INTRODUZIONE ALLE RETI DI CALCOLATORI ED AL PROTOCOLLO TCP-IP

a cura di

Eugenio DI SCIASCIO e Giacomo PISCITELLI
Dipartimento di Elettrotecnica ed Elettronica
Politecnico di Bari

Bari dicembre 2000

INDICE

Questa introduzione alle reti di calcolatori è stata ricavata dall'Appendice alla Tesi di Laurea del dott. Ing. Paolo MARRA dal titolo "Reti multiservizio: l'applicazione voce su IP"

1 Trasmissione dell'Informazione.

1.1 Concetto di Informazione.

– *Messaggio.*

Il dialogo tra due qualsiasi entità (uomini o macchine) avviene attraverso lo scambio di *messaggi* servendosi di un linguaggio opportuno, generalmente ben definito nell'aspetto sintattico, semantico e procedurale.

– *Informazione.*

Con il termine *informazione* indichiamo, in questa tesi, il *"contenuto"* di un messaggio ovvero ciò che "dà senso" al dialogo, indipendentemente dalla "forma" assunta dal messaggio: testo, dati, immagini, audio e video sono tutti esempi di "contenitori" di informazione.

– *Sorgenti informative.*

Le sorgenti informative sono i *"generatori di informazione"*, cioè quelle entità che "producono" informazione emettendo dei messaggi. Un telefono, un fax, una telecamera, uno strumento musicale sono tutti esempi di sorgenti informative ([1]).

– *Considerazioni.*

Nonostante la pluralità delle forme in cui l'informazione può "materializzarsi", è possibile, in generale, caratterizzare il modo in cui essa viene trasmessa individuando alcuni aspetti significativi. Tali "elementi distintivi" (descritti nei paragrafi successivi) esulano, oltretutto, dal particolare livello d'astrazione in cui ci si può porre nell'osservare l'intero sistema di comunicazione.

([1]) Sono sorgenti informative anche oggetti più complessi come ad esempio un sistema per il telerilevamento o per la diffusione di video musicali o film.

1.2 Trasmissione unicast, broadcast, multicast.

In base al numero di destinatari del messaggio, la trasmissione può essere:

- *unicast*: il messaggio è rivolto ad un solo destinatario;
- *broadcast*: il messaggio è indirizzato a tutti i possibili destinatari;
- *multicast*: il messaggio è indirizzato ad uno specifico gruppo di destinatari.

Una conversazione telefonica o il trasferimento dati da un server ad un calcolatore terminale sono esempi di trasmissione unicast; esempi di trasmissione broadcast o multicast li ritroviamo in applicazioni tipo la diffusione audio/video o la teleconferenza.

1.3 Trasmissione simplex, half-duplex, full-duplex.

Un dialogo può svolgersi in modo:

- *simplex*: unidirezionale;
- *half-duplex*: bidirezionale, alterno;
- *full-duplex*: bidirezionale, contemporaneo.

1.4 Trasmissione orientata alla connessione e priva di connessione.

Un dialogo tra due o più entità può, in alcuni casi, richiedere delle particolari procedure per stabilire e terminare la conversazione; distinguiamo allora fra:

– *Trasmissione orientata alla connessione (connection-oriented).*

La conversazione avviene in *3 fasi*: il mittente stabilisce una *connessione* con il ricevente, quindi la utilizza per la trasmissione del messaggio ed infine la rilascia in modo esplicito. Le unità informative vengono sempre ricevute nello *stesso ordine* in cui sono state trasmesse. Ciò è quanto accade per esempio in una conversazione telefonica.

– *Trasmissione priva di connessione (connection-less).*

In questo caso la trasmissione del messaggio avviene senza alcun coordinamento o pianificazione; conseguentemente, ogni messaggio deve contenere *"l'indirizzo"* completo del destinatario affinché possa essere recapitato in modo corretto. L'ordine con cui i messaggi vengono ricevuti generalmente *non* corrisponde all'ordine con cui sono stati spediti.

Un buon esempio di trasmissione connection-less è la spedizione di una lettera o di un telegramma.

1.5 Trasmissione affidabile e non-affidabile.

La trasmissione può avvenire in modo:

- *Affidabile.*

  Le unità informative non vengono mai perse, duplicate o modificate dal sistema di trasmissione; in pratica, il messaggio ricevuto è sempre esattamente corrispondente nel contenuto al messaggio trasmesso. Una trasmissione affidabile è appropriata ad esempio nel trasferimento dei file (archivi o programmi).

- *Non-affidabile.*

  Il messaggio può essere perso, duplicato o modificato (corrotto). Ad esempio, nella trasmissione della voce, la comprensibilità del parlato (e quindi il "contenuto" del messaggio vocale) non subisce, entro certi limiti, particolari alterazioni anche in presenza di rumore, distorsioni di ampiezza e fase, eco ed altre non-idealità.

2 Concetto di Rete.

2.1 Definizioni.

– *Rete (network).*

Una rete è un mezzo per trasferire *informazione* tra due o più dispositivi terminali ad essa collegati; può essere di 2 tipi: punto-punto e broadcast.

– *Rete punto-punto (point-to-point network).*

È composta da linee *(link)* e nodi *(switch o router)*. Il collegamento tra due utenti della rete, richiede la definizione di un percorso (*path* o *route*), generalmente composto da più linee interconnesse. Ogni linea è ad uso *esclusivo* dei nodi che la terminano (Figura A.1).

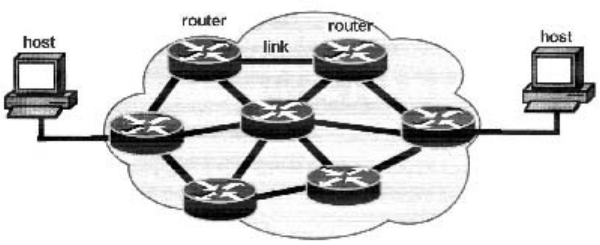


Figura A.1: Esempio di rete *punto-punto*.

– *Rete broadcast (broadcast network).*

Un *unico* canale di comunicazione (detto ancora linea o *link*) è *condiviso* da tutti gli utenti della rete; è necessario pertanto stabilire un meccanismo di contesa (ad accesso casuale oppure ordinato) per evitare che due o più utenti della rete tentino di trasmettere contemporaneamente, generando una *"collisione"* (Figura A.2).

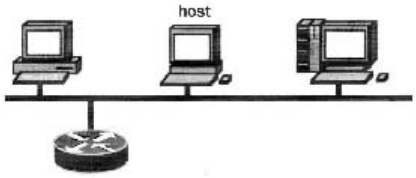


Figura A.2: Esempio di rete *broadcast*.

Poiché la linea broadcast collega tutti gli utenti, *non* occorre stabilire un percorso prima di effettuare una trasmissione: ogni messaggio inviato sulla rete è sempre "ascoltato" da tutti.

Rete punto-punto	Rete broadcast
è composta da linee e nodi; le linee sono ad uso esclusivo dei nodi che le terminano;	unico canale di comunicazione condiviso;
richiede un protocollo di routing.	richiede un protocollo di accesso multiplo.

Tabella A.1: Rete *punto-punto* e Rete *broadcast*.

– *Dispositivo terminale (end-system).*

Detto anche *host*, è un dispositivo connesso alla rete la cui funzione è quella di eseguire un'*applicazione* oppure un *servizio* utile all'utente.

Un end-system può essere un semplice telefono oppure un personal computer, una stazione di videoconferenza, un apparecchio TV, un centro informazioni elettronico o altro ancora.

Figura A.3: Esempi di *end-system*.

– *Nodo (switch o router).*

L'obiettivo della rete di permettere a due o più end-system di scambiarsi informazione, può essere risolto, piuttosto semplicemente, con l'uso di un mezzo *broadcast*, come ad esempio un bus condiviso *(Rete broadcast)*. Questa soluzione è ovviamente applicabile se la rete *non* è di grandi dimensioni: quando la distanza tra gli utenti comincia ad essere dell'ordine del chilometro o delle decine di chilometri, è indispensabile affidarsi a dei collegamenti *punto-punto*.

Con *N* utenti, possiamo pensare ad una *mesh completa*, cioè ad *N(N-1)/2* linee che collegano direttamente tutti a tutti, ma, se *N* è grande, è certamente meno dispendioso connettere gli utenti ad una *rete commutata* (*N* linee), composta da linee e nodi (Figura A.4).

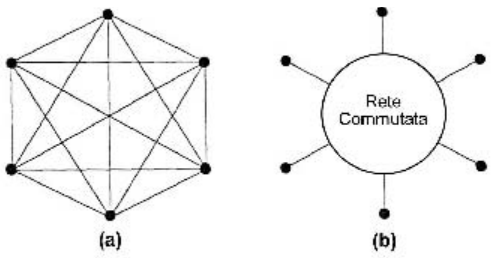


Figura A.4: *(a)* mesh; *(b)* rete commutata.

Un *nodo*, quindi, è un *dispositivo di commutazione*. Un nodo è più propriamente detto *switch* quando è chiamato a svolgere, su ogni unità informativa che lo attraversa, solamente un'operazione di commutazione (prestabilita); quando invece il percorso delle unità informative viene determinato mentre la trasmissione è in corso, la denominazione più corretta è quella di *router*.

Switch	Router
Commuta ogni unità informativa sulla prestabilita linea d'uscita.	Decide su quale linea d'uscita inoltrare l'unità informativa, quindi la commuta.

Tabella A.2: Elementi di commutazione.

- *Rete di Calcolatori.*

  Una rete di calcolatori è una collezione di calcolatori autonomi collegati (detti *host*).

  Due calcolatori sono *collegati* se in grado di scambiarsi informazioni. Richiedendo che i calcolatori siano *autonomi*, intendiamo escludere dalla definizione tutti sistemi gerarchici, nei quali cioè esiste una chiara relazione master/slave. Ad esempio, se un calcolatore può forzatamente accedere, bloccare o controllare un'altra unità, i calcolatori non sono autonomi: in una rete composta da unità autonome, ogni host può *liberamente* attivare un collegamento per effettuare una trasmissione.

- *Sistema Distribuito.*

  Un sistema distribuito è un sistema S/w costruito per una rete di calcolatori. Esso rende *"trasparente"* all'utente (cioè non-visibile) l'esistenza di molteplici calcolatori autonomi ma cooperanti. Ciò permette di elaborare e/o archiviare l'informazione in qualsiasi punto essa venga prodotta, ma nello stesso tempo di renderla automaticamente disponibile ovunque nella rete.

2.2 Classificazione delle reti per scala.

E' un importante metro di classificazione perché, come si è visto, su scale diverse sono generalmente utilizzate tecniche diverse. Come illustrato in Figura A.5, è possibile distinguere tra:

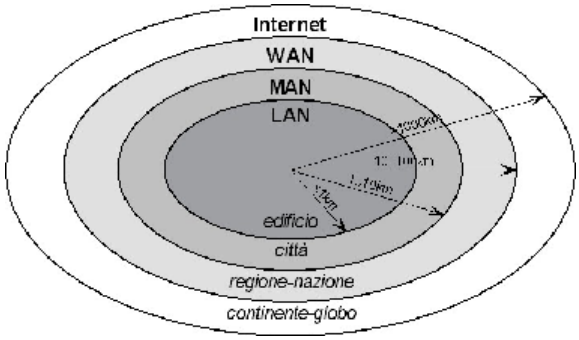


Figura A.5: Classificazione delle reti per scala.

– *Reti Locali o LAN (Local Area Network).*

La distanza tra gli utenti terminali varia tra pochi metri a circa 1 *km*; gli end-system possono trovarsi ad esempio nella stessa stanza, nello stesso edificio o nello stesso campus universitario. Essendo la dimensione limitata, è noto a priori il tempo di trasmissione nel caso peggiore; in genere adottano tecnologia trasmissiva broadcast. Velocità di trasmissione tipiche sono da 10 a 100 Mbps (megabit al secondo, cioé milioni di bit al secondo), con basso ritardo di propagazione del segnale da un capo all'altro del canale (qualche decina di microsecondi) e basso tasso di errore. Le LAN hanno in genere un topologia ben definita, bus o ring.

Esempi ben noti di LAN per la trasmissione dati sono *Ethernet* e *Token Ring*; anche una rete composta da un PABX *(Private Automatic Branch eXchange)* che connette tra loro dei telefoni costituisce una rete locale, anzi un PABX numerico potrebbe essere impiegato anche per la trasmissione dati a commutazione di circuito.

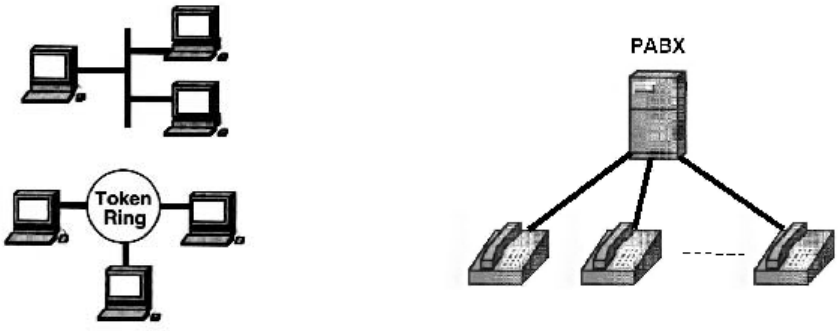


Figura A.6: Esempi di *LAN*.

– *Reti Metropolitane o MAN (Metropolitan Area Network).*

La distanza tra gli end-system, posti ad esempio in una stessa città o quartiere, è dell'ordine di 10 *km*. Le reti metropolitane sono generalmente pubbliche. Esistono due standard: IEEE 802.6 o DQDB (Distributed Queue Dual Bus) e *FDDI*.

– *Reti Geografiche o WAN (Wide Area Network).*

La distanza varia tra i 100÷1000 *km*, con una rete che copre ad esempio un'intera nazione o continente. Esempi di reti geografiche sono: la rete telefonica pubblica *(PSTN)*, le reti *X.25* (vedi Figura A.7) e *Frame-Relay*, progettate per la trasmissione dati, le reti *ISDN* e *ATM*, concepite invece per offrire servizi multipli.

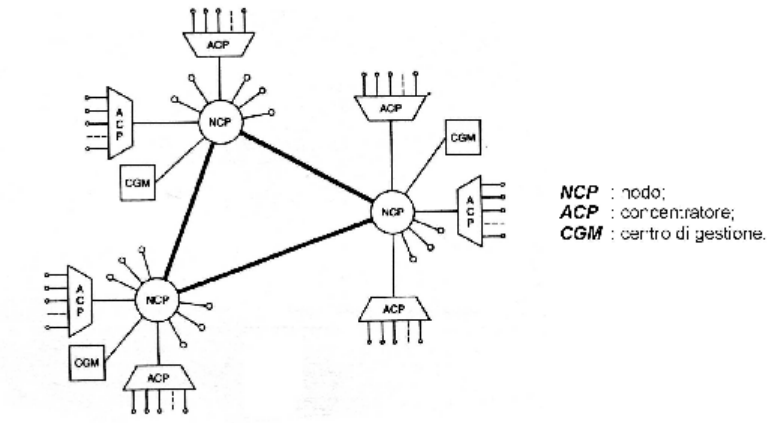


Figura A.7: Struttura di una rete *WAN*
(l'esempio si riferisce alla rete X.25 italiana ITAPAC).

Una WAN è tipicamente costituita di due componenti distinte:

- un insieme di elaboratori (*host* oppure *end system*);
- una *communication subnet* (o *subnet*), che connette gli *end system* fra loro. Il suo compito è trasportare messaggi da un *end system* all'altro.

Di norma la *subnet* consiste, a sua volta, di due componenti.

Le linee di trasmissione (dette anche circuiti o canali) e gli elementi di commutazione (*switching element*) tipicamente chiamati *router*; questi ultimi sono elaboratori specializzati utilizzati per connettere fra loro due o più linee di trasmissione. Quando arrivano dati su una linea, l'elemento di commutazione deve scegliere una linea in uscita sul quale instradarli.

Una tipica WAN è utilizzata per connettere più LAN fra loro mediante coppie di *router*. Ogni *router*, in generale, deve:

- ricevere un pacchetto da una linea in ingresso;

- memorizzarlo per intero in un buffer interno;
- appena la necessaria linea in uscita è libera, instradare il pacchetto su essa.

– *Internet.*

E' un'*interconnesione* di due o più *reti*, con distanze coperte anche dell'ordine di 10000 *km*. *Internet* ne è un esempio. Essa è una federazione di reti (una inter-net, per l'appunto), che condividono un insieme di protocolli (o protocol suite), noti come TCP/IP.

2.3 Topologie delle reti.

Varie topologie sono possibili (Figura A.8):

- bus,
- stella,
- anello,
- albero,
- mesh completa,
- mesh irregolare.

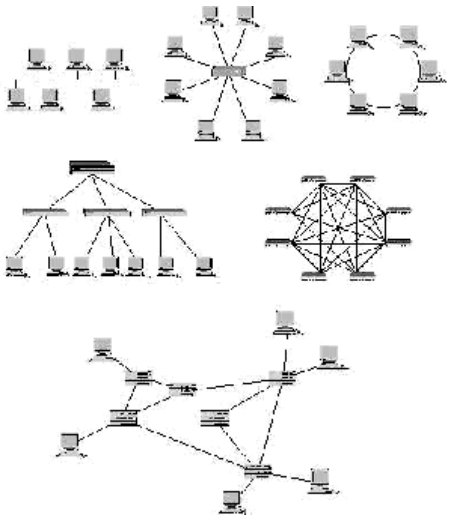


Figura A.8: *Topologie* delle reti.

Le topologie *regolari* sono tipiche in ambito LAN, le *irregolari* su scala WAN.

3 Tecniche di Multiplazione (Multiplexing).

Nelle reti per la trasmissione dell'informazione il mezzo trasmissivo viene generalmente suddiviso in più *canali* (suddivisione logica) in modo tale da ospitare, senza creare conflitti, flussi provenienti da più sorgenti indipendenti *(multiplexing).*

Le tecniche di multiplazione più frequentemente utilizzate sono FDM e TDM.

3.1 Frequency Division Multiplexing (FDM).

La banda a disposizione è suddivisa in *sottobande (canali)* separate da una *banda di guardia* necessaria per la loro separazione mediante dei filtri passa-banda (Figura A.9).

La tecnica *non* ottimizza l'uso della banda a causa delle indispensabili bande di guardia e *non* produce canali con uguali prestazioni (generalmente i canali collocati sugli estremi della banda ricevono un servizio peggiore).

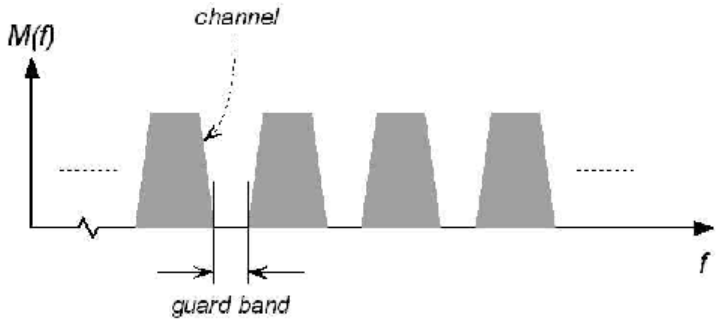


Figura A.9: Multiplazione a divisione di frequenza.

3.2 Time Division Multiplexing (TDM).

L'*intera* banda viene assegnata, a turno, ai vari canali per un tempo *finito*.

Il tempo può essere suddiviso in intervalli di lunghezza fissa *(time-slot)* o variabile; eventualmente più intervallini possono formare una trama o *frame*.

Distinguiamo gli schemi seguenti (vedi Figura A.10):

unslotted:	nessuna preliminare suddivisione del tempo;
(unframed) slotted:	preliminare suddivisione del tempo in intervalli di lunghezza fissa (slot);
framed and slotted:	gli slot sono raggruppati in frame; slot omologhi in frame distinti sono associati in modo logico tra loro (appartengono cioè allo stesso canale);
framed hybrid slotted-unslotted:	abbiamo una suddivisione del tempo in frame ed ogni frame contiene una sezione slotted ed una unslotted.

