Appunti del corso di
Sistemi di elaborazione: Reti I

PROF. G. BONGIOVANNI

Premessa

Questi appunti sono basati sul libro "Computer Networks" di A. Tanenbaum, terza edizione, ed. Prentice-Hall, adottato quale libro di testo del corso.

Essi rispecchiano piuttosto fedelmente il livello di dettaglio che viene seguito durante le lezioni, e costituiscono un ausilio didattico allo studio.

Tuttavia, è importante chiarire che gli appunti non vanno intesi come sostitutivi né del libro di testo né della frequenza alle lezioni, che rimangono fattori importanti per una buona preparazione dell'esame.

La realizzazione di questi appunti è stata resa possibile dalla collaborazione dello studente Federico Neri, che ha avuto la pazienza di convertire in forma elettronica il contenuto testuale dei manoscritti da me preparati per il corso. La realizzazione delle figure, la formattazione e la rifinitura del testo sono opera mia.

1) Introduzione

Gli ultimi tre secoli sono stati dominati ciascuno da una diversa tecnologia che lo ha caratterizzato ed ha avuto profonde influenze sulla vita dell'uomo:

- 18º secolo: sistemi meccanici, rivoluzione industriale;
- 19º secolo: motori a vapore;
- 20º secolo: tecnologie dell'informazione:
  - raccolta e memorizzazione;
  - elaborazione;
  - distribuzione.

Nel nostro secolo si sono via via diffusi:

- il sistema telefonico, a livello mondiale;
- la radio e la televisione;
- i computer;
- i satelliti per telecomunicazioni.

Queste tecnologie stanno rapidamente convergendo. In particolare, la combinazione di elaboratori e sistemi di telecomunicazione ha avuto una profonda influenza sull'organizzazione dei sistemi di calcolo.

Si è passati dal vecchio modello ***mainframe - terminali*** , in cui la potenza di calcolo è concentrata in un unico grande elaboratore a cui si accede per mezzo di un certo numero di terminali, a quello attuale in cui vi è un grande numero di elaboratori ***autonomi***, ***interconnessi*** fra loro:

- autonomi: significa che non deve esserci fra loro una relazione tipo master / slave (ad es., l'uno non può forzare lo spegnimento dell'altro);
- interconnessi: significa che devono essere capaci di scambiare informazioni (sfruttando un opportuno mezzo fisico).

Un sistema di calcolo siffatto è detto ***rete di elaboratori*** (***computer network***).

Si noti che rete di elaboratori non è sinonimo di ***sistema distribuito*****.** Infatti:

- in un sistema distribuito l'esistenza di più elaboratori è invisibile all'utente, che ha l'impressione di avere a che fare con un unico sistema di calcolo;
- in una rete di elaboratori, l'utente è conscio dell'esistenza di molteplici elaboratori, che devono essere esplicitamente riferiti.

In effetti, si può dire che:

Rete di Elaboratori + Sistema software di gestione = Sistema distribuito

dove il sistema software di gestione altro non è che un particolare tipo di sistema operativo, ossia un ***sistema operativo distribuito***.

1.1) Usi delle reti di elaboratori

Moltissimi sono gli usi delle reti di elaboratori, sia per le organizzazioni che per i singoli individui.

- Per le organizzazioni:
  - condivisione risorse: si possono rendere disponibili a chiunque programmi e informazioni anche distanti migliaia di km;
  - affidabilità: si ottiene mettendo in rete sorgenti alternative delle risorse (ad es. duplicando le applicazioni e i dati su più computer). E' importante in sistemi che devono funzionare a tutti i costi (traffico aereo, centrali nucleari, sistemi militari, ecc.);
  - diminuzione dei costi: una rete di personal computer costa molto meno di un mainframe. A volte alcuni elaboratori sono più potenti ed offrono agli altri dei servizi (modello ***client-server***, vedi figura sottostante);
  - scalabilità: si possono aumentare le prestazioni del sistema aumentando il numero di elaboratori (entro certi limiti);
  - comunicazione fra persone: è possibile inviarsi messaggi, scambiarsi file, ecc.

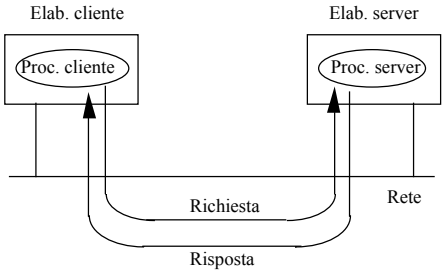


Figura 1-1: Il modello client-server

- Per i singoli individui: (di solito da casa propria tramite "fornitori di accesso"):
  - accesso ad informazioni remote, ad es.:
    - accesso a servizi bancari;
    - acquisti da casa;
    - navigazione sul World Wide Web;
  - comunicazioni fra persone:
    - posta elettronica;
    - videoconferenza;
    - gruppi di discussione;

- divertimento:
  - video on demand (selezione e ricezione via rete di un qualunque spettacolo tratto da un catalogo);
  - giochi interattivi (contro macchine o avversari umani).

1.2) Aspetti hardware delle reti

Due parametri sono utili per definire le caratteristiche di una rete, anche se non esiste una tassonomia universalmente accettata:
- tecnologia trasmissiva;
- scala dimensionale.

1.2.1) Tecnologia trasmissiva

Ci sono due tipologie per quanto riguarda la tecnologia trasmissiva:
- ***reti broadcast***;
- ***reti punto a punto***.

Le ***reti broadcast*** sono dotate di un unico "canale" di comunicazione che è condiviso da tutti gli elaboratori. Brevi messaggi (spesso chiamati ***pacchetti***) inviati da un elaboratore sono ricevuti da tutti gli altri elaboratori. Un indirizzo all'interno del pacchetto specifica il destinatario.

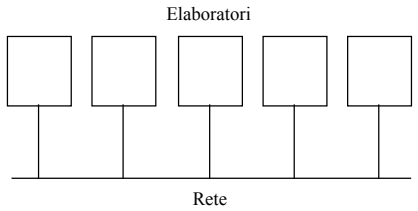


Figura 1-2: una rete broadcast

Quando un elaboratore riceve un pacchetto, esamina l'indirizzo di destinazione; se questo coincide col proprio indirizzo il pacchetto viene elaborato, altrimenti viene ignorato.

Le reti broadcast, in genere, consentono anche di inviare un pacchetto a tutti gli altri elaboratori, usando un opportuno indirizzo (***broadcasting***). In tal caso tutti prendono in considerazione il pacchetto.

Un' altra possibilità è inviare il pacchetto ad un sottoinsieme degli elaboratori (***multicasting***). In tal caso solo gli elaboratori di tale sottoinsieme lo prendono in

considerazione, gli altri lo ignorano. In questo caso, un bit dell'indirizzo indica che si tratta di una trasmissione in multicasting. I rimanenti (n-1) bit dell' indirizzo rappresentano l'indirizzo del gruppo destinatario.

Le ***reti punto a punto*** consistono invece di un insieme di connessioni fra coppie di elaboratori.

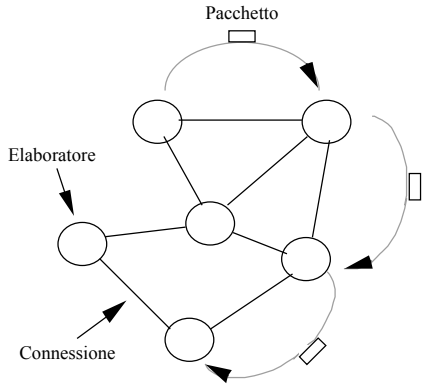


Figura 1-3: una rete punto a punto

Per arrivare dalla sorgente alla destinazione, un pacchetto può dover attraversare uno o più elaboratori intermedi. Spesso esistono più cammini alternativi, per cui gli algoritmi di ***instradamento*** (***routing***) hanno un ruolo molto importante.

In generale (ma con molte eccezioni):
- le reti geograficamente localizzate tendono ad essere broadcast;
- le reti geograficamente molto estese tendono ad essere punto a punto.

Alcune eccezioni:
- rete geografica realizzata via satellite (e quindi broadcast);
- rete locale basata su ATM (e quindi punto a punto).

1.2.2) Scala dimensionale

Un criterio alternativo di classificazione è la scala dimensionale delle reti. In questo contesto si distingue fra ***reti locali***, ***reti metropolitane*** e ***reti geografiche***.

Distanza fra processori	Ambito	Tipo di rete

10 m.	Stanza	Rete locale
100 m.	Edificio	Rete locale
1 km.	Campus	Rete locale
10 km.	Città	Rete metropolitana
100 km.	Nazione	Rete geografica
1000 km.	Continente	Rete geografica
10.000 km.	Pianeta	Internet (Rete geografica)

La distanza è un fattore molto importante, poiché a differenti scale dimensionali si usano differenti tecniche.

1.2.3) Reti locali

Le reti locali (***Local Area Network, LAN***), in genere:

- sono possedute da una organizzazione (reti private);
- hanno un'estensione che arriva fino a qualche km;
- si distendono nell'ambito di un singolo edificio o campus (non si possono, di norma, posare cavi sul suolo pubblico);
- sono usatissime per connettere PC o workstation.

Esse si distinguono dagli altri tipi di rete per tre caratteristiche:

- ***dimensione***: la dimensione non può andare oltre un certo limite, per cui è noto a priori il tempo di trasmissione nel caso peggiore. Questa conoscenza permette di utilizzare delle tecniche particolari per la gestione del canale di comunicazione;
- ***tecnologia trasmissiva***: come già accennato, le LAN sono in generale reti 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;
- ***topologia***: sono possibili diverse topologie, le più diffuse sono il ***bus*** ed il ***ring***;
  - topologia bus:
    - in ogni istante solo un elaboratore può trasmettere, gli altri devono astenersi;
    - è necessario un meccanismo di ***arbitraggio*** per risolvere i conflitti quando due o più elaboratori vogliono trasmettere contemporaneamente;
    - l'arbitraggio può essere centralizzato o distribuito;
    - lo standard ***IEEE 802.3*** (chiamato impropriamente ***Ethernet***) è per una rete broadcast, basata su un bus, con arbitraggio distribuito, operante a 10 oppure 100 Mbps;
    - gli elaboratori trasmettono quando vogliono; se c'è una collisione aspettano un tempo casuale e riprovano;
  - topologia ring:

  - in un ring ogni bit circumnaviga l'anello in un tempo tipicamente inferiore a quello di trasmissione di un pacchetto;
  - anche qui è necessario un meccanismo di arbitraggio (spesso basato sul possesso si un ***gettone*** (***token***) che abilita alla trasmissione);
  - lo standard ***IEEE 802.5*** (derivante dalla rete IBM ***Token Ring***) è una rete broadcast basata su ring, con arbitraggio distribuito, operante a 4 o 16 Mbps.

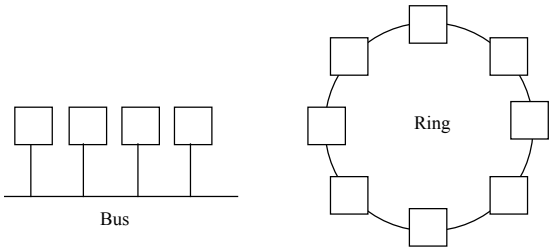


Figura 1-4: topologie bus e ring

Infine le reti broadcast possono essere classificate a seconda del meccanismo scelto per l'arbitraggio:

- ***Allocazione statica***: le regole per decidere chi sarà il prossimo a trasmettere sono fissate a priori, ad esempio assegnando un time slot ad ogni elaboratore con un algoritmo round-robin. Lo svantaggio è rappresentato dallo spreco dei time slot assegnati a stazioni che non devono trasmettere.
- ***Allocazione dinamica***: si decide di volta in volta chi sarà il prossimo a trasmettere; è necessario un meccanismo di arbitraggio delle contese, che può essere:
  - ***arbitraggio centralizzato***: un apposita apparecchiatura, ad esempio, una bus arbitration unit, accetta richieste di trasmissione e decide chi abilitare;
  - ***arbitraggio distribuito***: ognuno decide per conto proprio (come in 802.3); vedremo come si può evitare un prevedibile caos.

1.2.4) Reti metropolitane

Le reti metropolitane (***Metropolitan Area Network, MAN***)hanno un'estensione tipicamente urbana (quindi anche molto superiore a quella di una LAN) e sono generalmente pubbliche (cioé un'azienda, ad es. Telecom Italia, mette la rete a disposizione di chiunque desideri, previo pagamento di una opportuna tariffa).

Fino a qualche anno fa erano basate essenzialmente sulle tecnologie delle reti geografiche, utilizzate su scala urbana. Recentemente pero' e' stato definito un apposito standard, lo

IEEE 802.6 o ***DQDB*** (***Distributed Queue Dual Bus***), che e' effettivamente utilizzato in varie realizzazioni, molto piu' vicino alla tecnologia LAN che WAN.

Esiste un mezzo trasmissivo di tipo broadcast (due bus in 802.6) a cui tutti i computer sono attaccati.

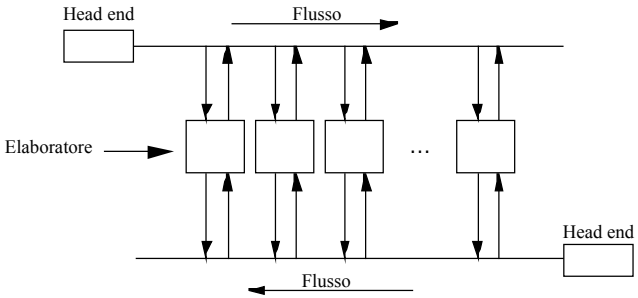


Figura 1-5: Distributed Queue Dual Bus

Ogni bus (cavo coassiale o fibra ottica) e' unidirezionale, ed ha una ***head-end*** che cadenza l'attivita' di trasmissione.

1.2.5) Reti geografiche

Le reti geografiche (***Wide Area Network, WAN***) si estendono a livello di una nazione, di un continente o dell'intero pianeta. Una WAN è tipicamente costituita di due componenti distinte:

- un insieme di elaboratori (***host*** oppure ***end system***) sui quali girano i programmi usati dagli utenti;
- una ***comunication subnet (***o ***subnet)***, che connette gli end system fra loro. Il suo compito è trasportare messaggi da un end system all'altro, così come il sistema telefonico trasporta parole da chi parla a chi ascolta.

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

- ***linee di trasmissione*** (dette anche ***circuiti***, ***canali***, ***trunk***):
- ***elementi di commutazione*** (***switching element***): gli elementi di commutazione 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. Non esiste una terminologia standard per identificare gli elementi di commutazione. Termini usati sono:
  - ***sistemi intermedi***;
  - ***nodi di commutazione pacchetti***;

- ***router*** (quello che utilizzeremo noi).

Una tipica WAN è utilizzata per connettere più LAN fra loro:

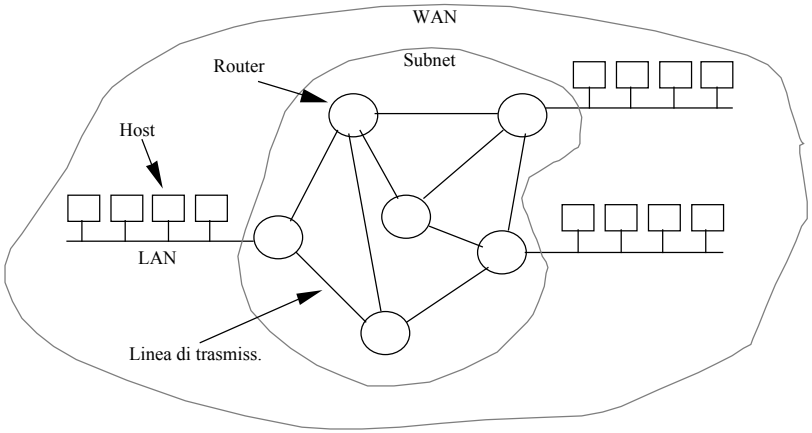


Figura 1-6: struttura tipica di una WAN

In generale una WAN contiene numerose linee (spesso telefoniche) che congiungono coppie di router.

Ogni router, in generale, deve:
1. ricevere un pacchetto da una linea in ingresso;
2. memorizzarlo per intero in un buffer interno;
3. appena la necessaria linea in uscita è libera, instradare il pacchetto su essa.

Una subnet basata su questo principio si chiama:

- ***punto a punto***;
- ***store and forward***;
- ***a commutazione di pacchetto (packet switched)***.

Molte topologie di interconnessione possono essere impiegate fra i router:

- ***a stella*** (ridondanza zero);
- ***ad anello*** (ridondanza zero);
- ***ad albero*** (ridondanza zero);
- ***magliata*** (ridondanza media);
- ***completamente connessa*** (ridondanza massima).

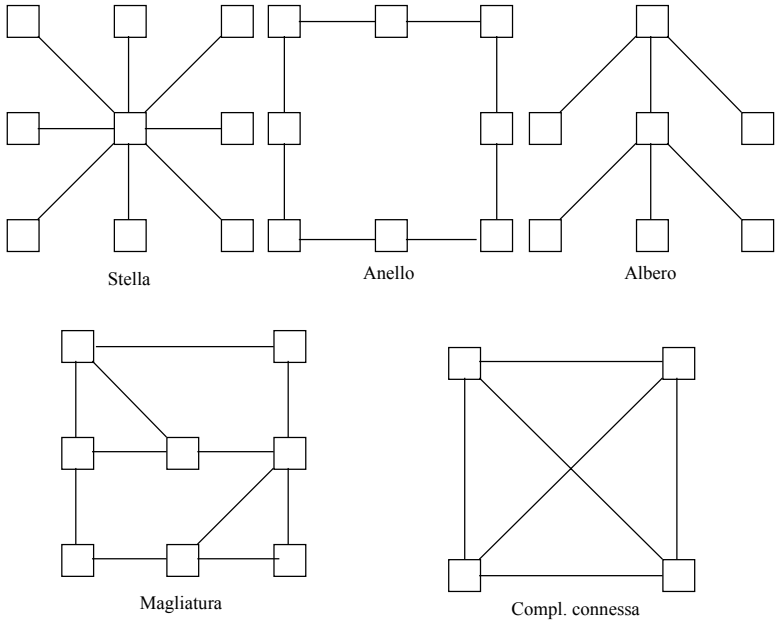


Figura 1-7: topologie di interconnessione

Un'altra possibilità è una WAN basata su satellite oppure radio al suolo.

- ***Satellite***: ogni router sente l'output del satellite e si fa sentire dal satellite. Dunque, in generale si ha:
  - broadcast downlink (cioé dal satellite a terra);
  - broadcast uplink (cioé da terra al satellite) se i router possono "sentire" quelli vicini, point to point altrimenti.

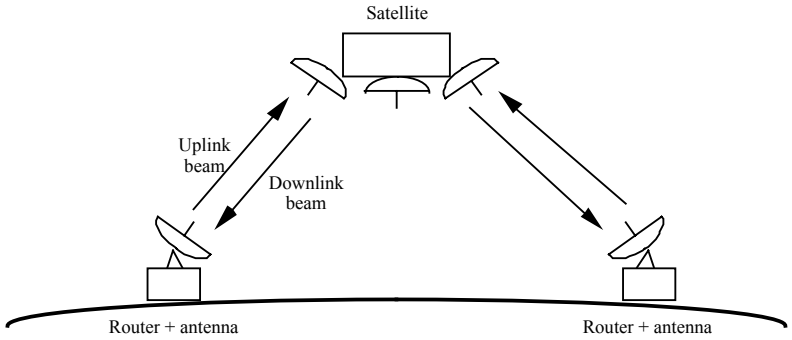


Figura 1-8: interconnessione di router via satellite

- ***Radio al suolo***: ogni router sente l'output dei propri vicini (entro una certa distanza massima):
  - anche qui siamo in presenza di una rete broadcast.

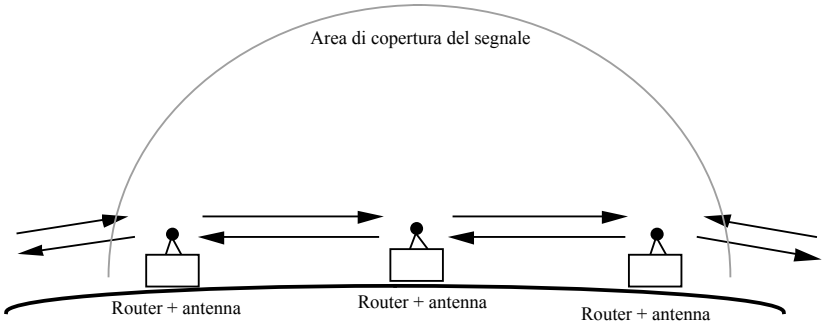


Figura 1-9: interconnessione di router via radio al suolo

Una WAN può essere anche realizzata in maniera mista: in parte cablata, in parte basata su radio o satellite.

1.2.6) Interconnessione di reti (Internetwork)

Una ***internetwork*** è formata quando reti diverse (sia LAN che MAN o WAN) sono collegate fra loro.

A prima vista, almeno in alcuni casi, la cosa è apparentemente uguale alla definizione di WAN vista precedentemente (se al posto di subnet si scrive WAN, abbiamo una internetwork costituita da una WAN e quattro LAN).

Alcuni problemi però sorgono quando si vogliono connettere fra di loro reti progettualmente diverse (spesso incompatibili fra loro). In questo caso si deve ricorrere a speciali attrezzature, dette ***gateway*** (o ***router multiprotocollo***), che oltre ad instradare i pacchetti da una rete all'altra, effettuano le operazioni necessarie per rendere possibili tali trasferimenti.

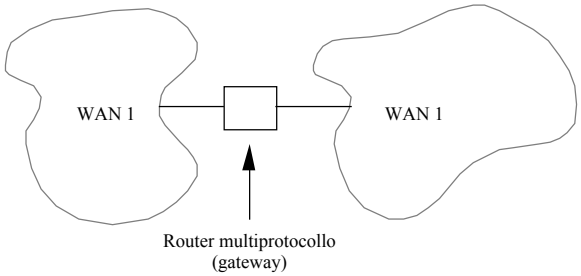


Figura 1-10: interconnessione di reti

Nel contesto del corso utilizzeremo:

- ***internet*** come sinonimo di internetwork, cioé la interconnessione di più reti generiche;
- ***Internet*** (con la I maiuscola) per riferirci alla specifica internetwork, basata su TCP/IP, che ormai tutti conoscono.

C'è molta confusione sui termini:

- ***sottorete (subnet)***, che nel contesto di una WAN è l'insieme dei router e delle linee di trasmissione;
- ***rete (network)***, che altro non è che una subnet più tutti gli host collegati;
- ***internetwork***, che è una collezione di più network, anche non omogenee, collegate per mezzo di gateway.

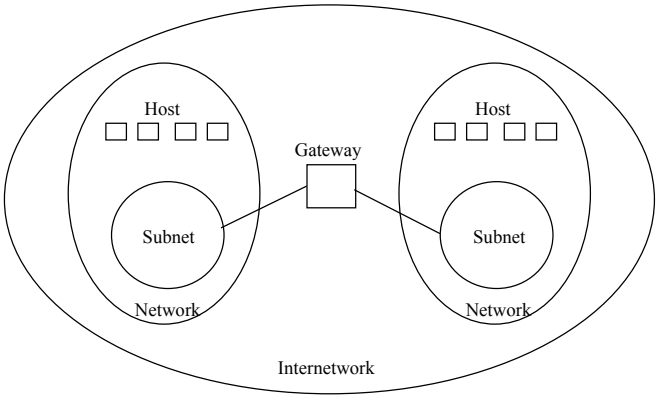


Figura 1-11: relazioni fra subnet, network e internetwork

1.3) Aspetti software delle reti

Le prime reti furono progettate cominciando dall'hardware e sviluppando il software solo successivamente, quasi come se esso fosse un'accessoria appendice dell'hardware.

Questo approccio non funziona più. Il SW di rete è oggi altamente strutturato. Esaminiamo ora, a grandi linee, tale strutturazione, che servirà da guida per l'intero corso e sulla quale torneremo spesso.

1.3.1) Gerarchie di protocollo

Per ridurre la complessità di progetto, le reti sono in generale organizzate a ***livelli***, ciascuno costruito sopra il precedente. Fra un tipo di rete ed un altra, possono essere diversi:

- il numero di livelli;
- i nomi dei livelli;
- il contenuto dei livelli;
- le funzioni dei livelli.

Comunque un principio generale è sempre rispettato:

- lo scopo di un livello è offrire certi ***servizi*** ai livelli più alti, nascondendo i dettagli sul come tali servizi siano implementati.

Il livello n su un host porta avanti una conversazione col livello n su di un'altro host. Le regole e le convenzioni che governano la conversazione sono collettivamente indicate col termine di ***protocollo di livello n***.

Le entità (processi) che effettuano tale conversazione si chiamano ***peer entitiy (entità di pari livello)***.

Il dialogo fra due peer entity di livello n viene materialmente realizzato tramite i servizi offerti dal livello (n-1).

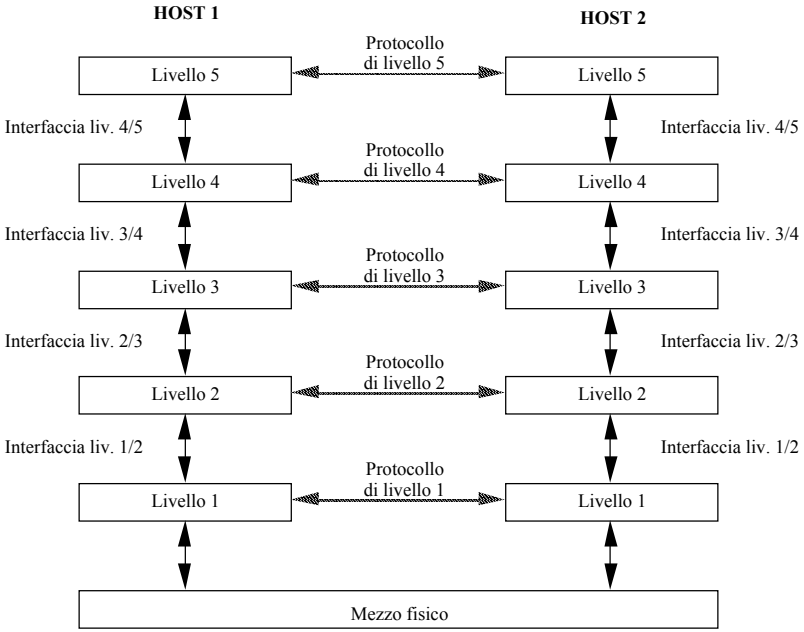


Figura 1-12: Dialogo fra peer entity

- In realtà non c'è un trasferimento diretto dal livello n di host 1 al livello n di host 2. Ogni livello di host 1 passa i ***dati***, assieme a delle ***informazioni di controllo***, al livello sottostante.
- Al di sotto del livello 1 c'è il mezzo fisico, attraverso il quale i dati vengono trasferiti da host 1 ad host 2.
- Quando arrivano a host 2, i dati vengono passati da ogni livello (a partire dal livello 1) a quello superiore, fino a raggiungere il livello n.

Fra ogni coppia di livelli adiacenti è definita una ***interfaccia***, che caratterizza:

- le operazioni primitive che possono essere richieste al livello sottostante;
- i servizi che possono essere offerti dal livello sottostante.

I vantaggi di una buona progettazione delle interfacce sono:

- minimizzazione delle informazioni da trasferire;

- possibilità di modificare l'implementazione del livello (ad es., ove le linee telefoniche venissero sostituite da canali satellitari) con una più attuale che offra gli stessi servizi.

1.3.2) Architettura di rete

L'insieme dei livelli e dei relativi protocolli è detto ***architettura di rete.***

La specifica dell' architettura deve essere abbastanza dettagliata da consentire la realizzazione di SW e/o HW che, per ogni livello, rispetti il relativo protocollo.

Viceversa, i dettagli implementativi di ogni livello e le interfacce fra livelli non sono parte dell'architettura, in quanto sono nascosti all'interno di un singolo host.

E' quindi possibile che sui vari host della rete ci siano implementazioni che differiscono fra di loro anche in termini di interfacce fra livelli, purché ogni host implementi correttamente i protocolli previsti dall'architettura. In questo caso possono dialogare fra loro anche host aventi caratteristiche (processore, sistema operativo, costruttore) diverse.

Dunque, nell'ambito di una specifica architettura di rete, si ha che:

- tutti gli host devono contenere implementazioni conformi in termini di livelli e di protocolli;
- gli host possono contenere implementazioni che differiscono in termini di dettagli implementativi e di interfacce fra livelli;

Un'architettura di rete può essere:

- ***proprietaria***;
- ***standard de facto***;
- ***standard de iure.***

Un'architettura proprietaria è basata su scelte indipendenti ed arbitrarie del costruttore, ed è generalmente incompatibile con architetture diverse. Nel senso più stretto del termine è un'architettura per la quale il costruttore non rende pubbliche le specifiche, per cui nessun altro può produrre apparati compatibili.

Esempi:

- IBM SNA (System Network Architecture)
- Digital Decnet Phase IV;
- Novell IPX;
- Appletalk.

Un'architettura standard de facto è un'architettura basata su specifiche di pubblico dominio (per cui diversi costruttori possono proporre la propria implementazione) che ha conosciuto una larghissima diffusione.

Esempi:

- Internet Protocol Suite (detta anche architettura TCP / IP).

Un'architettura standard de iure è un'architettura basata su specifiche (ovviamente di pubblico dominio) approvate da enti internazionali che si occupano di standardizzazione. Anche in questo caso ogni costruttore può proporne una propria implementazione.

Esempi:

- standard IEEE 802 per le reti locali;
- architettura OSI (Open Systems Interconnection);
- Decnet Phase V (conforme allo standard OSI).

L'insieme dei protocolli utilizzati su un host e relativi ad una specifica architettura di rete va sotto il nome di ***pila di protocolli (protocol stack)***. Si noti che un host può avere contemporaneamente attive più pile di protocolli.

1.3.3) Funzionamento del software di rete

Per comprendere i meccanismi basilari di funzionamento del software di rete si può pensare alla seguente analogia umana, nella quale un filosofo indiano vuole conversare con uno stregone africano:

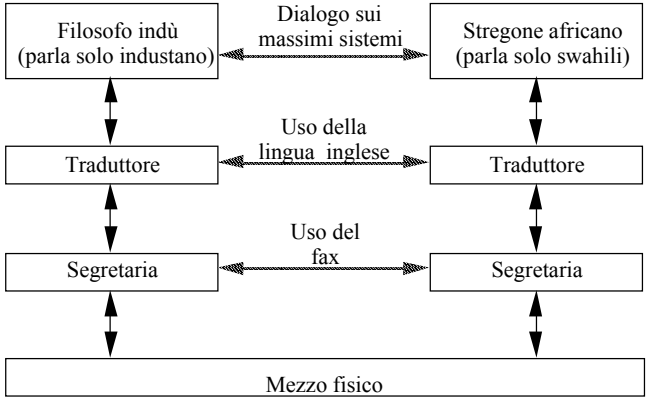


Figura 1-13: Dialogo fra grandi menti

Nel caso delle reti, la comunicazione fra le due entità di livello superiore avviene con una modalità che, almeno in linea di principio, è uguale in tutte le architetture di rete:

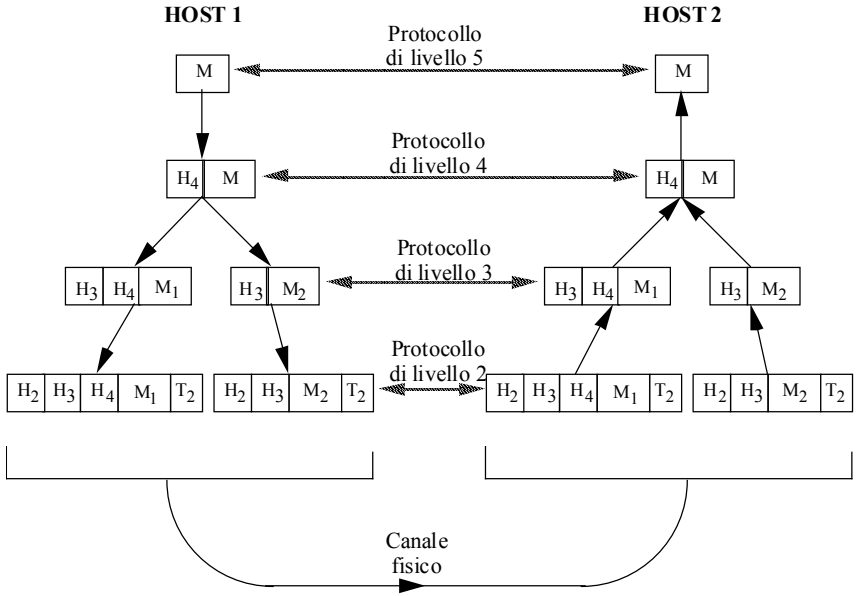


Figura 1-14: Flusso dell'informazione fra peer entity

Vediamo cosa accade:

1. il programma applicativo (livello 5) deve mandare un messaggio M alla sua peer entity;
2. il livello 5 consegna M al livello 4 per la trasmissione;
3. il livello 4 aggiunge un suo ***header*** in testa al messaggio (talvolta si dice che il messaggio è inserito nella ***busta*** di livello 4); questo header contiene informazioni di controllo, tra le quali:
    - numero di sequenza del messaggio;
    - dimensione del messaggio;
    - time stamp;
    - priorità;
4. il livello 4 consegna il risultato al livello 3;
5. il livello 3 può trovarsi nella necessità di frammentare i dati da trasmettere in unità più piccole, (***pacchetti***) a ciascuna delle quali aggiunge il suo header;
6. il livello 3 passa i pacchetti al livello 2;
7. il livello 2 aggiunge ad ogni pacchetto il proprio header (e magari un ***trailer***) e lo spedisce sul canale fisico;
8. nella macchina di destinazione i pacchetti fanno il percorso inverso, con ogni livello che elimina (elaborandoli) l'header ed il trailer di propria competenza, e passa il resto al livello superiore.

