

Reti elaboratori.

Capitolo 1.

Reti di calcolatori ed Internet.

What's the Internet: "nuts and bolts" view.

Internet permette il colloquio tra vari dispositivi remoti (decine di miliardi di dispositivi).

È una rete di calcolatori ad accesso pubblico che interconnette miliardi di dispositivi di calcolo in tutto il mondo.

Questi dispositivi vengono chiamati **hosts** (ospitano applicazioni distribuite che coinvolgono più sistemi finali che si scambiano dati ed utilizzano il servizio della rete internet) ed **end systems** (ai bordi della rete).

I sistemi periferici sono connessi tra loro tramite una **rete di collegamenti** (communication link) e **commutatori di pacchetti** (packet switch). Tali collegamenti, possono essere costituiti da *varie tipologie di mezzi fisici*, tra cui cavi coassiali, fili di rame, fibre ottiche e onde elettromagnetiche.

Esistono *varie tecnologie di accesso*, alcune sono cablate (con fili) altre sono non cablate, quindi wireless.

Collegamenti diversi possono trasmettere dati a *velocità differenti*, e tale **velocità di trasmissione** (transmission rate) viene misurata in bit/secondo (bps).

I sistemi periferici accedono a Internet tramite **Internet service provider (ISP)** che comprendono ISP residenziali, aziendali, universitari e ISP che forniscono accesso WiFi.

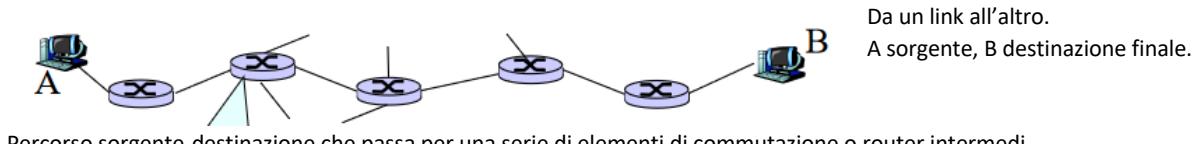
Un **provider** è insieme di commutatori di pacchetto e di collegamenti che sotto contratto fornisce ai (client) sistemi periferici svariati tipi di accesso alla rete, tra cui quello residenziale a larga banda come DSL, quello in rete locale ad alta velocità, quello senza fili (wireless) e dial-up tramite modem.

Quando un sistema periferico (end systems) vuole inviare dati a un altro sistema periferico, suddivide i dati in sotto parti e: l'insieme delle informazioni sono i **pacchetti**, sequenza di bit che viene suddivisa in pezzi.

Questi pacchetti sono inviati attraverso la rete alla destinazione, dove sono riassemblati per ottenere i dati originari.

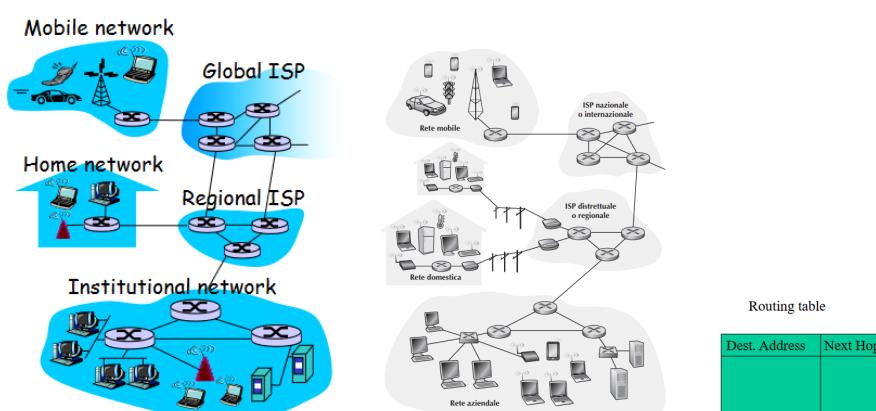
Un **commutatore di pacchetto** prende un pacchetto che arriva da uno dei collegamenti in ingresso e lo ritrasmette su uno di quelli in uscita. I pacchetti dell'odierna internet sono i router e i commutatori a livello di collegamento.

Router di larghezza di banda: **pacchetti di inoltro** (blocchi di dati). 
Inoltra un pezzo di informazioni (chiamato **pacchetto**) in arrivo su uno dei suoi collegamenti di comunicazione a uno dei suoi collegamenti di comunicazione in uscita (l'hop successivo sul percorso da sorgente a destinazione).



Forwarding: I router hanno una seconda funzionalità, quella di istradamento, un colloquio tra loro stessi che consente di riempire e aggiornare la tabella di istradamento. Quest'ultima viene riempita da algoritmi di routing e che continuano a funzionare in background con eventuali aggiornamenti nel tempo delle informazioni di tabella di istradamento. La routing table deve contenere le informazioni corrette che permettano di determinare il link prossimo.

Riassumendo: Il router riceve il pacchetto, basato su una tabella di routing e l'indirizzo di destinazione, calcola il 'next hop' alla destinazione. Inoltra il pacchetto al salto successivo. Il processo di calcolo e manutenzione la tabella di routing si chiama **Routing**.



Protocolli.

- Qualsiasi attività in Internet che coinvolge due o più entità remote in comunicazione viene governata da un protocollo. Il protocollo definisce regole che mittente e destinatario devono rispettare per comunicare.

Non può lasciare ambiguità, perché stanno comunicando delle macchine.

Macchine piuttosto che umani, tutte le attività di comunicazione in Internet regolate da protocolli,

Per esempio, i protocolli nei router determinano un percorso per il pacchetto dalla sorgente alla destinazione.

Cosa succede quando si invia una richiesta a un web server, ossia quando si digita l'indirizzo di una pagina web in un browser. Per prima cosa il calcolatore invierà un messaggio di richiesta di connessione al web server e si metterà in attesa di una risposta. Il web server alla fine riceverà il vostro messaggio di richiesta di connessione e restituirà un messaggio di risposta di connessione. Sapendo che ora è possibile richiedere un documento web, il computer invierà il nome della pagina che vuole prelevare dal server tramite un messaggio GET. Ora il web server restituirà la pagina web al vostro calcolatore.

- Lo scambio di messaggi e le azioni intraprese quando tali messaggi vengono inviati e ricevuti sono gli elementi chiave che definiscono un protocollo.

Un protocollo definisce il formato e l'ordine dei messaggi scambiati tra due o più entità in comunicazione, così come le azioni intraprese in fase di trasmissione e/o di ricezione di un messaggio o di un altro evento.

Questi protocolli controllano l'invio, ricezione di msg ad esempio, TCP, IP, HTTP, Skype, Ethernet.

Permettono di implementare lo scambio, le funzionalità alla base della rete internet e determinano il collegamento tra entità remote.

Ce ne sono di vario tipo:

http: per l'applicazione web; *Skype*: protocollo proprietario;

TCP consentono il dialogo tra entità remote end systems; *IP*: protocollo al cuore dell'istradamento.

Ethernet è una tecnologia e un protocollo che è funzionale allo scambio dati in una rete locale.

Rete di reti.

Internet non è una rete, la rete internet mette insieme tecnologie diverse, protocolli diversi.

Una cosa importante è stato progettare un'architettura di rete che interconnette in maniera efficiente e semplice sottoreti o reti eterogenee, in questa rete di reti che è stata poi la rete internet.

È una rete di reti gestita da entità amministrative autonome e diverse, che interconnettendo le loro reti ci permettono di avere il servizio di accesso alla rete e di scambio dati che è necessario nel nostro mondo attuale.

Negli ultimi anni servizi come Google si sono costruiti quasi un internet parallela di proprietà. Utilizzano ancora i servizi della rete pubblica, internet, ma utilizzano porzioni di rete private che solo alcuni operatori possono utilizzare.

Internet standards.

La rete internet è stata realizzata per essere pubblica e non utilizzata a fine privata da ricercatori, diversamente da altri standard, la standardizzazione sulla rete Internet è qualcosa di aperto, la IETF rende totalmente visibile la sua discussione interna e si apre a idee da parte di chiunque.

Topologia di una rete:

Tipologia fisica: (dispositivi fisici: cavo, pc, router,...)

dispositivi → hosts, end systems collegati tra loro e con un router e i router sono collegati tra loro.

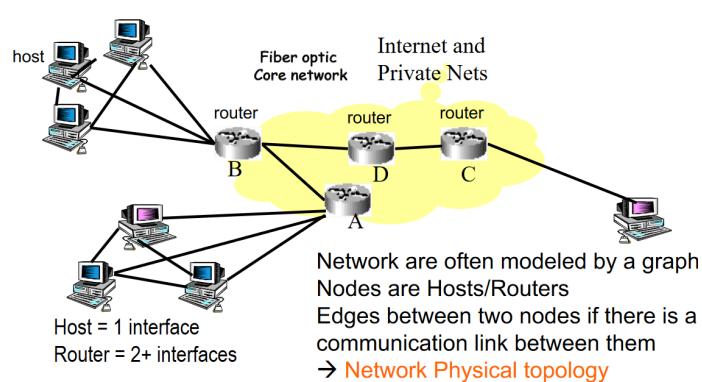
Nodi: end systems e router.

C'è un *arco* tra due nodi fisicamente connessi da un mezzo trasmissivo (link di comunicazione). Utile per determinare le logiche di istradamento in una rete.

Topologia logica della rete:

Nodo: elemento di commutazione, hosts.

Un arco esprime un percorso diretto che l'informazione può seguire tra hosts ed un elemento di commutazione, o tra due elementi di commutazione.



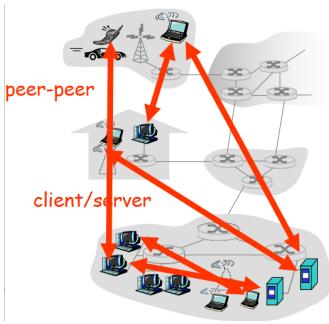
Le caratteristiche delle reti di accesso degli end systems.

Agli estremi della rete ci sono gli end systems.

Questi vengono detti anche *host* in quanto **eseguono** le *applicazioni distribuite* (che usufruiscono del servizio offerto dalla rete internet per poter operare) e hanno bisogno di interconnettersi al cuore della rete, lo fanno mediante reti di accesso, cioè la rete che connette fisicamente un sistema al suo edge router (router di bordo), che è il primo router sul percorso dal sistema d'origine a un qualsiasi altro sistema di destinazione collocato al di fuori della stessa rete di accesso.

Es: games online, colloquio. Ricerca in web, scambio informazioni web con le informazioni che stiamo cercando.

End systems (host) che eseguire programmi applicativi (Web, e-mail) seguono uno dei due paradigmi:



Modello client/server: client/server (uno dei due end system: **client** che richiede contenuti al secondo end system remoto: **server**, che memorizza contenuti, è pronto a ricevere richieste da altri end systems e le serve fornendo il contenuto richiesto).

Per esempio, richiesta Client-Server nel caso dell'applicazione Web. Il client, nel browser, faccia partire la richiesta a un server remoto che memorizza quel contenuto e lo fornisce su richiesta.

Richieste dell'host client, riceve il servizio dal server sempre attivo.

Per esempio. browser/server web; client/server di posta elettronica

Modello peer-peer: L'altro tipo di paradigma è peer-peer model: ci sono end systems che possono nel tempo svolgere ruoli di client e server. Ogni end systems può essere sia server(memorizza informazioni e le fornisce su richiesta) e client (richiede informazioni). Permettono rapidamente di distribuire contenuti. Utilizzo minimo (o nullo) di server dedicati.

Per esempio, scambio di file multimediale tra utenti, che possono mantenere localmente alcune porzioni di un video e le condividono tra di loro, richiedendo dei pezzi del contenuto multimediale del video non disponibile localmente e contemporaneamente fornendo pezzi del contenuto ad altri contenuti memorizzati localmente ad altri utenti che ne facciano richiesta. Skype, BitTorrent.

I mezzi trasmissivi sono dei mezzi **fisici** su cui passa Internet. Sono divisi in mezzi **vincolati** e mezzi **non vincolati**.

Quelli **vincolati** sono la fibra ottica, filo di rame o cavo coassiale.

Gli **altri** le onde si propagano nell'atmosfera e spazio esterno come nelle LAN wireless o nei canali digitali satellitari.

Principali componenti hardware detti anche dispositivi di rete: modem, switch, router, access point, client, server, DNS, proxy, nat.

Accedi a reti e media fisici.

Le **Access Network** (reti di accesso) sono reti che connettono fisicamente un sistema al suo **edge router** (router di bordo) cioè il *primo router sul percorso verso la destinazione*.

Diverse tipologie di rete di accesso (end systems al cuore della rete Internet). Come collegare gli end systems all'edge router?

- Reti di accesso **residenziali**: permettono l'accesso dalla loro abitazione.
- Reti di accesso **istituzionali** (scuola, azienda): rete locale da un ufficio.
- Reti di accesso **mobile**: quando si richiedono accessi ai contenuti internet in ogni momento dallo smartphone.

Ci sono altre caratteristiche che rendono diverse queste tecnologie di accesso; infatti, queste tecnologie possono essere diverse per la **banda** (data rate, quantità di bit al secondo) per le **prestazioni**, per la **latenza** (lunghezza del collegamento), affidabilità dei link di comunicazione e delle tecnologie di accesso in termini di bit error rate, in termini di probabilità di errore nella trasmissione di un singolo bit in quella tecnologia di accesso.

Se c'è un alto bit error rate, trasmetto pacchetti non bene ricevuti, bisogna gestire il tipo di problema con funzionalità di individuazione di presenza di errori nel pacchetto, poi cercare di diminuire la probabilità che un pacchetto non sia correttamente ricevuto e inoltre ci sono dei protocolli che ne richiedano la ritrasmissione.

Trasmissione attraverso un collegamento fisico.

Link con due **router**, due elementi di commutazione nella nostra rete di tipo **bidirezionale**, che trasmette di ricevere un router all'altro e viceversa. (Tx: trasmettitore, rx: ricevitore)

Quello che trasmettiamo sui nostri link di comunicazione è un **informazione digitale**, una sequenza di bit.

Questi bit propagano tra trasmettitore e ricevitore. Il collegamento fisico indica cosa c'è tra trasmettitore e ricevitore.

I mezzi di comunicazione tra trasmettitore e ricevitore possono essere:

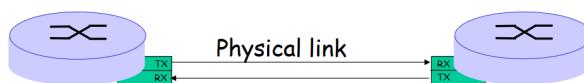
mezzi **guidati**: i segnali si propagano in **mezzi solidi**: rame, fibra, coassiale.

mezzi **non guidati**: i segnali si propagano liberamente tramite **onde elettromagnetiche**, ad es. radio

A seconda dei link trasmessi vengono utilizzate varie tecniche.

Posso modulare sinusoidi, segnale elettromagnetico cambiando la frequenza o l'ampiezza del segnale in funzione del fatto che sto trasmettendo un 0 o un 1. 1 valore alto, 0 valore basso.

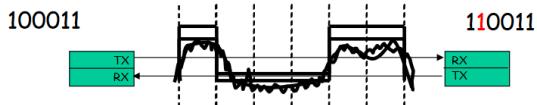
Posso inviare impulsi luminosi con dei link di comunicazione a fibra ottica.



Mentre il segnale viaggia, sperimenta:

- **Attenuazione** (assorbimento)
- **Distorsione** (larghezza di banda (frequenza) limitata)
- **Rumore** (interferenza, rumore termico)
- **Influenzato** da medio, bit rate e distanza.

Quindi potrebbe essere non riconosciuto il valore che era stato trasmesso.



Il primo valore che trasmettiamo è un valore alto (1), poi abbiamo per 3 tempi di simbolo due valori bassi (0) e poi per due tempi di simbolo abbiamo 2 valori alti (1).

La sequenza ricevuta potrebbe non essere corretta!!!

Questo porta a un tasso di errore nella trasmissione che dipende dalla particolare tecnologia e può variare.

Può essere molto basso o può essere significativo.

Un problema che si può verificare dalla situazione precedente è il: problema di sincronizzazione del ricevitore (nessuna transizione nel caso di sequenze di zeri o di uni) → NRZ 5B6B o 4B5B.

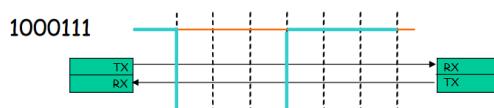
La loro visione del tempo è allineata.

Codifica NRZ (Non Return to Zero)

Ogni bit ha associato un valore stabile per la sua intera durata (1: High; 0: Low).

Nel caso di trasmissioni tutti 0 e tutti 1 posso perdere la sincronizzazione tra trasmettitore e ricevitore.

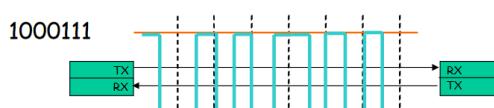
Quindi ogni tot tempo viene forzata una transizione alto basso, basso alto.



Codifica Manchester:

All'interno del tempo di simbolo, il tempo fra le due linette è dato una transizione basso-alto (codifica dello zero) o alto-basso (codifica del valore uno) in corrispondenza di ogni bit, per non perdere la sincronizzazione tra trasmettitore e ricevitore.

Usato in Ethernet 10Mbps e Token Ring.



Le tecnologie utilizzate adesso.

Accesso residenziale:

- Il primo modo che ha consentito l'accesso da un computer con una modalità di accesso residenziale è attraverso desktop. Questo tipo di connessione si basa sull'utilizzo di linee telefoniche tradizionali per trasportare pacchetti di dati e fornire agli utenti l'accesso al web.

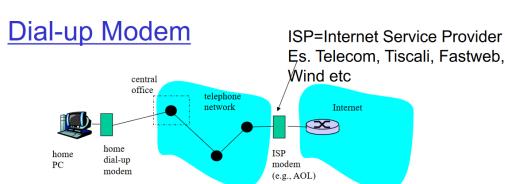
Viene utilizzato il **doppino telefonico** (cavo connesso alla rete telefonica), lo stesso tipo di link di comunicazione che collega il telefono alla rete telefonica tradizionale tramite un **modem dial-up** per connettere il computer al doppino.

Il **modem** permette di ricevere la sequenza di bit e modula il segnale analogico che viene essere trasmesso sul doppino.

Il modem trasforma l'analogico in digitale che può capire il router. L'informazione viene poi veicolata fino a un punto di accesso alla rete Internet. Nelle prime versione la velocità di accesso era molto limitata: 56 Kbps (56 mila bit al secondo).

Questa tecnologia prevedeva il fatto che un utente poteva o telefonare o connettere il desktop, a meno che non siano state installate 2 linee telefoniche separate e dedicate a ciascun servizio.

La tecnologia è evoluta nel tempo ed è diventata la base per la tecnologia **DSL**.



- Linea abbonati digitale (DSL).

Uno degli accessi residenziali a larga banda più diffusi. Un accesso residenziale a Internet di tipo **DSL** viene fornito dalla stessa compagnia telefonica che fornisce anche il servizio di telefonia fissa.

La prima caratteristica che distingue la DSL dal dial-up modem: nella DSL non c'è solo un modem, ma c'è anche uno **splitter**, il telefono è connesso allo splitter, e quest'ultimo dedica al segnale vocale la prima parte del doppino tra 0 e 4KHz e il resto lo utilizza per trasmettere la codifica di dati (sequenza di 0 e 1 che arrivano dal computer), cosicché si può utilizzare sia telefono che desktop.

Si utilizzano sempre doppi telefonici che sono evoluti e hanno fatto in modo che il data rate sia aumentato (utilizzare tutta la banda), per fare questo si deve trasmettere su una banda più grande oppure a trasmettere sullo stesso Hz con più bit al secondo.

Utilizza anche l'infrastruttura telefonica esistente fino a 1 Mbps in upstream (in genere < 256 kbps) fino a 8 Mbps in downstream (in genere < 1 Mbps) linea fisica dedicata alla centrale telefonica.

Se il central office è troppo distante si può ridurre la banda che posso utilizzare. Se si è distanti dal central office c'è un data rate minore. La velocità è notevolmente aumentata negli ultimi anni grazie alle tecnologie più resistenti alle interferenze e alla minor distanza tra modem DSM e DSLAM;

La DSL è stata espressamente progettata per distanze piccole tra l'abitazione e la centrale locale.

Digital Subscriber Line (DSL)

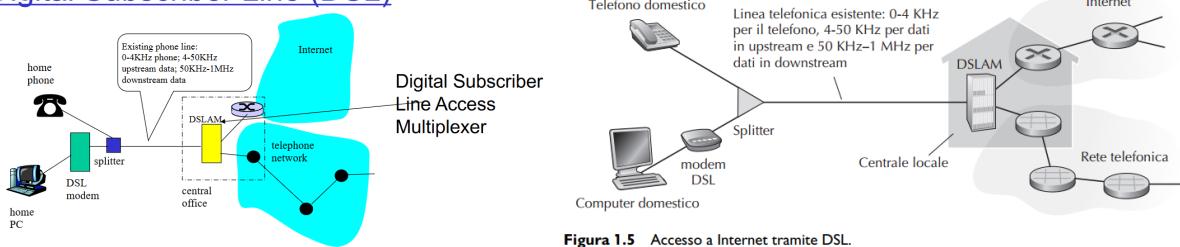


Figura 1.5 Accesso a Internet tramite DSL.

Estensore di loop ADSL.

Un loop extender **ADSL** o ripetitore ADSL è un dispositivo posizionato a metà strada tra l'abbonato e l'ufficio centrale della compagnia telefonica per estendere la distanza e aumentare la capacità del canale della propria connessione DSL.
In alcuni casi, il servizio può ora essere stabilito fino a 10 miglia dall'Ufficio Centrale (fattore di miglioramento 2)
Altre modalità di accesso dall'ambito residenziale.

- Via cavo.

Non utilizza l'infrastruttura telefonica, ma utilizza l'infrastruttura della TV via cavo.

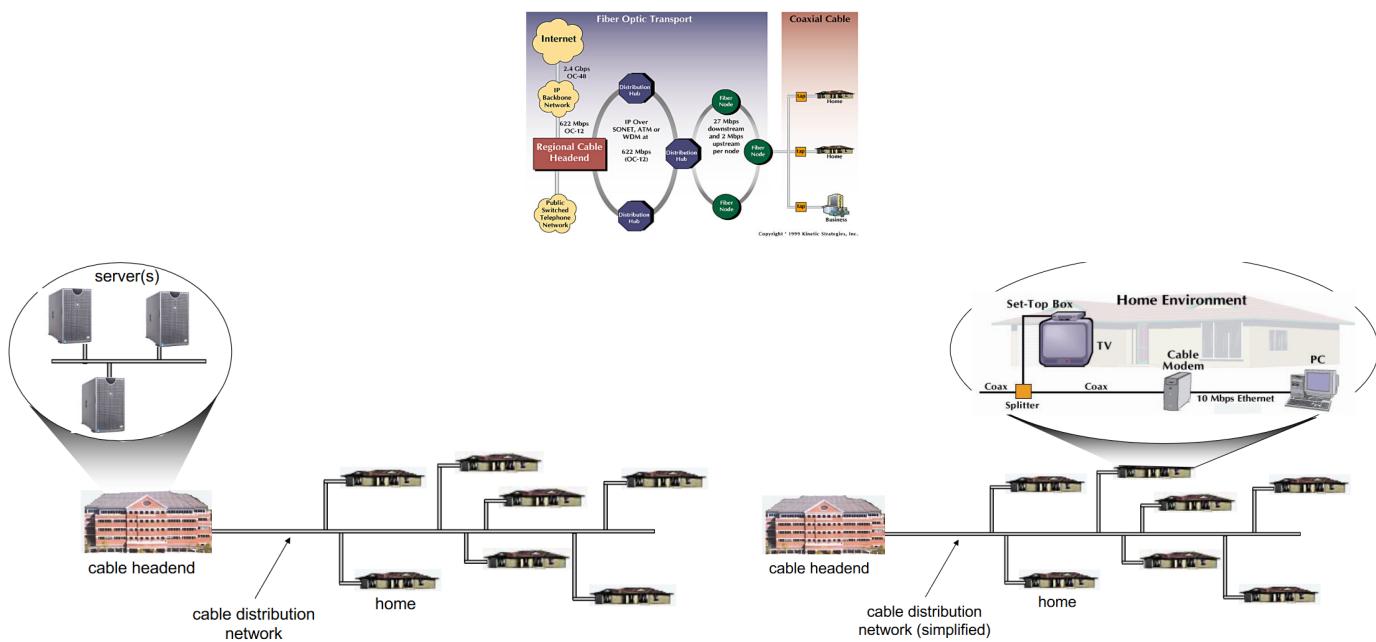
Migliaia di diverse abitazioni sono connesse al *gruppo di server* che **distribuiscono** i contenuti mediante la singola rete di accesso. Il collegamento era funzionale alla trasmissione delle informazioni alla codifica del segnale televisivo, mediante un collegamento dal monitor dello schermo tv e mediante un set-top box collegato a questa infrastruttura.

La stessa infrastruttura viene utilizzata per trasmettere informazioni digitali, verso computer connessi all'infrastruttura con uno split che divida l'utilizzo dell'infrastruttura dei canali fra quella porzione utilizzata per trasmettere il segnale video e quelle porzioni utilizzate per trasmettere i dati.

Lo split permette di dividere la banda a disposizione in più canali, alcuni utilizzati per la trasmissione di codifica video e per la trasmissione di codifica dei dati.

Questa infrastruttura è stata pensata per trasmettere il segnale video a tutti gli utenti, quindi condiviso tra i diversi utenti. Quando gli utenti trasmettono attraverso la fibra e verso internet contemporaneamente, essendo il mezzo condiviso c'è il problema di medium access control che deve essere gestito.

Quindi il segnale video condiviso non crea problemi, mentre quella di segnali dati sì.



- Fibra ottica. (alta velocità)

1. L'accesso di tipo **FTTH** (fiber to home) fornisce un collegamento in fibra ottica dalla centrale locale direttamente alle abitazioni. (velocità di accesso potenziale dell'ordine dei gigabit).
2. La rete di distribuzione ottica più semplice è chiamata **fibra diretta**, in cui una singola fibra collega una centrale locale a un'abitazione.

Di solito però una fibra uscente dalla centrale locale è in effetti **condivisa** da molte abitazioni e solo quando arriva relativamente vicina alle abitazioni viene suddivisa in più fibre, ognuna dedicata a un utente. Vi sono due architetture che eseguono questa suddivisione: una rete ottica passiva e una rete ottica attiva.

La **Fibra Ottica** è una tecnologia che **permette di trasmettere grandi quantità di dati ad altissima velocità**.

La fibra ottica è un cavo formato da sottilissimi filamenti trasparenti in fibra di vetro, per la precisione in silicio, o in polimeri plastici, tenuti insieme in una piccola guaina di materiale isolante. L'ADSL, invece, utilizza un cavo in rame, chiamato "doppino".

Il cavo in fibra ottica si presenta come un fascio di filamenti divisi in una sezione interna trasparente e una esterna più opaca capace di riflettere la luce.

I cavi in fibra ottica permettono di trasmettere dati molto più velocemente rispetto alle performance dell'ADSL. I materiali utilizzati sono leggeri e flessibili e più capaci, rispetto ai cavi in rame, di respingere eventuali disturbi elettrici o dovuti a perturbazioni atmosferiche.

Il meccanismo con cui viaggiano i dati è quello dello specchio tubolare. La luce che entra nel nucleo viaggia grazie a una serie di riflessioni tra i due materiali del nucleo e del mantello. In questo modo i dati vengono trasportati tra i diversi dispositivi che devono ricevere e trasmettere i dati: i modem, i router, i server e le infrastrutture degli operatori telefonici.

I cavi in fibra di vetro o in polimeri della fibra ottica sono capaci di sfruttare una banda di frequenze molto elevata. Questo consente di trasferire dati, sotto forma di segnali di luce, con una enorme velocità di trasmissione. Tutti i più importanti operatori telefonici offrono pacchetti che includono la fibra ottica e consentono di navigare fino a 2.5 Giga in download.

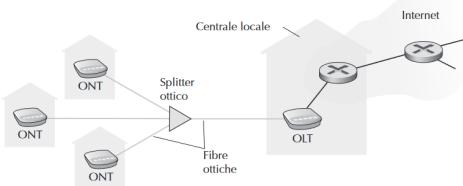


Figura 1.7 Accesso a Internet tramite FTTH.

Accesso aziendale (e residenziale): Ethernet e WiFi:

Per collegare i sistemi periferici al router di bordo si utilizza:

- **Rete locale (LAN, Local Area Network)** è una rete di collegamento che copre un'area limitata (abitazioni, aziende, università) e che utilizza tecnologia Ethernet (LAN) o Wireless (WLAN).

Una **LAN** può comprendere, oltre ai computer e alle periferiche condivise, dispositivi di instradamento come switch e router, dispositivi per il bilanciamento del carico (load balancing) e dispositivi di sicurezza come i firewall, i proxy e i sistemi di rilevamento di intrusioni. Le LAN più semplici si basano in genere su uno o più switch, che possono essere a loro volta collegati a un router, a un modem analogico o a un modem ADSL per l'accesso a Internet.

- La tecnologia **Ethernet** utilizza *un doppino di rame intrecciato* per **collegare numerosi sistemi periferici tra loro** e connetterli a uno switch Ethernet e portarli verso l'ISP tramite un router. L'accesso ha generalmente velocità intorno ai 100 Mbps.

Il principio di funzionamento di una rete Ethernet è che non esiste un dispositivo che faccia da master, ma ogni host è in grado di ricevere tutto quanto passa dal cavo coassiale, e solamente uno alla volta può trasmettere le informazioni.

Lo **switch** è un elemento di commutazione che implementa il livello fisico e il livello data link soltanto, non effettua istradamenti e non cerca il migliore percorso tra sorgente e destinazione.

Ethernet Internet access

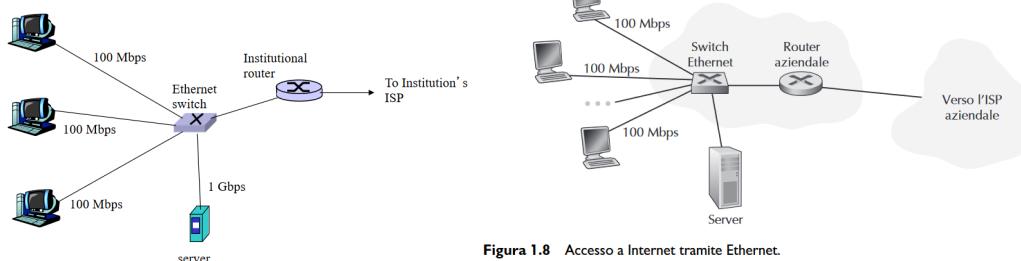


Figura 1.8 Accesso a Internet tramite Ethernet.

Le altre tipologie di accesso: siamo passati da DSL alla sua evoluzione, Ethernet (cablate) a una tipologia che è più wireless per la facilità di supportare la modalità degli utenti.

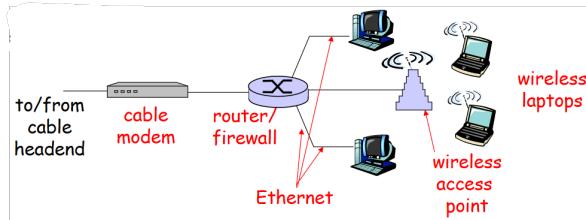
Nel caso accesso residenziale la tecnologia **WiFi**.

- **WiFi** è un insieme di tecnologie per reti locali **senza fili** (WLAN), il quale consente a più dispositivi (per esempio personal computer, smartphone, smart TV, ecc.) di essere connessi tra loro tramite onde radio (frequenze radio) e scambiare dati.

Si può dire che la connessione è "nell'aria" e che i computer la prendono al volo.

I dispositivi compatibili Wi-Fi possono connettersi a Internet tramite una WLAN e un punto di accesso wireless (access point).

In una LAN wireless gli utenti trasmettono e ricevono pacchetti entro un raggio di poche decine di metri da e verso un **access point wireless** (detto anche stazione base o base station, dispositivo fisico che **diffonde il segnale** in modo che i computer lo rilevino, riesce a trasmettere 100 Mbps) connesso a una rete aziendale, che probabilmente include una rete Ethernet cablata, a sua volta connessa a Internet. La rete Wi-Fi è facilmente estendibile una volta installata (si possono aggiungere antenne e parabole per ripetere il segnale fino a distanze molto elevate). Le **LAN wireless** (nota anche come Wi-Fi) si stanno rapidamente diffondendo in ambienti universitari, uffici, locali pubblici, abitazioni e persino sugli aeroplani.



Con la tecnologia disponibile al 2017, un access point (o un hotspot) all'interno di un edificio può avere una portata di circa 20 metri (il segnale a onde radio è attenuato dai muri), mentre all'esterno può coprire un raggio di circa 100 metri e, usando più punti di accesso sovrapposti, anche di diversi chilometri quadrati.

Accesso wireless su scala geografica: 3G e LTE:

I dispositivi iPhone e Android vengono sempre più frequentemente impiegati per inviare messaggi, condividere fotografie, guardare film e ascoltare musica durante i propri spostamenti. Tali dispositivi utilizzano la stessa infrastruttura wireless usata dalla telefonia cellulare per inviare/ricevere pacchetti tramite una stazione base gestita da un fornitore di telefonia cellulare.

A differenza del WiFi, l'utente può trovarsi a poche decine di chilometri dalla stazione base, invece che a poche decine di metri.

Le **compagnie di telecomunicazioni** hanno investito somme enormi nelle:

- reti wireless di terza generazione (**3G**), che consentono accesso wireless a Internet con commutazione a pacchetto e su scala geografica a velocità che superano 1 Mbps.

Tuttavia, sono già disponibili tecnologie di accesso su scala geografica a velocità ancora più alta:

- reti wireless di quarta generazione (**4G**), LTE che fonda le sue basi nella tecnologia 3G e può potenzialmente raggiungere velocità superiori a 10 Mbps. In ambiti industriali sono state riscontrate velocità LTE in downstream di molte decine di Mbps.

I prossimi a venire: i sistemi 5G.

Accesso satellitare:

Un ultima modalità di collegamento di link tipico della porzione della rete Internet che si utilizzano in diversi casi.

Quando andiamo in nave, ci possiamo collegare mediante **link satellitare**. Trasmette un segnale fino a un satellite che lo ritrasmette verso la terra. Consegniamo le informazioni da un area del pianeta a un'altra.

I satelliti sono a 500km dalla superficie terrestre. Si pagano latenze elevate perché il link è un link lungo e c'è un ritardo di propagazione necessario. Esistono delle costellazioni di satelliti più basse per i quali riusciamo a passare per inoltrare l'informazione. Il percorso è più corto.

Gli utenti di Internet di oggi sono 5 miliardi. C'è una distribuzione che dipende dalla zona del pianeta.

Si cerca di utilizzare tecnologie wireless.

35,5 Gb di traffico per utente a mese.

Le prestazioni relative ai diversi flussi di traffico sono diverse e devono essere gestite da internet.

Non ci sono più gli utenti umani, ma c'è il mondo dell'Internet delle cose e sempre più abbiamo contenuti e informazioni che sono connessi alla rete per connettersi agli utenti finali.

15 miliardi di dispositivi IoT connessi alla rete.

Vista dei servizi.

La rete Internet offre un servizio di comunicazione, un'infrastruttura di comunicazione che permette lo scambio dati per applicazioni distribuite: web, videogiochi, e-commerce, scambi di file.

La rete offre anche servizi di comunicazione forniti alle app:

- affidabile consegna dei dati da sorgente a destinazione.
- "best effort (del mio meglio)" (inaffidabili) consegna dati. Faccio del mio meglio, ma non ne sono sicuro.

Due servizi dalla rete Internet:

Connection-oriented service: prima del trasferimento dati c'è una stretta di mano, in cui si stabilisce una connessione tra entità remote e viene creato uno stato locale non all'interno della rete, ma tra due end systems direttamente. Dopo il successo della stretta di mano possono essere scambiate informazioni. Quindi parliamo di connection-oriented perché c'è questa connessione con stretta di mano.

Esempio: Servizio TCP [RFC 793]

Trasferimento dati affidabile e in ordine di flusso di byte.

Se arrivano troppe informazioni, troppo velocemente c'è perdita: riconoscimenti e ritrasmissioni.

Ho un buffer di dimensione limitata di destinazione in cui le informazioni sono memorizzate per essere lette dal livello applicativo. Se si satura questo buffer a destinazione perché arrivano troppe informazioni c'è un buffer overflow e c'è perdita di informazione.

Controllo del flusso: il mittente non sopraffare il destinatario, ci sono ulteriori protocolli di flusso per far arrivare informazioni non troppo velocemente.

Nei router ci sono buffer di dimensione limitata. Se arrivano troppe informazioni più rapidamente del tasso di smaltimento, cresce la dimensione del buffer e si perdono le informazioni in rete.

Controllo della congestione: i mittenti "rallentano la velocità di invio" quando la rete è congestionata.

Connectionless service:

La fase di stretta di mano non è presente.

Obiettivo: trasferimento dati tra end systems.

Si preferisce la **velocità** della trasmissione delle informazioni, rispetto all'affidabilità.

Esempio: Servizio UDP.

UDP: User Datagram Protocol [RFC 768]: Internet è una connessione senza servizio:

Trasferimento dei dati inaffidabile (best effort), nessun controllo del flusso e nessun controllo della congestione.

Leggero e veloce, da utilizzare quando non ci sono richieste di affidabilità.

App's using TCP: HTTP (Web), FTP (file transfer), Telnet (remote login), SMTP (email)

App's using UDP: streaming media, teleconferencing, DNS, Internet telephony

The network core (Il nucleo della rete).

Lo definiamo come una maglia di router interconnessi, una maglia di commutatori di pacchetti e collegamenti che interconnettono i sistemi periferici di Internet.

Come vengono trasferiti i dati attraverso la rete? 2 modalità:

Commutazione di pacchetto, packet-switching:

Dati inviati attraverso la rete in "pezzi" discreti.

La commutazione di pacchetto è una tecnica che consente di suddividere il messaggio da trasmettere in parti piccole detti **pacchetti**.

Ogni pacchetto è composto da un

- **Header:** utilizzato ai fini dell'identificazione e gestione;
- **Payload:** che contiene i dati veri e propri.

Tra sorgente e destinazione, ogni pacchetto viaggia attraverso collegamenti di comunicazione e switch di pacchetto (per i quali esistono due tipi predominanti, router e switch a livello di collegamento).

Ogni pacchetto utilizza l'intera larghezza di banda del collegamento e le risorse sono utilizzate secondo necessità. Quindi, se un sistema di origine o uno switch di pacchetto invia un pacchetto di L bit su un collegamento con velocità di trasmissione R bit/sec, il tempo per trasmettere il pacchetto è L / R secondi.

Ciascun pacchetto seguirà la sua strada per poi raggiungere il destinatario.

I pacchetti saranno poi ricostruiti a destinazione grazie al protocollo TCP.

Il pacchetto viene inviato alla rete senza riservare alcuna risorsa di collegamento; quindi, viaggia su più risorse di collegamenti.

Internet fa del suo meglio per fornire pacchetti in modo tempestivo, ma non fornisce alcuna garanzia.

Un vantaggio per quanto riguarda la commutazione a pacchetto è dato dal fatto che 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.

La maggior parte degli switch di pacchetto utilizza la trasmissione **store-and-forward** agli ingressi dei collegamenti.

Ciò significa che il commutatore deve ricevere l'intero pacchetto prima di poterne cominciare a trasmettere sul collegamento in uscita il primo bit.

Trascurando ritardi e tempi di elaborazione, se T è il tempo di trasmissione tra un host e il commutatore (numero bit/velocità mezzo), allora il tempo totale di trasmissione tra due host che vogliono scambiare un pacchetto attraverso un router è pari a 2T. Il tempo di trasmissione di P pacchetti attraverso un router è $(P+1)T$. Il ritardo totale per una trasmissione di P pacchetti su un percorso di N router è pari a $(P+N)T$.

Ritardi di accodamento e perdita di pacchetti.

Ogni commutatore, per ciascuno dei collegamenti a cui è connesso, mantiene un buffer di output (o coda) dove conserva i pacchetti che sta per inviare. Un pacchetto in arrivo che richiede l'invio attraverso un determinato collegamento, ma lo trova occupato dalla trasmissione di un altro, deve accodarsi nel buffer.

Questo comporta dei ritardi di accodamento che vanno ad incrementare i tempi di trasmissione di un pacchetto. Tali ritardi ovviamente sono variabili e dipendono dal livello di traffico sulla rete.

Dato che la dimensione del buffer è finita può accadere che un pacchetto in arrivo trovi il buffer completamente pieno. Si verificherà una perdita di pacchetto (packet loss).

Un problema di interesse ingegneristico è il controllo di congestione: adeguare la velocità di invio pacchetti da parte dell'end-system alla velocità di inoltro del router, in modo da evitare code di bufferizzazione piene.

Tabelle di inoltro e protocolli di instradamento.

Un **router** prende un pacchetto in arrivo su uno dei suoi collegamenti di comunicazione collegati e inoltra quel pacchetto su un altro dei suoi collegamenti di comunicazione collegati. Ma in che modo il router determina a quale collegamento deve inoltrare il pacchetto?

L'inoltro di pacchetti viene effettivamente eseguito in modi diversi in diversi tipi di reti di computer.

In Internet ogni end-system (sistema periferico) ha un **indirizzo IP**. Ogni pacchetto che percorre la rete contiene nella sua intestazione (**header**) l'indirizzo IP della sua destinazione. Ogni router ha una **tavella di inoltro** (forwarding table) che mette in relazione gli indirizzi di destinazione con i collegamenti in uscita del router stesso.

Quando il pacchetto giunge ad un router, quest'ultimo esamina l'indirizzo della destinazione e consulta la propria tabella per determinare il collegamento uscente appropriato verso cui dirigere il pacchetto in uscita.

Internet implementa parecchi protocolli di instradamento (routing protocol) per impostare automaticamente queste tabelle di inoltro.

Commutazione di circuito.

La **commutazione di circuito** è una tecnica che **prevede** un canale dedicato ed univoco per il trasferimento di dati e messaggi.

Questo è stato proprio progettato per la comunicazione vocale ed era meno adatto per la trasmissione di dati.

Nelle **reti a commutazione di circuito** le risorse richieste lungo un percorso (buffer, velocità di trasmissione del collegamento) per consentire la comunicazione tra sistemi periferici sono riservate per l'intera durata della sessione di comunicazione.

Le reti telefoniche tradizionali sono esempi di **reti a commutazione di circuito**.

Considera cosa succede quando una persona desidera inviare informazioni (voce o fax) a un'altra tramite una rete telefonica. *Prima* che il mittente possa inviare le informazioni, la rete deve stabilire una connessione (circuito) **end-to-end** tra il mittente e destinatario. Necessario stabilire percorso dedicato tra l'origine e la destinazione prima del trasferimento dei dati.

Questa è una connessione in buona fede per la quale gli interruttori sul percorso tra il mittente e destinatario mantengono lo stato di connessione per quella connessione.

Per esempio, per una chiamata telefonica c'è una connessione fisica di rame stabilita effettuando la chiamata.

Nel gergo della telefonia, questo connessione è chiamata **circuito**. Quando la rete stabilisce il circuito, riserva anche una costante velocità di trasmissione nei collegamenti della rete (che rappresentano una frazione della capacità di trasmissione di ciascun collegamento) per la durata del collegamento.

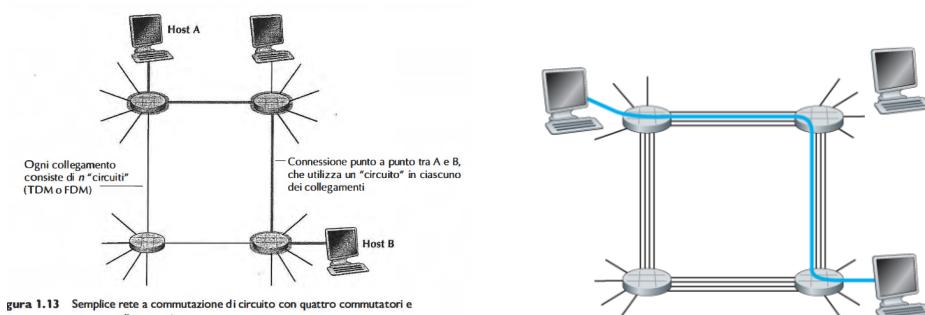
Poiché per questo mittente è stata riservata una determinata velocità di trasmissione connessione del ricevitore, il mittente può trasferire i dati al ricevitore alla velocità costante garantita.

Gli host (ad esempio PC e workstation) sono direttamente collegati a uno degli interruttori.

Quando due host vogliono comunicare, la rete stabilisce un terminale dedicato connessione to-end tra i due host. Pertanto, affinché l'Host A comunichi con l'Host B, la rete deve prima riservare un circuito su ciascuno dei due collegamenti. In questo esempio, l'end-to-end dedicato la connessione utilizza il secondo circuito nel primo collegamento e il quarto circuito nel secondo collegamento.

Perché ciascuno link ha quattro circuiti, per ogni link utilizzato dalla connessione end-to-end, la connessione ottiene un quarto della capacità trasmissiva totale del collegamento per la durata del collegamento.

Così, ad esempio, se ogni collegamento tra switch adiacenti ha una velocità di trasmissione di 1 Mbps, quindi ogni switch di circuito end-to-end la connessione ottiene 250 kbps di velocità di trasmissione dedicata.



La larghezza di banda disponibile in questo caso è fissa.

Un circuito all'interno di un collegamento è implementato tramite **multiplexing a divisione di frequenza** (FDM, frequency-division multiplexing) o **multiplexing a divisione di tempo** (TDM, time-division multiplexing).

Dividere la larghezza di banda del collegamento in "pezzi": divisione di frequenza e divisione del tempo

Con **FDM**, lo spettro di frequenza di un collegamento è suddiviso tra le connessioni stabilite con il collegamento.

In particolare, il collegamento dedica una banda di frequenza con un'ampiezza di 4 kHz (ovvero 4.000 hertz o 4.000 cicli al secondo) a ciascuna connessione per la durata della connessione.

La larghezza della banda è chiamata, non a caso, **larghezza di banda**. Le stazioni radio FM utilizzano anche FDM per condividere lo spettro di frequenza tra 88 MHz e 108 MHz, con ciascuna stazione assegnata a una banda di frequenza specifica.

Per un collegamento **TDM**, il tempo è **diviso** in *frame di durata fissa* e ogni **frame** è **diviso** in un *frame fisso numero di fasce orarie*. Quando la rete stabilisce una connessione attraverso un collegamento, la rete dedica un intervallo di tempo in ogni frame a questa connessione. Questi slot sono dedicati al solo uso di quella connessione, con un intervallo di tempo disponibile per l'uso (in ogni frame) per trasmettere i dati della connessione.

I fautori della commutazione di pacchetto hanno sempre sostenuto che la commutazione di circuito è uno spreco perché i circuiti dedicati sono inattivi durante i periodi di silenzio.

Ad esempio, quando una persona in una telefonata smette di parlare, le risorse di rete inattive (bande di frequenza o fasce orarie nei collegamenti lungo il percorso della connessione) non possono essere utilizzate da altre connessioni in corso.

Consideriamo il tempo necessario per inviare un file di 640.000 bit dall'host A all'host B su una rete a commutazione di circuito.

Supponiamo che tutti i collegamenti nella rete utilizzino TDM con 24 slot/sec e abbiano una velocità in bit di 1,536 Mbps. Si supponga inoltre che occorrono 500 msec per stabilire un circuito end-to-end prima che l'host A possa iniziare a trasmettere il file. *Quanto tempo ci vuole per inviare il file?*

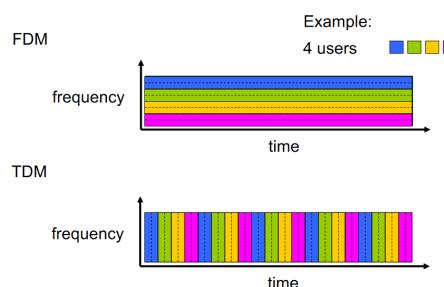
Ogni circuito ha una velocità di trasmissione di quindi ci vogliono secondi per trasmettere il file.

A questi 10 secondi aggiungiamo il tempo di costituzione del circuito, dando 10,5 secondi per inviare il file.

Si noti che il tempo di trasmissione è indipendente dal numero di collegamenti: il tempo di trasmissione sarebbe di 10 secondi se il circuito end-to-end passa attraverso un collegamento o cento collegamenti.

Il ritardo end-to-end effettivo $(1.536 \text{ Mbps})/24=64 \text{ kbps}$, $(640.000 \text{ bit})/(64 \text{ kbps})=10$ include anche un ritardo di propagazione include anche un ritardo di propagazione.

Più la realizzazione del circuito → 10,5s.



Vantaggi della commutazione di **circuito**:

- Otterrai l'intera larghezza di banda per la durata della chiamata.
- Riduce la quantità di ritardo che l'utente sperimenta prima e durante una chiamata.
- La commutazione del circuito per la chiamata verrà stabilita con canali, larghezza di banda e una velocità dati costante.
- Sotto la commutazione del circuito, la chiamata dovrebbe essere fornita con canali logici, larghezza di banda e una velocità dati continua.
- Un percorso / circuito dedicato fornisce la consegna dei dati garantita.

Svantaggi della commutazione di **circuito**:

- Ci vuole più tempo per impostare il circuito.
- Durante un disastro o una crisi, la rete potrebbe diventare instabile o non disponibile.
- Dedicare un canale a un singolo utilizzo lo rende non disponibile per altri servizi.
- Richiede più larghezza di banda.
- In questo tipo di metodo di commutazione del circuito, entrambe le estremità devono funzionare alla stessa velocità durante l'intera connessione.
- Fornisce un intero canale a un servizio e un percorso individuale.

Vantaggi della commutazione di **pacchetto**:

- Questo metodo aiuta i dispositivi di velocità diverse a comunicare tra loro.
- Elevata trasmissione dei dati.
- Ti aiuta a stabilire istantaneamente una connessione.
- Viaggio indipendente.
- Il ritardo nella consegna dei pacchetti è inferiore in quanto i pacchetti vengono inviati non appena sono disponibili.
- Il cambio di dispositivo non richiede spazio di archiviazione enorme.
- La consegna dei dati può essere portata avanti anche se alcune parti della rete devono affrontare problemi di errore di collegamento.
- Fornisce l'utilizzo simultaneo dello stesso canale con l'aiuto di più utenti.

Svantaggi della commutazione di **pacchetto**:

- In condizioni di utilizzo intenso, può verificarsi un ritardo significativo nel processo.
- Il cambio di pacchetto dipende da una serie di protocolli complessi che possono essere gestiti dalla distribuzione.
- I pacchetti di dati possono essere danneggiati o persi.
- I protocolli sono necessari per un trasferimento affidabile.
- La commutazione di pacchetto fornisce solo un'esperienza di chiamata vocale che può provocare un audio instabile, rendendo difficile per gli utenti capirsi a vicenda.
- Ti aiuta a ridurre i costi in diversi modi.

Una rete di reti.

Gli utenti che accedono a Internet e che eseguono le applicazioni distribuite lo fanno mediante entità che offrono un servizio di accesso alla rete che sono **internet service provider (ISP)**, che *gestiscono router connessi tra loro e offrono un servizio di scambio dati funzionali*, una connettività attraverso una rete cablata o senza fili con svariate tecnologie quali DSL, cavo, FTTH, Wi-Fi e cellulare.

L'obiettivo generale è quello di interconnettere gli ISP di accesso in modo che i sistemi periferici possano scambiarsi pacchetti.

Una backbone nei campi delle telecomunicazioni e dell'informatica è un collegamento ad alta velocità di trasmissione e capacità tra due server o router di smistamento informazioni e appartenente normalmente alla rete di trasporto di una rete di telecomunicazioni.

Questi **ISP** sono di diverso tipo e hanno una copertura globale, riescono a mettere in collegamento utenti che si trovano nelle diverse porzioni del mondo. Possiedono porzioni ampie della rete Internet.

Struttura gerarchica, quando un end systems accede alla rete invia delle informazioni al primo router che appartiene a ISP1 di rete locale e queste informazione verrà veicolate tramite una rete di reti che passa per ISP2 o ISP3 fino a raggiungere l'end system. Ci sono degli accordi commerciali che determinano il percorso.

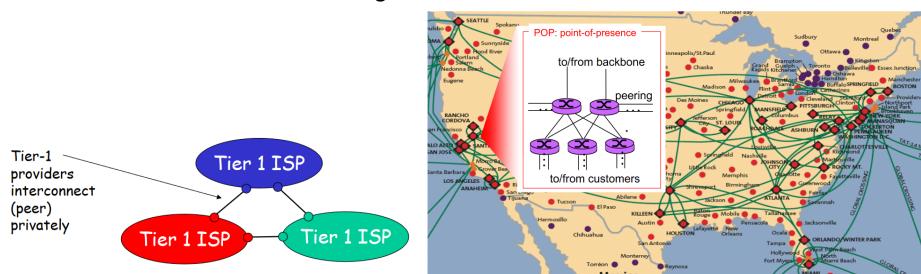
Gli **ISP di primo livello** (detti anche, soprattutto negli USA, **NSP**, Network Service Provider o Fornitori di Accesso alla Rete), hanno una rete di interconnessione velocissima, sono internazionali e sono direttamente connessi ad altri ISP di primo livello.

Sono queste le compagnie che fanno gli scavi e stendono migliaia di chilometri di cavo in fibra ottica per connessioni a lungo chilometraggio ed altissime velocità (dette dorsali) e ne sono quindi anche le proprietarie.

Tanto per fare un esempio, dal computer di casa non ci si può connettere direttamente ad un NSP perché occorre un hardware altamente specializzato e costoso e si deve richiedere una banda minima certamente superiore ai pochi MB/s di una ADSL domestica.

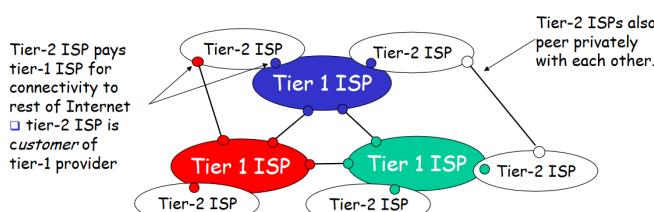
Un esempio di un backbone di un operatore Tier-1 è lo Sprint. Il cuore della rete è costituito da un numero limitato di punti del backbone che coprono le grandi popolazioni dal link (verde) ad altissima velocità realizzati in fibra ottica.

La fibra ottica riesce a mantenere altissima la velocità anche a grandi distanze.



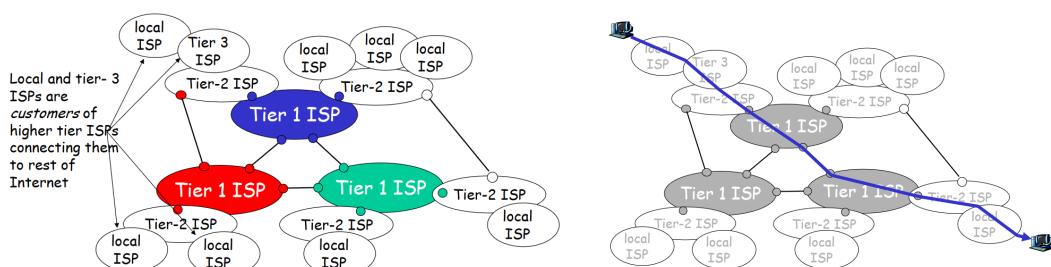
Questi ISP di livello 1 offrono questo il servizio di scambio dati agli **ISP di livello 2** (struttura gerarchica) che utilizzano il servizio di comunicazione da un capo all'altro del mondo per veicolare il traffico dei loro utenti.

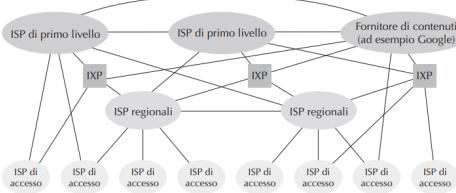
È possibile che ISP2 di livello 2 sono connessi tra di loro con 2 router di ISP2 connessi da link di alta velocità per scambiarsi subito informazioni.



Gli ISP di primo livello sono connessi a un certo numero di **ISP di secondo** e di **terzo livello** (questi ultimi detti anche **IAP**, Internet Access Provider o Fornitori di Accesso ad Internet) che non sono proprietari della dorsale ma che comprano una fetta di banda da un NSP per rivenderla ad utenti periferici come aziende o privati, generalmente aziende o enti pubblici. Questi **ISP di terzo livello** offrono servizio di accesso a Internet a ISP degli utenti stessi.

Il punto di contatto tra Internet e l'utente domestico è il **POP** o Point Of Presence (da non confondere con il protocollo di posta in arrivo, detto esso pure POP o Post Office Protocol).





I maggiori Internet Service Provider italiani sono nell'ordine: Telecom Italia, Vodafone Italia, NGI, TeleTu, Infostrada, Tiscali, MC-Link, Fastweb, Interfree, Brain Technology S.p.A. - Playnet, Aruba.

Oltre a fornire l'accesso a Internet, molti ISP forniscono anche il servizio di registrazione dei nomi di dominio. In Italia, gli ISP che forniscono questo tipo di servizio si chiamano anche Maintainers o MNT.

Prestazioni di una rete di telecomunicazione. Ritardo e perdita di informazioni in rete.

Idealmente, vorremmo che i servizi Internet fossero in grado di spostare una quantità di dati qualsiasi tra due sistemi periferici, instantaneamente e senza alcuna perdita di dati. Purtroppo, sebbene questo sia un nobile obiettivo, non è possibile raggiungerlo nella realtà.

Ricordiamo che un pacchetto parte da un host (la sorgente), passa attraverso una serie di router e conclude il viaggio in un altro host (la destinazione). A ogni tappa, il pacchetto subisce vari tipi di ritardo a ciascun nodo (host o router) del tragitto. Di tali ritardi, i principali sono il **ritardo di elaborazione**, il **ritardo di accodamento**, il **ritardo di trasmissione** e il **ritardo di propagazione**, che complessivamente formano il **ritardo totale di nodo** (nodal delay).

Le prestazioni di molte applicazioni per Internet come la ricerca, la navigazione sul Web, l'e-mail, le mappe, la messaggistica istantanea e il voice-over-IP sono pesantemente influenzate dai ritardi di rete.

Ritardo di elaborazione.

Il tempo richiesto per esaminare l'header del pacchetto e per determinare dove dirigerlo fa parte del **ritardo di elaborazione (processing delay)**. Questo può anche includere altri fattori, tra i quali il tempo richiesto per controllare errori a livello di bit durante la trasmissione dal nodo a monte al router A.

Nei router ad alta velocità questi ritardi sono solitamente dell'ordine dei microsecondi o inferiori. Dopo l'elaborazione, il router dirige il pacchetto verso la coda che precede il collegamento al router B.

Ritardo di accodamento.

Una volta in coda, il pacchetto subisce un **ritardo di accodamento (queuing delay)** mentre attende la trasmissione sul collegamento. La lunghezza di tale ritardo per uno specifico pacchetto dipenderà dal numero di pacchetti precedentemente arrivati, accodati e in attesa di trasmissione sullo stesso collegamento. Se la coda è vuota e non è in corso la trasmissione di altri pacchetti, il ritardo di accodamento per il nostro pacchetto è nullo.

D'altro canto, se il traffico è pesante e molti altri pacchetti stanno anch'essi aspettando la trasmissione, il ritardo di accodamento è elevato. Nella pratica i ritardi di accodamento possono essere dell'ordine dei microsecondi o dei millisecondi.

Ritardo di trasmissione.

Assumendo che i pacchetti siano trasmessi secondo la politica first-come-first-served (il primo che arriva è il primo a essere servito), come avviene comunemente nelle reti a commutazione di pacchetto, il nostro pacchetto può essere trasmesso solo dopo la trasmissione di tutti quelli che lo hanno preceduto nell'arrivo.

Il ritardo di trasmissione è la quantità di tempo impiegata dal router per trasmettere in uscita il pacchetto, ed è funzione della lunghezza del pacchetto e della velocità di trasmissione del collegamento, ma non ha niente a che fare con la distanza tra i due router.

Sia L la lunghezza del pacchetto, in bit, e R bps la velocità di trasmissione del collegamento dal router A al router B.

Il **ritardo di trasmissione (transmission delay)** risulta essere L/R . Questo è il tempo richiesto per trasmettere tutti i bit del pacchetto sul collegamento.

Anche i ritardi di trasmissione sono di solito dell'ordine dei microsecondi o dei millisecondi.

Ritardo di propagazione.

Il ritardo di propagazione, invece, è il tempo richiesto per la propagazione di un bit da un router a quello successivo, ed è funzione della distanza tra i due router, ma non ha niente a che fare con la lunghezza del pacchetto o con la velocità di trasmissione propria del collegamento.

Il tempo impiegato è il **ritardo di propagazione (propagation delay)**. Il bit viaggia alla velocità di propagazione del collegamento, che dipende dal mezzo fisico (fibra ottica, doppino in rame e così via) ed è compresa nell'intervallo che va dai 2×108 m/s ai 3×108 m/s, corrispondente quest'ultimo alla velocità della luce. Il ritardo di propagazione è dato da d/v , dove d è la distanza tra i due router, mentre v è la velocità di propagazione nel collegamento. Nelle reti molto estese i ritardi di propagazione sono dell'ordine dei millisecondi.

Per sottolineare la differenza c'è un'analogia tra il pacchetto e un gruppo di macchine che viaggiano insieme in un'autostrada. Il gruppo di 10 macchine equivale ai 10 bit inseriti in un pacchetto e devono essere inseriti nel tratto autostradale o nel link di uscita (analogia). Ho un tempo per servire una macchina, per immettere sul link di uscita quel particolare link. Una volta che ho immesso una macchina, ci mette un tot per arrivare all'altro casello stradale, questo dipende dalla velocità e dalla lunghezza del tratto autostradale.

Quanto ci mettono tutte le macchine ad arrivare all'altro casello? Prima devo immettere tutte le macchine (tutti i 10 bit sul link di uscita). Nel momento in cui immetto l'ultima macchina o bit ci vuole tempo per far sì che l'ultima macchina passi all'altro casello autostradale. Se cambiano le distanze, le velocità, le situazioni cambiano sicuramente.

Chiariamo meglio la situazione (un'autostrada, dove i tratti (di 100 km) tra un casello e l'altro corrispondono ai collegamenti e i caselli ai router). Si supponga:

- (1) che le automobili viaggino (ossia si propagano) a una velocità di 100 km/h (in altre parole, quando un'auto lascia un casello, accelera istantaneamente fino a 100 km/h e mantiene costantemente tale velocità);
- (2) che dieci automobili, accodate, procedano in ordine fisso (possiamo vedere ogni auto come un bit e l'insieme dei veicoli come un pacchetto);
- (3) che ciascun casello sia in grado di far transitare (ossia trasmettere) un'auto ogni dodici secondi (ovvero, 1 auto al minuto), e che queste siano le sole a percorrere in quel momento l'autostrada;
- (4) che la prima auto, una volta raggiunto il casello, prima di superarlo, attenda che gli altri nove veicoli siano allineati dietro di essa. Quindi, tutte le automobili devono raggiungere il casello prima di poter ripartire. Il tempo impiegato dal casello per far passare l'intera fila di macchine, che è di 10 auto / (5 auto/minuto) = 2 minuti, equivale al ritardo di trasmissione in un router. Il tempo richiesto a un'auto per spostarsi dall'uscita di un casello fino al casello successivo, pari a 100 km / (100 km/h) = 1 ora, corrisponde al ritardo di propagazione. Di conseguenza, il tempo che intercorre da quando l'intera coda di vetture si trova di fronte al casello di partenza fino al momento in cui raggiunge quello successivo è la somma del ritardo di trasmissione e del ritardo di propagazione, in questo caso 62 minuti.

Approfondiamo l'analisi dell'analogia. Che cosa succederebbe se il tempo di transito ai caselli fosse superiore al tempo richiesto a un'auto per spostarsi da un casello all'altro? Si supponga, per esempio, che le auto viaggino alla velocità di 1000 km/h e che il casello faccia passare le auto alla velocità di una al minuto. In questo caso il ritardo di viaggio tra due caselli sarebbe di 6 minuti, mentre il tempo per far defluire l'intera coda di auto sarebbe di 10 minuti, per cui le prime auto della fila arriverebbero al secondo casello prima che le ultime abbiano lasciato il primo. Questa situazione si riscontra anche nelle reti a commutazione di pacchetto: i primi bit di un pacchetto possono pervenire al router successivo mentre molti dei restanti bit del pacchetto sono ancora in attesa di essere trasmessi dal router precedente.