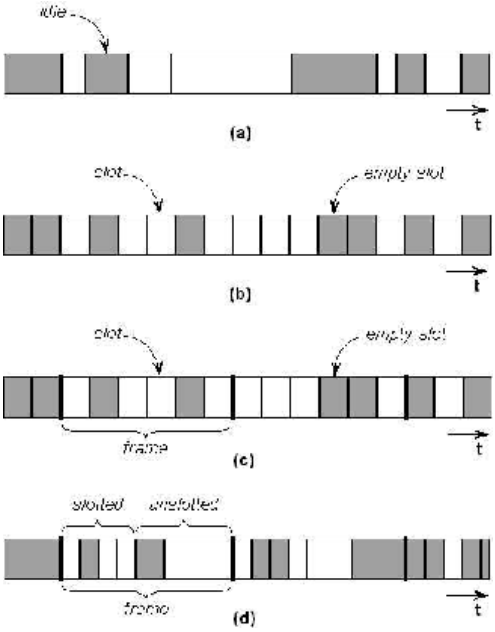


Figura A.10:
(a) TDM unslotted,
(b) slotted,
(c) framed and slotted,
(d) framed hybrid slotted-unslotted.

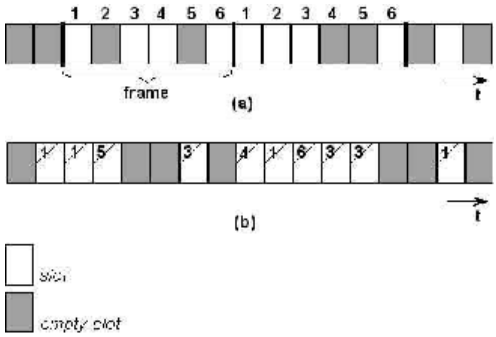


Figura A.11:
(a) S-TDM nel caso framed-and-slotted,
(b) A-TDM nel caso slotted.

Inoltre, ogni intervallino temporale può essere associato ad una particolare sorgente in modo statico o sincrono *(associazione implicita)* oppure dinamico o asincrono *(associazione esplicita)*. Precisamente (vedi Figura A.11):

– *TDM sincrono o S-TDM (solo nei casi framed-and-slotted e hybrid).*
  l'associazione è *statica* ed *implicita*. E' applicabile solo nel caso framed and slotted o ibrido nella sezione slotted. L'associazione è stabilita dalla *posizione* dello slot nel frame. Tende a sprecare banda se la sorgente non emette in modo *continuativo* (sorgente bursty).

– *TDM asincrono o A-TDM (sempre applicabile).*
  l'associazione è *dinamica* ed *esplicita*. E' sempre applicabile ed è stabilita da un'*etichetta* memorizzata nel blocco di dati (intestazione). Consentendo un utilizzo della linea su domanda, tende a massimizzarne l'utilizzo.

Dalle precedenti classificazioni possiamo derivare i modelli di multiplatori TDM elencati nella Tabella A.3.

	unslotted	slotted	framed and slotted	framed hybrid
TDM sincrono	**-**	**-**	**TDM**	**TDM Ibridi (*)**
TDM asincrono	**TDM statistici (STDM)**	**Cell Relay**	**-**	**TDM Ibridi (*)**

(*) il funzionamento è sincrono nella sezione slotted, asincrono in quella unslotted.

Tabella A.3: Multiplatori TDM.

4 Tecniche di Commutazione (Switching).

Le tecniche di commutazione sono di 3 tipi: di *circuito*, di *messaggio* e di *pacchetto*; in particolare, nella commutazione di pacchetto è possibile distinguere tra 2 differenti approcci, a *datagramma* e a *circuito virtuale*.

4.1 Commutazione di circuito.

In una rete a commutazione di circuito ad ogni connessione (effettuata su domanda o in modo semi-permanente) vengono assegnate *stabilmente* delle risorse di rete (un canale, rappresentato ad esempio da uno slot TDM), in modo tale da creare un *circuito* che connette i due utenti terminali (Figura A.12).

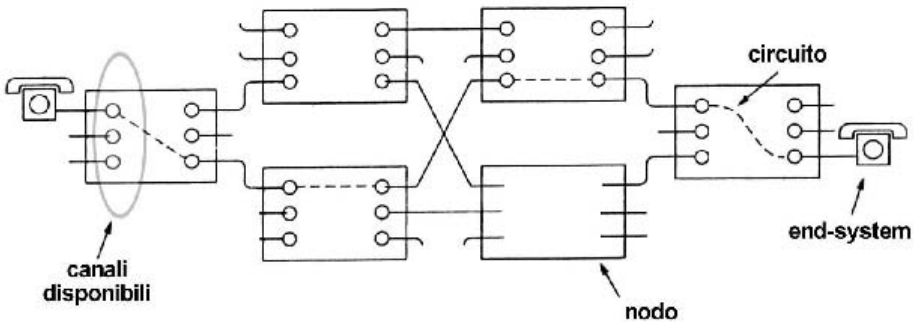


Figura A.12: Commutazione di Circuito.

Progettata per la trasmissione della *voce*, questa tecnica ha subìto negli anni un'evoluzione segnata principalmente dall'impiego di tecniche numeriche anziché analogiche per la trasmissione e la commutazione stessa. Ciò ha permesso il suo utilizzo per uno scopo completamente diverso: la trasmissione dei *dati*.

4.2 Commutazione di messaggio.

La trasmissione *non* richiede una preliminare fase di set up: non appena un utente ha un messaggio da spedire lo invia al nodo più vicino. Presso il nodo il messaggio viene processato con una tecnica detta "*store-and-forward*": viene prima completamente memorizzato *(store)*, poi ispezionato per individuare la sua destinazione ed infine inoltrato *(forward)* al nodo successivo non appena la linea d'uscita necessaria si rende disponibile.

In Figura A.13 sono illustrate le tecniche della commutazione di circuito e della commutazione di messaggio a confronto, supponendo che siano interessati 3 nodi della rete (*A*, *B* e *C*). Si può osservare che, nella commutazione di messaggio, la trasmissione inizia immediatamente, senza alcuna predisposizione dell'interconnessione $A \rightarrow C$; ciò comporta una riduzione del tempo complessivo d'impegno del collegamento nel caso di messaggi *brevi* ([2]).

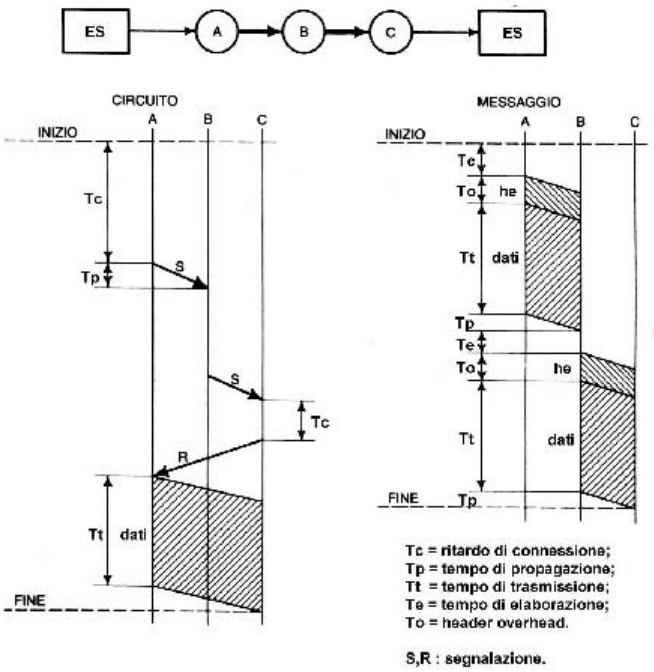


Figura A.13: Commutazione di *Circuito* vs Commutazione di *Messaggio*.

Osserviamo che, poiché non c'è limite alla *dimensione* del messaggio, una moderna implementazione di questa tecnica di commutazione richiederebbe delle memorie di massa presso ogni nodo della rete; inoltre, un grosso messaggio potrebbe impegnare una linea tra due nodi anche per minuti, bloccando il traffico alle sue spalle. Per questi motivi, nelle moderne reti di telecomunicazione, la commutazione di messaggio *non* è vantaggiosa: la tecnica store-and-forward è efficiente solo se i messaggi vengono frammentati in piccole unità informative.

4.3 Commutazione di pacchetto.

E' un'evoluzione della commutazione di messaggio, essendo anch'essa basata sulla tecnica store-and-forward; la differenza sostanziale è nella dimensione delle unità informative che

([2]) Grazie a questa proprietà, le reti a commutazione di messaggio hanno trovato una tipica applicazione nel *servizio*

circolano nella rete: un messaggio viene infatti suddiviso (prima della trasmissione) in piccole unità informative (ad esempio 1500 byte) dette pacchetti *(packets).*

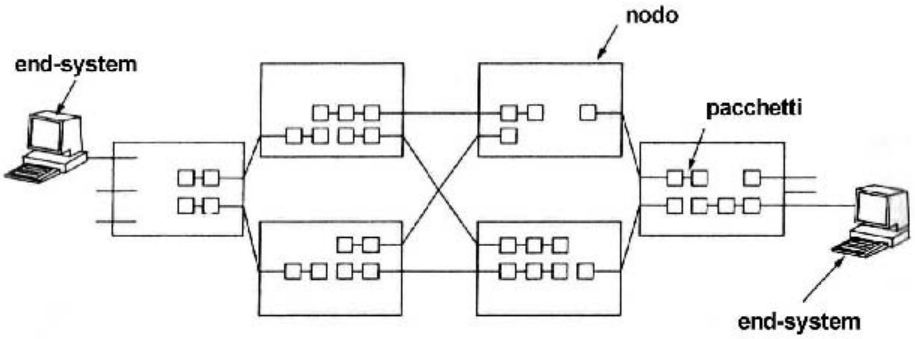


Figura A.14: Commutazione di Pacchetto.

Ogni pacchetto contiene un'intestazione *(header)*, che specifica l'indirizzo del destinatario più informazioni di controllo, seguita dai dati utili (carico o *payload*).

Il funzionamento è simile a quello della commutazione di messaggio: appena un end-system è pronto per la trasmissione, invia il primo pacchetto al nodo più vicino; quest'ultimo memorizza il pacchetto temporaneamente *(store)*, decide su quale linea d'uscita inoltrarlo ed infine lo incoda su quella linea. Non appena la linea è libera il pacchetto viene trasmesso *(forward).*

La suddivisione in pacchetti permette di ridurre il tempo di transito del messaggio attraverso la rete. Ciò è messo in evidenza nella Figura A.15 in cui si è supposto di trasmettere, attraverso *n* nodi, un messaggio divisibile in *p* pacchetti ([3]).

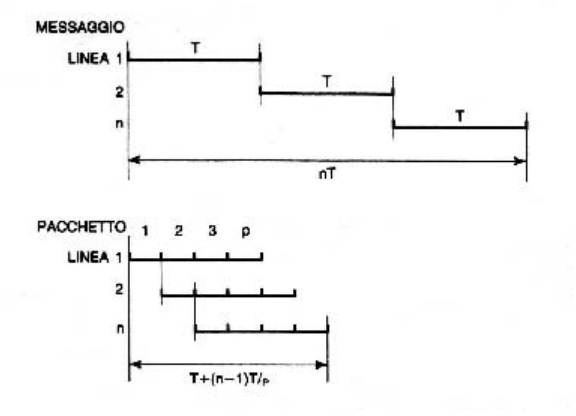


Figura A.15: Commutazione di *Messaggio* vs Commutazione di *Pacchetto*.

telegrammi.

([3]) Per semplicità si sono considerati trascurabili i tempi di propagazione, di elaborazione e di attesa nelle code.

Abbiamo due sottocasi (vedi Figura A.16 e Figura A.17):

– *Approccio a Datagramma.*

  Ogni pacchetto, detto datagramma *(datagram)*, è instradato indipendentemente dagli altri pacchetti che compongono lo stesso messaggio. In generale quindi, anche se diretti alla stessa destinazione, pacchetti diversi seguiranno *percorsi diversi*, giungendo a destinazione *fuori ordine*. Ogni nodo decide il percorso ottimale per ogni datagram che lo attraversa, basandosi sulla propria conoscenza dello stato (mutevole) della rete *(routing per pacchetto)*.

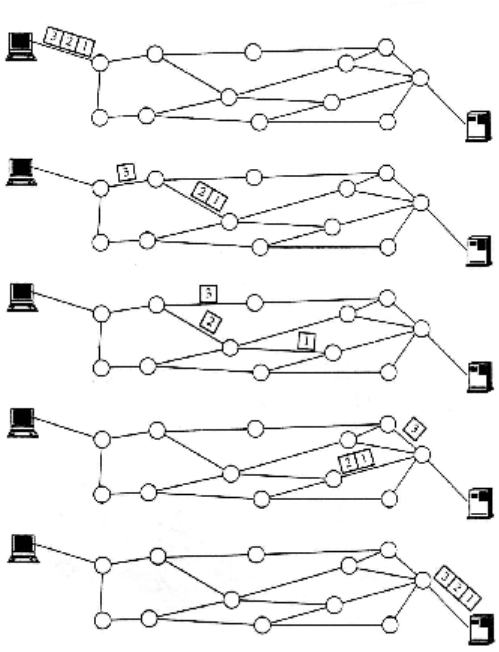


Figura A.16: Approccio a *Datagramma*.

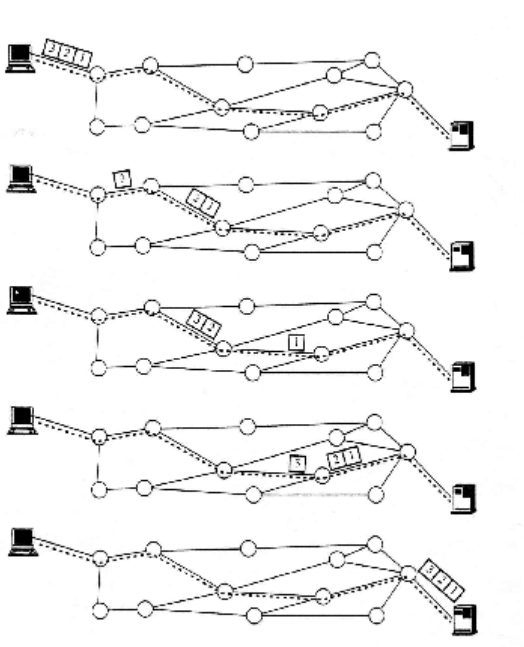


Figura A.17: Approccio a *Circuito Virtuale*.

– *Approccio a Circuito Virtuale.*

  Tutti i pacchetti di uno stesso messaggio vengono instradati allo stesso modo; il percorso, detto circuito virtuale *(virtual circuit)* in analogia con i circuiti della commutazione di circuito, viene deciso, una volta per tutte, *prima* che i pacchetti vengano spediti, utilizzando dei pacchetti di segnalazione. Con questa tecnica, ogni nodo non dovrà effettuare alcuna operazione di routing, eccetto nella fase di set up del circuito virtuale *(routing per sessione)*; inoltre, l'intestazione dei pacchetti che trasportano carico utile, anziché contenere un indirizzo completo, dovrà specificare solamente il *VC identifier*, necessario presso i nodi per commutare i pacchetti sulla *predefinita* linea d'uscita.

– *Datagram vs Virtual Circuit.*

Le differenze tra datagram e circuiti virtuali sono riassunte nella Tabella A.4.

	Datagram	**Virtual Circuit**
fase di set up	no	si
ogni pacchetto segue lo stesso percorso	no	si
i pacchetti vengono ricevuti in sequenza	no	si
indirizzamento	completo	identificatore
tempo di transito	elevato	basso
effetto di un guasto	pochi pacchetti persi	tutti i VC coinvolti vengono terminati
controllo delle congestioni	possibile ma complesso	semplice se è possibile prenotare le risorse

Tabella A.4: *Datagram* vs *Virtual Circuit*.

- Caratteristiche dei *Datagram*:
  1. *non* occorre una fase preliminare di *set up*;
  2. la rete può reagire efficacemente nel caso di nodi o linee malfunzionanti o congestionate, reinstradando i pacchetti altrove (elevata *affidabilità*);
  3. occorre *riordinare* i pacchetti presso il destinatario.

- Caratteristiche dei *Circuiti Virtuali*:
  1. i pacchetti vengono ricevuti *in sequenza*;
  2. la trasmissione è molto *rapida*: ogni nodo non deve ricalcolare il percorso per ogni pacchetto ma solamente commutarlo;
  3. se un nodo o una linea cessa di funzionare, tutti i VC che usano quel nodo o quella linea vengono necessariamente *terminati*.

5 Architetture di Rete.

5.1 Concetto di Architettura.

– *Il problema della comunicazione tra due end-system.*

Per ridurre la *complessità* della progettazione è opportuno suddividere il problema della trasmissione tra due o più utenti terminali in un insieme di sottoproblemi ragionevolmente autonomi e, come tali, più semplici da gestire. Ogni sottoproblema è risolto da un *modulo* (S/w o H/w); i moduli sono organizzati in una struttura *stratificata* per individuare diversi livelli d'astrazione, cioè vari modi di "vedere" la rete.

– *Strati o livelli (layers).*

Ogni *layer* ha il compito di offrire precisi *servizi* al livello superiore, nascondendo i dettagli della loro implementazione: il livello *n* è quindi un "fornitore di servizi" per il livello *n+1* ed un "utilizzatore di servizi" per il livello *n-1*.

Un livello soddisfa i seguenti requisiti:

1. svolge una ben determinata *funzione*;
2. è creato per realizzare un nuovo livello di *astrazione*;
3. i *limiti* dei livelli sono scelti in modo tale da minimizzare il flusso informativo alle interfacce;
4. l'implementazione di un livello può essere *sostituita* senza innescare modifiche nei livelli superiori o inferiori;
5. il *numero* di livelli è abbastanza grande da permettere a funzioni distinte di non essere forzatamente inserite nello stesso livello e, contemporaneamente, abbastanza piccolo da non appesantire l'intera architettura.

– *Interfacce.*

Due livelli adiacenti sono separati da un'*interfaccia*: essa definisce quali servizi il livello *n* rende disponibili al livello *n+1*; ogni servizio ha un indirizzo detto *SAP (Service Access Point)* e viene attivato usando dei comandi detti *primitive*.

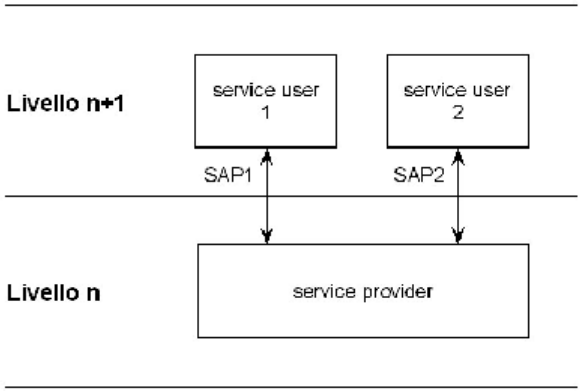


Figura A.18: Interfaccia tra due livelli.

La comunicazione tra due end-system *A* e *B* avviene facendo comunicare i layers corrispondenti sui due sistemi (chiamati entità pari o *"peers"*), anche se, come si vedrà, tale dialogo è generalmente indiretto.

– *Protocolli.*

La comunicazione tra entità pari del livello *n* avviene per mezzo dello scambio di blocchi di dati (detti *Protocol Data Unit* o *PDU*), formattati secondo delle convenzioni dette *"protocollo del livello n"*. Una PDU si compone di un'intestazione o *header*, eventualmente di una coda o *trailer*, contenenti le informazioni di controllo scambiate dalle entità pari, e di un campo dati (carico o *payload*) contenente la PDU del livello *n+1*.