Aspetti importanti sono i seguenti:

- le peer entity pensano concettualmente ad una comunicazione orizzontale fra loro, basata sul protocollo del proprio livello, mentre in realtà comunicano ciascuna col livello sottostante attraverso l'interfaccia fra i due livelli;
- spesso i livelli bassi sono implementati in hardware o firmware (per ragioni di efficienza). Nonostante questo, spesso gli algoritmi di gestione sono complessi.

1.3.4) Interfacce e servizi

La funzione di ogni livello è di offrire servizi al livello superiore. Il livello inferiore è il ***service provider***, quello superiore è il ***service user***.

Un livello n che usufruisce dei servizi di livello (n-1) può, per mezzo di questi, a sua volta offrire al livello (n+1) i propri servizi.

I servizi sono disponibili ai ***SAP (Service Access Point)***. I SAP del livello n, o ***n-SAP***, sono i punti di accesso nei quali il livello (n+1) può accedere ai servizi del livello n. Ogni n-SAP ha un ***indirizzo*** che lo identifica univocamente.

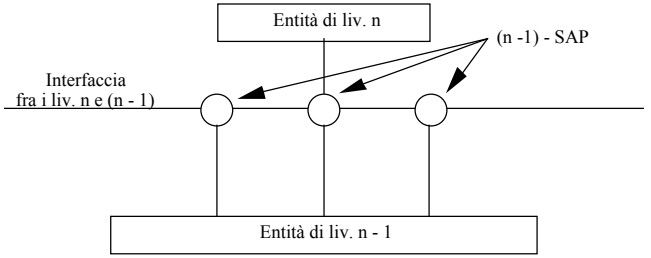


Figura 1-15: Livelli adiacenti e service access point

Analogia col telefono:

- SAP: presa a muro del telefono;
- SAP address: numero telefonico che identifica quella presa.

L'informazione passata dal livello n al livello (n-1), attraverso il (n-1)-SAP, si dice ***PDU (Protocol Data Unit) di livello n***, o ***n-PDU***.

Essa, entrando nel livello (n-1), diventa una ***SDU (Service Data Unit) di livello (n-1)***, o ***(n-1)-SDU***.

Entro il livello (n-1) viene aggiunta alla (n-1)-SDU una ***PCI (Protocol Control Information) di livello (n-1)***.

Il tutto diventa una (n-1)-PDU, che verrà passata al livello (n-2) attraverso un (n-2)-SAP.

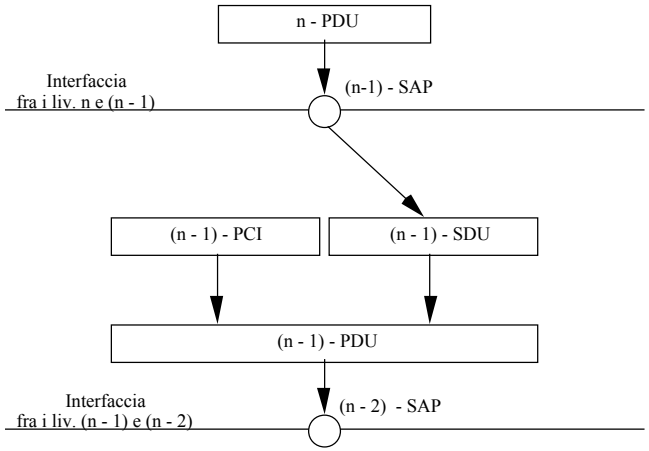


Figura 1-16: Passaggio dell'informazione fra livelli

Nomi spesso usati per i PDU:

- ***segmento*** (o***TPDU, Transport PDU***) a livello transport
- ***pacchetto*** (***packet***) a livello network
- ***trama*** (***frame***) a livello data link

Nome per il PCI:

- ***busta***

1.3.5) Servizi connection-oriented e connectionless

Ci sono due principali classi di servizi offerti da un livello a quello superiore:

- ***servizi connection-oriented***;
- ***servizi connectionless***.

Servizi connection-oriented

I servizi connection-oriented sono modellati secondo il sistema telefonico, dove per parlare con qualcuno si alza il telefono, si chiama, si parla e poi si riattacca. Ovvero:

1. si stabilisce una connessione;

2. si scambiano informazioni;
3. si rilascia la connessione.

Analogamente, un servizio connection-oriented si sviluppa in 3 fasi:

1. si stabilisce una connessione, cioé si crea con opportuni mezzi un "canale di comunicazione" fra la sorgente e la destinazione. La relativa attività tipicamente coinvolge un certo numero di elaboratori nel cammino fra sorgente e destinazione;
2. la connessione, una volta stabilita, agisce come un tubo digitale lungo il quale scorrono tutti i dati trasmessi, che arrivano nello steso ordine in cui sono partiti;
3. si rilascia la connessione (attività che coinvolge di nuovo tutti gli elaboratori sul cammino).

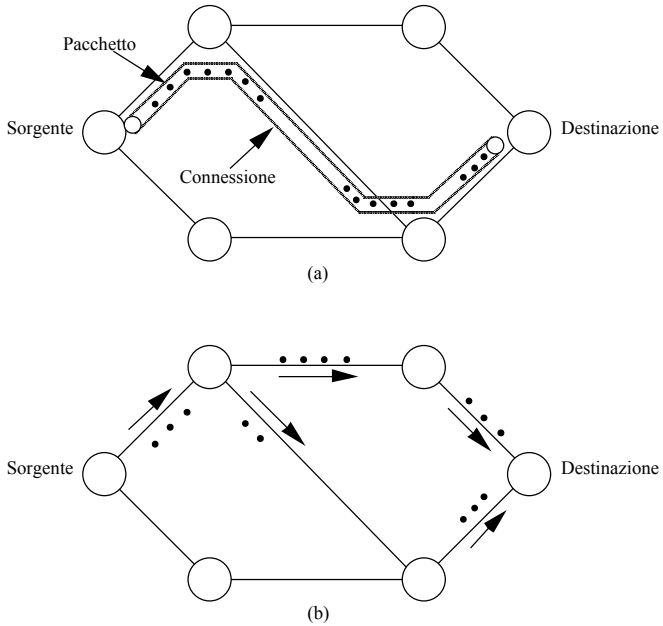


Figura 1-17: Servizi connection-oriented (a) e connectionless (b)

Servizi connectionless

I servizi connectionless sono modellati secondo il sistema postale: ogni lettera viaggia indipendentemente dalle altre; arriva quando arriva, e forse non arriva. Inoltre, due lettere con uguale mittente e destinatario possono viaggiare per strade diverse.

Analogamente, in un servizio connectionless, i pacchetti (PDU) viaggiano indipendentemente gli uni dagli altri, possono prendere strade diverse ed arrivare in ordine diverso da quello di partenza o non arrivare affatto.

La fase è una sola:

- invio del pacchetto (corrisponde all'immissione della lettera nella buca).

1.3.6) Affidabilità del servizio

Un servizio è generalmente caratterizzato dall'essere o no ***affidabile*** (***reliable***).

Un ***servizio affidabile*** non perde mai dati, cioé assicura che tutti i dati spediti verranno consegnati al destinatario. Ciò generalmente richiede che il ricevente invii un ***acknowledgement (conferma)*** alla sorgente per ogni pacchetto ricevuto. Si introduce ovviamente overhead, che in certe situazioni può non essere desiderabile.

Viceversa, un ***servizio non affidabile*** non offre la certezza che i dati spediti arrivino effettivamente a destinazione.

Si noti che se un certo livello non offre nessun servizio affidabile, qualora tale funzionalità sia desiderata dovrà essere fornita da almeno uno dei livelli superiori (vedremo che ciò accade spesso).

Esempi:

- ***reliable connection oriented***: trasferimento di file (non devono mancare pezzi e il file non deve essere "rimescolato");
- ***non reliable connection oriented***: nelle trasmissioni isocrone (quali voce e video) le relazioni temporali fra i bit del flusso devono essere mantenute. E' meglio qualche disturbo ogni tanto, piuttosto che interruzioni momentanee, ma avvertibili, del flusso di dati;
- ***non reliable connectionless*** (detto anche ***datagram service***, da telegram): distribuzione di posta elettronica pubblicitaria, non importa se qualche messaggio si perde.
- ***reliable connectionless*** (detto anche ***acknowledged datagram service***): si invia un breve messaggio e si vuole essere assolutamente sicuri che è arrivato.

1.3.7) Primitive di definizione del servizio

Un servizio di livello n è formalmente specificato da un insieme di ***primitive*** (cioé operazioni) che un'entità di livello (n+1) può adoperare per accedere al servizio. Esse possono indicare al servizio:

- l'azione da compiere (l'informazione viagga da livello n al livello (n-1));
- cosa riportare in merito ad una azione effettuata dalla peer entity di livello n (l'informazione viagga dal livello (n-1) al livello n).

Un esempio di primitive può essere il seguente:

Primitiva	Significato
`request()`	si chiede al servizio di fare qualcosa
`Indication()`	si viene avvertiti, dal servizio, di qualche evento
`response()`	si vuole rispondere ad un evento
`confirm()`	la risposta che si attendeva è arrivata

Per stabilire una connessione fra le peer entity A a B si avrà che:

Entity	Azione	Flusso informazione	Significato
A	invia una `connect.request()`	da livello n a livello (n-1)	A desidera connettersi
B	riceve una `connect.indication()`	da livello (n-1) a livello n	qualcuno vuole connettersi
B	invia una `connect.response()`	da livello n a livello (n-1)	B accetta (oppure no)
A	riceve una `connect.confirm()`	da livello (n-1) a livello n	B ha accettato (o no)

Le primitive hanno vari parametri (mittente, destinatario, tipo del servizio richiesto, ecc.), che possono essere usati dalle peer entity per negoziare le caratteristiche della connessione. I dettagli della negoziazione fanno parte del protocollo.

Ad esempio, un servizio può essere richiesto in modalità confermata oppure non confermato.

Per il servizio confermato avremo:

- `request()`;
- `indication()`;
- `response()`;
- `confirm()`.

Mentre per il servizio non confermato:

- `request()`;
- `indication()`.

Connect è sempre confermato (ovviamente), ma altri servizi no.

Vediamo ora un esempio di servizio connection oriented con 8 primitive:

1. `connect.request()`;
2. `connect.indication()`;
3. `connect.response()`;
4. `connect.confirm()`;
5. `data.request()`: si cerca di inviare dati;
6. `data.indication()`: sono arrivati dei dati;
7. `disconnect.request()`: si vuole terminare la connessione;
8. `disconnect.indication()`: l'altra entity vuole terminare.

La sequenza di primitive che entrano in gioco successivamente nel corso della gestione di una connessione è la seguente:

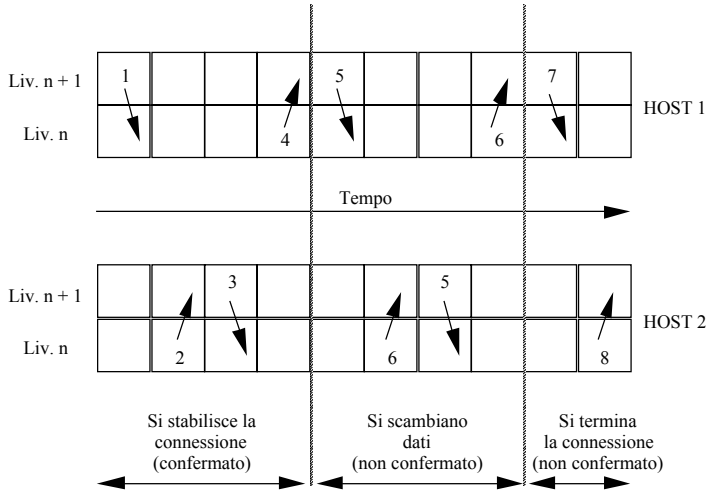


Figura 1-18: Esempio di attivazione, uso e rilascio di una connessione

1.3.8) Servizi vs. protocolli

Servizi e protocolli sono spesso confusi, ma sono concetti ben distinti.

Servizio	insieme di operazioni primitive che un livello offre al livello superiore. Come tali operazioni siano implementate non riguarda il livello superiore.
Protocollo	insieme di regole che governano il formato ed il significato delle informazioni (messaggi, frame, pacchetti) che le peer entity si scambiano fra loro. Le entità usano i protocolli per implementare i propri servizi.

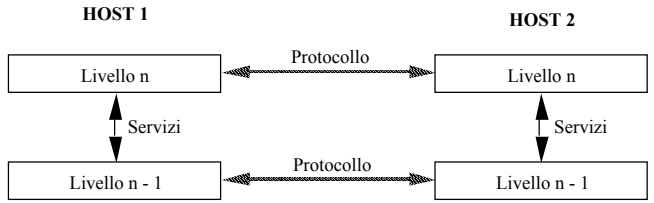


Figura 1-19: Relazione fra protocolli e servizi

Nota: una ipotetica primitiva `send_packet()`, alla quale l'utente fornisce il puntatore a un pacchetto già assemblato, non è conforme a questa filosofia.

1.3.9) Aspetti di progetto dei livelli

Decisioni di progetto vanno prese, nei vari livelli, in merito a diverse problematiche. Le principali sono:

1. Meccanismi di identificazione di mittente e destinatario (cioé ***indirizzamento***), in ogni livello.
2. Regole per il ***trasferimento dati*** (livelli bassi):
    - in una sola direzione (***simplex connection***);
    - in due direzioni ma non contemporaneamente (***half-duplex connection***).
    - in due direzioni contemporaneamente (***full-duplex connection***);
3. Meccanismi per il controllo degli ***errori di trasmissione***; è possibile:
    - rilevarli oppure no;
    - correggerli oppure no;
    - avvertire il mittente oppure no.
4. Meccanismi per il mantenimento (o la ricostruzione) dell'***ordine originario*** dei dati.
5. Meccanismi per regolare le ***velocità*** di sorgente e destinazione.
6. Decisioni sulla ***dimensione*** (minima o massima) dei messaggi da inviare, e su come eventualmente frammentarli.
7. Meccanismi di ***multiplexing*** di varie "conversazioni" su di un'unica connessione (se stabilire la connessione è costoso).
8. Meccanismi di ***routing*** dei messaggi se esistono più strade alternative, ed eventualmente di suddivisione di una "conversazione" su più connessioni contemporaneamente (per aumentare la velocità di trasferimento dei dati).

1.4) La realtà nel mondo delle reti

Iniziamo ad esaminare due importanti realtà nel mondo delle reti:

1. ***OSI Reference Model***;

2. ***Internet Protocol Suite*** (detta anche ***architettura TCP/IP*** o, piuttosto impropriamente, ***TCP/IP reference model***).

Un modello di riferimento è cosa diversa da un'architettura di rete:

Modello di riferimento	definisce il numero, le relazioni e le caratteristiche funzionali dei livelli, ma non definisce i protocolli effettivi
Architettura di rete	definisce, livello per livello, i protocolli effettivi

1.4.1) Modello OSI

L'OSI (Open Systems Interconnection) Reference Model è il frutto del lavoro della ISO (International Standard Organization), ed ha lo scopo di:

- fornire uno standard per la connessione di sistemi aperti, cioé in grado di colloquiare gli uni con gli altri;
- fornire una base comune per lo sviluppo di standard per l'interconnessione di sistemi;
- fornire un modello rispetto a cui confrontare le varie architetture di rete.

Esso non include di per se la definizione di protocolli specifici (che sono stati definiti successivamente, in documenti separati).

Principi di progetto seguiti durante lo sviluppo del modello OSI:

- ogni livello deve avere un diverso livello di astrazione;
- ogni livello deve avere una funzione ben definita;
- la scelta dei livelli deve:
    - minimizzare il passaggio delle informazioni fra livelli;
    - evitare:
        - troppe funzioni in un livello;
        - troppi livelli.

Il modello OSI consiste di 7 livelli (i maligni dicono che ciò fu dettato dal desiderio di rendere il modello compatibile con l'architettura SNA dell'IBM).

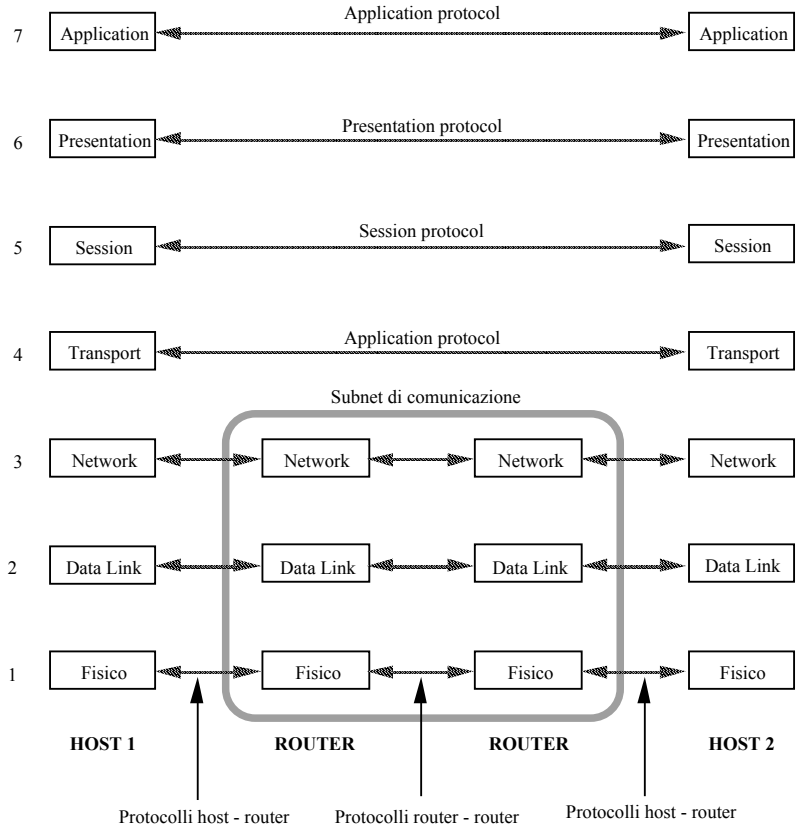


Figura 1-20: Il modello OSI

Spesso, per visualizzare le competenze (in termini di livelli gestiti) dei vari elaboratori sul cammino, si usano diagrammi simili al seguente:

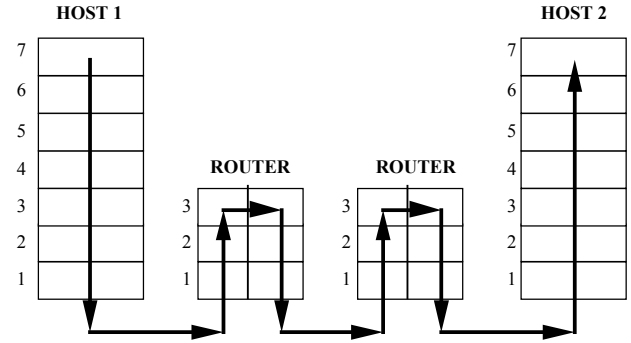


Figura 1-21: Rappresentazione schematica dei livelli gestiti lungo un cammino

Si noti che il modello OSI non è un'architettura di rete, perché dice solo cosa devono fare i livelli, ma non definisce né i servizi né i protocolli. Per questo ci sono separati documenti di definizione degli standard.

Livello fisico

Ha a che fare con la trasmissione di bit "grezzi" su un canale di comunicazione.

Gli aspetti di progetto sono:

- volti a garantire che se parte un 1, arrivi effettivamente un 1 e non uno zero;
- largamente riguardanti le caratteristiche meccaniche, elettriche e procedurali delle ***interfacce di rete*** (componenti che connettono l'elaboratore al mezzo fisico) e le caratteristiche del mezzo fisico.

Si caratterizzano, tra gli altri:

- tensioni scelte per rappresentare 0 ed 1;
- durata (in microsecondi) di un bit;
- trasmissione simultanea in due direzioni oppure no;
- forma dei connettori.

Livello Data Link

Lo scopo di questo livello è far si che un mezzo fisico trasmissivo appaia, al livello superiore, come una linea di trasmissione esente da errori di trasmissione non rilevati.

Normalmente funziona così:

- spezzetta i dati provenienti dal livello superiore in frame (da qualche centinaia a qualche migliaia di byte);
- invia i frame in sequenza;
- aspetta un ***acknowledgement frame*** (***ack***) per ogni frame inviato.

Incombenze:

- aggiunta di delimitatori (***framing***) all'inizio ed alla fine del frame (che succede se il delimitatore è presente dentro il frame ?);
- ***gestione di errori*** di trasmissione causati da:
    - errori in ricezione;
    - perdita di frame;
    - duplicazione di frame (da perdita di ack);
- ***regolazione del traffico*** (per impedire che il ricevente sia "sommerso" di dati);
- meccanismi per l'invio degli ack:
    - ***frame separati*** (che però competono col regolare traffico nella stessa direzione);
    - ***piggybacking*** (da pickaback, cioé trasportare sulle spalle).

Le reti broadcast hanno un'ulteriore problema: il controllo dell'accesso al canale trasmissivo, che è condiviso. Per questo hanno uno speciale sottolivello del livello data link, il ***sottolivello MAC (Medium Access Control)***.

Livello Network

Lo scopo del livello è controllare il funzionamento della subnet di comunicazione.

Inizialmente tale livello offriva solamente servizi connection oriented; successivamente fu aggiunta la modalità connectionless.

Incombenze:

- ***routing***, cioé scelta del cammino da utilizzare. Può essere:
    - statico (fissato ogni tanto e raramente variabile);
    - dinamico (continuamente aggiornato, anche da un pacchetto all'altro);

- ***gestione della congestione***: a volte troppi pacchetti arrivano ad un router (es.: da molte linee in ingresso ad un unica linea di uscita);
- ***accounting***: gli operatori della rete possono far pagare l'uso agli utenti sulla base del traffico generato;
- ***conversione di dati*** nel passaggio fra una rete ed un'altra (diversa):
    - indirizzi da rimappare;
    - pacchetti da frammentare;
    - protocolli diversi da gestire.

Livello Transport

Lo scopo di questo livello è accettare dati dal livello superiore, spezzettarli in pacchetti, passarli al livello network ed assicurarsi che arrivino alla peer entity che si trova all'altra estremità della connessione. In più, fare ciò efficientemente, isolando i livelli superiori dai cambiamenti della tecnologia di rete sottostante.

Il livello transport è il primo livello realmente ***end-to-end***, cioé da host sorgente a host destinatario: le peer entity di questo livello portano avanti una conversazione senza intermediari.

Si noterà che certe problematiche sono, in ambito end-to-end, le stesse che il livello data link ha nell'ambito di una singola linea di comunicazione; le soluzioni però sono alquanto diverse per la presenza della subnet di comunicazione.

Incombenze:

- ***creazione di connessioni di livello network*** (attraverso i servizi del livello network) per ogni connessione di livello transport richiesta:
    - normalmente, una connessione network per ciascuna connessione transport;
    - per ottenere un alto throughput: molte connessioni network per una singola connessione transport;
    - se è alto il costo di una connessione network: una singola connessione network viene usata per molte connessioni transport, con meccanismi di multiplexing;
- ***offerta di vari servizi*** al livello superiore:
    - canale punto a punto affidabile, che consegna i dati in ordine e senza errori (il servizio più diffuso, connection oriented);
    - invio di messaggi isolati, con o senza garanzia di consegna (connectionless);
    - broadcasting di messaggi a molti destinatari (connectionless).

Livello Session

Ha a che fare con servizi più raffinati che non quelli del transport layer, come ad es.:

- ***token management***: autorizza le due parti, a turno, alla trasmissione.

Come vedremo nel seguito, questo livello non ha avuto un grande successo.

Livello Presentation

E' interessato alla sintassi ed alla semantica delle informazioni da trasferire. Ad esempio, si occupa di convertire tipi di dati standard (caratteri, interi) da rappresentazioni specifiche della piattaforma HW di partenza in una rappresentazione "on the wire" e poi in quella specifica dell' HW di arrivo.

Anche questo livello non ha avuto molto successo.

Livello Application

Prevede che qui risieda tutta la varietà di protocolli che sono necessari per offrire i vari servizi agli utenti, quali ad esempio:

- ***terminale virtuale***;
- ***transferimento file***;
- ***posta elettronica***.

Attraverso l'uso di questi protocolli si possono scrivere applicazioni che offrono i suddetti servizi agli utenti finali.

1.4.2) Internet Protocol Suite

La "madre di tutte le reti" fu ***Arpanet***, originata da un progetto di ricerca finanziato dal DoD (Department of Defense) americano. Lo scopo era creare una rete estremamente affidabile anche in caso di catastrofi (o eventi bellici) che ne eliminassero una parte. Arpanet, attraverso varie evoluzioni, ha dato origine alla attuale Internet.

Nel corso dello sviluppo, per integrare via via tipi diversi di reti, si vide la necessità di una nuova architettura, mirata fin dall'inizio a consentire l'interconnessione di molteplici reti (internetwork).

L'architettura divenne, più tardi, nota coi nomi di ***Internet Protocol Suite***, ***architettura TCP/IP*** e ***TCP/IP reference model***, dal nome dei suoi due protocolli principali. Essa non è un modello nel senso stretto del termine, in quanto include i protocolli effettivi, che sono specificati per mezzo di documenti detti ***RFC*** (***Request For Comments***).

I livelli TCP / IP hanno questa relazione con quelli OSI:

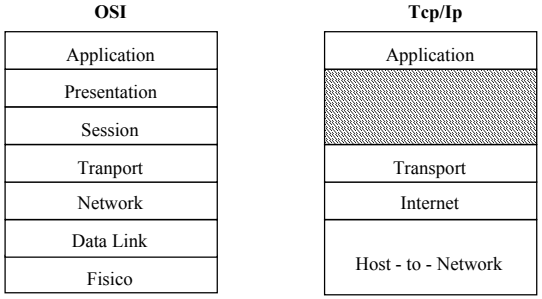


Figura 1-22: Relazione fra i livelli OSI e TCP / IP

I requisiti di progetto stabiliti fin dall'inizio (estrema affidabilità e tolleranza ai guasti, possibilità di interconnessione di più reti) portarono alla scelta di una rete:

- packet-switched;
- basata su un livello connectionless di internetwork.

Livello host-to-network

Il livello più basso non è specificato nell'architettura, che prevede di utilizzare quelli disponibili per le varie piattaforme HW e conformi agli standard.

Tutto ciò che si assume è la capacità dell'host di inviare pacchetti IP sulla rete.

Livello Internet

E' il livello che tiene insieme l'intera architettura. Il suo ruolo è permettere ad un host di iniettare pacchetti in una qualunque rete e fare il possibile per farli viaggiare, indipendentemente gli uni dagli altri e magari per strade diverse, fino alla destinazione, che può essere situata anche in un'altra rete. Dunque è ***connectionless***. E' un servizio ***best-effort datagram***. E' definito un formato ufficiale dei pacchetti ed un protocollo, ***IP (Internet Protocol)***.

Incombenze:

- routing;
- controllo congestione.

Livello Transport

E' progettato per consentire la conversazione delle peer entity sugli host sorgente e destinazione (end-to-end). Sono definiti due protocolli in questo livello:

- ***TCP (Transmission Control Protocol)***: è un protocollo connesso ed affidabile (ossia tutti i pacchetti arrivano, e nell'ordine giusto). Frammenta il flusso in arrivo dal livello superiore in messaggi separati che vengono passati al livello Internet. In arrivo, i pacchetti vengono riassemblati in un flusso di output per il livello superiore.
- ***UDP (User Datagram Protocol)***: è un protocollo non connesso e non affidabile, i pacchetti possono arrivare in ordine diverso o non arrivare affatto.

Livello Application

Nell'architettura TCP / IP non ci sono i livelli session e presentation (non furono ritenuti necessari; l'esperienza col modello OSI ha mostrato che questa visione è condivisibile).

Sopra il livello transport c'è direttamente il livello application, che contiene tutti i protocolli di alto livello vengono usati dalle applicazioni reali.

I primi protocolli furono:

- ***Telnet***: terminale virtuale;
- ***FTP (File Transfer Protocol)***: file transfer;
- ***SMTP (Simple Mail Transfer Protocol) e POP (Post Office Protocol)***: posta elettronica.

Successivamente se ne sono aggiunti altri, fra cui:

DNS (Domain Name Service): mapping fra nomi di host e indirizzi IP numerici;

NNTP (Network News Transfer Protocol): trasferimento di articoli per i newsgroup;

HTTP (HyperText Transfer Protocol): alla base del Word Wide Web.

I vari protocolli nell'architettura TCP / IP si collocano come segue:

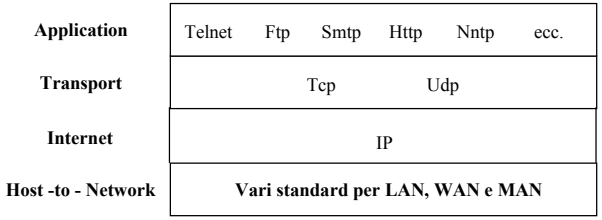


Figura 1-23: Relazione fra i livelli e i protocolli dell'architettura TCP / IP

1.4.3) Confronto fra modello di riferimento OSI e architettura TCP/IP

Somiglianze:

- basati entrambi sul concetto di pila di protocolli indipendenti;
- funzionalità simili in entrambi per i vari livelli.

Differenze di fondo:

- OSI nasce come modello di riferimento (utilissimo per le discussioni generali), i protocolli vengono solo successivamente;
- TCP / IP nasce coi protocolli, il modello di riferimento viene a posteriori.

Conseguenze:

essendo il modello OSI nato prima dei relativi protocolli, successe che:

- il modello era, ed è tuttora, molto generale (punto a favore);
- vi era insufficiente esperienza nella progettazione dei livelli (punto a sfavore). Ad esempio:
    - il livello data-link (pensato all'origine per linee punto-punto) ha dovuto essere sdoppiato per gestire reti broadcast;
    - mancò del tutto l'idea di internetworking: si pensava ad una rete separata, gestita dallo stato, per ogni nazione.

I protocolli dell'architettura TCP / IP sono invece il punto di partenza del progetto, per cui:

- l'architettura è molto efficiente (punto a favore);
- il reference model non è generale, in quanto descrive solo questa particolare architettura (punto a sfavore);

- è difficile rimpiazzare i protocolli se necessario (punto a sfavore).

Confronto fra pile di protocolli OSI e TCP/IP

I protocolli OSI non sono riusciti ad affermarsi sul mercato per una serie di ragioni:

- infelice scelta di tempo: la definizione dei protocolli è arrivata troppo tardi, quando cioé quelli TCP / IP si erano già considerevolmente diffusi. Le aziende non se la sono sentite di investire risorse nello sviluppo di una ulteriore architettura di rete;
- infelici scelte tecnologiche: i sette livelli (e i relativi protocolli) sono stati dettati in realtà dalla architettura SNA dell' IBM, più che da considerazioni di progetto. Per cui il progetto soffre di vari difetti:
    - grande complessità e conseguente difficoltà di implementazione;
    - inutili i livelli session e presentation;
    - non ottimali attribuzioni di funzioni ai vari livelli:
        - alcune funzioni appaiono in molti livelli (es. controllo errore e flusso in tutti i livelli);
        - altre funzioni mancano del tutto (ad es. sicurezza e gestione rete);
- infelice implementazione: le prime realizzazioni erano lente ed inefficienti, mentre contemporaneamente TCP / IP era molto ben implementato (e per di più gratis!). In effetti i protocolli dell'architettura TCP / IP invece sono stati implementati efficientemente fin dall'inizio, per cui si sono affermati sempre più, e quindi hanno goduto di un crescente supporto che li ha resi ancora migliori.

Ad ogni modo, neanche l'architettura TCP / IP è priva di problemi:

- l'architettura TCP / IP non ha utilità come modello (non serve ad altro che a descrivere se stessa);
- non c'è una chiara distinzione fra protocolli, servizi e interfacce, il che rende più difficile l'evoluzione dell'architettura;
- alcune scelte di progetto cominciano a pesare (ad es., indirizzi IP a soli 16 bit).

In conclusione:

- OSI è ottimo come modello, mentre i suoi protocolli hanno avuto poco successo;
- TCP / IP è ottima (per ora) come architettura di rete, ma inutile come modello.

Nel resto del corso ci concentreremo su un modello di riferimento OSI modificato:

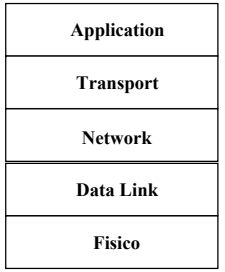


Figura 1-24: Il modello OSI modificato

1.4.4) Esempi di architetture di rete

Netware (Novell)

E' l'architettura di rete più diffusa nel mondo PC. E' precedente a OSI, e assomiglia molto a TCP / IP.

Application	File server, ecc.
Transport	NCP, SPX, Tcp
Network	IPX
Data Link	Ethernet, Token Ring, ecc.
Fisico	Ethernet, Token Ring, ecc.

Figura 1-25: L'architettura Novell Netware

- IPX: best-effort datagram (connectionless);
- NCP: reliable connection oriented;
- SPX: unreliable connectionless

Nota: la pratica di avere un livello network best-effort e connectionless, e sopra esso un livello transport reliable e connected, è molto diffusa. Sono così anche:

- Appletalk;
- TCP / IP.