Siano d_{elab} , d_{acc} , d_{trasm} e d_{prop} i ritardi di elaborazione, accodamento, trasmissione e propagazione; il ritardo totale di nodo è allora:

$$d_{nodo} = d_{elab} + d_{acc} + d_{trasm} + d_{prop}$$

Il contributo di queste componenti del ritardo può variare in modo significativo.

Per esempio, d_{prop} può essere trascurato (pochi microsecondi) per un collegamento che connette due router nello stesso campus universitario; è, invece, di centinaia di millisecondi per due router interconnessi tramite un satellite geostazionario e può risultare il termine dominante in d_{nodo} .

Anche il d_{trasm} può essere insignificante o molto importante.

Il suo contributo è in genere trascurabile per velocità trasmissive di 10 Mbps o superiori (come avviene nelle LAN); invece, può risultare di centinaia di millisecondi per grandi pacchetti Internet inviati su collegamenti a bassa velocità con modem dialup. Il ritardo di elaborazione, d_{elab} , è spesso trascurabile, ma può influenzare pesantemente il throughput massimo di un router, che rappresenta la velocità massima alla quale il router può inoltrare i pacchetti.

Ritardo di accodamento.

La componente più complessa e interessante del ritardo totale di nodo è il **ritardo di accodamento**, d_{acc}

A differenza degli altri tre ritardi questo varia da pacchetto a pacchetto.

Per esempio, se in una coda vuota arrivano 10 pacchetti contemporaneamente, il primo pacchetto trasmesso non subirà ritardo di accodamento, mentre l'ultimo subirà un ritardo di accodamento piuttosto grande (dovendo attendere la trasmissione dei restanti 9 pacchetti). Pertanto, nel caratterizzare il ritardo di accodamento, si fa uso solitamente di **misure statistiche**, quali il ritardo di accodamento medio, la varianza del ritardo di accodamento e la probabilità che il ritardo di accodamento superi un valore fissato.

Quando si considera rilevante e quando invece trascurabile il ritardo di accodamento?

- **a** la **velocità media di arrivo dei pacchetti nella coda**, espressa in pacchetti al secondo.
- **R** è la **velocità di trasmissione**, ossia la velocità (bit/secondo) alla quale i bit vengono trasmessi in uscita dalla coda.
- tutti i **pacchetti** consistano di **L bit**.

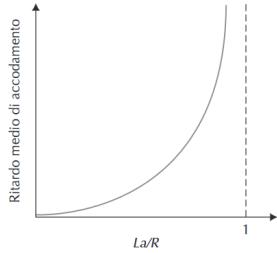
Quindi, la velocità media di arrivo dei bit in coda è di L_a bit/s. Infine, assumiamo che la coda possa mantenere un numero illimitato di bit.

Casi di L_a/R , detto intensità di traffico:

- **$L_a/R > 1$** allora la velocità media di arrivo dei bit supera la velocità alla quale vengono ritrasmessi in uscita;

In questa situazione la coda tenderà a crescere all'infinito e con essa il ritardo di accodamento.

- **$L_a/R \leq 1$** la velocità in uscita è più alta di quella in entrata e sul ritardo di accodamento influenza la natura del traffico. Se l'intensità di traffico è vicina a zero, allora gli arrivi di pacchetti sono pochi e piuttosto distanziati (poco ritardo di accodamento) mentre se è vicino ad 1 il tasso di arrivo dei pacchetti è molto alto e si forma facilmente una coda (ritardo di accodamento aumenta).



Un fondamentale aspetto evidenziato nella Figura 1.18 è che quanto più l'intensità di traffico si avvicina a 1, tanto più rapidamente cresce il ritardo medio di accodamento. Ossia, un piccolo incremento percentuale nell'intensità ha come risultato un incremento molto più accentuato nel ritardo. Forse avete sperimentato lo stesso fenomeno in autostrada: una strada che risulta sempre trafficata ha intensità di traffico vicina a 1. Se qualche evento causa un lieve incremento del traffico rispetto alla norma, il ritardo che subirete può diventare enorme.

Il grafico mostra come in funzione dell'intensità di traffico cresca il ritardo di coda.

Perdita di pacchetti.

Le **code** hanno **capacità finita**, sebbene le capacità di accodamento dipendano fortemente dalla struttura del router e dal suo costo. Poiché la **capacità delle code è finita** un pacchetto può trovare la coda piena. Non essendo possibile memorizzare tale pacchetto, il **router lo eliminerà** e il **pacchetto andrà perduto**.

La perdita sembrerà come se il pacchetto fosse stato inviato in rete, ma non fosse più riemerso alla destinazione.

Il pacchetto perso può essere ritrasmesso dal nodo precedente, dal sistema finale di origine o non ritrasmesso affatto.

Questo fenomeno, il fatto che si perde perché la coda è piena di pacchetti, è un problema che ci fa perdere il pacchetto in mezzo alla rete. Il fenomeno si chiama *congestione*.

Possono essere poi applicate diversi protocolli per garantire o meno l'arrivo di tutti i pacchetti (ritrasmissione, ricostruzione, ecc.).



Traceroute.

Per ottenere una **misura efficiente dei ritardi in una rete di calcolatori** possiamo fare uso del programma diagnostico Traceroute. Si tratta di un semplice programma eseguibile su qualsiasi host di Internet. Quando l'utente specifica il nome di un host di destinazione, il modulo dell'utente invia un certo numero di pacchetti speciali verso tale destinazione. Durante il loro percorso verso la destinazione, questi pacchetti passano attraverso una serie di router. Quando un router riceve uno di questi pacchetti speciali, invia un breve messaggio che torna all'origine. Il messaggio contiene il nome e l'indirizzo del router.

È in grado di determinare i ritardi di andata e ritorno per tutte le tratte.

```
traceroute: gaia.cs.umass.edu to www.eurecom.fr
Three delay measurements from
gaia.cs.umass.edu to cs-gw.cs.umass.edu
1 cs-gw (128.119.240.254) 1 ms 1 ms 2 ms
2 border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145) 1 ms 1 ms 2 ms
3 cht-vbns.gw.umass.edu (128.119.3.130) 6 ms 5 ms 5 ms
4 jnt-a1-0-0-19.wor.vbns.net (204.147.132.129) 16 ms 11 ms 13 ms
5 jnt-s07-0-0-0.wae.vbns.net (204.147.136.136) 21 ms 18 ms 18 ms
6 abilene-vbns.abilene.routinator.net (196.52.136.9) 2 ms 18 ms 22 ms
7 myriavbns.myriavbns.ucar.colorado.edu (194.32.8.46) 22 ms 18 ms 22 ms
8 62.40.103.253 (62.40.103.253) 104 ms 109 ms 106 ms
9 de2-1-de1.de1.geant.net (62.40.96.129) 109 ms 102 ms 104 ms
10 de1-fr1.fr1.geant.net (62.40.96.50) 113 ms 121 ms 114 ms
11 renater-gw1.fr1.geant.net (62.40.103.54) 112 ms 114 ms 112 ms
12 nio-n2cse.renater.fr (193.51.203.13) 111 ms 116 ms
13 p2p-eu.renater.fr (193.51.203.10) 109 ms 125 ms 124 ms
14 p2p-eu.renater.fr (193.51.203.12) 109 ms 126 ms 124 ms
15 1212-nice.casi.renater.fr (195.220.98.110) 126 ms 128 ms 124 ms
16 eurecom-vybonne.r32.ft.net (193.48.50.54) 135 ms 128 ms 133 ms
17 194.214.211.25 (194.214.211.25) 126 ms 128 ms 128 ms
18 *** * means no reponse (probe lost, router not replying)
19 fantasia.eurecom.fr (193.55.113.142) 132 ms 128 ms 136 ms
```

Name and address of router, round trip delays (3 samples)

Introduction 1-

Percorso dall'origine gaia.cs.umass.edu (presso l'University of Massachusetts) alla destinazione cis.poly.edu (presso la Polytechnic University of Brooklyn).

L'output presenta **sei colonne**:

la **prima** contiene il valore n descritto precedentemente, ossia il numero del router lungo il percorso;

la **seconda** è il nome del router;

la **terza** colonna è il suo indirizzo (nella forma xxx.xxx.xxx.xxx);

le ultime tre colonne rappresentano i ritardi di andata e ritorno nelle tre prove dell'esperimento.

Se l'origine riceve meno di tre messaggi da ogni dato router (per la perdita di pacchetti nella rete), Traceroute pone un asterisco subito dopo il numero del router e riporta meno di tre tempi di andata e ritorno per tale router.

Nel percorso seguito abbiamo nove router tra la sorgente e la destinazione. La maggior parte di essi ha un nome e tutti hanno un indirizzo. Per esempio, il nome del Router 3 è border4-rt-gi-1-3.gw.umass.edu e il suo indirizzo è 128.119.2.194.

Analizzando i dati forniti per questo stesso router, vediamo che nella prima delle tre prove il ritardo di andata e ritorno tra la sorgente e il router è stato di 1,03 ms. I ritardi di andata e ritorno per le successive due prove sono stati di 0,48 e 0,45 ms. Questi ritardi includono tutte le componenti di ritardo appena trattate, compresi i ritardi di trasmissione, di propagazione, di elaborazione e di accodamento. Dato che il ritardo di accodamento varia con il tempo, il ritardo di andata e ritorno del pacchetto n inviato

al router n può in realtà essere maggiore rispetto al ritardo di andata e ritorno del pacchetto n + 1 inviato al router n + 1. Infatti, nell'esempio si può notare che i ritardi nel Router 6 appaiono superiori ai ritardi nel Router 7.

Throughput.

Oltre al ritardo e alla perdita di pacchetti, un'altra misura critica delle prestazioni in una rete di calcolatori è il throughput end-to-end.

Il **throughput cattura quanto riusciamo a utilizzare in maniera efficace le capacità dei link attraversati sulla rete** e

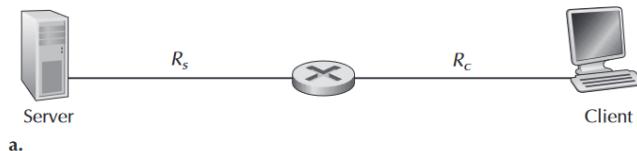
indica quant'è la **capacità del tubo che mi permette di veicolare le informazioni fra la sorgente e la destinazione relativa al flusso di traffico**.

Per darne una definizione, considerate il trasferimento di un file voluminoso da A a B, attraverso la rete. Questo file potrebbe essere un grosso videoclip da trasmettere da un peer a un altro in un sistema di condivisione di file P2P.

Il **throughput istantaneo** in ogni istante di tempo è la velocità (in bps) alla quale B sta ricevendo il file.

Se il file consiste di F bit e il trasferimento richiede T secondi affinché B riceva tutti gli F bit, allora il **throughput medio** del trasferimento del file è di F/T bps. Per alcune applicazioni, come la telefonia su Internet, è auspicabile avere un ritardo basso e un throughput istantaneo sopra una

certa soglia in modo continuativo (per esempio sopra i 24 kbps per alcune applicazioni di telefonia su IP e oltre i 256 kbps per alcune applicazioni video in tempo reale).



Ci sono due sistemi periferici, un server e un client, connessi da due collegamenti e da un router. Si consideri il throughput per un trasferimento di file dal server al client.

Sia R_s la *velocità del collegamento tra il server e il router* e R_c quella del collegamento tra il router e il client.

Qual è il throughput tra server e client? Pensiamo a un bit come a un fluido e ai collegamenti come a delle condotte.

Chiaramente il server non può pompare nel suo collegamento a una velocità maggiore di R_s bps e il router non può inoltrare bit a una velocità più alta di R_c bps.

Se $R_s < R_c$ i bit immessi dal server scorreranno attraverso il router e arriveranno al client a una velocità di R_s bps, dando un **throughput di R_s bps**.

Se, dall'altro lato, $R_c < R_s$ allora il router non sarà in grado di inoltrare i bit alla stessa velocità alla quale li riceve. In tal caso, i bit lasceranno il router a una velocità di R_c , dando un **throughput end-to-end di R_c** .

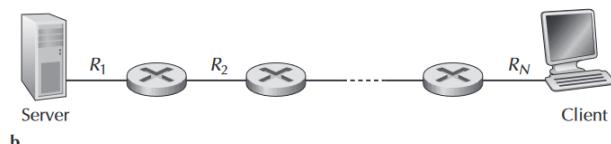
Si noti poi che se i bit continuano ad arrivare al router a una velocità R_s e a lasciarlo a una velocità di R_c , la quantità di bit accumulata al router in attesa di trasmissione al client cresce indefinitamente

Quindi, per questa semplice rete con due collegamenti, il throughput è il min (R_c, R_s), cioè la velocità di trasmissione del collegamento che fa da collo di bottiglia (**bottleneck link**).

Avendo determinato il throughput, possiamo ora stimare il tempo necessario a trasferire un grosso file di F bit dal server al client come $F / \min(R_s, R_c)$. Consideriamo un esempio specifico: supponete di stare scaricando un file MP3 di $F = 32$ milioni di bit.

Il server ha una velocità di trasmissione $R_s = 2$ Mbps e voi avete un collegamento di accesso di $R_c = 1$ Mbps. Il tempo necessario a trasferire il file è allora di 32 secondi.

Queste espressioni del throughput e del tempo di trasferimento sono solo approssimazioni, in quanto non tengono conto dei ritardi di elaborazione e store-and-forward e di quelli legati ai protocolli.



Se il file consiste di F bit e il trasferimento richiede T secondi affinché B riceva tutti gli F bit, allora il throughput medio del trasferimento del file è di F/T bps.



Ci sono 10 server e 10 client collegati al nucleo della rete. In questo esempio stanno avvenendo 10 download contemporanei, che coinvolgono 10 coppie Client-Server. Si supponga che questi 10 download siano il solo traffico sulla rete in questo momento. C'è un collegamento nel nucleo della rete che viene attraversato da tutti i 10 download.

R : velocità di trasmissione di questo collegamento. Supponiamo che tutti i collegamenti di accesso ai server abbiano la stessa velocità R_s , che tutti i collegamenti di accesso ai client abbiano la stessa velocità R_c e che le velocità di trasmissione di tutti i collegamenti del nucleo – eccetto l'unico collegamento comune a velocità R – siano molto più alte di R_s ed R_c . Ci chiediamo ora quali siano i throughput dei download. Chiaramente, se la velocità del collegamento comune R è grande, diciamo un centinaio di volte più grande di R_s e R_c , allora il throughput di ogni download sarà ancora una volta $\min(R_s, R_c)$. Ma che cosa accade se la velocità del collegamento comune è dello stesso ordine di grandezza di R_s e R_c ? Come sarebbe il throughput adesso? Supponete che $R_s = 2$ Mbps, $R_c = 1$ Mbps e $R = 5$ Mbps e che il collegamento comune suddivida la propria velocità di trasmissione equamente tra i 10 download. Quindi, il collo di bottiglia di ciascun download non è più nella rete di accesso, ma è invece il collegamento condiviso nel nucleo, che fornisce solo 500 kbps di throughput a ciascun download. Pertanto, il throughput end-to-end di ciascun download è ora ridotto a 500 kbps.

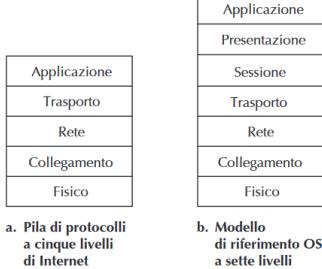
Quando non c'è altro traffico che interviene, il throughput può essere semplicemente approssimato alla velocità di trasmissione minima lungo il percorso tra la sorgente e la destinazione.

Il throughput dipende non solo dalla velocità di trasmissione dei collegamenti lungo il percorso, ma anche dal traffico sulla rete. In particolare, un collegamento con una velocità di trasmissione buona potrebbe essere il collo di bottiglia per un trasferimento di file, qualora molti altri flussi di dati passassero anch'essi attraverso quel collegamento.

Architettura a livelli.

Esiste una qualche speranza di organizzare l'architettura delle reti?

Un'architettura a livelli consente di definire singole parti specifiche e ben definite di un sistema articolato e complesso e rende inoltre possibile cambiare l'implementazione del servizio fornito da un determinato livello.



Ciascun protocollo di rete è dunque organizzato in **livelli o strati** che offrono dei servizi ai livelli superiori.
Nel caso di sistemi distribuiti, un protocollo di livello n si trova appunto distribuito tra i diversi end-system.
L'insieme dei protocolli dei vari livelli è detto **pila di protocolli** (protocol stack) e per Internet consiste di cinque livelli, anche se il modello di riferimento proposto da OSI ne consta sette.

Livello di applicazione.

Il **livello di applicazione** (*application layer*) è la sede delle applicazioni di rete e dei relativi protocolli.

Un protocollo a livello di applicazione è distribuito su più sistemi periferici: un'applicazione in un sistema periferico, tramite il protocollo, **scambia pacchetti di informazioni con l'applicazione in un altro sistema periferico**. È come se si scambiassero **MESSAGGI** direttamente, anche se il pacchetto passa attraverso tutti i livelli

Per quanto riguarda Internet, tale livello include molti protocolli, quali HTTP (richiesta e il trasferimento dei documenti web), SMTP (trasferimento dei messaggi di posta elettronica) e FTP (trasferimento di file tra due sistemi remoti).

Determinate funzioni di rete, quali la traduzione di nomi di host (per esempio, www.ietf.org) in indirizzi di rete a 32 bit, vengono anch'esse effettuate con l'aiuto di un protocollo a livello di applicazione, il DNS (domain name system).

Livello di trasporto.

Il **livello di trasporto** (*transport layer*) di Internet **trasferisce i messaggi del livello di applicazione tra punti periferici gestiti dalle applicazioni**.

Il transport layer fornisce un canale logico-affidabile di comunicazione end-to-end per pacchetti chiamati in questo livello **SEGMENTI**. Tra le funzionalità offerte possiamo trovare:

- un servizio che stabilisce una connessione persistente all'host, e la chiude quando non necessaria;
- verifica dell'ordine di consegna ed eventuale riordinamento;
- verifica delle perdite ed eventuale ritrasferimento dei pacchetti;
- controllo di flusso (sincronizzazione delle trasmissioni in caso di velocità di trasmissioni diverse);
- controllo di congestione che riconosce lo stato di congestione e adatta di conseguenza la velocità;
- multiplazione che permette di stabilire diverse connessioni contemporaneamente tra due stessi host

In Internet troviamo due protocolli di trasporto: **TCP** e **UDP**.

- **TCP** fornisce alle applicazioni un servizio orientato alla connessione, che include la consegna garantita dei messaggi a livello di applicazione alla destinazione e il controllo di flusso (ossia la corrispondenza tra le velocità di mittente e destinatario). Inoltre, TCP fraziona i messaggi lunghi in segmenti più piccoli e fornisce un meccanismo di controllo della congestione, in modo che una sorgente regoli la propria velocità trasmissiva quando la rete è congestionata.

- Il protocollo **UDP** fornisce alle proprie applicazioni un servizio non orientato alla connessione che è davvero un servizio senza fronzoli, senza affidabilità, né controllo di flusso e della congestione. Nel testo chiameremo segmenti i pacchetti a livello di trasporto.

Livello di rete.

Il **livello di rete** (*network layer*) di Internet **si occupa di trasferire i pacchetti a livello di rete**, detti **datagrammi**, da un host a un altro che in generale non sono direttamente connessi.

Il livello di rete mette poi a disposizione il servizio di consegna del segmento al livello di trasporto nell'host di destinazione.

In sostanza si occupa di **instradamento e indirizzamento verso la giusta destinazione attraverso il path di rete più appropriato**. Tra le funzionalità di questo livello troviamo:

- **Inoltro** (forwarding) ovvero ricevere un pacchetto su una porta, immagazzinarlo e ritrasmetterlo;
- **Frammentazione e riassemblaggio dei pacchetti**;
- **Instradamento** (routing) verso il percorso ideale tramite algoritmi dinamici che contengono informazioni sulle condizioni della rete, le tabelle di instradamento e le priorità del servizio;

Livello di collegamento.

Il **livello di rete di Internet** **instrada un datagramma attraverso una serie di router tra la sorgente e la destinazione**. Per trasferire un pacchetto da un nodo (host o router) a quello successivo sul percorso, il livello di rete si affida ai servizi del livello di collegamento. In particolare, a ogni nodo, il livello di rete passa il datagramma al livello sottostante, che lo trasporta al nodo successivo. In questo nodo, il livello di collegamento passa il datagramma al livello di rete superiore, spostare interi frame da un elemento della rete a quello adiacente.

Deve svolgere diverse funzioni:

- In trasmissione **raggruppare i bit provenienti dal livello rete in frame e mandarli al livello fisico**;
- In ricezione **controllare e gestire gli errori di trasmissione**
- Controllo di flusso
- Operare una multiplazione per l'accesso condiviso dello stesso canale fisico

Questo livello in effetti fa apparire in ricezione al livello fisico un flusso di pacchetti esenti da errori. Esempi

di protocolli che fanno parte di questo livello sono Ethernet, Wi-Fi e il protocollo di accesso alla rete DOCSIS. Dato che un pacchetto (datagramma) in genere attraversa diversi collegamenti tra la sorgente e la destinazione, questo potrebbe essere gestito da protocolli differenti lungo il suo tragitto.

Livello fisico.

Il ruolo del **livello fisico** (physical layer) è **trasferire i singoli bit del frame da un nodo a quello successivo**. Anche i protocolli di questo livello sono dipendenti dal collegamento e in più dipendono dall'effettivo mezzo trasmissivo (per esempio, doppino o fibra ottica). Gli standard (più che protocolli) che fanno parte di questo livello definiscono:

- Le caratteristiche fisiche del mezzo trasmissivo (forma, dimensioni, numero pin)
- Le caratteristiche funzionali (significato dei pin);
- Le caratteristiche elettriche (valori tensione per i livelli logici, la codifica e la durata di ogni bit);
- La codifica del segnale digitale su un mezzo trasmissivo che è analogico.

MODELLO OSI.

Il modello **OSI** (Open System Interconnection) proposto da ISO negli anni '70 prevedeva sette livelli ed aggiungeva allo stack di livelli già visti per Internet, il livello di presentazione e quello di sessione.

- Il livello di **presentazione** fornisce servizi che permettono ad applicazioni che vogliono comunicare di interpretare il significato dei dati scambiati. Comprende compressione e cifratura dati.
- Il livello di **sessione** fornisce la delimitazione e la sincronizzazione dello scambio dei dati, compresi i mezzi per costruire uno schema di controllo e di recupero degli stessi.

In internet questi due livelli sono incorporati nel livello applicazione e nel livello di trasporto per ridurre la complessità.

Implementazione livelli

Soltamente i router e i commutatori non implementano tutti i livelli della pila di protocolli ma solo quelli inferiori. I **commutatori a livello di collegamento** implementano i **livelli 1 e 2**, i **router** implementano i **livelli da 1 a 3**.

Gli **host** invece ragionevolmente implementano **tutti e cinque i livelli**. Ciò è coerente con l'idea che l'architettura Internet ponga la maggior parte della sua complessità alla periferia della rete.

INCAPSULAMENTO (imbustamento)

Presso un host mittente, un messaggio a livello di applicazione M viene passato al livello di trasporto.

Questo gli concatena informazioni aggiuntive (informazioni di intestazione di livello) H che saranno utilizzate e scorporate nel livello di trasporto dell'host ricevente. Il messaggio del livello di applicazione concatenato con le informazioni di intestazione del trasporto costituiscono il segmento del livello di trasporto, che di fatto incapsula il messaggio del livello di applicazione. Questo meccanismo di passaggio e arricchimento si intera per ogni livello, andando così a formare un datagramma a livello di rete, un frame a livello di collegamento.

Finita la fase di arricchimento ed arrivati al livello fisico, questo si collega con il livello fisico dell'host ricevente che inizia la fase di impoverimento, scorporando ad ogni livello l'header relativo e passando il payload al livello superiore.

Dunque, a ciascun livello il pacchetto ha due tipi di campi: un header, aggiunto dal livello stesso, e un payload proveniente dal livello superiore

Storia internet.

1961-1972: Early packet-switching principles

- 1961: Kleinrock - queueing theory shows effectiveness of packet-switching (MIT)
- 1964: Baran - packet-switching in military nets
- Davies at the National Physical Laboratory, UK was also developing ideas on packet switching
- 1967: ARPAnet conceived by Advanced Research Projects Agency
- 1969: first ARPAnet node operational
- Packet switches dubbed Interface Message Processors (IMP)

- 1972:
 - ARPAnet demonstrated publicly by Robert Kahn
 - NCP (Network Control Protocol) first host-host protocol
 - first e-mail program
 - ARPAnet has 15 nodes



Kleinrock's students:
Vinton Cerf
John Postel...

Network measurement center UCLA

1972-1980: Internetworking, new and proprietary nets

- 1970: ALOHAnet satellite network in Hawaii (Abramson)
- 1973: Metcalfe's PhD thesis proposes Ethernet
- 1974: Cerf and Kahn - architecture for interconnecting networks
- late 70's: proprietary architectures, e.g. IBM SNA (Schwartz)
- late 70's: switching fixed length packets (ATM precursor)
- 1979: ARPAnet has 200 nodes

Cerf and Kahn's internetworking principles:

- minimalism, autonomy - no internal changes required to interconnect networks
 - best effort service model
 - stateless routers
 - decentralized control
- define today's Internet architecture**

1980-1990: new protocols, a proliferation of networks

- ❑ 1983: deployment of TCP/IP
- ❑ 1982: SMTP e-mail protocol defined
- ❑ 1983: DNS defined for name-to-IP-address translation
- ❑ 1985: FTP protocol defined
- ❑ 1988: TCP congestion control

- ❑ new national networks: Cernet, BITnet, NSFnet, Minitel
- ❑ 100,000 hosts connected to confederation of networks

1990, 2000's: commercialization, the Web, new apps

- ❑ Early 1990's: ARPAnet decommissioned
- ❑ 1991: NSF lifts restrictions on commercial use of NSFnet (decommissioned, 1995)
- ❑ early 1990s: Web
 - hypertext [Bush 1945, Nelson 1960's]
 - HTML, HTTP: Berners-Lee
 - 1994: Mosaic, later Netscape
 - late 1990's: commercialization of the Web

- ❑ Late 1990's - 2000's:
 - ❑ more killer apps: instant messaging, peer2peer file sharing (e.g., Napster)
 - ❑ network security to forefront
 - ❑ est. 50 million host, 100 million+ users
 - ❑ backbone links running at Gbps

Significant late developments: P2P, broadband access, wireless Internet

CENNI SULLA SICUREZZA.

- **Malware**: indica qualsiasi programma usato per disturbare le operazioni svolte dal computer, rubare informazioni sensibili, accedere a sistemi privati, mostrare pubblicità indesiderata. Il modo di propagazione consiste di frammenti di software parassiti che si inseriscono in codice eseguibile già esistente;
- **Botnet**: i malware possono essere anche auto replicanti, nel senso che una volta che hanno infettato un host, cercano riferimenti ad altri host ed usano l'host iniziale per infettarne altri. Si possono diffondere a velocità esponenziali creando vere e proprie reti di Botnet;
- **Virus**: sono malware che richiedono una forma di interazione con l'utente per infettarne il dispositivo;
- **Worm**: sono malware che possono entrare in un dispositivo senza alcuna interazione esplicita con l'utente;
- **Negazione del servizio (DoS)**: è un attacco che rende inutilizzabile dagli utenti legittimi una rete, un host o un'altra parte di infrastruttura. Possono essere di tipo Bandwidth flooding (l'attaccante invia un'enorme quantità di pacchetti), Connection flooding (l'attaccante stabilisce un gran numero di connessioni TCP), oppure rivolte alle vulnerabilità dei sistemi.
- **Analisi del traffico**: è detto packet sniffer un ricevitore passivo che memorizza una copia di ciascun pacchetto che transita su una rete, che sia LAN o WLAN. Il lavoro oneroso non è tanto sniffare, quanto estrapolare o ricostruire i dati a livello applicativo. Una delle migliori difese dunque è la crittografia.
- **Mascheramento**: IP spoofing è la capacità di immettere in Internet pacchetti con un indirizzo sorgente falso "fatti a mano". Per prevenire questo tipo di attacco sono applicati meccanismi di end-point authentication.

Capitolo 2.

Livello di applicazione.

Il **livello applicativo** di un'applicazione di rete **consiste** nella codifica dei programmi distribuiti sui sistemi periferici che comunicano tra loro. Il cuore dello sviluppo delle applicazioni di rete è costituito dalla compilazione dei programmi che sono eseguiti dai sistemi periferici e che comunicano tra loro via rete.

Lo sviluppatore prima di procedere alla codifica vera e propria delle applicazioni deve progettare l'**architettura dell'applicazione** e stabilire l'organizzazione sui vari sistemi periferici, basandosi, probabilmente, su una delle due principali architetture di rete attualmente utilizzate: l'**architettura Client-Server** o **p2p**.

Nell'**architettura client-server** vi è un host sempre attivo, chiamato **server**, che *risponde alle richieste di servizio* di molti altri host, detti **client**.

Un processo **client** è solo in grado fare **richieste di servizio** (informazioni) e di **interpretare le risposte**.

Un processo **server** ha solo il compito di **interpretare le richieste e fornire le risposte**.

Messaggi di richiesta (request) generati dal lato client e messaggi di risposta (response) generati dal server.

Per esempio, nelle applicazioni web esistono due programmi diversi che comunicano tra di loro: il browser, che viene eseguito dall'host dell'utente (desktop, laptop, PDA, telefono cellulare e così via) e il web server, sempre attivo, risponde alle richieste del browser in funzione sui client.

Il client può quindi contattare il server in qualsiasi momento, inviandogli un pacchetto.

Tra le più note applicazioni con architettura client-server, ricordiamo il Web, il trasferimento dei file con FTP, Telnet e la posta elettronica.

Spesso in un'applicazione client-server un singolo host che esegue un server non è in grado di rispondere a tutte le richieste dei suoi client. Per esempio, una social network con molti iscritti potrebbe rapidamente soccombere qualora si basasse su un solo server per gestire tutte le richieste. Per questo motivo nelle architetture client-server si usano spesso **data center** che, ospitando molti **host** creano un potente **server virtuale**. (Google, Bing e Baidu, Amazon, eBay e Alibaba, Gmail e Yahoo Mail, Facebook, Instagram, Twitter e WeChat)

Un data center può ospitare fino a centinaia di migliaia di server a cui deve essere fornita alimentazione e manutenzione.

Processi comunicanti.

Come comunicano tra loro i programmi in esecuzione su diversi sistemi terminali?

Nel gergo dei sistemi operativi non si parla in effetti di programmi, ma di processi comunicanti.

Si può pensare a un processo come a un programma in esecuzione su un sistema. Processi in esecuzione sullo stesso sistema comunicano utilizzando un approccio interprocesso (**interprocess communication**).

I **processi su due sistemi terminali comunicano scambiandosi messaggi attraverso la rete**: il processo mittente crea e invia messaggi nella rete e il processo destinatario li riceve e, quando previsto, invia messaggi di risposta.

I processi comunicano utilizzando il livello di applicazione della **pila a cinque livelli** dei protocolli Internet.

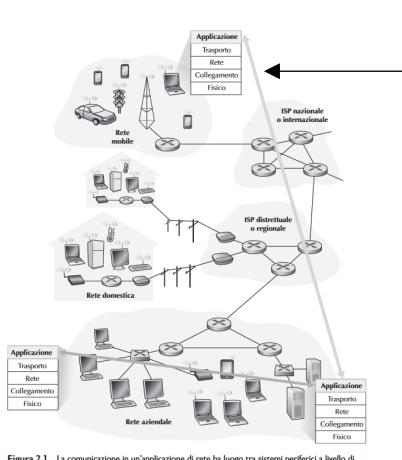


Figura 2.1 La comunicazione in un'applicazione di rete ha luogo tra sistemi periferici a livello di applicazione.

Processi client e server.

Le **applicazioni di rete** sono **costituite** da una coppia di processi che si scambiano messaggi su una rete.

Per ciascuna coppia di processi comunicanti, generalmente ne etichettiamo uno come **client** e l'altro come **server**.

In alcune applicazioni, come quelle di condivisione dei file via P2P, un processo può essere sia client sia server, in quanto può tanto inviare quanto ricevere file.

L'interfaccia tra il processo e la rete.

I client non comunicano direttamente tra loro; così, in una applicazione web, i browser non interagiscono direttamente tra loro.

Ogni messaggio inviato da un processo a un altro deve passare attraverso la rete sottostante. Un processo invia messaggi nella rete e riceve messaggi dalla rete attraverso un'**interfaccia software** detta **socket**.

La Figura 2.3 mostra la comunicazione tra le socket di due processi che comunicano via Internet. Tale figura ipotizza che il protocollo di trasporto sottostante sia TCP.

Una **socket** è *l'interfaccia* tra un **processo applicativo** e il **protocollo a livello di trasporto** all'interno di un host. La socket rappresenta l'interfaccia di programmazione con cui le applicazioni di rete vengono costruite.

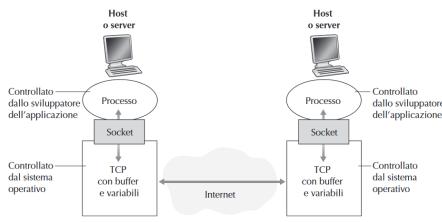


Figura 2.3 Processi, socket e protocollo di trasporto sottostante.

Indirizzamento.

Come nella posta tradizionale, affinché la consegna possa essere effettuata il **destinatario deve avere un indirizzo**. Anche in Internet i processi riceventi devono averne uno per ricevere i messaggi inviati da un processo in esecuzione su un altro host. Per identificare il processo ricevente, è necessario specificare due informazioni:

l'indirizzo dell'host e un **identificatore del processo** ricevente sull'host di destinazione.

In Internet, gli **host** vengono **identificati** attraverso i loro **indirizzi IP**, un numero di 32 bit che possiamo pensare identifichi univocamente l'host. Il mittente deve anche identificare il processo destinatario, più specificatamente la socket che deve ricevere il dato. Un **numero di porta** di destinazione assolve questo compito. Alle applicazioni più note sono stati assegnati numeri di porta specifici. Per esempio, i web server sono identificati dal numero di porta 80. Il processo di un server di posta che usa il protocollo SMTP è identificato dal numero di porta 25.

La lista di numeri di porta noti per tutti i protocolli standard Internet può essere reperita su <http://www.iana.org/>.

Servizi di trasporto disponibili per le applicazioni.

Nel progetto di un'applicazione occorre scegliere il **protocollo a livello di trasporto**.

Questi si possiamo classificare a grandi linee secondo quattro dimensioni: *trasferimento dati affidabile, throughput, temporizzazione e sicurezza*.

Servizi di trasporto offerti da Internet.

Internet (come ogni rete TCP/IP) mette a disposizione delle applicazioni **due protocolli di trasporto**: **UDP** e **TCP**. Quando gli sviluppatori di software realizzano una nuova applicazione di rete per Internet, scelgono tra TCP e UDP: due protocolli che offrono modelli di servizio diversi. Lo scambio di messaggi fra i processi applicativi avviene utilizzando i servizi dei livelli inferiori attraverso i SAP (Service Access Point).

Qualsiasi protocollo di livello applicativo accede al servizio di trasporto mediante socket.

Come si distingue a quale protocollo di trasporto passare il segmento? Protocol number

Servizi di TCP.

- **TCP** prevede un servizio **orientato alla connessione** e il **trasporto affidabile** dei dati.

Quando un'applicazione invoca TCP come protocollo di trasporto, riceve entrambi questi servizi.

Servizio orientato alla connessione. Client e server si scambiano informazioni di controllo a livello di trasporto prima che i messaggi a livello di applicazione comincino a fluire. Questa procedura, **handshaking**, mette in allerta client e server, preparandoli alla partenza dei pacchetti.

Dopo la fase di handshaking, si dice che esiste una connessione TCP tra le socket dei due processi.

- **Servizio di trasferimento affidabile**. I processi comunicanti possono contare su TCP per trasportare i **dati senza errori e nel giusto ordine**.

- TCP include anche un meccanismo di controllo della congestione, un servizio che permette di limitare la quantità di dati trasmessi, adattando il flusso dati inviato all'eventuale stato di congestione della rete.

I server che usano TCP vengono eseguiti in modalità parallela e sono dunque in grado di rispondere a più richieste insieme.

Servizi di UDP.

- **UDP** è un **protocollo di trasporto leggero e senza fronzoli**, dotato di un modello di servizio minimalista.

- UDP è **senza connessione**, non necessita quindi di handshaking.

- Fornisce un servizio di **trasferimento dati non affidabile**. Così, quando un processo invia un messaggio tramite la socket UDP, il protocollo non garantisce che questo raggiunga il processo di destinazione. Inoltre, i messaggi potrebbero giungere a destinazione non in ordine.

- UDP non include un meccanismo di controllo della congestione; pertanto, un processo d'invio UDP può "spingere" i dati al livello sottostante (di rete) a qualsiasi velocità.

Protocolli a livello di applicazione.

Un protocollo a livello di applicazione definisce come i processi di un'applicazione, in esecuzione su sistemi periferici diversi, si scambiano i messaggi. In particolare, un protocollo a livello di applicazione definisce:

- **i tipi di messaggi scambiati** (per esempio, di richiesta o di risposta);
- **la sintassi dei vari tipi di messaggio** (per esempio, quali sono i campi nel messaggio e come vengono descritti);
- **la semantica dei campi**, ossia il significato delle informazioni che contengono le regole per determinare quando e come un processo invia e risponde ai messaggi.

Alcuni protocolli a livello di applicazione vengono specificati nelle RFC e sono pertanto di pubblico dominio (HTTP (hypertext transfer protocol), Altri protocolli a livello di applicazione sono privati e volutamente non disponibili al pubblico (per esempio Skype).

È importante distinguere tra applicazioni di rete e protocolli a livello di applicazione. Un protocollo a livello di applicazione è solo una parte (benché molto importante) di un'applicazione di rete.

Web e http.

Nei primi anni '90, comparve una nuova importante applicazione: il World Wide Web [Berners-Lee 1994]. Il **Web** è stata la prima applicazione Internet che ha catturato l'attenzione del pubblico.

Forse ciò che attira di più gli utenti è che *il Web opera su richiesta (on demand): si può avere ciò che si vuole, quando lo si vuole.* Una delle caratteristiche principali della rete è che i nodi che la compongono sono tra loro collegati tramite i link (collegamenti ipertestuali) formando un enorme ipertesto.

Panoramica di http.

HTTP (hypertext transfer protocol) è protocollo a livello di applicazione del Web e costituisce il **cuore del Web**.

Questo protocollo è *implementato in due programmi, client e server*, in esecuzione su sistemi periferici diversi che comunicano tra loro scambiandosi messaggi HTTP.

Il protocollo definisce sia la struttura dei messaggi sia la modalità con cui client e server si scambiano i messaggi.

Prima di affrontare in dettaglio HTTP soffermiamoci brevemente sulla terminologia web.

Una **pagina web** (*web page*), detta anche documento, è costituita da oggetti.

Un oggetto è semplicemente un file (quale un file HTML, un'immagine JPEG, un applet Java, una clip video e così via) indirizzabile tramite un URL.

La maggioranza delle pagine web consiste di un file HTML principale e diversi oggetti referenziati da esso.

È una **risorsa** (cioè un file), specificata da un **URL**: *Uniform Resource Locator*.

URL è una *sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa su una rete di computer*, come ad esempio un documento, un'immagine, un video, tipicamente presente su un host server e resa accessibile a un client.

È perlopiù utilizzato per indicare risorse web (http), risorse recuperabili tramite protocolli di trasferimento file (ftp), condivisioni remote (smb) o accessi a sistemi esterni (ssh).

La **struttura di un URL** si compone normalmente di **sei parti**, alcune delle quali opzionali:

protocollo//[username[:password]@]host[:porta][</percorso>][?queryString][#fragment]

protocollo:

Identifica il **protocollo**, tipicamente di livello applicazioni, da utilizzare per l'accesso al server.

I protocolli più comuni sono l'HTTP, HTTPS, FTP, MMS ecc. Se il protocollo non viene specificato, generalmente il browser utilizza il protocollo "http" come predefinito.

// → Separatore tra il protocollo e il resto dell'URL: di solito l'host, o opzionalmente lo username.

username:password@ (opzionale):

Subito dopo il protocollo, è possibile specificare le credenziali di autenticazione (username e password) per l'accesso alla risorsa.

Host.

Identifica il server su cui risiede la risorsa. Può essere rappresentato direttamente da un indirizzo IP o (più comunemente) da un nome di dominio che il software converte in indirizzo IP avvalendosi del servizio DNS.

Porta. (opzionale)

Identifica la porta del servizio di rete al quale inoltrare la richiesta.

Percorso. (opzionale)

Percorso (pathname) nel file system del server che identifica la risorsa (generalmente una pagina web, una immagine o un file multimediale)

QueryString. (opzionale)

Se richiesto, al termine dell'url è possibile aggiungere una query string separandola con l'utilizzo del simbolo "?". La query string è una stringa di caratteri che consente di passare al server uno o più parametri.

Fragment. (opzionale)

Se presente, indica una parte o una posizione all'interno della risorsa, come la query string è possibile utilizzare più parametri, con la differenza che, essendo utilizzati dal client per sapere come muoversi all'interno di una risorsa non verranno inviati al server.

1. *Protocol (also called "scheme")*
 - o *how can a page be accessed? (application protocol used)*
 - <http://www.dsi.uniroma1.it/people/petrioli/index.html>

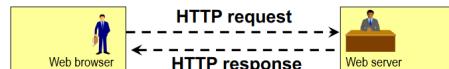
2. *Host name*
 - o *Where is the page located? (symbolic or numeric location)*
 - <http://WWW.dsi.uniroma1.it/people/petrioli/index.html>

3. *File (resource) name*
 - o *What is the page called? (with full path)*
 - <http://www.dsi.uniroma1.it/people/petrioli/index.html>



HTTP://cerbero.elet.polimi.it/people/bianchi/index.html

protocol location filename



Web browser (SW process) needs to send HTTP Request to Web Server.
Location not enough (it is just the computer address!)

Un browser web (come Internet Explorer o Firefox) implementa il lato client di HTTP (quando parliamo di Web useremo le parole browser e client in modo intercambiabile).

Un web server, che implementa il lato server di HTTP, ospita oggetti web, indirizzabili tramite URL.

HTTP definisce in che modo i client web richiedono le pagine ai web server e come questi ultimi le trasferiscono ai client.

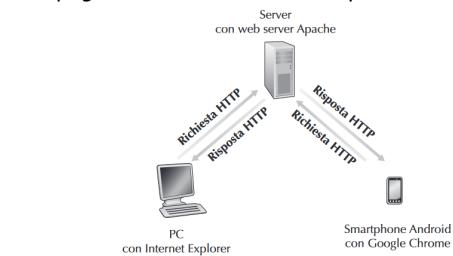


Figura 2.6 Comportamento richiesta-risposta di HTTP.

Quando l'**utente richiede una pagina web** (per esempio, cliccando su un collegamento ipertestuale), il **browser invia al server messaggi di richiesta HTTP per gli oggetti nella pagina**.

Il **server riceve le richieste e risponde con messaggi di risposta HTTP contenenti gli oggetti**.

- **HTTP utilizza come protocollo di trasporto TCP.** Quindi il client HTTP per prima cosa inizia una connessione TCP attraverso le proprie socket. Il client invia le richieste e riceve risposte http tramite la propria interfaccia socket, analogamente il server riceve richieste e invia messaggi di risposta attraverso la propria interfaccia di socket.
- HTTP non si preoccupa di dati smarriti nelle richieste, in quanto **TCP** mette a disposizione un servizio di **trasferimento dati affidabile**.
- In HTTP le connessioni vengono generalmente chiuse una volta che una richiesta è stata soddisfatta.
- HTTP è classificato come protocollo senza memoria di stato (stateless), ciò significa che il server non memorizza alcuna informazione di stato a proposito del client.

Per memorizzare le informazioni si utilizzano i **cookies**.

Connessioni persistenti e non persistenti.

In molte applicazioni per Internet, **client e server comunicano per un lungo periodo di tempo**, con il client che inoltra una serie di richieste e il server che risponde a ciascuna di esse.

A seconda dell'applicazione e del suo impiego la **serie di richieste potrebbe essere effettuata in sequenza, periodicamente a intervalli regolari o in maniera intermittente**.

Quando tale interazione client-server ha luogo su TCP, gli sviluppatori dell'applicazione devono prendere una decisione importante: ciascuna **coppia richiesta/risposta** deve essere inviata:

- Su una **connessione TCP separata (connessione non persistente)**
- Tutte sulla **stessa connessione TCP (connessioni persistenti)**

Http nella sua modalità di default usa connessioni persistenti.

HTTP con connessioni non persistenti.

Il **trasferimento di una pagina web dal server al client** nel caso di **connessioni non persistenti**.

Supponiamo che la pagina ha un file HTML principale e 10 immagini JPEG, e che tutti gli undici oggetti risiedano sullo stesso server. Ipotizziamo che l'URL del file HTML sia: <http://www.someSchool.edu/someDepartment/home.index>

Ecco che cosa avviene.

1. Il processo client HTTP inizializza una connessione TCP con il server www.someSchool.edu sulla porta 80.
Associate alla connessione TCP ci saranno una socket per il client e una per il server.

2. Il client HTTP, tramite la propria socket, invia al server un messaggio di richiesta HTTP che include il percorso </someDepartment/home.index>.

3. Il processo server HTTP riceve il messaggio di richiesta attraverso la propria socket associata alla connessione, recupera l'oggetto </someDepartment/home.index> dalla memoria (centrale o di massa), lo incapsula in un messaggio

di risposta HTTP che viene inviato al client attraverso la socket.

4.Il processo server HTTP comunica a TCP di chiudere la connessione. Questo, però, non termina la connessione finché non sia certo che il client abbia ricevuto integro il messaggio di risposta.

5.Il client HTTP riceve il messaggio di risposta. La connessione TCP termina. Il messaggio indica che l'oggetto incapsulato è un file HTML. Il client estrae il file dal messaggio di risposta, esamina il file HTML e trova i riferimenti ai 10 oggetti JPEG.

6.Vengono quindi ripetuti i primi quattro passi per ciascuno degli oggetti JPEG referenziati.

I passi appena riportati illustrano l'utilizzo di connessioni non persistenti, in cui ogni connessione TCP viene chiusa dopo l'invio dell'oggetto da parte del server: vale a dire che ciascuna trasporta soltanto un messaggio di richiesta e un messaggio di risposta. Pertanto, in questo esempio, quando l'utente richiede una pagina web, vengono generate **11 connessioni TCP**.

HTTP con connessioni persistenti.

Le connessioni **non** persistenti presentano alcuni limiti: il primo è che per ogni oggetto richiesto occorre stabilire e mantenere una nuova connessione. Per ciascuna di queste connessioni si devono allocare buffer e mantenere variabili TCP sia nel client sia nel server. Ciò pone un grave onere sul web server, che può dover servire contemporaneamente richieste provenienti da centinaia di diversi client. In secondo luogo, come abbiamo appena descritto, ciascun oggetto subisce un ritardo di consegna di due RTT, uno per stabilire la connessione TCP e uno per richiedere e ricevere un oggetto.

Con HTTP 1.1 nelle **connessioni persistenti** il server lascia la connessione TCP aperta dopo l'invio di una risposta, per cui le richieste e le risposte successive tra gli stessi client e server possono essere trasmesse sulla **stessa connessione**. In particolare, non solo il server può inviare un'intera pagina web (nell'esempio, il file HTML principale e le 10 immagini) su una sola connessione TCP permanente, ma può anche spedire allo stesso client più pagine web.

Queste richieste di oggetti possono essere effettuate una di seguito all'altra senza aspettare le risposte delle richieste pendenti (pipelining). In generale, il server HTTP chiude la connessione quando essa rimane inattiva per un dato lasso di tempo (un intervallo configurabile).

Formato dei messaggi http.

Le specifiche HTTP [RFC 1945; RFC 2616; RFC 7540] includono la definizione dei due **formati dei messaggi HTTP**, di richiesta e di risposta.

Messaggio di richiesta HTTP

Ecco un tipico messaggio di richiesta HTTP:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Notiamo innanzitutto che il messaggio è *scritto in testo ASCII*, in modo che l'utente sia in grado di leggerlo.

Inoltre, notiamo che consiste di cinque righe, ciascuna seguita da un carattere di ritorno a capo (carriage return) e un carattere di nuova linea (line feed).

L'ultima riga è seguita da una coppia di caratteri di ritorno a capo e nuova linea aggiuntivi.

La **prima riga** è detta riga di richiesta (**request line**) e quelle successive **righe di intestazione** (**header lines**).

La **riga di richiesta** presenta tre campi: il **campo metodo**, il **campo URL** e il **campo versione di HTTP**.

Il campo metodo può assumere diversi valori, tra cui GET, POST, HEAD, PUT e DELETE.

La maggioranza dei messaggi di richiesta HTTP usa il metodo GET, adottato quando il browser richiede un oggetto identificato dal campo URL.

La versione è auto esplicativa: nell'esempio, il browser, che implementa la versione HTTP/1.1, sta richiedendo l'oggetto /somedir/page.html.

Consideriamo ora le **righe di intestazione** dell'esempio.

- La **riga Host**: www.someschool.edu specifica l'**host** su cui risiede l'oggetto.

- La **linea di intestazione Connection**: close, il browser sta comunicando al server che non si deve occupare di connessioni persistenti, ma vuole che questi chiuda la connessione dopo aver inviato l'oggetto richiesto.

- La **riga di intestazione User-agent**: specifica il tipo di browser che sta effettuando la richiesta al server, in questo caso Mozilla/5.0, un browser Firefox.

- Infine, **Accept-language**: indica che l'utente preferisce ricevere una versione in francese dell'oggetto se disponibile; altrimenti, il server dovrebbe inviare la versione di default.

Campo Metodi.

-**GET**: server per richiedere una risorsa ad una server (solo header). Può richiedere una risorsa statica o dinamica (tramite query).

- **DELETE**: consente invece la cancellazione di un oggetto su un server.

GET	E' usato quando il client vuole scaricare un documento dal server. Il documento richiesto è specificato nell'URL. Il server normalmente risponde con il documento richiesto nel corpo del messaggio di risposta.
HEAD	E' usato quando il client non vuole scaricare il documento ma solo alcune informazioni sul documento (come ad esempio la data dell'ultima modifica). Nella risposta il server non inserisce il documento ma solo degli header informativi.
POST	E' usato per fornire degli input al server da utilizzare per un particolare oggetto (di solito un applicativo) identificato nell'URL.
PUT	E' utilizzato per memorizzare un documento nel server. Il documento viene fornito nel corpo del messaggio e la posizione di memorizzazione nell'URL.

Altri methods:
■DELETE, ■LINK, ■UNLINK, ...

Per inviare info al server
E' possibile anche usare GET
GET /search.cgi?string=greek-architects HTTP/1.0

Messaggio di risposta http.

Presentiamo ora un tipico messaggio di risposta HTTP che potrebbe rappresentare la **risposta** al messaggio di richiesta dell'esempio precedente.