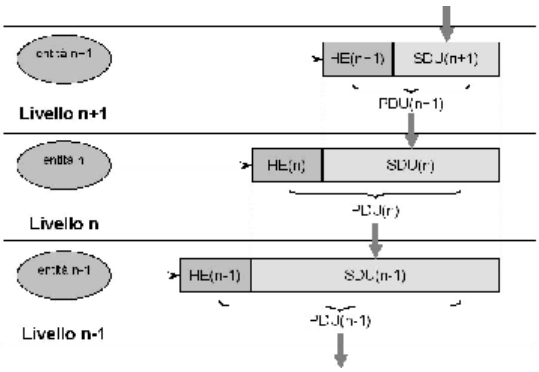


Figura A.19: Trasmissione.

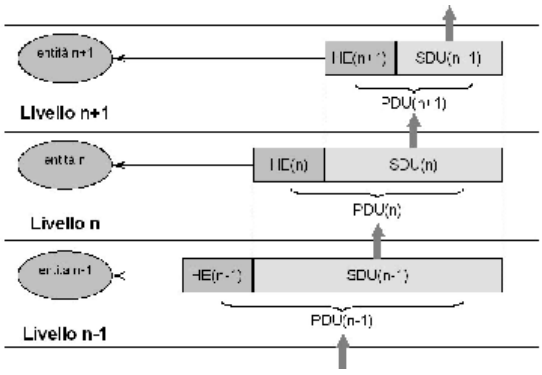


Figura A.20: Ricezione.

Quest'ultima, nel contesto del livello *n*, è detta più propriamente *SDU (Service Data Unit)* del livello *n*.

Le convenzioni del protocollo riguardano i seguenti aspetti:

1. *sintattico* (la struttura dei dati);
2. *semantico* (il significato dei dati);
3. *procedurale* (la sequenza di eventi).

– *Architettura di Rete.*

Un'architettura di rete è definita specificando i *livelli*, le *interfacce* ed i *protocolli*.

– *Comunicazione tra pari.*

Consideriamo un'applicazione presente sull'end-system *A* che desidera comunicare con un'applicazione presente sull'end-system *B*; supponiamo che tra i due end-system esista una connessione fisica diretta (Figura A.21).

L'applicazione invia l'informazione da trasmettere al livello *N* dell'architettura: poiché il livello *N* non possiede una connessione con il suo pari, aggiunge un header contenente ciò che ha da dirgli e passa tutta la *PDU(N)* al livello *N-1*. La discesa dei layer prosegue, con i dati che crescono complessivamente di dimensione, finché non viene raggiunto il *livello fisico* (layer *1*), in corrispondenza del quale la comunicazione può realmente aver luogo (Figura A.19).

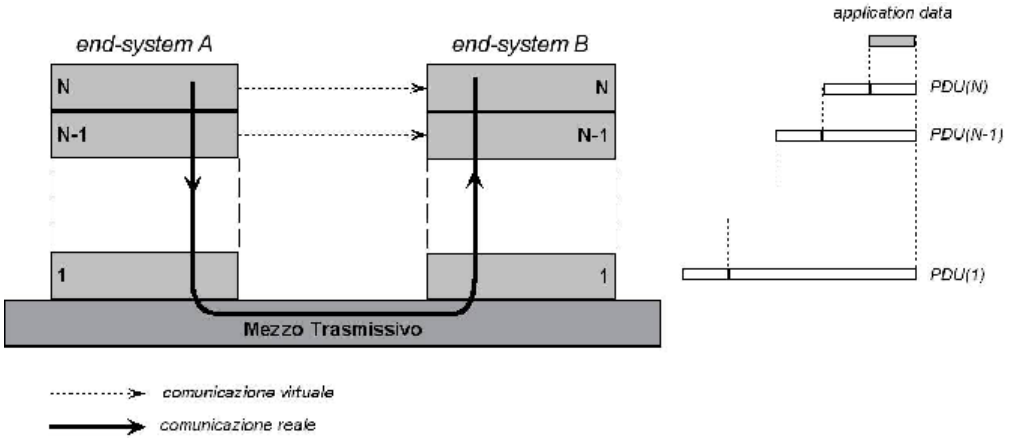


Figura A.21: Comunicazione tra entità pari.

Ricevuta la *PDU(1)*, il livello *1* dell'end-system *B* estrae il suo header ed invia l'*SDU(1)* al livello *2*. Quest'ultimo preleva il suo header, contenente il messaggio proveniente dal suo pari, ed inoltra il carico al livello superiore e così via (Figura A.20). I dati risalgono così i vari

strati ed infine vengono consegnati all'applicazione destinataria; nel contempo ogni livello ha potuto dialogare con il suo pari.

In conclusione, i processi di un dato livello *n*, ad eccezione di quelli appartenenti al livello fisico (i quali dialogano direttamente attraverso il mezzo trasmissivo), comunicano utilizzando un servizio del livello *n-1*: ogni livello dell'architettura utilizza le prestazioni del livello inferiore per fornire al livello superiore un servizio più "pregiato" ([4]).

5.2 Necessità di un modello di riferimento dei protocolli.

Il problema del trasporto dell'informazione è alquanto complesso: nel 1980 la *International Organization for Standardization* (ISO) comprese la necessità di definire un modello generale di rete che favorisse i costruttori *(vendors)* nella realizzazione di dispositivi di rete compatibili. Nacque così il modello OSI *(Open System Interconnection)* che rapidamente divenne un modello di riferimento universalmente riconosciuto.

Oggi esistono numerosi modelli architetturali, spesso proprietari, ma tutti possono essere ricondotti in modo più o meno diretto al modello OSI: tale confronto è spesso utilmente proposto proprio per favorire una rapida comprensione delle nuove (e vecchie) architetture.

5.3 Livelli OSI.

– *Caratteristiche generali.*

Il modello OSI è composto da *7 livelli*, ciascuno chiamato a svolgere un insieme ben definito di funzioni ([5]).

Per meglio comprendere la struttura del modello OSI, cerchiamo di individuare alcuni elementi essenziali per il funzionamento di una rete di telecomunicazioni.

Certamente è essenziale l'implementazione di un *servizio di trasporto* che, indipendentemente dal mezzo trasmissivo utilizzato, possa trasferire informazione tra due o più utenti della rete, possibilmente senza errori e con le necessarie prestazioni.

([4]) Si dice, pertanto, che un'architettura a strati funziona secondo il principio del "*valore aggiunto*": la sua descrizione definisce, per ogni livello *n*, i servizi resi disponibili dal livello inferiore *n-1* e quelli offerti al livello superiore *n+1*, vale a dire il "valore aggiunto" dal livello *n*.

([5]) Il modello di riferimento OSI definisce solamente le *funzioni* di ciascun livello: poiché non vi è alcun accenno né alle interfacce, né ai protocolli, il modello OSI *non* è un'architettura.

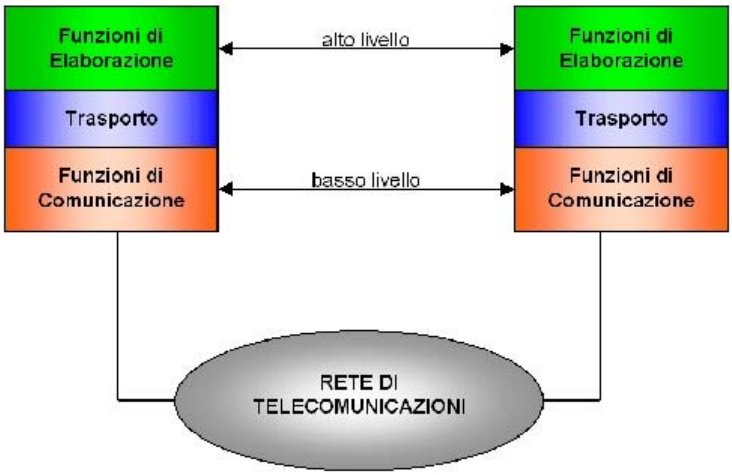


Figura A.22: Funzioni OSI.

Tale servizio è collocato, in OSI, al *livello 4*. La sua indipendenza dalle tecnologie impiegate nella rete è ottenuta mediante opportuni *protocolli di basso livello*, mentre la cooperazione tra i processi applicativi, poiché oltrepassa il campo delle comunicazioni per investire quello dell'elaborazione, richiede un insieme di *protocolli di alto livello* (vedi Figura A.22).

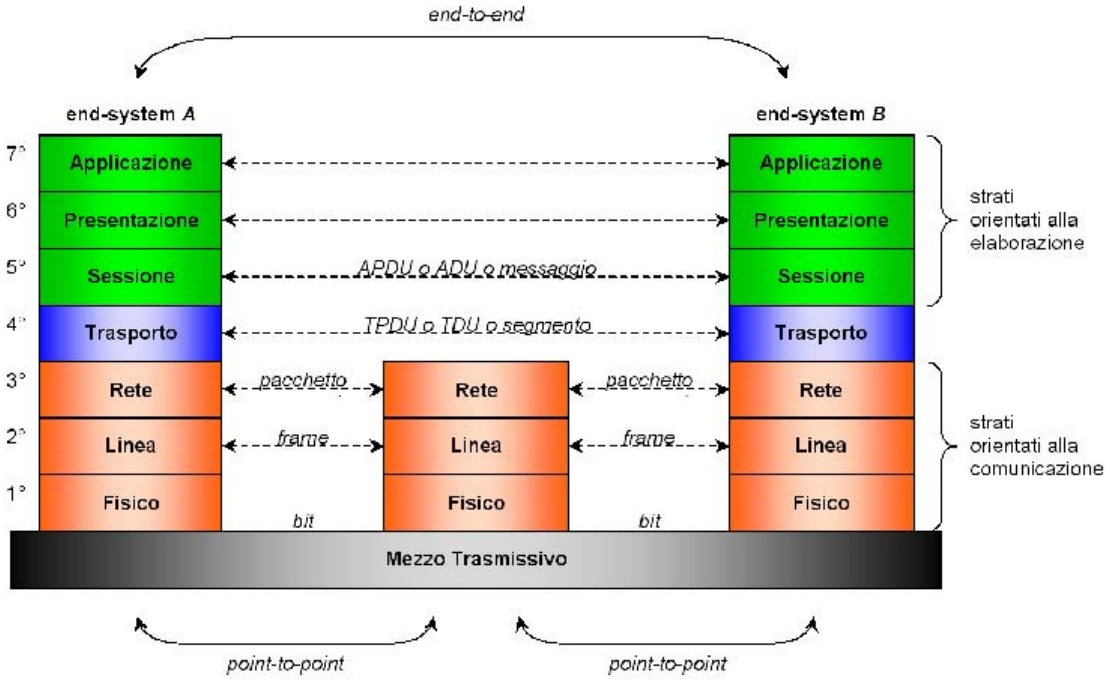


Figura A.23: Livelli OSI.

Per chiarire la differenza tra protocolli di basso ed alto livello, un esempio molto semplice è quello del *telefono*, dove nei primi possono essere comprese tutte le operazioni necessarie per

l'attivazione del collegamento tra i due interlocutori, e nei secondi le procedure per effettuare la conversazione vera e propria (riconoscimento reciproco, argomento noto ad entrambi, impiego della stessa lingua).

In OSI (Figura A.23), i protocolli di basso livello comprendono i primi 3 strati: il livello *fisico*, il livello di *linea* ed il livello di *rete*, con funzioni orientate alla comunicazione; mentre i protocolli di alto livello includono gli ultimi 3 strati (*sessione*, *presentazione* ed *applicazione*), con funzioni orientate verso l'elaborazione; al centro, si colloca il livello di *trasporto*.

5.3.1 L1: livello fisico (physical).

Definisce le specifiche elettriche, meccaniche e procedurali necessarie all'attivazione, gestione e disattivazione del *mezzo trasmissivo* posto sotto il livello fisico. Queste specifiche riguardano connettori, trasmettitori, ricevitori, segnali, forme d'onda, livelli di tensione, temporizzazioni, bit-rate, distanze, ecc.

Al livello fisico l'informazione si presenta sotto forma di *flusso di bit*: non è possibile perciò riconoscere il significato dei bit né vedere la loro suddivisione in blocchi di dati.

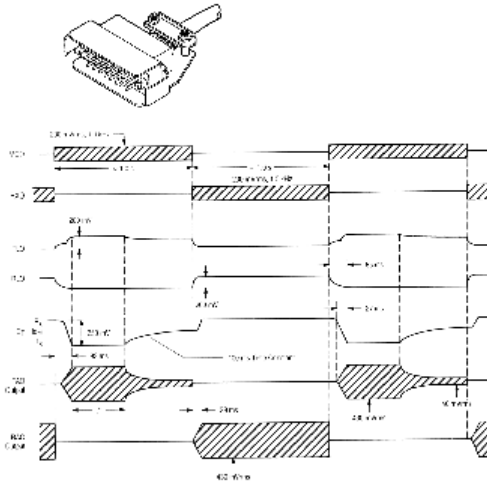


Figura A.24: Il livello fisico.

Particolare importanza riveste invece il problema della corretta *individuazione del livello logico* dei bit ricevuti; è fondamentale "sondare" ciascun bit in un istante scelto in modo tale da cadere con la massima probabilità nella zona centrale del "tempo di cifra". Ciò può essere attuato in *due* modi:

– *Trasmissione Asincrona.*

  La linea è normalmente in uno stato di riposo *(idle)*: prima di inviare il 1° bit, il trasmettitore deve portare la linea in condizioni di lavoro (impulso di *start*). Il ricevitore, quando rileva tale cambiamento, attiva un clock interno, isofrequenziale con quello di trasmissione. Conclusa la trasmissione dell'ultimo bit, la linea viene riportata nello stato di attesa (impulso di *stop*). Pertanto, è sufficiente che la sincronizzazione del clock, presso il destinatario, venga

mantenuta *solamente* per un intervallo di tempo *limitato*, quello compreso tra gli impulsi di start e stop.

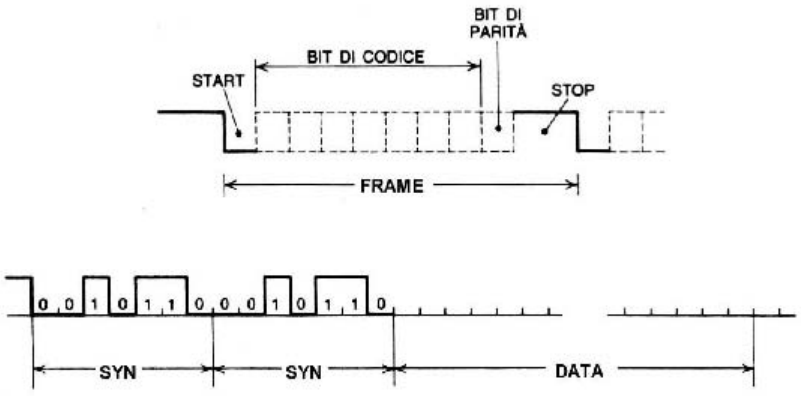


Figura A.25: Trasmissione Asincrona e Sincrona.

– *Trasmissione Sincrona.*

Non esistono impulsi di sincronizzazione: i bit sono trasmessi in modo continuo, senza alcuna interruzione. Presso il ricevitore, la temporizzazione necessaria viene estratta dallo stesso segnale ricevuto, grazie ad un circuito ausiliario.

Se il mezzo trasmissivo ha una banda utile incompatibile con quella del segnale che si desidera trasmettere, è necessario impiegare dei ricetrasmettitori opportuni *(tranceiver)* che facciano uso di forme d'onda con migliori caratteristiche spettrali (favorendo, possibilmente, l'estrazione della temporizzazione) oppure, se è necessario "spostare" nettamente la collocazione spettrale del segnale, dei veri e propri modulatori/demodulatori *(modem)*.

Maggiori dettagli sul livello fisico sono riportati in Appendice 1.

5.3.2 L2: livello di linea (data-link).

Il livello fisico accetta un flusso di bit ad un estremo della linea e lo consegna all'altro estremo senza garanzie: il numero di bit consegnati può essere maggiore, minore o uguale del numero di bit trasmessi ed alcuni bit possono assumere valori errati. Il livello di linea ha il compito di presentare al livello di rete un servizio di comunicazione point-to-point che trasmette frame, anziché flussi non specificati di bit, ed in grado di rilevare (e qualche volta anche correggere) eventuali errori di trasmissione (Figura A.26).

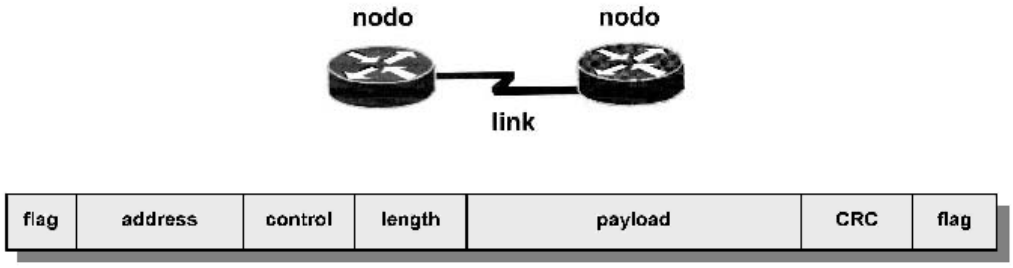


Figura A.26: Visibilità a livello di linea.

Un tipico protocollo di linea è HDLC (High-level Data Link Control) dal quale sono stati derivati molti protocolli "specializzati" come, ad esempio, LAP (Link Access Procedure), LAPB (Link Access Procedure, Balanced), IEEE 802.2, Ethernet (802.3) nelle sue varie realizzazioni, Token Ring (802.5) e Token Bus (802.4), PPP (Point-to-Point Protocol).
Distinguiamo le funzioni del livello di linea in base al tipo di rete:

– *Rete punto-punto.*

Le funzioni fondamentali sono le seguenti:

- Impacchettamento dei bit *(framing)*.
  I limiti dei frame vengono riconosciuti o creati tramite delle particolari sequenze di bit *(flag byte)* inserite in testa ed in coda al frame. Se la stessa sequenza occasionalmente compare nei dati viene modificata in modo noto con un procedimento detto *bit stuffing.* Eventualmente, può essere aggiunto un campo che contiene la dimensione del frame (ad esempio in byte) in modo da eseguire un controllo incrociato.
- Rilevazione degli errori *(error checking)*.
  Utilizzando un campo di *checksum* il destinatario può verificare se il frame è esente da errori; normalmente la checksum contiene un codice a ridondanza ciclica (*Cyclic Redoundancy Check* o *CRC*). I provvedimenti presi in caso di errore dipendono dal tipo di architettura: nelle architetture tradizionali viene chiesta la *ritrasmissione* del frame (*Automatic Repeat Request* o *ARQ*), in quelle *fast-packet* un frame errato viene semplicemente *scartato*, rimandando al livello di trasporto (end-to-end) la gestione delle ritrasmissioni (se è necessario un servizio affidabile).
- Controllo del flusso *(flow control).*
  Consente all'entità ricevente di regolare il flusso di frame trasmesso dalla entità sorgente. Al livello 2 questo controllo viene esercitato su ogni *linea*. La regolazione evita che una

sorgente troppo veloce inondi di frame un ricevitore lento, causando l'*overflow* dei buffer e la conseguente perdita dei frame in eccesso. Spesso il controllo del flusso è integrato nel meccanismo di controllo degli errori (se previsto).

– *Rete broadcast.*

Il mezzo trasmissivo è condiviso da tutti gli utenti della rete: per disciplinare gli accessi, è necessario implementare un meccanismo di contesa (ad accesso ordinato o casuale).

Il livello data-link è in tal caso suddiviso in 2 sottolivelli (vedi Figura A.27):

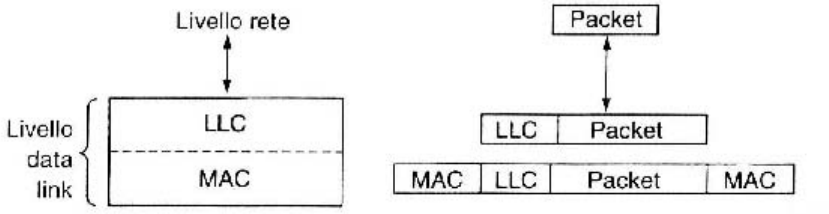


Figura A.27: Sottolivelli data-link.

