Reti il sistema telefonico

Il sistema telefonico riveste un ruolo centrale per le comunicazioni a distanza fra computer, per vari motivi:

- sarebbe proibitivo in termini di costi connettere, con appositi cavi, apparecchiature distanti centinaia di km o più;
- è illegale, praticamente in tutti i paesi, stendere cavi sul suolo pubblico.

Purtroppo il sistema telefonico, o ***rete pubblica telefonica commutata***, è nato e si è evoluto in funzione delle esigenze della fonia, anche se recentemente sta diventando sempre più adatto al traffico dati, grazie ai nuovi mezzi trasmissivi quali le fibre ottiche.

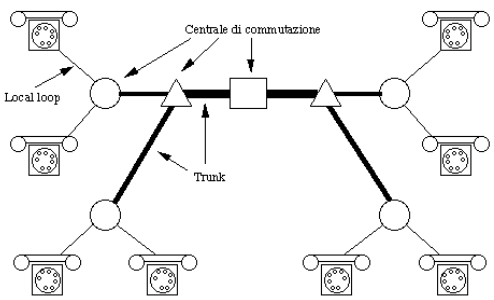
A titolo di esempio, si consideri la seguente tabella:

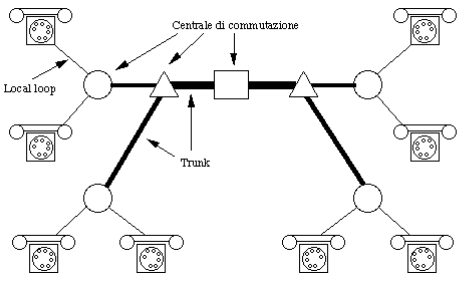
	Data rate	**Tasso di errore**
Cavo fra 2 computer	10^7 - 10^8 bps	1 su 10^{12} - 10^{13}
Linea telefonica	10^4 - 10^5 bps	1 su 10^5

Poiché gli uffici di commutazione nascevano come funghi, si ripropose lo stesso problema per il loro collegamento. Quindi vennero creati gli uffici di commutazione di secondo livello, e poi di terzo; alla fine la gerarchia si arrestò su cinque livelli (1890).

Tale tipo di struttura gerarchica è anche oggi alla base dei sistemi telefonici in tutto il mondo, con variazioni legate essenzialmente alle dimensioni dei vari sistemi. Attualmente ogni sistema telefonico è organizzato in una ***gerarchia multilivello*** con elevata ridondanza.

Nascita sistema telefonico





Struttura gerarchica del sistema telefonico

Al posto degli operatori vi sono delle ***centrali di commutazione***, una volta elettromeccaniche ed oggi quasi tutte digitali.

Il ***local loop***, cioè il collegamento dal telefono alla più vicina centrale di commutazione, è ancora oggi basato su doppino telefonico e può avere una lunghezza da 1 a 10 km. Trasporta un segnale analogico dotato di una banda molto modesta (3 kHz).

Per le altre connessioni (***trunk***) si usano molti altri mezzi:

- cavi coassiali;
- microonde;
- fibre ottiche, ormai molto diffuse.

Ormai quasi ovunque le centrali di commutazioni sono digitali e le linee che le collegano trasportano segnali digitali. I vantaggi principali sono i seguenti:

- è più facile ricostruire periodicamente il segnale senza introdurre errori (solo pochi valori);
- è più facile mescolare voce, dati, video e altri tipi di traffico;
- sono possibili data rate più alti usando le linee esistenti.

Il local loop trasporta un segnale analogico con una larghezza di banda di 3 kHz (0-3 kHz). Dunque, per trasmettere dati digitali, essi devono essere trasformati in analogici da un'apparecchio detto ***modem***. Quindi vengono ritrasformati in digitali nella centralina di commutazione da un apparecchio detto ***codec***, (cosa che succede anche alle conversazioni telefoniche), e quindi subiscono le conversioni inverse sul local loop di destinazione.

Ricordiamo che:

- le linee di trasmissione inducono attenuazione e distorsione (ed in più, specialmente sul local loop, sono soggette a disturbi);
- la trasmissione digitale genera onde quadre, che hanno un'ampio spettro di frequenze.

Quindi, se si trasmette un segnale digitale sul local loop, a causa della banda ridotta si deve usare una bassissima velocità di trasmissione. Per evitare questo inconveniente, si usa un segnale sinusoidale (quindi analogico) nella banda fra 1 e 2 kHz, detto ***portante***, che viene opportunamente modulato (variando nel tempo le sue caratteristiche) per trasmettere le informazioni.

Le principali tecniche di modulazione sono le seguenti:

- ***modulazione di ampiezza***: si varia l'ampiezza;
- ***modulazione di frequenza***: si varia la frequenza;
- ***modulazione di fase***: si varia la fase (cioè il "ritardo" rispetto al segnale originale).

Il modem accetta in ingresso un segnale digitale e produce in uscita una portante analogica opportunamente modulata. Ora, poiché la banda passante (e quindi la velocità di segnalazione) è limitata a 3 kHz, sappiamo che non si possono trasmettere più di 6 Kbps (per il teorema di Nyquist) se il segnale è a due valori. Per raggiungere velocità superiori si deve riuscire ad aumentare il numero dei possibili valori trasmessi. Ciò si ottiene usando in modo combinato le tecniche di modulazione sopra viste.

Ad esempio, modulando opportunamente sia ampiezza che fase si possono rappresentare 16 valori

diversi, quindi si possono ottenere 4 bit per baud. Dunque, su una linea a 2.400 baud (tipici del local loop), si può trasmettere alla velocità di 9.600 bps.

I diagrammi che definiscono i punti (nello spazio a coordinate polari ampiezza - fase) corrispondenti a valori validi del segnale da trasmettere si chiamano ***constellation pattern***. Quello sopra citato è definito nello standard ***V.32***, emesso da ITU. E' un esempio di standard per il livello fisico.

Un altro constellation pattern è definito nello standard ***V.32 bis***, per velocità di 14.400 bps su linea a 2.400 baud. Esso utilizza 64 punti per trasmettere 6 bit per baud.

Infine, il ***V.34*** viaggia a 28.800 bps, quindi trasmette 12 bit per baud. Si deve notare che con questi due standard possono sorgere problemi se la linea telefonica è più rumorosa del normale (teorema di Shannon).

Un'ulteriore modo per aumentare le prestazioni è ricorrere a meccanismi di ***compressione dei dati*** prima di trasmetterli. In tal modo, a parità di velocità, si inviano più informazioni.

Due standard importanti per la compressione dei dati sono :

V.42 bis, emesso da ITU;

MNP 5, standard de facto (Microcom Network Protocol).

Infine, per consentire una trasmissione contemporanea nei due sensi (***full-duplex***), ci sono due tecniche :

- suddividere la banda in due sottobande, una per ogni direzione; così però si dimezza la velocità. Questa tecnica è tipica degli standard per le velocità più basse, ad esempio V.21 per 300 bps;
- cancellazione dell'eco: ogni modem sfrutta l'intera banda e cancella in ricezione gli effetti della propria trasmissione.

Altri esempi di protocollo per il livello fisico sono quelli che stabiliscono le caratteristiche dell'interfaccia fra elaboratori (***DTE***, ***Data Terminal Equipment***) e modem (***DCE***, ***Data Circuit-terminating Equipment***), in termini:

- meccanici;
- elettrici;
- funzionali;
- procedurali.

Ad esempio, lo standard ***RS-232-C*** ed il molto simile ***V.24*** del CCITT caratterizzano:

- specifiche meccaniche: connettore a 25 pin con tutte le dimensioni specificate;
- specifiche elettrice:
    - valore 1 corrisponde a un segnale minore di -3 Volt;
    - valore 0 corrisponde a un segnale maggiore di +4 Volt;
    - data rate fino a 20 Kbps, lunghezza fino a 20 metri;
- specifiche funzionali: quali circuiti (trasmit, receive, clear to send, ecc.) sono collegati a

quali pin;
- specifiche procedurali: costituiscono il protocollo nel vero senso della parola, cioè le coppie azione/reazione fra DTE e DCE.

Vi sono molte attività di sperimentazione e tentativi di standardizzazione per superare il collo di bottiglia del local loop.

Tutte però richiedono :

- eliminazione del filtro a 3 kHz;
- local loop non troppo lunghi e con doppino di buona qualità (quindi, non tutti gli utenti potranno usufruirne).

Le principali proposte sono le seguenti:

- ***cable modem*** (velocità di 30 Mbps): si connette l'elaboratore al cavo coassiale fornito da un gestore di Cable TV. Il problema principale è che il cavo viene condiviso da molti utenti.
- ***ASDL*** (***Asymmetric Digital Subscriber Line***): velocità in ingresso 9 Mbps, in uscita 640 Kbps. Fornisce in casa un attacco ad alta velocità verso il sistema telefonico, ottenuto eliminando i filtri e usando più di un doppino. La banda non è condivisa con altri utenti.

Reti il sottolivello MAC

le reti sono divise in due categorie: ***punto a punto*** e ***broadcast***.

Nelle reti broadcast il problema principale è decidere quale elaboratore (detto anche ***stazione***) ha diritto di usare il mezzo trasmissivo quando c'è competizione (qui non si può alzare la mano per chiedere la parola!). Si deve evitare che molte stazioni trasmettano contemporaneamente, perché i relativi segnali si disturberebbero a vicenda.

I protocolli per decidere chi è il prossimo a trasmettere su un canale broadcast (detto anche ***multiaccess channel*** o ***random access channel***) appartengono ad un sottolivello del livello data link, detto ***sottolivello MAC***.

Essi sono usati sopratutto nelle LAN, ma anche nelle parti di WAN basate su satelliti.

Il problema principale è come allocare il canale ai vari utenti in competizione. Ci sono due meccanismi fondamentali:

- ***allocazione statica***, che viene decisa in anticipo;
- ***allocazione dinamica***, che si adatta alle esigenze di ogni momento.

L'allocazione statica prevede la suddivisione del canale fra gli N utenti, ciascuno dei quali riceve di conseguenza una frazione della banda totale. Si può fare, ad esempio, con tecniche quali FDM, allocando a ciascun utente una banda di frequenze distinta da quella degli altri utenti. Ciò va bene se

il numero di utenti non varia rapidamente e se tutti trasmettono con un data rate più o meno costante, però in genere comporta vari problemi:

- si verifica uno spreco di banda quando uno o più utenti non trasmettono;
- poiché il traffico è in generale molto ***bursty***, i picchi che si verificano non possono essere gestiti solamente con la sottobanda allocata.

