si trova la pagina, e quindi inviando un messaggio attraverso la connessione che richiede la pagina. Se non è possibile stabilire una connessione TCP verso una macchina qualsiasi collegata in Internet, il browser non funzionerà.

A volte il tempo che si aspetta per ottenere l'accesso al Web è spaventosamente lungo. Esiste almeno una società che offre il servizio Web via fax. Un cliente senza accesso a Internet chiama il server Web via fax e si collega attraverso la tastiera del telefono. Quindi digita un codice per identificare la pagina desiderata e questa viene inviata al fax del chiamante.

7.6.2 La parte server

Ogni sito Web ha un processo server in ascolto sulla porta 80 del TCP di connessioni in arrivo dai clienti (normalmente browser). Dopo che è stata stabilita una connessione, il cliente invia una richiesta e il server invia una risposta. Quindi la connessione viene abbandonata. Il protocollo che definisce la forma delle richieste e delle risposte è chiamato HTTP. Lo studieremo con un certo dettaglio più avanti, ma un semplice esempio del suo utilizzo può dare una ragionevole idea di come lavori il Web. La figura 7-59 mostra come si integrano le varie parti del modello Web.

Per questo esempio, possiamo immaginare che l'utente abbia appena fatto click su una parte di testo o anche su un'icona che punta alla pagina il cui nome (URL – Uniform Resource Locator) è <http://www.w3.org/hypertext/WWW/TheProject.html>.

Più avanti in questo capitolo esamineremo anche gli URL. Per il momento, è sufficiente sapere che una stringa ha tre parti: il nome del protocollo (*http*), il nome della macchina su cui si trova la pagina (www.w3.org) e il nome del file che contiene la pagina (*hyper-*

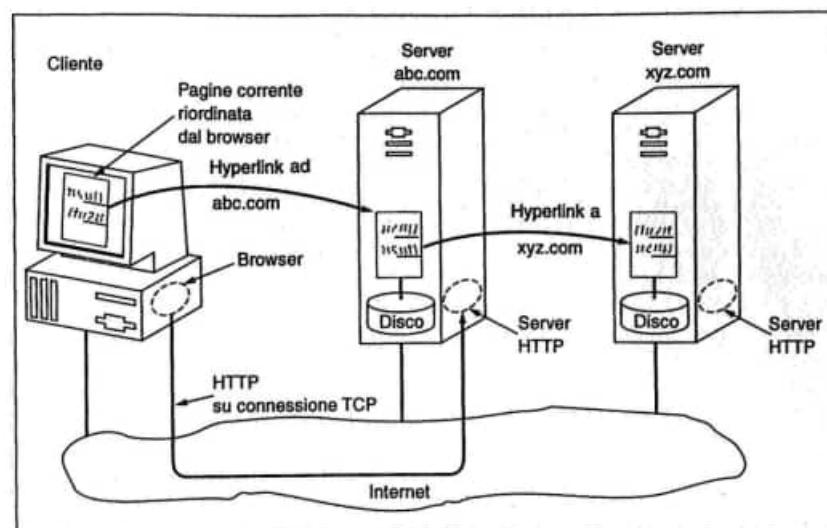


Fig. 7-59 I componenti del modello Web.

text/WWW/TheProject.html). Le fasi che si succedono tra il click dell'utente e la visualizzazione della pagina sono le seguenti:

1. Il browser determina l'URL (controllando ciò che è stato selezionato).
2. Il browser chiede al DNS l'indirizzo di www.w3.org.
3. Il DNS risponde con 18.23.0.23.
4. Il browser fa una connessione TCP alla porta 80 su 18.23.0.23.
5. Quindi invia un comando *GET/hypertext/WWW/TheProject.html*.
6. Il server www.w3.org invia il file *TheProject.html*.
7. Viene rilasciata la connessione TCP.
8. Il browser visualizza tutto il testo contenuto in *TheProject.html*.
9. Il browser recupera e visualizza tutte le immagini contenute in *TheProject.html*.

Molti browser indicano il passo che stanno eseguendo al momento in una linea di stato in fondo allo schermo. In questo modo, quando le prestazioni sono lente, l'utente può vedere se ciò è dovuto al DNS che non risponde, al server che non risponde o semplicemente alla congestione di rete durante la trasmissione della pagina.

Vale la pena di notare che per ogni immagine in linea (icona, disegno, foto ecc.) contenuta nella pagina, il browser stabilisce una nuova connessione TCP verso il server per recupe-

rare l'immagine. Inutile a dirsi, se una pagina contiene molte icone, tutte sullo stesso server, non è molto efficiente attivare, utilizzare e rilasciare una nuova connessione per ognuna di esse, ma l'implementazione resta comunque semplice. Le future versioni del protocollo penseranno all'efficienza. Troviamo una proposta in Mogul (1995).

Dato che HTTP è un protocollo ASCII come SMTP, è abbastanza semplice per un utente a un terminale (che non sia davanti a un browser) parlare direttamente con i server Web. Tutto ciò che serve è una connessione TCP alla porta 80 del server. Il modo più semplice per ottenere una connessione di questo tipo è utilizzare il programma Telnet. La figura 7-60 mostra uno scenario di come questo possa avvenire. In questo esempio, le linee marcate C: sono digitate dall'utente (il client), le linee marcate T: sono prodotte dal programma Telnet, mentre le linee marcate S: sono prodotte dal server del MIT.

I lettori sono incoraggiati a provare questo esempio personalmente (preferibilmente da un sistema UNIX, perché certi altri sistemi non restituiscono la linea di stato). Si notino gli spazi e la versione del protocollo sulla linea *GET*, e la linea bianca che segue la linea *GET*. Per inciso, il vero testo che si riceverà differirà da quello mostrato in figura 7-60 per tre ragioni. Primo, l'output dell'esempio qui è stato accorciato e modificato perché entrasse in una pagina. Secondo, qualcosa è stato cancellato per non mettere in imbarazzo l'autore, che senza dubbio si aspetta che siano migliaia le persone che osservano la pagina formattata, ma zero quelle che giudicano l'HTML che la produce. Terzo, i contenuti delle pagine vengono costantemente modificati. Ciò nonostante, questo esempio vuol dare un'idea ragionevole di come lavora il protocollo HTTP.

L'esempio mostra la cosa seguente: il client, in questo caso una persona, ma normalmente un browser, dapprima si connette a una particolare macchina e quindi invia un comando che richiede una pagina particolare e che specifica quale determinato protocollo e versione usare (HTTP/1.0). Alla linea 7, il server risponde con una linea di stato che indica il protocollo utilizzato (lo stesso del client) e il codice 200, che significa OK. Questa linea è seguita da un messaggio RFC 822 MIME, del quale in figura sono mostrate cinque linee dell'intestazione (molte altre sono state omesse per salvare spazio). Quindi arriva una linea bianca, seguita dal corpo del messaggio.

Per inviare una figura, il campo *Content-Type* dovrebbe essere

Content-Type: Image/GIF

In questo modo, i tipi MIME permettono di inviare oggetti arbitrari nella maniera standard. Per inciso, non è necessaria l'intestazione MIME *Content-Transfer-Encoding* perché il TCP permette di inviare senza modifica flussi di byte arbitrari, anche figure. Il significato dei comandi tra parentesi acute utilizzati nella pagina di esempio sarà discusso più avanti in questo capitolo.

Non tutti i server parlano HTTP. In particolare, molti server più vecchi utilizzano FTP, Gopher oppure altri protocolli. Dato che sui server FTP e Gopher è disponibile una grande quantità di informazioni utili, uno degli obiettivi di progetto del Web era di rendere disponibili queste informazioni agli utenti. Una soluzione è fare in modo che il browser utilizzi questi protocolli quando parla con un server FTP o Gopher. Alcuni di essi, infatti, utilizzano questa soluzione, ma fare in modo che i browser capiscano qualsiasi protocollo possibile li rende inutilmente grandi.

```
C: telnet www.w3.org 80
T: Trying 18.23.0.23 ...
T: Connected to www.w3.org.
T: Escape character is ']'.
C: GET /hypertext/WWW/TheProject.html HTTP/1.0
C:
S: HTTP/1.0 200 Document follows
S: MIME-Version: 1.0
S: Server: CERN/3.0
S: Content-Type: text/html
S: Content-Length: 8247
S:
S: <HEAD> <TITLE> The World Wide Web Consortium (W3C) </TITLE> </HEAD>
S: <BODY>
S: <H1> <IMG ALIGN=MIDDLE ALT="W3C" SRC="icons/WWW/w3c_96x67.gif">
S: The World Wide Web Consortium </H1> <P>
S:
S: The World Wide Web is the universe of network-accessible information.
S: The <A HREF="Consortium/"> World Wide Web Consortium </A>
S: exists to realize the full potential of the Web. <P>
S:
S: W3C works with the global community to produce
S: <A HREF="#Specifications"> specifications </A> and
S: <A HREF="#Reference"> reference software </A> .
S: W3C is funded by industrial
S: <A HREF="Consortium/Member>List.html"> members </A>
S: but its products are freely available to all. <P>
S:
S: In this document:
S: <menu>
S: <L1> <A HREF="#Specifications"> Web Specifications and Development Areas </A>
S: <L1> <A HREF="#Reference"> Web Software </A>
S: <L1> <A HREF="#Community"> The World Wide Web and the Web Community </A>
S: <L1> <A HREF="#Joining"> Getting involved with the W3C </A>
S: </menu>
S: <P> <HR>
S: <P> W3C is hosted by the
S: <A HREF="http://www.lcs.mit.edu/"> Laboratory for Computer Science </A> at
S: <A HREF="http://web.mit.edu/"> MIT </A> , and
S: in Europe by <A HREF="http://www.inria.fr/"> INRIA </A> .
S: </BODY>
```

Fig. 7-60 Uno scenario tipo per ottenere una pagina Web.

Invece, di solito viene usata una soluzione diversa: i server proxy (Luotonen, Altis (1994). Un **server proxy** è una specie di gateway che parla HTTP col browser, ma che parla FTP, Gopher, e con linguaggio di qualche altro protocollo col server. Accetta richieste HTTP e le traduce in, diciamo, richieste FTP, così il browser non ha bisogno di capire i protocolli diversi da HTTP. Il server proxy può essere un programma che gira sulla stessa macchina del browser, ma può anche essere su una macchina a sé stante, da qualche parte nella rete, in maniera da servire più browser. La figura 7-61 mostra la differenza tra un browser che parla FTP e uno che usa un proxy.

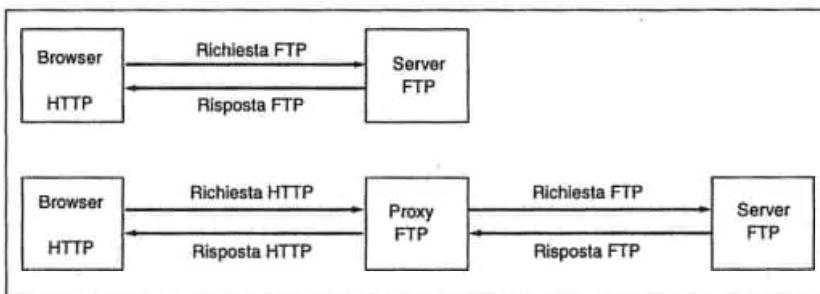


Fig. 7-61 (a) Un browser che parla FTP. (b) Un browser che non lo parla.

Spesso gli utenti possono configurare i loro browser con dei proxy per il linguaggio dei protocolli che il browser non parla. In questo modo aumentano le fonti di informazioni a cui il browser ha accesso.

Oltre ad agire come un tramite per i protocolli sconosciuti, i server proxy hanno un certo numero di altre funzioni importanti, come il caching. Un server proxy per caching colleziona e mantiene tutte le pagine che lo attraversano. Quando l'utente richiede una pagina, il server proxy controlla se ha già quella pagina. Se sì, verifica che la pagina sia ancora attuale. Se lo è, la passa all'utente, altrimenti recupera una nuova copia.

Infine, un'organizzazione può inserire un server proxy all'interno del suo firewall per permettere agli utenti di accedere al Web, ma senza dare loro accesso all'intera Internet. In questa configurazione, gli utenti colloquiano con il server proxy, ed è quest'ultimo che contatta i siti remoti e recupera le pagine per conto dei suoi clienti. Questo meccanismo può essere usato, ad esempio, nelle scuole superiori, per bloccare l'accesso ai siti Web che il preside ritiene siano inappropriati per ragazzi di quell'età.

Per informazioni su uno dei server più popolari (il demone HTTP dello NCSA) e le sue performance, si vedano Katz *et al.* (1994); Kwan *et al.* (1995).

HTTP – HyperText Transfer Protocol

Il protocollo di trasferimento del Web è **HTTP (HyperText Transfer Protocol)**. Ogni interazione consiste in una richiesta ASCII, seguita da una risposta del tipo RFC 822 MIME. Anche se l'uso di TCP per la connessione di trasporto è molto comune, non è

formalmente richiesta dallo standard. Se le reti ATM divenissero abbastanza affidabili, le richieste e risposte HTTP si potrebbero trasportare altrettanto bene in messaggi AAL 5. HTTP è in costante evoluzione. Ci sono diverse versioni in uso e altre sono in fase di sviluppo. Il materiale presentato di seguito è abbastanza fondamentale ed è improbabile che venga modificato nella sostanza, ma è possibile che nelle versioni future alcuni dettagli risultino un po' diversi.

Il protocollo HTTP consiste di due elementi distinti: l'insieme delle richieste da parte del browser ai server e l'insieme delle risposte che tornano indietro. Li tratteremo uno alla volta.

Tutte le ultime versioni di HTTP supportano due tipi di richieste: le richieste semplici e le richieste complete. Una richiesta semplice è una sola linea *GET* che nomina la pagina desiderata, senza la versione del protocollo. La risposta consiste solamente della pura pagina, senza intestazioni, MIME o codifica. Per vedere come funziona, provate a fare una connessione Telnet alla porta 80 di www.w3.org (come mostra la prima linea di figura 7-60) e quindi digitate

```
GET /hypertext/WWW/TheProject.html
```

ma questa volta senza specificare HTTP/1.0. Viene restituita la pagina senza alcuna indicazione sul tipo del suo contenuto. Questo meccanismo è necessario per la compatibilità con le vecchie versioni. Il suo uso scomparirà quando i browser e i server basati sulle richieste complete diventeranno lo standard.

La presenza della versione del protocollo sulla linea della richiesta *GET*, come in figura 7-60, indica una richiesta completa. Le richieste consistono in più linee, seguite da una linea bianca che indica la fine della richiesta, e questo spiega perché la linea bianca in figura 7-60 fosse necessaria. La prima linea di una richiesta completa contiene il comando (del quale *GET* è soltanto una delle possibilità), la pagina desiderata e il protocollo/versione. Le linee successive contengono le intestazioni RFC 822.

Sebbene HTTP sia stato inventato per essere utilizzato specificatamente nel Web, è stato reso intenzionalmente più versatile, pensando alle future applicazioni orientate agli oggetti. Per questa ragione, la prima parola sulla linea di una richiesta completa è semplicemente il nome del **metodo** (comando) da eseguire sulla pagina Web (o sull'oggetto generico). I metodi nativi sono elencati in figura 7-62. Quando si accede agli oggetti generici, possono essere disponibili altri metodi specifici all'oggetto. I nomi sono sensibili al maiuscolo o minuscolo, perciò *GET* è un metodo lecito mentre *get* non lo è.

Il metodo *GET* richiede al server di inviare la pagina (con la quale intendiamo l'oggetto, nel caso più generale), opportunamente codificata in MIME. In ogni caso, se una richiesta *GET* è seguita dall'intestazione *If-Modified-Since*, il server invia i dati solo nel caso in cui questi siano stati modificati dopo la data fornita. Utilizzando questo meccanismo, un browser che richieda una pagina che si trova nella cache può fare una richiesta condizionata al server, indicando la data di modifica associata con quella pagina. Se la pagina in cache è ancora valida, il server restituisce una linea di stato che annuncia questo fatto, eliminando così l'overhead per trasferire nuovamente la pagina.

Il metodo *HEAD* richiede solo l'intestazione del messaggio, senza la pagina vera e pro-

pria. Questo metodo può essere usato per ottenere la data dell'ultima modifica alla pagina, per collezionare informazioni allo scopo di inserirle in un indice, o semplicemente per verificare la validità di un URL. Le richieste *HEAD* condizionali non esistono.

Metodo	Descrizione
GET	Richiede lettura di una pagina Web
HEAD	Richiede lettura di un'intestazione di una pagina Web
PUT	Richiede di memorizzare una pagina Web
POST	Aggiunge a una risorsa indicata (ad esempio, una pagina WEB)
DELETE	Cancella una pagina Web
LINK	Connette due risorse esistenti
UNLINK	Interrompe una connessione esistente tra due risorse

Fig. 7-62 I metodi nativi per le richieste HTTP.

Il metodo *PUT* è l'inverso di *GET*, invece di leggere una pagina, la scrive. Con questo metodo è possibile costruire una collezione di pagine Web su un server remoto. Il corpo della richiesta contiene la pagina. Può essere codificata utilizzando MIME, nel qual caso le linee che seguono *PUT* possono includere *Content-Type* e intestazioni di autenticazione, per provare che il chiamante ha davvero i permessi per eseguire l'operazione richiesta.

Simile a *PUT* è il metodo *POST*. Anche questo trasporta un URL, ma invece di rimpiazzare i dati esistenti, quelli nuovi vengono genericamente "appesi" a essi. Pubblicare un messaggio in un gruppo di news o aggiungere un file a un sistema di bulletin board sono esempi di cosa significhi appendere in questo contesto. Qui chiaramente l'idea è che il Web supporti le funzionalità del sistema di news USENET.

DELETE fa quello che ci si può aspettare: cancella la pagina. Come con *PUT*, qui l'autenticazione e i diritti hanno un ruolo importante. Non c'è alcuna garanzia che *DELETE* abbia successo, poiché anche se il server HTTP remoto permette di cancellare la pagina, il file sottostante può avere una modalità che proibisce al server HTTP di modificarla o di cancellarla.

I metodi *LINK* e *UNLINK* permettono di stabilire delle connessioni tra pagine esistenti o altre risorse.

Ogni richiesta ottiene una risposta che consiste in una linea di stato e possibilmente di altre informazioni (ad es., tutta o parte della pagina Web). La linea di stato può contenere il codice 200 (OK), oppure uno qualsiasi dell'insieme dei codici di errore, per esempio 304 (non modificabile), 400 (richiesta non corretta) oppure 403 (proibito).

Gli standard HTTP descrivono le intestazioni e il corpo dei messaggi con particolare dettaglio. È sufficiente dire che questi sono molto simili ai messaggi MIME RFC 822, perciò qui non li vedremo.

7.6.3 Scrivere una pagina Web in HTML

Le pagine Web sono scritte in un linguaggio chiamato **HTML** (**H**yper**T**ext **M**arkup **L**anguage). HTML permette all'utente di costruire delle pagine Web che comprendono testo, grafica e puntatori ad altre pagine Web. Inizieremo a studiare HTML da questi puntatori, poiché sono il collante che tiene unito il Web.

URL – Uniform Resource Locator

Abbiamo detto parecchie volte che le pagine possono contenere più puntatori ad altre pagine. È il momento di vedere come sono implementati questi puntatori. Quando venne creato il Web, risultò immediatamente chiaro che il fatto di avere una pagina che punta a un'altra pagina richiedeva dei meccanismi per denotare e localizzare le pagine. In particolare, erano tre le domande che richiedevano una risposta prima che una pagina selezionata fosse visualizzata:

1. Qual è la pagina richiesta?
2. Dove si trova la pagina?
3. Come si può accedere alla pagina?

Se a ogni pagina veniva in qualche modo assegnato un nome, non ci poteva essere alcuna ambiguità nell'identificare le pagine. Nonostante ciò, il problema non era risolto. Confrontiamo le persone e le pagine. Negli Stati Uniti quasi tutti hanno un codice fiscale, che è un identificatore unico, in quanto non esistono due persone con lo stesso codice. Nonostante ciò, armati del solo codice fiscale, non c'è alcun modo per trovare l'indirizzo del proprietario, e sicuramente non c'è nessun modo per dire se a quella persona dovremmo scrivere in inglese, spagnolo o cinese. Il Web pone essenzialmente gli stessi problemi.

La soluzione scelta identifica le pagine in un modo che risolve tutti e tre i problemi in una volta sola. A ogni pagina è assegnato un **URL** (**U**niform **R**esource **L**ocator) che effettivamente ha la funzione di nome di pagina univoco per tutto il mondo. Gli URL hanno tre parti: il protocollo (detto anche schema), il nome DNS della macchina sulla quale si trova la pagina, e un nome locale che indica in maniera unica la pagina specifica (di solito semplicemente un nome di file sulla macchina dove risiede). Ad esempio, l'URL del dipartimento dell'autore

<http://www.cs.vu.nl/welcome.html>

Questo URL consiste di tre parti: il protocollo (*http*), il nome DNS della macchina (*www.cs.vu.nl*) e il nome del file (*welcome.html*), con la punteggiatura data che separa le parti. Alcuni siti hanno determinate abbreviazioni per i nomi dei file primitivi. Ad esempio, *~user* può corrispondere alla directory WWW di *user*, con la convenzione che un riferimento alla directory stessa implica un certo file, diciamo *index.html*. In questo modo la pagina principale dell'autore può essere ricercata a

<http://www.cs.vu.nl/~ast/>

Anche se il vero nome di file è diverso. In molti siti, un nome di file nullo porta per default alla pagina principale dell'organizzazione.

A questo punto dovrebbe essere chiaro come lavora un ipertesto. Per rendere "cliccabile" una parte di testo, colui che scrive la pagina deve dare due informazioni: il testo "cliccabile" da visualizzare e l'URL della pagina in cui andare se il testo viene selezionato. Quando il testo viene selezionato, il browser ricerca il nome della macchina utilizzando il DNS. A questo punto armato dell'indirizzo IP della macchina, il browser stabilisce una connessione TCP con la macchina. Tramite questa connessione invia il nome del file utilizzando il protocollo specifico. Fatto. Torna indietro la pagina. Questo è esattamente quello che vediamo in figura 7-60.

Questo schema è aperto, nel senso che dà la possibilità di specificare protocolli diversi da HTTP. Infatti, sono stati definiti URL per diversi altri protocolli, e molti browser sono in grado di capirli. In figura 7-63 vediamo le forme leggermente semplificate di quelli più comuni.

Nome	Usato per	Esempio
http	Hypertext (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	File locale	/usr/suzanne/prog.c
news	Gruppo di news	news:comp.os.minix
news	Articolo di news	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Invio di posta elettronica	mailto:kim@acm.org
telnet	Login remoto	telnet://www.w3.org:80

Fig. 7-63 Alcuni URL comuni.

Analizziamo brevemente l'elenco. Il protocollo *http* è il linguaggio nativo del Web, quello parlato dai server HTTP. Supporta tutti i metodi di figura 7-62, così come qualsiasi metodo specifico richiesto.

Il protocollo *ftp* viene usato per accedere ai file via FTP, il protocollo per il trasferimento di file di Internet. FTP è usato da più di due decenni ed è ben conosciuto. Molti server FTP in tutto il mondo permettono a chiunque sia su Internet di connettersi e scaricare qualsiasi file si trovi su un server FTP. Il Web non modifica questo schema; semplicemente rende più facile ottenere i file tramite FTP, dato che FTP ha un'interfaccia vecchio stile. Nel futuro probabilmente FTP scomparirà in quanto non c'è alcun particolare vantaggio per un sito nell'installare un server FTP rispetto a un server HTTP, che può fare tutto quel che fa il server FTP, e anche di più (anche se c'è qualcosa da dire sull'efficienza).

È possibile accedere a un file locale per vederlo come pagina Web, o usando il protocollo *file*, o più semplicemente solo dando il suo nome. Questo approccio è come usare FTP ma non richiede l'esistenza di un server. Ovviamente, funziona solo con i file locali.

Il protocollo *news* permette a un utente del Web di richiamare un articolo di news come

se fosse una pagina Web. Questo significa che un browser è anche un programma lettore di news. Infatti, molti browser hanno dei bottoni o delle voci di menu per poter leggere le news USENET in maniera persino più semplice dei lettori di news standard.

Per il protocollo *news* sono ammessi due formati. Il primo formato specifica un gruppo di news e può essere usato per ottenere un elenco di articoli da un sito di news prestabilito. Il secondo richiede l'identificatore di uno specifico articolo di news, in questo caso <AA0134223112@cs.utah.edu>. Il browser a questo punto recupera l'articolo dal suo sito di news prestabilito mediante il protocollo NNTP.

Il protocollo *gopher* viene usato dal sistema Gopher, progettato all'Università del Minnesota e così chiamato in onore della sua squadra di atletica, i Golden Gophers ("gopher" è anche un'espressione in gergo che significa "andare per", cioè recuperare). Gopher ha anticipato il Web di diversi anni. È un metodo per il recupero di informazioni, concettualmente simile allo stesso Web, ma che ammette solo il testo e non le immagini. Quando un utente si connette a un server Gopher, gli appare un menu di file e directory, ognuno dei quali può puntare a un altro menu Gopher in una qualsiasi parte del mondo.

Il grosso vantaggio di Gopher rispetto al Web è che lavora molto bene con i terminali ASCII 25 × 80, che sono ancora molto usati, e dato che è basato sul testo è molto veloce. Di conseguenza, esistono in giro per il mondo migliaia di server Gopher. Con il protocollo *gopher*, gli utenti del Web possono accedere al Gopher e vedere ogni menu Gopher presentato come una pagina Web "cliccabile". Se non conoscete il Gopher, provate a eseguire l'esempio in figura 7-63 o a far cercare al vostro motore di ricerca preferito la parola "gopher".

È anche possibile inviare una richiesta completa a un server Gopher usando il protocollo *gopher+*, anche se l'esempio non lo illustra. Ciò che viene visualizzato è il risultato dell'interrogazione sul server Gopher remoto.

Gli ultimi due protocolli in realtà non si occupano di recuperare delle pagine Web, e non sono supportati da tutti i browser, ma sono comunque utili. Il protocollo *mailto* permette agli utenti di inviare posta elettronica da un browser Web. Per farlo, fare click sul bottone OPEN e specificare un URL composto da *mailto:* seguito dall'indirizzo del destinatario. Molti browser risponderanno mostrando una maschera con dei campi per il soggetto e altre linee di intestazione e lo spazio per inserire il messaggio.

Il protocollo *telnet* viene usato per stabilire una connessione con una macchina remota. Viene usato allo stesso modo del programma Telnet, il che non sorprende in quanto molti browser richiamano semplicemente Telnet come un'applicazione di supporto. Come esercizio, provare nuovamente lo scenario di figura 7-60, utilizzando questa volta un browser Web.

In breve, gli URL sono stati progettati non solo per permettere agli utenti di navigare il Web, ma anche per lavorare con FTP, news, Gopher, posta elettronica e telnet, rendendo inutili tutti gli speciali programmi di interfaccia per tali servizi, e integrando così quasi tutti gli accessi su Internet in un unico programma, il browser Web. Se non fosse per il fatto che questo schema è stato progettato da un ricercatore di fisica, potrebbe facilmente passare per invenzione di qualche ufficio di pubblicità di una società di software.

A dispetto di tutte queste piacevoli proprietà, l'uso crescente del Web ha rivelato una debolezza nello schema degli URL. Un URL punta a uno specifico nodo. Per le pagine

che sono molto riferite, sarebbe desiderabile avere più copie tra loro distanti, in modo da ridurre il traffico di rete. Il problema è che gli URL non forniscono nessun modo per riferire una pagina senza dire contemporaneamente dove questa si trovi. Non c'è modo di dire: "Voglio la pagina xyz, ma non mi interessa dove la prendi". Per risolvere questo problema e rendere possibile la replicazione delle pagine, IETF sta lavorando a un sistema di **URI (Universal Resource Identifiers)**. Un URI può essere pensato come una URL generalizzata. Questo argomento è il soggetto di molte ricerche in corso.

Anche se qui abbiamo discusso solamente gli URL assoluti, esistono anche gli URL relativi. La differenza è analoga alla differenza tra il nome assoluto di file `/usr/ast/foobar` e il solo `foobar` quando il contesto è definito senza ambiguità.

HTML – HyperText Markup Language

Adesso che abbiamo una buona idea di come lavorano gli URL, possiamo studiare HTML. HTML è un'applicazione dello standard ISO 8879, che descrive **SGML (Standard Generalized Markup Language)**, però specializzato nel trattamento di ipertesti e adattato al Web.

Come abbiamo detto in precedenza, HTML è un linguaggio che usa annotazioni (markup) per descrivere il modo in cui i documenti vanno formattati. Il termine "markup" deriva dai vecchi tempi in cui i correttori di bozze annotavano sui documenti per lo stampatore – a quei tempi, un essere umano – le indicazioni tipografiche. I linguaggi con markup quindi contengono dei comandi esplicativi per la formattazione. Ad esempio, in HTML, `` significa inizia la modalità grassetto, e `` significa lascia la modalità grassetto. Il vantaggio di un linguaggio con markup esplicativi rispetto a uno che non li ha, è che scrivere un browser è semplice: basta che il browser capisca i comandi markup. TeX e troff sono altri esempi famosi di linguaggi con markup.

Possiamo mettere a confronto dei documenti scritti in un linguaggio con markup con dei documenti prodotti con un elaboratore di testi WYSIWYG (What You See Is What You Get), come MS-Word o WordPerfect. Questi sistemi possono memorizzare i loro file con dei markup nascosti in modo da riprodurli in un secondo tempo, ma non tutti lavorano in questo modo. Gli elaboratori di testi per il Macintosh, ad esempio, mantengono le informazioni di formattazione in strutture dati separate, e non in forma di comandi inseriti nei file dell'utente.

Inserendo i comandi markup all'interno di ogni file HTML e standardizzandoli, diventa possibile per qualsiasi browser Web leggere e riformattare qualsiasi pagina Web. Poter riformattare le pagine Web dopo averle ricevute è cruciale in quanto una pagina può essere stata prodotta per uno schermo 1024×768 con colori a 24 bit, ma deve venire visualizzata in una finestra piccola su uno schermo 640×480 con colori di 8 a bit. Non è possibile utilizzare gli elaboratori di testi WYSIWYG commerciali sul Web in quanto i loro linguaggi di markup interni (se esistono) non sono standardizzati tra loro, ciò rispetto a produttori, macchine e sistemi operativi. Inoltre, non sono ben manipolabili per essere riformattati per finestre di dimensioni diverse e per risoluzioni di schermo differenti. In ogni modo, i programmi per l'elaborazione di testi possono offrire l'opzione di salvare i documenti in HTML invece che nel proprio formato, e alcuni di essi lo fanno già.

Come HTTP, HTML è costantemente in evoluzione. Quando l'unico browser era

Mosaic, il linguaggio che esso interpretava, HTML 1.0, era lo standard de facto. Quando arrivarono altri browser nacque l'esigenza di uno standard Internet formale, per cui venne definito lo standard HTML 2.0. Lo standard HTML 3.0 fu creato inizialmente come prototipo di ricerca per aggiungere molte nuove caratteristiche all'HTML 2.0, comprese le tabelle, le barre di strumenti, le formule matematiche, fogli di stile avanzati (per la definizione di formati di pagina e il significato dei simboli), e altro ancora.

La standardizzazione ufficiale di HTML è gestita dal consorzio WWW, ma molti produttori di browser hanno aggiunto le loro estensioni particolari. Questi produttori sperano di convincere i produttori di pagine a scriverle con le loro estensioni, in modo che i lettori di tali pagine abbiano bisogno di quel particolare browser per visualizzarle nella maniera corretta. Questa tendenza non semplifica la standardizzazione dell'HTML.

In seguito introduciamo brevemente HTML, giusto per dare un'idea di cosa sia. Anche se è certamente possibile scrivere i documenti HTML con un qualsiasi editor, e molte persone lo fanno, è anche possibile utilizzare degli speciali editor HTML che fanno la maggior parte del lavoro (ma di conseguenza danno all'utente un controllo minore su tutti i dettagli del risultato finale).

Una pagina Web include un'intestazione e un corpo racchiusi tra i tag (comandi di formattazione) `<HTML>` e `</HTML>`, anche se molti browser non si curano della mancanza di questi tag. Come si vede dalla figura 7-64 (a), l'intestazione è racchiusa tra i tag `<HEAD>` e `</HEAD>` e il corpo è racchiuso tra i tag `<BODY>` e `</BODY>`.

I comandi interni a un tag sono detti **direttive**. Molti tag HTML hanno questo formato, vale a dire, `<QUALCOSA>` indica l'inizio di qualcosa e `</QUALCOSA>` indica la sua fine. Ci sono numerosi altri esempi di HTML. Molti browser hanno una voce di menu **VIEW SOURCE** o qualcosa di simile. Selezionando questa voce viene visualizzata la sorgente HTML della pagina corrente, invece dell'output formattato.

I tag possono essere sia in minuscolo che in maiuscolo. Così `<HEAD>` e `<head>` significano la stessa cosa, ma il primo si individua meglio. L'aspetto reale del documento HTML è irrilevante. I parser HTML ignorano gli spazi in più e il ritorno carrello in quanto il loro compito è riformattare il testo perché stia nell'area disponibile di visualizzazione. Di conseguenza, è possibile aggiungere spazi bianchi a piacere per rendere più leggibili i documenti HTML, cosa di cui hanno disperatamente bisogno. Come altra conseguenza, non si utilizzano le linee bianche per separare i paragrafi, in quanto vengono semplicemente ignorate: è richiesto un tag esplicito.

Alcuni tag hanno dei parametri (col nome). Ad esempio:

```
<IMG SRC="abc" ALT="foobar">
```

È un tag, ``, con il parametro `SRC` uguale ad `abc` e il parametro `ALT` uguale a `foobar`. Per ogni tag, lo standard HTML definisce una lista dei parametri permessi, se ci sono, e spiega cosa significano. Dato che ogni parametro ha un nome, l'ordine col quale sono dati non è significativo.

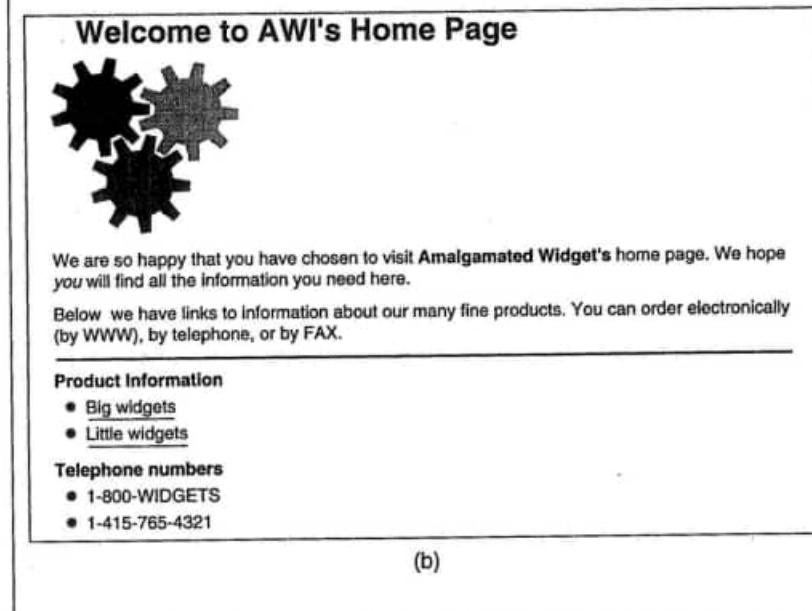
Dal punto di vista tecnico, i documenti HTML sono scritti nell'insieme di caratteri ISO 8859-1 Latin-1, ma per quegli utenti con tastiere che supportano solamente l'ASCII,

```

<HTML> <HEAD> <TITLE> AMALGAMATED WIDGET, INC. </TITLE> </HEAD>
<BODY> <H1> Welcome to AWI's Home Page </H1>
<IMG SRC="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <BR>
We are so happy that you have chosen to visit <B> Amalgamated Widget's</B>
home page. We hope <i>you</i> will find all the information you need here.
<P>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by fax. <HR>
<H2> Product information </H2>
<UL> <LI> <A HREF="http://widget.com/products/big"> Big widgets </A>
    <LI> <A HREF="http://widget.com/products/little"> Little widgets </A>
</UL>
<H2> Telephone numbers </H2>
<UL> <LI> By telephone: 1-800-WIDGETS
    <LI> By fax: 1-415-765-4321
</UL> </BODY> </HTML>

```

(a)



(b)

Fig. 7-64 (a) Codice HTML per una pagina WEB di esempio. (b) La pagina formattata.

esistono delle sequenze di escape per i caratteri speciali, ad esempio per è. Lo standard dà l'elenco dei caratteri speciali. Tutti iniziano con una e commerciale e terminano con un punto e virgola. Ad esempio, è produce "è" ed é produce "é". Dato che <, > ed & hanno un significato speciale, è possibile esprimere solamente con la loro sequenza di escape, rispettivamente < > ed &.

L'elemento principale dell'intestazione è il titolo, delimitato da <TITLE> e </TITLE>, ma possono essere presenti anche delle meta-informationi. Il titolo non viene visualizzato nella pagina. Alcuni browser lo utilizzano per dare un nome alla finestra che contiene la pagina.

Vediamo adesso alcune delle altre caratteristiche illustrate in figura 7-64. La figura 7-65 mostra tutti i tag usati in figura 7-64 e altri ancora. I titoli sono generati da un tag <Hn>, dove n è una cifra nell'intervallo da 1 a 6. <H1> è per il titolo più importante; <H6> per il meno importante. Dipende dal browser rappresentare tutto questo sullo schermo nella maniera opportuna. Tipicamente il titolo con un numero minore verrà visualizzato in un carattere più grande e più spesso. Il browser può anche scegliere di usare dei colori differenti per ogni livello di titolo. Tipicamente i titoli <H1> sono grandi e in grassetto con almeno una linea bianca al di sopra e al di sotto. In contrasto, i titoli <H2> sono in un carattere più piccolo, e con minor spazio al di sopra e al di sotto.

I tag e <I> vengono usati per entrare rispettivamente in modalità grassetto e corsivo. Se il browser non è in grado di visualizzare il grassetto e il corsivo, dovrà usare qualche altro modo per la loro rappresentazione, ad esempio, usando un colore diverso per ognuno e magari invertire i colori sul video. Invece di specificare degli stili fisici come grassetto e corsivo, gli autori possono anche usare degli stili logici come <DN> (definisci), (enfasi debole), (enfasi forte) e <VAR> (variabili di programma). Gli stili logici sono definiti in un documento chiamato **foglio di stile (style sheet)**. Il vantaggio degli stili logici è che modificando una definizione, è possibile modificare tutte le variabili, ad esempio, dal corsivo a un carattere con ampiezza costante.

HTML include diversi meccanismi per creare liste, comprese le liste annidate. Il tag inizia una lista non ordinata. Ai singoli elementi, che vengono indicati nella sorgente con il tag , vengono premessi dei punti (·).

Una variante di questo meccanismo è , che serve per le liste ordinate. Quando viene usato questo tag, il browser numera gli elementi .

Una terza opzione è <MENU>, che di solito produce sullo schermo una lista più compatta, senza punti e senza numeri. A parte l'uso di tag differenti per l'inizio e la fine, , e <MENU> hanno la stessa sintassi e risultati simili.

Oltre ai meccanismi di lista mostrati in figura 7-65, ce ne sono altri due che è bene menzionare brevemente. Per le tabelle brevi si può usare <DIR>. Inoltre, <DL> e </DL> creano delle liste di definizione (glossari) con elementi composti da due parti, che sono rispettivamente definite da <DT> e <DD>. Il primo tag è per il nome, il secondo per il significato. Questi costrutti sono stati in gran parte rimpiazzati dal meccanismo delle tabelle (più generale e complesso), che descriviamo in seguito.

Tag	Descrizione
<HTML>... </HTML>	Dichiara una pagina Web da scrivere in HTML
<HEAD>... </HEAD>	Delimita l'intestazione di pagina
<TITLE>... </TITLE>	Definisce il titolo (non visualizzato nella pagina)
<BODY>... </BODY>	Delimita il corpo della pagina
<Hn>... </Hn>	Delimita il livello n del titolo
... 	Imposta... in grassetto
<I>... </I>	Imposta... in corsivo
... 	Racchiude una lista non ordinata (per punti)
... 	Racchiude una lista ordinata
<MENU>... </MENU>	Racchiude un men di elementi
	Inizia un elemento della lista (non c' è)
 	Forza in questo punto un'interruzione di linea
<P>	Inizia un paragrafo
<HR>	Riga orizzontale
<PRE>... </PRE>	Testo preformatto; da non riformattare
	Carica in questo punto un'immagine
 ... 	Definisce un iperpuntatore

Fig. 7-65 Una selezione dei tag HTML più comuni. Alcuni richiedono parametri aggiuntivi.

I tag
, <P> e <HR> denotano tutti una separazione tra parti del testo. Il formato preciso si determina col foglio di stile associato alla pagina. Il tag
 forza un'interruzione di linea. Tipicamente, i browser non inseriscono una linea bianca dopo
. Al contrario, <P> inizia un nuovo paragrafo, e può ad esempio, inserire una linea bianca e magari una indentazione (in teoria esiste </P> per terminare il paragrafo, ma viene usato raramente (molti autori HTML non sanno neppure che esista). Infine, <HR> forza un'interruzione di linea e disegna una linea orizzontale attraverso lo schermo.

HTML 1.0 non permetteva di visualizzare tabelle o altre informazioni formattate. Ancor peggio, se lo scrittore HTML formattava minuziosamente una tabella utilizzando spazi e ritorni carrello, i browser ignoravano tutta la composizione e visualizzavano la pagina come se tutto il materiale formattato non lo fosse affatto. Per impedire che i browser rovinassero il lavoro fatto, esistevano i tag <PRE> e </PRE>. Sono istruzioni al browser per visualizzare tutto ciò che sta nel mezzo così com'è, carattere per carattere, senza cambiare niente. Quando verranno meglio implementate le tabelle e altri costrutti sofisticati di formattazione, diminuirà la necessità del <PRE>, eccetto che per i listati di programmi, per i quali la maggior parte dei programmati non tollererebbe una formattazione diversa dalla propria.

HTML permette di includere delle immagini sulle linee in una pagina Web. Il tag specifica il caricamento di un'immagine nella pagina nella posizione corrente. Può avere diversi parametri. Il parametro SRC indica la URL (o l'URI) dell'immagine. Lo standard

HTML non specifica quali siano i formati grafici permessi. In pratica, tutti i browser supportano i file GIF e molti anche i file JPEG. I browser sono liberi di supportare qualsiasi formato, ma questa estensione è un'arma a doppio taglio. Se un utente è abituato a un browser che supporta, diciamo, file BMP, può includere questi file nelle sue pagine Web e più tardi stupirsi che altri browser ignorino tutta la sua meravigliosa arte.

Altri parametri di sono *ALIGN*, che controlla l'allineamento dell'immagine rispetto alla linea base del testo (*TOP*, *MIDDLE*, *BOTTOM*), *ALT* che fornisce del testo da usare al posto dell'immagine quando l'utente ha disabilitato le immagini, e *ISMAP*, un flag che indica che l'immagine è una mappa attiva.

Infine, arriviamo agli iperpuntatori, che utilizzano i tag <A> (ancora) e . Come , <A> ha diversi parametri, che comprendono, tra gli altri, *HREF* (la URL), *NAME* (il nome dell'iperpuntatore) e *METHODS* (metodi d'accesso). Viene visualizzato il testo tra <A> e . Se selezionato, l'iperpuntatore conduce a una nuova pagina. In questo punto si può anche inserire un'immagine , in questo caso per attivare l'iperpuntatore si può anche fare click sull'immagine.

Come esempio, consideriamo il seguente frammento di HTML:

 NASA's home page

Quando viene visualizzata una pagina con questo frammento, ciò che appare sullo schermo

NASA's home page

Se l'utente successivamente fa click su questo testo, immediatamente il browser recupera la pagina il cui URL è <http://www.nasa.gov> e la visualizza.

Come secondo esempio, consideriamo adesso

Quando viene visualizzata, questa pagina mostra una figura (in questo caso, lo space shuttle). Cliccando sull'immagine si passa alla pagina principale, proprio come cliccando sul testo sottolineato come nell'esempio precedente. Se l'utente ha disabilitato la visualizzazione automatica delle immagini, verrà visualizzato il testo NASA al posto dell'immagine.

Il tag <A> può avere un parametro *NAME* per fissare un iperpuntatore, in modo che possa essere riferito dall'interno della pagina. Ad esempio, alcune pagine Web iniziano con un indice "cliccabile". Facendo click su un elemento dell'indice, l'utente salta al corrispondente paragrafo della pagina.

Una funzione non inclusa in HTML 2.0 e che invece sarebbe stata necessaria a molti autori di pagine è la creazione di tabelle aventi per elementi iperpuntatori attivi. Di conseguenza, è stato fatto un grosso sforzo per aggiungere tali tabelle in HTML 3.0. Di seguito diamo una breve introduzione delle tabelle, giusto per averne un'idea.

Una tabella HTML consiste di una o più righe, ognuna composta da una o più celle. Le celle possono contenere una grande varietà di materiale, compreso testo, figure e anche altre tabelle. È possibile raggruppare le celle, così ad esempio un titolo può comprendere

più colonne. Gli autori di pagine hanno un certo controllo sull'aspetto finale, compreso l'allineamento, lo stile dei bordi e i margini delle celle, comunque il browser ha la parola finale nella rappresentazione delle tabelle.

```
<HTML> <HEAD> <TITLE> A sample page with a table </TITLE> </HEAD>
<BODY>
<TABLE BORDER=ALL RULES=ALL>
<CAPTION> Some Differences between HTML Versions </CAPTION>
<COL ALIGN=LEFT>
<COL ALIGN=CENTER>
<COL ALIGN=CENTER>
<COL ALIGN=CENTER>
<TR> <TH>Item <TH>HTML 1.0 <TH>HTML 2.0 <TH>HTML 3.0
<TR> <TH> Active Maps and Images <TD> <TD> x <TD> x
<TR> <TH> Equations <TD> <TD> <TD> x
<TR> <TH> Forms <TD> <TD> x <TD> x
<TR> <TH> Hyperlinks x <TD> <TD> x <TD> x
<TR> <TH> Images <TD> x <TD> x <TD> x
<TR> <TH> Lists <TD> x <TD> x <TD> x
<TR> <TH> Toolbars <TD> <TD> <TD> x
<TR> <TH> Tables <TD> <TD> <TD> x
</TABLE> </BODY> </HTML>
```

(a)

Some Differences between HTML Versions

Item	HTML 1.0	HTML 2.0	HTML 3.0
Active Maps and Images		x	x
Equations			x
Forms		x	x
Hyperlinks	x	x	x
Images	x	x	x
Lists	x	x	x
Toolbars			x
Tables			x

(b)

Fig. 7-66 (a) Una tabella HTML. (b) Una sua possibile visualizzazione.