- LLC (Logical Link Control).
  Svolge le funzioni a, b, c come nelle reti punto-punto.
- MAC (Medium Access Control).
  Implementa il protocollo di accesso multiplo (Aloha, CSMA/CD, ecc.).

Maggiori dettagli sul livello 2 sono riportati in Appendice 2.

5.3.3 L3: livello di rete (network).

E' un livello complesso che si occupa di trasmettere *pacchetti* dalla sorgente alla destinazione (cioè tra end-points): questa funzione è chiaramente diversa da quella del livello data-link che ha il compito ben più modesto di trasportare *frame* da un estremo all'altro di una *linea* (vedi Figura A.28). Generalmente l'interfaccia rete/trasporto rappresenta il *confine della rete*: l'utente ha cioè il controllo sui protocolli e sulle interfacce da utilizzare dal livello di trasporto in sù.

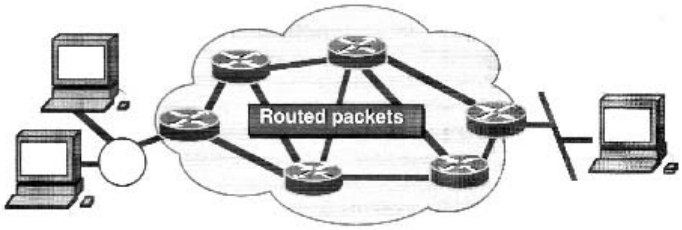


Figura A.28: Visibilità a livello di rete.

Distinguiamo nuovamente tra:

– *Rete punto-punto.*

Le funzioni di livello 3 sono:

- Instradamento *(routing).*

  Per permettere a due o più utenti di comunicare (unicast, multicast o broadcast), il livello di rete deve conoscere la *topologia* della rete (in modo totale o parziale) e scegliere percorsi appropriati per i pacchetti implementando un opportuno *algoritmo di routing*. E' al livello di rete che si pone l'importante distinzione tra:

  ⇨ Approccio a Datagram: *i pacchetti di uno stesso messaggio seguono percorsi generalmente diversi (routing per pacchetto);*

  ⇨ Approccio a Circuito Virtuale: *tutti i pacchetti di uno stesso messaggio seguono lo stesso predefinito percorso (routing per sessione).*

  Il percorso ottimale tra due end-points dipende da moltissimi fattori, ad esempio:

  - numero di linee e nodi coinvolti,
  - lunghezza delle linee,
  - bilanciamento del carico,
  - presenza di congestioni,
  - costo delle linee,
  - ecc.

  Nei casi di comunicazione *multicast* o *broadcast* il livello di rete deve evitare, fin quando è possibile, la duplicazione dei pacchetti provenienti dalla stessa sorgente ma diretti a più destinazioni, cercando di "clonare" i pacchetti solo in corrispondenza delle diramazioni del percorso (Figura A.29).

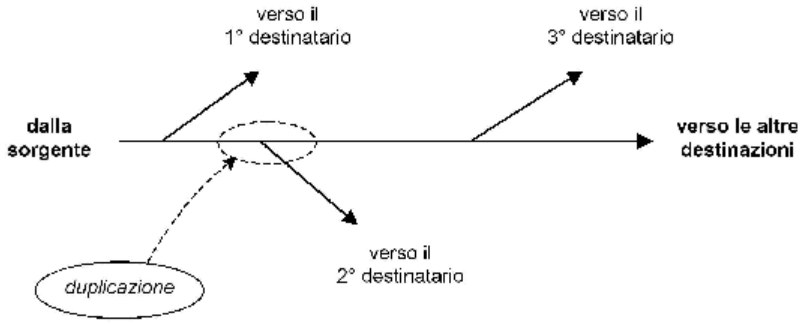


Figura A.29: Esempio di distribuzione *multicast*.

- Controllo delle congestioni *(congestion control).*

  Quando nella rete (o in una sua parte) sono presenti troppi pacchetti, le prestazioni degradano vistosamente: questo fenomeno è detto *congestione*. In presenza di congestioni le code nei nodi si allungano e cominciano a perdere pacchetti, allungando i tempi di trasporto. Le sorgenti tendono ad alimentare la congestioni una volta innescate: i pacchetti che non giungono a destinazione entro un certo tempo vengono generalmente ritrasmessi, anche più di una volta.

  Il livello di rete ha il compito di evitare concentrazioni di traffico distribuendo il carico sulla rete in modo uniforme *(load-balancing)*, di attivare in caso di overload eventuali linee di riserva *(backup)* ed infine, qualora le precedenti misure non siano sufficienti, ridurre i flussi d'ingresso ai confini della rete e/o negare nuove richieste di servizio.

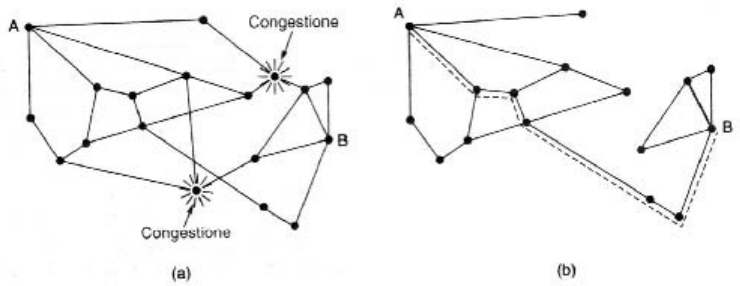


Figura A.30: (a) una rete congestionata; (b) rete ridisegnata che evita le congestioni.

– *Rete broadcast.*

Il livello di rete è assente.

5.3.4 L4: livello di trasporto (transport).

Il livello di trasporto costituisce la *"frontiera"* tra le funzioni di comunicazione e quelle di elaborazione. Esso dispone di mezzi per l'apertura, il mantenimento e la chiusura di *connessioni di trasporto* per la trasmissione di dati full-duplex tra interlocutori *end-to-end*; tutto ciò mascherando qualsiasi problema relativo alla trasmissione e garantendo, quindi, l'indipendenza delle funzioni di elaborazione dalle caratteristiche della rete utilizzata. Conseguentemente, a partire dal livello di trasporto, la rete *non* è più visibile (Figura A.31).

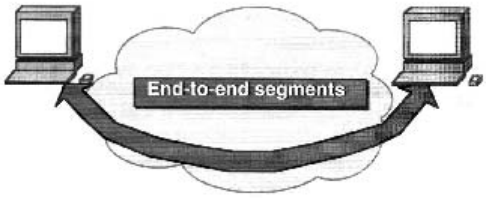


Figura A.31: Visibilità a livello di trasporto.

Funzioni principali:

- Fornitore di servizi di trasporto.
  L'entità di trasporto mette a disposizione delle applicazioni dei servizi di comunicazione *connection-oriented* oppure *connection-less*, *affidabili* o *non-affidabili* ed eventualmente con una specifica *QoS*. Nel caso connection-oriented gestisce le operazioni di creazione *(set up)* e rilascio ordinato *(tearing down)* delle connessioni.

- Frammentazione dei messaggi *(fragmentation)*.
  Lunghi messaggi provenienti dalle applicazioni vengono suddivisi in segmenti di dimensione opportuna. Eventuali ritrasmissioni interessano così solamente un segmento anziché l'intero messaggio. Come conseguenza della frammentazione, è necessario arricchire l'header con dei numeri di sequenza.

- Gestione degli errori *(error recovery)*.
  Il livello di rete fornisce generalmente un servizio non-affidabile.
  Se un'applicazione richiede un servizio *affidabile*, il livello di trasporto ha il compito di assicurarsi che tutti i segmenti giungano correttamente a destinazione, eventualmente dopo ripetute *ritrasmissioni* (ARQ). La tecnica qui implementata differisce da quella possibilmente presente a livello di linea; gli scopi sono, infatti, diversi: il *livello 4* si

occupa di trasmettere in modo affidabile interi messaggi su un percorso *end-to-end*, il *livello 2* di trasmettere senza errori piccoli frame su una linea *punto-punto*.

- Controllo del flusso *(flow control)*.

Al livello 4 il controllo viene esercitato per evitare che l'end-system *A* (mittente) inondi di segmenti l'end-system *B* (destinatario). Il controllo del flusso è spesso integrato nel meccanismo ARQ (protocolli *sliding windows*). La Figura A.32 illustra la differenza tra regolazione del flusso per connessione e regolazione del flusso a livello di linea.

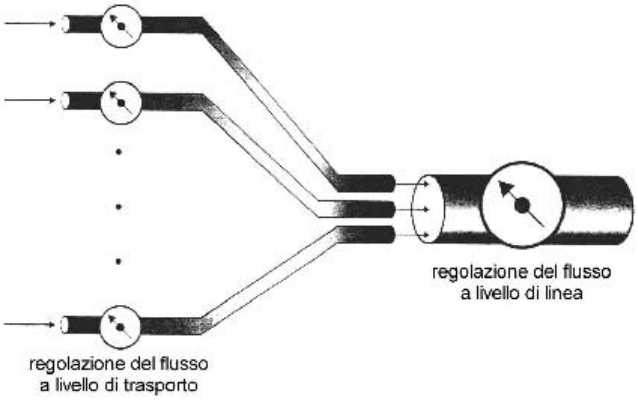


Figura A.32: Regolazione del flusso.

5.3.5 L5: livello di sessione (session).

E' il più basso tra i livelli di elaborazione; la sua funzione è quella di consentire lo scambio di messaggi tra *end-points* per mezzo di una connessione logica esente da interruzioni ed errori. Ogni connessione di sessione utilizza *una* connessione di trasporto, la quale non è necessariamente impegnata per tutta la durata della sessione: il livello di sessione, infatti, può autonomamente decidere di chiudere una connessione di trasporto e di riaprirne un'altra, qualora ciò fosse conveniente, senza produrre conseguenze sui livelli superiori. Se, ad esempio, si verifica un guasto mentre la trasmissione è in corso, la connessione di trasporto corrente viene *terminata* e ne viene *aperta* una seconda, riattivando così il trasferimento delle unità informative esattamente dal punto in cui era stato interrotto.

5.3.6 L6: livello di presentazione (presentation).

Consente la corretta *interpretazione* dei messaggi scambiati dalle *applicazioni*, indipendentemente dai formati, dai codici e da tutte le altre convenzioni impiegate in ogni end-

system. Si può dire che lo strato di presentazione agisce come un *interprete*: la situazione è infatti analoga a quella di due persone di lingue diverse che possono liberamente dialogare grazie a due interpreti che traducono i loro discorsi in un linguaggio assunto come riferimento comune (la lingua inglese, per esempio).

5.3.7 L7: livello di applicazione (application).

E' il livello OSI più vicino all'utente; esso ha la caratteristica di fornire servizi non ad un livello OSI, ma ad un'applicazione che è *fuori* dalla copertura del modello. I livelli inferiori (L1..L6) forniscono un supporto di comunicazione tra 2 o più end-system affidabile, ma non effettuano nessun compito *utile* per l'utente: al livello 7 troviamo dei protocolli che svolgono funzioni utili alle applicazioni reali e che pertanto non ha senso *replicare* in ciascuna di esse (ciò giustifica l'introduzione di un ulteriore layer sopra il livello 6).

Lo scenario è piuttosto vasto; abbiamo protocolli opportuni per svolgere varie *funzioni*:

- *trasferimento di file (FTP),*
- *posta elettronica (SMTP),*
- *newsgroup (NNTP),*
- *collegamento di documenti ipertestuali (HTTP),*
- *traduzione di indirizzi mnemonici in indirizzi di rete (DNS),*
- *crittografia, autenticazione, firme digitali,*
- *gestione e manutenzione della rete (SNMP),*
- *codifica e decodifica di audio e video in tempo reale,*
  ecc.

La Tabella A.5 riassume le funzioni principali dei vari livelli OSI.

livello	nome	funzioni
L1	*Fisico*	- attivazione/disattivazione del MT; - sincronizzazione e decisione; - eventuale modulazione/demodulazione.
L2	*Linea*	- framing; - controllo degli errori; - controllo del flusso (escluso fast-packet); - controllo dell'accesso al MT (solo reti broadcast).
L3	*Rete*	- routing; - controllo delle congestioni.
L4	*Trasporto*	- fornitore di servizi di trasporto; - frammentazione dei messaggi; - gestione degli errori; - controllo del flusso.
L5	*Sessione*	crea una connessione logica esente da interruzioni ed errori.
L6	*Presentazione*	permette la corretta interpretazione dei messaggi.
L7	*Applicazione*	fornitore di servizi utente.

Tabella A.5: Funzioni dei livelli OSI.

6 Appendice 1

6.1 Il livello 1: fisico

Lo studio del livello fisico richiede un preliminare studio della teoria dei segnali che esula dagli scopi di queste note. Ci limiteremo ad osservare alcuni aspetti principali in maniera elementare.
L'informazione può essere trasmessa a distanza variando opportunamente una qualche caratteristica fisica del mezzo scelto per la trasmissione. Tale variazione si propaga, con una certa velocità, lungo il mezzo di trasmissione e dopo un certo tempo arriva all'altra estremità del mezzo, dove può venir rilevata. Ad esempio, se il mezzo è un cavo metallico, si può variare la tensione applicata ad un'estremità. Tale variazione di tensione verrà successivamente rilevata all'altra estremità. I mezzi trasmissivi sono sostanzialmente di tre tipi:

- *mezzi elettrici* (cavi): in essi il fenomeno fisico utilizzato è l'energia elettrica;
- *mezzi ottici* (LED, laser e fibre ottiche): in essi il fenomeno utilizzato è la luce;
- *mezzi wireless* (onde radio): il fenomeno fisico è l'onda elettromagnetica, una combinazione di campo elettrico e campo magnetico variabili, che si propaga nello spazio e che induce a distanza una corrente elettrica in un dispositivo ricevente (antenna).

In linea di principio, la trasmissione può avvenire con due modalità differenti:

☞ trasmissione di segnale analogico;

☞ trasmissione di segnale digitale.

Il primo può variare gradualmente in un intervallo costituito da un numero infinito di possibili valori; il secondo può variare solamente passando bruscamente da uno all'altro di un insieme molto piccolo di valori (da due a qualche decina). Si tenga presente però che il fenomeno fisico utilizzato non è digitale, ma analogico. Un segnale quindi non può passare istantaneamente da un valore ad un altro, ma impiegherà un certo tempo per effettuare la transizione. La conseguenza è che un mezzo fisico farà del suo meglio per trasportare un segnale digitale, ma non riuscirà a farlo arrivare esattamente com'è partito.
Dalla analisi di Fourier sappiamo che un generico segnale variabile nel tempo è equivalente ad una somma di funzioni sinusoidali aventi ciascuna una propria ampiezza e frequenza. Si può quindi rappresentare un segnale di durata T attraverso il suo spettro di frequenze, ossia attraverso la sua scomposizione in sinusoidi.

Qualunque segnale è quindi caratterizzato da un intervallo di frequenze nel quale sono comprese le frequenze delle sinusoidi che lo descrivono. Esso va sotto il nome di banda di frequenza del segnale.

Anche i mezzi trasmissivi sono caratterizzati da un intervallo di frequenze, detta *banda passante*, che rappresenta l'intervallo di frequenze che il mezzo fisico è in grado di trasmettere senza alterarle oltre certi limiti. Le alterazioni principali sono la *attenuazione* e l'introduzione di *ritardo*, che di norma variano al variare delle frequenze trasmesse. In generale, i mezzi trasmissivi: attenuano i segnali in proporzione alla distanza percorsa e alla frequenza del segnale; propagano i segnali a velocità proporzionali alle loro frequenze. Una conseguenza è che, per qualunque mezzo trasmissivo, la banda passante si riduce all'aumentare della lunghezza del mezzo stesso.

Perché un segnale sia ricevuto come è stato trasmesso, è necessario che la banda passante sia uguale o più ampia della banda di frequenza del segnale stesso. Altrimenti, il segnale viene privato di alcune delle sue armoniche (tipicamente quelle di frequenza più elevata) e viene quindi distorto, cioè alterato. Se un numero sufficiente di armoniche arriva a destinazione, il segnale è comunque utilizzabile. Per maggiori informazioni si vedano i teoremi di Nyquist e di Shannon.

6.2 I mezzi trasmissivi

Analizziamo ora brevemente i mezzi trasmessivi tipicamente utilizzati nelle reti di calcolatori.

– *Doppino telefonico*

Consiste di una coppia di conduttori in rame intrecciati l'uno coll'altro in forma elicoidale. Ciò fa si che si minimizzino le interferenze fra coppie adiacenti (due fili paralleli costituiscono un'antenna; se sono intrecciati no). Usato, storicamente, per le connessioni terminali del sistema telefonico (da casa alla centrale più vicina).

La larghezza di banda dipende dalla lunghezza, ma comunque si può trasmettere a diversi Mbps su distanze fino a qualche km. Esistono varie categorie di doppini, in particolare:

categoria 3: due fili isolati, leggermente attorcigliati. Quattro coppie contenute in una guaina di plastica. Comune nei cablaggi telefonici interni agli edifici (si possono avere quattro telefoni per stanza);

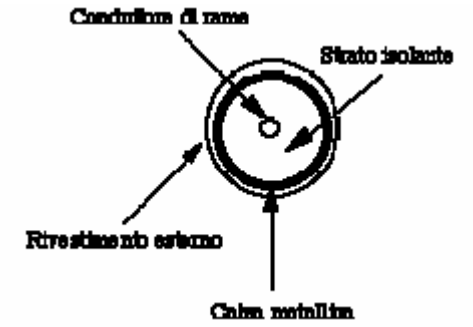
categoria 5: simile alla cat. 3, ma con un più fitto avvolgimento (più giri per centimetro) e con isolamento in teflon. Migliore qualità del segnale sulle lunghe distanze, adatto a collegamenti in alta velocità in ambito LAN (ad esempio per Ethernet a 100 Mbps, ATM a 34 Mbps).

Entrambi i tipi sono spesso chiamati UTP (Unshielded Twisted Pair), per distinguerli da un altro tipo, detto STP (Shielded Twisted Pair) che è schermato e quindi offre migliori prestazioni, ma è molto più ingombrante e, di fatto, non viene usato quasi più.

– *Cavo coassiale*

E' un altro comune mezzo di trasmissione; offre un miglior isolamento rispetto al doppino e quindi consente velocità di trasmissione maggiori su distanze superiori.

E' costituito da un conduttore centrale in rame circondato da uno strato isolante all'esterno del quale c'è una calza metallica.



In passato molto usato nel sistema telefonico per le tratte a lunga distanza, è ormai sostituito quasi ovunque dalla fibra ottica. Rimane in uso per la TV via cavo e in molte LAN. Tipi di cavo coassiale:

Baseband coaxial cable (50 ohm): il cavo baseband è usato per la trasmissione digitale, e consente velocità da 1 a 2 Gbps fino a circa 1 km. Per distanze superiori si devono interporre amplificatori. Per banda base intendiamo che l'intera banda passante è usata per una singola trasmissione, di tipo digitale.

Broadband coaxial cable (75 ohm): è usato per la trasmissione analogica. E' il cavo standard della TV. Offre una banda di 300 MHz e può estendersi fino a quasi 100 km. Per broadband si intende, nella telefonia qualunque trasmissione più ampia di 4 kHz; nella trasmissione dati si riferisce a un cavo su cui viaggia un segnale analogico che, con opportune tecniche di

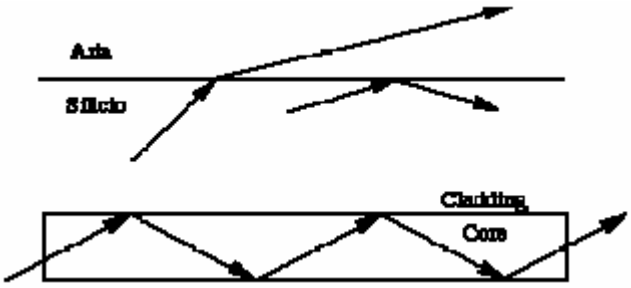
multiplazione, viene usato per effettuare contemporaneamente più trasmissioni distinte, separate in differenti bande di frequenza. La banda totale è suddivisa in canali di banda più piccola (ad es. 6 MHz per ciascun segnale TV) indipendenti gli uni dagli altri. Mentre un canale porta un segnale TV, un altro può portare una trasmissione dati (ovviamente con apparecchiature di conversione digitale/analogica e viceversa), tipicamente a 3 Mbps.