Viceversa, l'allocazione dinamica cerca di adeguarsi alle esigenze trasmissive, in modo da soddisfarle al meglio. Ci sono alcune assunzioni da fare:

1. ***modello a stazioni***: ci sono N stazioni indipendenti, ognuna delle quali genera nuovi frame per la trasmissione. La probabilità di generare un frame in un intervallo di tempo T è uguale a pT, dove p è una costante e rappresenta il tasso d'arrivo dei nuovi frame. Quando un frame è generato, la stazione si blocca finché esso non è trasmesso;
2. ***singolo canale***: un singolo canale, e null'altro, è disponibile per le comunicazioni; tutte le stazioni vi possono trasmettere e da esso possono ricevere, e tutte sono ad uguale livello;
3. ***collisioni***: se due frame vengono trasmessi contemporaneamente, si sovrappongono ed il segnale risultante è rovinato (si verifica collisione):
    - tutte le stazioni possono rilevare la collisione;
    - i frame devono essere ritrasmessi;
    - non ci sono altri tipi di errori;
4. ***tempo***: può essere gestito in due modi:
    - ***continuous time***: la trasmissione di un frame può iniziare in un qualunque istante;
    - ***slotted time***: il tempo è diviso in intervalli discreti (***slot***). Uno slot può contenere 0, 1 oppure più di un frame. Ciò corrisponde ad uno slot vuoto, ad uno slot con un frame e ad uno slot in cui si verifica una collisione. La trasmissione può iniziare solo all'inizio di uno slot;
5. ***ascolto del canale***: ci sono due possibilità,
    - ***carrier sense*** (tipico delle LAN): le stazioni, prima di trasmettere, ascoltano il canale; se è occupato non cercano di trasmettere;
    - ***no carrier sense*** (tipico dei canali via satellite, nei quali vi è un elevato round trip time): le stazioni non ascoltano, trasmettono senz'altro; si preoccuperanno dopo di vedere se c'è stata una collisione.

Reti il protocollo ALOHA

Nacque negli anni '70 per collegare tra loro, tramite radio al suolo, gli elaboratori sparsi nelle isole Hawaii.

Esistono due versioni, ***Pure Aloha*** e ***Slotted Aloha***.

Nel ***Pure Aloha*** le stazioni trasmettono quando vogliono, però durante la trasmissione ascoltano il

canale e confrontano ciò che ricevono con ciò che hanno spedito.

Dunque, se si verifica una collisione se ne accorgono, e in tal caso, dopo aver lasciato passare una quantità di tempo casuale, ritrasmettono il frame. La scelta di attendere per una quantità di tempo casuale discende dal fatto che altrimenti una collisione ne ricrea infinite altre.

Qual'è l'efficenza dello schema Aloha puro, in queste circostanze caotiche?

Definiamo come ***frame time*** il tempo necessario alla trasmissione di un frame, che ha lunghezza fissa. Supponiamo che vengano complessivamente generati dei frame con una distribuzione di Poisson avente media di S frame per frame time.

Ovviamente, se S >= 1, ci saranno quasi sempre collisioni. Per un throughput ragionevole ci aspettiamo 0 < S < 1. Purtroppo, oltre ai frame nuovi, ci sono anche quelli relativi alla ritrasmissione causata da collisioni precedenti.

Supponiamo che la distribuzione di tutti i frame (vecchi e nuovi) sia anch'essa di Poisson, con valor medio pari a G frame per frame time.

A basso carico ci aspettiamo poche collisioni, quindi G è circa uguale ad S. Ad alto carico invece avremo più collisioni, per cui G sarà maggiore di S.

In ogni caso, sotto qualunque condizione di carico il ***throughput*** (cioé la ***quantità di pacchetti che arrivano a destinazione***) è uguale al carico offerto moltiplicato per la probabilità che la trasmissione abbia successo, ossia:

Throughput = G*P(0)

dove P(0) è la probabilità che un frame non soffra collisioni.

Reti il protocollo CSMA

Nelle reti locali le stazioni possono ascoltare il canale e regolarsi di conseguenza, ottenendo un'efficienza molto più alta. I protocolli nei quali le stazioni ascoltano il canale prima di iniziare a trasmettere si dicono ***carrier sense***.

Ci sono vari tipi di protocolli carrier sense:

- ***1-persistent***
    - Quando una stazione deve trasmettere, ascolta il canale:
        - se è occupato, aspetta finché si libera e quindi trasmette;
        - se è libero, trasmette (con probabilità 1, da cui il nome).
    - Se avviene una collisione, la stazione aspetta un tempo random e riprova tutto da capo.
    - Problemi:
        - una stazione A trasmette, e prima che il suo segnale arrivi a B anche B inizia a

trasmette, dunque si verifica una collisione. Più alto è il tempo di propagazione fra A e B e più grave è il fenomeno;
        - A e B ascoltano contemporaneamente durante la trasmissione di C, e non appena quest'ultima termina iniziano entrambe a trasmettere: anche in questo caso si verifica una collisione.
- ***Nonpersistent***
    - Quando una stazione deve trasmettere, ascolta il canale:
        - se è occupato, invece di trasmettere non appena si libera come in 1-persistent la stazione aspetta comunque un tempo random e ripete tutto il procedimento da capo;
        - se è libero, si comporta come in 1-persistent.
    - Intuitivamente, ci si aspettano maggiori ritardi prima di riuscire a trasmettere un frame e meno collisioni rispetto a 1-persistent.
- ***P-persistent*** (si applica a canali slotted)
    - Quando una stazione deve trasmettere, ascolta il canale:
        - se è occupato, aspetta il prossimo slot e ricomincia da capo;
        - se è libero:
            - con probabilità p trasmette subito;
            - con probabilità 1 - p aspetta il prossimo slot; se anch'esso è libero, riapplica tale procedimento;
    - Il processo si ripete finché:
        - il frame è trasmesso, oppure
        - qualcun altro ha iniziato a trasmettere. In questo caso la stazione si comporta come in una collisione: aspetta un tempo random e ricomincia da capo.
    - Intuitivamente, al diminuire di p ci si aspettano crescenti ritardi prima di riuscire a trasmettere un frame ed una progressiva diminuzione delle collisioni.

Reti il protocollo CSMA-CD

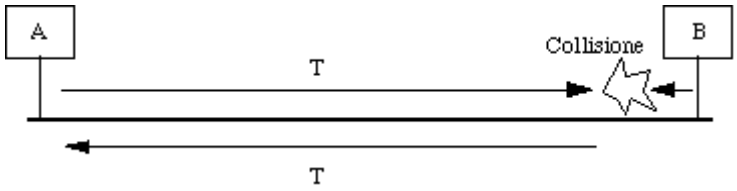
Un ulteriore miglioramento si ha se le stazioni interrompono la loro trasmissione non appena rilevano una collisione, invece di portarla a termine.

Rilevare la collisione è un processo analogico: si ascolta il canale durante la propria trasmissione, e se la potenza del segnale ricevuto è superiore a quella trasmessa si scopre la collisione.

Quando si verifica una collisione, la stazione aspetta una quantità casuale di tempo e riprova a trasmettere.

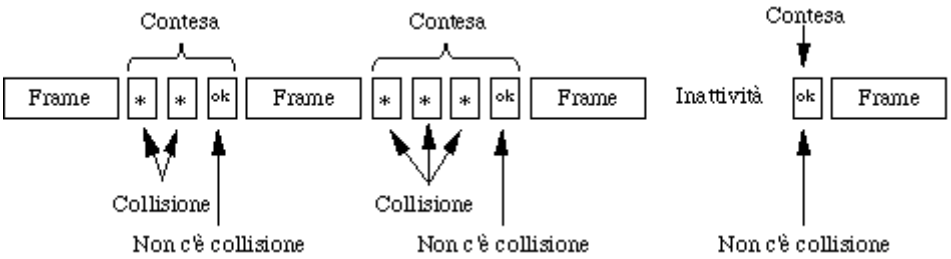
Posto uguale a T il tempo di propagazione del segnale da un capo all'altro della rete, è necessario che trascorra un tempo pari a 2T perché una stazione possa essere sicura di rilevare una collisione.

Infatti, se una stazione A posta ad una estremità della rete inizia a trasmettere al tempo t_0, il suo segnale arriva a B (posta all'altra estremità della rete) dopo al tempo $t_0 + T$; se un attimo prima di tale istante anche B inizia a trasmettere, la collisione conseguente viene rilevata da B quasi immediatamente, ma impiega una ulteriore quantità T di tempo per giungere ad A, che la può quindi rilevare solo un attimo prima dell'istante $t_0 + 2T$.



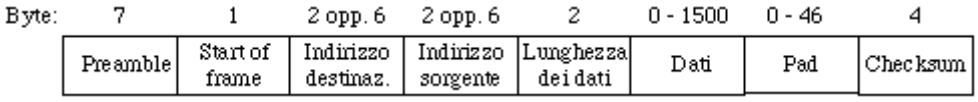
Il modello concettuale che si utilizza è il seguente:

- vi è un'alternanza di periodi di ***contesa***, di ***trasmissione*** e di ***inattività***;
- il periodo di contesa è modellato come uno Slotted Aloha con slot di durata 2T: a titolo di esempio, per un cavo di 1 km T vale circa 5 microsecondi.



Reti fast ethernet

La struttura di un frame 802.3 è la seguente:



I campi del frame hanno le seguenti funzioni:

Preamble	7 byte tutti uguali a 10101010. Producono, a 10 Mbps, un'onda quadra a 10 Mhz per 5,6 microsecondi, che consente al ricevitore di sincronizzare il suo clock con quello del trasmettitore.
Start of frame	un byte delimitatore, uguale a 10101011.
Indirizzi	gli indirizzi usati sono sempre a 6 byte, e sono univoci a livello mondiale (sono cablati dentro l'interfaccia). E' possibile specificare un singolo destinatario, un gruppo di destinatari (multicast) oppure un invio in broadcast a tutte le stazioni (indirizzo costituito da una sequenza di uni).
Lunghezza dei dati	indica quanti byte ci sono nel campo dati (da 0 a 1500).
Dati	contiene il payload del livello superiore.
Pad	Se il frame (esclusi preambolo e delimiter) è più corto di 64 byte, con questo campo lo si porta alla lunghezza di 64 byte, vedremo poi perché.
Checksum	è un codice CRC come quelli già visti.

Nessun livello MAC garantisce un servizio affidabile. Ciò è dettato dal fatto che, visto il bassissimo tasso d'errore delle LAN, si preferisce un protocollo datagram ad alte prestazioni.

Vediamo ora perché esiste un limite minimo di 64 byte per la lunghezza di un frame.

Abbiamo già visto che, perché una collisione possa essere certamente rilevata da chi trasmette, deve passare un tempo non inferiore a due volte il tempo di attraversamento dell'intera rete.

Nel caso di IEEE 802.3, che prevede 2,5 km di lunghezza massima totale e l'interposizione di un massimo di quattro ripetitori, si ha che il tempo massimo di attraversamento dell'intera rete moltiplicato per due è pari a 57,6 microsecondi.

Ora, è essenziale che la collisione venga rilevata durante la trasmissione e non dopo, altrimenti il mittente dedurrà erroneamente che la sua trasmissione è andata a buon fine.

Dunque, la trasmissione di un frame non deve durare meno di 57,6 microsecondi, che sono il tempo necessario per trasmettere (a 10 Mbps) proprio 72 byte (e cioé 576 bit, ciascuno dei quali viene trasmesso in un decimo di microsecondo). Dunque, il frame non può essere costituito da meno di 72 byte, 8 dei quali sono costituiti dal preambolo e dal delimitatore, e 64 dal resto del frame.

