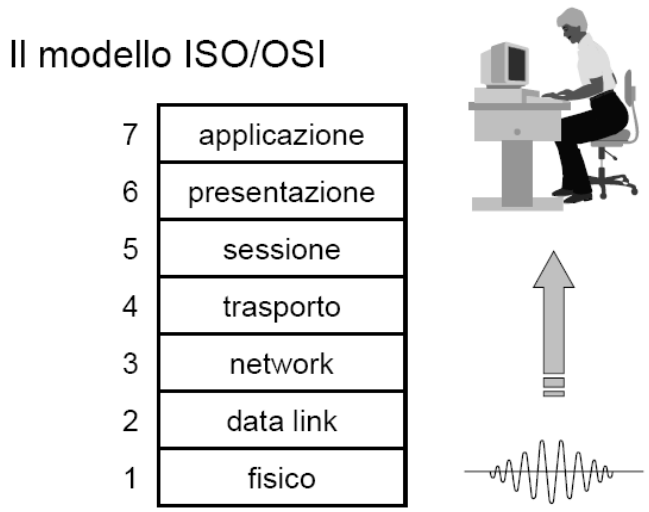
MODELLO OSI



Il modello OSI

Le tecniche di interconnessione fra computer furono storicamente messe a punto dai costruttori di sistemi informatici, primo fra tutti IBM. Le soluzioni che ne risultavano erano però delle reti di computer "chiuse", ossia costituite di apparati tutti dello stesso costruttore e incapaci di comunicare con macchine di altra origine.
L'avvento di sistemi informatici "aperti" , tali cioè da poter comunicare tra loro, attraverso l'utilizzo di opportuni standard, pur essendo realizzati con componenti hardware e software di fornitori diversi, fu facilitato dall'emergere di ARPAnet e delle reti pubbliche per dati, ma si scontrò presto con problemi che andavano oltre il semplice trasferimento di bit da un punto a un altro.
Si pensi ad esempio ai problemi connessi con l'uso di diverse codifiche per rappresentare testi; diverse strutture dei *file system*; diverse procedure di stampa, di login, ecc...
Il problema fu affrontato in modo sistematico dall'ISO che diede avvio alla fine degli anni '70 al progetto Open System Interconnection (OSI).
Le prime specifiche pubblicate nel 1978 dall'ISO definivano un modello di riferimento che descrive non come il sistema è costituito al suo interno, ma come si presenta verso il mondo esterno, ossia verso gli altri sistemi.

Il modello si basa su un'architettura modulare impostata su sette strati completamente indipendenti.

Ogni strato infatti, può essere modificato senza ripercussioni sugli altri strati; questo consente di scomporre il processo di comunicazione in una serie di passi più semplici, di integrare i prodotti di venditori diversi e dunque permette l'interoperabilità tra reti con caratteristiche differenti.

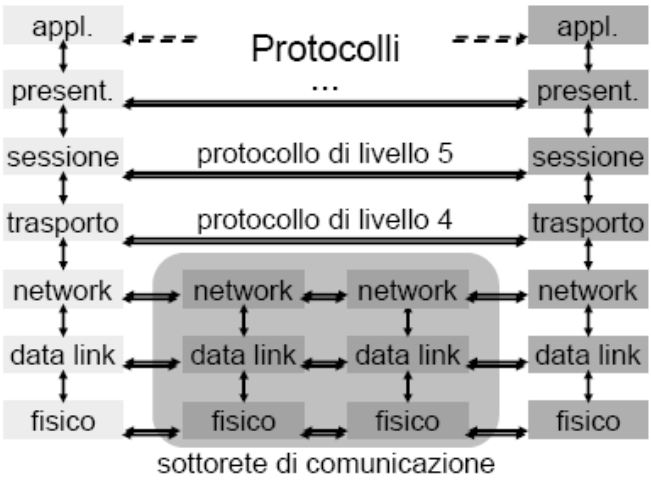
Lo scambio delle informazioni tra i vari strati è delegato ad opportune interfacce di comunicazione.

Vale la pena sottolineare la differenza tra "modello di riferimento" è "architettura di rete".
Un modello di riferimento come quello OSI **si limita a definire in maniera astratta il numero e le caratteristiche funzionali** dei singoli livelli, ovvero quali servizi devono svolgere i diversi strati del modello indipendentemente dal modo in cui questi saranno realizzati.
Un' "architettura di rete" al contrario definisce i protocolli e le interfacce effettivi di ogni livello, ovvero individua una specifica implementazione.
Uno strato può usare i protocolli che preferisce purché questi portino ai risultati desiderati(ovvero offrano i servizi specificati dal modello), e li si può cambiare senza modificare gli altri strati.
Il modello OSI è stato concepito prima di inventare i protocolli corrispondenti e ciò significa che il modello non è orientato verso un insieme specifico di protocolli.
Le idee che ne sono alla base rispecchiano alla perfezione i moderni concetti sulla programmazione ad oggetti.
Un oggetto come uno strato ha un insieme di metodi che sono invocati all'interno di un programma.
I metodi definiscono l'insieme di servizi offerti dall'oggetto, mentre i parametri dei metodi e i valori restituiti costituiscono la sua interfaccia.
Il codice interno all'oggetto è equivalente al suo protocollo, esso permette di fornire i servizi richiesti, ma è completamente invisibile all'esterno dell'oggetto.
Gli oggetti vengono utilizzati ignorandone il funzionamento interno. Semplicemente conoscendo la loro interfaccia e a patto che questa resti invariata, le eventuali modifiche al codice interno di un oggetto non si riflettono sul codice esterno che utilizza i suoi metodi.