Arpanet

Ncque a metà degli anni '60 (ai tempi della guerra fredda); il DoD volle una rete robustissima anche in caso di catastrofi, in grado di non interrompere le connessioni in atto anche se alcune sue componenti fossero state distrutte.

Si scelse quindi una rete packed-switched, formata dagli host e da una subnet di comunicazione costituita da vari IMP (Interface Message Processor) collegati da linee di trasmissione.

Il software venne diviso in due parti:

- SW per gli host
- SW per la subnet (ossia per gli IMP)

e furono definiti alcuni protocolli:

- Host-IMP protocol;
- IMP-IMP protocol;
- Source IMP-destination IMP protocol.

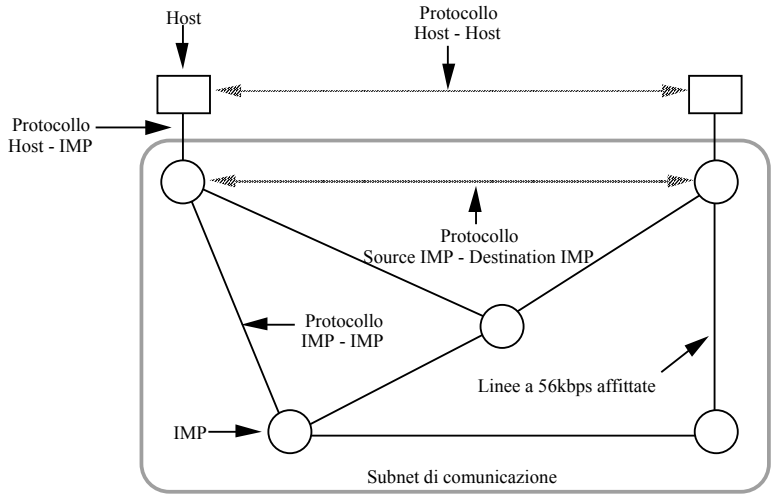


Figura 1-26: L'architettura di Arpanet

Successivamente Arpanet si sviluppò incorporando altre reti, il che mostrò l'inadeguatezza dei protocolli originari rispetto alle problematiche di internetworking.

Nacque di conseguenza, verso la metà degli anni '70, l'architettura TCP / IP, che il giorno 1 / 1 / 1983 divenne lo standard di Arpanet. TCP / IP fu poi mantenuta anche per l'evoluzione di Arpanet, NSFNET (metà degli anni '80), basata su linee da 56Kbps all'inizio e poi più veloci (1.5 Mbps nel 1990, 34 Mbps sulle linee principali oggi).

Il continuo aggiungersi di reti, ad Arpanet prima e ad NSFNET poi, ha creato quella che oggi viene comunemente chiamata Internet, costituita da milioni di host e utilizzata da decine di milioni di utenti. Internet si raddoppia all'incirca ogni anno, ed ha suscitato grande interesse sia per i privati cittadini che per le aziende.

Decnet Phase V (Digital)

Come già detto, Decnet Phase V è un'architettura di rete totalmente conforme al modello OSI. Decnet ha costruito una piattaforma comprendente i livelli 1, 2 e 3 del modello OSI, sulla quale possono appoggiarsi due diverse pile di protocolli:

- pila OSI;
- pila Decnet Phase IV (proprietaria).

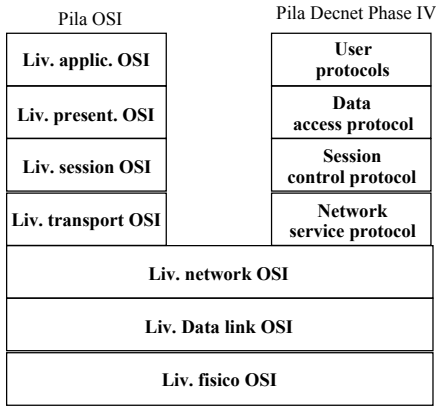


Figura 1-27: L'architettura Decnet Phase V

SNA (IBM)

SNA (System Network Architecture) è un'architettura di rete ancora molto diffusa, sopratutto nelle grandi aziende dotate di sistemi informativi IBM. Nacque a metà degli anni '70, periodo in cui i sistemi informativi erano basati sull'uso di mainframe e terminali.

Fu pensata per connettere fra loro più mainframe, ai quali dovevano poter essere connessi moltissimi terminali, anche geograficamente lontani.

E' un'architettura estremamente complessa e poco adatta all'attuale impostazione dei sistemi di calcolo (reti locali e applicazioni tipo client-server) per cui IBM sta migrando verso una strategia di networking più ampia, nella quale viene fornito il supporto ad SNA, APPN (successore di SNA: architettura proprietaria ma di dominio pubblico), OSI, TCP/IP ed altri (fra cui Novell IPX).

L'architettura SNA è la seguente:

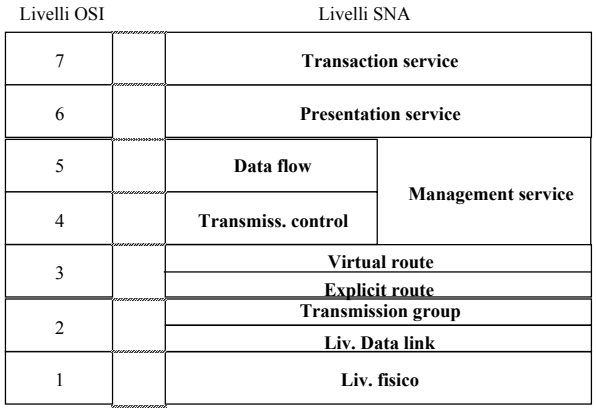


Figura 1-28: L'architettura IBM SNA

1.4.5) Autorità nel mondo degli standard

Queste sono le principali autorità nel mondo degli standard:

- ***PTT (Post, Telephone and Telegraph)***: amministrazione statale che gestisce i servizi trasmissivi (in Italia è il Ministero delle Poste);

- ***CCITT (Comité Consultatif International de Telegraphie et Telephonie)***: organismo internazionale che emette le specifiche tecniche che devono essere adottate dalle PTT. E' entrato da poco a far parte dell'***ITU (International Telecomunication Union)***;
- ***ISO (International Standard Organization)***: il principale ente di standardizzazione internazionale, che si occupa fra l'altro anche di reti;
- ***ANSI (American National Standards Institution)***: rappresentante USA nell' ISO;
- ***UNINFO***: rappresentante italiano, per le reti, nell'ISO;
- ***IEEE (Institute of Electrical and Electronic Engineers)***: organizzazione professionale mondiale degli ingegneri elettrici ed elettronici; ha gruppi di standardizzazione sulle reti;
- ***IRTF (Internet Research Task Force)***: comitato rivolto agli aspetti di ricerca a lungo termine in merito alla rete Internet;
- ***IETF (Internet Engineering Task Force)***: comitato rivolto agli aspetti di ingegnerizzazione a breve termine della rete Internet;
- ***IAF (Internet Architecture Board)***: comitato che prende le decisioni finali su nuovi standard da adottare per Internet, di solito proposti da IETF o IRTF.

2) Il livello uno (Fisico)

In questo capitolo verranno illustrati gli aspetti principali del livello fisico, che riguardano:

- la teoria della trasmissione delle informazioni, per capire quali sono i limiti fondamentali imposti da madre natura;
- le caratteristiche dei mezzi trasmissivi più comuni;
- le caratteristiche del sistema telefonico.

2.1) Basi teoriche della trasmissione dati

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 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);
- ***mezzi ottici*** (LED, laser e fibre ottiche): in essi il fenomeno utilizzato è la luce. Si tratta dei mezzi più recenti, che hanno rivoluzionato il settore.

Rappresentando il valore nel tempo del fenomeno fisico utilizzato come una funzione f(t), si può studiare matematicamente il segnale risultante.

In linea di principio, la trasmissione può avvenire con due modalità differenti: trasmissione di ***segnale analogico*** e trasmissione di ***segnale digitale***.

La differenza fondamentale fra un segnale analogico e uno digitale è che:
- 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.

Come vedremo in seguito, in certi casi (e con certe tecniche) è utile trasformare un segnale analogico in uno digitale e viceversa.

2.1.1) Analisi di Fourier (analisi armonica)

Premessa: una ***funzione sinusoidale*** , quale il seno o il coseno, è caratterizzata da alcuni parametri :
- ***ampiezza*** A (la differenza fra il valore massimo ed il minimo);
- ***periodo*** T (la quantità T di tempo trascorsa la quale la funzione si ripete);
- ***frequenza*** : l'inverso del periodo f = 1 / T, misurata in ***cicli al secondo*** (***Hz***).

Fourier (matematico francese dell'800) dimostrò che una funzione g(t), definita in un intervallo T, può essere espressa come una somma di un numero infinito di funzioni sinusoidali:

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

dove f = 1 / T è la ***frequenza fondamentale*** ed a_n e b_n sono le ampiezze dell'ennesima armonica (o termine), che ha una frequenza n volte più grande della frequenza

fondamentale. I valori di c, a_n e b_n sono tutti calcolabili come opportuni integrali di g(t) in t.

Dunque, un segnale variabile nel tempo è di fatto ***equivalente ad una somma di funzioni sinusoidali*** aventi ciascuna una propria ampiezza e frequenza. Si può quindi rappresentare un segnale g(t) di durata T in un modo diverso, e cioè attraverso il suo ***spettro di frequenze***, ossia attraverso la sua scomposizione in sinusoidi.

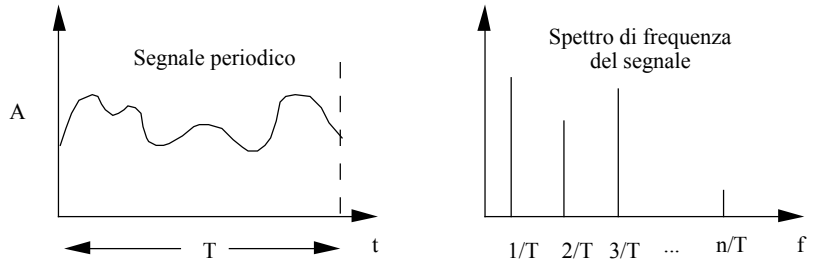


Figura 2-1: Un segnale e il suo spettro di frequenze

Qualunque segnale è dunque 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*** (***frequency band***) del segnale.

Diversi fattori influenzano le caratteristiche della banda:
- tanto più è breve la durata T del segnale, tanto più è alto il valore della frequenza fondamentale;
- tanto più velocemente nel tempo varia la g(t), tanto più numerose sono le armoniche necessarie a descriverlo.

Anche i mezzi fisici sono caratterizzati da una banda di frequenze, detta ***banda passante***. Essa 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.

A volte la dimensione della banda passante dipende dalle caratteristiche fisiche del mezzo trasmissivo, a volte deriva dalla presenza di opportuni ***filtri*** che tagliano le frequenze oltre una certa soglia (detta ***frequenza di taglio***, f_c). Ad esempio, nelle linee telefoniche la banda passante è 3 kHz (da 0 Hz a 3.000 Hz), ottenuta con ***filtri passa-basso***.

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.

Ci sono due teoremi fondamentali che caratterizzano i limiti per la trasmissione delle informazioni.

2.1.2) Teorema di Nyquist

Nyquist (1924) ha dimostrato che un segnale analogico di banda h (da 0 ad h Hz) può essere completamente ricostruito mediante una campionatura effettuata 2h volte al secondo. Dunque esso "convoglia" una quantità di informazione rappresentabile con un numero di bit pari a

2h*(logaritmo in base 2 del numero di possibili valori del segnale)

per ogni secondo.

Una conseguenza di tale teorema è che il massimo ***data rate*** (detto anche, con un termine non del tutto appropriato, ***velocità di trasmissione***) di un canale di comunicazione dotato di una banda passante da 0 Hz ad h Hz (passa-basso di banda h) che trasporta un segnale consistente di V livelli discreti è:

$$\textbf{massimo data rate (bit/sec.)} = 2h \log_2 V$$

Questo risultato implica che un segnale binario non va oltre i 6 kpbs su una linea di banda passante pari a 3 kHz. Come vedremo, i modem veloci sfruttano un segnale con un numero V di livelli piuttosto elevato per riuscire a trasmettere, su una linea funzionante ad x ***baud***, più di x bit/sec. (il termine baud indica la velocità di segnalazione di una linea, ossia quante volte al secondo essa è in grado di cambiare valore).

2.1.3) Teorema di Shannon

Il teorema di Nyquist è valido per canali totalmente privi di disturbi (il che purtroppo non è realistico). Per gli altri casi vale il teorema di Shannon (1948), che considera le caratteristiche di un canale rumoroso.

Prima di esporre il teorema è necessario chiarire il concetto di ***rapporto segnale/rumore*** (***signal to noise ratio***, ***S/N***): esso è il rapporto fra la potenza del segnale e quella del rumore. Si misura in ***decibel (dB)***, che crescono come $10\log_{10}$ (S / N). La tabella seguente riporta alcuni valori esemplificativi.

Rapporto S/N	Misura in Db
2	3
10	10
100	20
1.000	30

Il teorema di Shannon afferma che il massimo data rate di un canale rumoroso, con banda passante di h Hz e rapporto segnale/rumore pari a S/N, è data da:

$$\textbf{massimo data rate (bit/sec.)} = h \lg_2 (1 + S/N)$$

Si noti che in questo caso non conta più il numero V di livelli del segnale. Ciò perché, a causa del rumore, aumentarne il numero può renderli indistinguibili.

Ad esempio, su un canale con banda 3kHz e S/N = 30dB (tipici di una normale linea telefonica) si può arrivare al massimo a 30.000 bps.

In generale:

- più alto è il numero di bit/secondo che si vogliono trasmettere, più ampia diviene la banda passante che serve (T diminuisce);
- a parità di mezzo utilizzato, tanto più è corto il canale di trasmissione tanto più è alto il numero di bit/secondo raggiungibile (attenuazioni e sfasamenti restano accettabili);
- la trasmissione digitale è più critica di quella analogica (genera frequenze più alte), ma può essere più facilmente "rigenerata" lungo il percorso (è sufficiente distinguere fra pochi valori per ripristinare il segnale originario; nella trasmissione analogica ogni amplificazione introduce distorsione, che si somma a quella degli stadi precedenti).

2.2) Mezzi trasmissivi

2.2.1) Doppino intrecciato

E' il più anziano e diffuso. 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). E' usato, in

particolare, 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.

Due tipi di doppino sono importanti nella trasmissione dati:

- ***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*** (dal 1988): simile alla categoria 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.

2.2.2) 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 vi è una calza metallica.

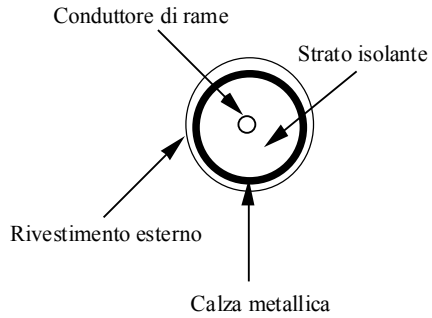


Figura 2-2: Sezione di un cavo coassiale

Era molto usato nel sistema telefonico per le tratte a lunga distanza, ma in tale ambito è ormai sostituito quasi ovunque dalla fibra ottica. Rimane in uso per la TV via cavo e in molte LAN.

Ci sono due tipi di cavo coassiale, per ragioni storiche più che tecniche.

Premessa: il termine ***baseband (banda base)*** significa che l'intera banda passante è usata per una singola trasmissione, di tipo digitale. Il termine ***broadband***, invece, nella telefonia indica qualunque trasmissione più ampia di 4 kHz, mentre 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.

- ***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.
- ***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. 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.

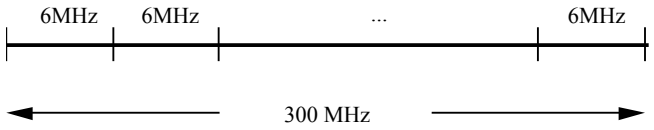


Figura 2-3: Multiplazione di più trasmissioni su un unico canale

Tecnicamente, il cavo broadband è inferiore a baseband per la trasmissione digitale, ma ha il vantaggio di essere già in opera in grandi quantità (TV via cavo). Dunque, attraverso essa, le compagnie pay-TV prevedibilmente entreranno in competizione con quelle telefoniche per l'offerta di servizi trasmissione dati.

2.2.3) Fibre ottiche

Sono uno dei mezzi più recenti, e stanno rivoluzionando il mondo delle telecomunicazioni. Sono fatte 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 contenitrice esterna.

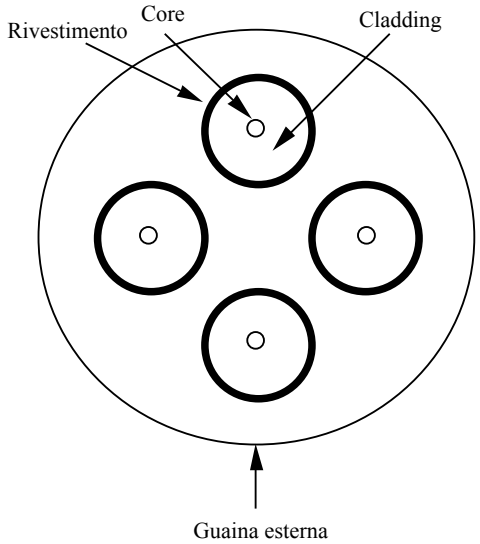


Figura 2-4: Sezione di un cavo contenente fibre ottiche

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.

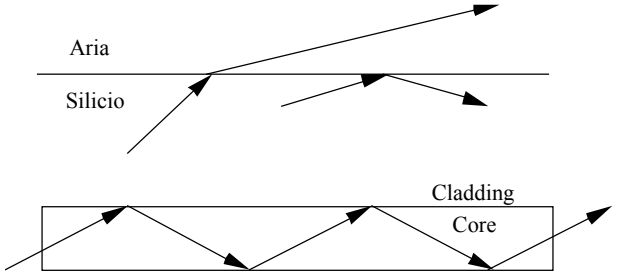


Figura 2-5: Deviazione del raggio luminoso

Le fibre ottiche sono di due tipi :

- ***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, come quello di un un capello;
- ***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 strepitose: con le correnti tecnologie è 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 fatte di un vetro speciale, molto trasparente (si vedrebbe il fondo del mare, se esso fosse di questo vetro), 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 :

- ***sorgente luminosa***: può essere un LED o un laser. Converte un segnale elettrico in impulsi luminosi;
- ***mezzo di trasmissione***: è la fibra ottica vera e propria;

- ***fotodiodo ricevitore***: 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 delle fibre ottiche rispetto al rame:
- leggerezza a parità 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 l'inserimento di intrusi per spiare il traffico.

Svantaggi delle fibre ottiche rispetto al rame:
- costo delle giunzioni;
- comunicazione unidirezionale (due fibre sono necessarie per una comunicazione two-way).

2.2.4) Trasmissione senza fili

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 (anche dalle gocce di pioggia!);
- 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.

2.3) Il sistema telefonico

Il sistema telefonico riveste un ruolo centrale per le comunicazioni a distanza fra computer, per vari motivi:
- sarebbe proibitivo in termini di costi connettere, con appositi cavi, apparecchiature distanti centinaia di km o più;
- è illegale, praticamente in tutti i paesi, stendere cavi sul suolo pubblico.

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

A titolo di esempio, si consideri la seguente tabella:

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

Ossia, vi sono 11 ordini di grandezza di differenza: la stessa differenza che c'è tra il costo del Progetto Apollo e quello di un biglietto dell'autobus.

2.3.1) Struttura generale

Agli albori della telefonia (il brevetto di Alexander Graham Bell è del 1876) i telefoni si vendevano a coppie, e gli acquirenti si preoccupavano di stendere il cavo (uno solo, con ritorno via terra) per collegarli. Le città divennero ben presto un groviglio di cavi, e quindi nacquero le società telefoniche (la prima fu la Bell) che aprirono ***uffici di commutazione*** nei quali un operatore smistava le chiamate fra i vari apparecchi. Questi non erano più collegati direttamente fra loro ma erano tutti connessi a un ufficio di commutazione.

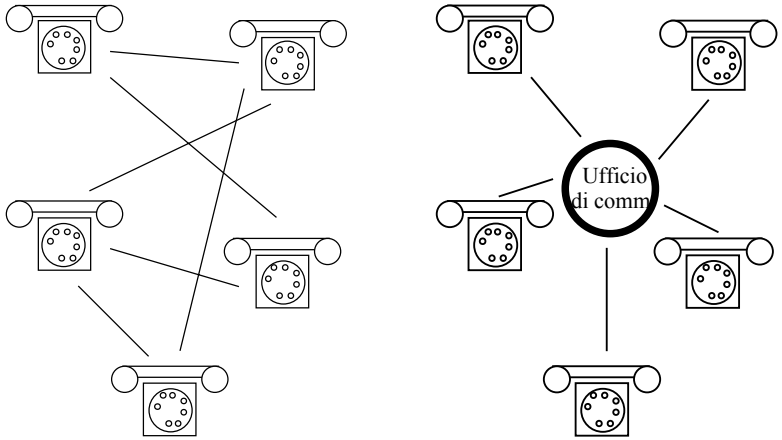


Figura 2-6: Nascita del sistema telefonico

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

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

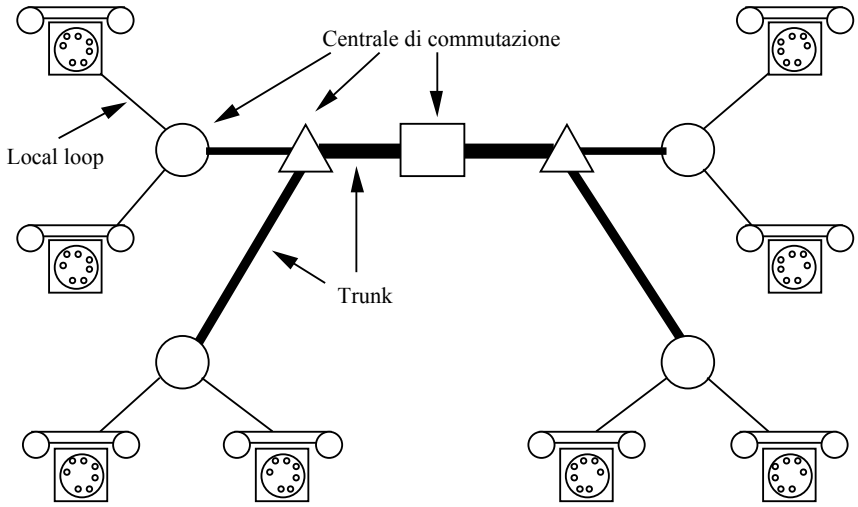


Figura 2-7: Struttura gerarchica del sistema telefonico

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

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

Per le altre connessioni (***trunk***) si usano molti altri mezzi:
- cavi coassiali;
- microonde;
- fibre ottiche, ormai molto diffuse.

Ormai quasi ovunque le centrali di commutazioni sono digitali e le linee che le collegano trasportano segnali digitali. I vantaggi principali sono i seguenti:
- è più facile ricostruire periodicamente il segnale senza introdurre errori (solo pochi valori);
- è più facile mescolare voce, dati, video e altri tipi di traffico;
- sono possibili data rate più alti usando le linee esistenti.

2.3.2) Il local loop

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

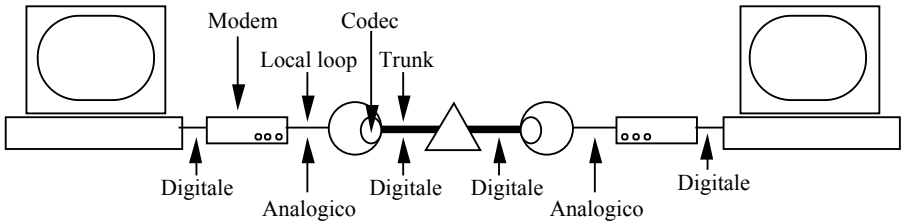


Figura 2-8: Trasmissione dati sul local loop

Ovviamente ciò non è l'ideale, ma bisogna accontentarsi.

Ricordiamo che:

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

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

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

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

Ad esempio, modulando opportunamente sia ampiezza che fase si possono rappresentare 16 valori diversi, quindi si possono ottenere 4 bit per baud. Dunque, su una linea a 2.400 baud (tipici del local loop), si può trasmettere alla velocità di 9.600 bps.

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

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

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

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

Due standard importanti per la compressione dei dati sono :
V.42 bis, emesso da ITU;
MNP 5, standard de facto (Microcom Network Protocol).

Infine, per consentire una trasmissione contemporanea nei due sensi (***full-duplex***), ci sono due tecniche :
- suddividere la banda in due sottobande, una per ogni direzione; così però si dimezza la velocità. Questa tecnica è tipica degli standard per le velocità più basse, ad esempio V.21 per 300 bps;
- cancellazione dell'eco: ogni modem sfrutta l'intera banda e cancella in ricezione gli effetti della propria trasmissione.

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

Ad esempio, lo standard ***RS-232-C*** ed il molto simile ***V.24*** del CCITT caratterizzano:
- specifiche meccaniche: connettore a 25 pin con tutte le dimensioni specificate;
- specifiche elettrice:
  - valore 1 corrisponde a un segnale minore di -3 Volt;
  - valore 0 corrisponde a un segnale maggiore di +4 Volt;
  - data rate fino a 20 Kbps, lunghezza fino a 20 metri;
- specifiche funzionali: quali circuiti (trasmit, receive, clear to send, ecc.) sono collegati a quali pin;
- specifiche procedurali: costituiscono il protocollo nel vero senso della parola, cioè le coppie azione / reazione fra DTE e DCE.

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

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

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

2.3.3) Trunk e multiplexing

Come abbiamo già visto, i ***trunk*** (cioè le connessioni fra una centrale e l'altra) sono realizzati con mezzi trasmissivi quali cavi coassiali, fibre ottiche, che offrono una banda passante molto ampia ed un bassissimo tasso d'errore.

Su essi devono poter essere convogliate contemporaneamente molte conversazioni indipendenti (e/o molte trasmissioni di dati).

C'è quindi la necessità di mettere in piedi un meccanismo di ***multiplazione*** (***multiplexing***). Ci sono due schemi principali:

- ***Frequency Division Multiplexing (FDM)***;
- ***Time Division Multiplexing (TDM)***.

Frequency division multiplexing
Lo spettro di frequenza disponibile è suddiviso in varie bande più piccole, e ogni utente ha l'esclusivo uso di una di esse.

Ad esempio, più canali telefonici (ciascuno avente la banda di 3 kHz) vengono multiplati allocando a ciascuno di essi una banda di 4 kHz, per avere un margine di 500 Hz di sicurezza su ciascun lato della banda. Ogni canale telefonico viene innalzato in frequenza fino ad occupare la banda assegnatagli.

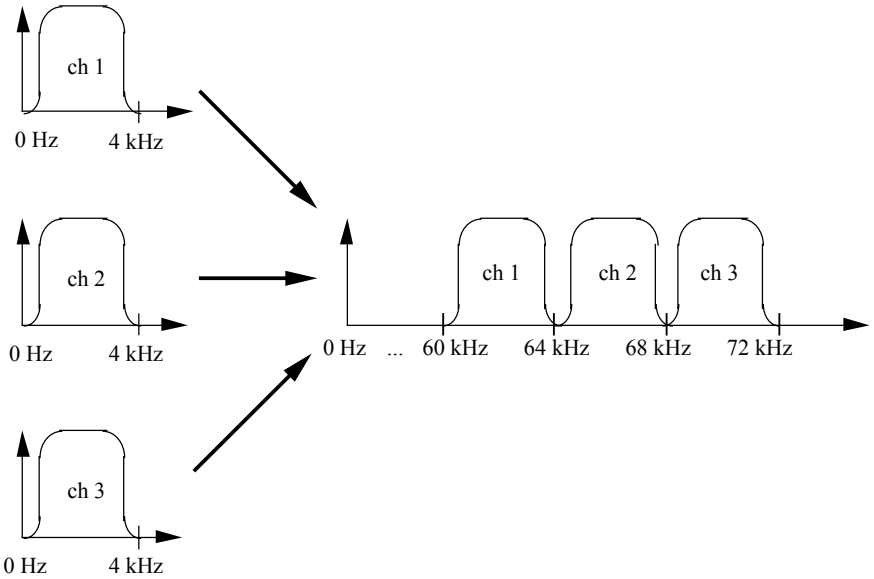


Figura 2-9: FDM

Uno standard CCITT prevede il multiplexing di 12 canali da 4 kHz nella banda 60-108 kHz. Ciò costituisce un ***group***. Molte società telefoniche offrono un servizio di trasmissione dati, a velocità tra i 48 e i 56 Kbps, basato su un gruppo.

Cinque group (60 canali) formano un ***supergroup***, cinque supergroup (300 canali) formano un un ***mastergroup***. Sono definiti gli standard fino a 230.000 canali.

Una variante di FDM per le fibre ottiche è il ***Wavelength Division Multiplexing*** (***WDM***). Si lavora in base alle lunghezze d'onda, inversamente proporzionali alle frequenze. Concettualmente è analogo a FDM. Dal punto di vista realizzativo si varia la lunghezza d'onda del raggio luminoso. Ciò si fa con dei sintonizzatori ottici, basati sugli ***interferometri Farbry-Perot o Mach-Zehnder***.

Time division multiplexing
FDM è adatto alla gestione di segnali analogici e richiede circuiteria analogica. Di conseguenza è poco adatto alla gestione di dati digitali quali quelli prodotti dai computer.

TDM invece è ideale per la gestione di dati in forma digitale. L'idea è semplice: i bit provenienti da diverse connessioni vengono prelevati a turno da ciascuna di esse ed inviati su un'unica connessione ad alta velocità :

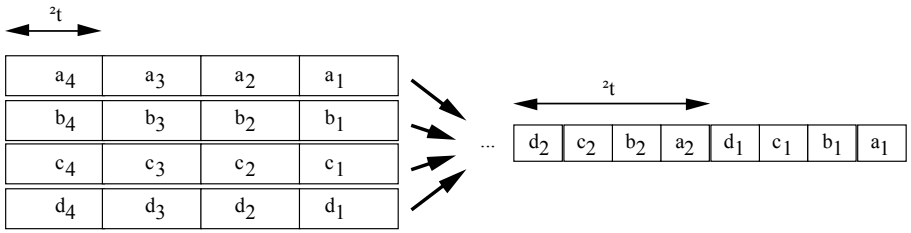


Figura 2-10: TDM

Poiché i local loop trasportano un segnale analogico, esso deve essere trasformato in digitale prima di essere combinato in TDM con gli altri. Questa operazione, come abbiamo già visto, viene fatta in centrale da un codec (coder-decoder). Esso effettua 8.000 campionamenti al secondo del segnale analogico (1 campione ogni 125 microsecondi: ciò è sufficiente, secondo il teorema di Nyquist, per un segnale caratterizzato da una banda di 4 kHz) e produce altrettanti valori a 7 bit (in USA) o 8 bit (in Europa).

Questa tecnica si chiama ***PCM (Pulse Code Modulation)***, e forma il cuore di tutti i sistemi telefonici moderni. La conseguenza è che il ritmo di funzionamento di tutti i sistemi telefonici è basato su un intervallo di tempo fondamentale di 125 microsecondi.

Si noti che il segnale vocale digitalizzato richiede 8 * 8.000 bps e cioé 64 Kpbs.

Non esiste uno standard internazionale per il TDM:

- in America ed in Giappone si è diffuso il ***T1 Carrier***, costituito da 24 canali; ogni canale trasferisce 7 bit di dati (per la voce) e 1 bit di controllo ogni 125 microsecondi, per cui si hanno 56.000 bps di dati e 8.000 bps di controllo. Un ***frame T1*** consiste di 192 bit (più uno di framing) ogni 125 microsecondi, per cui T1 trasmette alla velocità di 193 * 8.000 bps ossia di 1.544 Mbps;
- in Europa è invece diffuso ***E1 Carrier*** (conforme a raccomandazione CCITT), costituito da 32 canali con valori ad 8 bit (cioé 32 * 8 * 8.000 bps ossia 2.048 Mbps), di cui 30 per i dati e 2 per il controllo.

Ovviamente TDM può essere riapplicato, per cui si hanno le seguenti gerarchie (dette ***plesiocrone***, dal greco ***plesio*** che significa vicino: non è detto che il clock di due flussi nominalmente uguali sia perfettamente identico).

Carrier	Caratteristiche	Velocità
T2	4 canali T1	6.312 Mbps
T3	6 canali T2	44.736 Mbps
T4	7 canali T3	274.176 Mbps
E2	4 canali E1	8.848 Mbps
E3	4 canali E2	34.304 Mbps
E4	4 canali E3	139.264 Mbps
E5	4 canali E4	565.148 Mbps

Tutto ciò comporta la necessità di costose apparecchiature di conversione ai confini fra un sistema e l'altro.

2.3.4) SONET/SDH

Per superare tali difficoltà, a metà degli anni '80 è stata introdotta dal CCITT la ***gerarchia SDH*** (***Synchronous Digital Hierarchy***), unificata a livello mondiale. In USA si chiama ***SONET*** (***Synchronous Optical NETwork***) ed ha una velocità (51.84 Mbps) non presente in SDH.

I suoi scopi principali sono:

- interoperabilità dei vari ***Carrier*** (le aziende telefoniche);
- unificazione dei sistemi in esercizio in USA, Europa e Giappone;
- capacità di trasportare frame T1 e superiori ed E1 e superiori.

SONET/SDH è basato su un tradizionale TDM, ed è un sistema sincrono, controllato da un clock principale molto preciso (il tasso d'errore è tipicamente inferiore ad 1 su 10^9). Il multiplexing è fatto byte per byte, ciclicamente. La gerarchia è ***sincrona***, cioè si garantisce che tutti i clock che governano i vari flussi sono assolutamente identici.