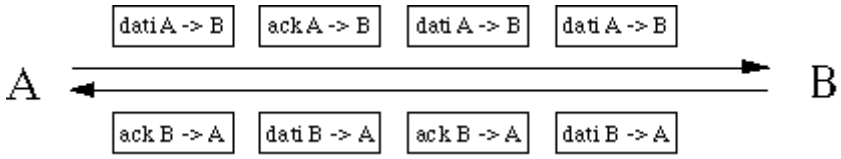
Si noti che se si vuole aumentare la velocità di un certo fattore, diciamo 10, si deve diminuire di 10 volte la lunghezza massima ammessa per la rete o aumentare di 10 volte la lunghezza minima del frame. Vedremo nel seguito come viene risolto il problema per il protocollo ***Fast Ethernet*** (100 Mbps).

Reti protocolli a finestra scorrevole

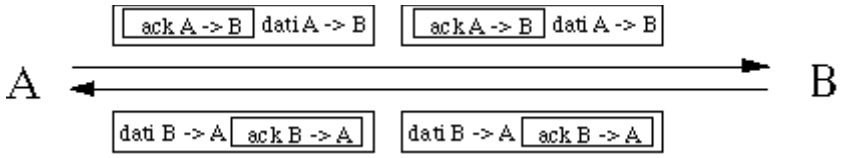
Volendo stabilire una comunicazione dati bidirezionale è necessario disporre di due circuiti, il che ovviamente è uno spreco di risorse. Un'idea migliore è usare un solo circuito, nel quale far convivere tutte le esigenze.

Supponendo che la comunicazione sia fra A e B, si avrà che:

- nella direzione da A a B viaggiano i frame dati inviati da A a B e i frame di ack inviati da A a B (in risposta ai frame dati inviati da B ad A);
- nella direzione da B a A viaggiano i frame dati inviati da B a A e i frame di ack inviati da B a A (in risposta ai frame dati inviati da A ad B);
- il campo ***kind*** serve a distinguere fra i due tipi di frame, dati e di ack, che viaggiano nella stessa direzione.



Però c'è un'idea ancora migliore: se, quando si deve inviare un ack da B ad A, si aspetta un pò fin che è pronto un frame dati che B deve inviare ad A, si può "fare autostop" e mettere dentro tale frame dati anche le informazioni relative all'ack in questione. Questa tecnica si chiama ***piggybacking*** (letteralmente, portare a spalle).



Il campo ***ack*** serve proprio a questo scopo, infatti è il campo in cui viene trasportato, se c'è, un ack.

Questa tecnica consente un notevole risparmio di:

- banda utilizzata;
- uso di cpu.

Infatti, con questa tecnica le informazioni di ack non richiedono la costruzione di un apposito frame (e quindi il tempo necessario alla creazione ed al riempimento della struttura, al calcolo del checksum, ecc.) né la sua trasmissione (e quindi l'uso di banda).

Però c'è un aspetto da non trascurare: per quanto si può aspettare un frame su cui trasportare un ack che è pronto e deve essere inviato? Non troppo, perché se l'ack non arriva in tempo il mittente ritrasmetterà il frame anche se ciò non è necessario. Dunque si stabilisce un limite al tempo di attesa di un frame sul quale trasportare l'ack; trascorso tale tempo si crea un frame apposito nel quale si

mette l'ack.

I protocolli che vedremo ora appartengono alla classe dei ***protocolli sliding window*** (finestra scorrevole), sono full-duplex (per i dati), sfruttano il piggybacking e sono più robusti di quelli precedenti.

Differiscono fra loro per efficienza, complessità e capacità dei buffer. Alcuni aspetti però sono comuni a tutti gli algoritmi:

- ogni frame inviato ha un numero di sequenza, da 0 a 2^n-1 (il campo ***seq*** è costituito da n bit);
- ad ogni istante il mittente mantiene una finestra scorrevole sugli indici dei frame, e solo quelli entro la finestra possono essere trasmessi. I numeri di sequenza entro la finestra rappresentano frame da spedire o spediti, ma non ancora confermati:
    - quando arriva dal livello network un pacchetto, un nuovo indice entra nella finestra;
    - quando arriva un ack, il corrispondente indice esce dalla finestra;
    - i frame dentro la finestra devono essere mantenuti in memoria per la possibile ritrasmissione; se il buffer è pieno, il livello data link deve costringere il livello network a sospendere la consegna di pacchetti;
- analogamente, il destinatario mantiene una finestra corrispondente agli indici dei frame che possono essere accettati:
    - se arriva un frame il cui indice è fuori dalla finestra:
        - il frame viene scartato (e non si invia il relativo ack);
    - se arriva un frame il cui indice è entro la finestra:
        - il frame viene accettato;
        - viene spedito il relativo ack;
        - la finestra viene spostata in avanti;
    - si noti che la finestra del destinatario rimane sempre della stessa dimensione, e se essa è pari a 1 il livello accetta i frame solo nell'ordine giusto (ma per dimensioni maggiori di 1 questo non è più detto).
- le finestre di mittente e destinatario non devono necessariamente avere uguali dimensioni, né uguali limiti inferiori o superiori.

  Nella figura segunete si osserva Finestra scorrevole sugli indici dei frame

In questi protocolli il livello data link ha più libertà nell'ordine di trasmissione, fermo restando che:

- i pacchetti devono essere riconsegnati al livello network nello stesso ordine di partenza;
- il canale fisico è ***wire-like***, cioé consegna i frame nell'ordine di partenza.

Protocollo a finestra scorrevole di un bit

In questo protocollo sia mittente che destinatario usano una finestra scorrevole di dimensione uno. Di fatto questo è un protocollo stop-and-wait.

Non c'è differenza di comportamento fra mittente e destinatario, tutt'e due usano lo stesso codice (questo vale anche per i protocolli successivi).

Il funzionamento, molto semplice, è il seguente:

- il mittente, quando invia un frame, fa partire un timer:
  - se prima che scada il timer arriva un ack con lo stesso numero di sequenza del frame che si sta cercando di trasmettere, si avanza la finestra e si passa a trasmettere il frame successivo;
  - se arriva un ack diverso o scade il timer, si ritrasmette il frame;
- il destinatario invece:
  - quando arriva un frame corretto, senza errori, invia un ack col corrispondente numero di sequenza;
  - se il frame non è un duplicato lo passa al livello network e avanza la finestra.

Qui sta la principale novità rispetto al protocollo 3: l'ack è etichettato col numero di sequenza del frame a cui si riferisce. I valori dell'etichetta possono solo essere 0 e 1, come nel protocollo 3.

Il peggio che può succedere è la ritrasmissione inutile di qualche frame, ma questo protocollo è sicuro.

Reti fast ethernet

FAST ethernet

Questo standard (803.2u), approvato nel 1995, prevede l'aumento di velocità di un fattore 10, da 10 Mbps a 100 Mbps.

Come si risolve il problema del minimo tempo di trasmissione e/o della massima lunghezza della rete? In modo diverso a seconda del supporto fisico utilizzato:

- Doppino classe 3 (100BaseT4)
  - si usano quattro doppini fra l'hub ed ogni stazione:
    - uno viene usato sempre per il traffico dall'hub alla stazione;
    - uno viene usato sempre per il traffico dalla stazione all'hub;
    - 2 vengono usati di volta in volta nella direzione della trasmissione in corso;
  - la codifica è ***8B6T***, cioé 8 bit vengono codificati con 6 ***trit*** (che hanno valore 0, 1 o 2);
  - la velocità di segnalazione è 25 Mhz (solo 25% in più di quella dello standard 802.3, che è di 20 Mhz);

        - si inviano 3 trit sui 3 doppini contemporaneamente a 25 Mhz, ossia 6 trit alla frequenza di 12,5 Mhz. Poiché 6 trit convogliano 8 bit, di fatto si inviano 8 bit a 12,5 Mhz, ottenendo così i 100 Mbps.
    - Doppino classe 5 (100BaseT)
        - velocità di segnalazione 124 Mhz;
        - codifica ***4B5B*** (4 bit codificati con 5 bit, introducendo ridondanza);
        - a seconda del tipo di hub:
            - hub tradizionale: la lunghezza massima di un ramo è 100 metri, quindi il diametro della rete è 200 metri (contro i 2,5 km di 802.3).
            - ***switched hub***: ogni ramo è un dominio di collisione separato, e quindi (poiché su esso vi è una sola stazione) non esiste più il problema delle collisioni, ma rimane il limite di 100 metri per i limiti di banda passante del doppino.
    - Fibra ottica (100BaseFX)
        - velocità di segnalazione 125 Mhz;
        - codifica 4B5B;
        - obbligatorio switched hub;
        - lunghezza rami fino a 2 km (con uno switched hub non c'è il problema delle collisioni, ed inoltre come sappiamo la fibra regge velocità dell'ordine dei Gbps a distanze anche superiori).

Reti 802.3

Il protocollo 802.3 è un CSMA/CD di tipo 1-persistent:

- prima di trasmettere, la stazione aspetta che il canale sia libero;
- appena è libero inizia a trasmettere;
- se c'è una collisione, la circuiteria contenuta nel transceiver invia una sequenza di jamming di 32 bit, per avvisare le altre stazioni;
- se la trasmissione non riesce, la stazione attende una quantità di tempo casuale e poi riprova.

La quantità di tempo che si lascia passare è regolata da un apposito algoritmo, il ***binary backoff exponential algorithm***:

- dopo una collisione, il tempo si considera discretizzato (slotted) con uno ***slot time*** pari a 51,2 microsecondi (corrispondenti al tempo di trasmissione di 512 bit, ossia 64 byte, pari alla lunghezza minima di un frame senza contare il preambolo ed il delimiter);
- il tempo di attesa prima della prossima ritrasmissione è un multiplo intero dello slot time, e viene scelto a caso in un intervallo i cui estremi dipendono da quante collisioni sono avvenute;
- dopo n collisioni, il numero r di slot time da lasciar passare è scelto a caso nell'intervallo $0 <= r <= 2^k-1$, con $k = \min(n, 10)$;
- dopo 16 collisioni si rinuncia (inviando un messaggio di errore al livello superiore).

La crescita esponenziale dell'intervallo garantisce una buona adattabilità ad un numero variabile di stazioni, infatti:

- se il range fosse sempre piccolo, con molte stazioni si avrebbero praticamente sempre collisioni;
- se il range fosse sempre grande, non ci sarebbero quasi mai collisioni ma il ritardo medio (metà range*slot time) causato da una collisione sarebbe molto elevato.

Le prestazioni osservate sono molto buone, migliori di quelle stimabili in via teorica.

Peraltro, queste ultime sono fortemente influenzate dal modello di traffico che si assume. Di solito lo si assume poissoniano, ma in realtà è bursty e per di più ***self similar***, ossia il suo andamento su un lungo periodo è simile a quello su un breve periodo, ricordando in questo le caratteristiche dei frattali.

La pratica ha mostrato che 802.3:

- può sopportare un carico medio del 30% (3 Mbps) con picchi del 60% (6 Mbps);
- sotto carico medio:
    - il 2-3% dei pacchetti ha una collisione;
    - qualche pacchetto su 10.000 ha più di una collisione.