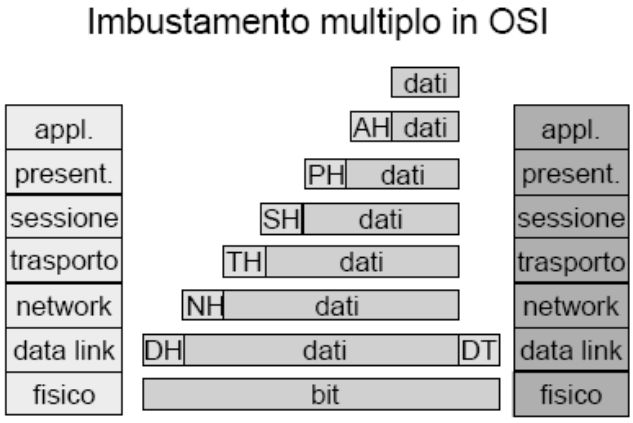
Il modello ISO/OSI è dunque costituito da strati (o livelli), i cosiddetti *layer*, che racchiudono uno o più aspetti fra loro correlati della comunicazione fra due nodi di una rete. I layers sono in totale 7 e vanno dal livello fisico (quello del mezzo fisico, ossia del cavo o delle onde radio) fino al livello delle applicazioni, attraverso cui si realizza la comunicazione di *alto livello*.

Ad ogni layer è associato un protocollo di comunicazione che varia a seconda delle caratteristiche della rete o del tipo di servizio richiesto(affidabile/non affidabile, orientato/non orientato alla connessione, etc…).

ISO/OSI realizza una **comunicazione per livelli**, ovvero, dati due nodi A e B, il livello n del nodo A può scambiare informazioni col livello n del nodo B ma non con gli altri.

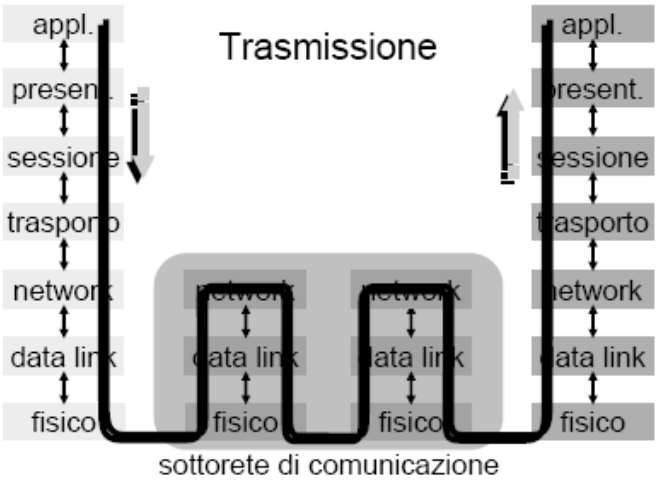


Inoltre ogni livello realizza la comunicazione col livello corrispondente su altri nodi usando i servizi offerti unicamente dal livello immediatamente sottostante. Sicché ISO/OSI incapsula i messaggi di livello n in messaggi del livello n-1 aggiungendo altre informazioni necessarie a svolgere le funzioni richieste a quel livello nella forma di un'intestazione(*header*) ed eventualmente di una coda(*trailer*) .



Così se A deve inviare, ad esempio, una e-mail a B, l'applicazione (liv. 7) di A propagherà il messaggio usando il layer sottostante (liv. 6) che a sua volta userà le funzioni offerte dal layer inferiore, fino ad arrivare alla trasmissione sul mezzo fisico.

Al nodo B i dati verranno estratti, livello per livello fino a quello dell'applicazione.



Passaggi analoghi avvengono anche in corrispondenza dei dispositivi della sottorete di comunicazione : i router (e nella LAN gli switch).

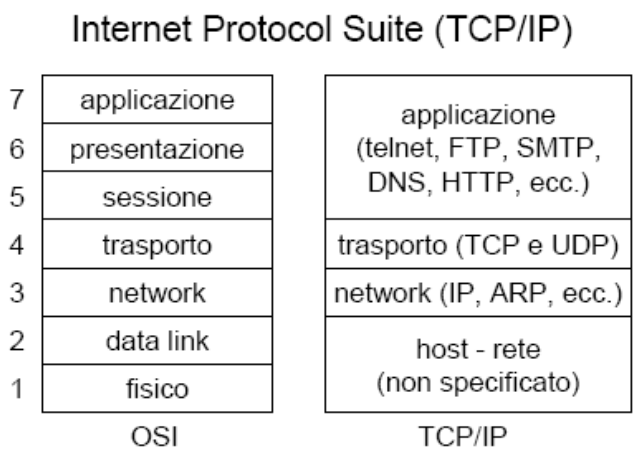
Differenze rispetto al modello DoD/ARPA

ISO/OSI è stato progettato per permettere la comunicazione in reti a "commutazione di pacchetto" , analogamente al modello DoD/ARPA.

Nel caso del modello DoD/ARPA tuttavia sono stati sviluppati prima i protocolli che costituiscono la suite TCP/IP e poi fu realizzato un modello che rappresentava una semplice descrizione dei protocolli esistenti.

La differenza sostanziale fra TCP/IP e ISO/OSI consiste nel fatto che nel TCP/IP i **layer sono solo 4** (applicazione, trasporto, rete, interfaccia) e **i livelli sessione, presentazione sono assenti** perché implementati (eventualmente) al livello applicativo **mentre il livello di interfaccia con il mezzo fisico ingloba sia il livello fisico che quello di data-link**.

Anche se a prima vista potrebbe sembrare sbagliato accorpare due livelli in uno, in effetti tali livelli non sono completamente indipendenti, ovvero una modifica del livello fisico può comportare l'esigenza di modificare anche il livello di data-link.



