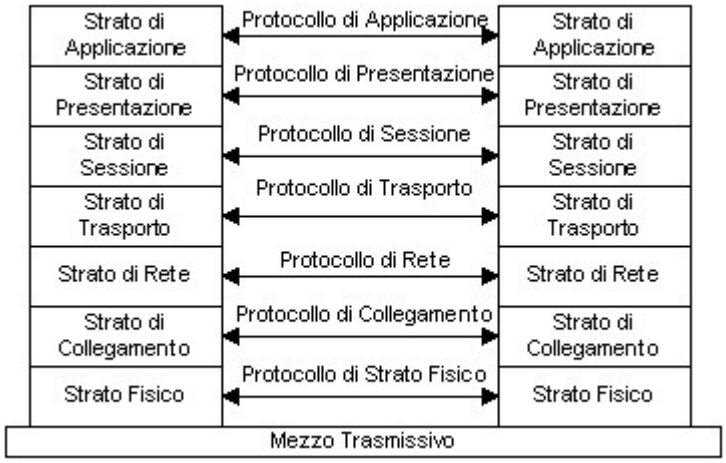
Modello di riferimento ISO-OSI
L'architettura

Come già visto nell'introduzione, il **modello ISO-OSI** (**Open System Interconnection**) prevede una architettura a strati. Nel modello di riferimento OSI gli strati sono 7, organizzati come mostrato nella figura seguente.

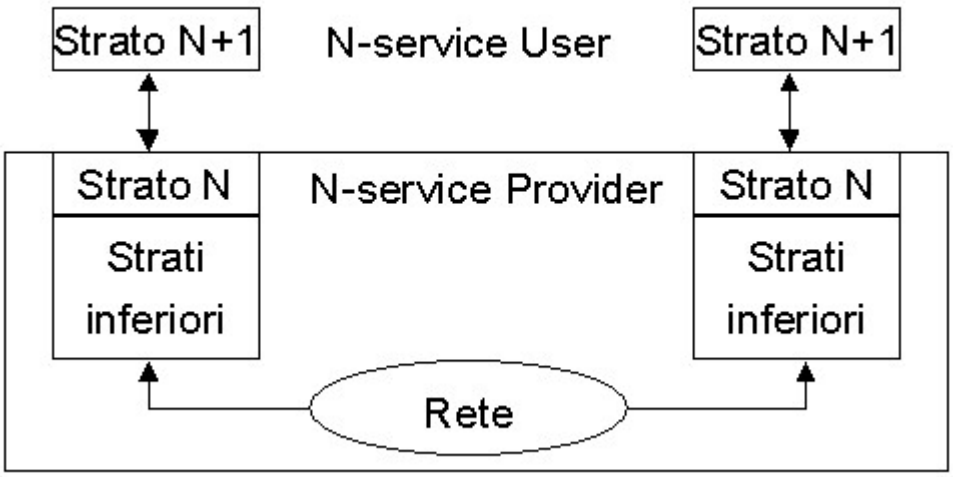



Secondo il modello di riferimento OSI il risultato dell'interazione verticale fra i livelli è un **servizio** (***service***). Un servizio viene definito sulla base delle relative interazioni e dei parametri scambiati.

Secondo la terminologia OSI si dice:

- *N-service provider*
  - Il fornitore di servizio a livello N comprende il livello N e tutti i livelli inferiori di cui il livello N fa uso.
- *N-service user*
  - L'utilizzatore di servizio di livello N è l'**entità** di livello N+1 che fa uso dei servizi del livello N.

Come già detto, il generico strato N, nel fornire il servizio al relativo strato N+1 maschera a questo l'esistenza degli strati inferiori.



Un servizio può essere di tipo ***Connection Oriented*** o ***Connectionless***.