Un sistema SONET consiste di un insieme di vari elementi, connessi da fibre ottiche:

- unità di commutazione;
- multiplexer;
- ripetitori.

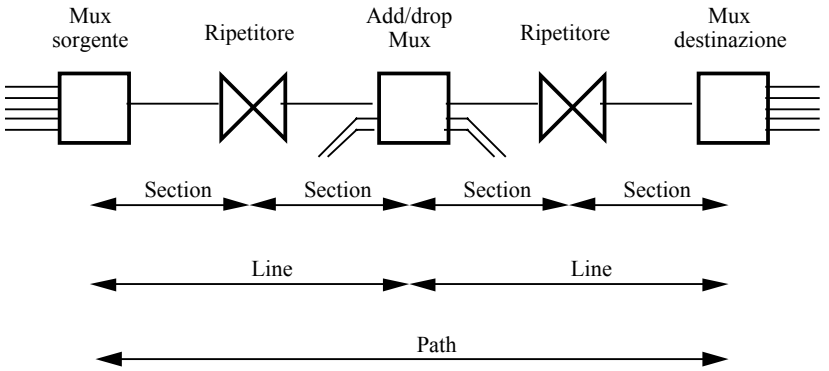


Figura 2-11: Componemti di SONET

L'unità base è un blocco di 810 byte emesso ogni 125 microsecondi. Questa unità definisce il canale fondamentale di SONET, che si chiama ***STS-1*** (***Synchronous Transport Signal - 1***). Esso ha una velocità di 810 * 8 * 8.000 bps, cioé 51.84 Mbps.

Vari livelli della gerarchia SONET/SDH sono stati definiti :

SONET	SDH	Mbps
STS-1		51.84
STS-3	STM-1	155.52
STS-9	STM-3	466.52
STS-12	STM-4	622.08
STS-18	STM-6	933.12
STS-24	STM-8	1244.16
STS-36	STM-12	1866.24
STS-45	STM-16	2488.32

ATM ha la velocità di 155 Mbps perché si pensa di trasportare le celle ATM su ***OC-3***, l'analogo su fibra ottica di STS-3 o STM-1.

Il livello fisico di SONET è diviso in quattro sottolivelli:

- sottolivello ***Photonic***: specifica caratteristiche della fibra e della luce da usare;
- sottolivello ***Section***: si occupa del multiplexing / demultiplexing di flussi multipli; ha a che fare, ad esempio, sul come e quanti flussi combinare;
- sottolivello ***Line***: si occupa di gestire un collegamento punto a punto in fibra ottica: genera un frame da una parte e lo elabora dall'altra;
- sottolivello ***Path***: si occupa degli aspetti end to end della comunicazione.

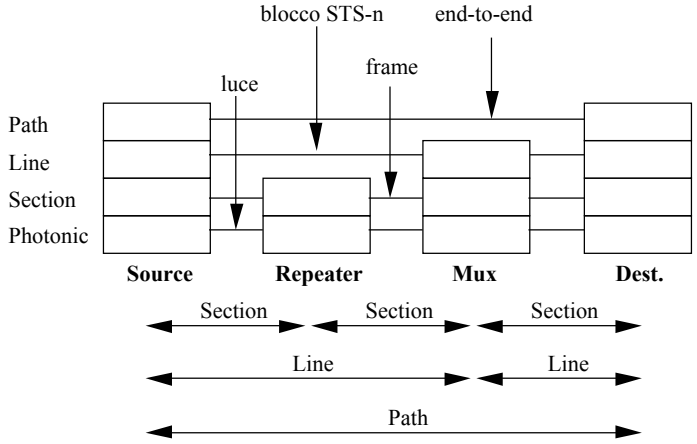


Figura 2-12: Livelli di SONET

2.3.5) Commutazione

Il sistema telefonico è ***commutato***, cioè le linee in ingresso alla centrale telefonica vengono commutate, ossia vengono connesse di volta in volta a differenti linee in uscita sulla base delle richieste di connessione degli utenti.

Ci sono due tipi principali di commutazione:

- ***Commutazione di circuito (circuit switching)***: Quando si effettua una telefonata, il sistema stabilisce una connessione fisica dedicata fra il chiamante ed il chiamato. Il traffico quindi fluisce su questa connessione. Naturalmente, la connessione in realtà è costituita in vari modi, prima sul local loop e poi come successione di canali a 64K ricavati all'interno di flussi T1, o T3, o STS-M. In questo caso c'è un certo tempo che

passa per il ***connection setup*** (anche alcuni secondi), dopo di che il segnale viaggia senza ulteriori ritardi. E' difficile che si presentino situazioni di congestione.

- ***Commutazione di pacchetto (packet switching)***: Non si stabilisce alcuna connessione fisica fra sorgente e destinazione. I dati da inviare sono spezzettati in pacchetti, che vengono inviati indipendentemente gli uni dagli altri, possono percorrere strade diverse e vengono riordinati solo a destinazione. Fra i vantaggi: non si riserva banda in anticipo (che verrebbe poi sprecata in assenza di traffico) e non c'è il ritardo per il connection setup. Uno svantaggio risiede nel fatto che la congestione può insorgere in ogni momento.

Attualmente i sistemi telefonici, ottimizzati per la fonia, sono circuit-switched. Viceversa, le reti di elaborazione sono comunemente packet-switched, pur offrendo in generale anche servizi di tipo connesso.

2.3.6) Dispositivi di commutazione

Il tipo più semplice è il ***crossbar***, dotato di n ingressi, n uscite ed n^2 punti di incrocio, realizzati ciascuno con un dispositivo a semiconduttori detto ***switch*** che ha due ingressi i_1 ed i_2 e due uscite u_1 ed u_2. Il dispositivo può avere due stati:

- i_1 è connesso a u_1 e i_2 è connesso a u_2;
- i_1 è connesso a u_2 e i_2 è connesso a u_1.

Un singolo switch commuta in pochi microsecondi e stabilisce una connessione elettrica diretta fra un ingresso ed una uscita. A seconda di come vengono pilotati gli switch, un crossbar è in grado di instradare contemporaneamente ogni linea in ingresso su una opportuna linea in uscita.

Uno svantaggio è che ci vogliono n^2 switch, ossia un numero considerevole. Per superare questo problema sono stati introdotti gli ***switch multilivello (multistage switch)***, basati sul principio di utilizzare molti crossbar di piccole dimensioni (ad es. 2 * 2), organizzati in gruppi su più livelli successivi. Una possibilità è la ***rete di Banyan*** .

Tali dispositivi richiedono un numero minore di switch ($O(n\lg_2 n)$ nel caso della rete Banyan), ma purtroppo presentano lo svantaggio di essere esposti a conflitti nell'instradamento delle informazioni. Tali conflitti richiedono meccanismi di buffering, che aumentano il costo del dispositivo.

2.3.7) Servizi per trasmissione dati

Per varie ragioni, fra cui gli elevati costi di cablaggio e l'esistenza di leggi nazionali che regolamentano il settore delle telecomunicazioni, è attualmente impossibile per una organizzazione realizzare una rete geografica provvedendo in proprio alla stesura materiale dei cavi, ove questi debbano passare su suolo pubblico. Ci si deve invece

rivolgere ad una società telefonica (in futuro anche ad altri, quali ad esempio Cable TV) per la realizzazione della subnet di comunicazione.

Esistono in proposito diverse possibilità, fra le quali:

- ***affitto di linee dedicate***: si possono stipulare contratti per l'affitto di linee telefoniche dedicate (cioè perennemente attive e di esclusivo uso del cliente) che poi costituiscono i collegamenti fra i nodi della subnet. Tali linee possono avere velocità varie (tipicamente comprese fra 2.400 bps e 2 Mbps). L'azienda telefonica implementa tali linee riservando al cliente un circuito permanente di opportuna velocità, e (se necessario) stendendo cavi di alta qualità dalle centrali più vicine alle sedi utente, per superare i limiti già visti che affliggono il local loop. Questa soluzione è molto costosa: una linea a 2 Mbps costa diverse decine di milioni l'anno. L'utente, ricevute le linee dedicate, configura la rete geografica utilizzando l'architettura e le attrezzature di rete di suo gradimento.
- ***affitto di servizi trasmissione dati***: una possibilità economicamente più vantaggiosa è quella di ricorrere ai servizi di trasmissione dati offerti (ormai da vari anni) da tutte le società telefoniche. L'idea è di realizzare le connessioni fra i nodi della propria subnet non più attraverso il ricorso a linee dedicate, ma "simulandole" attraverso l'affitto di tali servizi di trasmissione dati. I più importanti, per i quali esistono degli standard, sono:
  - ***Frame Relay***;
  - ***ISDN (Integrated Services Digital Network)***;
  - ***SMDS (Switched Multimegabit Data Service)***;
  - ***ATM (Asynchronous Transfer Mode)***.

3) Il livello due (Data Link)

Questo livello ha il compito di offrire una comunicazione affidabile ed efficiente a due macchine ***adiacenti***, cioé connesse fisicamente da un canale di comunicazione (ad es.: cavo coassiale, doppino, linea telefonica).

Esso si comporta come un "tubo digitale", cioé i bit partono e arrivano nello stesso ordine. La cosa non è banale come sembra, perché:

- ci sono errori e disturbi occasionali;
- il canale ha un data rate finito;
- c'è un ritardo nella propagazione.

I protocolli usati per la comunicazione devono tenere in conto tutto questo.

Competenze del livello data link

Questo livello ha le seguenti incombenze principali:

- 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 Network

Il servizio principale è trasferire dati dal livello network della macchina di origine al livello network della macchina di destinazione.

Come sappiamo, la trasmissione reale passa attraverso il livello fisico, ma noi ragioniamo in termini di dialogo fra due processi a livello data link che usano un protocollo di livello data link.

I servizi offerti possono essere diversi. I più comuni sono:

- ***connectionless non confermato***
  - si mandano frame indipendenti;
  - i frame non vengono confermati;
  - non si stabilisce una connessione;
  - i frame persi non si recuperano (in questo livello);
  - è appropriato per:
    - canali con tasso d'errore molto basso;
    - traffico real-time (es. voce);
    - le LAN, nelle quali, in effetti, è molto comune.
- ***connectionless confermato***
  - come sopra, però i frame vengono confermati;
  - se la conferma non arriva, il mittente può rispedire il frame;
  - è utile su canali non affidabili (ad es. in sistemi wireless);

    - nota 1: la perdita di un ack può causare la trasmissione di più copie dello stesso frame;
    - nota 2: avere la conferma a questo livello è un'ottimizzazione, mai una necessità. Infatti la conferma può sempre essere fatta a livello superiore, ma con linee disturbate ciò può essere gravoso.
- ***connection oriented confermato***
    - è il servizio più sofisticato;
    - prevede tre fasi (apertura conn. / invio dati / chiusura conn.);
    - garantisce che ogni frame sia ricevuto esattamente una volta e nell'ordine giusto;
    - fornisce al livello network un flusso di bit affidabile.

Vediamo ora un tipico esempio di funzionamento. Consideriamo un router con alcune linee in ingresso ed alcune in uscita. Ricordiamo che il routing avviene a livello tre (network), quindi il router gestisce i livelli uno, due e tre.

1. Quando al router arrivano dei bit da una linea fisica, l'hardware apposito se ne accorge (siamo a livello 1) e li passa al corrispondente SW / HW di livello due.
2. Il SW / HW di livello due (data link), che in genere è contenuto in un chip sulla ***scheda (o adattatore) di rete*** (tipico esempio è una scheda Ethernet) fa i controlli opportuni:
    - framing;
    - controllo errori di trasmissione;
    - controllo numero di frame (se necessario).
3. Se tutto è OK, il SW / HW di livello due genera un interrupt alla cpu, che chiama in causa il SW di livello tre (network):
    - 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.
4. 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. Però:
- il flusso non è esente da errori;
- possono arrivare più o meno bit di quanti sono stati inviati.

E' compito del livello due rilevare, e se possibile correggere, tali errori. L'approccio usuale del livello due è il seguente.
- In trasmissione:
    - spezza il flusso di bit che arriva dal livello tre in una serie di frame;
    - calcola un'apposita funzione (***checksum***) per ciascun frame;
    - inserisce il checksum nel frame;
    - consegna il frame al livello uno, il quale lo spedisce come sequenza di bit.
- In ricezione:

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

Dunque, la prima cosa (più difficile di quanto sembri) di cui occuparsi, è come delimitare un singolo frame.

3.1) Framing

E' rischioso usare lo spazio temporale che intercorre tra i frame per delimitarli, perché nelle reti in genere non ci sono garanzie di temporizzazione. Quindi, si devono usare degli appositi marcatori per designare l'inizio e la fine dei frame.

Ci sono vari metodi:
- ***conteggio dei caratteri***;
- ***caratteri di inizio e fine***, con ***character stuffing***;
- ***bit pattern di inizio e fine***, con ***bit stuffing***;
- ***violazioni della codifica*** dei bit usata nel livello fisico.

3.1.1) Conteggio

Si utilizza un campo nell'header, per indicare quanti caratteri ci sono del frame

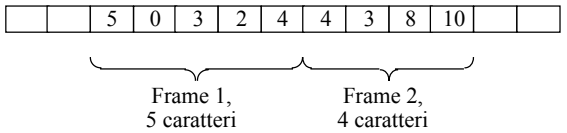


Figura 3-1: Il metodo del conteggio

Se durante la trasmissione si rovina il campo del frame che contiene il conteggio, diventa praticamente impossibile ritrovare l'inizio del prossimo frame e di conseguenza anche quello dei successivi. A causa della sua scarsa affidabilità questo metodo è usato ormai pochissimo.

3.1.2) Caratteri di inizio e fine

Ogni frame inizia e finisce con una particolare la sequenza di caratteri ASCII.

Una scelta diffusa è la seguente:

- inizio frame:
  - ***DLE*** (Data Link Escape), ***STX*** (Start of TeXt)
- fine frame:
  - ***DLE, ETX*** (End of TeXt)

In questo modo, se la destinazione perde traccia dei confini di un frame la riacquista all'arrivo della prossima coppia DLE, STX e DLE, ETX.

Esiste però un problema: nella trasmissione di dati binari, il byte corrispondente alla codifica di DLE può apparire dentro il frame, imbrogliando le cose.

Per evitare questa evenienza, il livello due sorgente aggiunge davanti a tale byte un altro DLE, per cui in arrivo solo i singoli DLE segnano i confini dei frame. Naturalmente, il livello due di destinazione rimuove i DLE aggiunti dentro ai dati prima di consegnarli al livello tre. Questa tecnica si chiama ***character stuffing***.

3.1.3) Bit pattern di inizio e fine

La tecnica precedente è legata alla codifica ASCII ad 8 bit, e non va bene per codifiche più moderne (es. UNICODE). Una tecnica analoga e più recente permette di avere un numero qualunque di bit dentro il frame.

Ogni frame inizia e finisce con una specifica sequenza di bit (bit pattern), ad es.:

```
01111110
```

chiamata un ***flag byte***.

Ovviamente esiste un problema analogo al caso precedente: tale sequenza di bit può infatti apparire all'interno dei dati che devono essere trasmessi.

In questo caso:

- in trasmissione: ogni volta che il livello due incontra nei dati da trasmettere 5 bit consecutivi uguali a 1 inserisce uno zero aggiuntivo;
- in ricezione: quando nei dati ricevuti compaiono 5 bit uguali a uno, si rimuove lo zero che li segue.

Dunque, il flag byte può apparire solo all'inizio ed alla fine dei frame. Questa tecnica va sotto il nome di ***bit stuffing***.

3.1.4) Violazioni della codifica

In molte reti (sopratutto LAN) si codificano i bit al livello fisico con una certa ridondanza. Ad esempio:

- il valore 1 di un bit di dati è codificato con la coppia ***high/low*** di bit fisici;
- 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. Codifiche come questa (***Manchester encoding***, usata in IEEE 802.3) hanno lo scopo di consentire una facile determinazione dei confini di un bit dati, poiché esso ha sempre una trasmissione nel mezzo (però ci vuole una bandwidth doppia per trasmetterlo a parità di velocità).

Le coppie high / high e low / low possono quindi essere usate per delimitare i frame.

3.2) Rilevamento e correzione errori

Ci sono molti fattori che possono provocare errori, sopratutto sul local loop e nelle trasmissioni wireless. Viceversa, essi sono piuttosto rari nei mezzi più moderni quali le fibre ottiche.

Gli errori sono dovuti in generale a:

- rumore di fondo;
- disturbi (ad es. fulmini) improvvisi;
- interferenze (ad es. motori elettrici).

Ci sono due approcci al trattamento degli errori:

- includere abbastanza informazione aggiuntiva in modo da poter ricostruire il messaggio originario (***correzione dell'errore***);
- includere meno informazione aggiuntiva, in modo da accorgersi che c'è stato un errore, senza per questo essere in grado di correggerlo (***rilevazione dell'errore***).

3.2.1) Codici per la correzione degli errori

Normalmente, un frame (a parte i delimitatori) consiste di

$$n = m + r$$

bit, dove:

- m bit costituiscono il messaggio vero e proprio;
- r bit sono ridondanti, e sono detti ***reduntant bit*** (o ***check bit***).

Una sequenza di n bit fatta in tal modo si dice ***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 la ***distanza di Hamming*** delle due codeword (Hamming, 1956). Se due codeword hanno una distanza di Hamming uguale a d, ci vogliono 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.

Ora, le capacità di rilevare o correggere gli errori dipendono 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.

Vediamo ora, come esempio, 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 even parity)

```
<-- m -->  r

1011 0101  1
1000 0111  0
```

Questo codice, detto ***codice di parità*** (***parity code***) 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.

Come secondo esempio, consideriamo il codice costituito dalle seguenti codeword:

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

Esso ha distanza 5, per cui corregge due errori (cifre sottolineate). Infatti, se arriva

```
00000 001111
```

si può risalire correttamente al codeword più vicino, che è

```
00000 11111
```

Però, se ci sono tre errori ed arriva

```
00000 00111
```

siamo nei guai, perché esso verrà intrepretato 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 $\lg_2 m$ bit. Esiste un codice (***codice di Hamming***) che raggiunge questo limite teorico.

Ora, c'è una semplice tecnica con la quale tale codice può essere impiegato per correggere gruppi contigui di errori (detti ***burst di errori***) di lunghezza massima prefissata.

Supponiamo di voler correggere burst di lunghezza k:

- si accumulano k codeword, riga per riga;
- i codeword si trasmettono per colonne;
- quando arrivano si riassemblano per righe. Un burst di k errori comporta un singolo errore in ciascun codeword, che quindi può essere completamente ricostruito.

3.2.2) Codici per il rilevamento degli errori

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 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.

Infatti, ad esempio, supponiamo di avere:
- canale con errori isolati e probabilità di errore uguale a 10^{-6} per bit;
- blocchi dati di 1.000 bit.

Per correggere errori singoli su un blocco di 1.000 bit, ci vogliono 10 bit, per cui un Megabit richiede 10.000 check bit.

Viceversa, per rilevare l'errore in un blocco, basta un bit (con parity code). Ora, con 10^{-6} di tasso d'errore, solo un blocco su 1.000 è sbagliato e quindi deve essere ritrasmesso. Di conseguenza, per ogni Megabit si devono rispedire 1.001 bit (un blocco più il parity bit).

Dunque, l'overhead totale su un Megabit è:
- 1.000 bit per parity bit su 1.000 blocchi,
- 1.001 bit per il blocco ritrasmesso,
- per un totale di 2.000 bit contro i 10.000 del caso precedente.

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.

Esiste però un altro metodo che nella pratica viene usato quasi sempre, il ***Cyclic Redundancy Code*** (***CRC***), noto 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 sempio, la stringa di bit 1101 corrisponde al polinomio $x^3 + x^2 + x^0$.

L'aritmetica polinomiale è fatta modulo 2, secondo le regole della teoria algebrica dei campi. In particolare:
- addizione e sottrazione sono equivalenti all'or esclusivo (non c'è riporto o prestito);
- la divisione è come in binario, calcolata attraverso la sottrazione modulo 2.

Il mittente ed il destinatario si mettono d'accordo 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 il ricevitore riceve il frame più il checksum, divide il tutto per G(x). Se il risultato è zero è tutto OK, altrimenti c'è stato un errore.

Il calcolo del checksum si effettua come segue:
1. Appendere r bit a destra del frame, che quindi ha m+r bit, e corrisponde ad $x^rM(x)$;
2. Dividere $x^rM(x)$ per G(x);
3. Sottrarre ad $x^rM(x)$ 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;
- rileva tutti i burst di errori di lunghezza <= r.

Tra i polinomi sono diventati standard internazionali:

- ***CRC-12***: $x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$;
- ***CRC-16***: $x^{16} + x^{15} + x^2 + 1$;
- ***CRC-CCITT***: $x^{16} + x^{12} + x^5 + 1$;

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.

Questi risultati valgono sotto l'ipotesi che gli m bit del messagio 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 si creda.

3.3) Gestione sequenza di trasmissione e flusso

Dopo aver trovato il modo di delineare l'inizio e la fine dei frame e di gestire gli errori di trasmissione, bisogna trovare la maniera di informare opportunamente il mittente se i frame spediti sono anche arrivati, e senza errori, dall'altra parte:
- servizi connectionless non confermati: non c'è bisogno di alcuna conferma;
- servizi connectionless confermati: sono arrivati tutti e senza errori?
- servizi connection oriented confermati: sono arrivati tutti, senza errori e nell'ordine giusto?

Per saperlo si introduce il concetto di ***acknowledgement***, che è un messaggio inviato dal destinatario al mittente per informarlo che:

- il frame è arrivato correttamente (***positive ack***);
- il frame è errato (***negative ack***).

Nel seguito, ove necessario per evitare ambiguità, col termine ***frame dati*** indicheremo un frame che trasporta informazioni generate nel colloquio fra le peer entity; col termine ***frame di ack*** indicheremo un frame il cui solo scopo è trasportare un acknowledgement.

Introducendo l'uso degli ack sorgono però alcuni problemi, fra cui:

Problema 1	un frame può anche sparire del tutto, per cui il mittente rimane bloccato in attesa di un ack che non arriverà mai;
Soluzione 1	il mittente stabilisce un time-out per la ricezione dell'ack. Se questo non arriva in tempo, il frame si ritrasmette.

Problema 2	se sparisce l'ack, il destinatario può trovarsi due (o più) copie dello stesso frame;
Soluzione 2	il mittente inserisce un numero di sequenza all'interno di ogni frame dati.

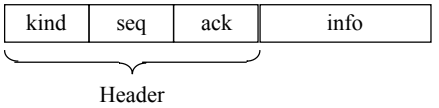
Un altro aspetto importante, che richiede anch'esso un feedback dal destinatario al mittente, è il controllo del flusso, onde impedire che il mittente spedisca dati più velocemente di quanto il destinatario sia in grado di gestirli. Spesso si usano meccanismi basati sull'esplicita autorizzazione, data da parte del destinatario al mittente, di inviare un ben preciso numero di frame.

Vedremo ora le caratteristiche di diversi protocolli, di complessità crescente, per il livello data link.

Prima però dobbiamo fare alcune assunzioni:

- nei livelli fisico, data link e network, ci sono processi (HW o SW) indipendenti, che comunicano fra loro, ad esempio scambiandosi messaggi;
- quando 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 è tutto OK consegna il pacchetto, e solo quello, al livello network;
    - altrimenti intrapende quanto è necessario fare per recuperare la situazione e non consegna il pacchetto al livello network.

Nei nostri esempi, la struttura di un frame è questa:



I campi del frame hanno le seguenti funzioni:

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
info	contiene un pacchetto di livello network completo (comprendente quindi le informazioni di controllo di tale livello).

Naturalmente, i vari algoritmi useranno via via un numero maggiore di tali informazioni.

3.3.1) Protocollo 1: Heaven

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

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

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

Mittente (loop infinito):

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

Destinatario (loop infinito):

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

3.3.2) Protocollo 2: Simplex Stop and Wait

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

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

Mittente (loop infinito):

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


Destinatario (loop infinito):

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

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

3.3.3) Protocollo 3: simplex per canale rumoroso

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

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

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

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

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

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

Notiamo che basta un bit per il numero di sequenza, poiché l'unica ambiguità in ricezione è tra un frame ed il suo immediato successore (siano ancora in stop and wait): infatti, fino

a che un frame non viene confermato, è sempre lui ad essere ritrasmesso, altrimenti è il suo successore.

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

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

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

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

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

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

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

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


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

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

In sintesi:

- Il mittente etichetta i frame dati con la sequenza `...0,1,0,1...`, ma passa all'etichetta e frame successivi solo quando arriva un ack; finché ciò non succede, 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...`.

I disegni 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:

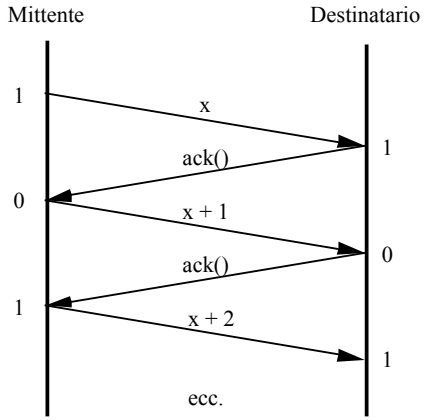


Figura 3-2: Normale funzionamento

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

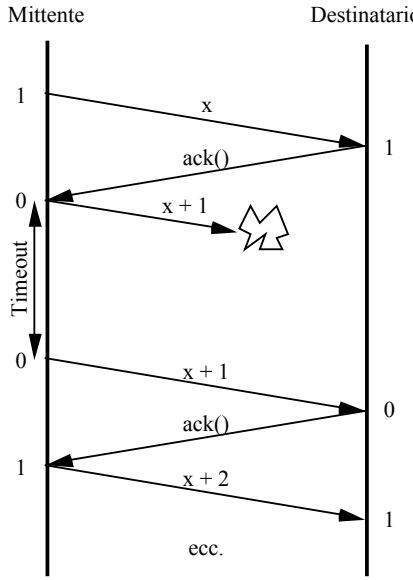


Figura 3-3: Perdita (o danneggiamento) di un frame

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

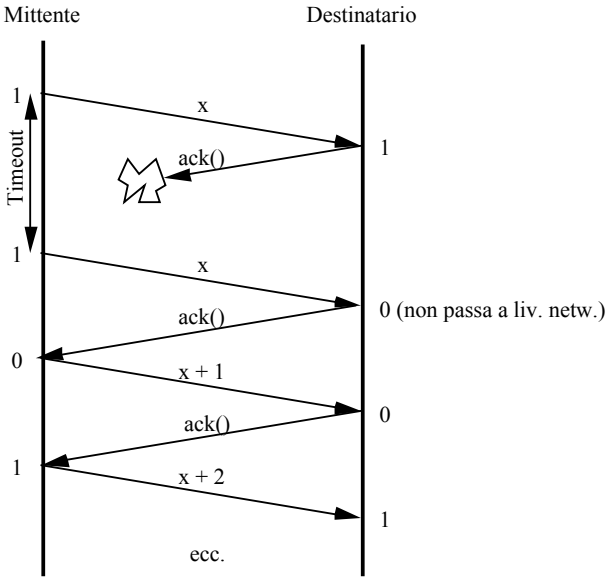


Figura 3-4: Perdita (o danneggiamento) di un ack

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

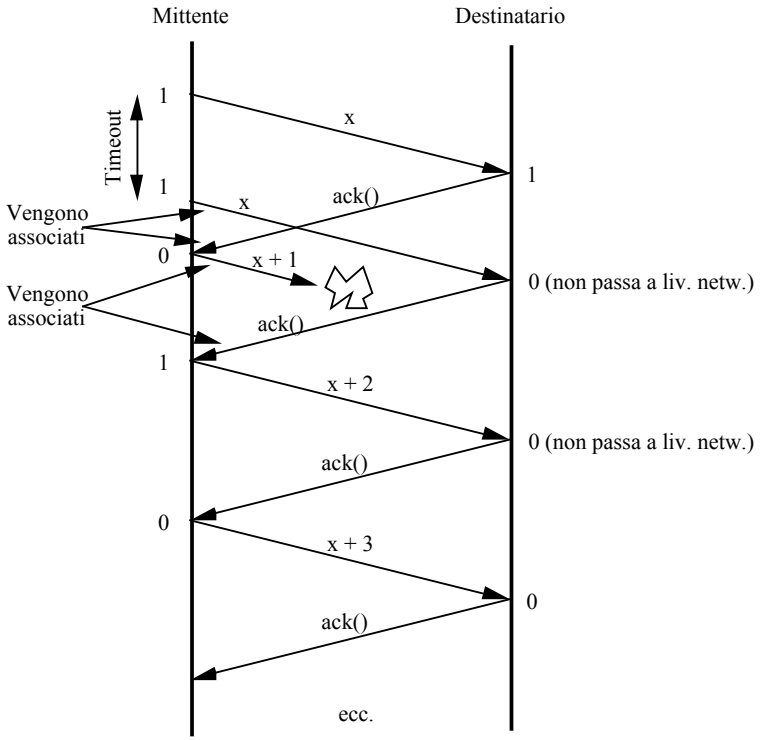


Figura 3-5: Timeout troppo ridotto

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:

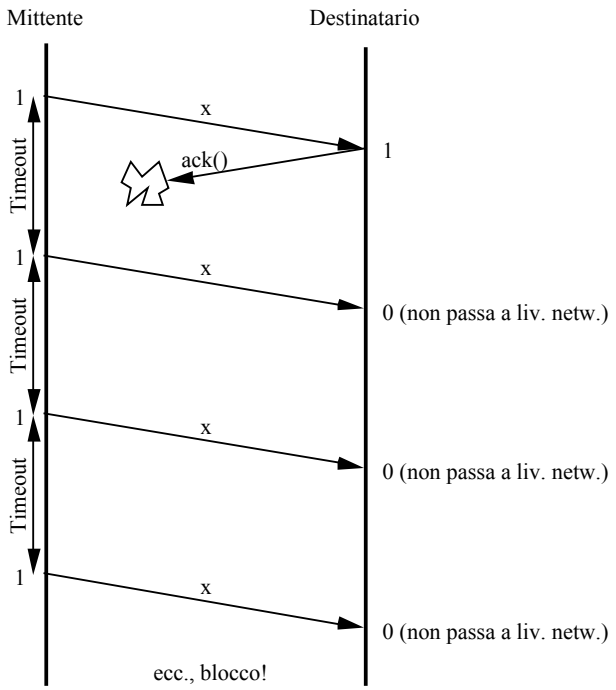


Figura 3-6: Ack inviato solo per frame non duplicati

3.3.4) Protocolli a finestra scorrevole

Nei casi precedenti i frame dati viaggiano in una sola direzione e i frame di ack nella direzione contraria, quindi esistono dei frame che viaggiano in entrambe le direzioni.

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

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

- nella direzione da A a B viaggiano i frame dati inviati da A a B e i frame di ack inviati da A a B (in risposta ai frame dati inviati da B ad A);

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

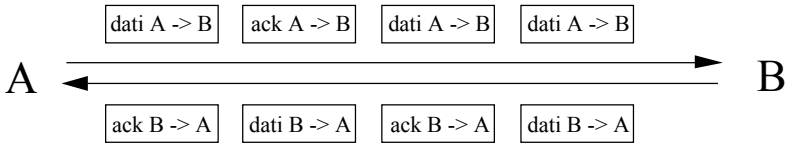


Figura 3-7: Comunicazione dati bidirezionale su unico canale

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

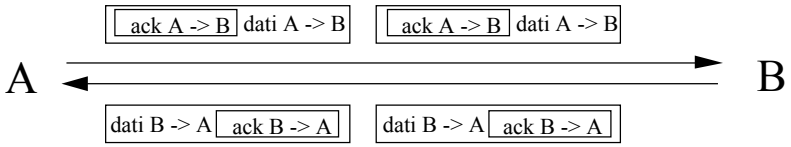


Figura 3-8: Piggybacking

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;
- uso di cpu.

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

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

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

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

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

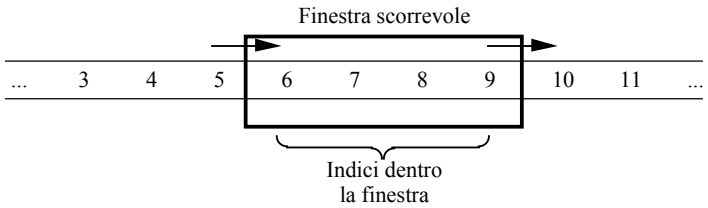


Figura 3-9: Finestra scorrevole sugli indici dei frame

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

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

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

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

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

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

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

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 sta quasi sempre fermi aspettando l'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.

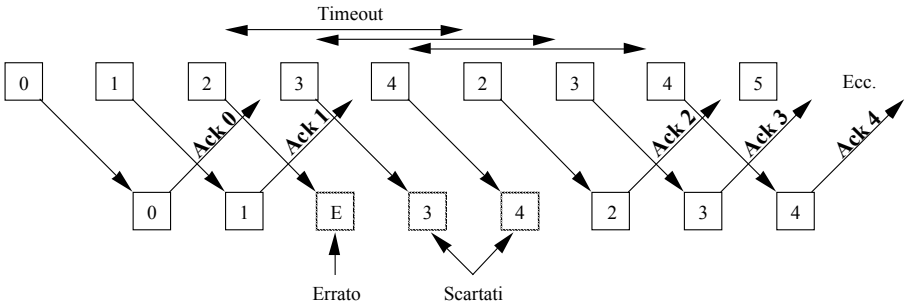


Figura 3-10: Funzionamento del protocollo go-back-n

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 bene, 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.