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(data data data data data ...)
```

Analizzando in dettaglio questo messaggio di risposta, osserviamo tre sezioni: *una riga di stato iniziale*, *sei righe di intestazione* e *il corpo*. Il **corpo** contiene l'oggetto richiesto (rappresentato da: data data data data data ...).

La *riga di stato iniziale* presenta tre campi: la **versione del protocollo**, un **codice di stato** e un **corrispettivo messaggio di stato**.

In questo esempio, la riga di stato indica che il server sta usando HTTP/1.1 e che tutto va bene (ossia che il server ha trovato e sta inviando l'oggetto richiesto).

Osserviamo ora le *righe di intestazione*.

- **Connection:** close per comunicare al client che ha intenzione di chiudere la connessione TCP dopo l'invio del messaggio.
- **Date:** indica l'ora e la data di creazione e invio, da parte del server, della risposta HTTP.
- Si noti che non si tratta dell'istante in cui l'oggetto è stato creato o modificato per l'ultima volta, ma del momento in cui il server recupera l'oggetto dal proprio file system, lo inserisce nel messaggio di risposta e invia il messaggio.
- **Server:** indica che il messaggio è stato generato da un web server Apache; essa è analoga alla riga User-agent: nel messaggio di richiesta HTTP.
- **Last-Modified:** indica l'istante e la data il cui l'oggetto è stato creato o modificato per l'ultima volta, importante per la gestione dell'oggetto nelle cache, sia nel client locale sia in alcuni server in rete (proxy server o proxy).
- **Content-Length:** contiene il numero di byte dell'oggetto inviato.
- **Content-Type:** indica che l'oggetto nel corpo è testo HTML, cioè il tipo dell'oggetto

Il codice di stato e l'espressione associata indicano il risultato della richiesta. Tra i più comuni codici di stato e relative espressioni troviamo:

- 200 OK: la richiesta ha avuto successo e in risposta si invia l'informazione.
- 301 Moved Permanently: l'oggetto richiesto è stato trasferito in modo permanente; il nuovo URL è specificato nell'intestazione Location: del messaggio di risposta. Il client recupererà automaticamente il nuovo URL.
- 400 Bad Request: si tratta di un codice di errore generico che indica che la richiesta non è stata compresa dal server.
- 404 Not Found: il documento richiesto non esiste sul server.
- 505 HTTP Version Not Supported: il server non dispone della versione di protocollo HTTP richiesta.

■1xx Informational

Status codes:

■2xx Success

200 OK:	La richiesta ha avuto successo, l'informazione è inclusa
201 Created:	La risorsa è stata creata con successo in seguito ad UN POST (se non puo' essere creata subito 202 Accepted)
202 Accepted:	The request has been received but has not yet been handled in full (e.g., since a program must run which requires a long time → the user agent can continue with its task without waiting for the action to complete at the origin server)
204 No content:	Successfully received, no change required in what the user actually sees

■3xx Redirection

301 Moved Permanently:	L'oggetto è stato spostato nell'URL indicato
302 Moved Temporarily:	L'oggetto è stato spostato nell'URL indicato
304 Not Modified:	L'oggetto non modificato dal tempo incluso nella richiesta

■4xx Client error

400 Bad Request:	errore generico (sintassi errata/irriconoscibile)
401 Unauthorized:	Autenticazione fallita (e.g., Accesso senza necessari account e password)
403 Forbidden:	La richiesta è stata ricevuta e compresa ma il server ha stabilito di non servirla. Le ragioni possono essere indicate nell'entity body.
404 Not Found:	L'oggetto non esiste sul server

■5xx Server error

500 Internal server error	Generico. Errore o guasto nel server
501 Not implemented	Funzione non implementata
503 Service unavailable	Servizio non disponibile

Autenticazione.

HTTP è **stateless** e quindi *non si possono riconoscere richieste successive dello stesso utente*.

In HTTP esiste un elementare meccanismo di autenticazione (account e password) che serve a riconoscere gli utenti.

Normalmente il browser memorizza passwd e account in modo da non richiedere la digitazione ogni volta.

C'è un altro modo per riconoscere richieste successive di uno stesso utente che non richiede di ricordare password:

Interazione utente-server: i cookie.

I server HTTP sono privi di stato.

È spesso gradito che i web server possano autenticare gli utenti, sia per limitare l'accesso da parte di questi ultimi sia per fornire contenuti in funzione della loro identità.

A questo scopo, http adotta i **cookie**. I cookie **consentono ai server di tener traccia degli utenti**. I **cookie possono essere usati per identificare gli utenti** e mantenere delle info di stato su una transazione che richiede vari scambi di messaggi http. La maggior parte dei siti commerciali usa i cookie.

La tecnologia dei cookie presenta quattro componenti:

- (1) una riga di intestazione nel messaggio di risposta HTTP,
- (2) una riga di intestazione nel messaggio di richiesta HTTP,
- (3) un file mantenuto sul sistema dell'utente e gestito dal browser e
- (4) un database sul sito.

Nonostante i cookie semplificano molte attività via internet, sono fonte di controversie, in quanto possono essere considerati una violazione della privacy e dell'utente. Inoltre, potrebbero essere intercettati tramite diverse tecniche (cross-site scripting, cross-site request) ed usati per ottenere l'accesso (credenziali) al sito web a cui il cookie appartiene.

Web caching.

Una web cache, nota anche come **proxy server**, è un'entità di rete che soddisfa richieste HTTP al posto del web server effettivo. Il proxy ha una propria memoria su disco (una cache) in cui conserva copie di oggetti recentemente richiesti. Il browser di un utente può essere configurato in modo che tutte le richieste HTTP dell'utente vengano innanzitutto dirette al proxy server. Supponiamo per esempio che un browser stia richiedendo l'oggetto <http://www.someschool.edu/campus.gif>. Ecco che cosa succede.

1. Il browser stabilisce una connessione TCP con il proxy server e invia una richiesta HTTP per l'oggetto specificato.

2. Il proxy controlla la presenza di una copia dell'oggetto memorizzata localmente. Se l'oggetto viene rilevato, il proxy lo inoltra all'interno di un messaggio di risposta HTTP al browser.

3. Se, invece, la cache non dispone dell'oggetto, apre una connessione TCP verso il server di origine; ossia, nel nostro esempio, www.someschool.edu. Poi, il proxy invia al server una richiesta HTTP per l'oggetto. Una volta ricevuta tale richiesta, il server di origine invia al proxy l'oggetto all'interno di una risposta HTTP.

4. Quando il proxy riceve l'oggetto ne salva una copia nella propria memoria locale e ne inoltra un'altra copia, all'interno di un messaggio di risposta HTTP, al browser (sulla connessione TCP esistente tra il browser e il proxy).

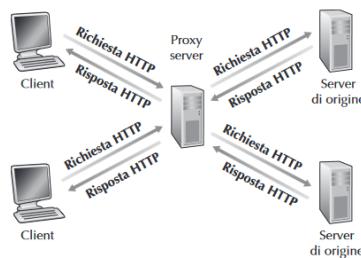


Figura 2.11 Client che richiedono oggetti attraverso un proxy server.

Si noti che il **proxy** è **contemporaneamente server e client**: quando riceve richieste da un browser e gli invia risposte agisce da server, quando invia richieste e riceve risposte da un server di origine funziona da client.

Generalmente un proxy server è acquistato e installato da un ISP.

Per esempio, un'università può installare un proxy sulla propria rete e configurare tutti i browser della propria sede per puntare a quel proxy. Oppure un grande ISP (quale ComCast) potrebbe installare uno o più proxy nella propria rete e preconfigurare i browser in modo che vi puntino.

Il web caching si è sviluppato in Internet per due ragioni.

Le motivazioni alla base dell'utilizzo di questo meccanismo sono prima di tutto la **riduzione massiccia dei tempi di risposta** alle richieste e in secondo luogo la **riduzione del traffico**.

HTTP fornisce due meccanismi per la gestione della coerenza nel caching. Nel TTL (Time To Live) il server quando fornisce un oggetto dice anche quando scade (header Expired), oppure tramite GET condizionali.

GET condizionale.

Sebbene il web caching riduca i tempi di risposta percepiti dall'utente, introduce un nuovo problema: la copia di un oggetto che risiede in cache potrebbe essere scaduta.

In altre parole, l'oggetto ospitato nel web server potrebbe esser stato modificato rispetto alla copia nel client (sia esso un proxy o un browser). Fortunatamente, http presenta un meccanismo che permette alla cache di verificare se i suoi oggetti sono aggiornati. Questo meccanismo è chiamato GET condizionale (conditional GET).

Un messaggio di richiesta HTTP viene detto messaggio di GET condizionale se (1) usa il metodo GET e (2) include una riga di intestazione If-modified-since::

Notiamo che in risposta a un GET condizionale, il web server invia ancora un messaggio di risposta, ma non include l'oggetto richiesto, in quanto ciò implicherebbe solo spreco di banda e incrementerebbe il tempo di risposta percepito dall'utente, in particolare se l'oggetto è grande. La riga di stato 304 Not Modified comunica al proxy che può procedere e inoltrare al browser richiedente la copia dell'oggetto presente in cache.

PROTOCOLLO HTTPS.

Il traffico effettuato attraverso il browser con il *protocollo HTTP* è completamente in chiaro **senza alcun genere di sicurezza**. Qualunque informazione personale trasmessa tramite questo protocollo può essere intercettata sulla rete tramite un semplice programma di sniffing. Il **protocollo HTTPS** introduce invece un canale di comunicazione criptato attraverso lo scambio di certificati in modo da garantire l'identità delle parti e la riservatezza dei dati. L'HTTPS si appoggia generalmente sulla porta 443. Sintatticamente è identico all'HTTP, al quale è sovrapposto un livello di SSL (Secure Sockets Layer).

PROTOCOLLO FTP.

È il protocollo utilizzato a livello applicativo per trasferire i file (File Transfer Protocol), basato su TCP e definito nel RFC 959.

A differenza di altri protocolli, FTP per trasportare un file utilizza **due canali TCP** separati che agiscono in parallelo:

- Una connessione di controllo usata per **trasmettere informazioni** di controllo tra client e server come l'identificazione dell'utente, password, ecc.

- Una connessione per il **trasferimento dati**.

Dato che l'FTP usa un connessione di controllo separata, si dice che invia le sue informazioni di controllo fuori banda (out-of-band).

FTP è un protocollo statefull perché mantiene lo stato sulle directory e sui dati di autenticazione durante le trasmissioni.

Client e Server FTP.

Il protocollo FTP si riferisce ad un modello client/server dove il client da luogo al trasferimento in entrambi i versi e il server resta in attesa delle connessioni (in ascolto sulla porta 21).

FTP Attivo.

Quando un utente avvia una sessione FTP con un server remoto, il lato client apre due porte con numero random superiore a 1023, una per la trasmissione e ricezione dei dati e l'altra per il controllo.

Il client instaura una connessione TCP di controllo con il server sulla porta 21 del server specificando il numero di porta dati del client.

Il client ottiene dal server l'autorizzazione alla connessione sulla linea di controllo.

Il client invia l'identificazione dell'utente e la password sulla connessione di controllo.

Il server apre il data channel verso la porta indicata dal client per iniziare il trasferimento. Al termine del trasferimento il server chiude la connessione.

Una problematica legata a questa modalità è il fatto che i porti si stabiliscono dinamicamente in fase di controllo; quindi, i firewall potrebbero bloccare le connessioni in ingresso sulle porte stabilite.

FTP Passivo.

Questa modalità consente al client di aprire sia la connessione di controllo che quella per i dati.

Il client alloca due porte con numero maggiore di 1023, ma non viene indicato al server il numero di porta riservata ai dati: è il server a comunicare al client il numero di porta che lui riserva per il trasferimento dei dati, e quindi non è più la porta numero 21 di default.

La risposta del server comprende anche il suo numero di porta data in modo che il client attivi la connessione anche su di essa, inviando un comando di richiesta di apertura per il canale data.

Comandi FTP.

USER <username>, PASS <password>, LIST elenca i file della directory corrente, RETR <filename> recupera (get) un file dalla directory corrente STOR <filename> memorizza (put) un file nell'host remoto, CWD <directory> cambia directory corrente

Sia comandi che risposte FTP sono codificate in testo ASCII

DNS: il servizio di directory di Internet

Le persone possono essere identificate in molti modi, il nome di battesimo, il codice fiscale o il numero della patente.

Proprio come le persone, anche gli host Internet possono essere identificati in vari modi, uno dei modi è identificarli dai cosiddetti *indirizzi IP*.

- Un **indirizzo IP** consiste di *quattro byte* e presenta una rigida struttura gerarchica. Ha una forma del tipo 121.7.106.83, in cui ogni *punto* separa uno dei byte espressi con un numero decimale compreso tra 0 e 255. Un indirizzo IP è gerarchico perché, leggendolo da sinistra a

destra, otteniamo informazioni sempre più specifiche sulla collocazione dell'host in Internet (ossia sulla sua appartenenza a quale rete all'interno della rete di reti).

Servizi forniti da DNS.

Abbiamo appena visto che esistono *due modi per identificare gli host*: il **nome** e l'**indirizzo IP**. Le persone preferiscono il primo, mentre i router prediligono gli indirizzi IP a lunghezza fissa e strutturati in modo gerarchico.

Al fine di conciliare i due approcci è necessario un *servizio in grado di tradurre i nomi degli host nei loro indirizzi IP*.

Si tratta del principale compito del **domain name system (DNS)** di Internet.

DNS è (1) un database distribuito implementato in una gerarchia di DNS server e (2) un protocollo a livello di applicazione che consente agli host di interrogare il database. I DNS server sono generalmente macchine UNIX che eseguono un software chiamato BIND (Berkeley Internet name domain) [BIND 2016].

Il **protocollo DNS utilizza UDP** e la **porta 53**.

DNS viene comunemente utilizzato da altri protocolli a livello di applicazione, tra cui HTTP e SMTP, per **tradurre i nomi di host forniti dall'utente in indirizzi IP**.

Cosa succede quando un browser (ossia un client HTTP) in esecuzione sull'host di un utente richiede l'URL www.someschool.edu/index.html. L'**host dell'utente**, per essere in grado di inviare un messaggio di richiesta HTTP al web server www.someschool.edu, deve come prima cosa **ottenere il suo indirizzo IP**. Ciò avviene come segue.

1. La stessa macchina utente esegue il lato client dell'applicazione DNS.
2. Il browser estrae il nome dell'host, www.someschool.edu, dall'URL e lo passa al lato client dell'applicazione DNS.
3. Il client DNS invia una interrogazione (query) contenente l'hostname a un DNS server.
4. Il client DNS prima o poi riceve una risposta, che include l'indirizzo IP corrispondente all'hostname.
5. Una volta ricevuto l'indirizzo IP dal DNS, il browser può dare inizio a una connessione TCP verso il processo server HTTP collegato alla porta 80 di quell'indirizzo IP.

Panoramica ad alto livello del funzionamento del DNS. **Servizio di traduzione da hostname a indirizzo IP**.

Supponiamo che una certa applicazione (browser web o un programma per la lettura della posta) in esecuzione sull'host di un utente abbia necessità di tradurre un hostname in un indirizzo IP. L'applicazione invocherà il lato client del DNS, specificando l'hostname da tradurre. Su macchine basate su UNIX, `gethostbyname()` è la chiamata di funzione per ottenere il servizio di traduzione. Il DNS sull'host prende poi il controllo, inviando un messaggio di richiesta (query) sulla rete. Tutte le query DNS e i messaggi di risposta vengono inviati all'interno di datagrammi UDP diretti alla porta 53. Dopo un ritardo che varia dai millisecondi ai secondi, il client DNS sull'host dell'utente riceve un messaggio di risposta contenente la corrispondenza desiderata, che viene poi passata all'applicazione che ne ha fatto richiesta.

Da questo esempio vediamo che il **DNS introduce un ritardo aggiuntivo**, talvolta sostanziale, alle applicazioni Internet che lo utilizzano. L'indirizzo IP desiderato si trova spesso nella cache di un DNS server vicino, il che aiuta a ridurre il traffico DNS in rete e il ritardo medio del servizio.

Oltre alla traduzione degli hostname in indirizzi IP, DNS mette a disposizione *altri importanti servizi*.

- **Host aliasing.** Un host dal nome complicato può avere uno o più sinonimi (alias).

Il DNS può essere invocato da un'applicazione per ottenere l'hostname canonico di un sinonimo, così come l'indirizzo IP dell'host.

- **Mail server aliasing.** Per ovvi motivi è fortemente auspicabile che gli indirizzi di posta elettronica siano facili da ricordare. Un'applicazione di posta può invocare il DNS per ottenere il nome canonico di un sinonimo fornito, così come l'indirizzo IP dell'host.

- **Distribuzione del carico di rete** (load distribution). Il **DNS** viene anche **utilizzato** per *distribuire il carico tra server replicati*, per esempio dei web server. Con la stessa richiesta di nome, il client può essere reindirizzato verso diversi server replicati.

Quando un client effettua una query DNS per un nome associato ad un insieme di indirizzi, il server risponde con l'intero insieme di indirizzi ma ne varia l'ordinamento ad ogni risposta. Dato che il client invia la richiesta HTTP al primo indirizzo, di fatto si distribuisce il traffico sui diversi server.

DNS Centralizzato.

Un DNS centralizzato prevede un singolo server che contiene un database con le corrispondenze IP/nomi.

Seppur di semplice implementazione, un sistema del genere implica una serie di problemi tra i quali:

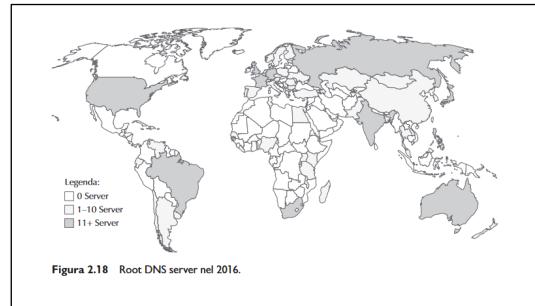
- Un solo punto di fallimento. Se il DNS server si guasta, ne soffre l'intera Internet.
- Volume di traffico. Un singolo DNS server dovrebbe gestire tutte le richieste generate da migliaia di host.
- Database centralizzato distante. Un singolo DNS server non può essere vicino a tutti i client causando ritardi significativi.
- Manutenzione. Il singolo DNS server dovrebbe contenere record relativi a tutti gli host di Internet. Non solo tale sarebbe vasto, ma dovrebbe essere aggiornato frequentemente per tener conto di ogni nuovo host.

In conclusione, un database centralizzato su un singolo DNS server non è in grado di adattarsi alla crescita esponenziale della rete (ovvero, non è scalabile). Di conseguenza, il DNS è stato progettato in maniera distribuita e costituisce un magnifico esempio di come si possa implementare un database distribuito su Internet.

Un database distribuito e gerarchico.

Per trattare il problema della scalabilità, il DNS utilizza un grande numero di server, organizzati in maniera gerarchica e distribuiti nel mondo. In prima approssimazione, esistono tre classi di DNS server: i **root server**, i **top-level domain (TLD) server** e i **server autoritativi**, organizzati in una gerarchia, come mostrato nella Figura 2.17.

- **Root server.** sono i server DNS responsabili del dominio di radice (punto . alla fine dei domini), possiedono l'elenco dei server autoritativi di tutti i domini di primo livello (TLD) conosciuti e lo forniscono in risposta a ciascuna richiesta. I root server sono 13 in tutto il mondo, 7 dei quali collocati negli Stati Uniti; Gli altri sono dislocati sul globo tramite indirizzamento anycast (permette di assegnare a più computer lo stesso indirizzo IP);



- **Top-level domain (TLD) server.** Questi server si occupano dei domini di primo livello quali com, org, net, edu e gov, e di tutti i domini di primo livello relativi ai vari paesi, come uk, fr, ca e jp.

I server TLD forniscono gli indirizzi IP dei server autoritativi.

- **DNS server autoritativi.** Ogni organizzazione dotata di host pubblicamente accessibili tramite Internet (quali web server e-mail server) deve fornire record DNS pubblicamente accessibili che associno i nomi di tali host a indirizzi IP. Il DNS server autoritativo dell'organizzazione ospita questi record. L'organizzazione può decidere di pagare terzi o di implementare il proprio server autoritativo per ospitare tali record.

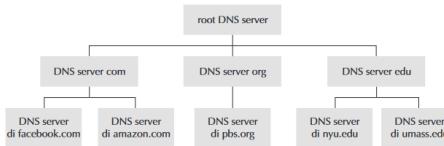


Figura 2.17 Gerarchia parziale di server DNS.

Con questo primo approccio se un Client richiede l'IP di www.amazon.com:

1. Il client contatta un root server per avere la lista degli indirizzi IP dei TLD per il dominio .com;
2. Il client contatta uno dei TLD server che gli restituisce l'indirizzo IP del server autorevole per amazon.com;
3. Il client contatta il server autorevole per amazon.com che gli restituisce l'indirizzo IP di www.amazon.com;

I DNS server appena descritti sono tutti collocati in una gerarchia di DNS server.

DNS Server locali.

Esiste un altro importante tipo di DNS, detto **DNS server locale**, che non appartiene strettamente alla gerarchia di server, ma che è comunque *centrale nell'architettura DNS*. Ciascun ISP, come un'università, un dipartimento universitario, un'azienda o un ISP residenziale, ha un DNS server locale (detto anche default name server).

Quando un host effettua una richiesta DNS, la query viene inviata al DNS locale, che opera da proxy e inoltra la richiesta alla gerarchia dei DNS server.

Ciascun host deve essere configurato con l'indirizzo del DNS server locale. Questa configurazione avviene in manuale o in automatico tramite l'utilizzo del DHCP.

DNS Caching.

Il DNS sfrutta in modo estensivo il caching per migliorare le prestazioni di ritardo e per ridurre il numero di messaggi DNS che "rimbalzano" su Internet. L'idea alla base del DNS caching è molto semplice. In una concatenazione di richieste, il DNS server che riceve una risposta DNS (contenente, per esempio, la traduzione da hostname a indirizzo IP), può mettere in cache le informazioni contenute.

Se una coppia hostname/indirizzo IP è nella cache di un DNS server e giunge al server un'altra richiesta con lo stesso hostname, il DNS server può fornire l'indirizzo IP desiderato, anche se non è autoritativo per tale indirizzo.

Dato che gli host e le associazioni tra nome e indirizzo IP non sono in alcun modo permanenti, i DNS server invalidano le informazioni in cache dopo un periodo di tempo fissato (in genere di 2 giorni).

Record e messaggi DNS.

I server che implementano il database distribuito di DNS memorizzano i cosiddetti record di risorsa (RR, resource record), tra cui quelli che forniscono le corrispondenze tra nomi e indirizzi. Ogni messaggio di risposta DNS trasporta uno o più record di risorse. Un record di risorsa contiene i seguenti campi: (Name, Value, Type, TTL)

TTL è il time to live, il tempo residuo di vita di un record e determina quando una risorsa vada rimossa dalla cache. Nei record di esempio di seguito riportati ignoreremo il campo TTL. Il significato di Name e Value dipende da Type:

- **Tipo A:** un record A fornisce la corrispondenza hostname/indirizzo IP.
 - o nome=hostname;
 - o valore = indirizzo IP;

- Tipo NS: usato per instradare le richieste DNS successive alla prima concatenazione delle query;
 - o nome=dominio;
 - o valore=indirizzo IP del NameServer Autoritativo;
- Tipo CNAME:
 - o nome=alias per il nome canonico,
 - o valore=nome canonico;
- Tipo MX:
 - o nome=dominio di posta;
 - o valore = nome dell'host associato al nome;

Messaggi DNS.

Gli unici **due tipi di messaggio DNS** sono le **query** e i **messaggi di risposta** che presentano, entrambi, lo stesso formato (Figura 2.21). La semantica dei campi dei messaggi DNS è la seguente.

Identificazione	Flag	
Numero di domande	Numero di RR di risposta	-12 byte
Numero di RR autoritativi	Numero di RR addizionali	
Sezione delle domande (numero variabile di domande)	Campi di nome e tipo di una query	
Sezione delle risposte (numero variabile di record di risorsa)	RR in risposta alla query	
Sezione autoritativa (numero variabile di record di risorsa)	Record per i server autoritativi	
Sezione aggiuntiva (numero variabile di record di risorsa)	Informazione aggiuntiva "d'aiuto" che può essere usata	

Figura 2.21 Formato dei messaggi DNS.

- I primi 12 byte rappresentano la sezione di intestazione, che a sua volta contiene un certo numero di campi.
 - Il primo è un numero di 16 bit che identifica la richiesta.
 - Campo flag. Il primo di questi, il bit di richiesta/risposta (query/reply), indica se il messaggio è una richiesta (0) o una risposta (1).
 - Un bit viene impostato nei messaggi di risposta quando il DNS server è autoritativo per il nome richiesto.
 - Un bit di richiesta di ricorsione (recursion-desired flag), viene impostato quando un client (sia esso un host o un DNS server) desidera che il DNS server effettui ricorsione quando non dispone del record.
 - Quattro campi il cui nome comincia con "numero di", che indicano il numero di occorrenze delle quattro sezioni di tipo dati che seguono l'intestazione.

- La sezione delle domande contiene informazioni sulle richieste che stanno per essere effettuate. Inoltre, include (1) un campo nome con il nome che sta per essere richiesto, e (2) un campo tipo che indica il tipo della domanda sul nome.
- In una risposta proveniente da DNS server, la sezione delle risposte contiene i record di risorsa relativi al nome originariamente richiesto. Ricordiamo che in ogni record di risorsa troviamo Type (A, NS, CNAME o MX), Value e TTL.
- La sezione autoritativa contiene i record di altri server autoritativi.
- La sezione aggiuntiva racchiude altri record utili.

Inserimento di record nel database DNS.

Come sono inseriti i record nel database? Vediamolo nel contesto di uno specifico esempio.

- Registrare il nome di dominio presso un ente di registrazione (registrar). Un registrar è un'azienda che verifica l'unicità del nome di dominio, lo inserisce nel database DNS e vi richiede una piccola somma di denaro per i propri servizi.

Quando registrate il nome di dominio fornire anche i nomi e gli indirizzi IP dei vostri DNS server autoritativi primario e secondario.

- Dovrete inoltre accertarvi che il record di risorsa di tipo A per il vostro web e che il record di risorsa di tipo MX per il vostro server di posta vengano immessi nei vostri DNS server autoritativi.

Più di recente, al protocollo DNS è stata aggiunta l'opzione UPDATE, per consentire l'aggiunta o la cancellazione dinamica di dati dal database attraverso messaggi DNS.

Una volta completati questi passi sarà possibile visitare il vostro sito web.

Posta elettronica in Internet.

L'e-mail rappresenta un mezzo di comunicazione asincrono: le persone inviano e leggono i messaggi nel momento per loro più opportuno, senza doversi coordinare con altri utenti. La posta elettronica è veloce, facile da distribuire e gratuita. La moderna posta elettronica ha molte caratteristiche importanti quali gli allegati (attachment), i collegamenti ipertestuali, il testo con formattazione HTML e le foto incorporate.

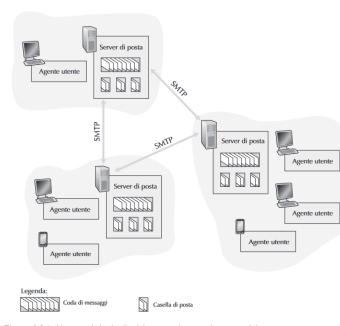


Figura 2.14 Visione ad alto livello del sistema di posta elettronica di Internet.

Da questo diagramma notiamo la presenza di tre componenti principali:

- Gli **user agent** (o *agenti utente*): (per esempio Microsoft Outlook e Apple Mail) consentono agli utenti di leggere, rispondere, inoltrare, salvare e comporre i messaggi.

- I **server di posta** (o *mail server*): costituiscono la parte centrale dell'infrastruttura del servizio di posta elettronica:

- Ciascun destinatario ha una mail box contenente i messaggi in entrata non letti;
- Il mail server contiene una coda di messaggi in uscita che contiene i messaggi non ancora recapitati;

- Il **protocollo SMTP** (*simple mail transfer protocol*): principale protocollo a livello di applicazione per la posta elettronica su Internet. Fa uso del servizio di trasferimento dati affidabile proprio di TCP per trasferire la mail dal server del mittente a quello del destinatario.

SMTP presenta un lato client, in esecuzione sul mail server del mittente e un lato server, in esecuzione sul server del destinatario. Entrambi i lati possono essere eseguiti su tutti i server di posta. Quando un server invia posta a un altro agisce come client SMTP; quando invece la riceve, funziona come server SMTP.

SMTP.

Questo protocollo è il cuore della posta elettronica su Internet. **SMTP** trasferisce i messaggi dal mail server del mittente a quello del destinatario.

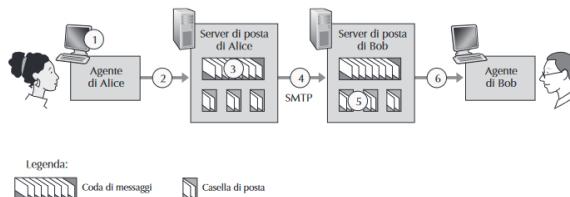
SMTP è assai più vecchio di http e rappresenta una tecnologia ereditata con caratteristiche "arcaiche".

- Tratta il corpo (non solo le intestazioni) di tutti i messaggi di posta come semplice ASCII a 7 bit. La restrizione all'ASCII a 7 bit è piuttosto penalizzante, in quanto richiede che i dati multimediali binari vengano codificati in ASCII prima di essere inviati e che il messaggio venga nuovamente decodificato in binario dopo il trasporto.

Al fine di illustrare le operazioni di base di SMTP, presentiamo uno scenario tipico.

Supponiamo che Alice voglia inviare a Bob un semplice messaggio ASCII.

1. Alice invoca il proprio user agent per la posta elettronica, fornisce l'indirizzo di posta di Bob (per esempio bob@someschool.edu), compone il messaggio e dà istruzione allo user agent di inviarlo.
2. Lo user agent di Alice invia il messaggio al suo mail server, dove è collocato in una coda di messaggi.
3. Il lato client di SMTP, eseguito sul server di Alice, vede il messaggio nella coda dei messaggi e apre una connessione TCP verso un server SMTP in esecuzione sul mail server di Bob.
4. Dopo un handshaking SMTP, il client SMTP invia il messaggio di Alice sulla connessione TCP.
5. Presso il mail server di Bob, il lato server di SMTP riceve il messaggio, che viene posizionato nella casella di Bob.
6. Bob, quando lo ritiene opportuno, invoca il proprio user agent per leggere il messaggio.



SMTP non usa mail server intermedi per inviare la posta, anche quando i mail server finali sono collocati agli angoli opposti del mondo. Se il mail server di Bob è spento, il messaggio rimane nel mail server di Alice e attende un nuovo tentativo.

- Il protocollo SMTP presenta molte somiglianze con i protocolli usati per l'interazione umana faccia a faccia.

Primo, il client SMTP (in esecuzione sul mail server di invio) fa stabilire a TCP una connessione sulla porta 25 verso il server SMTP (in esecuzione sul mail server in ricezione).

Se il server è inattivo, il client riprova più tardi.

Una volta stabilita la connessione, il server e il client effettuano una qualche forma di handshaking a livello applicativo. Client e server SMTP si presentano prima di effettuare scambi di informazioni. Durante questa fase, il client indica l'indirizzo e-mail del mittente (la persona che ha generato il messaggio) e quello del destinatario. Dopo la reciproca presentazione, il client invia il messaggio.

SMTP può contare sul servizio di trasferimento dati affidabile proprio di TCP per recapitare il messaggio senza errori.

Il client ripete il processo sulla stessa connessione TCP se ha altri messaggi da inviare al server, altrimenti ordina a TCP di chiudere la connessione.

Diamo ora uno sguardo a un esempio di trascrizione di messaggi scambiati tra un client SMTP (C) e un server SMTP (S). Il nome dell'host del client è crepes.fr mentre il nome dell'host del server è hamburger.edu. Le righe di testo ASCII precedute da C: sono esattamente quelle che il client invia nella propria socket TCP, mentre le righe precedute da S: sono esattamente quelle che il server invia nella propria socket TCP. La seguente trascrizione inizia appena si stabilisce la connessione TCP:

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Ti piace il ketchup?
C: Che cosa ne pensi dei cetrioli?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Il client ha inviato cinque comandi: HELO (abbreviazione di HELLO), MAIL FROM, RCPT TO, DATA e QUIT.

("ciao", "mail da", "recapitare a", "dati", "basta").

Il client invia anche una riga che consiste unicamente di un punto, che indica al server la fine del messaggio.

Per ciascun messaggio il client inizia il processo con un nuovo MAIL FROM: crepes.fr e stabilisce la fine del messaggio con un punto isolato.

Il comando QUIT viene inviato solo dopo aver spedito tutti i messaggi. Per effettuare un dialogo diretto con un server SMTP è possibile usare Telnet. Per farlo, usate il comando telnet serverName 25 dove serverName è il nome di un mail server locale: ciò stabilisce una connessione TCP tra il vostro host locale e il mail server. Dopo aver digitato questa riga, dovrà ricevere immediatamente la risposta 220 da parte del server. Quindi, immettete i comandi SMTP HELO, MAIL FROM, RCPT TO, DATA, "." e QUIT al momento opportuno.

Confronto con http.

Durante il trasferimento, sia **HTTP** che **SMTP** utilizzano **connessioni persistenti**.

HTTP è però principalmente un protocollo pull, nel senso che gli utenti usano HTTP per richiedere risorse (pull) dal web server; la connessione TCP viene iniziata da qualcuno che vuole ricevere il file.

SMTP è al contrario un protocollo push, perché il mail server di invio spinge (push) i file al mail server in ricezione; la connessione TCP viene iniziata dall'host che vuole spedire il file.

Una seconda differenza è che mentre l'STMP deve comporre l'intero messaggio con codifica ASCII a 7 bit, HTTP non impone questo vincolo.

Formati dei messaggi di posta.

Il corpo dei messaggi di posta elettronica è preceduto da un'intestazione contenente informazioni di servizio. Tale informazione periferica è contenuta in una serie di righe di intestazione.

Queste righe sono separate dal corpo del messaggio mediante una riga senza contenuto.

Come avviene per HTTP, queste righe contengono testo leggibile, costituito da una parola chiave seguita da due punti a loro volta seguiti da un valore. Alcune parole chiave sono obbligatorie, mentre altre sono opzionali.

Ogni intestazione deve avere una riga From: e una riga To:.

Un'intestazione può includere una riga Subject: e altre righe di intestazione opzionali. È importante osservare che queste righe sono differenti dai comandi SMTP visti (anche se contengono alcune parole comuni quali "from" e "to").

I comandi di tale paragrafo facevano parte del protocollo SMTP; le righe di intestazione esaminate qui sono invece parte del messaggio stesso. Ecco una tipica intestazione di messaggio.

From: alice@crepes.fr

To: bob@hamburger.edu

Subject: Alla ricerca del significato della vita.

Dopo l'intestazione, segue una riga vuota; quindi, troviamo il corpo del messaggio (in ASCII).

Protocolli di accesso alla posta.

Attualmente, l'accesso alla posta elettronica utilizza un'architettura **client-server**: l'utente legge le e-mail con un client in esecuzione sul proprio sistema periferico (il PC dell'ufficio, un laptop o uno smartphone). Eseguendo un client di posta su un PC locale, gli utenti beneficiano di molteplici possibilità, tra cui la capacità di visualizzare messaggi multimediali e allegati.

Ricordiamo che un server di posta gestisce caselle ed esegue il lato client e server di SMTP. Se il server di posta dovesse risiedere sul suo PC locale, quest'ultimo dovrebbe rimanere sempre acceso e connesso a Internet al fine di ricevere nuova posta che può giungere in qualsiasi istante. Tale procedura è poco pratica per cui si preferisce che l'utente abbia in esecuzione uno user agent sul PC locale, ma acceda alla propria casella memorizzata su un mail server condiviso con altri utenti e sempre attivo. Questo server è generalmente gestito dall'ISP dell'utente (per esempio, un'università o un'azienda).

POP3.

POP3 è un *protocollo di accesso alla posta* estremamente semplice con delle funzionalità limitate.

POP3 entra in azione quando lo user agent (il client) apre una connessione TCP verso il mail server (il server) sulla porta 110.

Quando la connessione TCP è stabilita, POP3 procede in tre fasi:

- **Autorizzazione:** lo user agent invia nome utente e password (in chiaro) per autenticare l'utente.
- **Transazione:** lo user agent **recupera i messaggi**, marca i messaggi per la cancellazione, rimuove i marcatori di cancellazione e ottiene statistiche sulla posta.
- **Aggiornamento:** ha luogo dopo che il client ha inviato il comando quit, che conclude la sessione POP3; in questo istante, il server di posta rimuove i messaggi che sono stati marcati per la cancellazione.

Il mail server può rispondere ai comandi inviati dall'user con +OK (talvolta seguito da dati dal server al client) o -ERR, utilizzato dal server per indicare che qualcosa non ha funzionato nel precedente comando.

Vari comandi:

- user : specifica la username
- pass : specifica la password
- list : visualizza la lista dei messaggi
- retr : preleva il messaggio per numero
- dele : elimina il messaggio dal server
- quit : chiude la sessione

La modalità POP3 consente di scaricare tutti i messaggi della posta in arrivo dal server direttamente sul pc o sul dispositivo che stai utilizzando.

Un user agent che usa POP3 può essere configurato in una modalità ‘scarica e cancella’ dove l’utente scarica il messaggio da leggere e il server lo elimina direttamente dalla mail box; ed una modalità ‘scarica e mantieni’ dove il mail server mantiene anche i messaggi letti. Per la prima modalità se un user agent è configurato su più macchine, non è possibile leggere un messaggio già scaricato da una macchina con un’altra macchina.

IMAP.

Il protocollo IMAP (*Interactive Mail Access Protocol*) si contrappone al POP3

IMAP è un protocollo che presenta maggiori potenzialità rispetto a POP3 ed è quindi assai più complesso.

- POP3 permette di **scaricare e mantenere i messaggi in locale sull’host**.

- IMAP invece è pensato per **consentire direttamente l’accesso ai messaggi memorizzati su un server remoto**.

Il protocollo IMAP lascia tutta la posta sul server, così che potrai consultarla anche da Webmail.

Configurando la casella e-mail in IMAP, visualizzerai sul tuo dispositivo, come sulla webmail, tutti i messaggi in tutte le sottocartelle (comprese Posta inviata, Bozze, Cestino, e così via ...).

Se elimini un messaggio dalla webmail, questo sparirà anche da tutti i dispositivi configurati con IMAP e viceversa.

Se decidi di utilizzare IMAP per le tue caselle di posta, assicurati che tutti gli altri dispositivi che leggono tale casella utilizzino la stessa modalità: se un solo dispositivo fosse configurato in modalità POP3, infatti, questo dispositivo scaricherebbe tutti i messaggi, eliminandoli dalla webmail e da tutti i dispositivi configurati in IMAP.

Posta basata sul Web.

Ultimamente gli *utenti accedono alle proprie e-mail* tramite un **browser web**. In questo caso, lo user agent è un semplice browser web e l’utente comunica con la propria casella remota via HTTP.

Un destinatario che vuole accedere alla propria casella vede il messaggio e-mail spedito dal server di posta al browser usando il protocollo HTTP anziché POP3 o IMAP.

Quando un mittente vuole inviare un messaggio di posta elettronica, quest’ultimo viene spedito dal suo browser al suo server di posta su HTTP anziché su SMTP. Il server di posta manda ancora i messaggi e li riceve utilizzando SMTP.

STREAMING VIDEO.

Generalmente i contenuti video messi a disposizione su Internet sono di due tipologie: in presa diretta (distribuiti contemporaneamente a più utenti) o di tipo Video-On-Deman (cioè pre-registrati). Nei primi vengono utilizzati protocolli multicast per la distribuzione simultanea e nei secondi connessioni unicast tra l’utente e la piattaforma che ospita il servizio.

Nello streaming HTTP il video viene memorizzato in un server HTTP come un file ordinario con un url specifico. Quando un utente vuole vedere un video, stabilisce una connessione TCP con il server e invia un messaggio GET per il suo URL. Il server invia il video all’interno del messaggio HTTP di risposta. Sul lato client i dati vengono bufferizzati. Lo svantaggio è che il client ricevono la stessa versione codificata del video, nonostante abbiano disponibile una larghezza di banda che può variare da un caso all’altro.

Per superare il problema è stato sviluppato lo streaming dinamico adattivo su HTTP (DASH dynamic adaptive streaming over HTTP). In DASH i video vengono codificati in diverse versioni, ognuna avendo un bit rate differente e quindi differente livello di qualità. Il client seleziona automaticamente in modo dinamico la versione con bit rate adeguato alla banda disponibile.

CDN Content Delivery Network.

Nella distribuzione dei contenuti vengono introdotte le problematiche del delay e del throughput dato che le singole risorse devono essere trasferite da un singolo provider verso una molteplicità di utenti.

La soluzione è rappresentata dal Content Delivery Network, un sistema di server collegati in rete che collaborano per distribuire contenuti agli utenti finali. La tendenza è quella di replicare il contenuto su quanti più nodi possibili e soprattutto vicino agli utenti finali.

L’utente, quindi, richiede la risorsa al server CDN installato nell’ISP di provenienza, e non il server originario.

L’user experience risulta essere migliore se il contenuto è replicato su molti server, dato che l’accesso avviene su server geograficamente più vicini.

La CDN crea una mappa che indica la distanza tra i vari ISP e i nodi CDN. Quando arriva una query al DNS autoritativo si determina l’ISP che ha originato la query, si usa la mappa per la scelta del server CDN più vicino e ruta la richiesta a quest’ultimo.

Le CDN solitamente adottano due politiche di dislocazione dei server:

Enter deep: filosofia introdotta da Akamai che prevede di entrare in profondità nelle reti di accesso degli ISP installando gruppi di server, detti anche cluster. L’obiettivo è quello di essere vicini agli utenti finali in modo da limitare il ritardo percepito e il throughput;

Bring home: filosofia di Limelight che prevede di portarsi in casa l’ISP costruendo grandi cluster in pochi punti chiave e interconnetterli usando una rete privata ad alata velocità; Invece di entrare negli ISP pongono un cluster vicino ai PoP di molti ISP di livello 1;

Distribuzione di file P2P.

È un modello di architettura logica in cui i nodi non sono gerarchizzati sotto forma di client e di server, ma sotto forma di nodi equivalenti o paritari (**peer**).

Ci sono *coppie di host* connessi in modo intermittente, chiamati **peer**, che **comunicano direttamente l’uno con l’altro**. Qualsiasi nodo, dunque, è in grado di **avviare o completare** una *transazione*.

Due host della rete sono connessi in modo intermittente e comunicano direttamente tra di loro.

I peer non appartengono ai fornitori dei servizi, ma sono computer fissi e portatili controllati dagli utenti.

Uno tra i più diffusi protocolli di distribuzione di file tramite P2P è BitTorrent.

Tra i vantaggi dell'utilizzo di quest'architettura:

- Non è necessario un server con potenza elevata; la capacità di servizio è distribuita;
- Elevata scalabilità, ovvero possibilità di aggiungere capacità di servizio al sistema (più peer);
- La velocità media di trasmissione è più elevata dal momento che l'informazione richiesta può essere reperita da diversi client connessi (peer) anziché un unico server.

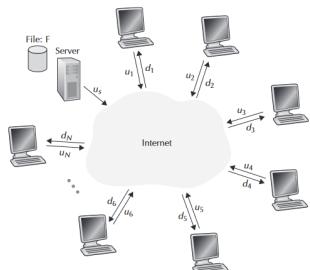
Scalabilità dell'architettura P2P.

Server e peer sono collegati a Internet con collegamenti di accesso:

- U_s la banda di upload del collegamento di accesso del server;
- U_i la banda di upload del collegamento di accesso dell'i-esimo peer;
- D_i la banda di download del collegamento di accesso dell'i-esimo peer.

F la dimensione del file da distribuire (in bit) e N il numero di peer che vuole una copia del file.

Il tempo di distribuzione è il tempo richiesto perché tutti gli N peer ottengano una copia, sotto l'ipotesi che la rete abbia banda elevata (cioè implica che i colli di bottiglia siano le reti di accesso).



Il server trasmette NF bit (N copie di F bit). Il tempo richiesto per inviare le copie è NF/U_s .
 Il client i impiega un tempo pari a F/D_i per scaricare una copia.
 Il tempo di distribuzione sarà pari al max { NF/U_s , $F/\min(D_i)$ }

Figura 2.22 Un problema illustrativo di distribuzione file.

Con questa architettura il tempo cresce linearmente con il crescere dei client N che scaricano il file.

In questa architettura ciascun client che abbia ricevuto alcuni dati del file concorre nella distribuzione del file stesso con la propria banda di upload.

Il server deve inviare almeno una copia ad un peer. Tempo richiesto F/U_s . Il client i impiega un tempo pari a F/D_i per il download. NF bit totali devono essere scaricati. Velocità massima di upload $U_s + \sum U_i$

Il tempo di distribuzione sarà pari al max { F/U_s , $F/\min(D_i)$, $NF/(U_s + \sum U_i)$ }

Con il crescere dei peer N che scaricano il file, il tempo di distribuzione diminuisce. È stato dimostrato quindi che l'architettura P2P (al contrario di Client-Server) può essere scalabile e questo è la diretta conseguenza del fatto che i peer re-distribuiscono i bit oltre che a scaricarli.

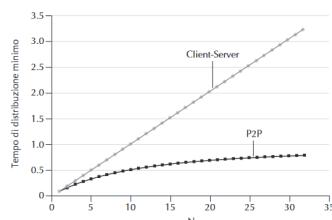


Figura 2.23 Tempo di distribuzione per le architetture P2P e client-server.

BitTorrent.

BitTorrent è un diffuso protocollo P2P per la distribuzione di file.

L'insieme di tutti i peer che partecipano alla distribuzione di un particolare file è chiamato **torrent** (torrente).

I peer in un torrent scaricano chunk (parti) del file di uguale dimensione, con una dimensione tipica di 256 kbyte.

Quando un peer entra a far parte di un torrent per la prima volta, non ha chunk del file. Col passare del tempo accumula sempre più parti che, mentre scarica, invia agli altri peer del torrent.

Una volta che un peer ha acquisito l'intero file, può (egoisticamente) lasciare il torrent o (altruisticamente) rimanere nel torrent e continuare a inviare chunk agli altri peer.

Qualsiasi peer può lasciare il torrent in qualsiasi momento con solo un sottoinsieme dei chunk del file e rientrare a far parte del torrent in seguito.

Ciascun torrent ha un nodo di infrastruttura chiamato **tracker**. Quando il peer si aggiunge ad un torrent esso si registra presso il tracker per avere la lista dei peer e si connette ad un sottoinsieme di tali peer (neighbors) direttamente con connessioni TCP.

Periodicamente, un peer chiede a tutti i neighbor la lista dei chunk in loro possesso ed invia richieste per i chunk mancanti.

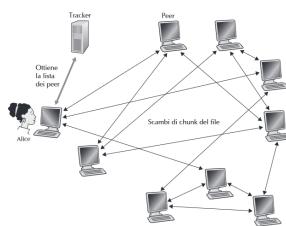


Figura 2.24 Distribuzione di file con BitTorrent.

Prelievo Chunk.

Nel decidere quali chunk richiedere si adotta la tecnica **rarest first**, cioè si determinano i chunk più rari tra quelli che mancano. In questo modo i chunk più rari vengono distribuiti più velocemente.

Invio Chunk.

Per determinare a quali richieste di chunk il peer debba rispondere si utilizza un algoritmo di tit-for-tat, cioè si dà priorità ai peer che forniscono dati al rate più alto. In particolare, vengono scelti i top 4, chiamati unchocked e che sono ricalcolati ogni 10 secondi. Ogni 30 secondi si seleziona un peer nuovo in maniera casuale (optimistically unchoke) e si inizia ad inviargli chunk. Il peer scelto può essere aggiunto ai top 4 se più veloce di uno dei 4.

Questo meccanismo limita la pratica del free-riding, ovvero di scaricare senza contribuire al torrent, in quanto incentiva scambi tra peer con velocità compatibili.

P2P con directory centralizzata. (Napster)

Il peer si connette inizialmente ad un server centralizzato fornendo il proprio indirizzo IP e il nome degli oggetti resi disponibili per la condivisione. In questo modo il server raccoglie le info sui peer attivi e le aggiorna dinamicamente.

Gli svantaggi di questo tipo di architettura sono: la presenza di un singolo punto di fallimento, un collo di bottiglia per le prestazioni, e soprattutto il server potrebbe contenere informazioni che violano il copyright.

Il trasferimento dei file è di fatto decentralizzato, ma la locazione dei contenuti è pesantemente centralizzata.

P2P Decentralizzato. (Gnutella)

Questa architettura è completamente decentralizzata (non esistono server). Si realizza con un'architettura di rete sovrapposta (overlay network) fatta da connessioni TCP in corso. L'overlay network ha una struttura paritetica dove ogni peer è connesso al massimo con 10 altri peer. Le problematiche da risolvere sono come costruire e gestire la rete di peer e come localizzare i contenuti all'interno della rete. Un peer A inizialmente si connette ad un solo peer B della rete e con la tecnica del flooding (inondazione) propaga le query di richiesta (messaggi di PING) in uno scope con raggio limitato. Cioè la richiesta si propaga ai peer collegati al peer B ma solo fino ad un certo punto (es 7 peer a partire dal primo). Il peer A quindi riceve tutti gli indirizzi (messaggi di PONG) IP intercettati nella rete con il flooding e a questo punto può connettersi direttamente tramite TCP.

P2P con directory decentralizzata.

Sfrutta le caratteristiche positive dell'architettura centralizzata di Napster e quelle dell'architettura decentralizzata di Gnutella. Ogni peer è associato ad un gruppo leader (mini hub) che è esso stesso un peer.

Un group leader memorizza le informazioni in condivisione dei "figli". Ogni group leader è in grado di interrogare altri group leader. Quindi il meccanismo delle query flooding è applicato alla rete dei group leader.

In una fase di bootstrap un peer che si connette deve essere associato ad un group leader o deve essere designato esso stesso group leader. L'overlay è costituito da connessioni TCP tra peer e group leader e tra coppie di group leader. Ogni file possiede un identificatore hash e un descrittore. I peer spediscono le query al proprio group leader; quest'ultimo risponde per ogni richiesta con l'indirizzo IP del detentore della risorsa, l'hash, e dei metadati associati alla risorsa. Il group leader inoltra sia le richieste sia le eventuali risposte da parte di altri group leader. Il peer seleziona la risorsa file per il download e invia una richiesta HTTP al detentore della risorsa usando l'hash come identificatore

Capitolo 3.

Livello di trasporto.

Introduzione e servizi a livello di trasporto.

Un **protocollo a livello di trasporto** mette a disposizione una comunicazione logica tra processi applicativi (dell'app in esecuzione) di host differenti.

Per comunicazione logica si intende che tutto proceda come se gli host che eseguono i processi fossero direttamente connessi; in realtà, gli host si possono trovare agli antipodi del pianeta, connessi da numerosi router e da svariati tipi di collegamenti.

All'interno di un sistema periferico, un protocollo di trasporto trasferisce i messaggi dai processi applicativi al bordo della rete (ossia, al livello di rete) e viceversa, ma non fornisce alcuna indicazione su come i messaggi siano trasferiti all'interno della rete.

I **protocolli a livello di trasporto** sono **implementati** nei sistemi periferici, ma non nei router della rete.

Lato mittente, il livello di trasporto converte i messaggi che riceve da un processo applicativo in pacchetti a livello di trasporto, noti come **segmenti** (transport-layer segment). Questo avviene spezzando i messaggi applicativi in parti più piccole e aggiungendo a ciascuna di esse un'intestazione di trasporto per creare un segmento.

Il **livello di trasporto**, quindi, passa il segmento al livello di rete, dove viene incapsulato all'interno di un pacchetto a livello di rete (datagramma) e inviato a destinazione.

Lato ricevente, il livello di rete estrae il segmento dal datagramma e lo passa al livello superiore, quello di trasporto. Quest'ultimo elabora il segmento ricevuto, rendendo disponibili all'applicazione destinataria i dati del segmento.

È possibile mettere a disposizione delle applicazioni di rete più di un protocollo a livello di trasporto.

Internet, per esempio, ne possiede due, TCP e UDP, che forniscono servizi diversi alle applicazioni che ne fanno uso.

Relazione tra i livelli di trasporto e di rete.

Il livello di trasporto è collocato esattamente sopra quello di rete nella pila di protocolli.

- Un **protocollo a livello di trasporto** mette a disposizione una comunicazione logica tra processi che vengono eseguiti su host diversi;
- Un **protocollo a livello di rete** fornisce comunicazione logica tra host.

I servizi che un protocollo di trasporto può offrire sono vincolati al modello di servizio del protocollo sottostante a livello di rete.

Determinati servizi possono essere garantiti da un protocollo di trasporto anche se il sottostante protocollo di rete non offre un servizio corrispondente.

Panoramica del livello di trasporto di Internet.

Internet, e più in generale una rete TCP/IP, mette a disposizione del livello di applicazione due diversi protocolli:

- Uno è **UDP** (*user datagram protocol*), che fornisce alle applicazioni un servizio non affidabile e non orientato alla connessione.
- L'altro è **TCP** (*transmission control protocol*), che offre un servizio affidabile e orientato alla connessione.

Lo sviluppatore delle applicazioni sceglie tra UDP e TCP quando crea le socket.

UDP e TCP forniscono, anche un controllo di integrità includendo campi per il riconoscimento di errori nelle intestazioni dei propri segmenti. In particolare, esattamente come IP, **UDP costituisce un servizio inaffidabile**: non garantisce che i dati inviati da un processo arrivino intatti (e neppure che arrivino) al processo destinatario.

TCP, d'altra parte, offre alle applicazioni diversi servizi aggiuntivi:

- fornisce un *trasferimento dati affidabile*. Grazie al controllo di flusso, ai numeri di sequenza, agli acknowledgement e ai timer, assicura che i dati sono trasferiti da un processo a un altro in modo corretto e ordinato.

Pertanto, TCP converte il servizio inaffidabile tra sistemi periferici, tipico di IP, in un servizio affidabile di trasporto dati tra processi.

- TCP fornisce anche il *controllo di congestione*, un servizio offerto alle applicazioni e a Internet che evita che le connessioni TCP intasino i collegamenti e i router tra gli host comunicanti con un'eccessiva quantità di traffico.

- Il traffico UDP, al contrario, non viene regolato. Le applicazioni che usano UDP possono spedire a qualsiasi velocità desiderata e per tutto il tempo voluto.

Segmento == il pacchetto a livello di trasporto.

Il **protocollo a livello di rete** di Internet ha un nome: **IP** (Internet protocol) che **fornisce comunicazione logica tra host**. Il suo modello di servizio prende il nome di best effort (letteralmente, "massimo sforzo"): questo significa che IP fa "del suo meglio" per consegnare i segmenti tra host comunicanti, ma non assicura né la consegna dei segmenti né il rispetto dell'ordine originario e non garantisce neppure l'integrità dei dati all'interno dei segmenti. Per queste ragioni si dice che **IP offre un servizio non affidabile**.

Ciascun host possiede un indirizzo IP.

A questo punto diamo uno sguardo ai **modelli di servizio offerti da UDP e TCP**, il cui principale compito è l'estensione del servizio di consegna di IP "tra sistemi periferici" a quello di consegna "tra processi in esecuzione sui sistemi periferici".

Questo passaggio, da consegna host-to-host consegna process-to-process, viene detto **multiplexing** e **demultiplexing** a livello di trasporto.

Multiplexing e demultiplexing.

Analizziamo il *multiplexing* e il *demultiplexing*, cioè come il servizio di trasporto da host a host fornito dal livello di rete possa diventare un servizio di trasporto da processo a processo per le applicazioni in esecuzione sugli host.

Nell'host destinatario il livello di trasporto riceve segmenti dal livello di rete immediatamente sottostante. Il livello di trasporto ha il compito di consegnare i dati di questi segmenti al processo applicativo appropriato in esecuzione nell'host.

Il livello di trasporto nell'host di ricezione in realtà non trasferisce i dati direttamente a un processo, ma piuttosto a una socket che fa da intermediario. Siccome, a ogni dato istante, può esserci più di una socket nell'host di ricezione, ciascuna avrà un identificatore univoco il cui formato dipende dal fatto che si tratti di socket UDP o TCP, come vedremo tra breve.

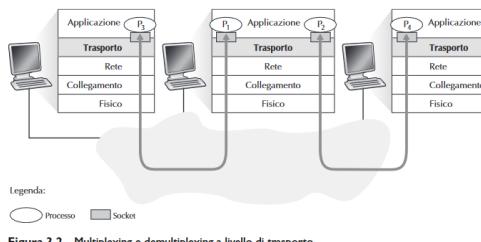


Figura 3.2 Multiplexing e demultiplexing a livello di trasporto.

- Il compito di trasportare i dati dei segmenti a livello di trasporto verso la giusta socket viene detto **demultiplexing**.

- Il compito di radunare frammenti di dati da diverse socket sull'host di origine, creare dei segmenti e passarli al livello di rete, viene detto **multiplexing**.

Il multiplexing a livello di trasporto richiede (1) che le socket abbiano identificatori unici e (2) che ciascun segmento presenti campi che indichino la socket cui va consegnato il segmento.

Questi (Figura 3.3) sono il campo del numero di porta di origine e il campo del numero di porta di destinazione.

Quando si sviluppa una nuova applicazione, è necessario assegnarle un numero di porta.



Figura 3.3 I campi del numero di porta di origine e di destinazione nei segmenti a livello di trasporto.

Ora dovrebbe essere chiaro come il livello di trasporto possa implementare il servizio di demultiplexing: ogni socket nell'host deve avere un numero di porta, e quando un segmento arriva all'host il livello di trasporto esamina il numero della porta di destinazione e dirige il segmento verso la socket corrispondente. I dati del segmento passano, quindi, dalla socket al processo assegnato. Come vedremo, questo è fondamentalmente il modo in cui agisce UDP, mentre il multiplexing/demultiplexing TCP è ancora più raffinato.

Multiplexing e demultiplexing non orientati alla connessione. (UDP)

Una socket UDP viene **identificata** completamente da una coppia che consiste di un **indirizzo IP** e di un **numero di porta di destinazione**. Di conseguenza, se due segmenti UDP hanno lo stesso indirizzo IP e lo stesso numero di porta di destinazione, saranno diretti allo stesso processo di destinazione tramite la medesima socket.

Multiplexing e demultiplexing orientati alla connessione. (TCP)

Una sottile differenza tra una socket TCP e una socket UDP risiede nel fatto che la prima è **identificata** da quattro parametri:

indirizzo IP di origine | numero di porta di origine | indirizzo IP di destinazione | numero di porta di destinazione.

Pertanto, quando un segmento TCP giunge dalla rete in un host, quest'ultimo utilizza i quattro valori per dirigere (fare demultiplexing) il segmento verso la socket appropriata. In particolare, e al contrario di UDP, due segmenti TCP in arrivo, aventi indirizzi IP di origine o numeri di porta di origine diversi, vengono diretti a due socket differenti, anche a fronte di indirizzo IP e porta di destinazione uguali, con l'eccezione dei segmenti TCP che trasportano la richiesta per stabilire la connessione.

Trasporto non orientato alla connessione: UDP.

Analizziamo ora da vicino le azioni di UDP:

-*Lato mittente*, prendiamo i messaggi dal processo applicativo e li passiamo direttamente a livello di rete;

-*Lato ricevente*, prendiamo i messaggi in arrivo dal livello di rete e li trasferiamo direttamente al processo applicativo.

Il livello di trasporto deve fornire un servizio di multiplexing/demultiplexing al fine di trasferire dati tra il livello di rete e il processo corretto a livello di applicazione.

UDP, a parte multiplexing/demultiplexing e una forma di controllo degli errori molto semplice, non aggiunge nulla a IP.

- Prende i messaggi dal processo applicativo, aggiunge il numero di porta di origine e di destinazione per il multiplexing/demultiplexing, aggiunge altri due piccoli campi e passa il segmento risultante al livello di rete. Effettua un tentativo di consegnarlo all'host di destinazione in modalità best-effort.
- Se il segmento arriva a destinazione, UDP utilizza il numero di porta di destinazione per consegnare i dati del segmento al processo applicativo corretto.
- UDP non orientato alla connessione.
- DNS è un tipico esempio di protocollo a livello applicativo che utilizza UDP.

Perché uno sviluppatore dovrebbe scegliere di costruire un'applicazione su UDP anziché su TCP?

- **UDP non introduce alcun ritardo nello stabilire una connessione.**

Non appena un processo applicativo passa dei dati a UDP, quest'ultimo li impacchetta in un segmento che trasferisce immediatamente al livello di rete. TCP, invece, dispone di un meccanismo di controllo della congestione e continua perciò non si adatta particolarmente bene a queste esigenze.

Questo è probabilmente il motivo principale per cui DNS utilizza UDP anziché TCP (con cui risulterebbe molto più lento). HTTP invece usa TCP, dato che l'affidabilità risulta critica per le pagine web con testo.

- **Nessuno stato di connessione:** un server dedicato a una particolare applicazione può generalmente supportare molti più client attivi.
- **Minor spazio usato per l'intestazione del pacchetto.** L'intestazione dei pacchetti TCP aggiunge 20 byte, mentre UDP solo 8.

Come potevamo aspettarci, la posta elettronica, l'accesso a terminali remoti, il Web e il trasferimento di file utilizzano TCP in quanto richiedono un servizio di trasferimento dati affidabile.

UDP viene inoltre utilizzato per trasportare dati di gestione della rete.

In questo caso UDP viene preferito a TCP perché le applicazioni di gestione della rete vanno spesso in esecuzione quando la rete DNS fa uso di UDP per evitare i ritardi dovuti alla creazione di una connessione TCP.

Sia UDP che TCP sono oggi utilizzati per le applicazioni multimediali.

Le applicazioni possono ottenere un trasferimento dati affidabile anche con UDP. Come già menzionato, il protocollo QUIC (Quick UDP Internet Connection, [Iyengar 2015]), utilizzato nel browser Chrome di Google, utilizza UDP.

Struttura dei segmenti UDP.

L'intestazione UDP presenta solo **quattro campi** di *due byte* ciascuno.

- I numeri di porta consentono all'host di destinazione di trasferire i dati applicativi al processo corretto (ossia di effettuare il demultiplexing).

- Il campo lunghezza specifica il numero di byte del segmento UDP (intestazione più dati).

L'host ricevente utilizza il checksum per verificare se sono avvenuti errori nel segmento. In realtà, oltre che sul segmento UDP, il checksum è calcolato anche su alcuni campi dell'intestazione IP.



Figura 3.7 Struttura dei segmenti UDP.

Checksum UDP.

Il **checksum UDP** serve per il rilevamento degli errori. È utilizzato per determinare se i bit del segmento UDP sono stati **alterati** durante il loro trasferimento da sorgente a destinazione (a causa di disturbi nei collegamenti o quando sono stati memorizzati in un router).

Lato mittente UDP effettua il complemento a 1 della somma di tutte le parole da 16 bit nel segmento, e l'eventuale riporto finale viene sommato al primo bit. Tale risultato viene posto nel campo checksum del segmento UDP.

Dato che non sono garantiti né l'affidabilità del singolo collegamento né il rilevamento di errori in memoria, UDP deve mettere a disposizione a livello di trasporto un meccanismo di verifica su base end-to-end se si vuole che il servizio trasferimento dati sia in grado di rilevare eventuali errori.

UDP, in ogni caso, **non fa nulla per risolvere le situazioni di errore**; alcune implementazioni di UDP si limitano a scartare il segmento danneggiato, altre lo trasmettono all'applicazione con un avvertimento.

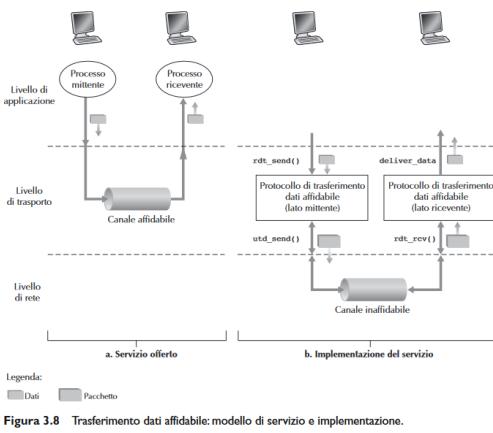


Figura 3.8 Trasferimento dati affidabile: modello di servizio e implementazione.

Principi del trasferimento dati affidabile.

Il problema dell'implementazione di un trasferimento dati affidabile si verifica non solo a livello di trasporto, ma anche a livello di collegamento e di applicazione.

Con un **canale affidabile** a disposizione nessun bit dei dati trasferiti è corrotto o va perduto e tutti i bit sono consegnati nell'ordine di invio. Si tratta precisamente del modello di servizio offerto da **TCP** alle applicazioni per Internet che ne fanno uso.

Ciò è reso difficile dalla possibile inaffidabilità del livello "al di sotto" del protocollo di trasferimento dati.

Per esempio, TCP è un protocollo di trasferimento dati affidabile implementato appoggiandosi a un livello di rete (IP) che non è affidabile end-to-end. Questo livello inferiore è un canale punto a punto inaffidabile.

Nella nostra discussione assumeremo che i pacchetti vengano consegnati nell'ordine con cui sono stati inviati, ma alcuni possono andare persi; vale a dire che il canale sottostante non riordina i pacchetti.

Il lato mittente del protocollo di trasferimento dati sarà invocato tramite una chiamata a **rdt_send()** e trasferirà i dati da consegnare al livello superiore sul lato ricevente.

Quando un **pacchetto raggiunge il lato ricevente del canale**, verrà chiamata **rdt_recv()**.

Vedremo che, oltre a scambiare pacchetti contenenti dati da trasferire, i due lati di rdt necessiteranno anche del reciproco scambio di pacchetti di controllo: entrambi inviano pacchetti tramite una chiamata a **utd_send()** (UDT, unreliable data transfer).

Nel momento in cui il **protocollo rdt voglia consegnare i dati al livello superiore**, lo farà chiamando **deliver_data()**.

Da qui in avanti useremo pacchetto anziché segmento per il livello di trasporto.

In questo paragrafo consideriamo solo il caso di trasferimento dati unidirezionale.

Costruzione di un protocollo di trasferimento dati affidabile.

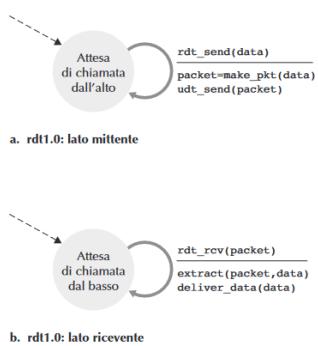
Passiamo ora in rassegna una serie di protocolli via via sempre più complessi, per arrivare a un impeccabile protocollo di trasferimento dati affidabile.

Trasferimento dati affidabile su un canale perfettamente affidabile: rdt1.0.

Canale sottostante completamente affidabile → rdt1.0.

La FSM (a) definisce le operazioni del mittente, l'altra (b) mostra come opera il destinatario.

Esistono due FSM separate, una per il mittente e una per il destinatario. Dato che le FSM della figura hanno un unico stato, le transizioni (indicate dalle frecce) hanno luogo necessariamente tra quello stato e sé stesso. Lo stato iniziale della FSM è indicato dalla freccia tratteggiata.



Il *lato mittente* di rdt accetta semplicemente dati dal livello superiore tramite l'evento **rdt_send(data)**, crea un pacchetto contenente dati con l'azione **make_pkt(data)** e lo invia sul canale. In pratica, l'evento **rdt_send(data)** è il risultato di una chiamata a procedura, per esempio, a **rdt_send()**, da parte dell'applicazione al livello superiore. Con **utd_send(packet)** passo al livello sottostante. Ora devo aspettare di ricevere un ulteriore unità informativa dal livello superiore e immetterla sul canale affidabile delle informazioni.

Lato ricevente, rdt raccoglie i pacchetti dal sottostante canale tramite l'evento **rdt_rcv(packet)**, rimuove i dati dai pacchetti tramite l'azione **extract(packet,data)** e li passa al livello superiore con l'azione **deliver_data(data)**.

In pratica, l'evento **rdt_rcv(packet)** è il risultato di una chiamata a procedura, per esempio a **rdt_rcv()**, da parte del protocollo di livello inferiore.

Figura 3.9 rdt1.0: protocollo per un canale completamente affidabile.

Inoltre, tutti i pacchetti fluiscono dal mittente al destinatario; con un canale prettamente affidabile, non c'è alcun bisogno che il lato ricevente fornisca informazioni al mittente, dato che nulla può andare storto.

Trasferimento dati affidabile su un canale con errori sui bit: rdt2.0.

Canale sottostante con i bit in un pacchetto che possono essere corrotti.

Si assume che tutti i pacchetti trasmessi vengano ricevuti nell'ordine di invio, anche se i loro bit possono essere corrotti.

Questo protocollo infatti fa uso di notifiche(acknowledgment) positive, ovvero "OK", e notifiche negative, "Per favore, ripeti".

Nel contesto di una rete di calcolatori, i protocolli di trasferimento dati affidabili basati su ritrasmissioni sono noti come protocolli ARQ (automatic repeat request) e devono avere tre funzionalità aggiuntive per gli errori:

- **Rilevamento dell'errore:** è richiesto un meccanismo che consenta al destinatario di rilevare gli errori sui bit.

UDP utilizza il campo di checksum per questo preciso scopo. Tali tecniche richiedono l'invio di bit extra (oltre a quelli dei dati da trasferire) tra mittente e destinatario.

- **Feedback del destinatario.** C'è bisogno di una verifica del destinatario.

Le risposte di notifica positiva: ACK e negativa: NAK sono esempi di feedback, rdt2.0 manderà pacchetti ACK e NAK dal destinatario al mittente. In linea di principio, tali pacchetti potrebbero essere costituiti da un solo bit: per esempio, 0 per NAK | 1 per ACK.

- **Ritrasmissione.** Un pacchetto ricevuto con errori sarà ritrasmesso dal mittente.

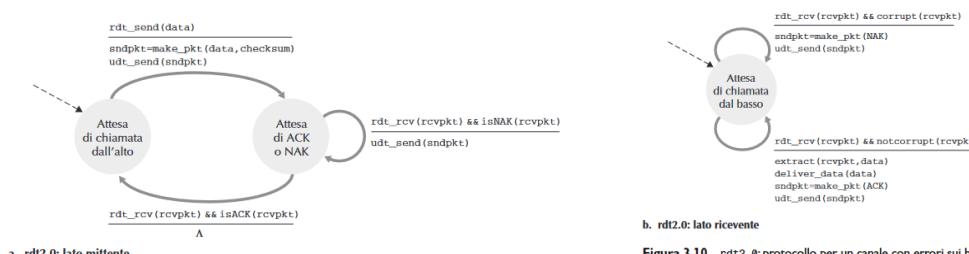


Figura 3.10 rdt2.0: protocollo per un canale con errori sui bit.

Il *lato mittente* di rdt2.0 presenta due stati. In quello di sinistra, il protocollo lato mittente sta attendendo i dati da raccogliere dal livello superiore. Quando si verifica l'evento **rdt_send(data)**, il mittente crea un pacchetto (**sndpkt**) contenente i dati da inviare, insieme al **checksum** e infine spedisce il pacchetto tramite l'operazione **udt_send(sndpkt)**. Nello stato di destra, il protocollo mittente è in attesa di un pacchetto ACK o NAK dal destinatario.

Se riceve un ACK il mittente sa che il pacchetto trasmesso più di recente è stato **ricevuto correttamente**.

Invece, se riceve un NAK, il protocollo **ritrasmette l'ultimo pacchetto** e attende una risposta alla ritrasmissione.

Il mittente non invia nuovi dati finché non è certo che il destinatario abbia ricevuto correttamente il pacchetto corrente. È proprio per questo comportamento che i protocolli quali rdt2.0 sono noti come protocolli stop-and-wait.

La FSM *lato ricevente* di rdt2.0 ha ancora un solo stato. All'arrivo del pacchetto, il destinatario risponde o con un ACK o con un NAK, a seconda che il pacchetto sia corrotto o meno.

Il protocollo rdt2.0 sembrerebbe funzionare ma, sfortunatamente, presenta un grave difetto; infatti, non abbiamo tenuto conto della possibilità che i pacchetti ACK o NAK possano a loro volta essere alterati. Dovremmo aggiungere dei bit di checksum ai pacchetti ACK/NAK per rilevare tali errori.

Prendiamo in considerazione **tre possibilità per gestire gli ACK e i NAK corrotti**.

• Si può ricorrere all'*aggiunta di bit di checksum sufficienti a consentire al mittente non solo di trovare, ma anche di correggere gli errori sui bit*. Ciò risolve il problema solo per un canale che può danneggiare pacchetti, ma non perderli.

• Un altro approccio prevede semplicemente che il *mittente rinvii il pacchetto di dati corrente a seguito della ricezione di un pacchetto ACK o NAK alterato*. Questo approccio, tuttavia, introduce pacchetti duplicati nel canale. La fondamentale difficoltà insita nella duplicazione di pacchetti è che il destinatario non sa se l'ultimo ACK o NAK inviato sia stato ricevuto correttamente dal mittente. Di conseguenza, non può sapere "a priori" se un pacchetto in arrivo contenga dati nuovi o rappresenti una ritrasmissione.

Una soluzione semplice a questo nuovo problema (adottata in quasi tutti i protocolli di trasferimento dati, tra cui TCP) consiste nell'aggiungere un campo al pacchetto dati, obbligando il mittente a numerare i propri pacchetti dati con un numero di sequenza nel nuovo campo. Al destinatario sarà sufficiente controllare questo numero per sapere se il pacchetto ricevuto rappresenti o meno una ritrasmissione. Per questo semplice protocollo **stop-and-wait**: logica per cui trasmetto, aspetto un riscontro e poi ritrasmetto fino a quando non sono sicuro che il pacchetto è stato correttamente ricevuto, un numero di sequenza da 1 bit sarà sufficiente, dato che consentirà al destinatario di sapere se il mittente stia ritrasmettendo un pacchetto o inviandone uno già trasmesso.

Trasferimento dati affidabile su un canale con perdite ed errori sui bit: rdt3.0.

Supponiamo ora che il canale di trasmissione, oltre a danneggiare i bit, possa anche smarrire i pacchetti, un evento non raro sulle odierne reti di calcolatori (Internet compresa).

Per lo smarrimento pacchetti la soluzione consiste nell'utilizzo del checksum, numeri di sequenza, pacchetti ACK e ritrasmissione .

Per gestire il bit danneggiati dobbiamo aggiungere al protocollo un nuovo meccanismo.

Supponiamo che il mittente spedisca un pacchetto dati e che questo o l'ACK corrispondente del ricevente vada smarrito.

In entrambi i casi, il mittente non otterrà alcuna risposta da parte del destinatario.

Se il mittente è disposto ad attendere un tempo sufficiente per essere certo dello smarrimento del pacchetto, può semplicemente ritrasmettere il pacchetto di dati.

Quanto tempo deve attendere il mittente? Questo ritardo relativo al caso peggiore è difficile da stimare, oltre che da sapere con certezza. L'approccio adottato nella pratica è scegliere in modo assennato un valore di tempo tale per cui la perdita di pacchetti risulti probabile, anche se non garantita. Se non si riceve un ACK in questo lasso di tempo, il pacchetto viene ritrasmesso. Notiamo che il mittente potrebbe ritrasmettere un pacchetto che sperimenta un ritardo particolarmente lungo, anche se né il pacchetto di dati stessa né il suo ACK sono stati smarriti. Ciò introduce la possibilità di pacchetti dati duplicati sul canale tra mittente e destinatario. Fortunatamente, il protocollo rdt2.2 presenta già una funzionalità (i numeri di sequenza) per gestire il caso dei pacchetti duplicati.

Dal punto di vista del mittente, la ritrasmissione è una panacea. Il mittente non sa se un pacchetto dati sia andato perduto, se sia stato smarrito un ACK o se il pacchetto o l'ACK abbiano semplicemente subito un notevole ritardo.

In tutti questi casi, l'azione intrapresa è la stessa: **ritrasmettere**.

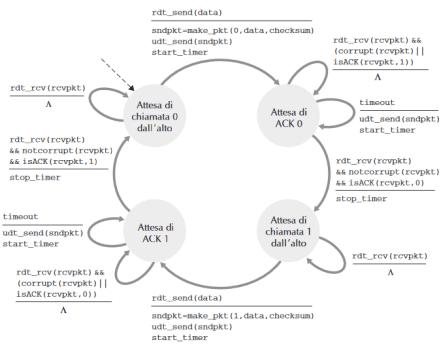


Figura 3.15 Mittente rdt3.0.

Implementare un meccanismo di ritrasmissione basato sul tempo richiede un contatore (countdown timer) in grado di segnalare al mittente l'avvenuta scadenza di un dato lasso di tempo. Il mittente dovrà quindi essere in grado di:

1. inizializzare il contatore ogni volta che invia un pacchetto (che si tratti del primo invio o di una ritrasmissione),
2. rispondere a un interrupt generato dal timer con l'azione appropriata e;
3. fermare il contatore.

Protocolli per il trasferimento dati affidabile con pipeline.

Il protocollo rdt3.0 è corretto dal punto di vista funzionale, ma non ha buone prestazioni poiché si tratta di un protocollo stop-and-wait.

Un protocollo stop-and-wait presenta un basso utilizzo del mittente e quindi si riscontra un throughput basso.

I protocolli di rete possono limitare il rendimento dell'hardware di rete sottostante. Se si considerano i tempi di elaborazione del protocollo ai livelli inferiori presso il mittente e il destinatario, così come i ritardi di elaborazione e di accodamento che si verificherebbero sui router intermedi tra il mittente e il destinatario. Considerare questi non farebbe altro che incrementare ulteriormente il ritardo, aumentato le prestazioni scadenti.

La soluzione a questo particolare problema è semplice: anziché operare in modalità stop-and-wait, si consente al mittente di inviare più pacchetti senza attendere gli acknowledgment.

Se si consente al mittente di trasmettere tre pacchetti senza dover aspettare gli acknowledgment, l'utilizzo viene sostanzialmente triplicato. Questa tecnica è nota come **pipelining** (da pipe, letteralmente, "tubo").

Le conseguenze su un protocollo di trasferimento dati affidabile sono le seguenti:

- L'intervallo dei numeri di sequenza disponibili deve essere incrementato, dato che ogni pacchetto in transito (senza contare le ritrasmissioni) deve presentare un numero di sequenza univoco e che ci potrebbero essere più pacchetti in transito ancora in attesa di acknowledgment.
- I lati di invio e di ricezione dei protocolli possono dover memorizzare in un buffer più di un pacchetto. Inoltre, come vedremo più avanti, anche al destinatario potrebbe essere richiesta la memorizzazione dei pacchetti ricevuti correttamente.
- La quantità di numeri di sequenza necessari e i requisiti di buffer dipendono dal modo in cui il protocollo di trasferimento dati reagisce ai pacchetti smarriti, alterati o troppo in ritardo.

Si possono identificare due approcci di base verso la risoluzione degli errori con pipeline: Go-Back-N e ripetizione selettiva (selective repeat).

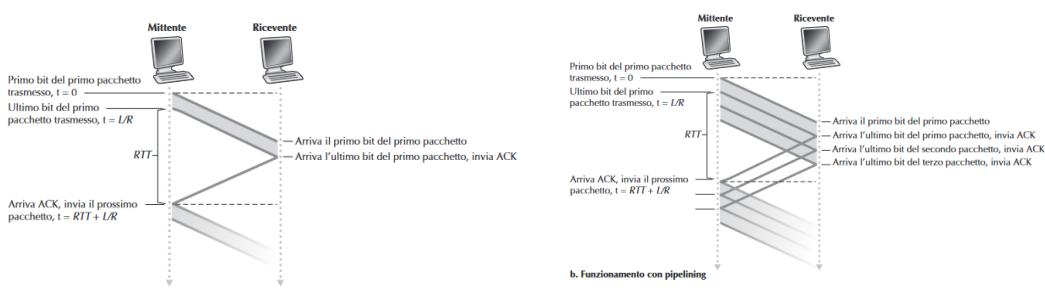


Figura 3.18 Invio per il protocollo stop-and-wait e con pipelining.

Go-Back-N (GBN).

In un protocollo **Go-Back-N** (GBN) il mittente può trasmettere più pacchetti senza dover attendere alcun *acknowledgment*, ma non può avere più di un dato numero massimo consentito N di pacchetti (se disponibili) in attesa di acknowledgment nella pipeline.

Il protocollo GBN consente al mittente di “riempire la tubatura” con pacchetti, evitando così i problemi di scarso utilizzo del canale che abbiamo riscontrato nei protocolli stop-and-wait.

Se definiamo *base* come il **numero di sequenza del pacchetto più vecchio che non ha ancora ricevuto un acknowledgment** e *nextseqnum* **il più piccolo numero di sequenza inutilizzato** (il numero di sequenza del prossimo pacchetto da inviare), allora si possono identificare quattro intervalli di numeri di sequenza.

- I numeri di sequenza nell’intervallo [0, base–1] corrispondono ai pacchetti già trasmessi e che hanno ricevuto acknowledgment.
- L’intervallo [base, nextseqnum–1] corrisponde ai pacchetti inviati, ma che non hanno ancora ricevuto alcun acknowledgment.
- I numeri di sequenza nell’intervallo [nextseqnum, base+N–1] possono essere utilizzati per i pacchetti da inviare immediatamente.
- Infine, i numeri di sequenza maggiori o uguali a base+N non possono essere utilizzati finché il mittente non riceva un acknowledgment relativo a un pacchetto che si trova nella pipeline ed è ancora privo di riscontro.

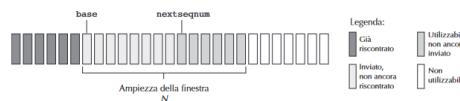


Figura 3.19 Visione del mittente sui numeri di sequenza nel protocollo Go-Back-N.

N viene spesso chiamato **ampiezza della finestra** (window size) e il protocollo GBN viene detto protocollo a finestra scorrevole (sliding-window protocol). Ci si potrebbe chiedere per quale motivo dovremmo limitare il numero di pacchetti in sospeso a N; perché non consentire un numero illimitato di tali pacchetti? Il controllo di flusso rappresenta una delle ragioni per impostare un limite al mittente.

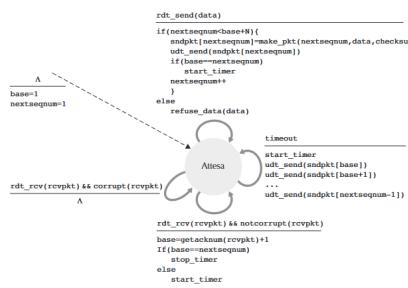


Figura 3.20 Descrizione con automa esteso del mittente GBN.

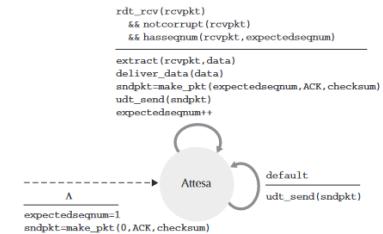


Figura 3.21 Descrizione con automa esteso del ricevente GBN.

Il mittente GBN deve rispondere a tre tipi di evento.

- **Invocazione dall’alto.** Quando dall’alto si chiama `rdt_send()`, come prima cosa il mittente controlla se la finestra sia piena, ossia se vi siano N pacchetti in sospeso senza acknowledgment. Se la finestra non è piena, crea e invia un pacchetto e le variabili vengono aggiornate di conseguenza. Se la finestra è piena, il mittente restituisce i dati al livello superiore che, presumibilmente, ritenterà più tardi.
- **Ricezione di un ACK.** Nel nostro protocollo GBN, l’acknowledgment del pacchetto con il numero di sequenza n verrà considerato un acknowledgment cumulativo (cumulative acknowledgment), che indica che tutti i pacchetti con un numero di sequenza minore o uguale a n sono stati correttamente ricevuti dal destinatario.
- **Evento di timeout.** Il nome del protocollo “Go-Back-N” deriva dal comportamento del mittente in presenza di pacchetti persi o eccessivamente in ritardo. Come nei protocolli stop-and-wait, si usa ancora un contatore per risolvere il problema di pacchetti dati o acknowledgment persi. Quando si verifica un timeout, il mittente invia nuovamente tutti i pacchetti spediti che ancora non hanno ricevuto un acknowledgment.

Anche le azioni del destinatario GBN sono semplici. Se un pacchetto con numero di sequenza n viene ricevuto correttamente ed è in ordine il destinatario manda un ACK per quel pacchetto e consegna i suoi dati al livello superiore. In tutti gli altri casi, il destinatario scarta i pacchetti e rimanda un ACK per il pacchetto in ordine ricevuto più di recente. Notiamo che, essendo i pacchetti consegnati uno alla volta al livello superiore, se il pacchetto k è stato ricevuto e consegnato, tutti i pacchetti con un numero di sequenza inferiore sono anch’essi stati consegnati. Pertanto, l’uso di acknowledgment cumulativi è una scelta naturale per GBN.

Nel nostro protocollo GBN il destinatario scarta i pacchetti fuori sequenza.

Ovviamente lo svantaggio di eliminare un pacchetto ricevuto correttamente è che la sua ritrasmissione potrebbe andare persa o essere alterata, il che richiederebbe ulteriori ritrasmissioni.

Ripetizione selettiva.

Esistono, tuttavia, scenari in cui lo stesso GBN ha **problemi di prestazioni**. In particolare, quando l'ampiezza della finestra e il prodotto tra larghezza di banda e ritardo sono entrambi grandi, nella pipeline si possono trovare numerosi pacchetti. Un errore su un solo pacchetto può pertanto provocare un elevato numero di ritrasmissioni, in molti casi inutili. Al crescere della probabilità di errore sul canale, la pipeline può saturarsi a causa di queste ritrasmissioni non necessarie.

Come suggerisce il nome, i **protocolli a ripetizione selettiva** (SR, selective-repeat protocol) **evitano le ritrasmissioni non necessarie facendo ritrasmettere al mittente solo quei pacchetti su cui esistono sospetti di errore** (ossia, smarrimento o alterazione). Questa forma di ritrasmissione a richiesta e personalizzata costringe il destinatario a mandare acknowledgment specifici per i pacchetti ricevuti in modo corretto.

Si userà nuovamente un'ampiezza di finestra pari a N per limitare il numero di pacchetti privi di acknowledgment nella pipeline. Tuttavia, a differenza di GBN, il mittente avrà già ricevuto gli ACK di qualche pacchetto nella finestra.

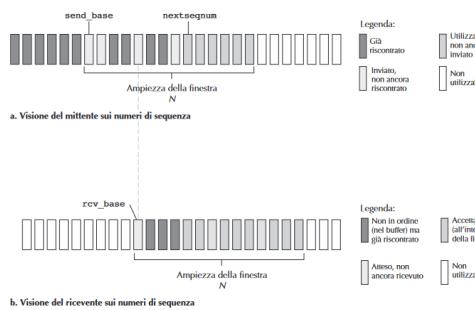


Figura 3.23 Visione del mittente e del ricevente a ripetizione selettiva sullo spazio dei numeri di sequenza.

1. Dati ricevuti dall'alto. Quando si ricevono dati dall'alto, il mittente SR controlla il successivo numero di sequenza disponibile per il pacchetto. Se è all'interno della finestra del mittente, i dati vengono impacchettati e inviati; altrimenti sono salvati nei buffer o restituiti al livello superiore per una successiva ritrasmissione, come in GBN.

2. Timeout. Vengono usati ancora i contatori per cautelarsi contro la perdita di pacchetti. Ora però ogni pacchetto deve avere un proprio timer logico, dato che al timeout sarà ritrasmesso un solo pacchetto. Si può utilizzare solo contatori hardware per simulare le operazioni di più timer logici [Varghese 1997].

3. ACK ricevuto. Se si riceve un ACK, il mittente SR etichetta tale pacchetto come ricevuto, ammesso che sia nella finestra. Se il numero di sequenza del pacchetto è uguale a *send_base*, la base della finestra si muove verso il pacchetto che non ha ricevuto acknowledgement con il più piccolo numero di sequenza. Se la finestra si sposta e ci sono pacchetti non trasmessi con numero di sequenza che ora cade all'interno della finestra, questi vengono trasmessi.