Un'altra importante differenza risiede nella modalità di comunicazione, orientata o meno alla connessione.

Il modello OSI prevede entrambe le modalità al livello di rete, mentre al livello di trasporto supporta solo quella orientata alla connessione. Viceversa il modello TCP/IP ha solo una modalità al livello di rete(protocollo IP non orientato alla connessione), mentre le supporta entrambe al livello di trasporto(TCP o UDP).

Esistono altre differenze ovviamente tra i due protocolli. **Nel modello OSI abbiamo una chiara distinzione tra servizi, protocolli e interfacce, al contrario del modello TCP/IP.**

Ciò significa che è più semplice cambiare un protocollo di un livello nel modello OSI senza influenzare il livelli adiacenti.

In pratica ISO/OSI è più flessibile rispetto al paradigma di TCP/IP, ma soltanto perché risulta più astratto rispetto a questo.

Quando fu sviluppato, molti esperti credevano che questo modello avrebbe soppiantato rapidamente tutti gli altri dominando la scena mondiale. Ciò non è successo in realtà.

Le ragioni di tale fallimento vanno ricercate nella tempistica, nella complessità del modello e nel fatto che le prime implementazioni erano di scarsa qualità.

Tutto questo giocò a favore del modello concorrente che al contrario, sebbene meno generale, poteva fornire implementazioni che funzionavano ragionevolmente bene ed erano già ampiamente diffuse nel mondo accademico da creare un mercato.

Dunque sebbene il modello OSI abbia conquistato un certo rilievo a livello accademico, ovvero per discutere di teoria delle reti, non si è affermato a livello commerciale venendo invece rapidamente soppiantato dal TCP/IP.

In pratica non esistono implementazioni 'complete' di ISO/OSI, a parte quelle proprietarie (ad esempio DECNET della Digital) e di interesse accademico.

Descrizione dei livelli

Livello 1: Fisico

Obiettivo: trasmettere un flusso di dati non strutturati attraverso un collegamento fisico, occupandosi della forma e del voltaggio del segnale. Ha a che fare con le procedure meccaniche e elettroniche necessarie a stabilire, mantenere e disattivare un collegamento fisico.

Semplicemente si occupa di controllare la rete, gli hardware che la compongono e i dispositivi che permettono la connessione.

In questo livello si decidono:

- Le tensioni scelte per rappresentare i valori logici 0 e 1
- La durata in microsecondi del segnale elettrico che identifica un bit
- L'eventuale trasmissione simultanea in due direzioni
- La forma e la meccanica dei connettori usati per collegare l'hardware al mezzo trasmissivo

Livello 2: Data-link

Obiettivo: gestire la comunicazione tra nodi direttamente connessi, garantire il controllo di flusso(ovvero la necessaria sincronizzazione tra sorgente e destinazione) e dirimere le contese per l'accesso al canale di comunicazione nel caso in cui questo risulti condiviso, rilevare eventuali errori di trasmissione e gestire eventualmente la perdita o la duplicazione dei frames(a seconda che il protocollo preveda un riscontro oppure no) per permettere un trasferimento affidabile di dati attraverso il livello fisico. Tutto ciò consente di far apparire, al livello superiore, il mezzo fisico come una linea di trasmissione esente da errori di trasmissione

Questo livello si occupa di formare i dati da inviare attraverso il livello fisico, incapsulando i dati in una trama o *frame* provvisto di *header*(intestazione) e *trailer* (coda).
Questa frammentazione è detta *framing* e i campi che compongono l' *header* e il *trailer* variano a seconda del protocollo utilizzato a questo livello(quello più diffuso per le LAN è Ethernet, mentre per le WAN vi sono vari protocolli come PPP).

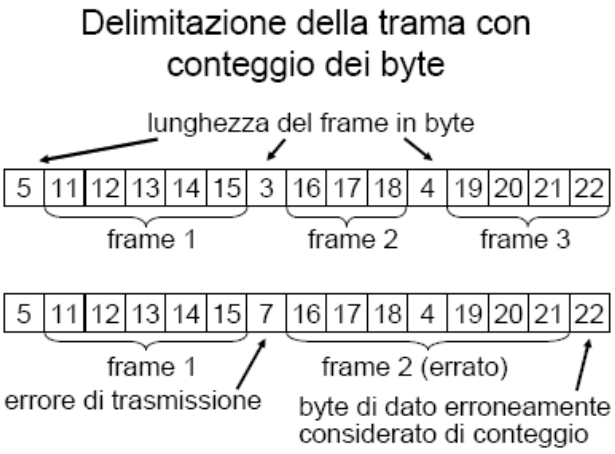
Framing

Per poter individuare l'inizio e la fine di un frame occorre in qualche modo delimitarli, perché sarebbe troppo rischioso utilizzare lo spazio che intercorre tra un frame ed un altro per tale delimitazione.

Vengono utilizzati vari metodi per la delimitazione dei frames:

- Conteggio dei caratteri.
- Caratteri di inizio e fine.
- Indicatori (flag) di inizio e fine.