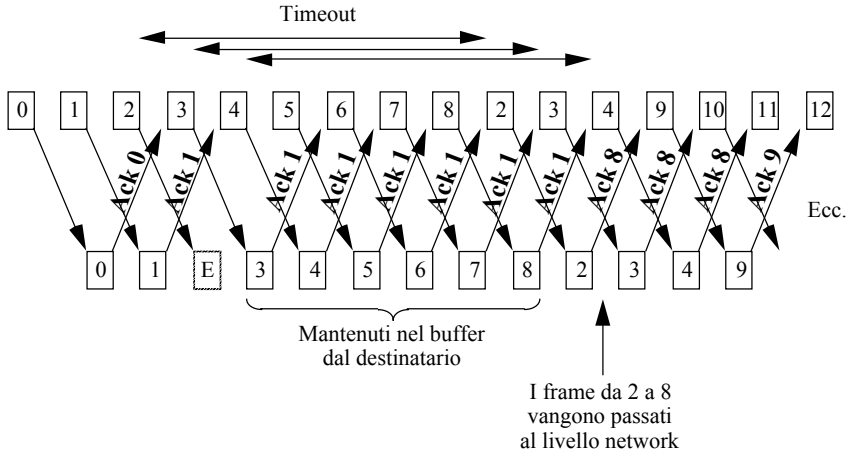


Figura 3-11: Funzionamento del protocollo selective repeat

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.

3.4) Esempi di protocolli 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).

3.4.1) 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.

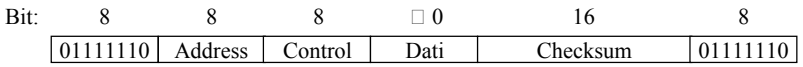


Figura 3-12: Frame HDLC

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:

- usa una finestra scorrevole con numeri di sequenza a 3 bit, contenuti dentro un campo ***Seq*** situato all'interno del campo Control;
- utilizza il campo ***Next***, anch'esso contenuto in Control, per il piggybacking degli ack;
- ha tre tipi di frame (identificati dai primi due bit di Control):
  - ***Information***, per la trasmissione dati;
  - ***Supervisory***, per comandare diverse modalità di ritrasmissione;
  - ***Unnumbered*** (non ha il numero di sequenza), per finalità di controllo o per trasportare il traffico di connessioni non affidabili.

3.4.2) 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 character stuffing.
Ha diverse limitazioni:

- non c'è controllo degli errori;
- supporta solo IP, e per di più solo indirizzi statici;
- non è uno standard ufficiale di Internet.

3.4.3) 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***);
- supporto di molteplici protocolli di livello network;
- un protocollo per negoziare opzioni di livello network (***NCP, Network Control Protocol***):
    - per ogni livello network supportato c'è un differente NCP;
    - 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);
- c'è un campo apposito per il supporto multiprotocollo offerto al livello network.

Il formato del frame PPP è il seguente.

Byte:	1	1	1	1	Variabile	2 oppure 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Dati	Checksum	Flag 01111110

Figura 3-13: 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)

4) Il sottolivello MAC (Medium Access Control)

Come già chiarito, le reti sono divise in due categorie: ***punto a punto*** e ***broadcast***.

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

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

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

Il problema principale è come allocare il canale ai vari utenti in competizione. Ci sono due meccanismi fondamentali:
- ***allocazione statica***, che viene decisa in anticipo;
- ***allocazione dinamica***, che si adatta alle esigenze di ogni momento.

L'allocazione statica prevede la suddivisione del canale fra gli N utenti, ciascuno dei quali riceve di conseguenza una frazione della banda totale. Si può fare, ad esempio, con tecniche quali FDM, allocando a ciascun utente una banda di frequenze distinta da quella degli altri utenti. Ciò va bene se il numero di utenti non varia rapidamente e se tutti trasmettono con un data rate più o meno costante, però in genere comporta vari problemi:
- si verifica uno spreco di banda quando uno o più utenti non trasmettono;
- poiché il traffico è in generale molto ***bursty***, i picchi che si verificano non possono essere gestiti solamente con la sottobanda allocata.

Viceversa, l'allocazione dinamica cerca di adeguarsi alle esigenze trasmissive, in modo da soddisfarle al meglio. Ci sono alcune assunzioni da fare:
1. ***modello a stazioni***: ci sono N stazioni indipendenti, ognuna delle quali genera nuovi frame per la trasmissione. La probabilità di generare un frame in un intervallo di tempo T è uguale a pT, dove p è una costante e rappresenta il tasso d'arrivo dei nuovi frame. Quando un frame è generato, la stazione si blocca finché esso non è trasmesso;
2. ***singolo canale***: un singolo canale, e null'altro, è disponibile per le comunicazioni; tutte le stazioni vi possono trasmettere e da esso possono ricevere, e tutte sono ad uguale livello;
3. ***collisioni***: se due frame vengono trasmessi contemporaneamente, si sovrappongono ed il segnale risultante è rovinato (si verifica collisione):
    - tutte le stazioni possono rilevare la collisione;
    - i frame devono essere ritrasmessi;
    - non ci sono altri tipi di errori;

4. ***tempo***: può essere gestito in due modi:
    - ***continuous time***: la trasmissione di un frame può iniziare in un qualunque istante;
    - ***slotted time***: il tempo è diviso in intervalli discreti (***slot***). Uno slot può contenere 0, 1 oppure più di un frame. Ciò corrisponde ad uno slot vuoto, ad uno slot con un frame e ad uno slot in cui si verifica una collisione. La trasmissione può iniziare solo all'inizio di uno slot;
5. ***ascolto del canale***: ci sono due possibilità,
    - ***carrier sense*** (tipico delle LAN): le stazioni, prima di trasmettere, ascoltano il canale; se è occupato non cercano di trasmettere;
    - ***no carrier sense*** (tipico dei canali via satellite, nei quali vi è un elevato round trip time): le stazioni non ascoltano, trasmettono senz'altro; si preoccuperanno dopo di vedere se c'è stata una collisione.

4.1) 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.

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

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

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

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

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

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

Throughput = G*P(0)

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

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:

$$P(k) = \frac{G^k e^{-G}}{k!}$$

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.

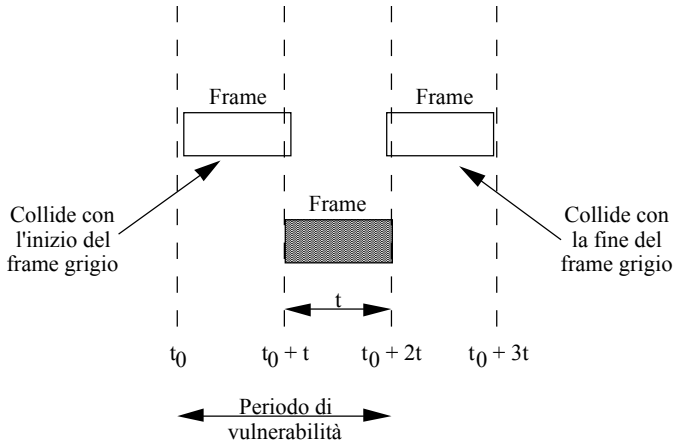


Figura 4-1: Il periodo di vulnerabilità

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

$$Throughput = Ge^{-2G}$$

ed ha la seguente forma:

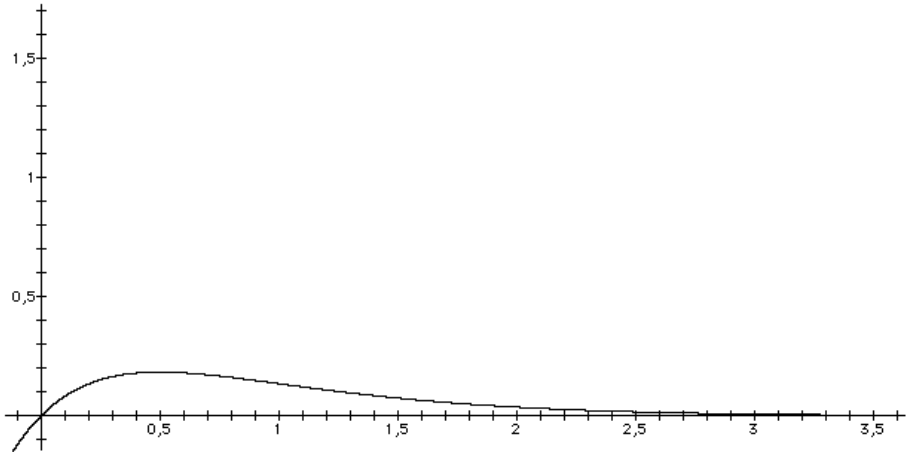


Figura 4-2: Throughput del protocollo Pure Aloha

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.

Un modo per aumentare l'efficienza di Aloha (Roberts, 1972) 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:

$$Throughput = Ge^{-G}$$

ed ha la seguente forma:

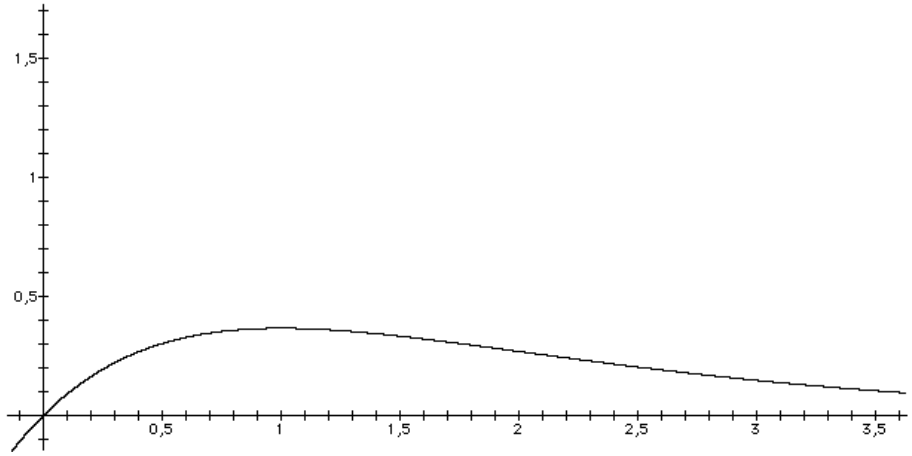


Figura 4-3: Throughput del protocollo Slotted Aloha

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

4.2) Protocolli CSMA (Carrier Sense Multiple Access)

Anche Slotted Aloha ha una bassa efficienza, il che d'altronde è comprensibile visto che le stazioni trasmettono senza preoccuparsi se il canale è libero.

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

Ci sono vari tipi di protocolli carrier sense:

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

4.3) Protocolli CSMA/CD (CSMA 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 $t_0 + 2T$.

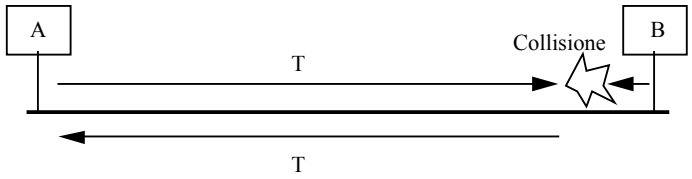


Figura 4-4: Rilevazione di una collisione

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.

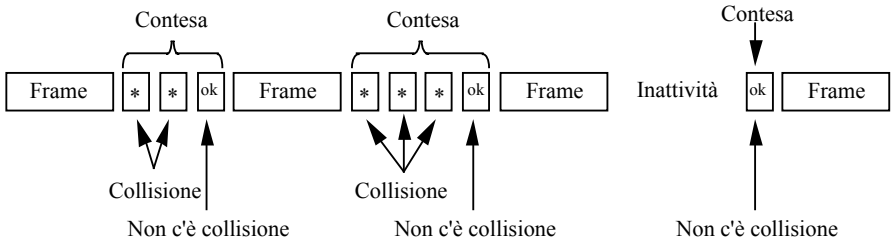


Figura 4-5: Modello concettuale per CSMA / CD

4.4) Le reti ad anello

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

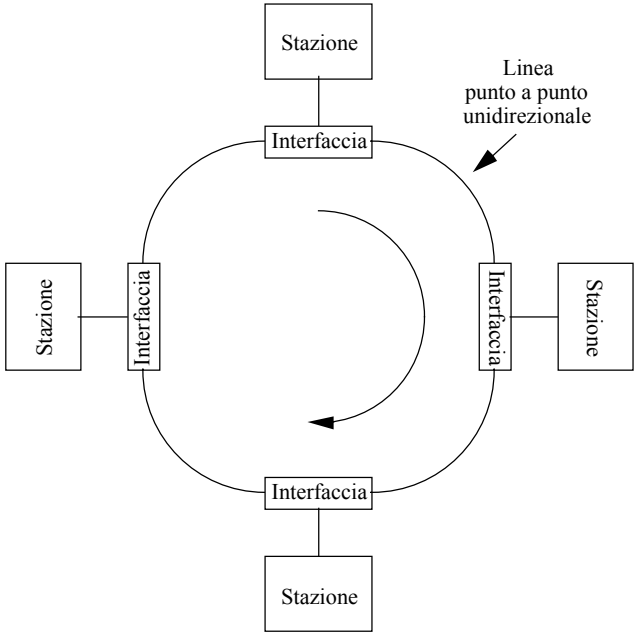


Figura 4-6: Struttura di una rete ad anello

Le reti ad anello hanno diverse attrattive:
- 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***:

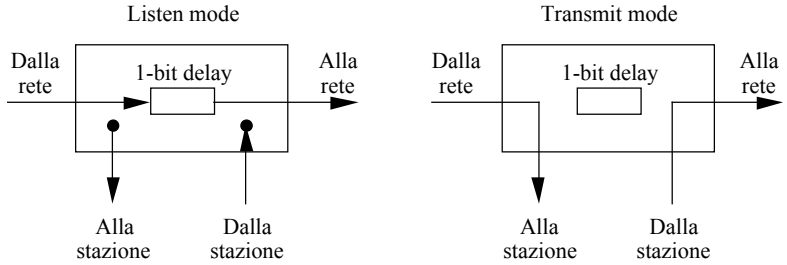


Figura 4-7: Modalità di funzionamento dell'interfaccia di rete

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:
1. aspettare che arrivi il token (in listen mode);
2. rimuoverlo dal ring (in listen mode, vedremo come);
3. trasmettere i dati (in transmit mode);
4. rigenerare il token (in transmit mode);
5. 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 non è così ovvio come sembra (qual'è la lunghezza di un bit?);
- un frame, invece, non è necessario che ci stia tutto 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***).

4.5) Lo standard IEEE 802

IEEE ha prodotto diversi standard per le LAN, collettivamente noti come ***IEEE 802***. Essi includono gli standard per:
- ***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.

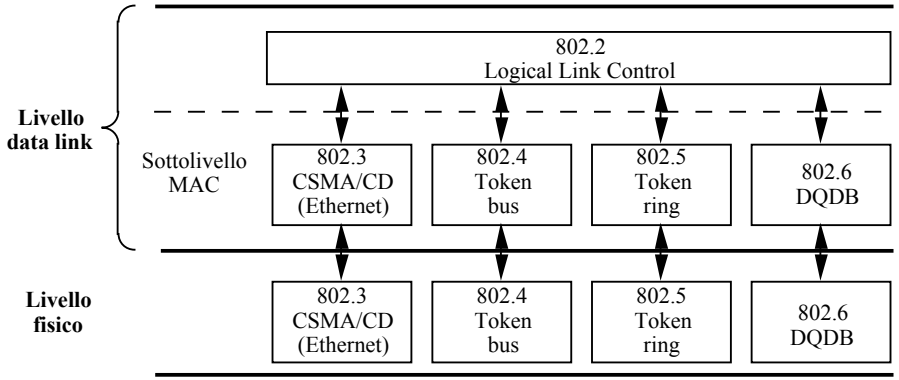


Figura 4-8: Lo standard IEEE 802

4.5.1) 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.

4.5.1.1) Cablaggio

Sono previsti vari cablaggi:
- ***Thick ethernet***: è il primo storicamente; consiste di un cavo coassiale spesso (lo standard suggerisce il colore giallo per la guaina esterna).
  - Ufficialmente si chiama ***10Base5***, ossia:
    - 10 Mbps;
    - Baseband signaling;
    - 500 metri di lunghezza massima.
  - Possono essere installate 100 macchine su un segmento.
  - 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;
    - 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).
  - 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.
- ***Thin ethernet***: è un cavo coassiale più sottile, e si piega più facilmente.
  - Ufficialmente si chiama ***10Base2***, ossia:
    - 10 Mbps;
    - Baseband signaling;
    - 200 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.

- Doppino telefonico:
  - Lo standard ***10BaseT*** (twisted) prevede il collegamento fra una sola coppia di stazioni.
  - La lunghezza massima è 100 metri (150 se il doppino è di classe 5).
  - Per connettere più di due stazioni serve un ***ripetitore multiporta*** (detto ***HUB***).

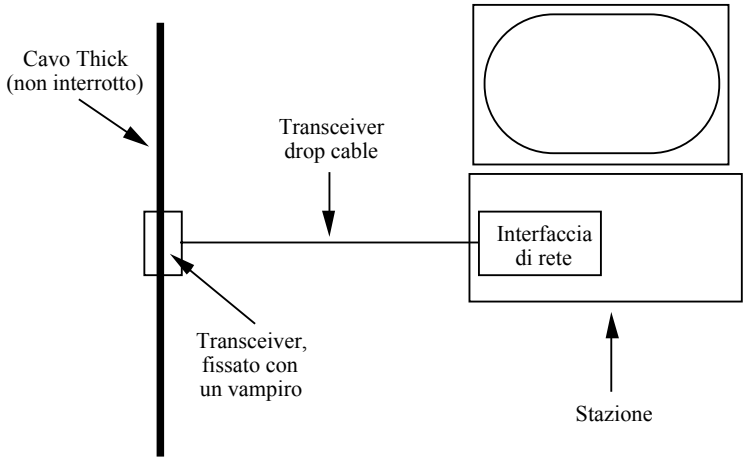


Figura 4-9: Cablaggio Ethernet cavo Thick

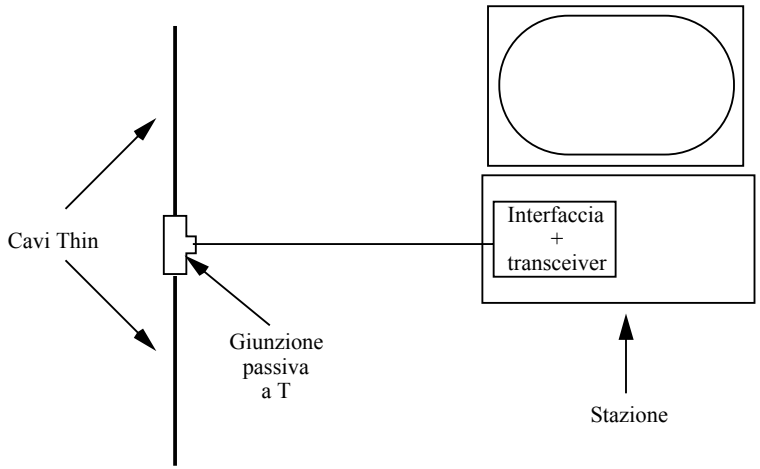


Figura 4-10: Cablaggio Ethernet tramite cavo Thin

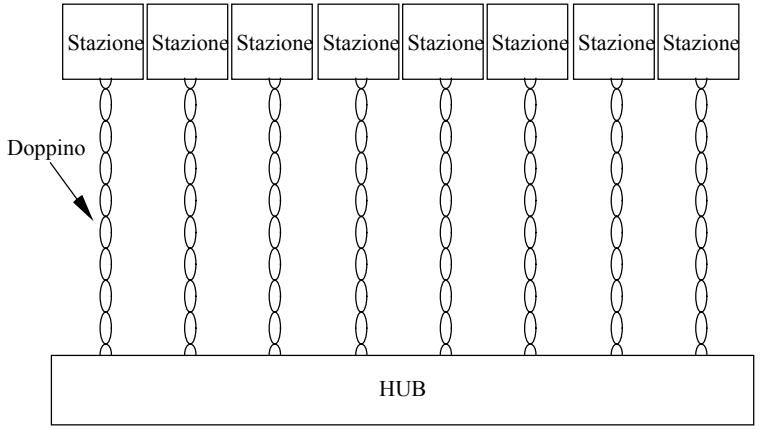


Figura 4-11: Cablaggio Ethernet tramite HUB

Un 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. I ripetitori 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.

4.5.1.2) Codifica dei dati

In 802.3 non si usa una codifica diretta dei dati (ad esempio, zero volt per lo zero e cinque volt per l'uno), perché sarebbe difficile rilevare le collisioni. Inoltre, si vuole delimitare con facilità l'inizio e la fine di ogni singolo bit.

Si usa una codifica, detta ***Manchester***, che prevede una transizione del valore del segnale nel mezzo di ogni bit, zero o uno che sia.

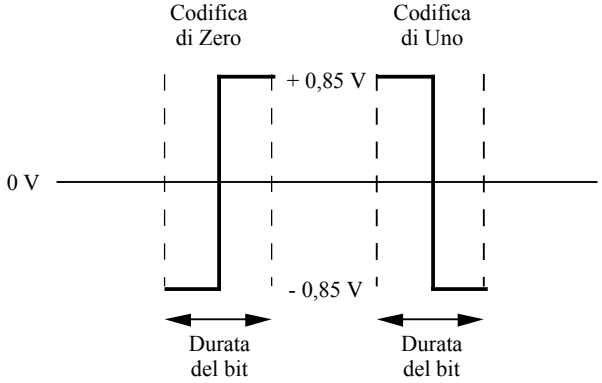


Figura 4-12: Codifica Manchester

Fra i vantaggi di tale codifica:

- facilità di sincronizzazione fra mittente e destinatario;
- il codice trasmissivo è ***bilanciato***, cioé vi è 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;
- è facile rilevare le collisioni.

Si noti però che tale codifica richiede, a parità di velocità di trasmissione, una banda doppia rispetto alla codifica diretta (ogni bit richiede la trasmissione di due valori distinti).

4.5.1.3) Protocollo MAC 802.3

La struttura di un frame 802.3 è la seguente:

Byte:	7	1	2 opp. 6	2 opp. 6	2	0 - 1500	0 - 46	4
	Preamble	Start of frame	Indirizzo destinaz.	Indirizzo sorgente	Lunghezza dei dati	Dati	Pad	Checksum

Figura 4-13: 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.
Indirizzi	gli indirizzi usati sono sempre a 6 byte, e sono univoci a livello mondiale (sono cablati dentro l'interfaccia). E' possibile specificare un singolo destinatario, un gruppo di destinatari (multicast) oppure un invio in broadcast a tutte le stazioni (indirizzo costituito da una sequenza di uni).
Lunghezza dei dati	indica quanti byte ci sono nel campo dati (da 0 a 1500).
Dati	contiene il payload del livello superiore.
Pad	Se il frame (esclusi preambolo e delimiter) è più corto di 64 byte, con questo campo lo si porta alla lunghezza di 64 byte, vedremo poi perché.
Checksum	è un codice CRC come quelli già visti.

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

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

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

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

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

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

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

4.5.1.4) Funzionamento di 802.3

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

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

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

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

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

- se il range fosse sempre piccolo, con molte stazioni si avrebbero praticamente sempre collisioni;

- se il range fosse sempre grande, non ci sarebbero quasi mai collisioni ma il ritardo medio (metà range*slot time) causato da una collisione sarebbe molto elevato.

4.5.1.5) Prestazioni

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

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

La pratica ha mostrato che 802.3:

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

4.5.1.6) Fast Ethernet

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

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

- Doppino classe 3 (100BaseT4)
    - si usano quattro doppini fra l'hub ed ogni stazione:
        - uno viene usato sempre per il traffico dall'hub alla stazione;
        - uno viene usato sempre per il traffico dalla stazione all'hub;
        - 2 vengono usati di volta in volta nella direzione della trasmissione in corso;
    - la codifica è ***8B6T***, cioé 8 bit vengono codificati con 6 ***trit*** (che hanno valore 0, 1 o 2);
    - la velocità di segnalazione è 25 Mhz (solo 25% in più di quella dello standard 802.3, che è di 20 Mhz);
    - si inviano 3 trit sui 3 doppini contemporaneamente a 25 Mhz, ossia 6 trit alla frequenza di 12,5 Mhz. Poiché 6 trit convogliano 8 bit, di fatto si inviano 8 bit a 12,5 Mhz, ottenendo così i 100 Mbps.
- Doppino classe 5 (100BaseT)
    - velocità di segnalazione 124 Mhz;
    - codifica ***4B5B*** (4 bit codificati con 5 bit, introducendo ridondanza);
    - a seconda del tipo di hub:

    - hub tradizionale: la lunghezza massima di un ramo è 100 metri, quindi il diametro della rete è 200 metri (contro i 2,5 km di 802.3).
    - ***switched hub***: ogni ramo è un dominio di collisione separato, e quindi (poiché su esso vi è una sola stazione) non esiste più il problema delle collisioni, ma rimane il limite di 100 metri per i limiti di banda passante del doppino.
- Fibra ottica (100BaseFX)
    - velocità di segnalazione 125 Mhz;
    - codifica 4B5B;
    - obbligatorio switched hub;
    - lunghezza rami fino a 2 km (con uno switched hub non c'è il problema delle collisioni, ed inoltre come sappiamo la fibra regge velocità dell'ordine dei Gbps a distanze anche superiori).

4.5.2) IEEE 802.5

Nel 1972 IBM scelse l'anello per la sua architettura di LAN, a cui diede il nome di ***Token Ring***. Successivamente, IEEE ha definito lo standard IEEE 802.5 sulla base di tale architettura.

Le differenze principali sono che la rete IBM prevede velocità di 4 Mbps e 16 Mbps, mentre 802.5 prevede oltre ad esse anche la velocità di 1 Mbps.

4.5.2.1) Cablaggio

Il cablaggio più diffuso è 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.

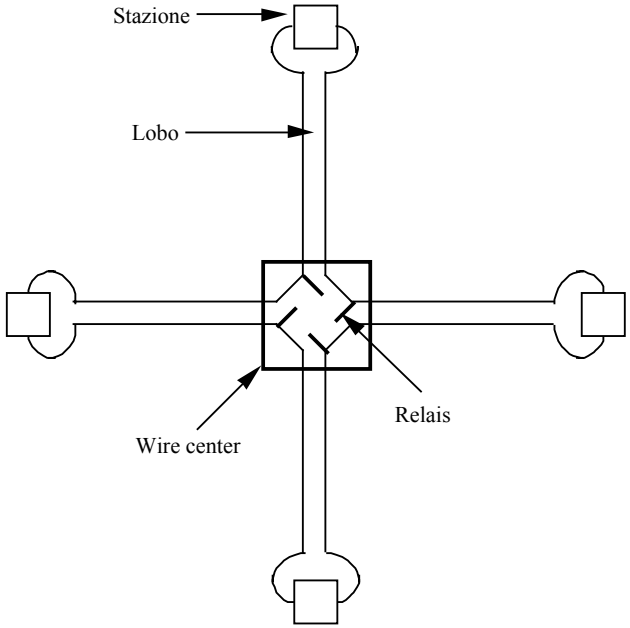


Figura 4-14: Cablaggio con wire center

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.

4.5.2.2) Codifica dei dati

Si usa la codifica ***Differential Manchester Encoding***, definita così:

- valore zero: all'inizio della trasmissione del bit si fa una transizione;
- valore uno: all'inizio della trasmissione del bit non si fa una transizione;
- a metà del bit si fa in entrambi i casi una transizione.

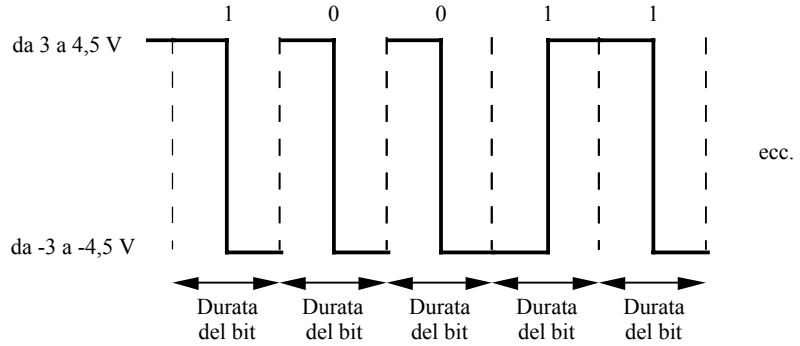


Figura 4-15: Codifica Differential Manchester

4.5.2.3) Protocollo MAC 802.5

La struttura del token e del frame di 802.5 è la seguente:

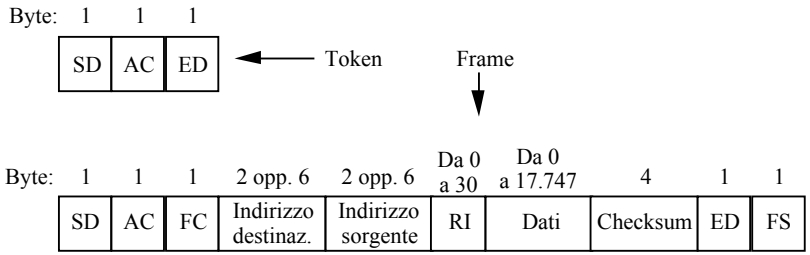


Figura 4-16: Token e frame 802.5

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	come 802.3.
RI	Routing information, contiene (se c'è) le informazioni necessarie al source routing (vedremo più avanti).
Dati	contiene il payload del livello superiore.
Checksum	è un codice CRC come quelli già visti.
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.

4.5.2.4) Funzionamento di 802.5

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;

    - 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.

4.5.3) Confronto fra 802.3 ed 802.5

Vantaggi di 802.3:

- ha un'enorme diffusione;
- esibisce un buon funzionamento a dispetto della teoria.

Svantaggi di 802.3

- ha sostanziose componenti analogiche (per il rilevamento delle collisioni);
- il funzionamento peggiora con forte carico.

Vantaggi di 802.5:

- è totalmente digitale;
- va molto bene sotto forte carico.

Svantaggi di 802.5

- c'è ritardo anche senza carico (per avere il token);
- ha bisogno di un monitor (e se è "malato", cioé malfunzionante, e nessuno se ne accorge?).

In definitiva, nessuna delle due può essere giudicata la migliore in assoluto.

4.5.4) IEEE 802.2

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

Ha due funzioni principali:

- fornire al livello network un'interfaccia unica, nascondendo le differenze fra i vari sottolivelli MAC;
- fornire, se è richiesto dal livello superiore, un servizio più sofisticato di quello offerto dai vari sottolivelli MAC (che, ricordiamo, offrono solo servizi datagram). Esso infatti offre:
    - servizi datagram;
    - servizi datagram confermati;
    - 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.

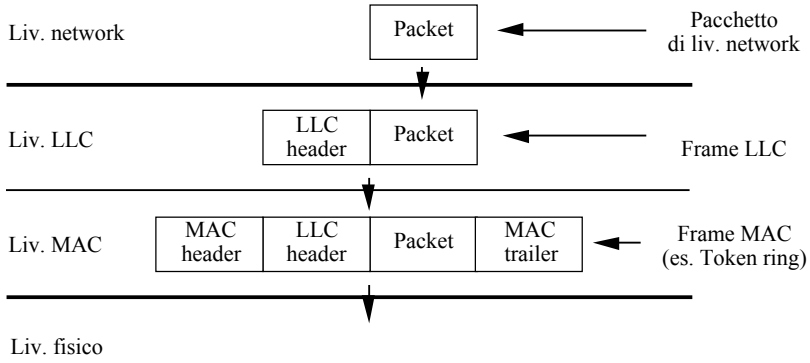


Figura 4-17: Buste LLC e MAC

4.6) Il bridge

Molto spesso c'è la necessità di connettere fra di loro LAN distinte, per molte ragioni:

- 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;
- si vuole una LAN la cui lunghezza superi i limiti massimi consentiti (ad esempio, 2,5 km per Ethernet);

- si desidera, nel caso di una LAN contenente molti host, suddividerla in molteplici LAN interconnesse. Questo per tenere separato 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 essere interconnesse con dispositivi detti ***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 affatto in considerazione quelle di livello tre. Questa è la caratteristica fondamentale che li differenzia dai ***router***, che invece agiscono a livello tre.

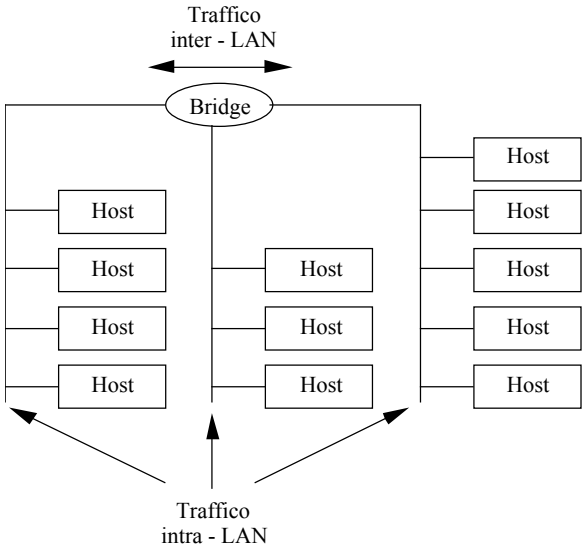


Figura 4-18: Interconnessione di LAN tramite bridge

In questo esempio il traffico totale (se è tutto confinato entro le singole LAN) può arrivare a tre volte quello di una singola LAN. Solo il traffico fra host di LAN diverse attraversa il 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, quindi il bridge appartiene in tutto e per tutto al livello data link.

Il funzionamento di un bridge, che ha tante interfacce di rete quante sono le LAN alle quali è fisicamente collegato, è il seguente:

- 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 ripetitore, 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*** (cioé non far passare) 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: 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; ad esempio, nello standard 802 i frame troppo grandi devono essere scartati;
- 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.