Tecnicamente, il cavo broadband è inferiore a baseband per la trasmissione digitale, ma ha il vantaggio di essere già in opera in grandi quantità.

– *Fibre ottiche*

Sono mezzi trasmessivi costituiti di un sottilissimo cilindro centrale in vetro, (*core*) circondato da uno strato esterno (*cladding*) di vetro avente un diverso indice di rifrazione e da una guaina protettiva. Sono quindi raggruppate insieme in una guaina esterna.

Le fibre ottiche sfruttano il principio della deviazione che un raggio di luce subisce quando attraversa il confine fra due materiali diversi (core e cladding nel caso delle fibre). La deviazione dipende dagli indici di rifrazione dei due materiali. Oltre un certo angolo, il raggio rimane intrappolato all'interno del materiale.



Deviazione del raggio luminoso

Tipi di fibre ottiche:

- *multimodali*: raggi diversi possono colpire la superficie con diversi angoli (detti mode), proseguendo quindi con diversi cammini. Il diametro del core è di 50 micron;
- *monomodali*: sono così sottili (il diametro del core è 8-10 micron) che si comportano come una guida d'onda: la luce avanza in modo rettilineo, senza rimbalzare. Sono più costose ma reggono distanze più lunghe (fino a 30 km).

Le fibre ottiche hanno prestazioni molto maggiori dei precedentemente analizzati mezzi trasmessivi: è facilmente raggiungibile una velocità di trasmissione di 50.000 Gbps (50 Tbps)

con un bassissimo tasso d'errore. La pratica attuale di usare velocità dell'ordine dei Gbps dipende dall'incapacità di convertire più velocemente segnali elettrici in luminosi. Infatti, nelle fibre ottiche, il mezzo fisico utilizzato è ovviamente la luce, e un impulso luminoso rappresenta un 1 mentre la sua assenza uno zero.

Le fibre ottiche sono costruite con un vetro speciale, molto trasparente, per cui offrono una bassissima attenuazione del segnale luminoso. L'attenuazione dipende anche dalla lunghezza d'onda della luce, per cui si usano comunemente tre particolari bande per la trasmissione (tutte nell'infrarosso vicino), larghe da 25.000 GHz a 30.000 Ghz ciascuna (un'enormità). Un sistema di trasmissione ottica ha tre componenti. La sorgente luminosa che può essere un LED o un laser, essa converte un segnale elettrico in impulsi luminosi; il mezzo di trasmissione che è la fibra vera e propria; il fotodiodo ricevitore che converte gli impulsi luminosi in segnali elettrici. Il tipico tempo di risposta di un fotodiodo è 1 nsec., da cui il limite di 1 Gbps.

Ci sono due topologie comuni per le reti basate su fibre ottiche:

- *anello*: mediante la concatenazione di più spezzoni di fibre ottiche si crea un anello. Tutti collegamenti sono punto a punto. L'interfaccia può essere passiva (fa passare l'impulso luminoso nell'anello) o attiva (converte l'impulso in elettricità, lo amplifica e lo riconverte in luce);
- *stella passiva*: l'impulso, inviato da un trasmettitore, arriva in un cilindro di vetro al quale sono attaccate tutte le fibre ottiche; viene poi distribuito alle fibre ottiche uscenti. Si realizza così una rete broadcast.

Vantaggi e svantaggi delle fibre ottiche:

- ☞ leggerezza a parità di larghezza di banda (due fibre sono più capaci di 1.000 doppini, 100 kg/km contro 8.000 kg/km);
- ☞ totale insensibilità a disturbi elettromagnetici;
- ☞ difficile inserimento di intrusi per spiare il traffico;
- ☞ elevato costo delle giunzioni;
- ☞ comunicazione unidirezionale (necessarie due fibre per comunicazione duplex).

– *Trasmissioni senza filo (wireless)*

Le onde elettromagnetiche, create dal movimento degli elettroni, viaggiano nello spazio (anche vuoto) alla velocità della luce e possono indurre una corrente in un dispositivo

ricevente (antenna) anche molto distante. Le porzioni dello spettro elettromagnetico utilizzabili per la trasmissione dati includono: onde radio; microonde; raggi infrarossi; luce visibile; raggi ultravioletti.

In generale, almeno per le onde radio, l'allocazione delle frequenze dipende da un'autorità statale.

Man mano che si sale di frequenza si hanno comportamenti diversi. Le onde radio, di frequenza più bassa, passano attraverso gli edifici, percorrono lunghe distanze e vengono riflesse dalla ionosfera; a frequenze più elevate (lunghezza d'onda dell'ordine dei cm o mm) sono estremamente direzionali e vengono fermate degli ostacoli, compresa la pioggia e in tutti i casi sono soggette a interferenze elettromagnetiche. La trasmissione (almeno per basse frequenze) è inerentemente di tipo broadcast.

Anche in questo ambito la velocità di trasmissione è funzione dell'ampiezza della banda utilizzata. Si trasmettono informazioni modulando l'ampiezza, la frequenza e/o la fase dell'onda.

7 Appendice 2

7.1 Il livello 2: data link

Esso ha il compito di offrire una comunicazione affidabile ed efficiente tra due macchine adiacenti, cioè connesse fisicamente da un canale di comunicazione (ad es.: cavo coassiale, doppino, linea telefonica). Esso si comporta come un "tubo digitale", cioè i bit partono e arrivano nello stesso ordine.

Compiti del livello:

- offrire servizi al livello network con un'interfaccia ben definita;
- determinare come i bit del livello fisico sono raggruppati in frame (*framing*);
- gestire gli errori di trasmissione;
- regolare il flusso della trasmissione fra sorgente e destinatario.

– *Servizi offerti al livello superiore (network):*

trasferire dati dal livello network della macchina di origine al livello network della macchina di destinazione mediante servizi di vario tipo:

- *connectionless non confermato*: si mandano frame indipendenti; i frame non vengono confermati; non si stabilisce una connessione; i frame persi non si recuperano. Esso è appropriato per: canali con tasso d'errore molto basso; traffico real-time (es. voce); LAN, nelle quali, in effetti, è molto comune.
- *connectionless confermato*: i frame vengono confermati mediante un acknowledgment (ack); se la conferma non arriva, il mittente rispedisce il frame; è utile su canali non affidabili (es. sistemi wireless). Richiedere la conferma a livello 2 non è indispensabile. Infatti la conferma può sempre essere fatta a un livello superiore, ma con linee disturbate ciò può essere gravoso.
- *connection oriented confermato*: prevede tre fasi (apertura connessione/invio dati/chiusura connessione); garantisce che ogni frame sia ricevuto esattamente una volta e nell'ordine giusto; fornisce al livello network un flusso di bit affidabile.

7.1.1 Un esempio di funzionamento

Si consideri un router con alcune linee in ingresso ed alcune in uscita. Il routing avviene a livello di rete, quindi esso gestisce i livelli uno, due e tre.

- ✓ Quando al router arrivano dei bit da una linea fisica, l'hardware apposito li passa al corrispondente SW/HW di livello due.
- ✓ Il SW/HW di livello due (data link), che in genere è contenuto in un chip sulla scheda (o adattatore) di rete esegue dei controlli, quali framing; controllo errori di trasmissione; controllo numero di frame (se necessario).
- ✓ Se i controlli hanno successo, il SW/HW di livello due genera un interrupt che chiama in causa il SW di livello rete:
- ✓ questo è tipicamente un processo di sistema, al quale viene passato il pacchetto contenuto nel frame di livello due per l'ulteriore elaborazione;
- ✓ l'elaborazione consiste nel determinare, sulla base delle caratteristiche del pacchetto (in particolare dell'indirizzo di destinazione), su quale linea in uscita instradarlo.
- ✓ Presa questa decisione, il SW di livello tre consegna il pacchetto al corrispondente SW/HW di livello due, che lo imbusta in un nuovo frame e lo consegna al sottostante livello fisico (ossia quello relativo alla linea in uscita prescelta).
- ✓ Il livello uno accetta un flusso di bit grezzi e cerca di farli arrivare a destinazione.

7.2 Rilevazione e correzione di errori

Il problema è che il flusso, in generale, non è esente da errori. E' compito del livello data link rilevare e, se possibile, correggere tali errori. L'approccio usuale del livello data link è il seguente:

in trasmissione → si spezza il flusso di bit che arriva dal livello tre in una serie di frame; si calcola, per ciascun frame, un'apposita funzione di controllo (checksum) per verificare di presenza errori; si inserisce il checksum nel frame; si consegna il frame al livello uno, il quale lo spedisce come sequenza di bit.

in ricezione → si riceve una sequenza di bit dal livello uno; si ricostruisce da essa un frame dopo l'altro; per ciascun frame si ricalcola il checksum; se esso è uguale a quello contenuto nel frame questo viene accettato, altrimenti viene considerato errato e quindi scartato.

Per delimitare i frame si usano degli appositi marcatori per designarne l'inizio e la fine. Esistono vari metodi:

a) conteggio dei caratteri. È poco utilizzato, poiché richiede un campo apposito nel frame per indicarne la lunghezza, ma se durante la trasmissione si deteriora proprio questo campo, diventa praticamente impossibile ritrovare l'inizio del prossimo frame e di conseguenza anche quello dei successivi;
b) caratteri di inizio e fine, con *character stuffing*. Ciascun frame inizia e finisce con una particolare la sequenza di caratteri ASCII. Tipicamente: per l'inizio del frame: *DLE (Data Link Escape), STX (Start of TeXt)*; per la fine del frame: *DLE, ETX (End of TeXt)*. In questo modo, se la destinazione perde traccia dei confini di un frame, la riacquista all'arrivo del frame successivo. Ovviamente, poiché DLE è un byte come qualsiasi altro, è necessario distinguere la sua presenza all'interno dei dati trasmessi e ciò si ottiene aggiungendo davanti ad un eventuale DLE nel *payload* un ulteriore DLE (che va rimosso all'arrivo);
c) bit pattern di inizio e fine, con *bit stuffing*. E' una variante del precedente, che prescinde dalla codifica ASCII ad 8 bit. Esso permette di avere un numero qualunque di bit all'interno del frame. Ogni frame inizia e finisce con uno specifico pattern di bit, ad es.: `01111110` chiamato flag byte. Ovviamente esiste un problema analogo al caso precedente, dato che anche la sequenza di bit può apparire all'interno del *payload* e non va confusa con un inizio di frame. Tipicamente, pertanto, ogni volta che il livello due incontra nei dati da trasmettere 5 bit consecutivi uguali a 1 inserisce uno zero aggiuntivo; all'arrivo, se nei dati ricevuti compaiono 5 bit uguali a uno, si rimuove lo zero che li segue;
d) violazione controllata della codifica dei bit usata nel livello fisico. I bit al livello fisico vengono normalmente codificati con una certa ridondanza per aumentare l'affidabilità. Ad esempio nella codifica Manchester, largamente utilizzata, il valore 1 di un bit di dati è codificato con la coppia high/low di bit fisici e il valore 0 di un bit di dati è codificato con la coppia low/high di bit fisici (le coppie low/low ed high/high non sono utilizzate). Ciò ovviamente introduce un dimezzamento della banda utile, ma consente una facile determinazione dei confini di un bit dati che ha sempre una transizione di stato. Poiché le coppie high/high e low/low sono non utilizzate in codifica possono essere usate per delimitare i frame.

7.3 Tecniche di correzione e rilevazione degli errori

Errori possono essere presenti, tipicamente dovuti a: rumore di fondo, disturbi impulsivi, interferenze. La rilevazione e gestione degli errori richiede sempre l'aggiunta di informazione ridondante:

- tecniche di correzione dell'errore: includere tanta informazione ridondante da poter ricostruire il messaggio originario;
- tecniche di rilevazione dell'errore: includere meno informazione ridondante, sufficiente a rilevare se c'è stato un errore, ma non a correggerlo.

Per comprendere l'uso di queste tecniche è utile partire dalla considerazione che normalmente un frame (a parte i delimitatori) consiste di: n = m + r bit, dove i primi m bit costituiscono il messaggio vero e proprio e gli altri r bit sono ridondanti.

Una sequenza di n bit fatta in tal modo si definisce *codeword*, o parola di codice. Date due qualunque parole di codice, ad es.:

```
1000 1001
1011 0001
```

è possibile determinare il numero di bit che in esse differiscono (tre nell'esempio) tramite un semplice XOR fatto bit a bit. Tale numero si dice distanza di Hamming delle due codeword. Se due codeword hanno una distanza di Hamming uguale a d, sono necessari esattamente d errori su singoli bit per trasformare l'una nell'altra. Un insieme prefissato di codeword costituisce un codice (code). La distanza di Hamming di un codice è il minimo delle distanze di Hamming fra tutte le possibili coppie di codeword del codice.

E' facile evincere che la capacità di rilevare o correggere gli errori dipende strettamente dalla distanza di Hamming del codice scelto. In particolare:

Per rilevare d errori serve un codice con distanza di Hamming (d+1). Infatti, in questo caso qualunque combinazione di d errori non riesce a trasformare un codeword valido in un altro codeword valido, per cui si ha per forza un codeword non valido, che quindi rivela il fatto che ci sono stati degli errori;

Per correggere d errori, serve un codice di Hamming con distanza (2d+1). Infatti in questo caso un codeword contenente fino a d errori è più vicino a quello originario che a qualunque altro codeword valido.

Dunque, a seconda degli scopi che si vogliono raggiungere, si progetta un algoritmo per il calcolo degli r check bit (in funzione degli m bit del messaggio) in modo che i codeword di n = m + r bit risultanti costituiscano un codice con la desiderata distanza di Hamming.

Esempio1. Un codice ottenuto mediante l'aggiunta di un bit di parità, calcolato in modo tale che il numero totale di bit uguali ad 1 sia pari (in questo caso si parla di parità pari)

```
<-- m --> r
1011 0101 1
1000 0111 0
```

Questo codice, detto *codice di parità*, ha distanza di Hamming uguale a due, e dunque è in grado di rilevare singoli errori. Infatti, un singolo errore produce un numero dispari di 1 e quindi un codeword non valido, ma un doppio errore, su due bit potrebbe non essere rilevato.

Esempio 2. Un codice costituito dalle seguenti codeword:

```
00000 00000
00000 11111
11111 00000
11111 11111
```

ha distanza 5, per cui corregge fino a due errori. Infatti, se viene trasmesso `00000 11111` ed arriva `00000 001111` si può risalire correttamente al codeword più vicino, che è `00000 11111`

Però, se viene trasmesso `00000 00000` e ci sono tre errori ed arriva `00000 00111` esso verrà interpretato erroneamente come `00000 11111` anziché come `00000 00000`

Per correggere un singolo errore su m bit, si devono impiegare almeno *r* check bit, con $2^r >= m + r + 1$, ossia sono necessari circa $\log_2 (m)$ bit. Esiste un codice (codice di Hamming) che raggiunge questo limite teorico.

Per correggere gruppi contigui di errori (detti *burst di errori*) di lunghezza massima prefissata (*k*) è sufficiente: accumulare *k* codeword, riga per riga; trasmettere le codeword per colonne e infine a destinazione riassemblare per righe. Un burst di *k* errori comporta un singolo errore in ciascuna codeword che quindi può essere completamente ricostruito.

I codici correttori di errore sono usati raramente (ad esempio in presenza di trasmissioni simplex, nelle quali non è possibile inviare al mittente una richiesta di ritrasmissione), perché in generale è molto più efficiente limitarsi a rilevare gli errori e ritrasmettere saltuariamente i dati piuttosto che impiegare un codice (più dispendioso in termini di ridondanza) per la correzione degli errori.

Ad esempio, si supponga uno scenario con canale con errori isolati, probabilità di errore uguale a 10^{-6} per bit e blocchi dati di 1.000 bit. Per correggere errori singoli su un blocco di

1.000 bit, sono necessari 10 bit, per cui un MBit richiede un overhead di 10.000 check bit. Al contrario, per limitarsi a rilevare l'errore in un blocco, basta un bit (con bit di parità). Avendo un tasso d'errore pari a 10^{-6}, solo un blocco su 1.000 è sbagliato e quindi deve essere ritrasmesso. Di conseguenza, per ogni Mbit si devono rispedire 1.001 bit (un blocco più il parity bit). L'overhead totale su un MBit è pertanto: 1.000 bit per parity bit su 1.000 blocchi + 1.001 bit per il blocco ritrasmesso. Tra i due casi ci sono pertanto circa 8.000 bit di differente overhead.

L'uso del parity bit può servire (con un meccanismo analogo a quello visto per la correzione di burst di *k* errori) per rilevare burst di errori di lunghezza <= *k*. La differenza è che non si usano *r* check bit per ogni codeword, ma uno solo.

La maggior parte delle implementazioni reali usa comunque una tecnica diversa, detta *Cyclic Redundancy Code* (CRC), nota anche come *polynomial code*. I polynomial code sono basati sull'idea di considerare le stringhe di bit come rappresentazioni di polinomi a coefficenti 0 e 1 (un numero ad *m* bit corrisponde ad un polinomio di grado *m*-1). Ad esempio, la stringa di bit 1101 corrisponde al polinomio $x^3 + x^2 + x^0$.
Notare che l'aritmetica polinomiale è modulo 2. In particolare: addizione e sottrazione sono equivalenti all'or esclusivo (non c'è riporto o prestito) e la divisione è come in binario, calcolata attraverso la sottrazione modulo 2.
Il mittente ed il destinatario si accordano su un polinomio generatore G(x), che deve avere il bit più significativo e quello meno significativo entrambi uguali ad 1. Supponiamo che G(x) abbia *r* bit.
Il frame M(x), del quale si vuole calcolare il checksum, dev'essere più lungo di G(x). Supponiamo che abbia *m* bit, con $m > r$.
L'idea è di appendere in coda al frame un checksum tale che il polinomio corrispondente (che ha grado $m + r - 1$) sia divisibile per G(x).
Quando al ricevitore giunge il frame più il checksum, divide il tutto per G(x). Se il risultato è zero il frame è giunto correttamente, altrimenti c'è stato un errore.
Il calcolo del checksum si effettua come segue:

- Appendere *r* bit a destra del frame, che quindi ha $m + r$ bit, e corrisponde ad $x^r M(x)$;
- Dividere $x^r M(x)$ per G(x);
- Sottrarre ad $x^r M(x)$ il resto della divisione effettuata al passo precedente.

Ciò che si ottiene è il frame più il checksum da trasmettere, che è ovviamente divisibile per G(x). Si noti che di fatto questa è un'operazione di XOR fatta sugli *r* bit meno significativi, e quindi non modifica il frame.

Questo metodo è molto potente, infatti un codice polinomiale con *r* bit rileva tutti gli errori singoli e doppi; rileva tutti gli errori di x bit, x dispari e rileva tutti i burst di errori di lunghezza <= *r*.

Tipici polinomi standard sono:

CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$;

CRC-12: $x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$;

CRC-16: $x^{16} + x^{15} + x^2 + 1$;

Un checksum a 16 bit corregge: errori singoli e doppi; errori di numero dispari di bit; errori burst di lunghezza <= 16; 99.997% di burst lunghi 17; 99.998% di burst lunghi 18.

N.B. Questi risultati valgono nell'ipotesi che gli *m* bit del messaggio siano distribuiti casualmente, il che però non è vero nella realtà, per cui i burst di 17 e 18 possono sfuggire più spesso di quanto calcolato teoricamente.

7.4 La gestione della sequenza e del flusso

Si è già introdotto il concetto di acknowledgement (*ack*), un messaggio inviato dal destinatario al mittente per informarlo circa l'esito della trasmissione di un frame.