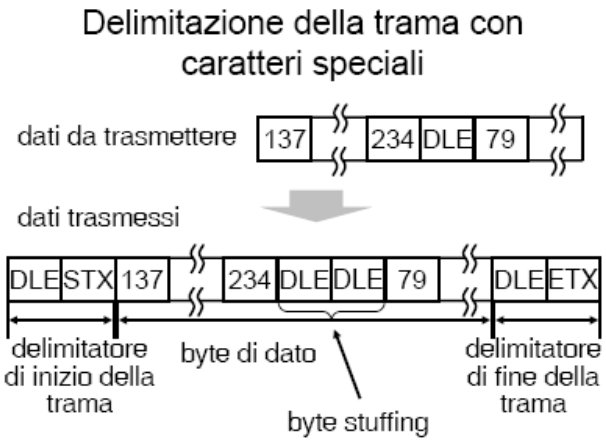
Il metodo del **conteggio di caratteri** (specificando in un campo dell'intestazione il numero di caratteri del frame) è raramente utilizzabile perché, se il campo che contiene il conteggio si rovina durante la trasmissione, non si può più individuare dove comincia il frame successivo pertanto vengono utilizzate le tecniche più affidabili .



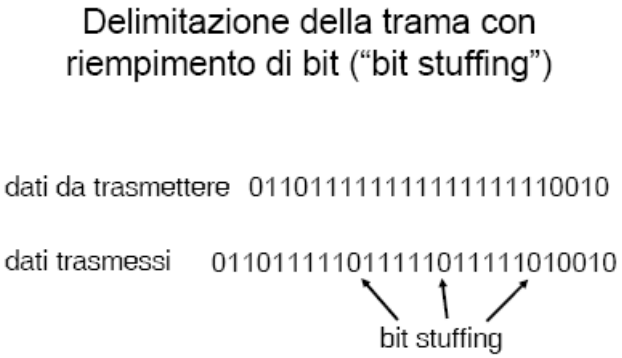
Nella **trasmissione orientata al byte** (il frame mantiene la suddivisione in byte) il frame viene preceduto dalla sequenza di caratteri ASCII **DLE STX** (Data Link Escape Start of TeXt) e finisce con la sequenza **DLE ETX** (Data Link Escape End of TeXt).

Se un frame si rovina e la destinazione perde la sincronizzazione basta trovare il successivo DLE STX o DLE ETX.

Il carattere DLE però può comparire casualmente dentro al frame; perché questi caratteri non interferiscano viene aggiunto un ulteriore DLE (che viene rimosso a destinazione prima di passare al frame al livello di rete) in modo che solo i DLE singoli vengano interpretati come delimitatori; questa tecnica si chiama *character stuffing*.



Nella **trasmissione orientata al bit** (il frame può contenere un numero qualsiasi di bite) **ogni frame inizia e finisce con la sequenza 01111110** chiamata flag: questa sequenza può comparire casualmente nei dati, perciò in trasmissione **dopo cinque 1 consecutivi viene sempre inserito uno 0** nel flusso di bit, indipendentemente dal fatto che il bit successivo sia 1 o 0, mentre in ricezione bisogna provvedere ad eliminare i bit inseriti, rimuovendo sempre uno 0 dopo cinque; questa tecnica è chiamata *bit stuffing*.



Rilevamento e correzione degli errori di trasmissione

Per ogni frame da inviare viene calcolato un codice di controllo detto *checksum* (o somma di controllo) legato alla sequenza di bit da trasmettere e il valore ottenuto è trasmesso insieme al frame.

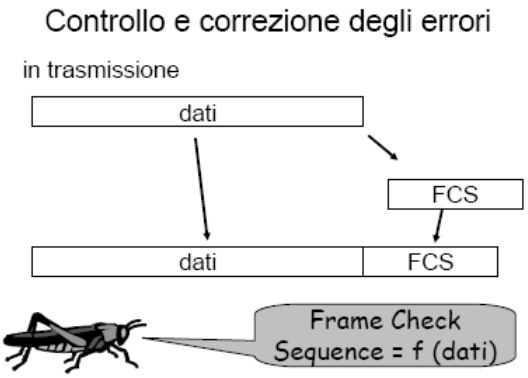
Quando un *frame* arriva a destinazione, il *checksum* viene ricalcolato. Se il nuovo risultato è diverso da quello contenuto nel pacchetto, il livello data link riconosce che deve essersi verificato un errore di trasmissione e provvede ad esempio a scartare il pacchetto ed eventualmente a spedire in risposta un messaggio di errore.

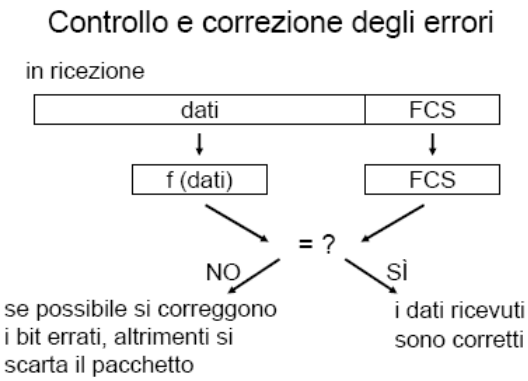
A seconda dell’algoritmo utilizzato per il calcolo del *checksum* è possibile anche un’altra alternativa: correggere l’errore.

Questo richiede ovviamente un algoritmo di calcolo di checksum che permetta non solo la rilevazione dell’errore, ma che fornisca anche la possibilità di risalire alla configurazione originaria.

Un codice per che permetta la correzione degli errori richiede però più spazio nel frame, tanto più quanto maggiore è il numero di errori che riesce a rilevare.

La scelta tra codici rilevatori e correttori dipende in genere dalla velocità delle linee (per linee a bassa velocità aspettare la ritrasmissione potrebbe richiedere troppo tempo) e dalla loro affidabilità (se il tasso di errore sulla linea è molto basso non vale la pena sprecare molta per un codice correttore).