Figura 3.24 Eventi e azioni di un mittente SR.

Il destinatario SR invia un riscontro per i pacchetti correttamente ricevuti sia in ordine sia fuori sequenza. Questi vengono memorizzati in un buffer finché non sono stati ricevuti tutti i pacchetti mancanti (ossia quelli con numeri di sequenza più bassi), momento in cui un blocco di pacchetti può essere trasportato in ordine al livello superiore.

Nei protocolli SR ciò significa che le finestre del mittente e del destinatario non sempre coincidono.

La mancanza di sincronizzazione tra le finestre del mittente e del destinatario ha conseguenze importanti quando abbiamo a che fare con un intervallo finito di numeri di sequenza.

Trasporto orientato alla connessione: TCP.

Connessione TCP.

TCP viene detto **orientato alla connessione** in quanto, prima di effettuare lo scambio dei dati, i processi devono effettuare l'**handshake**, ossia devono scambiarsi informazioni client e server.

Lo stato della connessione TCP risiede completamente nei due sistemi periferici e non negli elementi di rete intermedi (router e switch a livello di collegamento), questi ultimi non salvano lo stato della connessione TCP.

Una **connessione TCP** offre un servizio **full-duplex**: su una connessione TCP tra il processo A su un host e il processo B su un altro host, i dati a livello di applicazione possono fluire dal processo A al processo B nello stesso momento in cui fluiscono in direzione opposta.

Una **connessione TCP** è anche **punto a punto**, ossia ha luogo tra un singolo mittente e un singolo destinatario. Il cosiddetto "multicast", ossia il trasferimento di dati da un mittente a molti destinatari in un'unica operazione, con TCP non è possibile.

Supponiamo che il processo di un host voglia inizializzare una connessione con il processo in un altro host.

Il processo applicativo client innanzitutto informa il livello di trasporto client di voler stabilire una connessione verso un processo nel server. Un programma client in Python effettua l'operazione così: `clientSocket.connect((serverName, serverPort))` dove `serverName` è il nome del server, mentre `serverPort` identifica il processo sul server.

Il TCP in esecuzione sul client procede quindi a stabilire una connessione con il TCP del server.

Il client invia per primo uno speciale segmento TCP; il server risponde con un secondo segmento speciale TCP; infine, il client risponde con un terzo segmento speciale. I primi due non trasportano payload, ossia non hanno dati a livello applicativo; il terzo può invece trasportare informazioni utili. Dato che i due host si scambiano tre segmenti, la connessione viene spesso detta **handshake a tre vie**.

Una volta instaurata una connessione TCP, i due processi applicativi si possono scambiare dati.

Il processo client manda un flusso di dati attraverso la socket. Questi, quando hanno attraversato il punto di uscita, sono nelle mani di TCP in esecuzione nel client.

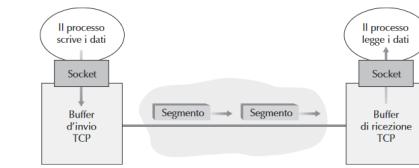


Figura 3.28 Buffer di invio e ricezione TCP.

TCP dirige i dati al buffer di invio della connessione, uno dei buffer riservato durante l'handshake a tre vie, da cui, di tanto in tanto, preleverà blocchi di dati e li passerà al livello di rete.

TCP deve "spedire tali dati in segmenti quando è più conveniente". La massima quantità di dati prelevabili e posizionabili in un segmento viene limitata dalla dimensione massima di segmento (**MSS**, *maximum segment size*). L'MSS rappresenta la massima quantità di dati a livello di applicazione nel segmento e non la massima dimensione del segmento TCP con intestazioni incluse.

TCP accoppia ogni blocco di dati del client a una intestazione TCP, andando pertanto a formare segmenti TCP. Questi vengono passati al sottostante livello di rete, dove sono incapsulati separatamente nei datagrammi IP a livello di rete che vengono poi immessi nella rete. Quando all'altro capo TCP riceve un segmento, i dati del segmento vengono memorizzati nel buffer di ricezione della connessione TCP (Figura 3.28). L'applicazione legge il flusso di dati da questo buffer. Ogni lato della connessione presenta un proprio buffer di invio e di ricezione.

Abbiamo visto che una connessione TCP è costituita da buffer, variabili e una connessione socket al processo in un host e da un altro insieme di buffer, variabili e connessione socket al processo in un altro host. Come precedentemente accennato, negli elementi di rete tra gli host (router, switch e ripetitori) non vengono allocati alla connessione buffer o variabili.

Struttura dei segmenti TCP.

Il segmento TCP consiste di campi intestazione e di un campo contenente un blocco di dati proveniente dall'applicazione.

La MSS limita la dimensione massima del campo dati di un segmento. TCP, quando invia un file di grandi dimensioni come, per esempio, un'immagine che fa parte di una pagina web, di solito frammenta il file in porzioni di dimensione MSS (a eccezione dell'ultima, che avrà generalmente dimensioni inferiori). Le applicazioni interattive, invece, trasmettono spesso blocchi di dati più piccoli di MSS.

L'intestazione TCP occupa comunemente 20 byte (12 in più dell'intestazione UDP).

Struttura segmenti TCP: come in UDP, l'intestazione include numeri di porta di origine e di destinazione, utilizzati per il multiplexing/demultiplexing dei dati da e verso le applicazioni del livello superiore, e un campo checksum. L'intestazione dei segmenti TCP comprende poi i seguenti campi.

- Il **campo numero di sequenza** (sequence number) e il **campo numero di acknowledgment** (acknowledgment number), entrambi di 32 bit, vengono utilizzati dal mittente e dal destinatario TCP per implementare il trasferimento dati affidabile.
- Il **campo finestra di ricezione** (receive window), di 16 bit, viene utilizzato per il controllo di flusso.
- Il **campo lunghezza dell'intestazione** TCP in multipli di 32 bit. L'intestazione TCP ha lunghezza variabile a causa del campo delle opzioni TCP. Generalmente, il campo delle opzioni è vuoto e, pertanto, la lunghezza consueta è di 20 byte.
- Il **campo opzioni** (options), facoltativo e di lunghezza variabile, viene utilizzato quando mittente e destinatario negoziano la dimensione massima del segmento (MSS) o come fattore di scala per la finestra nelle reti ad alta velocità.
- Il **campo flag** è di 6 bit. Il bit ACK viene usato per indicare che il valore trasportato nel campo di acknowledgment è valido; ossia, il segmento contiene un acknowledgment per un segmento che è stato ricevuto con successo.

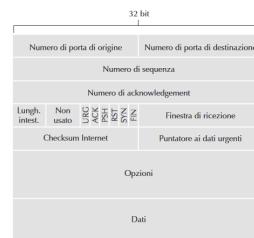


Figura 3.29 Struttura dei segmenti TCP.

Numeri di sequenza e numeri di acknowledgment.

Due tra i campi più importanti dell'intestazione del segmento TCP contengono il **numero di sequenza** e il **numero di acknowledgment**, che rappresentano una parte critica del servizio di trasferimento dati affidabile proprio di TCP.

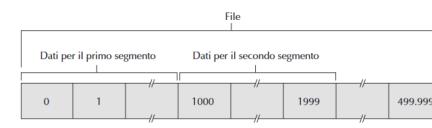


Figura 3.30 Divisione di un file di dati in segmenti TCP.

Telnet: un caso di studio per i numeri di sequenza e di acknowledgment.

Telnet, è un protocollo a livello di applicazione impiegato per il login remoto che utilizza TCP ed è progettato per funzionare tra qualsiasi coppia di host.

Telnet è quindi, un protocollo di rete, utilizzato tramite interfaccia a riga di comando per fornire all'utente sessioni di login remoto. È un protocollo client-server basato su TCP; i client solitamente si connettono alla porta 23 sul server. Stabilisce una connessione interattiva ad un qualche altro servizio su un server internet.

Telnet è usato ogni tanto nel debug di servizi di networking come i server SMTP e HTTP, in quanto rappresenta un modo semplice per mandare comandi al server ed esaminare le risposte.

Timeout e stima del tempo di andata e ritorno.

TCP, come pure il protocollo rdt, utilizza un **meccanismo di timeout e ritrasmissione** per **recuperare i segmenti persi**. Il timeout dovrebbe essere più grande del tempo di andata e ritorno sulla connessione (RTT), ossia del tempo trascorso da quando si invia un segmento a quando se ne riceve l'acknowledgment, altrimenti ci sarebbero delle ritrasmissioni inutili. Ma di quanto deve essere maggiore? E, innanzitutto, come dovrebbe essere stimato RTT?

Stima del tempo di andata e ritorno.

Cominciamo lo **studio della gestione dei timer TCP** considerando come il protocollo stimi il tempo di andata e ritorno tra mittente e destinatario. L'RTT misurato di un segmento, denotato come:

- **SampleRTT**, è la quantità di tempo che intercorre tra l'istante di invio del segmento (ossia quando viene passato a IP) e quello di ricezione dell'acknowledgment del segmento.

In ogni istante di tempo, SampleRTT viene valutato per uno solo dei segmenti trasmessi e per cui non si è ancora ricevuto acknowledgment, il che comporta la misurazione di un nuovo valore di SampleRTT a ogni RTT. TCP non calcola mai il SampleRTT per i segmenti ritrasmessi, cioè lo calcola soltanto per i segmenti trasmessi una volta sola.

Ovviamente, i campioni variano da segmento a segmento in base alla congestione nei router e al diverso carico sui sistemi periferici.

Per effettuare una stima risulta naturale calcolare una media ponderata dei valori di SampleRTT, che TCP chiama **EstimatedRTT**. Quando si ottiene un nuovo SampleRTT, TCP aggiorna EstimatedRTT secondo la formula:

Tale media attribuisce maggiore importanza ai campioni recenti rispetto a quelli vecchi.

Oltre ad avere una stima di RTT, è anche importante possedere **la misura della sua variabilità**. [RFC 6298] definisce la variazione RTT, **DevRTT**, come una stima di quanto SampleRTT generalmente si discosta da EstimatedRTT:

$$\text{DevRTT} = (1 - \beta) \times \text{DevRTT} + \beta \times | \text{SampleRTT} - \text{EstimatedRTT} |$$

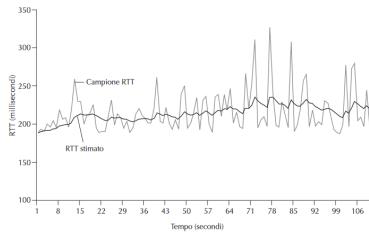


Figura 3.32 Campioni e stima di RTT.

Si noti che DevRTT è un EWMA della differenza tra SampleRTT e EstimatedRTT. Se i valori di SampleRTT presentano fluttuazioni limitate, allora DevRTT sarà piccolo; in caso di notevoli fluttuazioni, DevRTT sarà grande.

Impostazione e gestione del timeout di ritrasmissione.

Dati i valori di EstimatedRTT e DevRTT, quali valori andrebbero usati per l'intervallo di timeout di TCP? L'intervallo non può essere inferiore a quello di EstimatedRTT → ritrasmissioni non necessarie.

Ma l'intervallo stesso non dovrebbe essere molto maggiore di EstimatedRTT → TCP non ritrasmetterebbe rapidamente il segmento perduto, il che comporterebbe gravi ritardi sul trasferimento dei dati. È pertanto auspicabile impostare il timeout a EstimatedRTT più un certo margine che dovrebbe essere grande quando c'è molta fluttuazione nei valori di SampleRTT e piccolo in caso contrario. Qui entra in gioco il valore di DevRTT. Tutti questi aspetti vengono presi in considerazione dal modo in cui TCP determina l'intervallo di timeout di ritrasmissione:

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 \times \text{DevRTT}$$

Trasferimento dati affidabile.

Abbiamo già visto che il servizio di Internet a livello di rete (**servizio IP**) **non è affidabile**: non garantisce la consegna in sequenza, né la consegna stessa dei datagrammi, né l'integrità dei dati.

Con il servizio IP, i datagrammi possono sovrappollare i buffer dei router e non raggiungere la destinazione o arrivare in ordine casuale e con bit alterati.

Dato che sono inglobati nei datagrammi IP, anche i segmenti a livello di trasporto possono risentire di questi problemi.

TCP crea un servizio di **trasporto dati affidabile** al di sopra del servizio inaffidabile e best-effort di IP, assicurando che **il flusso di byte che i processi leggono dal buffer di ricezione TCP non sia alterato, non abbia buchi, non presenti duplicazioni e rispetti la sequenza originaria**; in altre parole, il flusso di dati in arrivo è esattamente quello spedito.

Supponiamo che i dati vengano inviati dall'Host A all'Host B e che il primo stia trasmettendo un file di grandi dimensioni.

Esistono tre eventi principali relativi alla trasmissione e ritrasmissione dei dati:

- TCP incapsula i dati che gli giungono dall'applicazione in un segmento e lo passa a IP. Ciascun segmento include un numero di sequenza che rappresenta il numero del primo byte di dati del segmento nel flusso di byte. Inoltre, se il timer non è già in funzione per qualche altro segmento, TCP lo avvia quando il segmento viene passato a IP.

- Timeout: TCP risponde ritrasmettendo il segmento che lo ha causato e quindi riavviando il timer.

- Arrivo del segmento di acknowledgment con un valore valido nel campo ACK.

Raddoppio dell'intervallo di timeout.

Varianti utilizzate dalla maggior parte delle implementazioni TCP. La prima riguarda la lunghezza dell'intervallo di timeout dopo la scadenza di un timer. Con questa modifica, in tutti i casi in cui si verifica un timeout, TCP ritrasmette il segmento con il più basso numero di sequenza che non abbia ancora ricevuto acknowledgment. Tuttavia, ogni volta che questo si verifica, TCP imposta il successivo intervallo di timeout al doppio del valore precedente, anziché derivarlo dagli ultimi EstimatedRTT e DevRTT. Tuttavia, tutte le volte che il timer viene avviato dopo uno degli altri due eventi possibili (ossia la ricezione di dati dall'applicazione superiore e la ricezione di un ACK), il TimeoutInterval viene ricavato dai più recenti valori di EstimatedRTT e DevRTT.

Questa modifica offre una forma limitata di controllo di congestione. La scadenza del timer viene probabilmente causata dalla congestione nella rete, ossia troppi pacchetti arrivano presso una (o più) code dei router nel percorso tra l'origine e la destinazione, provocando l'eliminazione dei pacchetti e/o lunghi ritardi di accodamento. Nei periodi di congestione, se le sorgenti continuano a ritrasmettere pacchetti, la congestione può peggiorare. TCP si comporta in maniera più rispettosa della rete, dato che ciascun mittente ritrasmette dopo intervalli sempre più lunghi.

Ritrasmissione rapida.

Uno dei problemi legati alle ritrasmissioni è che il periodo di timeout può rivelarsi relativamente lungo. Quando si smarrisce un segmento, il lungo periodo di timeout impone al mittente di ritardare il nuovo invio del pacchetto perso, incrementando di conseguenza il ritardo end-to-end. Fortunatamente, il mittente può in molti casi rilevare la perdita dei pacchetti ben prima che si verifichi l'evento di timeout grazie agli ACK duplicati relativi a un segmento il cui ACK è già stato ricevuto dal mittente.

Dato che in molti casi il mittente invia un gran numero di segmenti, se uno di questi viene smarrito ci saranno probabilmente molti ACK duplicati. Se il mittente TCP riceve tre ACK duplicati per lo stesso dato, considera questo evento come indice che il segmento che lo segue è andato perduto (nei problemi alla fine del capitolo ci chiede- remo il motivo per cui il mittente attenda tre ACK ripetuti anziché uno solo). Nel caso in cui siano stati ricevuti tre ACK duplicati, il mittente TCP effettua una ritrasmissione rapida (fast retransmit) [RFC 5681], rispedendo il segmento mancante prima che scada il timer (si veda la Figura 3.37 dove viene perduto il secondo segmento e ritrasmesso prima che il suo timer scada). In TCP con ritrasmissione rapida il seguente frammento di codice sostituisce l'evento di ACK ricevuto nella Figura 3.33.

Avevamo già fatto notare come, quando in un vero protocollo si implementa un meccanismo di timeout e ritrasmissione, sorgano molti problemi. Le procedure qui descritte, evolute nel tempo e risultato di più di vent'anni di esperienza con i timer TCP, confermano questa osservazione.

Go-Back-N o ripetizione selettiva?

è un protocollo GBN o SR? Ricordiamo che gli acknowledgment TCP sono cumulativi e che i segmenti ricevuti correttamente, ma in modo disordinato, non vengono notificati singolarmente dal destinatario. Quindi (Figure 3.33

e 3.19), il mittente TCP deve solo memorizzare il numero di sequenza più basso tra i byte trasmessi che non hanno ancora ricevuto acknowledgment (SendBase) e il numero di sequenza del successivo byte da inviare (NextSeqNum). In questo senso, TCP assomiglia molto a un protocollo di tipo GBN. Tuttavia, esistono alcune differenze chiave tra TCP e Go-Back-N. Molte implementazioni TCP memorizzano in un buffer i segmenti ricevuti correttamente, ma non in ordine [Stevens 1994].

Una modifica proposta di TCP, il cosiddetto riscontro selettivo (selective acknowledgment) [RFC 2018], consente al destinatario di mandare acknowledgment in modo selettivo per i segmenti non in ordine anziché cumulativamente per l'ultimo segmento ricevuto senza errori e nell'ordine giusto. Se combinato con la ritrasmissione selettiva (vale a dire, evitando la ritrasmissione dei segmenti per cui si è già ricevuto un acknowledgment in modo selettivo dal destinatario), TCP è molto simile a un generico protocollo SR. È quindi opportuno classificare il meccanismo di ripristino dagli errori proprio di TCP come un ibrido tra i protocolli di tipo GBN e SR.

Controllo di flusso.

Quando la connessione TCP riceve byte corretti e in sequenza, li posiziona nel buffer di ricezione.

Il processo applicativo associato legge i dati da questo buffer, ma non necessariamente nell'istante in cui arrivano.

Se l'applicazione è relativamente lenta nella lettura dei dati può accadere che il mittente mandi in overflow il buffer di ricezione inviando molti dati troppo rapidamente.

TCP offre un servizio di controllo di flusso (flow-control service) alle proprie applicazioni per evitare che il mittente saturi il buffer del ricevente. Il controllo di flusso è pertanto un servizio di confronto sulla velocità, dato che paragona:

- La frequenza di invio del mittente.
- La frequenza di lettura dell'applicazione ricevente.

I mittenti TCP possono anche essere rallentati dalla congestione nella rete IP.

Questa forma di controllo del mittente viene detta **controllo di congestione** (*congestion control*).

Sebbene le azioni intraprese dal controllo di flusso e di congestione siano simili (il rallentamento del mittente), sono causate da ragioni molto differenti.

TCP offre controllo di flusso facendo mantenere al mittente una variabile chiamata finestra di ricezione (receive window) che, fornisce al mittente un'indicazione dello spazio libero disponibile nel buffer del destinatario.

Dato che TCP è full-duplex, i due mittenti mantengono finestre di ricezione distinte.

UDP non offre controllo di flusso. In una tipica implementazione UDP, tale protocollo metterà in coda i segmenti in un buffer di dimensione finita che "precede" la corrispondente socket (che fa da punto di collegamento con il processo). Se il processo non legge i segmenti dal buffer a velocità sufficiente, si verifica un overflow e alcuni segmenti verranno persi.

Gestione della connessione TCP: stabilire e rilasciare una connessione TCP.

Un processo in esecuzione in un host (*client*) vuole iniziare una **connessione** con un altro processo in un altro host (*server*).

Il processo applicativo client informa il lato client di TCP di voler stabilire una connessione verso un processo nel server.

Il TCP nel client, quindi, procede a stabilire una connessione TCP con il TCP nel server:

1. **TCP lato client invia uno speciale segmento al TCP lato server.** Questo segmento speciale non contiene dati a livello applicativo, ma uno dei bit nell'intestazione del segmento, il bit SYN, è posto a 1.

Inoltre, il client sceglie a caso un numero di sequenza iniziale (client_isn) e lo pone nel campo numero di sequenza del segmento SYN iniziale. Quest'ultimo viene incapsulato in un datagramma IP e inviato al server.

2. Quando il datagramma IP contenente il segmento TCP SYN arriva all'host server (ammesso che arrivi), il **server estrae il segmento dal datagramma, alloca i buffer e le variabili TCP alla connessione e invia un segmento di connessione approvata** al **client TCP**. Anche questo segmento non contiene dati a livello applicativo, ma nella sua intestazione il bit SYN è posto a 1. Il campo ACK assume il valore client_isn+1. Infine, il server sceglie il proprio numero di sequenza iniziale (server_isn) e lo pone nel campo del numero di sequenza. È come se il segmento stesse dicendo: "Ho ricevuto il tuo pacchetto SYN per iniziare una connessione con il tuo numero di sequenza iniziale, client_isn. Sono d'accordo nello stabilire questa connessione. Il mio numero di sequenza iniziale è server_isn".

Il **segmento di connessione approvata** viene talvolta detto segmento **SYNACK**.

3. Alla ricezione del segmento SYNACK, anche il **client** alloca buffer e variabili alla connessione. L'**host client invia** quindi al **server** un altro **segmento** in risposta al segmento di connessione approvata del server. Tale operazione viene svolta dal client ponendo il valore server_isn+1 nel campo ACK dell'intestazione del segmento TCP. Il bit SYN è posto a zero, dato che la connessione è stata stabilita.

Una volta completati questi tre passi, gli **host client e server possono scambiarsi segmenti contenenti dati**. In ciascuno di questi futuri segmenti, il bit SYN sarà posto a zero. Notiamo che al fine di stabilire la connessione, i due host si scambiano tre pacchetti → handshake a tre vie.

Ciascuno dei due processi che partecipano alla connessione può terminarla. E, in tal caso, le "risorse" negli host (ossia buffer e variabili) vengono deallocate.

- Il **processo applicativo client** invia un comando di chiusura inviando un segmento TCP speciale al processo server dove troviamo il bit **FIN** con valore **1**. Quando il server riceve questo segmento, risponde inviando un acknowledgment al client. Il server spedisce quindi il proprio segmento di shutdown, con il bit FIN uguale a 1. Infine, il client manda a sua volta un acknowledgment a quest'ultimo segmento del server.
- Notiamo che anche il **server** potrebbe scegliere di terminarla. Ciò spinge il client TCP a inviare un segmento con il bit **FIN** impostato a **1** e a entrare nello stato **FIN_WAIT_1**. Quando si trova in questo stato, il client TCP attende dal server un segmento TCP con un acknowledgment e, quando lo riceve, entra nello stato **FIN_WAIT_2**, in cui il client attende un altro segmento dal server con bit FIN impostato a 1. Dopo aver ricevuto questo segmento, il client TCP manda un acknowledgment ed entra nello stato **TIME_WAIT**, che consente al client TCP di inviare nuovamente l'ultimo acknowledgment nel caso in cui l'ACK vada perduto. Dopo l'attesa, la connessione viene formalmente chiusa e tutte le risorse sul lato client (compresi i numeri di porta) vengono rilasciate.

Consideriamo ora che cosa succeda quando un host riceva un segmento TCP i cui **numeri di porta** o il cui **indirizzo IP di origine** non corrispondano ad alcuna socket attiva nell'host. In tal caso invierà al **mittente** un **segmento speciale di reset** con il bit **RST** impostato a **1**, allo scopo di comunicare alla sorgente: "Non ho una socket per quel segmento. Per favore non rimandarlo". Quando un host riceve un pacchetto UDP il cui numero di porta di destinazione non corrisponde a una socket UDP attiva, invia uno speciale datagramma ICMP.

Nmap è uno strumento potente, che può raccogliere informazioni non solo sulle porte TCP aperte, ma anche sulle porte UDP aperte, sui firewall e sulle loro configurazioni, e anche sulle versioni delle applicazioni e dei sistemi operativi. Molto di ciò viene fatto manipolando i segmenti per la gestione delle connessioni TCP.

Potete scaricare nmap per altri sistemi operativi da <http://www.nmap.org>.

Principi del controllo di congestione.

Servizio di trasferimento dati affidabile a fronte di una perdita di pacchetti. Tale perdita è tipicamente il risultato di un overflow dei buffer nei router quando il traffico in rete diventa eccessivo. La ritrasmissione di pacchetti tratta quindi un sintomo della congestione di rete (la perdita di uno specifico segmento a livello di trasporto), cioè il tentativo da parte di troppe sorgenti di inviare dati a ritmi troppo elevati. In questo caso sono richiesti meccanismi per adeguare l'attività dei mittenti in relazione al traffico.

Consideriamo ora il problema del controllo di congestione in un contesto generale.

Cause e costi della congestione.

Controllo della congestione con **tre** scenari. In ciascun caso vedremo perché la rete sia congestionata e ne valuteremo le conseguenze in termini di basso utilizzo delle risorse e di scarse prestazioni percepite dai sistemi periferici.

Scenario 1: due mittenti e un router con buffer illimitati.

Scenario di congestione più semplice possibile: due host (Ae B) con una connessione che **condivide** un singolo router intermedio.

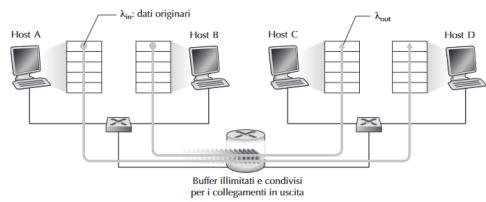


Figura 3.43 Scenario di congestione 1: due connessioni che condividono un hop con buffer illimitato.

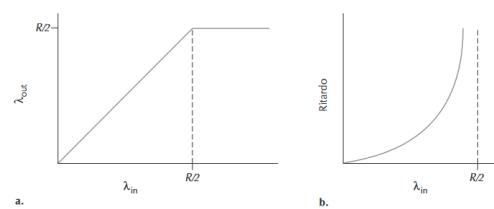


Figura 3.44 Scenario di congestione 1: throughput e ritardi in funzione della frequenza trasmittiva dell'host.

Ipotizziamo che un'applicazione nell'Host A stia inviando dati sulla connessione a una frequenza media di λ_{in} byte/s.

Tali dati sono originali, nel senso che ciascuna unità di dati viene mandata nella socket solo una volta.

Il sottostante protocollo a livello di trasporto è semplice. I dati vengono incapsulati e inviati, senza porre rimedio a eventuali errori.

Il tasso al quale l'Host A presenta traffico al router in questo primo scenario è pertanto λ_{in} byte/s. L'Host B opera in modo simile, e noi assumeremo per semplicità che stia trasmettendo anch'esso a λ_{in} byte/s. I pacchetti dall'Host A e dall'Host B passano attraverso un router e un collegamento uscente condiviso di capacità R. Il router possiede buffer che gli consentono di memorizzare i pacchetti entranti quando la loro velocità di arrivo supera la capacità del collegamento

Perfino in questo scenario (estremamente) idealizzato abbiamo già trovato un costo dovuto alla congestione delle reti: quando il tasso di arrivo dei pacchetti si avvicina alla capacità del collegamento, si rilevano lunghi ritardi di accodamento.

Scenario 2: due mittenti e un router con buffer limitati.

Una prima modifica dello scenario precedente consiste nell'assumere che la dimensione dei buffer nel router sia limitata. Una conseguenza pratica di quest'ipotesi è che i pacchetti che giungono in un buffer già pieno sono scartati. In secondo luogo, supponiamo che le due connessioni siano affidabili: se un pacchetto che contiene un segmento a livello di trasporto viene scartato dal router, il mittente prima o poi lo ritrasmetterà. Dato che i pacchetti possono essere ritrasmessi, dobbiamo ora prestare attenzione all'uso della locuzione "tasso di invio".

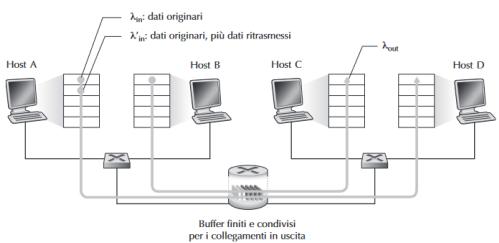


Figura 3.45 Scenario 2: due host (con ritrasmissioni) e un router con buffer di dimensione finita.

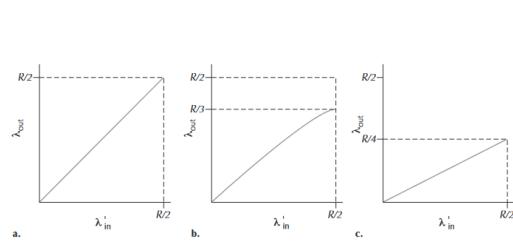


Figura 3.46 Prestazioni dello Scenario 2 con buffer di dimensione finita.

In particolare, denotiamo ancora il tasso di trasmissione verso la socket con λ_{in} byte/s, e indichiamo con λ'_{in} byte/s il tasso al quale il livello di trasporto invia segmenti (contenenti dati originali e dati ritrasmessi), una grandezza talvolta detta carico offerto (offered load) alla rete.

Le prestazioni che si riscontrano in questo scenario dipendono fortemente da come si effettua la ritrasmissione.

Innanzitutto, consideriamo il caso poco probabile in cui l'Host A sia in grado di determinare, come per magia, se il buffer nel router abbia spazio a disposizione e trasmetta pertanto un pacchetto solo quando il buffer è libero.

In questo caso non si verificherebbe alcun smarrimento. Dal punto di vista del throughput, le prestazioni sono ideali: tutto quanto viene trasmesso è ricevuto. Notiamo che la velocità di invio media dell'host in questo scenario non supera $R/2$, visto che abbiamo ipotizzato che nessun pacchetto vada smarrito.

Consideriamo ora il caso, un po' più realistico, in cui il mittente ritrasmette solo quando è certo che un pacchetto sia andato perduto.

Un'ipotesi leggermente forzata. Tuttavia, l'host mittente potrebbe impostare il proprio timeout con un valore sufficientemente grande da essere praticamente certo che il pacchetto di cui non si è ancora ricevuto acknowledgment sia stato perduto.

Prendiamo in esame, infine, la situazione in cui il mittente possa andare in timeout prematuramente e ritrasmettere un pacchetto che abbia subito ritardi in coda, ma non sia stato perduto. In questo caso, sia il pacchetto originale sia quello ritrasmesso possono raggiungere il destinatario. Ovviamente, al destinatario è sufficiente una copia di tale pacchetto e scarterà le altre. In questo caso, il lavoro effettuato dal router per instradare la copia ritrasmessa del pacchetto è sprecato, dato che il destinatario avrà già ricevuto la copia originale. Il router avrebbe potuto utilizzare meglio la capacità trasmittiva del collegamento trasmettendo un altro pacchetto.

In questo scenario, il throughput assumerà asintoticamente il valore $R/4$ quando il carico offerto tende a $R/2$.

Scenario 3: quattro mittenti, router con buffer finiti e percorsi composti da più collegamenti.

In questo caso supponiamo che i pacchetti siano trasmessi da quattro host, ciascuno su percorsi composti da due collegamenti sovrapposti tra loro (Figura 3.47); ciascun host, inoltre, utilizza un meccanismo di timeout e ritrasmissione per implementare il servizio affidabile di trasferimento dati, e tutti e quattro hanno lo stesso valore di λ_{in} .

Supponiamo anche che la capacità dei collegamenti dei router sia di R byte/s.

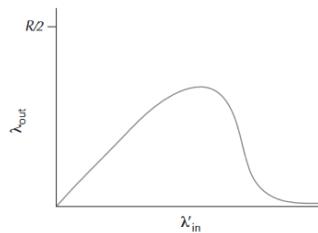
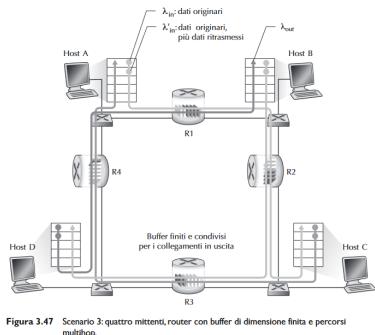


Figura 3.48 Prestazioni dello Scenario 3 (buffer di dimensione finita e percorsi multihop).

Ogni qualvolta un pacchetto viene scartato sul router del secondo hop, il lavoro effettuato dal router del primo hop nell'instradamento del pacchetto verso il secondo router finisce per essere "sprecato". La rete avrebbe funzionato altrettanto bene (anzi, altrettanto male) se il primo router avesse semplicemente scartato il pacchetto e fosse rimasto inattivo. Più nello specifico, la capacità trasmissiva utilizzata dal primo router per instradare il pacchetto al secondo potrebbe essere utilizzata in modo più proficuo trasmettendo un altro pacchetto. Per esempio, nel selezionare un pacchetto per la trasmissione, sarebbe preferibile che il router desse priorità ai pacchetti che hanno già superato un certo numero di router.

Quindi, anche in questo caso, vediamo un costo legato all'eliminazione dei pacchetti per via della congestione: quando un pacchetto viene scartato lungo il percorso, la capacità trasmissiva, utilizzata sui collegamenti per instradare il pacchetto fino al punto in cui è scartato, risulta sprecata.

Approcci al controllo di congestione.

Ci sono **2 principali orientamenti al controllo di congestione** utilizzati nella pratica.

Ad alto livello, si può distinguere tra *livelli di rete* che **offrono** o **meno assistenza esplicita** al livello di trasporto per controllare la congestione.

- **Controllo di congestione end-to-end.**

TCP deve necessariamente utilizzare questo approccio end-to-end, dato che il livello IP non offre feedback relativamente alla congestione della rete. La perdita di segmenti TCP (indicata da un timeout o da acknowledgment triplicati) viene considerata chiara indicazione di congestione di rete e TCP diminuisce, di conseguenza, l'ampiezza della propria finestra.

- **Controllo di congestione assistito dalla rete.**

I **componenti a livello di rete** (ossia i *router*) forniscono un feedback esplicito al mittente sullo stato di congestione della rete.

Questo avviso può essere semplicemente un bit che indica traffico su un collegamento o anche qualcosa di più sofisticato.

L'informazione di congestione viene solitamente fornita dalla rete al mittente in due modi. Può essere trasmesso un avviso diretto da un router al mittente tramite un chokepacket ("pacchetto di strozzatura"), che riferisce: "Sono congestionato!".

Il secondo tipo di notifica ha luogo quando un router impone un campo in un pacchetto che fluisce dal mittente al destinatario, per indicare congestione. Alla ricezione di un pacchetto marcato, il destinatario notifica al mittente l'indicazione di congestione. Notiamo che questa forma di notifica richiede quantomeno un RTT.

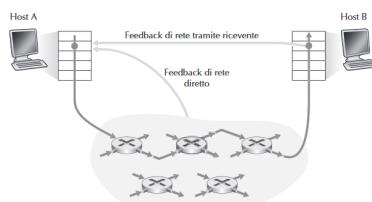


Figura 3.49 Due modalità per ricevere feedback riguardo la congestione di rete.

Controllo di congestione TCP.

L'approccio scelto da **TCP** consiste nell'imporre a ciascun mittente un limite alla velocità di invio sulla propria connessione in funzione della congestione di rete percepita. Se il mittente TCP si accorge di condizioni di scarso traffico sul percorso che porta alla destinazione, incrementa il proprio tasso trasmissivo; se, invece, percepisce traffico lungo il percorso, lo riduce. Ma tale approccio solleva tre domande. Innanzitutto, come può il mittente TCP limitare la velocità di invio del traffico sulla propria connessione? Secondo, come percepisce la congestione sul percorso che porta alla destinazione? E, infine, quale algoritmo dovrebbe essere usato dal mittente per variare la velocità di invio in funzione della congestione end-to-end?

Innanzitutto, esaminiamo come **TCP possa ridurre la velocità di invio del traffico sulla propria connessione**.

Il meccanismo di controllo di congestione TCP fa tener traccia agli estremi della connessione di una variabile aggiuntiva: la finestra di congestione (congestion window), indicata con **cwnd**, che impone un vincolo alla velocità di immissione di traffico sulla rete da parte del mittente.

Al fine di concentrarci sul controllo di congestione (anziché su quello di flusso), assumiamo che il buffer di ricezione sia sufficientemente capiente da poter ignorare il vincolo della finestra di ricezione; pertanto, la quantità di dati che non hanno ricevuto acknowledgment è limitata soltanto da cwnd. Assumiamo inoltre che il mittente abbia sempre dati da inviare, per cui la finestra di congestione sia sempre completamente in uso. Modificando il valore di cwnd, il mittente può regolare la velocità di invio dei dati sulla propria connessione.

Consideriamo ora come il mittente TCP percepisce la presenza di congestione sul cammino verso la destinazione. Definiamo “evento di perdita” per il mittente TCP l’occorrenza o di un timeout o della ricezione di tre ACK duplicati da parte del destinatario.

In presenza di una congestione eccessiva, uno o più buffer dei router lungo il percorso vanno in overflow, causando l’eliminazione di un datagramma (che contiene un segmento TCP). Il datagramma eliminato, a sua volta, costituisce un evento di perdita presso il mittente (sotto forma di timeout o come ricezione di tre ACK duplicati), che lo considera come un’indicazione di congestione sul percorso tra sé e il destinatario.

Dato il meccanismo di modifica del valore di cwnd per controllare il tasso di trasmissione, la questione critica è come un mittente TCP debba determinare il tasso a cui dovrebbe trasmettere. Se i mittenti TCP tutti insieme trasmettessero troppo velocemente, potrebbero congestionare la rete portandola al tipo di collasso di congestione.

Tuttavia, se i mittenti TCP fossero troppo cauti e trasmettessero troppo lentamente, potrebbero sottoutilizzare l’ampiezza di banda della rete; i mittenti TCP potrebbero trasmettere a tasso più elevato senza congestione la rete. Allora come fanno i mittenti TCP a determinare la loro velocità di trasmissione in modo da non congestione la rete, ma allo stesso tempo utilizzare tutta la banda disponibile? I mittenti TCP sono esplicitamente coordinati o esiste un approccio distribuito in cui stabiliscono i loro tassi di trasmissione basandosi solo su informazioni locali? TCP risponde a queste domande sulla base dei seguenti principi guida.

- Un segmento perso implica congestione, quindi i tassi di trasmissione del mittente TCP dovrebbero essere decrementati quando un segmento viene perso.
- Un acknowledgment indica che la rete sta consegnando i segmenti del mittente al ricevente e quindi il tasso di trasmissione del mittente può essere aumentato quando arriva un acknowledgment non duplicato. Di conseguenza la finestra di congestione può essere incrementata.
- Rilevamento della larghezza di banda. Avendo acknowledgment che indicano che il cammino dalla sorgente alla destinazione è privo di congestione ed eventi di perdita che indicano un percorso congestionato, la strategia di TCP per regolare la velocità di trasmissione è di incrementarla in risposta all’arrivo di acknowledgment finché non si verifica un evento di perdita; a questo punto la velocità di trasmissione viene decrementata. Il mittente TCP aumenta quindi la sua velocità di trasmissione per rilevare a che tasso trasmisivo comincia a verificarsi la congestione, rallenta e quindi inizia di nuovo la fase di rilevamento per trovare se il punto di inizio della congestione è cambiato.

Possiamo ora prendere in considerazione i dettagli del celebrato **algoritmo di controllo di congestione di TCP**.

L’algoritmo presenta tre componenti o fasi principali: (1) **slow start**, (2) **congestion avoidance** e (3) **fast recovery**.

Slow start e congestion avoidance sono componenti obbligatorie di TCP e differiscono nel modo in cui aumentano la grandezza di cwnd in risposta agli acknowledgment ricevuti. Vedremo tra breve che slow start incrementa la dimensione della cwnd molto più rapidamente di congestion avoidance. La fast recovery è suggerita, ma non obbligatoria, per i mittenti TCP.

Slow start.

Quando si stabilisce una connessione TCP, il valore di **cwnd** viene in genere **inizializzato a 1 MSS**, il che comporta una velocità di invio iniziale di circa MSS/RTT. Se MSS = 500 byte e RTT = 200 ms, la velocità iniziale è solo di circa 20 kbps. Dato che la banda disponibile alla connessione può essere molto più grande di MSS/RTT, il mittente TCP gradirebbe scoprire velocemente la banda disponibile. Quindi, durante la fase iniziale, detta slow start, il valore di cwnd parte da 1 MSS e si incrementa di 1 MSS ogni volta che un segmento trasmesso riceve un acknowledgment. Questo processo ha come effetto il raddoppio della velocità trasmissiva a ogni RTT. Quindi, in TCP, la velocità di trasmissione parte lentamente, ma cresce in modo esponenziale durante la fase di slow start.

Quando, tuttavia, dovrebbe terminare questa crescita esponenziale? Se c’è un evento di perdita (e quindi una congestione) indicato da un evento di timeout e quando vengono rilevati tre acknowledgment duplicati, nel qual caso TCP opera una ritrasmissione rapida ed entra nello stato di fast recovery, come discusso di seguito.

Congestion avoidance.

Quando TCP entra nello stato di **congestion avoidance**, il valore di **cwnd** è circa la **metà di quello che aveva l’ultima volta in cui era stata rilevata la congestione** (che potrebbe essere ancora dietro l’angolo). Quindi, invece di raddoppiare il valore di cwnd ogni RTT, TCP adotta un approccio più conservativo, incrementando cwnd di 1 MSS ogni RTT. Ciò si può ottenere in diversi modi: un approccio comune è l’incremento da parte del mittente TCP della propria cwnd di $MSS \times (MSS/cwnd)$ byte ogni qualvolta riceva un nuovo acknowledgment. Ciascun ACK in arrivo (assumendo un ACK per segmento) incrementa l’ampiezza della finestra di congestione di 1/10 MSS e quindi, dopo la ricezione degli acknowledgment relativi a tutti e dieci i segmenti, il valore della finestra di congestione sarà stato aumentato di un MSS.

Ma quando finisce l’incremento lineare (1 MSS per RTT) durante la congestion avoidance? L’algoritmo di congestion avoidance quando si verifica un timeout si comporta nello stesso modo di slow start: il valore di cwnd è posto uguale a 1 MSS e il valore di ssthresh viene impostato alla metà del valore di cwnd al momento del timeout. Si ricordi, tuttavia, che un evento di perdita può essere anche il risultato della ricezione di tre acknowledgment duplicati.

Fast recovery.

Durante la fase di **fast recovery** il valore di **cwnd** è **incrementato di 1 MSS per ogni ACK duplicato ricevuto relativamente al segmento perso che ha causato l’entrata di TCP nello stato di fast recovery**. Infine, quando arriva un ACK per il segmento perso, TCP entra nello stato di congestion avoidance dopo aver ridotto il valore di cwnd. Se si verifica un timeout vi è invece una transizione dallo stato di fast recovery a quello di slow start dopo avere effettuato le stesse azioni presenti sia in slow start che in congestion avoidance: il valore di cwnd è posto a 1 MSS e il valore di ssthresh è impostato a metà del valore di cwnd nel momento in cui si è riscontrato l’evento di perdita.

Fast recovery è un componente raccomandato, ma non obbligatorio di TCP. È interessante sapere che una prima versione di TCP, nota come **TCP Tahoe**, portava in modo incondizionato la finestra di congestione a 1 MSS ed entrava nella fase di slow start dopo qualsiasi tipo di evento di perdita. La versione più recente, **TCP Reno**, adotta invece fast recovery.

L'evoluzione della finestra di congestione TCP con Reno e con Tahoe, il valore iniziale della soglia è di 8 MSS. Per i primi 8 cicli di trasmissione Tahoe e Reno si comportano allo stesso modo. La finestra di congestione cresce in modo esponenziale e supera la soglia nel quarto turno di trasmissione. La finestra di congestione, quindi, sale in modo lineare fino al verificarsi di un triplice ACK duplicato, appena dopo l'ottavo turno di trasmissione. Notiamo che la finestra di congestione vale 12 MSS quando si verifica l'evento di perdita. La soglia ssthresh viene quindi impostata a $0,5 \times \text{cwnd} = 6$ MSS. Con TCP Reno, la finestra di congestione è posta a 6 MSS e poi cresce in modo lineare. Con TCP Tahoe, la finestra di congestione viene posta a 1 MSS e cresce in modo esponenziale fino a raggiungere la soglia ssthresh, punto dal quale cresce linearmente.

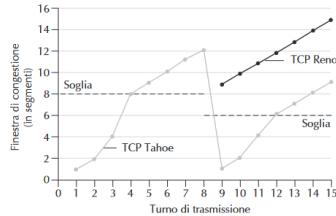


Figura 3.52 Evoluzione della finestra di congestione TCP (Tahoe e Reno).

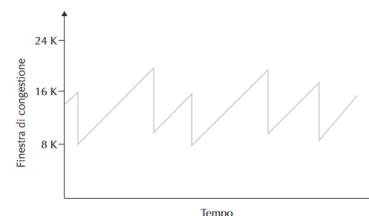


Figura 3.53 Controllo di congestione a incremento additivo e decremento moltiplicativo.

Retrospettiva sul controllo di congestione di TCP.

Dopo aver visto i dettagli delle fasi di slow start, congestion avoidance e fast recovery, vale la pena fare un passo indietro per averne una visione complessiva. Ignorando la parte iniziale della fase di slow start, quando inizia la connessione, e assumendo che le perdite siano indicate da un triplice ACK duplicato piuttosto che da eventi di timeout, il controllo di congestione di TCP consiste in un incremento additivo lineare della cwnd pari a 1 MSS per RTT e quindi di un decremento moltiplicativo che dimezza la cwnd in corrispondenza di un evento di triplice ACK duplicato. Il controllo di congestione di TCP è spesso indicato come una forma di controllo di congestione incremento additivo, decremento moltiplicativo (AIMD, additive increase multiplicative-decrease). Il controllo di congestione AIMD si basa sull'intuizione sulla rilevazione della larghezza di banda: TCP incrementa linearmente l'ampiezza della propria finestra di congestione e quindi del tasso di trasmissione, finché si verifica un evento di triplice ACK duplicato. Quindi decremente la propria finestra di congestione di un fattore due, ma riprende ancora a crescere linearmente per capire se ci sia ulteriore ampiezza di banda disponibile.

La maggior parte delle attuali implementazioni di TCP utilizza l'algoritmo Reno.

L'*algoritmo Vegas* tenta di evitare la congestione mantenendo allo stesso tempo un buon throughput.

L'idea alla base di Vegas è

- (1) rilevare la congestione nei router tra origine e destinazione prima che si verifichi un evento di perdita;
- (2) abbassare la velocità in modo lineare quando si profila l'imminente perdita di un pacchetto.

L'evento di perdita viene predetto osservando il valore RTT: più l'RTT dei pacchetti è grande, maggiore sarà la congestione nei router.

Throughput di TCP.

Assodato il comportamento a dente di sega di TCP, è naturale chiedersi quale potrebbe essere il **throughput medio** (ossia la frequenza media di trasmissioni) di una connessione TCP prolungata.

Queste ipotesi forniscono un modello macroscopico estremamente semplificato del comportamento a regime di TCP. La rete elimina un pacchetto dalla connessione quando la velocità sale a W/RTT ; la velocità viene poi dimezzata e quindi incrementata di MSS/RTT a ogni RTT fino a quando raggiunge ancora W/RTT . Questo processo si ripete in continuazione.

Il throughput (cioè la velocità) cresce in modo lineare tra i due valori estremi.

Fairness.

Consideriamo K connessioni TCP, ciascuna con un differente percorso end-to-end, ma che passano tutte attraverso un collegamento con capacità trasmissiva di R bps che costituisce il collo di bottiglia del sistema; con ciò intendiamo che tutti gli altri collegamenti non sono congestionati e dispongono di elevata capacità trasmissiva, in confronto a quella del collegamento che fa da collo di bottiglia. Supponiamo che ogni connessione stia trasferendo un file di grandi dimensioni e che non ci sia traffico UDP attraverso il collo di bottiglia. Si dice che un meccanismo di controllo di congestione sia equo (fair) se la velocità trasmissiva media di ciascuna connessione è approssimativamente R/K ; in altre parole, ciascuna connessione ottiene la stessa porzione di banda del collegamento.

AIMD è un algoritmo fair? La risposta deve tener conto del fatto che le connessioni TCP possono aver inizio in istanti diversi e quindi possono avere diverse dimensioni di finestra.

Consideriamo il caso semplice di due connessioni TCP che condividono un collegamento con capacità trasmissiva R (Figura 3.54). Assumiamo che le connessioni abbiano gli stessi valori di MSS e RTT (e pertanto abbiano uguali finestre di congestione e lo stesso throughput), che debbano trasmettere una gran quantità di dati e che non vi siano altre connessioni TCP o datagrammi UDP che attraversano questo collegamento condiviso.

L'obiettivo dovrebbe essere ottenere throughput situati vicino all'intersezione tra la bisettrice e la linea di massimo utilizzo della banda.

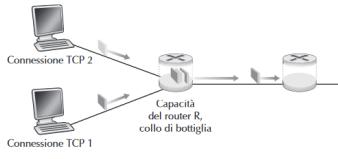


Figura 3.54 Due connessioni TCP che condividono un singolo collegamento che fa da collo di bottiglia.

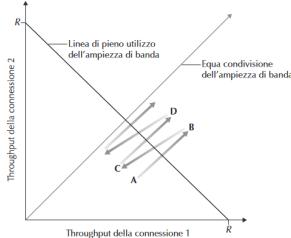


Figura 3.55 Throughput delle connessioni TCP 1 e 2.

Supponiamo che le dimensioni della finestra TCP siano tali che a un certo istante di tempo le connessioni 1 e 2 raggiungano i throughput corrispondenti al punto A della Figura 3.55. Dato che la porzione di banda del collegamento utilizzata congiuntamente è minore di R, non si verificheranno perdite e le due connessioni aumenteranno la loro finestra di 1 MSS per RTT come risultato dell'algoritmo di congestion avoidance. Di conseguenza, il throughput congiunto procede lungo la semiretta a 45° (pari incremento per entrambe le connessioni) uscente dal punto A. La banda congiunta potrebbe essere maggiore di R, e si potrebbe quindi verificare una perdita di pacchetti.

Se le connessioni 1 e 2 subiscono una perdita di pacchetti quando raggiungono i throughput indicati dal punto B, allora decrementeranno le loro finestre di un fattore 2. I throughput raggiunti di conseguenza si troveranno pertanto sul punto C, a metà strada lungo un segmento che collega il punto B all'origine. Essendo l'utilizzo di banda condivisa minore di R in prossimità del punto C, le due connessioni ancora una volta incrementano il loro throughput lungo la linea a 45° passante per C. Al punto D si possono ancora verificare delle perdite, nel qual caso le due connessioni decrementeranno di nuovo l'ampiezza delle loro finestre di un fattore 2, e così via. Dovreste esservi convinti che la banda utilizzata dalle due connessioni può fluttuare lungo la linea di equa condivisione della banda e che le due connessioni convergeranno a questo comportamento indipendentemente dal punto del piano in cui si trovano inizialmente. Sebbene basato su numerose ipotesi, tale scenario fornisce un'idea intuitiva del perché TCP porti a un'equa ripartizione della banda tra le connessioni.

Nel nostro scenario abbiamo ipotizzato che solo connessioni TCP attraversino il collo di bottiglia, che abbiano lo stesso RTT e che non ci siano connessioni multiple tra coppie di host. Nella pratica, queste condizioni non si verificano quasi mai e quindi le applicazioni client/server ottengono porzioni di banda assai diverse. In particolare, è stato dimostrato che quando più connessioni condividono un collo di bottiglia, quelle con RTT inferiore sono in grado di acquisire più rapidamente larghezza di banda su un particolare collegamento, non appena la banda si libera. In altre parole, aprono le proprie finestre di congestione più rapidamente e quindi avranno throughput superiori rispetto alle connessioni con RTT più alto [Lakshman 1997].

Fairness e UDP.

Abbiamo appena visto come il meccanismo della finestra di congestione consenta al controllo di congestione TCP di regolare il tasso trasmissivo delle applicazioni. Molte *applicazioni multimediali*, quali la fonia e la videoconferenza, non fanno uso di TCP: **non vogliono che il loro tasso trasmissivo venga ridotto**, anche se la rete è molto congestionata. Piuttosto, queste applicazioni preferiscono utilizzare UDP, che non incorpora il controllo di congestione; in tal modo possono immettere il proprio audio e video sulla rete a frequenza costante e occasionalmente perdere pacchetti, piuttosto che non perderli, ma dover ridurre il loro tasso trasmissivo a livelli "equi" nei momenti di traffico.

Fairness e connessioni TCP parallele.

Anche se potessimo forzare il traffico UDP a comportarsi in modo equo, il problema della fairness non sarebbe tuttavia completamente risolto, perché nulla può impedire a un'applicazione basata su TCP di usare **più connessioni in parallelo**.

Per esempio, spesso i *browser web* usano **più connessioni TCP** in parallelo per trasferire il contenuto delle pagine. Nella maggior parte dei browser è possibile configurare il numero esatto di connessioni multiple. Le applicazioni che utilizzano connessioni in parallelo ottengono una porzione di banda maggiore sui collegamenti congestionati. Consideriamo per esempio un collegamento di capacità R cui accedono nove applicazioni client/server, ciascuna delle quali utilizza una connessione TCP. Se giunge un'altra applicazione con una connessione TCP, allora la frequenza trasmissiva di tutte le applicazioni sarà circa uguale a R/10. Ma se, invece, la nuova applicazione usa 11 connessioni TCP in parallelo, allora otterrà un'allocazione iniqua superiore a R/2. Dato che il traffico web è assai diffuso in Internet, le connessioni in parallelo non sono rare.

Capitolo 4.

Data la sua complessità, il **livello di rete** può essere in due parti interagenti:

- Il **piano dei dati**: racchiude tutte le funzioni che ogni router implementa singolarmente per determinare come un datagramma che arriva a uno dei collegamenti in entrata del router è inoltrato a uno dei collegamenti in uscita del router;
- Il **piano di controllo**: racchiude la logica globale di rete che controlla come i datagrammi sono instradati tra i router su un percorso end-to-end tra sorgente e destinazione.

Ad un livello di astrazione alto, il livello di rete prende dall'host mittente i segmenti del livello di trasporto, li incapsula in un datagramma e li trasmette ad un router vicino; nell'host destinatario estrae i segmenti e li consegna al livello di trasporto.

Le funzioni principali del livello rete sono quella di **forwarding (inoltro)** e quella di **routing (instradamento)**.

- Con *inoltro* si fa riferimento all'azione locale del router che associa i pacchetti da un'interfaccia di ingresso ad una di uscita. Per inoltrare i pacchetti i router estraggono i valori (instradamento) da uno o più campi dell'header che utilizzano come indice nella forwarding table. Il risultato indica a quale interfaccia di uscita il pacchetto debba essere diretto.

È implementato generalmente in hardware perché avviene su scala temporale molto piccola.

- Con *instradamento* si indica invece il processo globale di rete che determina i percorsi dei pacchetti nel loro viaggio dalla sorgente alla destinazione. Coincide con l'operazione di "riempimento" delle forwarding table.

Il livello di rete deve determinare il percorso che i pacchetti devono seguire tramite algoritmi di instradamento (algoritmi di routing). È implementato in software perché avviene su scale temporali più grandi.

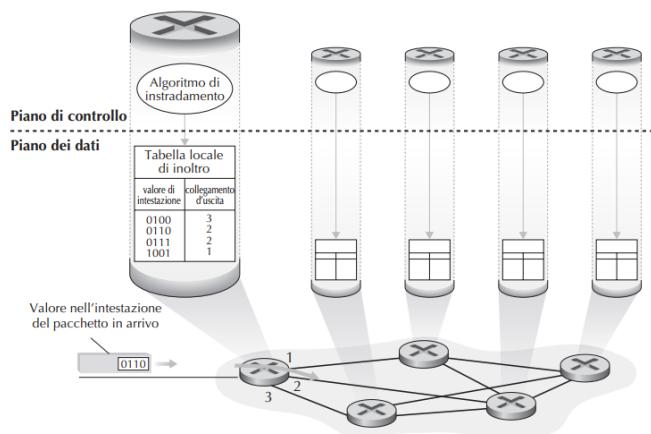


Figura 4.2 Gli algoritmi di instradamento determinano i valori nelle tabelle di inoltro.

Modello di servizio della rete, che definisce le caratteristiche del trasporto end-to-end di pacchetti tra host di origine e di destinazione. Consideriamo ora alcuni servizi che il livello di rete potrebbe offrire:

- **Consegna garantita.** Questo servizio assicura che il pacchetto giunga, prima o poi, alla propria destinazione.
- **Consegna ordinata.** Questo servizio garantisce che i pacchetti giungano alla destinazione nell'ordine in cui sono stati inviati.
- **Banda minima garantita.** Questo servizio a livello di rete emula il comportamento di un collegamento trasmissivo con bit rate specificato (per esempio, 1 Mbps) tra host di invio e di destinazione, anche se l'effettivo percorso end-to-end può attraversare diversi collegamenti fisici.
- **Servizi di sicurezza.** Il livello di rete dell'host sorgente può cifrare tutti i datagrammi inviati; il livello di rete nell'host di destinazione avrà il compito di decifrarli. Con questo tipo di servizio, la riservatezza viene fornita a tutti i segmenti del livello di trasporto.

Il **livello di rete** mette a disposizione un servizio **best-effort** cioè "col massimo impegno possibile", per cui non c'è garanzia che i pacchetti vengano ricevuti nell'ordine in cui sono stati inviati e non è garantita la loro consegna. Non c'è garanzia sul ritardo end-to-end, né su una larghezza di banda minima. Nonostante ciò, il livello rete è best-effort by design, perché si dimostra essere efficiente e soprattutto più veloce se combinato con determinate politiche di controllo implementate ai livelli superiori.

Il modello di servizio *best-effort* combinato con un'adeguata larghezza di banda si è dimostrato abbastanza buono da supportare una gamma di applicazioni quali i servizi di streaming video come Netflix, le applicazioni voice-and-video-over-IP e Skype e Facetime.

Servizi Datagram e a circuito virtuale.

Esistono due tipologie di servizio a livello rete: **Datagram** e a **circuito virtuale**.

- Il servizio **datagram**: i pacchetti entrano nella rete e seguono ognuno un percorso diverso in base a diversi fattori.

Tutti arrivano a destinazione con un ordine di arrivo non definito. Ogni nodo intermedio deve avere un modo per fare routing, ossia ogni pacchetto deve essere identificato con gli indirizzi IP mittente e destinazione.

- Il servizio a *circuito virtuale* utilizza invece un **canale virtuale logico** attraverso cui viaggiano i pacchetti.
La difficoltà iniziale sta nel creare, mediante algoritmi di routing, il canale virtuale. Una volta creato può essere numerato con un id, un identificatore di flusso, ed utilizzato per instradare i pacchetti. Il router deve solo verificare l'id ed instradare il pacchetto nel relativo canale.

Approccio tradizionale e approccio SDN.

In un approccio tradizionale, le funzioni di routing e di forwarding, il piano dei dati e quello di controllo, sono implementate internamente ad ogni router. Il router, quindi, riempie le proprie tabelle con algoritmi di routing poi sceglie con algoritmi di forwarding il link di uscita per l'inoltro. La gestione di rete è statica e affidata ai singoli operatori che configurano switch e router.

In un approccio alternativo, un controller remoto, separato fisicamente dai router, calcola e distribuisce le tabelle di inoltro a tutti i router, permettendo di fatto la separazione tra il piano dei dati e quello di controllo.

Il controller potrebbe essere implementato in un data center. L'architettura **SND Software-Defined-Network** permette questo approccio. L'infrastruttura di rete fisica resta astratta e direttamente programmabile tramite un'interfaccia di controllo software.

Router.

Il **router** è un dispositivo a livello di rete quindi deputato alla commutazione di livello 3 nello stack OSI.

Ci sono quattro componenti fisici principali:

- **Porte di ingresso:** svolgono le funzioni di **terminazione (elettrica)** di un collegamento in ingresso al router e di **elaborazione a livello di collegamento**. Queste implementano rispettivamente il livello fisico e di collegamento associati a un singolo collegamento in ingresso al router. Durante l'elaborazione alla porta di ingresso, utilizzando le informazioni della tabella di inoltro, viene determinata la porta di uscita a cui dirigere un pacchetto attraverso la struttura di commutazione.

La tabella di inoltro viene elaborata e aggiornata dal processore. Una sua copia conforme è memorizzata su ogni porta di ingresso.

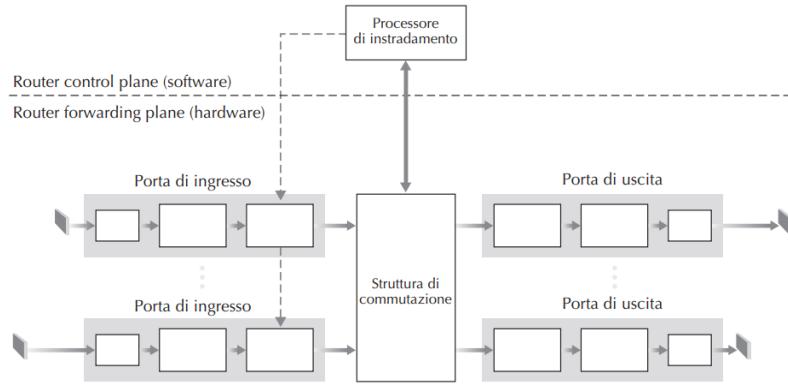


Figura 4.4 Architettura di un router.

- **Struttura di commutazione** (switching fabric): **connette fisicamente le porte di ingresso a quelle di uscita**. Lo switching può essere effettuato in memoria, via bus o attraverso una rete di interconnessione.

- **Porte di uscita:** che **memorizzano i pacchetti** che provengono dalla struttura di commutazione e li **trasmettono sul collegamento in uscita**, operando le funzionalità necessarie del livello di collegamento e fisico.

- **Processore di instradamento** (routing processor): **esegue le funzioni del piano di controllo, cioè di routing**.

Nei router tradizionali esegue i protocolli di instradamento, gestisce le tabelle di inoltro e le informazioni sui collegamenti attivi, ed elabora la tabella di inoltro per il router. Nei router SDN, il processore è responsabile della comunicazione con il controller remoto.

Le funzioni del piano dei dati (quindi routing), quindi svolte da porte di ingresso, porte di uscita e struttura di commutazione sono implementate quasi sempre in hardware perché operano su scala temporale dei nanosecondi. Le funzioni del piano di controllo (forwarding) sono invece implementate via software ed eseguite dal processore di instradamento, perché operano sulla scala dei millisecondi o secondi.

Protocollo Internet IP.

È un protocollo di interconnessione classificato al livello di rete (3) del modello ISO/OSI.

È un protocollo a pacchetti **senza connessione** e di tipo **best effort**, cioè che non garantisce alcuna forma di affidabilità della comunicazione in termini di controllo di errore, controllo di flusso e di congestione, che viene realizzata allivello superiore (TCP).

Attualmente sono in uso due versioni del protocollo, **IPv4** e **IPv6**.

Il pacchetto di rete è noto come **datagramma**. È costituito da un header e da un'area dati.

I principali campi dei datagrammi **IPv4** sono:

- **Numero di versione** [4 bit]: specifica la versione del protocollo.

- **Lunghezza dell'header** (IHL) [4 bit]: indica la lunghezza dell'header del pacchetto IP, ovvero l'offset del campo dati. È necessario perché un datagramma può contenere un numero variabile di opzioni (incluse nell'intestazione);

- **Tipo di servizio** [8 bit]: servono all'host mittente per specificare il modo in cui il destinatario deve trattare il datagramma (precedenza, latenza, throughput, affidabilità).
- **Lunghezza del datagramma** [16 bit]: rappresenta la dimensione in byte del datagramma IP (header + dati). Considerato che questo campo è di 16 bit, la massima dimensione dei datagrammi è di 65535 byte, anche se raramente superano i 1500 byte in modo da non superare la lunghezza massima del campo dei dati dei frame Ethernet.
- **Identificatore** [16 bit]: questo campo è un identificativo del datagramma
- **Flag** [3 bit]: il bit DF (Don't Fragment) se settato ad 1 indica che il pacchetto non può essere frammentato; MF (More Fragment) se settato a 0 indica che il pacchetto è l'ultimo frammento;
- **Fragment offset** [13 bit]: indica l'offset di un particolare frammento di un pacchetto IP;
- **Time to live (TTL)** [8 bit]: un contatore di 8 bit incluso per assicurare che i datagrammi non restino in loop per sempre nella rete. Questo campo viene decrementato (o dovrebbe) di un'unità ogni volta che il datagramma è elaborato da un router. Quando raggiunge 0, il datagramma deve essere scartato.
- **Protocollo** [8 bit]: indica il codice associato al protocollo utilizzato nel campo dati del pacchetto IP.
Ad esempio, il numero 6 indica che i dati sono destinati al protocollo TCP mentre il 17 designa UDP.
- **Checksum dell'intestazione** [16 bit]: consente ai router di rilevare gli errori dell'header. Ad ogni hop il checksum viene ricalcolato e confrontato con il valore di questo campo; se non c'è corrispondenza il datagramma viene scartato. Non viene effettuato nessun controllo sulla presenza di errori del campo Dati, che viene deputato ai livelli superiori.
- **Source/Destination address** [32 bit]: indirizzi IP che identificano il nodo mittente e quello ricevente.

- **Opzioni:** (multipli di 4 byte) campo che consente di estendere l'intestazione IP con un numero variabile di opzioni che comprendono la security, source routing, router recording, stream identification, timestamping. Se l'opzione non occupa 4 byte, vengono inseriti dei bit di riempimento (zero).
- **Dati** (payload): contiene il segmento a livello di trasporto (TCP o UDP) da consegnare alla destinazione e in alcuni casi anche altri tipi di dati, quali messaggi ICMP.

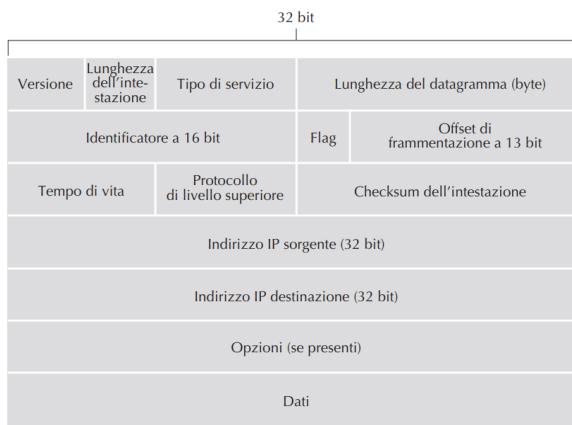


Figura 4.16 Formato dei datagrammi IPv4.

Frammentazione dei datagrammi IPv4.

La massima quantità di dati che un frame di livello collegamento può trasportare è detta **maximum transmission unit (MTU)**. Non tutti i protocolli a livello di collegamento possono trasportare pacchetti della stessa dimensione a livello di rete (per esempio i frame Ethernet possono trasportare fino a 1500 byte) e sicuramente tra il mittente e il destinatario di un pacchetto verranno utilizzati differenti protocolli di collegamento con differenti MTU.

Il **protocollo di rete**, e in particolare IP deve **frammentare i datagrammi in frammenti adatti all'MTU** del protocollo di collegamento ricevente e trasferirli in uscita.

I frammenti dovranno essere riassemblati prima di raggiungere il livello di trasporto alla destinazione.

Tenendo fede al principio di mantenere semplice il nucleo della rete si prevede di operare il riassemblaggio dei datagrammi sui sistemi periferici, anziché nei router interni.

Quando un host riceve una serie di datagrammi alla stessa origine, deve individuare i frammenti, determinare quando ha ricevuto l'ultimo e stabilire l'ordine di riassemblaggio. I campi di identificazione, flag e offset contenuti nei datagrammi IP assolvono a questo compito.

Indirizzamento IPv4.

Generalmente un **host ha un solo collegamento con la rete** e lo utilizza per **inviare e ricevere datagrammi**.

Il confine tra host e collegamento è detto **interfaccia**. Un router invece ha il compito di ricevere datagrammi da un collegamento ed inviarli su un altro, dunque presenta più interfacce, almeno 2, una su ciascuno dei suoi collegamenti.

Il protocollo IP è assegnato a un indirizzo IP ad ognuna di queste interfacce, e non al singolo host.

L'**indirizzo IPv4** è formato da 32 bit (4 byte) per tanto esistono circa 4 miliardi di indirizzi possibili. Tali indirizzi sono scritti in dotted-decimal notation cioè ciascun byte è indicato in forma decimale ed è separato con un punto dagli altri (es. 192.128.1.1).

Nell'indirizzo IP 193.32.216.9 il numero 193 è l'equivalente decimale dei primi 8 bit dell'indirizzo; 32 è l'equivalente dei secondi 8, e così via.

Pertanto, l'indirizzo 193.32.216.9 in notazione binaria diventa 11000001 00100000 11011000 00001001

Gli indirizzi IP pubblici sono univoci e assegnati da ICANN Internet Corporation for Assigned Names and Numbers.

Concettualmente l'**indirizzo IP** si compone di **due parti**:

- **Identificatore di rete e precisamente della sottorete (Net_ID)**
- **Identificatore di host (Host_ID)**

Per determinare le sottoreti di una rete complessa si sganciano le interfacce da host e router in maniera tale da creare isole di reti isolate delimitate dalle interfacce.

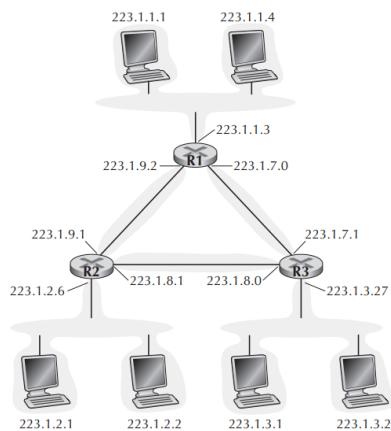


Figura 4.20 Tre router che interconnettono sei sottoreti.

CLASSFULL ADDRESSING.

La strategia di assegnazione degli indirizzi nota come **classfull addressing** prevede che la **suddivisione dei bit tra le due componenti di un indirizzo è effettuata in base alla classe di appartenenza**.

Esistono dunque **tre classi**, i cui indirizzi hanno rispettivamente 8, 16, 26 bit assegnati alla parte della rete.

La **classe di appartenenza** è determinata dalla **subnet mask**, che se in notazione /x determina il numero di bit che ricadono nel prefisso e che individuano la sottorete. Il restante numero (32-x) costituisce il suffisso e identifica l'host specifico all'interno della rete.

La **subnet mask** però può essere scritta anche in dot notation su 4 byte (X.X.X.X). Se un byte è 1 la porzione corrispondente dell'indirizzo fa parte del prefisso, se è 0 al contrario fa parte del suffisso.

Il **network address** si ottiene poi mediante un AND bit a bit tra l'indirizzo IP e la subnet mask.

Indirizzo IP	192.168.1.1	
Subnet mask	255.255.255.0	
Subnet Address	192.168.1.0	
Indirizzo IP	192.168.5.130	(130 -> 10000010)
Subnet Mask	255.255.255.192	(192-> 11000000)
Subnet Address	192.168.5.128	(10000000 = 128)

Utilizzo Subnet.

Al momento dell'invio di un pacchetto, l'host confronta l'indirizzo IP della destinazione con il proprio. Se la destinazione è sulla sua stessa sottorete invia i pacchetti sulla LAN, altrimenti li invia ad un Gateway, che sarà connesso alle altre reti e si occuperà di inoltrare i pacchetti ricevuti.

A seconda della subnet mask utilizzata l'indirizzo ricade in una delle seguenti classi:

CLASSE	UTILIZZO BIT	SUBNET MASK	RETI	HOST
A	0NNNNNNN . HHHHHHHH . HHHHHHHH . HHHHHHHH	255.0.0.0 / 8	128	16M
B	10NNNNNN . NNNNNNNN . HHHHHHHH . HHHHHHHH	255.255.0.0 / 16	16K	65K
C	110NNNNN . NNNNNNNN . NNNNNNNN . HHHHHHHH	255.255.255.0 / 24	2M	256
D	1110XXXX . XXXXXXXX . XXXXXXXX . XXXXXXXX	MULTICAST		

Classe A:

- Il primo byte rappresenta la rete (va da 0-128), gli altri tre gli host per ogni rete.
- La maschera di rete è 255.0.0.0 oppure /8 perché i primi 8 bit sono dedicati alla rete.
- Questi indirizzi iniziano con i bit 0

Classe B:

- I primi due byte rappresentano la rete (da 128-191), gli altri due gli host per ogni rete
- La maschera di rete è 255.255.0.0 oppure /16
- Gli indirizzi iniziano con i bit 10

Classe C:

- I primi tre byte rappresentano la rete (192- 223), l'ultimo gli host per ogni rete
- La maschera di rete è 255.255.255.0 oppure /24)
- Gli indirizzi iniziano con i bit 110

Classe D:

- È riservata agli indirizzi Multicast.
- Non è definita una maschera di rete, essendo tutti i 32 bit utilizzati per indicare un gruppo, non un singolo host.
- Questi indirizzi in binario iniziano con i bit 1110.

I seguenti sono indirizzi speciali riservati ad usi specifici.

Network address: indirizzi di qualsiasi classe, il cui suffisso è costituito da tutti '0' e dunque non sono assegnati a nessun host ma identificano una rete specifica. ES. x.0.0.0/8

Direct Broadcast Address: indirizzi che consentono l'invio di un messaggio a tutti gli host sulla stessa sottorete. Il suffisso dell'indirizzo è costituito da tutti '1'. ES x.x.1.1/16

Limited Broadcast Address: 1.1.1.1 (255.255.255.255) viene utilizzato da un host per inviare un messaggio a tutti gli host della stessa rete (il router blocca questi messaggi verso reti esterne).

This Computer Address: 0.0.0.0 utilizzato da un host durante la procedura di avvio del suo stack IP in quanto non è ancora a conoscenza del proprio indirizzo IP; il destinatario di tale pacchetto è un server di bootstrap presente sulla rete

Zero Address: è un indirizzo che contiene tutti 0 nel prefisso e indica un particolare host nella rete locale con i bit del suffisso. ES 0.0.0.46/24

Loopback Address (o Localhost): è un indirizzo ip con il primo byte pari a 127 utilizzati per il test dello stack IP. Individuano la stessa macchina e non abbandonano mai l'host che li ha generati ES 127.0.0.1 L'indirizzamento a classi presenta diversi limiti dovuti soprattutto al numero di host gestibili dalle diverse classi. In pratica se si esauriscono gli indirizzi univoci resi disponibili da una classe, occorre fare ricorso ad un indirizzo di classe superiore.

CLASSLESS INTERDOMAIN ROUTING. (CIDR)

A partire dal 1993 si abbandonò il concetto di classful routing in favore di una nuova strategia detta *classless interdomain routing* (**CIDR**) che generalizza la nozione di indirizzamento di sottorete.

L'indirizzo IP è ancora diviso in prefisso e suffisso e mantiene la forma decimale puntata a.b.c.d/x. La novità essenziale consiste nell'utilizzare subnet mask di lunghezza arbitraria x, quando l'indirizzamento classfull ammette solo tre lunghezze (/8, /16, /24).

All'interno di una tradizionale Classe C (/24) sono possibili le seguenti sottoreti:

Con il vecchio approccio classfull in una sottorete di classe C, 24 bit individuano la singola sottorete, e i restanti 8 bit sono per identificare 256 host. Per cui se sono necessari ad esempio 300 indirizzi una rete di classe.

C'è insufficiente ed una rete di classe B è sovrabbondante.

Con l'approccio classless possono essere forniti blocchi di indirizzi con 31 netmask diverse. Ciò rende più flessibile la suddivisione di reti in più sottoreti mantenendo un adeguato numero di host in ogni sottorete.

Esempio:

Se un blocco di 256 indirizzi (classe C) è possibile usare una netmask /25 (cioè 255.255.255.128) per ottenere 2 subnet da 128 indirizzi. È possibile suddividere la seconda in ulteriori 2 subnet da 64 indirizzi con una mask 255.255.255.192 (/26).

INDIRIZZI PRIVATI.

Tra gli indirizzi speciali riservati, alcuni blocchi sono riservati per uso privato.

- 10.0.0.0 – 10.255.255.255 CLASSE A
- 172.16.0.0 – 172.31.255.255 CLASSE B
- 192.168.0.0 – 192.168.255.255 CLASSE C

Sono indirizzi riservati alle reti locali. I pacchetti con tali indirizzi non vengono utilizzati per l'indirizzamento e l'instradamento tramite protocollo IP dai router Internet verso la rete di trasporto. Risolvono dunque il problema della limitatezza degli indirizzi IP pubblici perché possono essere replicati su reti differenti senza avere conflitti. Nel caso occorra connettere ad Internet una rete locale che utilizza indirizzi privati si deve ricorrere al NAT il quale multiplica (o mappa) più indirizzi IP privati su un solo indirizzo IP pubblico, visibile all'esterno della sottorete ed utilizzabile per l'instradamento.

NAT – Network Address Translation

Il **NAT** è una tecnica che funziona al livello di rete e in particolare è implementato nei router e nei firewall interposti tra le reti private e Internet.

All'interno di una rete privata, sono utilizzati solitamente indirizzi IP privati, che non permettono il routing verso l'esterno. Quando il router che implementa il NAT riceve un pacchetto dalla rete privata, scambia l'indirizzo IP privato (e la porta) con uno pubblico valido per il routing assegnato al router stesso. In pratica un insieme di macchine interne alla rete è vista dall'esterno con un unico ip pubblico, che è quello del router.

Il NAT mantiene una tabella di routing per gestire le connessioni e le diverse comunicazioni fra host. Quando riceve un pacchetto dall'esterno, verifica con la tabella di routing se c'è un host interno che aspetta un pacchetto e lo inoltra.

Il NAT può essere di tipo:

- **Statico**: il NAT associa un indirizzo IP pubblico ad un indirizzo IP privato interno alla rete. Permette di connettere terminali della rete interna ad internet in maniera trasparente ma non risolve il problema della scarsità di indirizzi.

• **Dinamico** (Network Address and Port Translation): il NAT permette di condividere un indirizzo IP routabile fra più terminali in indirizzamento privato. Tutti i terminali della rete interna hanno virtualmente, lo stesso indirizzo IP, se visti dall'esterno. Questa meccanismo può essere infatti chiamato anche IP masquerading (maschera IP). Per multiplexare i diversi indirizzi IP su un indirizzo pubblico il NAT dinamico usa il meccanismo di traslazione della porta (PAT Port Address Translation), cioè l'attribuzione di una porta sorgente diversa ad ogni nuova richiesta in maniera tale da poter mantenere una corrispondenza tra le richieste che provengono dalla rete interna e le risposte dei terminali su Internet, tutte indirizzata all'indirizzo IP del router.

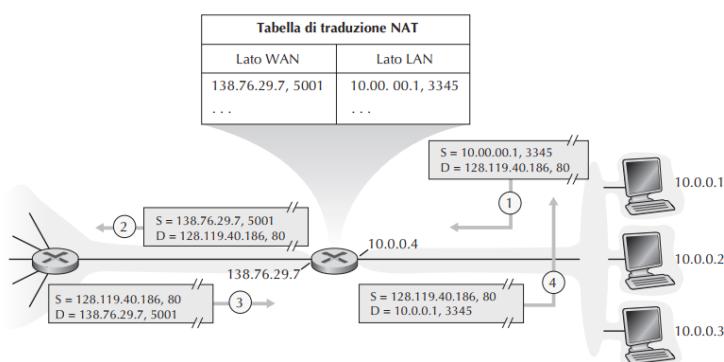


Figura 4.25 NAT.

PORT FORWARDING.

La traslazione di indirizzo, tramite NAT, permette di collegare solo delle richieste che provengono dalla rete interna verso quella esterna, il che significa che è impossibile per un terminale esterno inviare un pacchetto verso un terminale della rete interna.

Esiste un'estensione del NAT detta port forwarding che consiste nel configurare in gateway per trasmettere ad un terminale specifico della rete interna, tutti i pacchetti ricevuti su una particolare porta. Così, ad esempio, se si vuole accedere ad un server web (porta 80) funzionante su un terminale 192.168.1.2 (privato) dall'esterno, sarà necessario definire una regola di forwarding della porta sul gateway, ridirigendo tutti i pacchetti TCP ricevuti sulla porta 80 verso il terminale 192.168.1.2.

DHCP (DYNAMIC HOST CONFIGURATION PROTOCOL).

DHCP è un **protocollo di rete di livello applicativo** che permette agli host di una rete locale di ricevere automaticamente ad ogni richiesta di accesso ad una rete, la configurazione IP necessaria per stabilire una connessione e inter operare con tutte le altre sottoreti.

Oltre che all'indirizzo IP da assegnare all'host, DHCP fornisce dinamicamente, al client richiedete:

- Maschera di sottorete
- Gateway di default
- Indirizzi del server DNS
- Indirizzi del server WINS
- Indirizzi del server NTP
- Altro

DHCP è implementato come protocollo **Client-Server**. Un **client** DHCP di solito è un host che, connesso alla rete, richiede informazioni sulla configurazione di rete. Di solito ogni sottorete dispone di un **server** DHCP che assegna gli indirizzi, ma in caso contrario è necessario un agente di Relay DHCP, che si occupa di inoltrare le richieste ad un server.

Sessione tipica DHCP:

1. Individuazione del server DHCP: il client collegato sulla rete invia un pacchetto UDP chiamato DHCPDISCOVER in broadcast sulla porta 67, con indirizzo IP sorgente 0.0.0.0 (this computer address) e destinazione 255.255.255.255 (indirizzo di broadcast);
2. Offerta del server DHCP: il pacchetto è ricevuto da tutto il dominio di broadcast e in particolare da tutti i server DHCP presenti, i quali possono rispondere (o meno) ciascuno con un pacchetto di DHCPOFFER in cui propongono un indirizzo IP e gli altri parametri di configurazione al client. Questo pacchetto di ritorno è indirizzato in broadcast all'indirizzo di livello datalink del client (che non ha ancora un indirizzo IP) cioè in unicast, per cui può essere inviato solo da un server che si trovi sullo stesso dominio di broadcast. Il pacchetto contiene oltre che a tutte le configurazioni di rete che DHCP offre, anche un lease time, cioè il lasso di tempo durante il quale l'indirizzo IP è valido.

3. Richiesta DHCP: Il client sceglie tra le offerte dei server dopodiché invia un pacchetto di DHCPREQUEST in broadcast, indicando all'interno del pacchetto quale server ha selezionato. Anche questo pacchetto raggiunge tutti i server DHCP presenti sulla rete.

4. Conferma DHCP: Il server che è stato selezionato conferma l'assegnazione dell'indirizzo con un pacchetto di DHCPACK (nuovamente indirizzato in broadcast all'indirizzo di livello datalink del client); gli altri server vengono automaticamente informati che la loro offerta non è stata scelta dal client, e che sulla sottorete è presente un altro server DHCP. Sicurezza DHCP

Il client si identifica verso il server attraverso un campo client-id dei pacchetti DHCP. Questo campo normalmente ha come valore il mac address della scheda di rete per cui si richiede l'indirizzo. Questa è l'unica forma di autenticazione disponibile ed è piuttosto debole, in quanto utilizza un dato che viene inviato in broadcast sulla rete locale, e quindi può essere facilmente trovato da qualunque host connesso alla stessa rete. Un host malevolo, connesso alla rete, e configurato come server DHCP potrebbe rispondere ad una richiesta DHCP e fornire come gateway di default il proprio mac address. A questo punto l'host malevolo potrà sniffare tutto il traffico generato dal client, e tramite IP masquerading può ridirigere le connessioni verso il gateway ufficiale (MITM). Per prevenire questi attacchi, alcuni switch offrono la funzionalità DHCP snooping che permette di fermare i pacchetti che non sono originati da server autorizzati.

PROTOCOLLO ICMP.

Il protocollo ICMP (*Internet Control Message Protocol*) viene usato da host e router per scambiarsi informazioni a livello di rete, tipicamente riguardanti errori, malfunzionamenti o informazioni di controllo.

ICMP è encapsulato direttamente in IP e non è quindi garantita la consegna a destinazione di tali pacchetti (best effort). ICMP può essere usato per veicolare diversi tipi di messaggi di gestione, identificati dal tipo e dal relativo codice.

PROTOCOLLO IPv6.

Nei primi anni '90 l'IETF diede inizio alla versione successiva del protocollo IPv4. La prima problematica affrontata fu quella dello spazio di indirizzamento. I 4 miliardi di indirizzi possibili con IPv4 stavano per terminare.

Inoltre, sulla base dell'esperienza accumulata, i progettisti colsero l'occasione per apportare altre migliorie e risolvere alcune problematiche:

- Spazio di indirizzamento;
- Routing;
- Sicurezza; (confidenzialità, integrità, autenticazione)
- Configurazione automatica;
- Servizi di tipo real-time. (IPv6 cerca di garantire un servizio efficiente a differenza di IPv4)

L'indirizzo di rete è lungo 268 bit, cioè 32 cifre esadecimali. Questo porta il numero di indirizzi esprimibili dall'IPv6 a $2^{128} = 1632 = 3,4 \times 10^{38}$. Viene definito un nuovo tipo di indirizzo, anycast, che si riferisce ad un insieme di interfacce.

Un pacchetto inviato ad un indirizzo anycast viene recapitato ad una delle interfacce che fanno parte dell'insieme da esso individuato, tipicamente quella più vicina secondo la metrica utilizzata dal protocollo di routing.

L'header è stato semplificato. Alcuni campo sono stati eliminati o resi opzionali. Ciò ha consentito che, malgrado gli indirizzi IPv6 siano 4 volte più lunghi di quelli di IPv4, l'header del primo è soltanto il doppio di quello del secondo.

Viene introdotto il supporto per la Quality of Service, cioè una funzionalità che permette di etichettare (flow label) i pacchetti appartenenti a flussi di dati particolari per i quali si richiede un trattamento di tipo non default.

L'header consiste di due parti: *header principale* ed *extension headers*. Gli extension headers sono introdotti per ospitare opzioni aggiuntive, tra le quali opzioni per la gestione del routing, della frammentazione, dell'autenticazione e della sicurezza.

Campi datagramma IPv6.

- **Versione**: Campo a 4 bit che identifica il numero di versione IP
- **Campo di traffico**: Campo a 8 bit, simile al campo TOS di IPv4. Può essere utilizzato per attribuire priorità a determinati datagrammi all'interno di un flusso o provenienti da specifiche applicazioni (per esempio, voice-over-IP) rispetto a quelli di altri servizi (per esempio SMTP).
- **Etichetta di flusso**: campo a 20 bit utilizzato per identificare un flusso di datagrammi.
- **Lunghezza del payload**: Questo valore a 16 bit è trattato come un intero senza segno e indica il numero di byte nel datagramma IPv6 che seguono l'intestazione a lunghezza fissa di 40 byte.

- **Intestazione successiva:** Campo che identifica il protocollo a cui verranno consegnati i contenuti (campo dati) del datagramma, per esempio TCP o UDP. Utilizza gli stessi valori del campo protocollo nell'intestazione IPv4.
- **Limite di hop:** contatore del numero di hops (analogo a TTL)

- **Limite di hop:** Il contenuto di questo campo è decrementato di 1 da ciascun router che inoltra il datagramma. Quando il suo valore raggiunge 0, il datagramma viene eliminato.

- **Indirizzo di sorgente:** indirizzo del mittente

- **Indirizzo di destinazione:** indirizzo del destinatario. Differenze con l'header IPv4

Confrontando il formato IPv6 con IPv4 notiamo che sono stati eliminati vari campi:

- Frammentazione/riassembaggio. IPv6 non consente frammentazione né riassembaggio sui router intermedi; queste operazioni possono essere effettuate soltanto da sorgente o destinazione.

- Checksum dell'intestazione. Dal momento che i protocolli Internet a livello di trasporto (per esempio, TCP e UDP) e di collegamento (per esempio, Ethernet) calcolano un loro checksum, i progettisti di IP hanno probabilmente ritenuto questa funzionalità talmente ridondante nel livello di rete da decidere di rimuoverla.

- Opzioni. Il campo Opzioni non fa più parte dell'intestazione IP standard, anche se non è del tutto scomparso.

Le priorità dei pacchetti possono essere: il meccanismo degli header, incapsulati nel payload.

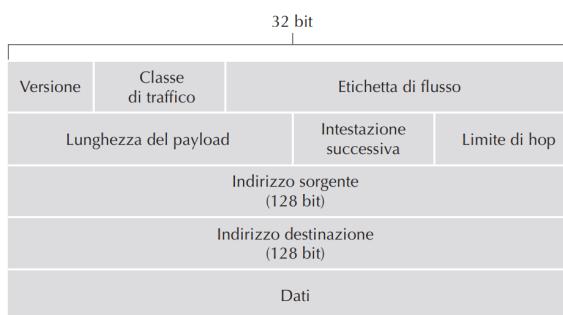


Figura 4.26 Formato dei datagrammi IPv6.

Frammentazione.

Un'altra differenza sostanziale con IPv4 è la modalità di frammentazione dei pacchetti.

In IPv4 la frammentazione avviene all'occorrenza sui router intermedi, quando il pacchetto è più grande del MTU del router.

In IPv6 la frammentazione viene effettuata invece in modalità end-to-end.

Se un IPv6 datagram ha dimensioni maggiori della MTU, questo viene scomposto (dalla sorgente) in datagrammi più piccoli, detti **frammenti**, ognuno dei quali contiene una parte del payload relativo all'IP datagram originale. Il livello Network prima di inviare il datagramma richiede all'interfaccia di rete di destinazione la sua MTU per poter eventualmente effettuare la frammentazione. L'header non è frammentabile ed è inviato con ciascun frammento.

Questa frammentazione avviene con l'ausilio di un particolare extension header, detto Fragment Header, identificato dal valore 44 nel campo Next Header. L'header in questione contiene i campi relativi alla frammentazione presenti anche nell'header dell'IPv4 (fragment offset, M, identification).

In questo modo si riduce l'overhead dei router, in modo che essi possano gestire più pacchetti per unità di tempo.

Indirizzi IPv6.

Gli indirizzi sono rappresentati come 8 gruppi di 4 cifre esadecimali separati dal carattere :

Se uno dei gruppi è composto da una sequenza di quattro zeri può essere contratto ad un solo zero;

Una sequenza di zeri contigui (e una soltanto) composta da 2 o più gruppi può essere contratta con ::

Di seguito sono riportate varie rappresentazioni dello stesso indirizzo:

```
2001:0db8:0000:0000:0000:1428:57ab
2001:0db8:0000:0000::1428:57ab
2001:0db8:0:0:0:1428:57ab
2001:0db8:0::0:1428:57ab
2001:0db8::1428:57ab
```

Inoltre, possono essere omessi gli zeri iniziali di ogni gruppo: 2001:db8:2de::e13

Gli ultimi 32 bit possono essere scritti in decimale, nella notazione dotted decimal, rendendo così la sintassi IPv6 retro compatibile con quella IPv4. Esistono due rappresentazioni per gli IPv4 utilizzando indirizzi IPv6.

IPv4-mapped address. I primi 80 bit sono posti a 0, i successivi 16 sono posti a 1 (ffff) e gli ultimi 32 rappresentano l'IPv4.

```
0:0:0:0:ffff:192.168.0.1
::ffff:192.168.0.1
```

IPv4-compatible address. I primi 96 bit sono posti a 0 e gli ultimi 32 rappresentano l'indirizzo IPv4 (deprecati in favore di mapped)

0:0:0:0:0:0:192.168.0.1

::192.168.0.1

Transizione IPv4/IPv6.

Il passaggio nella rete Internet del protocollo di livello rete IP dalla versione 4 alla versione 6 ha portato ingenti problemi di compatibilità. La politica naturalmente adoperabile consisterebbe nel costruire router e switch in grado di interpretare entrambi i protocolli. Questo risulta di difficile applicazione per la vastità dei router da sostituire nel mondo. Si adottano dunque soluzioni software quali dual stack o tunneling.

La tecnica del dual stack prevede l'utilizzo del doppio stack IP, nella pila protocollare. Questo è di semplice implementazione ma aumenta la complessità della rete, nel senso che i router e gli switch devono interpretare più istanze dello stesso protocollo.

Inoltre, non risolve il problema dell'indirizzamento, perché ogni interfaccia deve comunque essere dotata di un indirizzo IPv4 e uno IPv6.

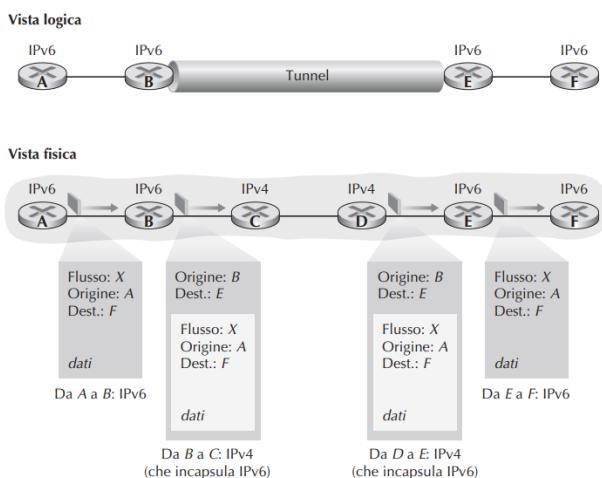


Figura 4.27 Tunneling.

Indirizzi Speciali.

È stato definito un certo numero di indirizzi speciali riservati ad usi particolari. Nella notazione CIDR:

- ::/128 – indirizzo composto da tutti zeri utilizzato per inizializzare l'host prima che esso conosca il proprio indirizzo.
- ::1/128 – indirizzo di loopback utilizzato per identificare la macchina locale (localhost) su cui i programmi sono in esecuzione
- ::/96 – utilizzato per interconnettere le due tecnologie IPv4/IPv6 nelle reti ibride (deprecati)
- ::ffff:0:0/96 – indirizzo IPv4 mapped address utilizzato nei dispositivi dual stack
- fe80:/10 – il prefisso link-local specifica che l'indirizzo è valido solo sullo specifico link fisico
- ff00::/8 – indirizzo multicast

Livello di rete: piano di controllo.

Trattiamo il **piano di controllo** (control plane), la logica di rete globale che controlla non solo come i datagrammi vengono inoltrati tra i router lungo i percorsi end-to-end dall'host sorgente all'host destinatario, ma anche come le componenti e i servizi del livello di rete vengono configurati e gestiti. Verranno trattati gli algoritmi di instradamento tradizionali per calcolare i percorsi a costo minimo in un grafo; tali algoritmi sono alla base dei due protocolli di instradamento di Internet: OSPF e BGP.

OSPF è il protocollo di instradamento che opera all'interno della rete di un singolo ISP.

BGP è il protocollo di instradamento che serve a interconnettere tutte le reti in Internet; BGP è un collage che tiene insieme Internet.

Le tabelle di inoltro, nel caso di inoltro basato sulla destinazione, e le tabelle dei flussi, nel caso di inoltro generalizzato, sono gli elementi principali che collegano il piano dei dati e il piano di controllo del livello di rete. Queste tabelle specificano il comportamento di inoltro nel piano dei dati locale di un router.

Nel definire la tecnica con la quale le tabelle di inoltro e dei flussi vengono calcolate possono essere adottati due approcci:

- **Controllo locale:** L'algoritmo di instradamento viene eseguito su ogni singolo router, all'interno del quale vengono effettuate sia le funzioni di inoltro (piano dei dati) che quelle di instradamento (piano di controllo). Questo approccio di controllo locale per router è stato usato in Internet per decenni; i protocolli OSPF e BGP sono basati su tale approccio.
- **Controllo logicamente centralizzato:** il controller centralizzato calcola e distribuisce le tabelle di inoltro che devono essere utilizzate da ogni router. Il controller interagisce i control agent di ogni router, i quali non partecipano attivamente all'elaborazione della tabella di inoltro. Le architetture SDN utilizzano tale approccio.

ALGORITMI DI INSTRADAMENTO.

Gli **algoritmi di instradamento** hanno lo scopo di **determinare** i **percorsi**, o **cammini**, tra le sorgenti e le destinazioni attraverso la rete di router. Tipicamente il percorso migliore è quello che ha un costo minimo, ma ci sono anche problematiche di interesse concreto, quali possono essere le policy adottate dalle parti interagenti (es. un utente della rete potrebbe non voler far passare i pacchetti per determinati router.).

- Per formulare i problemi di instradamento, le reti sono modellate attraverso dei **grafo**: un grafo $G = (N, E)$ è un insieme N di nodi e un insieme E di archi (edge), ove ciascun arco collega una coppia di nodi di N .

- I nodi rappresentano i router della rete;
- Gli archi rappresentano i link fisici che connettono i vari router della rete.

Ad un arco è associato un valore che ne indica il costo.

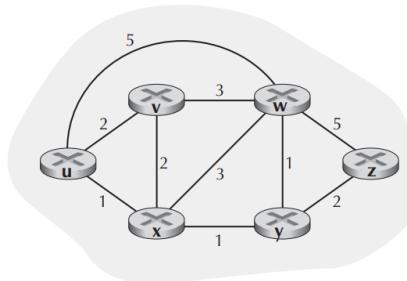
Se gli archi non sono orientati (sono bidirezionali), si presume un costo uguale per entrambi i versi di percorrenza.

In generale un link può non essere simmetrico, e presentare un costo diverso per ognuno dei versi.

Dati due nodi qualsiasi, esistono più percorsi che li congiungono, ciascuno con il proprio costo (somma dei costi di tutti gli archi attraversati).

L'obiettivo di un algoritmo di instradamento è ***l'individuazione dei percorsi a costo minimo tra la sorgente e la destinazione***.

Si noti che se tutti gli archi del grafo hanno lo stesso costo, il percorso a costo minimo rappresenta anche il percorso più breve (shortest path), ossia il percorso con il minor numero di collegamenti tra sorgente e destinazione.



In generale si distinguono, in base al tipo di controllo adottato, due macro classi di algoritmi:

• **Algoritmi centralizzati**: sono algoritmi globali che ricevono in ingresso tutti i collegamenti tra i nodi e i relativi costi. Quindi hanno una conoscenza globale e completa della rete (anche connettività e costi) che gli permette determinare il percorso a costo minimo tra una sorgente e una destinazione. Gli algoritmi con informazioni di stato globali sono spesso detti algoritmi link-state (LS).

• **Algoritmi decentralizzati**: il percorso a costo minimo è calcolato in modo distribuito e iterativo.

Nessun nodo possiede informazioni complete sul costo di tutti i collegamenti di rete. Inizialmente i nodi conoscono solo i costi dei collegamenti a loro incidenti. Attraverso un processo iterativo, e tramite lo scambio di informazioni con i nodi adiacenti un nodo gradualmente calcola il percorso a costo minimo. Si dice che un singolo nodo deve prendere una decisione a livello globale sulla base di informazioni locali.

Un secondo criterio per classificare gli algoritmi di instradamento li suddivide in:

• **Algoritmi statici** (non adattivi): non basano le loro decisione di routing su misurazioni o stime del traffico corrente e della tipologia. La scelta dei percorsi da usare è calcolata in anticipo e caricata nei router. I percorsi cambiano molto raramente, spesso come risultato di un intervento umano.

• **Algoritmi dinamici** (adattivi): i percorsi sono aggiornati automaticamente in funzione delle modifiche di topologia o di traffico.

Un algoritmo dinamico può essere eseguito sia periodicamente o come conseguenza diretta di un cambiamento nella topologia o nel costo di un collegamento. Gli algoritmi dinamici rispondono meglio ai cambiamenti della rete, ma sono anche maggiormente soggetti a problemi quali l'instradamento in loop e l'oscillazione dei percorsi.

Un terzo modo per classificare gli algoritmi di instradamento è il fatto di essere più o meno **sensibili** (load-sensitive o load-insensitive) al carico della rete. In un algoritmo sensibile al carico i costi dei collegamenti variano dinamicamente per riflettere il livello corrente di congestione.

Se si associa un alto costo a un collegamento attualmente trafficato, un algoritmo di instradamento tenderà a evitare di usarlo.

Gli attuali algoritmi di instradamento Internet (quali RIP, OSPF e BGP) sono algoritmi insensibili al carico, dato che il costo di un collegamento non riflette esplicitamente il suo attuale (o recente) livello di congestione.

Instradamento "link-state". (LS)

In un instradamento **link-state** la topologia di rete e tutti i costi dei collegamenti sono disponibili in input all'algoritmo.

Ciò si ottiene facendo inviare a ciascun nodo pacchetti sullo stato dei suoi collegamenti a tutti gli altri nodi della rete.

Così tutti i nodi dispongono di una vista identica e completa della rete e ciascun nodo che lancerà l'algoritmo LS otterrà gli stessi risultati.

L'algoritmo di calcolo dei percorsi che presentiamo associato all'instradamento link-state è noto come **algoritmo di Dijkstra**.

L'**algoritmo di Dijkstra** calcola il percorso a costo minimo da un nodo (l'origine, che chiameremo u) a tutti gli altri nodi nella rete, è iterativo e ha le seguenti proprietà: dopo la k -esima iterazione, i percorsi a costo minimo sono noti a k nodi di destinazione e, tra i percorsi a costo minimo verso tutti i nodi di destinazione, questi k percorsi hanno i k costi più bassi.

L'algoritmo di instradamento centralizzato consiste in un passo di inizializzazione seguito da un ciclo che viene eseguito una volta per ogni nodo del grafo. Quando termina, l'algoritmo avrà calcolato il percorso minimo dal nodo origine u a tutti gli altri nodi.

```

Algoritmo Dijkstra dal nodo origine u
1 Inizializzazione:
2    $N' = \{u\}$ 
3   per tutti i nodi v
4     se v è adiacente a u
5       allora  $D(v) = c(u,v)$ 
6       altrimenti  $D(v) =$ 
7
8 Ciclo
9   determina un w non in  $N'$  tale che  $D(w)$  sia minimo
10  aggiungi w a  $N'$ 
11  aggiorna  $D(v)$  per ciascun nodo v adiacente a w e non in  $N'$ :
12     $D(v) = \min((D(v), D(w) + c(w,v)))$ 
13  /* il nuovo costo verso v è il vecchio costo verso v oppure
14  il costo del percorso minimo noto verso w più il costo da w a v */
15  ripeti in ciclo finché non si verifica che  $N' = N$ 

```

Quando l'algoritmo Dijkstra termina, abbiamo **per ciascun nodo il suo predecessore** lungo il percorso a costo minimo dal nodo origine. Per ciascun predecessore abbiamo il rispettivo predecessore, e in questo modo riusciamo a costruire l'intero percorso dall'origine a tutte le destinazioni. La tabella di inoltro in un nodo può essere costruita da queste informazioni memorizzando, per ciascuna destinazione, il nodo del successivo hop sul percorso a costo minimo da u alla destinazione.

Qual è la complessità computazionale di questo algoritmo? Dati n nodi per determinare i percorsi a costo minimo dall'origine a tutte le destinazioni vengono fatte $n(n + 1)/2$ iterazioni e, di conseguenza, diciamo che la precedente implementazione dell'algoritmo Dijkstra ha, nel caso peggiore, una **complessità di ordine quadratico: $O(n^2)$** . Utilizzando una struttura dati nota come heap, si riesce a ridurre la complessità di questo algoritmo.

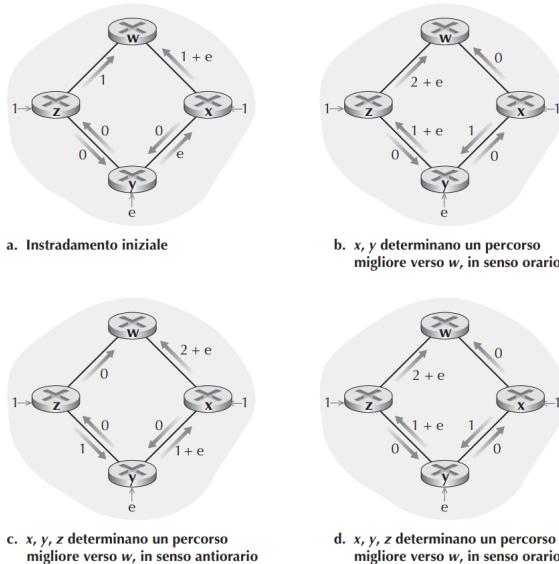


Figura 5.5 Oscillazioni con instradamento sensibile alla congestione.

Instradamento "distance-vector". (DV)

LS usa informazioni globali, **distance-vector** è **iterativo, asincrono e distribuito**.

- È **distribuita** nel senso che ciascun nodo riceve parte dell'informazione da uno o più dei suoi vicini direttamente connessi, a cui, dopo aver effettuato il calcolo, restituisce i risultati.
- È **iterativo** nel senso che questo processo si ripete fino a quando non avviene ulteriore scambio informativo tra vicini.
- L'algoritmo è anche **auto-terminante**: non vi è alcun segnale che il calcolo debba fermarsi, semplicemente si blocca.
- È **asincrono** nel senso che non richiede che tutti i nodi operino al passo con gli altri.

L'algoritmo di Bellman-Ford calcola i cammini minimi di un'unica sorgente su un grafo diretto pesato (dove alcuni pesi degli archi possono essere negativi). L'algoritmo di Dijkstra risolve lo stesso problema in un tempo computazionale inferiore, ma richiede che i pesi degli archi siano non-negativi. Per questo, Bellman-Ford è usato di solito quando sul grafo sono presenti pesi degli archi negativi.

L'algoritmo di Bellman-Ford è nella sua struttura base molto simile a quello di Dijkstra, ma invece di selezionare il nodo di peso minimo, tra quelli non ancora processati, con tecnica **greedy**, semplicemente processa tutti gli archi e lo fa $|V| - 1$ volte, dove $|V|$ è il numero di vertici nel grafo. Le ripetizioni permettono alle distanze minime di propagarsi accuratamente attraverso il grafo, poiché, in assenza di cicli negativi il cammino minimo può solo visitare ciascun nodo al più una volta. Diversamente da quello con tecnica greedy, il quale dipende da certe assunzioni strutturali derivate dai pesi positivi, questo semplice approccio si applica al caso più generale.

L'idea è quella di partire dal nodo sorgente e cominciare a guardare i nodi adiacenti, cioè che distano un passo solo. Si assegna loro il valore del costo per raggiungerli (determinato dal costo dell'arco + il valore del nodo da cui si è partiti, che in questo caso è 0 visto che stiamo partendo dalla sorgente). A questo punto per ciascuno dei nodi raggiunti si procede allo stesso modo: si vedono i nodi che gli distano uno e si assegna

loro il valore dell'arco percorso per raggiungerli più quello già assegnato al nodo da cui si è partiti (assegno loro il nuovo valore solo se è più piccolo di quello che già avevano: la prima volta che li raggiungi è certamente più piccolo in quanto abbiamo detto che inizialmente diamo a tutti i nodi valore infinito). Ad ogni passaggio i nodi raggiunti lo step precedente (considera nodi raggiunti solo quelli a cui hai aggiornato il valore) diventano punto di partenza per raggiungere i nodi adiacenti, il cui valore diventa (se è minore di quello che già possiedono) quello dell'arco percorso per raggiungerli + il valore del nodo da cui li si è raggiunti (e in tal caso diventano a loro volta nuovi punti di partenza). Se il grafo ha N nodi è certo che dopo N-1 giri tutti i nodi hanno a loro assegnato il costo minimo per essere raggiunti dal nodo sorgente. Ovviamente ad ogni giro quando aggiorni il valore di un nodo devi salvarti il percorso associato per raggiungerlo dalla sorgente, così quando avrai finito le iterazioni oltre ad avere tutti i costi minimi avrai anche i percorsi associati cioè i percorsi a costo minimo per raggiungere ogni nodo del grafo dal nodo sorgente.

L'algoritmo di Bellman-Ford ha una complessità temporale $O(|V| |E|)$, dove $|V|$ ed $|E|$ sono rispettivamente il numero di vertici e di archi del grafo.

Algoritmo Bellman Ford