4.6.1) Standard IEEE per i bridge

Ci sono due tipi di bridge standardizzati da IEEE:

- ***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. Incredibile a dirsi, la cosa funziona!

Il mecccanismo è il seguente:

- Dal momento in cui il bridge viene attivato, esamina tutti i frame che gli 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*** (che vedremo meglio più avanti);
    - 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.

5) Il livello tre (Network)

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

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

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

Nel progetto e nella realizzazione del livello network di una architettura di rete si devono prendere decisioni importanti in merito a:
- ***servizi offerti*** al livello transport;
- ***organizzazione interna*** della subnet di comunicazione.

5.1) Servizi offerti

In merito ai servizi offerti al livello superiore, ci sono (come abbiamo già accennato) due tipologie fondamentali di servizi:
- servizi connection-oriented;
- servizi connectionless.

In proposito, ci sono due scuole di pensiero:
- fautori dei servizi connection-oriented (compagnie telefoniche);
- fautori dei servizi connectionless (Internet Community).

La prima scuola di pensiero afferma che il livello network deve fornire un servizio sostanzialmente affidabile e orientato alla connessione. In questa visione, succede che:
- le peer entitiy stabiliscono una connessione, negoziandone i parametri (di qualità, di costo, ecc.), alla quale viene associato un identificatore;
- tale identificatore viene inserito in ogni pacchetto che verrà inviato;
- la comunicazione è bidirezionale e i pacchetti viaggiano, in sequenza, lungo il cammino assegnato alla connessione;
- il controllo di flusso è fornito automaticamente (attraverso alcuni dei parametri negoziati, come ad esempio il dimensionamento di una o più finestre scorrevoli).

La seconda scuola di pensiero ritiene invece che la sottorete debba solo muovere dati e nient'altro:
- la sottorete è giudicata inerentemente inaffidabile, per cui gli host devono provvedere per conto proprio alla correzione degli errori e al controllo di flusso;
- una ovvia conseguenza è che il servizio offerto dal livello network dev'essere datagram, visto che è inutile inserire le funzioni di controllo degli errori e del flusso in due diversi livelli;
- i pacchetti viaggiano indipendentemente, e dunque devono tutti contenere un identificatore (ossia l'indirizzo) della destinazione.

Di fatto, il problema è dove mettere la complessità della realizzazione:
- la prima scuola la mette nei nodi della subnet, che si devono occupare del ***setup delle connessioni*** e di fornire la necessaria affidabilità;
- la seconda scuola la mette negli host, i cui livelli transport forniscono l'affidabilità e l'orientamento alla connessione.

In realtà le decisioni sono due, separate:
- offrire o no un servizio affidabile;
- offrire o no un servizio orientato alla connessione.

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

5.2) Organizzazione interna della subnet

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

Una subnet può essere organizzata con un funzionamento interno:
- basato su connessioni (che in questo contesto si chiamano ***circuiti virtuali***):
  - la subnet stabilisce un circuito virtuale (sul quale verrà tipicamente veicolato il traffico di un servizio connection oriented), cioé crea un cammino fra la sorgente e la destinazione;
  - tutti i router lungo tale cammino ricordano, in una apposita struttura dati, la parte di loro competenza di tale cammino (e cioé quale linea in entrata e quale in uscita sono assegnate al cammino);
  - quando arrivano pacchetti che contengono l' ID di tale circuito virtuale, essi vengono instradati di conseguenza (tutti nello stesso modo).
- connectionless:
  - i router si limitano a instradare ogni pacchetto che arriva sulla base del suo indirizzo di destinazione, decidendo di volta in volta come farlo proseguire;
  - i router hanno delle ***tabelle di instradamento*** (***routing table***)che indicano, per ogni possibile destinazione, quale linea in uscita utilizzare; si noti che queste

tabelle esistono anche nelle subnet del tipo precedente, dove però servono solamente nella fase di setup della connessione (per decidere come instradare i pacchetti di setup);
- quando offre un servizio connection-oriented, questo livello fa credere al livello superiore che esista una connessione, ma poi i pacchetti viaggiano indipendentemente (e quindi hanno tutti l'indirizzo del destinatario) e vengono rimessi in ordine dal livello network solo a destinazione, prima di essere consegnati al livello superiore.

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

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

Dev'essere chiaro che i servizi offerti sono indipendenti dalla realizzazione interna della subnet. E' possibile avere tutte le quattro combinazioni di servizio offerto e implementazione della subnet:
- servizi connection oriented su circuiti virtuali;
- servizi connectionless su subnet datagram;
- servizi connection oriented su subnet datagram (si cerca di fornire comunque un servizio robusto);
- servizi connectionless su circuito virtuale (esempio: IP su subnet ATM).

5.3) Algoritmi di routing

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

Un ***algoritmo di routing*** è quella parte del software di livello network che decide su quale linea di uscita instradare un pacchetto che è arrivato:
- in una subnet datagram l'algoritmo viene applicato ex novo ad ogni pacchetto;
- in una subnet basata su circuiti virtuali l'algoritmo viene applicato solo nella fase di setup del circuito; in tale contesto si usa spesso il termine ***session routing***.

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

Purtroppo, gli ultimi due requisiti sono spesso in conflitto fra loro; inoltre, a proposito dell'ottimalità, non sempre è chiaro cosa si voglia ottimizzare. Infatti, supponiamo che si vogliano:
- minimizzare il ritardo medio pacchetti;
- massimizzare il throughput totale dei pacchetti.

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

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

Gli algoritmi adattivi differiscono fra loro per:
- come ricevono le informazioni:
  - localmente;
  - dai router adiacenti;
  - da tutti i router;
- quanto spesso rivedono le decisioni:
  - a intervalli di tempo prefissati;
  - quando il carico cambia;
  - quando la topologia cambia;
- quale metrica di valutazione adottano:
  - distanza;

- numero di hop;
- tempo di transito stimato.

5.3.1) Il principio di ottimalità

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

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

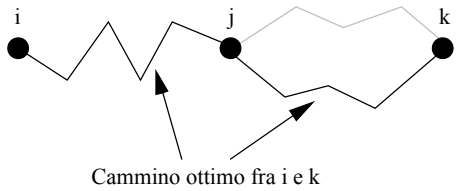


Figura 5-1: Principio di ottimalità

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

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

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

5.3.2) Algoritmi statici

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

Esamineremo i seguenti algoritmi statici:
- ***shortest path routing***;
- ***flooding***;
- ***flow-based routing***.

Shortest path routing
L'idea è semplice: un host di gestione della rete mantiene un grafo che rappresenta la subnet:
- i nodi rappresentano i router;
- gli archi rappresentano le linee punto-punto.

All'avvio della rete (o quando ci sono variazioni permanenti della topologia) l'algoritmo:
- applica al grafo un algoritmo per il calcolo del ***cammino minimo*** fra ogni coppia di nodi; ad esempio, il noto algoritmo di Dijkstra ('59) può essere usato;
- invia tali informazioni a tutti i router.

Quale sia il cammino minimo dipende da qual'è la grandezza che si vuole minimizzare. Tipicamente si usano:
- numero di hop, cioè di archi, da attraversare;
- lunghezza dei collegamenti;
- tempo medio di accodamento e trasmissione;
- una combinazione di lunghezza, banda trasmissiva, traffico medio, ecc.

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

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

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

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

Flow-based routing
Questo algoritmo è basato sull'idea di:
- calcolare in anticipo il traffico atteso su ogni linea;
- da questi calcoli derivare una stima del ritardo medio atteso per ciascuna linea;
- basare su tali informazioni le decisioni di routing.

Le informazioni necessarie per poter applicare l'algoritmo sono:
- la topologia della rete;
- la matrice delle quantità di traffico T(i,j) stimate fra ogni coppia (i,j) di router;
- le capacità (in bps ad esempio) delle linee point to point.

Vengono fatte le seguenti assunzioni:
- il traffico è stabile nel tempo e noto in anticipo;
- il ritardo su ciascuna linea aumenta all'aumentare del traffico sulla linea e diminuisce all'aumentare della velocità della linea secondo le leggi della teoria delle code.

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

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

5.3.3) Algoritmi dinamici

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

Distance vector routing
Ogni router mantiene una tabella (***vector***) contenente un elemento per ogni altro router. Ogni elemento della tabella contiene:
- la distanza (numero di hop, ritardo, ecc.) che lo separa dal router in oggetto;
- la linea in uscita da usare per arrivarci;

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

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

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

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

per ogni destinazione.

Ovviamente, la migliore è la concatenazione che produce la minore somma di:
- distanza fra il router stesso ed un suo vicino immediato (viene dalla misurazione diretta);
- distanza fra quel vicino immediato ed il router remoto di destinazione (viene dalla tabella ricevuta dal vicino immediato).

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

Infatti, consideriamo questo esempio:

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

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

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

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

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

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

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

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

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

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

L'idea è questa:
- ogni router tiene sott'occhio lo stato dei collegamenti fra se e i suoi vicini immediati (misurando il ritardo di ogni linea) e distribuisce tali informazioni a tutti gli altri;
- sulla base di tali informazioni, ogni router ricostruisce localmente la topologia completa dell'intera rete e calcola il cammino minimo fra se e tutti gli altri.

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

6. calcolare il cammino più breve a tutti gli altri router.

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

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

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

La costruzione e l'invio di tali pacchetti si verifica tipicamente:
- a intervalli regolari;
- quando accade un evento significativo (es.: una linea va giù o torna su).

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

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

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

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

5.3.4) Routing gerarchico

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

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

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

Di conseguenza, ci solo due livelli di routing:

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

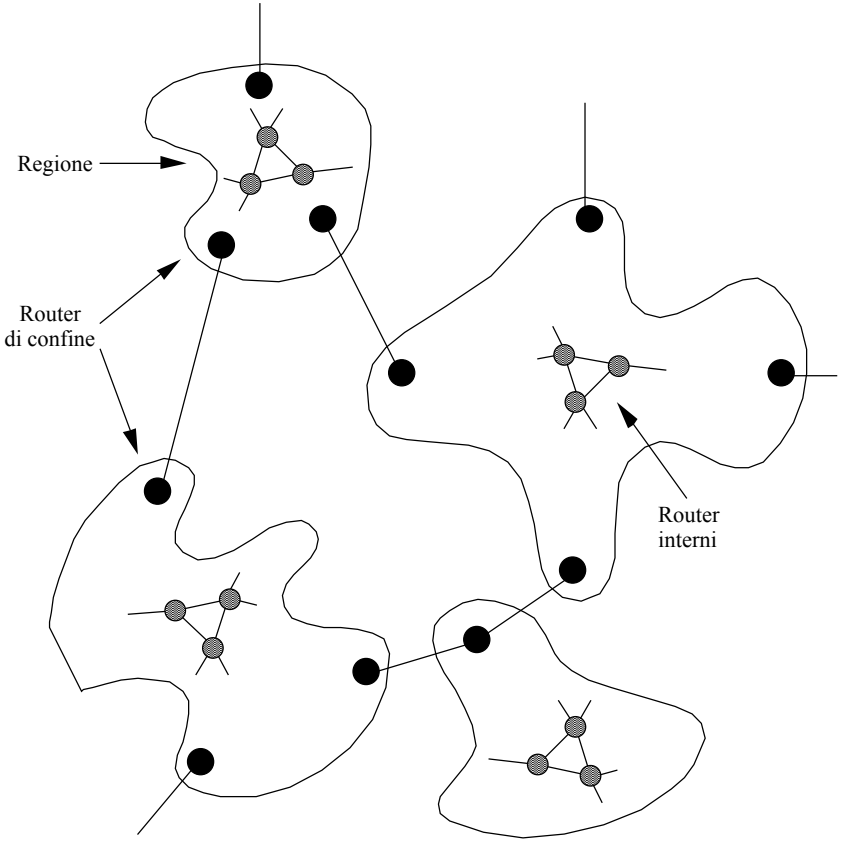


Figura 5-2: Routing gerarchico

I router interni mantengono nelle loro tabelle di routing:

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

I router di confine, invece, mantengono:

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

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

5.4) Controllo della congestione

Quando troppi pacchetti sono presenti in una parte della subnet, si verifica una ***congestione*** che degrada le prestazioni. Ciò dipende dal fatto che, quando un router non riesce a gestire tutti i pacchetti che gli arrivano, comincia a perderli, e ciò causa delle ritrasmissioni che aggravano ancor più la congestione.

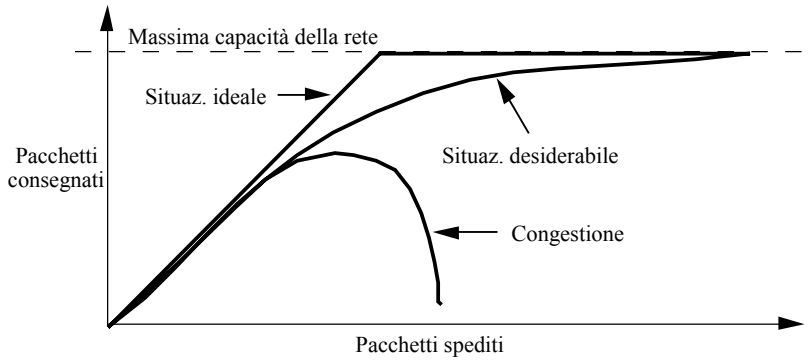


Figura 5-3: Effetti della congestione

La congestione in un router può derivare da diversi fattori:

- troppi pochi buffer nel router;
- processore troppo lento nel router;
- linea di trasmissione troppo lenta (si allunga la coda nel router di partenza).

Inoltre, la congestione in un router tende a propagarsi ai suoi vicini che gli inviano dati. Infatti, quando tale router è costretto a scartare i pacchetti che riceve non li conferma più, e

quindi i router che li hanno spediti devono mantenerli nei propri buffer, aggravando così anche la propria situazione.

Il ***controllo della congestione*** è un problema globale di tutta la rete, ed è ben diverso dal problema del controllo di flusso nei livelli data link, network (nel caso dei servizi connection oriented) e trasporto, che invece riguarda una singola connessione sorgente-destinazione.

Ci sono due approcci al problema della congestione:
- ***open loop*** (senza controreazione);
- ***closed loop*** (con controreazione).

Il primo cerca di impostare le cose in modo che la congestione non si verifichi, ma poi non effettua azioni correttive.

Il secondo tiene sott'occhio la situazione della rete, intraprendendo le azioni opportune quando necessario.

5.4.1) Traffic shaping

In questo approccio, di tipo open loop, l'idea è di forzare la trasmissione dei pacchetti a un ritmo piuttosto regolare, onde limitare la possibilità di congestioni.

Verdemo tre tecniche per implementare il traffic shaping:
- ***leaky bucket***;
- ***token bucket***;
- ***flow specification***.

Algoritmo Leaky bucket (secchio che perde)
L'idea è semplice, e trova un analogia reale in un secchio che viene riempito da un rubinetto (che può essere continuamente manovrato in modo da risultare più o meno aperto) e riversa l'acqua che contiene attraverso un forellino sul fondo, a ritmo costante. Se viene immessa troppa acqua, essa fuoriesce dal bordo superiore del secchio e si perde.

Sull'host si realizza (nell'interfaccia di rete o in software) un leaky bucket, che è autorizzato a riversare sulla rete pacchetti con un fissato data rate (diciamo b bps) e che mantiene, nei suoi buffer, quelli accodati per la trasmissione.

Se l'host genera più pacchetti di quelli che possono essere contenuti nei buffer, essi si perdono.

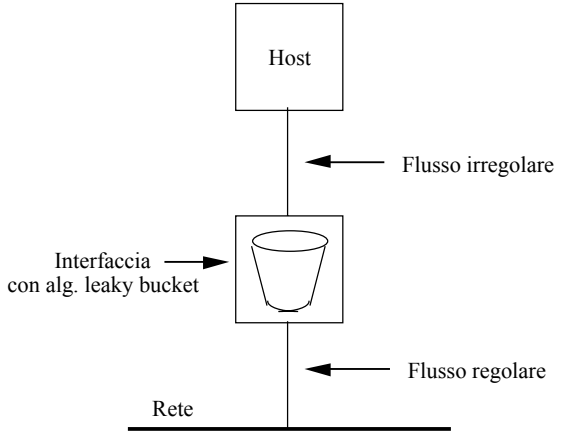


Figura 5-4: Leaky bucket

In questo modo, l'host può anche produrre un traffico bursty senza creare problemi sulla rete; finché il data rate medio non supera i b bps tutto funziona regolarmente, oltre si cominciano a perdere pacchetti.

Algoritmo token bucket (secchio di gettoni)
E' una tecnica per consentire un grado di irregolarità controllato anche nel flusso che esce sulla rete.

Essenzialmente, si accumula un credito trasmissivo con un certo data rate (fino ad un massimo consentito) quando non si trasmette nulla.

Quando poi c'è da trasmettere, lo si fa sfruttando tutto il credito disponibile per trasmettere, fino all'esaurimento di tale credito, alla massima velocità consentita dalla linea.

Il secchio contiene dei ***token***, che si creano con una cadenza prefissata (ad esempio uno ogni millisecondo) fino a che il loro numero raggiunge un valore M prefissato, che corrisponde all'aver riempito il secchio di token.

Per poter tramettere un pacchetto (o una certa quantità di byte), deve essere disponibile un token.

Se ci sono k token nel secchiello e h > k pacchetti da trasmettere, i primi k sono trasmessi subito (al data rate consentito dalla linea) e gli altri devono aspettare dei nuovi token.

Dunque, potenzialmente dei burst di M pacchetti possono essere trasmessi in un colpo solo, fermo restando che mediamente non si riesce a trasmettere ad una velocità più alta di quella di generazione dei token.

Un'altra differenza col leaky bucket è che i pacchetti non vengono mai scartati (il secchio contiene token, non pacchetti). Se necessario, si avverte il livello superiore, produttore dei dati, di fermarsi per un pò.

Questi due algoritmi possono essere usati per regolare il traffico host-router e router-router; in quest'ultimo caso però, se il router sorgente è costretto a fermarsi invece di inviare dati e non ha spazio di buffer a sufficienza, questi possono perdersi.

Flow specification
Il traffic shaping è molto efficace se tutti (sorgente, subnet e destinazione) si accordano in merito.

Un modo di ottenere tale accordo consiste nello specificare:
- le caratteristiche del traffico che si vuole inviare (data rate, grado di burstiness, ecc.);
- la qualità del servizio (ritardo massimo, frazione di pacchetti che si può perdere, ecc.).

Tale accordo si chiama ***flow specification*** e consiste in una struttura dati che descrive le grandezze in questione.

Sorgente, subnet e destinatario si accordano di conseguenza per la trasmissione.
Questo accordo viene preso prima di trasmettere, e può essere fatto sia in subnet connesse (e allora si riferisce al circuito virtuale) che in subnet non connesse (e allora si riferisce alla sequenza di pacchetti che sarà trasmessa).

Controllo della congestione nelle reti basate su circuiti virtuali
In generale nelle reti connesse è più facile il controllo della congestione, perché le risorse per una connessione sono allocate in anticipo.

Quindi, per evitare la congestione è possibile negare l'attivazione di nuovi circuiti virtuali ove non vi siano sufficienti risorse per gestirli. Questa tecnica va sotto il nome di ***admission control***.

5.4.2) Choke packet

In questo approccio, di tipo closed loop, è previsto che un router tenga d'occhio il grado di utilizzo delle sue linee di uscita. Il router misura, per ciascuna linea, l'utilizzo istantaneo U e accumula, entro una media esponenziale M, la storia passata:

$$M_{nuovo} = a\, M_{vecchio} + (1 - a)U$$

dove

- il parametro a (compreso fra 0 ed 1) è il peso dato alla storia passata;
- (1 - a) è il peso dato all'informazione più recente.

Quando, per una delle linee in uscita, M si avvicina a una soglia di pericolo prefissata, il router esamina i pacchetti in ingresso per vedere se sono destinati alla linea d'uscita che è in allarme.

In caso affermativo, invia all'host di origine del pacchetto un ***choke packet*** (to choke significa soffocare) per avvertirlo di diminuire il flusso.

Quando l'host sorgente riceve il choke packet diminuisce il flusso (tipicamente lo dimezza) e ignora i successivi choke packet per un tempo prefissato, perché tipicamente ne arriveranno molti in sequenza.

Trascorso tale tempo prefissato, l'host si rimette in attesa di altri choke packet. Se ne arrivano altri, riduce ancora il flusso. Altrimenti, aumenta di nuovo il flusso.

Hop-by-hop choke packet
L'unico problema della tecnica precedente è la lentezza di reazione, perché l'host che produce i pacchetti ci mette un certo tempo a ricevere i choke packet ed a diminuire di conseguenza il ritmo della trasmissione. Per migliorare le cose si può costringere ogni router sul percorso, appena riceve tali pacchetti, a rallentare subito il ritmo. In tal caso si parla di ***hop-by-hop choke packet***.

Questa tecnica rende molto più veloce il sollievo del router che ha per primo i problemi di congestione, ma richiede più spazio di buffer nei router sul percorso dall'host originario a quel router.

5.5) Internetworking

Come sappiamo, esistono diverse architetture di rete, ciascuna caratterizzata dalle scelte effettuate in molti settori fra i quali ricordiamo:
- i servizi offerti dai vari livelli (connection oriented o no, reliable o no, ecc.);
- le modalità di indirizzamento;
- la dimensione massima dei pacchetti.

Per connettere fra loro reti eterogenee si devono superare problemi non banali, tra i quali:
- difformità nei servizi offerti (ad esempio, un servizio connected viene offerto solo su una delle reti);
- difformità nei formati dei pacchetti e degli indirizzi;
- difformità, nelle subnet, dei meccanismi di controllo dell'errore e della congestione;
- difformità nella dimensione massima dei pacchetti.
-

A causa di tali problemi, in generale (a meno che le architetture di rete non siano molto simili) non si usa questo approccio generale, ma altre tecniche più specifiche.

Tali tecniche sono basate sull'uso di particolari attrezzature, che operano a livelli diversi:

- i bridge, che abbiamo già visto e che operano a livello data link;
- i ***router multiprotocollo***: sono dei router in grado di gestire contemporaneamente più pile di protocolli.

Reti di router multiprotocollo

Mediante ricorso a tali apparecchiature diventa possibile, per esempio, impostare una internetwork costituita di reti eterogenee. Ogni rete può comunicare con le altre reti a lei conformi attraverso una porzione di rete costituita di router multiprotocollo.

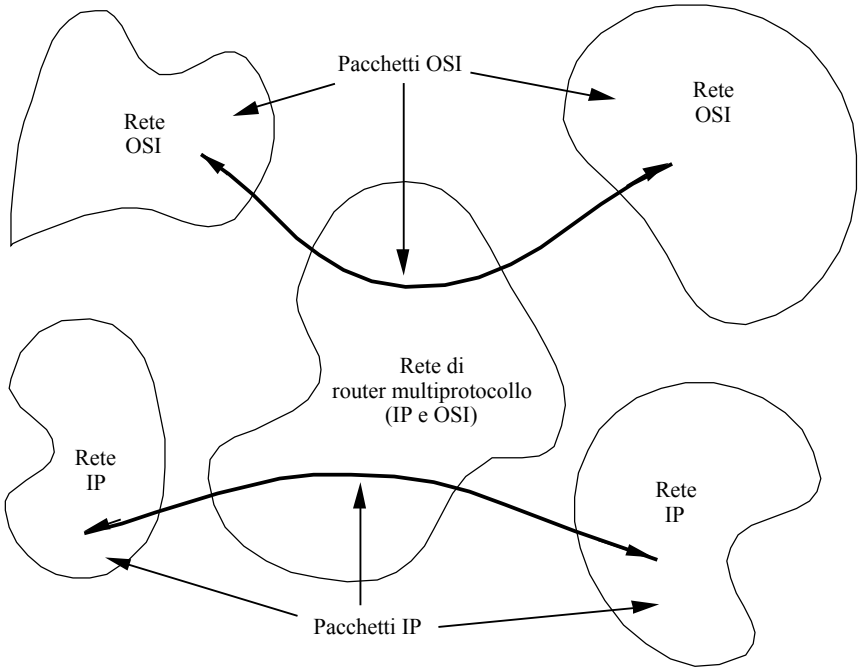


Figura 5-5: Internetwork basata su router multiprotocollo

Le reti OSI possono parlare fra loro, e così quelle IP. Nella subnet costituita dai router multiprotocollo circolano pacchetti di entrambe le architetture, che vengono instradati secondo le regole di competenza dell'architettura di sui fanno parte.

Tunneling

Un' altra tecnica che risolve un problema analogo è il ***tunneling***, che si utilizza per mettere in comunicazione due reti uguali per mezzo di una rete diversa.

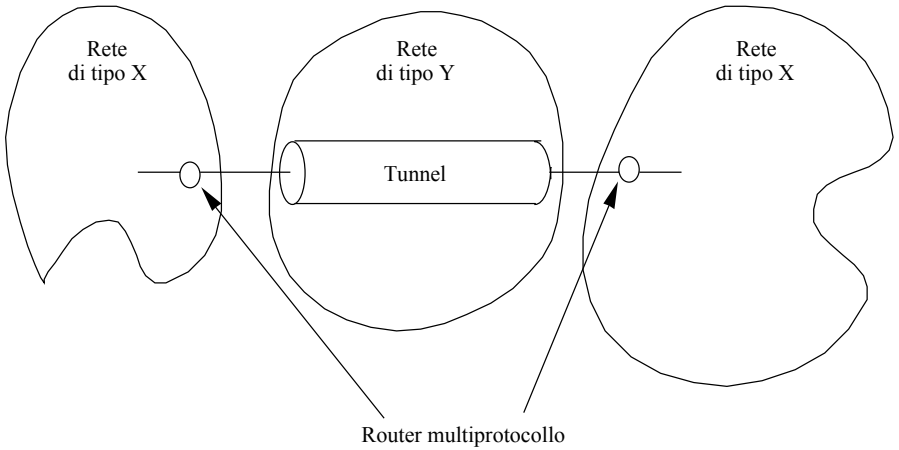


Figura 5-6: Tunneling

In questo caso la rete di tipo Y non è dotata di router multiprotocollo. Invece, un router designato in ciascuna delle due reti di tipo X è multiprotocollo e incapsula i pacchetti delle reti di tipo X dentro pacchetti di tipo Y, consegnandoli poi alla rete di tipo Y. Si noti che in questi pacchetti ci sono due buste di livello network:

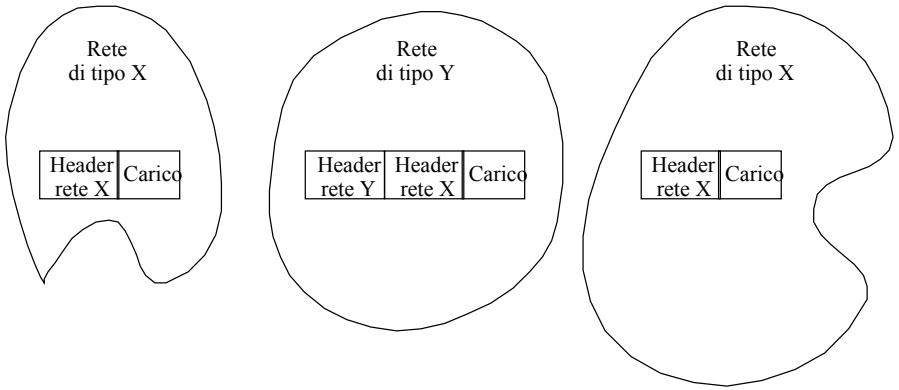


Figura 5-7: Doppio livello network nel tunneling

Frammentazione
Un diverso problema che talvolta si presenta è legato al fatto che in generale è possibile che i pacchetti in arrivo da una rete siano troppo grandi per transitare su un altra.

In questo caso è necessario che il livello network della rete di origine (e di quella di destinazione) prevedano meccanismi di ***spezzettamento*** del pacchetto in ***frammenti*** prima di consegnarli alla rete di transito, e di ***ricomposizione*** dei frammenti appena essi giungono dalla rete di transito in quella di destinazione (come vedremo, il protocollo IP è fornito di questa funzionalità).

Circuiti virtuali concatenati
Se tutte le reti interessate offrono servizi connessi nel livello network, è possibile costruire un circuito virtuale che si estende attraverso più reti eterogenee come ***concatenazione di circuiti virtuali*** che attraversano ciascuno una delle reti. Ai confini fra ogni rete ci sono dei router multiprotocollo che:

- creano la porzione di circuito virtuale che attraversa la rete di competenza, arrivando fino ad un router multiprotocollo situato all'altra estremità della rete;
- instradano successivamente i pacchetti lungo tale circuito virtuale.

Transport gateway
Un meccanismo simile è piuttosto diffuso a livello transport, dato che esso offre servizi connessi in quasi tutte le architetture. In tale ambito le apparecchiature interessate prendono il nome di ***transport gateway***.

Poiché i transport gateway operano a livello transport, essi ricevono i dati dal livello application di partenza e li consegnano al livello application di arrivo.

Application gateways
Un ultimo tipo di attrezzatura è costituito dagli ***application gateway***, che effettuano una conversione di dati a livello application.

Ad esempio, per mandare da un host Internet un messaggio di posta elettronica a un utente OSI:

- si compone il messaggio secondo il formato Internet; l'indirizzo del destinatario è un indirizzo OSI, ma viene espresso secondo regole valide per Internet;
- si invia il messaggio, che viene consegnato a un ***mail gateway***, cioè a un server di posta elettronica dotato della capacità di conversione dei formati;

il mail gateway estrae il testo dal messaggio, lo inserisce in un nuovo messaggio costruito secondo il formato OSI (che, per la posta elettronica, è definito dallo standard X.400) e lo invia sulla rete OSI (consegnandolo al pertinente livello trasporto).

5.5.1) Internetwork routing

In generale, in una internetwork le singole reti componenti sono entità autonome e vengono chiamate ***AS*** (***Autonomous System***).

In questo caso, il routing complessivo è a due livelli:

- un primo livello è costituito dall'***Interior Gateway Protocol*** (***IGP***). Questo termine identifica l' algoritmo di routing usato da un AS al proprio interno. Naturalmente, diversi AS possono utilizzare diversi IGP. Come abbiamo già visto, un IGP può anche essere gerarchico, in particolare quando le dimensioni dell'AS sono considerevoli;
- un secondo livello è dato dall'***Exterior Gateway Protocol*** (***EGP***), che è l'algoritmo che si usa per gestire il routing fra diversi AS. Tipicamente, in questo scenario EGP è l'algoritmo di routing che riguarda i router multiprotocollo. L'aspetto più interessante relativo a EGP è che esso deve spesso tener conto di specifiche leggi nazionali (ad esempio, divieto di far transitare dati sul suolo di una nazione ostile), per cui le decisioni di routing devono adattarsi a tali direttive.

5.6) Il livello network in Internet

Internet è una collezione di AS connessi gli uni con gli altri. Non esiste una struttura rigida, ma comunque si possono distinguere alcune componenti:

- backbone principali (linee ad alta velocità);
- reti regionali (USA);
- reti nazionali (Europa e resto del mondo);
- reti locali.

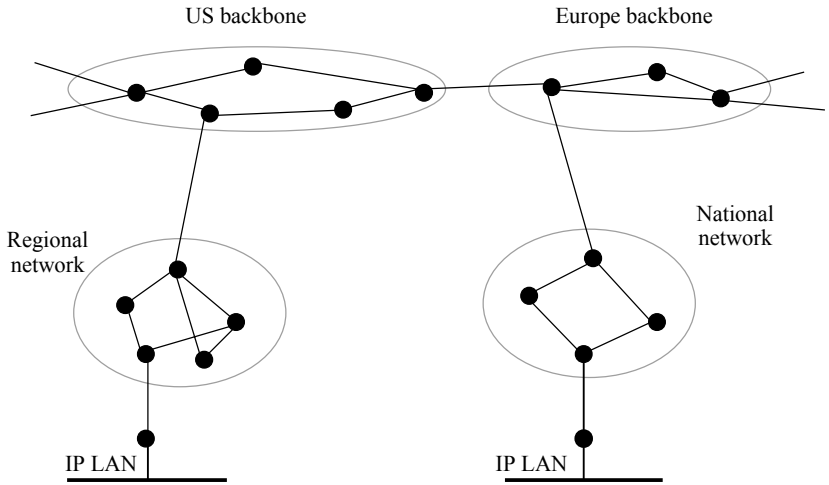


Figura 5-8: Organizzazione della rete Internet

Ciò che tiene tutto insieme è il protocollo di livello network dell'architettura TCP / IP, e cioè ***IP*** (***Internet Protocol***, RFC 791).

IP è un protocollo datagram, quindi non connesso e non affidabile, che opera come segue:

- riceve i dati dal livello transport e li incapsula in pacchetti di dimensione massima pari a 64 Kbyte (normalmente circa 1.500 byte);
- instrada i pacchetti sulla subnet, eventualmente frammentandoli lungo il viaggio;
- a destinazione:
  - riassembla (se necessario) i frammenti in pacchetti;
  - estrae da questi i dati del livello transport;
  - consegna al livello transport i dati nell'ordine in cui sono arrivati (che non è necessariamente quello in cui sono partiti).

5.6.1) Lo header IP (versione 4)

Un pacchetto IP è costituito da un ***header*** e da una parte dati.

L'header ha una parte fissa di 20 byte e una parte, opzionale, di lunghezza variabile.