Reti PROTOCOLLI HEAVEN E SIMPLEX

Protovollo heaven

Questo protocollo, per canale simplex, è molto semplice ed è basato sulle ipotesi (non realistiche) che:

- i frame dati vengono trasmessi in una sola direzione;
- le peer entity di livello network sono sempre pronte (non devono mai attendere per inviare o ricevere al/dal livello data link);
- si ignora il tempo di elaborazione del livello data link;
- c'è spazio infinito per il buffering nel ricevitore;
- il canale fisico non fa mai errori.

Il protocollo consiste di due procedure, relative rispettivamente al mittente e al destinatario.

Mittente (loop infinito):

```
1) attende un pacchetto dal livello network;
2) costruisce un frame dati;
3) passa il frame al livello fisico;
4) torna ad 1).
```

Destinatario (loop infinito):

```
1) attende evento:
     * arriva frame da livello fisico:
         2) estrae pacchetto;
         3) lo passa al livello network;
         4) torna ad 1).
```

Protocollo SIMPLEX STOP AND WAIT

Rilasciamo l'ipotesi (poco realistica) che esista un buffer infinito nel ricevitore. Tutte le altre rimangono, incluso il fatto che il traffico dati viaggia in una sola direzione. La novità è che il mittente deve essere opportunamente rallentato. Ciò però non si può fare con ritardi prefissati: sarebbe troppo gravoso, perché questi dovrebbero essere calibrati sul caso pessimo, che non sempre si verificherà.

La soluzione consiste nell'invio, da parte del destinatario, di una esplicita autorizzazione all'invio del prossimo frame. Questo tipo di protocolli, nei quali il mittente attende un OK dal destinatario, si chiamano ***stop and wait***.

Mittente (loop infinito):

```
1) attende un pacchetto dal livello network;
2) costruisce un frame dati;
3) passa il frame al livello fisico;
4) attende evento:
     * arriva frame di ack (vuoto):
         5) torna ad 1).
```

Destinatario (loop infinito):

```
1) attende evento:
     * arriva frame dati da livello fisico:
         2) estrae il pacchetto;
         3) consegna il pacchetto al livello network;
         4) invia un frame di ack (vuoto) al mittente;
         5) torna ad 1).
```

Si noti che, sebbene il traffico dati viaggi in una sola direzione, i frame viaggiano in entrambe, dunque ci vuole un canale almeno half-duplex (c'è alternanza stretta nelle due direzioni).

Assumiamo ora che il canale possa fare errori. I frame (dati o di ack) possono essere danneggiati o persi completamente. Se un frame arriva danneggiato, l'HW di controllo del checksum se ne accorge e informa il SW di livello data link; se il frame si perde del tutto, ovviamente, la cosa non si rileva.

L'aggiunta di un timer al protocollo 2) può bastare? Cioé, è adeguato uno schema quale il seguente?

- quando il mittente invia un frame dati, fa anche partire un timer; se non arriva l'ack entro la scadenza del timer, invia nuovamente il frame;

- il destinatario invia un ack quando un frame dati arriva senza errori;

Questo semplice schema in realtà non basta, infatti può verificarsi la seguente sequenza di eventi:

1. il frame dati x arriva bene;
2. l'ack del frame x si perde completamente (gli errori non discriminano tra frame dati e frame di ack);
3. il frame dati x viene inviato di nuovo e arriva bene;
4. il livello network di destinazione riceve due copie di x, errore!

E' necessario che il destinatario possa riconoscere gli eventuali doppioni. Ciò si ottiene sfruttando il campo ***seq*** dell'header, dove il mittente mette il numero di sequenza del frame dati inviato.

Notiamo che basta un bit per il numero di sequenza, poiché l'unica ambiguità in ricezione è tra un frame ed il suo immediato successore (siano ancora in stop and wait): infatti, fino a che un frame non viene confermato, è sempre lui ad essere ritrasmesso, altrimenti è il suo successore.

Dunque, sia mittente che destinatario useranno, come valori per i numeri di sequenza, la successione

```
...01010101...
```

Il mittente trasmette i frame dati alternando zero ed uno nel campo seq; passa a trasmettere il prossimo frame solo quando riceve l'ack di quello precedente.

Il destinatario invia un frame di ack per tutti quelli ricevuti senza errori, ma passa al livello network solo quelli con il numero di sequenza atteso.

Per quanto riguarda le possibilità che i frame dati arrivino rovinati o non arrivino affatto, un meccanismo basato su timer va bene, bisogna però che esso sia regolato in modo da permettere sicuramente l'arrivo dell'ack, pena la ritrasmissione errata di un duplicato del frame, che può creare grossi guai (vedremo in seguito).

Protocolli come questo, in cui il mittente aspetta un ack di conferma prima di trasmettere il prossimo frame, si chiamano ***PAR (Positive Ack with Retransmission)*** o ***ARQ (Automatic Repeat Request)***.

Mittente (loop infinito; [seq] rappresenta il campo seq di un frame):

```
0) n_seq = 0;
1) n_seq = 1 - n_seq;
2) attende un pacchetto dal livello network;
3) costruisce frame dati e copia n_seq in [seq];
4) passa il frame dati al livello fisico;
5) resetta il timer;
6) attende un evento:
     * timer scaduto: torna a 4)
     * arriva frame di ack (vuoto) non valido: torna a 4)
     * arriva frame di ack (vuoto) valido: torna ad 1)
```

Destinatario (loop infinito; [seq] rappresenta il campo seq di un frame):

```
0) n_exp = 1;
1) attende evento;
     * arriva frame dati valido da livello fisico:
```

```
        2) se ([seq] == n_exp)
           2.1) estrae pacchetto
           2.2) lo consegna al livello network
           2.3) n_exp = 1 - n_exp
        3) invia frame di ack (vuoto)
        4) torna ad 1)
    * arriva frame non valido: torna ad 1)
```

Reti livello 5 e posta elettronica

La ***posta elettronica*** è uno dei servizi più consolidati ed usati nelle reti. In Internet è in uso da circa 20 anni, e prima del WWW era senza dubbio il servizio più utilizzato.

Un servizio di posta elettronica, nel suo complesso, consente di effettuare le seguenti operazioni:

- comporre un messaggio;
- spedire il messaggio (a uno o più destinatari);
- ricevere messaggi da altri utenti;
- leggere i messaggi ricevuti;
- stampare, memorizzare, eliminare i messaggi spediti o ricevuti.

Di norma, un messaggio ha un formato ben preciso. In Internet un messaggio ha un formato (definito nell'RFC 822) costituito da un ***header*** e da un ***body***, separati da una linea vuota.

Lo header è a sua volta costituito da una serie di linee, ciascuna relativa a una specifica informazione (identificata da una parola chiave che è la prima sulla linea); alcune informazioni sono:

`To`	indirizzo di uno o più destinatari.
`From`	indirizzo del mittente.
`Cc`	indirizzo di uno o più destinatari a cui si invia per conoscenza.
`Bcc`	blind Cc: gli altri destinatari non sanno che anche lui riceve il messaggio.
`Subject`	argomento del messaggio.
`Sender`	chi materialmente effettua l'invio (ad es. nome della segretaria).

Il body contiene il testo del messaggio, in caratteri ASCII. L'ultima riga contiene solo un punto, che identifica la fine del messaggio.

Gli indirizzi di posta elettronica in Internet hanno la forma:

username@hostname

dove username è una stringa di caratteri che identifica il destinatario, e hostname è un nome DNS oppure un indirizzo IP.

Ad esempio, bongiovanni@dsi.uniroma1.it è l'indirizzo di posta elettronica dell'autore di queste dispense.

La posta elettronica viene implementata in Internet attraverso la cooperazione di due tipi di sottosistemi:

- ***Mail User Agent*** (***MUA***);
- ***Mail Transport Agent*** (***MTA***).

Il primo permette all'utente finale di:

- comporre messaggi;
- consegnarli a un MTA per la trasmissione;
- ricevere e leggere messaggi;
- salvarli o eliminarli.

Il secondo si occupa di:

- trasportare i messaggi sulla rete, fino alla consegna a un MTA di destinazione;
- rispondere ai MUA dei vari utenti per consegnare loro la posta arrivata; in questa fase l'MTA richiede ad ogni utente una password per consentire l'accesso ai messaggi.

Corrispondentemente, sono definiti due protocolli principali per la posta elettronica:

- ***SMTP*** (***Simple Mail Transfer Protocol***, RFC 821) per il trasporto dei messaggi:
  - dal MUA di origine ad un MTA;
  - fra vari MTA, da quello di partenza fino a quello di destinazione;
- ***POP3*** (***Post Office Protocol*** versione 3, RFC 1225) per la consegna di un messaggio da parte di un MTA al MUA di destinazione.

Recentemente sono stati introdotti altri protocolli più sofisticati, quali ***IMAP*** (***Interactive Mail Acces Protocol***, RFC 1064) e ***DMSP*** (***Distributed Mail System Protocol***, RFC 1056), il cui supporto però non è ancora molto diffuso nel software disponibile agli utenti.

Come avviene la trasmissione di un messaggio? Supponiamo che l'utente

`pippo@topolinia.wd`

spedisca un messaggio a

`minnie@paperinia.wd`

e immaginiamo che:

- Pippo usi un MUA configurato per consegnare la posta ad un SMTP server in esecuzione sull'host `mailer.topolinia.wd`;
- Minnie abbia un MUA configurato per farsi consegnare la posta da un POP3 server in esecuzione sull'host `mailer.paperinia.wd`.

La sequenza di azioni che hanno luogo è la seguente:

1. Pippo compone il messaggio col suo MUA, che tipicamente è un programma in esecuzione su un PC in rete;
2. appena Pippo preme il pulsante SEND, il suo MUA:
    - interroga il DNS per sapere l'indirizzo IP dell'host `mailer.topolinia.wd`;
    - apre una connessione TCP ed effettua una conversazione SMTP con il server SMTP in esecuzione sull'host `mailer.topolinia.wd`, per mezzo della quale gli consegna il messaggio;
    - chiude la connessione TCP;
3. Pippo se ne va per i fatti suoi;
4. il server SMTP di `mailer.topolinia.wd`:
    - chiede al DNS l'indirizzo IP di `paperinia.wd`;
    - scopre che è quello dell'host `mailer.paperinia.wd`;
    - apre una connessione TCP e poi una conversazione SMTP con il server SMTP in esecuzione su quell'host e gli consegna il messaggio scritto da Pippo;
5. Minnie lancia il suo MUA;
6. appena Minnie preme il pulsante "check mail", il suo MUA:
    - interroga il DNS per avere l'indirizzo IP dell'host `mailer.paperinia.wd`;
    - apre una connessione TCP e poi una conversazione POP3 col server POP in esecuzione su `mailer.paperinia.wd` e preleva il messaggio di Pippo, che viene mostrato a Minnie.

Si noti che `topolinia.wd` e `paperinia.wd` in genere corrispondono a un dominio nel suo complesso e non ad un singolo host, al fine di rendere gli indirizzi di posta elettronica indipendenti da variazioni del numero, dei nomi logici e degli indirizzi IP degli host presenti nel dominio.