In figura 7-66 (a) vediamo la definizione di una tabella HTML e in figura 7-66 (b) una possibile visualizzazione. Questo esempio vuole mostrare in maniera semplice alcune delle caratteristiche di base delle tabelle HTML. Le tabelle iniziano con il tag **<TABLE>**. È possibile dare altre informazioni per descrivere le proprietà generali delle tabelle. Il tag **<CAPTION>** si usa per assegnare una didascalia a una figura. Ogni riga inizia con un tag **<TR>** (riga di tabella). Le singole celle sono individuate da **<TD>** (titolo di tabella)

o **<TD>** (dati di tabella). La distinzione è fatta per permettere ai browser di usare per essi una rappresentazione differente, come abbiamo fatto nell'esempio.

Per le tabelle esistono numerosi altri tag, come ad esempio per specificare l'allineamento orizzontale e verticale di celle, la giustificazione all'interno di una cella, bordi, raggruppamenti di celle, unità e altro ancora.

Forms (moduli)

I documenti scritti in HTML 1.0 erano in sostanza a senso unico. Gli utenti potevano richiamare pagine da fornitori di informazione, ma era difficile trasmettere dati nell'altro senso. Quando molte organizzazioni commerciali iniziarono a usare il Web, nacque una forte domanda per traffico a due sensi. Ad esempio, molte società volevano raccogliere ordini per i loro prodotti via pagine Web; i venditori di software volevano distribuire i programmi ordinati mediante moduli elettronici riempiti direttamente dai clienti, e le società che offrivano motori di ricerca volevano che i loro clienti potessero inviare chiavi di ricerca.

Queste richieste portarono all'inclusione in HTML 2.0 del costrutto delle **forms** (moduli). Una form contiene scatole o pulsanti che permettono agli utenti di inserirvi informazioni o di fare delle scelte e quindi di rimandare le informazioni al proprietario della pagina. A questo scopo utilizzano il tag **<INPUT>**. Questo ha una varietà di parametri per determinare la dimensione, la natura e l'utilizzo della scatola visualizzata. Le form più comuni sono dei campi bianchi per accettare del testo dell'utente, quadretti per la scelta di un'opzione, mappe attive e pulsanti tipo SUBMIT. L'esempio di figura 7-67 illustra qualcuna di queste scelte.

Iniziamo il nostro discorso sulle form analizzando quest'esempio. Come per tutte le form, questa è racchiusa tra i tag **<FORM>** e **</FORM>**. Il testo non racchiuso in un tag è solamente visualizzato. In una form sono ammessi tutti i tag usuali (come ****). In questa form sono usati tre tipi di scatole di input.

Il primo tipo di scatola di input segue il testo "Nome". La scatola è ampia 46 caratteri e si aspetta che l'utente vi immetta una stringa, che viene quindi memorizzata nella variabile *customer* per essere elaborata in seguito. Il tag **<P>** dice al browser di visualizzare il testo e le scatole che seguono sulla linea successiva, anche se sulla linea corrente c'è ancora spazio. Utilizzando **<P>** e gli altri tag di visualizzazione, l'autore della pagina può controllare l'aspetto della form sullo schermo.

La linea successiva della form chiede la via dell'indirizzo dell'utente, su 40 colonne, anche questo su una linea a sé. Quindi segue una linea per richiedere la città, lo stato e il paese. Tra questi campi non viene usato alcun tag **<P>**, perciò il browser li visualizzerà tutti su una linea se ci stanno. Per quanto riguarda il browser, questa linea contiene sei elementi: tre stringhe che si alternano con tre scatole. Li visualizza linearmente da sinistra a destra, andando a riga nuova se la linea corrente non può contenere l'elemento successivo. Così è probabile che su uno schermo 1024 × 768 tutte e tre le stringhe e le scatole corrispondenti appaiano sulla stessa linea, mentre su uno schermo 640 × 480 si possono suddividere su due linee. Nel caso peggiore, la parola "Country" è alla fine di una linea e la sua scatola è all'inizio della linea successiva. Non c'è modo di ordinare al browser di forzare l'adiacenza tra scatola e testo.

```
<HTML> <HEAD> <TITLE> AWI CUSTOMER ORDERING FORM </TITLE> </HEAD>
<BODY>
<H1> Widget Order Form </H1>
<FORM ACTION="http://widget.com/cgi-bin/widgetorder" METHOD=POST>
Name <INPUT NAME="customer" SIZE=46> <P>
Street Address <INPUT NAME="address" SIZE=40> <P>
City <INPUT NAME="city" SIZE=20> State <INPUT NAME="state" SIZE =4>
Country <INPUT NAME="country" SIZE=10> <P>
Credit card # <INPUT NAME="cardno" SIZE=10>
Expires <INPUT NAME="expires" SIZE=4>
M/C <INPUT NAME="cc" TYPE=RADIO VALUE="mastercard">
VISA <INPUT NAME="cc" TYPE=RADIO VALUE="visacard"> <P>
Widget size Big <INPUT NAME="product" TYPE=RADIO VALUE="expensive">
Little <INPUT NAME="product" TYPE=RADIO VALUE="cheap">
Ship by express courier <INPUT NAME="express" TYPE=CHECKBOX> <P>
<INPUT TYPE=SUBMIT VALUE="Submit order"> <P>
Thank you for ordering an AWI widget, the best widget money can buy!
</FORM> </BODY> </HTML>
```

(a)

Widget Order Form

Name

Street address

City State Country

Credit card # Expires M/C Visa

Widget size Big Little Ship by express courier

Thank you for ordering an AWI widget, the best widget money can buy!

(b)

Fig. 7-67 (a) Codice HTML di un modulo d'ordine. (b) La corrispondente pagina formattata.

La linea successiva richiede il numero di carta di credito e la data di scadenza. La trasmissione dei numeri di carta di credito tramite Internet dovrebbe essere fatta solamente quando siano state prese adeguate misure di sicurezza. Ad esempio, alcuni browser, ma non tutti, cifrano le informazioni inviate dall'utente. Anche in questo caso, la comunicazione sicura e la gestione delle chiavi sono materia complicata e sono soggette a molti tipi di attacchi, come quelli già visti.

Dopo la data di scadenza incontriamo un nuovo costrutto: i bottoni radio. Questi vanno usati quando si deve scegliere tra due o più alternative. Il modello ispiratore è la radio di un'automobile con una mezza dozzina di bottoni per la scelta delle stazioni. Il browser visualizza queste scatole in una form che permetta all'utente di selezionarle e deselezionarle facendo click su di esse (oppure utilizzando la tastiera). Facendo click su una si annullano tutte le altre nello stesso gruppo. La presentazione visuale dipende dall'interfaccia grafica utilizzata. È compito del browser scegliere una form che sia consistente con Windows, Motif, OS/2 Warp o qualsiasi altro sistema di interfaccia a finestre. Anche la dimensione del widget utilizza i bottoni radio. I due gruppi sono distinti dal loro campo NAME, non dallo scoping statico usando qualcosa come <RADIOBUTTON> ... </RADIOBUTTON>.

I parametri VALUE sono usati per indicare quali bottoni radio siano stati premuti. A seconda di quale opzione tra le carte di credito sceglie l'utente, la variabile cc viene impostata o con la stringa "mastercard" o con la stringa "visacard".

Dopo i due insiemi di bottoni radio, arriviamo all'opzione di spedizione, rappresentata da una scatola di tipo CHECKBOX. Può essere selezionata o meno. Diversamente dai bottoni radio, dove deve essere scelto esattamente uno tra tutti quelli dell'insieme, ogni box di tipo CHECKBOX può essere scelto o non scelto, indipendentemente da tutti gli altri. Ad esempio, ordinando una pizza attraverso la pagina Web di Electropizza, l'utente può scegliere sardine e cipolle e ananas (se gli piace), ma non può scegliere piccola e media e grande per la stessa pizza. I condimenti per la pizza dovrebbero essere rappresentati da tre scatole separate di tipo CHECKBOX, mentre la dimensione della pizza dovrebbe essere un insieme di bottoni radio.

Tra parentesi, se le liste su cui fare una scelta sono molto lunghe, i bottoni radio sono scomodi. A questo scopo, esistono i tag <SELECT> e </SELECT> per racchiudere una lista di alternative, ma con la semantica dei bottoni radio (almeno che non sia presente il parametro MULTIPLE, nel qual caso la semantica è quella delle scatole checkbox). Alcuni browser rappresentano gli elementi tra <SELECT> e </SELECT> con un menù a scomparsa.

Abbiamo a questo punto visto due tipi nativi per il tag <INPUT>: RADIO e CHECKBOX. In effetti, abbiamo già visto anche il terzo: TEXT. Dato che questo è il tipo di default, non dobbiamo preoccuparci di includere il parametro TYPE = TEXT, ma potremmo farlo. Ci sono altri due tipi, PASSWORD e TEXTAREA. Una scatola PASSWORD è uguale a una scatola TEXT, eccetto che i caratteri digitati non vengono visualizzati. Una scatola TEXTAREA è ancora lo stesso di una scatola TEXT, eccetto che può contenere più linee.

Tornando indietro all'esempio di figura 7-67, vediamo un esempio di bottone SUBMIT. Quando questo viene "cliccato", l'informazione dell'utente sulla form viene rispedita alla macchina che ha fornito la form. Come per tutti gli altri tipi, SUBMIT è una parola riservata riconosciuta dal browser. La stringa VALUE qui è l'etichetta del bottone e viene visualizzata. Tutte le scatole possono avere dei valori; noi ne avevamo bisogno solamente qui. Per le scatole TEXT, il contenuto del campo VALUE viene visualizzato assieme alla form, ma l'utente può editarlo o cancellarlo. È possibile anche inizializzare le scatole

CHECKBOX e *RADIO*, ma mediante un campo di nome *CHECKED* (perché *VALUE* dà semplicemente il testo, ma non indica la scelta preferenziale).

Il browser inoltre riconosce il bottone *RESET*. Quando viene cliccato, riporta la form nel suo stato iniziale.

Dobbiamo menzionare altri due tipi. Il primo è *HIDDEN*. Questo è solamente di uscita; non può essere "cliccato" o modificato. Ad esempio, quando si lavora con una serie di pagine nelle quali vanno fatte diverse scelte, le scelte fatte in precedenza possono essere di tipo *HIDDEN*, per impedirne la modifica.

L'ultimo tipo è *IMAGE*, che serve per le mappe attive (e altre immagini "cliccibili"). Quando l'utente fa click sulla mappa, le coordinate (*x*, *y*) del pixel selezionato (cioè la posizione corrente del mouse) vengono memorizzate in alcune variabili e la form viene immediatamente restituita al proprietario per ulteriori elaborazioni.

Le form si possono sottomettere in tre modi: mediante il bottone di sottomissione, "cliccando" su una mappa attiva o battendo ENTER su una form costituita da un elemento di tipo *TEXT*. Al momento della sottomissione di una form, vengono intraprese delle azioni. L'azione è specificata dai parametri del tag <FORM>. Il parametro *ACTION* specifica la URL (o l'URI) da avvisare per la sottomissione, e il parametro *METHOD* indica quale metodo usare. L'ordine di questi parametri (e di tutti gli altri) non è significativo.

Il modo in cui le variabili della form vengono spedite indietro al proprietario della pagina dipende dal valore del parametro *METHOD*. Per *GET*, il solo modo per restituire dei valori è conversazionale: sono appesi alla URL, separati da un punto di domanda. Questo approccio può portare ad avere URL lunghe migliaia di caratteri. Nonostante ciò, questo metodo è usato di frequente perché è semplice.

Se viene usato il metodo *POST* (vedi fig. 7-62), il corpo del messaggio contiene le variabili della form e i loro valori. Il carattere & viene usato per separare i campi; il + rappresenta il carattere spazio. Ad esempio, la risposta alla form widget potrebbe essere

```
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
```

La stringa si potrebbe rispedire al server in una sola linea, non in tre. Se non viene selezionata una *CHECKBOX*, questa è omessa dalla stringa. È compito del server dare un significato a tale stringa.

Fortunatamente, è già disponibile uno standard per la manipolazione dei dati delle form. Si chiama **CGI (Common Gateway Interface)**. Consideriamo un modo comune per utilizzare CGI. Supponiamo che qualcuno abbia una base di dati interessante (ad es., un indice delle pagine Web per parole chiave e argomento) e voglia metterla a disposizione degli utenti del Web. Il modo in cui la CGI rende disponibile la base di dati è mediante uno script (o programma) di interfaccia (cioè un gateway) tra la base di dati e il Web. Questo programma è associato a una URL, per convenzione nella directory *cgi-bin*. I server HTTP sanno (o si può far sapere loro) che quando devono invocare un metodo su una pagina che si trova in *cgi-bin*, devono interpretare il file come se fosse uno script eseguibile e avviarlo.

Supponiamo che un certo utente apra la form associata col nostro script widget e la

visualizzi. Dopo la compilazione della form, l'utente fa click sul bottone *SUBMIT*. Questa azione fa sì che il browser stabilisca una connessione TCP verso l'URL presente nel parametro *ACTION* della form – il programma nel directory *cgi-bin*. Quindi il browser invoca l'operazione specificata dal parametro *METHOD* della form, di solito *POST*. Il risultato di questa operazione consiste nell'avvio del programma, che viene presentato (attraverso la connessione TCP, sull'ingresso standard) con la stringa lunga vista prima. Inoltre, vengono impostate diverse variabili d'ambiente. Ad esempio, la variabile d'ambiente *CONTENT_LENGTH* indica la lunghezza della stringa in ingresso.

A questo punto, molti script hanno bisogno di elaborare i loro ingressi per porli in una forma più conveniente. Questo obiettivo può essere raggiunto richiamando una delle molte librerie o procedure script disponibili. Lo script può quindi interagire con la propria base di dati nella maniera desiderata. Ad esempio, le mappe attive normalmente utilizzano degli script CGI per intraprendere azioni diverse a seconda di dove l'utente punta.

Gli script CGI inoltre possono produrre delle uscite e fare molte altre cose, ad esempio accettare degli ingressi dalle form. Se un iperpuntatore punta a uno script CGI, quando quell'iperpuntatore viene invocato, viene avviato lo script, con le diverse variabili d'ambiente impostate in modo da fornire alcune informazioni riguardo all'utente. Lo script quindi scrive un file (ad es., una pagina HTML) sull'uscita standard, file che viene quindi inviato al browser e da questo interpretato. Con questo meccanismo lo script può generare delle pagine Web personalizzate al volo.

Nella buona o nella cattiva sorte, alcuni siti Web che rispondono a domande hanno una lista di pubblicità che può essere inserita in maniera selettiva nella pagina Web in costruzione, a seconda di ciò che l'utente sta cercando. Ad esempio, se l'utente sta cercando una "automobile" può essere visualizzata una pubblicità della General Motors, mentre se ricerca "vacanza" può ottenere una pubblicità della United Airlines. Queste pubblicità di solito comprendono testo e immagini "cliccibili".

7.6.4 Java

Con HTML è possibile descrivere il modo in cui appaiono pagine statiche, comprese le tabelle e le immagini. Con il meccanismo cgi-bin è anche possibile avere una limitata interazione a due vie (form ecc.). Comunque, non è possibile un'interazione rapida con pagine scritte in HTML. Perché sia possibile avere delle pagine altamente interattive, è necessario un meccanismo differente. In questo paragrafo descriveremo un meccanismo del genere, il linguaggio Java e la sua macchina astratta.

Java è nato quando alcune persone della Sun Microsystems decisero di sviluppare un nuovo linguaggio che fosse adatto a programmare degli elettrodomestici informatizzati. Più tardi fu reindirizzato verso il World Wide Web. Sebbene Java derivi molte idee e parte della sintassi dal C e C++, è un linguaggio orientato agli oggetti del tutto originale, incompatibile con entrambi. Qualche volta si dice che "in the large" Java è simile allo Smalltalk, ma che "in the small" è come il C o il C++.

L'idea principale nell'utilizzare Java per le pagine interattive è che una pagina Web può puntare a un piccolo programma Java, detto **applet**. Quando il browser lo raggiunge, l'applet viene caricato sulla macchina utente e qui eseguito in un modo sicuro. Sicuro vuol dire che deve essere strutturalmente impossibile per un applet leggere o scrivere

qualsiasi file che non sia autorizzato ad accedere. Deve inoltre essere impossibile per l'applet introdurre dei virus o provocare un qualsiasi altro danno. Per queste ragioni, e per conseguire portabilità su più piattaforme, gli applet vengono compilati in un codice intermedio (bytecode) dopo essere stati scritti e verificati. Sono questi programmi in bytecode che vengono puntati dalle pagine Web, in modo simile a come vengono puntate le immagini. Quando arriva, l'applet viene interpretato in un ambiente sicuro.

Prima di entrare nei dettagli del linguaggio Java, è bene spendere alcune parole sulla sua utilità e sul perché gli applet Java vengano inclusi in pagine Web. Per prima cosa, gli applet consentono alle pagine Web di divenire interattive. Ad esempio, una pagina Web può contenere una scacchiera per il tic tac toe, l'otello o gli scacchi e giocare una partita con l'utente. Il programma che gioca (scritto in Java) viene caricato insieme alla sua pagina Web. Come secondo esempio, è possibile visualizzare delle form complesse (come i fogli elettronici), in cui l'utente compila i campi e vede istantaneamente i calcoli impostati.

È sicuramente possibile che alla lunga il modello per cui gli utenti comprano programmi, li installano e li eseguono localmente verrà rimpiazzato da un modello in cui gli utenti fanno click sulle pagine Web, e carcano degli applet perché eseguano un determinato compito, probabilmente in combinazione con un server remoto o una base di dati. Invece di compilare manualmente il modello delle tasse o mediante uno speciale programma, i contribuenti potrebbero fare click sulla pagina principale dell'ufficio tributi per caricare l'applet delle tasse. Questo applet potrebbe porre alcune domande, quindi contattare il datore di lavoro, la banca e l'agente di cambio del contribuente per raccogliere informazioni su stipendio, interessi e dividendi, compilare il modello delle tasse e quindi visualizzarlo per una verifica e per la sottomissione.

Un'altra ragione per eseguire degli applet sulla macchina cliente è che rendono possibile aggiungere animazione e suono alle pagine Web senza dover eseguire dei programmi di aiuto esterni. Il suono proviene riprodotto al momento del caricamento della pagina, come musica di sottofondo, o quando accade qualche specifico evento (ad es., facendo click sul gatto questo miagola). Lo stesso vale per l'animazione. Dato che l'applet gira localmente, anche se deve essere interpretato, può riscrivere su tutto lo schermo (o una sua parte) in qualsiasi modo e a una velocità elevata (almeno in confronto allo script shell cgi-bin remoto).

Il sistema Java ha tre parti:

1. Un compilatore Java che genera codice intermedio bytecode.
2. Un browser che riconosce gli applet.
3. Un interprete di bytecode.

Lo sviluppatore scrive l'applet in Java, quindi lo compila in bytecode. Per includerlo compilato in una pagina Web, è stato inventato un nuovo tag HTML, <APPLET>. Un uso tipico

```
<APPLET CODE=game.class WIDTH=100 HEIGHT=200> </APPLET>
```

Quando il browser vede il tag <APPLET>, recupera l'applet compilato *game.class* dal sito della pagina Web (oppure, se è presente un altro parametro, *CODEBASE*, dalla URL

7.6 Il World Wide Web

che specifica). Il browser quindi passa l'applet all'interprete locale di bytecode (oppure interpreta l'applet da solo, se possiede un interprete interno). I parametri *WIDTH* e *HEIGHT* danno la dimensione della finestra di default dell'applet, in pixel.

In un certo senso, il tag <APPLET> è simile al tag . In entrambi i casi, il browser va a prendere un file e quindi lo sottopone a un interprete (possibilmente interno) per visualizzarlo in una determinata area dello schermo. Quindi continua a elaborare la pagina Web. Se serve, nel caso di applicazioni che hanno bisogno di prestazioni molto alte, alcuni interpreti Java sono in grado di compilare al volo i programmi bytecode in linguaggio macchina nativo.

Di conseguenza, i browser basati su Java sono estensibili in modo molto diverso dai browser della prima generazione. I browser della prima generazione sono essenzialmente degli interpreti HTML che hanno dei moduli nativi per parlare al bisogno, il linguaggio dei vari protocolli, come per HTTP 1.0, FTP ecc., così come i decodificatori per vari formati di immagini. La figura 7-68 (a) mostra un esempio. Se qualcuno inventa o reclamizza un nuovo formato, di tipo audio o MPEG-2, questi vecchi browser non sono in grado di leggere le pagine che li contengono. Alla meglio, l'utente deve trovare, caricare e installare un opportuno visualizzatore esterno.

Con un browser basato su Java, la situazione è differente. Al momento dell'avvio, il browser è effettivamente una macchina virtuale Java vuota, come mostra la figura 7-68 (b). Caricando gli applet HTML e HTTP, diventa capace di leggere le pagine Web standard. Comunque, quando sono necessari dei nuovi protocolli o decodificatori, le loro classi vengono caricate dinamicamente, magari via rete dai siti specificati nelle pagine Web. Dopo un po', il browser somiglierà a quello di figura 7-68 (c).

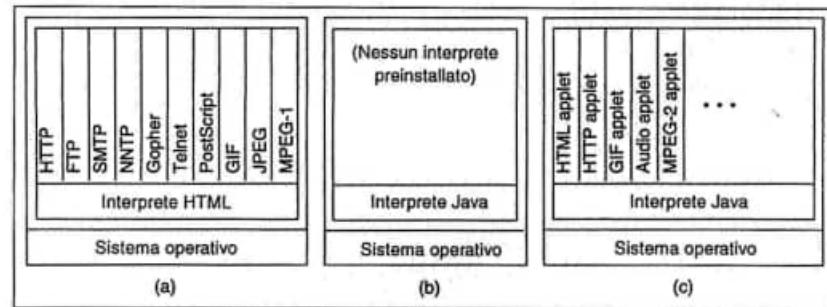


Fig. 7-68 (a) Un browser di prima generazione. (b) Un browser basato su Java all'avvio. (c) Il browser di (b) dopo un po' che è in esecuzione.

In questo modo se qualcuno inventasse un nuovo formato, tutto ciò che andrebbe fatto sarebbe di includere la URL di un applet per la gestione in una pagina Web, e il browser automaticamente recupererà e caricherà l'applet. Nessun browser di prima generazione è in grado di caricare e installare automaticamente al volo dei nuovi visualizzatori. La capacità di caricare dinamicamente gli applet significa che le persone possono sperimentare con facilità i nuovi formati senza dover prima trovare un accordo in riunioni senza fine.

Questa estensibilità si applica anche ai protocolli. Per alcune applicazioni sono necessari dei protocolli speciali, ad esempio dei protocolli sicuri per le banche e per il commercio. Con Java, è possibile caricare dinamicamente questi protocolli al momento del bisogno, senza alcuna necessità di raggiungere una standardizzazione universale. Per comunicare con la società X, è sufficiente recuperare il suo applet per il protocollo. Per parlare con la società Y, è necessario recuperare il suo applet per il protocollo. Non c'è alcun bisogno che X e Y si accordino su un protocollo standard.

Introduzione al linguaggio Java

Gli obiettivi elencati sopra hanno portato a un linguaggio con tipizzazione sicura, orientato agli oggetti concorrenti (multithreading) e nessuna caratteristica indefinita o dipendente dal sistema. Quella che segue è una descrizione molto semplificata di Java, giusto per darne un'idea. Per brevità sono stati omessi molti dettagli, funzioni, opzioni e casi speciali. La specifica completa del linguaggio, e molto altro ancora, è disponibile attraverso lo stesso Web (naturalmente) a <http://java.sun.com>. Per delle introduzioni a Java, si vedano Campione, Walrath (1996); Van der Linden (1996). Per l'intera storia, si vedano Arnold, Gosling (1996); Gosling *et al.* (1996). Per un breve confronto tra Java e la risposta della Microsoft (Blackbird), si veda Singleton (1996).

Come abbiamo già detto, Java è, per la programmazione in piccolo (algoritmica), simile al C e al C++. Le regole lessicali, ad esempio, sono più o meno le stesse (ad es., i token sono delimitati dallo spazio bianco, ed è possibile inserire delle linee nuove tra due token qualsiasi). È possibile inserire ovunque dei commenti usando sia la sintassi del C (`/* ... */`) che la sintassi del C++ (`// ...`).

Java ha otto tipi di dato primitivi, elencati in figura 7-69. Ogni tipo ha una dimensione specifica, indipendente dall'implementazione locale. A differenza del C, dove un intero può essere di 16, 32 o 64 bit, a seconda dell'architettura della macchina sottostante, un *int* Java è sempre di 32 bit, nè più nè meno, indipendentemente dal tipo di macchina su cui l'interprete sta girando. Questa consistenza è essenziale dato che lo stesso applet deve girare su macchine a 16, 32 o 64 bit, e dare su tutte lo stesso risultato.

Tipo	Dimensione	Descrizione
Byte	1 byte	Un intero con segno tra - 128 e +127
Short	2 byte	Un intero con segno di 2 byte
Int	4 byte	Un intero con segno di 4 byte
Long	8 byte	Un intero con segno di 8 byte
Float	4 byte	Un numero in virgola mobile IEEE di 4 byte
Double	8 byte	Un numero in virgola mobile IEEE di 8 byte
Boolean	1 bit	Gli unici sono vero e falso
Char	2 byte	Un carattere in Unicode

Fig. 7-69 I tipi base di Java.

È possibile combinare tra loro le variabili aritmetiche (i primi 6 tipi) mediante le usuali operazioni aritmetiche (compreso `++` e `- -`) e confrontarle mediante gli usuali operatori relazionali (ad es., `<`, `<=`, `==`, `!=`). Le conversioni tra tipi sono permesse là dove hanno senso.

Java usa per i caratteri il codice Unicode a 16 bit invece di quello ASCII, per cui le variabili di tipo carattere sono lunghe 2 byte. I primi 127 caratteri Unicode sono gli stessi dell'ASCII per la compatibilità retrospettiva. Oltre a questi ci sono i simboli grafici e quindi i caratteri necessari per russo, arabo, ebraico, giapponese (kanji, katakana e hiragana), e virtualmente qualsiasi altra lingua. I caratteri non presenti nell'ASCII si possono rappresentare con una sequenza \u seguita da quattro cifre esadecimali. Ad esempio, \u0ae6 è lo 0 gujarati.

Java permette la dichiarazione di array unidimensionali. Ad esempio:

```
int [] table;
```

dichiara un array, *table*, ma non alloca per esso dello spazio. Questo può essere fatto in seguito, come nel C++, ad esempio, con

```
table = new int [1024];
```

per allocare un array con 1024 elementi. Non è necessario (né possibile) restituire array più lunghi del necessario; il garbage collector li reclama. In questo modo non sono necessarie per la gestione della memoria le routine di libreria *malloc* e *free*, che di solito producono molti errori di programmazione. È possibile inizializzare gli array e usare gli array di array per ottenere una dimensionalità maggiore, come in C. Sono disponibili le stringhe, ma sono definite in una classe, piuttosto che essere dei semplici array di caratteri che terminano con un byte nullo.

La figura 7-70 mostra le istruzioni di controllo Java. Le prime nove hanno essenzialmente la stessa sintassi e semantica del C, tranne che, dove è richiesta un'espressione booleana, il linguaggio insiste che si usi una vera espressione booleana. Inoltre, le istruzioni *break* e *continue* in Java accettano delle etichette per indicare quel è il ciclo dal quale uscire o da ripetere.

Le due istruzioni successive sono in C++, non in C. Le istruzioni *throw* e *try* si occupano della gestione delle eccezioni. Java definisce una varietà di eccezioni standard, come il tentativo di dividere per 0, e permette ai programmati di definire e segnalare le loro personali eccezioni. I programmati possono scrivere le procedure di gestione delle eccezioni, rendendo così inutile verificare costantemente che qualcosa sta andando male (ad esempio, in fase di lettura da un file). L'istruzione *throw* segnala un'eccezione, e l'istruzione *try* definisce l'associazione tra la procedura di gestione dell'eccezione e un blocco di codice nel quale l'eccezione può avvenire.

L'istruzione *synchronized* è originale di Java e ha a che fare con i programmi concorrenti. Per evitare condizioni di conflitto, viene usata questa istruzione per delimitare un blocco di codice (o un'intera procedura) in cui è attivo un solo contatore programma alla volta. Questi blocchi di codice di solito vengono detti **regioni critiche**. Quando viene

eseguita l'istruzione `synchronized`, il contatore programma che la esegue deve acquisire il lock associato con la regione critica, eseguire il codice, e quindi rilasciare il lock. Se il lock non è disponibile, il contatore programma attende che sia libero (cioè si sospende). Mettendo delle guardie di questo tipo alle procedure e utilizzando le variabili di condizione, i programmati hanno tutto il potere dei monitor (Hoare, 1974).

Istruzione	Descrizione	Esempio
Assegnamento	Assegna un valore	<code>n = i + j;</code>
If	Scelta booleana	<code>if (k < 0) k = 0; else k = 2*k;</code>
Switch	Selezione un caso	<code>switch (b) {case 1: n++; case 2: n--;}</code>
For	Iterazione	<code>for (i = 0; i < n; i++) a [i] = b [i];</code>
While	Ripetizione	<code>while (n < k) n += i;</code>
Do	Ripetizione	<code>do {n = n + n} while (n < m);</code>
Break	Istruzione di uscita	<code>break etichetta;</code>
Return	Ritorno	<code>return n;</code>
Continue	Prossima iterazione	<code>continue etichetta;</code>
Throw	Solleva un'eccezione	<code>throw new IllegalArgumentException();</code>
Try	Scoping dell'eccezione	<code>try { ... } catch (Exception e) {return -1;};</code>
Synchronized	Mutua esclusione	<code>synchronized void update (int s) { ... }</code>

Fig. 7-70 Le istruzioni Java. La notazione `{ ... }` indica un blocco di codice.

È possibile invocare un programma Java con argomenti. L'elaborazione della linea di comando è simile a quella del C, tranne che l'array degli argomenti è chiamato `args` invece di `argv` e che `args[0]` è il primo parametro, non il nome del programma. La figura 7-71 illustra il calcolo di una tabella di fattoriali, giusto per dare l'idea di un breve programma Java.

Anche se sono entrambi linguaggi orientati agli oggetti che si basano sul C, Java e C++ differiscono in diversi modi. Da Java sono state rimossi alcuni costrutti per renderlo sicuro rispetto ai tipi e più leggibile. Le rimozioni comprendono `#define`, `typedef`, `enum`, `union`, `struct`, gli operatori di overloading, i puntatori esplicativi, le variabili globali, le funzioni e le funzioni "amiche". Va da sé che l'istruzione `goto` sia stata spedita nel posto speciale riservato ai costrutti obsoleti dei linguaggi di programmazione. Sono stati aggiunti altri costrutti per dare più potere al linguaggio. Le aggiunte comprendono il garbage collection, la concorrenza, le interfacce degli oggetti e la modularità (packages).

Oggetti Java

Nei tradizionali linguaggi procedurali come Pascal o C, un programma consiste in una collezione di variabili e procedure, senza alcun principio generale di organizzazione. Al contrario nei linguaggi orientati agli oggetti, (quasi) tutto è un oggetto. Un oggetto normalmente contiene alcune variabili interne (cioè nascoste) di stato, insieme ad alcune procedure pubbliche, dette metodi, per accedere a tali variabili. I programmi che utiliz-

```
class Factorial /* This program consists of a single class with two methods. */

public static void main (int argc, String args[]) { // main program
    long i, f, lower = 1, upper = 20; // declarations of four longs

    for (i = lower; i <= upper; i++) { // loop from lower to upper
        f = factorial(i); // f = i!
        System.out.println(i + " " + f); // print i and f
    }
}

static long factorial (long k) { // recursive factorial function
    if (k == 0)
        return 1; // 0! = 1
    else
        return k * factorial(k-1); // k! = k * (k-1)!)
}
```

Fig. 7-71 Un programma Java per calcolare e stampare da 0! a 20!.

zano l'oggetto di solito invocano i metodi per manipolare lo stato dell'oggetto (o sono costretti a farlo). In questo modo, chi scrive l'oggetto può controllare il modo in cui i programmi usano le informazioni all'interno degli oggetti. Questo principio è chiamato **incapsulazione**, ed è la base di tutta la programmazione orientata agli oggetti.

Java cerca di prendere il meglio di entrambi i mondi. Si può usare come un tradizionale linguaggio procedurale oppure come un linguaggio orientato agli oggetti. Infatti l'esempio di figura 7-71 si poteva scrivere altrettanto bene in C, in sostanza persino allo stesso modo. Comunque, per scrivere delle pagine Web, Java dev'essere considerato un linguaggio a oggetti, perciò in questo paragrafo studieremo la sua orientazione agli oggetti.

Un programma Java consiste di uno o più **package**, ognuno dei quali contiene alcune definizioni di classi. È possibile accedere in maniera remota ai package tramite la rete, perciò i package che si pensa avranno una grande audience è bene che abbiano un nome unico. Di solito, vengono usati dei nomi gerarchici, che iniziano con il nome DNS della propria macchina scritto a rovescio, ad esempio

EDU.univ.cs.catie.games.chess

Una definizione di **classe** è la descrizione della struttura delle istanze di oggetto, ognuna delle quali contiene le stesse variabili di stato e gli stessi metodi di tutte le altre istanze d'oggetto della sua classe. I valori delle variabili di stato all'interno di oggetti differenti sono, ovviamente, indipendenti. Le classi sono come gli stampi per i biscotti: non sono di per sé biscotti, ma vengono usati per ottenere biscotti della stessa forma, e ogni stampo

diverso produce biscotti in una forma diversa. Una volta fatti, i diversi biscotti (oggetti) sono indipendenti l'uno dall'altro.

È possibile produrre dinamicamente durante l'esecuzione gli oggetti Java, ad esempio con

```
object = new ClassName()
```

Questi oggetti vengono memorizzati nello heap e rimossi dal garbage collector quando non sono più necessari. In questo modo, la gestione della memoria in Java è gestita dal sistema, senza che siano necessarie a tale scopo le procedure *malloc* e *free*, e neppure i puntatori esplicativi.

Ogni classe si basa su un'altra classe. Una nuova classe definita è detta **sottoclasse** della classe su cui si basa, a sua volta detta **superclasse**. Una (sotto) classe eredita sempre i metodi della sua superclasse. Può avere o non avere l'accesso diretto alle variabili interne della sua superclasse, a seconda del fatto che la superclasse lo consenta o no. Ad esempio, se una superclasse *A* ha i metodi *M1*, *M2* e *M3* e una sottoclasse *B* definisce un nuovo metodo *M4*, allora gli oggetti creati da *B* avranno i metodi *M1*, *M2*, *M3* e *M4*. La proprietà per cui una classe eredita automaticamente tutti i metodi della sua superclasse è detta **ereditarietà** ed è una proprietà importante in Java. L'aggiungere nuovi metodi ai metodi della superclasse è detto **estensione** di una superclasse. Tra l'altro, alcuni linguaggi orientati agli oggetti consentono alle classi di ereditare i metodi da due o più superclassi (ereditarietà multipla), ma i progettisti di Java hanno pensato che questa proprietà creasse confusione e l'hanno lasciata intenzionalmente fuori.

Dato che ogni classe ha esattamente una superclasse immediata, in Java l'insieme di tutte le classi forma un albero. La classe radice dell'albero è detta **Object**. Tutte le altre classi ereditano i suoi metodi. Qualsiasi classe che nella sua definizione non abbia identificata in maniera esplicita la superclasse per default sarà una sottoclasse della classe *Object*. Per questo motivo, ad esempio, la classe *Fattoriale* di figura 7-71 è una sottoclasse di *Object*. Vediamo adesso un esempio riguardante i concetti di orientazione agli oggetti presentati fino a questo momento. In figura 7-72 abbiamo un package che definisce due classi, *NumeriComplessi*, per definire e utilizzare i numeri complessi (cioè i numeri con una parte reale e una parte immaginaria) e *test*, per mostrare come usare i *NumeriComplessi*. Come per *Fattoriale*, la classe *NumeriComplessi* si basa su *Object*, dato che nella sua definizione non viene nominata nessun'altra superclasse. Ogni oggetto della classe *NumeriComplessi* rappresenta un numero complesso. Ogni oggetto di questa classe contiene due variabili nascoste, *re* e *im*, entrambe numeri in virgola mobile di 64 bit, per rappresentare rispettivamente le parti intera e immaginaria. Non è possibile accedere loro dall'esterno della definizione di classe (e delle sue sottoclassi), dato che sono state dichiarate *protected*. Se fossero state dichiarate *private*, sarebbero state visibili solamente da *NumeriComplessi* e non dalle sottoclassi. Per il momento *private* potrebbe anche andare bene, ma presto definiremo una sottoclasse. Se fossero state dichiarate *public*, sarebbero state visibili ovunque sia visibile il package, distruggendo così la maggior parte del valore della programmazione orientata agli oggetti. Nonostante questo, esistono delle situazioni nelle quali a volte è necessario che lo stato interno di un oggetto sia pubblico.

```
class ComplexNumber {           // Define a subclass of Object called ComplexNumber
    // Hidden data.
    protected double re, im;      // real and imaginary parts

    // Five methods that manage the hidden data.
    public void Complex(double x, double y) {re = x; im = y;}
    public double Real() {return re;}
    public double Imaginary() {return im;}
    public double Magnitude() {return Math.sqrt(re*re + im*im);}
    public double Angle() {return Math.atan(im/re);}
}

class test {                   // A second class, for testing ComplexNumber
    public static void main (String args[]) {
        ComplexNumber c;          // declare an object of class ComplexNumber

        c = new ComplexNumber();   // actually allocate storage for c
        c.Complex(3.0, 4.0);       // invoke the Complex method to initialize c
        System.out.println("The magnitude of c is " + c.Magnitude());
    }
}
```

Fig. 7-72 Un package che definisce due classi.

Sugli oggetti appartenenti alla classe *NumeriComplessi* sono definiti cinque metodi. Le operazioni a disposizione degli utenti della classe sono così solamente queste cinque, e non arrivano direttamente allo stato. In *test* vediamo un esempio di creazione, inizializzazione e utilizzo degli oggetti della classe *NumeriComplessi*.

Quando questo package viene compilato, il compilatore produce due file binari (byte-code), ognuno contenente una classe, e il cui nome deriva da questa. Invia il comando

```
java test
```

viene invocato l'interprete Java con la classe *test* come parametro. A questo punto l'interprete cerca un metodo di nome *main*, e appena lo trova, lo esegue. Il risultato dell'esecuzione è la stampa della linea

The magnitude of c is 5

Definiamo adesso una sottoclasse dei *NumeriComplessi*, solo per vedere come lavora. Inizia importando la classe originale, per imparare cosa sia e quali metodi usi. Quindi definisce un'estensione dei *NumeriComplessi*, che sarà chiamata *NumeriSottili*. La nuova classe eredita automaticamente i cinque metodi presenti nella superclasse. Per rendere

interessante il gioco, definiamo nella sottoclass un sesto metodo, *SommaA*, che somma un numero complesso all'oggetto, incrementando le sue parti reale e immaginaria. La definizione di sottoclasse è data in figura 7-73, insieme a un altro programma di prova che mostra come usare un oggetto che appartiene alla classe *NumeriSottili*. Quando viene eseguito il nuovo programma test, questo stamperà

$h = (-0.5, 6)$

Ricordiamo che su *a* e *h* si possono usare tutti e sei i metodi, non importa dove ciascuno di essi sia stato definito. Se adesso definissimo un'altra sottoclasse basata su *NumeriSottili* e su di essa definissimo, diciamo, altri tre metodi, gli oggetti prodotti da questa avrebbero nove metodi validi.

Oltre che aggiungere nuovi metodi alla sua superclasse, una sottoclasse può sostituire (rimpiazzare) i metodi esistenti semplicemente ridefinendoli. Così è possibile per una sottoclasse ridefinire tutti i metodi ereditati dalla sua superclasse, e quindi due oggetti appartenenti alle due classi non avrebbero niente in comune. Fare questo, comunque, è da evitare.

Infine, una classe Java può definire più metodi con lo stesso nome ma con parametri differenti e definizioni differenti. Quando il compilatore trova un metodo invocato con questo nome, deve usare i tipi dei parametri per determinare quale metodo usare. Questa proprietà è detta **sovraffunzione o polimorfismo**. A differenza del C++, dove le operazioni possono sempre essere sovraccaricate, in Java è possibile sovraccaricare solamente i metodi, non le operazioni, per rendere i programmi più semplici da capire.

L'interfaccia per i programmatore di applicazioni (Java API)

Oltre al linguaggio in quanto tale, i progettisti di Java hanno definito e implementato con la versione iniziale circa 200 classi. I metodi contenuti in queste classi formano una sorta di ambiente standard per chi sviluppa programmi Java. Le classi sono scritte in Java, e quindi sono portabili su qualsiasi piattaforma e sistema operativo. Anche se una discussione dettagliata di tutte queste classi va oltre lo scopo di questo libro, può essere interessante una breve descrizione. Le 200 classi sono raggruppate in 7 package di dimensione diversa, ognuno dei quali è finalizzato a qualche tema principale. Gli applet che necessitano di un package particolare lo possono includere con l'istruzione Java import. I metodi in esso contenuti verranno usati solo se necessario. Questo meccanismo sostituisce quello comune in C di includere i file di intestazione. Inoltre elimina la necessità di collegare staticamente le librerie, in quanto i package sono caricati dinamicamente durante l'esecuzione al momento della loro invocazione.

La figura 7-74 elenca i 7 package. Il package *java.lang* contiene le classi che possono essere viste come parte del linguaggio, ma che tecnicamente non lo sono. Queste includono classi per la gestione delle classi stesse, dei thread, e per la gestione delle eccezioni. Qui si trovano anche le librerie standard matematiche e per la manipolazione di stringhe.

```
import ComplexNumber;           // import the ComplexNumber package

class HairyNumber extends ComplexNumber { // define a new class
    public void AddTo(ComplexNumber z) { // with one method
        re = re + z.Real();
        im = im + z.Imaginary();
    }
}

class test2 {                   // test program for HairyNumber
    public static void main(String args[]) {
        HairyNumber a, h;          // declare two HairyNumbers

        a = new HairyNumber();     // allocate storage for a
        h = new HairyNumber();     // allocate storage for h
        a.Complex(1.0, 2.0);      // assign a value to a
        h.Complex(-1.5, 4.0);    // assign a value to h
        h.AddTo(a);              // invoke the AddTo method on h
        System.out.println("h = (" + h.Real() + "," + h.Imaginary() + ")");
    }
}
```

Fig. 7-73 Una sottoclasse di *NumeriComplessi* che definisce un nuovo metodo.

Package	Funzionalità di esempio
<i>Java.lang</i>	Classi, Thread, eccezioni, matematiche, stringhe
<i>Java.io</i>	I/O su file sequenziali e ad accesso casuale
<i>Java.net</i>	Socket, indirizzi IP, URL, datagrammi
<i>Java.util</i>	Pile, tabelle hash, vettori, tempo, date
<i>Java.applet</i>	Recupero e visualizzazione di pagine Web
<i>Java.awt</i>	Gestione di eventi, dialog, menu, font, grafici, gestione finestre
<i>Java.awt.image</i>	Colori, collezione di immagini, filtraggio e conversione
<i>Java.awt.peer</i>	Accesso al sistema di finestre sottostante

Fig. 7-74 I package compresi nella Java API standard.

Come il C, il linguaggio Java non contiene primitive di I/O, che invece si ottengono caricando e utilizzando il package *java.io*. È analogo alla libreria standard del C. Ci sono metodi per leggere e scrivere file sequenziali, ad accesso casuale e per eseguire tutte le

necessarie formattazioni di stampa. Abbiamo usato uno di questi metodi, *println*, in figura 7-71, per formattare una stampa.

Il trasporto di rete è strettamente collegato all'I/O. In *java.net* si trovano i metodi per cercare e gestire gli indirizzi IP. Fa parte di questo package anche l'accesso ai socket. Lo stesso per la preparazione dei datagram. La vera trasmissione è gestita in *java.io*.

La successiva classe è *java.util*. Contiene le classi e i metodi per le strutture dati comuni, come le pile e le tabelle hash, in modo che i programmati non debbano ogni volta riscoprire l'acqua calda. Qui si trova anche la gestione del tempo e della data.

Il package *java.applet* contiene alcuni meccanismi di base per gli applet, compresi i metodi per recuperare le pagine Web, a cominciare dalle loro URL. Inoltre contiene i metodi per visualizzare le pagine e per far suonare i frammenti audio (ad es., una musica di sottofondo). Il package *java.applet* inoltre contiene la classe Object. Tutti gli oggetti ereditano i suoi metodi, a meno che non vengano sovrascritti. Questi metodi comprendono la clonazione di un oggetto, il confronto di due oggetti per uguaglianza, la conversione di un oggetto in una stringa e altro ancora.