Può ovviamente accadere che un frame sparisca del tutto, per cui il mittente rimarrebbe bloccato in attesa di un ack che non arriverà mai; in tal caso il mittente stabilisce un time-out per la ricezione dell'ack. Se questo non arriva in tempo, il frame si ritrasmette. Ma poiché anche il frame contenente l'ack può perdersi il mittente potrebbe inviare più copie dello stesso frame; in tal caso è necessario inserire numero di sequenza all'interno di ogni frame dati.

Per quanto attiene al controllo di flusso, è necessario impedire che il mittente spedisca dati più velocemente di quanto il destinatario sia in grado di smaltirli. Spesso si usano meccanismi basati sull'esplicita autorizzazione, data da parte del destinatario al mittente, di inviare un ben preciso numero di frame.

Analizzeremo nel seguito le caratteristiche di diversi protocolli, di complessità crescente, per il livello data link. Si assumerà che:

- ☞ nei livelli fisico, data link e network, ci sono processi (HW o SW) indipendenti, che comunicano fra loro, ad esempio scambiandosi messaggi;

- ☞ in trasmissione → il SW di livello data link riceve un pacchetto dal livello network, lo incapsula in un header ed un trailer contenenti informazioni di controllo; quindi vengono calcolati il checksum e i delimitatori (di norma a cura di un HW apposito di livello data link);
- ☞ il frame viene passato al livello sottostante, che trasmette il tutto;
- ☞ in ricezione → l' HW di livello data link identifica i delimitatori, estrae il frame, ricalcola il checksum: se è sbagliato, il SW di livello data link viene informato dell'errore; altrimenti il SW di livello data link riceve il frame (senza più checksum).
- ☞ il SW di livello data link, quando riceve un frame, esamina le informazioni di controllo (ossia lo header e il trailer):
- ☞ se tutto è corretto consegna il payload al livello network, altrimenti intrapende quanto è necessario per recuperare la situazione e non consegna il pacchetto al livello network.

Si assumerà inoltre la seguente struttura di un frame:

kind	Seq.	Ack.	payload

kind	servirà a distinguere il tipo di frame (contiene dati, è solo di controllo, ecc.)
seq	contiene il numero progressivo del frame
ack	contiene informazioni legate all'acknowledgement
payload	contiene un pacchetto di livello network completo (comprendente quindi le informazioni di controllo di tale livello).

7.4.1 Protocollo 1: Heaven

Pensato per 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 è privo di 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).

7.4.2 Protocollo 2: Simplex Stop and Wait

Rilascia esclusivamente l'ipotesi (poco realistica) che esista un buffer infinito nel ricevitore. Pertanto il mittente deve poter essere opportunamente rallentato per non perdere dati. 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.

– *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 è necessario un canale almeno half-duplex (c'è alternanza stretta nelle due direzioni).

7.4.3 Protocollo 3: simplex per canale rumoroso

Si assume un canale rumoroso. 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, non viene rilevato.
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:

- ❖ il frame dati x arriva bene;
- ❖ l'ack del frame x si perde completamente (gli errori non discriminano tra frame dati e frame di ack);
- ❖ il frame dati x viene inviato di nuovo e arriva bene;
- ❖ 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 introduce 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.
Il mittente trasmette i frame dati alternando zero ed uno nel campo seq e 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.

Questo tipo di protocolli, in cui il mittente aspetta un ack di conferma prima di trasmettere il prossimo frame, è noto come 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)

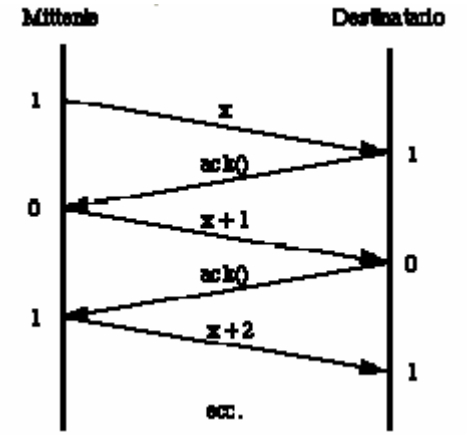
In sintesi: il mittente etichetta i frame dati con la sequenza `...0,1,0,1...`, ma passa alla etichetta e frame successivi solo quando arriva un ack; finché ciò non accade, continua a ritrasmettere lo stesso frame.

Il ricevente invia un ack di conferma per tutti i frame dati privi di errori, ma consegna al livello network solo quelli giusti, e cioé etichettati secondo la sequenza `...0,1,0,1...`.

Gli schemi seguenti illustrano varie eventualità che possono verificarsi durante il dialogo secondo il protocollo 3. I numeri ai lati delle figure indicano i numeri di sequenza che, rispettivamente:

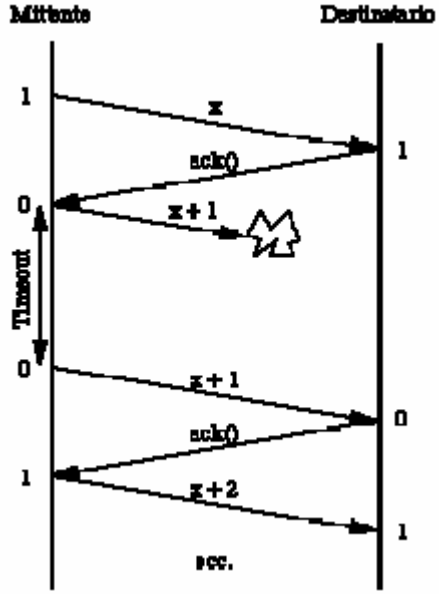
- il mittente usa per etichettare il frame dati da trasmettere;
- il destinatario usa per decidere se il prossimo frame che arriva va passato al livello network o no.

In assenza di errori il funzionamento è il seguente:



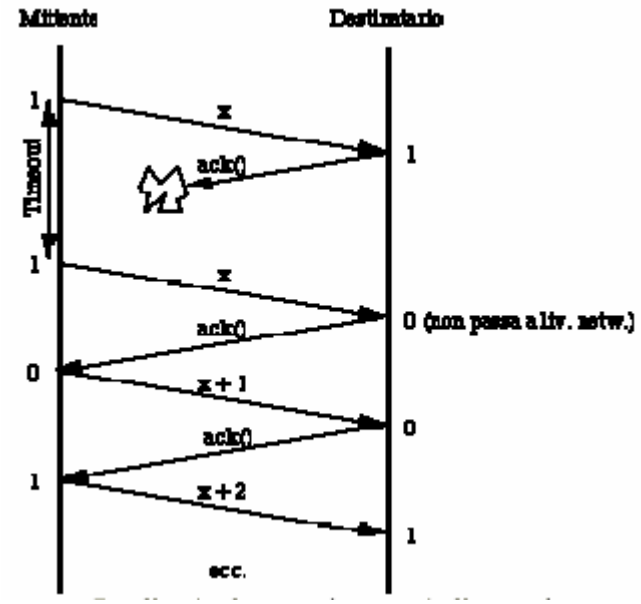
Normale funzionamento

Nel caso in cui un frame dati si perda o si danneggi, la situazione è la seguente:



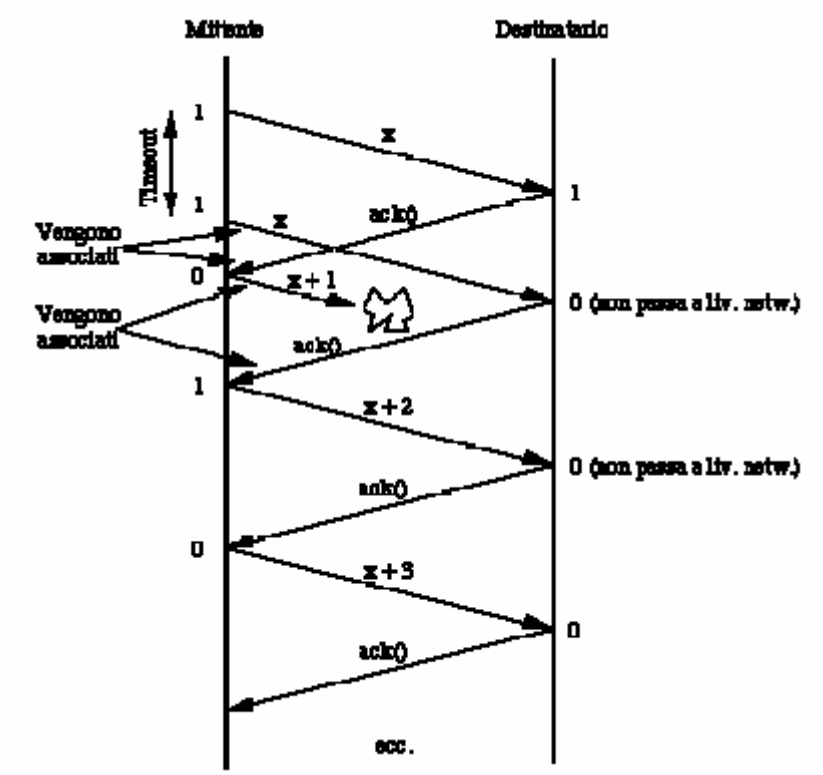
Perdita (o danneggiamento) di un frame

Nel caso in cui invece si perda o si danneggi un frame di ack, la situazione è la seguente:



Perdita (o danneggiamento) di un ack

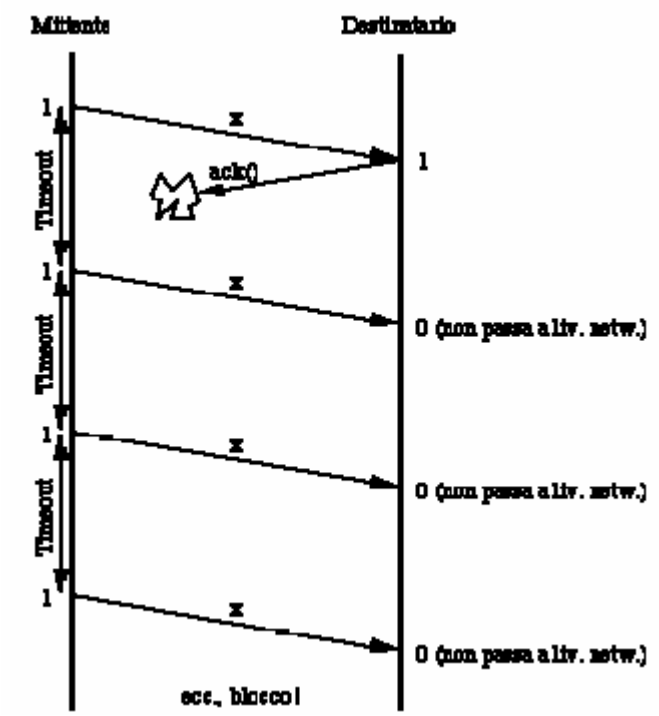
Un problema importante è legato, come abbiamo già accennato, alla lunghezza del timeout. Se esso è troppo breve, può succedere questo:



Timeout troppo ridotto

Nell'esempio, i frame dati (x+1) e (x+2) si perdono (nel senso che non vengono consegnati al livello network) e nessuno se ne accorge.

D'altronde, se cambiamo il protocollo mandando un frame di ack solo per i frame dati che non sono duplicati, ci si può bloccare:

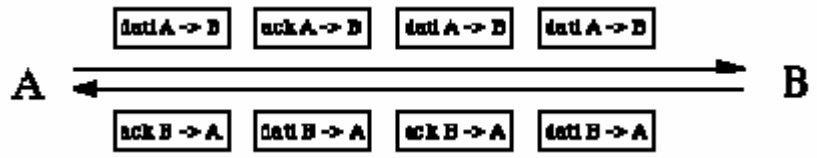


Ack inviato solo per frame non duplicati

7.4.4 Protocolli sliding window (a finestra scorrevole)

Si assume una trasmissione duplex (bidirezionale) in cui dati A e B sorgente e destinazione, rispettivamente:

- ☞ 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.



Comunicazione dati bidirezionale su unico canale

E' inoltre possibile che, quando si deve inviare un ack da B ad A, si aspetti fino a che è pronto un frame dati che B deve inviare ad A, si può "fare autostop" e introdurre dentro tale frame dati anche le informazioni relative all'ack in questione. Questa tecnica è nota come piggybacking (letteralmente, portare sulle spalle).

Piggybacking

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 e uso di CPU dato che 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).

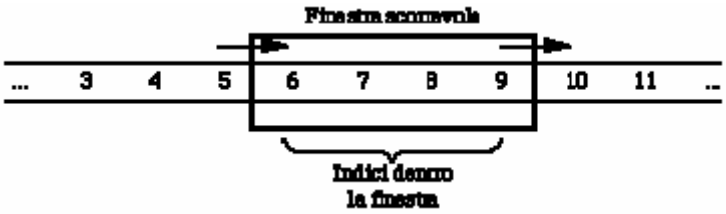
Però c'è un aspetto da non trascurare: per quanto si può aspettare un frame su cui trasportare un ack che è pronto e deve essere inviato? Non troppo, perché se l'ack non arriva in tempo il mittente ritrasmetterà il frame anche se ciò non è necessario. Dunque si stabilisce un limite al tempo di attesa 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 analizzeremo nel seguito appartengono alla classe dei protocolli sliding window, sono full-duplex, 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.



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 e il canale fisico è wire-like, cioé consegna i frame nell'ordine di partenza.

7.4.5 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, entrambi usano lo stesso codice.

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;

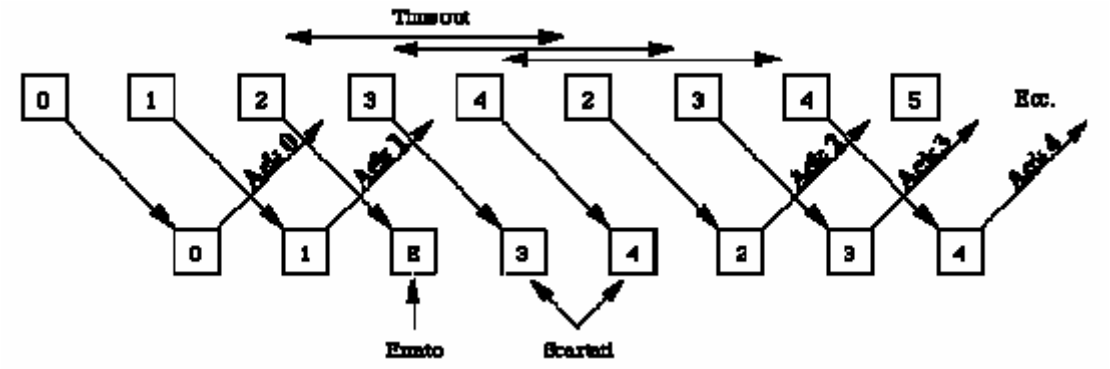
– *il destinatario invece*
  - quando arriva un frame corretto, senza errori, invia un ack col corrispondente numero di sequenza;
  - se il frame non è un duplicato lo passa al livello network e avanza la finestra.

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 caso peggiore è la ritrasmissione inutile di qualche frame, ma questo protocollo è sicuro.

7.4.6 Protocolli go-back-n e selective repeat

Se il tempo di andata e ritorno del segnale (round-trip time) è alto, come ad esempio nel caso dei canali satellitari nei quali è tipicamente pari a 500 + 500 msec, c'è un enorme inefficienza coi protocolli stop-and-wait, perché si rimane in attesa dell'ack.
Per migliorare le cose, si può consentire l'invio di un certo numero di frame anche senza aver ricevuto l'ack del primo. Questa tecnica va sotto il nome di *pipelining*.
Ciò però pone un serio problema, perché se un frame nel mezzo della sequenza si rovina molti altri frame vengono spediti prima che il mittente sappia che qualcosa è andato storto.
Il primo approccio al problema è quello del *protocollo go-back-n*:

- se arriva un frame danneggiato o con un numero di sequenza non progressivo, il destinatario ignora tale frame e tutti i successivi, non inviando i relativi ack. Ciò corrisponde ad una finestra di dimensione uno nel ricevitore, che quindi accetta i frame solo nell'ordine giusto;
- il mittente ad un certo punto va in time-out sul frame sbagliato, e poi su tutti quelli successivi (scartati dal destinatario), e quindi provvede a ritrasmettere la sequenza di frame che inizia con quello per il quale si è verificato il time-out.

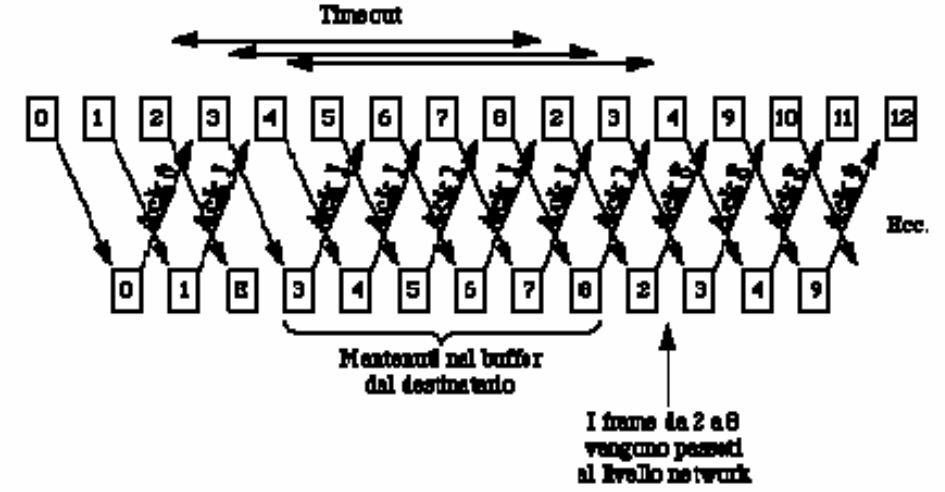


Funzionamento del protocollo go-back-n

Si noti che il mittente deve mantenere in un apposito buffer tutti i frame non confermati per poterli eventualmente ritrasmettere. Se il buffer si riempie, il mittente deve bloccare il livello network fino a che non si ricrea dello spazio. Inoltre, vi è spreco di banda se il tasso d'errore è alto e/o il time-out è lungo.

Il secondo approccio è più efficiente, ed è chiamato *selective repeat*:

- il destinatario mantiene nel suo buffer tutti i frame ricevuti successivamente ad un eventuale frame rovinato;
- non appena questo arriva nuovamente (senza errori), esso e tutti i successivi frame contigui che il destinatario ha mantenuto nel buffer vengono consegnati al livello network;
- per ogni frame arrivato correttamente, il destinatario invia un ack col numero più alto della sequenza completa arrivata fino a quel momento;
- quando si verifica un timeout, il mittente rispedisce il frame corrispondente.



Funzionamento del protocollo selective repeat

Alcune considerazioni a proposito del protocollo selective repeat:

- ☞ mittente e destinatario devono entrambi gestire un buffer per mantenervi i frame:
  - non confermati (mittente);
  - successivi ad un errore (destinatario);
- ☞ vi è un basso spreco di banda, che si si può ulteriormente diminuire mandando un NACK (Negative ACKnowledgement) quando:
  - arriva un frame danneggiato;

- arriva un frame diverso da quello atteso (ciò può indicare l'avvenuta perdita del frame precedente).

Infine, si noti che per entrambi i precedenti protocolli:

☞ è necessaria la gestione di timer multipli (uno per ogni frame inviato e non confermato);

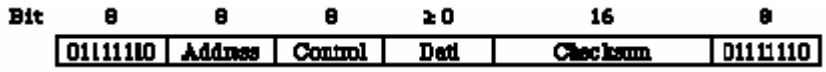
☞ il ricevente, per inviare gli ack, usa il piggybacking se possibile, altrimenti invia un apposito frame.

7.4.7 Protocolli standard di data link

I protocolli data link più diffusi oggi sono discendenti del protocollo SDLC (Synchronous Data Link Control), nato nell'ambito dell'architettura SNA. Nel seguito verranno illustrate brevemente le caratteristiche di tre diffusi protocolli: HDLC (standard ISO), SLIP (architettura TCP/IP) e PPP (suo successore).