```

A ciascun nodo x:
1 Inizializzazione:
2   per tutte le destinazioni y in N:
3      $D_x(y) = c(x,y)$  /* se y non è adiacente, allora  $c(x,y) = \infty$  */
4   per ciascun vicino w
5      $D_w(y) = ?$  per tutte le destinazioni y in N
6   per ciascun vicino w
7     invia il vettore delle distanze  $D_x = [D_x(y): y \in N]$  a w
8
9 ciclo
10  attendi (finché vedi cambiare il costo di un collegamento verso
11    qualche vicino w o finché ricevi un vettore delle distanze
12    da qualche vicino w)
13  per ogni y in N:
14     $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$ 
15
16  se  $D_x(y)$  è cambiato per qualche destinazione y
17    invia il vettore delle distanze  $D_x = [D_x(y): y \in N]$  a tutti i vicini
18
19 ripeti il ciclo indefinitamente

```

Ricordiamo che l'instradamento LS è centralizzato nel senso che richiede a ciascun nodo di ottenere innanzitutto una mappa completa della rete prima di mandare in esecuzione l'algoritmo di Dijkstra. L'algoritmo Bellman Ford è invece decentralizzato e non usa tale informazione globale. Infatti, le sole informazioni detenute dal nodo sono attende aggiornamenti dai suoi vicini (righe 10, 11 e 12), quando riceve un aggiornamento calcola il proprio nuovo vettore delle distanze (riga 14) e lo distribuisce ai suoi vicini (righe 16 e 17). L'instradamento DV viene utilizzato in molti protocolli reali, tra cui RIP e BGP per Internet, ISO IDRP, IPX di Novell e ARPAnet originale.

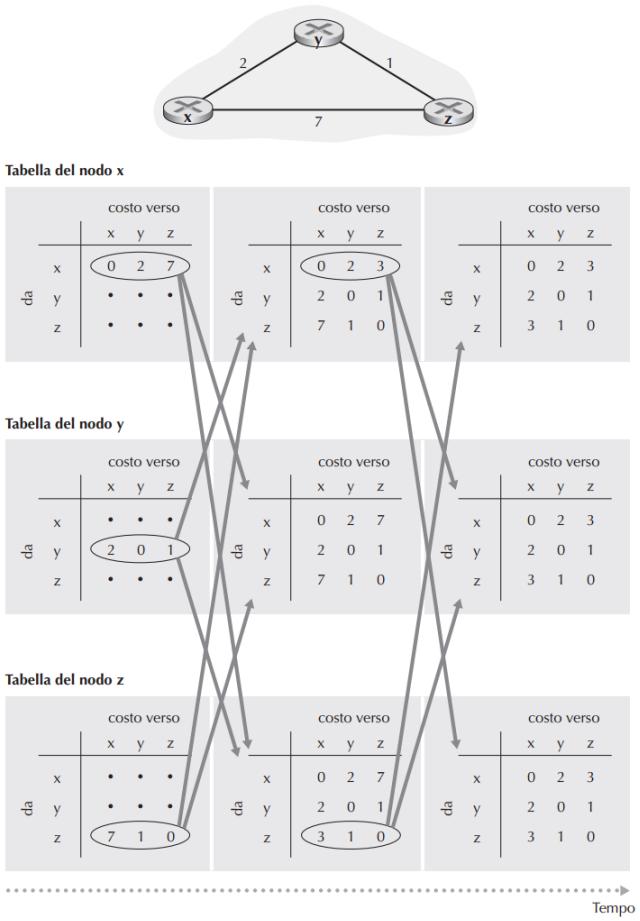


Figura 5.6 Instradamento distance vector (DV).

Instradamento distance-vector: modifica dei costi e guasti dei collegamenti.

Quando un nodo che esegue l'instradamento DV rileva un cambiamento nel costo dei collegamenti con un vicino (righe 10-11-12) aggiorna il proprio vettore delle distanze (righe 13-14) e, se si verifica un cambiamento nel costo del percorso a costo minimo, trasmette ai suoi vicini (righe 16-17) il proprio nuovo vettore delle distanze. Il processo che si viene a creare viene detto **problema di conteggio all'infinito**.

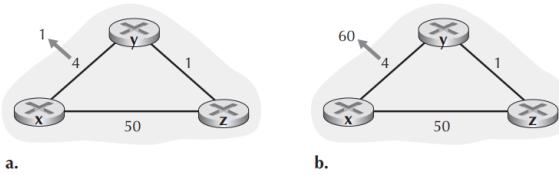


Figura 5.7 Variazioni nel costo dei collegamenti.

Confronto tra gli instradamenti LS e DV.

I due meccanismi di instradamento studiati utilizzano approcci complementari nel calcolare i percorsi. Nel DV ciascun nodo dialoga solo con i vicini direttamente connessi, informandoli delle stime a costo minimo da sé stesso a tutti i nodi (che conosce) nella rete.

Nel LS, ciascun nodo dialoga con tutti gli altri nodi via broadcast, ma comunica loro solo i costi dei collegamenti direttamente connessi. Concludiamo la nostra trattazione degli instradamenti LS e DV con un veloce confronto di alcuni dei rispettivi attributi. Ricordiamo che N rappresenta l'insieme dei nodi (router) ed E l'insieme degli archi (collegamenti).

- **Complessità dei messaggi.** LS richiede che ciascun nodo conosca il costo di ogni collegamento nella rete.

A ogni iterazione l'instradamento DV richiede scambi di messaggi tra nodi adiacenti. Il tempo richiesto perché l'algoritmo converga può dipendere da molti fattori. Quando cambiano i costi dei collegamenti, l'instradamento DV propaga i risultati dei costi cambiati se il nuovo costo ha causato la variazione del percorso a costo minimo per uno o più nodi connessi a tale collegamento.

- **Velocità di convergenza.** LS è un algoritmo $O(|N|^2)$ che richiede $O(|N| \cdot |E|)$ messaggi.

L'algoritmo DV può convergere lentamente e può presentare cicli di instradamento. DV presenta anche il problema del conteggio all'infinito.

- **Robustezza.** Che cosa avviene se un router si guasta, funziona male o viene sabotato?

Con LS, un router può comunicare via broadcast un costo sbagliato per uno dei suoi collegamenti connessi (ma non per altri). Un nodo può anche alterare o eliminare i pacchetti ricevuti in broadcast LS. Ma i nodi LS si occupano di calcolare soltanto le proprie tabelle di inoltro, e gli altri nodi effettuano calcoli simili per quanto li riguarda. ↗ robustezza.

Con DV, un nodo può comunicare percorsi a costo minimo errati a tutte le destinazioni. Più in generale notiamo che, a ogni iterazione, il calcolo di DV in un nodo viene comunicato ai suoi vicini e quindi ai vicini dei vicini alla successiva iterazione.

In questo senso, un calcolo errato su un nodo si può diffondere per l'intera rete.

In conclusione, nessuno dei due meccanismi di instradamento è nettamente superiore all'altro ed entrambi vengono utilizzati in Internet.

Intradamento gerarchico.

In LS e DV abbiamo visto la rete come una collezione di router interconnessi. Ciascuno di questi era indistinguibile dagli altri, tutti eseguivano lo stesso algoritmo per calcolare l'istradamento attraverso la rete.

Questo modello risulta semplicistico per due motivi:

- **Scalabilità:** al crescere del numero di router, il tempo per calcolare, memorizzare e comunicare le informazioni di istradamento diventa proibitivo. Internet è costituita da migliaia di host, archiviare le informazioni di istradamento su essi richiede troppa memoria e non c'è neanche banda per i pacchetti di dati.

- **Autonomia amministrativa:** ciascuno dovrebbe essere in grado di amministrare la propria rete in modo auspicabile, pur mantenendo la possibilità di connetterla alle reti esterne.

Questi problemi possono essere risolti **organizzando i router** in **sistemi autonomi (AS)**, composti da gruppi di router posti sotto lo stesso controllo amministrativo (ISP).

I router di un AS eseguono lo stesso algoritmo di istradamento e gli uni hanno informazioni sugli altri. L'algoritmo di istradamento in esecuzione in un AS è detto protocollo di istradamento interno al sistema autonomo (*intra-AS routing protocol*).

È necessario connettere gli AS tra loro, e pertanto uno o più router (router **gateway**) avranno il compito aggiuntivo di inoltrare pacchetti a destinazioni esterne.

Mostriamo tre sistemi autonomi (AS1, AS2, AS3), le righe marcate indicano connessioni a collegamento diretto tra coppie di router, quelle più sottili indicano sottoreti direttamente connesse a questi.

Nel caso in cui un router di un AS deve sapere come istradare un pacchetto verso una destinazione esterna al sistema stesso.

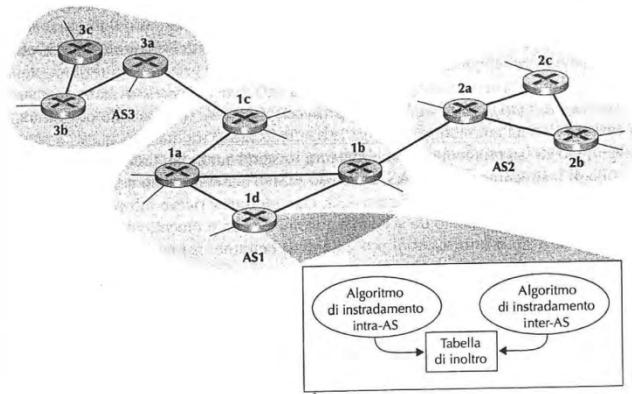


Figura 4.32 Esempio di sistemi autonomi interconnessi.

Come fa un router di sistema autonomo a sapere come istradare un pacchetto verso una destinazione esterna al sistema stesso?

Se il sistema autonomo *presenta un solo gateway* connesso a un altro sistema autonomo, dato che l'algoritmo di istradamento interno ha determinato il percorso a costo minimo da ogni router interno al gateway, questi sanno come inoltrare il pacchetto.

Il gateway, alla ricezione del pacchetto lo inoltra sul collegamento che conduce fuori dal sistema autonomo. Il sistema autonomo all'altro lato del collegamento, si assume poi la responsabilità di istradare il pacchetto verso la sua destinazione ultima.

Se il sistema autonomo *presenta più collegamenti verso l'esterno (gateway)* la questione è più impegnativa.

Per gestire questi due compiti si fa ricorso a un unico protocollo di istradamento tra sistemi autonomi (*inter-AS routing protocol*)

La procedura è questa:

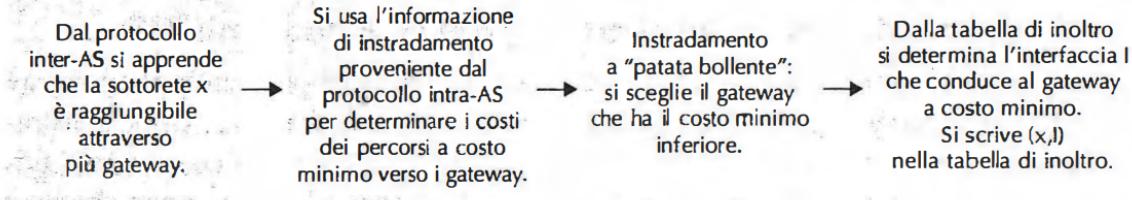


Figura 4.33 Passi per aggiungere una destinazione esterna al sistema autonomo in una tabella di inoltro di un router.

Tutti i sistemi autonomi usano lo stesso protocollo di instradamento **intra-AS**, chiamato BGP4.

hot potato routing: il sistema autonomo si sbarazza del pacchetto non appena possibile, nel modo meno costoso possibile. Questo avviene facendo inviare il pacchetto da un router al gateway con il minor costo.

Internet è costituita da una gerarchia di ISP interconnessi, ma qual è la relazione tra ISP e sistemi autonomi? In alcuni casi i router e i collegamenti che li connettono costituiscono un singolo sistema autonomo, in altri le reti sono ripartite in più sistemi autonomi. Alcuni ISP di primo livello usano, per esempio, un solo sistema autonomo e per la loro intera rete; altre suddividono la loro rete in decine di sistemi autonomi interconnessi.

Instradamento in Internet.

Il **compito dell'instradamento in Internet** è determinare il percorso che un datagramma seguirà dalla sorgente alla destinazione.

Un sistema autonomo è un insieme di router che eseguono lo stesso protocollo di instradamento e che ricadono sotto lo stesso controllo amministrativo e tecnico. Ciascun sistema autonomo è costituito da più sottoreti.

I **protocolli di instradamento intra-AS**, noti come **protocollo per gateway interni (IGP, interior gateway protocol)**, sono usati per **determinare come l'instradamento viene eseguito all'interno dei sistemi autonomi**. Vediamone due molto utilizzati:

- Routing information protocol (RIP);
- Open shortest path first (OSPF) a cui è correlato il protocollo intermediate-system-to-intermediate-system (IS-IS).

RIP è stato uno dei primi protocolli di Internet per l'**instradamento all'interno di un sistema autonomo**.

L'ampio utilizzo di RIP è legato all'inclusione nel 1982 in UNIX BSD che supportava TCP/IP.

RIP è di tipo distance-vector. I costi sono calcolati tra il router sorgente e la sottorete destinazione.

RIP utilizza il termine *hop* che rappresenta il numero di sottoreti attraversate lungo il percorso minimo dal router sorgente e la sottorete destinazione. I router adiacenti si scambiano gli aggiornamenti di instradamento approssimativamente ogni 30 secondi utilizzando un messaggio RIP response. Questo messaggio, inviato da un router o un host, contiene un elenco di 25 sottoreti di destinazione all'interno del sistema autonomo, nonché la distanza del mittente rispetto a ciascuna di tali sottoreti. I messaggi di risposta sono tali anche come RIP advertisement.

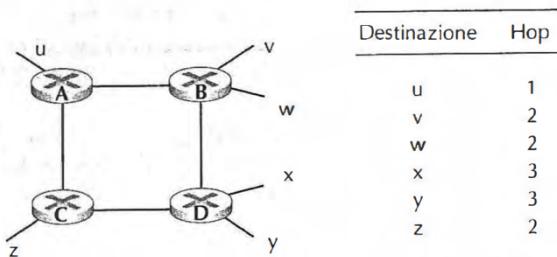


Figura 4.34 Numero di hop dal router sorgente A alle varie sottoreti.

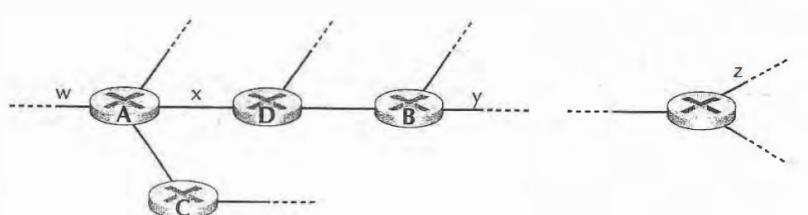


Figura 4.35 Una sottoparte di un sistema autonomo.

Ciascun router mantiene una **tavola di instradamento RIP**, che include il vettore delle distanze e la tabella di inoltro.

Sottorete di destinazione	Router successivo	Numero di hop verso la destinazione
w	A	2
y	B	2
z	B	7
x	—	1
...

Figura 4.36 Tabella di instradamento nel router D prima di ricevere notifica dal router A.

In linea di principio una tabella di instradamento avrà una riga per ciascuna sottorete del sistema autonomo.

Come abbiamo detto, i router RIP si scambiano annunci ogni 30 secondi. Dopo questo scambio le tabelle di instradamento vengono aggiornate. Se non vengono ricevute notizie per 180 secondi dal router vicino, o è stato spento o è caduto il collegamento.

Un router può anche richiedere informazioni sul costo dei vicini usando i messaggi di richiesta RIP. I router inviano messaggi di richiesta e di risposta RIP a tutti gli altri con UDP sulla porta 520. Il segmento UDP viene trasportato in un datagram standard IP.

Un processo chiamato routed esegue RIP, ossia mantiene informazioni di instradamento e scambia messaggi con i processi routed nei router vicini. Visto che RIP viene implementato come un processo a livello di applicazione, può inviare e ricevere messaggi su una socket standard e utilizzare un protocollo di trasporto standard. RIP è implementato come un protocollo a livello di applicazione che fa uso di UDP.

Anche **OSPF** viene utilizzato in Internet per il **routing interno ai sistemi autonomi**.

OSPF (open shortest path first) e IS-IS sono impiegati negli ISP di primo livello, mentre RIP in livelli inferiori e nelle reti aziendali.

Il termine open indica che le specifiche del protocollo sono disponibili.

OSPF è un protocollo **link state** che utilizza il **flooding** (*inondazione*) di informazioni riguardo lo stato dei collegamenti e l'**algoritmo di Dijkstra** per la determinazione del percorso a costo minimo.

Un router costruisce una mappa topologica, un grafo, dell'intero sistema autonomo e manda in esecuzione l'algoritmo di Dijkstra per determinare un albero dei percorsi minimi verso tutte le sottoreti. I costi dei collegamenti vengono fissati dall'amministratore di rete.

In OSPF, ogni qualvolta che si verifica un cambiamento nello stato di un collegamento, il router manda informazioni di instradamento via broadcast a tutti gli altri router nel sistema autonomo. Inoltre, invia lo stato dei collegamenti anche se questo non è cambiato.

Gli annunci OSPF sono contenuti in messaggi OSPF che vengono trasportati direttamente da IP come un protocollo di livello superiore con identificativo 89.

OSPF deve implementare **funzionalità**, tra cui il **trasferimento affidabile dei messaggi** e il **broadcast dello stato dei collegamenti**.

Tra i **vantaggi di OSPF** ricordiamo:

- **Sicurezza**: Gli scambi tra router OSPF possono essere autenticati, soltanto router fidati possono prendere parte al protocollo OSPF in un sistema autonomo, evitando che malintenzionati immettano informazioni errate nelle tabelle dei router.
- **Percorsi con lo stesso costo**: quando più percorsi verso una destinazione hanno lo stesso costo, OSPF consente di usarli senza doverne sceglierli uno per trasportare tutto il traffico.
- **Supporto integrato per l'instradamento unicast e multicast**: MOSPF che utilizza il database dei collegamenti OSPF e aggiunge un nuovo tipo di annuncio sullo stato dei collegamenti al meccanismo di broadcast.
- **Supporto alle gerarchie in un dominio di instradamenti**: Strutturare i sistemi autonomi in modo gerarchico.

Un sistema autonomo OSPF può essere configurato in aree che eseguono diversi algoritmi di instradamento OSPF: ciascun router in un'area invia lo stato dei suoi collegamenti agli altri router nell'area.

In ogni area, uno o più router di confine d'area si fa carico dell'instradamento dei pacchetti indirizzati all'esterno.

Un'area di un sistema autonomo OSPF è configurata per essere l'area di dorsale, il cui ruolo principale è quello di instradare il traffico tra le altre aree nel sistema autonomo. La dorsale contiene sempre tutti i router di confine del sistema autonomo, ma non necessariamente solo quelli. L'instradamento nell'area di un sistema autonomo richiede che i pacchetti siano instradati dapprima verso il proprio router di confine e da questi, attraverso la dorsale, al router di confine dell'area di destinazione.

Broadcasting.

Instradamento **broadcast** e **multicast**.

Con l'instradamento **broadcast** il livello di rete offre un servizio di consegna di un pacchetto spedito da un nodo origine a tutti gli altri nodi nella rete. L'instradamento **multicast** consente un nodo origine di inviare la copia di un pacchetto a un sottoinsieme di nodi della rete.

Algoritmi di instradamento broadcast.

Un nodo ottiene una comunicazione broadcast attraverso l'invio di una copia del pacchetto a ogni destinazione. Dati N nodi destinazione, il nodo sorgente prepara N copie del pacchetto, le indirizza alle varie destinazioni e infine invia le N copie alle N destinazioni utilizzando l'instradamento unicast.

Questo approccio, unicast a N vie è semplice, ma presenta lati negativi:

- **Inefficiente**: se il nodo origine è connesso al resto della rete tramite un unico collegamento sarà attraversato da N copie dello stesso pacchetto.
- Si ammette che i destinatari del broadcast, e i loro indirizzi, siano noti al mittente.
- Non ci si può affidare all'infrastruttura di instradamento unicast per ottenere un instradamento broadcast in situazioni in cui il broadcast stesso viene utilizzato per creare e aggiornare i percorsi.

Flooding non controllato.

Tecnica più ovvia per ottenere broadcast è basata sul **flooding** ("inondazione"), il nodo origine invia una copia del pacchetto a tutti i propri vicini. Quando un nodo riceve un pacchetto broadcast, lo duplica e lo inoltra a tutti i propri vicini, a eccezione di quello da cui lo ha ricevuto. Se il grafo è connesso, questo schema farà prevenire una copia del pacchetto broadcast a tutti i nodi.

Difetto: Se nel grafo c'è un ciclo, allora una o più copie di un pacchetto broadcast continueranno a percorre il ciclo indefinitamente. Quello che si viene a creare è una **tempesta di broadcast** e questa renderà la rete inutilizzabile.

Flooding controllato.

Per evitare la **tempesta di broadcast** è necessario che i nodi scelgano quando eseguire il flooding del pacchetto e quando no.

Per esempio, sulla base del fatto che sia già stata ricevuta e inoltrata una precedente copia del pacchetto.

Nel caso di **flooding controllato con numero di sequenza**, un nodo sorgente pone il proprio indirizzo e un numero di sequenza di broadcast nei pacchetti, prima di inviarli ai suoi vicini. Ciascun nodo mantiene una lista di indirizzi di origine e di numeri di sequenza per ogni pacchetto broadcast ricevuto, duplicato e inoltrato. Quando un nodo ne riceve uno, controlla se si trova nella lista, e in caso affermativo, lo elimina, altrimenti viene duplicato e inviato ai nodi adiacenti.

Un altro approccio per il flooding controllato è il **reverse path forwarding** (RPF, inoltro su percorso inverso) o anche **reverse path broadcast** (RPB, broadcast su percorso inverso).

Quando un router riceve un pacchetto broadcast, lo trasmette su tutti i propri collegamenti in uscita solo se è pervenuto attraverso il percorso unicast più breve tra il router e la sorgente.

In caso contrario, scarta il pacchetto in ingresso senza inoltrarlo in quanto sa che riceverà o avrà già ricevuto una copia del pacchetto sul collegamento che si trova sul suo percorso più breve verso il mittente.

RPF richiede solo che ogni nodo sappia quale, tra i propri vicini, si trova sul percorso unicast più breve che lo collega al mittente.

Il nodo utilizza questa informazione per decidere se applicare il flooding ai pacchetti broadcast che riceve.

Broadcast con albero di copertura.

RPF e **flooding controllato** con numeri di sequenza evitano le tempeste di broadcast, ma non sono in grado di eliminare le trasmissioni di pacchetti ridondanti.

Un albero di copertura di un grafo $G = (N, E)$ è un grafo $G' = (N, E')$ in cui E' è un sottoinsieme di E , G' è connesso e non contiene cicli e contiene tutti i nodi originali di G . Se gli archi hanno un costo e il costo di un albero è la somma dei costi dei suoi archi, allora un albero di copertura il cui costo sia il minimo, viene detto anche **minimum spanning tree**.

Un altro approccio per fornire broadcast è che i nodi della rete costruiscano uno **spanning tree**. Quando un nodo origine vuole inviare un pacchetto broadcast, lo spedisce su tutti i collegamenti incidenti che appartengono allo spanning tree. Il nodo che riceve un pacchetto broadcast lo inoltra a tutti i suoi vicini nello spanning tree (a eccezione di quello che glielo ha spedito). Gli alberi di copertura non solo eliminano i pacchetti broadcast ridondanti, ma una volta creati possono essere utilizzati dai nodi per dare inizio a un broadcast. Un nodo non deve necessariamente conoscere l'intero albero; è sufficiente che sappia quali tra i suoi vicini appartengano allo spanning tree.

Per la costruzione di uno spanning tree, con l'approccio basato su **nodo centrale** si definisce appunto un nodo centrale (punto di rendezvous o core). Utilizzando l'instradamento unicast, i nodi inoltrano al nodo centrale il messaggio di adesione che prosegue lungo le connessioni fino a quando raggiunge un router che già appartiene allo spanning tree o arriva al nodo centrale. In entrambi i casi, il percorso seguito dal messaggio definisce il ramo dell'albero tra il nodo periferico, che ha originato il messaggio, e il centro. Questo nuovo percorso potrebbe essere considerato come un innesto sullo spanning tree esistente.

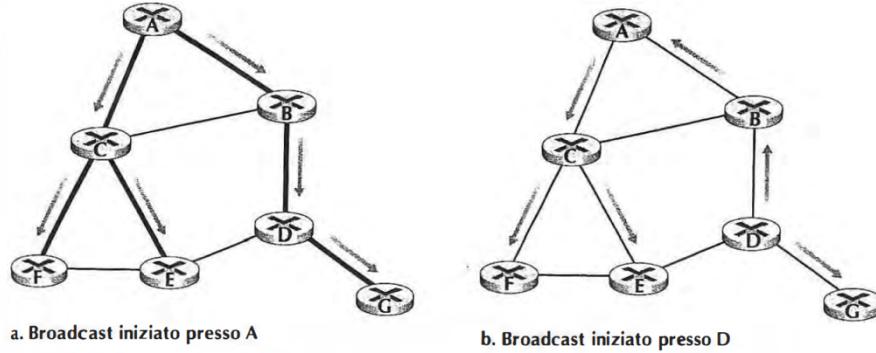


Figura 4.45 Broadcast su un albero di copertura.

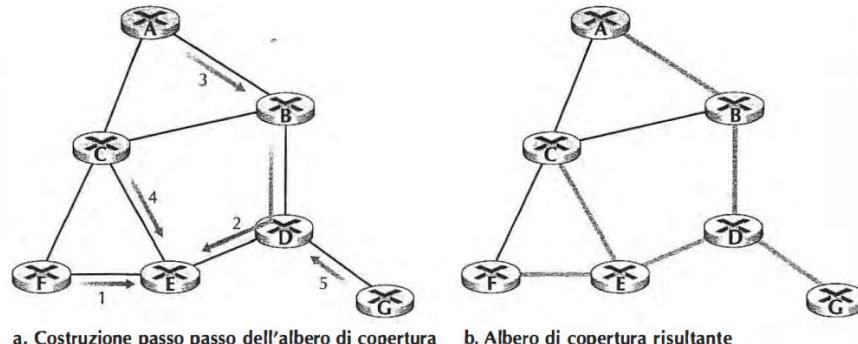


Figura 4.46 Costruzione di un albero di copertura basata su un nodo centrale.

Instradamento tra sistemi autonomi: BGP.

Gli ISP utilizzano RIP o OSPF per determinare i percorsi ottimali per le coppie sorgente-destinazione interne a un sistema autonomo.

Per i percorsi tra le coppie sorgente-destinazione che interessano più sistemi autonomi si utilizza il border gateway protocol version 4, detto **BGP4**, detto semplicemente **BGP**.

Questo è un protocollo di tipo inter-AS, ma può essere utilizzato anche per instradamento all'interno di sistemi autonomi.

In BGP coppie di router si scambiano informazioni di instradamento su connessioni TCP semi-permanenti usando la porta 179. Esistono anche connessioni TCP semi-permanenti tra router interni ai sistemi autonomi.

I router ai capi di una connessione TCP sono detti **BGP peer** e la connessione TCP con tutti i messaggi BGP inviati è detta **sessione BGP**.

- **sessione BGP esterna** (*eBGP*) nel caso in cui coinvolga due sistemi autonomi;
- **sessione BGP interna** (*iBGP*) nel caso in cui ci siano router dello stesso sistema autonomo.

In BGP le destinazioni non sono host ma prefissi CIDR che rappresentano una sottorete o una collezione di sottoreti.

Quando un gateway di un qualsiasi sistema autonomo riceve prefissi appresi tramite *eBGP*, utilizza le proprie sessioni *iBGP* per distribuire i prefissi agli altri router del sistema autonomo. Quando un router (gateway o meno) viene a conoscenza di un nuovo prefisso, lo memorizza in una nuova riga della propria tabella di inoltro.

In **BGP**, un **sistema autonomo** viene **identificato** dal suo numero di sistema autonomo (**ASN**) di solito univoco.

Quando un router annuncia un prefisso per una sessione BGP, include anche un **certo numero di attributi BGP**. Un prefisso insieme agli attributi viene detto anche rota. Due tra i più importanti attributi sono **AS-PATH** e **NEXT-HOP**.

- **AS-PATH**: elenca i sistemi autonomi attraverso i quali è passato l'annuncio del prefisso. Quando un prefisso attraversa un sistema autonomo aggiunge il proprio ASN all'attributo AS-PATH.

NEXT-HOP: fornire il delicato collegamento tra i protocolli di instradamento intra-AS e inter-AS.

Quando un router gateway riceve un annuncio di rota, utilizza le proprie politiche di importazione, per decidere se accettare o filtrare la rota e se impostare determinati attributi quali i parametri di scelta. Le politiche possono cancellare una rota.

Selezione dei percorsi.

BGP utilizza *eBGP* e *iBGP* per **distribuire le rotte ai router all'interno dei sistemi autonomi**. Un router può ricavare più di una rota verso un determinato prefisso. Se ne esistono 2 ci sono delle regole per individuare un'unica possibilità:

- Alle rotte viene assegnato come attributo un valore di preferenza locale, impostato direttamente dal router o appreso da un altro router nello stesso AS.
- Tra le rotte con lo stesso valore di preferenza locale, si seleziona quella con AS-PATH più breve.
- Tra le rotte con lo stesso valore di preferenza locale e la stessa lunghezza AS-PATH si seleziona quella il cui router di NEXT-HOP è più vicino, per con il percorso con costo minore.

Ci sono **vari tipi di messaggi BGP** scambiati tra peer tramite connessione TCP.

OPEN: apre la connessione TCP al peer e autentica il mittente;

UPDATE: pubblicizza il nuovo percorso (o ritira il vecchio)

KEEPALIVE: mantiene attiva la connessione in assenza di AGGIORNAMENTI; anche ACKs OPEN richiesta.

NOTIFICATION: segnala errori nel messaggio precedente; utilizzato anche per chiudere la connessione.

Politiche di instradamento:

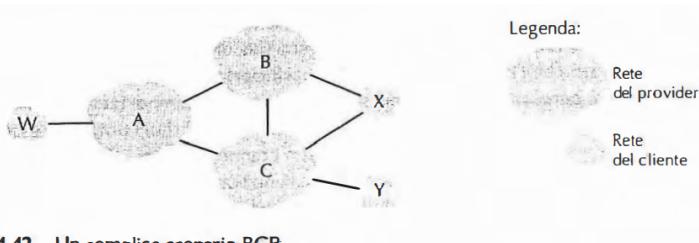


Figura 4.42 Un semplice scenario BGP.

Per evitare a X di smaltire il traffico tra B e C si utilizza il controllo degli annunci delle rotte BGP. X opererà come stub se annuncia ai suoi vicini B e C di non avere percorsi verso altre destinazioni tranne sé stessa.

Concentriamoci ora sulla rete di un provider, per esempio il sistema autonomo B, e supponiamo che questo abbia appreso (da A) che A presenta un percorso AW verso W. Allora B può installare il percorso BAW nella sua struttura dati per l'instradamento. Chiaramente, B vuole anche annunciare il percorso BAW al suo cliente X, per permettergli di raggiungere W tramite B. Per farlo, C potrebbe instradare il traffico verso W via CBAW. Se A, B e C sono provider di dorsale, allora B potrebbe giustamente pensare di non dover sopportare il peso (e il costo) del traffico in transito tra A e C e che sia compito di A e C assicurarsi che C possa instradare da e verso i clienti di A tramite una connessione diretta tra A e C. Una regola pratica seguita dagli ISP commerciali è quella per cui tutto il traffico che fluisce attraverso la rete di dorsale di un ISP deve avere origine e/o destinazione in una sua rete cliente.

Multicast.

L'instradamento **multicast** consente a un nodo origine di inviare la copia di un pacchetto a un sottoinsieme di nodi della rete.

Molte applicazioni richiedono la consegna di pacchetti da uno o più mittenti a un gruppo di destinatari. Queste includono il trasferimento di grandi quantità di dati, lo streaming di contenuti multimediali, le applicazioni con dati condivisi, l'aggiornamento di dati e i giochi interattivi.

Nella comunicazione **multicast** ci troviamo ad affrontare due problemi:

- l'identificazione dei destinatari dei pacchetti;
- l'indirizzamento dei pacchetti.

Nella comunicazione unicast l'*indirizzo IP del destinatario* è presente in ciascun datagramma e identifica il singolo destinatario.

Nel caso di broadcast, tutti i nodi devono ricevere il pacchetto e pertanto sono richiesti indirizzi di destinazione.

I **pacchetti multicast** vengono indirizzati utilizzando **indirizzi indiretti**. Si usa un identificatore singolo per un gruppo di destinatari, cui viene consegnata una copia del pacchetto.

In Internet, l'identificatore che rappresenta un gruppo di destinatari è un indirizzo P multipla multicast di classe D. Il gruppo di destinatari associati a un indirizzo di classe D viene detto **gruppo multicast**.

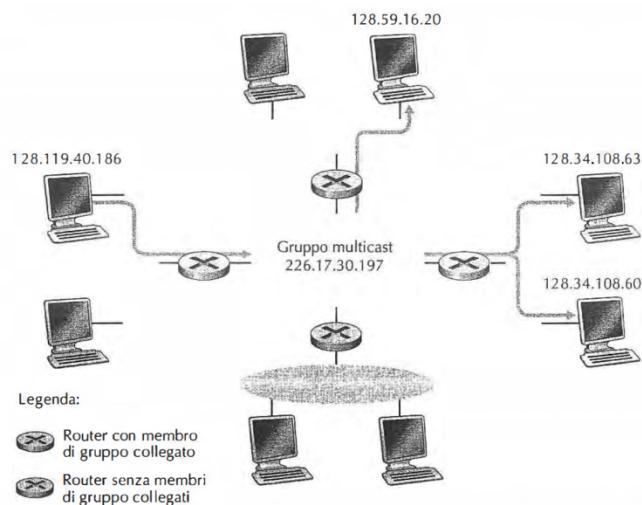


Figura 4.47 Gruppo multicast: il datagramma A indirizzato al gruppo viene consegnato a tutti i suoi membri.

Quattro host associati all'indirizzo del gruppo multicast 226.17.30.197 e riceveranno tutti i datagrammi destinati a tale indirizzo.

Ogni host ha un indirizzo IP completamente indipendente dall'indirizzo del gruppo multicast di cui fa parte.

Internet group management protocol.

Il protocollo IGMP versione 3 opera tra un host e il router che gli è direttamente connesso. Primo router che un host vede sul percorso verso qualsiasi altro host al di fuori della propria rete locale, o come l'ultimo router su qualsiasi percorso verso l'host.

In generale, una rete locale connette vari host di cui, in ogni dato istante, solo alcuni apparterranno a un particolare gruppo multicast. Per recapitare i datagrammi multicast a livello di rete è necessariamente richiesto un altro protocollo che coordini i router multicast in Internet.

Il multicast a livello di Internet consiste di IGMP e i protocolli multicast di instradamento.

IGMP ha 3 **tipi di messaggio** encapsulati nei datagrammi IP.

I messaggi **membership query** servono a determinare l'insieme di tutti i gruppi multicast cui gli host su tale interfaccia hanno aderito.

Gli host rispondono a tale messaggio con un messaggio IGMP **membership report**, generato da un host quando un'applicazione aderisce a un gruppo multicast per la prima volta.

L'ultimo tipo di messaggio IGMP è **leave_group**, un router deduce che non ci sono host aderenti a un dato gruppo multicast quando non ottiene risposta a un messaggio membership_query con quel dato indirizzo di gruppo.

Questo è quello che viene chiamato **soft-state**, in cui lo stato viene alterato tramite timeout se non viene esplicitamente rinnovato.

Algoritmi di instradamento multicast.

Obiettivo: trovare un albero di collegamenti che collega tutti i router connessi a host appartenenti al gruppo multicast.

I pacchetti multicast verranno instradati lungo l'albero dal mittente a tutti gli host che appartengono all'albero multicast. L'albero potrebbe contenere router non connessi a host del gruppo multicast.

Ci sono 2 approcci:

- **Intradamento multicast con albero condiviso dal gruppo:** si basa sulla costruzione di un albero che include tutti i router periferici con host connessi e appartenenti al gruppo multicast. Si usa un approccio centralizzato per costruire l'albero di instradamento multicast e i router periferici con host connessi, che appartengono al gruppo multicast, inviano messaggi indirizzati al nodo centrale. Come nel broadcast si inoltra verso il centro un messaggio di adesione fino a quando si raggiunge un router che già appartiene all'albero multicast o si arriva al centro. Tutti i router lungo il percorso seguito dal messaggio di adesione inoltrano poi pacchetti multicast ricevuti ai router periferici che hanno dato il via un'adesione multicast.

- **Intradamento multicast con albero basato sull'origine:** il secondo approccio crea un albero per ciascuna origine nel gruppo multicast. In pratica, si utilizza un algoritmo **RPF** (con nodo origine x) per costruire un albero di distribuzione multicast dedicato ai datagrammi multicast generati da x. Si potrebbero avere pacchetti multicast indesiderati e la soluzione di RPF a questo è nota come **potatura (pruning)**. Un router multicast, che riceve pacchetti multicast e non è connesso a host aderenti al gruppo, invierà un messaggio di potatura al proprio router di upstream. Se un router riceve questi messaggi da tutti i suoi router di downstream, può inoltrare il messaggio di potatura in upstream.

Intradamento multicast in Internet.

Il **primo** protocollo di intradamento multicast utilizzato in Internet è il **distance-vector multicast routing protocol (DVMRP)** che implementa alberi basati sull'origine con RPF e potatura.

Il protocollo di intradamento multicast **più diffuso** in Internet è il **protocol-Independent multicast (PIM)**, che prende esplicitamente in considerazione due scenari di distribuzione multicast. La modalità densa PIM è una tecnica RPF diffondi-e-soltisci simile concettualmente a DVMRP.

PIM Sparse Mode (PIM-SM) parte dal presupposto che i destinatari di un particolare gruppo multicast saranno scarsamente distribuiti nella rete. In altre parole, si presume che la maggior parte delle sottoreti nella rete non desideri alcun dato pacchetto multicast. Per ricevere dati

multicast, i router devono informare esplicitamente i loro vicini a monte del loro interesse per particolari gruppi e fonti. I router utilizzano i messaggi PIM Join e Prune per entrare e uscire dagli alberi di distribuzione multicast.

PIM-SM è un protocollo soft-state. Ovvero, tutto lo stato è scaduto per un po' di tempo dopo aver ricevuto il messaggio di controllo che ne ha creato l'istanza. Per mantenere attivo lo stato, tutti i messaggi di unione PIM vengono periodicamente ritrasmessi.

PIM Dense Mode (PIM-DM) è un protocollo di routing multicast progettato con il presupposto opposto a PIM-SM, ovvero che i ricevitori per qualsiasi gruppo multicast siano distribuiti densamente nella rete. Cioè, si presume che la maggior parte (o almeno molte) sottoreti nella rete vorranno un dato pacchetto multicast. I dati multicast vengono inizialmente inviati a tutti gli host della rete. I router che non hanno host interessati inviano quindi messaggi PIM Prune per rimuoversi dall'albero.

PIM bidirezionale (BIDIR-PIM) è un terzo protocollo PIM, basato su PIM-SM. Il modo principale in cui BIDIR-PIM differisce da PIM-SM è nel metodo utilizzato per inviare i dati da una sorgente all'RP. Mentre in PIM-SM i dati vengono inviati utilizzando l'incapsulamento o un albero basato sul sorgente, in BIDIR-PIM i dati fluiscono verso l'RP lungo l'albero condiviso, che è bidirezionale - i dati fluiscono in entrambe le direzioni lungo un determinato ramo.

Capitolo 5/6.

Livello di collegamento e reti locali.

Il livello di rete fornisce un servizio di comunicazione tra due qualsiasi host della rete. I datagrammi attraversano una serie di collegamenti, cablati e wireless, iniziano all'host sorgente, passano attraverso una serie di router e switch (commutatori) e raggiungono la destinazione. Scendendo nella pila dei protocolli, dal livello di rete troviamo quello di collegamento.

Nel **livello di collegamento** esistono due tipologie fondamentali di canali.

- Al *primo tipo* appartengono i canali broadcast che connettono un gruppo di host nelle LAN wireless, nelle reti satellitari e nelle reti di accesso HFC.
- Al *secondo tipo* di canale a livello di collegamento appartiene il canale di comunicazione punto a punto, come quello che spesso si trova tra due router collegati da un canale a lunga distanza o tra il computer di un ufficio e lo switch Ethernet nelle vicinanze.

Qualunque dispositivo che opera a **livello di collegamento** viene chiamato **nodo**.

Ci riferiremo inoltre ai canali di comunicazione, che collegano nodi adiacenti lungo un cammino, come a **collegamenti (link)**.

Quindi, i datagrammi che devono essere trasferiti da un host sorgente a uno di destinazione, devono essere trasportati lungo ciascun collegamento nel percorso da un estremo all'altro.

Supponiamo di voler inviare un datagramma da uno degli host wireless a uno dei server. Il datagramma dovrà attraversare sei collegamenti. Su ogni collegamento, un nodo trasmittente incapsula il datagramma in un frame del livello di collegamento (link-layer frame) e lo trasmette lungo il collegamento stesso.

Servizi offerti dal livello di collegamento.

I **possibili servizi** che possono essere offerti dai protocolli a **livello di collegamento** sono:

- **Framing.** Quasi tutti i protocolli incapsulano i datagrammi del livello di rete all'interno di un frame a livello di collegamento, prima di trasmetterlo. I frame sono costituiti da un campo dati, nel quale è inserito il datagramma, e da vari campi di intestazione. La struttura del frame è specificata dal protocollo.

• **Accesso al collegamento.** Un protocollo che controlla l'accesso al mezzo trasmissivo (**MAC, medium access control**) specifica le regole con cui immettere i frame nel collegamento. Nei collegamenti punto a punto, un mittente e un destinatario, in cui il protocollo MAC è semplice (o non esiste), il mittente può inviare il frame quando il canale risulta libero. Nel caso in cui vari nodi condividono un singolo canale broadcast (il cosiddetto problema degli accessi multipli) il protocollo MAC aiuta a coordinare la trasmissione dei frame da parte dei nodi.

• **Consegna affidabile.** I protocolli a livello di collegamento che forniscono un servizio di consegna affidabile garantiscono il trasporto senza errori di ciascun datagramma. Abbiamo già alcuni protocolli di trasporto (come TCP) che forniscono un servizio di consegna affidabile. Anche il servizio di consegna affidabile del livello di collegamento può essere realizzato attraverso acknowledgment e ritrasmissioni.

• **Rilevazione e correzione degli errori.** Il nodo ricevente può decidere, erroneamente, che un bit in un frame sia 0 quando questo era stato trasmesso come 1, e viceversa. Gli errori di bit sono causati dall'attenuazione di segnale e dai disturbi elettromagnetici. C'è un meccanismo per rilevarne la presenza. Ciò è possibile grazie all'inserimento, da parte del nodo trasmittente, di bit di controllo di errore all'interno del frame e all'esecuzione di un semplice controllo da parte del nodo ricevente. Anche il livello di trasporto e di rete di Internet fornisce un rilevamento degli errori: il checksum, ma a livello di collegamento è però solitamente più sofisticato, in quanto implementato in hardware.

Dov'è implementato il livello di collegamento?

Il **livello di collegamento** è **implementato** nelle **line card dei router**.

Per un dato collegamento, il protocollo del livello di collegamento è sostanzialmente realizzato da un **adattatore di rete** (*network adapter*), o **scheda di rete** (*NIC, network interface card*). Il cuore della scheda di rete è il **controller a livello di collegamento** (link layer controller), che è di solito un chip dedicato, che implementa molti dei servizi a livello di collegamento. (framing, accesso al collegamento, rilevazione degli errori)

La maggior parte, quindi, delle funzionalità del controller è implementata in hardware. Prima molti adattatori di rete erano su schede fisicamente separate, ora sono sempre più spesso integrati sulla scheda madre del calcolatore.

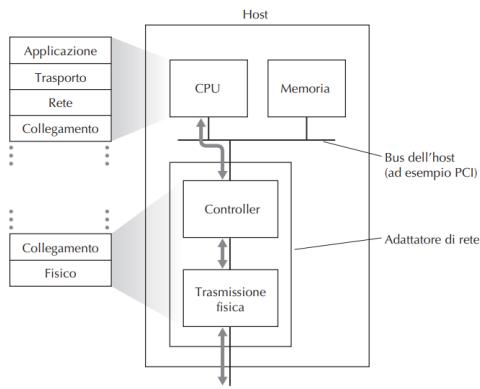


Figura 6.2 Adattatore di rete: le sue relazioni con gli altri componenti dell'host e con le funzionalità della pila di protocolli.

Tecniche di rilevazione e correzione degli errori.

Rilevamento e correzione degli errori sui bit sono due servizi forniti dal livello di collegamento, ma anche dal livello di trasporto.

Anche con l'utilizzo dei bit di rilevazione degli errori è possibile che ci siano degli errori non rilevati. Per risolvere questo problema ci sono tre tecniche per rilevare gli errori nei dati trasmessi: **controllo di parità** (*parity check*), per illustrare l'idea di base della rilevazione e correzione degli errori, tecniche di **checksum**, solitamente utilizzate nel livello di trasporto e **controllo a ridondanza ciclica** (*cyclic redundancy check*), in genere impiegato dalle schede di rete a livello di collegamento.

Controllo di parità.

La forma più semplice di rilevamento degli errori è quella che utilizza un **unico bit di parità** (*parity bit*).

Tale sistema prevede l'aggiunta di un bit ridondante ai dati, calcolato a seconda che il numero di bit che valgono 1 sia pari o dispari.

Quando si usa un bit di parità pari, si pone tale bit uguale a 1 se il numero di "1" in un certo insieme di bit è dispari (facendo diventare il numero totale di "1", incluso il bit di parità, pari). Quando invece si usa un bit di parità dispari, si pone tale bit uguale a 1 se il numero di "1" in un certo insieme di bit è pari (facendo diventare il numero totale di "1", incluso il bit di parità, dispari).

Se un numero dispari di bit (incluso il bit di parità) è cambiato durante la trasmissione di un insieme di bit, allora il bit di parità non risulterà in accordo con il numero di bit a 1 e di conseguenza indicherà che è avvenuto un errore durante la trasmissione. Usando un canale molto disturbato, può essere necessario un lungo tempo per effettuare una trasmissione corretta, o può non accadere mai.

La **capacità del ricevente sia di rilevare sia correggere** gli errori è conosciuta come **forward error correction** (**FEC**, correzione degli errori in avanti). Queste tecniche sono comunemente utilizzate in dispositivi audio di registrazione e riproduzione, come i lettori CD audio. Le tecniche FEC sono molto utili, perché possono diminuire il numero di ritrasmissioni e permettono al ricevente l'immediata correzione degli errori.

Checksum.

Un semplice metodo per eseguire il checksum è quello di **sommare interi da k bit** e usare i bit del risultato come bit per la rilevazione degli errori. Il checksum di Internet si basa su questo approccio: i dati sono trattati come interi di 16 bit e sommati. Il complemento a 1 di questa somma costituisce il checksum di Internet che viene trasposto nell'intestazione dei segmenti. Il ricevente controlla il checksum calcolando il complemento a 1 della somma dei dati ricevuti (compreso il checksum stesso) e verifica che tutti i bit del risultato siano 1. Se non è così, viene segnalato un errore.

Nei protocolli TCP e UDP il checksum è calcolato per tutti i campi (intestazione e dati). In IP il checksum è calcolato solo sull'intestazione, perché i segmenti TCP/UDP hanno il proprio.

A livello di trasporto (generalmente eseguito dal software) conviene utilizzare il checksum e a livello di collegamento (implementata mediante hardware dedicato nelle schede di rete) conviene il CRC.

Controllo a ridondanza ciclica. (CRC)

Una tecnica largamente utilizzata nelle reti è basata sui **codici di controllo a ridondanza ciclica**. I codici CRC sono anche detti **codici polinomiali**, in quanto è possibile vedere la stringa di bit da trasmettere come un polinomio in cui i coefficienti sono i bit della stringa, con le operazioni sulla stringa di bit interpretate come aritmetica polinomiale.

Consideriamo d bit costituenti i dati D da trasmettere e supponiamo che sorgente e destinazione si siano accordate su una stringa di $r+1$ bit, conosciuta come generatore G . È necessario che il bit più significativo (a sinistra) di G sia 1.

Dato un blocco di dati D , il mittente sceglierà r bit addizionali, R , e li unirà a D in modo da ottenere una stringa di $d+r$ bit che, interpretata come numero binario, sia esattamente divisibile per G (modulo 2)

Il processo di controllo CRC:

- se la divisione $(d+r)/G$ ha un resto diverso da zero, il ricevente sa che si è verificato un errore;
- altrimenti i dati sono accettati come corretti.

Con tale tecnica si possono rilevare tutti gli errori (a raffica) che coinvolgono meno di $r+1$ bit. Inoltre, il CRC può rilevare qualsiasi numero dispari di errori.

Distanza di Hamming.

- Date due parole codice e.g., 10001001 e 10110001 è possibile determinare in quanti bit ‘differiscono’ (XOR delle due parole e contate il numero di 1 del risultato)

Il numero di posizioni nelle quali le due parole di codice differiscono determina la loro distanza di Hamming.

- Se due parole codice hanno una distanza di Hamming d ci vorranno d errori sui singoli bit per tramutare una parola di codice nell’altra

Per come sono usati i bit di ridondanza se la lunghezza delle parole di codice è $n=m+r$ sono possibili 2^m messaggi dati ma non tutte le 2^n parole codice

- la distanza di Hamming di un codice è la minima distanza di Hamming tra due parole codice

Per come sono usati i bit di ridondanza se la lunghezza delle parole di codice è $n=m+r$ sono possibili 2^m messaggi dati ma non tutti 2^n parole codice

- la distanza di Hamming di un codice è la minima distanza di Hamming tra due parole codice

Per fare il detection di d errori serve un codice con distanza di Hamming $d+1$. Per correggere d errori serve un codice con distanza di Hamming $2d+1$

Collegamenti broadcast e protocolli di accesso multiplo.

Esistono **due tipi di collegamento di rete**: **punto a punto** e **broadcast**.

- Il collegamento **punto a punto** è costituito da un trasmittente a un'estremità del collegamento e da un unico ricevente all'altra.

- Il collegamento **broadcast** può avere più nodi trasmittenti e riceventi connessi allo stesso canale broadcast condiviso. Il termine broadcast indica che, quando un nodo trasmette un frame, il canale lo diffonde e tutti gli altri nodi ne ricevono una copia. Ethernet e Wireless LAN sono esempi di tecnologie con collegamenti broadcast.

Bisogna coordinare l'accesso di più nodi trasmittenti e riceventi in un canale broadcast condiviso, ossia il **problema dell'accesso multiplo**.

I nodi su un canale broadcast di una rete di calcolatori possono sia trasmettere sia ricevere. Un'analogia più appropriata è quella di una festa dove molte persone sono in una stanza parlando e ascoltandosi a vicenda.

Le reti di calcolatori usano protocolli simili – i cosiddetti **protocolli di accesso multiplo** – che fissano le modalità con cui i nodi regolano le loro trasmissioni sul canale condiviso. (Nelle LAN cablate, quelle wireless e le reti satellitari).

In pratica, centinaia o anche migliaia di nodi possono comunicare direttamente su un canale broadcast.

Dato che tutti i nodi sono in grado di trasmettere frame, è possibile che due o più lo facciano nello stesso istante, per cui tutti i nodi riceveranno contemporaneamente più frame. Tra questi si genera una collisione a causa della quale nessuno dei nodi riceventi riuscirà a interpretare i frame che, in un certo senso, risultano ingarbugliati tra loro. Di conseguenza con la collisione si verifica una perdita di frame, mentre il canale rimane inutilizzato.

Per far sì che il canale broadcast venga utilizzato in modo efficiente, occorre coordinare le trasmissioni dei nodi attivi.

Si possono classificare praticamente tutti i protocolli di accesso multiplo in una di queste categorie: **protocolli a suddivisione del canale** (*channel partitioning protocol*), **protocolli ad accesso casuale** (*random access protocol*) e **protocolli a rotazione** (*taking-turn protocol*).

Caratteristiche di un protocollo di accesso multiplo per un **canale broadcast con velocità di R bit al secondo**.

1. Quando un solo nodo deve inviare dati, il throughput = R bps.
2. Quando M nodi devono inviare dati, questi dispongono di un throughput pari a R/M bps.
3. Il protocollo è **decentralizzato**: non ci sono nodi principali che qualora non funzionassero bene potrebbero rendere inattivo l'intero sistema.
4. Il protocollo è **semplice**, in modo che risulti economico da implementare.

Protocolli a suddivisione del canale.

Il **multiplexing a divisione di tempo**, **TDM** (*time division multiplexing*) e quello a **divisione di frequenza**, **FDM** (*frequency division multiplexing*), vengono utilizzate per **suddividere la larghezza di banda di un canale broadcast fra i nodi che lo condividono**.

Se il canale supporti N nodi e che la sua velocità di trasmissione sia di R bps.

TDM suddivide il canale in intervalli di tempo (time frame) e poi divide ciascun intervallo di tempo in N slot temporali (time slot).

Ogni slot è quindi assegnato a uno degli N nodi. Ogni volta che un nodo ha un pacchetto da inviare, trasmette i bit del pacchetto durante lo slot di tempo che gli è stato assegnato. Le dimensioni dello slot sono stabilite in modo da consentire la trasmissione di un singolo pacchetto. Lo stesso tempo spetterà quindi a un altro invitato, e così via. Una volta che tutti hanno avuto l'opportunità di parlare, la sequenza si ripete. TDM viene utilizzato in quanto riesce a evitare le collisioni ed è perfettamente imparziale: ciascun nodo ottiene, durante ciascun intervallo di tempo, un tasso trasmissivo di R/N bps.

TDM presenta due specifici inconvenienti. In effetti anche quando non vi sono altri nodi che devono inviare un pacchetto, quello che vuole trasmettere è vincolato ad attendere il suo turno nella sequenza di trasmissione e a non superare il limite medio di R/N bps.

FDM suddivide il canale condiviso in frequenze differenti (ciascuna con una larghezza di banda di R/N) e assegna ciascuna frequenza a un nodo. Quindi, a partire da un canale da R bps, FDM crea N canali di R/N bps. FDM evita le collisioni e divide equamente la larghezza di banda tra gli N nodi. Ma, anche con FDM la larghezza di banda è limitata a R/N, pure quando vi è un solo nodo che ha un pacchetto da spedire.

C'è anche un terzo protocollo di suddivisione del canale è l'**accesso multiplo a divisione di codice (CDMA, code division multiple access)**.

Mentre **TDM** e **FDM** assegnano rispettivamente ai *nodi slot di tempo e di frequenze*, **CDMA** assegna loro un *codice*.

Ciascun nodo, quindi, utilizza il proprio codice univoco per codificare i dati inviati.

Se i codici sono scelti accuratamente, le reti CDMA avranno un'eccellente proprietà:

- consentiranno a nodi differenti di trasmettere simultaneamente e ai rispettivi destinatari di ricevere correttamente i bit dei dati codificati (assumendo che il ricevente conosca il codice) nonostante le interferenze derivanti dalle trasmissioni degli altri nodi.

Protocolli ad accesso casuale.

Avviene *in situazioni in cui un nodo trasmette sempre alla massima velocità consentita dal canale, cioè R bps*. Quando si verifica una **collisione**, i nodi coinvolti ritrasmettono ripetutamente i loro frame (cioè i pacchetti) fino a quando raggiungono la destinazione, senza collisioni.

La ritrasmissione del frame non è immediata, ma il nodo attende per un periodo di tempo casuale (random delay); ogni nodo coinvolto in una collisione seleziona un ritardo casuale indipendente da quello degli altri nodi.

Le differenti scelte arbitrarie del tempo di attesa operate dai diversi nodi possono consentire ai frame di attraversare il canale senza ulteriori collisioni.

Vediamo il **protocollo ALOHA** e i **protocolli di accesso multiplo con rilevazione della portante (CSMA, carrier sense multiple access)**.

Ethernet è un famoso protocollo CSMA largamente utilizzato.

Slotted ALOHA.

Si assume nella trattazione di questo protocollo che:

- Tutti i frame sono lunghi L bit e il tempo è diviso in slot di uguale durata L/R secondi;
- I nodi cominciano la trasmissione dei frame solo all'inizio degli slot;
- I nodi siano sincronizzati in modo che tutti sappiano quando iniziano gli slot.

Il protocollo prevede che quando un nodo ha un nuovo frame da spedire, attende fino all'inizio dello slot successivo e poi trasmette l'intero frame. Se non si verifica una collisione non occorre effettuare una ritrasmissione. Se si verifica una collisione, il nodo ritrasmette con probabilità p il suo frame durante gli slot successivi. La probabilità è un numero tra 0 e 1, quindi il nodo al prossimo slot può trasmettere o non trasmettere. Inoltre, le probabilità tra tutti i nodi sono indipendenti tra loro.

A differenza della suddivisione del canale, **slotted ALOHA** consente a un singolo nodo di trasmettere continuamente pacchetti alla massima velocità del canale, quando è il solo nodo attivo. Un nodo si dice **attivo** se ha un *frame da trasmettere*. Slotted ALOHA è anche fortemente **decentralizzato**, in quanto ciascun nodo rileva le collisioni e decide indipendentemente quando ritrasmettere, anche se è comunque necessario che gli slot siano sincronizzati ai nodi.

Slotted ALOHA funziona bene quando è attivo un solo nodo, ma è inefficiente in presenza di molti nodi attivi perché in primo luogo, una certa frazione degli slot presenterà collisioni e di conseguenza andrà "sprecata".

ALOHA.

Slotted ALOHA richiede che tutti i nodi sincronizzino le loro trasmissioni a partire dall'inizio di uno slot.

Il primo protocollo **ALOHA** era in realtà un *protocollo privo di slot*, completamente decentralizzato. Nel protocollo **ALOHA puro**, appena arriva un frame il nodo lo trasmette immediatamente e integralmente nel canale broadcast.

Se un frame va in collisione, allora il nodo lo ritrasmette immediatamente con probabilità p o aspetterà, restando inattivo per un altro periodo di tempo, con probabilità 1-p.

CSMA: accesso multiplo con rilevamento della portante.

Nei due protocolli ALOHA i nodi prendono la decisione di trasmettere indipendentemente dall'attività degli altri nodi collegati al canale broadcast. Un nodo non presta attenzione al fatto che vi sia un altro nodo che sta trasmettendo né arresta la trasmissione se un altro nodo inizia a interferire con la sua trasmissione.

I protocolli ALOHA corrispondono a un maleducato che continua a parlare senza preoccuparsi del fatto che altri stiano conversando tra loro.

In particolare, esistono due regole importanti per dialogare in modo civile, che sono le seguenti.

- Ascoltare prima di parlare. Nel mondo delle reti, ciò è chiamato **rilevamento della portante (carrier sensing)**: un nodo ascolta il canale prima di trasmettere. Se il canale sta già trasmettendo un frame, il nodo aspetta finché rileva che il canale è libero per un intervallo di tempo e quindi inizia la trasmissione.

- Se qualcun altro comincia a parlare insieme a voi, smettete di parlare. Nel mondo delle reti, ciò si chiama **rilevamento della collisione (collision detection)**: il nodo che sta trasmettendo rimane contemporaneamente in ascolto del canale. Se osserva che un altro nodo sta trasmettendo un frame che interferisce col suo, arresta la propria trasmissione, aspetta un intervallo di tempo casuale e poi ripete il processo.

Queste due regole sono alla base dei **protocolli CSMA** (*carrier sense multiple access, accesso multiplo con rilevamento della portante*) e **CSMA/CD** (*CSMA with collision detection, CSMA con rilevamento della collisione*).

Il ritardo di propagazione da un estremo all'altro di un canale broadcast (il tempo richiesto da un segnale per propagarsi da un nodo all'altro) avrà un ruolo cruciale nel determinare le sue prestazioni. Maggiore è questo ritardo, maggiore sarà la possibilità che il nodo, pur attento a rilevare la portante, non si accorga che è già cominciata la trasmissione da parte di un altro nodo.

Accesso multiplo a rilevazione della portante con rilevamento delle collisioni. (CSMA/CD)

I nodi non eseguono il rilevamento delle collisioni. Quando un nodo rileva una collisione, cessa immediatamente la trasmissione.

Ora i due nodi terminano la loro trasmissione poco dopo aver rilevato la collisione. L'introduzione della rilevazione di collisione in un protocollo di accesso multiplo migliorerà le prestazioni. Prima di analizzare il protocollo CSMA/CD riassumiamo le sue operazioni dal punto di vista di una scheda di rete collegata a un canale broadcast.

1. La scheda ottiene direttamente un datagramma dal livello di rete, prepara un frame a livello di collegamento e lo sistema in un suo buffer.
2. Quando riscontra che il canale è libero, inizia la trasmissione del frame. Se il canale risulta occupato, resta in attesa sino al momento in cui non rileva più il segnale e solo allora inizia l'invio del frame.
3. Durante la trasmissione verifica la presenza di eventuali segnali provenienti da altre schede di rete sul canale broadcast.
4. La scheda di rete, se trasmette l'intero frame senza rilevare energia di segnale proveniente da altre schede, ha finito il suo lavoro; altrimenti, se riscontra energia di segnale durante la trasmissione, interrompe immediatamente la trasmissione del frame.
5. Dopo aver annullato la trasmissione, la scheda di rete aspetta per un tempo casuale e poi ritorna al passo 2.

Se due nodi trasmettono frame simultaneamente e quindi aspettano per lo stesso periodo di tempo, continueranno a entrare in collisione per sempre. Ma qual è un intervallo di tempo opportuno da scegliere per il tempo di attesa casuale (detto anche tempo di backoff)?

Occorrerebbe un intervallo di tempo piccolo, quando il numero di nodi in collisione è piccolo, e uno grande quando il numero di nodi è grande.

L'algoritmo di **binary exponential backoff** (attesa binaria esponenziale), usato sia in Ethernet sia nei protocolli di accesso multiplo delle reti HFC DOCSIS, risolve elegantemente tale problema. Quando il trasmettitore riscontra l'n-esima collisione durante la trasmissione di un dato frame, stabilisce casualmente un valore K nell'insieme {0, 1, 2, ..., 2n – 1}. Quindi più alto è il numero di collisioni, più grande sarà l'intervallo da cui K viene estratto.

Efficienza di CSMA/CD.

Quando un solo nodo ha un frame da inviare può trasmettere alla massima velocità del canale (10 Mbps, 100 Mbps o 1 Gbps per Ethernet). Se, invece, i nodi che vogliono trasmettere sono numerosi, l'elevativa velocità di trasmissione sul canale può risultare notevolmente inferiore.

Definiamo **efficienza di CSMA/CD** la frazione di tempo media durante la quale i frame sono trasferiti sul canale senza collisioni in presenza di un alto numero di nodi attivi, con un'elevata quantità di frame da inviare.

Quando il ritardo di propagazione è nullo, i nodi in cui si verifica una collisione interromperanno immediatamente la trasmissione senza sprecare la capacità del canale.

Protocolli a rotazione.

Ricordiamo che due proprietà auspicabili di un protocollo di accesso multiplo sono:

(1) quando un solo nodo è attivo, deve avere un throughput di R bps, e (2) quando sono attivi M nodi, ciascuno deve avere un throughput vicino a R/M bps.

I protocolli ALOHA e CSMA possiedono la prima proprietà, ma non la seconda. Questo ha incoraggiato i ricercatori a creare un'altra classe di protocolli: i **protocolli a rotazione** (*taking-turn protocol*). Esistono decine di questi protocolli, ciascuno con molte varianti; ne vedremo qui due tra le più importanti.

Il primo è il **protocollo polling** nel quale uno dei nodi, il principale, interpella "a turno" gli altri. In particolare, il nodo principale invia un messaggio al nodo 1, comunicandogli che può trasmettere fino a un dato numero massimo di frame.

Dopo che il nodo 1 ha trasmesso dei frame, il nodo principale avvisa il nodo 2 che può iniziare a inviare un dato numero massimo di frame. Il nodo principale può determinare che un nodo ha terminato di inviare i propri frame, osservando la mancanza di segnale sul canale. La procedura prosegue in questo modo, con il nodo principale che interpella in modo ciclico tutti gli altri nodi.

Il protocollo polling elimina le collisioni e gli slot vuoti che si riscontrano nei protocolli ad accesso casuale e quindi può avere un'efficienza più elevata, ma presenta anche alcuni svantaggi. Il primo è l'introduzione del ritardo di polling: il tempo richiesto per notificare a un nodo il permesso di trasmettere.

Il secondo inconveniente, potenzialmente molto più serio, è che, se il nodo principale si guasta, l'intero canale diventa inattivo. Il protocollo 802.15 e quello Bluetooth sono esempi di protocolli polling.

Il secondo **protocollo a rotazione** è il **protocollo token-passing** (*a passaggio di gettone*). In questo caso non esiste un nodo principale, ma un frame, un messaggio di controllo detto **token** (gettone), che circola fra i nodi seguendo un ordine prefissato.

Per esempio: il nodo 1 deve sempre spedire il token al nodo 2 che deve inviarlo al nodo 3 e così via fino a quando il nodo N lo restituirà al nodo 1 che riprenderà la procedura. Se il nodo che riceve il token non ha pacchetti da inviare, lo inoltra immediatamente al successivo. Altrimenti, procede a trasmettere il numero massimo di frame consentito, prima di inoltrare il token al nodo successivo. Questo protocollo è decentralizzato e altamente efficiente, ma non è privo di problemi. Per esempio, il guasto di un nodo può mettere fuori servizio l'intero canale. Oppure, se un nodo non riesce a inoltrare il token, occorre invocare procedure di recupero, per rimetterlo in circolazione.

Reti locali commutate.

Gli switch, operando a livello di collegamento, commutano frame piuttosto che datagrammi a livello di rete, per cui non riconoscono gli indirizzi a livello di rete e non usano algoritmi di instradamento quali RIP e OSPF per determinare i percorsi attraverso la rete degli switch di livello 2. Al posto degli indirizzi IP usano indirizzi a livello di collegamento per inoltrare i frame attraverso la rete.

Host e router hanno indirizzi a livello di collegamento e i nodi hanno anche un indirizzo a livello di rete.

Il protocollo per la risoluzione degli indirizzi (ARP, address resolution protocol) fornisce ai nodi un meccanismo per trasformare indirizzi IP in indirizzi a livello di collegamento.

Indirizzi MAC.

Non sono i nodi (cioè, gli host e i router) ad avere indirizzi a livello di collegamento, ma sono i loro **adattatori** (*schede di rete*) a possederli. Questi sono indicati con varie terminologie che vanno da **indirizzo fisico**, **indirizzo LAN** o **indirizzo MAC**.

Per molte LAN (come Ethernet e 802.11), l'**indirizzo MAC** è lungo sei byte, il che consente di avere 248 possibili indirizzi. Questi indirizzi sono generalmente espressi in *notazione esadecimale*, scrivendo due cifre esadecimali per ogni byte. Sebbene sia stato progettato per essere permanente è ora possibile cambiare l'indirizzo MAC della scheda di rete via software.

Non esistono due schede di rete con lo stesso indirizzo MAC.

L'indirizzo MAC di una scheda di rete ha una struttura piatta (cioè **non gerarchica**) e non cambia mai. Un portatile con una scheda Ethernet avrà sempre lo stesso indirizzo MAC, indipendentemente dal luogo in cui il computer è utilizzato. Al contrario, gli indirizzi IP hanno una struttura gerarchica, e che l'indirizzo IP cambia la rete al quale è collegato. L'indirizzo MAC = numero di codice fiscale di una persona, che ha non varia a seconda del luogo in cui la persona si trasferisce. Gli indirizzi IP = all'indirizzo postale di una persona. Questi indirizzi hanno una struttura gerarchica e devono essere aggiornati quando la persona cambia residenza.

Protocollo per la risoluzione degli indirizzi. (ARP)

Dato che esistono sia *indirizzi del livello di rete* (come gli **indirizzi IP** di Internet) sia *indirizzi del livello di collegamento* (gli **indirizzi MAC**), si presenta la necessità della loro **conversione**.

Internet affida questo compito al protocollo di risoluzione degli indirizzi (**ARP**, address resolution protocol).

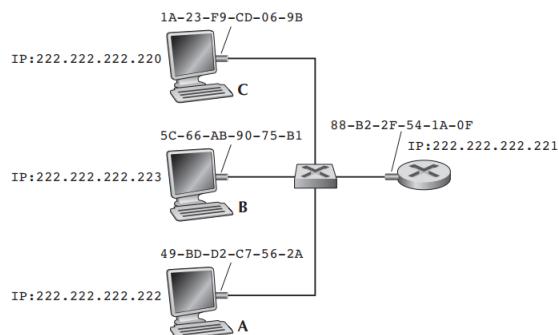


Figura 6.17 Ogni nodo di una LAN ha un indirizzo IP e l'adattatore di ciascun nodo ha un indirizzo MAC.

Un *modulo ARP traduce un indirizzo IP in un indirizzo MAC*. Per molti aspetti è simile al DNS, che traduce i nomi degli host in indirizzi IP. Ma DNS esegue l'operazione per host localizzati in qualunque punto di Internet, mentre ARP risolve soltanto gli indirizzi IP per i nodi nella stessa sottorete.

Se un nodo a Torino cerca di utilizzare ARP per risolvere l'indirizzo IP di un nodo di Venezia, ARP segnala un errore.

Funzionamento ARP.

Nella RAM dei nodi vi è una **tavella ARP** che contiene la corrispondenza tra indirizzi IP e MAC.

Tabella ARP nel nodo 222.222.222.220.

Indirizzo IP	Indirizzo MAC	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Figura 6.18 Una possibile tabella ARP per il nodo 222.222.222.220.

TTL (time-to-live tempo di vita) indica quando bisognerà eliminare una data voce dalla tabella. Il tipico TTL è di 20 minuti.

La tabella non contiene necessariamente una voce per ciascun nodo della sottorete perché alcuni possono essere stati cancellati (perché scaduto il loro TTL), altri possono non essere mai stati inseriti.

- Supponiamo adesso che il nodo 222.222.222.220 voglia inviare un datagramma a un altro nodo della sottorete.
- Il nodo trasmittente ha la necessità di ottenere l'indirizzo MAC del nodo di destinazione partendo dall'indirizzo IP di quel nodo.

- Ciò è semplice se la tabella ARP del nodo trasmittente ha una voce per il nodo di destinazione.
- Se la tabella al momento non la possiede il nodo trasmittente utilizza il protocollo ARP per la conversione dell'indirizzo. Lo scopo di un pacchetto ARP di richiesta è interrogare tutti gli altri nodi della sottorete riguardo all'indirizzo MAC corrispondente all'indirizzo IP da risolvere. Il frame contenente la richiesta ARP è ricevuto da tutte le altre schede di rete sulla sottorete. Ciascuna di queste, poiché l'indirizzo è quello broadcast, trasferisce il pacchetto ARP al proprio nodo che controlla se il proprio indirizzo IP corrisponde a quello di destinazione indicato nel pacchetto ARP.
 - L'unico nodo (se esiste) che ha l'indirizzo corrispondente invia al nodo richiedente un frame di risposta ARP con la corrispondenza desiderata. Il nodo richiedente 222.222.222.220 può quindi aggiornare la sua tabella ARP e inviare il datagramma IP, incapsulato in un frame, il cui MAC di destinazione è quello del nodo che ha risposto alla richiesta ARP precedente.