Infine, arriviamo al *java.awt* e ai suoi sottopackage. AWT sta per **A**bstract **W**indow **T**oolkit, ed è stato progettato per rendere gli applet portabili tra diversi sistemi a finestre. Per esempio, come fa un applet a disegnare un rettangolo su uno schermo in modo che la stessa versione compilata (bytecode) dell'applet giri su UNIX, Windows e Macintosh, anche se ognuno di questi ha il proprio sistema di finestre? Parte del package si occupa di come disegnare sullo schermo, perciò ci sono metodi per piazzare sullo schermo linee, figure geometriche, testo, menù, bottoni, barre di scorrimento e molti altri elementi. È compito del package *java.awt* fare le opportune chiamate al sistema operativo locale per eseguire quel compito. Questa strategia comporta che *java.awt* vada riscritto per ogni nuova piattaforma, ma quegli applet sono indipendenti dalla piattaforma, cosa molto più importante.

Un altro compito molto importante di questa classe è la gestione degli eventi. La maggior parte dei sistemi a finestre è fondamentalmente diretta da eventi. Questo significa che il sistema operativo rileva la battitura di un tasto, il movimento del mouse, la pressione e rilascio di un bottone, e molti altri eventi, e li converte in chiamate a procedure d'utente. Nel caso di Java, in *java.awt* viene fornita una grossa libreria di metodi per il trattamento di questi eventi. Il loro utilizzo facilita la scrittura dei programmi che interagiscono con il sistema di finestre locale ed è ancora portabile al 100% su macchine con sistemi operativi differenti e sistemi a finestre differenti.

Parte del lavoro di questo package viene fatto da *java.image*, in quanto gestione di immagini, e in *java.awt.peer*, che consente l'accesso al sistema a finestre sottostante.

Sicurezza

Uno degli aspetti più importanti di Java sono le sue proprietà di sicurezza. Quando viene recuperata una pagina Web che contiene un applet, questo viene automaticamente eseguito sulla macchina del cliente. Idealmente, non dovrebbe fallire o guastare la macchina del cliente.

Inoltre, non ci vuole molto a immaginare qualche studente intraprendente che produca una pagina Web con qualche nuovo gioco carino, che pubblicizza la sua pagina URL

in lungo e in largo (ad es., inserendola in tutti i newsgroup). Nell'informazione data non fa riferimento al piccolo dettaglio che la pagina contiene anche un applet che appena arriva immediatamente crittografa tutti i file sul disco rigido dell'utente. Quando ha finito, l'applet annuncia cosa ha fatto ed educatamente informa che gli utenti che se volessero comprare la chiave per la decifratura dovrebbero inviare 1000 dollari in banconote di piccolo taglio a una certa cassetta postale a Panama.

Oltre al precedente schema "come-diventare-ricchi-velocemente", ci sono anche dei pericoli intrinseci nel permettere che codice sconosciuto giri sulla propria macchina. Un applet potrebbe andare a caccia di informazioni interessanti (posta elettronica salvata, il file delle password, le stringhe locali di ambiente ecc.) e quindi restituirle o spedirle tramite email al mittente. Potrebbe anche consumare delle risorse (ad es., riempire il disco), visualizzare sullo schermo figure oscene o slogan politici, oppure creare una microspia per ascoltare e trasmettere conversazioni attraverso la scheda audio.

I progettisti di Java conoscevano questi problemi, ovviamente, e hanno eretto contro di essi una serie di barriere. La prima linea di difesa consiste in un linguaggio *sicuro rispetto ai tipi*. Java ha una tipizzazione molto forte, veri array con il controllo sugli estremi e nessun puntatore. Queste restrizioni rendono impossibile a un programma Java di costruire un puntatore per leggere e scrivere in locazioni arbitrarie di memoria.

Comunque, Trudy, che ha smesso di cercare di violare i protocolli crittografici e cerca di trovare maggior fortuna nello scrivere degli applet Java con cattive intenzioni, può semplicemente scrivere o modificare un compilatore C per produrre il bytecode Java, così potrà oltrepassare tutti i controlli forniti dal linguaggio e dal compilatore Java.

La seconda linea di difesa è che prima che un applet esterno vada in esecuzione, viene fatto girare attraverso un *verificatore di bytecode*. Questo cerca i tentativi di costruzione di puntatori, l'esecuzione di istruzioni e le chiamate di metodi con parametri non validi, l'uso di variabili prima che siano state inizializzate, e così via. Lo scopo di questi controlli è di garantire che solamente gli applet legali vadano in esecuzione, ma Trudy certamente lavorerà sodo per trovare un trucco che il verificatore non controllerà.

La terza linea di difesa è il *caricatore di classi*. Dato che le classi vengono caricate al volo, c'è il pericolo che un applet possa caricare una delle sue proprie classi per sostituire una classe critica del sistema, in modo da oltrepassare tutti i controlli di sicurezza sulle classi. Questo attacco del tipo "cavalo di Troia" è reso impossibile grazie al fatto che ogni classe ha il proprio spazio dei nomi (una sorta di directory astratto), e che le classi di sistema vengono ricercate prima delle classi dell'utente. In altre parole, se l'utente caricasse una versione malvagia di *println*, questa non verrebbe mai usata perché verrebbe sempre trovata prima la versione ufficiale di *println*.

La quarta linea di difesa è che alcune classi standard hanno le proprie misure di difesa originali. Ad esempio, la classe per l'accesso ai file mantiene una lista di file accessibili dagli applet, e mostrano una finestra di dialogo ogni volta che un applet cerca di far qualcosa per violare le regole di protezione.

A dispetto di tutte queste misure di sicurezza, ci sono ancora dei problemi. Primo, è possibile che ci siano dei bachi nel software Java che dei programmati intelligenti possano sfruttare per oltrepassare la sicurezza. L'infame virus Internet del 1988 utilizzò

un baco del demone Finger di UNIX per mettere in ginocchio migliaia di macchine su tutta Internet (Hafner, Markoff 1991; Spafford 1989).

Secondo, mentre è possibile evitare che un applet faccia tutto tranne che scrivere sullo schermo, molti applet hanno bisogno di più potere, così quando richiedono dei privilegi speciali, gli utenti possono a malincuore (o ingenuamente) accordarglieli. Ad esempio, gli applet debbono poter scrivere dei file temporanei, così gli utenti possono permettere l'accesso alla directory `/tmp`, pensando che lì non ci sia niente di importante. Sfortunatamente, molti editor tengono proprio lì le versioni temporanee dei documenti e dei messaggi di posta elettronica, perciò gli applet malintenzionati possono copiarli e cercare di spedirli via rete. Ovviamente, è possibile bloccare l'accesso alla rete agli applet, ma allora molti non funzionerebbero, perciò devono avere questo privilegio.

Ma anche nell'eventualità inverosimile che non siano assolutamente consentiti gli accessi in rete agli applet, questi potrebbero comunque essere in grado di trasmettere informazioni mediante i canali coperti (Lampson, 1973). Ad esempio, dopo aver acquisito delle informazioni, un applet può formare una stringa di bit utilizzando l'orologio reale del sistema locale. Per inviare un 1, lavorerà molto per Δt ; per inviare uno 0, semplicemente attende per Δt . Per ottenere questa informazione, il proprietario dell'applet può stabilire una connessione alla macchina del cliente per leggere alcune delle sue pagine Web o per prendere tramite FTP alcuni dei suoi file pubblici. Monitorando accuratamente il flusso dei dati in arrivo, il proprietario dell'applet può vedere se l'applet sta lavorando (e questo rallenta l'uscita dei dati osservati) o sta riposo. Ovviamente questo canale è disturbato, ma è possibile gestirlo con delle tecniche standard. I dati possono venire suddivisi in pacchetti delimitati da byte tipo flag, i singoli pacchetti possono usare un codice con una forte correzione degli errori, e tutti i pacchetti vengono inviati due o tre volte. Esistono molti altri canali coperti ed è particolarmente difficile scoprirli e bloccarli. Per ulteriori informazioni riguardo ai problemi di sicurezza in Java, si vedano Dean, Wallach (1995).

In breve, Java introduce molte nuove possibilità e opportunità nel World Wide Web. Permette alle pagine Web di diventare interattive, e di contenere animazione e suono. Inoltre permette una infinita estensibilità dei browser. Comunque, il modello Java di caricamento degli applet introduce anche alcuni seri problemi di sicurezza che non sono stati ancora interamente risolti.

7.6.5 Come localizzare le informazioni sul Web

Anche se il Web contiene una grande quantità di informazioni, trovare l'elemento giusto non è sempre una cosa facile. Per facilitare il ritrovamento delle pagine utili, molti ricercatori hanno scritto dei programmi per indicizzare il Web in vari modi. Alcuni di questi programmi sono diventati così popolari che sono stati commercializzati. I programmi che ricercano sul Web sono di volta in volta detti motori di ricerca, spider, crawler, worm o knowbots (knowledge robots). In questo paragrafo daremo una breve introduzione di questo argomento. Per maggiori informazioni, si vedano Pinkerton (1994); McBryan (1994).

Sebbene il Web sia enorme, ridotto alla sua parte essenziale è solo un grosso grafo, in cui i nodi sono le pagine e gli archi sono gli iperpuntatori. Gli algoritmi per visitare tutti i nodi di un grafo sono noti. Quello che rende difficile indicizzare il Web è

l'enorme quantità di dati che deve essere gestita e il fatto che questi dati siano in costante cambiamento.

Iniziamo la nostra discussione con un semplice obiettivo: indicizzare tutte le parole chiave nei titoli delle pagine Web. Per il nostro algoritmo, abbiamo bisogno di tre strutture dati. Per primo, abbiamo bisogno di un grosso vettore lineare, la `url_table`, che contiene milioni di elementi, almeno uno per ogni pagina Web. Dovrebbe essere conservato in memoria virtuale, in modo che le parti meno usate vengano automaticamente paginate su disco. Ogni elemento contiene due puntatori, uno alla URL e uno al titolo della pagina. Entrambi questi elementi sono stringhe a lunghezza variabile e possono essere mantenute in un heap (un grosso insieme non strutturato di memoria virtuale nel quale è possibile appendere le stringhe). L'heap è la nostra seconda struttura dati.

La terza struttura dati è una tabella hash di dimensione n . Viene usata nella maniera seguente. Qualsiasi URL si può elaborare con una funzione hash per produrre un intero non negativo minore di n . Tutte le URL che la funzione hash trasforma nel valore k vengono raccolte insieme in una lista collegata che inizia con l'elemento k della tabella hash. Se una URL viene inserito nella `url_table`, finisce anche nella tabella hash. L'uso principale della tabella hash è quello di determinare velocemente se una data URL sia già presente nella `url_table`. La figura 7-75 illustra queste tre strutture dati.

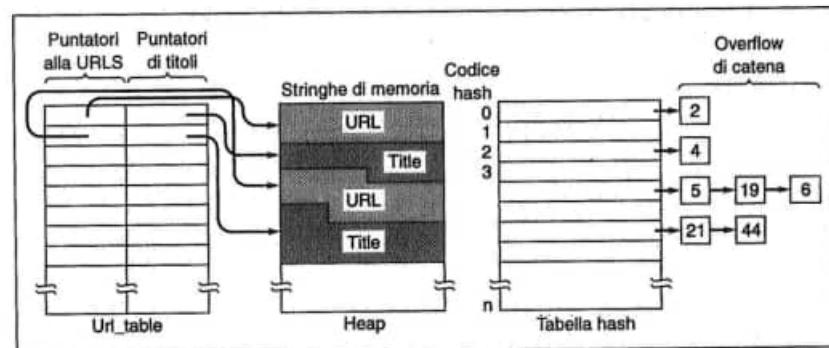


Fig. 7-75 Strutture dati necessarie per un semplice motore di ricerca.

La costruzione dell'indice richiede due fasi: ricerca e indicizzazione. Iniziamo con un semplice motore di ricerca. Il cuore del motore di ricerca è una procedura ricorsiva `process_url` che prende in ingresso una stringa URL e funziona come segue. Per prima cosa, elabora l'URL con la funzione hash per vedere se è già presente nella `url_table`. Se lo è, ha finito e termina immediatamente. Ogni URL si elabora una volta sola.

Se l'URL non è già conosciuta, viene recuperata la pagina. Quindi vengono copiati URL e titolo nell'heap e i puntatori a queste due stringhe vengono immessi nella `url_table`. La stringa URL viene anche inserita nella tabella hash.

Infine, `process_url` estrae dalla pagina tutti gli iperpuntatori e chiama `process_url` una volta per ogni iperpuntatore, passando la stringa URL dell'iperpuntatore come parametro di ingresso.

Per eseguire il motore di ricerca, viene chiamata *process_url* con una certa URL iniziale. Quando termina, nella *url_table* sono state inserite tutte le pagine raggiungibili da quella URL e la fase di ricerca è terminata.

Anche se questo schema è semplice e teoricamente corretto, presenta un grosso problema in un sistema vasto come il Web. Il problema è che questo algoritmo fa una ricerca in profondità (a scandaglio), e dunque scatenerà la ricorsione tante volte quanto è lungo il massimo cammino non ciclico. Nessuno sa quanto sia lungo questo cammino, ma probabilmente è lungo migliaia di iperpuntatori. Di conseguenza, qualsiasi motore di ricerca che utilizzi questa tecnica di ricerca a scandaglio probabilmente provocherà un trabocco della pila prima che il lavoro sia terminato.

In pratica, i motori di ricerca attuali dapprima collezionano tutti gli iperpuntatori per ogni pagina che leggono, eliminano tutti quelli che sono già stati elaborati e salvano il resto. Quindi esegue una ricerca a ventaglio sul Web; questo significa che viene seguito ogni puntatore su una pagina e che vengono collezionati tutti gli iperpuntatori di tutte le pagine puntate, ma questi non vengono seguiti nell'ordine ottenuto.

La seconda fase effettua l'indicizzazione delle parole chiave. La procedura di indicizzazione attraversa linearmente la *url_table*, elaborando a turno ogni elemento. Per ogni elemento, esamina il titolo e seleziona tutte le parole che non sono sulla lista di esclusioni. (Questa lista evita che vengano indicizzate le preposizioni, le congiunzioni, gli articoli e le altre parole molto comuni ma di scarso significato). Per ogni parola selezionata, scrive in un file una linea formata dalla parola seguita dal numero dell'elemento corrente nella *url_table*. Quando è stata esaminata l'intera tabella, il file viene ordinato sulle parole. L'indice verrà memorizzato sul disco e verrà utilizzato nella maniera seguente. L'utente completa un modulo con una o più parole chiave e fa click sul bottone SUBMIT. Questa azione provoca l'invio di una richiesta *POST* al programma CGI della macchina su cui si trova l'indice. Questo programma quindi cerca le parole chiave nell'indice per trovare l'insieme degli indici della *url_table* per ognuna di esse. Se l'utente vuole un AND BOLEANO delle parole chiave, viene calcolata l'intersezione degli insiemi. Se invece si vuole l'OR BOLEANO, viene calcolata l'unione degli insiemi.

Il programma CGI a questo punto effettua una ricerca con gli indici nella *url_table* per trovare tutti i titoli e le URL. Questi vengono poi combinati per formare una pagina Web e quindi vengono restituiti all'utente in risposta alla *POST*. A questo punto il browser visualizza la pagina, consentendo all'utente di fare click su qualsiasi elemento che appaia interessante. Sembra facile? Non lo è. In qualsiasi sistema pratico si devono risolvere i problemi seguenti:

1. Alcune URL sono obsolete (cioè, puntano a pagine che non esistono più).
2. Alcune macchine saranno temporaneamente irraggiungibili.
3. Non tutte le pagine saranno raggiungibili dalla URL iniziale.
4. Alcune pagine possono essere raggiunte solo da mappe attive.
5. Non è possibile indicizzare alcuni documenti (ad es., le tracce audio).
6. Non tutti i documenti hanno dei titoli (che siano utili).

7. Il motore di ricerca potrebbe esaurire la memoria o lo spazio disco.

8. L'intero processo potrebbe richiedere troppo tempo.

Le URL obsolete fanno perdere tempo, ma costituiscono un problema trascurabile perché il server su cui dovrebbero trovarsi risponde immediatamente con un codice di errore. Al contrario, quando il server è giù, tutto il motore di ricerca osserva un grosso ritardo nello stabilire la connessione TCP. Per evitare l'attesa infinita, occorre una temporizzazione (timeout). Se il timeout è troppo breve, potrebbero essere scartati delle URL valide. Se è troppo lungo, la ricerca verrebbe sensibilmente rallentata.

La scelta della URL iniziale è ovviamente fondamentale. Se il motore di ricerca inizia con la pagina principale di qualche astrofisico, sarà possibile trovare quasi tutto su astronomia, fisica, chimica e scienza dello spazio, ma mancheranno completamente pagine sulla medicina veterinaria, l'inglese e il rock'n roll. Infatti questi insiemi possono essere disgiunti. Una soluzione è raccogliere il più grande insieme possibile di URL, e usare ciascun elemento dell'insieme come pagina di partenza. Le URL iniziali si possono prendere dagli articoli delle news di USENET e dall'ultima versione settimanale della *url_table*, dato che alcune di queste pagine possono essere state modificate di recente (ad esempio, uno degli astrofisici si è sposato con una veterinaria ed entrambi hanno solennemente modificato le loro pagine principali perché puntassero l'una all'altra).

L'indicizzazione lavora bene sul testo, ma sempre più pagine contengono elementi non testuali, come immagini, audio e video. In questo caso una soluzione è interrogare ogni nuova URL trovata con il metodo *HEAD*, giusto per ottenere la sua intestazione MIME. Qualsiasi cosa che non sia di tipo *text* non viene ricercata.

Circa il 20% di tutte le pagine Web non ha titolo, e in molte di quelle pagine che lo hanno è inutile (ad es. "Pagina di Guido"). Un grosso miglioramento all'indice base è quello di non includere solamente i titoli, ma anche tutto l'ipertesto. In questo modo, quando viene esaminata una pagina, vengono anche registrati tutti gli iperpuntatori, insieme alla pagina dalla quale provengono e alla pagina alla quale puntano. Dopo che la fase di ricerca è stata completata, è possibile indicizzare anche tutte le iperparole.

Ancora più ambizioso è il progetto di indicizzare tutte le parole importanti di ogni pagina. Per determinare quali siano le parole importanti, è possibile calcolare (per ogni pagina Web) la frequenza di occorrenza di tutte le parole che non sono nella lista delle esclusioni. Probabilmente è bene indicizzare le prime 10 o 20 parole. Dopo tutto, se la parola "fegato" è la parola più usata in una pagina, c'è la possibilità che la pagina sia di interesse per i chirurghi biliari (o per i cuochi). Alcuni motori di ricerca (come Lycos) utilizzano questa strategia.

Infine, il motore di ricerca può esaurire la memoria o il tempo. Una soluzione è quella di riprogettare gli algoritmi più attentamente. Un approccio completamente differente è quello di Harvest, che distribuisce il lavoro (Bowman et al., 1994, 1996). In particolare, Harvest si basa su un programma che gira su server cooperanti. Questo programma effettua tutta la ricerca localmente e restituisce alla fine l'indice locale. In un sito centrale vengono fusi insieme tutti gli indici locali in un indice principale. Questo approccio riduce di parecchi ordini di grandezza la quantità di memoria, il tempo di CPU e la banda di rete

necessari ma ha il grande svantaggio di richiedere che tutti i server Web cooperino eseguendo software alieno. Dati i possibili problemi con virus e vermi, quando a un amministratore di sistema viene fatta la richiesta "Per favore, può eseguire questo programma sulla sua macchina?", non dovrebbe sorprendere il rifiuto di molti.

È d'uopo una piccola preghiera. Sebbene sembri facile scrivere un motore di ricerca, uno che risulti scorretto può travolgere la rete generando un grande numero di richieste spurie, e non solo sprecherebbe banda di comunicazione, ma metterebbe in ginocchio molti server col suo carico computazionale. Se non potete resistere alla tentazione di scrivere il vostro motore di ricerca, l'etichetta di rete richiede che agisca solo sul vostro DNS locale fino a che non è completamente scuro da errori.

7.7 Il multimediale

Il multimediale è il sacro Graal della rete. Quando se ne parla, tutti i cani della muta cominciano a salivare come se avessero trovato la traccia da seguire. I tecnici vedono immensi problemi da risolvere nel portare il video interattivo a richiesta in ogni casa. I commerciali vedono invece immensi profitti. Nessun libro sulle reti sarebbe completo senza almeno introdurre l'argomento. Data la lunghezza raggiunta sinora dal presente libro, la nostra introduzione sarà necessariamente sintetica. Altre informazioni su questo argomento affascinante e potenzialmente assai remunerativo si trovano in Buford (1994); Deloddere *et al.* (1994); Dixit, Skelly (1995); Fluckiger (1995); Minoli (1995); Steinmetz, Nahrstedt (1995).

Multimediale letteralmente significa due o più media. Se l'editore di questo libro volesse cavalcare a poco prezzo la moda del multimediale, potrebbe asserire che il libro usa tecnologia multimediale. Dopotutto, questo libro contiene testo e grafica: le figure. D'altra parte, quando si parla di multimediale in genere si intende la combinazione di due o più **media continui**, cioè media che debbono essere riprodotti in sincrono durante qualche ben definito intervallo di tempo, di solito con qualche interazione d'utente. In realtà i due media sono quasi sempre audio e video, cioè suoni più immagini in movimento. Per questo motivo noi cominceremo il nostro studio con una introduzione alle tecnologie audio e video. Poi le combineremo e introdurremo i veri sistemi multimediali, che includono il video a richiesta e il sistema multimediale di Internet, ovvero MBone.

7.7.1 Audio

Un'onda audio è un'onda acustica (pressione) monodimensionale. Quando un'onda acustica entra nell'orecchio, il timpano vibra, il che obbliga gli ossicini dell'orecchio interno a vibrare a loro volta, trasmettendo impulsi nervosi al cervello. Questi impulsi vengono percepiti come suoni dall'ascoltatore. allo stesso modo, quando un'onda acustica colpisce un microfono, questo genera un segnale elettrico, che rappresenta l'ampiezza del suono come funzione del tempo. La rappresentazione, l'elaborazione, la memorizzazione e la trasmissione di tali segnali audio sono una parte importante dello studio dei sistemi multimediali.

Le frequenze udibili dall'orecchio umano rientrano nell'intervallo da 20 a 20.000 Hz; invece alcuni animali, specialmente i cani, possono sentire frequenze più alte.

L'orecchio ascolta in modo logaritmico, quindi il rapporto di due suoni con ampiezze A e B viene convenzionalmente espresso in **dB (decibel)** usando la formula

$$\text{dB} = 20 \log_{10} (A/B)$$

Se definiamo il limite inferiore dell'udibilità (pressione di circa 0,0003 dyne/cm²) per un'onda seno di 1 kHz come 0 dB, una conversazione con tono normale vale 50 dB e la soglia del dolore è di circa 120 dB, un intervallo dinamico di un fattore di circa 1.000.000. Per evitare ogni confusione, A e B citate sopra sono **ampiezze**. Se dovessimo usare il livello di energia, che è proporzionale al quadrato dell'ampiezza, il coefficiente del logaritmo sarebbe 10 e non 20.

L'orecchio è sorprendentemente sensibile alle variazioni di suono che durano solo pochi millisecondi. L'occhio, invece, non è capace di notare cambiamenti del livello luminoso che durano solo pochi millisecondi. Questa osservazione ha la conseguenza che un'interferenza di appena pochi millisecondi durante una trasmissione multimediale influenza la qualità del suono percepita assai più di quanto influenzi la qualità dell'immagine.

Le onde radio possono essere convertite in forma digitale da un **ADC (convertitore analogico digitale)**. Un ADC prende in ingresso un voltaggio e genera in uscita un numero binario. Nella figura 7-76 (a) vediamo un esempio di un'onda seno. Per rappresentare questo segnale in forma digitale, lo campioniamo ogni ΔT secondi, come mostrano le barre in figura 7-76 (b). Se un'onda sonora non ha la forma pura di un'onda seno, ma è una combinazione lineare di onde seno in cui f è la componente di frequenza più alta, allora il teorema di Nyquist (vedi capitolo 2) stabilisce che è sufficiente campionare a una frequenza $2f$. Campionare più spesso non ha senso perché sono assenti le frequenze superiori che tale campionamento potrebbe rilevare.

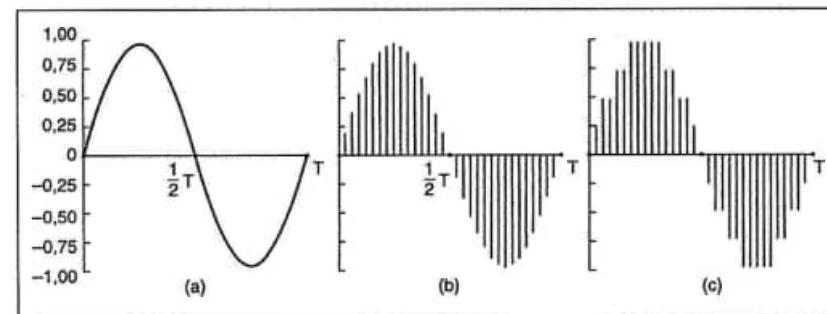


Fig. 7-76 (a) Onda seno. (b) Campionamento dell'onda seno. (c) Quantizzazione su 3 bit del campionamento.

I campionamenti digitali non sono mai esatti. I campionamenti su 3 bit di figura 7-76(c) riportano solo 8 valori, da -1,00 a +1,00 in passi di 0,25. Un campionamento su 8 bit permetterebbe di avere 256 valori distinti. Un campionamento su 16 bit permetterebbe di

avere 65.536 valori distinti. L'errore introdotto dal numero finito di bit per campionamento si chiama **rumore di quantizzazione**. Se è troppo grande l'orecchio se ne accorge. Due esempi ben conosciuti di suono campionato sono il telefono e i CD audio. La modulazione d'impulso che si usa nel sistema telefonico utilizza campionamenti su 7 bit (Nord America e Giappone) o su 8 bit (Europa) 8000 volte/s. Questo sistema corrisponde a una velocità di trasmissione di 56.000 o 64.000 bps. Con un tasso di solo 8000 campionamenti/s si perdono le frequenze oltre i 4 kHz.

I CD audio sono digitalizzati con un tasso di campionamento di 44.100 campioni/s, sufficiente per catturare le frequenze fino a 22.050 Hz. I campionamenti sono su 16 bit, e sono distribuiti linearmente nell'intervallo delle ampiezze. Si noti che i campionamenti su 16 bit risultano in solo 65.536 valori diversi, anche se l'intervallo dinamico dell'orecchio è di circa 1.000.000 quando viene misurato in termini di passi elementari di suoni udibili. Quindi l'uso di campionamenti su 16 bit introduce un certo rumore di quantizzazione (anche se l'intervallo completo di udibilità non viene coperto: i CD non debbono far male all'orecchio). Se ogni secondo contiene 44.100 campionamenti su 16 bit, un CD audio necessita di una banda di 705,6 kbps per il suono normale e 1,411 Mbps per lo stereo. Anche se queste misure sono minori di quel che serve alla riproduzione video (vedi sotto), occorre comunque un canale T1 completo per trasmettere senza compressione suono stereo di qualità CD.

Il suono digitalizzato si può facilmente manipolare con software specifici. Esistono dozzine di programmi su PC che permettono di registrare, ascoltare, editare, miscelare, e memorizzare onde sonore provenienti da fonti multiple. In pratica tutti i sistemi professionali moderni di registrazione e manipolazione dei suoni sono oggi digitali.

Persino gran parte degli strumenti musicali hanno un'interfaccia digitale. Quando vennero messi in commercio i primi strumenti digitali ciascuno possedeva un'interfaccia specifica, ma dopo qualche tempo venne definito uno standard, **MIDI (Music Instrument Digital Interface)** che venne adottato da tutta l'industria musicale. Questo standard specifica il connettore, il cavo e il formato dei messaggi. Ogni messaggio MIDI comprende un byte di stato seguito da 0 o più byte di dati. Un messaggio MIDI trasporta un evento musicalmente significativo. Eventi tipici sono la pressione su un tasto, il movimento di una manopola, un pedale che viene rilasciato. Il byte di stato indica l'evento, mentre i byte di dati riportano i parametri, come quale tasto è stato premuto e a quale velocità si è mosso.

Ogni strumento è associato a un codice MIDI. Ad esempio, un grande pianoforte è identificato da 0, una marimba da 12, un violino da 40. Questo è necessario per evitare che un concerto per flauto venga riprodotto come concerto per tuba. Il numero di "strumenti" definiti è 127. Tuttavia, non tutti i numeri sono associati a strumenti, ma anche a effetti speciali come uccelli cinguettanti, elicotteri, e l'applauso artificiale che accompagna molti programmi televisivi.

Il cuore di ogni sistema MIDI è il sintetizzatore (spesso è un computer) che accetta messaggi e li traduce in musica. Il sintetizzatore può riprodurre tutti i 127 strumenti, e quindi è capace di generare un diverso spettro di energia per un Do suonato da una tromba piuttosto che da uno xilofono. Lo svantaggio di MIDI è che il ricevitore necessita di un sintetizzatore MIDI per ricostruire il suono musicale, e sintetizzatori diversi potrebbero riprodurlo in modi leggermente differenti.

Ovviamente la musica è solo un caso speciale di audio, ma è importante. Un altro caso speciale è il linguaggio parlato. La voce umana rientra di solito nello spettro 600-6000 Hz. Il linguaggio parlato si compone di vocali e consonanti, che hanno proprietà diverse. Le vocali vengono pronunciate quando il tratto vocale non è ostruito, il che produce una risonanza la cui frequenza fondamentale dipende dalla dimensione e dalla forma del sistema vocale e dalla posizione della lingua e della masella del parlante. Questi suoni sono quasi periodici in intervalli di circa 30 ms. Le consonanti vengono pronunciate quando il tratto vocale è parzialmente ostruito. Questi suoni sono meno regolari delle vocali.

Alcuni sistemi di generazione e trasmissione del parlato invece di campionare l'onda sonora vocale usano modelli del sistema vocale per rappresentare la voce usando alcuni parametri (ad es. le dimensioni e le forme di alcune cavità).

7.7.2 Video

L'occhio umano ha la proprietà che quando un'immagine viene impressa sulla retina, viene mantenuta per alcuni millisecondi prima di svanire. Se una serie di immagini viene proiettata alla velocità di 50 o più immagini/s, l'occhio non si accorge che ciò che sta vedendo sono immagini discrete. Ogni sistema video, compresa la televisione, sfrutta questo principio per produrre immagini in movimento.

Sistemi analogici

Per comprendere i sistemi video è meglio cominciare con la vecchia televisione bianco e nero. Per rappresentare un'immagine bidimensionale come voltaggio monodimensionale in funzione del tempo la telecamera punta un pennello elettronico sulla scena che ha di fronte muovendolo rapidamente in senso orizzontale e lentamente in verticale, registrando tutte le intensità luminose. Alla fine della scansione si ottiene un'immagine, e il pennello elettronico ricomincia daccapo. Le intensità luminose registrate in funzione del tempo vengono trasmesse ai ricevitori, che ripetono il processo di scansione per ricostruire l'immagine. Lo schema di scansione usato sia dalla telecamera che dal ricevitore televisivo viene mostrato in figura 7-77. (Diremo anzi che le telecamere CCD usano un metodo di integrazione invece che una scansione, ma comunque tutti i monitor e alcune telecamere usano la scansione).

I parametri precisi che regolano la scansione variano da paese a paese. Il sistema usato in America e in Giappone include 525 linee di scansione, un rapporto orizzontale/verticale 4:3 e 30 immagini/s. Il sistema europeo ha 625 linee, lo stesso rapporto 4:3 e 25 immagini/s. In entrambi i sistemi le linee in alto e in basso non vengono mostrate (per approssimare l'immagine rettangolare dei CRT originali, che erano rotondi). Nel sistema NTSC vengono mostrate solo 483 delle 525 linee (e 576 su 625 nel sistema PAL/SECAM). Il pennello elettronico si spegne durante il movimento di reinizializzazione verticale, e molte stazioni, specie in Europa, usano questo intervallo per trasmettere TeleText (pagine di testo che contengono notizie di cronaca e sportive, previsioni meteo, prezzi di borsa ecc.).

Anche se 25 immagini/s sono sufficienti per rappresentare il movimento fluido, a quella velocità alcune persone, specialmente tra gli anziani, percepiscono un'immagine disturbata, perché la vecchia immagine svanisce dalla retina prima che appaia la nuova. Piut-

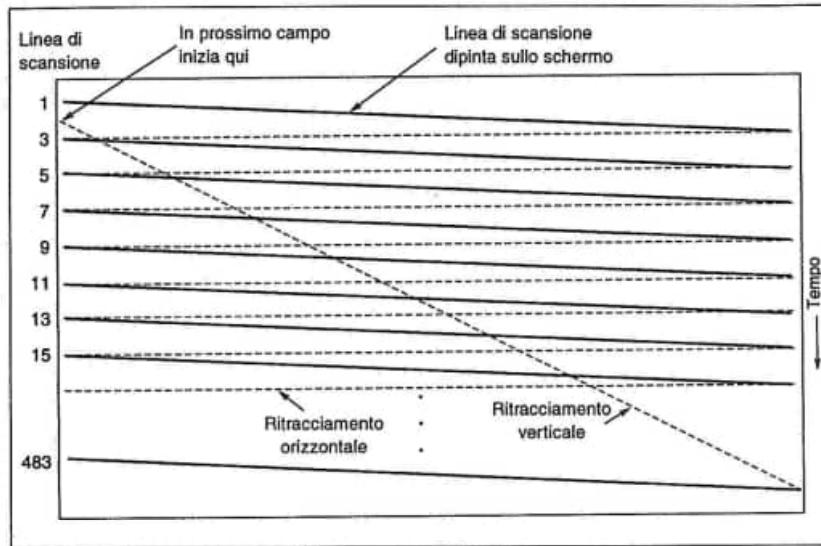


Fig. 7-77 Lo schema di scansione usato dal video e dalla televisione col sistema NTSC.

tosto che aumentare la velocità delle immagini, che richiederebbe di aumentare l'uso di banda di trasmissione, che invece è scarsa, si usa un sistema diverso. Invece di mostrare le linee di scansione nell'ordine corretto, prima vengono mostrate tutte le linee dispari, poi quelle pari. Queste mezze immagini si chiamano **campi**. Molti esperimenti hanno dimostrato che sebbene la gente percepisca uno sfarfallio a 25 immagini/s, nessuna si lamenta se la velocità è di 50 immagini/s. Questa tecnica si chiama **interallacciamento**. Il video o la televisione non interallacciati si chiamano **progressivi**.

Il video a colori usa lo stesso schema di scansione della TV in bianco e nero, solo che invece di usare un solo pennello elettronico se ne usano tre che si muovono all'unisono. Viene usato un pennello per ciascuno dei colori primari additivi: rosso, verde e blu (RGB – red, green, blue). Questa tecnica funziona perché qualsiasi colore si può costruire per sovrapposizione lineare di rosso, verde e blu di intensità appropriata. Tuttavia, quando si usa la trasmissione su singolo canale, i tre segnali di colore vanno combinati in un unico segnale **composito**.

Quando venne inventata la televisione a colori erano possibili diversi metodi per riprodurre il colore, e nazioni diverse fecero scelte differenti, che portarono a sistemi che sono tuttora incompatibili. (Si noti che queste scelte non hanno nulla a che vedere con il VHS, il Betamax, o il P2000, che sono metodi di registrazione). In tutte le nazioni un requisito fu che i programmi trasmessi a colori dovessero essere ricevibili sui televisori esistenti in bianco e nero. Di conseguenza, non era accettabile lo schema più semplice, cioè la codifica separata dei segnali RGB. Inoltre, RGB non è lo schema più efficiente.

Il primo sistema a colori fu standardizzato negli Stati Uniti dal Comitato per gli

standard televisivi nazionali (National Television Standards Committee), che battezzò col suo acronimo lo standard: **NTSC**. La televisione a colori fu introdotta in Europa parecchi anni più tardi, in un momento in cui la tecnologia era molto migliorata, per cui era possibile produrre sistemi molto meno rumore e con colori migliori. Tali sistemi si chiamano **SECAM** (SEquentiel Couleur Avec Memoire), usato in Francia ed Europa dell'Est, e **PAL** (Phase Alternating Line), usato nel resto d'Europa. La differenza in qualità di colore tra NTSC e PAL/SECAM ha fatto nascere una battuta popolare tra i tecnici per cui NTSC significa: "Never Twice the Same Color" (mai due volte lo stesso colore).

Per far sì che le trasmissioni a colori possano essere riprodotte su ricevitori in bianco e nero tutti i sistemi combinano linearmente i segnali RGB in un segnale di luminanza (luminosità), e due segnali di crominanza (colore). Essi tuttavia usano diversi coefficienti per costruire tali segnali dai segnali RGB. È interessante notare che l'occhio è molto più sensibile al segnale di luminanza che a quelli di crominanza, quindi questi ultimi possono non essere trasmessi con precisione. Di conseguenza, il segnale di luminanza si può trasmettere alla stessa frequenza del vecchio segnale bianco e nero. I due segnali di crominanza vengono trasmessi in bande strette a frequenze più alte. Per controllare tali segnali in modo indipendente alcuni televisori hanno delle manopole che si chiamano luminosità, colore e saturazione. La spiegazione sulla luminanza e la crominanza è necessaria per capire come funziona la compressione video.

Alcuni anni fa c'era un considerevole interesse per la TV ad alta definizione (**HDTV – High Definition TV**), che produce immagini più contrastate raddoppiando il numero di linee di scansione. Gli USA, l'Europa e il Giappone hanno sviluppato sistemi HDTV tutti diversi e mutuamente incompatibili. I principi base della HDTV riguardo la scansione, la luminanza, la crominanza e così via sono tutti simili a quelli dei sistemi esistenti. Tuttavia, tutti i formati hanno un rapporto di immagine 16:9 invece di 4:3 per potersi adattare al formato usato per i film (registrati su pellicola a 35 mm).

Per un'introduzione alla tecnologia televisiva, si veda Buford (1994).

Sistemi digitali

La rappresentazione più semplice del video digitale è una sequenza di inquadrature (frame), ciascuna avente forma di matrice rettangolare composta da punti immagine elementari detti **pixel**. Ciascun pixel può essere un singolo bit, per rappresentare il bianco o il nero. La qualità di tale sistema è simile a quella che si ottiene quando si trasmette una foto a colori via fax – terribile. (Provateci; oppure fotocopiate una foto a colori su una fotocopiatrice che non "rasterizza" l'immagine.)

Il passo successivo è di usare 8 bit/pixel per rappresentare 256 livelli di grigio. Questo schema produce video in bianco e nero di alta qualità. Nel caso di video a colori i sistemi buoni usano 8 bit per ciascuno dei tre colori RGB, anche se quasi tutti i sistemi li mescolano in video composito per la trasmissione. Usando 24 bit/pixel si ottengono circa 16.000.000 di colori, che l'occhio umano certamente non distingue completamente. Le immagini digitali a colori vengono prodotte usando tre pennelli di scansione, uno per colore. La geometria è la stessa del sistema analogico di figura 7-77 eccetto che le linee continue di scansione sono ora rimpiazzate da righe di pixel discreti.

Per produrre il movimento continuo, il video digitale, come quello analogico, deve riprodurre almeno 25 immagini/s. Tuttavia, siccome i monitor di buona qualità spesso riproducono sullo schermo immagini prese dalla memoria 75 volte/s o più, l'interallacciamento non è necessario e quindi di norma non viene usato. La riproduzione della stessa immagine tre volte di fila elimina qualsiasi sfarfallio.

In altri termini, la fluidità del moto è determinata dal numero di immagini diverse per secondo, mentre lo sfarfallio è determinato dal numero di volte per secondo in cui lo schermo è ridisegnato. Questi due parametri sono diversi. Un'immagine ferma riprodotta a 20 immagini/s non mostrerà movimenti a scatti ma conterrà sfarfallio perché un'immagine decadrà dalla retina prima che appaia la successiva. Un film con 20 immagini/s, ciascuna riprodotta 4 volte di fila, non avrà sfarfallio, ma il moto apparirà discontinuo.

Il significato di questi due parametri diventa ancora più chiaro se consideriamo la banda richiesta per trasmettere video digitale su una rete. I monitor attuali usano il rapporto 4:3 in modo da poter usare i tubi catodici economici prodotti in massa per il mercato televisivo. Le configurazioni più comuni sono 640×480 (VGA), 800×600 (SVGA), e 1024×768 (XGA). Un video XGA con 24 bit/pixel e 25 immagini/s va trasmesso a 472 Mbps. Non basta nemmeno OC-9, portare SONETOC-9 in ogni casa non è certo in agenda. Il raddoppio di questo tasso per evitare lo sfarfallio è anche più impensabile. Una miglior soluzione è di trasmettere 25 immagini/s, e farle memorizzare al computer per riprodurlle due volte. La televisione in broadcast non usa questa strategia perché i televisori non hanno memoria, e in ogni caso i segnali analogici non possono essere memorizzati in RAM senza prima convertirli in forma digitale, che richiede hardware extra. Di conseguenza, l'interallacciamento è necessario per le trasmissioni TV ma non per il video digitale.

7.7.3 Compressione dei dati

A questo punto dovrebbe essere ovvio che la trasmissione in forma non compressa di materiale multimediale è del tutto fuori questione. L'unica speranza è che sia possibile compressione massiccia. Per fortuna, il gran numero di ricerche negli ultimi decenni ha generato tante tecniche e algoritmi di compressione che rendono possibile la trasmissione multimediale. In questo paragrafo studieremo alcuni metodi per comprimere i dati multimediali, specie le immagini. Per maggiori dettagli si vedano Fluckiger (1995); Steinmetz, Nahrstedt (1995).

Tutti i sistemi di compressione richiedono due algoritmi: uno per comprimere i dati alla fonte, e uno per decomprimerli alla destinazione. Nella letteratura questi algoritmi vengono detti algoritmi rispettivamente di **codifica** e **decodifica**. Noi ci atterremo a tale terminologia.

Tali algoritmi presentano alcune asimmetrie che sono importanti da capire. Per molte applicazioni un documento multimediale, ad esempio un film, verrà codificato una volta sola (quando è memorizzato in un servizio multimediale), ma sarà decodificato migliaia di volte (quando viene visto dai clienti del servizio). Questa asimmetria implica che possiamo accettare che l'algoritmo di codifica sia lento e richieda hardware costoso, purché l'algoritmo di decodifica sia veloce e non richieda hardware costoso. Dopo tutto,

l'operatore del servizio multimediale potrebbe accettare di noleggiare un super computer parallelo per alcune settimane per codificare tutta la sua videoteca, mentre la richiesta ai clienti di noleggiare un supercomputer per gustarsi un film di due ore non avrà un gran successo. In pratica molti sistemi di compressione si sforzano di rendere la decodifica facile e veloce, anche al prezzo di rallentare e complicare la codifica.

D'altra parte, nel caso di trasmissioni in tempo reale, come la videoconferenza in diretta, una codifica lenta è inaccettabile. La codifica deve anch'essa procedere in diretta, in tempo reale. Di conseguenza, il multimediale in tempo reale usa algoritmi o parametri diversi rispetto a quando i video vengono memorizzati su disco, spesso accontentandosi di un fattore di compressione apprezzabilmente minore.

Una seconda asimmetria è che il processo codifica/decodifica può non essere invertibile. Cioè quando si comprime un file, lo si trasmette, e poi lo si decomprime, l'utente si aspetta di avere una copia fedele all'originale fino all'ultimo bit. Nel caso di riproduzione multimediale questo requisito non è necessario. È solitamente accettabile che il segnale video ottenuto dopo la codifica e successiva decodifica sia leggermente diverso dall'originale. Quando l'output decodificato non è esattamente uguale all'input originale si dice che il sistema è **infedele** (*lossy*). Se l'input e l'output sono identici il sistema è **fedele** (*lossless*). I sistemi infedeli sono importanti, perché accettare una piccola quantità di perdita di informazione può far guadagnare molto in termini di costi di compressione.

Codifica entropica

Gli schemi di compressione si possono dividere in due categorie generali: codifica entropica e codifica sorgente. Li discuteremo uno dopo l'altro.

La codifica entropica manipola i flussi di bit senza preoccuparsi del loro significato. È una tecnica generale, fedele, completamente reversibile, applicabile a qualsiasi tipo di dato. La illustreremo mediante tre esempi.

Il primo esempio entropico è la **codifica di stringa**. In molti tipi di dato sono comuni sottostringhe di simboli ripetuti (bit, cifre ecc.). Tali sottostringhe possono essere rimpiazzate da una marca speciale che non è contenuta nel dato originale, seguita dal simbolo, seguita da quante volte è ripetuto. Se la marca speciale è contenuta nei dati, viene duplicata. Ad esempio, si consideri la seguente stringa di cifre decimali:

Se usiamo A come marca e usiamo numeri a due cifre per contare le ripetizioni possiamo codificare tale stringa nel modo che segue:

315A01284587A1136354674A02265

In questo caso la codifica di stringa ha dimezzato la lunghezza della stringa di partenza. Le stringhe di simboli tutti uguali sono comuni nei dati multimediali. Il silenzio audio viene spesso rappresentato da sequenze di 0. Nel video capitano sequenze di colore nelle immagini del cielo, dei muri, e di molte superfici piatte. Tutte queste sequenze possono essere compresse in gran misura.

Il secondo esempio di codifica entropica è la **codifica statistica**. Con questo termine si intende un codice corto per rappresentare simboli frequenti e uno lungo per rappresentare quelli meno frequenti. Il codice Morse si basa su tale principio: la lettera E si scrive –, mentre la Q è – .. –, e così via. Questa codifica è usata anche dal codice Huffman e dall'algoritmo Ziv-Lempel usato dal programma Compress di UNIX.

Il terzo esempio di codifica entropica si chiama **CLUT** (**C**olor **L**ook **U**p **T**able – tabella di ricerca del colore). Si consideri un'immagine che usa una codifica RGB con 3 byte/pixel. In teoria l'immagine potrebbe contenere 2^{24} diversi valori di colore. In pratica conterrà di solito un numero molto minore di valori, specie se l'immagine non è una foto ma un cartone animato o generata da computer. Supponiamo che vengano usati solo 256 colori. Si può facilmente conseguire un fattore di compressione triplo costruendo una tabella di 768 byte che elenca i valori RGB dei 256 colori, e quindi rappresentando ciascun pixel come indice entro la tabella. Abbiamo quindi un esempio evidente in cui la codifica è più lenta della decodifica, perché la codifica richiede una ricerca in tabella mentre la decodifica avviene al prezzo di un solo accesso via l'indice.