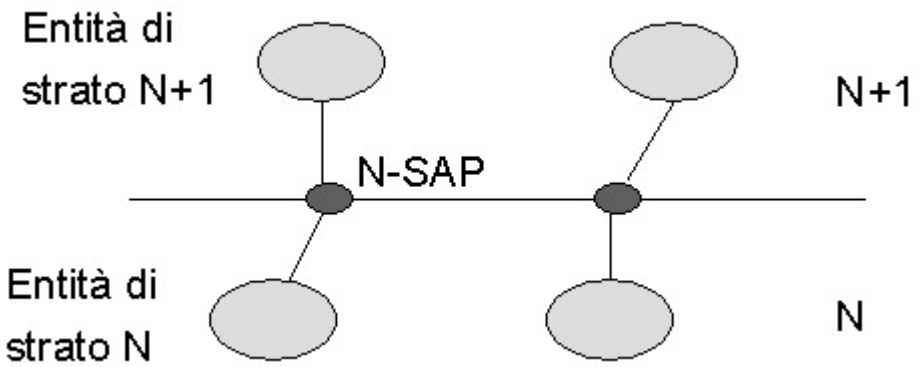
Una modalità di fornire un servizio si dice *Connection Oriented* quando si stabilisce una connessione:

- la connessione associa due o più sistemi al fine di trasferire dati;
- il processo di comunicazione si compone normalmente di tre fasi;
- instaurazione della connessione, tramite lo scambio di opportune informazioni iniziali;
- trasferimento dei dati veri e propri;
- chiusura della connessione.

Qualora i dati vengano trasferiti senza prima stabilire una connessione si parla si servizio *Connectionless*.

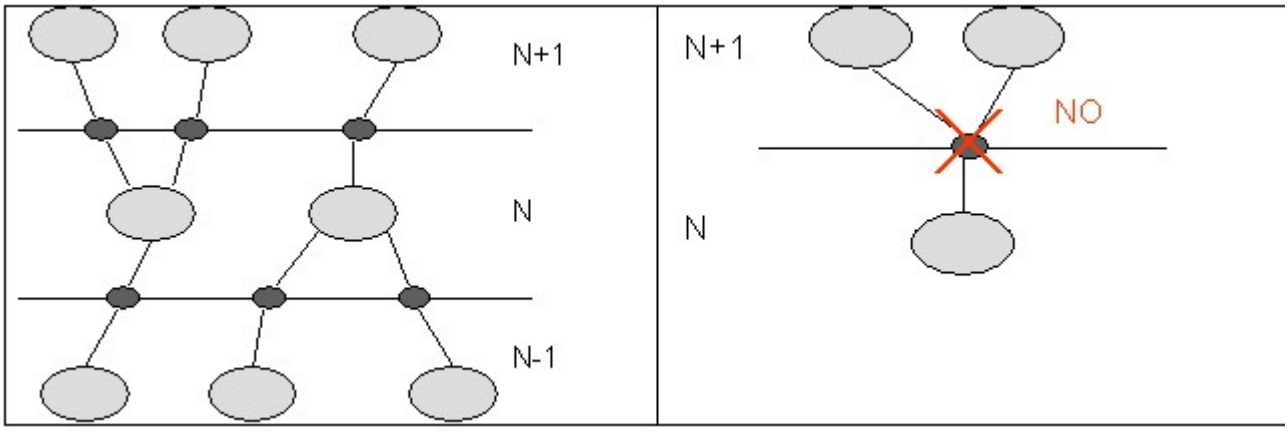
- Per ogni accesso al servizio vengono fornite tutte le informazioni necessarie per il trasferimento dei dati.
- Ogni unità di dati viene trasferita in modo indipendente dalle altre.

Ogni elemento attivo in uno strato viene detto entità. Ciascuno strato comprende una o più entità.




Il **Service Access Point** (**SAP**) è l'interfaccia logica fra un'entità di livello N+1 e una di livello N, attraverso la quale viene fornito un servizio:

- ogni N-SAP ha un **indirizzo** (*address*) unico.
- Un'entità di livello N può servire più N-SAP contemporaneamente.
- Un utilizzatore di livello N può servirsi di più N-SAP contemporaneamente.
- Non è permesso connettere più N-*user* allo stesso N-SAP:
  - si genererebbe ambiguità sulla provenienza/destinazione dei dati.