– *HDLC (High Level Data Link Control)*

E' un protocollo bit oriented, e quindi usa la tecnica del bit stuffing. Il formato del frame HDLC è illustrato nella figura seguente.



Frame HDLC

I campi del frame hanno le seguenti funzioni:

Address	utilizzato nelle linee multipunto, dove identifica i diversi terminali (il protocollo offre funzioni per il dialogo fra un concentratore e diversi terminali)
Control	contiene numeri di sequenza, ack, ecc.
Dati	contiene i dati da trasportare
Checksum	è calcolata con CRC-CCITT

Le caratteristiche salienti sono le seguenti:

1) usa una finestra scorrevole con numeri di sequenza a 3 bit, contenuti dentro un campo Seq situato all'interno del campo Control;
2) utilizza il campo Next, anch'esso contenuto in Control, per il piggybacking degli ack;
3) ha tre tipi di frame (identificati dai primi due bit di Control):
   a) Information, per la trasmissione dati;

b) Supervisory, per comandare diverse modalità di ritrasmissione;

c) Unnumbered (non ha il numero di sequenza), per finalità di controllo o per trasportare il traffico di connessioni non affidabili.

– *SLIP (Serial Line IP)*

E' nato nel 1984 ed è il più vecchio protocollo di livello data link dell'Internet Protocol Suite.
Molto semplice, nacque per collegare via modem macchine Sun ad Internet. Spedisce sulla linea pacchetti IP terminati col byte `0xC0`. Usa il character stuffing.
Ha diverse limitazioni, poiché non prevde controllo degli errori; supporta solo IP, e per di più solo indirizzi statici; non è uno standard ufficiale di Internet.

– *PPP (Point to Point Protocol)*

Per migliorare le cose, IETF ha prodotto uno standard ufficiale, il Point to Point Protocol (RFC 1661, 1662 e 1663). Esso è adatto sia a connessioni telefoniche che a linee router-router.
Esso fornisce le seguenti funzionalità:

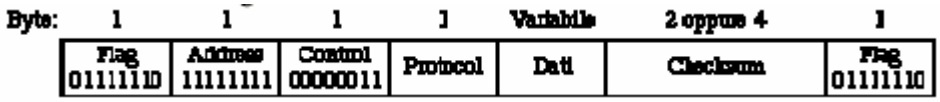
- framing;
- rilevamento degli errori;
- un protocollo di controllo per attivare, testare e diasattivare la linea (LCP, Link Control Protocol, RFC 1570);
- supporto di molteplici protocolli di livello network;
- una famiglia di protocolli per negoziare opzioni di livello network (NCP, Network Control Protocol): per ogni livello network supportato c'è un differente NCP e ad esempio, nel caso di IP, NCP viene usato per negoziare un indirizzo IP dinamico;
- il traffico derivante (nelle fasi iniziali e finali della connessione) dall'uso dei protocolli LCP e NCP viene trasportato dentro i frame PPP.

Il protocollo è modellato su HDLC, ma con alcune differenze:

✓ è character-oriented anziché bit-oriented, e utilizza il character stuffing (quindi i frame sono costituiti da un numero intero di byte);

✓ è previsto un campo apposito per il supporto multiprotocollo offerto al livello network.

Il formato del frame PPP è il seguente.



Frame PPP

I campi del frame hanno le seguenti funzioni:

Flag	come in HDLC
Address	sempre 11111111: di fatto non ci sono indirizzi, in quanto non c'è più l'idea di gestire linee multipunto
Control	il default (00000011) indica un unnumbered frame, quindi relativo ad un servizio non affidabile
Protocol	indica il protocollo relativo al pacchetto che si trova nel payload (LCP, NCP, IP, IPX, Appletalk, ecc.)
Payload	è di lunghezza variabile e negoziabile, il default è 1500 byte
Checksum	normalmente è di due byte (quattro sono negoziabili)

7.5 Il sottolivello di accesso al mezzo (MAC)

7.6 Reti a bus

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 realizzare, 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. Nell'affrontare questo approccio sono necessarie alcune definizioni.

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 p * T, 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):
   a) tutte le stazioni possono rilevare la collisione;
   b) i frame devono essere ritrasmessi;
4) non ci sono altri tipi di errori;
5) *tempo*: può essere gestito in due modi:
   a) *continuous time*: la trasmissione di un frame può iniziare in un qualunque istante;
   b) *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;
6) ascolto del canale: ci sono due possibilità,
   a) *carrier sense* (tipico delle LAN): le stazioni, prima di trasmettere, ascoltano il canale; se è occupato non cercano di trasmettere;
   b) *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.

Nel seguito verranno analizzati alcuni protocolli che si basano su alcune caratteristiche precedentemente descritte.

7.6.1 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.

Per determinare l'efficienza 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:

G*P(0)

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

Per calcolare il throughput effettivo, e quindi l'efficienza, ottenibile col protocollo Pure Aloha, si devono fare due considerazioni.

La prima è che la probabilità di generare k frame durante un intervallo di tempo pari ad un frame time è data, per la distribuzione di Poisson sopra definita (avente, si ricordi, valor medio pari a G frame per frame time) dalla relazione:

$$P(K) = G^K \times e^{-G} / K!$$

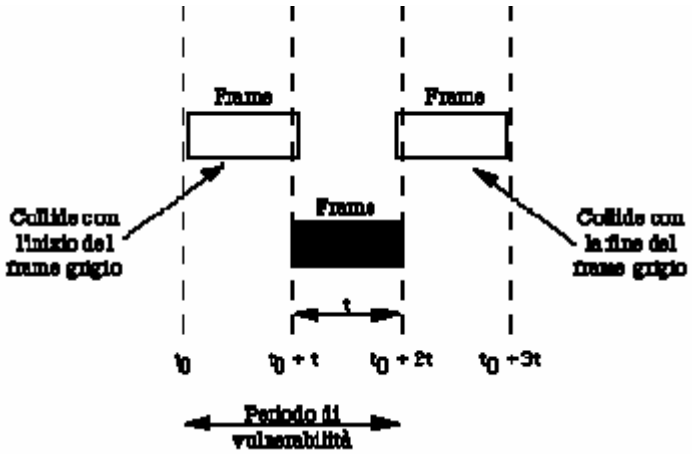
Dunque, la probabilità che si generino zero frame in un intervallo di tempo pari ad un frame time è pari a $P(0) = e^{-G}$.

La seconda considerazione è che il periodo di vulnerabilità di un frame, cioé l'intervallo di tempo nel quale esso è a rischio di collisioni, è lungo 2 volte il frame time.

In tale periodo vengono generati mediamente 2G frame. Di conseguenza, la probabilità che non si generino nuovi frame per tutto il periodo di vulnerabilità di un frame è: $P(0) = e^{-2G}$

Utilizzando tale probabilità nella relazione vista sopra per il throughput, otteniamo la stima del throughput ragggiungibile col protocollo Pure Aloha, che è $G \times e^{-2G}$

Il massimo throughput è 0,184, cioé meno del 20% (due frame su 10 slot) in corrispondenza di un carico G pari a 0,5 frame per frame time.



Il periodo di vulnerabilità

Un modo per aumentare l'efficienza di Aloha consiste nel dividere il tempo in intervalli discreti, ciascuno corrispondente ad un frame time. Ovviamente gli utenti devono essere d'accordo nel confine fra gli intervalli, e ciò può essere fatto facendo emettere da una attrezzatura speciale un breve segnale all'inizio di ogni intervallo.

Le stazioni non possono iniziare a trasmettere quando vogliono, ma solo all'inizio dell'intervallo. Questo protocollo, che prende il nome di Slotted Aloha, dimezza il periodo di vulnerabilità che diviene uguale ad un solo frame time.

In tale periodo vengono generati mediamente G frame, per cui la probabilità che non si generino nuovi frame per tutto il periodo di vulnerabilità di un frame è: $P(0) = e^{-G}$

Utilizzando tale probabilità nella relazione vista precedentemente per il throughput, otteniamo la stima del throughput ragggiungibile col protocollo Slotted Aloha, che è: Ge^{-G}

Il massimo throughput è 0,368, in corrispondenza di un carico G pari a 1 frame per frame time.

7.6.2 Protocolli CSMA (Carrier Sense Multiple Access)

Nelle LAN le stazioni possono ascoltare il canale e regolarsi di conseguenza. 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.

Ci sono, ovviamente, dei 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.

7.6.3 Protocolli CSMA/CD (Carrier Sense Multiple Access with Collision Detection)

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 $t_0 + 2T$.

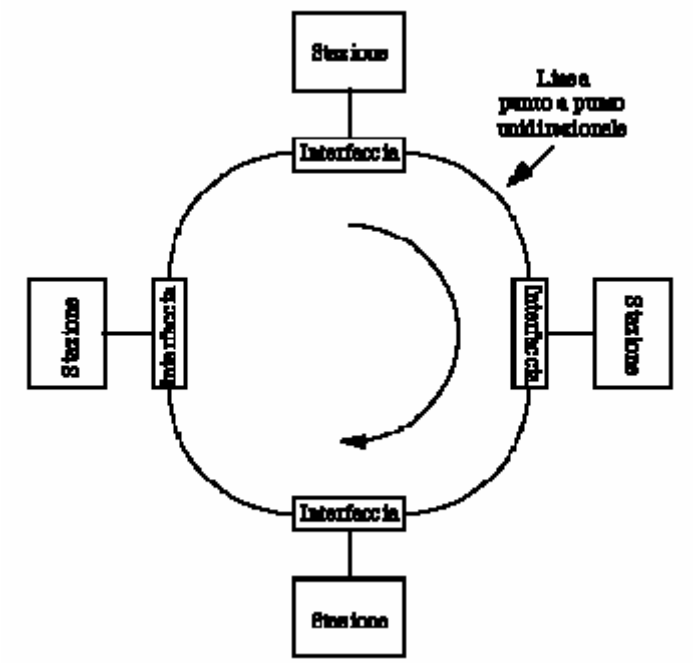
Il modello concettuale che si utilizza è il seguente:

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.

7.7 Reti ad anello

Una rete ad anello consiste di una collezione di interfacce di rete, collegate a coppie da linee punto a punto:

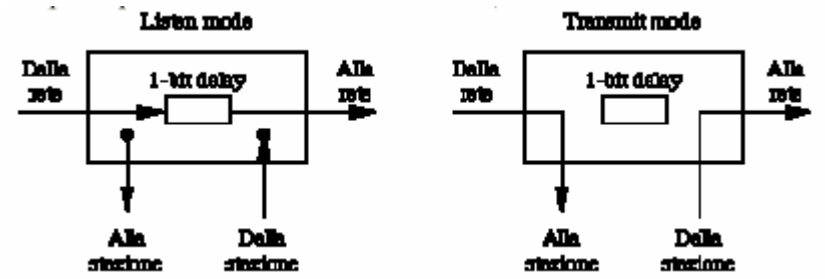


Struttura di una rete ad anello

Le reti ad anello hanno diverse caratteristiche interessanti: non sono reti basate su un mezzo trasmissivo broadcast; non c'è una significativa componente analogica per la rilevazione delle collisioni (che non possono verificarsi); l'anello è intrinsecamente equo.

Ogni bit che arriva all'interfaccia è copiato in un buffer interno, poi rigenerato e ritrasmesso sul ring. Può essere modificato prima di essere ritrasmesso.

L'interfaccia di rete può operare in due diverse modalità, listen mode e transmit mode:



Modalità di funzionamento dell'interfaccia di rete

In listen mode i bit in ingresso vengono copiati nel buffer interno (dove possono essere anche modificati) e quindi ritrasmessi con un ritardo di un bit (1-bit delay).

In transmit mode l'anello è aperto, e i bit in arrivo vengono rimossi; nuovi bit vengono trasmessi sull'anello.

Una speciale configurazione binaria, detta token (gettone) circola in continuazione se nessuno vuole trasmettere.

Quando una stazione vuole trasmettere, deve:

- aspettare che arrivi il token (in listen mode);
- rimuoverlo dal ring (in listen mode);
- trasmettere i dati (in transmit mode);
- rigenerare il token (in transmit mode);
- rimettersi in listen mode.

Poiché c'è un solo token, questo meccanismo risolve senza conflitti il problema dell'accesso al mezzo.

Alcune considerazioni sono degne di nota:

- il token deve essere contenuto per intero sull'anello, il che richiede una certa attenzione;
- un frame, invece, non è necessario che sia per intero sull'anello (che in trasmissione è aperto), quindi non ci sono limiti alla dimensione dei frame;
- in genere esiste un tempo massimo entro il quale, una volta preso il token, si deve completare la trasmissione; ciò permette di ottenere una schedulazione round-robin delle trasmissioni;
- quando tutte le stazioni hanno qualcosa da trasmettere, l'efficienza si avvicina al 100%;
- viceversa, quando non c'è traffico, una stazione deve attendere un pò più che in CSMA/CD per trasmettere (mediamente dovrà attendere un tempo pari a quello di attraversamento di mezzo anello, per ricevere il token).

La velocità di propagazione del segnale nel rame è circa 200 metri per microsecondo. Con un data rate (ad esempio) di 1 Mbps, si genera un bit al microsecondo. Dunque, un bit è lungo in tal caso circa 200 metri, per cui per contenere 10 bit un anello dovrebbe essere lungo almeno 2 km.

In realtà sul ring trovano posto:

- x bit sull'anello, in funzione della sua lunghezza totale;
- y bit nei buffer delle interfacce delle y stazioni presenti (1 bit delay).

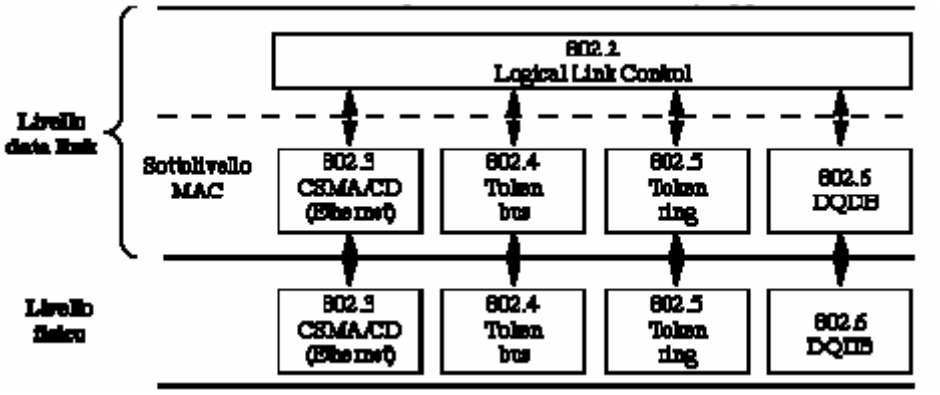
In definitiva, è necessario che x + y sia maggiore del numero di bit del token. Ciò significa che, a seconda delle caratteristiche dimensionali della rete in questione, può essere necessario ricavare un ritardo addizionale, sotto forma di buffer aggiuntivi, in una stazione (che ha un ruolo particolare, quello di monitor dell'anello).

7.8 Lo standard IEEE 802 per le LAN

IEEE 802 nasce dalla standardizzazione di alcuni protocolli, originariamente proprietari, ma successivamente, con lievi modifiche, adottati universalmente per le reti LAN.

E' costituito da diverse sotto parti. Specifiche generali del progetto (802.1); Logical Link Control, LLC (802.2); CSMA/CD (802.3); token bus (802.4, destinato a LAN per automazione industriale); token ring (802.5); DQDB (802.6, destinato alle MAN).

I vari standard differiscono a livello fisico e nel sottolivello MAC, ma sono compatibili a livello data link. Ciò è ottenuto separando dal resto, attraverso l'apposito standard LLC, la parte superiore del livello data link, che viene usata da tutti i protocolli standard del gruppo.



Lo standard IEEE 802

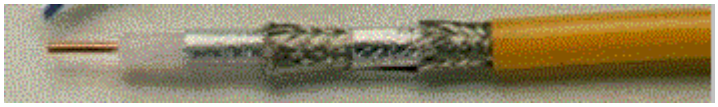
7.8.1 IEEE 802.3

E' lo standard per un protocollo CSMA/CD, di tipo 1-persistent, funzionante a 10Mbps. 802.3 è l'evoluzione dello standard Ethernet, proposto da Xerox, DEC e INTEL sulla base dell'esperienza maturata con Aloha prima e nei laboratori Xerox PARC poi.

802.3 e Ethernet hanno alcune differenze, ma sono largamente compatibili.

Sono previsti vari cablaggi descritti brevemente più avanti.

- *10Base5 (Thick ethernet)*



un cavo coassiale RG-8

è il primo storicamente; consiste di un cavo coassiale spesso (lo standard suggerisce il colore giallo per la guaina esterna).

Caratteristiche

- Cavo coassiale RG-8
  - 10 Mbps;
  - Baseband signaling;
  - 500 metri di lunghezza massima per segmento.
  - Possono essere installate 100 macchine su un segmento.
  - Richiede un terminatore a ciascun capo con impedenza di 50 Ohm
- Ogni stazione contiene un'interfaccia di rete (detta anche scheda ethernet) che incapsula i dati del livello superiore, gestisce il protocollo MAC, codifica i dati da trasmettere e in ricezione decapsula i dati e li consegna al livello superiore (o lo informa dell'errore).
- All'interfaccia di rete viene collegata una estremità di un corto cavo (pochi metri), detto transceiver drop cable, all'altra estremità del quale è connesso un transceiver che si aggancia, con un dispositivo detto vampiro, al cavo thick (che non viene interrotto).

Un transceiver per 10 base 5

- Il transceiver contiene la circuiteria analogica per l'ascolto del canale e la rilevazione delle collisioni. Quando c'è una collisione, il transceiver informa l'interfaccia ed invia sulla rete uno speciale segnale di 32 bit (jamming sequence) per avvisare le altre stazioni, che così scartano quanto già ricevuto.

Esempio

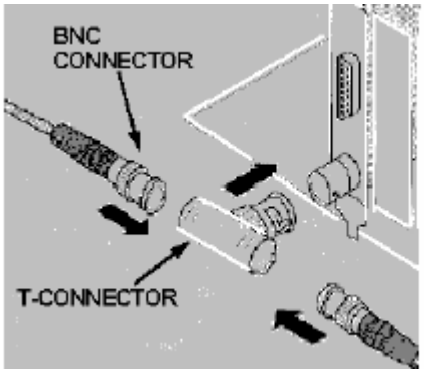
– *10Base2 (Thin ethernet)*

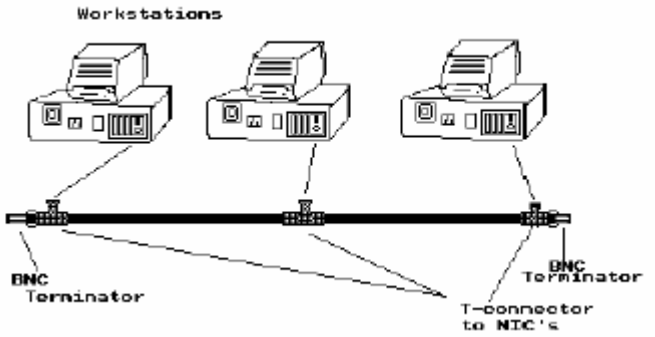
un cavo coassiale RG-58 con connettore BNC

E’ un cavo coassiale più sottile, e si piega più facilmente.