Codifica sorgente

La codifica sorgente sfrutta proprietà dei dati per comprimerli di più, di solito in modo infedele. Anche qui abbiamo tre esempi. Il primo è la **codifica differenziale**, in cui una sequenza di valori (ad es. audio) viene codificata rappresentando ciascun valore per differenza rispetto al precedente. Un esempio di questa tecnica è la modulazione differenziale di codice a impulsi, vista nel capitolo 2. È una tecnica infedele perché il segnale potrebbe fare un salto così grande tra due valori consecutivi che la differenza non potrebbe essere rappresentata dall'intervallo utilizzabile per esprimere le differenze, quindi verrebbe registrato almeno un valore scorretto e qualche informazione sarebbe perduta.

La codifica differenziale è un tipo di codifica sorgente perché sfrutta il fatto che grossi salti tra due dati consecutivi siano improbabili. Ma non tutte le sequenze di numeri godono di questa proprietà. Ad esempio, si pensi a una lista di numeri di telefono generati a caso e utilizzati per le ricerche di mercato che importunanano la gente a ora di cena. In queste liste le differenze tra numeri di telefono consecutivi richiederanno tanti bit quanti ne servono per i numeri stessi.

Il secondo esempio sono le **trasformazioni**. La compressione può diventare molto più semplice trasformando i segnali da un dominio a un altro. Ad esempio, si consideri la trasformata di Fourier di figura 2-1 (e): rappresenta una funzione del tempo come lista di ampiezze. Avendo i valori precisi delle ampiezze si può ricostruire perfettamente la funzione originale. D'altra parte, se sono note solo le prime 8 ampiezze arrotondate ai primi 2 decimali può comunque essere possibile ricostruire il segnale così bene che l'ascoltatore non si accorge che è stata persa informazione. Il vantaggio è che la trasmissione di 8 ampiezze richiede assai meno bit che trasmettere tutta la forma d'onda.

Le trasformazioni si applicano anche ai dati di immagini a due dimensioni. Si supponga che la matrice 4×4 di figura 7-78 (a) rappresenti i valori a toni di grigio di un'immagine monocromatica.

Possiamo trasformare questi dati sottraendo il valore dell'elemento in alto a sinistra da tutti gli altri, come in figura 7-78 (b). Questa trasformazione potrebbe essere utile se si

Figure 7-78 consists of two parts. Part (a) shows a 4x4 grid of pixel values labeled 'Valore del pixel'. The values are: Row 1: 160, 160, 161, 160; Row 2: 161, 165, 166, 158; Row 3: 160, 167, 165, 161; Row 4: 159, 160, 160, 160. An arrow points to the first cell (160). Part (b) shows the same grid with the top-left value (160) subtracted from all other cells. The resulting values are: Row 1: 0, 1, 0; Row 2: 1, 5, 6, -2; Row 3: 0, 7, 5, 1; Row 4: -1, 0, 1, 0. A bracket above the first column is labeled '4 pixel'.

Fig. 7-78 (a) Valori dei pixel per parte di un'immagine. (b) Trasformata in cui l'elemento in alto a sinistra è sottratto da tutti gli elementi eccetto se stesso.

usa la codifica statistica. Per esempio, i valori tra –7 e +7 si potrebbero codificare con numeri a 4 bit, mentre i valori tra 0 e 255 si potrebbero codificare con un codice speciale a 4 bit (–8) seguito da un numero a 8 bit.

Sebbene tale trasformata sia fedele, ce ne sono altre più utili che non lo sono. Una trasformata spaziale bidimensionale importante è **DCT** (**D**iscrete **C**osine **T**ransformation – trasformata a coseno discreto) (Feig, Winograd, 1992). Questa trasformata ha la proprietà che nel caso di immagini con forti discontinuità la maggior parte del potere di spettro risiede nei primi termini, il che permette di ignorare gli altri senza gravi perdite di informazione. Torneremo tra breve su DCT.

Il terzo esempio di codifica sorgente è la **quantizzazione vettoriale**, che si applica direttamente ai dati di immagine. L'immagine viene divisa in rettangoli di dimensione fissa. Oltre all'immagine, occorre rappresentare anche una tabella di rettangoli delle stesse dimensioni e possibilmente costruita a partire dall'immagine stessa. Tale tabella viene chiamata **libro codice**. Di ciascun rettangolo viene trasmesso solo l'indice ottenuto cercando nel libro codice. Va trasmesso anche quest'ultimo, se viene creato dinamicamente (immagine per immagine). I risparmi sono possibili se esiste un piccolo numero di rettangoli che viene usato più volte nell'immagine.

La figura 7-79 mostra un esempio di quantizzazione vettoriale. In (a) abbiamo una griglia di rettangoli di dimensione non specificata. In (b) abbiamo il libro codice. Il flusso di output è la lista di interi 001022032200400. Ciascuno rappresenta un elemento nel libro codice.

In un certo senso la quantizzazione vettoriale è una generalizzazione della codifica CLUT. La vera differenza si vede quando non si trovano associazioni perfette. Sono possibili tre strategie. La prima è di usare la miglior associazione. La seconda è di usare la miglior associazione più qualche informazione per migliorarla (ad es. il vero valore medio). La terza è di usare la miglior associazione più tutta l'informazione necessaria per permettere al decodificatore di ricostruire perfettamente l'immagine. Le prime due strategie sono infedeli, ma raggiungono ottime prestazioni di compressione. La terza è fedele ma com-

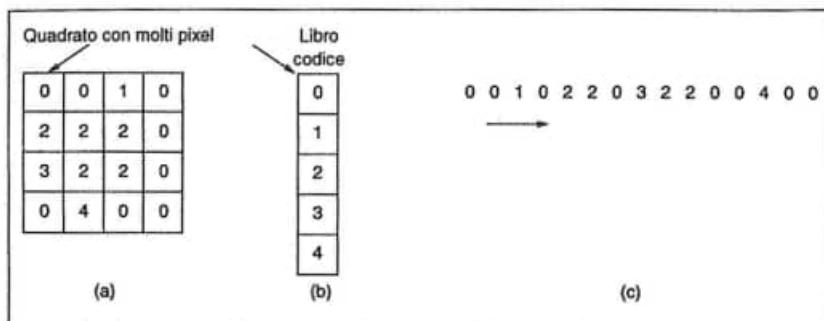


Fig. 7-79 Esempio di quantizzazione vettoriale. (a) Immagine divisa in quadrati. (b) Libro codice dell'immagine. (c) Immagine codificata.

prime di meno. Anche in questo caso la codifica (basata su pattern matching) è molto più costosa della decodifica (accesso in tabella via indice).

Lo standard JPEG

Lo standard **JPEG** (Joint Photographic Experts Group) è stato sviluppato da esperti di fotografia riuniti sotto gli auspici di ITU, ISO e IEC (sono tutte organizzazioni che pubblicano standard) per la compressione di immagini ferme a tono continuo (ad es. le fotografie). È importante perché lo standard multimediale per le immagini in movimento, **MPEG**, è grosso modo la codifica JPEG di ciascuna immagine separatamente, più alcune funzioni extra per la compressione di serie di immagini e il rilevamento del movimento. JPEG è definito nello Standard Internazionale 10918.

JPEG ha quattro modi e molte opzioni. Somiglia più a una lista per la spesa che a un algoritmo. Per i nostri scopi tuttavia è rilevante solo la modalità sequenziale infedele, illustrata in figura 7-80. Ci concentreremo inoltre sul modo in cui JPEG viene usato normalmente per codificare immagini video RGB a 24 bit, mentre trascureremo alcuni dettagli minori per semplificarci la vita.



Fig. 7-80 Le operazioni effettuate da JPEG in modalità sequenziale infedele.

Il passo 1 della codifica JPEG di un'immagine è la preparazione dei blocchi. Più specificamente, assumiamo che l'input a JPEG sia un'immagine RGB 640×480 con 24 bit/pixel, come in figura 7-81 (a).

Siccome se si usano la luminanza e crominanza si ottiene una compressione migliore, prima calcoliamo la luminanza Y , poi le due crominanze I e Q (per NTSC), usando le seguenti formule:

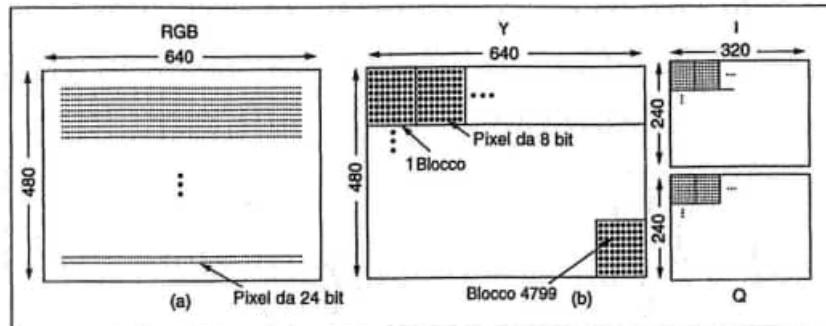


Fig. 7-81. (a) Dati RGB di ingresso. (b) Dopo la preparazione dei blocchi.

$$Y = 0.30R + 0.59G + 0.11B$$

$$I = 0.60R - 0.28G - 0.32B$$

$$Q = 0.21R - 0.52G + 0.31B$$

Nel caso di immagini PAL le crominanze si chiamano U e V e i coefficienti sono diversi, ma l'idea è la stessa. Il SECAM differisce sia da PAL che da NTSC.

Occorre costruire delle matrici per Y , I e Q , i cui elementi variano nell'intervallo 0.255. Poi si fa la media di blocchi quadrati di 4 pixel nelle matrici I e Q per ridurle a dimensione 320×240 . Questa riduzione è infedele, ma l'occhio se ne accorge appena perché è più sensibile alla luminanza che alla crominanza. In questo modo si guadagna un fattore di compressione pari a 2. A questo punto viene sottratto 128 da ciascun elemento delle tre matrici per mettere 0 nel mezzo dell'intervallo. Infine, ogni matrice viene suddivisa in blocchi 8×8 . La matrice Y ha 4800 blocchi; le altre hanno 1200 blocchi ciascuna, come in figura 7-81 (b).

Il passo 2 di JPEG applica una trasformazione discreta coseno (DCT) a ciascuno dei 7200 blocchi separatamente. L'output di ciascun DCT è una matrice 8×8 di coefficienti DCT. L'elemento DCT (0, 0) è il valore medio del blocco. Gli altri elementi dicono quando potere spettrale è presente e ciascuna frequenza spaziale. In teoria la trasformata DCT è fedele, ma in pratica siccome si usano numeri in virgola mobile e funzioni trascendenti si introducono errori di rappresentazione che perdono un po' di informazione. Di solito questi elementi decadono rapidamente allontanandosi dall'origine (0, 0), come si vede in figura 7-82.

Completata la trasformata DCT, JPEG passa al passo 3, la quantizzazione, in cui si buttano via i coefficienti DCT meno importanti. Questa trasformazione infedele viene fatta dividendo ciascun coefficiente nella matrice 8×8 per un peso ottenuto da una tabella. Se tutti i pesi sono 1, la trasformata non fa nulla. Ma se i pesi crescono fortemente a partire dall'origine le frequenze spaziali più alte vengono cancellate rapidamente. La figura 7-83 ne dà un esempio. Vediamo la matrice DCT iniziale, la tabella di quantizzazione, e il risultato ottenuto dividendo ciascun elemento DCT per il corrispondente elemento della tabella di quantizzazione. I valori di tale tabella non fanno parte dello standard JPEG.

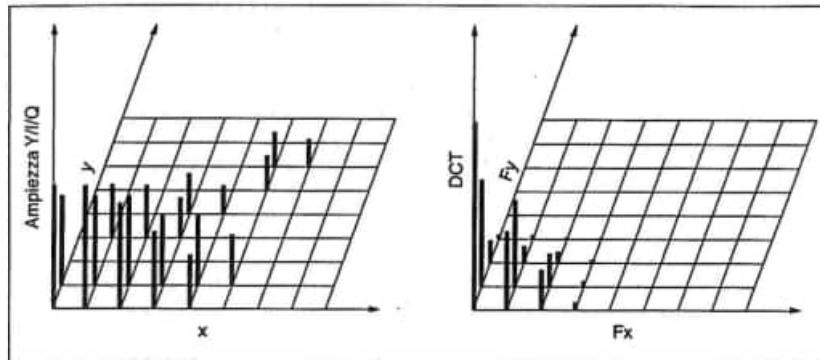


Fig. 7-82 (a) Un blocco della matrice Y. (B) I coefficienti DCT.

Ogni applicazione deve averne una sua propria, che permette di bilanciare la compressione rispetto alla perdita di fedeltà.

Fig. 7-83 Calcolo dei coefficienti DCT quantizzati.

Il passo 4 riduce il valore (0, 0) di ciascun blocco (quello in alto a sinistra) rimpiazzandolo con la sua differenza rispetto all'elemento corrispondente del blocco precedente. Siccome questi elementi sono medie dei loro blocchi, dovrebbero cambiare lentamente, quindi prendendo le differenze si ottengono in genere numeri piccoli. Per gli altri valori non si calcolano le differenze. I valori (0, 0) sono detti componenti DC; gli altri valori sono i componenti AC.

Il passo 5 linearizza i 64 elementi e applica alla lista la codifica di stringa. La scansione del blocco da sinistra a destra e alto-basso non concentra gli zeri, quindi viene usato uno schema di scansione a zigzag, come in figura 7-84.

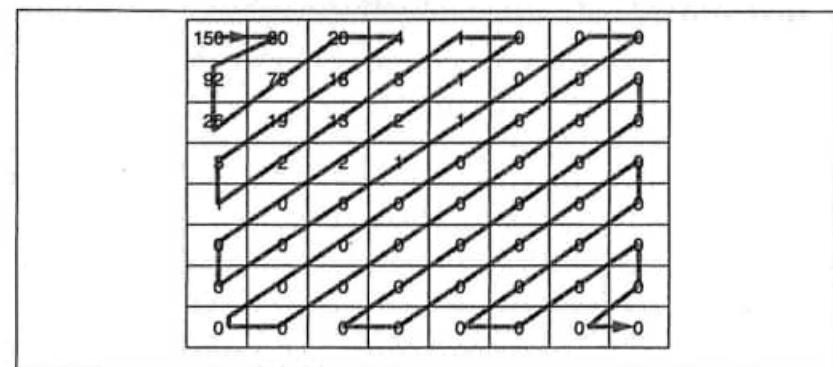


Fig. 7-84 Ordine in cui vengono trasmessi i valori quantizzati.

In questo esempio lo schema a zigzag produce 38 zeri consecutivi alla fine della matrice. La stringa si può ridurre di un colpo dicendo che ci sono 38 zeri.

Abbiamo adesso una lista di numeri che rappresenta l'immagine (nello spazio trasformato). Il passo 6 esegue la codifica di Huffman dei numeri per l'archiviazione o la trasmissione.

JPEG può sembrare complicato, e in effetti lo è. Eppure, siccome spesso comprime di venti volte o anche di più, viene largamente usato. La decodifica di un'immagine JPEG richiede l'esecuzione dell'algoritmo all'indietro. Diversamente da altri algoritmi che abbiamo visto, JPEG è simmetrico: la decodifica prende tanto tempo quanto la codifica.

È interessante notare che a causa delle proprietà matematiche di DCT è possibile eseguire certe trasformazioni geometriche (ad es. la rotazione) direttamente sulla matrice trasformata, senza dover rigenerare l'immagine originale. Queste trasformazioni vengono discusse in Shen, Sethi, (1995). Proprietà analoghe si applicano anche ai dati audio compressi MPEG (Broadhead, Owen, 1995).

Lo standard MPEG

Finalmente veniamo al nocciolo della questione: gli standard **MPEG (Motion Picture Experts Group)**. Sono gli algoritmi principali usati per comprimere i video; sono standard internazionali sin dal 1993. Siccome i film contengono sia immagini che suoni, MPEG

comprime entrambi, ma poiché il video richiede banda maggiore e contiene inoltre più ridondanza esponiamo prima la compressione video.

Il primo standard pubblicato fu MPEG-1 (Standard Internazionale 11172). Il suo obiettivo era di produrre output video di qualità equivalente a quella di un videoregistratore (352 × 240 per NTSC) usando un tasso di bit di 1,2 Mbps. Abbiamo visto prima che il video non compresso necessita di 472 Mbps, quindi ridurlo a 1,2 Mbps non è semplice, anche a bassa risoluzione. MPEG-1 può essere trasmesso su doppino per brevi distanze. Viene anche usato per memorizzare i film sui CDROM nel formato CD-I e CD-Video.

Lo standard successivo è MPEG-2 (Standard Internazionale 13818), progettato inizialmente per comprimere video di qualità equivalente a quella trasmessa via etere in 4 o 6 Mbps, in modo da poter transitare sul canale di broadcast NTSC o PAL. Più tardi MPEG-2 fu esteso per supportare risoluzioni superiori, compreso HDTV. MPEG-4 serve per la videoconferenza a media risoluzione a bassa velocità (10 immagini/s) e bassa banda (64 Kbps). Questo permette videoconferenze su un singolo canale N-ISDN B.

Se la numerazione è questa, potremmo immaginare che il prossimo standard sarà MPEG-8. In realtà ISO usa una numerazione lineare, non esponenziale. Una volta esisteva anche MPEG-3. Doveva essere usato per HDTV, ma tale progetto fu cancellato, e HDTV è stata aggiunta a MPEG-2.

I principi base di MPEG-1 e 2 sono simili, ma i dettagli sono diversi. Grossso modo MPEG-2 è un sovrainsieme di MPEG-1, cui aggiunge i certi formati di pacchetto e certe opzioni di codifica. È probabile che alla lunga MPEG-1 dominerà il mercato dei film su CDROM, mentre MPEG-2 dominerà le trasmissioni a lunga distanza. Discuteremo prima MPEG-1 e poi MPEG-2.

MPEG-1 ha tre componenti: audio, video e sistema, che integra gli altri due, come mostra la figura 7-85.

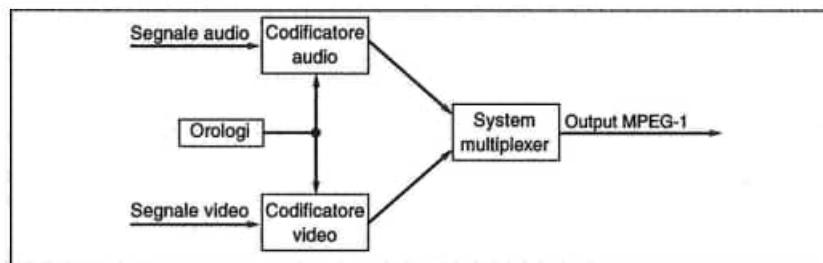


Fig. 7-85 Sincronizzazione dei flussi audio e video in MPEG-1.

I codificatori audio e video funzionano in modo indipendente, e quindi occorre risincronizzare i due flussi dal lato ricevente. Questo problema si risolve mediante un orologio di sistema da 90 kHz che riporta l'orario attuale a entrambi i codificatori. Questi valori sono rappresentati su 33 bit, il che permette a un film di andare avanti per 24 h senza dover reiniziare il ciclo dell'orologio. Le marche temporali vengono incluse nell'output codificato e propagate al ricevitore, che le usa per risincronizzare i flussi audio e video.

La compressione audio MPEG viene effettuata campionando la forma d'onda a 32 kHz, 44,1 kHz o 48 kHz. Può gestire suono mono, stereo disgiunto (ciascun canale viene compresso separatamente) e stereo congiunto (viene sfruttata la ridondanza tra i canali). È strutturato su tre livelli, e tutti applicano ottimizzazioni ulteriori per ottenere maggior compressione a un costo addizionale. Il livello 1 è lo schema fondamentale. Viene usato ad esempio nel sistema di nastri digitali DCC. Il livello 2 aggiunge allo schema base l'allocazione avanzata dei bit. Viene usato per l'audio dei CD-ROM e le piste sonore dei filmati. Il livello 3 aggiunge dei filtri ibridi, una quantizzazione non uniforme, la codifica di Huffman e altre tecniche avanzate.

L'audio MPEG può comprimere un CD di rock'n roll fino a 96 kbps senza alcuna perdita di qualità percepibile, persino per i fan del rock'n roll che non abbiano subito danni all'udito. Per un concerto di pianoforte sono necessari almeno 128 kbps. Questa differenza è dovuta al fatto che il rapporto segnale-rumore è molto più alto per il rock'n roll che per un concerto di pianoforte (in senso ingegneristico, in ogni modo).

La compressione audio è raggiunta effettuando una trasformata veloce di Fourier sul segnale audio per trasformarlo dal dominio del tempo a quello delle frequenze. Lo spettro risultante viene quindi diviso fino a 32 bande di frequenza, ognuna delle quali viene elaborata separatamente. Quando sono presenti due canali stereo, viene anche sfruttata la ridondanza inherente nell'avere due sorgenti audio altamente sovrapponibili. Lo stream audio MPEG-1 risultante è aggiustabile da 32 kbps a 448 kbps. In Pan (1995) viene descritto tale processo in modo introduttivo.

Consideriamo adesso la compressione video MPEG-1. Nei filmati esistono due tipi di ridondanze: spaziale e temporale; MPEG-1 le usa entrambe. La ridondanza spaziale viene utilizzata codificando separatamente ciascun fotogramma con JPEG. Questo approccio viene utilizzato qualche volta, specialmente quando è necessario un accesso casuale a ciascun fotogramma, come nelle produzioni di video alla moviola. In questo modo è possibile ottenere una banda compressa nell'intervallo da 8 a 10 Mbps.

È possibile ottenere compressione aggiuntiva sfruttando il fatto che fotogrammi consecutivi siano spesso identici. Questo effetto è minore di quello che può sembrare in un primo momento, dato che molti di coloro che fanno un film cambiano scena ogni 3 o 4 s (misurate il tempo di un film e contate le scene). Nonostante ciò, anche una sequenza di 75 fotogrammi molto simili offre il potenziale di una riduzione maggiore sulla sola codifica di ogni singolo fotogramma con JPEG.

Per le scene in cui la camera e lo sfondo sono ferme e uno o due attori si muovono lentamente, quasi tutti i pixel saranno identici tra un fotogramma e l'altro. In questo caso, andrà bene sottrarre ciascun fotogramma dal precedente ed eseguire JPEG sulla differenza. Comunque, per scene in cui la camera fa panoramiche oppure zoom, questa tecnica fallisce miseramente. Ciò che serve è un modo per compensare questo movimento. Questo è esattamente ciò che fa MPEG; è la principale differenza tra MPEG e JPEG. L'uscita MPEG-1 consiste di quattro tipi di fotogrammi:

1. Fotogrammi I (intracodificato): immagini ferme autocontenute codificate con JPEG.
2. Fotogrammi P (profetici): differenza blocco-a-blocco con l'ultimo fotogramma.

3. Fotogrammi B (bidirezionali): differenze con l'ultimo e il prossimo fotogramma.
4. Fotogrammi D (codificati DC): medie a blocchi utilizzate per l'avanti veloce.

I fotogrammi I sono codificati mediante JPEG, utilizzando anche la luminanza ad alta risoluzione e la crominanza a metà risoluzione lungo ogni asse. È necessario che nell'uscita vi siano dei fotogrammi I per tre ragioni. Primo, MPEG-1 può essere utilizzato per la trasmissione multicast, in cui chi riceve si sintonizza a suo piacimento. Se tutti i fotogrammi dipendessero dai loro predecessori fino al primo fotogramma, chiunque perdesse il primo fotogramma non potrebbe mai decodificare un qualsiasi fotogramma che lo seguisse. Secondo, se un fotogramma ricevuto è errato, non sarebbe possibile effettuare nessun'altra decodifica. Terzo, senza fotogrammi I, durante un avanti o un indietro veloce, il decodificatore dovrebbe calcolare ogni fotogramma che passa in modo da conoscere il vero valore di quello su cui ci si ferma. Per queste ragioni, vengono inseriti uno o due fotogrammi I al secondo nell'uscita.

I fotogrammi P, al contrario, codificano le differenze tra i fotogrammi. Si basano sull'idea dei **macroblocchi**, che ricoprono 16×16 pixel nello spazio di luminanza e 8×8 pixel nello spazio di crominanza. Un macroblocco viene codificato ricercando per esso il fotogramma precedente o un altro che ne differisca di poco.

La figura 7-86 contiene un esempio nel quale i fotogrammi P sono utili. Qui vediamo tre fotogrammi consecutivi che hanno lo stesso sfondo, ma differiscono per la posizione della persona. I macroblocchi che contengono la scena dello sfondo coincideranno esattamente, mentre i macroblocchi contenenti la persona differiranno nella posizione per un certo valore e dovranno venire tracciati.

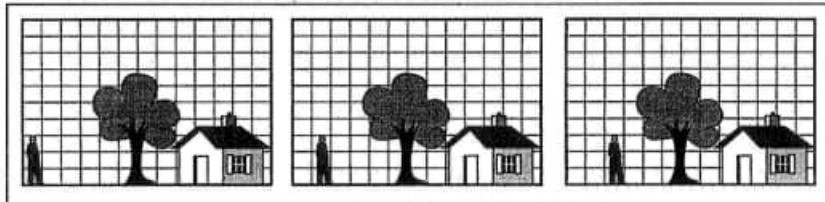


Fig. 7-86 Tre fotogrammi consecutivi.

Lo standard MPEG-1 non specifica come e quanto lontano si debba cercare oppure quanto deve essere buona una corrispondenza usata per contare. Questo dipende dalla singola implementazione. Ad esempio, un'implementazione potrebbe cercare un macroblocco alla posizione corrente nel fotogramma precedente, e tutte le altre posizioni differiscono di $\pm\Delta x$ nella direzione x e di $\pm\Delta y$ nella direzione y. Per ogni posizione andrebbe calcolato il numero delle corrispondenze nella matrice di luminanza. La posizione con il valore maggiore verrebbe dichiarata vincente, grazie al fatto che supera una certa soglia predefinita. In caso contrario, il macroblocco verrebbe classificato come mancante. Ovviamente sono possibili molti altri algoritmi più sofisticati.

Se viene trovato un macroblocco, questo viene codificato prendendo la differenza col suo

valore nel fotogramma precedente (sia per luminanza che per crominanza). Queste matrici delle differenze sono quindi sottoposte a una trasformata DCT, quantizzazione, codifica di stringa e codifica di Huffman, proprio come per JPEG. Il valore del macroblocco nello stream di uscita è quindi il vettore di movimento (quanto si sposta il macroblocco in ogni direzione rispetto alla sua posizione precedente), seguito dalla lista dei numeri con codifica di Huffman. Se nel fotogramma precedente il macroblocco non si trova, il valore corrente viene codificato con JPEG, proprio come in un fotogramma I.

Chiaramente questo algoritmo è fortemente asimmetrico. Ogni implementazione è libera di provare qualsiasi posizione plausibile nel fotogramma precedente se lo vuole, nel disperato tentativo di localizzare fino all'ultimo macroblocco. Questo approccio minimizzerà lo stream MPEG-1 codificato al costo di una elaborazione molto lenta. Questo può andare bene per la codifica una tantum di una libreria di film, ma sarebbe terribile per una videoconferenza in tempo reale.

Allo stesso modo, ogni implementazione è libera di decidere cosa costituisca un macroblocco "ritrovato". Questa libertà permette agli implementatori di competere sulla qualità e velocità dei loro algoritmi, ma produce comunque output compatibile con MPEG-1. Qualunque sia l'algoritmo di ricerca che viene usato, il risultato finale è o la codifica JPEG del macroblocco corrente, o la codifica JPEG della differenza tra il macroblocco corrente e quello dell'immagine precedente a una distanza specificata da quello corrente. Fin qui la decodifica MPEG-1 è semplice. La decodifica di immagini I è la stessa della decodifica di immagini JPEG. La decodifica di immagini P richiede al decodificatore di memorizzare l'immagine precedente e poi costruire la nuova usando i macroblocchi completamente codificati e quelli che contengono le differenze con l'immagine precedente, la nuova immagine viene costruita macroblocco per macroblocco.

Le immagini B sono simili a quelle P, solo che è ammesso referenziare macroblocchi in un'immagine precedente o successiva. Questo maggior grado di libertà migliora la compensazione del movimento, ed è utile anche quando un oggetto passa davanti o dietro a un altro oggetto. La codifica delle immagini B richiede di tenere in memoria tre immagini alla volta: quella passata, quella presente e quella futura. Sebbene diano la miglior compressione, non tutte le implementazioni supportano le immagini B.

Le immagini D si usano solo per rendere possibile visualizzare un'immagine a bassa risoluzione durante un passo veloce di moviola avanti o indietro. Eseguire in tempo reale la normale decodifica MPEG-1 è abbastanza complesso. Aspettarsi che il decodificatore la esegua e per di più mostri su video le immagini a 10 volte la velocità normale vuol dire chiedere troppo. Si usano invece le immagini D per avere una riproduzione a bassa risoluzione. Ciascun elemento è la media di un blocco senza codifica, quindi è facile la sua visualizzazione in tempo reale. Questa funzione è importante per permettere all'utente di cercare una scena particolare ad alta velocità.

Ora che abbiamo esposto MPEG-1 passiamo a MPEG-2. La codifica è simile a quella di MPEG-1, con immagini I, P e B, ma senza quelle D. Inoltre la trasformata DCT è 10×10 invece che 8×8 , per avere il 50% in più di coefficienti, quindi una miglior qualità. Siccome MPEG-2 serve per le trasmissioni TV e per i CDROM, supporta immagini sia progressive che interallacciate, mentre MPEG-1 solo quelle progressive. Esistono altre differenze meno importanti.

Invece di supportare un solo livello di risoluzione MPEG-2 ne ha quattro: bassa (352×240), principale (720×480), alta-1440 (1440×1152) e alta (1920×1080). La bassa risoluzione serve per i videoregistratori e come compatibilità all'indietro con MPEG-1. Quella principale si usa per trasmissioni NTSC, mentre le altre due servono per la TV ad alta definizione (HDTV).

Oltre ai quattro livelli di risoluzione MPEG-2 ha 5 profili, uno per area di applicazione. Il profilo principale è di uso generico; verosimilmente molti processori saranno ottimizzati per gestire il profilo principale e il relativo livello di risoluzione. Il profilo semplice è come quello principale, ma esclude l'uso di immagini B, per semplificare la codifica e decodifica a software. Gli altri profili servono per la scalabilità e l'HDTV. Le differenze tra i profili riguardano la presenza delle immagini B, la risoluzione di crominanza e la scalabilità dello stream di bit codificato verso altri formati.

Il tasso di dati compressi per ciascuna combinazione di risoluzione e profilo è diverso. Si va da circa 3 Mbps a 100 Mbps per la TV ad alta definizione. Il caso normale è da 3 a 4 Mbps. In Pancha, El Zarki (1994) si trovano alcuni dati sulle prestazioni di MPEG. MPEG-2 gestisce audio e video in multiplex in modo più generale rispetto al modello MPEG-1 di figura 7-85. Definisce un numero illimitato di stream elementari, che comprendono video e audio, ma anche stream di dati che devono essere sincronizzati, ad esempio sottotitoli in più linguaggi. Ogni stream viene prima impacchettato con marche temporali. Un semplice esempio con due stream è mostrato in figura 7-87.

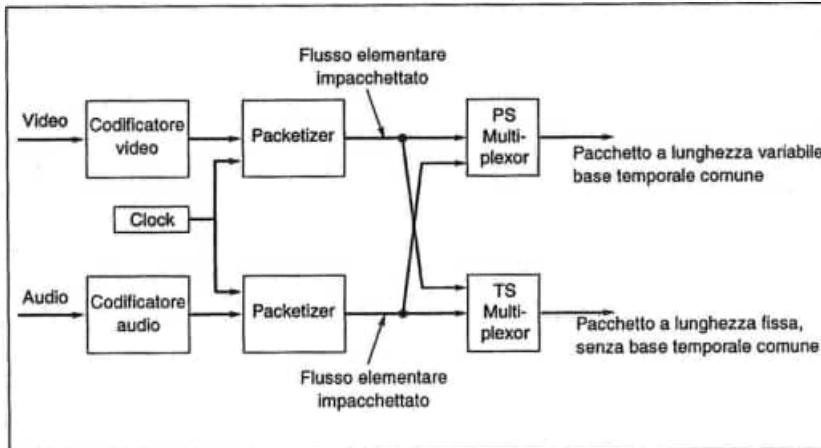


Fig. 7-87 Multiplexing di due stream in MPEG-2.

L'uscita di ogni elemento che impacchetta è un PES (Packet Elementary Stream). Ogni pacchetto PES ha circa 30 campi e flag di intestazione, compresi le lunghezze, gli identificatori di stream il controllo della codifica, lo stato del copyright, le marche di tempo e un CRC. Gli stream PES per audio, video e possibilmente per dati vengono quindi trasmessi assieme in multiplex su un singolo stream di uscita.

Lo stream di programma di MPEG-2 è simile agli stream dei sistemi MPEG-1 di figura 7-85. Viene usato per inviare insieme via multiplex gli stream elementari che hanno una base temporale comune e che vanno visualizzati in modo sincronizzato. Lo stream di programma utilizza lunghi pacchetti a lunghezza variabile.

L'altro stream MPEG-2 è lo stream di trasporto. Viene usato per inviare via multiplex gli stream (compresi gli stream di programma) che non usano una base temporale comune. I pacchetti dello stream di trasporto hanno lunghezza fissa (188 byte), in modo che sia più semplice circoscrivere l'effetto di pacchetti danneggiati o persi durante la trasmissione.

È bene notare che tutti gli schemi di codifica che abbiamo discusso sono basati su un modello di codifica infedele trasmessa in modo fedele. Né JPEG né MPEG, ad esempio, sono in grado di recuperare bene in seguito a una perdita o a un danneggiamento di pacchetti. Un approccio differente per la trasmissione di immagini è quello che le trasforma in modo da separare le informazioni importanti da quelle meno importanti come fa il DCT. Quindi aggiunge una considerevole quantità di ridondanza (addirittura duplica i pacchetti) alle informazioni importanti e niente a quelle meno importanti. Se qualche pacchetto viene perso o danneggiato, è comunque possibile visualizzare in maniera ragionevole le immagini senza che si debbano ritrasmettere. Queste idee sono descritte più in dettaglio in Danskin *et al.* (1995). Sono applicabili in special modo alle trasmissioni multicast, dove è impossibile avere un riscontro da ciascun ricevente.

7.7.4 Video su richiesta

Il video su richiesta viene talvolta paragonato a un negozio virtuale che noleggia cassette video. Nei negozi reali l'utente (il cliente) seleziona uno qualunque tra i tanti video disponibili e lo porta a casa per vederlo. Invece con il video su richiesta la selezione viene fatta da casa utilizzando il telecomando del televisore, e il video inizia immediatamente. Non è necessario andare al negozio. Ovviamente implementare il video su richiesta è un po' più complicato che descriverlo. In questo paragrafo, introdurremo le idee base e la loro implementazione. Una descrizione di una implementazione reale si può trovare in Nelson, Linton, (1995). Una trattazione più generale della televisione interattiva è in Hodge (1995). Altri riferimenti interessanti sono Chang *et al.* (1994); Hodge *et al.* (1993); Little, Verkatesh (1994).

Il video su richiesta è simile al noleggio di una cassetta, o è molto più simile alla scelta di un film in un sistema con 500 o 5000 canali? La risposta ha delle importanti implicazioni tecniche. In particolare, gli utenti che noleggiano video sono abituati all'idea di poter fermare un video, fare un rapido giro in cucina o nel bagno, e quindi riprendere da dove ci si era fermati. Chi guarda la televisione non pensa di fermare i programmi per un po' di tempo.

Se il video su richiesta vuole competere con successo con i negozi di noleggio video, è necessario che agli utenti sia permesso fermare, iniziare e riavvolgere i video a piacere. Dare agli utenti questa funzione costringe in pratica i fornitori di video a trasmettere a ciascun cliente una copia privata.

D'altra parte, se il video su richiesta viene visto come un sistema televisivo avanzato, potrebbe essere sufficiente che i fornitori inizino la proiezione dei film più popolari,

diciamo, ogni 10 min, e li trasmettano senza interruzioni. Un utente che voglia vedere un film popolare potrebbe dover aspettare al massimo 10 min perché questo inizi. Anche se non è possibile la funzione di pausa/riavvio, lo spettatore che ritorna nel soggiorno dopo una breve pausa può selezionare un altro canale che visualizza lo stesso film ma con 10 min di ritardo. Alcune scene andrebbero riviste, ma non si perderebbe niente. Questo schema è chiamato **video su semirichiesta**. Offre lo stesso potenziale a un costo molto minore, in quanto la stessa proiezione può raggiungere più utenti contemporaneamente. La differenza tra video su richiesta e semirichiesta è simile alla differenza che c'è tra guidare la macchina e prendere l'autobus.

Guardare i film su semirichiesta non è che uno tra un gran numero di nuovi servizi che saranno potenzialmente disponibili una volta che sarà realizzata la rete a larga banda. Il modello generale è illustrato in figura 7-88. Vediamo al centro del sistema una rete su supporto ad alta banda, che collega una vasta area (nazionale o internazionale). A essa sono connessi migliaia di reti locali di distribuzione, come sistemi di distribuzione di TV via cavo o della società telefonica. I sistemi di distribuzione locali entrano nelle case delle persone, dove arrivano in **scatole set-top**, che di fatto sono potenti personal computer specializzati.

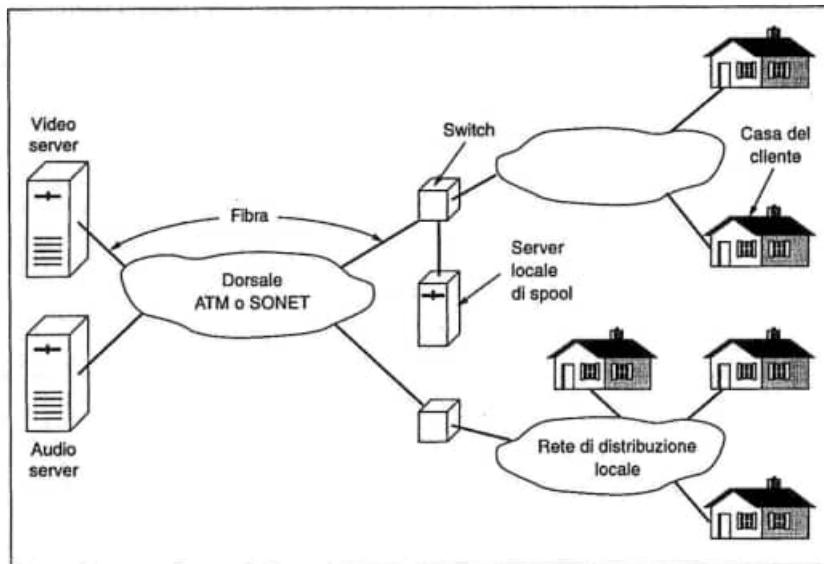


Fig. 7-88 Schema generico di un sistema di video su richiesta.

Al supporto ad alta banda di fibre ottiche sono attaccati migliaia di fornitori di informazioni. Alcuni di questi offriranno video a pagamento o CD audio a pagamento. Altri offriranno dei servizi specializzati, come un servizio di acquisti a casa (con la possibilità di ruotare una lattina di minestra e ingrandire la lista degli ingredienti oppure vedere un breve filmato su come guidare una falciatrice per prati alimentata a benzina). Senza

dubbio in breve tempo sarebbero disponibili sport, notizie, repliche di serial, accesso al WWW e innumerevoli altre possibilità.

Inoltre nel servizio sarebbe incluso anche un server locale in grado di permettere all'utente di riposizionare i video a suo piacimento, in modo da salvare banda durante le ore di punta. Come queste componenti possano essere assemblate e chi è proprietario di cosa sono argomenti di dibattito all'interno dell'industria.

Qui di seguito esamineremo il progetto delle parti principali del sistema: i server di video, la rete di distribuzione e le scatole set-top.

I server di video

Per avere video su (semi-) richiesta, abbiamo bisogno di **server di video** capaci di memorizzare e riprodurre un gran numero di film simultaneamente. Il numero totale di film è di circa 65.000 (Minoli, 1995). Se compressi con MPEG-2, un film normale occupa circa 4 GB di memoria, perciò 65.000 film richiederebbero circa 260 TB (terabyte). Aggiungete a questi tutti i vecchi programmi televisivi, i filmati sportivi, i servizi di informazione, i cataloghi di vendite presentati ecc., ed è chiaro che abbiamo tra le mani un problema enorme di memoria. Il modo più economico di memorizzare dei grossi volumi di informazione è su nastro magnetico. È sempre stato così e probabilmente lo sarà sempre. Un nastro DAT può memorizzare 8 GB (due film) al costo di circa 5 dollari/GB. Sono attualmente disponibili in commercio grossi server meccanici che gestiscono migliaia di nastri e che hanno un braccio robotizzato per recuperare un qualsiasi nastro e inserirlo nel drive. Con questi sistemi il problema è il tempo di accesso (specialmente per il secondo film su nastro), la velocità di trasmissione e il numero limitato di drive di nastro (per servire contemporaneamente n film, l'unità avrà bisogno di n drive).

Fortunatamente, l'esperienza con i negozi di noleggio, le biblioteche pubbliche e simili dimostrano che non tutte le cassette hanno la stessa popolarità. Sperimentalmente, quando ci sono N film disponibili, la frazione di tutte le richieste per il k -esimo in ordine di popolarità è di circa C/k (Chervenak, 1994). Calcoliamo C normalizzando la somma a 1, cioè

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + \dots + 1/N)$$

In questo modo il film più popolare è sette volte più popolare del settimo film. Questo principio si chiama **legge di Zipf** (Zipf, 1949).

Il fatto che alcuni film siano molto più popolari di altri suggerisce una possibile soluzione in forma di gerarchia di memorizzazione, come in figura 7-89. Qui le prestazioni crescono dal basso verso l'alto.



Fig. 7-89 Una gerarchia di memorizzazione per server di video.

Un'alternativa al nastro è il disco ottico. Al momento i CD-ROM hanno una capacità di soli 650 MB, ma la prossima generazione conterrà circa 4 GB, e così saranno adatti per la distribuzione di filmati MPEG-2. Sebbene il tempo di posizionamento sia lento in confronto a quello dei dischi magnetici (100 ms contro 10 ms), il loro basso costo e la loro alta affidabilità fa dei juke box di dischi ottici con migliaia di CD-ROM una buona alternativa al nastro per i film che sono molto più visti.

Poi vengono i dischi magnetici. Questi hanno un tempo di accesso breve (10 ms), un'alta velocità di trasferimento (10 MB/s) e una capacità considerevole (10 GB), che li rende adatti a contenere i film che vengono attualmente trasmessi (in contrasto a quelli appena memorizzati nell'ipotesi che qualcuno li volesse mai vedere). Il rovescio della medaglia è l'alto costo per la memorizzazione di quei film che vengono richiesti di rado.

In cima alla piramide della figura 7-89 si trova la RAM: è il mezzo di memorizzazione più veloce, ma anche il più costoso. È adatta a quei film dei quali parti diverse vengono inviate a destinatari diversi nello stesso momento (ad es., il vero video su richiesta a 100 utenti che iniziano tutti la proiezione in momenti differenti). Quando il prezzo della RAM crollerà a 10 dollari/MB, un film di 4 GB richiederà 40.000 dollari di RAM, perciò avere 100 film in RAM costerà 4.000.000 di dollari per 400 GB di memoria. Eppure, il costo di un server di video di 10.000.000 di dollari si potrebbe ammortizzare se ciascun video venisse pagato simultaneamente da un numero sufficiente di clienti.

Dato che un server di video è di fatto solo un grosso dispositivo di I/O in tempo reale, ha bisogno di un'architettura hardware e software differente da quella di un PC o di un elaboratore UNIX. L'architettura hardware di un tipico server di video è illustrata in figura 7-90. Il server possiede una o più CPU RISC ad alte prestazioni, ciascuna con la sua memoria locale, una memoria principale condivisa, una grossa cache RAM per i film popolari, una varietà di dispositivi di memorizzazione per la raccolta di film e hardware per la rete, di solito un'interfaccia ottica verso una rete ATM (o SONET) a OC-3 o maggiore.

Questi sottosistemi sono connessi tramite un bus a velocità estremamente alta (almeno 1 GB/s).

Diamo una breve occhiata al software per i server di video. La CPU viene usata per accettare le richieste degli utenti, per localizzare i film, per spostare i dati tra i dispositivi, per fatturare i clienti e per molte altre funzioni. Alcune di queste non hanno problemi sui tempi, mentre molte altre sì, perciò su alcune CPU, se non su tutte, gireranno dei sistemi operativi con caratteristiche di tempo reale, come microkernel con caratteristiche di tempo reale. Di solito questi sistemi suddividono il lavoro in piccoli compiti, ognuno con un tempo di terminazione noto. Lo scheduler può quindi usare una politica del tipo "selezione del compito con la scadenza più vicina" oppure "l'algoritmo di velocità monotona" (Liu, Layland, 1973).

Il software della CPU inoltre definisce la natura dell'interfaccia che il server presenta ai clienti (server di spool e scatole set-top). Sono due gli schemi più usati. Il primo è un file system tradizionale, nel quale i clienti possono aprire, leggere, scrivere e chiudere i file. A parte la complicazione introdotta dalla gerarchia di memorizzazione e dalle considerazioni sul tempo reale, un server di questo tipo può avere un file system su modello di quello di UNIX.