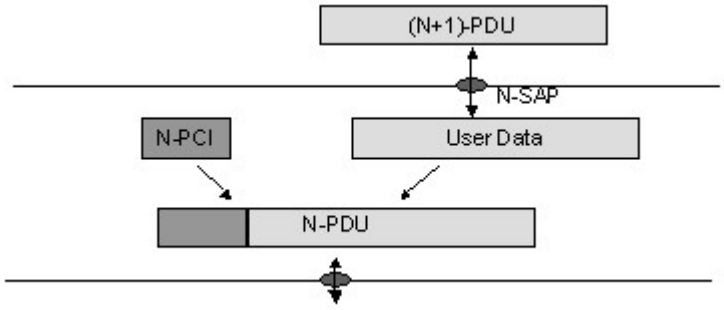



Ciascuno strato deve gestire i dati che gli provengono dallo strato superiore e aggiungere a questo quelle informazioni che servano per la realizzazione delle funzioni proprie dello strato in oggetto.

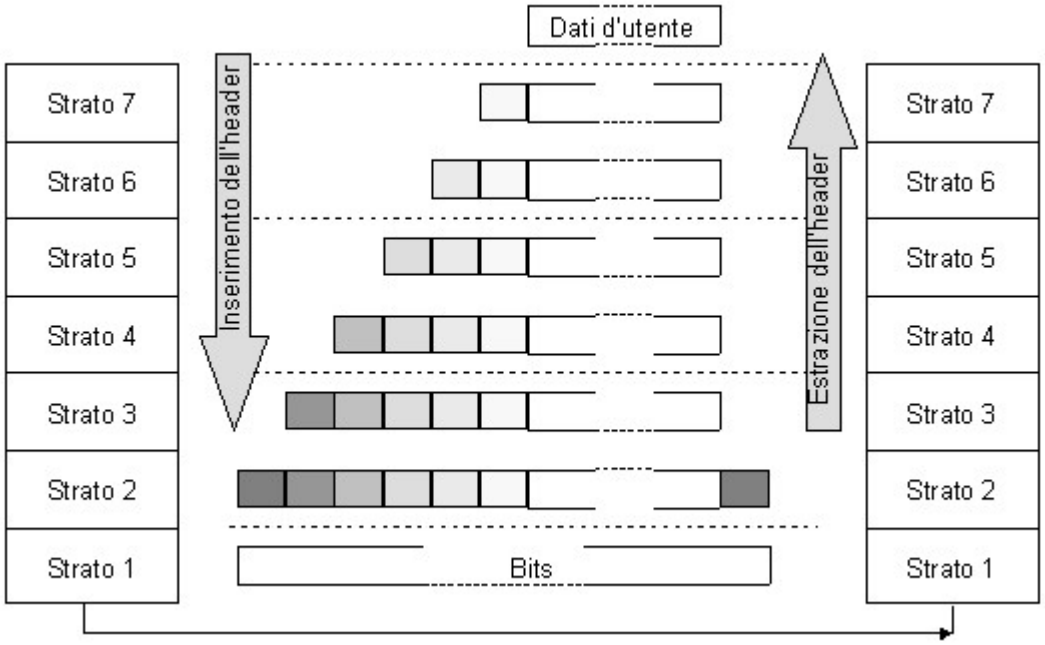
Si possono quindi evidenziare tre tipologie di dati:

- *User Data*:
  - I dati passati al livello N attraverso un N-SAP.
- *Protocol Control Information* (PCI):
  - Informazioni di controllo aggiunte dal livello N.
- **Protocol Data Unit** (**PDU**):
  - I dati trasferiti fra entità dello stesso livello.

Come mostrato in figura la PDU altro non è che l'unione di PCI e *User Data*. Le PCI sono anche dette **intestazione** o **header** della PDU.




Il trasferimento dei dati fra due applicazioni avviene quindi tramite un progressivo imbustamento di questi con PCI dei vari livelli, fino ad arrivare allo strato fisico che si limita alla trasmissione dei bit.