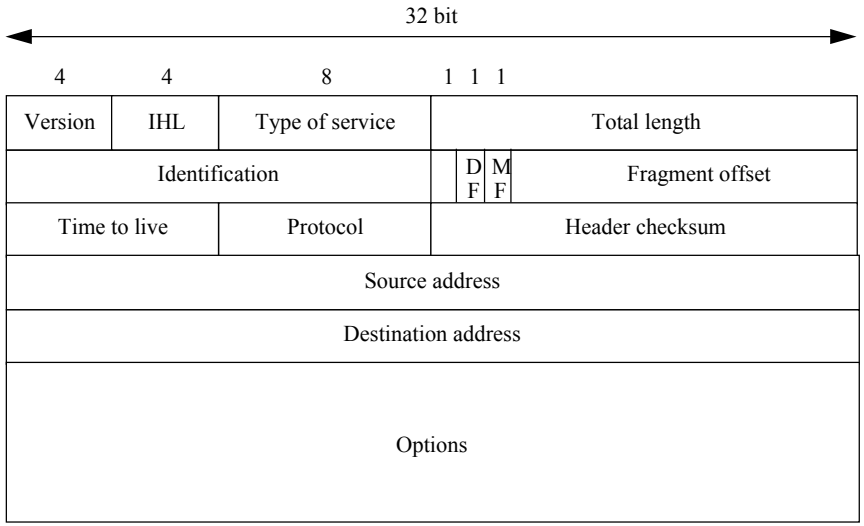


Figura 5-9: Formato dell'header IP

I campi dell'header hanno le seguenti funzioni:

Version	il numero di versione del protocollo (oggi è 4).
IHL	lunghezza dell'header in parole di 32 bit (minimo 5, massimo 15).
Type of service	caratterizza affidabilità e velocità richieste. E' di fatto ignorato dai router.
Total length	lunghezza del pacchetto (inclusi dati), massimo 65.535 byte.
Identification	tutti i frammenti di uno stesso pacchetto hanno lo stesso valore.
DF	***don't fragment*** (se uguale a 1, non si deve frammentare il pacchetto a costo di scegliere una strada meno veloce).
MF	***more fragments*** (se uguale a 1, il pacchetto non è ancora finito).
Fragment offset	indice del frammento nel pacchetto.
Time to live	contatore (inizializzato a 255) che viene decrementato di uno a ogni hop (o ad ogni secondo). Quando arriva a zero, il pacchetto viene scartato.
Protocol	codice del protocollo di livello transport a cui consegnare i dati (i codici sono definiti in RFC 1700).
Header checksum	checksum di controllo del solo header: • si sommano (in complemento ad uno) le parole a 16 bit dello header, considerando il checksum a zero; • si complementa ad uno il risultato; • viene ricalcolato ad ogni hop (time to live cambia).
Source e destination address	indirizzi di mittente e destinatario.
Options	opzioni, solo cinque sono definite oggi: • ***security***: quanto è segreto il pacchetto; • ***strict source routing***: cammino da seguire; • ***loose source routing***: lista di router da non mancare; • ***record route***: ogni router appende il suo indirizzo; • ***timestamp***: ogni router appende il suo indirizzo più un timestamp.

5.6.2) Indirizzi IP

Un indirizzo IP è formato da 32 bit e codifica due cose:

- ***network number***, cioé il numero assegnato alla rete IP (detta ***network***) su cui si trova l'elaboratore; in questo contesto una network è caratterizzata dal fatto di essere costituita da un unico canale di comunicazione cui sono connessi tutti gli host della network stessa (e quindi, ad esempio, una LAN oppure una linea punto punto fra due router);
- ***host number***, cioé il numero assegnato all'elaboratore.

La combinazione è unica: non possono esistere nell'intera rete Internet due indirizzi IP uguali.

Si noti che solitamente si ritiene che ogni host sulla rete abbia un singolo indirizzo IP. In realtà gli indirizzi sono assegnati alle interfacce di rete, quindi:

- se un host ha un'unica interfaccia di rete (come è il caso di un PC in LAN) allora ha un unico indirizzo IP;
- se un host ha X interfacce di rete (come è il caso di un router connesso ad una LAN e ad X-1 linee punto-punto) ha X indirizzi.

Gli indirizzi IP sono assegnati da autorità nazionali (***NIC***, ***Network Information Center***) coordinate a livello mondiale.

I formati possibili degli indirizzi sono i seguenti:

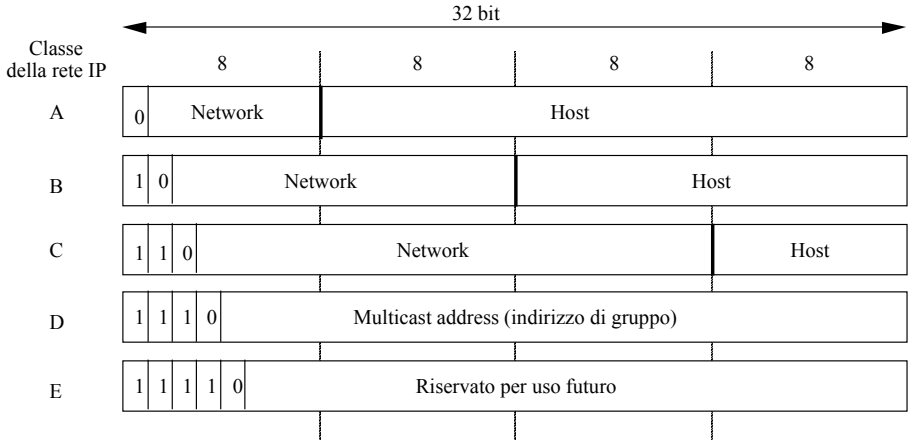


Figura 5-10: Formati degli indirizzi IP

Inoltre, esistono alcuni indirizzi con un significato speciale:

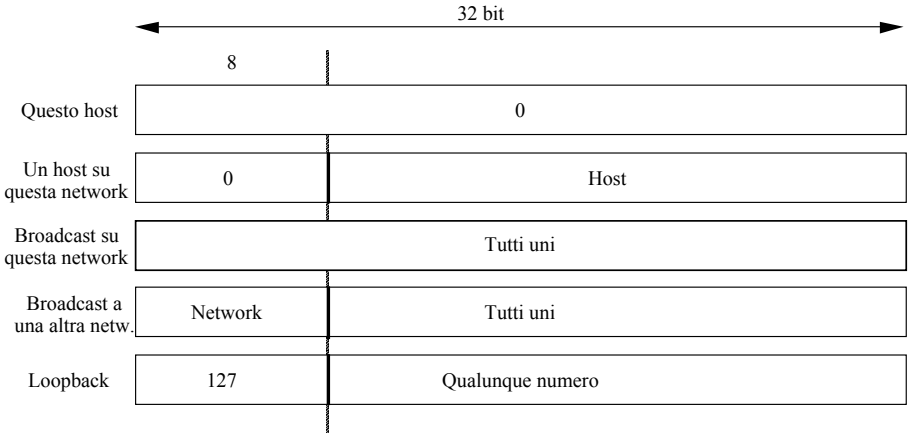


Figura 5-11: Indirizzi IP speciali

Quando si utilizza il ***loopback***, il pacchetto non viene inviato sulla rete ma viene elaborato come se fosse in arrivo: questo è molto utile, ad esempio, per effettuare localmente dei test su un software di rete in fase di sviluppo.

Ricapitolando, poiché alcune configurazioni binarie per gli indirizzi sono impegnate per gli indirizzi speciali, possono esistere:

- 126 network di classe A, le quali possono contenere 16 milioni di host ciascuna;
- 16382 network di classe B, con circa 64.000 host ciascuna;
- 2 milioni di network di classe C, con 254 host ciascuna.

Gli indirizzi sono usualmente espressi nella ***dotted decimal notation***, cioè i valori dei singoli byte sono espressi in decimale e sono separati da un punto, come nell'indirizzo:

```
141.192.140.37
```

In tale notazione, è possibile rappresentare separatamente il network number e l'host number. Per distinguerli, il primo è seguito da un punto. Ad esempio, nel caso dell'indirizzo IP precedente (che è relativo ad una network di tipo B), si ha:

- il network number è `141.192.` (notare il punto finale);
- l'host number è `140.37` (non c'è il punto finale).

5.6.3) Routing IP

Il collegamento fra due router non avviene direttamente, ma attraverso una network (in realtà di solito una ***subnet***, come vedremo poi) che li collega, e che di fatto è costituita dalle due interfacce di rete e dalla linea di comunicazione che le collega:

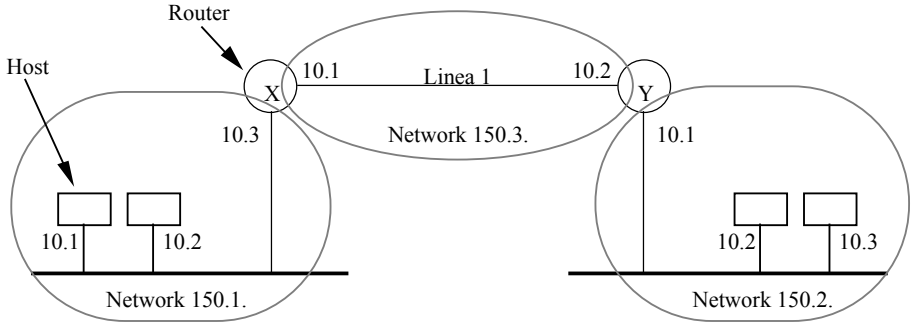


Figura 5-12: Collegamento di router IP

Ogni router possiede una tabella (costruita e mantenuta dinamicamente dall'algoritmo di routing in esercizio) che contiene elementi del tipo:

1. (this network number, host number) per ciascun host della network a cui il router è direttamente collegato;
2. (network number, 0) per ciascuna network lontana di cui il router conosce l'esistenza.

Associate a tali elementi ci sono le informazioni sull'interfaccia di rete da usare per instradare il pacchetto e, nel caso delle linee punto punto, l'indirizzo del router che si trova dall'altra parte.

Inoltre, viene mantenuto l'indirizzo di un ***default router*** (e la corrispondente linea seriale da usare per raggiungerlo) a cui inviare tutti i pacchetti destinati a network sconosciute.

Nell'esempio della figura precedente, il router X si comporta come segue:

- se arriva da fuori un pacchetto destinato a `150.1.10.1` il router, attraverso un elemento di tipo 1, scopre che deve inviarlo sulla LAN;
- se arriva dalla LAN un pacchetto per `150.2.10.3` e il router ha un elemento di tipo 2 quale (150.2., 0), allora invia il pacchetto (al router Y) sulla linea 1.

5.6.4) Subnet

Al fine di economizzare nel numero di network da usare (utilizzando al meglio quelle che possono contenere migliaia o milioni di host) una network può essere divisa in varie ***subnet***, ciascuna contenente i suoi host.

Questo è un fatto privato della network che viene suddivisa, e non ha bisogno di essere comunicato all'esterno.

Il meccanismo usato è di considerare, nell'indirizzo IP originario, l'host number come una coppia di valori: un ***subnet number*** e l'host number. Ciò avviene sulla base di una maschera di bit, detta ***subnet mask***, che deve essere unica per tutta la network e che delimita la parte di host number che viene usata come subnet number.

Ad esempio, per una rete di classe A si potrà avere:

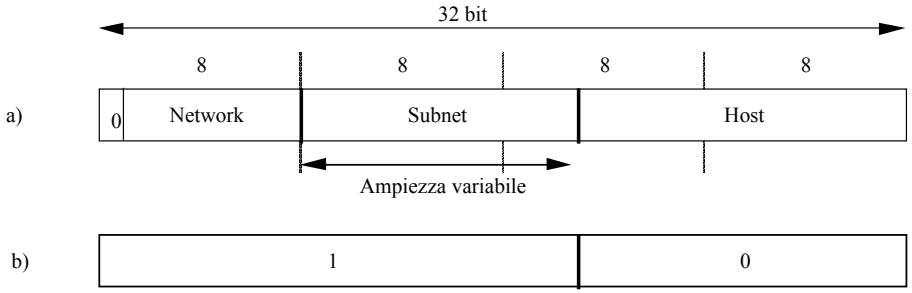


Figura 5-13: Indirizzo IP con subnet (a) e relativa subnet mask (b)

A seconda dell'ampiezza del campo dedicato alla subnet, si possono ottenere molte subnet contenenti ciascuna pochi host oppure poche subnet che però potranno contenere molti host.

Nei router si aggiungono elementi del tipo:
- (this network number, this subnet number, host number) per gli host delle subnet a cui il router è collegato direttamente;
- (this network number, subnet number, 0) per le altre subnet a cui il router non è collegato direttamente;

con le relative informazioni sulle interfacce di rete da usare.

La rete di figura 5-12, che nella sua formulazione originaria impegnava ben tre network di classe B, può essere realizzata con un sola network di classe B suddivisa in tre subnet. A tal fine, usiamo un network number uguale a 150.1. ed una subnet mask (espressa in dotted decimal notation) uguale a 255.255.255.0, cioé:
- i primi due byte dell'indirizzo identificano la network;
- il terzo byte identifica la subnet;
- il quarto byte identifica l'host.

Il risultato è il seguente:

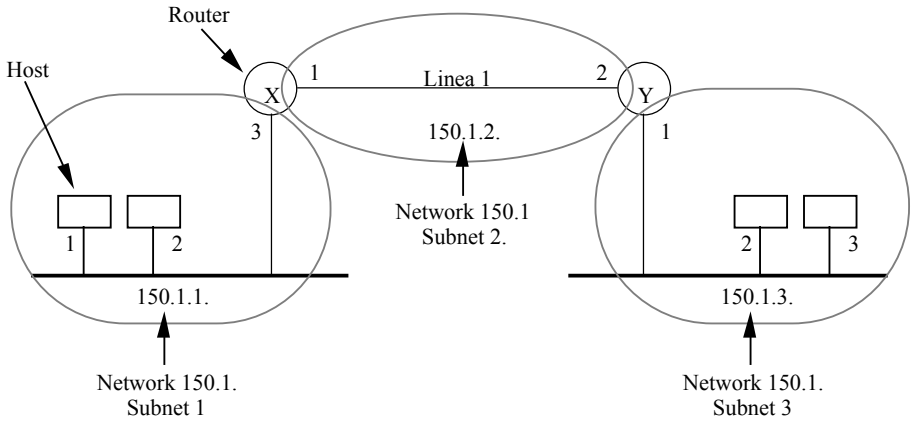


Figura 5-14: Collegamento di router IP mediante uso di subnet

5.6.5) Protocolli di controllo

Assieme a IP esistono diversi protocolli per il controllo del funzionamento della subnet.

ICMP (Internet Control Message Protocol, RFC 792)
L'operatività della subnet è controllata continuamente dai router, che si scambiano informazioni mediante messaggi conformi al protocollo ***ICMP*** (tali messaggi viaggiano dentro pacchetti IP).

Esistono molti tipi di messaggi, fra i quali:
- ***destination unreachable***: non si trova la destinazione del pacchetto. Viene inviato al mittente del pacchetto;
- ***time exceeded***: il contatore di un pacchetto è arrivato a zero. Viene inviato al mittente del pacchetto;

- ***redirect***: il router ha ragione di pensare che il pacchetto gli è arrivato per errore, ad esempio perché un host mobile si è spostato. Viene inviato al mittente del pacchetto;
- ***echo request, reply***: si vuole sapere se una destinazione è viva e raggiungibile. Si invia request, si aspetta reply;
- ***timestamp request, reply***: come il precedente, con in più la registrazione dell'istante di arrivo e partenza, per misurare le prestazioni della rete.

ARP (Address Resolution Protocol, RFC 826)
Il protocollo ***ARP*** serve per derivare, dall'indirizzo IP dell'host di destinazione, l'indirizzo di livello data link necessario per inviare il frame che incapsulerà il pacchetto destinato all'host di cui all'indirizzo IP.

Esso opera appoggiandosi direttamente sul livello data link e non su IP:
- viene inviata a tutte le stazioni della LAN, in data link broadcast, una richiesta del tipo: "chi ha l'indirizzo IP uguale a 151.100.17.102 ?"
- solo l'host che ha quell'indirizzo IP risponde, inserendo nella risposta il proprio indirizzo data link;
- quando riceve la risposta, l'host la mantiene in memoria per circa 15 minuti.

Se l'indirizzo IP è relativo ad un'altra network:
- la soluzione più semplice è mandare il pacchetto ARP come prima, configurando però il router in modo che risponda alle richieste ARP relative ad altre reti fornendo il proprio indirizzo ethernet(***proxy ARP***); il router farà poi da tramite nella conversazione IP fra il mittente e il destinatario, inviando di volta in volta all'uno i pacchetti IP che gli giungono dall'altro;
- alternativamente, si configura l'host impostando al suo interno l'indirizzo ethernet di default (quello del router) a cui mandare i pacchetti IP per le altre reti; anche in questo caso il router deve fare da tramite nella conversazione IP fra il mittente e il destinatario.

RARP (Reverse Address Resolution Protocol, RFC 903)
Il protocollo ***RARP*** risolve il problema inverso, cioè consente di trovare quale indirizzo IP corrisponda a un determinato indirizzo data link.

Esso è utile nel caso di stazioni senza disco, che al momento dell'avvio caricano l'immagine del codice binario del sistema operativo da un server.

Il vantaggio che si ottiene è che una sola immagine binaria va bene per tutte le stazioni: ogni stazione personalizza poi l'immagine binaria con la determinazione del proprio indirizzo IP mediante una richiesta RARP, nella quale invia il proprio indirizzo data link (che è cablato nell'interfaccia di rete).

5.6.6) Protocolli di routing

Come già detto, Internet è una collezione di AS connessi da backbone ad alta velocità.

Ciò che caratterizza un AS è il fatto di essere gestito da una singola autorità.

Esempi tipici di AS sono:
- la rete degli enti di ricerca di una nazione (rete GARR in Italia);
- la rete di una singola azienda.

Il routing complessivo è organizzato in modo gerarchico:
- all'interno di un singolo AS si usa un solo Interior Gateway Protocol (IGP);
- viceversa, per il routing fra gli AS si usa un Exterior Gateway Protocol (EGP).

Interior Gateway Protocol per Internet
Il protocollo originario era il ***RIP*** (***Routing Information Protocol***, RFC 1058) di tipo distance vector, ormai sostituito da ***OSPF*** (***Open Shortest Path First***, RFC 1247), che è di tipo link state.

OSPF consente fra l'altro un routing gerarchico all'interno dell'AS, che viene suddivisa in diverse aree:
- ***backbone area*** (che è connessa a tutte le altre)
- altre aree

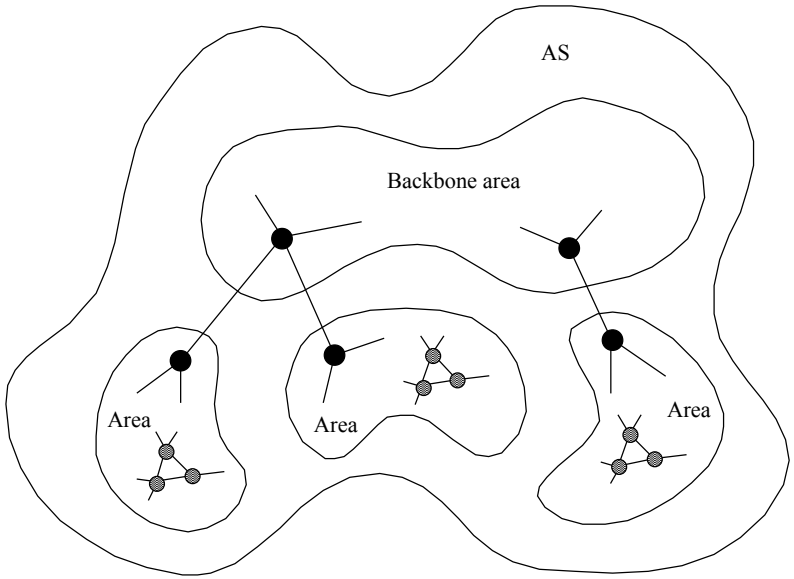


Figura 5-15: Suddivisione di un AS in aree

I router di un AS possono essere:

- ***interni a un area***: si occupano del routing all'interno dell'area;
- ***sul confine di un'area***: si occupano del routing fra le diverse aree via backbone;
- ***nell'area backbone***: si occupano del routing sul backbone;
- ***sul confine dell'AS***: si occupano del routing fra gli AS (applicando EGP); possono trovarsi anche sul confine di un'area.

Exterior Gateway Protocol per Internet
Il protocollo EGP, usato dai router sul confine dell'AS e da quelli sui backbone ad alta velocità che connettono gli AS, si chiama ***BGP*** (***Border Gateway Protocol***, RFC 1654).

E' fondamentalmente di tipo distance vector, con due novità:
- possiede la capacità di gestire politiche di instradamento (derivanti, ad esempio, da leggi nazionali) che vengono configurate manualmente nei router;
- mantiene (e scambia con gli altri router) non solo il costo per raggiungere le altre destinazioni, ma anche il cammino completo. Ciò consente di risolvere il problema del count to infinity, perché se una linea va giù il router può subito scartare, senza quindi poi distribuirli, tutti i cammini che ci passano.

5.6.7) IP versione 6

Dopo un lungo lavoro, IETF ha approvato il successore di IP versione 4, cioè la versione 6 (***IPv6***, RFC 1883 - 1887).

I requisiti principali di progetto erano:
- aumentare il numero di indirizzi, ormai quasi esauriti;
- ottenere una maggiore efficienza nei router (tavole più piccole, routing più veloce);
- supportare meglio il traffico real time;
- offrire maggiore sicurezza ai dati riservati.

Le principali differenze rispetto alla versione 4 sono:
- indirizzi di 16 byte, il che significa disporre di 2^{128} indirizzi IP, e cioé $7*10^{23}$ indirizzi IP per metro quadro su tutto il nostro pianeta;
- header semplificato: 7 campi contro 13;
- funzioni di autentificazione e privacy, basate su crittografia;
- gestione della qualità di servizio attraverso un campo ***flow label***, che consente di istituire delle pseudoconnessioni con caratteristiche negoziate in anticipo.

6) Il livello quattro (Transport)

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

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

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

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

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

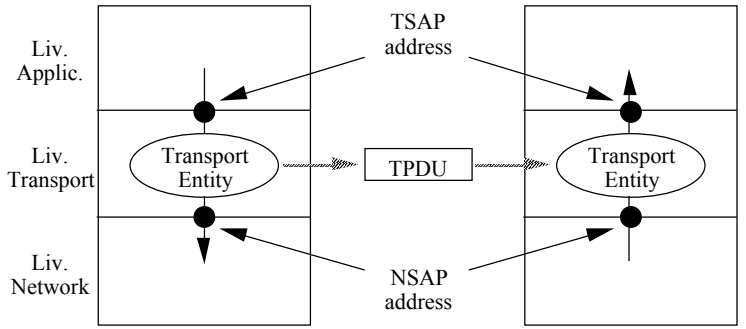


Figura 6-1: I servizi transport sono basati sui servizi network

Così come ci sono due tipi di servizi di livello network, ce ne sono due anche a livello transport:
- servizi affidabili orientati alla connessione (tipici di questo livello);
- servizi datagram (poco usati in questo livello).

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

Ma allora, perché duplicare le cose in due diversi livelli? Il fatto è che l'utente (che accede ai servizi di rete dall'alto) non ha alcun controllo sulla subnet di comunicazione, e vuole comunque certe garanzie di servizio (ad esempio, il trasferimento corretto di un file). Dunque, tali garanzie devono essere fornite al di fuori della subnet, per cui devono risiedere in un livello superiore a quello network. In sostanza, il livello transport permette di offrire un servizio più affidabile di quanto la subnet sia in grado di fare.

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

Un ultimo aspetto riguarda la possibilità di specificare la ***QoS*** (***Quality of Service***) desiderata. Questo in particolare è adatto soprattutto ai servizi connection oriented, nei quali il richiedente può specificare esigenze quali:
- massimo ritardo per l'attivazione della connessione;
- throughput richiesto;
- massimo ritardo di transito ammesso;
- tasso d'errore tollerato;
- tipo di protezione da accessi non autorizzati ai dati in transito.

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

Primitive di definizione del servizio
Esse definiscono il modo di accedere ai servizi del livello.

Tipicamente sono progettate in modo da nascondere i dettagli della subnet (di più, in modo da essere indipendenti da qualunque subnet) ed essere facili da usare.

Ad esempio, questo può essere un tipico insieme di primitive:

Primitiva	TPDU spedito	Note
`accept()`	-	Si blocca finché qualcuno cerca di connettersi
`connect()`	`conn.request`	Cerca di stabilire una connessione
`send()`	`dati`	Invia dei dati
`receive()`	-	Si blocca finché arriva un TPDU
`disconnect()`	`disconn.request`	Chiede di terminare la

		connessione

Ad esempio, consideriamo i seguenti frammenti di codice di un'applicazione client-server:

Server	Client

`...`
`accept();`
`send();`
`receive();`
`...`
	`...`
	`connect();`
	`receive();`
	`send();`
	`...`
	`disconnect();`

Questi due frammenti sono in grado di portare avanti un dialogo senza errori (se il servizio utilizzato è affidabile, il che è la norma), ignorando completamente tutto quanto riguarda la meccanica della comunicazione.

6.1) Protocolli di livello transport

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

Ci sono però anche delle importanti differenze. Quella principale è che:
- nel livello data link fra le peer entity c'è un singolo canale di comunicazione;
- nel livello transport c'è di mezzo l'intera subnet di comunicazione.

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

6.2) Indirizzamento

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

Tipicamente un TSAP address ha la forma

(NSAP address, informazione supplementare)

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

(IP address:port number)

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

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

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

6.3) Attivazione della connessione

Questa operazione sembra facile ma non lo è, perché la subnet può perdere o duplicare (a causa di ritardi interni) dei pacchetti.

Ad esempio, a causa di ripetuti ritardi nell'invio degli ack può succedere che vengano duplicati e successivamente arrivino in perfetto ordine a destinazione tutti i pacchetti precedentemente generati nel corso di una connessione. Ciò in linea di principio può significare che l'attivazione della connessione e tutto il suo svolgimento abbiano luogo due volte.

Si immaginino le conseguenze di tale inconveniente se lo scopo di tale connessione fosse stato la richiesta (a una banca) di versare un miliardo sul conto di un personaggio dalla dubbia onestà.

Il problema di fondo risiede nella possibile esistenza di ***duplicati ritardatari*** che arrivano a destinazione molto dopo essere partiti.

Già sappiamo che, una volta che la connessione è stabilita, un qualunque protocollo a finestra scorrevole è adatto allo scopo. Infatti durante il setup della connessione le peer entity si accordano sul numero iniziale di sequenza, e quindi i doppioni ritardatari non vengono accettati.

Viceversa, per risolvere il problema dei duplicati relativi alla fase di attivazione della connessione esiste una soluzione detta ***three-way handshaking*** (Tomlinson, 1975).

Il protocollo funziona così:

- il richiedente invia un TPDU di tipo `conn.request` con un numero x proposto come inizio della sequenza;
- il destinatario invia un TPDU di tipo ack contenente:
  - la conferma di x;
  - la proposte di un proprio numero y di inizio sequenza;
- il richiedente invia un TPDU di tipo dati contenente:
  - i primi dati del dialogo;
  - la conferma di y.

I valori x e y possono essere generati, ad esempio, sfruttando l'orologio di sistema, in modo da avere valori ogni volta diversi.

In assenza di errori, il funzionamento è il seguente:

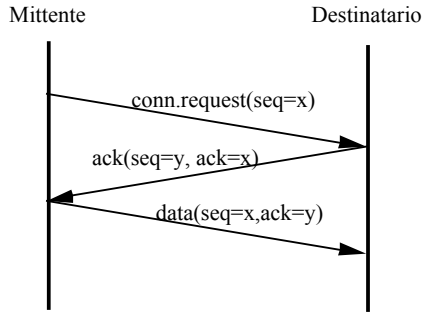


Figura 6-2: Three-way handshake

Se arriva a destinazione un duplicato della richiesta di attivazione, il destinatario risponde come prima ma il mittente, che sa di non aver richiesto una seconda connessione, lo informa dell'errore:

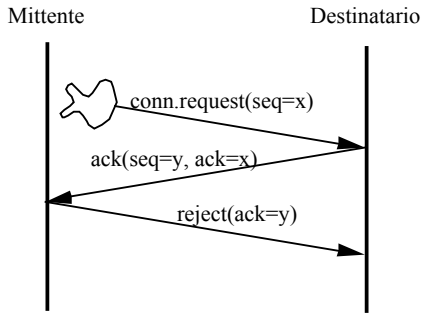


Figura 6-3: Duplicato della richiesta di attivazione

Se infine arrivano al destinatario sia un duplicato della richiesta di attivazione che un duplicato del primo TPDU dati, la situazione è la seguente:

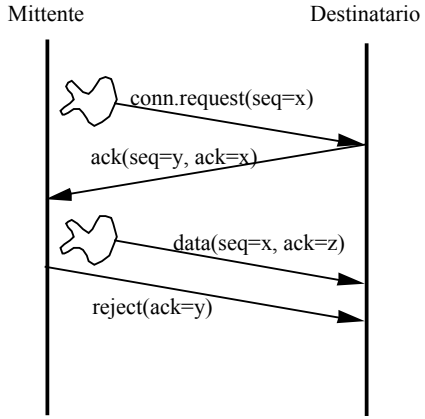


Figura 6-4: Duplicati della richiesta di attivazione e del primo TPDU dati

Infatti, in questo caso:

- il mittente invia un reject alla risposta del destinatario, perché sa di non aver richiesto una seconda connessione (come nel caso precedente);
- il destinatario scarta il TPDU dati, perché questo reca un ack relativo ad un numero di sequenza (z) precedente e non a quello (y) da lui testé inviato.

6.4) Rilascio di una connessione

Rilasciare una connessione è più semplice che stabilirla, ma comunque qualche piccolo problema c'è anche in questa fase.

In questo contesto, rilasciare la connessione significa che l'entità di trasporto rimuove le informazioni sulla connessione dalle proprie tavole e informa l'utente di livello superiore che la connessione è chiusa.

Ci sono due tipi di rilasci:

- ***asimmetrico***;
- ***simmetrico***.

Nel primo caso (esemplificato dal sistema telefonico) quando una delle due parti "mette giù" si chiude immediatamente la connessione. Ciò però può portare alla perdita di dati, in particolare di tutti quelli che l'altra parte ha inviato e non sono ancora arrivati.

Nel secondo caso si considera la connessione come una coppia di connessioni unidirezionali, che devono essere rilasciate indipendentemente. Quindi, lungo una direzione possono ancora scorrere dei dati anche se la connessione lungo l'altra direzione è stata chiusa. Il rilascio simmetrico è utile quando un processo sa esattamente quanti dati deve spedire, e quindi può autonomamente decidere quando rilasciare la sua connessione in uscita.

Se invece le due entità vogliono essere d'accordo prima di rilasciare la connessione, un modo di raggiungere lo scopo potrebbe essere questo:

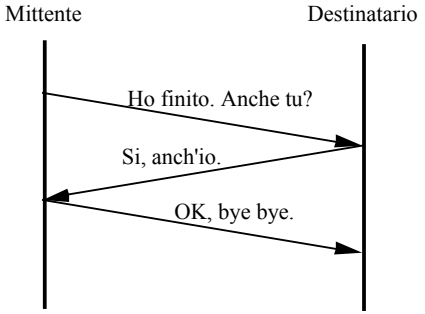


Figura 6-5: Semplice schema per il rilascio concordato della connessione

Purtroppo, le cose non sono così semplici: c'è un problema famoso in proposito, il ***problema delle due armate***:

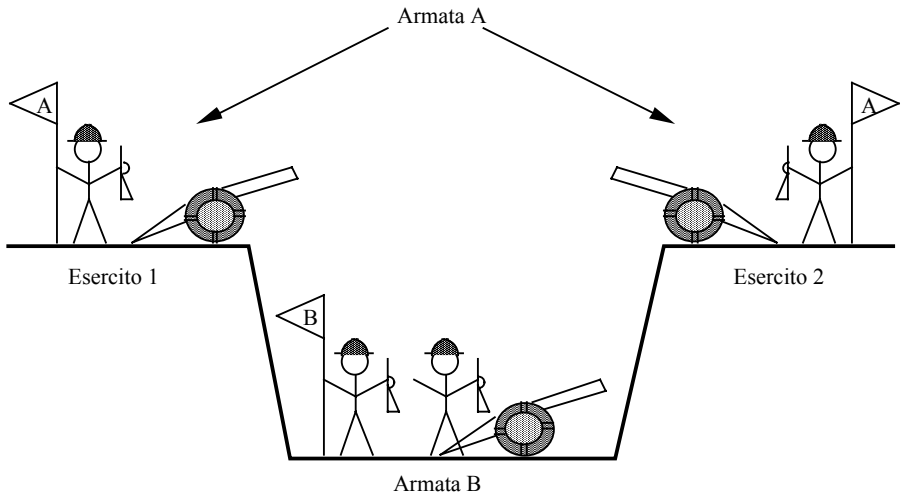


Figura 6-6: Il problema delle due armate

La definizione del problema è la seguente:

- i due eserciti che compongono l'armata A sono ciascuno più debole dell'esercito che costituisce l'armata B;
- l'armata A però nel suo complesso è più forte dell'armata B;
- i due eserciti dell'armata A possono vincere solo se attaccano contemporaneamente;
- i messaggi fra gli eserciti dell'armata A sono portati da messaggeri che devono attraversare il territorio dell'armata B, dove possono essere catturati.

Come fanno ad accordarsi gli eserciti dell'armata A sull'ora dell'attacco? Una possibilità è la seguente:

- il comandante dell'esercito 1 manda il messaggio "attacchiamo a mezzanotte. Siete d'accordo?";
- il messaggio arriva, un ok di risposta parte e arriva a destinazione, ma il comandante dell'esercito 2 esita perché non può essere sicuro che la sua risposta sia arrivata.