Nel DNS ci sono opportuni record detti di tipo ***MX*** (***Mail Exchange***), che si occupano di indicare quale host effetttivamente fa da server SMTP per un dominio.

Nel nostro esempio avremo, nel DNS server di Paperinia, i record:

```
mailer.paperinia.wd A     100.10.10.5
paperinia.wd        MX    mailer.paperinia.wd
```

Il secondo record è un record di tipo MX, e indica che tutta la posta in arrivo per

`chiunque@paperinia.wd`

deve essere consegnata all'host `mailer.paperinia.wd`, e cioé quello che ha l'indirizzo IP

`100.10.10.5`

Inoltre, non è detto che `mailer.topolinia.wd` consegni i messagi direttamente a `mailer.paperinia.wd`. E' possibile che le macchine siano configurate in modo da trasferire i messaggi attraverso un certo numero di server SMTP intermedi.

Infine, vanno citate due significative estensioni di funzionalità della posta elettronica, in via di progressiva diffusione:

- possibilità di inviare messaggi di posta contenenti informazioni di qualunque tipo (per esempio programmi eseguibili, immagini, filmati, suoni, ecc.) attraverso lo standard ***MIME*** (***Multipurpose Internet Mail Extension***, RFC 1341 e 1521);
- possibilità di inviare messaggi corredati di firma digitale o crittografati, attraverso lo standard in via di definizione ***S/MIME*** (***Secure/MIME***, RFC 1847).

Reti livello 4 Transport

Il livello transport è il cuore di tutta la gerarchia di protocolli. Il suo compito è di fornire un trasporto affidabile ed efficace dall'host di origine a quello di destinazione, indipendentemente dalla rete utilizzata.

Questo è il livello in cui si gestisce per la prima volta (dal basso verso l'alto) una conversazione diretta, cioè senza intermediari, fra sorgente e destinazione.

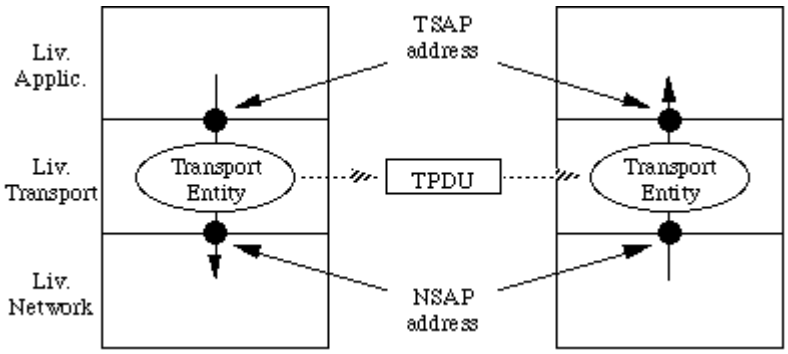
Da ciò discende che il software di livello transport è presente solo sugli host, e non nei router della subnet di comunicazione.

Servizi offerti dal livello transport

I servizi principali offerti ai livelli superiori sono vari tipi di trasporto delle informazioni fra una transport entity su un host e la sua peer entity su un altro host.

Naturalmente, tali servizi sono realizzati dal livello transport per mezzo dei servizi ad esso offerti dal livello network.

I servizi transport sono basati sui servizi network



Così come ci sono due tipi di servizi di livello network, ce ne sono due anche a livello transport:

- servizi affidabili orientati alla connessione (tipici di questo livello);
- servizi datagram (poco usati in questo livello).

Essi sono molto simili, come caratteristiche, a quelli corrispondenti del livello network, ed hanno gli analoghi vantaggi e svantaggi.

Ma allora, perché duplicare le cose in due diversi livelli? Il fatto è che l'utente (che accede ai servizi

di rete dall'alto) non ha alcun controllo sulla subnet di comunicazione, e vuole comunque certe garanzie di servizio (ad esempio, il trasferimento corretto di un file). Dunque, tali garanzie devono essere fornite al di fuori della subnet, per cui devono risiedere in un livello superiore a quello network. In sostanza, il livello transport permette di offrire un servizio più affidabile di quanto la subnet sia in grado di fare.

Inoltre, ha un altro importante scopo, quello di isolare i livelli superiori dai dettagli implementativi della subnet di comunicazione. Ottiene ciò offrendo un insieme di primitive (di definizione dei servizi) semplici da utilizzare ed indipendenti dai servizi dei livelli sottostanti. Questo perché mentre solo poche persone scrivono componenti software che usano i servizi di livello network, molte scrivono applicazioni di rete, che si basano sui servizi di livello transport.

Un ultimo aspetto riguarda la possibilità di specificare la ***QoS*** (***Quality of Service***) desiderata. Questo in particolare è adatto soprattutto ai servizi connection oriented, nei quali il richiedente può specificare esigenze quali:

- massimo ritardo per l'attivazione della connessione;
- throughput richiesto;
- massimo ritardo di transito ammesso;
- tasso d'errore tollerato;
- tipo di protezione da accessi non autorizzati ai dati in transito.

In questo scenario:

- le peer entity avviano una fase di negoziazione per mettersi d'accordo sulla QoS, anche in funzione della qualità dei servizi di livello network di cui dispongono;
- quando l'accordo è raggiunto, esso vale per tutta la durata della connessione.

Reti livello 4 i protocolli di livelloTransport e Internet

Il livello transport di Internet è basato su due protocolli:

- ***TCP*** (***Transmission Control Protocol***) RFC 793, 1122 e 1323;
- ***UDP*** (***User Data Protocol***) RFC 768.

Il secondo è di fatto IP con l'aggiunta di un breve header, e fornisce un servizio di trasporto datagram (quindi non affidabile). Lo vedremo brevemente nel seguito.

Il protocollo TCP è stato progettato per fornire un flusso di byte affidabile, da sorgente a destinazione, su una rete non affidabile.

Dunque, offre un servizio reliable e connection oriented, e si occupa di:

- accettare dati dal livello application;
- spezzarli in ***segment***, il nome usato per i TPDU (dimensione massima 64 Kbyte, tipicamente circa 1.500 byte);

- consegnarli al livello network, eventualmente ritrasmettendoli;
- ricevere segmenti dal livello network;
- rimetterli in ordine, eliminando buchi e doppioni;
- consegnare i dati, in ordine, al livello application.

E' un servizio full-duplex con gestione di ack e controllo del flusso.

servizi di TCP si ottengono creando connessione di livello transport identificata da una coppia di punti d'accesso detti ***socket***. Ogni socket ha un ***socket number*** che consiste della coppia:

IP address: Port number

Il socket number costituisce il TSAP.

I port number hanno 16 bit. Quelli minori di 256 sono i cosidetti ***well-known port***, riservati per i servizi standard. Ad esempio:

Port number	Servizio
7	Echo
20	Ftp (control)
21	Ftp (data)
23	Telnet
25	Smtp
80	Http
110	Pop versione 3

Poiché le connessioni TCP, che sono full duplex e point to point, sono identificate dalla coppia di socket number alle due estremità, è possibile che su un singolo host più connessioni siano attestate localmente sullo stesso socket number.

Le connessioni TCP trasportano un flusso di byte, non di messaggi: i confini fra messaggi non sono né definiti né preservati. Ad esempio, se il processo mittente (di livello application) invia 4 blocchi di 512 byte, quello destinatario può ricevere:

- 8 "pezzi" da 256 byte;
- 1 "pezzo" da 2.048 byte;
- ecc.

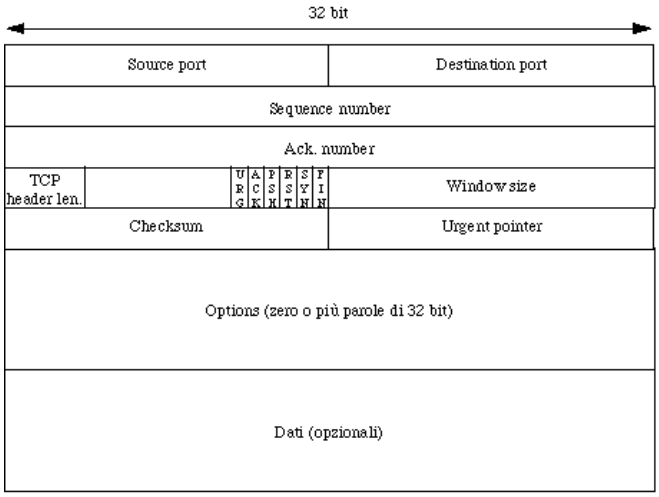
Ci pensano le entità TCP a suddividere il flusso in arrivo dal livello application in segmenti, a trasmetterli e a ricombinarli in un flusso che viene consegnato al livello application di destinazione.

C'è comunque la possibilità, per il livello application, di forzare l'invio immediato di dati; ciò causa l'invio di un flag ***urgent*** che, quando arriva dall'altra parte, fa sì che l'applicazione venga interrotta e si dedichi a esaminare i dati urgenti (questo succede quando, ad esempio, l'utente durante una sessione di emulazione di terminale digita il comando ABORT (CTRL-C) della computazione corrente).

Protocollo TCP

Le caratteristiche più importanti sono le seguenti:

- ogni byte del flusso TCP è numerato con un numero d'ordine a 32 bit, usato sia per il controllo di flusso che per la gestione degli ack;
- un segmento TCP non può superare i 65.535 byte;
- un segmento TCP è formato da:



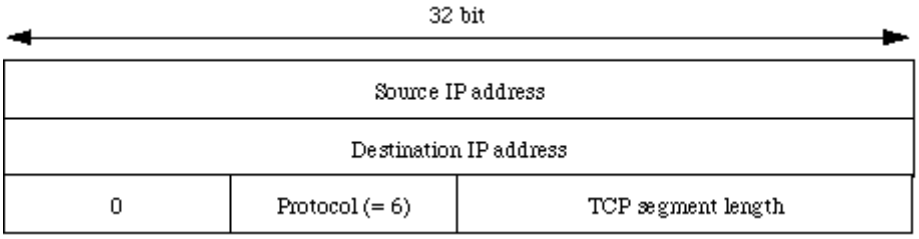
  - uno header, a sua volta costituito da:
    - una parte fissa di 20 byte;
    - una parte opzionale;
  - i dati da trasportare;
- TCP usa un meccanismo di sliding window di tipo go-back-n con timeout. Se questo scade, il segmento si ritrasmette. Si noti che le dimensioni della finestra scorrevole e i valori degli ack sono espressi in numero di byte, non in numero di segmenti.