L'insieme della regole utilizzate dal generico strato N per il trasferimento dei dati, comprendente la specifica del formato e del significato delle PCI utilizzate si dice **protocollo** utilizzato dallo strato N.

Vedremo con maggiore dettaglio le funzioni che il modello di riferimento demanda a ciascuno strato.

Strato 1 - Fisico

Lo strato **fisico** è strutturato per regolamentare tutto ciò che riguarda le caratteristiche dell'interconnessione fisica fra due nodi della rete.

Per fare questo deve specificare le caratteristiche:

- meccaniche:
  - forma di prese e spine, numero di contatti;
- elettriche:
  - voltaggio e caratteristiche elettriche dei segnali associati all'interfaccia;
- funzionali:
  - significato dei vari segnali;
- procedurali:
  - combinazioni e sequenze dei segnali all'interfaccia necessarie al fine di regolarne il funzionamento;

Uno degli esempi più diffusi di standard di strato 1 è l'interfaccia di comunicazione seriale RS-232, emanato dall'EIA. Esistono anche standard per comunicazione parallela quali IEEE 488 e, più recenti standard per la comunicazione veloce fra calcolatori e periferiche quali l'*Universal Serial Bus* (USB).

Inoltre tutti gli standard per reti locali prevedono specifiche per l'interconnessione fisica del calcolatore al mezzo trasmissivo.

Nel caso in cui il calcolatore debba essere connesso ad una rete geografica, generalmente di tipo pubblico, solitamente ci si rifà ad uno schema di collegamento fisico in cui il calcolatore, denominato anche **DTE** (***Data Terminal Equipment***), utilizza per l'interconnessione alla rete un apparato di interfaccia denominato **DCE** (***Data Communication Equipment***). Lo standard di strato 1 in questo caso si prende carico di specificare solamente le caratteristiche che deve avere il

collegamento fra DTE e DCE, mentre non dice nulla relativamente alle modalità di connessione del DCE alla rete. Questo tipo di approccio ha il notevole vantaggio di rendere trasparente al terminale di utente (calcolatore) il tipo di tecnologia e di interfaccia di accesso utilizzata per l'interconnessione alla rete pubblica. Infatti le modalità di connessione alla rete possono essere cambiate, modificando il DCE. Se si mantiene costante l'interfaccia del DCE verso il calcolatore queste modifiche sono del tutto invisibili all'utente del calcolatore stesso.

Un esempio di questo approccio è il collegamento ad **Internet** tramite **modem**. Il calcolatore (DTE) si collega alla rete pubblica tramite un apparato detto modem (DCE) utilizzando una certa interfaccia di strato 1 (RS-232, USB, *wireless* ...). Qualora si decida di modificare il tipo di accesso alla rete pubblica, per esempio passando da un normale accesso di tipo telefonico all'accesso di tipo ADSL, è sufficiente cambiare il modem esistente con uno nuovo avente la medesima interfaccia verso il calcolatore di quello precedente, con il risultato che il collegamento alla rete viene modificato senza modificare il calcolatore.

Strato 2 - Data Link

Il livello **data link** è il secondo livello partendo dal basso nella pila di strati. Si occupa delle procedure di colloquio necessarie per un trasferimento di informazioni sufficientemente affidabile ed efficiente attraverso ogni linea di collegamento fra nodi di rete adiacenti e nodi di rete e terminali di utente. Nel caso delle reti locali, questo livello gestisce anche le procedure per l'accesso condiviso al mezzo trasmissivo.

Il livello *data link* fornisce servizi di :

- controllo e recupero di errore/perdita/duplicazione di dati;
- controllo e recupero di sequenza dei dati; i bit del livello fisico sono organizzati in trame (**frame**) cioè vengono raggruppati in pacchetti;
- **controllo di flusso**.