Protocolli orientati alla connessione e non

Il livello di data-link può offrire un servizio:

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)
- appropriato per canali con tasso d'errore molto basso

Connectionless confermato

- I frame vengono confermati
- Se la conferma non arriva, il mittente può rispedire il frame
- appropriato per canali non affidabili (sistemi wireless)

Connection oriented confermato

- Tre fasi: apertura connessione, invio dati, chiusura connessione
- Garantisce che ogni frame sia ricevuto esattamente una volta e nell'ordine giusto
- Fornisce al livello network un flusso di bit affidabile

Nel caso in cui il protocollo utilizzato a questo livello prevede il riscontro, per ogni *frame* ricevuto, il destinatario invia al mittente un *frame* **ACK** di *acknowledgement* ovvero di conferma dell'avvenuta ricezione del *frame* inviato.

Il mittente ripete l'invio dei pacchetti alterati da errori di trasmissione(in questo caso il frame di riscontro è indicato con NACK che sta per *Not - Acknowledged)* e di quelli di cui non ha ricevuto un ACK.

Per ottimizzare l'invio degli ACK, si usa una tecnica detta **Piggybacking**, che consiste nell'accodare ai messaggi in uscita gli ACK relativi ad una connessione in entrata, per ottimizzare l'uso del livello fisico i pacchetti ACK possono anche essere raggruppati e mandati in blocchi.

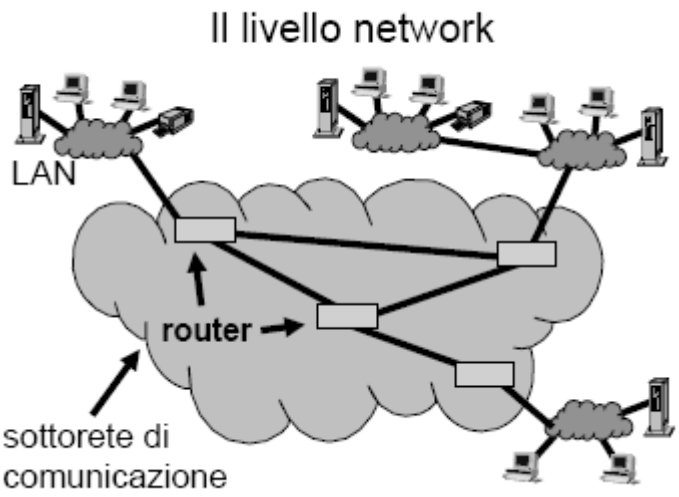
Nel caso in cui il protocollo utilizzato a questo livello non fornisca tali servizi devono essere i livelli superiori ad effettuare il controllo di flusso, il controllo degli errori e gestire le conferme (e relative ritrasmissioni).

Controllo di flusso

Un altro importante problema di progettazione che si ritrova nel livello di data link è quello gestire una linea condivisa quando più nodi vogliono inviare messaggi nello stesso tempo e inoltre deve decidere cosa fare di un mittente che sistematicamente tende a trasmettere pacchetti più velocemente di quanto il ricevente li accetti. Questa situazione può facilmente essere riscontrata quando il mittente è dislocato su una macchina veloce e il ricevente su una macchina lenta. Il mittente continua a spedire pacchetti ad alta velocità, fino a quando il ricevente non è completamente sopraffatto. Anche se la trasmissione è esente da errori, a un certo punto il ricevente non sarà in grado di gestire i pacchetti in arrivo e inizierà a perderli. La tipica soluzione è quella di introdurre un controllo di flusso per obbligare il mittente a rispettare la velocità del ricevente nello spedire i pacchetti. Questa imposizione solitamente richiede un certo tipo di meccanismo di riscontro in modo che il mittente possa essere avvisato se il ricevente è in grado di ricevere o meno. Nel caso in cui invece più nodi vogliono inviare contemporaneamente dei messaggi, si tende ad introdurre un controllo centralizzato, creando un singolo nodo di controllo, responsabile di determinare chi ottiene la priorità all'interno della rete; il nodo successivo quindi, controllerà quando la rete non sarà più occupata, così da poter inviare il messaggio appena questa diventerà libera. Può accadere però, che più nodi monitorizzano la rete e che appena questa sia libera, inviano immediatamente i messaggi, in questo caso si avranno dei problemi di collisione; per ovviare a questo problema, i nodi che monitorizzano la rete attenderanno un tempo casuale prima di inviare i messaggi, poiché è improbabile che i nodi scelgano lo stesso istante per inviare i dati(esempi sono il protocollo CSMA/CD di Ethernet per le reti cablate e CSMA/CA per le reti wireless).

Livello 3: Rete

Obiettivo: Il compito del livello di rete è la trasmissione di pacchetti tra due host arbitrari, che in generale non sono direttamente connessi (ovvero non hanno un collegamento diretto tra di loro), rendendo i livelli superiori indipendenti dai meccanismi e dalle tecnologie di trasmissione usate per la connessione.



È responsabile di:

- **routing**: scelta ottimale del percorso da utilizzare per garantire la consegna delle informazioni
- **gestione della congestione**: evitare che troppi pacchetti arrivino allo stesso router contemporaneamente
- **indirizzamento**
- **conversione dei dati nel passaggio fra una rete ed un'altra con diverse caratteristiche**. Deve, quindi:
    - tradurre gli indirizzi
    - valutare la necessita' di frammentare i dati se la nuova rete ha una diversa Maximum Transmission Unit (MTU)
    - valutare la necessita' di gestire diversi protocolli attraverso l'impiego di gateway

La sua unità dati fondamentale è il *pacchetto*.