Si potrebbe pensare di risolvere il problema con un passaggio in più (ossia con un three-way handshake): l'arrivo della risposta dell'esercito 2 deve essere a sua volta confermato. Ora però chi esita è il comandante dell'esercito 1, perché se tale conferma si perde, l'armata 2 non saprà che la sua conferma alla proposta di attaccare è arrivata e quindi non attaccherà.

Aggiungere ulteriori passaggi non aiuta, perché c'è sempre un messaggio di conferma che è l'ultimo, e chi lo spedisce non può essere sicuro che sia arrivato. Dunque, non esiste soluzione.

Se ora sostituiamo la frase "attacchiamo l'armata B" con "rilasciamo la connessione", vediamo che si rischia effettivamente di non rilasciare mai una connessione.

Per fortuna, rilasciare una connessione è meno critico che attaccare un'armata nemica. Quindi, qualche rischio si può anche prendere.

Un protocollo di tipo three-way handshaking arricchito con la gestione di timeout è considerato adeguato, anche se non infallibile:

- il mittente invia un `disconn.request` e, se non arriva risposta entro un tempo prefissato (timeout), lo invia nuovamente per un massimo di n volte:
  - appena arriva una risposta (`disconn.request`) rilascia la connessione in ingresso e invia un ack di conferma;
  - se non arriva nessuna risposta, dopo l'ultimo timeout rilascia comunque la connessione in ingresso;
- il destinatario, quando riceve `disconn.request`, fa partire un timer, invia a sua volta un `disconn.request` e attende l'ack di conferma. Quando arriva l'ack o scade il timer, rilascia la connessione in ingresso.

Normalmente il funzionamento è il seguente:

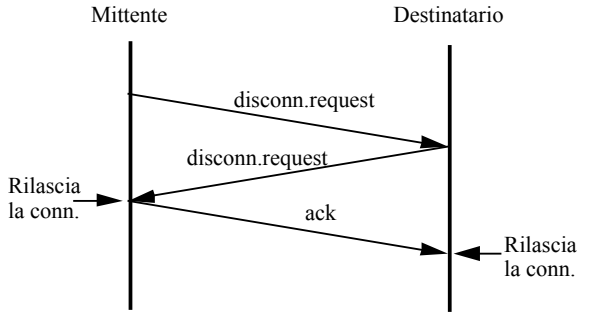


Figura 6-7: Rilascio concordato di una connessione transport

Se si perde l'ack:

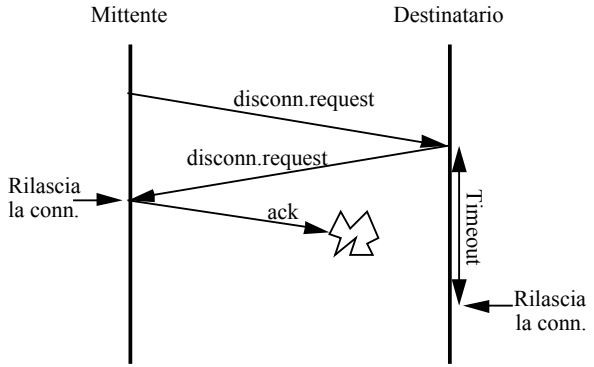


Figura 6-8: Perdita dell'ack durante il rilascio della connessione

Se invece si perde la risposta alla prima proposta di chiusura (supponendo che il timeout del destinatario sia molto più ampio di quello del mittente):

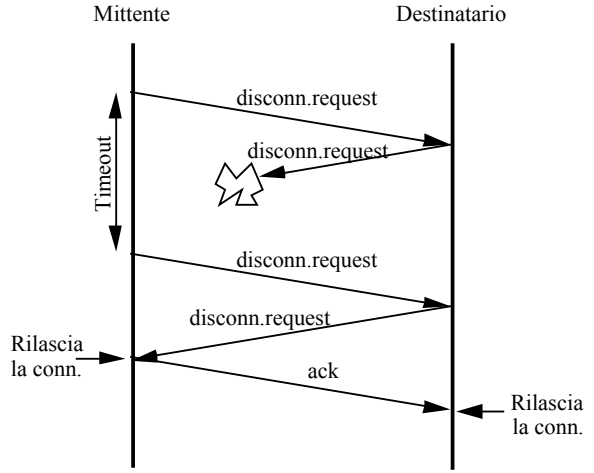


Figura 6-9: Perdita della risposta alla proposta di chiusura durante il rilascio della connessione

Infine, si può perdere la risposta alla prima proposta di chiusura e tutte le successive proposte di chiusura:

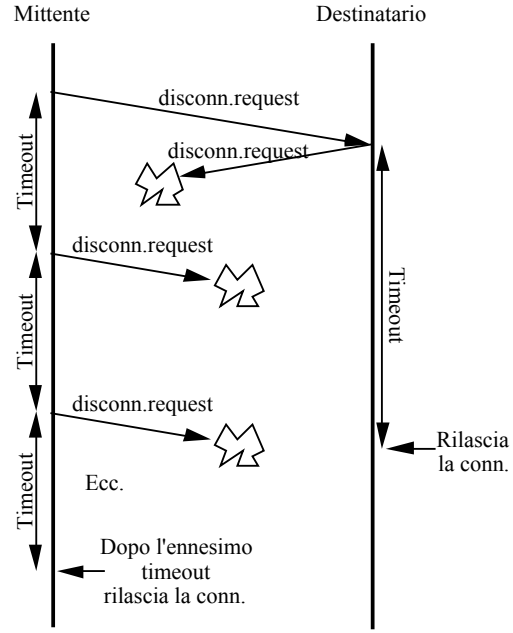


Figura 6-10: Perdita di tutti i messaggi tranne il primo

Il protocollo fallisce se tutte le trasmissioni di `disconn.request` della peer entity che inizia la procedura si perdono: in tal caso essa rilascia la connessione, ma l'altra entity non se ne accorge e la tiene aperta. Il risultato è una ***half-open connection***.

Un modo per risolvere il problema è che le parti rilascino una connessione quando passa un certo lasso di tempo (ad es. 60 secondi) senza che arrivino dati. Naturalmente, in tal caso, i processi devono scambiarsi dei ***dummy TPDU***, cioé dei TPDU vuoti, con una certa frequenza anche se non c'è necessità di comunicare alcunché, per evitare che la connessione cada.

6.5) Controllo di flusso e buffering

Per alcuni aspetti il controllo di flusso a livello transport è simile a quello di livello data link: è necessario un meccanismo, quale la gestione di una sliding window, per evitare che il ricevente sia sommerso di TPDU.

Però ci sono anche importanti differenze:

- il livello data link non ha alcun servizio, tranne la trasmissione fisica, a cui appoggiarsi. Il livello transport invece usa i servizi del livello network, che possono essere affidabili o no, e quindi può organizzarsi di conseguenza;
- il numero di connessioni data link è relativamente piccolo (uno per linea fisica) e stabile nel tempo, mentre le connessioni transport sono potenzialmente molte e di numero ampiamente variabile nel tempo;
- le dimensioni dei frame sono piuttosto stabili, quelle dei TPDU sono molto più variabili.

Se il livello transport dispone solo di servizi di livello network di tipo datagram:

- la transport entity sorgente deve mantenere in un buffer i TPDU spediti finché non sono confermati, come si fa nel livello data link;
- la transport entity di destinazione può anche non mantenere dei buffer specifici per ogni connessione, ma solo un pool globale: quando arriva un TPDU, se non c'è spazio nel pool, non lo accetta. Poiché il mittente sa che la subnet può perdere dei pacchetti, riproverà a trasmettere il TPDU finché non arriva la conferma. Al peggio, si perde un pò di efficienza.

Viceversa, ove siano disponibili servizi network di tipo affidabile ci possono essere altre possibilità per il livello transport:

- se il mittente sa che il ricevente ha sempre spazio disponibile nei buffer (e quindi è sempre in grado di accettare i dati che gli arrivano) può evitare di mantenere lui dei buffer, dato che ciò che spedisce:
    - arriva sicuramente a destinazione;
    - viene sempre accettato dal destinatario;
- se non c'è questa garanzia, il mittente deve comunque mantenere i buffer, dato che il destinatario riceverà sicuramente i TPDU spediti, ma potrebbe non essere in grado di accettarli.

Un problema legato al buffering è come gestirlo, vista la grande variabilità di dimensione dei TPDU:

- un pool di buffer tutti uguali: poco adatto, dato che comporta uno spreco di spazio per i TPDU di piccole dimensioni e complicazioni per quelli grandi;
- un pool di buffer di dimensioni variabili: più complicato da gestire ma efficace;
- un singolo array (piuttosto grande) per ogni connessione, gestito circolarmente: adatto per connessioni gravose, comporta spreco di spazio per quelle con poco scambio di dati.

In generale, data la grande variabilità di condizioni che si possono verificare, conviene adottare regole diverse di caso in caso:

- traffico bursty ma poco gravoso (ad esempio in una sessione di emulazione di terminale, che genera solo caratteri): niente buffer a destinazione, dove i TPDU si possono acquisire senza problemi man mano che arrivano. Di conseguenza, bastano buffer alla sorgente;
- traffico gravoso (ad esempio file transfer): buffer cospicui a destinazione, in modo da poter sempre accettare ciò che arriva.

Per gestire al meglio queste situazioni, di norma i protocolli di livello transport consentono alle peer entity di mettersi d'accordo in anticipo (al setup della connessione) sui meccanismi di bufferizzazione da usare.

Inoltre, spesso la gestione della dimensione delle finestre scorrevoli è disaccoppiata dalla gestione degli ack (vedremo il caso di TCP).

6.6) Multiplexing

Una importante opportunità è data dal multiplexing delle conversazioni di livello transport su quelle di livello network.

Se le connessioni di livello network (in una rete che le offre) sono costose da istituire, allora è conveniente convogliare molte conversazioni transport su un'unica connessione network (***upward multiplexing***).

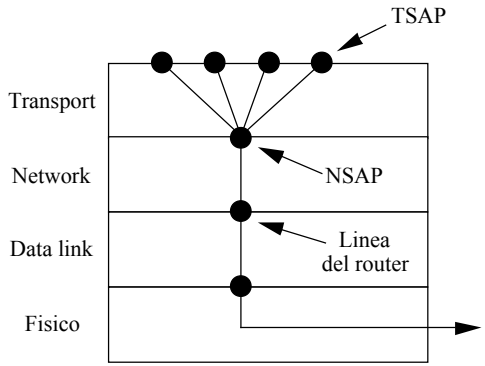


Figura 6-11: Upward multiplexing

Viceversa, se si vuole ottenere una banda superiore a quella consentita a una singola connessione network, allora si può guadagnare banda ripartendo la conversazione transport su più connessioni network (***downward multiplexing***).

Ciò può essere utile, ad sempio, in una situazione dove:
- la subnet applica a livello network un controllo di flusso sliding window, con numeri di sequenza a 8 bit;
- il canale fisico è satellitare ed ha 540 msec di round-trip delay.

Supponendo che la dimensione dei pacchetti sia di 128 byte, il mittente può spedire al massimo $2^8 - 1$ pacchetti ogni 540 msec, e quindi ha a disposizione una banda di 484 kbps, anche se la banda del canale satellitare è tipicamente 100 volte più grande.

Se la conversazione transport è suddivisa su, ad esempio, 10 connessioni network, essa potrà disporre di una banda 10 volte maggiore.

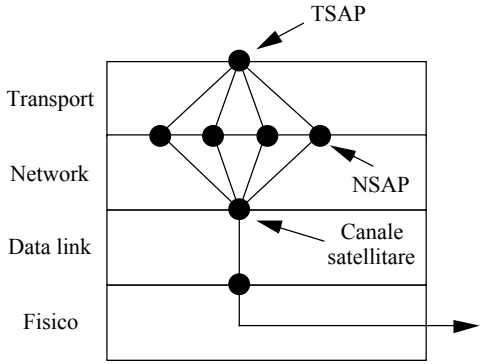


Figura 6-12: Downward multiplexing

6.7) Il livello transport in Internet

Il livello transport di Internet è basato su due protocolli:
- ***TCP*** (***Transmission Control Protocol***) RFC 793, 1122 e 1323;
- ***UDP*** (***User Data Protocol***) RFC 768.

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

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

Dunque, offre un servizio reliable e connection oriented, e si occupa di:
- accettare dati dal livello application;
- spezzarli in ***segment***, il nome usato per i TPDU (dimensione massima 64 Kbyte, tipicamente circa 1.500 byte);
- consegnarli al livello network, eventualmente ritrasmettendoli;
- ricevere segmenti dal livello network;
- rimetterli in ordine, eliminando buchi e doppioni;
- consegnare i dati, in ordine, al livello application.

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

6.7.1) Indirizzamento

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

IP address: Port number

Il socket number costituisce il TSAP.

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

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

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

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

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

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

C'è comunque la possibilità, per il livello application, di forzare l'invio immediato di dati; ciò causa l'invio di un flag ***urgent*** che, quando arriva dall'altra parte, fa sì che l'applicazione venga interrotta e si dedichi a esaminare i dati urgenti (questo succede

quando, ad esempio, l'utente durante una sessione di emulazione di terminale digita il comando ABORT (CTRL-C) della computazione corrente).

6.7.2) Il protocollo TCP

Le caratteristiche più importanti sono le seguenti:

- ogni byte del flusso TCP è numerato con un numero d'ordine a 32 bit, usato sia per il controllo di flusso che per la gestione degli ack;
- un segmento TCP non può superare i 65.535 byte;
- un segmento TCP è formato da:
  - uno header, a sua volta costituito da:
    - una parte fissa di 20 byte;
    - una parte opzionale;
  - i dati da trasportare;
- TCP usa un meccanismo di sliding window di tipo go-back-n con timeout. Se questo scade, il segmento si ritrasmette. Si noti che le dimensioni della finestra scorrevole e i valori degli ack sono espressi in numero di byte, non in numero di segmenti.

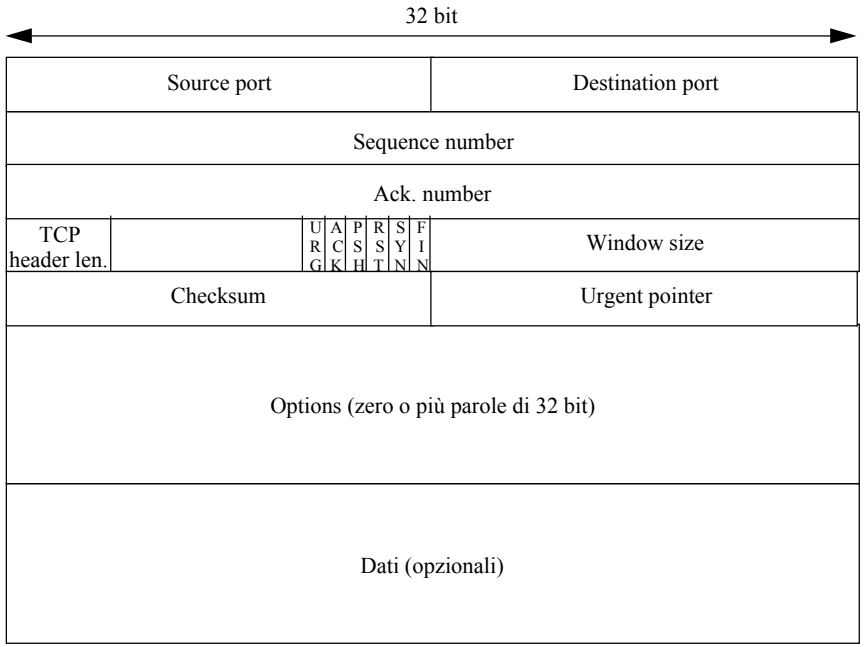


Figura 6-13: Formato del segmento TCP

I campi dell'header hanno le seguenti funzioni:

Source port, destination port	identificano gli end point (locali ai due host) della connessione. Essi, assieme ai corrispondenti numeri IP, formano i due TSAP.
Sequence number	il numero d'ordine del primo byte contenuto nel campo dati.
Ack. number	il numero d'ordine del prossimo byte aspettato.
TCP header length	quante parole di 32 bit ci sono nell'header (necessario perché il campo options è di dimensione variabile).
URG	1 se urgent pointer è usato, 0 altrimenti.
ACK	1 se l'ack number è valido (cioè se si convoglia un ack), 0 altrimenti.
PSH	dati urgenti (***pushed data***), da consegnare senza aspettare che il buffer si riempia.
RST	richiesta di reset della connessione (ci sono problemi!).
SYN	usato nella fase di setup della connessione: • SYN=1 ACK=0 richiesta connessione; • SYN=1 ACK=1 accettata connessione.
FIN	usato per rilasciare una connessione.
Window size	il controllo di flusso è di tipo sliding window di dimensione variabile. Window size dice quanti byte possono essere spediti a partire da quello (compreso) che viene confermato con l'ack number. Un valore zero significa: fermati per un pò, riprenderai quando ti arriverà un uguale ack number con un valore di window size diverso da zero.
Checksum	simile a quello di IP; il calcolo include uno pseudoheader.
Urgent pointer	puntatore ai dati urgenti.
Options	fra le più importanti, negoziabili al setup: • dimensione massima dei segmenti da spedire; • uso di selective repeat invece che go-back-n; • uso di NAK.

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

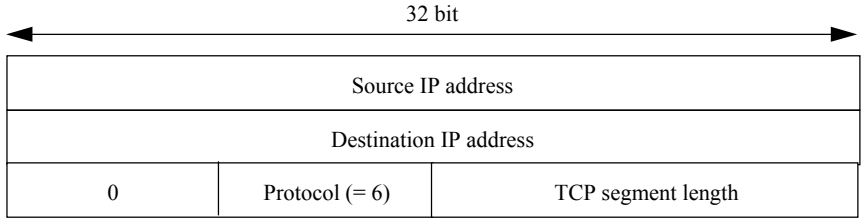


Figura 6-14: Formato dello pseudoheader TCP

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

Source IP address, destination IP address	indirizzi IP (a 32 bit) di sorgente e destinatario.
Protocol	il codice numerico del protocollo TCP (pari a 6).
TCP segment length	il numero di byte del segmento TCP, header compreso.

6.7.3) Attivazione della connessione

Si usa il three-way handshake visto precedentemente:

- una delle due parti (diciamo il server) esegue due primitive, `listen()` e poi `accept()` rimanendo così in attesa di una richiesta di connessione su un determinato port number e, quando essa arriva, accettandola;
- l'altra parte (diciamo un client) esegue la primitiva `connect()`, specificando host, port number e altri parametri quali la dimensione massima dei segmenti, per stabilire la connessione; tale primitiva causa l'invio di un segmento TCP col bit `syn` a uno e il bit `ack` a zero;
- quando tale segmento arriva a destinazione, l'entity di livello transport controlla se c'è un processo in ascolto sul port number in questione:
  - se non c'è nessuno in ascolto, invia un segmento di risposta col bit `rst` a uno, per rifiutare la connessione;
  - altrimenti, consegna il segmento arrivato al processo in ascolto; se esso accetta la connessione, l'entity invia un segmento di conferma, con entrambi i bit `syn` ed `ack` ad uno, secondo lo schema sotto riportato.

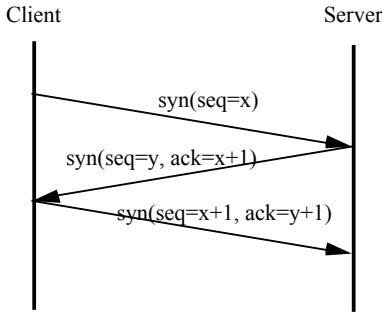


Figura 6-15: Attivazione di una connessione TCP

I valori di x e y sono ricavati dagli host sulla base dei loro clock di sistema; il valore si incrementa di una unità ogni 4 microsecondi.

6.7.4) Rilascio della connessione

Il rilascio della connessione avviene considerando la connessione full-duplex come una coppia di connessioni simplex indipendenti, e si svolge nel seguente modo:

- quando una delle due parti non ha più nulla da trasmettere, invia un `fin`;
- quando esso viene confermato, la connessione in uscita viene rilasciata;
- quando anche l'altra parte completa lo stesso procedimento e rilascia la connessione nell'altra direzione, la connessione full-duplex termina.

Per evitare il problema dei 3 eserciti si usano i timer, impostati al doppio della vita massima di un pacchetto.

Il protocollo di gestione delle connessioni si rappresenta comunemente come una ***macchina a stati finiti***. Questa è una rappresentazione molto usata nel campo dei protocolli, perché permette di definire, con una certa facilità e senza ambiguità, protocolli anche molto complessi.

6.7.5) Politica di trasmissione

L'idea di fondo è la seguente: la dimensione delle finestre scorrevoli non è strettamente legata agli ack (come invece di solito avviene), ma viene continuamente adattata mediante un dialogo fra destinazione e sorgente.

In particolare, quando la destinazione invia un ack di conferma, dice anche quanti ulteriori byte possono essere spediti.

Nell'esempio che segue, le peer entity si sono preventivamente accordate su un buffer di 4K a destinazione.

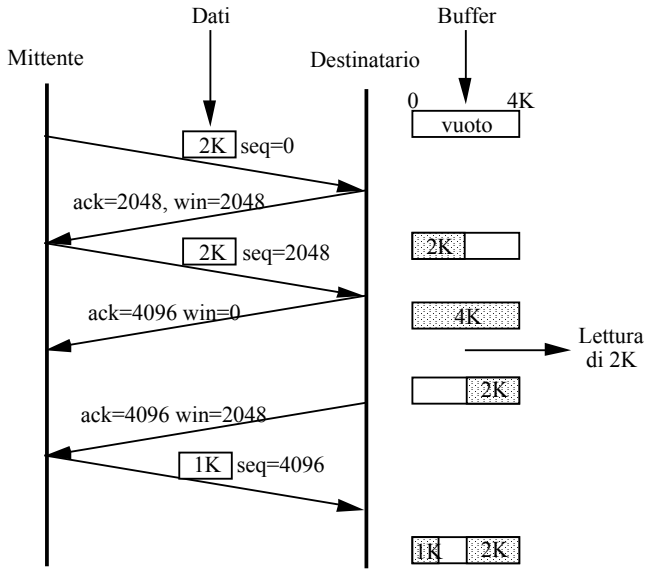


Figura 6-16: Esempio di controllo del flusso TCP

Anche se riceve win=0, il mittente può comunque inviare:

- dati urgenti;
- richieste di reinvio dell'ultimo ack spedito (per evitare il deadlock se esso si è perso).

6.7.6) Controllo congestione

Il protocollo TCP assume che, se gli ack non tornano in tempo, ciò sia dovuto a congestione della subnet piuttosto che a errori di trasmissione (dato che le moderne linee di trasmissione sono molto affidabili).

Dunque, TCP è preparato ad affrontare due tipi di problemi:

- scarsità di buffer a destinazione;
- congestione della subnet.

Ciascuno dei problemi viene gestito da una specifica finestra mantenuta dal mittente:

- la finestra del buffer del ricevitore (quella di cui all'esempio precedente);
- la ***congestion window***, che rappresenta quanto si può spedire senza causare congestione.

Il mittente si regola sulla più piccola delle due.

La congestion window viene gestita in questo modo:

- il valore iniziale è pari alla dimensione del massimo segmento usato nella connessione;
- ogni volta che un ack torna indietro in tempo la finestra si raddoppia, fino a un valore ***threshold***, inizialmente pari a 64 Kbyte, dopodiché aumenta linearmente di 1 segmento alla volta;
- quando si verifica un timeout per un segmento:
    - il valore di threshold viene impostato alla metà della dimensione della congestion window;
    - la dimensione della congestion window viene impostata alla dimensione del massimo segmento usato nella connessione.

Vediamo ora un esempio, con segmenti di dimensione 1 Kbyte, threshold a 32 Kbyte e congestion window arrivata a 40 Kbyte:

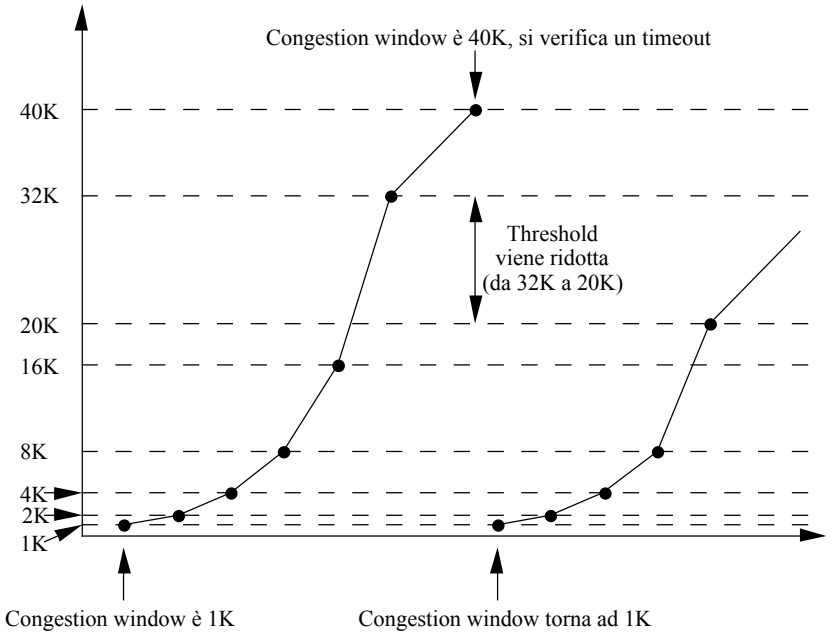


Figura 6-17: Esempio di controllo della congestione TCP

6.7.7) Il protocollo UDP

Il livello transport fornisce anche un protocollo non connesso e non affidabile, utile per inviare dati senza stabilire connessioni (ad esempio per applicazioni client-server).

Lo header di un segmento UDP è molto semplice:

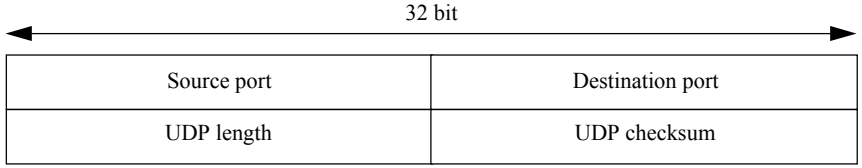


Figura 6-18: Formato dello header UDP

La funzione di calcolo del checksum può essere disattivata, tipicamente nel caso di traffico in tempo reale (come voce e video) per il quale è in genere più importante mantenere un'elevato tasso di arrivo dei segmenti piuttosto che evitare i rari errori che possono accadere.

7) Il livello cinque (Application)

Nel nostro modello di architettura (e anche nell'architettura TCP/IP) sopra il livello transport c'è il livello application, nel quale viene effettivamente svolto il lavoro utile per l'utente.

In questo livello si trovano diverse tipologie di oggetti:
- protocolli di supporto a tutte le applicazioni, come per esempio il ***DNS*** (***Domain Name System***, RFC 1034 e 1035);
- protocolli di supporto ad applicazioni di tipo standardizzato, come ad esempio:
  - ***SNMP*** (***Simple Network Management Protocol***, RFC 1157) per la gestione della rete;
  - ***FTP*** (***File Transfer Protocol***, RFC 959) per il trasferimento di file;
  - ***SMTP*** e ***POP3*** (***Simple Mail Transfer Protocol***, RFC 821, e ***Post Office Protocol***, RFC 1225) per la posta elettronica;
  - ***HTTP*** (***HyperText Transfer Protocol***, RFC 1945) alla base del ***World Wide Web*** (***WWW***);
- applicazioni scritte in conformità ai protocolli di cui sopra;
- applicazioni proprietarie, basate su regole di dialogo private (ad esempio, un'applicazione di tipo client/server per la gestione remota di un magazzino).

7.1) Il DNS

Poiché riferirsi a una risorsa (sia essa un host oppure l'indirizzo di posta elettronica di un utente) utilizzando un indirizzo IP numerico (della forma x.y.z.w) è estremamente scomodo, si è creato un meccanismo tramite il quale tali risorse possono essere identificate tramite un ***nome logico***, cioé una stringa di caratteri (molto pù comprensibile per un essere umano) quale ad esempio:
- `sparc1.unimi.it` (riferimento ad un host);
- `john@cern.ch` (indirizzo di posta elettronica).

La corrispondenza fra gli indirizzi IP numerici ed i nomi logici si effettua mediante l'uso del DNS.

Esso consiste di:
1. uno ***schema gerarchico di nominazione***, basato sul concetto di dominio (***domain***);
2. un ***database distribuito*** che implementa lo schema di nominazione;
3. un ***protocollo*** per il mantenimento e la distribuzione delle informazioni sulle corrispondenze.

Il funzionamento, in breve, è il seguente:
- quando un'applicazione deve collegarsi ad una risorsa di cui conosce il nome logico (ad es. sparc1.unimi.it), invia una richiesta al ***DNS server*** locale (l'applicazione chiama per questo una apposita procedura di libreria detta ***resolver***);
- il DNS server locale, se conosce la risposta, la invia direttamente al richiedente. Altrimenti interroga a sua volta un DNS server di livello superiore, e così via. Quando finalmente arriva la risposta, il DNS server locale la passa al richiedente;
- quando l'applicazione riceve la risposta (costituita del numero IP della risorsa in questione) crea una connessione TCP con la (o spedisce segmenti UDP alla) destinazione, usando l'indirizzo IP testé ricevuto.

Lo spazio dei nomi DNS è uno ***spazio gerarchico***, ***organizzato in dominii***, ciascuno dei quali può avere dei sottodominii.

Esiste un insieme di dominii di massimo livello (***top-level domain***), i più alti nella gerarchia.

Nel caso di un host, la forma del nome logico è costituita da un certo numero di sottostringhe separate da punti, come nell'esempio seguente:

host.subdomain3.subdomain2.subdomain1.topleveldomain

dove:
- la prima sottostringa (quella più a sinistra) identifica il nome dell'host;
- le altre sottostringhe (tranne quella più a destra) identificano ciascuna un sottodominio del dominio di cui alla sottostringa seguente;
- l'ultima sottostringa (quella più a destra) identifica il top-level domain di appartenenza.

Per gli USA sono definiti, fra gli altri, i seguenti top-level domain:

com	aziende
edu	università
gov	istituzioni governative
mil	istituzioni militari
net	fornitori d'accesso
org	organizzazioni non-profit

Fuori degli USA, ogni nazione ha un suo top-level domain. Ad esempio:

au	Australia
ch	Svizzera
fr	Francia
it	Italia
jp	Giappone
uk	Inghilterra

Ogni dominio è responsabile della creazione dei suoi sottodominii, che devono essere registrati presso una appositá autorità.

Esempi di sottodominii sono:

cern.ch	il CERN a Ginevra, Svizzera
cnr.it	il Consiglio Nazionale delle Ricerche, Italia
mit.edu	il Massachussetts Institute of Technology, USA
nasa.gov	la NASA, USA
unige.ch	Università di Ginevra, Svizzera
uniroma1.it	Università di Roma "La Sapienza", Italia
dsi.uniroma1.it	Dip. di Scienze dell'Informazione, Università di Roma "La Sapienza"

L'estensione di un dominio è del tutto indipendente da quella delle reti e sottoreti IP.

Ogni dominio ha la responsabilità di fornire il servizio DNS per quanto di propria competenza. Ossia, deve poter rispondere a interrogazioni riguardanti tutti gli host contenuti nel dominio stesso.

Ciò si ottiene predisponendo un ***name server*** (o più di uno), che è un processo in grado di gestire le seguenti informazioni:

- informazioni di corrispondenza fra nomi simbolici e indirizzi IP. Per ogni host del dominio esiste un ***resource record*** che contiene tali informazioni; tale record è detto ***authoritative record***, in quanto è gestito dal DNS server responsabile del dominio (che è supposto fornire sempre informazioni corrette e aggiornate);
- l'identità dei name server responsabili dei sottodominii inclusi nel dominio, così da poter inviare loro le richieste che gli pervengono dall'alto della gerarchia;

- l'identità del name server responsabile del dominio di livello immediatamente superiore, così da potergli inviare le richieste che gli pervengono dal basso della gerarchia.

Una richiesta che arriva a un name server può dunque viaggiare verso l'alto nella gerarchia oppure (dal momento in cui perviene a un top-level domain server) verso il basso, a seconda dei casi. Quando una risposta ritorna indietro, essa viene tenuta dal server in una sua cache per un certo periodo; qui costituisce un nuovo record, detto ***cached record*** perché contiene della informazione che potrebbe anche divenire, col passare del tempo, obsoleta e non più corretta.

Un esempio di resource record (relativo a un host) è:

```
spcw.dsi.uniroma1.it 86400 IN A 151.100.17.110
```

dove:

`spcw.dsi.uniroma1.it`	***domain_name***: nome simbolico.
`86400`	***time_to_live***: la quantità di tempo (in secondi) trascorsa la quale il record viene tolto dalla cache.
`IN`	***class***: classe del record (Internet in questo caso).
`A`	***type***: tipo del record (Address in questo caso).
`151.100.17.110`	***value***: indirizzo IP numerico.

7.2) La posta elettronica

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

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

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

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

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

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

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

Gli indirizzi di posta elettronica in Internet hanno la forma:

username@hostname

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

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

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

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

Il primo permette all'utente finale di:

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

Il secondo si occupa di:

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

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

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

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

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

`pippo@topolinia.wd`

spedisca un messaggio a

`minnie@paperinia.wd`

e immaginiamo che:

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

La sequenza di azioni che hanno luogo è la seguente:

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

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

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

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

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

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

`chiunque@paperinia.wd`

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

`100.10.10.5`

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

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

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