I servizi offerti da questo livello sono :

- senza connessione e senza riscontro; vengono inviati dei *frame* indipendenti e non vengono confermati dal destinatario quando questi vengono ricevuti, non viene instaurata una connessione diretta tra i 2 sistemi in comunicazione. Può capitare che alcuni *frame* non vengano ricevuti, e con questa strategia viene ignorato (a livello *data link*) il mancato recupero. è un servizio appropriato per reti con basso tasso di errore, con traffico che richiede una elevata trasparenza temporale (ad esempio per traffico vocale), in particolare viene usato nelle reti locali quando si preferisce la velocità all'integrità dei dati;
- senza connessione e con riscontro; caso analogo al precedente, solo che al momento della ricezione viene inviato dal destinatario un messaggio che conferma la corretta ricezione **(acknowledge - ack**) del *frame*. Il mancato ricevimento dalla sorgente del segnale *ack* comporta la **ritrasmissione** del *frame* non confermato. Questo servizio è utile per reti non affidabili, ad esempio connessioni *wireless*. È possibile che un *frame* non riscontrato sia spedito più volte, inoltre questo meccanismo di riscontro è utile ma non necessario, infatti è possibile implementarlo a livelli superiori;
- con connessione e con riscontro; è il servizio più sofisticato, prevede tre fasi, instaurazione della connessione, invio dei dati e chiusura connessione. In questo modo è possibile garantire che ogni *frame* sia consegnato correttamente e nell'ordine giusto. Viene fornito al livello di rete un flusso di bit affidabile.

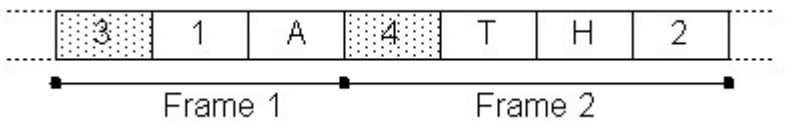
Un tipico esempio di funzionamento del livello *data link* è il seguente:

- in trasmissione:

    - segmenta il flusso di bit proveniente dal livello di rete in *frame*;
    - calcola un'apposita funzione, detta **checksum** per ogni *frame*, e la inserisce nel *frame* stesso;
    - consegna il *frame* come flusso di bit al livello fisico;
- in ricezione:
    - ricostruisce una serie di *frame* partendo dal flusso di bit fornito dal livello fisico;
    - per ogni *frame* ricalcola il *checksum*, se il *checksum* incapsulato nel *frame* corrisponde con quello appena calcolato, allora il *frame* è considerato esente da errori e viene accettato, altrimenti viene scartato.

Per delimitare i *frame* possono essere usate diverse strategie:

- conteggio dei caratteri; si utilizza un campo all'inizio del *frame* (*header*) per indicare di quanti caratteri è composto il *frame* stesso. Il problema di questo approccio è che se si dovesse rovinare il campo che contiene il conteggio dei caratteri allora il messaggio sarebbe completamente errato in quanto sarebbe impossibile arrivare all'inizio del *frame* successivo.