Alcuni protocolli di rete forniscono un servizio di gestione delle **connessioni** (X.25, frame relay, Asynchronous Transfer Mode), ovvero richiedono che venga stabilito un canale di comunicazione prima che due host possano scambiarsi dati; altri trasportano semplicemente i datagrammi a destinazione (IP, IPX). I protocolli orientati alla connessione possono offrire garanzie di consegna in ordine dei pacchetti, mentre questo non avviene normalmente nei protocolli non orientati alla connessione.
Valgono considerazioni simili a quelle viste per il livello di data-link.
I protocolli orientati alla connessione prevedono che all'atto di stabilire una connessione venga creato un percorso logico che tutti i pacchetti seguiranno, tale percorso viene detto circuito virtuale e si differenzia dal percorso fisico creato nella commutazione di circuito(nelle centraline telefoniche) perché non viene creato un percorso fisico, bensì si stabilisce semplicemente attraverso quali router verranno inoltrati tutti i pacchetti.

Ciò fa sì che i pacchetti possano arrivare in ordine. Inoltre tali protocolli prevedono forme di riscontro che garantiscono una connessione affidabile.

Poiché il percorso compiuto dai vari pacchetti attraverso la rete è lo stesso, non è necessario specificare l'indirizzo completo del mittente e del destinatario, ma solo il numero del circuito virtuale.

Al contrario qualora si utilizzi un protocollo non orientato alla connessione è necessario specificare gli indirizzi completi perché ogni pacchetto segue un percorso indipendente.

Ciò che avviene in un protocollo orientato alla connessione è paragonabile ad una chiamata telefonica, mentre l'invio di pacchetti con un protocollo non orientato alla connessione è simile all'invio di un telegramma da qui il termine **datagram** per designare i pacchetti in quest'ultimo caso.

Un protocollo non orientato alla connessione è in compenso più veloce nella consegna dei pacchetti anche se non fornisce alcuna garanzia che non ne sia stato perso qualcuno o che l'ordine di arrivo corrisponda all'ordine di trasmissione.

Circuiti virtuali e datagram

	datagram	circuiti virtuali
creazione circuito	non richiesto	richiesto
indirizzi	ogni pacchetto contiene gli indirizzi del mittente e del destinatario	ogni pacchetto contiene l'identificatore di VC
routing	ogni pacchetto è instradato indipendentemente	il percorso è scelto all'inizializzazione
effetti dei guasti	solo sui pacchetti persi durante il guasto	tutti i VC interessati dal guasto terminano
controllo della congestione	complesso	semplice (gestione preventiva durante il setup)

Per quanto riguarda l'instradamento, nella maggior parte dei casi, questa funzione viene svolta dinamicamente tramite appositi algoritmi, che analizzano le condizioni della rete, le tabelle di instradamento, la priorità del servizio e altri elementi secondari.

Nel modello **TCP/IP**, il livello 3 viene detto livello internet oppure livello di internetworking, in quanto interconnette reti eterogenee, che possono essere basate su protocolli di livello collegamento (ad esempio **Ethernet, PPP**) o su protocolli di rete (ad esempio **Frame Relay**, **Asynchronous Transfer Mode**), per realizzare un'unica rete in modo trasparente agli utilizzatori. La forza di IP sta proprio in questo agnosticismo rispetto al livello di rete, che permette di usare o riusare tecnologie già disponibili, e di adattarsi con naturalezza a nuove tecnologie.

IP determina il miglior cammino (detto routing o instradamento) per l'inoltro dei pacchetti, attraverso la consultazione delle tabelle di routing.

Tali tabelle possono essere di tipo statico (realizzate manualmente dai gestori della rete) o dinamico (composte con l'utilizzo di protocolli di routing tipo l'OSPF, il RIP o il BGP che servono a popolare aggiornare le tabelle attraverso lo scambio di informazioni in maniera da riflettere lo stato attuale della rete).

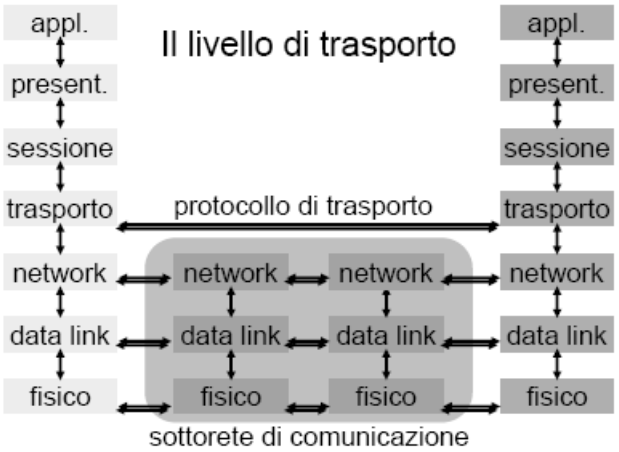
Il protocollo IP è un protocollo non affidabile e non orientato alla connessione.

Livello 4: Trasporto

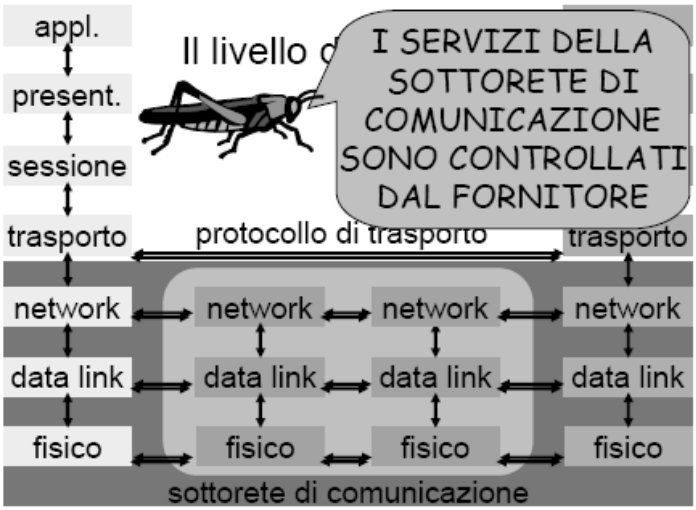
Obiettivo: permettere un trasferimento di dati trasparente e affidabile (implementando anche un controllo degli errori e delle perdite) tra due host. È il primo livello realmente end-to-end, cioè da host sorgente a destinatario.