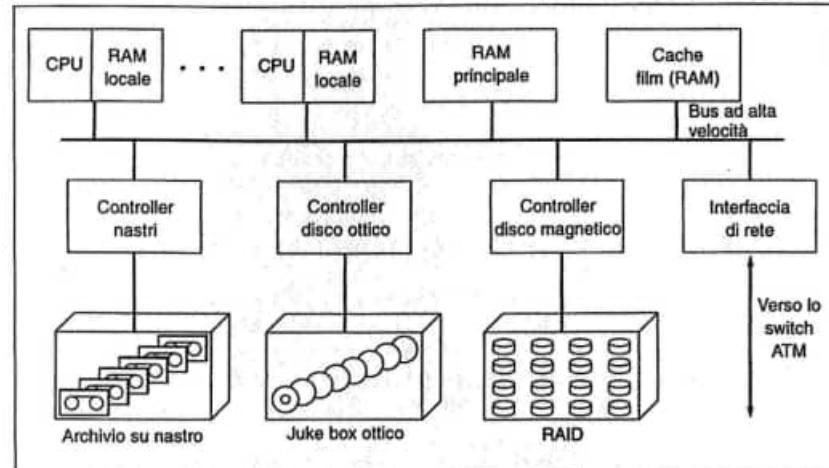


Fig. 7-90 L'architettura hardware di un tipico server di video.

Il secondo tipo di interfaccia si basa sul modello del videoregistratore. I comandi al server richiedono di aprire, riprodurre, fermare, mandare avanti velocemente e riavvolgere i file. La differenza col modello UNIX è che una volta dato un comando PLAY, il server deve solamente inviare i dati a velocità costante, senza che occorrono nuovi comandi.

Il cuore del software per server di video è la gestione dei dischi. Ha due compiti principali: porre i film sul disco magnetico quando devono essere recuperati dal supporto ottico o dal nastro, e gestire le richieste al disco per più stream di uscita. Il posizionamento dei film è importante perché può influenzare notevolmente le prestazioni.

Due modi possibili per organizzare la memorizzazione su disco sono la farm di dischi e l'array di dischi.

Con la **farm di dischi**, ogni drive mantiene alcuni film per intero. Per ragioni di efficienza e di affidabilità, ogni film dovrebbe essere presente almeno su due drive, meglio se più. L'altro tipo di organizzazione è l'**array di dischi** o **RAID** (**Redundant Array of Inexpensive Disks**), nel quale ogni film viene distribuito su più drive, ad esempio, il blocco 0 sul disco 0, il blocco 1 sul disco 1 e così via, con il blocco $n - 1$ sul disco $n - 1$. In seguito il ciclo si ripete, con il blocco n sul disco 0 e così via. Questa organizzazione è detta **striping**.

Un array di dischi in striping ha diversi vantaggi rispetto a una farm di dischi. Innanzitutto tutti gli n drive possono girare in parallelo, aumentando le prestazioni di un fattore n . Secondo, è possibile inserire della ridondanza aggiungendo un drive in più per ogni gruppo di n , dove il drive ridondante contenga l'OR ESCLUSIVO blocco per blocco degli altri drive, in modo da consentire il ripristino dei dati nel caso in cui un drive si dovesse guastare. Infine, è risolto il problema del bilanciamento del carico (non è necessario il posizionamento manuale per evitare di avere tutti i film più popolari sullo stesso drive). D'altra parte, l'organizzazione con l'array di dischi è più complicata di una farm di dischi.

ed è molto più sensibile ai guasti multipli. Inoltre non è adatta alle tipiche operazioni di un videoregistratore come il riavvolgimento e l'avanti veloce di un film. Il testo di Cher-venak *et al.* (1995) è uno studio di simulazione che confronta le due organizzazioni. La ricerca dei blocchi sui dischi è strettamente collegata al posizionamento dei blocchi. Di solito lo schema UNIX di albero non bilanciato di blocchi disco puntati dall'i-nodo non è accettabile perché i file video sono enormi, perciò è possibile localizzare molti blocchi solamente passando attraverso blocchi indiretti tripli, il che significa molti accessi in più al disco (Tanenbaum, 1992). Invece, è normale collegare tra loro i blocchi in una lista con singoli o doppi puntatori. A volte viene anche usato un indice stile UNIX (i-nodi) per permettere un accesso casuale.

L'altro compito del software di disco è servire tutti gli stream di uscita in tempo reale, rispettando le loro limitazioni di tempo. Uno stream video MPEG-2 a 25 immagini/s ha bisogno di recuperare e trasmettere circa 14 KB ogni 40 ms, mentre la quantità reale varia considerevolmente perché le immagini I, P e B hanno percentuali di compressione diverse. Di conseguenza, per mantenere una velocità di uscita uniforme, è necessaria la "bufferizzazione" a entrambi i lati dello stream.

In figura 7-91 vediamo una scala che mostra la quantità totale di dati recuperati dal disco per un dato stream video (assumendo che il film sia su disco). Si muove verso l'alto con salti discreti, un salto per ogni blocco letto. Ciò nonostante, la trasmissione deve avvenire a velocità costante, perciò il processo di lettura da disco deve tener conto del processo di trasmissione. La regione ombreggiata della figura mostra i dati recuperati dal disco ma non ancora trasmessi.

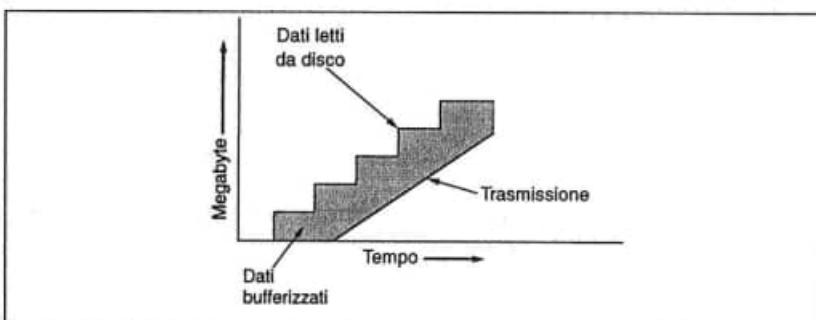


Fig. 7-91 Bufferizzazione del disco nel server.

Di solito i dischi vengono gestiti usando l'algoritmo dell'ascensore, che attiva il braccio muovendolo verso l'interno e continua fino al cilindro centrale, servendo le richieste in ordine di cilindro. Quando arriva a fine corsa, il braccio inverte il movimento ed elabora le richieste pendenti in ordine inverso. Questo algoritmo minimizza il tempo di seek, ma non offre garanzie di efficienza in tempo reale, quindi non è adatto per un server di video. Un algoritmo migliore tiene traccia di tutti gli stream video e fa una lista del blocco successivo richiesto da ciascuno stream. Tali numeri di blocco sono ordinati e letti in

ordine di cilindro. Quando viene letto l'ultimo blocco il giro successivo comincia dal numero del blocco che si trova in testa a ciascuno stream. Tali numeri vengono ordinati e letti in ordine di cilindro, e così via. Questo algoritmo offre prestazioni efficaci in tempo reale e minimizza il tempo di seek in confronto a un algoritmo FIFO puro.

Un altro problema software è il controllo dell'ammissione. Quando arriva una richiesta di attivare un nuovo stream, può essere accettata senza danneggiare le prestazioni degli stream esistenti? Un algoritmo che permette di decidere questa questione esamina il caso peggiore per vedere se, conoscendo le prestazioni della CPU, della RAM e del disco, sia possibile in modo certo passare da k a $k + 1$ stream. Un altro algoritmo usa semplicemente delle informazioni statistiche.

Un altro problema del software del server è la gestione della visualizzazione durante l'avanti o indietro veloci (per cercare scene). Le immagini D offrono l'informazione necessaria per MPEG-1, ma a meno che non vengano marcate e conservate in modo speciale il server non riuscirà a trovarle senza decodificare tutto lo stream; normalmente i server non effettuano la decodifica MPEG durante la trasmissione. Un diverso meccanismo è necessario per MPEG-2, almeno per semplificare la ricerca e la decodifica delle immagini I.

Infine, è un problema la crittazione. Quando i film vengono trasmessi in multicast, ad esempio quando la rete locale distributiva è la TV via cavo, è necessaria la crittazione per garantire che i film vengano visti solo da chi paga per essi. Sono possibili due soluzioni: precrittazione oppure crittazione in linea. Se i film vengono memorizzati in forma crittata allora chiunque scopra la chiave può poi usarla altre volte perché tutti i film usano la stessa chiave. È più sicuro crittare ogni stream separatamente, ma costa di più in termini di risorse di calcolo.

Un ultimo problema è la gestione delle chiavi. Di solito si usa la crittazione in linea con un algoritmo semplice, ma conviene anche cambiare spesso la chiave, in modo che anche se un intruso la scoprissse con 10 min di calcoli, alla fine sarebbe già obsoleta.

La rete distributiva

La rete distributiva è l'insieme dei commutatori e delle linee tra la sorgente e la destinazione. Come abbiamo visto in figura 7-88 consiste in una dorsale SONET o ATM (o ATM su SONET) connessa a una rete distributiva locale. Di solito la dorsale è commutata mentre la rete distributiva locale no.

La dorsale ha come requisito principale banda alta e jitter basso. Se la dorsale è SONET è facile conseguire tali requisiti: la banda è garantita e lo jitter è 0 perché la rete è sincrona. Se la dorsale è ATM o ATM su SONET la qualità del servizio è molto importante. È gestita mediante l'algoritmo del secchio bucato e tutte le altre tecniche che abbiamo studiato dettagliatamente nel capitolo 5, quindi non ripeteremo quella discussione. Altre informazioni su MPEG real-time su dorsali ATM sono contenute in Dixit, Skelly (1995); Morales *et al.* (1995). Affrontiamo adesso un argomento che abbiamo sinora toccato appena: la rete di distribuzione locale.

La distribuzione locale è molto caotica: società diverse usano reti diverse in regioni diverse. Le società telefoniche, di TV via cavo, e tutti gli altri sono ben convinti che chiunque arrivi primo sarà il gran vincitore, quindi assistiamo alla proliferazione delle

tecnologie che vengono installate. I quattro schemi principali di distribuzione locale per il video a richiesta sono noti mediante gli acronimi ADSL, FTTC, FTTH e HFC. Li spiegheremo uno per uno.

ADSL (Asymmetric Digital Subscriber Line) è stato il primo tentativo (Chen, Waring 1994). L'idea è che ogni casa del mondo occidentale possiede già un doppino per il servizio telefonico analogico. Se si potessero usare tali cavi per il video a richiesta le società telefoniche avrebbero già vinto.

Il problema ovviamente è che questi cavi non consentono di trasmettere nemmeno MPEG-1 sulla loro estensione tipica di 10 km, figurarsi MPEG-2. La soluzione ADSL sfrutta gli avanzamenti nell'elaborazione del segnale digitale per eliminare gli echi e altri rumori di linea. La figura 7-92 mostra che ogni abbonato ADSL ha in casa un'unità ADSL che contiene un processore che elabora il segnale digitale. Dall'altra parte del cavo c'è un'altra unità ADSL. Questa può trovarsi o nell'ufficio terminale della società dei telefoni oppure, se il collegamento locale è troppo lungo, alla fine di una fibra ottica nelle vicinanze della casa.

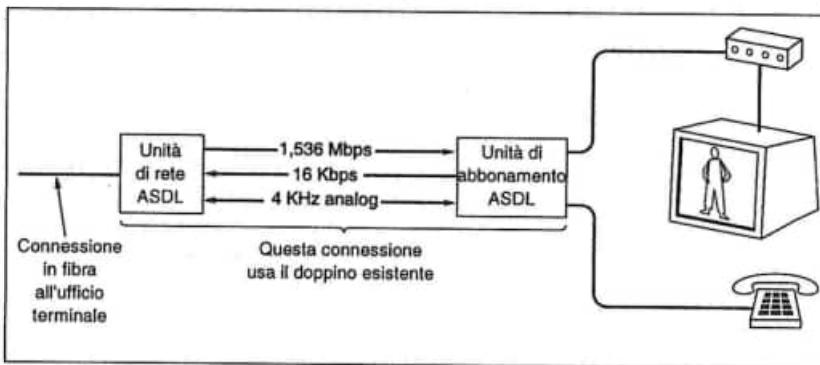


Fig. 7-92. Rete locale distributiva di tipo ADSL.

ADSL-1 ha un canale di downlink a 1,536 Mbps (T1 meno il 193-esimo bit), ma per l'uplink ha solo 16 kbps. Inoltre c'è anche il vecchio canale analogico telefonico a 4 kHz (oppure in certi casi due canali digitali N-ISDN). L'idea è che il canale di uplink è sufficiente per le ordinazioni di film da parte dell'utente, mentre quello di downlink ha banda sufficiente per trasmettere film codificati in MPEG-1. ADSL va considerata come una pezza veloce e sporca più che una soluzione a lungo termine, ma viene installata in parecchie città. Esistono versioni migliorate che si chiamano ADSL-2 e ADSL-3. Quest'ultima permette di trasmettere MPEG-2 su circuiti locali al massimo di 2 km.

Il secondo schema utilizzato dalle società telefoniche è **FTTC (Fiber To The Curb)**. Abbiamo già descritto questo schema in figura 2-23 (a). Con FTTC la società telefonica stende una fibra ottica dall'ufficio terminale in ogni quartiere residenziale, terminandola con un dispositivo che si chiama **ONU (Optical Network Unit)**. In figura 2-23 la ONU è stata chiamata "scatola di giunzione". Circa 16 circuiti locali di rame possono terminare in una

ONU. Questi circuiti sono adesso così corti che supportano T1 o T2 full duplex, e quindi film rispettivamente MPEG-1 o MPEG-2. Inoltre, viene resa possibile la videoconferenza per coloro che lavorano in casa e in piccole aziende, perché FTTC è simmetrico.

La terza soluzione che usano le società telefoniche è di portare direttamente in casa la fibra ottica. Questo sistema si chiama **FTTH (Fiber To The Home)**. In questo schema tutti possono avere OC-1, OC-2 o anche di più. FTTH è molto costoso e passeranno anni prima che si diffonda, ma chiaramente quando succederà ci saranno nuove possibilità di servizio. ADSL, FTTC e FTTH sono tutte reti di distribuzione locale punto-a-punto, il che non ci sorprende tenendo conto dell'organizzazione attuale del sistema telefonico. Una soluzione completamente diversa è **HFC (Hybrid Fiber Coax)**, che è la soluzione al momento preferita dalle società che forniscono il servizio di TV via cavo. È stata illustrata in figura 2-23 (b). La storia è all'incirca questa. I cavi coassiali da 300 e 450 MHz verranno rimpiazzati da cavi a 750 MHz, aumentando la capacità da 50-75 canali a 125 canali da 6 MHz. Di questi 125, 75 vengono usati per trasmettere TV analogica. Gli altri 50 canali vengono modulati ciascuno usando QAM-256, che produce circa 40 Mbps per canale, per un totale di 2 Gbps. Vengono posati più terminali per quartiere, in modo che ciascun cavo connetta al più 500 case. Chiaramente, suddividendo la banda si ha che ciascun appartamento ottiene un canale dedicato da 4 Mbps, che può essere usato per una combinazione di programmi MPEG-1, MPEG-2, dati trasmessi dall'utente, telefonia analogica e digitale ecc.

Tutto ciò sembra meraviglioso, ma bisogna che i fornitori di cavi sostituiscano i coax mettendone da 750 MHz, istallino nuovi terminali, e rimuovano tutti gli amplificatori a senso unico: in altre parole, bisogna sostituire tutto il sistema di TV via cavo. Di conseguenza la quantità di nuove infrastrutture qui è paragonabile a quella necessaria alle società telefoniche per FTTC. In entrambi i casi il fornitore di rete locale deve stendere la fibra nei quartieri residenziali. E in entrambi i casi la fibra termina in un convertitore optoelettrico. In FTTC il segmento finale è un circuito locale punto-a-punto su doppino. In HFC il segmento finale è un cavo coassiale condiviso. Tecnicamente questi due sistemi non sono così diversi quanto vorrebbero far credere i rispettivi proponenti.

Nondimeno esiste una differenza sostanziale su cui vale la pena soffermarsi. HFC usa un mezzo condiviso senza commutazione e routing. Qualsiasi dato messo sul cavo può essere ricevuto da qualsiasi abbonato senza altri problemi. FTTC è completamente commutato e non gode di questa proprietà. La conseguenza è che gli operatori HFC usano server di video che distribuiscono stream crittati, che possono essere visti solo da chi paga. Gli operatori FTTC non sono interessati alla crittazione perché aggiunge complessità al sistema, peggiora le prestazioni, e non aggiunge sicurezza al loro sistema. Ora, dal punto di vista della società che gestisce il server video, è meglio crittare o no? Un server gestito da una società telefonica o da una delle sue sussidiarie potrebbe decidere di non crittare i film, sostenendo di aumentare l'efficienza ma in realtà causando perdite economiche ai competitori HFC.

È probabile che ogni rete distributiva locale venga corredata di uno o più server di spool. Questi sono versioni ridotte dei server di video discussi in precedenza. Il grande vantaggio di tali server locali è che siccome le reti distributive sono corte e in genere non commutate non introducono jitter come farebbe una rete su dorsale ATM.

Tali server possono essere caricati con film o dinamicamente o su prenotazione. Ad esempio, quando un utente sceglie un film il primo minuto potrebbe essere trasmesso al server locale in meno di 2 s a OC-3. Dopo 55 s il minuto successivo verrà trasmesso in due secondi, e così via. In questo modo non occorre che il traffico sulla dorsale ATM sia libero dal jitter, quindi è possibile usare il servizio ABR invece del più costoso CBR.

Se gli utenti prenotano i film con consistente anticipo, si possono scaricare sul server durante le ore di traffico minore, aumentando il risparmio. Questa osservazione potrebbe significare che i fornitori di film debbano assumere i funzionari addetti alle tariffe delle linee aeree per far calcolare loro il prezzo del servizio. Possiamo prevedere tariffe in cui i film che vengono ordinati da 24 a 72 h in anticipo per la proiezione il martedì o il giovedì sera prima delle 18 o dopo le 23 otterranno uno sconto del 27%. I film ordinati la prima domenica del mese prima delle 8, da vedere mercoledì pomeriggio in un giorno la cui data è un numero primo sono scontati del 43%, e così via.

La scelta della pila di protocolli da usare per il video a richiesta è ancora vaga. ATM è chiaramente la miglior tecnologia, ma quale protocollo di adattamento dovremmo usare? AAL 1 è stato progettato apposta per il video, quindi è un ottimo candidato, ma corrisponde alla categoria di servizio CBR. Dedicare la massima banda disponibile è costoso, specialmente perché MPEG è traffico intrisicamente VBR quindi il circuito virtuale dovrebbe essere sovradiandimensionato.

AAL 2 non è completo (e probabilmente non lo sarà mai), e AAL 3/4 è troppo complicato, quindi l'unico competitore è AAL 5. Non è legato al servizio CBR, e trasmettere un grosso blocco di MPEG in ciascun messaggio sarebbe estremamente inefficiente, ottenendo quasi il 100% della banda utente per lo stream video. D'altra parte AAL 5 effettua la rilevazione degli errori. Scartare un intero blocco a causa di un errore di 1 bit è una pessima idea, soprattutto perché la maggior parte degli errori sono proprio da un bit nel mezzo dei dati. Di conseguenza, c'è un certo consenso per cambiare AAL 5 in modo da permettere alle applicazioni di chiedere tutti i blocchi, più 1 bit che dica se la somma di controllo è a posto.

La pila di protocolli per il video a richiesta che abbiamo descritto viene illustrata in figura 7-93. Al di sopra del livello AAL vediamo il programma MPEG e il livello di trasporto dello stream. Vengono poi la codifica e decodifica di audio e video MPEG. In cima troviamo l'applicazione.

Scatole set-top

Alla fine della storia, tutti i metodi di distribuzione locale che abbiamo descritto servono a portare nelle case uno o più stream MPEG. Per decodificarli e vederli sono necessari un'interfaccia di rete, un decodificatore MPEG e altri componenti elettronici. Esistono due soluzioni.

La prima soluzione è usare un personal computer per decodificare e vedere i film. È necessaria una speciale scheda che contenga alcuni processori e un connettore per collegare la rete di distribuzione locale. I film appaiono sul monitor, magari in una finestra. Potremmo chiamare questa configurazione scatola set-bottom perché quando si usa un computer di solito la scatola è sotto il monitor. Questo approccio è economico – serve solo la scheda e il software –, usa un monitor ad alta risoluzione non interallacciato, offre

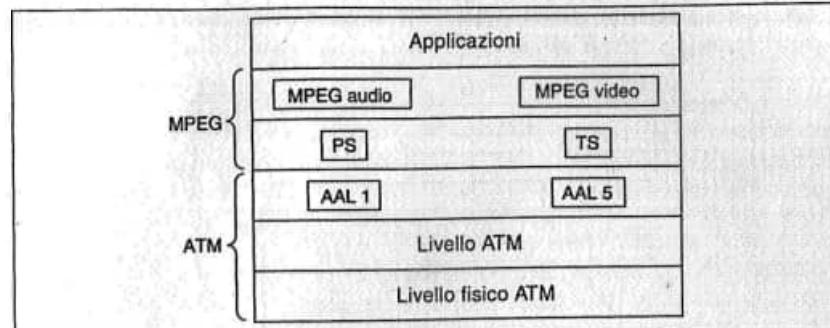


Fig. 7-93 Una pila di protocolli per il video a richiesta.

un'interfaccia utente sofisticata basata su mouse, e si può integrare facilmente con il WWW e altre fonti di informazioni e servizi. D'altra parte, di solito i PC hanno schermi piccoli, sono posti in uffici e studi più che nei salotti, e vengono tradizionalmente usati da una persona alla volta. Sono anche significativamente meno luminosi dei televisori. La seconda soluzione consiste nel noleggiare o comprare una **scatola set-top** cui siano connesse sia la rete che il televisore. Questo approccio ha il vantaggio che tutti hanno un televisore ma non un computer, e per di più i computer posseduti dalle famiglie di solito sono vecchi o non adatti alla codifica MPEG. Inoltre il televisore di solito è posto in una stanza progettata per la visione di gruppo.

D'altra parte, il monitor ha bassa risoluzione interallacciata, inadatta a materiale testuale come il WWW. Inoltre, l'interfaccia utente è orrenda (il telecomando), il che costringe a un'interazione completamente guidata a menu. Anche inserire il nome di un film diventa complicato, e scordiamoci di poter dialogare col server alla ricerca di tutti i film fatti da un certo attore, regista o casa di produzione in un certo periodo. Infine, le scatole set-top con date prestazioni non sono facili da produrre a un prezzo accettabile (che dovrebbe essere di qualche centinaio di dollari).

Considerati tutti questi fattori, la maggior parte dei sistemi di video a richiesta ha adottato il modello a scatola set-top, perché i funzionari commerciali detestano escludere clienti potenziali (quelli senza PC). Inoltre, si crea un mercato per vendere o noleggiare le scatole. D'altra parte, il mercato delle schede PC è sufficientemente grande quindi senza dubbio anche tali schede verranno prodotte.

Le funzioni primarie della scatola sono di interfacciare la rete di distribuzione locale, decodificare il segnale MPEG, sincronizzare audio e video, produrre un segnale composto NTSC, PAL o SECAM per il televisore, reagire al telecomando, e gestire l'interfaccia utente. Altre funzioni potrebbero essere la connessione con uno stereo, con il telefono, o altri dispositivi. È in corso nell'industria una grossa battaglia per decidere quali funzioni spettano alla scatola e quali alla rete. Chi vivrà vedrà.

La figura 7-94 mostra una possibile architettura di una semplice scatola set-top. Il dispositivo include una CPU, ROM, RAM, controllore I/O, decodificatore MPEG e interfaccia

di rete. È opzionale un processore di sicurezza per decrittare i film e crittare i messaggi in uscita (numeri di carte di credito ecc.).

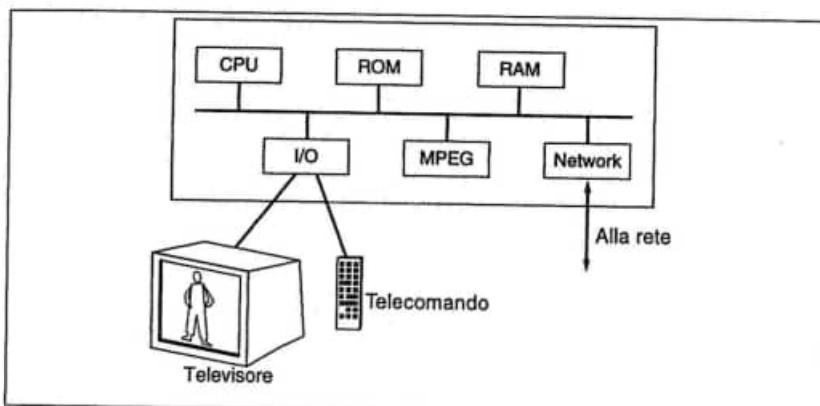


Fig. 7-94 L'architettura hardware di una semplice scatola set-top.

Un problema importante è la sincronizzazione audio/video e la gestione del jitter. 500 KB di RAM permettono di "bufferizzare" 1 s di MPEG-2, ma aumentano il costo di un dispositivo che i fabbricanti sperano di vendere al massimo per poche centinaia di dollari. Siccome la scatola è un computer, ha bisogno di software, forse un sistema operativo in tempo reale con architettura a microkernel memorizzato su ROM. Per ragioni di flessibilità e adattabilità è probabilmente una buona idea permettere di scaricare dalla rete altro software. Questa possibilità crea il problema di cosa accadrebbe quando il proprietario di una scatola basata su MIPS volesse giocare con un videogioco scritto per una scatola basata su SPARC. L'uso di un linguaggio interpretato come Java risolverebbe il problema di compatibilità ma abbasserebbe le prestazioni in un ambiente di tempo reale in cui le prestazioni sono cruciali.

Gli standard

Non possiamo ignorare l'economia del video a richiesta. Un grosso server di video può facilmente costare più di un mainframe, almeno 10.000.000 di dollari. Si supponga che serva 100.000 case, nelle quali troviamo scatole set-top da 300 dollari. Ipotizziamo altri 10.000.000 di dollari di investimento sulla rete e un periodo di ammortamento di 4 anni: il sistema deve generare per ogni casa 10 dollari al mese. A 5 dollari per film, tutti debbono comprare due in modo di andare in parità; non abbiamo ancora incluso gli stipendi, la commercializzazione, e tutti gli altri costi. Dunque non è ovvio che il gioco valga la candela.

I numeri che abbiamo dato possono essere ricalcolati in molti modi (ad es., si può far pagare 6 dollari al mese per il noleggio della scatola e 2 dollari per film), e i costi cambiano velocemente nel tempo, ma è chiaro che se non si crea un mercato di massa

non c'è modo che il video a richiesta diventi un affare. È essenziale che tutti i componenti diventino standardizzati. Se ciascun fornitore di servizi, operatore di rete fabbricante di scatole progetterà la propria interfaccia particolare, nessun componente riuscirà a operare con gli altri. Finora l'unico standard accettato da tutti è MPEG-2 per la codifica video. Tutto il resto è in discussione. Alcune questioni cui va data risposta prima di costruire un sistema nazionale sono elencate in figura 7-95.

- Quale tecnologia usa la dorsale (SONET, ATM, SONET+ATM)?
- A quale velocità va la dorsale (OC-3, OC-12)?
- Com'è fatta la distribuzione locale (HFC, FTTC)?
- Quanta banda d'ingresso è concessa all'utente (16 kbps, 1,5 Mbps)?
- I film sono crittati? Se sì, come?
- Si ha correzione di errore (obbligatoria, opzionale, assente)?
- Chi possiede la scatola set-top (l'utente, l'operatore)?
- La telefonia è parte del sistema (analogico, N-ISDN)?
- Sono supportate le applicazioni ipertestuali ad alta risoluzione (ad es. WWW)?

Fig. 7-95 Alcune aree in cui sono necessari gli standard.

Se tutto questo può essere standardizzato, possiamo facilmente immaginare molti vendori i cui prodotti consistano in scatole contenenti un connettore telefonico, un monitor, una tastiera e un mouse da usare per il video e per calcolare, magari allo stesso tempo. A quel punto sarà una realtà la molto discussa convergenza delle industrie informatica, telematica e televisiva.

7.7.5 MBone

Mentre tutte queste industrie stanno facendo grandi piani, molto pubblicizzati, per il futuro video a richiesta digitale internazionale, la comunità Internet in silenzio ha realizzato il proprio sistema digitale multimediale: **MBone** (**M**ulticast **B**ackbone). In questo paragrafo spiegheremo in breve che cos'è e come funziona. Il testo di Kumar (1996) è un intero libro su MBone; articoli sullo stesso argomento sono in Eriksson (1994); Macedonia, Brutzmann (1994).

MBone costituisce la TV e la radio di Internet. Diversamente dal video a richiesta, in cui l'enfasi è sulla richiesta e la visione di film precompresi memorizzati su un server, MBone si usa per trasmettere in broadcast su Internet a tutto il mondo audio e video in diretta in forma digitale. È stato reso operativo nel 1992. Sono state trasmesse molte conferenze scientifiche, specie IETF, così come alcuni eventi notevoli come le missioni della navetta spaziale. Una volta è stato trasmesso un concerto dei Rolling Stones. È difficile classificare questo come evento scientifico. È inoltre disponibile software per registrare in formato digitale una trasmissione MBone (Hofstader 1995).

La maggior parte delle ricerche su MBone si è concentrata sul multicast efficiente su Internet, che è orientata ai datagrammi. Pochi sforzi sono stati dedicati alla codifica audio

o video. Le fonti MBone sono libere di sperimentare MPEG o altra tecnica di codifica.

Non esistono standard Internet sul contenuto o la codifica.

Tecnicamente MBone è una rete virtuale realizzata su Internet. Comprende alcune isole capaci di fare multicast, connesse da gallerie, come in figura 7-96.

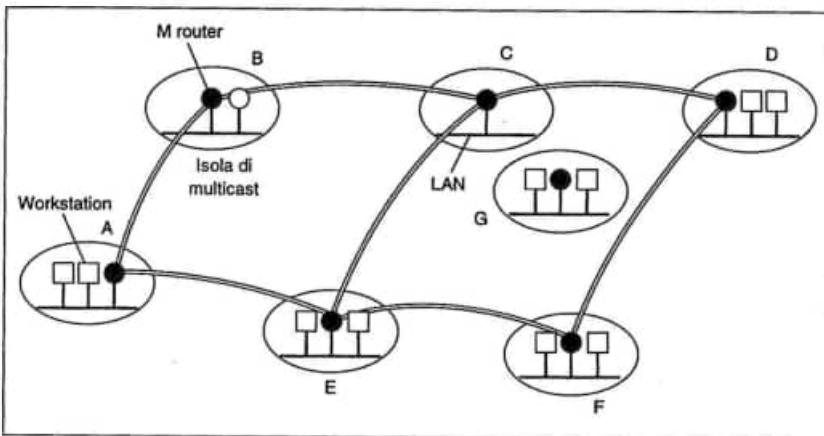


Fig. 7-96. MBone è composto da isole di multicast connesse da gallerie.

In questa figura abbiamo sei isole, da A a F, connesse da sette gallerie. Ogni isola (una LAN o un gruppo di LAN interconnesse) supporta multicast hardware ai suoi host. Le gallerie propagano pacchetti MBone tra le isole. Un giorno nel futuro quando tutti i router saranno capaci di gestire direttamente il traffico multicast questa sovrastruttura non sarà più necessaria, ma per il momento funziona bene.

Ciascuna isola contiene uno o più router speciali chiamati **mrouter** (**multicast router**). Alcuni sono router normali, ma la maggior parte è composta da elaboratori UNIX che eseguono del software speciale con privilegi di root. Gli mrouter sono logicamente connessi da gallerie. Nel passato i pacchetti MBone erano instradati da mrouter a mrouter di solito via uno o più router ignari di MBone, usando routing sorgente lasco. Oggi i pacchetti MBone vengono incapsulati in pacchetti IP (tunneling) e trasmessi come pacchetti normali unicast all'indirizzo IP del router destinazione. Se tutti i router coinvolti supportano il multicast il tunneling è inutile.

Le gallerie vengono configurate a mano. Di solito una galleria è un cammino virtuale su connessioni fisiche, ma non sempre. Se per caso il cammino fisico che supporta la galleria si guasta, gli mrouter che usano la galleria non se ne accorgono, perché Internet reinstraderebbe automaticamente tutto il traffico IP su altre linee.

Se una nuova isola vuole connettersi a MBone, come G in fig. 7-96, l'amministratore manda un messaggio annunciando la sua esistenza alla lista di posta elettronica di MBone. Gli amministratori dei siti vicini allora lo contattano per configurare le gallerie. A volte le gallerie esistenti vengono rimescolate per ottimizzare la topologia. Dopo tutto le gal-

lerie non sono entità fisiche, ma sono definite da tabelle negli mrouter e possono essere aggiunte, rimosse o spostate modificando tali tabelle. Di solito in ciascuna nazione esiste una dorsale MBone, cui si attaccano isole regionali. Normalmente MBone viene configurata con una o due gallerie che attraversano gli oceani Atlantico e Pacifico, raggiungendo una scala globale.

In ogni momento MBone ha una topologia specifica fatta di isole e gallerie, indipendente dal numero di indirizzi di multicast in uso e dal numero di utenti audio o video. Questa situazione è molto simile a una sottorete fisica normale, quindi si applicano i normali algoritmi di routing. Di conseguenza MBone inizialmente usò un algoritmo, **DVMRP** (**Distance Vector Multicast Routing Protocol**), derivato dall'algoritmo di Bellman-Ford della distanza vettoriale. Ad esempio, nella figura 7-96 l'isola C può instradare su A o via B o via E (o persino via D). La scelta viene fatta prendendo i valori comunicati da tali nodi che riportano le loro distanze da A e poi comando la sua distanza verso di essi. In questo modo ogni isola determina la strada migliore verso ogni altra isola. Tuttavia, come vedremo, le strade non vengono usate davvero in questo modo.

Consideriamo come avviene il multicast. Per trasmettere un programma audio o video una sorgente deve prima acquisire un indirizzo di multicast di classe D, che funziona come una frequenza o un numero di canale. Gli indirizzi di classe D vengono riservati usando un programma che cerca gli indirizzi liberi in un database. Possono essere simultaneamente in corso molte trasmissioni e un host si sintonizza su quella di interesse ascoltando il relativo indirizzo multicast.

Periodicamente ogni mrouter emette un pacchetto di broadcast IGMP limitato alla sua isola chiedendo chi è interessato a quale canale. Gli host che vogliono continuare a ricevere uno o più canali rispondono con un altro pacchetto IGMP. Tali risposte vengono scagliate nel tempo, per evitare di sovraccaricare la LAN. Ogni mrouter possiede una tabella dei canali che deve immettere sulla propria LAN, per evitare di sprecare banda per canali che nessuno vuole.

Le trasmissioni si propagano come segue. Quando una fonte audio o video genera un nuovo pacchetto, lo trasmette alla sua isola usando la funzione hardware di multicast. Questo pacchetto viene preso dall'mrouter locale, che lo copia su tutte le gallerie cui è connesso.

Quando un mrouter riceve un pacchetto da una galleria controlla se arriva dalla strada migliore, ciò quella che la sua tabella gli dice essere la migliore per raggiungere la fonte (come se fosse una destinazione). Se è così l'mrouter copia il pacchetto sulle altre gallerie; altrimenti viene scartato. Ad esempio, in figura 7-96 se la tabella di C dice di usare B per raggiungere A allora quando arriva un pacchetto di A via B il pacchetto viene copiato su D ed E. Se invece arriva un pacchetto da A via E (che non è il miglior cammino), viene scartato. Questo algoritmo è l'inverso di quello di avanzamento sul cammino visto nel capitolo 5. Non è perfetto, ma è buono e facile da implementare.

Oltre a usare il cammino inverso per evitare di inondare Internet, il campo IP *Time to live* viene usato per limitare l'azione del multicasting. Ogni pacchetto comincia con qualche valore determinato dalla fonte e ogni galleria ha un peso. Un pacchetto attraversa una galleria solo se ha abbastanza peso, altrimenti si scarta. Ad esempio, le gallerie transoceaniche di solito hanno un peso di 128, quindi per vincolare i pacchetti al continente di

origine basta assegnare un valore *Time to live* minore o uguale di 127. Dopo aver attraversato una galleria il campo *Time to live* viene decrementato del peso della galleria stessa. Anche se questo algoritmo funziona, sono state fatte molte ricerche per migliorarlo. Una proposta conserva il routing con distanza vettoriale, ma rende l'algoritmo gerarchico raggruppando i siti MBone in regioni e facendo routing su queste (Thyagarajan, Deering, 1995). Un'altra proposta suggerisce di usare una forma modificata del routing link state. In particolare, un gruppo di lavoro IETF sta modificando OSPF per adattarlo al multicast entro un sistema autonomo. Il risultato si chiama MOSPF (Moy, 1994). Le modifiche costruiscono la mappa completa per tener traccia delle isole e gallerie di multicast, oltre all'informazione normale per il routing. Avendo la topologia completa, è immediato calcolare il cammino migliore da un'isola a un'altra; si può usare ad esempio l'algoritmo di Dijkstra.

Una seconda area di ricerche è il routing tra le AS. Esiste un algoritmo, **PIM (Protocol Independent Multicast)**, che è in via di sviluppo presso un altro gruppo di lavoro IETF (Huitema, 1995). Ce ne sono due versioni, a seconda che le isole siano dense (tutti vogliono guardare) o sparse (quasi nessuno vuol guardare). Entrambe le versioni usano le tabelle standard di routing unicast, invece di creare una topologia sovrapposta come fanno DVMRP e MOSPF.

In PIM-denso l'idea è di tagliare i cammini inutili. I tagli si fanno come segue. Quando arriva un pacchetto multicast dalla galleria "sbagliata", lungo la stessa galleria viene trasmesso all'indietro un pacchetto di tagli che dice al mittente di smettere di mandare pacchetti. Se il pacchetto arriva dalla galleria "giusta", viene copiato su tutte le altre che non sono state tagliate. Se tutte le gallerie sono state tagliate e localmente quel canale non è interessante l'mrouter taglia anche la galleria "giusta". In questo modo si ha un adattamento automatico e la trasmissione arriva solo dove è richiesta.

PIM-sparso funziona diversamente. L'idea è di impedire che Internet vada in saturazione perché tre persone a Berkeley vogliono fare una conferenza utilizzando un indirizzo di classe D. PIM-sparso funziona costruendo dei punti di rendez-vous. Tutte le sorgenti del gruppo PIM-sparso mandano i pacchetti ai punti di rendez-vous. Ogni sito interessato chiede ai punti di aprire una galleria. In questo modo il traffico PIM-sparso viene trasportato per unicast invece di multicast.

Tutto considerato il multimediale è un'area eccitante e in rapida evoluzione. Quasi quotidianamente vengono annunciate nuove tecnologie e applicazioni, e complessivamente questo argomento probabilmente rimarrà importante nei prossimi decenni.

7.8 Riassunto

Le reti di computer sono inherentemente insicure. Per mantenere segreta l'informazione occorre crittarla. Gli algoritmi crittografici si classificano in due gruppi: a chiave segreta (DES, IDEA) e a chiave pubblica (RSA). L'uso di tali algoritmi è semplice; la parte difficile è la gestione della chiave.

Oltre a garantire segretezza, gli algoritmi crittografici possono anche fornire autenticazione, così quando Alice crede di comunicare con Bob, sta davvero comunicando con lui e non con Trudy. Infine, si può usare la crittografia anche per firmare i messaggi di modo che dopo averli mandati il mittente non possa disconoscerli.

7.8 Riassunto

In Internet si usa il DNS, un sistema di database distribuiti per gestire i nomi. Il DNS contiene record con indirizzi IP, scambi di posta e altre informazioni. Un processo che interroga un server DNS può trasformare un nome di dominio Internet nell'indirizzo IP necessario per comunicare con esso.

Via via che le reti crescono diventano più difficili da gestire. È stata sviluppata proprio per questo motivo una serie di sistemi e protocolli speciali per la gestione della rete, di cui il più popolare è SNMP. Questo protocollo permette ai gestori di comunicare con agenti dentro i dispositivi per leggere il loro stato e trasmettere loro dei comandi.

Le quattro principali applicazioni della rete sono la posta elettronica, le notizie di USENET, il World Wide Web, e il multimediale (video a richiesta ed MBone). La maggior parte dei sistemi di posta elettronica usa il sistema definito in RFC 821 e 822. I messaggi mandati in questo sistema usa intestazioni ASCII per definire i propri attributi. I messaggi stessi vengono trasmessi usando SMTP. Esistono due sistemi per assicurare la segretezza dei messaggi, PGP e PEM.

Le notizie di USENET comprendono migliaia di gruppi su ogni genere di argomento. Gli utenti si associano ai gruppi e possono mandare messaggi a tutto il mondo usando il protocollo NNTP, che somiglia in qualche modo a SMTP.

Il WWW è un sistema per collegare documenti ipertestuali. Ogni documento è una pagina scritta in HTML, che include iperpuntatori ad altri documenti. Un browser è un visualizzatore di documenti che si collega via TCP con un server, cui chiede il documento e poi chiude la connessione. Quando l'utente seleziona un iperpuntatore, il nuovo documento viene ottenuto nello stesso modo. In pratica i documenti sono collegati formando una gigantesca ragnatela mondiale.

Il multimediale è l'astro nascente nel firmamento della rete. Comporta la digitalizzazione di audio e video che vengono trasmessi elettronicamente e riprodotti. La maggior parte dei progetti multimediali usa gli standard MPEG e trasmettere dati su connessioni ATM. MBone è un servizio radiotelevisivo digitale sperimentale sull'Internet mondiale.

Esercizi

- Violare il seguente cifrario monoalfabetico. Il testo in chiaro, che è in inglese e consiste solamente di lettere, è un ben noto brano scelto da un poema di Lewis Carroll.

```
kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm
ur mfudm zhx mftnm zhx mdzythc pzq ur ezsszcdm zhx gthcm
zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthcm
sok ptzk t stk kfd uamkdim eitdx sdruid pd fzld uoi efzk
rui mudb ur om zid uok ur sidzkf zhx zyy ur om zid rzk
hu foiai mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk
```

- Violare il seguente cifrario per trasposizione a colonne. Il testo in chiaro è in inglese e tratto da un noto testo di computer, quindi "computer" è una parola probabile. Il testo in chiaro consiste interamente di lettere (senza spazi). Il testo cifrato è stato suddiviso in blocchi di cinque caratteri per leggibilità.