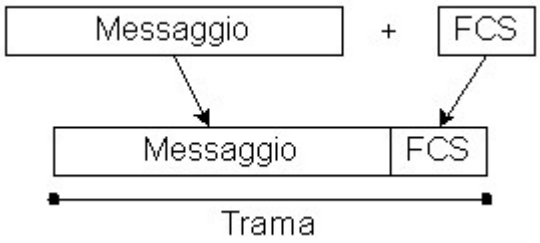

- Caratteri di inizio e fine, con *character stuffing*; viene usato un particolare carattere **ASCII** per delimitare l'inizio del *frame* ed un altro per la fine. In questo modo se la destinazione perde traccia dei confini di un *frame*, la ricezione del carattere di inizio permette il riallineamento. Sorge però un problema, ossia i *byte* che rappresentano i caratteri utilizzati per delimitare i *frame* possono essere contenuti all'interno del messaggio stesso. Per evitare questo inconveniente, quando si presenta un tale *byte* all'interno del *frame*, il livello ne aggiunge un altro subito dopo, in modo che solo i *byte* che rappresentano l'inizio e la fine sono contenuti una volta sola nel flusso, quando se ne presentano due il secondo viene rimosso prima di essere trasferito al livello superiore. Questa tecnica viene detta di *character stuffing*.
- *Bit pattern* di inizio e fine con **bit stuffing**; la tecnica precedente è legata all'utilizzo di codifica ASCII, e può non andare bene per codifiche piu moderne. Per evitare questo problema viene utilizzata una particolare sequenza di bit (*bit pattern*) che indicano l'inizio e la fine del *frame*, ad esempio la sequenza 01111110. Anche in questo caso può presentarsi questa sequenza all'interno dei bit da trasmettere, perciò se si presentano 5 bit pari a 1 consecutivamente da trasmettere, il livello *data link* inserisce uno 0 successivamente a questa serie di 1. In ricezione se si presentano una serie di 5 1 consecutivi, viene rimosso lo 0 che segue. Pertanto la sequenza 01111110 può apparire solo all'inizio e alla fine dei *frame*. Questa tecnica viene detta *bit stuffing*.
- Violazione della codifica; in molte reti, soprattutto nelle **LAN**, si codificano i bit al livello fisico con ridondanza, ossia gli 1 vengono trasmessi con una coppia alto/basso di tensione, mentre gli 0 con una coppia basso/alto. In questo modo non si incontrano coppie del tipo alto/alto e basso/basso. Queste due coppie possono essere utilizzate per delimitare i *frame*.

L'altro importante compito dello strato di linea è quello di controllare gli errori di trasmissione sui bit, che possono verificarsi con modalità diverse a seconda del mezzo utilizzato. Gli errori sono dovuti in genere a rumori di fondo, disturbi improvvisi e interferenze. Gli approcci al problema sono due:

- possiamo includere una quantità di informazione aggiuntiva in modo da poter ricostruire il messaggio, ossia una strategia di correzione dell'errore;
- includiamo una quantità minore di informazione aggiuntiva solo per permettere il riconoscimento dell'errore in fase di trasmissione, ossia riconoscimento dell'errore.

La parte che viene aggiunta si chiama *Frame Sequence Check* (FCS) e dipende dal tipo di codice

utilizzato. I codici di *Hamming* permettono la correzione dell'errore e sono adatti quando il tasso di errori è così alto che si impiega meno tempo a mandare un messaggio più lungo (con più bit di ridondanza) piuttosto che ad aspettare la ritrasmissione. I codici polinomiali o **Codici a Ridondanza Ciclica** (**CRC**) permettono solo il rilevamento dell'errore, ma richiedono meno bit di ridondanza rispetto ai codici di *Hamming*.




Strato 3 - Rete

Il livello di **rete** ha come compito quello di instradare i pacchetti dalla sorgente fino al destinatario, attraversando i nodi intermedi (**router**) che uniscono le varie *subnet*. La differenza tra il livello di rete ed il livello **data link** è che il livello *data link* deve solo trasferire i pacchetti da un capo all'altro del ramo, mentre il livello di rete deve effettuare una vera e propria funzione di instradamento.

I compiti di questo livello sono di conoscere la topologia della rete, di scelta di percorso, di gestire il flusso dei dati e le eventuali congestioni, di gestire la presenza di più reti differenti.

I servizi offerti dal livello di rete possono essere sia con connessione che senza:

- con connessione; le **entità** coinvolte nella comunicazione stabiliscono una connessione, negoziando i vari parametri, e a questa connessione viene associato un identificatore. Questo identificatore viene inserito in ogni **pacchetto** inviato dalle due entità, la comunicazione è bidirezionale e i pacchetti viaggiano in sequenza lungo il percorso prestabilito in fase di negoziazione. Con questa strategia il **controllo di flusso** è operato automaticamente grazie ai parametri prestabiliti all'inizio. In questo modo si opera in modo da fornire un servizio di tipo affidabile;
- senza connessione; la sottorete viene considerata inaffidabile, pertanto sono il sorgente e il destinatario del flusso informativo che devono preoccuparsi di gestire sia gli errori che il controllo di flusso, in pratica è il livello di trasporto che si deve occupare di queste cose. Il servizio offerto è di tipo **datagramma**, cioè i pacchetti viaggiano indipendentemente l'uno dall'altro e devono contenere tutti un **indirizzo** di destinazione.