A differenza dei livelli precedenti, che si occupano di connessioni tra nodi contigui di una rete, il Trasporto (a livello logico) si occupa solo del punto di partenza e di quello di arrivo.
Si occupa anche di effettuare la frammentazione dei dati provenienti dal livello superiore in pacchetti, detti 'segmenti' e trasmetterli in modo efficiente ed affidabile usando il livello rete ed isolando da questo i livelli superiori. Inoltre, si preoccupa di ottimizzare l'uso delle risorse di rete e di prevenire la congestione.
La sua unità dati fondamentale è il *messaggio o segmento*.

Lo scopo dello strato di trasporto è quello di fornire un servizio efficiente ed affidabile ai processi utenti dello strato di sessione.
Le relazioni fra livello di trasporto e quelli di rete e sessione è rappresentata dalla seguente figura:



A prima vista sembrerebbe che il livello di trasporto abbia una funzione del tutto identica a quella del livello di rete. Per capire la funzione dello strato di trasporto occorre tener presente che lo strato di rete fa parte della rete di comunicazione per cui è gestito dalle società di telecomunicazioni.

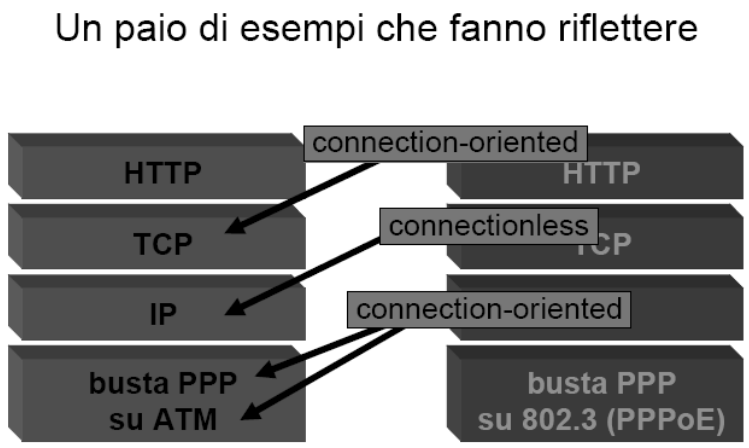


Ciò significa che, poiché gli utenti non hanno alcun controllo sulla rete di comunicazione, essi non potrebbero intervenire nel caso venisse fornito un servizio inaffidabile dallo strato di rete.

Ad esempio, se lo strato di rete non riesce a mantenere la connessione fra due utenti e vi è l'interruzione della connessione fra le due entità di trasporto, lo strato di trasporto rioccuperà del ripristino della connessione e di tutte le operazioni necessarie per individuare quali dati sono giunti in maniera corretta, quali sono andati persi, quali bisogna ritrasmettere e così via.

In sostanza il livello di trasporto può porre rimedio ai pacchetti perduti, ai dati danneggiati, al reset delle connessioni da parte del livello di rete. Inoltre il livello di trasporto rende la rete di comunicazione trasparente alle applicazioni. In sostanza grazie a questo strato si possono scrivere applicazioni senza tener conto delle differenze delle interfacce che compongono le varie sottoreti di comunicazione.

La figura seguente illustra delle stranezze che possono accadere



TCP

Transmission Control Protocol (**TCP**) è un protocollo di livello di trasporto della suite di protocolli Internet. È definito nella RFC 793, e su di esso si appoggiano gran parte delle applicazioni Internet.

Il TCP può essere classificato al livello trasporto (OSI livello 4) del modello di riferimento OSI, e di solito è usato in combinazione con il protocollo di livello rete (OSI livello 3) IP. La corrispondenza con il modello OSI non è perfetta, in quanto il TCP e l'IP sono antecedenti al modello OSI. La loro combinazione è indicata come TCP/IP e, alle volte, è erroneamente considerata un unico protocollo.

Il TCP è stato progettato per utilizzare i servizi del protocollo IP, che non offre alcuna garanzia in ordine alla consegna dei pacchetti, al ritardo, alla congestione, e costruire un **canale di comunicazione affidabile tra due processi applicativi**. Il canale di comunicazione è costituito da un flusso bidirezionale di byte. Inoltre, alcune funzionalità di TCP sono vitali per il buon funzionamento complessivo di una rete IP.

Il TCP nacque nel 1970 come frutto del lavoro di un gruppo di ricerca del dipartimento di difesa statunitense. I suoi punti di forza sono l'alta affidabilità e robustezza. La sua popolarità si deve anche grazie ad una sua implementazione open source diffusa dall'Università di Berkeley

Il servizio offerto da TCP è il trasporto di un **flusso di byte** bidirezionale tra due applicazioni in esecuzione su host differenti. Il protocollo permette alle due applicazioni di trasmettere contemporaneamente nelle due direzioni, quindi il servizio può essere considerato "**Full Duplex**" anche se non tutti i protocolli applicativi basati su TCP utilizzano questa possibilità.

Il flusso di byte viene **frazionato** in blocchi per la trasmissione dall'applicazione a TCP (che normalmente è implementato all'interno del sistema operativo), per la trasmissione all'interno di segmenti TCP, per la consegna all'applicazione che lo riceve, ma questa divisione in blocchi non è per forza la stessa nei diversi passaggi.