I campi dell'header hanno le seguenti funzioni:

Source port, ***destination port***	identificano gli end point (locali ai due host) della connessione. Essi, assieme ai corrispondenti numeri IP, formano i due TSAP.
Sequence number	il numero d'ordine del primo byte contenuto nel campo dati.
Ack. number	il numero d'ordine del prossimo byte aspettato.
TCP header length	quante parole di 32 bit ci sono nell'header (necessario perché il campo options è di dimensione variabile).
URG	1 se urgent pointer è usato, 0 altrimenti.
ACK	1 se l'ack number è valido (cioè se si convoglia un ack), 0 altrimenti.
PSH	dati urgenti (***pushed data***), da consegnare senza aspettare che il buffer si riempia.
RST	richiesta di reset della connessione (ci sono

	problemi!).
SYN	usato nella fase di setup della connessione: • SYN=1 ACK=0 richiesta connessione; • SYN=1 ACK=1 accettata connessione.
FIN	usato per rilasciare una connessione.
Window size	il controllo di flusso è di tipo sliding window di dimensione variabile. Window size dice quanti byte possono essere spediti a partire da quello (compreso) che viene confermato con l'ack number. Un valore zero significa: fermati per un pò, riprenderai quando ti arriverà un uguale ack number con un valore di window size diverso da zero.
Checksum	simile a quello di IP; il calcolo include uno pseudoheader.
Urgent pointer	puntatore ai dati urgenti.
Options	fra le più importanti, negoziabili al setup: • dimensione massima dei segmenti da spedire; • uso di selective repeat invece che go-back-n; • uso di NAK.

Nel calcolo del checksum entra anche uno ***pseudoheader*** , in aperta violazione della gerarchia, dato che il livello TCP in questo calcolo opera su indirizzi IP.



Lo pseudoheader non viene trasmesso, ma precede concettualmente l'header. I suoi campi hanno le seguenti funzioni:

Source IP address, ***destination IP address***	indirizzi IP (a 32 bit) di sorgente e destinatario.

Protocol	il codice numerico del protocollo TCP (pari a 6).
TCP segment length	il numero di byte del segmento TCP, header compreso.

Reti livello 4 i protocolli di livelloTransport

I protocolli di livello transport (sulla base dei quali si implementano i servizi) assomigliano per certi aspetti a quelli di livello data link. Infatti si occupano, fra le altre cose, anche di:

- controllo degli errori;
- controllo di flusso;
- riordino dei TPDU.

Ci sono però anche delle importanti differenze. Quella principale è che:

- nel livello data link fra le peer entity c'è un singolo canale di comunicazione;
- nel livello transport c'è di mezzo l'intera subnet di comunicazione.

Questo implica che, a livello transport:

- è necessario indirizzare esplicitamente il destinatario;
- è più complicato stabilire la connessione;
- la rete ha una capacità di memorizzazione, per cui dei TPDU possono saltare fuori quando la destinazione meno se li aspetta;
- buffering e controllo di flusso richiedono un approccio differente che nel livello data link, a causa del numero molto variabile di connessioni che si possono avere di momento in momento.

Quando si vuole attivare una connessione, si deve ovviamente specificare con chi la si desidera. Dunque, si deve decidere come è fatto l'indirizzo di livello transport, detto ***TSAP address*** (***Transport Service Access Point address***).

Tipicamente un TSAP address ha la forma

(NSAP address, informazione supplementare)

Ad esempio, in Internet un TSAP address (ossia un indirizzo TCP o UDP) ha la forma:

(IP address:port number)

dove IP address è il NSAP address, e port number è l'informazione supplementare.

Questo meccanismo di formazione degli indirizzi dei TSAP ha il vantaggio di determinare implicitamente l'indirizzo di livello network da usare per stabilire la connessione.

In assenza di tale meccanismo, diviene necessario che l'architettura preveda un servizio per effettuare il mapping fra gli indirizzi di livello transport e i corrispondenti indirizzi di livello network.

Reti livello 3

Il livello network è incaricato di muovere i pacchetti dalla sorgente fino alla destinazione finale, attraversando tanti sistemi intermedi (***router***) della subnet di comunicazione quanti è necessario.

Ciò è molto diverso dal compito del livello data link, che è di muovere informazioni solo da un capo all'altro di un singolo canale di comunicazione wire-like.

Le incombenze principali di questo livello sono:

- conoscere la topologia della rete;
- scegliere di volta in volta il cammino migliore (***routing***);
- gestire il flusso dei dati e le congestioni (***flow control*** e ***congestion control***);
- gestire le problematiche derivanti dalla presenza di più reti diverse (***internetworking***).

Nel progetto e nella realizzazione del livello network di una architettura di rete si devono prendere decisioni importanti in merito a:

- ***servizi offerti*** al livello transport;
- ***organizzazione interna*** della subnet di comunicazione.

In merito ai servizi offerti al livello superiore, ci sono (come abbiamo già accennato) due tipologie fondamentali di servizi:

- servizi connection-oriented;
- servizi connectionless.

In proposito, ci sono due scuole di pensiero:

- fautori dei servizi connection-oriented (compagnie telefoniche);
- fautori dei servizi connectionless (Internet Community).

La prima scuola di pensiero afferma che il livello network deve fornire un servizio sostanzialmente affidabile e orientato alla connessione. In questa visione, succede che:

- le peer entitiy stabiliscono una connessione, negoziandone i parametri (di qualità, di costo, ecc.), alla quale viene associato un identificatore;
- tale identificatore viene inserito in ogni pacchetto che verrà inviato;
- la comunicazione è bidirezionale e i pacchetti viaggiano, in sequenza, lungo il cammino assegnato alla connessione;
- il controllo di flusso è fornito automaticamente (attraverso alcuni dei parametri negoziati, come ad esempio il dimensionamento di una o più finestre scorrevoli).

La seconda scuola di pensiero ritiene invece che la sottorete debba solo muovere dati e nient'altro:

- la sottorete è giudicata inerentemente inaffidabile, per cui gli host devono provvedere per conto proprio alla correzione degli errori e al controllo di flusso;

- una ovvia conseguenza è che il servizio offerto dal livello network dev'essere datagram, visto che è inutile inserire le funzioni di controllo degli errori e del flusso in due diversi livelli;
- i pacchetti viaggiano indipendentemente, e dunque devono tutti contenere un identificatore (ossia l'indirizzo) della destinazione.

Di fatto, il problema è dove mettere la complessità della realizzazione:

- la prima scuola la mette nei nodi della subnet, che si devono occupare del ***setup delle connessioni*** e di fornire la necessaria affidabilità;
- la seconda scuola la mette negli host, i cui livelli transport forniscono l'affidabilità e l'orientamento alla connessione.

In realtà le decisioni sono due, separate:

- offrire o no un servizio affidabile;
- offrire o no un servizio orientato alla connessione.

Le scelte più comuni sono di offrire ***servizi connection oriented affidabili*** oppure ***servizi connectionless non affidabili***, mentre le altre due combinazioni, anche se tecnicamente possibili, non sono diffuse.

uesto è un problema separato ed indipendente da quello dei servizi offerti, anche se spesso c'è una relazione fra i due.

Una subnet può essere organizzata con un funzionamento interno:

- basato su connessioni (che in questo contesto si chiamano ***circuiti virtuali***):
    - la subnet stabilisce un circuito virtuale (sul quale verrà tipicamente veicolato il traffico di un servizio connection oriented), cioé crea un cammino fra la sorgente e la destinazione;
    - tutti i router lungo tale cammino ricordano, in una apposita struttura dati, la parte di loro competenza di tale cammino (e cioé quale linea in entrata e quale in uscita sono assegnate al cammino);
    - quando arrivano pacchetti che contengono l' ID di tale circuito virtuale, essi vengono instradati di conseguenza (tutti nello stesso modo).
- connectionless:
    - i router si limitano a instradare ogni pacchetto che arriva sulla base del suo indirizzo di destinazione, decidendo di volta in volta come farlo proseguire;
    - i router hanno delle ***tabelle di instradamento*** (***routing table***)che indicano, per ogni possibile destinazione, quale linea in uscita utilizzare; si noti che queste tabelle esistono anche nelle subnet del tipo precedente, dove però servono solamente nella fase di setup della connessione (per decidere come instradare i pacchetti di setup);
    - quando offre un servizio connection-oriented, questo livello fa credere al livello superiore che esista una connessione, ma poi i pacchetti viaggiano indipendentemente (e quindi hanno tutti l'indirizzo del destinatario) e vengono rimessi in ordine dal livello network solo a destinazione, prima di essere consegnati

al livello superiore.

Ognuna delle due organizzazioni della subnet sopra viste ha i suoi supporter e i suoi detrattori, anche sulla base delle seguenti considerazioni:

	Subnet basata su connessioni	**Subnet connectionless**
Banda trasmissiva	***Minore*** (piccole ID in ogni pacchetto)	***Maggiore*** (intero indirizzo di dest. in ogni pacchetto)
Spazio sui router	***Maggiore*** (strutture dati per i circuiti virtuali)	***Minore***
Ritardo per il setup	***Presente***	***Assente***
Ritardo per il routing	***Assente***	***Presente***
Congestione	***Minore*** (risorse allocate in anticipo)	***Maggiore*** (possibile in ogni momento)
Vulnerabilità	***Alta***	***Bassa***

Dev'essere chiaro che i servizi offerti sono indipendenti dalla realizzazione interna della subnet. E' possibile avere tutte le quattro combinazioni di servizio offerto e implementazione della subnet:

- servizi connection oriented su circuiti virtuali;
- servizi connectionless su subnet datagram;
- servizi connection oriented su subnet datagram (si cerca di fornire comunque un servizio robusto);
- servizi connectionless su circuito virtuale (esempio: IP su subnet ATM).

Reti livello 5 e posta elettronica

La ***posta elettronica*** è uno dei servizi più consolidati ed usati nelle reti. In Internet è in uso da circa 20 anni, e prima del WWW era senza dubbio il servizio più utilizzato.

Un servizio di posta elettronica, nel suo complesso, consente di effettuare le seguenti operazioni:

- comporre un messaggio;
- spedire il messaggio (a uno o più destinatari);

- ricevere messaggi da altri utenti;
- leggere i messaggi ricevuti;
- stampare, memorizzare, eliminare i messaggi spediti o ricevuti.

Di norma, un messaggio ha un formato ben preciso. In Internet un messaggio ha un formato (definito nell'RFC 822) costituito da un ***header*** e da un ***body***, separati da una linea vuota.

Lo header è a sua volta costituito da una serie di linee, ciascuna relativa a una specifica informazione (identificata da una parola chiave che è la prima sulla linea); alcune informazioni sono:

`To`	indirizzo di uno o più destinatari.
`From`	indirizzo del mittente.
`Cc`	indirizzo di uno o più destinatari a cui si invia per conoscenza.
`Bcc`	blind Cc: gli altri destinatari non sanno che anche lui riceve il messaggio.
`Subject`	argomento del messaggio.
`Sender`	chi materialmente effettua l'invio (ad es. nome della segretaria).

Il body contiene il testo del messaggio, in caratteri ASCII. L'ultima riga contiene solo un punto, che identifica la fine del messaggio.

Gli indirizzi di posta elettronica in Internet hanno la forma:

username@hostname

dove username è una stringa di caratteri che identifica il destinatario, e hostname è un nome DNS oppure un indirizzo IP.

Ad esempio, bongiovanni@dsi.uniroma1.it è l'indirizzo di posta elettronica dell'autore di queste dispense.

La posta elettronica viene implementata in Internet attraverso la cooperazione di due tipi di sottosistemi:

- ***Mail User Agent*** (***MUA***);
- ***Mail Transport Agent*** (***MTA***).