- aauan cvlre runn dltme aeepb ytust iceat npmey iicgo gorch srsoc
nntii imiha oofpa gsivt tpsit lboir otoex
3. In figura 7-4, i blocchi-P e i blocchi-S si alternano. Anche se quest'alternanza è esteticamente piacevole, è in qualche modo più sicura rispetto ad avere prima tutti i blocchi-P e dopo tutti i blocchi-S?
 4. Supponiamo che un messaggio sia stato cifrato usando il DES in modalità di concatenazione a blocchi del cifrario. Un bit del testo cifrato nel blocco C_i viene accidentalmente trasformato da 0 a 1 durante la trasmissione. Quanto testo in chiaro viene perduto per tale ragione?
 5. Si consideri nuovamente la concatenazione a blocchi del cifrario. Invece della trasformazione di un singolo bit 0 in un bit 1, nel cifrario viene inserito un bit 0 in più dopo il blocco C_i . Quanto testo in chiaro viene perduto per tale ragione?
 6. Progettare un attacco al DES sapendo che il testo in chiaro consiste esclusivamente di lettere ASCII maiuscole, più spazio, virgola, punto, punto e virgola, ritorno carrello e "a capo". Non si sa niente riguardo ai bit di parità del testo in chiaro.
 7. Confrontare la concatenazione a blocchi del cifrario con la modalità feedback del cifrario in termini di numero di operazioni di cifratura necessarie per trasmettere un grosso file. Qual è più efficiente e quanto?
 8. Usando il crittosistema a chiave pubblica RSA, con $a = 1$, $b = 2$ ecc.
 - (a) Se $p = 7$ e $q = 11$, elencare 4 valori validi per d .
 - (b) Se $p = 13$, $q = 31$ e $d = 7$, trovare e .
 - (c) Usando $p = 5$, $q = 11$ e $d = 27$, trovare e cifrare "abcdefghijklm".
 9. Lo scambio di chiave Diffie-Hellman viene usato per stabilire una chiave segreta tra Alice e Bob. Alice trasmette a Bob (719, 3, 191). Bob risponde con (543). Il numero segreto, x , di Alice è 16. Qual è la chiave segreta?
 10. Modificare un messaggio nel protocollo di figura 7-14 in una maniera semplice in modo che resista all'attacco riflessivo. Spiegare perché la modifica funziona.
 11. Nel protocollo della rana dalla bocca larga, perché A invia il testo in chiaro assieme alla chiave di sessione cifrata?
 12. Nel protocollo della rana dalla bocca larga, abbiamo fatto notare che iniziare ogni messaggio in chiaro con 32 bit 0 è un rischio per la sicurezza. Supporre che ciascun messaggio inizi con un numero casuale per ciascun utente, in pratica una seconda chiave segreta conosciuta solo dal suo utente e dal KDC. Questo elimina il noto attacco del testo in chiaro?
 13. Nel protocollo Needham-Schroeder, Alice genera due sfide, R_A ed R_{A2} . Questo sembra troppo. Non ne basta una?
 14. Nel protocollo di autenticazione a chiave pubblica della figura 7-21, nel messaggio

- 3, R_B viene crittata con K_S . Questa citazione è necessaria o sarebbe bastato spedirla indietro in chiaro?
15. Il protocollo di firma della figura 7-22 presenta il seguente punto debole. Se Bob si guasta, perde il contenuto della sua RAM. Che problemi ne conseguono e come si risolvono?
 16. Dopo che Ellen confessa a Marilyn il suo trucco sull'assunzione di Tom, Marilyn per evitare in futuro questo problema, decide di dettare a voce i contenuti degli eventuali messaggi in un dittafono e chiedendo alla segretaria di ribatterli. Marilyn si ripromette di esaminare i messaggi al terminale dopo la loro battitura per assicurarsi che siano fedeli alle sue parole. La nuova segretaria può usare comunque l'attacco del compleanno per falsificare un messaggio, e se sì, come? Suggerimento: sì, si può.
 17. I terminali POS (Point-Of-Sale) che usano carta magnetica e codici PIN hanno un problema fatale: un commerciante con cattive intenzioni potrebbe modificare il suo lettore di carte per catturare e memorizzare tutta l'informazione sulla carta, più il codice PIN per poter effettuare false transazioni in futuro. La prossima generazione di terminali POS userà carte complete di CPU, tastiera, e un piccolo video sulla carta. Si sviluppi un protocollo per tali dispositivi che non possa essere infranto.
 18. Secondo le informazioni date in figura 7-27, *little-sister.cs.vu.nl* è su una rete di classe A, B o C?
 19. Perché nella figura 7-27 non c'è un punto dopo *rowboat*?
 20. Che cos'è l'**OBJECT IDENTIFIER** per l'oggetto *tcp*?
 21. Si deve trasmettere un intero SNMP il cui valore è 200. Si mostri la rappresentazione binaria dei bit trasmessi usando la sintassi di trasferimento ASN.1.
 22. Qual è la rappresentazione della stringa binaria '11100001111' nella sintassi di trasferimento ASN.1?
 23. Un venditore di bridge vi assume per scrivere codice conforme a SNMP per uno dei suoi bridge. Vi leggete gli RFC e comunque vi restano dei dubbi. Decidete di suggerire alla IAB di definire una grammatica formale e completa del linguaggio usato per definire le variabili SNMP. IAB approva e vi affida l'incarico. La grammatica dovrebbe essere aggiunta a RFC 1442 o a RFC 1213? Perché? Suggerimento: non è necessario studiare i documenti RFC; il testo contiene informazioni sufficienti per rispondere.
 24. Alcuni sistemi di posta elettronica supportano un attributo *Content Return*. Specificare se il corpo del messaggio dev'essere restituito nel caso di mancata consegna. Questo campo appartiene alla busta o all'intestazione?
 25. I sistemi di posta elettronica hanno bisogno di guide in cui cercare gli indirizzi della gente. Per costruire tali guide bisognerebbe decomporre i nomi in componenti stan-

- dard (ad es. nome e cognome). Si indichino alcuni problemi da risolvere prima di definire uno standard mondiale.
26. Un file binario è lungo 3072 byte. Quanto si allungherà usando una codifica base64, inserendo una coppia CR+LF ogni 80 byte e alla fine?
 27. Si consideri lo schema di codifica MIME quoted-printable. Scoprire un problema non descritto nel testo e proporre una soluzione.
 28. Dare due ragioni per cui PGP comprime i messaggi.
 29. Si supponga che qualcuno configuri un demone di vacanza e trasmetta un messaggio prima di disconnettersi. Sfortunatamente il destinatario è in vacanza da una settimana e ha attivato anche lui un demone di vacanza. Che succede? Si crea un ciclo di risposte finché qualcuno ritorna?
 30. Se tutti gli utenti Internet usassero PGP, potrebbe essere mandato a un indirizzo arbitrario di Internet un messaggio PGP ed essere questo decodificato da tutti gli interessati? Discutere la risposta.
 31. PGP non supporta la canonicalizzazione, come invece fa PEM. Perché?
 32. Provare a indovinare cosa significa il simbolo smiley : – X (a volte viene scritto come : – #).
 33. Quanto tempo occorre per distribuire una giornata di news mediante un canale satellitare da 50 Mbps?
 34. Quali dei comandi elencati in figura 7-56 sono teoricamente ridondanti?
 35. Una grossa rete è una griglia di $n \times n$ macchine. Tutti i nodi interni hanno 4 vicini; quelli sui lati (o gli angoli) hanno 3 (2) vicini. Se un articolo di m byte viene impostato su una macchina usando NNTP, quanti byte di banda occorrono per trasmetterlo a tutte le altre macchine? (Ignorare l'overhead di NNTP e contare i byte dei messaggi).
 36. Si ripeta il problema precedente calcolando la banda approssimata necessaria per distribuire il messaggio usando una mailing list. Di quanto il risultato è più grande rispetto al problema precedente?
 37. Quando vengono trasmesse le pagine Web hanno un prefisso che è un'intestazione MIME. Perché?
 38. Quando servono i visualizzatori esterni? Come fa un browser a sapere quale vada usato?
 39. Qualcuno al dipartimento di Informatica di Stanford ha scritto un nuovo programma che vuole distribuire mediante FTP. Mette il programma nella directory FTP `ftp/pub/freebies/newprog.c`. Quale sarà la URL di questo programma?
 40. In figura 7-60 il parametro `ALT` viene "settato" nel tag ``. Quando viene usato dai browser, e come?

41. Come si costruisce in HTML un'immagine "cliccabile"? Dare un esempio.
42. Si mostri il tag `<A>` necessario per trasformare la stringa "ACM" nel puntatore a <http://www.acm.org>.
43. Si progetti il form di una nuova società, Interburger, che serve ordinazioni di hamburger via Internet. Il form dovrebbe comprendere il nome del cliente, l'indirizzo, la città, e la scelta delle dimensioni (gigante o immenso) e un'opzione formaggio. Gli hamburger verranno pagati in contanti o alla consegna, quindi non è necessaria l'informazione della carta di credito.
44. Java non ha le strutture come in C, né i record come in Pascal. C'è qualche altro sistema per ottenere lo stesso effetto di aggregare un gruppo di variabili diverse per formare un solo tipo di dati? Se sì, qual'è?
45. Usando le strutture dati della figura 7-75, elencare i passi precisi necessari per controllare una nuova URL per vedere se è già contenuta in `url_table`.
46. Si supponga che nel suo sforzo di riorientamento al mercato, il KGB offra servizi commerciali e ingaggi un'agenzia pubblicitaria che progetta una pagina Web per tali servizi. Siete stati assunti come consulenti esterni per realizzare tale pagina. Scrivete il codice HTML per produrre la seguente pagina Web.

Benvenuti nella pagina WWW del KGB

In seguito alla recente privatizzazione, il KGB è lieto di annunciare la disponibilità commerciale di molti ottimi prodotti e servizi precedentemente disponibili solo ai governi.

Prezzi competitivi! Servizio discreto!

- Prodotti
 - Armi nucleari* (piccole, medie, grandi, jumbo)
 - Satelliti spia* (controllate i vostri vicini)
 - Aerei supersonici con basso profilo radar* (sorvolate gli amici senza essere visti)
- Servizi
 - Immissione di talpe in un'organizzazione di vostra scelta*
 - Colpi di stato* (politici o industriali)
 - Assistenza per costruire il vostro laboratorio per la guerra biologica*
- Offerte speciali a prezzi scontati
 - L'opera omnia di Felix Dzerzhinsky* (tiratura limitata)
 - Fotografie aeree dell'Afghanistan* (risalenti al 1984)
 - Carri armati di qualità bulgara* (sconto del 95%)

Webmaster@kgb.ru

47. In C e C++ il linguaggio non definisce la dimensione di un intero. In Java invece sì. Dare un argomento per il modo C e uno per quello Java.
48. Si supponga che il Web contenga 10.000.000 di pagine, ciascuna con una media di 10 iperpuntatori. Ottenere una pagina richiede 100 ms. Qual è il tempo minimo per indicizzare l'intero Web?
49. Un compact disc contiene 650 MB di dati. I CD audio sono compressi? Spiegare il ragionamento seguito.
50. Qual è il tasso di bit necessario per trasmettere VGA a colori senza compressione con 8 bit/pixel a 40 immagini/s?
51. Nella figura 7-76 (c) si ha rumore di quantizzazione a causa dell'uso di campioni su 3 bit. Il primo campione, a 0, è esatto, ma i successivi non lo sono. Qual è la percentuale di errore dei campioni a 1/32, 2/32 e 3/32 del periodo?
52. È possibile che un errore di un bit in un'immagine MPEG influenzi altri frame a parte quello in cui è contenuto? Spiegare la risposta.
53. Si consideri l'esempio del video server da 100.000 clienti descritto nel testo. Si supponga che metà di tutti i film venga servita di sera dalle 8 alle 10. Quanti film deve trasmettere il server simultaneamente durante questo periodo? Se ciascun film richiede 4 Mbps, quante connessioni OC-1 servono tra il server e la rete?
54. Si supponga che legge di Zipf valga per gli accessi a un video server da 10.000 film. Se il server conserva i più popolari 1000 film su disco magnetico e i restanti su disco ottico, definire un'espressione per la frazione di tutti i riferimenti al disco magnetico. Scrivere un piccolo programma che valuti tale espressione numericamente.
55. I pacchetti PES di MPEG contengono un campo che restituisce lo stato dei diritti d'autore della trasmissione corrente. A che può servire tale campo?

Capitolo 8

RIFERIMENTI BIBLIOGRAFICI

Abbiamo terminato il nostro studio delle reti di computer, ma questo è solo l'inizio. Abbiamo trascurato di approfondire molti argomenti che avrebbero meritato una miglior esposizione, mentre per mancanza di spazio abbiamo completamente omesso altri argomenti. In questo capitolo intendiamo offrire alcuni suggerimenti per letture ulteriori e una bibliografia a beneficio dei lettori che vogliono continuare lo studio delle reti di calcolatori.

8.1 Suggerimenti per letture ulteriori

La letteratura sulle reti e sui sistemi distribuiti è molto estesa. Le riviste scientifiche che pubblicano articoli su questi argomenti sono "IEEE Transactions on Communications", "IEEE Journal on Selected Areas in Communications", "Computer Communication Review", e "Computer Networks and ISDN Systems". Molte altre riviste pubblicano occasionalmente articoli su tali argomenti.

L'IEEE pubblica anche due riviste a grande diffusione, "IEEE Network Magazine" e "IEEE Communications Magazine", che contengono articoli di rassegna, articoli introduttivi, e presentazioni di esempi di reti. La prima rivista mette in risalto l'architettura, gli standard, e il software, mentre la seconda copre la tecnologia delle comunicazioni (fibre ottiche, satelliti ecc.).

Esistono poi parecchie conferenze annuali o periodiche che spesso attraggono molti articoli sulle reti e i sistemi distribuiti. In particolare, si consultino gli atti di SIGCOMM '9x, The International Conference on Distributed Computer Systems, The Symposium on Operating Systems Principles e The N-th Data Communications Symposium.

Inoltre, l'IEEE ha pubblicato in un formato conveniente molti volumi contenenti ristampe di articoli importanti sulle reti.

Nel seguito elenchiamo i nostri suggerimenti per letture ulteriori, classificati in base ai capitoli di questo libro.

8.1.1 Introduzione e lavori di ordine generale

Jabbari et al., *Network Issues for Wireless Communication* (art.)

Un'introduzione ai sistemi radio cellulari, che tratta il controllo della chiamata, il routing, e altri aspetti dei sistemi mobili di comunicazione.

Comer, *The Internet Book*

Questo libro è adatto a chi cerca un'introduzione semplice a Internet. Comer descrive la

storia, lo sviluppo, le tecnologie, i protocolli, e i servizi di Internet in termini comprensibili per i novizi. Il libro contiene così tanto materiale da interessare anche i lettori più tecnici.

Kwok, *A Vision for Residential Broadband Service* (art.)

Se volete sapere come la pensa la Microsoft sul video a richiesta e come dovrebbe essere organizzato, questo è l'articolo che fa per voi. Il principale architetto ATM della Microsoft spiega la visione della sua società. In breve, questa è l'idea della Microsoft: ATM nelle case è la strada giusta; lasciamo perdere tutte le soluzioni "realistiche" (cioè ad hoc), come ADSL, e facciamo la cosa giusta.

Le Boudec, *The Asynchronous Transfer Mode: A Tutorial* (art.)

ATM è una nuova tecnologia, e questo articolo è un'ottima introduzione. Sono coperti tutti i livelli rilevanti: fisico, ATM e AAL. Inoltre, la parte finale descrive il dibattito in corso su ATM.

Pahlavan et al., *Trends in Local Wireless Networks* (art.)

Le LAN senza filo indubbiamente diventeranno sempre più importanti in futuro. In questo articolo, gli autori discutono lo stato dell'arte e le tendenze tecnologiche e nell'uso dello spettro.

Siu, Jain, *A Brief Overview of ATM* (art.)

Questo articolo introduttivo copre molti aspetti dei sistemi ATM; si concentra comunque su emulazione di LAN e gestione del traffico. Costituisce l'introduzione a un numero speciale di *Computer Communication Review* dedicato alla tecnologia ATM.

Stallings, *Data and Computer Communications*, quarta ed.

Invece di descrivere solo i modelli di riferimento OSI e TCP/IP, Stallings ne aggiunge per buona misura un terzo, SNA. Sono tutti descritti nel capitolo 10.

8.1.2 Il livello fisico

Awdeh, Mouftah, *Survey of ATM Switch Architectures* (art.)

Chiunque sia interessato al progetto di switch dovrebbe studiare questo lavoro. Dopo un'introduzione generale ai commutatori e alle strategie di "bufferizzazione", gli autori descrivono molti tipi di commutatori crossbar, disjoint-path, e banyan. L'articolo include oltre 200 riferimenti.

Bellamy, *Digital Telephony*

Tutto quel che avreste voluto sapere sul sistema telefonico, e altro, si trova in questo libro autorevole. Molto interessanti sono i capitoli sulla trasmissione e sul multiplexing, sulla commutazione digitale, sulle fibre ottiche e ISDN.

8.1 Suggerimenti per letture ulteriori

De Prycker, *Asynchronous Transfer Mode*, seconda ed.

Il capitolo 4 contiene ricchissime informazioni sugli switch ATM. L'idea è illustrata da numerosi esempi di commutazione, compresi knockout, Roxanne, Coprin, e Athena.

Held, *The Complete Modem Reference*, seconda ed.

Questo libro contiene tutto lo scibile sui modem, dalle leggi dei governi statunitense e canadese, alle tecniche e gli standard di modulazione, a come diagnosticare un modem guasto.

"IEEE Communications Mag.", *Wireless Personal Communications* gennaio 1995 (art.)

Questo numero speciale contiene sette articoli su diversi aspetti della comunicazione personale senza filo. Complessivamente coprono la propagazione, i metodi di accesso, i principi di ricezione, gli aspetti di sistema e le reti.

Metcalfe, *Computer/Network Interface Design: Lessons from Arpanet & Ethernet* (art.)

Sebbene da decenni gli ingegneri siano capaci di costruire interfacce di rete, ci si chiede spesso cosa abbiano appreso da tutta questa esperienza. In questo articolo, il progettista di Ethernet racconta come si costruisce un'interfaccia di rete, e cosa farsene quando la si costruisce. L'autore non nasconde niente, e confessa quel che ha fatto male e quel che ha fatto bene.

Padgett et al., *Overview of Wireless Personal Communications* (art.)

Un'introduzione e un confronto tra i sistemi di comunicazione cellulare e senza filo. Sono trattati entrambi gli standard americano ed europeo.

Palais, *Fiber Optic Communication*, terza ed.

I libri sulla tecnologia delle fibre ottiche di solito sono per specialisti, ma questo è uno dei più leggibili. Tratta le guide d'onda, le sorgenti luminose, gli accoppiatori, il rumore di modulazione, e molto altro.

Pandya, *Emerging Mobile and Personal Communications Systems* (art.)

Questo articolo è utile se si desidera un'introduzione rapida e semplice ai piccoli sistemi di comunicazione personale. Una delle nove pagine contiene una lista di acronimi usati nelle altre otto.

Partridge, *Gigabit Networking*

Oltre a descrivere parecchi tipi di switch ATM, il capitolo 5 confronta la "bufferizzazione" di ingresso con quella di uscita e deriva formule per le loro prestazioni.

Spragins et al., *Telecommunications Protocols and Design*

Il capitolo 2 contiene una buona introduzione alla tecnologia di trasmissione, inclusi i fili di rame, le fibre ottiche, la radio cellulare e i satelliti. Include anche una lunga discussione sui limiti di Nyquist e Shannon e sulle loro implicazioni.

8.1.3 Il livello data link

Black, *Data Link Protocols*

Un intero libro dedicato al livello data link. Si sofferma sugli aspetti pratici, con notevole materiale su HDLC, LLC, PPP, e altri protocolli commerciali importanti.

Holzmann, *Design and Validation of Computer Protocols*

Chi è interessato agli aspetti più formali dei protocolli data link (e similari) dovrebbe consultare questo libro, che tratta la specifica, il progetto, la correttezza e la verifica di tali protocolli.

Spragins et al., *Telecommunications Protocols and Design*

Il capitolo 6 di questo libro contiene notizie utili sui codici che rilevano e correggono gli errori. Il libro descrive inoltre i principi dei protocolli elementari data link più o meno allo stesso livello del libro che avete davanti. Il capitolo 7 continua dettagliatamente la discussione di diversi protocolli data link.

Walrand, *Communication Networks: A First Course*

Il capitolo 4 tratta i protocolli data link, con enfasi sull'analisi delle prestazioni. Sono trattati gli approcci a macchina a stati finiti e le reti di Petri.

8.1.4. Il sottolivello medium access control

Abeyendumara, Kamal, *High-Speed Local Area Networks and Their Performance* (art.)

Dal momento che le LAN ad alta velocità sono interessanti per la loro velocità, un articolo che analizza e discute le loro prestazioni è il benvenuto. In questo ci si concentra sui diversi tipi di LAN basate su bus, anello, albero e stella, sui ritardi nonché sulle loro caratteristiche d'uso.

Jain, *FDDI Handbook-High-Speed Networking Using Fiber and other Media*

Questo libro è una buona scelta per chi cerca un approfondito trattamento di FDDI (comprese delle belle introduzioni alla tecnologia a fibre ottiche e a SONET). Oltre a lunghi paragrafi su hardware e software FDDI, include un paragrafo sulle prestazioni e persino consigli per gli acquisti di cavi a fibra ottica.

Perlman, *Interconnections: Bridges and Routers*

Questo è il libro da consultare per una discussione autorevole ma divertente sui bridge e i routers. L'autore è un esperto perché è il progettista dell'algoritmo usato nel bridge a spanning tree IEEE 802 nonché degli algoritmi di routing di DECnet.

Stallings, *Local and Metropolitan Area Networks*, quarta ed.

Il nucleo di questo libro è formato dalle tre LAN IEEE 802, ma è incluso anche materiale su altre LAN e MAN.

8.1 Suggerimenti per letture ulteriori

Walrand, *Communication Networks: A First Course*

Come il libro di Stallings di cui sopra, il capitolo 5 di questo espone il materiale base di 802, più FDDI e DQDB. L'enfasi è sull'analisi delle prestazioni dei protocolli.

8.1.5. Il livello rete

Comer, *Internetworking with TCP/IP*, vol. 1, terza ed.

Comer ha scritto il lavoro definitivo sui protocolli TCP/IP. I capitoli da 4 a 11 discutono IP e protocolli correlati nel livello rete. Gli altri capitoli trattano soprattutto dei livelli superiori, e vale la pena di leggerli.

Huitema, *Routing in the Internet*

Chi vuol sapere tutto quel che c'è da sapere sul routing di Internet deve leggere questo libro. Sono esposti in dettaglio sia gli algoritmi pronunciabili (ad es. RIP, CIDR, e MBO-NE) che quelli impronunciabili (per esempio, OSPF, IGRP, EGP, e BGP). Contiene anche una descrizione delle novità, come il multicast, l'IP mobile, e la prenotazione di risorse.

Perlman, *Interconnections: Bridges and Routers*

Nel capitolo 9 Perlman descrive molti aspetti del progetto di algoritmi di routing unicast e multicast, per WAN e reti di LAN, e la loro realizzazione in vari dispositivi. L'autore chiaramente ha molto a cuore l'argomento, perché ha intitolato il paragrafo 9.13 "La mia opinione sul multicast IP nel livello rete".

Sterbenz et al., *Report on the IEEE ComSoc Gigabit Networking Workshop* (art.)

Prima che le connessioni di rete da 1 Gb e oltre divengano utilizzabili, dovranno essere risolti parecchi problemi fondamentali. Uno dei più importanti è se tali reti utilizzeranno ATM, TCP/IP o entrambi. Allo scopo di studiare meglio questi problemi, lo IEEE nell'aprile del 1995 organizzò una conferenza, di cui questa pubblicazione costituisce un riassunto. Chi crede che ATM sia la soluzione dei problemi di telecomunicazione mondiale dovrebbe leggere la critica fatta da Schulzrinne.

Stevens, *TCP/IP Illustrated*, vol. I

I capitoli 3-10 costituiscono un trattamento approfondito di IP e dei protocolli relativi (ARP, RARP, e ICMP) illustrato da esempi.

Yang, Reddy, *A Taxonomy for Congestion Control Algorithms in Packet Switching Networks* (art.)

Gli autori hanno sviluppato una tassonomia per gli algoritmi di controllo della congestione. Le categorie principali sono "ciclo aperto con controllo della sorgente", "ciclo aperto con controllo della destinazione", "ciclo chiuso con feedback esplicito", e "ciclo chiuso con feedback implicito". La tassonomia viene usata per descrivere e classificare 23 algoritmi esistenti.

8.1.6 Il livello trasporto

Comer, *Internetworking with TCP/IP*, vol. 1, terza ed.

Abbiamo già detto prima che Comer ha scritto il lavoro definitivo su TCP/IP. Il capitolo 12 descrive UDP; Il capitolo 13 descrive TCP.

Mogul, *IP Network Performance* (art.)

Malgrado il suo titolo, questo articolo riguarda le prestazioni di TCP e di rete in generale, più che quelle di IP in particolare. È pieno di consigli e suggerimenti utili.

Stallings, *Data and Computer Communications*, quarta ed.

Il capitolo 12 riguarda i protocolli di trasporto e tratta in astratto i servizi e i meccanismi, oltre a descrivere in dettaglio i protocolli di trasporto OSI e TCP.

Stevens, *TCP/IP Illustrated*, vol. I

I capitoli 17-24 descrivono approfonditamente TCP con esempi.

8.1.7 Il livello delle applicazioni

Anderson, R., *Why Cryptosystems Fail* (art.)

Secondo Anderson, la sicurezza nei sistemi bancari è pessima, ma non a causa di intrusori furbi che rompono DES sui loro PC. I veri problemi vanno dagli impiegati disonesti (un commesso che riscrive l'indirizzo di un cliente per intercettare la sua carta Bancomat e il suo numero segreto) agli errori di programmazione (dare a tutti i clienti lo stesso codice PIN). Molto illuminanti sono le risposte delle banche quando qualcuno contesta loro un errore: "I nostri sistemi sono perfetti e perciò tutti gli inconvenienti sono dovuti a errori o frodi dei clienti".

Berghel, *The Client Side of the Web* (art.)

Un'introduzione semplice ai browser Web e alle funzioni che offrono. Gli argomenti principali sono la conformità HTML/HTTP, le prestazioni, la riconfigurabilità, l'integrazione con l'interfaccia principale, e i supporti alla navigazione. Vengono paragonati nove browser molto popolari.

Berners-Lee et al., *The World Wide Web* (art.)

Un articolo che presenta una prospettiva sul Web e sul suo futuro, scritto dalla persona che l'ha inventato. L'articolo presenta l'architettura, HTTP, HTML e le evoluzioni future.

Carl-Mitchell, Quarterman, *Practical Internetworking with TCP/IP and UNIX*

Il capitolo 5 è un'introduzione ai problemi del naming e al DNS, inclusi le autorità, l'architettura operativa e il database DNS.

Choudhury et al., *Copyright Protection for Electronic Publishing on Computer Networks* (art.)

Sebbene parecchi libri e articoli descrivano gli algoritmi crittografici, pochi si soffermano su come si possono usare per impedire l'ulteriore distribuzione dei documenti una volta che siano stati decrittati. Questo articolo presenta una varietà di meccanismi che possono aiutare a proteggere i diritti d'autore nell'era elettronica.

8.1 Suggerimenti per letture ulteriori

Furht et al., *Design Issues for Interactive Television Systems* (art.)

Il video a richiesta propone molti problemi tecnici complessi relativi all'architettura di sistema, alla topologia di rete, al progetto dei server e dei dispositivi di visualizzazione. In questo articolo gli autori presentano un'introduzione ai problemi principali e alle soluzioni che sono state proposte.

Handley, Crowcroft, *The World Wide Web-Beneath the Surf*

Mentre il 99% dei libri sul WWW dice solo come si usa un certo browser o elenca URL interessanti, questo spiega come funziona dentro. Vengono esposti il lato servente, il lato cliente e HTML.

Kaufman et al., *Network Security*

Questo libro autorevole e spesso spiritoso è la prima scelta in cui guardare per informazioni sulla sicurezza di rete. Sono spiegati in dettaglio algoritmi e protocolli a chiave pubblica o segreta, hash di messaggi, autenticazione, Kerberos e la posta elettronica. Le parti migliori sono i dialoghi tra gli autori, etichettati da pedici, come in "Io, non ho potuto forzare me₂ stesso ad essere più specifico...".

Kumar, *MBone: Interactive Multimedia on the Internet*

La quarta di copertina annuncia: "Scoprite come potete trasmettere, pubblicizzare e visualizzare i vostri prodotti su Internet". Fortunatamente questo argomento non è trattato affatto nel libro. Il suo argomento è invece l'architettura e l'implementazione di MBone, compreso molto materiale su come funziona e come si usa.

Nemeth et al., *UNIX System Administration Handbook*

Il capitolo 16 è una lunga introduzione al DNS. Espone tutti i dettagli, spiega i diversi archivi e i descrittori di risorsa con esempi numerosi. Sono anche esposti in dettaglio i programmi e gli strumenti che si usano nella gestione del DNS.

Rose, *The Internet Message*

Se volete che vi venga servita la posta elettronica in modo iconoclasta, questo libro è un buon investimento. Di tanto in tanto l'autore sale su una scatola e annuncia cosa va male nel mondo. Tuttavia, il gusto non è cattivo.

Schneier, *Applied Cryptography*, seconda ed.

Questo compendio monumentale è il peggior incubo di NSA: un solo libro che descrive ogni algoritmo crittografico conosciuto. Peggio (o meglio, dipende dal punto di vista), il libro descrive la maggior parte degli algoritmi in forma di programmi eseguibili (in C). Contiene anche 1600 riferimenti alla letteratura rilevante. Se volete davvero tenere al sicuro i vostri file, leggete questo libro.

Steinmetz, Nahrstedt, *Multimedia: Computing, Communications and Applications*

Sebbene sia un po' disordinato, questo libro copre buona parte dei temi multimediali. Gli argomenti includono audio, immagini ferme, immagini in movimento, compressione,

dispositivi ottici di memorizzazione, sistemi operativi per il multimediale, reti, ipertesti, sincronizzazione di flussi, e applicazioni multimediali.

Van der Linden, *Just Java*

Quando il primo capitolo di un libro si intitola "Vieni a casa mia, disse il ragno alla mosca", potete scommettere che è una fiaba per bimbi oppure un lavoro sul World Wide Web. Questo riguarda il Web, soprattutto il linguaggio Java e il suo ambiente. Se il lettore vuole giocare con Java, il libro include su CDROM il sistema completo.

8.2 Bibliografia

Abeymundara B.W., Kamal A.E., 1991, *High-Speed Local Area Networks and Their Performance* "Computing Surveys", giugno, vol. 23, pp. 221-64.

Abramson N., 1985, *Development of the ALOHANET*, "IEEE Trans. on Information Theory", marzo, vol. IT-31, pp. 119-23.

Adam J.A., 1995, *Privacy and Computers*, "IEEE Spectrum", dicembre, vol. 32, pp. 46-52.

Adams N., Gold R., Schilit B.N., Tso M.M., Want R., 1993, *An Infrared Network for Mobile Computers*, Proc. USENIX Mobile and Location-Independent Computing Symposium, USENIX, pp. 41-51.

Anderson R.J., 1994, *Why Cryptosystems Fail*, "Commun. of the ACM", novembre, vol. 37, pp. 32-40.

Armbuster H., 1995, *The Flexibility of ATM: Supporting Future Multimedia and Mobile Communications*, "IEEE Commun. Magazine", febbraio, vol. 33, pp. 76-84.

Armitage G.J., Adams K.M., 1995, *How Efficient is IP over ATM Anyway?*, "IEEE Network Magazine", gennaio-febbraio, vol. 9, pp. 18-26.

Arnold K., Gosling J., 1996, *The Java Programming Language*, Addison-Wesley, Reading (MA).

At&t, Bellcore, 1989, *Observations of Error Characteristics of Fiber Optic Transmission Systems*, CCITT SG XVIII, San Diego gennaio.

Awdeh R.Y., Mouftah H.T., 1995, *Survey of ATM Switch Architectures*, "Computer Networks and ISDN Systems", novembre, vol. 27, pp. 1567-613.

Bakne A., Badrinath B.R., 1995, *I-TCP: Indirect TCP for Mobile Hosts*, Proc. Fifteenth Int'l. Conf. on Distr. Computer Systems, IEEE, pp. 136-43.

Balakrishnan H., Seshan S., Katz R.H., 1995, *Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks*, Proc. ACM Mobile Computing and Networking Conf., ACM, pp. 2-11.

Ballardie T., Francis P., Crowcroft J., 1993, *Core Based Trees (CBT)*, Proc. SIGCOMM'93 Conf., ACM, pp. 85-95.

Bantz D.F., Bauchot F.J., 1994, *Wireless LAN Design Alternatives*, "IEEE Network Magazine", marzo-aprile, vol. 8, pp. 43-53.

Baransel C., Dobosiewicz W., Gburzynski P., 1995, *Routing in Multihop Packet Switching Networks: Gb/s Challenge*, "IEEE Network Magazine", maggio-giugno, vol. 9, pp. 38-61.

8.2 Bibliografia

Barlow J.P., 1995, *Property and Speech: Who Owns What You Say in Cyberspace*, "Commun. of the ACM", dicembre, vol. 38, pp. 19-22.

Batcher K.E., 1968, *Sorting Networks and Their Applications*, Proc. AFIPS Spring Joint Computer Conf., vol. 32, pp. 307-15.

Bates R.J., 1994, *Wireless Networked Communications*, McGraw-Hill, New York.

Berghel H.L., 1996, *The Client Side of the Web*, "Commun. of the ACM", gennaio, vol. 39, pp. 33-40.

Bellamy J., 1991, *Digital Telephony*, John Wiley, New York.

Bellman R.E., 1957, *Dynamic Programming*, Princeton University Press, Princeton, (NJ).

Belsnes D., 1975, *Flow Control in the Packet Switching Networks*, "Communications Networks", Uxbridge, England: Online", pp. 349-61.

Berners-Lee T., Caillau A., Louton A., Nielsen H.F., Secret A., 1994, *The World Wide Web*, "Commun. of the ACM", agosto, vol. 37, pp. 76-82.

Bertsekas D., Gallager R., 1992, *Data Networks*, seconda ed., Prentice Hall, Englewood Cliffs (NJ).

Bharghavan V., Demers A., Shenker S., Zhang L., 1994, *MACAW: A Media Access Protocol for Wireless LANs*, Proc. SIGCOMM 1994 Conf. ACM, pp. 212-25.

Biham E., Shamir A., 1993, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, New York.

Binder R., 1975, *A Dynamie Packet Switching System for Satellite Broadcast Channels*, Proc. Int'l. Conf. on Commun., pp. 41-1/41-5a.

Black U.D., 1993, *Data Link Protocols*, Prentice Hall, Englewood Cliffs, (NJ).

— 1994, *Emerging Communications Technologies*, Prentice Hall, Englewood Cliffs, (NJ).

— 1995, *TCP/IP and Related Protocols*, McGraw-Hill, New York.

Blaze M., 1994, *Protocol Failure in the Escrowed Encryption Standard*, Proc. Second ACM Conf. on Computer and Commun. Security, ACM, pp. 59-67.

Bogineni K., Sivalingam K.M., 1993, *Low-Complexity Multiple Access Protocols for Wavelength-Division Multiplexed Photonic Networks*, "IEEE Journal on Selected Areas in Commun.", maggio, vol. 11, pp. 590-604.

Bonomi F., Fendick K.W., 1995, *The Rate-Based Flow Control Framework for the Available Bit-rate ATM Service*, "IEEE Network Magazine", marzo, vol. 9, pp. 25-39.

Bowman C.M., Danzig P.B., Manber U., Schwartz M.F., 1994, *Scalable Internet Resource Discovery: Research Problems and Approaches*, "Commun. of the ACM", agosto, vol. 37, pp. 98-107.

Bowman C. M., Danzig P.B., Hardy D.R., Manber U., Schwartz M.F., 1995, *The Harvest Information Discovery and Access System*, "Computer Networks and ISDN Systems", dicembre, vol. 28, pp. 119-25.

Brakmo L.S., O'Malley S.W., Peterson L.L., 1994, *TCP Vegas: New Techn. for Congestion Detection and Avoidance*, Proc. SIGCOMM 1994 Conf. ACM, pp. 24-35.

Broadhead M.A., Owen C.B., 1995, *Direct Manipulation of MPEG Compressed Digital Audio*, Proc. of ACM Multimedia 1995, ACM, pp. 499-507.

Brown L., Kwan M., Pieprzyk J., Seberry J., 1991, *Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI*, ASIACRYPT 1991 Abstracts, pp. 25-30.

- Buford J.F.K. (ed.), 1994, *Multimedia Systems*, Addison-Wesley, Reading (MA).
- Burrows M., Abadi M., Needham R.M., 1990, *A Logic of Authentication*, "DEC System Research Center Report", febbraio.
- Campbell A., Coulson G., Hutchison D., 1994, *A Quality of Service Architecture*, "Computer Commun. Rev.", aprile, vol. 24, pp. 6-27.
- Campione M., Walrath K., 1996, *The Java Language Tutorial: Object-Oriented Programming for the Internet*, Addison-Wesley, Reading (MA).
- Capetanakis J.I., 1979, *Tree Algorithms for Packet Broadcast Channels*, "IEEE Trans. on Information Theory", settembre, vol. IT-25, pp. 505-15.
- Carl-Mitchell S., Quarterman J.S., 1993, *Practical Internetworking with TCP/IP and UNIX*, Addison-Wesley, Reading (MA).
- Catlett C.E., 1992, *In Search of Gigabit Applications*, "IEEE Commun. Magazine", aprile, vol. 30, pp. 42-51.
- Cerf V., Kahn R., 1974, *A Protocol for Packet Network Interconnection*, "IEEE Trans. on Commun.", maggio, vol. COM-22, pp. 637-48.
- Chandranmenon G.P., Varghese G., 1995, *Trading Packet Headers for Packet Processing*, Proc. SIGCOMM 1995 Conf., ACM, pp. 162-73.
- Chang Y.-H., Coggins D., Pitt D., Skellern D., Thapar M., Venkatraman C., 1994, *An Open-System Approach to Video on Demand*, "IEEE Commun. Magazine", maggio, vol. 32, pp. 68-80.
- Chao J.J., Ghosal D., Saha D., Tripathi S.K., 1994, *IP on ATM Local Area Networks*, "IEEE Commun. Magazine", agosto, vol. 32, pp. 52-9.
- Chapman D.E., Zwicky E.D., 1995, *Building Internet Firewalls*, O'Reilly, Sebastopol (CA).
- Chen K.-C., 1994, *Medium Access Control of Wireless LANs for Mobile Computing*, "IEEE Network Magazine", settembre-ottobre, vol. 8, pp. 50-63.
- Chen M., Yum T.-S., 1991, *A Conflict-Free Protocol for Optical WDMA Networks*, Proc. Globecom 1991, pp. 1276-81.
- Chen W.Y., Waring D.L., 1994, *Applicability of ADSL to Support Video Dial Tone in the Copper Loop*, "IEEE Commun. Magazine", maggio, vol. 32, pp. 102-6.
- Cheriton D., Williamson C., 1989, *VMTP as the Transport Layer for High-Performance Distributed Systems*, "IEEE Commun. Magazine", giugno, vol. 27, pp. 37-44.
- Chervenak A.L., 1994, *Tertiary Storage: An Evaluation of New Applications*, Ph. D. thesis., CSD, Univ. of California at Berkeley.
- Chervenak A.L., Patterson D.A., Katz R.H., 1995, *Choosing the Best Storage System for Video Service*, Proc. of ACM Multimedia 1995, ACM, pp. 109-19.
- Chesson G.L., 1989, *XTP/PE Design Considerations*, IFIP Workshop on Protocols for High-Speed Networks, IFIP, pp. 27-33.
- Cheswick W.R., Bellovin S.M., 1994, *Firewalls and Interwalls-Repelling the Wily Hacker*, Addison-Wesley, Reading, (MA).
- Choudhury A.K., Maxemchuk N.F., Paul S., Schulzrinne H.G., 1995, *Copyright Protection for Electronic Publishing on Computer Networks*, "IEEE Network Magazine", maggio-giugno, vol. 9, pp. 12-20.
- Clark D.D., 1982, *Window and Acknowledgement Strategy in TCP*, "RFC 813", luglio.

- Clark D.D., 1987, *NETBLT: A Bulk Data Transfer Protocol*, "RFC 998".
- 1988, *The Design Philosophy of the DARPA Internet Protocols*, Proc. SIGCOMM 1988 Conf., ACM, pp. 106-14.
- Clark D.D., Davie B.S., Farber D.J., Gopal I.S., Kadaba B.K., Sincoskie W.D., Smith J.M., Tennehouse D.L., 1993, *The Aurora Gigabit Testbed*, "Computer Networks and ISDN Systems", gennaio, vol. 25, pp. 599-621.
- Clark D.D., Jacobson V., Romkey J., Salwen H., 1989, *An Analysis of TCP Processing Overhead*, "IEEE Commun. Magazine", giugno, vol. 27, pp. 23-9.
- Clark D.D., Lambert M., Zhang L., 1987, *NETBLT: A High Throughput Transport Protocol*, Proc. SIGCOMM 1987 Conf., ACM, pp. 353-9.
- Clos C., 1953, *A Study of Non-Blocking Switching Networks*, "Bell System Tech. J.", marzo, vol. 32, pp. 406-24.
- Comer D.E., 1995a, *The Internet Book*, Prentice Hall, Englewood Cliffs (NJ).
- 1995b, *Internetworking with TCP/IP*, terza ed., vol. 1, Prentice Hall, Englewood Cliffs (NJ).
- Cook A., Stern J., 1994, *Optical Fiber Access-Perspectives Toward the 21st Century*, "IEEE Commun. Magazine", febbraio, vol. 32, pp. 78-86.
- Cooper E., 1986, *Broadband Network Technology*, Prentice Hall, Englewood Cliffs (NJ).
- Couloris G.F., Dollimore J., Kindberg T., 1994, *Distributed Systems: Concepts and Design*, seconda ed., Addison-Wesley, Reading (MA).
- Crespo P.M., Honig M.L., Salehi J.A., 1995, *Spread-Time Code-Division Multiple Access*, "IEEE Trans. on Commun.", giugno, vol. 43, pp. 2139-48.
- Cronin W.J., Hutchinson J.D., Ramakrishnan K.K., Yang H., 1994, *A Comparison of High Speed LANs*, Proc. 19th Conf. on Local Computer Networks, IEEE, pp. 40-9.
- Crowcroft J., Wang Z., Smith A., Adams J., 1995, *A Rough Comparison of the IETF and ATM Service Models* "IEEE Network Magazine", novembre-dicembre, vol. 9, pp. 12-6.
- Crowther W., Rettberg R., Walden D., Ornstein S., Heart F., 1973, *A System for Broadcast Communication: Reservation-Aloha*, Proc. Sixth Hawaii Int. Conf. System Sci., pp. 371-74.
- Cusick T. W., Wood M.C., 1991, *The REDOC-II Cryptosystem*, Advances in Cryptology-CRYPTO 1990 Proceedings, Springer-Verlag, New York, pp. 545-63.
- Dagdeviren N., Newell J.A., Spindel L.A., Stefanick M.J., 1994, *Global Networking with ISDN*, "IEEE Commun. Magazine", giugno, vol. 32, pp. 26-32.
- Danskin J. M., Davis G.M., Song X., 1995, *Fast Lossy Internet Image Transmission*, Proc. of ACM Multimedia 1995, ACM, pp. 321-32.
- Danthine A.A.S., 1980, *Protocol Representation with Finite-State Models*, "IEEE Trans. on Commun.", aprile, vol. COM-28, pp. 632-43.
- Davis P.T., McGuffin C.R., 1995, *Wireless Local Area Networks*, McGraw-Hill, New York.
- Dav J.D., 1995, *The (Un) Revised OSI Reference Model*, "Computer Commun. Rev.", ottobre, vol. 25, pp. 39-55.
- Day J.D., Zimmermann, H., 1983, *The OSI Reference Model*, Proc. of the IEEE, dicembre, vol. 71, pp. 1334-40.

- De Jonge W., Chaum D., 1987, *Some Variations on RSA Signatures and Their Security*, in A. M. (ed.) Odlyzko, *Advances in Cryptology ecc.*, Springer Verlag, New York.
- De Prycker M., 1993, *Asynchronous Transfer Mode*, seconda ed., Ellis Horwood, New York.
- Dean D., Wallach D.S., 1995, *Security Flaws in the HotJava Web Browser*, Technical Report 502, Dept. of Computer Science, Princeton Univ.
- Deering S.E., 1993, *SIP: Simple Internet Protocol*, "IEEE Network Magazine", maggio-giugno, vol. 7, pp. 16-28.
- Deering S.E., Cheriton D.R., 1990, *Multicast Routing in Datagram Internetworks and Extended LANS*, "ACM Trans. on Computer Systems", maggio, vol. 8, pp. 85-110.
- Deering S.E., Estrin D., Farinacci D., Jacobson V., Liu C.-G., Wei L., 1994, *An Architecture for Wide-Area Multicast Routing*, Proc. SIGCOMM 1994 Conf., ACM, pp. 126-35.
- Deloddere D., Verbiest W., Verhille H., 1994, *Interactive Video on Demand*, "IEEE Commun. Magazine", maggio, vol. 32, pp. 82-8.
- Demers A., Keshav S., Shenker S., 1990, *Analysis and Simulation of a Fair Queueing Algorithm*, "Internetwork: Research and Experience", settembre, vol. 1, pp. 3-26.
- Denning D.E., Sacco G.M., 1981, *Timestamps in Key Distribution Protocols*, "Commun. of the ACM", agosto, vol. 24, pp. 533-6.
- Diffee W., Hellman M.E., 1976, *New Directions in Cryptography*, "IEEE Trans. on Information Theo", novembre, vol. IT-22, pp. 644-54.
- 1977, *Exhaustive Cryptanalysis of the NBS Data Encryption Standard*, "IEEE Computer Magazine", giugno, vol. 10, pp. 74-84.
- Dijkstra E.W., 1959, *A Note on Two Problems in Connexion with Graphs*, "Numer. Math.", ottobre, vol. 1, pp. 269-71.
- Dirvin R.A., Miller A.R., 1986, *The MC68824 Token Bus Controller: VLSI for the Factory LAN*, "IEEE Micro Magazine", giugno, vol. 6, pp. 15-25.
- Dixit S., Skelly P., 1995, *MPEG-2 over ATM for Video Dial Tone Network*, "IEEE Network Magazine", settembre-ottobre, vol. 9, pp. 30-40.
- Dixon R.C., 1987, *Lore of the Token Ring*, "IEEE Network Magazine", gennaio-febbraio, vol. 1, pp. 11-8.
- Doeringer W.A., Dykeman D., Kaiserswerth M., Meister B.W., Rudin H., Williamson R., 1990, *A Survey of Light-Weight Transport Protocols for High-Speed Networks*, "IEEE Trans. on Commun.", novembre, vol. 38, pp. 2025-39.
- Dorfman R., 1943, *Detection of Defective Members of a Large Population*, Annals Math. Statistics, vol. 14, pp. 436-40.
- Eckberg A.E., 1992, *B-ISDN/ATM Traffic and Congestion Control*, "IEEE Network Magazine", settembre-ottobre, vol. 6, pp. 28-37.
- Eckberg A.E., Doshi B.T., Zoccolillo R., 1991, *Controlling Congestion in B-ISDN/ATM: Issues and Strategies*, "IEEE Commun. Magazine", settembre, vol. 29, pp. 64-70.
- Edwards A., Muir S., 1995, *Experience Implementing a High-Performance TCP in User-Space*, Proc. SIGCOMM 1995 Conf., ACM, pp. 197-205.
- El Gamal T., 1985, *A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, "IEEE Trans. on Information Theory", luglio, vol. IT-31, pp. 469-72.

- Eriksson H., 1994, *MBone: The Multicast Backbone*, "Commun. of the ACM", agosto, vol. 37, pp. 54-60.
- Estrin D., Rekhter I.Y., Hotz S., 1992, *Scalable Inter-Domain Routing Architecture*, Proc. SIGCOMM 1992 Conf., ACM, pp. 40-52.
- Feig E., Winograd S., 1992, *Fast Algorithms for Discrete Cosine Transformations*, "IEEE Trans. on Signal Processing", settembre, vol. 40.
- Feit S., 1995, *SNMP-A Guide to Network Management*, McGraw-Hill, New York.
- Fiorini D., Chiani M., Tralli V., Salati C., 1995, *Problems with HDLC*, "Computer Commun. Rev.", ottobre, vol. 25, pp. 61-80.
- Fischer W., Wallmeier E., Worster T., Davis S.P., Hayter A., 1994, *Data Communications Using ATM: Architectures, Protocols, and Resource Management*, "IEEE Commun. Magazine", agosto, vol. 32, pp. 24-33.
- Floyd S., Jacobson V., 1993, *Random Early Detection for Congestion Avoidance*, IEEE/ACM Trans. on Networking, agosto, vol. 1, pp. 397-413.
- Fluckiger F., 1995, *Understanding Networked Multimedia*, Prentice Hall, Englewood Cliffs (NJ).
- Ford L.R., Jr., Fulkerson D.R., 1962, *Flows in Networks*, Princeton University, Princeton (NJ).
- Ford P.S., Rekhter Y., Braun H.-W., 1993, *Improving the Routing and Addressing of IP*, "IEEE Network Magazine", maggio-giugno, vol. 7, pp. 10-5.
- Forman G.H., Zahorjan J., 1994, *The Challenges of Mobile Computing*, "IEEE Computer Magazine", aprile, vol. 27, pp. 38-47.
- Francis P., 1993, *A Near-Term Architecture for Deploying Pip*, "IEEE Network Magazine", maggio-giugno, vol. 7, pp. 30-7.
- Fraser A.G., 1993, *Early Experiments with Asynchronous Time Division Networks*, "IEEE Network Magazine", gennaio-febbraio, vol. 7, pp. 12-27.
- Fraser A.G., 1987, *Towards a Universal Data Transport System*, in K. Kummerle, F. Tobagi, J. O. Limb (ed.), *Advances in local Area Networks*, IEEE Press, New York.
- Furht B., Kalra D., Kitson F.L., Rodriguez, Wall W.E., 1995, *Design Issues for Interactive Televisions Systems*, "IEEE Computer Magazine", maggio, vol. 28, pp. 25-39.
- Garcia-Haro J., Jajszczyk A., 1994, *ATM Shared-Memory Switching Architectures*, "IEEE Network Magazine", luglio-agosto, vol. 8, pp. 18-26.
- Garg V., Wilkes J.E., 1996, *Wireless and Personal Communication Systems*, Prentice Hall, Englewood Cliffs (NJ).
- Gasman L., 1994, *Broadband Networking*, Van Nostrand Reinhold, New York.
- Giacopelli J.N., Hickey J.J., Marcus W.S., Sincoskie W.D., Littlewood M., 1991, *Sunshine: A High-Performance Self-Routing Broadband Packet Switch Architecture*, "IEEE Journal on Selected Areas in Commun.", ottobre, vol. 9, pp. 1289-98.
- Goodman D.J., 1991, *Trends in Cellular and Cordless Communications*, "IEEE Commun. Magazine", giugno, vol. 29, pp. 31-40.
- Goralski W.J., 1995, *Introduction to ATM Networking*, McGraw-Hill, New York.
- Gosling J., Joy B., Steele G., 1996, *The Java Language Specification*, Addison-Wesley, Reading (MA).
- Green P.F., ir., 1993, *Fiber Optic Networks*, Prentice Hall, Englewood Cliffs (NJ).