Caratteristiche

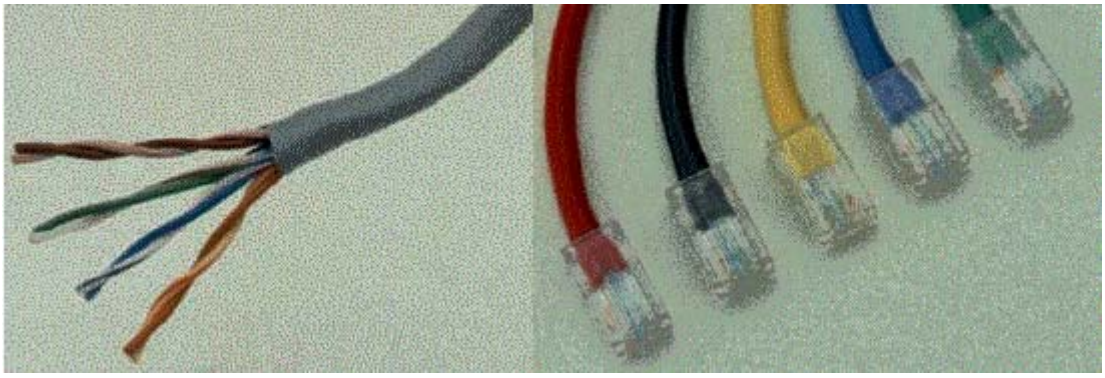
- Cavo coassiale RG-58
  - Connessione mediante BNC (British Naval Connector)
  - Richiede un terminatore a ciascun capo con impedenza di 50 Ohm
  - 10 Mbps;
  - Baseband signaling;
  - 200 (185 meglio) metri di lunghezza massima per un singolo segmento.
  - Possono essere installate 30 macchine su un segmento.
- Di norma l'interfaccia di rete contiene anche il transceiver.
- L'allaccio di una stazione alla rete avviene con una giunzione a T, alla quale sono collegati il cavo che porta alla stazione e due cavi thin che costituiscono una porzione del segmento. Le varie stazioni sono collegate in cascata (daisy-chain) sul segmento.





Modalità di connessione in 10 base 2

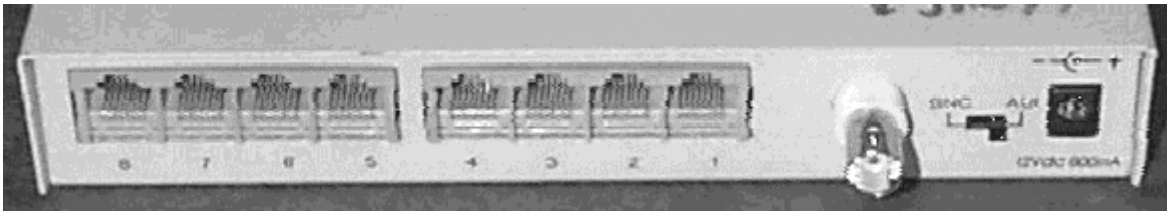
– *10BaseT (Doppino telefonico)*



Doppino telefonico a 4 coppie e connettori RJ-45

Caratteristiche

- Lo standard 10BaseT (twisted) prevede il collegamento fra una sola coppia di stazioni.
- Il cavo è normalmente un UTP a 4 coppie AWG 22, 24 o 26 (solo una coppia viene utilizzata)
- Connettori RJ45
- La lunghezza massima è 100 metri (150 se il doppino è di classe 5).
- Per connettere più di due stazioni serve un *concentratore multiporta* (HUB).



Un concentratore (HUB) a 8 porte

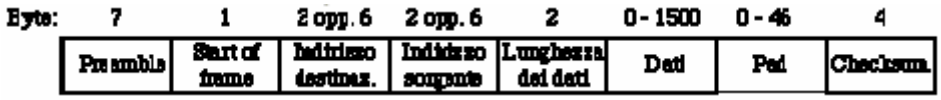
Un concentratore (detto anche ripetitore) è un dispositivo che opera a livello uno (fisico): riceve il segnale da un segmento, lo amplifica e lo ritrasmette su tutti gli altri segmenti. Gli hub possono essere usati anche per aumentare la lunghezza complessiva della rete.

Comunque, sono in vigore delle regole generali stabilite dallo standard:

- la lunghezza massima dell'intera rete, fra qualunque coppia di stazioni, non deve superare i 2,5 km;
- fra qualunque coppia di stazioni non devono trovarsi più di quattro ripetitori;
- possono esservi al massimo 1024 stazioni sulla rete.

La codifica dei dati utilizzata in IEEE 802.3 è la già citata Codifica Manchester che consente una facilità di sincronizzazione fra mittente e destinatario e un bilanciamento nell'energia per cui si ha uguale energia per lo zero e per l'uno, e quindi la trasmissione di dati, anche se genera diverse quantità di zeri e uni, non produce componenti in corrente continua, molto dannose perché ostacolano la trasmissione dei segnali.

La struttura di un frame 802.3 è la seguente:



Frame 802.3

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
Indirizzo sorgente/destinazione	gli indirizzi usati sono sempre a 6 byte, e sono assegnati univocamente, a livello mondiale, ai produttori di schede che li cablano 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 1)
Lunghezza dei dati	indica quanti byte ci sono nel campo dati (da 0 a 1500)
Dati	contiene il payload
Pad	Se il frame è più corto di 64 byte, con questo campo lo si porta alla lunghezza di 64 byte. (Per consentire comunque la rilevazione di una collisione)
Checksum	è un codice CRC

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.

La necessità del campo Pad e il limite minimo di 64 byte per la lunghezza di un frame ha come conseguenza che affinché 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.
E' 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.
Pertanto, la trasmissione di un frame non deve durare meno di 57,6 microsecondi che è 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.

Modo operativo

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 certa quantità di tempo 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, al contrario 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.

La pratica ha mostrato che 802.3: può sopportare un carico medio del 30% (3 Mbps) con picchi del 60% (6 Mbps) e in presenza di un carico medio il 2-3% dei pacchetti ha una collisione e solo qualche pacchetto su 10.000 ha più di una collisione.

- *Lo standard Fast Ethernet*

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

100BaseT4 (Doppino classe 3)

- si usano tutti e quattro i 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.

100BaseT (Doppino classe 5)

- 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.

100BaseFX (Fibra ottica)

– velocità di segnalazione 125 Mhz;
– codifica 4B5B;
– obbligatorio switched hub;
– lunghezza rami fino a 2 km.

IEEE 802.5 (Token Ring)

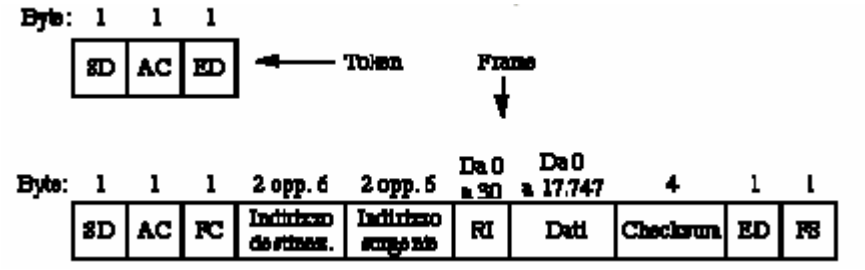
– prevede velocità di 4 Mbps e 16 Mbps (802.5, ma non token ring nella sua versione originale IBM) prevede anche la velocità di 1 Mbps.
– Utilizza tipicamente un cablaggio basato su doppino telefonico:
– schermato (STP);
– non schermato (UTP):
– categoria 3, 4 o 5 per 4 Mbps;
– categoria 4 o 5 per 16 Mbps.

Normalmente il cablaggio è fatto utilizzando un wire center, che ha la possibilità di isolare parti dell'anello guaste: se manca corrente su un lobo il corrispondente relais si chiude automaticamente. I lobi hanno una lunghezza massima variabile, a seconda del cablaggio utilizzato:

✓ UTP cat. 4: 150 metri;
✓ UTP cat. 5: 195 metri;
✓ STP: 340 metri.

Le stazioni possono essere al massimo 260.

La codifica utilizzata è Differential Manchester Encoding



Token e frame 802.5

I campi del frame hanno le seguenti funzioni:

SD, ED	Starting e ending delimiter: contengono all'interno due coppie di bit codificati con i valori high-high e low-low, in violazione della codifica Manchester. Si usano high-high e low-low accoppiati per non introdurre uno sbilanciamento del codice.
AC	Access control, serve per il controllo dell'accesso. E' costituito di 8 bit: PPPTMRRR i tre bit P indicano la priorità attuale; il bit M serve per il controllo di frame orfani: il monitor lo setta ad 1 al passaggio del frame, e se lo ritrova ad uno al passaggio successivo il frame è orfano e viene tolto dall'anello; il bit T, detto token bit, identifica un token (se vale 0) o un frame (se vale 1); i tre bit R indicano la priorità richiesta.
FC	Frame control, distingue frame contenenti dati da frame con funzioni di controllo.
Indirizzi	Analoghi ad 802.3.
RI	Routing information, contiene (se c'è) le informazioni necessarie al source routing (vedremo più avanti).
Dati	contiene il payload
Checksum	è un codice CRC
FS	Frame status, serve per sapere cosa è successo del frame. Contiene, fra l'altro, due bit, A e C, gestiti come segue: bit A: viene messo ad 1 (dal destinatario) quando il frame gli arriva; bit C: viene messo ad 1 (dal destinatario) quando il frame gli arriva ed il destinatario lo copia al suo interno.

Modo operativo

Quando il token circola e una stazione vuole trasmettere, essa, che è in listen mode, opera come segue:

- aspetta che arrivi il token;
- quando il token arriva:
  - lascia passare SD;
  - lascia passare i bit PPP di AC;

- quando ha nel buffer il token bit T:
- lo cambia in uno, trasformando il token in un frame e invia il bit T modificato sul ring;

- si mette immediatamente in transmit mode;
- invia il resto del frame;
- quando il frame è trasmesso:
  - se non ha esaurito il THT (Token holding time) può trasmettere un altro frame, altrimenti rigenera un nuovo token e lo trasmette;
- appena trasmesso l'ultimo bit del token si rimette immediatamente in listen mode.

Ogni ring ha una stazione con un ruolo speciale, il monitor (ogni stazione è in grado di diventare il monitor). Il monitor viene designato all'avvio dell'anello. I suoi compiti principali sono: rigenerare il token se esso si perde; ripulire il ring dai resti di frame danneggiati; ripulire il ring dai frame orfani.

– *IEEE802.2*

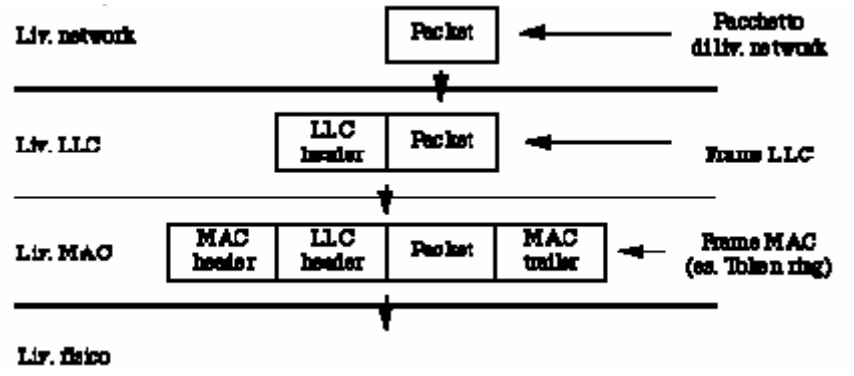
Lo standard, noto come Logical Link Control (LLC), definisce la parte superiore del livello data link in modo indipendente dai vari sottolivelli MAC.

Ha le funzioni principali di fornire al livello superiore un'interfaccia unica, nascondendo le differenze fra i vari sottolivelli MAC e, se richiesto dal livello superiore, un servizio più sofisticato di quello offerto dai vari sottolivelli MAC. Esso infatti può fornire: servizi datagram, ma anche servizi datagram confermati e servizi affidabili orientati alla connessione.

Il frame LLC è modellato ispirandosi a HDLC, con indirizzi di mittente e destinatario, numeri di sequenze, numeri di ack (questi ultimi due omessi per i servizi datagram), ecc.

Gli indirizzi LLC sono lunghi un byte e servono sostanzialmente ad indicare quale protocollo di livello superiore deve ricevere il pacchetto di livello tre; in questo modo LLC offre un supporto multiprotocollo al livello superiore.

Il frame LLC viene imbustato, in trasmissione, in un frame dell'opportuno sottolivello MAC. Il processo inverso ha luogo in ricezione.

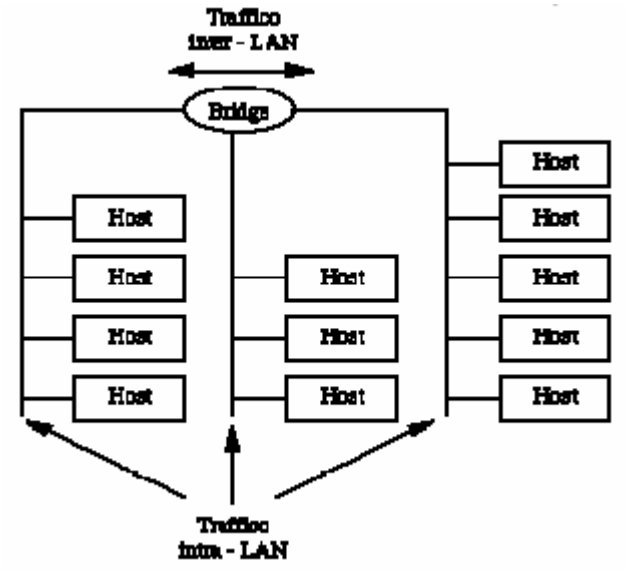


LLC e MAC

7.9 Apparati attivi di interconnessione: bridge

Un bridge è un apparato utilizzabile quando due LAN di tipo differente (ad esempio una Ethernet ed una Token ring), che non si possono semplicemente collegare l'una con l'altra, contengono host che vogliono dialogare fra loro. Esso è però utilizzabile anche per LAN omogenee quando si voglia per esempio una LAN la cui lunghezza superi i limiti massimi consentiti (ad esempio, 2,5 km per Ethernet) ovvero si desideri, nel caso di una LAN contenente molti host, suddividerla in molteplici LAN interconnesse. Questo per segmentare il traffico generato nelle sue parti, in modo da avere un traffico totale molto superiore a quello possibile su una singola LAN.

Due o più LAN possono quindi essere interconnesse con bridge, che operano a livello data link. Ciò significa che la loro operatività è basata esclusivamente sulle informazioni contenute nelle buste di livello due, mentre non vengono prese in considerazione quelle di livello tre. Questa caratteristica fondamentale che li differenzia dai router, che invece agiscono a livello tre.



Interconnessione di LAN tramite bridge

I bridge si occupano di instradare il traffico da una LAN all'altra. E' importante sottolineare che, anche se l'instradamento di per se è una funzione tipica del livello tre, qui avviene sulla base dei soli indirizzi di livello due.

Il funzionamento di un bridge, che ha tante interfacce di rete quante sono le LAN alle quali è fisicamente collegato, è di seguito descritto.

- Quando una delle interfacce di rete del bridge riceve un frame MAC, lo passa al relativo software di livello MAC che toglie la busta MAC;
- il resto viene passato dal livello MAC al software di livello LLC del bridge, nel quale, sulla base dell'indirizzo di destinazione, si decide a quale LAN inviarlo:
- se la destinazione si trova sulla LAN di provenienza il frame viene scartato;
- altrimenti, il frame LLC viene passato al livello MAC competente per la LAN di destinazione, che lo imbusta in un frame MAC e provvede ad inviarlo su tale LAN, secondo le regole di quest'ultima.

Si noti che un bridge è ben diverso da un hub, che copia pedissequamente tutto ciò che riceve da una linea su tutte le altre. Il bridge infatti acquisisce un frame, lo analizza, lo ricostruisce e lo instrada, quindi può anche essere configurato in modo da filtrare alcuni tipi di traffico. Ciò tipicamente avviene in funzione dell'indirizzo LLC, che identifica il protocollo di livello superiore, o sulla base dell'indirizzo MAC del mittente o del destinatario.

I bridge progettati per interconnettere LAN di tipo diverso devono risolvere vari problemi legati alle diverse regole in vigore su tali LAN, tra cui:

- ✓ formati dei frame differenti;
- ✓ data rate differenti;
- ✓ massima lunghezza di frame differente, poiché è fuori questione spezzare un frame in questo livello, dato che tutti i protocolli si aspettano che il frame o arrivi per intero o non arrivi affatto;
- ✓ funzioni previste da un tipo di LAN ma non dall'altra: ad esempio, il concetto di priorità ed i bit A e C presenti in 802.5 non hanno un equivalente in 802.3.

Lo standard IEEE 802 propone due tipi di bridge:

- *transparent bridge* (promossi dai comitati 802.3 e 802.4)
- *source-routing bridge* (scelti dal comitato 802.5)

Il transparent bridge (IEEE 802.1 part D) può essere installato e diventare operativo in modo totalmente trasparente, senza richiedere niente altro che la connessione fisica e l'accensione.
In pratica, dal momento in cui il bridge viene attivato, esamina tutti i frame che arrivano dalle varie LAN e sulla base di questi costruisce progressivamente le sue tabelle di instradamento. Infatti, ogni frame ricevuto consente al bridge di sapere su quale LAN si trova la stazione che lo ha inviato.
Ogni frame che arriva al bridge viene ritrasmesso:

- se il bridge ha nelle sue tabelle di instradamento l'indirizzo del destinatario, invia il frame sulla corrispondente LAN;
- altrimenti il frame viene inviato a tutte le LAN tranne quella di provenienza, con una tecnica detta flooding;
- man mano che il bridge aumenta la sua conoscenza degli indirizzi delle varie macchine, la ritrasmissione diventa sempre più selettiva (e quindi più efficiente).

Le tabelle vengono aggiornate ogni qualche minuto, rimuovendo gli indirizzi che non si sono fatti vivi nell'ultimo periodo (così, se una macchina si sposta, entro pochi minuti viene di nuovo indirizzata correttamente) Questa tecnica si chiama backward learning.
Se ci sono maglie nella topologia di connessione delle LAN, i bridge si costruiscono di essa uno spanning tree, che poi utilizzano per l'instradamento, al fine di evitare la generazione di un infinito numero di duplicati durante il flooding.

Il source-routing bridge (nato per le reti 802.5) è progettato invece per ottenere l'instradamento più efficiente possibile, anche a scapito della trasparenza.

L'idea di base è che il mittente indichi esplicitamente il cammino (espresso come sequenza di bridge e reti) che il frame deve percorrere. L'amministratore di sistema deve assegnare numeri di identificazione distinti ad ogni rete e ad ogni bridge, operazione che deve essere fatta manualmente.

Tali informazioni sono incluse in un apposito campo RI (Routing Information) del frame 802.5, e la loro eventuale presenza è indicata dal valore 1 del bit più significativo dell'indirizzo sorgente (che, essendo sempre relativo a un indirizzo singolo e mai di gruppo o broadcast, originariamente è sempre zero). Il bridge esamina solo i frame che hanno tale bit a uno.

E' ovvio che ogni host deve avere il quadro della topologia delle connessioni, memorizzato in un'apposita struttura dati. Per costruirla e mantenerla, il meccanismo usato è il seguente:

- quando un host deve spedire un frame ma non conosce il cammino da seguire per raggiungere la destinazione, invia un discovery frame, chiedendo tale informazione;
- il discovery frame viene inviato in flooding da ogni bridge a tutti gli altri, e quindi raggiunge tutti gli host. In questa fase, ogni bridge scrive nel discovery frame il suo ID, che si aggiunge a quello dei bridge precedentemente incontrati. Quando un discovery frame arriva alla destinazione, contiene tutto il cammino percorso;
- quando l'host di destinazione riceve un discovery frame, lo invia indietro al mittente;
- il mittente, sulla base del primo discovery frame che ritorna (considerando il relativo cammino quello più conveniente) aggiorna le sue tabelle e può mandare il frame che voleva spedire originariamente.

Un vantaggio di questo schema di funzionamento è che si trova sempre il cammino ottimo; uno svantaggio è l'esplosione del numero di discovery frame.

Dopo un periodo in cui entrambi gli standard sopra descritti erano abbastanza diffusi, oggi praticamente tutti i bridge costruiti sono di tipo transparent, ed al più offrono la funzionalità source-routing come un'opzione supplementare.