Il primo permette all'utente finale di:

- comporre messaggi;
- consegnarli a un MTA per la trasmissione;
- ricevere e leggere messaggi;
- salvarli o eliminarli.

Il secondo si occupa di:

- trasportare i messaggi sulla rete, fino alla consegna a un MTA di destinazione;

- rispondere ai MUA dei vari utenti per consegnare loro la posta arrivata; in questa fase l'MTA richiede ad ogni utente una password per consentire l'accesso ai messaggi.

Corrispondentemente, sono definiti due protocolli principali per la posta elettronica:

- ***SMTP*** (***Simple Mail Transfer Protocol***, RFC 821) per il trasporto dei messaggi:
  - dal MUA di origine ad un MTA;
  - fra vari MTA, da quello di partenza fino a quello di destinazione;
- ***POP3*** (***Post Office Protocol*** versione 3, RFC 1225) per la consegna di un messaggio da parte di un MTA al MUA di destinazione.

Recentemente sono stati introdotti altri protocolli più sofisticati, quali ***IMAP*** (***Interactive Mail Acces Protocol***, RFC 1064) e ***DMSP*** (***Distributed Mail System Protocol***, RFC 1056), il cui supporto però non è ancora molto diffuso nel software disponibile agli utenti.

Come avviene la trasmissione di un messaggio? Supponiamo che l'utente

`pippo@topolinia.wd`

spedisca un messaggio a

`minnie@paperinia.wd`

e immaginiamo che:

- Pippo usi un MUA configurato per consegnare la posta ad un SMTP server in esecuzione sull'host `mailer.topolinia.wd`;
- Minnie abbia un MUA configurato per farsi consegnare la posta da un POP3 server in esecuzione sull'host `mailer.paperinia.wd`.

La sequenza di azioni che hanno luogo è la seguente:

1. Pippo compone il messaggio col suo MUA, che tipicamente è un programma in esecuzione su un PC in rete;
2. appena Pippo preme il pulsante SEND, il suo MUA:
   - interroga il DNS per sapere l'indirizzo IP dell'host `mailer.topolinia.wd`;
   - apre una connessione TCP ed effettua una conversazione SMTP con il server SMTP in esecuzione sull'host `mailer.topolinia.wd`, per mezzo della quale gli consegna il messaggio;
   - chiude la connessione TCP;
3. Pippo se ne va per i fatti suoi;
4. il server SMTP di `mailer.topolinia.wd`:
   - chiede al DNS l'indirizzo IP di `paperinia.wd`;
   - scopre che è quello dell'host `mailer.paperinia.wd`;
   - apre una connessione TCP e poi una conversazione SMTP con il server SMTP in esecuzione su quell'host e gli consegna il messaggio scritto da Pippo;
5. Minnie lancia il suo MUA;
6. appena Minnie preme il pulsante "check mail", il suo MUA:
   - interroga il DNS per avere l'indirizzo IP dell'host `mailer.paperinia.wd`;

- apre una connessione TCP e poi una conversazione POP3 col server POP in esecuzione su `mailer.paperinia.wd` e preleva il messaggio di Pippo, che viene mostrato a Minnie.

Si noti che `topolinia.wd` e `paperinia.wd` in genere corrispondono a un dominio nel suo complesso e non ad un singolo host, al fine di rendere gli indirizzi di posta elettronica indipendenti da variazioni del numero, dei nomi logici e degli indirizzi IP degli host presenti nel dominio.

Nel DNS ci sono opportuni record detti di tipo ***MX*** (***Mail Exchange***), che si occupano di indicare quale host effetttivamente fa da server SMTP per un dominio.

Nel nostro esempio avremo, nel DNS server di Paperinia, i record:

```
mailer.paperinia.wd A     100.10.10.5
paperinia.wd        MX    mailer.paperinia.wd
```

Il secondo record è un record di tipo MX, e indica che tutta la posta in arrivo per

`chiunque@paperinia.wd`

deve essere consegnata all'host `mailer.paperinia.wd`, e cioé quello che ha l'indirizzo IP

`100.10.10.5`

Inoltre, non è detto che `mailer.topolinia.wd` consegni i messagi direttamente a `mailer.paperinia.wd`. E' possibile che le macchine siano configurate in modo da trasferire i messaggi attraverso un certo numero di server SMTP intermedi.

Infine, vanno citate due significative estensioni di funzionalità della posta elettronica, in via di progressiva diffusione:

- possibilità di inviare messaggi di posta contenenti informazioni di qualunque tipo (per esempio programmi eseguibili, immagini, filmati, suoni, ecc.) attraverso lo standard ***MIME*** (***Multipurpose Internet Mail Extension***, RFC 1341 e 1521);
- possibilità di inviare messaggi corredati di firma digitale o crittografati, attraverso lo standard in via di definizione ***S/MIME*** (***Secure/MIME***, RFC 1847).

Reti algoritmi di routing

La funzione principale del livello network è di instradare i pacchetti sulla subnet, tipicamente facendo fare loro molti ***hop*** (letteralmente, salti) da un router ad un altro.

Un ***algoritmo di routing*** è quella parte del software di livello network che decide su quale linea di uscita instradare un pacchetto che è arrivato:

- in una subnet datagram l'algoritmo viene applicato ex novo ad ogni pacchetto;
- in una subnet basata su circuiti virtuali l'algoritmo viene applicato solo nella fase di setup del circuito; in tale contesto si usa spesso il termine ***session routing***.

Da un algoritmo di routing desideriamo:

- correttezza (deve muovere il pacchetto nella giusta direzione);
- semplicità (l'implementazione non deve essere troppo complicata);
- robustezza (deve funzionare anche in caso di cadute di linee e/o router e di riconfigurazioni della topologia);
- stabilità (deve convergere, e possibilmente in fretta);
- equità (non deve favorire nessuno);
- ottimalità (deve scegliere la soluzione globalmente migliore.

Purtroppo, gli ultimi due requisiti sono spesso in conflitto fra loro; inoltre, a proposito dell'ottimalità, non sempre è chiaro cosa si voglia ottimizzare. Infatti, supponiamo che si vogliano:

- minimizzare il ritardo medio pacchetti;
- massimizzare il throughput totale dei pacchetti.

Si scopre facilmente che questi due obiettivi sono in conflitto fra loro, perché di solito aumentare il throughput allunga le code sui router e quindi aumenta il ritardo: questo è vero per qualunque sistema basato su code gestito in prossimità della sua capacità massima.

Gli algoritmi di routing si dividono in due classi principali:

- ***algoritmi non adattivi*** (***static routing***):
    - le decisioni di routing sono prese in anticipo, all'avvio della rete, e sono comunicate ai router che poi si attengono sempre a quelle;
- ***algoritmi adattivi*** (***dynamic routing***):
    - le decisioni di routing sono riformulate (sulla base del traffico, della topologia della rete, ecc.) molto spesso.

Gli algoritmi adattivi differiscono fra loro per:

- come ricevono le informazioni:
    - localmente;
    - dai router adiacenti;
    - da tutti i router;
- quanto spesso rivedono le decisioni:
    - a intervalli di tempo prefissati;
    - quando il carico cambia;
    - quando la topologia cambia;
- quale metrica di valutazione adottano:
    - distanza;
    - numero di hop;
    - tempo di transito stimato.

E' possibile fare una considerazione generale sull'ottimalità dei cammini, indipendentemente dallo specifico algoritmo adottato per selezionarli.

Il ***principio di ottimalità*** dice che se il router j è nel cammino ottimo fra i e k, allora anche il cammino ottimo fra j e k è sulla stessa strada:

Se così non fosse, ci sarebbe un altro cammino (ad es. quello tratteggiato in figura) fra j e k migliore di quello che è parte del cammino ottimo fra i e k, ma allora ci sarebbe anche un cammino fra i e k migliore di quello ottimo.

Una diretta conseguenza è che l'insieme dei cammini ottimi da tutti i router a uno specifico router di destinazione costituiscono un ***albero***, detto ***sink tree*** per quel router.

In sostanza, gli algoritmi di routing cercano e trovano i sink tree relativi a tutti i possibili router di destinazione, e quindi instradano i pacchetti esclusivamente lungo tali sink tree.

Reti algoritmi di routing statici

Questi algoritmi, come abbiamo già accennato, sono eseguiti solamente all'avvio della rete, e le decisioni di routing a cui essi pervengono sono poi applicate senza più essere modificate.

Esamineremo i seguenti algoritmi statici:

- ***shortest path routing***;
- ***flooding***;
- ***flow-based routing***.

Shortest path routing

L'idea è semplice: un host di gestione della rete mantiene un grafo che rappresenta la subnet:

- i nodi rappresentano i router;
- gli archi rappresentano le linee punto-punto.

All'avvio della rete (o quando ci sono variazioni permanenti della topologia) l'algoritmo:

- applica al grafo un algoritmo per il calcolo del ***cammino minimo*** fra ogni coppia di nodi; ad esempio, il noto algoritmo di Dijkstra ('59) può essere usato;
- invia tali informazioni a tutti i router.

Quale sia il cammino minimo dipende da qual'è la grandezza che si vuole minimizzare. Tipicamente

si usano:

- numero di hop, cioè di archi, da attraversare;
- lunghezza dei collegamenti;
- tempo medio di accodamento e trasmissione;
- una combinazione di lunghezza, banda trasmissiva, traffico medio, ecc.

Flooding

La tecnica del ***flooding*** consiste nell'inviare ogni pacchetto su tutte le linee eccetto quella da cui è arrivato.

In linea di principio il flooding può essere usato come algoritmo di routing (ogni pacchetto inviato arriva a tutti i router) ma presenta l'inconveniente di generare un numero enorme (teoricamente infinito!) di pacchetti.

Ci sono delle tecniche per limitare il traffico generato:

- inserire in ogni pacchetto un ***contatore*** che viene decrementato ad ogni hop. Quando il contatore arriva a zero, il pacchetto viene scartato. Un appropriato valore iniziale può essere il diametro della subnet;
- inserire la coppia (***source router ID, sequence number***) in ogni pacchetto. Ogni router esamina tali informazioni e ne tiene traccia, e quando le vede per la seconda volta scarta il pacchetto;
- ***selective flooding***: i pacchetti vengono duplicati solo sulle linee che vanno all'incirca nella giusta direzione (per questo si devono mantenere apposite tabelle a bordo).

Il flooding non è utilizzabile in generale come algoritmo di routing, però:

- è utile in campo militare (offre la massima affidabilità e robustezza);
- è utile per l'aggiornamento contemporaneo di informazioni distribuite;
- è utile come strumento di paragone per altri algoritmi, visto che trova sempre, fra gli altri, il cammino minimo.

Flow-based routing

Questo algoritmo è basato sull'idea di:

- calcolare in anticipo il traffico atteso su ogni linea;
- da questi calcoli derivare una stima del ritardo medio atteso per ciascuna linea;
- basare su tali informazioni le decisioni di routing.

Le informazioni necessarie per poter applicare l'algoritmo sono:

- la topologia della rete;
- la matrice delle quantità di traffico T(i,j) stimate fra ogni coppia (i,j) di router;
- le capacità (in bps ad esempio) delle linee point to point.