- Il messaggio di richiesta ARP è inviato in un frame **broadcast**, mentre il messaggio di risposta ARP è inviato in un **frame standard**.
- ARP è **plug-and-play**, nel senso che la tabella ARP di un nodo si costruisce automaticamente e non deve essere configurata dall'amministratore del sistema; inoltre, se un nodo viene scollegato dalla sottorete, verrà eliminato dalla tabella dei nodi della sottorete.

ARP è considerato come un protocollo che sta al confine tra i livelli di collegamento e quelli di rete.

Come inviare un datagramma a un nodo esterno alla sottorete.

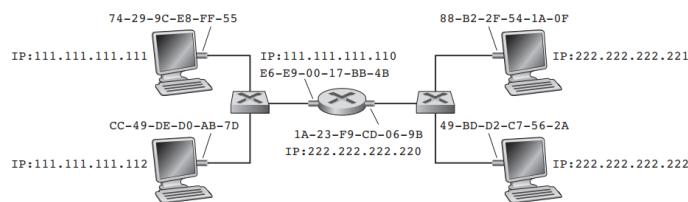


Figura 6.19 Due sottoreti connesse da un router.

Una rete costituita da due sottoreti interconnesse da un router. Ci sono due tipi di nodi: host e router. Ciascun host ha esattamente un indirizzo IP e una scheda di rete. Un router ha un indirizzo IP per ogni interfaccia che, a sua volta, dispone anche di propri moduli ARP (nel router) e di proprie schede di rete. Il router ha due interfacce, e quindi avrà due indirizzi IP, due moduli ARP e due schede di rete. Naturalmente, ogni scheda di rete ha il proprio indirizzo MAC.

- Se l'host 111.111.111.111 vuole inviare un datagramma IP all'host 222.222.222.222, l'host trasmittente passa il datagramma alla sua scheda di rete, ma deve anche comunicargli un appropriato indirizzo MAC di destinazione.
- Affinché un datagramma possa passare a un nodo sulla Sottorete 2, deve prima essere inviato all'interfaccia del router.
- Ma come può l'host trasmittente acquisire l'indirizzo MAC del router? Ovviamente, tramite l'utilizzo del protocollo ARP.
- Dopo aver ottenuto l'indirizzo MAC, la scheda di rete trasmittente crea un frame e lo trasferisce nella Sottorete 1. La scheda di rete del router sulla Sottorete 1 riconosce che il pacchetto è indirizzato a lui e allora lo passa al livello di rete del router.
- A questo punto, il datagramma IP è stato trasportato con successo dall'host sorgente al router.
- Il router consulta quindi la propria tabella e individua la scheda cui va inoltrato il datagramma, che a sua volta lo trasferisce alla sua scheda di rete; questa incapsula il datagramma nel nuovo frame e lo spedisce nella Sottorete 2.
- Adesso, l'indirizzo MAC di destinazione del frame è davvero quello finale, che il router ha ottenuto tramite ARP.

Ethernet.

- Ethernet ha conquistato il mercato delle reti cablate. **Ethernet** è di gran lunga la tecnologia per LAN cablate più diffusa ad alta velocità con vasta diffusione.
- Ethernet è stata per le reti locali quello che Internet è stata per la rete globale.

La grande diffusione di Ethernet ha fatto sì che le sue **componenti hardware** (schede di rete e switch) siano facilmente reperibili e particolarmente economiche.

- Le prime LAN Ethernet usavano un **cavo coassiale** come bus per connettere i nodi. Ethernet con topologia a bus è una LAN broadcast, in quanto tutti i frame trasmessi sono elaborati da tutte le schede di rete collegate al bus.
- Alla fine degli anni '90, la maggior parte delle aziende e università aveva sostituito le proprie LAN con installazioni Ethernet usando una topologia a stella, basata su **hub** dove gli **host e i router sono direttamente collegati a un hub tramite doppino intrecciato in rame**.

Un **hub** è un dispositivo a livello fisico che agisce sui singoli bit, piuttosto che sui frame. Quando un bit, rappresentato da 0 o 1, arriva a un'interfaccia, l'hub semplicemente rigenera il bit, amplifica la sua potenza e lo trasmette su tutte le altre interfacce.

- Ethernet con topologia a stella basata su hub, quindi, è anch'essa una LAN broadcast, poichéogniqualvolta l'hub riceve un bit da una delle sue interfacce, ne manda una copia a tutte le altre. In particolare, se l'hub riceve dei frame da due diverse interfacce contemporaneamente si verifica una collisione e i nodi che hanno creato i frame devono ritrasmetterli.
- All'inizio del 2000, Ethernet ha sperimentato un altro cambiamento rivoluzionario. Le installazioni Ethernet continuavano a usare una topologia a stella, ma l'hub al centro veniva sostituito da uno switch.

Struttura dei frame Ethernet.

Invio di un datagramma IP da un host a un altro, sulla stessa LAN Ethernet.

Il payload del frame Ethernet è un datagramma IP, ma Ethernet può anche trasportare altri tipi di pacchetti a livello di rete.

- La scheda di rete trasmittente, A, ha indirizzo MAC AA-AA-AA-AA-AA-AA e che il ricevente, B, ha indirizzo MAC BB-BB-BB-BB-BB-BB.
- La scheda di rete A incapsula il datagramma IP in un frame Ethernet e lo passa al livello fisico.
- B ottiene il frame dal livello fisico, estrae il datagramma IP e lo passa al livello di rete.

Esaminiamo adesso i sei campi del pacchetto di Ethernet.



Figura 6.20 Struttura di un frame Ethernet.

• **Campo dati** (da 46 a 1500 byte). Contiene il datagramma IP. L'**unità massima di trasmissione (MTU, maximum transfer unit)** per Ethernet è di 1500 byte, se il datagramma IP supera questo valore, l'host deve frammentare il datagramma. Altrimenti, se il datagramma IP è inferiore alla dimensione minima del campo dati, equivalente a 46 byte, il campo dovrà essere "riempito" (stuffed) fino a raggiungere quel dato valore.

- **Indirizzo di destinazione** (6 byte). Campo che contiene l'indirizzo MAC della scheda di rete di destinazione (BB-BB-BB-BB-BB-BB).
- **Indirizzo sorgente** (6 byte). Campo che include l'indirizzo MAC della scheda che trasmette il pacchetto (AA-AA-AA-AA-AA-AA).
- **Tipo** (2 byte). Indica il tipo di protocollo del livello di rete in uso durante la trasmissione.
- **Controllo a ridondanza ciclica (CRC)** (4 byte). Consente alla scheda di rete ricevente di rilevare la presenza di un errore nei bit del frame.
- **Preamble** (8 byte). I primi 6 byte hanno tutti valore 10101010 e servono a "svegliare" gli adattatori del ricevente e a sincronizzare i suoi clock con quelli del trasmittente. L'ultimo byte ha valore 10101010 e la serie dei due bit a 1 indica al destinatario che sta iniziando la parte dei dati utili;

Ethernet servizio senza connessione.

- A livello di rete, tutte le tecnologie **Ethernet** forniscono un **servizio senza connessione**, nel senso che quando una scheda di rete vuole inviare un datagramma a un host della rete, non fa altro che incapsularlo in un frame Ethernet e immetterlo nella LAN, senza alcuna forma di handshake con il destinatario. (analogo ai servizi senza connessione dei datagrammi IP a livello 3 e UDP a livello 4)
- Tutte le tecnologie Ethernet forniscono un servizio non affidabile a livello di rete.

Questa mancanza di affidabilità nel trasporto (a livello di collegamento) aiuta a mantenere Ethernet semplice ed economica, ma significa anche che il flusso dei datagrammi che attraversano il livello di rete può presentare delle lacune.

- Se l'applicazione utilizza UDP, allora l'applicazione vedrà delle lacune nei dati.
- Se, invece, l'applicazione sta usando TCP, quando TCP ritrasmette delle informazioni, queste ritorneranno alla scheda di rete che le ha scartate.

Per cui, in un certo senso, Ethernet effettua una ritrasmissione, anche se non ha idea se si tratta di dati nuovi o già trasmessi.

Tecnologie Ethernet.

Ethernet compare in molte forme differenti con svariate denominazioni come:

10 BASE-T, 10BASE-2, 100BASE-T, 1000BASE-LX e 10GBASE-T

standardizzate dai gruppi di lavoro IEEE 802.3 CSMA/CD. (Ethernet)

- La prima parte fa infatti riferimento alla **velocità dello standard** (10, 100, 1000, o 10G), 10 Mbps, 100 Mbps, 1 Gbps e 10 Gbps e 40 Gbps.
- "BASE" si riferisce a Ethernet in banda base, cioè che il mezzo fisico trasporta solo **traffico Ethernet**, ed è usata da quasi tutti gli standard 802.3.
- La parte finale del l'acronimo rimanda al **mezzo fisico**; Ethernet viene usata con un'ampia varietà di mezzi fisici, compresi cavi coassiali, fili di rame e fibre ottiche. Tipicamente, "T" si riferisce ai doppini in rame intrecciati.

Nella maggior parte delle installazioni odierne i nodi sono **collegati a uno switch** con segmenti punto a punto, realizzati con doppini in rame o fibre ottiche. In una LAN Ethernet basata su switch non ci sono collisioni e, quindi, non c'è bisogno di un protocollo MAC. Ciò che non è mai cambiato è il formato del frame Ethernet.

Switch a livello di collegamento.

Lo **switch** è un dispositivo di rete che si occupa di **commutazione a livello di collegamento**.

Lo switch agisce sull'indirizzamento e sull'instradamento all'interno delle reti LAN mediante indirizzo fisico, selezionando i frame ricevuti e dirigendoli verso la porta in uscita corretta. Perciò il **ruolo dello switch** è **ricevere i frame in ingresso e inoltrarli sui collegamenti in uscita**. Le interfacce di uscita dello switch hanno dei buffer, analogamente alle interfacce di uscita dei router per i datagrammi.

Inoltro e filtraggio.

Il **filtraggio (filtering)** è la **funzionalità dello switch** che **determina se un frame debba essere inoltrato a una qualche interfaccia o scartato**.

L'**inoltro (forwarding)** consiste **nell'individuazione dell'interfaccia verso cui il frame deve essere diretto** e, quindi, nell'inviarlo a quell'interfaccia.

Le operazioni di filtraggio e inoltro di uno switch sono eseguite mediante una **tavella di commutazione (switch table)** che contiene:

1. Indirizzo MAC del nodo;
2. Interfaccia dello switch che conduce al nodo;
3. Il momento in cui la voce per quel nodo è stata inserita nella tabella.

Anche se molti switch possono essere utilizzati per effettuare l'inoltro in base all'indirizzo MAC o IP di destinazione noi manterremo la distinzione tale per cui gli switch utilizzano gli indirizzi MAC e non quelli IP.

Nell'ipotesi in cui un frame con indirizzo MAC di destinazione A giunge allo switch sull'interfaccia x gli scenari possibili sono tre:

1. Nella switch table non vi è una voce per l'indirizzo A; lo switch inoltra copie del frame ai buffer di uscita di tutte le interfacce, eccetto x (manda il frame in broadcast);
2. Nella switch table vi è una voce che associa l'indirizzo A all'interfaccia x. Il frame proviene da un segmento di rete che contiene la scheda di rete A; quindi, non deve uscire su nessuna interfaccia e viene dunque scartato;
3. Nella switch table vi è una voce che associa l'indirizzo A all'interfaccia y \neq x; il frame viene inoltrato al segmento di LAN collegato all'interfaccia y.

Autoapprendimento.

Uno **switch** ha la pregevole *proprietà di costruire automaticamente, dinamicamente e in modo autonomo le proprie tabelle*, senza l'intervento di un operatore o di un protocollo di configurazione (gli switch auto-apprendono).

Quindi vengono detti dispositivi plug-and-play, in quanto non richiedono interventi dell'amministratore di rete o dell'utente. Basta semplicemente collegare i segmenti di LAN alle sue interfacce, senza dover configurare le tabelle al momento dell'installazione o quando un host è rimosso da un segmento LAN.

Proprietà della commutazione a livello di collegamento.

Caratteristiche e proprietà switch.

- **Eliminazione delle collisioni.** In una LAN costituita da switch e senza hub **non vi è spreco di banda a causa delle collisioni**. Gli switch, quindi, forniscono un significativo miglioramento delle prestazioni su LAN con collegamenti broadcast.

- **Collegamenti eterogenei.** Dato che uno switch isola un collegamento da un altro, i diversi collegamenti nella LAN possono funzionare a velocità diverse e possono usare mezzi trasmissivi diversi. Uno switch è ideale per combinare dispositivi già installati con altri nuovi.

- **Gestione.** Oltre a fornire una maggiore sicurezza uno **switch facilita anche la gestione di rete**. Per esempio, se una scheda di rete ha un malfunzionamento e manda continuamente frame Ethernet, uno switch può individuare il problema e disconnettere internamente la scheda di rete non funzionante. L'amministratore di rete non ha pertanto bisogno di essere sul posto per risolvere il problema. In modo analogo, il taglio di un cavo disconnette solo il nodo che stava usando quel cavo per collegarsi allo switch.

Gli switch raccolgono anche statistiche sull'uso della banda, tasso di collisioni e tipi di traffico, e rendono queste informazioni disponibili ai gestori di rete, informazioni che possono essere usate per rilevare e correggere i problemi e per pianificare come la LAN dovrà evolvere nel futuro.

Router e switch a confronto.

Router:	Switch:
<ul style="list-style-type: none">• I router sono commutatori di pacchetti store-and-forward che inoltrano pacchetti usando gli indirizzi a livello di rete.• I router sono commutatori di pacchetto di livello 3.	<ul style="list-style-type: none">• Gli switch sono commutatori di pacchetto store-and-forward, ma questi inoltrano i pacchetti utilizzando indirizzi MAC.• Gli switch sono commutatori di pacchetto di livello 2.

<ul style="list-style-type: none"> Dato che gli indirizzi di rete sono sovente gerarchici (e non lineari come gli indirizzi MAC), i pacchetti non presentano cicli attraverso i router, anche nel caso in cui la rete presenti percorsi ridondanti. In ogni modo i pacchetti rischiano di percorrere dei cicli soltanto se le tabelle dei router sono configurate male. Però, IP utilizza uno speciale campo di intestazione del datagramma per limitare la percorrenza dei cicli. Inoltre, contrariamente agli switch, i router proteggono dalle tempeste di broadcast a livello 2, ma non sono plug-and-play e quindi il loro indirizzo IP e quello degli host a essi collegati deve essere configurato. Inoltre, molte volte presentano un tempo di elaborazione per pacchetto più lungo di quello degli switch, in quanto devono eseguire l'elaborazione fino ai campi del livello 3. 	<ul style="list-style-type: none"> I moderni switch operanti in modalità "match-action" possono essere usati sia per inoltrare frame di livello 2 basandosi sull'indirizzo MAC di destinazione sia datagrammi di livello 3 usando l'indirizzo IP di destinazione. Gli switch, dispositivi plug-and-play che possono avere capacità di filtraggio e inoltro dei pacchetti relativamente alte ma, devono elaborare pacchetti solo fino al livello 2. Inoltre, una rete di switch molto grande richiederebbe delle grandi tabelle ARP nei nodi e nei router generando un considerevole traffico ARP e richiedendo molta elaborazione. Gli switch non offrono alcuna protezione contro le tempeste di broadcast.
---	--

Per le reti costituite al massimo da alcune centinaia di host e da pochi segmenti risultano sufficienti gli switch, in quanto localizzano il traffico e incrementano il throughput aggregato senza richiedere la configurazione degli indirizzi IP.

Le reti più grandi, costituite da migliaia di host, invece, includono tipicamente oltre agli switch anche dei router, che forniscono un più efficace isolamento del traffico, evitano le tempeste di broadcast, e utilizzano percorsi più funzionali fra gli host della rete.

Retrospettiva: cronaca di una richiesta di una pagina web.

Dobbiamo scaricare una pagina web. Esempio: uno studente, Bob, connette il proprio laptop a uno switch Ethernet della sua scuola e scarica una pagina web, per esempio, la home page di www.google.com. Questa richiesta apparentemente semplice sottintende moltissime cose.

DHCP, UDP, IP e Ethernet.

Supponiamo che Bob accenda il suo laptop e lo colleghi a un cavo Ethernet connesso allo switch Ethernet della scuola che a sua volta è connesso al router della scuola. Il router della scuola è connesso a un ISP, in questo esempio Comcast.net, che fornisce il servizio DNS alla scuola. Quindi il server DNS risiede nella rete Comcast piuttosto che nella rete della scuola. Assumiamo che il server DHCP sia eseguito nel router, come avviene spesso. Quando Bob connette il suo laptop alla rete, non può fare niente, nemmeno scaricare una pagina web, senza un indirizzo IP. Quindi la prima azione riguardante la rete intrapresa dal laptop di Bob è quella di eseguire il protocollo DHCP per ottenere un indirizzo IP, insieme ad altre informazioni, dal server DHCP locale.

- Il sistema operativo del laptop di Bob crea un messaggio DHCP request e inserisce questo messaggio in un segmento UDP con porta di destinazione 67 (server DHCP) e porta sorgente 68 (client DHCP). Il segmento UDP viene quindi inserito all'interno di un datagramma IP con indirizzo IP di destinazione broadcast (255.255.255.255) e indirizzo IP sorgente 0.0.0.0, in quanto il laptop di Bob non ha ancora un indirizzo IP.
- Il datagramma IP contenente il messaggio di richiesta DHCP è quindi posto in un frame Ethernet. Il frame Ethernet ha indirizzo MAC di destinazione FF:FF:FF:FF:FF:FF in modo che il frame venga inviato in broadcast a tutti i dispositivi connessi allo switch, tra i quali si spera ci sia un server DHCP; l'indirizzo MAC della sorgente del frame è quello del laptop di Bob, 00:16:D3:23:68:8A.
- Il frame Ethernet broadcast contenente la richiesta DHCP è il primo frame inviato dal laptop di Bob allo switch Ethernet. Lo switch invia in broadcast il frame in entrata a tutte le porte in uscita, compresa la porta che lo connette al router.
- Il router riceve il frame broadcast Ethernet che contiene la richiesta DHCP sulla sua interfaccia con indirizzo MAC 00:22:6B:45:1F:1B e il datagramma IP è estratto dal frame Ethernet. L'indirizzo IP di destinazione del datagramma broadcast indica che tale datagramma IP dovrebbe essere elaborato dai protocolli di livello superiore su questo nodo; quindi sul payload del datagramma (un segmento UDP) viene effettuato un

demultiplexing (Paragrafo 3.2) verso UDP e il messaggio di richiesta DHCP viene estratto dal segmento UDP. Ora il server DHCP ha il messaggio di richiesta DHCP.

5. Supponiamo ora che il server DHCP in esecuzione sul router possa allocare indirizzi IP nel blocco CIDR (Paragrafo 4.3.3) 68.85.2.0/24. In questo esempio tutti gli indirizzi IP usati all'interno della scuola sono contenuti nel blocco di indirizzi di Comcast. Supponiamo che il server DHCP allochi l'indirizzo 68.85.2.101 al laptop di Bob. Il server DHCP crea un messaggio DHCP ACK (Paragrafo 4.3.3) contenente tale indirizzo IP, insieme all'indirizzo IP del server DNS (68.87.71.226), l'indirizzo IP del gateway di default (68.85.2.1) e il blocco della sottorete (68.85.2.0/24 o equivalentemente la maschera di rete). Il messaggio DHCP è posto all'interno di un segmento UDP inserito in un datagramma IP a sua volta posto in un frame Ethernet. Il frame Ethernet ha come indirizzo MAC sorgente quello dell'interfaccia del router sulla rete domestica (00:22:6B:45:1F:1B) e come indirizzo MAC di destinazione quello del laptop di Bob (00:16:D3:23:68:8A).

6. Il frame Ethernet contenente il messaggio DHCP ACK è inviato (in unicast) dal router allo switch. Lo switch, poiché è in grado di auto-apprendere (Paragrafo 6.4.3) e ha precedentemente ricevuto dal laptop di Bob un frame Ethernet contenente la richiesta DHCP, sa di dover inoltrare un frame indirizzato a 00:16:D3:23:68:8A solo alla porta di uscita verso il laptop di Bob.

7. Il laptop di Bob riceve il frame Ethernet contenente il messaggio DHCP ACK, estraе dal frame Ethernet il datagramma IP, estraе dal datagramma IP il segmento UDP ed estraе dal segmento UDP il messaggio DHCP ACK. Il client DHCP di Bob memorizza il suo indirizzo IP e quello del server DNS. Inoltre installa l'indirizzo del gateway di default nella sua tabella di inoltro (Paragrafo 4.1). Il laptop di Bob invierà tutti i datagrammi il cui indirizzo di destinazione è all'esterno della sua sottorete 68.85.2.0/24 al gateway di default. A questo punto, il laptop di Bob ha inizializzato le sue componenti di rete ed è pronto a iniziare a elaborare la lettura della pagina web. Si noti che solo gli ultimi due passi DHCP dei quattro presentati nel Capitolo 4 sono realmente necessari

Siamo ancora all'inizio: DNS e ARP

Quando Bob scrive l'URL di www.google.com nel suo browser web inizia la lunga catena di eventi che alla fine porterà a visualizzare la home page di Google. Il web browser di Bob inizia il processo creando una socket TCP (Paragrafo 2.7) che verrà usata per inviare una richiesta HTTP (Paragrafo 2.2) a www.google.com. Il laptop di Bob, per creare la socket, deve conoscere l'indirizzo IP di www.google.com. Abbiamo visto nel Paragrafo 2.5 che il protocollo DNS viene usato per fornire il servizio di traduzione da nome a indirizzo IP.

8. Il sistema operativo del laptop di Bob crea un messaggio di DNS query (Paragrafo 2.5.3) inserendo la stringa “www.google.com” nella sezione riguardante la richiesta del messaggio DNS. Tale messaggio DNS è quindi posto all'interno di un segmento UDP con porta di destinazione 53 (server DNS). Il segmento UDP è quindi posto all'interno di un datagramma IP con indirizzo IP di destinazione 68.87.71.226 (l'indirizzo del server DNS restituito nel messaggio DHCP ACK al passo 5) e indirizzo IP sorgente 68.85.2.101.

9. Il laptop di Bob, quindi, inserisce il datagramma contenente il messaggio di richiesta DNS in un frame Ethernet che verrà inviato, con indirizzo a livello di collegamento, al gateway della rete della scuola di Bob. Il laptop di Bob, pur conoscendo l'indirizzo IP del gateway della scuola (68.85.2.1) tramite il messaggio DHCP ACK del passo 5, non conosce l'indirizzo MAC, per ottenere il quale deve usare il protocollo ARP.

10. Il laptop di Bob crea un messaggio di ARP query con indirizzo IP di destinazione 68.85.2.1 (il gateway di default), pone il messaggio ARP all'interno di un frame Ethernet con indirizzo di destinazione broadcast FF:FF:FF:FF:FF:FF e invia il frame Ethernet allo switch, che lo consegna a tutti i dispositivi connessi compreso il gateway.

11. Il router gateway riceve il frame contenente il messaggio di interrogazione ARP sull'interfaccia della rete della scuola e scopre che l'indirizzo IP 68.85.2.1 nel messaggio ARP corrisponde all'indirizzo IP della sua interfaccia. Il gateway prepara quindi un messaggio ARP reply che indica che il suo indirizzo MAC 00:22:6B:45:1F:1B corrisponde all'indirizzo IP 68.85.2.1. Pone il messaggio di risposta ARP in un frame Ethernet con l'indirizzo di destinazione 00:16:D3:23:68:8A (il laptop di Bob) e invia il frame allo switch che lo consegna al laptop di Bob.

12. Il laptop di Bob riceve il frame contenente il messaggio di risposta ARP ed estraе l'indirizzo MAC del gateway (00:22:6B:45:1F:1B) dal messaggio di risposta ARP.

13. Il laptop di Bob può ora (finalmente!) indirizzare il frame Ethernet contenente l'interrogazione DNS all'indirizzo MAC del gateway. Si noti che il datagramma IP in questo frame ha indirizzo IP di destinazione 68.87.71.226 (il server DNS), mentre il frame ha indirizzo di destinazione 00:22:6B:45:1F:1B (il router gateway). Il laptop di Bob invia questo frame allo switch che lo consegna al gateway.

Instradamento intra-dominio al server DNS

14. Il gateway riceve il frame ed estraе il datagramma IP contenente l'interrogazione DNS. Il router ricerca l'indirizzo di destinazione di tale datagramma (68.87.71.226) e determina sulla base della sua tabella di inoltro che il datagramma dovrebbe essere inviato al router più a sinistra nella rete Comcast mostrata nella Figura 6.32. Il datagramma IP è posto all'interno di un frame appropriato per il collegamento che connette il router della scuola al router Comcast più a sinistra; il frame viene trasmesso.

15. Il router più a sinistra nella rete Comcast riceve il frame, estraе il datagramma IP, esamina l'indirizzo di destinazione del datagramma (68.87.71.226) e determina l'interfaccia di uscita sulla quale inoltrare il datagramma al server DNS sulla base della sua tabella di inoltro, che è stata riempita dal protocollo intra-dominio di Comcast (come RIP, OSPF o IS-IS, Paragrafo 5.3) e dal protocollo inter-dominio di Internet, BGP.

16. Infine, il datagramma IP contenente l'interrogazione DNS arriva al server DNS, che estraе il messaggio di interrogazione DNS, ricerca il nome www.google.com nel suo database DNS (Paragrafo 2.5) e trova il record di risorsa DNS che contiene l'indirizzo IP (64.233.169.105) di www.google.com (assumendo che sia nella cache del server DNS). Si ricordi che i dati nella cache sono originati dal DNS server autoritativo (Paragrafo 2.5.2) di google.com. Il server DNS scrive un messaggio DNS reply contenente la corrispondenza tra il nome dell'host e l'indirizzo

IP e lo pone in un segmento UDP; infine pone il segmento all'interno di un datagramma IP indirizzato al laptop di Bob (68.85.2.101). Questo datagramma verrà restituito al router della scuola attraverso la rete Comcast e da qua al laptop di Bob tramite lo switch Ethernet.

17. Il laptop di Bob estrae l'indirizzo IP del server www.google.com dal messaggio DNS. Finalmente, dopo un sacco di lavoro, il laptop di Bob è pronto a contattare il server www.google.com!

Interazione client-server: TCP e HTTP

18. Il laptop di Bob, ora che ha l'indirizzo IP di www.google.com, può creare la socket TCP (Paragrafo 2.7) che verrà usata per inviare un messaggio di HTTP GET (Paragrafo 2.2.3) a www.google.com. Quando Bob crea la socket TCP, per prima cosa il laptop di Bob effettua l'handshake a tre vie (Paragrafo 3.5.6) con TCP su www.google.com. Quindi il laptop di Bob crea un segmento TCP SYN con porta di destinazione 80 (per HTTP), pone il segmento TCP all'interno di un datagramma IP con indirizzo IP di destinazione 64.233.169.105 (www.google.com), pone il datagramma all'interno di un frame con indirizzo MAC di destinazione 00:22:6B:45:1F:1B (il gateway) e invia il frame allo switch.

19. I router nelle reti della scuola, di Comcast e di Google inoltrano il datagramma contenente TCP SYN a www.google.com, usando ognuno la propria tabella di inoltro, come spiegato nei passi da 14 a 16. Si ricordi che le righe delle tabelle di inoltro dei router che governano la procedura di inoltro dei pacchetti sul collegamento inter-dominio tra Comcast e Google sono determinate dal protocollo BGP

20. Infine, il datagramma contenente il TCP SYN arriva a www.google.com. Il messaggio TCP SYN viene estratto dal datagramma e viene effettuato un demultiplexing verso la socket di benvenuto associata alla porta 80. Viene creata una socket di connessione (Paragrafo 2.7) per la connessione TCP tra il server HTTP di Google e il laptop di Bob. Un segmento TCP SYNACK (Paragrafo 3.5.6) viene generato, posto all'interno di un datagramma indirizzato al laptop di Bob e infine inserito in un frame appropriato al collegamento tra www.google.com e il primo router.

21. Il datagramma contenente il segmento TCP SYNACK viene inoltrato attraverso le reti di Google, Comcast e della scuola per arrivare infine alla scheda Ethernet del laptop di Bob. Viene effettuato un demultiplexing del datagramma all'interno del sistema operativo alla socket TCP creata al passo 18, che entra nello stato di connessione.

22. Con la socket sul laptop di Bob finalmente pronta per trasmettere byte a www.google.com, il browser di Bob crea il messaggio HTTP GET (Paragrafo 2.2.3) contenente l'URL da leggere. Quindi il messaggio HTTP GET viene scritto nella socket e diventa il payload di un segmento TCP. Il segmento TCP viene posto in un datagramma, inviato e consegnato a www.google.com come descritto dei passi da 18 a 20.

23. Il server HTTP di www.google.com legge dalla socket TCP il messaggio http GET, crea un messaggio di risposta HTTP (Paragrafo 2.2), pone il contenuto della pagina web richiesta nel corpo del messaggio di risposta HTTP che invia alla socket TCP.

24. Il datagramma contenente il messaggio di risposta HTTP viene inoltrato attraverso le reti di Google, Comcast e della scuola al laptop di Bob. Il browser di Bob legge dalla socket la risposta HTTP, estrae il codice HTML della pagina web dal corpo della risposta HTTP e finalmente (finalmente!) visualizza la pagina web!

Capitolo 6/7.

Reti mobili e wireless.

In una rete wireless possiamo identificare i seguenti elementi:

- **Host wireless:** dispositivi periferici che eseguono applicazioni. L'**host wireless** può essere un portatile, un tablet, un telefono o un computer desktop. Gli host possono essere mobili o meno.

- **Collegamenti wireless:** attraverso cui gli host si connettono alle stazioni base. Vedremo che i collegamenti wireless sono utilizzati per connettere router, commutatori e anche altri dispositivi di rete.

- **Stazione base (base station).** È il **componente chiave dell'infrastruttura** delle **reti wireless**.

Una stazione base è responsabile dell'invio e della ricezione dei dati (pacchetti) tra gli host wireless a essa associati.

Quando diciamo che un host wireless è "associato" a una stazione base, intendiamo che

l'host si trova nell'area di copertura della stazione base e che (2) la utilizza per trasmettere dati verso il resto della rete. Un esempio di queste stazioni base sono i **ripetitori di cella** (cell tower) nelle reti cellulari e gli **access point** (punti di accesso) nelle LAN 802.11.

Gli host associati a una stazione base sono considerati come operanti in **modalità infrastruttura (infrastructure mode)**, in quanto tutti i tradizionali servizi di rete sono forniti dalla rete attraverso la stazione base. Nelle cosiddette **reti ad hoc**, gli host wireless non hanno alcuna infrastruttura cui connettersi. In sua assenza, essi stessi devono provvedere a tutti i servizi di rete.

L'host, quando si sposta dall'area di copertura di una stazione base a un'altra cambia la stazione base cui è associato. Questo processo è chiamato **handoff**.

- **Infrastruttura di rete.** È la rete più ampia alla quale l'host wireless potrebbe volersi connettere.

Formazione diverse tipologie di rete. A livello più alto possiamo classificare le reti wireless secondo due criteri: (i) se un pacchetto nella rete attraversa un solo collegamento wireless (hop) oppure più d'uno; (ii) se vi è un'infrastruttura, come una stazione base, oppure no.

- **Hop singolo, con infrastruttura.** Queste reti hanno una stazione base che si collega a una rete cablata più grande, per esempio Internet. Inoltre, tutte le comunicazioni hanno luogo tra la stazione base e gli host su un singolo hop wirel

ess.

- **Hop singolo, senza infrastruttura.** Non vi è alcuna stazione base collegata alla rete wireless. Uno dei nodi di questa rete può coordinare la trasmissione degli altri nodi. Le reti Bluetooth e 802.11 in modalità ad hoc ricadono in questa categoria.

• **Hop multipli, con infrastruttura.** È presente una stazione base, collegata tramite cavo alla rete più grande. Tuttavia, alcuni nodi potrebbero dover fare affidamento per le loro comunicazioni su altri nodi wireless, per comunicare con la stazione base. Alcune reti di sensori wireless e le **reti mesh wireless** ricadono in questa categoria.

• **Hop multipli, senza infrastruttura.** Non c'è una stazione base e i nodi possono dover ritrasmettere i messaggi a parecchi altri nodi per raggiungere la destinazione. I nodi possono anche essere mobili e la connettività tra loro può cambiare, come in una classe di reti nota come mobile ad hoc network (MANET, rete mobile ad hoc). Se i nodi mobili sono veicoli, la rete è una **vehicular ad hoc network** (VANET, rete veicolare ad hoc).

Collegamenti wireless e caratteristiche di rete.

Rete cablata, come una rete domestica, con uno switch Ethernet che connette alcuni host. Se sostituissimo una Ethernet cablata con una rete 802.11, la scheda wireless sostituirebbe la scheda Ethernet cablata nell'host e l'access point sostituirebbe lo switch Ethernet ma, in linea di principio, non sarebbe necessario alcun altro cambiamento a livello di rete o superiore.

Ciò ci suggerisce di focalizzare l'attenzione sul livello di collegamento per analizzare le importanti differenze tra le reti cablate e wireless. Ecco le principali.

• **Attenuazione del segnale.** Le radiazioni elettromagnetiche si attenuano quando attraversano determinati ostacoli (come le pareti delle case). Anche nello spazio libero l'intensità del segnale si attenua al crescere della distanza percorsa (path loss).

• **Interferenze da parte di altre sorgenti.** Sorgenti radio che trasmettono nella stessa banda di frequenza interferiscono tra loro. Inoltre, il rumore elettromagnetico ambientale può generare interferenze.

• **Propagazione su più cammini (multipath propagation).** La propagazione su più cammini si verifica quando una parte delle onde elettromagnetiche si riflette su oggetti e sul terreno. Questo fenomeno disturba il segnale che giunge al destinatario.

Oggetti in movimento tra il trasmittente e il ricevente possono causare il **fenomeno di propagazione multipla**, che varia in ogni momento.

I protocolli di collegamento wireless (come il protocollo 802.11) utilizzano non solo potenti codici CRC di rilevazione degli errori, ma anche un protocollo di trasferimento affidabile a livello di collegamento, che ritrasmette i pacchetti danneggiati.

Gli host ricevono un segnale elettromagnetico che è la combinazione di una forma degradata del segnale originale trasmesso dal mittente e un rumore di fondo dell'ambiente. Il **rapporto segnale rumore (SNR, signal-to-noise ratio)** è una misura relativa dell'intensità del segnale ricevuto, cioè dell'informazione che è stata trasmessa, e del rumore. L'SNR viene tipicamente misurato in decibel (dB). Per i nostri scopi, ci basti sapere che un SNR più grande facilita il ricevente nel distinguere il segnale trasmesso dal rumore di fondo.

L'elevato tasso di errori nei bit non è l'unica differenza tra i collegamenti wireless e quelli cablati. Ricordiamo che, nel caso dei canali cablati broadcast, ciascun nodo riceve la trasmissione da tutti gli altri nodi. Nei collegamenti wireless, la situazione non è così semplice. Il cosiddetto **problema del terminale nascosto (hidden terminal problem)** nasce dal fatto che ostacoli fisici presenti nell'ambiente (per esempio, una montagna o un palazzo) potrebbero impedire a due stazioni di sentirsi l'un l'altro, anche se le loro trasmissioni interferiscono presso un'altra destinazione. Un secondo scenario nel quale si crea una collisione non rilevabile dalla stazione ricevente è quella in cui si verifica il **fading (evanescenza)** del segnale che si propaga nell'ambiente.

CDMA.

Quando due host condividono un canale di comunicazione è necessario un protocollo che ne regoli l'accesso.

Ci sono tre classi di questi protocolli: a suddivisione del canale, ad accesso casuale e a rotazione.

CDMA (code division multiple access, accesso multiplo a divisione di codice), appartenente alla famiglia di protocolli a suddivisione del canale, è il protocollo di accesso al canale condiviso più diffuso nelle reti wireless e nelle tecnologie cellulari.

Nel protocollo CDMA ogni bit inviato è codificato moltiplicandolo con un segnale (il codice) che cambia con frequenza (detta chipping rate) di molto superiore a quella con cui variano i bit dei dati.

Intanto, affinché i riceventi CDMA possano essere in grado di estrarre un particolare segnale del trasmittente, occorre scegliere accuratamente i codici CDMA.

Wi-Fi: LAN wireless 802.11.

Le **LAN wireless** sono una delle tecnologie più importanti per l'accesso a Internet.

Con il termine WLAN o Wireless LAN, si indica una rete locale che sfrutta tecnologia wireless, invece di una connessione cablata via cavo.

Lo standard **IEEE 802.11 wireless LAN**, conosciuto anche come **Wi-Fi**, risulta essere l'incontestato "vincitore".

Esistono numerosi standard 802.11 ("WiFi"). La Tabella 7.1 ne riassume le principali caratteristiche.

Gli standard 802.11 hanno in comune diverse caratteristiche.

- Utilizzano lo stesso protocollo di accesso al mezzo, **CSMA/CA**, e la stessa struttura del frame a livello di collegamento.
- Tutti possono ridurre la frequenza trasmissiva per raggiungere distanze maggiori.
- Inoltre, gli standard 802.11 hanno **retrocompatibilità**: per esempio un dispositivo mobile abilitato solo 802.11g può interagire con una più recente base station abilitata 802.11ac.

Tuttavia, i tre standard presentano notevoli differenze a livello fisico (Tabella 7.1).

Tabella 7.1 Riepilogo degli standard IEEE 802.11.

Standard	Gamma di frequenze (negli Stati Uniti)	Velocità di trasferimento dati
802.11b	2,4 GHz	Fino a 11 Mbps
802.11a	5 GHz	Fino a 54 Mbps
802.11g	2,4 GHz	Fino a 54 Mbps
802.11n	2,5 GHz e 5 GHz	Fino a 450 Mbps
802.11ac	5 GHz	Fino a 1300 Mbps

La rete **802.11b** opera nelle due bande di frequenze:

- Da 2,4 a 2,485 GHz, utilizzata anche dai telefoni a 2,4 GHz e dai forni a microonde, e
- Da 5,1 a 5,8 GHz, con una portata di trasmissione più breve e danneggiata dalla propagazione su cammini multipli.

I due standard più recenti, **802.11n** e **802.11ac** usano due o più antenne sul lato di trasmissione e due o più antenne sul lato ricevente, che trasmettono/ricevono segnali diversi. (MIMO, multiple-input multiple-output)

Le base station **802.11ac** possono trasmettere a più stazioni simultaneamente e possono usare antenne “intelligenti” con beamforming adattativo in modo da ridurre le interferenze e aumentare le distanze raggiungibili a un certo data rate.

Architettura di 802.11.

I componenti basilari dell’architettura dello standard 802.11.

Il principale blocco costitutivo è il **basic service set** (BSS, insieme di servizi di base), che contiene una o più stazioni wireless e una stazione base centrale, detta **access point** (AP, punto di accesso).

In una tipica rete domestica, si trovano un AP e un router (spesso assemblati nello stesso dispositivo) che **connettono il BSS a Internet**.

- Le stazioni 802.11 hanno indirizzi MAC di 6 byte che sono cablati nel firmware della scheda di rete.
- Ciascun AP ha anche un indirizzo MAC per le interfacce wireless. Questi indirizzi MAC sono gestiti da IEEE e sono univoci.

Le reti wireless che utilizzano AP sono anche chiamate **wireless LAN con infrastruttura**, dove “l’infrastruttura” è formata dagli AP, dalla rete Ethernet che li collega e da un router.

Una **rete ad hoc** potrebbe essere composta da un gruppo di persone riunite in uno stesso luogo (per esempio, in una sala conferenze, in treno o in macchina), che vogliono scambiarsi dati con i loro computer portatili, nonostante l’assenza di un AP.

Canali e associazione.

Prima di inviare o ricevere pacchetti dati 802.11 le stazioni devono associarsi a un AP.

Quando un amministratore di rete installa un AP, gli assegna uno o due identificativi, detti **SSID** (service set identifier).

Per esempio, quando si utilizza la funzionalità “visualizza reti disponibili” su un iPhone, viene mostrata una lista contenente gli SSID degli AP raggiungibili. L’amministratore deve anche assegnare all’AP un numero di canale.

802.11 opera nella banda di frequenza da **2,4 GHz a 2,4835 GHz**. In questi 85 MHz di banda definisce 11 canali parzialmente sovrapposti. Un amministratore potrebbe creare una LAN wireless con una capacità trasmissiva globale di 33 Mbps, installando tre AP 802.11b nello stesso luogo, assegnando i canali 1, 6 e 11 agli AP e connettendoli con uno switch.

Giungla Wi-Fi (Wi-Fi jungle) è un luogo in cui **la stazione wireless riceve un segnale sufficientemente intenso da due o più AP**. Ciascuno di questi AP potrebbe appartenere a una diversa sottorete e dovrebbe avere un canale indipendente.

Per ottenere l’accesso a Internet, la nostra stazione wireless avrà bisogno di associarsi a una specifica sottorete e quindi a un unico AP. Solo l’AP associato potrà inviare pacchetti di dati verso la nostra stazione wireless e questa potrà inviare pacchetti in Internet soltanto attraverso quel dato AP.

Lo standard 802.11 richiede che l’AP invii periodicamente dei **frame beacon**, che contengono il proprio codice SSID e il proprio indirizzo MAC. La nostra stazione wireless, sapendo che l’AP sta inviando i frame beacon, analizza gli 11 canali in cerca di questi frame provenienti dagli AP situati nelle vicinanze (alcuni dei quali potrebbero trasmettere sul medesimo canale) e seleziona quello a cui associarsi.

Non c’è un algoritmo per selezionare l’AP con il quale associarsi, solitamente l’host sceglie l’AP i cui frame beacon vengono ricevuti con la potenza di segnale più alta, anche se questa non è la sola caratteristica dell’AP che determinerà le prestazioni ricevute dall’host.

Il processo di **scansione dei canali e di ascolto** dei frame beacon è chiamato **scansione passiva**.

Un host wireless esegue anche una **scansione attiva**, inviando in broadcast un frame sonda che verrà ricevuto da tutti gli AP nel raggio di copertura dell’host wireless. L’AP risponde al frame sonda di richiesta con un frame di risposta.

L’host wireless **può quindi scegliere l’AP** con il quale associarsi **tra quelli che hanno risposto**.

Dopo aver individuato il punto d’accesso con il quale associarsi, l’host wireless invia un frame di richiesta di associazione all’AP, il quale risponde con un frame di risposta di associazione. Il secondo scambio richiesta/risposta è necessario con la scansione attiva, in quanto un AP che risponde all’iniziale frame sonda di richiesta non sa con quale dei (magari molti) AP che rispondono l’host sceglierà di associarsi.

Una **volta associato con un AP**, l’host vorrà entrare nella sottorete, dell’indirizzamento IP, alla quale appartiene l’AP.

Una **volta ottenuto l’indirizzo**, l’host viene visto come un altro host con un indirizzo IP in quella sottorete.

Per creare un’associazione con un particolare AP, alla stazione wireless potrebbe essere richiesto di autenticarsi, un approccio diffuso consente l’accesso alla rete wireless sulla base dell’indirizzo MAC della stazione. Un secondo approccio utilizza “utente” e password.

Protocollo MAC di 802.11.

Una volta che una stazione wireless è associata a un AP, può iniziare a trasmettere e a ricevere frame dati da e verso l'AP. Ma, poiché stazioni multiple potrebbero voler trasmettere frame di dati contemporaneamente sullo stesso canale, è necessario un protocollo ad accesso multiplo per coordinare le trasmissioni. Ci sono tre classi di protocolli ad accesso multiplo: a suddivisione del canale (compreso CDMA), ad accesso casuale e a rotazione.

Ispirandosi al grande successo di Ethernet, i progettisti di 802.11 scelsero proprio il protocollo ad accesso casuale. Questo protocollo è chiamato **CSMA** con prevenzione di collisioni o più sinteticamente CSMA/CA (CA, collision avoidance).

L'acronimo "CSMA" significa **accesso multiplo con rilevazione della portante**, ovvero ciascuna stazione ascolta il canale prima di trasmettere e si astiene dal farlo se rileva che il canale è occupato. Sebbene sia Ethernet sia 802.11 utilizzino CSMA con accesso casuale, i loro protocolli MAC presentano sostanziali differenze.

Se una stazione che sta trasmettendo rileva la trasmissione di un'altra stazione, la prima interrompe la trasmissione e ritenta dopo un breve intervallo di tempo di durata casuale. Al contrario del protocollo Ethernet 802.3, il protocollo MAC 802.11 non implementa la rilevazione delle collisioni.

Dato che le reti 802.11 non utilizzano la rilevazione delle collisioni, una volta che una stazione inizia a trasmettere un frame lo trasmette interamente: cioè, non può tornare indietro. Per ridurre il rischio di collisioni, 802.11 implementa numerose tecniche di prevenzione.

- **Schema di avvenuta ricezione:** Un frame inviato da una stazione in una rete wireless potrebbe non raggiungere la stazione di destinazione per numerosi motivi. Perciò MAC 802.11 utilizza la **conferma di avvenuta ricezione a livello di collegamento**. Quando la stazione di destinazione riceve un frame che passa il controllo CRC, attende per un breve periodo di tempo, noto come **SIFS (short inter-frame space**, spazio breve inter-frame), dopo il quale invia al mittente un frame di conferma di avvenuta ricezione. Se la stazione trasmittente non riceverà questo riscontro entro un arco di tempo stabilito, presupporrà un errore e ritrasmetterà il frame, utilizzando ancora il protocollo CSMA/CA per accedere al canale. Se il frame di conferma non sarà ricevuto dopo un numero prefissato di ritrasmissioni, la stazione trasmittente passerà oltre e scarterà il frame.

Il protocollo CSMA/CA.

Supponiamo che una stazione (una stazione wireless o un AP) abbia un frame da trasmettere.

1. Se inizialmente la stazione percepisce il canale come inattivo, trasmette il suo frame dopo un breve periodo di tempo (**DIFS**).
2. Altrimenti, la stazione sceglie un valore casuale di ritardo usando una attesa binaria esponenziale e decrementa questo valore dopo DIFS solo quando il canale viene percepito come inattivo. Se il canale è percepito come occupato, il contatore rimane fermo.
3. Quando il contatore arriva a zero (canale inattivo), la stazione trasmette l'intero frame e aspetta il frame di conferma.
4. Se riceve la conferma, la stazione sa che il frame è stato ricevuto correttamente e, qualora avesse un altro frame da inviare, riattiva il protocollo CSMA/CA dal passo 2.

Se il frame di conferma non viene ricevuto, la stazione trasmittente ritorna al passo 2, ma con un valore di ritardo maggiore.

Nel protocollo Ethernet di accesso, una stazione inizia a trasmettere appena percepisce che il canale è inattivo. Con CSMA/CA, invece, la stazione evita di trasmettere mentre decrementa il contatore, anche se ha rilevato che il canale è inattivo.

Terminali nascosti: RTS e CTS.

Il protocollo MAC 802.11 include anche un elegante (opzionale) *schema di prenotazione* che aiuta a evitare collisioni anche in presenza di terminali nascosti.

Consideriamo due stazioni wireless e un AP.

Le due stazioni sono collocate nel raggio dell'AP al quale sono associate. Ciascuna delle stazioni wireless è nascosta all'altra, sebbene nessuna sia nascosta all'AP.

I terminali nascosti possono essere problematici perché si potrebbero causare collisioni e il canale risulterà pertanto sprecato.

Per evitare questo problema, il protocollo IEEE 802.11 prevede **due frame di controllo**:

- **RTS** (request to send, richiesta di invio); e
- **CTS** (clear to send, abilitazione a trasmettere) per riservare l'accesso al canale.

Il trasmittente, quando vuole inviare il frame DATI, invia innanzitutto il frame RTS all'AP, indicando il tempo totale richiesto per la trasmissione del frame DATI e del frame di conferma ACK. L'AP, quando riceve il frame RTS, risponde diffondendo in broadcast il frame CTS. Questo ha due scopi: comunica al trasmittente il permesso esplicito di inviare e comunica alle altre stazioni di non trasmettere durante il periodo di tempo riservato.

L'utilizzo dei frame RTS e CTS può incrementare le prestazioni per due motivi.

- Risolve il problema del terminale nascosto, in quanto il frame DATI viene trasmesso solamente dopo che il canale è stato prenotato.
- Dato che i frame RTS e CTS sono piccoli, una collisione che li coinvolgesse sarebbe di breve durata. Una volta che questi sono stati trasmessi con successo, i successivi frame DATI e ACK dovrebbero essere trasmessi senza collisioni.

Pacchetto IEEE 802.11.

I frame 802.11 sono molto simili ai frame Ethernet, ma contengono campi specifici per l'utilizzo nei collegamenti wireless. I numeri sopra ciascun campo ne rappresentano la lunghezza in byte; quelli sopra i sottocampi del campo di controllo rappresentano la lunghezza in bit.



Dettaglio del campo di controllo: (I numeri indicano la lunghezza del campo in bit)

2	2	4	1	1	1	1	1	1	1	1	1
Versione del protocollo	Tipo	Sottotipo	Verso AP	Da AP	Frammentazione	Copia	Alimentazione	Altri dati	WEP	Riservato	

Campi Payload e CRC.

Il cuore del frame è il campo **payload**, che solitamente consiste di un datagramma IP o di un pacchetto ARP.

I frame 802.11 hanno un campo CRC di 32 bit che permette al ricevente il rilevamento degli errori nei bit. Questi errori sono molto più frequenti nelle reti wireless che nelle LAN cablate; quindi, il campo CRC è più utile qui.

Campi Indirizzo.

Il frame 802.11 contiene quattro campi indirizzi MAC di 6 byte.

- **L'indirizzo 2** è l'indirizzo MAC della stazione che **trasmette il frame**. Quindi, se una stazione wireless trasmette il frame, viene inserito in questo campo il suo indirizzo MAC. Se invece è l'AP a trasmettere, in questo campo si inserirà l'indirizzo MAC dell'AP.

- **L'indirizzo 1** è l'indirizzo MAC della stazione wireless che **deve ricevere il frame**.

Quindi, se è una stazione wireless a trasmettere il frame, questo campo conterrà l'indirizzo MAC dell'AP di destinazione. In modo analogo, se è l'AP che trasmette il frame, questo campo conterrà l'indirizzo MAC della stazione wireless di destinazione.

- Per comprendere il **campo indirizzo 3**, ricordiamo che il BSS fa parte di una sottorete e che questa connette le altre sottoreti attraverso una interfaccia del router. L'indirizzo 3 contiene l'indirizzo MAC di **questa interfaccia del router**.

In sintesi, l'indirizzo 3 gioca un ruolo cruciale per interconnettere il BSS con una rete cablata.

Campi numero di sequenza: permette al ricevente di distinguere tra un frame appena trasmesso e la ritrasmissione di un frame.

Campo durata: usato dalla stazione trasmittente per riservare il canale per un periodo di tempo che include la trasmissione del suo frame dati e quella del frame di consegna.

Campo controllo del frame: contiene diversi sottocampi:

- o **Tipo:** può essere data, RTS frame, CTS frame, ACK
- o **Protocol:** specifica la versione di 802.11 utilizzata
- o **Verso AP e da Ap:** definiscono la funzione dei diversi campi indirizzo.

Mobilità all'interno di una sottorete IP.

Per aumentare la copertura di una LAN wireless, aziende e università dispongono spesso **vari BSS all'interno di una stessa sottorete IP**.

La mobilità può essere trattata in modo semplice quando i BSS fanno parte di una stessa sottorete.

Quando una stazione si muove da una sottorete a un'altra, è necessario un protocollo più sofisticato.

Se il dispositivo che connette due BSS non è un router, tutte le stazioni all'interno di queste, inclusi AP, apparterranno alla stessa sottorete IP.

Se il dispositivo d'interconnessione fosse stato un router, allora si sarebbe dovuto cambiare l'indirizzo IP e chiudere tutte le connessioni TCP, oppure usare un protocollo di mobilità a livello di rete come, per esempio, IP mobile.

Come fa uno switch a sapere che l'host si è spostato da un AP a un altro?

Gli switch auto-apprendono, costruendo automaticamente le proprie tabelle d'inoltro. Gli switch, però, non sono stati progettati per supportare utenti che si spostano di frequente. Una soluzione potrebbe essere che, appena realizzata la nuova associazione, AP invii in broadcast allo switch un frame Ethernet, con l'indirizzo sorgente di H. Ricevuto il frame, lo switch aggiornerà la sua tabella d'inoltro.

Funzionalità avanzate di 802.11.

Adattamento del tasso in 802.11.

Esistono tecniche diverse di **modulazione** (ognuna con i relativi tassi trasmissivi) che si associano a scenari di SNR distinti.

Alcune implementazioni di 802.11 sono in grado di variare il tasso trasmissivo selezionando in modo adattativo la tecnica di modulazione da usare a livello fisico in base alle attuali o recenti caratteristiche del canale.

- Se un nodo invia due frame consecutivi senza ricevere un riscontro (indicazione implicita di errori sui bit nel canale), il tasso trasmissivo ricade al successivo valore più basso.
- Se per 10 frame consecutivi viene confermata la ricezione o scade il timer che tiene traccia del tempo dall'ultima sostituzione del tasso, il tasso trasmissivo si incrementa al successivo valore più alto.

Questo meccanismo di adattamento del tasso condivide la stessa filosofia del "probing" con il meccanismo di controllo della congestione di TCP: quando le condizioni sono buone (ne è prova la ricezione degli ACK), il tasso trasmissivo viene incrementato fino a che si verifica un evento "brutto" (la mancata ricezione degli ACK), dopo di che il tasso trasmissivo viene ridotto.

Gestione dell'energia.

Lo standard 802.11 fornisce funzionalità di gestione dell'energia che consentono ai **nodi 802.11** di **minimizzare la quantità di tempo nel quale le loro funzionalità di ascolto, trasmissione, ricezione nonché altri circuiti elettrici debbano rimanere attivi**. Un nodo è in grado di alternare esplicitamente tra gli stati di "attivo" (wake) e "inattivo" (sleep); un nodo indica all'access point che sta per disattivarsi impostando a 1 il bit di gestione energetica nell'intestazione di un frame 802.11. Il nodo imposta un timer perché torni ad attivarsi appena prima dell'istante in cui l'AP è programmato per mandare il frame beacon (ricordiamo che un AP tipicamente invia un frame beacon ogni 100 millisecondi). Dato che l'AP è a conoscenza del fatto che il nodo sta per disattivarsi, grazie all'impostazione del bit di gestione energetica, l'AP sa che non dovrà inviare frame al nodo e memorizza i frame destinati all'host disattivo per una trasmissione successiva.

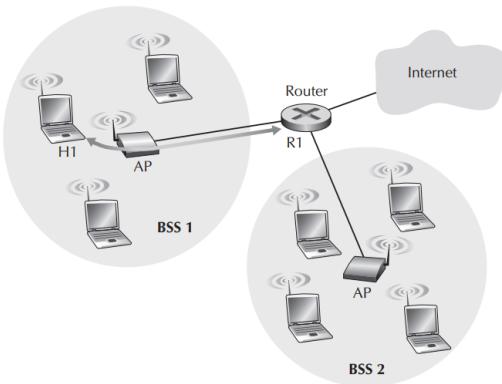
Un nodo si attiva immediatamente prima che l'AP invii un frame beacon e in breve tempo risulta completamente attivo; il processo richiede solo 250 microsecondi.

I frame beacon, inviati dall'AP, contengono una lista dei nodi i cui frame sono stati memorizzati all'AP. Se non vi sono frame memorizzati per quel nodo, quest'ultimo torna a disattivarsi. Altrimenti, il nodo può richiedere esplicitamente che i frame memorizzati siano inviati, mandando

un messaggio di interrogazione all'AP. Un nodo che non ha frame da inviare o da ricevere può essere disattivo il 99 per cento del tempo, con un conseguente significativo risparmio di energia.

Indirizzamento 802.11.

Supponiamo si trasportare un datagramma dall'interfaccia R1 del router alla stazione wireless H1.



Il router non è consapevole della presenza di un AP tra sé e H1.

1. Il router utilizza ARP per determinare l'indirizzo MAC di H1 (dato che conosce il suo IP).
2. Dopo aver ottenuto l'indirizzo MAC di H1, l'interfaccia del router incapsula il datagramma in un frame Ethernet.
3. Quando il frame Ethernet giunge all'AP, questo lo converte in un frame 802.11. L'AP riempie i campi indirizzo 1 e 2 con l'indirizzo MAC di H1 e il proprio indirizzo MAC. Per l'indirizzo 3, AP inserisce l'indirizzo MAC di R1.

Quando l'host H1 risponde a R1:

1. H1 crea un frame 802.11, riempiendo i campi indirizzo 1 e 2 con l'indirizzo MAC dell'AP e l'indirizzo MAC di H1. Per l'indirizzo 3, H1 inserisce l'indirizzo MAC di R1.
2. Quando l'AP riceve il frame 802.11, lo converte in frame Ethernet. Il campo dell'indirizzo sorgente per questo pacchetto è l'indirizzo MAC di H1, mentre il campo di indirizzo di destinazione è l'indirizzo MAC di R1.

Personal Area Network: Bluetooth e Zigbee.

Lo standard IEEE 802.11 Wi-Fi è rivolto alla comunicazione tra dispositivi che distano fino a 100 metri, eccetto quando 802.11 viene usato in configurazione punto a punto con un'antenna direzionale. Esistono due altri protocolli IEEE 802, Bluetooth e Zigbee.

Bluetooth.

Una rete 802.15.1 opera su un raggio limitato, a bassa potenza e a basso costo.

802.11 prevede maggior potenza, copertura media e alta velocità di accesso.

Le reti 802.15.1 sono qualche volta chiamate **WPAN** (*wireless personal area network, reti wireless personali*).

La rete 802.15.1 opera nella banda a 2,4 GHz in modalità TDM, con slot di tempo di 625 microsecondi. In ogni slot, si può trasmettere in uno tra 79 canali, cambiando il canale in modo pseudo-casuale da uno slot all'altro. Questa strategia, conosciuta come **FHSS** (frequency-hopping spread spectrum, spettro distribuito a frequenza variabile) ripartisce nel tempo le trasmissioni sullo spettro di frequenza. Queste reti possono raggiungere un tasso trasmissivo di 4 Mbps.

Le reti 802.15.1 sono **reti ad hoc**, nel senso che **non è necessaria alcuna infrastruttura** (cioè un AP) per interconnettere i dispositivi 802.15.1.

Accesso cellulare a Internet.

Un **host può accedere a Internet** quando si trova **all'interno di un hotspot Wi-Fi**, cioè quando è nelle vicinanze di un AP 802.11.

Molti hotspot Wi-Fi dispongono però di un raggio di copertura limitato, tra i 10 e i 100 metri di diametro.

Dato che il telefono cellulare è ormai diffusissimo, si potrebbe pensare di estendere queste reti in modo da consentire loro di supportare non soltanto la voce, ma anche gli accessi wireless a Internet.

Panoramica dell'architettura di una rete cellulare.

Spesso le tecnologie cellulari sono classificate in "generazioni".

- I sistemi di prima generazione, **1G**, erano analoghi ai sistemi FDMA progettati solamente per le comunicazioni audio.
- Questi sistemi (estinti) sono stati sostituiti dai sistemi **2G digitali**. Anche questi di seconda generazione sono progettati per la telefonia.

Global system for mobile communication: standard di seconda generazione di telefonia mobile. (GSM)

- Ma più tardi vennero estesi, **2.5G**, per supportare il traffico dati, per esempio Internet, oltre al servizio di telefonia.
- I sistemi **3G** attualmente installati supportano sia la voce sia i dati, con una sempre maggiore enfasi sulla capacità di trasporto dati e collegamenti di accesso radio sempre più veloci.
- Gli attuali sistemi **4G** sono basati sulla tecnologia LTE, nucleo di rete completamente basato su IP e forniscono voce e dati integrati a velocità di Megabit.

Architettura della rete cellulare 2G: connessioni voce alla rete telefonica.

Il termine cellulare si riferisce al fatto che un'area geografica è suddivisa in aree di copertura dette celle.

Ogni cella contiene una stazione base che prende il nome di **BTS** (*base transceiver station*) che **scambia segnali con le stazioni mobili della cella**. L'**area di copertura** di una cella **dipende da molti fattori**, inclusi la potenza di trasmissione del BTS e quella della stazione mobile, la presenza di

palazzi nella cella, e l'altezza dell'antenna della stazione base. Molti sistemi preferiscono installarla agli angoli dove tre celle si intersecano, in modo che una sola stazione dotata di antenne direzionali possa servire tre celle.

Lo standard GSM per i sistemi cellulari 2G usa una combinazione di FDM e TDM per l'interfaccia aerea.

Con FDM puro il canale è suddiviso in bande di frequenza, ciascuna dedicata a una chiamata.

Invece TDM suddivide il tempo in frame ulteriormente ripartiti in slot e a ogni chiamata deve essere assegnato un particolare slot di un frame. Nei sistemi misti FDM/TDM, il canale è ripartito in sottobande di frequenza all'interno delle quali il tempo è suddiviso in frame e slot. Quindi, in questi sistemi, se il canale è suddiviso in F sottobande e il tempo è suddiviso in T slot, i canali potranno supportare $F \times T$ chiamate contemporanee.

Le reti di accesso HFC usano un approccio FDM/TDM. GSM consiste di bande di frequenza di 200 kHz, ciascuna delle quali supporta 8 chiamate TDM. GSM codifica le chiamate a 13 e a 12,2 kbps.

Un **base station controller (BSC)** nelle reti GSM serve tipicamente alcune decine di BTS.

Il ruolo del BSC è di allocare canali radio BTS agli utenti mobili, eseguendo la procedura di **paging**, che consiste nel trovare la cella in cui l'utente mobile risiede, ed eseguire l'handoff dell'utente mobile. Il BSC insieme alle BTS che controlla costituiscono il **base station system (BSS)** di una rete GSM.

Il **mobile switching center (MSC)** gioca il ruolo centrale nell'autorizzazione e identificazione degli utenti (per esempio determinando se un dispositivo mobile è autorizzato a connettersi alla rete cellulare), nello stabilire o terminare una chiamata e nell'handoff.

Rete dati cellulare 3G: accesso Internet agli utenti delle reti cellulari.

Mentre ci muoviamo vorremmo anche leggere le e-mail, accedere al Web. Per fare questo il nostro smartphone deve eseguire l'intero stack di protocolli TCP/IP (compresi i livelli fisico, di collegamento, di rete, di trasporto e di applicazione) e connettersi a Internet attraverso la rete dati cellulare.

3G core network.

Il nucleo (core) della rete dati cellulare 3G connette reti di accesso radio a Internet. La core network interopera con le componenti della rete cellulare esistente per la voce (rete voce cellulare), in particolare l'MSC.

Nella core network 3G ci sono due tipi di nodi: **Serving GPRS Support Nodes (SGSN)** e **Gateway GPRS Support Nodes (GGSN)**.

GPRS è l'acronimo di un servizio dati cellulari di 2G: **Generalized Racket Radio Service**.

Un **SGSN** ha il compito di consegnare datagrammi a/da nodi mobili nella rete di accesso radio a cui l'SGSN è collegato. L'SGSN interagisce con l'MSC della rete voce cellulare di quell'area, fornendo agli utenti i servizi di autorizzazione e handoff, memorizzando le informazioni di posizione (cella) dei nodi mobili attivi e inoltrando i datagrammi tra i nodi mobili in una rete di accesso radio e un GGSN, che agisce come gateway connettendo più SGSN a Internet. Quindi il GGSN è l'ultima parte dell'infrastruttura 3G.

Rete di accesso radio 3G: il confine wireless.

La rete di accesso radio 3G è la rete wireless di primo hop vista da un utente 3G.

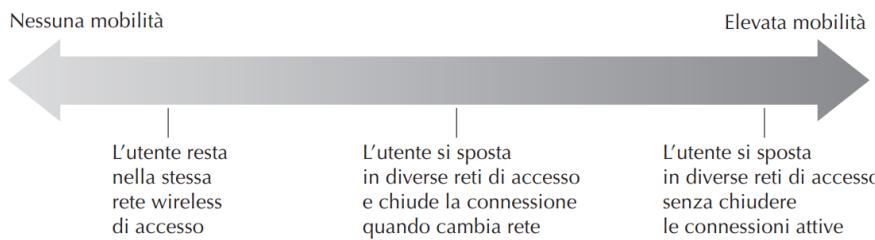
Il **Radio Network Controller (RNC)** controlla tipicamente alcune stazioni base simili ai BTS incontrati nei sistemi 2G che nel gergo di UMTS vengono chiamati "Node B". Ogni collegamento wireless all'interno di una cella opera tra i nodi mobili e una BTS, come nelle reti 2G.

L'RNC è connesso sia alla rete voce cellulare a commutazione di circuito tramite un MSC sia a Internet a commutazione di pacchetto tramite un SGSN. Quindi, sebbene i servizi voce e i servizi dati 3G usino nuclei di reti differenti, hanno in comune la rete di accesso radio di primo e ultimo hop. Un cambiamento significativo di UMTS rispetto alle reti 2G è quello di usare una tecnica CDMA nota come Direct Sequence Wideband CDMA (DS-WCDMA) all'interno degli slot TDMA al posto dello schema GSM FDM/TDM; gli slot TDMA sono disponibili su più frequenze.

Questo cambiamento richiede una nuova rete di accesso cellulare 3G che operi in parallelo con la rete radio BSS 2G. Il servizio dati associato alle specifiche WCDMA è noto come HSPA (high speed packet access) e prospetta velocità di downlink per i dati fino a 14 Mbps.

Gestione della mobilità.

Un **nodo mobile** è un nodo che cambia nel tempo il suo punto di connessione con la rete.



• Che cos'è un utente mobile dal punto di vista del livello di rete?

- Se un utente vuole trasferire un computer portatile con una scheda di rete wireless all'interno di un edificio non è un utente mobile dal punto di vista del livello di rete. Inoltre, se l'utente si associa allo stesso AP indipendentemente dalla posizione, l'utente non è mobile neppure per il livello di collegamento.
- Se un utente passa attraverso diverse reti di accesso wireless e volendo mantenere ininterrotta la sua connessione TCP a un'applicazione remota questo utente è certamente mobile.