Per realizzare correttamente la principale funzione del livello di rete, ossia l'instradamento o **routing**, sono necessari opportuni algoritmi. L'algoritmo di *routing* deve calcolare su quale uscita di un commutatore instradare il flusso dati in ingresso. Se il servizio è con connessione, questo algoritmo si applica solo in fase di *setup* della connessione, se invece è senza connessione allora si applica su ogni pacchetto.

I requisiti per un algoritmo di *routing* sono:

- semplicità, al fine di non richiedere ai nodi grandi sforzi di elaborazione;
- robustezza, per garantire buone funzionalità anche in presenza di malfunzionamenti della rete;
- stabilità, per non creare inconsistenze che possano rendere inefficace l'intradamento dei dati;
- equità al fine di fornire la stessa qualità di servizio a tutte le connessioni;
- ottimalità nelle scelte di percorso.

Gli algoritmi di *routing* possono essere statici e adattivi. Gli algoritmi statici sono eseguiti solamente

all'avvio della rete, e le decisioni prese non vengono più modificate. Rientrano in questa classe gli algoritmi:

- *shortest path routing*; ogni **router** viene considerato come un **nodo**, e una connessione punto punto come un ramo. Vengono calcolati i cammini minimi tra ogni coppia di nodi e vengono inviati a ogni *router*. I cammini minimi vengono calcolati in base al numero di nodi che devono essere attraversati, alla lunghezza dei rami, tempo medio di immagazzinamento e rilancio;
- **flooding**; il pacchetto viene rinviato su tutti i rami tranne quello da cui è arrivato. In questo modo però si potrebbe generare un numero infinito di pacchetti, quindi per ridurre il traffico generato si possono utilizzare alcuni stratagemmi. Uno di questi richiede l'inserimento nei pacchetti di un numero massimo di nodi da attraversare, dopodiché se questo numero viene superato allora il pacchetto viene scartato.Un altro richiede la verifica da parte di ogni *router* che quel pacchetto non sia già transitato, altrimenti lo scarta. Questò è l'algoritmo più robusto e affidabile anche se genera una quantità di dati tale da non essere usabile con efficacia;
- *flow-based routing*; questo algoritmo effettua una stima del traffico atteso su ogni ramo, poi calcola il tempo medio di attraversamento, quindi decide su quale ramo instradare.

Nelle reti moderne sono in uso algoritmi dinamici di *routing*, in grado di adattarsi ai cambiamenti della rete stessa. Questi algoritmi funzionano non solo all'avvio della rete, ma rimangono in esecuzione durante il normale funzionamento della rete stessa. Fanno parte degli algoritmi dinamici di *routing*:

- ***distance vector*** *routing*; ogni *router* mantiene al proprio interno una tabella in cui vengono indicizzati tutti gli altri *router* conosciuti fino a quel momento nella rete. In questa tabella viene memorizzato per ogni altro *router* la distanza e il ramo d'uscita per arrivarci. Il *router* a intervalli di tempo manda degli speciali pacchetti chiamati *echo* a tutti gli altri **nodi adiacenti** e misura il tempo di risposta. Appena completa la tabella la invia ai nodi adiacenti. In questo modo non viene a conoscenza del *router* la topologia totale della rete, ed inoltre la convergenza dell'algoritmo è piuttosto lenta quando si verificano eventi che modificano la topologia della rete;
- ***link state*** *routing*; ogni *router* controlla lo stato dei collegamenti con i *router* adiacenti, misurando i ritardi di attraversamento, e distribuisce queste informazioni agli altri nodi adiacenti. Considerando tutti i pacchetti arrivati, si costruisce un grafo della rete e si calcola il cammino minimo per ogni nodo della *subnet*. Questo algoritmo è molto usato, ad esempio una sua implementazione in **Internet** è piuttosto affermata, questa corrisponde al nome di **OSPF** - *open shortest path first*.