Vengono fatte le seguenti assunzioni:

- il traffico è stabile nel tempo e noto in anticipo;

- il ritardo su ciascuna linea aumenta all'aumentare del traffico sulla linea e diminuisce all'aumentare della velocità della linea secondo le leggi della teoria delle code.

Dai ritardi calcolati per le singole linee si può calcolare il ritardo medio dell'intera rete, espresso come somma pesata dei ritardi delle singole linee. Il peso di ogni linea è dato dal traffico su quella linea diviso il traffico totale sulla rete.

Il metodo nel suo complesso funziona così:

- si sceglie un algoritmo di routing;
- sulla base di tale algoritmo si determinano i percorsi che verranno seguiti per il collegamento fra ogni coppia di router;
- si calcola il traffico che incide su ogni linea (che è uguale alla somma di tutti i T(i,j) instradati su quella linea);
- si calcola il ritardo di ogni linea;
- si calcola il ritardo medio della rete;
- si ripete il procedimento con vari algoritmi di routing, scegliendo alla fine quello che minimizza il ritardo medio dell'intera rete.

Reti algoritmi di routing dinamici

Nelle moderne reti si usano algoritmi dinamici, che si adattano automaticamente ai cambiamenti della rete. Questi algoritmi non sono eseguiti solo all'avvio della rete, ma rimangono in esecuzione sui router durante il normale funzionamento della rete.

Distance vector routing

Ogni router mantiene una tabella (***vector***) contenente un elemento per ogni altro router. Ogni elemento della tabella contiene:

- la distanza (numero di hop, ritardo, ecc.) che lo separa dal router in oggetto;
- la linea in uscita da usare per arrivarci;

Per i suoi vicini immediati il router stima direttamente la distanza dei collegamenti corrispondenti, mandando speciali ***pacchetti ECHO*** e misurando quanto tempo ci mette la risposta a tornare.

A intervalli regolari ogni router manda la sua tabella a tutti i vicini, e riceve quelle dei vicini.

Quando un router riceve le nuove informazioni, calcola una nuova tabella scegliendo, fra tutte, la concatenazione migliore

se stesso -> vicino immediato -> router remoto di destinazione

per ogni destinazione.

Ovviamente, la migliore è la concatenazione che produce la minore somma di:

- distanza fra il router stesso ed un suo vicino immediato (viene dalla misurazione diretta);
- distanza fra quel vicino immediato ed il router remoto di destinazione (viene dalla tabella ricevuta dal vicino immediato).

L'algoritmo distance vector routing funziona piuttosto bene, ma è molto lento nel reagire alle cattive notizie, cioè quando un collegamento va giù. Ciò è legato al fatto che i router non conoscono la topologia della rete.

Infatti, consideriamo questo esempio:

```
A     B     C     D     E   <- Router
*-----*-----*-----*-----*   <- Collegamenti (topologia lineare)
      1     2     3     4   <- Distanze da A
```

Se ora cade la linea fra A e B, dopo uno scambio succede questo:

```
A     B     C     D     E   <- Router
*     *-----*-----*-----*   <- Collegamenti
      3     2     3     4   <- Distanze da A (dopo uno scambio)
```

Ciò perché B, non ricevendo risposta da A, crede di poterci arrivare via C, che ha distanza due da A. Col proseguire degli scambi, si ha la seguente evoluzione:

```
A     B     C     D     E   <- Router
*     *-----*-----*-----*   <- Collegamenti
      3     4     3     4   <- Distanze da A (dopo due scambi)
      5     4     5     4   <- Distanze da A (dopo tre scambi)
      5     6     5     6   <- Distanze da A (dopo quattro scambi)
ecc.
```

A lungo andare, tutti i router vedono lentamente aumentare sempre più la distanza per arrivare ad A. Questo è il problema del ***count-to-infinity***.

Se la distanza rappresenta il numero di hop si può porre come limite il diametro della rete, ma se essa rappresenta il ritardo l'upper bound dev'essere molto alto, altrimenti cammini con un ritardo occasionalmente alto (magari a causa di congestione) verrebbero considerati interrotti.

Sono state proposte molte soluzioni al problema count-to-infinity, ma nessuna veramente efficace.

Nonostante ciò, il distance vector routing era l'algoritmo di routing di ARPANET ed è usato anche in Internet col nome di ***RIP*** (***Routing Internet Protocol***), e nelle prime versioni di DECnet e IPX.

Link state routing

Sopratutto a causa della lentezza di convergenza del distance vector routing, si è cercato un approccio diverso, che ha dato origine al ***link state routing***.

L'idea è questa:

- ogni router tiene sott'occhio lo stato dei collegamenti fra se e i suoi vicini immediati (misurando il ritardo di ogni linea) e distribuisce tali informazioni a tutti gli altri;
- sulla base di tali informazioni, ogni router ricostruisce localmente la topologia completa

dell'intera rete e calcola il cammino minimo fra se e tutti gli altri.

I passi da seguire sono:

1. scoprire i vicini e identificarli;
2. misurare il costo (ritardo o altro) delle relative linee;
3. costruire un pacchetto con tali informazioni;
4. mandare il pacchetto a tutti gli altri router;
5. previa ricezione degli analoghi pacchetti che arrivano dagli altri router, costruire la topologia dell'intera rete;
6. calcolare il cammino più breve a tutti gli altri router.

Quando il router si avvia, invia un ***pacchetto HELLO*** su tutte le linee in uscita. In risposta riceve dai vicini i loro indirizzi (univoci in tutta la rete).

Inviando vari pacchetti ECHO, misurando il tempo di arrivo della risposta (diviso 2) e mediando su vari pacchetti si deriva il ritardo della linea.

Si costruisce un pacchetto con:

- identità del mittente;
- numero di sequenza del pacchetto;
- età del pacchetto;
- lista dei vicini con i relativi ritardi.

La costruzione e l'invio di tali pacchetti si verifica tipicamente:

- a intervalli regolari;
- quando accade un evento significativo (es.: una linea va giù o torna su).

La distribuzione dei pacchetti è la parte più delicata, perché errori in questa fase possono portare qualche router ad avere idee sbagliate sulla topologia, con conseguenti malfunzionamenti.

Di base si usa il flooding, inserendo nei pacchetti le coppie (source router ID, sequence number) per eliminare i duplicati. Tutti i pacchetti sono confermati. Inoltre, per evitare che pacchetti vaganti (per qualche errore) girino per sempre, l'età del pacchetto viene decrementata ogni secondo, e quando arriva a zero il pacchetto viene scartato.

Combinando tutte le informazioni arrivate, ogni router costruisce il grafo della subnet e calcola il cammino minimo a tutti gli altri router.

Il link state routing è molto usato attualmente:

- in Internet ***OSPF*** (***Open Shortest Path First***) è basato su tale principio e si avvia ad essere l'algoritmo più utilizzato;
- un altro importante esempio è ***IS-IS*** (***Intermediate System-Intermediate System***), progettato per DECnet e poi adottato da OSI. La sua principale caratteristica è di poter gestire indirizzi di diverse architetture (OSI, IP, IPX) per cui può essere usato in reti miste o multiprotocollo.

Reti routing gerarchico

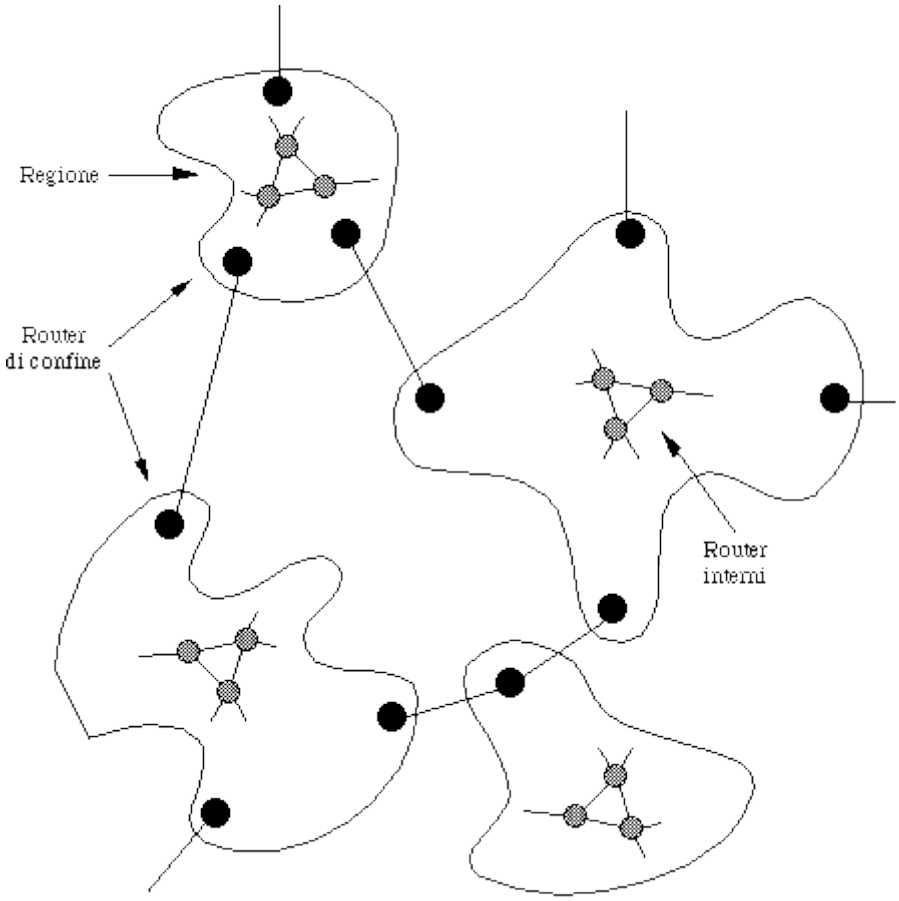
Quando la rete cresce fino contenere decine di migliaia di nodi, diventa troppo gravoso mantenere in ogni router la completa topologia. Il routing va quindi impostato in modo gerarchico, come succede nei sistemi telefonici.

La rete viene divisa in ***zone*** (spesso dette ***regioni***):

- all'interno di una regione vale quanto visto finora, cioè i router (detti ***router interni***) sanno come arrivare a tutti gli altri router della regione;
- viceversa, quando un router interno deve spedire qualcosa a un router di un'altra regione sa soltanto che deve farlo pervenire a un particolare router della propria regione, detto ***router di confine***.
- il router di confine sa a quale altro router di confine deve inviare i dati perché arrivino alla regione di destinazione.

Di conseguenza, ci solo due livelli di routing:

- un primo livello di routing all'interno di ogni regione;
- un secondo livello di routing fra tutti i router di confine.



I router interni mantengono nelle loro tabelle di routing:

- una entrata per ogni altro router interno, con la relativa linea da usare per arrivarci;
- una entrata per ogni altra regione, con l'indicazione del relativo router di confine e della linea da usare per arrivarci.

I router di confine, invece, mantengono:

- una entrata per ogni altra regione, con l'indicazione del prossimo router di confine da contattare e della linea da usare per arrivarci.

Non è detto che due livelli siano sufficienti. In tal caso il discorso si ripete su più livelli.