- Hac A., 1995, *Wireless and Cellular Architecture and Services*, "IEEE Commun. Magazine", novembre, vol. 33, pp. 98-104.
- Hafner K., Markoff J., 1991, *Cyberpunk*, Simon and Schuster, New York.
- Hamming R.W., 1950, *Error Detecting and Error Correcting Codes*, "Bell System Tech. J.", aprile, vol. 29, pp. 147-60.
- Handel R., Huber M.N., Schroder S., 1994, *ATM Concepts, Protocols, and Applications*, seconda ed., Addison-Wesley, Reading (MA).
- Handley M., Crowcroft J., 1994, *The World Wide Web-Beneath the Surf*, UCL Press, Londra.
- Hawley G.T., 1991, *Historical Perspectives on the U. S. Telephone System*, "IEEE Commun. Magazine", marzo, vol. 29, pp. 24-8.
- Hein M., Griffiths D., 1995, *SNMP*, Thompson, Londra.
- Held G., 1994, *The Complete Modem Reference*, seconda ed., John Wiley, New York.
- Hellman M.E., 1980, *A Cryptanalytic Time-Memory Tradeoff*, IEEE Trans. on Information Theory, luglio, vol. IT-26, pp. 401-6.
- Henderson T.R., 1995, *Design Principles and Performance Analysis of SSCOP: A New ATM Adaptation Layer Protocol*, "Computer Commun. Review", aprile, vol. 25, pp. 47-59.
- Hoare C.A.R., 1974, *Monitors, An Operating System Structuring Concept*, "Commun. of the ACM", ottobre, vol. 17, pp. 549-57, 1975, *Erratum* in "Commun. of the ACM" febbraio, vol. 18, p. 95.
- Hodge W.W., 1995, *Interactive Television*, McGraw-Hill, New York.
- Hodge W.W., Martin S., Powers J.T., Jr., 1993, *Video on Demand: Architectures, Systems, and Applications*, "Society of Motion Picture and Television Engineers Journal", settembre, vol. 102, pp. 791-803.
- Hoffman L.J. (ed.), 1995, *Building in Big Brother: The Cryptographic Policy Debate*, Springer-Verlag, New York.
- Hofelder W., 1995, *MBone VCR-Video Conference Recording on the MBone*, Proc. of ACM Multimedia 1995, ACM, pp. 237-8.
- Holzmann G.J., 1991, *Design and Validation of Computer Protocols*, Prentice Hall, Englewood Cliffs (NJ).
- Hong D., Suda T., 1991, *Congestion Control and Prevention in ATM Networks*, "IEEE Network Magazine", luglio-agosto, vol. 5, pp. 10-6.
- Huang A., Knauer S., 1984, *Starlite: A Wideband Digital Switch*, Proc. Globecom. 1984, pp. 121-5.
- Hughes J.P., Franta W.R., 1994, *Geographic Extension of HIPPI Channels*, "IEEE Network Magazine", maggio-giugno, vol. 8, pp. 42-53.
- Hui J., 1987, *A Broadband Packet Switch for Multi-rate Services*, Proc. Int'l. Conf. on Communications, IEEE, pp. 782-8.
- Huitema C., 1995, *Routing in the Internet*, Prentice Hall, Englewood Cliffs (NJ).
- 1996, *IPv6: The New Internet Protocol*, Prentice Hall, Englewood Cliffs (NJ).
- Humblet P.A., Ramaswami R., Sivarajan K.N., 1992, *An Efficient Communication Protocol for High-Speed Packet-Switched Multichannel Networks*, Proc. SIGCOMM 1992 Conf., ACM, pp. 2-13.

- IEEE, 1995, "Communications Magazine", gennaio, vol. 33.
- IEEE, 802.3, 1985a, *Carrier Sense Multiple Access with Collision Detection*, IEEE, New York.
- IEEE, 802.4, 1985b, *Token-Passing Bus Access Method*, IEEE, New York.
- IEEE, 802.5, 1985c, *Token Ring Access Method*, IEEE, New York.
- Ioannidis J., Maquire G.Q. Jr., 1993, *The Design and Implementation of a Mobile Internetworking Architecture*, Proc. Winter USENIX Conf., USENIX, gennaio, pp. 491-502.
- Irmer T., 1994, *Shaping Future Telecommunications: The Challenge of Global Standardization*, "IEEE Commun. Magazine", gennaio, vol. 32, pp. 20-8.
- Ivancic W.D., Shalkausier M.J., Quintana J.A., 1994, *A Network Architecture for a Geostationary Communication Satellite*, "IEEE Commun. Magazine", luglio, vol. 32, pp. 72-84.
- Jabbari B., Colombo G., Nakajima A., Kulkarni J., 1995, *Network Issues for Wireless Communications*, "IEEE Commun. Magazine", gennaio, vol. 33, pp. 88-98.
- Jacobson V., 1988, *Congestion Avoidance and Control*, Proc. SIGCOMM 1988 Conf., ACM, pp. 314-29.
- Jain R., 1990, *Congestion Control in Computer Networks: Issues and Trends*, "IEEE Network Magazine", maggio-giugno, vol. 4, pp. 24-30.
- 1991, *The Art of Computer Systems Performance Analysis*, John Wiley, New York.
- 1994, *FDDI Handbook-High-Speed Networking Using Fiber and other Media*, Addison-Wesley, Reading (MA).
- 1995, *Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey*, "Computer Networks and ISDN Systems", novembre, vol. 27.
- Jia F., Mukherjee B., 1993, *The Receiver Collision Avoidance (RCA) Protocol for a Single-Hop WDM Lightwave Network*, "Journal of Lightwave Technology", maggio-giugno, vol. 11, pp. 1053-65.
- Johnson D.B., 1995, *Scalable Support for Transparent Mobile Host Internetworking*, "Wireless Networks", ottobre, vol. 1, pp. 311-21.
- Johnson H.W., 1996, *Fast Ethernet-Dawn of a New Network*, Prentice Hall, Englewood Cliffs (NJ).
- Kahn D., 1967, *The Codebreakers*, Macmillan, New York.
- 1980, *Cryptology Goes Public*, "IEEE Commun. Magazine", marzo, vol. 18, pp. 19-28.
- Kaliski B.S., Robshaw M.J.B., 1994, *Fast Block Cipher Proposal*, Proc. Cambridge Security Workshop, Springer-Verlag, New York, pp. 26-39.
- Kamoun F., Kleinrock L., 1979, *Stochastic Performance Evaluation of Hierarchical Routing for Large Networks*, "Computer Networks", novembre, vol. 3, pp. 337-53.
- Karn P., 1990, *MACA-A New Channel Access Protocol for Packet Radio*, ARRL/CRRRL Amateur Radio Ninth Computer Networking Conf., pp. 134-40.
- Karol M.J., Hluchyj M.G., Morgan S.P., 1987, *Input Versus Output Queueing on a Space-Division Packet Switch*, "IEEE Trans. on Commun.", dicembre, vol. 35, pp. 1347-56.
- Karshmer A.I., Thomas J.N., 1992, *Computer Networking on Cable TV Plants*, "IEEE Commun. Magazine", novembre, vol. 30, pp. 32-40.

- Katz D., Ford P.S., 1993, *TUBA: Replacing IP with CLNP*, "IEEE Network Magazine", maggio-giugno, vol. 7, pp. 38-47.
- Katz D., Butler M., McGrath R., 1994, *A Scalable HTTP Server: The NCSA Prototype*, "Computer Networks and ISDN Systems", novembre, vol. 27, pp. 155-64.
- Kaufman C., Perlman R., Speciner M., 1995, *Network Security*, Prentice Hall, Englewood Cliffs (NJ).
- Kavak N., 1995, *Data Communication in ATM Networks*, "IEEE Network Magazine", maggio-giugno, vol. 9, pp. 28-37.
- Kent C.A., Mogul J.C., 1987, *Fragmentation Considered Harmful*, Proc. SIGCOMM 1987 Conf., ACM, pp. 390-401.
- Kent S.T., 1993, *Internet Privacy Enhanced Mail*, "Commun. of the ACM", agosto, vol. 36, pp. 48-60.
- Kessler G.C., 1993, *ISDN*, seconda ed., McGraw-Hill, New York.
- Kessler G.C., Train D., 1992, *Metropolitan Area Networks: Concepts, Standards, and Services*, McGraw-Hill, New York.
- Kim J.B., Suda T., Yoshimura M., 1994, *International Standardization of B-ISDN*, "Computer Networks and ISDN Systems", ottobre, vol. 27, pp. 5-27.
- Kleinrock L., Tobagi F., 1975, *Random Access Techniques for Data Transmission over Packet-Switched Radio Channels*, Proc. Nat. Computer Conf., pp. 187-201.
- Kohno R., Meidan R., Milstein L.B., 1995, *Spread Spectrum Access Methods for Wireless Communication*, "IEEE Commun. Magazine", gennaio, vol. 33, pp. 58-67.
- Kumar V., 1996, *MBone: Interactive Multimedia on the Internet*, New Riders, Indianapolis (IN).
- Kung H.T., Morris R., 1995, *Credit-Based Flow Control for ATM Networks*, "IEEE Network Magazine", marzo-aprile, vol. 9, pp. 40-4.
- Kwan T.T., McGrath R.E., Reed D.A., 1995, *NCSA's WWW Server: Design and Performance*, "IEEE Computer Magazine", novembre, vol. 28, pp. 68-74.
- Kwok T., 1995, *A Vision for Residential Broadband Service: ATM to the Home*, "IEEE Network Magazine", settembre-ottobre, vol. 9, pp. 14-28.
- Kyas O., 1995, *ATM Networks*, International Thomson Publishing, Londra.
- Lai X., 1992, *On the Design and Security of Block Ciphers*, Hartung-Gorre, Costanza.
- Lai X., Massey J., 1990, *A Proposal for a New Block Encryption Standard*, Advances in Cryptology-Eurocrypt 1990 Proceedings, Springer-Verlag, New York, pp. 389-404.
- Lampson B.W., 1973, *A Note on the Confinement Problem*, "Commun. of the ACM", ottobre, vol. 10, pp. 613-5.
- Landau S., 1988, *Zero-Knowledge and the Department of Defense*, "Notices of the American Mathematical Society", gennaio, vol. 35, pp. 5-12.
- Langford A., 1984, *The Open System User's Programming Interfaces*, "Computer Networks", vol. 8, pp. 3-12.
- La Porta T.F., Veeraraghavan M., Ayanoglu E., Karol M., Gitlin R.D., 1994, *B-ISDN: A Technological Discontinuity*, "IEEE Commun. Magazine", ottobre, vol. 32, pp. 84-97.
- Latif A., Rowlance E.J., Adams R.H., 1992, *The IBM 8209 LAN Bridge*, "IEEE Network Magazine", maggio-giugno, vol. 6, pp. 28-37.

- Laudon K.C., 1995, *Ethical Concepts and Information Technology*, "Commun. of the ACM", dicembre, vol. 38, pp. 33-9.
- Le Boudec J.-Y., 1992, *The Asynchronous Transfer Mode: A Tutorial*, "Computer Networks and ISDN Systems", maggio, vol. 24, pp. 279-309.
- Leiner B.M., Cole R., Postel J., Mills D., 1985, *The DARPA Internet Protocol Suite*, "IEEE Commun. Magazine", marzo, vol. 23, pp. 29-34.
- Levine D.A., Akyildiz I.A., 1995, *PROTON: A Media Access Control Protocol for Optical Networks with Star Topology*, "IEEE/ACM Trans. on Networking", aprile, vol. 3, pp. 158-68.
- Levy S., *Crypto Rebels*, 1993, "Wired", maggio-giugno, pp. 54-61.
- Lin F., Chu P., Liu M., 1987, *Protocol Verification Using Reachability Analysis: The State Space Explosion Problem and Relief Strategies*, Proc. SIGCOMM 1987 Conf., ACM, pp. 126-35.
- Lipper E.H., Rumsewicz M.P., 1994, *Teletraffic Considerations for Widespread Deployment of PCS*, "IEEE Network Magazine", settembre-ottobre, vol. 8, pp. 40-9.
- Little T.D.C., Venkatesh D., 1994, *Prospects for Interactive Video on Demand*, "IEEE Multimedia Magazine", vol. 1, Fall, pp. 14-24.
- Liu C.L., Layland J.W., 1973, *Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment*, "Journal of the ACM", gennaio, vol. 20, pp. 46-61.
- Luotonen A., Altis K., 1994, *World Wide Web Proxies*, "Computer Networks and ISDN Systems", novembre, vol. 27, pp. 147-54.
- Macario R.C.V., 1993, *Cellular Radio-Principles and Design*, McGraw-Hill, New York.
- Macedonia M.R., Brutzman D.P., 1994, *MBone Provides Audio and Video Across the Internet*, "IEEE Computer Magazine", aprile, vol. 27, pp. 30-6.
- Massey J.L., 1994, *SAFER K-64: A Byte-Oriented Block Ciphering Algorithm*, Proc. Cambridge Security Workshop, Springer-Verlag, New York, pp. 1-17.
- Matsui M., 1994, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology-Eurocrypt 1993 Proceedings, Springer-Verlag, New York, pp. 386-97.
- McBryan O., 1994, *GENVL and WWW: Tools for Taming the Web*, Proc. First Int'l. WWW Conference, pp. 79-90.
- McDysan D.E., Spohn D.L., 1995, *ATM-Theory and Application*, McGraw-Hill, New York.
- McKenney P.E., Dove K.F., 1992, *Efficient Demultiplexing of Incoming TCP Packets*, Proc. SIGCOMM 1992 Conf., ACM, pp. 269-79.
- Menezes A.J., Vanstone S.A., 1993, *Elliptic Curve Cryptosystems and Their Implementation*, "Journal of Cryptology", vol. 6, pp. 209-24.
- Merkle R.C., 1991, *Fast Software Encryption Functions*, Advances in Cryptology-CRYPTO 1990 Proceedings, Springer-Verlag, New York, pp. 476-501.
- 1978, *Hiding and Signatures in Trapdoor Knapsacks*, "IEEE Trans. on Information Theory", settembre, vol. IT-24, pp. 525-30.
- Merkle R.C., Hellman M., 1981, *On the Security of Multiple Encryption*, "Commun. of the ACM", luglio, vol. 24, pp. 465-67.
- Metcalfe R.M., 1993, *Computer/Network Interface Design: Lessons from Arpanet and Ethernet*, "IEEE Journal on Selected Areas in Commun.", febbraio, vol. 11, pp. 173-79.

- Metcalfe R.M., 1995, *On Mobile Computing*, "Byte", settembre, vol. 20, p. 110.
- Metcalfe R.M., Boggs D.R., 1976, *Ethernet: Distributed Packet Switching for Local Computer Networks*, "Commun. of the ACM", luglio, vol. 19, pp. 395-404.
- Miki T., 1994a, *The Potential of Photonic Networks*, "IEEE Commun. Magazine", dicembre, vol. 32, pp. 23-7.
- Miki T., 1994b, *Toward the Service-Rich Era*, "IEEE Commun. Magazine", febbraio, vol. 32, pp. 34-9.
- Minoli D., 1995, *Video Dialtone Technology*, McGraw-Hill, New York.
- Minoli D., Vitella M., 1994, *ATM & Cell Relay for Corporate Environments*, McGraw-Hill, New York.
- Mirchandani S., Khanna R. (ed.), 1993, *FDDI Technologies and Applications*, John Wiley, New York.
- Mishra P.P., Kanakia H., 1992, *A Hop by Hop Rate-Based Congestion Control Scheme*, Proc. SIGCOMM 1992 Conf., ACM, pp. 112-23.
- Mochida Y., 1994, *Technologies for Local-Access Fiberizing*, "IEEE Commun. Magazine", febbraio, vol. 32, pp. 64-73.
- Mogul J.C., 1995, *The Case for Persistent-Connection HTTP*, Proc. SIGCOMM 1995 Conf. ACM, pp. 299-314.
- 1993, *IP Network Performance*, "in D. C. Lynch, M. T. Rose (ed.), Internet System Handbook", Addison Wesley, Reading (MA), pp. 575-675.
- Mok A.K., Ward S.A., 1979, *Distributed Broadcast Channel Access*, "Computer Networks", novembre, vol. 3, pp. 327-35.
- Morales J., Patka A., Choa P., Kui J., 1995, *Video Dial Tone Sessions*, "IEEE Network Magazine", settembre-ottobre, vol. 9, pp. 42-7.
- Moy J., 1994, *Multicast Routing Extensions*, "Commun. of the ACM", agosto, vol. 37, pp. 61-6.
- Mullender S.J. (ed.), 1993, *Distributed Systems*, seconda ed., ACM Press, New York.
- Myles A., Skellern D., 1993, *Comparison of Mobile Host Protocols for IP*, "Computer Networks and ISDN Systems", dicembre, vol. 26, pp. 349-55.
- Nagle J., 1984, *Congestion Control in TCP/IP Internetworks*, "Computer Commun. Rev.", ottobre, vol. 14, pp. 11-7.
- 1987, *On Packet Switches with Infinite Storage*, "IEEE Trans. on Commun.", aprile, vol. COM-35, pp. 435-8.
- Needham R.M., Schroeder M.D., 1978, *Using Encryption for Authentication in Large Networks of Computers*, "Commun. of the ACM", dicembre, vol. 21, pp. 993-9.
- 1987, Schroeder M.D., *Authentication Revisited*, "Operating Systems Rev.", gennaio, vol. 21, p. 7.
- Nelson M.N., Linton M., 1995, *A Highly Available, Scalable ITV System*, Proc. Fifteenth Symp. on Operating Systems Prin., ACM, pp. 54-67.
- Nemeth E., Snyder G., Seebass S., Hein T.R., 1995, *UNIX System Administration Handbook*, Prentice Hall, Englewood Cliffs (NJ).
- Nemzow M., 1995, *Implementing Wireless Networks*, McGraw-Hill, New York.

- Neuman B.C., Ts' O T., 1994, *Kerberos: An Authentication Service for Computer Networks*, "IEEE Commun. Magazine", settembre, vol. 32, pp. 33-8.
- Newman P., 1994a, *ATM Local Area Networks*, "IEEE Commun. Magazine", marzo, vol. 32, pp. 86-98.
- 1994b, *Traffic Management for ATM Local Area Networks*, "IEEE Commun. Magazine", agosto, vol. 32, pp. 44-50.
- NIST, 1993, *Secure Hash Algorithm*, US. Government Federal Information Processing Standard 180.
- Omidyar C.G., Aldridge A., 1993, *Introduction to SDH/SONET*, "IEEE Commun. Magazine", settembre, vol. 31, pp. 30-3.
- Otway D., Rees O., 1987, *Efficient and Timely Mutual Authentication*, "Operating Systems Rev.", gennaio, pp. 8-10.
- Padgett J.E., Gunther C.G., Hattori T., 1995, *Overview of Wireless Personal Communications*, "IEEE Commun. Magazine", gennaio, vol. 33, pp. 28-41.
- Paff A., 1995, *Hybrid Fiber/Coax in the Public Telecommunications Infrastructure*, "IEEE Commun. Magazine", aprile, vol. 33, pp. 40-5.
- Pahlavan K., Probert T.H., Chase M.E., 1995, *Trends in Local Wireless Networks*, "IEEE Commun. Magazine", marzo, vol. 33, pp. 88-95.
- Palais J.C., 1992, *Fiber Optic Commun.*, terza ed., Prentice Hall, Englewood Cliffs (NJ).
- Palmer L.C., White L.W., 1990, *Demand Assignment in the ACTS LBR System*, "IEEE Trans. on Commun.", maggio, vol. 38, pp. 684-92.
- Pan D., 1995, *A Tutorial on MPEG/Audio Compression*, "IEEE Multimedia Magazine", estate, vol. 2, pp. 60-74.
- Pancha P., El Zarki M., 1994, *MPEG Coding for Variable Bit Rate Video Transmission*, "IEEE Commun. Magazine", maggio, vol. 32, pp. 54-66.
- Pandya R., 1995, *Emerging Mobile and Personal Communication Systems*, "IEEE Commun. Magazine", giugno, vol. 33, pp. 44-52.
- Partridge C., 1992, *A Proposed Flow Specification*, "Internet RFC 1363", settembre.
- 1994, *Gigabit Networking*, Addison-Wesley, Reading (MA).
- Partridge C., Hughes J., Stone J., 1995, *Performance of Checksums and CRCs over Real Data*, Proc. SIGCOMM 1995 Conf., ACM, pp. 68-76.
- Parulkar G., Schmidt D.C., Turner J.S., 1995, *AITPM: A Strategy for Integrating IP with ATM*, Proc. SIGCOMM 1995 Conf., ACM, pp. 49-58.
- Paxson V., 1994, *Growth Trends in Wide-Area TCP Connections*, "IEEE Network Magazine", luglio-agosto, vol. 8, pp. 8-17.
- Paxson V., Floyd S., 1995, *Wide-Area Traffic: The Failure of Poisson Modeling*, Proc. SIGCOMM 1994 Conf., ACM, pp. 257-68.
- Perkins C., 1993, *Providing Continuous Network Access to Mobile Hosts Using TCP/IP*, "Computer Networks and ISDN Systems", novembre, vol. 26, pp. 357-70.
- Perlman R., 1988, *Network Layer Protocols with Byzantine Robustness*, Ph. D. thesis, MIT.
- Perlman R., 1992, *Interconnections: Bridges and Routers*, Addison-Wesley, Reading (MA).

- Perry T.S., Adam J.A., 1992, *E-Mail: Pervasive and Persuasive*, "IEEE Spectrum", ottobre, vol. 29, pp. 22-8.
- Peterson W.W., Brown D.T., 1961, *Cyclic Codes for Error Detection*, Proc. IRE, gennaio, vol. 49, pp. 228-35.
- Pickholtz R.L., Schilling D.L., Milstein L.B., 1982, *Theory of Spread Spectrum Communication-A Tutorial*, "IEEE Trans. on Commun.", maggio, vol. COM-30, pp. 855-84.
- Pierce J., 1972, *How Far Can Data Loops Go?*, "IEEE Trans. on Commun.", giugno, vol. COM-20, pp. 527-30.
- Pinkerton B., 1994, *Finding What People Want: Experiences with the WebCrawler*, Proc. First Int'l. WorldWide Web Conference.
- Piscitello D.M., Chapin A.L., 1993, *Open Systems Networking: TCP/IP and OSI*, Addison-Wesley, Reading (MA).
- Pitt D.A., 1988, *Bridging-The Double Standard*, "IEEE Network Magazine", gennaio, vol. 2, pp. 94-5.
- Quick R.F. Jr., Balachandran K., 1993, *An Overview of the Cellular Digital Packet Data (CDPD) System*, Fourth Int'l. Symp. on Personal, Indoor, and Mobile Radio Commun., pp. 338-43.
- Quisquater J.-J., Girault M., 1991, *Chinese Lotto as an Exhaustive Code-Breaking Machine*, "IEEE Computer Magazine", novembre, vol. 24, pp. 14-22.
- Rabin M.O., 1979, *Digital Signatures and Public-Key Functions as Intractable as Factorization*, Technical Report LCS-TR-212, MIT, gennaio.
- Rahnema M., 1993, *Overview of the GSM System and Protocol Architecture*, "IEEE Commun. Magazine", aprile, vol. 31, pp. 92-100.
- Rajagopalan B., 1992, *Reliability and Scaling Issues in Multicast Communication*, Proc. SIGCOMM 1992 Conf, ACM, pp. 188-98.
- Ransom M.N., 1992, *The VISTAnet Gigabit Network Testbed*, "Journal of High Speed Networks", vol. 1, pp. 49-60.
- Rao S.K., Hatamian M., 1995, *The ATM Physical Layer*, "Computer Commun. Rev.", aprile, vol. 25, pp. 73-81.
- Rivest R.L., 1992, *The MD5 Message-Digest Algorithm*, "RFC 1320", aprile.
- Rivest R.L., Shamir A., 1984, *How to Expose an Eavesdropper*, "Commun. of the ACM", aprile, vol. 27, pp. 393-5.
- Rivest R.L., Shamir A., Adleman L., 1978, *On a Method for Obtaining Digital Signatures and Public Key Cryptosystems*, "Commun. of the ACM", febbraio, vol. 21, pp. 120-6.
- Roberts L., 1972, *Extensions of Packet Communication Technology to a Hand Held Personal Terminal*, Proc. Spring Joint Computer Conference, AFIPS, pp. 295-8.
- 1973, *Dynamic Allocation of Satellite Capacity through Packet Reservation*, Proc. NCC, AFIPS, pp. 711-6.
- Romanow A., Floyd S., 1994, *Dynamics of TCP Traffic over ATM Networks*, Proc. SIGCOMM 1984 Conf., ACM, pp. 79-88.
- Rose M.T., 1993, *The Internet Message*, Prentice Hall, Englewood Cliffs (NJ).
- 1994, *The Simple Book*, Prentice Hall, Englewood Cliffs (NJ).

- Rose M.T., McCloghrie K., 1995, *How to Manage Your Network Using SNMP*, Prentice Hall, Englewood Cliffs (NJ).
- Ross F.E., Hamstra J.R., 1993, *Forging FDDI*, "IEEE Journal on Selected Areas in Commun.", febbraio, vol. 11, pp. 181-90.
- Sadiku M.N.O., Arvind A.S., 1994, *Annotated Bibliography on Distributed Queue Dual Bus (DQDB)*, "Computer Commun. Rev.", gennaio, vol. 24, pp. 21-36.
- Saltzer J.H., Pogran K.T., Clark D.D., 1983, *Why a Ring?*, "Computer Networks", agosto, vol. 7, pp. 223-30.
- Saltzer J.H., Reed D.P., Clark D.D., 1984, *End-to-End Arguments in System Design*, "ACM Trans. on Computer Systems", novembre, vol. 2, pp. 277-88.
- Sanderson D.W., Dougherty D., 1993, *Smileys*, O'Reilly, Sebastopol (CA).
- Santifaller M., 1994, *TCP/IP e ONC/NFS*, Addison-Wesley, Reading (MA).
- Schneier B., 1994, *Description of a New Variable-Length Key, 64-Bit Block Cipher [Blowfish]*, Proc. of the Cambridge Security Workshop, Springer-Verlag, pp. 191-204.
- 1995, *E-Mail Security*, John Wiley, New York.
- 1996, *Applied Cryptography*, seconda ed., John Wiley, New York.
- Schnorr C.P., 1991, *Efficient Signature Generation for Smart Cards*, "Journal of Cryptology", vol. 4, pp. 161-74.
- Scholtz R.A., 1982, *The Origins of Spread-Spectrum Communications*, "IEEE Trans. on Commun.", maggio, vol. COM-30, pp. 822-54.
- Scott R., 1985, *Wide Open Encryption Design Offers Flexible Implementations*, "Cryptologia", gennaio, vol. 9, pp. 75-90.
- Selfridge O.G., Schwartz R.T., 1980, *Telephone Technology and Privacy*, "Technology Rev.", maggio, vol. 82, pp. 56-65.
- Seybold A.M., 1994, *Using Wireless Communications in Business*, Van Nostrand Reinhold, New York.
- Shacham N., Mckenney P., 1990, *Packet Recovery in High-Speed Networks Using Coding and Buffer Management*, Proc. INFO Com. 1990, IEEE, pp. 124-30.
- Shah A., Ramakrishnan G., 1994, *FDDI-A High Speed Network*, Prentice Hall, Englewood Cliffs (NJ).
- Shannon C., 1948, *A Mathematical Theory of Communication*, "Bell System Tech. J.", luglio, vol. 27, pp. 379-423, ottobre, pp. 623-56.
- Shen B., Sethi I.K., 1995, *Inner-Block Operations on Compressed Images*, Proc. of ACM Multimedia 1995, ACM, pp. 489-98.
- Shimizu A., Miyaguchi S., 1988, *Fast Data Encipherment Algorithm FEAL*, Advances in Cryptology-Eurocrypt '87 Proceedings, New York, Springer-Verlag, pp. 267-78.
- Shreedhar M., Varghese G., 1995, *Efficient Fair Queueing Using Deficit Round Robin*, "Proc. SIGCOMM 1995 Conf.", ACM, pp. 231-43.
- Singleton A., 1996, *Wired on the Web*, "Byte", gennaio, vol. 21, pp. 77-80.
- Sipior J.C., Wardi B.T., 1995, *The Ethical and Legal Quandary of Email Privacy*, "Commun. of the ACM", dicembre, vol. 38, pp. 48-54.

- Siu K.-Y., Jain R., 1995, *A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic Management*, "Computer Commun. Rev.", aprile, vol. 25, pp. 6-20.
- Smith P., 1993, *Frame Relay*, Addison-Wesley, Reading (MA).
- Soha M., Perlman R., 1988, *Comparison of Two LAN Bridge Approaches*, "IEEE Network Magazine", gennaio-febbraio, vol. 2, pp. 37-43.
- Spafford E.H., 1989, *The Internet Worm: Crisis and Aftermath*, "Commun. of the ACM", giugno, vol. 32, pp. 678-87.
- Spragins J.D., Hammond J.L., Pawlikowski K., 1991, *Telecommunications Protocols and Design*, Addison-Wesley, Reading (MA).
- Stallings W., 1993a, *SNMP, SNMPv2, and CMIP*, Addison-Wesley, Reading (MA).
- 1993b, *Local and Metropolitan Area Networks*, quarta ed., Macmillan, New York.
 - 1994, *Data and Computer Communications*, quarta ed., Macmillan, New York.
 - 1995a, *ISDN and Broadband ISDN with Frame Relay and ATM*, Prentice Hall, Englewood Cliffs (NJ).
 - 1995b, *Network and Internetwork Security*, Prentice Hall, Englewood Cliffs (NJ).
 - 1995c, *Protect Your Privacy: The PGP User's Guide*, Prentice Hall, Englewood Cliffs (NJ).
- Steele R., Whitehead J., Wong W.C., 1995a, *System Aspects of Cellular Radio*, "IEEE Commun. Magazine", gennaio, vol. 33, pp. 80-6.
- Steele R., Williams J., Chandler D., Dehghan S., Collard A., 1995b, *Teletraffic Performance of GSM900/DCS1800 in Street Microcells*, "IEEE Commun. Magazine", marzo, vol. 33, pp. 102-8.
- Steiner J.G., Neuman B.C., Schiller J.I., 1988, *Kerberos: An Authentication Service for Open Network Systems*, Proc. Winter USENIX Conf., USENIX, pp. 191-201.
- Steinmetz R., Nahrstedt K., 1995, *Multimedia: Computing, Communications and Applications*, Prentice Hall, Englewood Cliffs (NJ).
- Stephens W.E., Banwell T.C., 1995, *155.52 Mb/s Data Transmission on Category 5 Cable Plant*, "IEEE Commun. Magazine", aprile, vol. 33, pp. 62-9.
- Sterbenz J.P.G., Schulzrinne H.G., Touch J.D., 1995, *Report and Discussion of the IEEE ComSoc TCGN Gigabit Networking Workshop 1995*, "IEEE Network Magazine", luglio-agosto, vol. 9, pp. 9-29.
- Stevens W.R., 1994, *TCP/IP Illustrated*, vol. 1, Addison-Wesley, Reading (MA).
- Stiller B., 1995, *A Survey of UNI Signaling Systems and Protocols*, "Computer Commun. Rev.", aprile, vol. 25, pp. 21-33.
- Stinson D.R., 1995, *Cryptography Theory and Practice*, CRC Press, Boca Raton (FL).
- Sunshine C.A., Dalal Y.K., 1978, *Connection Management in Transport Protocols*, "Computer Networks", vol. 2, pp. 454-73.
- Suzuki T., 1994, *ATM Adaptation Layer Protocol*, "IEEE Commun. Magazine", aprile, vol. 32, pp. 80-3.
- Tanenbaum A.S., 1992, *Modern Operating Systems*, Prentice Hall, Englewood Cliffs (NJ).
- 1995, *Distributed Operating Systems*, Prentice Hall, Englewood Cliffs (NJ).

- Teraoka F., Yokote Y., Tokoro M., 1993, *Host Migration Transparency in IP Networks*, "Computer Commun. Rev.", gennaio, vol. 23, pp. 45-65.
- Thyagarajan A.S., Deering S.E., 1995, *Hierarchical Distance-Vector Multicast Routing for the MBone*, Proc. SIGCOMM 1995 Conf., ACM, pp. 60-6.
- Tokoro M., Tamaru K., 1977, *Acknowledging Ethernet*, Compcon, Con. IEEE, Fall, pp. 320-5.
- Tolmie D.E., 1992, *Gigabit Networking*, "IEEE LTS", maggio, vol. 3, pp. 28-36.
- Tolmie D.E., 1995, *Gigabit LAN Issues-HIPPI, Fibre Channel, and ATM*, in B. Hertzberger, G. Serazzi (ed.), Proc. High-Performance Computing and Networking, Springer Verlag, Berlino, pp. 45-53.
- Tolmie D.E., Renwick J., 1993, *hippl: Simplicity Yields Success*, "IEEE Network Magazine", gennaio-febbraio, vol. 7, pp. 28-32.
- Tomlinson R.S., 1975, *Selecting Sequence Numbers*, Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop, ACM, pp. 11-23.
- Touch J.D., 1995, *Performance Analysis of MD5*, Proc. SIGCOMM 1995 Conf., ACM, pp. 77-86.
- Truong H.L., Ellington W.W.J., Le Boudec J.-Y., Meier A.X., Pace J.W., 1995, *LAN Emulation on an ATM Network*, "IEEE Commun. Magazine", maggio, vol. 33, pp. 70-85.
- Tuchman W., 1979, *Hellman Presents No Shortcut Solutions to DES*, "IEEE Spectrum", luglio, vol. 16, pp. 40-1.
- Turner J.S., 1986, *New Directions in Communications (or Which Way to the Information Age)*, "IEEE Commun. Magazine", ottobre, vol. 24, pp. 8-15.
- Van Der Linden P., 1996, *JUST Java*, Prentice Hall, Englewood Cliffs (NJ).
- Van Oorschot P.C., Wiener M.J., 1988, *A Known-Plaintext Attack on Two-Key Triple Encryption*, Advances in Cryptology-CRYPTO 1988 Proceedings, Springer-Verlag, New York, pp. 119-31.
- Van Renesse R., Van Staveren H., Tanenbaum A.S., 1988, *Performance of the World's Fastest Distributed Operating System*, "Operating Systems Rev.", ottobre, vol. 22, pp. 25-34.
- Varghese G., Lauck T., 1987, *Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility*, Proc. Eleventh Symp. on Operating Systems Prin., ACM, pp. 25-38.
- Venkatasubramanian C., Chiueh T., 1995, *Design, Implementation, and Evaluation of a Software-Based Real-Time Ethernet Protocol*, Proc. SIGCOMM 1995 Conf., ACM, pp. 27-37.
- Vetter R.J., Spell C., Ward C., 1994, *Mosaic and the World-Wide Web*, "IEEE Computer Magazine", ottobre, vol. 27, pp. 49-57.
- Villamizar C., Song C., 1995, *High Performance TCP in ANSNET*, "Computer Commun. Rev.", ottobre, vol. 25, pp. 45-60.
- Viterbi A.J., 1995, *CDMA Principles of Spread Spectrum Communication*, Addison-Wesley, Reading (MA).
- Wada H., Yozawa T., Ohnishi T., Tanaka Y., 1993, *Mobile Computing Environment Based on Internet Packet Forwarding*, Proc. Winter USENIX Conf., USENIX, gennaio, pp. 503-17.
- Walrand J., 1991, *Communication Networks: A First Course*, Irwin, Homewood (IL).
- Watson R.W., 1981, *Timer-Based Mechanisms in Reliable Transport Protocol Connection Management*, "Computer Networks", febbraio, vol. 5, pp. 47-56.
- Wayner P., 1995, *Picking the Crypto Lock*, "Byte", ottobre, pp. 77-80.

- Weisband S.P., Reinig B.A., 1995, *Managing User Perceptions of Email Privacy*, "Commun. of the ACM", dicembre, vol. 38, pp. 40-7.
- Wiener M.J., 1994, *Efficient DES Key Search*, Technical Report TR-244, School of Computer Science, Carleton Univ., Ottawa.
- Williams K.A., Dam T.Q., Du D.H.-C., 1993, *A Media Access Protocol for Time and Wavelength-Division Multiplexed Passive Star Networks*, "IEEE Journal on Selected Areas in Commun.", maggio, vol. 11, pp. 560-7.
- Willinger W., Taqqu M.S., Sherman R., Wilson D.V., 1995, *Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level*, Proc. SIGCOMM 1995 Conf., ACM, pp. 100-13.
- Wolter M.S., 1990, *Fiber Distributed Data Interface-A Tutorial*, "ConneXions", ottobre, pp. 16-26.
- Yang C.-Q., Reddy A.V.S., 1995, *A Taxonomy for Congestion Control Algorithms in Packet Switching Networks*, "IEEE Network Magazine", luglio-agosto, vol. 9, pp. 34-45.
- Yeh Y.-S., Hluchyj M.G., Acampora A.S., 1987, *The Knockout Switch: A Simple, Modular Architecture for High-Performance Packet Switching*, "IEEE Journal on Selected Areas in Commun.", ottobre, vol. 5, pp. 1274-83.
- Youssef A.M., Kalmar E., Benzoni L., 1995, *Technico-Economic Methods of Radio Spectrum Assignment*, "IEEE Commun. Magazine", giugno, vol. 33, pp. 88-94.
- Yuval G., 1979, *How to Swindle Rabin*, "Cryptologia", luglio, vol. 3, pp. 187-90.
- Zhang L., 1988, *Comparison of Two Bridge Routing Approaches*, "IEEE Network Magazine", gennaio-febbraio, vol. 2, pp. 44-8.
- 1993, *RSVP A New Resource ReSerVation Protocol*, "IEEE Network Magazine", settembre-ottobre, vol. 7, pp. 8-18.
- Zimmermann P.R., 1995a, *The Official PGP User's Guide*, MIT Press, Cambridge (MA).
- 1995b, *PGP: Source Code and Internals*, MIT Press, Cambridge (MA).
- Zipf G.K., 1949, *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Addison-Wesley, Cambridge (MA).
- Ziv J., Lempel Z., 1977, *A Universal Algorithm for Sequential Data Compression*, "IEEE Trans. on Information Theory", maggio, vol. IT-23, pp. 337-43.