Quando la grandezza della rete diventa tale da non permettere un efficace utilizzo di questi algoritmi, viene utilizzato il metodo del **routing gerarchico**. La rete viene suddivisa in regioni, e gli algoritmi si applicano su due livelli, all'inizio si opera all'interno di una regione, successivamente si applica l'algoritmo una seconda volta su tutti i *router* che fanno parte del confine di una regione.

Strato 4 - Trasporto

Scopo dello strato di **trasporto** è fornire un **canale** sicuro *end-to-end*, svincolandoli da tutti i problemi di rete.

Si occupa tipicamente di adattare la dimensione dei frammenti forniti dagli strati superiori (*files*) a quella richiesta dalle reti (**pacchetto**):
funzione di Pacchettizzazione (*fragmenting*)

Può avere molte altre funzioni fra cui:

- **controllo dell'errore**;
- **controllo di flusso**;
- gestione di dati prioritari, eccetera...

Non tutti le applicazioni hanno bisogno delle stesse funzioni, per cui si possono definire diverse Classi di servizi di trasporto.

Ad esempio nel modello **Internet** lo strato di trasporto prevede diversi protocolli per la fornitura di diverse tipologie di servizio. I più usati fra questi protocolli sono:

- **TCP** per un trasferimento dati *end-to-end connection oriented* con recupero degli errori e controllo del flusso;
- **UDP** per un trasferimento dati *end-to-end connectionless*;
- RTP per un trasferimento dati *end-to-end* che rispetti il più possibile stretti requisiti di temporizzazione.

Ad esempio un'applicazione di trasferimento *file* tipicamente utilizzerà TCP, mentre l'applicazione di invio di un **e-mail** utilizzerà UDP ed un collegamento audio via Internet utilizzerà RTP.

Strato 5 - Sessione

Lo strato di **sessione** suddivide il dialogo fra le applicazioni in unità logiche (dette appunto sessioni), in modo tale che una sessione possa essere individuata, interrotta e ripresa, per fare fronte a vari eventi catastrofici: perdita di dati, caduta della linea, momentaneo *crash* di uno dei due interlocutori...

Permette la chiusura ordinata (*soft*) del dialogo, con la garanzia che tutti i dati trasmessi siano arrivati a destinazione.

Anche gli strati di sessione hanno molte funzionalità e possono essere più o meno completi a seconda delle richieste delle applicazioni.

Strato 6 - Presentazione

Lo strato di **presentazione** adatta il formato dei dati usato dai due interlocutori preservandone il significato.

La descrizione del tipo di dati usati per una applicazione e del loro formato si dice una sintassi. Ogni interlocutore ha una sua Sintassi locale e durante il dialogo bisogna concordare una Sintassi di trasferimento. È stato definito un linguaggio detto *Abstract Sintax Notation* 1 (ASN 1) per descrivere e negoziare le sintassi.

Nell'architettura dei protocolli di **Internet** non sono previsti strati di sessione e presentazione, per cui le relative funzioni sono demandate alle applicazioni. Per questa ragione, ad esempio, dobbiamo preoccuparci di specificare il tipo di codifica quando inviamo un allegato ad un **e-mail**, oppure dobbiamo ricominciare da capo una navigazione se per qualche ragione cade il collegamento. Se queste funzioni fossero previste in rete, la rete stessa si preoccuperebbe di svolgerle, senza renderle visibili all'utente.

Strato 7 - Applicazione

Lo strato di **applicazione** è l'utente della rete di calcolatori.

Rappresenta il programma applicativo (o Applicazione) che per svolgere i suoi compiti ha bisogno di

comunicare con altre applicazioni remote.

Sono applicazioni i programmi che utilizziamo normalmente per accedere a servizi di rete, quali il ***browser*** Internet, il programma di invio e ricezione dell'**e-mail**, i programmi di trasferimento *file* e condivisione delle risorse eccetera.

Secondo il modello di riferimento ISO-**OSI** l'applicazione dovrebbe occuparsi solamente dell'interazione con l'utente e della gestione dei dati relativi. Come già detto nel caso di **Internet** all'applicazione sono demandate anche funzioni di gestione della sessione e di presentazione dei dati.