TCP è un protocollo **orientato alla connessione**, ovvero prima di poter trasmettere dati deve stabilire la comunicazione, negoziando una connessione tra mittente e destinatario, che viene esplicitamente chiusa quando non più necessaria. Esso quindi ha le funzionalità per creare, mantenere e chiudere una connessione.

TCP garantisce che i dati trasmessi, se giungono a destinazione, lo facciano in ordine e una volta sola ("**at most once**"). Più formalmente, il protocollo fornisce ai livelli superiori un servizio equivalente ad una connessione fisica diretta che trasporta un flusso di byte. Questo è realizzato attraverso vari meccanismi di acknowledgment e di ritrasmissione su timeout.

TCP possiede funzionalità di controllo di flusso e di controllo della congestione sulla connessione, attraverso il meccanismo della finestra scorrevole. Questo permette di ottimizzare l'utilizzo della rete anche in caso di congestione, e di condividere equamente la capacità disponibile tra diverse sessioni TCP attive su un collegamento.

TCP fornisce un servizio di multiplazione delle connessioni su un host, attraverso il meccanismo delle porte.

Confronto con UDP

Le principali differenze tra TCP e UDP(User Datagram Protocol), l'altro principale protocollo di trasporto della suite di protocolli Internet, sono:

1. UDP non offre nessuna garanzia dell'arrivo dei datagrammi né sul loro ordine di arrivo, al contrario il TCP tramite i meccanismi di acknowledgement e di ritrasmissione su timeout riesce a garantire la consegna dei dati, anche se al costo di un maggiore overhead
2. TCP è un protocollo orientato alla connessione, pertanto per stabilire, mantenere e chiudere una connessione, è necessario inviare pacchetti di servizio i quali aumentano l'overhead di comunicazione. Al contrario, UDP invia solo i datagrammi richiesti dal livello applicativo;
3. l'oggetto della comunicazione di TCP è il *flusso di byte* mentre quello di UDP è il *singolo datagramma*.

L'utilizzo del protocollo TCP rispetto a UDP è preferito quando è necessario avere garanzie sulla consegna dei dati o sull'ordine di arrivo dei vari segmenti (come per esempio nel caso di trasferimenti di file).

Al contrario UDP viene principalmente utilizzato per le comunicazioni broadcast e multi cast o quando è più importante la velocità di trasmissione più che l'affidabilità della stessa ovvero nel caso di applicazioni multimediali Streaming audio/video.

Livello 5: Sessione

Obiettivo: controllare la comunicazione tra applicazioni. Stabilire, mantenere e terminare connessioni (sessioni) tra applicazioni cooperanti.

Esso consente di aggiungere, ai servizi forniti dal livello di trasporto, servizi più avanzati, quali l'autenticazione, la gestione del dialogo (mono o bidirezionale), la sincronizzazione tra i due estremi della comunicazione(è utilizzato per esempio nelle videoconferenze).
Si occupa anche di inserire dei punti di controllo nel flusso dati: in caso di errori nell'invio dei pacchetti, la comunicazione riprende dall'ultimo punto di controllo andato a buon fine.

Vi e' molta rassomiglianza tra lo stabilimento di una sessione e quello di un Circuito Virtuale a livello trasporto, mai i due non coincidono esattamente. Sono possibili tre scenari:

- lo stabilimento di una sessione coincide con lo stabilimento di un circuito virtuale, la terminazione di una sessione equivale all'abbattimento del circuito virtuale
- alla chiusura di una sessione non viene terminato il circuito virtuale, per l'elevata probabilita' di una sessione successiva che usi lo stesso circuito
- un circuito virtuale viene chiuso facilmente in maniera anomala a causa di problemi di rete; la sessione sopravvive alla chiusura del circuito e si fa carico del ristabilire nuovi circuiti ogni volta serva, e continua lo scambio dati dal punto di interruzione

Livello 6: Presentazione

Obiettivo: trasformare i dati forniti dalle applicazioni in un formato standardizzato e offrire servizi di comunicazione comuni, come la crittografia, la compressione del testo e la riformattazione.

Grazie a questo livello, le applicazioni che girano su computer con sistemi diversi possono comunicare tra loro in modo indipendente dalle applicazioni stesse.

Al livello 6 vengono infatti definiti gli standard ASCII ed EBCDIC per gestire i file di testo, gli standard GIF (Graphic Interchange Format), JPEG (Joint Photographic Experts Group) e TIFF (Tagged Image File Format) per rappresentare le immagini, lo standard MPEG (Motion Picture Experts Group) per la codifica e la compressione dei dati su memorie di massa digitali, lo standard MIDI (Musical Instrument Digital Interface) per l'audio digitale, lo standard Quick Time per i file con audio e video. Questo livello gestisce inoltre i file binari delle applicazioni multimediali e di FTP, e anche i file in formato HTML (Hypertext Markup Language) usati dai vari web browser. Il livello 6 si occupa infine della compressione dei dati (per ridurne la dimensione e velocizzare la trasmissione) e della crittografia, per proteggere i dati da intromissioni esterne; se sulla linea viaggiano crittografati, anche se intercettati da "estranei", i dati non possono infatti essere interpretati senza la chiave di decodifica.

Livello 7: Applicazione

Obiettivo: fornire servizi alle applicazioni.

Fornisce un insieme di protocolli che operano a stretto contatto con le applicazioni. È errato identificare un'applicazione utente come parte del livello applicazione.
I protocolli delle applicazioni tipiche di questo livello realizzano operazioni come:

- Trasferimento di file(FTP)
- Terminale virtuale(TELNET)
- Posta elettronica(SMTP,POP)
- DNS
- Web(HTTP)