- L'utente che sposta un computer portatile da una posizione (un ufficio o una camera d'albergo) a un'altra (una caffetteria o in classe) e che vuole riconnettersi alla rete nella nuova posizione. Anche questo utente è mobile, sebbene meno del precedente, ma non pretende di mantenere attive le connessioni TCP mentre si sposta tra i punti di connessione alla rete.

- Quanto è importante che l'indirizzo del nodo mobile resti invariato?

Se l'entità mobile è in grado di mantenere il proprio indirizzo IP quando si sposta, la mobilità diventa trasparente dal punto di vista dell'applicazione. Questa trasparenza è molto importante: un'applicazione non si deve preoccupare di un potenziale cambio d'indirizzo IP. IP mobile fornisce questa trasparenza, permettendo a un nodo mobile di conservare il proprio indirizzo mentre si sposta da una rete all'altra.

- Qual è l'infrastruttura cablata disponibile? Negli scenari precedenti abbiamo implicitamente assunto un'infrastruttura fissa alla quale gli utenti mobili si connettono. Che cosa accadrebbe se l'infrastruttura non esistesse?

Le reti ad hoc forniscono precisamente la funzionalità di stabilire una connessione in assenza di un'altra infrastruttura di rete.

Si tratta di un campo in rapida espansione, al centro delle ricerche sulle reti mobili che esula dagli obiettivi di questo libro.

In una rete, il luogo permanente in cui risiede un nodo mobile (come un portatile o un palmare) è detto **rete di appartenenza** (*home network*) e le entità che gestiscono la mobilità all'interno di questa sono conosciute come **agenti domestici** (*home agent*).

La rete in cui il nodo mobile viene a trovarsi occasionalmente costituisce la **rete ospitante** (*foreign network* o *visited network*) mentre l'entità al suo interno che si occupa della mobilità è detta **agente ospitante** (*foreign agent*).

Indirizzamento.

Per rendere l'utente mobile trasparente alle applicazioni di rete, un **nodo mobile deve conservare il proprio indirizzo mentre si sposta da una rete a un'altra**. Quando un nodo è ospitato in una rete, tutto il traffico diretto all'indirizzo permanente del nodo deve essere instradato verso quella rete. Per fare questo ci sono varie possibilità:

1. La rete ospitante avverte tutte le altre che il nodo mobile si trova attualmente al suo interno.
2. Oppure, la rete potrebbe semplicemente avvertire i propri vicini che possiede percorsi specifici per l'indirizzo permanente del nodo mobile. A loro volta i vicini dovranno propagare queste informazioni attraverso la rete. Quando il nodo mobile esce da una rete e si connette a un'altra, la nuova rete ospitante dovrà comunicare una nuova specifica rotta per questo nodo, mentre la vecchia dovrà cancellare le sue informazioni di instradamento.
3. Un approccio alternativo è di portare le funzionalità di mobilità dal nucleo della rete alla sua periferia. È utilizzata la rete di appartenenza del nodo mobile. Gli agenti nella rete di appartenenza del nodo mobile possono monitorare la rete nella quale il nodo mobile si trova a risiedere. È certamente necessario un protocollo tra il nodo mobile (o l'agente ospitante che lo rappresenta) e l'agente domestico per aggiornare la localizzazione del nodo stesso.

Consideriamo ora l'**agente ospitante**. Un compito di questo agente è:

- la definizione di un **indirizzo di mediazione**, detto **COA** (*care-of address*) per il nodo mobile. La parte di rete del COA sarà quella della rete ospitante. Gli indirizzi associati alla modalità mobile sono quindi due: quello permanente e il COA.
- Il secondo ruolo dell'agente ospitante è di informare l'agente domestico che il nodo mobile risiede nella sua rete e di essere a conoscenza del suo COA.

Il COA sarà utilizzato per "reinstradare" il datagramma al nodo mobile attraverso il suo agente ospitante.

Il nodo mobile può anche assumere le responsabilità dell'agente ospitante. Per esempio, il nodo mobile potrebbe ottenere il COA nella rete ospitante (utilizzando un protocollo come DHCP) e comunicare all'agente domestico il proprio COA.

Instradamento verso il nodo mobile.

Dato che soltanto l'agente domestico (e non il router dell'intera rete) conosce la localizzazione di questo nodo, non sarà sufficiente indirizzare il datagramma all'indirizzo permanente del nodo e inviarlo nell'infrastruttura al livello di rete. Bisogna fare qualcosa di più. In questo caso, esistono due approcci: **l'instradamento indiretto e quello diretto**.

Instradamento indiretto verso il nodo mobile.

- Consideriamo, innanzitutto, il caso del corrispondente che voglia inviare il datagramma a un nodo mobile. Nell'approccio dell'**instradamento indiretto**, il corrispondente non fa altro che indirizzare il datagramma all'indirizzo permanente del nodo e inviarlo nella rete, inconsapevole dell'effettiva localizzazione del nodo. La mobilità, quindi, è completamente trasparente al corrispondente.
- L'**agente domestico** oltre alla responsabilità di interagire con l'agente ospitante per monitorare il COA del nodo mobile, ha un'altra funzione molto importante: **il controllo dei datagrammi in entrata indirizzati ai nodi che fanno parte della rete di appartenenza**, ma che si trovano al momento in una rete esterna. L'agente intercetta questi datagrammi e li invia al nodo mobile con un processo articolato in due passi. Il datagramma è innanzitutto inviato all'agente ospitante del nodo mobile, utilizzando il COA di quest'ultimo e poi viene inoltrato al nodo mobile stesso.

Consideriamo ora un nodo mobile che invia un datagramma al corrispondente. Il nodo può indirizzare il datagramma direttamente al destinatario utilizzando il proprio indirizzo permanente come indirizzo sorgente, e l'indirizzo del corrispondente come indirizzo di destinazione. Conoscendo l'indirizzo del corrispondente, non è necessario instradare il datagramma attraverso l'agente domestico.

Riassumiamo la nostra discussione sull'instradamento indiretto elencando le funzionalità richieste a livello di rete per supportare la mobilità.

- **Protocollo da nodo mobile ad agente ospitante**. Il nodo mobile si registra presso l'agente ospitante quando si collega alla rete esterna e si cancella quando l'abbandona.

- **Protocollo di registrazione da agente ospitante ad agente domestico**. L'agente ospitante registra il COA del nodo presso l'agente domestico.

• **Protocollo dell'agente domestico per l'incapsulamento del datagramma.** Riguarda l'incapsulamento e l'invio del datagramma originale del corrispondente all'interno del datagramma indirizzato al COA.

• **Protocollo di decapsulamento del datagramma dell'agente ospitante.** Riguarda l'estrazione del datagramma originale del corrispondente dal datagramma encapsulato e suo invio al nodo mobile. La precedente discussione affronta tutti i componenti (agenti ospitanti e domestici e instradamento indiretto) necessari a un nodo mobile per mantenere attive le connessioni mentre si sposta tra reti diverse.

Lo standard IP mobile utilizza l'approccio dell'instradamento indiretto.

Instradamento diretto verso un nodo mobile.

Esiste problema dell'**intradamento triangolare** (*triangle routing problem*), dove i datagrammi indirizzati al nodo mobile devono prima essere instradati all'agente domestico e poi alla rete ospitante, anche in presenza di percorsi più efficienti tra il corrispondente e il nodo mobile.

L'instradamento **diretto** supera l'inefficienza insita nell'instradamento triangolare, ma al costo di una maggiore complessità.

Un **agente corrispondente**, nella rete del corrispondente, ottiene innanzitutto il COA del nodo mobile (attraverso una richiesta all'agente domestico), poi, invia tramite un tunnel i datagrammi direttamente al COA del nodo mobile, uguale a come faceva l'agente domestico.

L'instradamento diretto risolve il problema dell'instradamento triangolare introduce due problemi.

- La necessità di un protocollo di localizzazione dell'utente mobile tramite il quale l'agente corrispondente possa interrogare l'agente domestico per ottenere il COA del nodo mobile.

- Quando il nodo mobile si sposta da una rete a un'altra, come sono inviati i dati alla nuova rete?

Nel caso dell'instradamento indiretto era sufficiente aggiornare il COA presso l'agente domestico.

Con l'instradamento diretto, l'agente domestico è interrogato dall'agente corrispondente solo una volta, all'inizio della sessione.

Aggiornare il COA nell'agente domestico NON risolverà il problema dell'instradamento dei dati verso la nuova rete del nodo mobile.

Una soluzione potrebbe essere un nuovo protocollo per avvisare il corrispondente del cambio di COA.

Un'altra soluzione (quella delle reti GSM) opera nel modo seguente:

supponiamo che, in una determinata sessione, siano inviati dati a un nodo mobile nella rete ospitante, nella quale identifichiamo l'agente in quella rete come **agente di appoggio** (*anchor foreign agent*).

Quando il nodo si sposta in una nuova rete, registra il nuovo COA presso il nuovo agente ospitante, che fornirà all'agente di appoggio il nuovo COA del nodo mobile. Quando l'agente di appoggio riceverà il datagramma encapsulato per il nodo non più presente, lo reincapsulerà e lo invierà a quel nodo utilizzando il nuovo COA. Se in futuro il nodo si sposta in un'altra rete, l'agente della nuova rete ospitante dovrà contattare l'agente di appoggio per impostare l'invio dei datagrammi verso quella nuova rete.

IP mobile.

L'architettura e i protocolli di Internet per la mobilità, conosciuti come **IP mobile**, sono definiti per IPv4 nell'RFC 5944.

IP mobile è uno **standard flessibile**, che supporta diverse modalità operative (con o senza agenti ospitanti), vari modi con cui gli agenti e i nodi mobili si rilevano l'un l'altro, l'utilizzo di COA singoli o multipli e varie forme d'incapsulamento.

L'attuale standard, che specifica l'utilizzo dell'instradamento indiretto, consiste di tre parti principali.

- **Ricerca dell'agente.** Sono definiti i protocolli utilizzati dagli agenti per informare i nodi mobili dei propri servizi e dai nodi mobili per richiederli.

- **Registrazione presso l'agente domestico.** IP mobile definisce i protocolli utilizzati dal nodo mobile e/o dall'agente ospitante per registrare e cancellare i COA presso l'agente domestico del nodo.

- **Intradamento indiretto dei datagrammi.** Lo standard definisce anche il modo in cui i datagrammi sono inviati al nodo mobile dall'agente domestico, incluse le regole per l'invio dei datagrammi, la gestione degli errori e varie forme d'incapsulamento.

Nello standard IP mobile i problemi della sicurezza sono posti in primo piano. È necessaria l'autenticazione dei nodi mobili per evitare che utenti malintenzionati registrino indirizzi contraffatti, che portano all'invio di datagrammi a un indirizzo IP per arrivare a un malintenzionato.

Ricerca dell'agente.

Il nodo IP mobile che giunge in una nuova rete, per connettersi a una rete ospitante o per ritornare alla sua rete di appartenenza deve ricercare un nuovo agente, con un nuovo indirizzo di rete, che consenta di accorgersi dello spostamento in una nuova rete.

Questo processo è conosciuto come **ricerca dell'agente** (agent discovery) e può essere realizzato tramite un **avviso** (dell'agente da scoprire) o una **richiesta** (all'agente interessato).

Con l'**avviso dell'agente** (agent advertisement) un agente rende noti i suoi servizi utilizzando un'estensione del protocollo esistente di ricerca di un router.

L'agente invia periodicamente, in broadcast, un messaggio ICMP che contiene 9 nel campo tipo (ricerca router) su tutti i collegamenti cui è connesso. Il messaggio di ricerca del router contiene l'indirizzo IP del router (che è l'agente), in modo da permettere ai nodi mobili di conoscere l'indirizzo IP dell'agente. Questo messaggio contiene anche un'estensione dell'avviso dell'agente mobile che contiene informazioni addizionali necessarie al nodo mobile. I campi più importanti nell'estensione sono:

- **Bit agente domestico (H).** Indica che si tratta di un agente domestico per la rete nella quale risiede.
- **Bit agente ospitante (F).** Indica che si tratta di un agente ospitante per la rete nella quale risiede.
- **Bit di richiesta di registrazione (R).** Indica che l'utente mobile in questa rete deve registrarsi presso l'agente ospitante. L'utente mobile non può ottenere l'indirizzo COA nella rete e usufruire delle funzionalità dell'agente per sé stesso senza registrarsi con l'agente ospitante.
- **Bit d'incapsulamento (M e G).** Indicano se sarà utilizzata un'altra forma d'incapsulamento oltre a quello di IP in IP.
- **Campo COA.** Lista di uno o più indirizzi fornita dall'agente ospitante. Il COA sarà associato all'agente ospitante, che riceverà i datagrammi inviati al COA e li invierà al nodo mobile appropriato. L'utente mobile selezionerà uno di questi indirizzi come suo COA quando si registrerà presso il suo agente domestico.

Con la **richiesta dell'agente** (agent solicitation), il nodo mobile che vuole informazioni riguardo agli agenti, senza aspettare di ricevere un avviso dell'agente, può inviare in broadcast un messaggio di richiesta dell'agente, che è semplicemente un messaggio ICMP che contiene 10 nel campo tipo. L'agente che riceve questa richiesta invierà in unicast un avviso dell'agente direttamente al nodo mobile, che potrà quindi procedere come se avesse ricevuto un avviso non richiesto.

Registrazione presso l'agente domestico.

Una volta che il nodo IP mobile ha ricevuto il COA, l'indirizzo deve essere comunicato all'agente domestico.

Ciò può avvenire tramite l'agente ospitante (che poi registra il COA presso l'agente domestico) o direttamente dal nodo stesso.

Il primo caso prevede quattro fasi.

1. In seguito alla ricezione dell'avviso dell'agente ospitante, il nodo mobile gli invia il messaggio di registrazione IP mobile.

Il messaggio di registrazione, trasportato in un datagramma UDP e inviato alla porta 434, contiene la comunicazione del COA da parte dell'agente ospitante, l'indirizzo dell'agente domestico (HA), l'indirizzo permanente del nodo mobile (MA), il tempo di scadenza della registrazione richiesto e l'identificazione di registrazione a 64 bit. Il tempo di scadenza è il numero di secondi per i quali la registrazione risulta essere valida. Se la registrazione non viene rinnovata entro il tempo specificato, perde validità.

2. L'agente ospitante riceve il messaggio di registrazione e memorizza l'indirizzo IP permanente del nodo mobile. Ora sa che deve prestare attenzione ai datagrammi d'incapsulamento il cui indirizzo di destinazione corrisponde a quello permanente del nodo mobile. L'agente ospitante, quindi, invia un messaggio di registrazione IP mobile (ancora, in un datagramma UDP) alla porta 434 dell'agente domestico.

3. L'agente domestico riceve la richiesta di registrazione e ne controlla autenticità e correttezza. Successivamente, associa l'indirizzo IP permanente del nodo mobile al COA; d'ora in poi, i datagrammi indirizzati al nodo mobile che arriveranno all'agente domestico saranno incapsulati e inviati al COA. L'agente domestico invia una risposta di registrazione IP mobile contenente HA, MA, la scadenza corrente e l'identificazione di registrazione della richiesta.

4. L'agente ospitante riceve la risposta di registrazione e la invia al nodo mobile.

Un agente ospitante non ha bisogno di un'esplicita cancellazione del COA quando un nodo lascia la sua rete. Ciò avverrà automaticamente, quando il nodo mobile si sposterà in una nuova rete (in un'altra rete ospitante o nella sua rete di appartenenza) e registrerà il nuovo COA.

Gestione della mobilità nelle reti cellulari.

Reti telefoniche cellulari.

Come IP mobile, **GSM** adotta un instradamento indiretto inviando dapprima la chiamata del corrispondente alla rete di appartenenza dell'utente mobile e da lì alla rete visitata. Nella terminologia GSM, la **rete di appartenenza** dell'utente mobile è identificata come **home PLMN** (*home public land mobile network*, rete di appartenenza mobile di area pubblica). Questa è costituita dal fornitore di servizi cellulari con cui l'utente mobile ha un contratto (cioè, il gestore di telefonia mobile).

La PLMN visitata, alla quale faremo riferimento come **rete visitata**, è la rete in cui l'utente mobile di volta in volta risiede.

Come nel caso di IP mobile, le responsabilità delle reti domestiche e di quelle visitate sono differenti.

- La **rete di appartenenza** mantiene un database detto **HLR** (*home location register*, registro di localizzazione di appartenenza), che contiene il numero telefonico permanente del telefono cellulare e le informazioni sul profilo degli utenti (tra cui anche l'informazione sulla localizzazione corrente di questi utenti). Quindi, se al momento l'utente mobile sta operando nella rete cellulare di un altro fornitore, HLR conterrà le informazioni necessarie per ottenere l'indirizzo nella rete visitata al quale instradare le chiamate verso l'utente.

Uno switch speciale nella rete di appartenenza, il **GMSC** (*gateway mobile services switching center*) viene contattato per le chiamate dirette agli utenti mobili.

- La **rete visitata** mantiene un database detto **VLR** (*visitor location register*, registro di localizzazione dei visitatori), che contiene una voce per ogni utente mobile che si trova attualmente nella parte della rete da lui servita. Queste voci nascono e muoiono ogni volta che l'utente entra o lascia la rete.

In pratica, la rete del gestore servirà come rete di appartenenza per i suoi utenti e come rete visitata per gli utenti mobili di altri fornitori.

Instrandamento delle chiamate verso utenti mobili.

Vediamo come sono instradate le **chiamate dirette a un utente GSM che si sposta in una rete ospitante**. Considereremo un semplice esempio.

1. Il corrispondente compone il numero di telefono dell'utente mobile. Numero di telefono è fisso e l'utente è mobile.

La prima parte del numero identifica globalmente la rete di appartenenza dell'utente. La chiamata viene instradata dal corrispondente attraverso la rete pubblica commutata verso l'MSC della rete di appartenenza. Questo è il primo tratto della connessione.

2. L'MSC di appartenenza riceve la chiamata e interroga l'HLR per determinare la localizzazione dell'utente. In questo semplice caso, l'HLR restituisce il **MSRN** (*mobile station roaming number*, numero di roaming della stazione mobile): **numero di roaming**. Quest'ultimo è effimero: è assegnato temporaneamente all'utente quando visita una rete. Questo numero ha un ruolo simile al COA di IP mobile ed è invisibile al corrispondente e all'utente.

Se HLR non ha il numero di roaming, restituisce l'indirizzo del VLR della rete visitata. In questo caso l'MSC di appartenenza avrà bisogno di interrogare il VLR per ottenere il numero di roaming dell'utente mobile.

3. Dato il numero di roaming, la chiamata viene instradata lungo il percorso che va dal corrispondente all'MSC di appartenenza all'MSC visitato, fino alla stazione base, che serve l'utente mobile. A questo punto la chiamata è terminata.

Vediamo come l'HLR ottenga informazioni sulla localizzazione dell'utente.

Quando il telefono mobile viene acceso o entra a far parte di una nuova rete visitata, coperta da un nuovo VLR, dovrà registrarsi tra il nodo mobile e il VLR. Quest'ultimo invia un messaggio di richiesta di aggiornamento di localizzazione all'HLR del nodo mobile. Questo messaggio informa l'HLR del numero di roaming con cui il nodo mobile può essere contattato, o dell'indirizzo del VLR (a cui può essere richiesto il numero di roaming). Durante questo scambio, il VLR ottiene anche informazioni sull'utente.

Handoff in GSM.

Quando la **stazione mobile cambia** la sua associazione da una **stazione base a un'altra** nel corso di una chiamata si verifica un **handoff** (passaggio di mano).

La chiamata della stazione mobile è inizialmente (prima dell'handoff) instradata alla stazione mobile attraverso una stazione base (che indicheremo come la **vecchia stazione base**) e dopo l'handoff è instradata attraverso un'altra stazione base (la **nuova stazione base**).

- Assumiamo, inizialmente, che la **vecchia** e la **nuova stazione base** condividano lo **stesso MSC** e che questo effettui il re-instradamento.

Ci sono numerose situazioni in cui si può verificare un handoff. Per esempio,

1. il segnale tra la stazione base attuale e quella mobile potrebbe subire un deterioramento, oppure
2. una cella potrebbe essere sovraccaricata per l'alto numero di chiamate. Questa congestione può essere risolta passando delle stazioni mobili a celle vicine, meno congestionate.

Le **stazioni mobili analizzano** periodicamente l'**intensità del segnale di beacon** (radiofaro) della stazione base cui sono correntemente associate e dei segnali di beacon che provengono dalle stazioni vicine. Queste analisi sono riportate alla stazione base corrente una o due volte al secondo. Questi dati e altri fattori spingono la stazione base a dare inizio all'handoff.

Lo standard GSM non specifica un particolare algoritmo per decidere se iniziare l'handoff.

Le fasi dell'handoff.

1. La vecchia stazione base (BS) comunica all'MSC visitato che sta per essere eseguito un handoff e la nuova stazione BS (o il gruppo di possibili BS) cui l'utente mobile sarà associato.

2. L'MSC visitato inizializza un percorso verso la nuova BS, allocando le risorse necessarie per re-instradare la chiamata e segnalandole che l'handoff sta per essere eseguito.

3. La nuova BS alloca e attiva un canale radio per la stazione mobile.

4. La nuova BS trasmette all'MSC visitato e alla vecchia BS che il percorso da MSC visitato alla nuova BS è stato stabilito e che la stazione mobile dovrà essere informata dell'imminente handoff. La nuova BS fornirà tutte le informazioni di cui la stazione mobile avrà bisogno per associarsi.

5. La stazione mobile è informata che dovrà eseguire l'handoff.

6. Le stazioni mobili e la nuova BS si scambiano uno o più messaggi per completare l'attivazione del nuovo canale, nella nuova BS.

7. La stazione mobile invia un messaggio di completamento dell'handoff alla nuova BS, la quale lo invierà all'MSC visitato. A sua volta, questo re-instraderà la chiamata attiva alla stazione mobile attraverso la nuova BS.

8. Le risorse allocate lungo il percorso verso la vecchia BS vengono rilasciate.

Cosa accade quando una stazione mobile si sposta in una stazione base che è associata a un **MSC differente** rispetto a quello della vecchia stazione base: che cosa accade quando questo handoff inter MSC si verifica più di una volta?

GSM definisce la nozione di **MSC di appoggio** (*anchor MSC*), ovvero quello visitato dalla stazione mobile quando vengono inizializzate le chiamate, e che non può cambiare durante la chiamata. Per tutta la sua durata, la chiamata è instradata dall'MSC di appartenenza all'MSC di appoggio e, in seguito, da questo a quello visitato, dove la stazione mobile è attualmente ubicata. Quando un utente mobile si sposta dall'area di copertura di un MSC a un'altra, le chiamate in esecuzione sono re-instradate dall'MSC di appoggio al nuovo MSC visitato che contiene la nuova stazione base. Quindi, in ogni momento, ci sono al massimo tre MSC (di competenza, d'appoggio e visitato) tra il corrispondente e l'utente mobile.

Piuttosto che mantenere un unico punto di connessione MSC, da quello d'appoggio a quello corrente, sarebbe possibile concatenare gli MSC visitati dall'utente mobile e imporre al vecchio MSC di trasferire la chiamata in corso al nuovo MSC ogni volta che la stazione mobile cambia

centro di smistamento. Questo concatenamento può di fatto verificarsi nelle reti cellulari IS-41, che prevedono anche un passo (opzionale) di ottimizzazione per rimuovere cicli di MSC intermedi che si possono creare.

Wireless e mobilità: l'impatto sui protocolli.

Le reti wireless differiscono in modo significativo dalle loro controparti cablate sia a livello di collegamento sia a livello di rete

Ma esistono importanti differenze ai livelli di trasporto e di applicazione?

I protocolli di trasporto e TCP in particolare possono qualche volta avere prestazioni molto differenti nelle reti cablate e in quelle wireless.

TCP ritrasmette un segmento che può venir perso o essere corrotto lungo il cammino dal trasmittente al ricevente.

Nel caso degli utenti mobili, la perdita può essere causata sia dalla congestione della rete sia dall'handoff.

In tutti i casi, gli acknowledgment TCP che vanno da ricevente a trasmittente indicano soltanto che il segmento non è stato ricevuto intatto; quindi, il trasmittente non è a conoscenza del motivo (congestione, handoff o rilevazione di errori nei bit) per cui è andato perso.

In ogni caso, la reazione è la stessa: invia nuovamente il segmento. Anche la risposta del controllo di congestione di TCP è, in ogni caso, la stessa: TCP riduce la sua finestra di congestione.

In tal modo, TCP implicitamente assume che la perdita del segmento avvenga a causa della congestione piuttosto che della corruzione del segmento o dell'handoff.

Abbiamo visto che gli errori nei bit sono molto più comuni nelle reti wireless che in quelle cablate. Nel caso di errori nei bit o di perdita per handoff, non vi è realmente alcuna ragione perché il trasmittente TCP riduca la finestra di congestione (e quindi la propria frequenza trasmittiva). In verità, potrebbe verificarsi il caso in cui i buffer dei router siano vuoti e che i pacchetti fluiscano lungo il percorso da un estremo all'altro senza problemi di congestione. A causa dell'elevato tasso di errori nei bit dei collegamenti wireless e della possibilità di perdita dovuta all'handoff, la risposta del controllo di congestione di TCP avrebbe potuto causare dei problemi in certe situazioni. Esistono tre approcci per affrontare questo problema.

- Recupero locale (local recovery). Questo approccio mira a recuperare gli errori nei bit quando e dove si verificano.
- Consapevolezza del trasmittente TCP del collegamento wireless. Trasmittenti e riceventi TCP possono essere consapevoli dell'esistenza di un canale wireless, per distinguere le perdite dovute alla congestione nella rete cablata da quelle che si verificano sul collegamento wireless, per corruzione o perdita. In tal modo sarà possibile invocare il controllo di congestione solamente se questa si verifica in una rete cablata.
- Approcci di suddivisione della connessione. In questi approcci, la connessione end-to-end tra l'utente mobile e l'altro punto finale viene spezzata in due connessioni a livello di trasporto: una dall'utente mobile all'access point wireless e una da quest'ultimo all'altro punto finale della comunicazione, che assumiamo sia un'host cablato.

Capitolo 8.

Sicurezza nelle reti.

La sicurezza della comunicazione in rete coinvolge diversi aspetti:

- **Riservatezza:** solo mittente e destinatario dovrebbero essere in grado di comprendere il contenuto del messaggio trasmesso.

Dato che questo può essere intercettato da qualche "spione", è necessario cifrarlo in modo da renderlo incomprensibile a chi lo intercetta.

- **Integrità del messaggio:** occorre che il contenuto della comunicazione non subisca, durante la trasmissione, alterazioni dovute a cause fortuite o a manipolazioni.

- **Autenticazione:** mittente e destinatario devono essere reciprocamente sicuri della loro identità, cioè devono poter confermare che l'altra parte sia effettivamente chi dichiara di essere.

- **Sicurezza operativa:** i servizi offerti in rete devono essere protetti da eventuali attacchi che pregiudichino la loro accessibilità e disponibilità. Un firewall si pone tra la rete dell'istituzione e quella pubblica e controlla i pacchetti in transito. Un sistema di rilevamento delle intrusioni esegue un controllo approfondito dei pacchetti, avvisando gli amministratori di rete in caso di attività sospette.

Esempi di comportamenti malevoli:

- **Eavesdropping:** intercettare i messaggi in una comunicazione tra due interlocutori;
- Inserire messaggi fasulli nel flusso di una comunicazione;
- **Spoofing:** inviare pacchetti con il campo source address fasullo;
- **Hijacking:** dirottare la comunicazione piazzandosi "in mezzo" alla comunicazione;
- **Denial of service:** impedire ad un servizio offerto in rete di essere utilizzabile.

La **crittografia** si pone come pietra angolare della sicurezza di rete e il suo utilizzo risulta fondamentale non solo per fornire riservatezza, ma anche come un importante strumento per il servizio di autenticazione e l'integrità del messaggio.

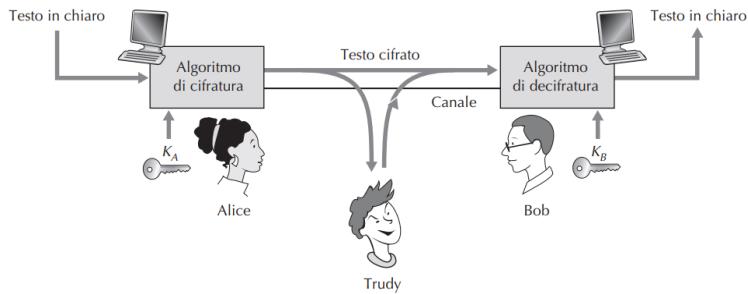
Le tecniche di crittografia consentono al trasmittente di mascherare i dati in modo che un intruso non possa comprenderne il contenuto. Il ricevente, naturalmente, deve invece essere in grado di recuperare i dati originali.

Un **sistema crittografico** è un sistema in **grado di cifrare** (encryption) e **decifrare** (decryption) un messaggio attraverso l'uso di un algoritmo e una chiave (stringa alfanumerica). Il messaggio da cifrare è detto **testo in chiaro** (plaintext o cleartext) mentre il risultato dell'algoritmo crittografico è detto **testo cifrato** (ciphertext).

Esistono due tipologie di crittografia:

- **Crittografia a chiave simmetrica:** mittente e destinatario usano la stessa chiave (segreto condiviso) per encryption e decryption;

- **Crittografia a chiave pubblica:** la chiave per la encryption è pubblica (nota a tutti) mentre la chiave per la decryption è segreta (privata).



Alice fornisce all'algoritmo di cifratura una **chiave**, K_A , cioè una stringa alfanumerica che genera il testo cifrato. La notazione $K_A(m)$ denota la forma cifrata utilizzando la chiave K_A del messaggio in chiaro, m . Bob fornisce una chiave, K_B , all'algoritmo di decifratura che legge il testo cifrato e restituisce quello originale in chiaro. Cioè, quando Bob riceve un messaggio cifrato $K_A(m)$, lo può decifrare con $K_B(K_A(m)) = m$.

CRITTOGRAFIA A CHIAVE SIMMETRICA.

In un sistema a **chiave simmetrica**, entrambi gli interlocutori devono conoscere la chiave K_{A-B} per cifrare e decifrare il messaggio.

Si è dimostrato che con una chiave di almeno 128 bit (13 caratteri) è impossibile decrittare un codice in tempi relativamente limitati. Il problema di questo tipo di crittografia, dunque, non è tanto legato alla complessità di crittazione ma tanto alla funzionalità logistica. La chiave deve essere distribuita tra mittente e destinatario mantenendola rigorosamente segreta e quindi adoperando un canale di comunicazione sicuro.

Due esempi di crittografia simmetrica:

Cifrario per sostituzione: unità di testo delle lettere in chiaro sono sostituite con corrispondenti sequenze di simboli nel testo cifrato secondo uno schema regolare. In particolare, in un **cifrario monoalfabetico**, abbiano una corrispondenza fissa tra ciascuna lettere dell'alfabeto in chiaro ed una lettera dell'alfabeto cifrato.

Il cifrario monoalfabetico risulta più efficiente di quello di Cesare poiché in quello di Cesare ci sono 25 lettere (alfabeto inglese) $\leq 25! < 26!$

Lettere 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
Lettere cifrate:	m n b v c x z a s d f g h j k l p o i u y t r e w q

Con un attacco a forza bruta di 26 lettere, con $26!$ (nell'ordine di 10^6) tentativi sembrerebbe difficile decifrare il codice poiché richiederebbe un tempo eccessivo. Ma con tecniche di critto-analisi che prevedono lo studio di pattern ricorrenti ed analisi statistiche delle frequenze di occorrenza si può arrivare a decifrare il messaggio.

Un approccio più sofisticato è quello della **cifratura polialfabetica** che utilizza l'uso di molteplici sostituzioni monoalfabetiche.

In sostanza, le varie occorrenze di una stessa lettera vengono codificate in modo diverso a seconda della posizione in cui appaiono nel messaggio in chiaro.

Lettere 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
$C_1(k=5)$:	f g h i j k l m n o p q r s t u v w x y z a b c d e
$C_2(k=19)$:	t u v w x y z a b c d e f g h i j k l m n o p q r s

Due diversi cifrari di Cesare, C_1 con $k = 5$ e C_2 con $k = 19$ che potremmo scegliere di utilizzare seguendo la sequenza C_1, C_2, C_2, C_1, C_2 . Cioè, la prima lettera del testo in chiaro deve essere sostituita con C_1 , la seconda e la terza con C_2 , la quarta con C_1 e la quinta con C_2 . Lo schema si ripete in modo ciclico, per cui la sesta lettera verrà scambiata con C_1 , e così via.

Cifrari a blocchi: un blocco di k bit del testo in chiaro è codificato con altri k bit nel testo in codice secondo uno schema fisso, ad esempio:

Ingresso	Uscita	Ingresso	Uscita
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

In pratica vi è una **corrispondenza a uno a uno**: cioè vi è un'uscita diversa per ciascun ingresso.

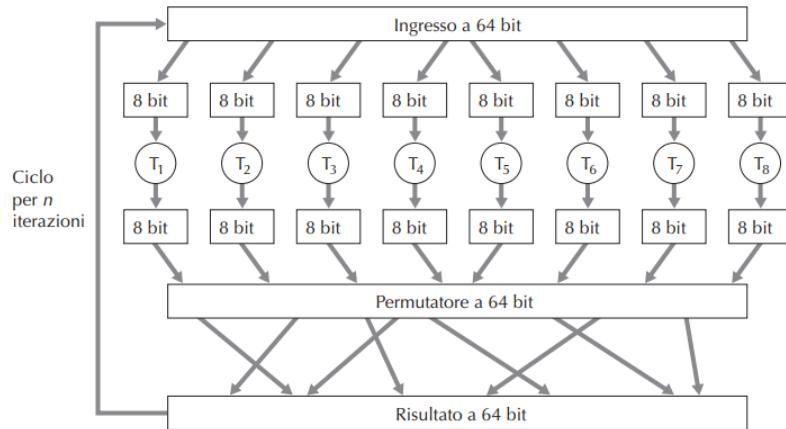
Prendendo blocchi di k bit è possibile creare 2^k permutazioni (corrispondenze) diverse. Si può vedere ciascuna di queste corrispondenze come una chiave. Aumentando quindi il valore di k è possibile aumentare la sicurezza.

L'attacco a forza bruta per questo cifrario consiste nel provare a decifrare il testo cifrato usando tutte le possibili corrispondenze. Con solo 40.320 corrispondenze (quando $k = 3$) questo può essere svolto velocemente da un PC. Per contrastare un attacco a forza bruta, i cifrari a blocchi tipicamente usano blocchi molto più grandi, con $k = 64$ o maggiore.

Il problema è mantenere, sia per la cifratura che per la decifratura una tabella di 2^k elementi in memoria.

Quello che si usa fare tipicamente è usare delle funzioni che simulano in modo casuale tabelle permutate.

Quello che segue è un esempio di una funzione di quel tipo.



1. Suddivide il blocco di 64 bit in 8 parti di 8 bit ciascuno.
 2. Ciascuna parte viene elaborata da una tabella di 8x8 bit.
 3. Le parti criptate sono riassembrate e le posizioni dei 64 bit sono rimescolate.
 4. Questo risultato viene rinvia all'ingresso a 64 bit, dove inizia un'altra interazione (solitamente 16 iterazioni).
- Lo scopo delle iterazioni è quello di far in modo che ciascun ingresso influenzi molti, se non tutti, i bit del risultato finale.

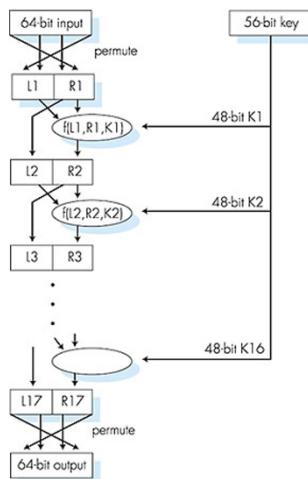
Oggi ci sono parecchi cifrari a blocchi comuni, compreso **DES** (data encryption standard), **3DES**, e **AES** (advanced encryption standard).

Data Encryption Standard (DES).

Si tratta di un **algoritmo a chiave simmetrica** con chiave a 64 bit (ma solo 56 utili poiché 8 sono di controllo).

Sui 64 bit dopo ogni 7 bit, l'ottavo è utilizzato come bit di parità per i 7 precedenti (ottavo, 16-esimo,...,64-esimo).

L'algoritmo prevede 16 cicli identici. Prima dell'iterazione principale il blocco è diviso in due metà di 32 bit e processato alternativamente. Questo assicura che la cifratura e la decifratura siano processi simili, con l'unica differenza che le sottochiavi sono applicate nell'ordine inverso.



Ad ogni iterazione viene usata la cosiddetta **funzione di Festel** che mescola metà del blocco con una parte della chiave. Il risultato della funzione è poi combinato con l'altra metà del blocco, e le due metà sono scambiate prima del ciclo successivo.

Attualmente DES è considerato insicuro per via della chiave di solo 56 bit. Con la potenza di calcolo disponibili oggi si può forzare una chiave DES in poche ore. L'algoritmo è ritenuto più sicuro reiterandolo 3 volte nel Triple DES. Negli ultimi anni DES è stato sostituito da AES che elimina molti dei problemi del DES.

Advanced Encryption Standard (AES)

A differenza del DES elabora i dati a blocchi da 128 bit e la chiave può essere di 128, 192 o 256 bit.

L'algoritmo opera utilizzando matrici di 4x4 byte chiamati states. Per cifrare sono previsti diversi round o cicli di processamento in cui vengono operate:

- sostituzioni non lineari di tutti i byte
- spostamento dei byte in base alla riga di appartenenza
- combinazione dei byte con un'operazione lineare
- combinazione dei byte con la chiave di sessione.

Con la forza bruta una chiave AES a 64 bit è stata violata in 5 anni.

Cipher Block Chaining.

La debolezza più forte dei cifrari a blocchi è che blocchi in input uguali producono lo stesso testo cifrato e si prestano dunque a crittoanalisi.

Per ovviare a questo problema si usa la tecnica detta **cipher block chaining**. Con questa modalità, l'algoritmo a blocchi crittografa il messaggio creando una catena di blocchi in cui ognuno di essi dipende dalla cifratura del blocco precedente. Questa interdipendenza assicura che un cambiamento ad un qualsiasi bit del testo in chiaro causerà un cambiamento nel blocco finale crittografato.

L'inconveniente della crittografia a chiave simmetrica è che mittente e destinatario devono in qualche modo scambiarsi la chiave tramite un canale sicuro.

CRITTOGRAFIA A CHIAVE PUBBLICA.

In un sistema a **chiave pubblica**:

- la chiave **pubblica** usata solitamente per la cifratura deve essere nota a tutti.
- la chiave **privata** usata solitamente per la decifratura è nota solo al destinatario.

Se con una delle due chiavi si cifra il messaggio, allora quest'ultimo sarà decifrato solo con l'altra. È importante notare che le due chiavi sono intercambiabili nei ruoli di cifratura e decifratura.

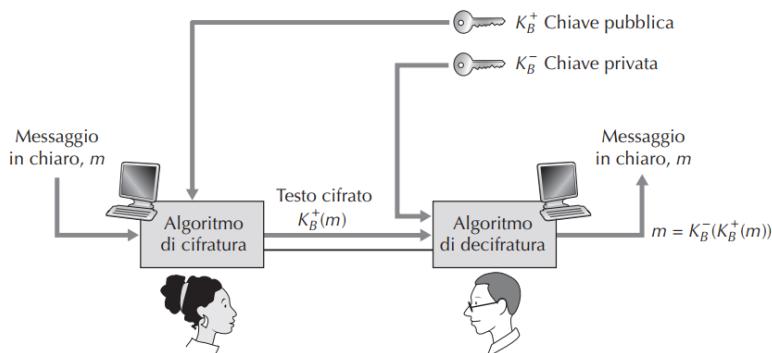
La forza di un sistema a chiave pubblica si basa sulla difficoltà di determinare la chiave privata corrispondente alla chiave pubblica.

A causa del peso computazionale della crittografia a chiave pubblica, essa di solito è usata per piccoli blocchi di dati, in genere il trasferimento di una chiave simmetrica. Questa è poi utilizzata per cifrare messaggi lunghi.

Fasi.

1. **Inviare un messaggio cifrato ad un destinatario.** Il mittente cifra il messaggio con la chiave pubblica del destinatario. L'unico a poter decifrare è il destinatario, possessore della chiave privata.

2. **Verificare l'autenticità di un messaggio.** Il destinatario verifica l'autenticità del messaggio decifrando con la chiave pubblica del mittente. Tutti i possessori della chiave pubblica possono leggere il messaggio.



Alice si procura la chiave pubblica di Bob e codifica quindi il suo testo in chiaro (m), utilizzando la chiave pubblica di Bob e un dato di cifratura, e genera un messaggio crittato $\square KB+m$.

Quando Bob riceve il messaggio cifrato, utilizza la sua chiave privata e un algoritmo per decodificarlo $\square KB-KB+m$

Alice può utilizzare la chiave pubblica di Bob per inviargli un messaggio segreto senza che nessuno di loro debba distribuire chiavi segrete. Invertendo la cifratura con chiave pubblica e quella con chiave privata si ottiene lo stesso risultato, cioè $KB-KB+m = KB-KB+m=m$.

RSA. (Generare una coppia di chiavi)

Per utilizzare questo tipo di crittografia, è necessario dunque creare una coppia di chiavi tale che:

$$K_B^-(K_B^+(m)) = m$$

Gli algoritmi utilizzati per generare le chiavi si basano spesso su problemi matematici che attualmente non ammettono alcuna soluzione particolarmente efficiente. Uno di questi è l'**algoritmo RSA** (Rivest, Shamir, Adleman):

1. Si scelgono due numeri primi grandi p e q , più grande è il loro valore più difficile risulterà violare RSA.
2. Si calcolano: $n=(p) \times (q)$ modulo e prodotto $z=(p-1) \times (q-1)$
3. Si sceglie un numero e ($e < n$) chiamato esponente pubblico che non abbia fattori comuni con z .
4. Si sceglie un numero d chiamato esponente privato tale che $(e) \times (d)-1$ sia esattamente divisibile per z (($e) \times (d) \bmod z = 1$)

La **chiave pubblica** è (n,e) , la **chiave privata** è (n,d)

La robustezza dell'algoritmo sta nel fatto che per calcolare d da e (o viceversa) non basta la conoscenza di n ma serve il numero $z=(p-1)(q-1)$, e che il suo calcolo richiede tempi molto elevati. Infatti, fattorizzare in numeri primi è un'operazione computazionalmente costosa.

RSA cifratura e decifratura.

Dati (n,e) e (n,d) come calcolati precedentemente:

$$c = m^e \bmod n \quad (\text{resto della divisione di } m^e \text{ per } n)$$

$$m = c^d \bmod n \quad (\text{resto della divisione di } c^d \text{ per } n)$$

- La cifratura del testo in chiaro m si effettua calcolando:

$$m = \underbrace{(m^e \bmod n)}_c^d \bmod n$$

L'efficacia di RSA consiste nel fatto che non si conoscono algoritmi veloci per la fattorizzazione dei numeri interi. Quindi, anche conoscendo il valore pubblico n risulta computazionalmente proibitivo determinare i fattori primi p e q (con i quali, dato il valore pubblico e , si può facilmente calcolare la chiave segreta). D'altra parte, non si sa se esiste o no un algoritmo veloce per la fattorizzazione di un numero, ed è in questo senso che la sicurezza di RSA non è garantita.

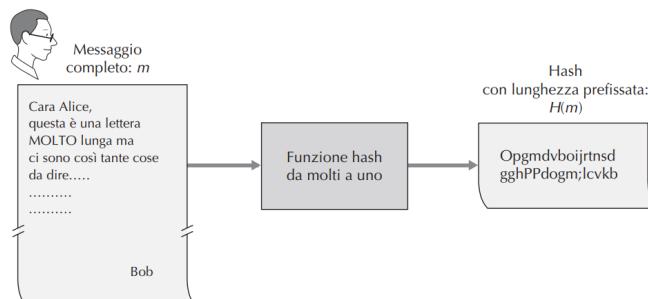
Integrità dei messaggi.

Un altro tema ugualmente importante, dopo la trattazione della riservatezza di una comunicazione è quello dell'**integrità dei messaggi**, anche noto come **autenticazione dei messaggi**. Una soluzione a questo problema sono le **funzioni di hash crittografiche**.

Una **funzione hash** prende un testo in ingresso m , e produce una stringa di lunghezza prefissa $H(m)$ detta hash. La funzione H è tale che è computazionalmente impossibile trovare 2 messaggi diversi x e y tali che $H(x)=H(y)$.

Vale a dire che, dal punto di vista computazionale, un malintenzionato non deve avere alcuna possibilità di poter sostituire un messaggio con un altro messaggio che sia protetto dalla funzione hash.

Equivalentemente noto $m = H(x)$.



MD5: È un algoritmo di hashing molto usato ed è in grado di calcolare una hash di 128 bit con un processo a quattro fasi.

Questa funzione prende in input una stringa di lunghezza arbitraria e ne produce in output un'altra a 128 bit. Il processo avviene molto velocemente e l'output (noto anche come "MD5 Checksum" o "MD5 Hash") restituito è tale per cui è altamente improbabile ottenere con due diverse stringhe in input uno stesso valore hash in output.

SHA-1: È un altro importante algoritmo di hashing basato su principi simili a quelli di MD4. Produce una sintesi (digest) del messaggio di 160 bit. La maggiore lunghezza del risultato rende SHA-1 più sicuro.

Autenticazione dei messaggi.

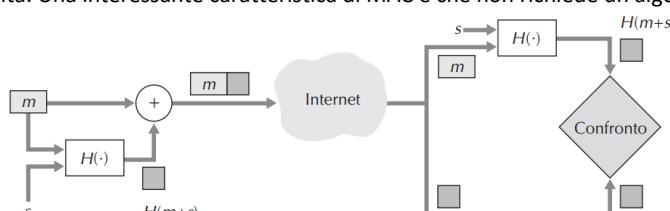
Per garantire l'integrità dei messaggi, oltre alle funzioni di hash, **mittente e destinatario hanno bisogno di un segreto condiviso**, chiamato **chiave di autenticazione**. L'integrità è realizzata come segue:

1. Il mittente crea un messaggio m , concatena s con m per creare $m+s$, calcola la stringa di hash $H(s+m)$. Questa stringa calcolata è detta **codice di autenticazione del messaggio (MAC)**

2. Il mittente aggiunge il MAC al messaggio m e lo manda al destinatario.

3. Il destinatario riceve il messaggio esteso ($m+MAC$) e avendo ricevuto m e conoscendo s , calcola il MAC $H(m+s)$.

Se $H(m+s)=h$ è stata realizzata l'integrità. Una interessante caratteristica di MAC è che non richiede un algoritmo di cifratura.



Legenda:

- m = Messaggio
- s = Segreto condiviso

AUTENTICAZIONE.

La **firma digitale** è una tecnica crittografica che permette di realizzare l'autenticazione nelle comunicazioni in rete.

Le firme digitali si propongono di avere tre importanti requisiti:

- **Autenticazione:** il destinatario deve poter verificare l'identità del mittente;
- **Non ripudio:** il mittente deve poter disconoscere un documento da lui firmato;
- **Integrità:** il destinatario non deve poter modificare un documento firmato da qualcun altro.

Un tipico schema di firma elettronica basata sulla crittografia a chiave pubblica prevede i seguenti step:

- Il mittente A firma il messaggio m crittografandolo con la sua chiave privata creando un messaggio firmato.
- Il destinatario B riceve il messaggio m con la firma digitale di A.
- Il destinatario verifica che m sia stato effettivamente firmato da A decifrando con la chiave pubblica di B.

Autenticazione di un punto terminale.

Con **autenticazione di un punto terminale** (*end-point authentication*) si intende il processo attraverso il quale una entità prova la sua identità a un'altra entità su una rete di calcolatori.

Analizziamo varie versione di un protocollo di autenticazione che si chiama *ap*.

Cominciamo supponendo che Alice debba autenticarsi a Bob.

Protocollo di autenticazione ap1.0.

Alice invia un messaggio a Bob dicendo semplicemente di essere Alice. Ovviamente, il destinatario non può assolutamente essere sicuro che la persona che ha inviato il messaggio "Sono Alice" sia davvero Alice, perché anche Trudy potrebbe averlo fatto.

Protocollo di autenticazione ap2.0.

Se Alice avesse un indirizzo di rete conosciuto (per esempio, l'indirizzo IP) che utilizza abitualmente, Bob potrebbe autenticarla verificando la corrispondenza fra l'indirizzo sorgente sul datagramma IP che trasporta il messaggio e quello di Alice, ma non è difficile creare un datagramma IP contenente un indirizzo sorgente artefatto e poi inviarlo attraverso il protocollo a livello di collegamento al primo router. Ciò è possibile nel caso in cui si abbia accesso al codice del sistema operativo e si possa costruire un proprio kernel di sistema. Questo approccio è una forma di **spoofing di IP**. Lo **spoofing** (impersonificazione) potrebbe essere evitato se il router del primo hop del mittente malintenzionato fosse configurato in modo da inoltrare solo datagrammi con il suo indirizzo IP sorgente.

Protocollo di autenticazione ap3.0.

Un **classico approccio all'autenticazione** è l'**utilizzo di una password**. Una password è un segreto condiviso dall'entità che svolge il processo di riconoscimento e da quella che deve essere autenticata. Nel protocollo ap3.0 Alice invia quindi la sua password a Bob.

Dato che l'impiego di password è ampiamente diffuso, potremmo aspettarci che il protocollo ap3.0 sia piuttosto sicuro. E invece no. Se Trudy intercetta la comunicazione di Alice, può scoprirla la password.

Protocollo di autenticazione ap3.1

Un modo per superare i limiti di ap3.0 consiste nel cifrare la password in modo da impedire all'intruso di appropriarsene. Se Alice condividesse una chiave simmetrica segreta kA-B con Bob, potrebbe utilizzarla per cifrare il suo messaggio di identificazione e la password. Bob provvederebbe quindi a decodificarla e a controllarne la veridicità.

Anche se questa versione del protocollo, che chiameremo ap3.1, impedisce di scoprire la password di Alice, l'utilizzo della crittografia non risolve il problema dell'autenticazione. Bob è ancora soggetto al cosiddetto **attacco di replica** (*playback attack*). Trudy può infatti inserirsi nella comunicazione, registrare la versione cifrata della chiave e successivamente riprodurla per inviarla a Bob sostenendo di essere Alice. In sostanza, l'impiego della password cifrata non ha reso la situazione diversa da quella del protocollo ap3.0.

Protocollo di autenticazione ap4.0.

Lo scenario con fallimento del ap3.1 deriva dal fatto che Bob non riesce a distinguere fra l'autenticazione originale di Alice e la successiva riproduzione.

Un nonce è un numero che il protocollo userà soltanto una volta nel seguente modo.

- Alice invia il messaggio, "Sono Alice", a Bob.
- Bob sceglie un nonce, R, e lo trasmette ad Alice.
- Alice utilizza kA-B, la chiave simmetrica segreta che condivide con Bob, per codificare il nonce, e gli rinvia il valore risultante, kA-B(R).

Il nonce viene usato per assicurare che Alice è attiva.

- Bob decifra il messaggio ricevuto: se il nonce è quello da lui inviato, Alice è autenticata.

Attraverso l'utilizzo del nonce, R, e con il controllo del valore di ritorno, kA-B(R), Bob può essere sicuro che si tratta veramente di Alice e che si trova in quel momento all'altro capo del collegamento.

Per risolvere il problema dell'autenticazione nel protocollo ap4.0 è stato impiegato un nonce e la crittografia a chiave simmetrica.

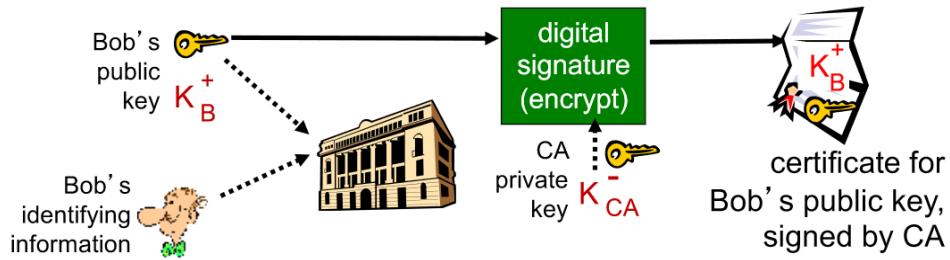
Certificazione della chiave pubblica.

Un importante applicazione della firma digitale è la **certificazione della chiave pubblica**, cioè la certificazione che una chiave pubblica appartenga a una specifica entità. Per utilizzare la crittografia a chiave pubblica, infatti, è necessario che gli utenti abbiano garanzie sulle chiavi pubbliche fornite.

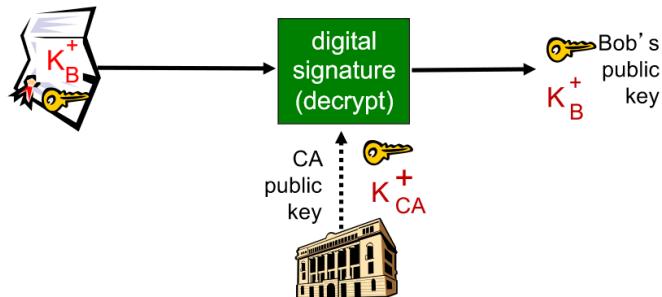
Generalmente, la relazione tra una data chiave pubblica e una determinata entità è stabilita da un'**autorità di certificazione** (*CA certification authority*), il cui compito è di validare l'identità ed emettere certificati di autenticazione.

- Verifica che un'entità (persona fisica, router o altro) sia veramente chi afferma di essere. In effetti, non esistono specifiche procedure che stabiliscono come questa mansione debba essere svolta per cui, quando si definisce un accordo con una CA, occorre sperare che questa esegua un appropriato e rigoroso controllo dell'identità.

- Una volta verificata l'identità, la CA rilascia un certificato che autentica la corrispondenza fra chiave pubblica ed entità. Il certificato contiene la chiave pubblica e le informazioni di identificazione globali e uniche del suo proprietario (per esempio, il nome di una persona o un indirizzo IP), ed è firmato digitalmente dalla CA.



Quando un utente A desidera la chiave pubblica di un'altra entità B deve ottenere il certificato di B e poi applicare la chiave pubblica della CA al certificato di B per ottenere la chiave pubblica di B.



Rendere sicure le connessioni TCP: SSL.

La crittografia può essere usata per arricchire TCP con servizi di sicurezza, comprese la riservatezza, l'integrità dei dati e l'autenticazione endpoint. Questa versione arricchita di TCP è comunemente nota come **secure sockets layer (SSL)**.

SSL è supportato da tutti i più comuni browser e web server e viene utilizzato da gmail e da tutti i siti di commercio elettronico in Internet (compresi Amazon, eBay, e TaoBao). Se avete mai comperato qualcosa su Internet con la vostra carta di credito, la comunicazione tra il vostro browser e il web server è quasi certamente avvenuta usando SSL.

Immaginiamo che Bob si connetta al sito della Alice Incorporated, che vende profumi. Desideroso di fare acquisti, Bob compila una scheda in cui indica la tipologia e la quantità desiderata di profumi, il suo indirizzo e il numero della carta di credito e clicca su "invia".

Tutto ciò sembra funzionare senza intoppi, ma se non si adottassero specifiche misure di sicurezza Bob potrebbe avere qualche brutta sorpresa.

- Se non venisse assicurata la **riservatezza (cifratura)**, un intruso potrebbe intercettare l'ordine, ottenere le informazioni relative alla carta di credito di Bob, e fare acquisti a sue spese.
- Se non venisse assicurata l'**integrità dei dati**, un intruso potrebbe modificare l'ordine di Bob e fargli comperare, per esempio, 10 volte più bottiglie di profumo di quelle che desidera.
- Infine, se non venisse usata l'**autenticazione del server**, il sito potrebbe mostrare il logo della Alice Incorporated, ma essere in realtà gestito da Trudy che potrebbe incassare i soldi di Bob e sparire.

SSL cerca di risolvere tutti questi problemi aggiungendo a TCP la riservatezza, l'integrità dei dati e l'autenticazione del client e del server.

SSL viene spesso usato per fornire sicurezza alle transazioni che hanno luogo tramite HTTP, tuttavia, dato che SSL rende sicuro TCP, può essere sfruttato da qualsiasi altra applicazione che faccia uso di TCP. SSL fornisce una semplice API (application programmer interface) verso le socket, analoga a quella fornita da TCP: quando un'applicazione vuole usare SSL, include le classi/librerie SSL che forniscono l'interfaccia socket SSL allo sviluppatore dell'applicazione.

Quadro generale

versione semplificata di SSL o "quasi-SSL".

"Quasi-SSL" e SSL hanno tre fasi: handshake, derivazione delle chiavi e trasferimento dati.

Descriveremo queste tre fasi per una sessione di comunicazione tra un client (Bob) e un server (Alice) che dispone di una coppia di chiavi privata/pubblica e di un certificato che lega la sua identità alla sua chiave pubblica.

Handshake.

Durante la fase di handshake, Bob ha bisogno di (a) stabilire una connessione TCP con Alice, (b) verificare che Alice sia realmente Alice e (c) inviarle una chiave segreta principale che verrà utilizzata da entrambi per generare tutte le chiavi simmetriche di cui hanno bisogno per la sessione SSL (Figura 8.25). Notate che una volta che la connessione TCP è stata stabilita, Bob manda ad Alice un messaggio "hello" e Alice risponde con il proprio certificato che contiene la sua chiave pubblica. Dato che il certificato è garantito da una CA, Bob sa con certezza che la chiave pubblica del certificato appartiene ad Alice. Bob genera un master secret (MS): un valore segreto che verrà usato solo per questa sessione SSL e dal quale verranno derivate altre chiavi. Il master secret viene cifrato con la chiave pubblica di Alice, per creare un encrypted master secret (EMS) che viene inviato ad Alice, la quale decifra l'EMS con la sua chiave privata e ottiene MS. Dopo questa fase Bob ha autenticato Alice, ed entrambi (ma nessun'altro) conoscono il master secret per questa sessione SSL.

Derivazione delle chiavi.

In linea di principio, MS, ora condiviso da Bob e Alice, potrebbe essere usato come la chiave simmetrica di sessione per tutte le successive cifrature e verifiche dell'integrità dei dati. È però generalmente considerato più sicuro che Bob e Alice usino ciascuno chiavi crittografiche differenti, oltre che a impiegare chiavi diverse per la cifratura e la verifica dell'integrità dei dati. Quindi, Alice e Bob usano MS per generare 4 chiavi:

- EB = chiave di cifratura di sessione per i dati inviati da Bob ad Alice
- MB = chiave MAC di sessione per i dati inviati da Bob ad Alice
- EA = chiave di cifratura di sessione per i dati inviati da Alice a Bob
- MA = chiave MAC di sessione per i dati inviati da Alice a Bob

Le chiavi possono essere generate semplicemente suddividendo MS in quattro parti (anche se nella reale implementazione di SSL è un po' più complicato, come vedremo fra breve). Alla fine della fase di derivazione delle chiavi, sia Alice sia Bob hanno 4 chiavi. Le due chiavi di cifratura verranno usate per cifrare i dati, e le due chiavi MAC verranno usate per verificarne l'integrità.

Trasferimento dati.

Ora che Bob sa con certezza che sta comunicando con Alice, ed entrambi condividono le stesse 4 chiavi di sessione (EB, MB, EA, e MA), i due possono iniziare a scambiarsi dati in sicurezza sulla connessione TCP. Dato che TCP è un protocollo che si basa su un flusso di byte, un approccio naturale per SSL sarebbe cifrare al volo i dati applicativi e passarli poi a TCP. Ma se facessimo così, dove metteremmo il MAC per la verifica dell'integrità? Certamente non vorremmo aspettare fino alla fine della sessione TCP per verificare l'integrità di tutti i dati di Bob che sono stati inviati. Per risolvere questo problema, SSL suddivide il flusso di dati in record a cui aggiunge un MAC per la verifica dell'integrità e che poi cifra. Per creare il MAC, Bob passa i dati del record assieme alla chiave M

B come a una funzione hash, come spiegato nel Paragrafo 8.3. Per cifrare il pacchetto composto da record e MAC, Bob usa la sua chiave di cifratura di sessione E B. Il pacchetto cifrato viene poi passato a TCP per essere trasportato in Internet.

Sebbene questo approccio contribuisca a fornire l'integrità dei dati per l'intero flusso del messaggio, non è ancora a prova di bomba. In particolare, supponete che Trudy stia usando un attacco di tipo "man-in-the-middle" e abbia la capacità di inserire, cancellare e sostituire segmenti nel flusso di segmenti TCP inviati da Alice a Bob. Trudy, per esempio, può catturare due segmenti inviati da Bob, invertirne l'ordine, aggiustarne i numeri di sequenza TCP (che non possono essere cifrati) e poi inviare i due segmenti invertiti ad Alice. Assumendo che ciascun segmento TCP incapsuli esattamente un record, diamo uno sguardo a come Alice elaborerà i due segmenti:

1. il TCP di Alice pensa che tutto vada bene e passa i due record al sottolivello SSL.
2. SSL di Alice decifra i due record.
3. SSL di Alice usa il MAC in ciascun record, per verificare l'integrità dei dati.
4. SSL passa i flussi di byte decifrati dei due record al livello applicativo, ma il flusso di byte completo ricevuto da Alice non sarà nell'ordine corretto, a causa dei due record invertiti.

Siete incoraggiati a considerare scenari simili per quando Trudy elimina o sostituisce segmenti.

La soluzione a questo problema, come probabilmente immaginerete, è di usare dei numeri di sequenza oltre che quelli di TCP. SSL si comporta come segue: Bob mantiene un contatore di numeri di sequenza che inizia da zero e viene incrementato per ciascun record che SSL invia. Bob non include effettivamente il numero di sequenza nel record vero e proprio, ma lo include nel MAC, quando ne effettua il calcolo. Quindi, il MAC è ora un hash di dati, cui si aggiunge la chiave MAC MB e il numero di sequenza corretto. Alice conserva traccia dei numeri di sequenza di Bob, per verificare l'integrità dei dati di un record, includendo nel calcolo del MAC il numero di sequenza appropriato. Questo uso dei numeri di sequenza da parte di SSL evita che Trudy possa eseguire un attacco di tipo man-in-the-middle, come nel caso del riordino e della sostituzione dei segmenti.

Record SSL

Il record SSL (come quello di "quasi-SSL") è mostrato nella Figura 8.26. Il record consiste nei seguenti campi: tipo, versione, lunghezza, dati e MAC. Si noti che i primi tre campi non sono cifrati.

Il campo tipo indica se il record è un messaggio di handshake o uno che contiene dati applicativi. Viene anche usato per chiudere una sessione SSL, come vedremo in seguito. SSL al lato ricevente usa il campo lunghezza per estrarre i record SSL dal flusso di byte TCP in ingresso. Il campo versione è auto esplicativo.

Un quadro più completo

Handshake SSL.

SSL non richiede che Alice e Bob usino uno specifico algoritmo a chiave simmetrica o uno specifico a chiave pubblica, o uno specifico MAC, ma consente loro di accordarsi all'inizio della sessione SSL, durante la fase di handshake, su quali algoritmi crittografici useranno. Inoltre, durante la fase di handshake, Alice e Bob si scambiano dei nonce che vengono usati nella creazione delle chiavi di sessione (EB, MB, EA, e MA).

I passi del reale handshake SSL sono i seguenti.

1. Il client invia, assieme al proprio nonce, la lista degli algoritmi crittografici da lui supportati.
2. Dalla lista, il server sceglie un algoritmo a chiave simmetrica (per esempio, AES), uno a chiave pubblica (per esempio RSA, con una specifica lunghezza di chiave) e un algoritmo MAC. Restituisce al client le proprie scelte, insieme a un certificato e al nonce del server.
3. Il client verifica il certificato, estrae la chiave pubblica del server, genera un premaster secret (PMS) e lo cifra con la chiave pubblica del server per poi mandarlo cifrato al server.
4. Usando la stessa funzione di derivazione della chiave, come specificato dallo standard SSL, il client e il server calcolano indipendentemente il master secret partendo da PMS e nonce. Il master secret viene poi suddiviso per generare le due chiavi di cifratura e le due chiavi MAC. Inoltre, se l'algoritmo a chiave simmetrica va uso di CBC (come nel caso di 3DES e AES) i due vettori di inizializzazione (uno per ogni capo della connessione) sono anch'essi ottenuti dal master secret. D'ora in poi tutti i messaggi inviati tra il client e il server sono cifrati e autenticati con il MAC.
5. Il client invia un MAC di tutti i messaggi di handshake.
6. Il server manda un MAC di tutti i messaggi di handshake.

Gli ultimi due passi proteggono l'handshake dalla manomissione. Per accorgersene, si osservi che nel Passo 1, il client offre tipicamente una lista di algoritmi, alcuni forti e alcuni deboli. Questa lista di algoritmi viene inviata in chiaro, in quanto né gli algoritmi di cifratura né le chiavi sono state ancora accordate. Trudy, se usa un approccio di tipo man-in-the-middle, potrebbe cancellare dalla lista gli algoritmi più sicuri,

costringendo il client a selezionare un algoritmo debole. Per evitare questo tipo di attacco di alterazione, nel Passo 5 il client fa pervenire un MAC della concatenazione di tutti i messaggi di handshake, che ha spedito e ricevuto. Il server può confrontare questo MAC con quello dei messaggi di handshake da lui mandati e ricevuti: se c'è un'inconsistenza il server può chiudere la connessione. In modo analogo, il server invia un MAC dei messaggi di handshake che ha visto, consentendo al client di verificarne la inconsistenza.

Potreste stupirvi dell'esistenza delle nonce ai passi 1 e 2. I numeri in sequenza non sono sufficienti a prevenire gli attacchi basati sulla ripetizione dei segmenti? La risposta è sì, ma da soli non riescono a prevenire quelli basati sulla ripetizione delle connessioni. Supponiamo, per esempio, che Trudy spiò tutti i messaggi tra Alice e Bob. Il giorno successivo Trudy finge di essere Bob e invia ad Alice esattamente la stessa sequenza di messaggi che Bob le aveva inviato il giorno precedente. Se Alice non usa un nonce risponde esattamente con la stessa sequenza di messaggi che ha inviato il giorno prima. Alice non avrà alcun sospetto perché ogni messaggio che riceve supera

il controllo di integrità. Se Alice fosse un server di e-commerce, penserebbe che Bob sta facendo un secondo ordine esattamente per la stessa cosa. D'altra parte, includendo una nonce nel protocollo, Alice invia nonce diverse su ogni sessione TCP, forzando le chiavi di cifratura a essere diverse nelle due occasioni. Perciò, quando Alice riceve i

record SSL replicati da Trudy, questi falliscono i controlli di integrità e quindi la transazione non avrà successo. Riassumendo, in SSL si usano i nonce per difesa contro la ripetizione delle connessioni (connection reply attack) e i numeri di sequenza per difendersi contro il rinvio di pacchetti individuali durante una sessione aperta.

Chiusura della connessione.

A un certo punto, tanto Bob quanto Alice vorranno terminare la sessione SSL. Un approccio potrebbe essere quello di lasciare che sia Bob a concludere la sessione, semplicemente terminando la connessione TCP sottostante, cioè Bob invia un segmento TCP FIN ad Alice. Ma un modello così semplice prepara il campo a un attacco di tipo truncation, dove Trudy ancora un volta si pone nel mezzo di una sessione SSL e la termina prematuramente con un messaggio TCP FIN. Se Trudy facesse questo, Alice penserebbe di aver ricevuto tutti i dati da Bob, quando effettivamente ne ha ricevuto solo una parte. La soluzione a questo problema è indicare nel campo tipo se il record serve a terminare la sessione SSL. Sebbene il tipo SSL venga inviato in chiaro, è autenticato dal ricevente usando il MAC del record. Includendo questo tipo di campo, se Alice dovesse ricevere un TCP FIN prima di aver ricevuto il record SSL di chiusura, saprebbe che sta accadendo qualcosa di insolito.