INDICE ANALITICO

A

- AAL (*vedi* ATM Adaptation Layer)
 AAL 1, 530-31, 730
 AAL 2, 532, 730
 AAL 3/4, 532-35, 730
 AAL 5, 535-36, 730
 AAL, confronto di protocolli, 536-37
 ABR (*vedi* Available Bit Rate)
 Abstract Syntax Notation-1, 613-19
 Abstract Window Toolkit, 696
 Ack, 29
 ACR (*vedi* Actual Cell Rate)
 ACTS (*vedi* Advanced Communication Technology Satellite)
 Actual Cell Rate, 454
 ADC (*vedi* Convertitore analogico digitale)
 ADCCP (*vedi* Advanced Data Communication Control Procedure)
 ADSL (*vedi* Asymmetric Digital Subscriber Line)
 Advanced Communication Technology Satellite, 318
 Advanced Data Communication Control Procedure, 216
 Agente base, 355
 Agente di trasferimento messaggi, 625
 Agente SNMP, 612
 Agente straniero, 355
 Agente utente, 625-30
 Alias, email, 627-28
 Allocazione del canale nelle LAN, 234-35
 ALOHA, 236-40
 puro, 236-39
 satellitare, 316
 slot, a, 239-40
 American National Standards Institute, 67
 AMPS (*vedi* Sistema telefonico mobile avanzato)
 Analisi di Fourier, 75
 Anello delle chiavi private, PGP, 646
 stellato, 283
 ANSI (*vedi* American National Standards Institute)
 ANSNET, 49
 Anycast, 427
 Apocalisse dei due elefanti, 38-39
 Appendice, 40, 310
 Applet, 685-88
 Applicazione di aiuto, 663
 Architettura di rete, 16
 Area, OSPF, 410
 ARP (*vedi* Risoluzione degli indirizzi)
 ARPANET 33, 45-48, 50, 67, 343, 550, 603
 Array di dischi, 725
 ARQ (*vedi* Automatic Repeat Request)
 Articolo di news, esempio, 656
 AS (*vedi* Sistemi Autonomi)
 ASCII, protezione, 634
 ASN.1 (*vedi* Abstract Syntax Notation-1)
 ASN.1, sintassi di trasferimento, 617-19
 Aspetti sociali, 5-6
 relativi alla crittografia, 601-603
 Asymmetric Digital Subscriber Line, 728
 Asynchronous Transfer Mode, 59-63
 categorie di servizi, 442-44
 celle, formato delle, 436-37
 connessioni, creazione delle, 438-40
 controllo della congestione, 451-55
 CS, sottolivello di convergenza, 62
 data link, livello, 225-28
 Forum ATM, 63
 instradamento e commutazione, 440-42
 NNI, 436
 piano del controllo, 61
 piano utente, 61
 PMD, sottolivello, 61
 prospettive, 63
 qualità del servizio, 444-47
 SAR, sottolivello, 62
 secchio bucato, 451

TC, sottolivello, 62
 traffico, normalizzazione e pattugliamento, 447-51
 UNI, 436
 virtual channel, 435
 virtual path, 435
ATM (*vedi* Asynchronous Transfer Mode)
ATM Adaptation Layer, 527-37
ATMARP, server, 456
Attacco del compleanno, 599-601
Attacco riflesso, 585
Attenuazione, 105
 nella fibra, 86
Attraversamento dell'albero, protocollo, 247-49
Audio CD, 701-704
Audio digitale, 701-705
Aurora, 53
Automatic Repeat Request, 194
Autorità di certificazione, 647
Autenticazione, protocolli di, 582-94
 chiave pubblica, 593-94
 Kerberos, 591-93
 usando KDC, 588-601
Available Bit Rate, 443-44
Aziende comuni, 65, 115

B

Backbone, OSPF, 411
Backward learning, algoritmo, 299
Banda base, cavo a, 83, 84, 266, 267, 269
Banda del seccchio, attacco, 587
Banda di frequenza, 92
Banda industriale-scientifica-medica, 96
Base64, 634
Batcher-banyan, commutatore, 146-49
Baud, 78
Bell Operating Company, 102
Bell System, 100
Bellman-Ford routing, 343
BGP (*vedi* Routing tra gateway esterni)
Big endian, macchina, 398
Bigrammi, 564
 Bit di parità, 178
Bit pipe, 135
BITNET, 50
Blanca, 53
Blocchi cifrati, 566, 571, 577
Blocco-P, 568
Blocco-S, 569
Blocco della testa della linea, 145
BOC (*vedi* Bell Operating Company)

BOOTP, 409
Bridge, 293-306
 instradamento da sorgente, 302-303
 remoti, 305-306
 spanning tree (albero di attraversamento), 300-301
 tra LAN IEEE 802, 294-97
 trasparenti, 298-300
Broadband, cavo, 82-84
Broadband ISDN, 58-63, 138-49
Broadcast, indirizzo, 269
Broadcast, reti, 7
Broadcast routing, 357-59
Broadcast/Unknown server, 457
Broadcasting, 7
Browser, World Wide Web, 661
BUS (*vedi* Broadcast/Unknown server)
Busta, email, 626

C

Campo, video, 706
Canale, 11
Canali ad accesso casuale, 233
Canali a fibra, 314-15
Canali coperti, 698
Canali multiaccesso, 233
Care-of-address, 418
Carnegie-Mellon University, 6
CASA, 53
Cavo coassiale, 81-84
 a banda base, 81-82
 a banda larga, 82-84
CBR (*vedi* Constant Bit Rate)
CCITT, 65, 115, 117, 118, 120, 138, 624
CDMA (*vedi* Code Division Multiple Access)
CDPD (*vedi* Cellular Digital Packet Data)
CDV (*vedi* Cell Delay Variation)
Cell Delay Variation, 446
Cell Error Ratio, 446
Cell Loss Ratio, 446
Cell Misinsertion Rate, 447
Cell relay, 59
Cell Transit Delay, 446
Cell Variation Delay Tolerance, 446
Celle
 ATM, 59
 HTML, 679
 oziouse, 226
 radio cellulare, 153
Cellular Digital Packet Data, 14, 258-60
Censura
 da CMU, 6
 da CompuServe, 655

Centro distribuzione chiavi, 588-91
CER (*vedi* Cell Error Ratio)
CGI (*vedi* Common Gateway Interface)
Checksum, 172, 173, 177-84, 225
Chiave crittografica, 562
Chiave di sessione, 583
Chiave segreta, crittografia a, 568-78
Chiave simmetrica, crittografia a, 580
Chip, 261
Chiudere una connessione, 481-85, 512-14
Choke packet, 374
CIDR (*vedi* Routing interdominio senza classe)
Cifrario
 blocco, 570-77
 Cesare, 564-65
 sostituzione, 564-65
 trasposizione, 565-66
Cifratura con modalità output feedback, 573-74
Circuito, 11
Circuito locale, 100, 104-13
 fibra, 111-13
Circuito virtuale, 329-33
 confronto con reti a datagramma, 331-33
Circuito virtuale concatenato, 386-87
Circuito virtuale permanente, 57, 139-40
Client-Server, modello, 3-4
CLR (*vedi* Cell Loss Ratio)
CLUT (*vedi* Color Look Up Table)
CMR (*vedi* Cell Misinsertion Rate)
Code, gestione pesata, 375
Code Division Multiple Access, 260-64, 320
Codec, 105
Codice di correzione errori, 177-79
Codice di ridondanza ciclica, 180
Codice di rilevazione errori, 177, 180-84
Codice polinomiale, 180
Codifica differenziale, 710
Codifica Manchester differenziale, 268-69
Collegamento remoto, 51
Color Look Up Table, 710
Common Gateway Interface, 684-85
Commutatore
 a divisione di spazio, 131-33
 a divisione di tempo, 133-34
 crossbar, 130-31
Commutatori ATM, 142-49
Commutatori di dati, 11
Commutazione di circuito, 125-29
Commutazione di messaggio, 127
Commutazione di pacchetto, 128
 nodo, 12
 sottorete, 12

Commutazione store-and-forward, 127
Commutazione telefonica, 125-34
Compressione dei dati, 708-21
CLUT, 710
 codifica di stringa, 709
 codifica differenziale, 710
 codifica entropica, 709-10
 codifica sorgente, 710-12
Discrete Cosine Transformation, 711
fedele (lossless), 709
infedele (lossy), 709
 quantizzazione vettoriale, 711
 trasformazioni, 710

CompuServe, 655
Concatenazione a blocchi del cifrario, 572
Conferma, 25
Congestione, politiche di prevenzione, 362-64
Connessione iniziale, protocollo, 474
Constant Bit Rate, 443
Conteggio all'infinito, 354
Conteggio binario a ritroso, 245-46
Contesa limitata, protocollo a, 246-49
Controllo del flusso, 176-77, 485-89
Controllo degli errori, 175-76
Controllo della congestione, algoritmi di, 360-81
 ATM, 453-55
 basato sulla velocità, 454-55
 gestione pesata delle code, 374-75
 multicasting, 379-81
 nelle reti basate su circuito virtuale, 372-73
 pacchetti regolatori, 373-76
 principi generali, 362-64
 seccchio bucato, algoritmo, 366-68
 TCP, 519-21
 token bucket, 368-71
Controllo di ammissioni, 372, 451
Convergence sublayer, AAL, 529
Convertitore analogico digitale, 703
Cordless (*vedi* Telefono senza filo)
Core base tree, 360
Corpo, email, 626
Correzione errori, 177-84
Crawler, 698
CRC (*vedi* Codice di ridondanza ciclica)
Create una connessione, 476-81
 TCP, 511-12
Crittoanalisi, 562
 differenziale, 577
 lineare, 577
Crittografia, 561-82
 a chiave pubblica, 578-82
 a chiave segreta, 568-78
 tradizionale, 561-67

Crittologia, 562
 Cromiumanza, 707-708
 Crossposting, 651
 CS sublayer, ATM, 62
 CSMA (*vedi* Multiaccesso con rilevamento della portante)
 CSMA/CD, 242-43
 CSNET, 48
 CTD (*vedi* Cell Transit Delay)
 CVPT (*vedi* Cell Variation Delay Tolerance)

D

Data Circuit-Terminating Equipment, 110
 Data Encryption Standard, 569-77
 concatenato, 571-74
 controversie, 575
 rompere, 574-77
 Data Terminal Equipment, 110
 Datagram, 329-30
 confronto con circuiti virtuali, 331-33
 servizio, 23
 Dati urgenti, 507
 DCE (*vedi* Data Circuit-Terminating Equipment)
 DCS 1800, 255
 Decibel, 79, 703
 Decodifica, 708
 DES (*vedi* Data Encryption Standard)
 Descrittore di traffico, 445
 Diffie-Hellman, scambio di chiavi, 578-80
 Distorsione da ritardo, 105
 Digital Sense Multiple Access, 259-60
 Digital Signature Standard, 597
 Directory server, 475
 Discrete Cosine Transformation, 711
 Dispersione nelle fibre, 87
 Distance Vector Multicast Routing, protocollo, 735
 Distance Vector Routing, 342
 Distanza di Hamming, 178
 Distributed Mail System Protocol, 642
 Distributed Queue Dual Bus, 10, 289-92
 Direttive HTML, 674-81
 DMSP (*vedi* Distributed Mail System Protocol)
 DNS (*vedi* Domain Name System)
 Domain Name System, 406, 603-11
 Dominio, 603
 Doppini, 81
 categoria 3, 81
 categoria 5, 81
 Dorsali, 11, 113-25
 di connessione di pedaggio, 101

interpedaggio, 101
 interufficio, 101
 DQDB (*vedi* Distributed Queue Dual Bus)
 DS, 117
 DSMA (*vedi* Digital Sense Multiple Access)
 DSS (*vedi* Digital Signature Standard)
 DTE (*vedi* Data Terminal Equipment)
 DVMRP (*vedi* Distance Vector Multicast Routing)

E

EARN, 50
 Eco
 cancellatori, 109
 soppressori, 108-109
 Elaborazione veloce dei TPDU, 546-49
 Electronic code book mode, 571
 Electronic mail (*vedi* Email)
 Elefanti, apocalisse, 38-39
 Email, 5, 51, 623-48
 agente trasferimento messaggi, 625
 agente utente, 625, 626-30
 architettura e servizi, 625-26
 busta, 626
 comandi utente, 630
 corpo, 626
 filtri, 642
 formato dei messaggi, 630-37
 formato MIME, 632-37
 funzioni, 625
 gateway, 640-41
 intestazione, 626
 lettura, 628-30
 primi sistemi, 624
 privacy, 643-48
 spedizione, 626-28
 trasferimento dei messaggi, 637-43
 trasmissione finale, 641-43
 Emoticon, 653
 Entità, 21
 Entità di trasporto, 463
 Entità pari, 21
 Entropia, codifica, 709-10
 ER (*vedi* Explicit Rate)
 Ereditarietà, Java, 692
 Esente da collisioni, protocollo, 243-46
 Eserciti, problema dei due, 482-83
 Ethernet, 9, 10, 265 (*vedi anche* IEEE 802.3)
 Ethernet commutata, 274-75
 Explicit rate, 454
 Extended SMTP, 640
 Exterior gateway protocol, 391, 409, 414-16

F
 Fabry-Perot, interferometro, 88, 250
 FAQ (*vedi* Frequently Asked Question)
 Farm di dischi, 725
 Fast Ethernet, 309-12
 FCC (*vedi* Federal Communications Commission)
 FDDI (*vedi* Fiber Distributed Data Interface)
 FDM (*vedi* Frequency Division Multiplexing)
 Federal Communications Commission, 93, 97, 160
 Fiber Distributed Data Interface, 306-309
 Fiber To The Curb, 111-13, 116, 728-29
 Fiber To The Home, 111-13, 729
 fibra multimodo, 85
 fibre ottiche, 84-91
 attenuazione, 86
 confronto con rame, 90-91
 confronto con satellite, 162-63
 dispersione, 87
 FDDI, 306-309
 modo singolo, 85
 multimodo, 85
 principi base, 84-86
 SONET, 120-25
 WDM, 115-16
 File server, 3
 File Transfer Protocol, 672
 Filo di rame, confrontato con fibre ottiche, 90-91
 Filtro per pacchetti, 396
 Finestra di congestione, 519
 Finestra di ricezione, 196
 Finestra di trasmissione, 196
 Firewall, 394-97
 Firma digitale, 594-601
 a chiave pubblica, 596-97
 a chiave segreta, 595-96
 Flamewar, 651-52
 Flooding, 338-40
 selettivo, 340
 Foglio di stile, 677
 Ford-Fulkerson routing, 343
 Form, HTML, 681-85
 Fornitore di servizi, 21
 Fornitore di servizi di trasporto, 465
 Fotogrammi F, MPEG, 719
 Fotogrammi I, MPEG, 717
 Fotogrammi P, MPEG, 717
 Frame
 ack, 29
 data, 29
 video, 705

G
 Gateway, 15
 di applicazione, 384, 397
 di trasporto, 384
 GCRA (*vedi* Generic Cell Rate Algorithm)
 Generatore polinomiale, 181
 Generic Cell Rate Algorithm, 447-51
 Gerarchia di commutazione, telefono, 129-30
 Gerarchia di memorizzazione, 723
 Gerarchia di protocolli, 16-19
 Gerarchia in USENET, 650
 Gigabit, reti di, 52-54, 549-54
 Global System for Mobile Communications, 255-58
 Go back n, protocollo, 200-204
 Gopher, 673
 Gruppo, 115
 GSM (*vedi* Global System for Mobile Communications)

H
 Half duplex, 20
 Half-gateway, 384
 Hamming, distanza, 178
 Handoff, 153
 HDLC (*vedi* High-Level Data Link Control)
 HDTV (*vedi* High Definition TeleVision)
 head-end, cavo, 83
 header (*vedi* Intestazione)
 Header Error Control, ATM, 225
 HEC (*vedi* Header Error Control, ATM)
 HEPNET, 50
 HFC (*vedi* Hybrid Fiber Coax)
 High Definition TeleVision, 707
 High-Level Data Link Control, 216-19
 High Performance Parallel Interface, 312-14
 HIPPI (*vedi* High Performance Parallel Interface)
 Host, 11

Host-to-network, livello, 36
HTML (*vedi* HyperText Markup Language)
HTTP (*vedi* HyperText Transfer Protocol)
Hub, satellite, 159
Hybrid Fiber Coax, 729
HyperText Markup Language, 671-85
 form (moduli), 681-85
 versioni, 679-81
HyperText Transfer Protocol, 668-70

I

IAB (*vedi* Internet Architecture Board)
IBM, 41, 216, 302, 569, 574-76
ICMP (*vedi* Internet Control Message Protocol)
IDEA (*vedi* International Data Encryption Algorithm)
IDU (*vedi* Interface Data Unit)
IEEE, 67
IEEE 802, 264-93
 confronto tra LAN, 288-89
IEEE 802.2, 292-93
IEEE 802.3, 9, 265-75
 cablaggio, 265-68
 codifica dei segnali, 268-69
 commutata, 274-75
 fast Ethernet, 309-12
 formato dei frame, 269
 prestazioni, 272-74
 protocollo, 269-71
IEEE 802.3u, 309-12
IEEE 802.4, 275-81
 manutenzione dell'anello, 279-81
 protocollo, 277-79
IEEE 802.5, 9, 281-88
 manutenzione dell'anello, 284-86
 protocollo, 286-88
IEEE 802.6, 10, 289-92
IETF (*vedi* Internet Engineering Task Force)
IGMP (*vedi* Internet Group Management Protocol)
IMAP (*vedi* Interactive Mail Access Protocol)
Immagini, video, 705
IMP (*vedi* Interface Message Processor)
Impacchettamento, 172
Improved Mobile Telephone System, 153
IMTS (*vedi* Improved Mobile Telephone System)
Incapsulazione, Java, 691
Indication, 23-26
Indirizzamento, 472-76
Indirizzamento piatto, 476

Indirizzi broadcast, 269
 Indirizzi multicast, 269
 Indirizzi gerarchici, 475
 Indirizzi IP, 406-409
 Indirizzo, 474
 di trasporto, 472
 Information frame, 216
 Infrarossi, trasmissione a, 96-97
 Instradamento da sorgente, 302
Integrated Services Digital Network, 59, 135-38
Interactive Mail Access Protocol, 641
 Interallacciamento video, 706
Inter Exchange Carrier, 103-104
 Interfaccia parallela ad alte prestazioni, 312-14
 Interfaccia tra livelli, 16
Interface Data Unit, 22
Interferometro,
 Fabry-Perot, 88, 250
 Mach-Zehnder, 88, 250
Interface Message Processor, 45
 Interior gateway protocol, 391, 409-14
International Data Encryption Algorithm, 577-78
International Standard Organization, 66
International Telecommunication Union, 65, 438, 455, 535, 615, 647, 712
Internet, 15
 CIDR, 419-22
 data link, 219-25
 gestione delle connessioni, 511-14
 IP, 34, 398-401
 IP mobile, 417-19
 IPv6, 422-34
 livello internet, 34, 397-434
 multicasting, 416-17
 protocolli di routing, 409-16
 storia, 50-52
 TCP, 35, 504-25
Internet, applicazioni
 email, 623-48
 MBone, 733-36
 net news, 648-59
 World Wide Web, 659-702
Internet Architecture Board, 68
Internet Control Message Protocol, 405-406
Internet Engineering Task Force, 68
Internet Group Management Protocol, 417, 735
Internet Policy Registration Authority, 647
Internet, protocolli
 ARP, 406-408
 BGP, 414-16

DVMRP, 735
HTTP, 668-70
ICMP, 405-406, 735
IGMP, 417
 IP, 34, 398-401
NNTP, 657-59
OSPF, 409-14
PIM, 736
PPP, 221-25, 664
RARP, 408-409
RSVP, 379-81
SLIP, 220-21, 664
SMTP, 637-40
TCP, 35, 504-24, 637, 659, 665
UDP, 35, 525-27
Internet provider, 219
Internet Society, 51, 68
Internetwork, 8, 15
Internetwork routing, 390-91
Internetworking, 381-97
 orientato alla connessione, 386-87
 perché è necessario, 385-86
 senza connessioni, 387-89
Interrogazione ricorsiva, 610-11
Intestazione, 18
 email, 626
 frame, 185
 news, 655-56
Intruso, 562
IP (*vedi* Internet, protocolli)
Ipermediale, 663
Iperpuntatore, 661
Iperfesto, 661
IPRA (*vedi* Internet Policy Registration Authority)
IPv4, 398-401
IPv5, 423
IPv6, 422-34
 controversie, 432-34
 indirizzi, 426-27
 jumbogram, 430
 preamboli di estensione, 428-32
 preambolo principale, 424-28
IPX, 43
IS-IS routing, 352
ISDN (*vedi* Integrated Services Digital Network)
ISDN a banda stretta, 134-38
ISM band (*vedi* Banda industriale-scientifica-medica)
ISO (*vedi* International Standard Organization)
ITU (*vedi* International Telecommunication Union)

ITU-R, 65
ITU-T, 65-66
IXC (*vedi* Inter Exchange Carrier)

J

Java, 685-98
 Abstract Window Toolkit, 696
API, 694-96
 classe, 691-92
 descrizione linguaggio, 688-96
 oggetti, 690-94
 polimorfismo, 694
 sicurezza, 696-98
Jitter, 372
Jitter, controllo del, 378-79
JPEG, 712-15
Jumbogram, 430

K

Karn, algoritmo di, 524
KDC (*vedi* Centro distribuzione chiavi)
Keepalive timer, TCP, 524
Kerberos, 591-93
Killfile, 652
Knockout, commutatore, 145
Knowbot, 698

L

LAN (*vedi* Local Area Network)
LAN Emulation Server, 455
LAN ad alte velocità, 306-15
LAN ATM, 455-57
LAN volante, 15
LAP (*vedi* Link Access Procedure)
LATA (*vedi* Local Access and Transport Area)
LCP (*vedi* Link Control Protocol)
LEC (*vedi* Local Exchange Carrier)
LES (*vedi* LAN Emulation Server)
Linea seriale IP, protocollo, 220-21, 664
Linea SONET, 121
Linea voice-grade, 78
Link Access Procedure, 216
Link Control Protocol, 221-25
Link Encryption, 561
Link state routing, 346-52
LIS (*vedi* Logical IP Subnet)
Little endian computer, 398
Livello, 16
 applicazione, 32, 35, 559-742
 data link, 28, 169-232

fisico, 27, 75-168
 presentazione, 31-32
 rete, 29, 34
 sessione, 31
 trasporto, 30, 34-35, 463-558
 Livello AAL, 61, 527-37
 Livello applicazione, 32, 35, 559-742
 Domain Name System, 603-11
 email, 623-48
 multimediale, 702-36
 network management, 611-23
 notizie di rete, 648-59
 sicurezza in rete, 559-603
 World Wide Web, 659-702
 Livello ATM, 434-57, 61
 Livello data link, 169-231
 ATM, 225-28
 controllo degli errori, 175-76
 controllo del flusso, 176
 esempi di protocolli, 216-28
 HDLC, 216-19
 impacchettamento (framing), 172-75
 LLC, 264, 292-93
 OSI, 28-29
 principi di progettazione, 169-77
 protocolli elementari, 184-94
 riempimento di bit, 174
 riempimento di caratteri, 174
 Livello di presentazione, 31-32
 Livello di rete, 327-462
 ATM, 434-57
 caratteristiche di progetto, 327-33
 controllo della congestione, 360-81
 Internet, 397-434
 internetworking, 381-97
 organizzazione interna, 329-31
 OSI, 29
 routing, algoritmi, 333-60
 servizi offerti, 327-29
 Livello di sessione, OSI, 31
 Livello fisico, 75-168
 comunicazioni satellitari, 157-63
 mezzi di trasmissione, 80-91
 OSI, 27-28
 radio cellulare, 149-57
 sistema telefonico, 98-134
 trasmissione senza filo, 91-98
 Livello host-to-network, 36
 Livello trasporto, 463-558
 AAL, ATM, 527-38
 elementi del protocollo, 471-93
 esempio, 493-504
 Internet, 504-27
 OSI, 30

prestazioni, 538-54
 servizi offerti, 463-71
 LLC (*vedi* Logical Link Control)
 Load shedding, 376
 Local Access and Transport Area, 103
 Local Area Network, 8-10
 allocazione del canale, 234-36
 alte velocità, 306-15
 ATM, 455-57
 confronto fra LAN, 288-89
 Ethernet, 10, 265
 fast Ethernet, 309-12
 IEEE 802, 264-93
 token bus, 275-81
 token ring, 281-88
 Local Exchange Carrier, 103
 Logical IP Subnet, 456
 Logical Link Control, 264, 292-93
 Lotteria cinese, 575
 Luminanza, 707
 Luminiferous ether, 265
 Lunghezza d'onda, 91

M

MAC sublayer (*vedi* Sottolivello di accesso al mezzo)
 MACA (*vedi* Multiple Access with Collision Avoidance)
 MACAW, 254
 Macchine a flusso di dati, 7
 Macchine a stati finiti, 209-14, 503-504
 Mach-Zehnder, interferometro, 88, 250
 Macrobloccchi, 718
 Mailbox, 625
 Mailing list, 625
 Mailto, 672
 MAN (*vedi* Metropolitan Area Network)
 Management Information Base, 613, 621-22
 Manchester, codifica, 268-69
 Mappa di bit, protocollo a, 244-45
 Mappe attive, 663
 Markup language, 674
 Maschera di sottorete, 404
 Maximum Transfer Unit, 507
 MBone (*vedi* Multicast Backbone)
 MCR (*vedi* Minimum Cell Rate)
 MD5, 598, 643
 Media continui, 702
 Meet-in-the-middle, attacco, 575
 Message digest, 598
 Messaggio di credito, 502-503
 Metodo, HTTP, 669
 Metropolitan Area Network, 10

Mezzi di trasmissione, 80-91
 Mezzo fisico, 16
 MFJ (*vedi* Modified Final Judgement)
 MIB (*vedi* Management Information Base)
 MIDI (*vedi* Music Instrument Digital Interface)
 Midsplit, cavo coassiale, 83
 MILNET, 48
 MIME (*vedi* Multipurpose Internet Mail Extensions)
 Minimum Cell Rate, 446
 Mobile Switching Center, 153
 Mobile Telephon Switching Office, 153
 Modalità output feedback, 573-74
 Modalità promiscue, 294
 Modelli di riferimento, 27-41
 B-ISDN, 60-63
 confronto tra OSI e TCP/IP, 36-38
 OSI, 27-33
 TCP/IP, 33-36
 Modem, 106-109
 Modem nullo, 111
 Modified Final Judgement, 103
 Modulazione, 106-109
 ampiezza, 107
 fase, 107
 frequenza, 107
 Modulazione delta, 119
 Motore di ricerca, WWW, 698-702
 Mosaic, 659
 MOSPF (*vedi* Multicast OSPF)
 MOTIS, 624
 MPEG, standard, 715-21, 729-30
 Fotogrammi B, 718
 Fotogrammi I, 717
 Fotogrammi P, 717
 macrobloccchi, 718
 MPEG-1, 716-19
 MPEG-2, 719-21
 profili, 720
 streams, 721
 MRouter (*vedi* Multicast router)
 MSO (*vedi* Mobile Switching Center)
 MTSO (*vedi* Mobile Telephon Switching Office)
 MTU (*vedi* Maximum Transfer Unit)
 Multiaccesso con rilevamento della portante, protocollo, 240-43
 Multicast Backbone, 733-37
 Multicast OSPF, 736
 Multicast router, 734
 Multicast routing, 359-360
 Multicasting, 7, 359, 379-81
 Internet, 416-17

N

N-ISDN (*vedi* Narrowband ISDN)
 Nagle, algoritmo di, 517-18
 NAK (*vedi* Negative Acknowledgement)
 Name Server, 475, 609-11
 NAP (*vedi* Network Access Point)
 Narrowband ISDN, 134-38
 National Institute of Standards and Technology, 67
 National Security Agency, 574
 National Television Standards Committee, 707
 NCP (*vedi* Network Control Protocol)
 NCP (*vedi* Network Core Protocol)
 Nectar, 54
 Needham-Schoeder, protocollo, 589-90
 Negative acknowledgement, 208
 Negoziazione, 24-25
 Negoziazione delle opzioni, 467
 NETBLT, 553
 NetWare, Novell, 43-44
 Network Access Point, 50
 Network Control Protocol, 221-25
 Network Core Protocol, 44
 Network Information Center, 402
 Network News Transfer Protocol, 657-59
 Network Service Access Point, 472
 News, 51, 648-59, 672, 673
 Newsfeed, 655
 Newsgroup,
 creazione, 654
 esempi, 651
 NIC (*vedi* Network Information Center)

NIST (*vedi* National Institute of Standards and Technology)
 NNTP (*vedi* Network News Transfer Protocol)
 Nonce, 589
 Notazione decimale a punti, 402
 Novel Netware, 43-44
 NREN, 53
 NSA (*vedi* National Security Agency)
 NSAP (*vedi* Network Service Access Point)
 NSFNET, 48-50
 NT1, 135-37
 NT2, 135-37
 NTSC (*vedi* National Television Standards Committee)
 Nyquist, limite, 78-79

O

OAM (*vedi* Operation and Maintenance cell)
 OC-n (*vedi* Optical Carrier)
 Oggetto,
 Java, 690-94
 SNMP, 612, 622-23,
 One-bit sliding window, protocollo, 196
 One-time pad, 566
 ONU (*vedi* Optical Network Unit)
 Open Shortest Path First, 409-14
 Operation and Maintenance cell, 226
 Optical carrier, 123-24
 Optical Network Unit, 728-29
 Oryctolagus cuniculus, 17
 OSI, modello di riferimento, 27-33
 confronto con TCP/IP, 36-38
 critiche, 38-42
 OSPF (*vedi* Open Shortest Path First)
 Otway-Rees, protocollo, 590-91

P

Pacchetto, 7
 Package, Java, 690
 Packet Assembler Disassembler, 57
 Packet Elementary Stream, 720
 PAD (*vedi* Packet Assembler Disassembler)
 Pagina Web, 661, 662, 676
 Paging, 150
 PAL (*vedi* Phase Alternating Line)
 Pari, 16
 PBX (*vedi* Private Branch eXchange)
 PCA (*vedi* Policy Certification Authority)
 PCM (*vedi* Pulse Code Modulation)
 Percorso, SONET, 121
 PCN (*vedi* Personal Communication Network)

PCR (*vedi* Peak Cell Rate)
 PCS (*vedi* Personal Communication Services)
 PDU (*vedi* Protocol Data Unit)
 Peak Cell Rate, 445-46
 Peer (*vedi* Pari)
 PEM (*vedi* Private Enhanced Mail)
 Permanent virtual circuit, 57, 139-40
 Personal Communication Network, 156-57
 Personal Communication Services, 156-57
 PES (*vedi* Packet Elementary Stream)
 PGP (*vedi* Pretty Good Privacy)
 Phase Alternating Line, 707
 Piano di controllo, ATM, 63
 Piano utente, ATM, 63
 Piggybacking, 195
 Pila di protocolli, 17
 PIM (*vedi* Protocol Independent Multicast)
 Pipelining, 200
 Pixel, 707
 Plain Old Telephone Service, 137
 PMD, sottolivello ATM, 61, 142
 Point of Presence, 104
 Policy Certification Authority, 647
 Polimorfismo, Java, 694
 Politica, 41
 Politica del latte, 376
 Politica del vino, 376
 Polling, 316
 Polling orientato al trap, 613
 POP (*vedi* Point of Presence)
 POP3 (*vedi* Post Office Protocol-3)
 Porta, TCP, 506
 ben nota, 506
 Portante T1, 117-19
 Portante T2, 119
 Portante T3, 119
 Portante T4, 119
 Post Office Protocol-3, 641
 Post, Telegraph and Telephone Administration, 65
 Posta elettronica (*vedi* Email)
 POTS (*vedi* Plain Old Telephone Service)
 PPP (*vedi* Protocollo Punto-a-Punto)
 Predizione del preambolo, 659
 Prestazioni della rete, 538-54
 IEEE 802.3, 272-74
 Pretty Good Privacy, 643-46
 confrontato con PEM, 647-48
 Primitive dei servizi, 23-26
 Primitive del servizio di trasporto, 467-71
 Principio di ottimalità, 335-36
 Privacy Enhanced Mail, 646-47
 confrontato con PGP, 647-48

Private Branch eXchange, 135
 Problema della stazione esposta, 253
 Problema della stazione nascosta, 252
 Prodotto banda-ritardo, 540
 Profilo utente, 628
 Protocol Data Unit, 22
 Protocol Independent Multicast, 736
 Protocolli di accesso multiplo, 236-64
 Protocollo/i, 16
 1-bit, 196
 AAL, 536-37
 accesso multiplo, 236-64
 ADCCP, 226
 ARP, 406-408
 ARP gratuito, 418
 ARQ, 192-94
 attraversamento dell'albero, 247-49
 autenticazione, 582-94
 BGP, 414-16
 BOOTP, 409
 canale disturbato, 191-94
 conteggio binario, 245-46
 contesa limitata, 246-49
 CSMA, 240-43
 data link elementari, 184-94
 DMSP, 642
 DSMA, 259-60
 DVMPR, 735
 esente da collisioni, 243-46
 exterior gateway, 391
 FTP, 672
 Go back n, 200-204
 HDL, 216-19
 HTTP, 668-70
 ICMP, 405-406
 IEEE 802.3, 269-71
 IEEE 802.4, 277-79
 IEEE 802.5, 286-88
 IGMP, 417
 IMAP, 641
 interior gateway, 391
 IP, 34, 398-401
 IPX, 43
 MACA, 253-54
 MACAW, 254
 LAN senza filo, 252-55
 LAP, 216
 LAPB, 216
 LCP, 221-25
 Mappa di bit 244-45
 NCP (Network Control Protocol), 221-25
 NCP (Network Core Protocol), 44
 Needham-Schoeder, 589-90
 NNTP, 657-59

OSPF, 409-14
 Otway-Rees, 590-91
 PAR, 192-94
 PIM, 736
 POP-3, 641
 PPP, 221-25, 664
 Q.2391, 438
 RARP, 408-409
 reti di gigabit, 549-54
 ripetizione selettiva, 204-209
 RSVP, 379-81
 SDLC, 216
 sfida-risposta, 583-86
 simplex non limitato, 188
 sliding window, 196-99
 SLIP, 220-21, 664
 SMTP, 637-40
 SNMP, 622-23
 SSCOP, 537
 TCP, 35, 504-24, 637, 659, 665
 UDP, 35, 525-27
 WDMA, 249-51
 Protocolli di trasporto in Internet, 504-27
 Protocollo Punto-a-Punto, 221-25, 664
 Provider (*vedi* Fornitore di servizi)
 Proxy ARP, 408
 PSTN (*vedi* Public Switched Telephone Network)
 PTT (*vedi* Post, Telegraph and Telephone Administration)
 Public Switched Telephone Network, 98
 Pubblicazione multipla (*vedi* Crossposting)
 Pubblicità, IP mobile, 418
 Pulse Code Modulation, 117
 Punti di incrocio, 130
 Punti di riferimento, ISDN, 135
 Push to talk, 157
 PVC (*vedi* Permanent virtual circuit)

Q

Q.2391, 438
 QAM (*vedi* Quadrature Amplitude Modulation)
 QoS (*vedi* Quality of Service)
 Quadrature Amplitude Modulation, 107
 Quality of Service, 22, 444-47, 465-67
 ATM, 444-47
 Quoted-printable encoding, 634

R

Radio cellulare, 149-57
 digitale, 255-64

- RAID (*vedi* Redundant Array of Inexpensive Disks)
 Rana dalla bocca larga, protocollo, 588
 Rapporto segnale-rumore, 78
 RARP (*vedi* Risoluzione inversa degli indirizzi)
 Record di autorità, 610
 Redundant Array of Inexpensive Disks, 725
 Regressione binaria esponenziale, algoritmo, 271-72
 Remailer anonimi, 654
 Reply attack, 589
 Request for Comment, 68
 Resource Management cell, 454
 Resource Record, 605
 Resource reSerVation Protocol, 379-81
 Rete distributiva, 727-30
 Rete multiaccesso, 410
 Rete telefonica pubblica commutata, 98
 Reti ATM, 138-49
 Reti di calcolatori, 2
 uso, 3-7
 Reti di fibre ottiche, 88-90
 Reti di Petri, modello, 214-15
 Reti di transito, 415
 Reti point-to-point, 7
 Reti senza filo, 13-14
 host mobili, 354-57, 417-19
 LAN senza filo, 252-55
 onde elettromagnetiche, 91-98
 radio analogica, 149-56
 radio digitale, 255-64
 TCP senza filo, 525-27
 Reverse path forwarding, 358
 RFC (*vedi anche* Request for Comment)
 RFC 768, 525
 RFC 792, 406
 RFC 793, 504
 RFC 821, 624, 630, 640, 738
 RFC 822, 624, 630-32, 636, 641, 644, 647, 655, 656, 666, 668, 669, 670, 737
 RFC 826, 407
 RFC 903, 409
 RFC 951, 409
 RFC 977, 657
 RFC 1028, 611
 RFC 1034, 603
 RFC 1035, 603
 RFC 1036, 655
 RFC 1048, 409
 RFC 1055, 220
 RFC 1056, 642
 RFC 1064, 641
 RFC 1067, 611
 Riempimento di bit, 174
 Riempimento di caratteri, 174
 Riflessometria a dominio di tempo, 266
 Rilevazione errori, 177-84
 Ripetitori attivi, 89
 Ripetizione selettiva, 201
 Ripetizione selettiva, protocollo, 204-209
 Ripetitore 89, 383
 Ripristino dai guasti, 491-93
 RFC 1084, 409
 RFC 1106, 511
 RFC 1112, 417
 RFC 1122, 504
 RFC 1144, 220
 RFC 1155, 611
 RFC 1157, 611
 RFC 1213, 622
 RFC 1225, 641
 RFC 1247, 409
 RFC 1268, 416
 RFC 1323, 504, 511
 RFC 1421, 647
 RFC 1422, 647
 RFC 1423, 647
 RFC 1424, 647
 RFC 1425, 640
 RFC 1441, 611
 RFC 1442, 611, 619
 RFC 1443, 611
 RFC 1444, 611
 RFC 1445, 611
 RFC 1446, 611
 RFC 1447, 611
 RFC 1448, 611, 623
 RFC 1449, 611
 RFC 1450, 611
 RFC 1451, 611
 RFC 1452, 611
 RFC 1483, 455, 536
 RFC 1519, 420
 RFC 1521, 633, 634, 635
 RFC 1550, 423
 RFC 1577, 329, 456, 536
 RFC 1654, 416
 RFC 1661, 221, 224
 RFC 1662, 221
 RFC 1663, 221, 222
 RFC 1700, 400, 506
 RFC 1715, 428
 RFC 1883, 423
 RFC 1884, 423
 RFC 1885, 423
 RFC 1886, 423
 RFC 1887, 423

- Risoluzione degli indirizzi, protocollo, 406-408
 Risoluzione inversa degli indirizzi, protocollo, 408-409
 RM cell (*vedi* Resource Management cell)
 Router designato, 413
 Router multiprotocollo, 383
 Routing, algoritmi, 333-60
 adattivo, 335
 basato su vettori di distanza, 342
 basato sul flusso, 340-42
 broadcast, 357-59
 flooding, 338-40
 gerarchico, 352-54
 interdominio senza classe, 419-22
 internetwork, 390-94
 link state, 346-52
 lungo cammino minimo, 336-38
 multicast, 359-60
 non adattivo, 335
 per host mobili, 354-57
 reverse path forwarding, 358
 Routing di sessione, 334
 Routing multidestinazione, 357
 Routing tra gateway esterni, protocollo, 414-16
 RS-232, 110
 RS-422-A, 111
 RS-423-A, 111
 RS-449, 111
 RSA, algoritmo, 580-81
 RSVP (*vedi* Resource reSerVation Protocol)
 Ruota dei tempi, 550
 Rumore, 105
 Rumore di quantizzazione, 704
- S**
- SABME (*vedi* Set Asynchronous Balanced Mode Extended)
 SAP (*vedi* Service Access Point)
 SAP (*vedi* Service Advertising Protocol)
 SAR sublayer, ATM, 62, 529
 Satelliti, reti, 157-63
 a orbita bassa, 160-62
 confronto con le fibre, 162-63
 geosincroni, 158-60
 per comunicazioni, 157-63
 Scatole set-top, 730-32
 Schedulazione virtuale, algoritmo, 449
 Schemi a costellazione, 107
 SCR (*vedi* Sustained Cell Rate)
 SDH (*vedi* Synchronous Digital Hierarchy)
 SDLC (*vedi* Synchronous Data Link Control)
 SDU (*vedi* Service Data Unit)
- SEAL (*vedi* Simple Efficient Adaptation Layer)
 SECAM (*vedi* SEquentiel Couleur Avec Mémoire)
 SECBR (*vedi* Severely-Errored Cell Block Ratio)
 Secchio bucato, algoritmo, 366-71
 Secure Hash Algorithm, 599
 Segmento, TCP, 508
 Segnalazione a segnale comune, 118
 Segnalazione associata al canale, 118
 Segnalazione in banda, 109
 SEquentiel Couleur Avec Memoire, 707
 Sequenza di chip, 261
 Server proxy, 668
 Service Access Point, 21
 Service Advertising Protocol, 44
 Service Data Unit, 22
 Service Specific Connection-Oriented Protocol, 537
 Servizi confermati, 25
 Servizi di comunicazione personale, 156-57
 Servizi non confermati, 25
 Servizio
 datagramma, 23
 orientato alla connessione, 22-23
 privo di connessione, 22-23
 richiesta-risposta, 23
 Set Asynchronous Balanced Mode Extended, 218
 Set Normal Response Mode Extended, 218
 Severely-Errored Cell Block Ratio, 447
 Sezione, SONET, 121
 SGML (*vedi* Standard Generalized Markup Language)
 SHA (*vedi* Secure Hash Algorithm)
 Shannon, limite di, 78
 Shell account, 219
 Sicurezza,
 in rete, 559-603
 Java, 696-98
 telefono cellulare, 155
 Silly window syndrome, 517
 Simple Efficient Adaptation Layer, 535-36
 Simple Internet Protocol Plus, 423
 Simple Mail Transfer Protocol, 637-40
 Simple Network Management Protocol, 611-23
 Simplex, comunicazione, 20
 Sincronizzazione, 31
 Sintesi di messaggio, 598
 SIPP (*vedi* Simple Internet Protocol Plus)
 Sistemi autonomi, 391, 397
 Sistemi di contesa, 236-37, 241-47

Sistemi di rintracciamento, 150-51
 Sistema distribuito, 2
 Sistemi premi-per-parlare, 152
 Sistema intermedio, 11
 Sistema telefonico mobile avanzato, 152-53
 Sistema terminale, 10
 Sliding window ad un solo bit, protocollo, 196
 SLIP (*vedi* Linea seriale IP, protocollo)
 Slow start, algoritmo, 521
 SMDS (*vedi* Switched Multimegabit Data Service)
 SMI (*vedi* Struttura delle informazioni di gestione)
 Smiley, 653
 SMTP (*vedi* Simple Mail Transfer Protocol)
 SNA (*vedi* System Network Architecture)
 SNMP (*vedi* Simple Network Management Protocol)
 SNMP, agente, 612
 SNRME (*vedi* Set Normal Response Mode Extended)
 Società A-side, 154
 Società B-side, 154
 Società wireline, 154
 Socket, 469-71
 Software delle reti, 16-27
 Soglie di congestione, 521
 Solitone, 87
 SONET (*vedi* Synchronous Optical NET-work)
 Sottoclassi, Java, 692
 Sottolivello della sezione, SONET, 125
 Sottolivello di accesso al mezzo, 233-236
 Sottorete, 11
 Internet, 403-404
 Sottoreti di comunicazione, 11
 Sovraccaricamento, Java, 694
 SPADE, 317
 SPAN, 50
 Spanning tree, 357
 SPE (*vedi* Synchronous Payload Envelope)
 Specifiche di flusso, 371-73
 Spettro elettromagnetico, 91-94
 Spettro diffuso, 93
 a sequenza diretta, 94
 Spider, 698
 Split horizon, algoritmo, 345-46
 Spot beam, 159
 SPX, 44
 SSCOP (*vedi* Service Specific Connection-Oriented Protocol)
 Stallo, 212
 Standard de facto, 64

Standard de jure, 64
 Standard Generalized Markup Language, 674
 Standard internazionali, 66
 Standard ISO
 ISO 3166, 604
 ISO 8802, 67
 ISO 8859-1, 675
 Standardizzazione dei video a richiesta, 732-33
 Standardizzazione delle reti, 64-69
 Stazione di gestione, 612
 Stazione di riferimento, 316
 Stazione nascosta, 252
 Stella passiva, 89
 Store-and-forward, sottorete, 12
 Store-and-forward, commutazione, 127
 Stream di programma, 721
 Stream di trasporto, 721
 Striping (*vedi* Array di dischi)
 Struttura delle informazioni di gestione, 619-21
 Struttura di commutazione, 142
 STS-1 (*vedi* Synchronous Transport Signal-1)
 Stub, rete, 415
 Style sheet, 677
 Subsplit, cavo coassiale, 83
 Sustained Cell Rate, 336
 Switched Multimegabit Data Service, 54-57
 Switched Virtual Circuit (*vedi* Circuito virtuale commutato)
 Synchronous Data Link Control, 216
 Synchronous Digital Hierarchy, 120-25
 Synchronous Optical NETwork, 120-25
 Synchronous Payload Envelope, 121-22
 Synchronous Transport Signal-1, 121
 System Network Architecture, 39

T

Tag, HTML, 675-81
 Tandem, uffici, 101
 Tariffario, 65
 Tasso base, ISDN, 137, 138
 TC sublayer, 62, 225-28
 TCP (*vedi* Transmission Control Protocol)
 TCP indiretto, 526
 TCP/IP, modello di riferimento, 33-36
 confronto con OSI, 36-38
 TDM (*vedi* Time Division Multiplexing)
 Telecomunicazioni e standardizzazione, 64-66
 Telefonia,
 circuiti locali, 104-13
 commutazione, 125-34

dorsali e multiplexing, 113-25
 politica, 102-104
 portante T1, 117-19
 SONET, 120-25
 Telefono cellulare, 151-56
 AMPS, 152-53
 analogico, 151-52
 digitale, 156
 gestione della chiamata, 154-55
 sicurezza, 155
 Telefono senza filo, 150-51
 Televisione
 analogica, 705-707
 digitale, 707-708
 Telnet, 666-69, 672, 673
 Tempesta di broadcast, 539
 Tempo di possesso del token, 284
 Terminal Interface Processor, 46
 Terminali virtuale di rete, 32
 Testo cifrato, 562
 Testo cifrato semplice, attacco, 563
 Testo in chiaro conosciuto, attacco, 563
 Testo in chiaro selezionato, attacco, 563
 Thin Ethernet, 266
 Three-way handshake, 480
 Time Division Multiplexing, 114, 116-20, 318-20
 Timer, token, 309
 Timer di persistenza, TCP, 524
 Timer di ritrasmissione, TCP, 521-24
 TIP (*vedi* Terminal Interface Processor)
 Token, 275-76
 Token, gestione, 31
 Token Bucket, algoritmo, 368-71
 Token bus, LAN (*vedi* IEEE 802.4)
 Token ring, LAN, 281-88
 Torn tape office, 128
 TPDU (*vedi* Transport Protocol Data Unit)
 Traffic policing, 366
 Traffic shaping, 365-66, 451-55
 Transceiver, 266
 Transmission Control Protocol, 35, 504-24, 637, 659, 665
 controllo di congestione, 519-21
 gestione dei timer, 521-25
 gestione delle connessioni, 511-14
 Karn, algoritmo di, 524
 modello del servizio, 506-507
 Nagle, algoritmo di, 517-18
 politica di trasmissione,
 preamble del segmento, 508-11
 senza filo, 525-27
 silly window syndrome, 517
 Transponditori, 157

Transport Protocol Data Unit, 468
 Transport Service Access Point, 472-75
 Trap, SNMP, 613
 Trasformazioni, 710-11
 Trasmissione ad onde luminose, 97-98
 Trasmissione ad onde infrarosse, 96-97
 Trasmissione ad onde millimetriche, 96-97
 Trasmissione a microonde, 95-96
 Trasmissione bilanciata, 111
 Trasmissione radio, 94-95
 Trasmissione sbilanciata, 111
 Trasporto, protocollo, 471-72
 controllo di flusso, 485-89
 elementi, 471-93
 indirizzamento, 472-76
 Internet, 504-27
 multiplexing, 489-91
 Trigrammi, 564
 Tripla X, 58
 TSAP (*vedi* Transport Service Access Point)
 Tunneling, 389-91
 TV via cavo, 83-84, 103, 138, 165

U

UBR (*vedi* Unspecified Bit Rate service)
 UDP (*vedi* User Datagram Protocol)
 Ufficio centrale, 100
 Ufficio centrale locale, 100
 Ufficio di pedaggio, 101
 Ufficio tandem, 101
 Ufficio terminale, 100
 Uniform Resource Locator, 671-74
 Universal Resource Identifier, 674
 Unnumbered, frame, 217
 Unshielded Twisted Pair, 81
 Unspecified Bit Rate service, 444
 Uomo nel mezzo, attacco, 587
 URI (*vedi* Universal Resource Identifier)
 URL (*vedi* Uniform Resource Locator)
 USENET, 648-59, 670
 punto di vista utente, 649-54
 realizzazione, 654-59
 relazioni con Internet, 649

User Datagram Protocol, 35, 525-27
 Utente di servizi, 21
 Utilizzatore di servizi di trasporto, 465
 UTP (*vedi* Unshielded Twisted Pair)
 UUCP, 649

V

V.24, 110
 V.32, 108

- V.32.bis, 108
 V.34, 108
 V.42bis, 108
 Vacanza, demone di, 642
 Vano di fili, 81
 Variable Bit Rate, 443
 VBR (*vedi* Variable Bit Rate)
 Velocità della luce, 91
 Verifica di protocolli, 209-15
 Vettore di inizializzazione, 572
 Vettore, quantizzazione di, 711
 Very High Frequency, banda, 92, 95
 Very Low Frequency, banda, 92, 94
 Very Small Aperture Terminal, 159
 VHF, banda (*vedi* Very High Frequency, banda)
 Video composito, 706
 Video, 705-708
 analogico, 705-707
 campo, 706
 digitale, 707-708
 interallacciamento, 706
 progressivo, 706
 Video su richiesta, 721-33
 rete distributiva, 727-30
 server, 723-27
 scatole set-top, 730-33
 Video server, 723-27
 software, 725-27
 Videoconferenza, 5
 Virtual channel, ATM, 435
 Virtual circuit (*vedi* Circuito virtuale)
 Virtual path, ATM, 435
 Virus Internet, 698
 VISTAnet, 54
 Visualizzatore esterno, 663
 VLF, banda (*vedi* Very Low Frequency, banda)
 VSAT (*vedi* Very Small Aperture Terminal)
 VTMP, 553
- W**
- WAN (*vedi* Wide Area Network)
- WARC (*vedi* World Administrative Radio Conference)
 Wavelength Division Multiple Access, 249-51
 Wavelength Division Multiplexing, 115-16
 WDM (*vedi* Wavelength Division Multiplexing)
 WDMA (*vedi* Wavelength Division Multiple Access)
 Web (*vedi* World Wide Web)
 Web page (*vedi* Pagina Web)
 Wide Area Network, 10-12
 World Administrative Radio Conference, 93
 World Wide Web, 51, 659-702
 browser, 661
 CGI, 684-85
 HTML, 671-85
 HTTP, 668-70
 ipermediale, 663
 iperpuntatore, 661
 ipertesto, 661
 Java, 685-98
 motore di ricerca, 698
 server, 664-70
 URL, 671-74
 visualizzatore esterno, 663
 WWW (*vedi* World Wide Web)
 WYSIWYG, 674
- X**
- X.3, 57
 X.21, 57
 X.25, 57-58
 X.28, 58
 X.29, 58
 X.400, 624-42
 X.509, 647-48
 XTP, 553
- Z**
- Zipf, legge di, 723
 Zone, DNS, 609

