Lezione 3: Il modello OSI

Un modello per l'architettura a strati: OSI

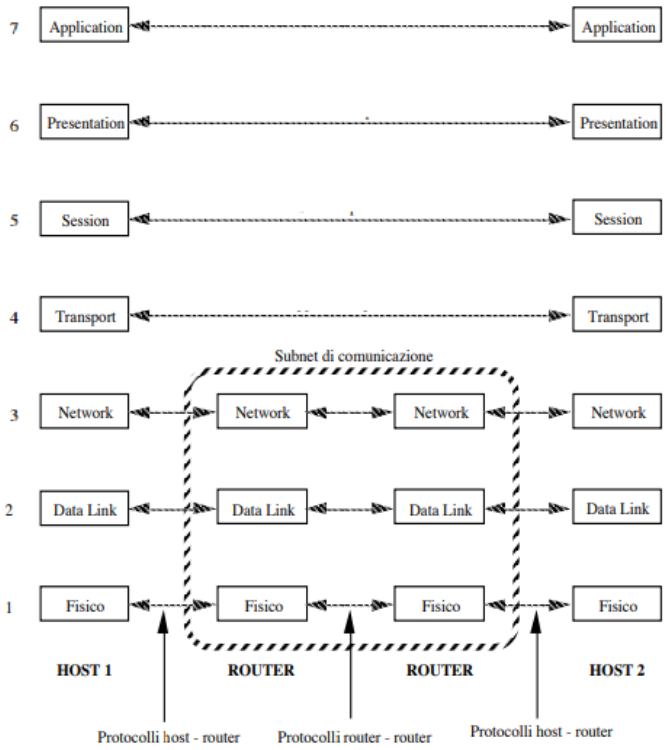
A partire dai problemi visti precedentemente, **l'ISO - International Standard Organization** ha definito nel 1988 il modello **OSI - Open Systems Interconnection**.

Gli obbiettivi dell'OSI erano:

- Fornire uno standard per la connssione a **sistemi aperti (open systems)**
- Fornire un modello per lo **sviluppo di standards** per sistemi aperti all'interconnessione
- Fornire un **modello per confrontare** diverse architetture di reti.

Per raggiungere questi obbiettivi:

- Il modello si basa su un **concetto di strati**.
- Ogni strato dovrebbe avere **delle funzioni ben definiti a diversi livelli di astrazione**
- Il numero e funzioni ad ogni livello dovrebbe essere selezionato per:
    - Evitare troppe funzioni per ogni strato
    - Evitare troppi strati per problemi di inefficienza
    - Minimizzare lo scambio di informazioni tra strati



OSI model is not a network architecture

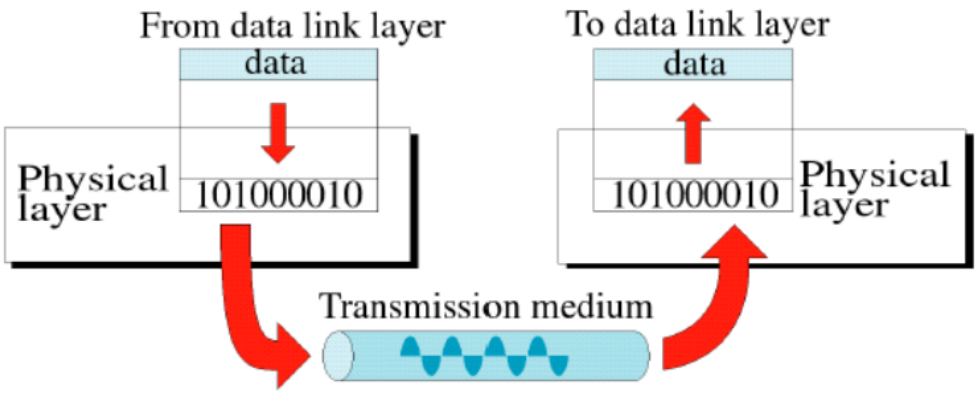
Il numero di strati selezionati per il modello OSI è di 7.

Perchè abbiamo precisamente 7 livelli?

Il primo motivo è che avere troppi, o troppo pochi, livelli, non sarebbe la scelta giusta; questo perchè ogni livello deve essere in grado di comunicare con i livelli adiacenti, e qualora avessimo troppi livelli, introdurremmo un **overhead** notevole, proprio a livello di funzioni usate per comunicare tra livelli.

Strato fisico

Importante: partiamo dal livello più basso.



Questo livello ha due grandi responsabilità: da un lato deve consentire la trasmissione corretta e controllata dei segnali sul collegamento fisico, in modo che, in ricezione, si possa ricostruire la medesima sequenza di bit trasmessa. Un'altra responsabilità del livello è quella di definire le caratteristiche meccaniche, elettriche e procedurali dell'interfaccia di rete.

in poche parole

Questo strato punta ad abilitare la trasmissione di bits su un mezzo di trasmissione.

A questo punto, i seguenti aspetti sono importanti:

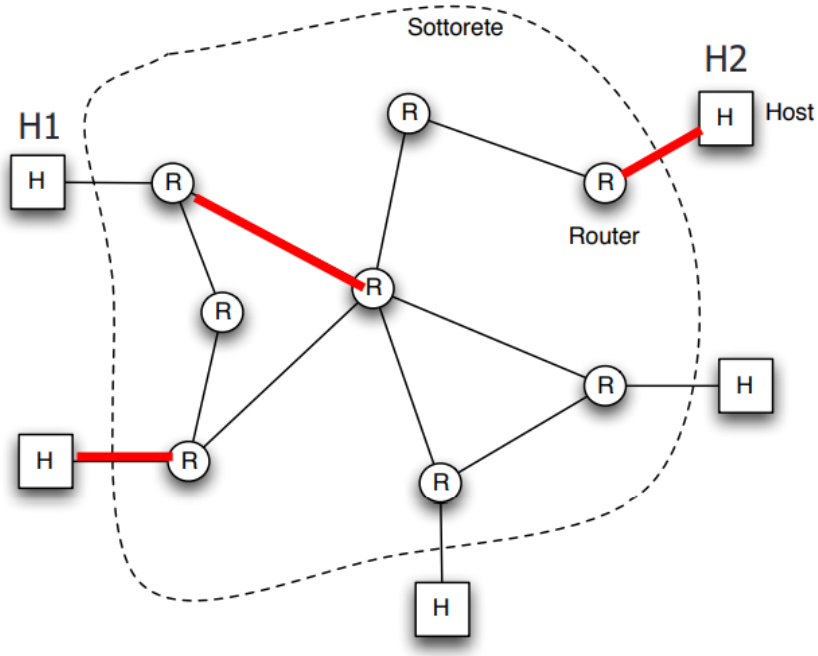
- Avere **una trasmissione di segnali corretti e controllati** in modo da garantire la ricezione della stessa **sequenza di bit che è stata trasmessa**.
- Avere delle caratteristiche **meccaniche, elettroniche e procedurali** di **network cards**, ovvero componenti hardware che connettono il computer al mezzo di trasmissione.

Lo strato è inoltre responsabile di definire:

- Il voltaggio di segnale per rappresentare il **valore logico 0 ed 1**.
- La **durata** del segnale elettrico che caratterizza **un bit**.
- Possibile **trasmissione simultanea** in entrambe le direzioni
- Aspetti meccanici dei connettori adottati per collegare un computer al mezzo di trasmissione.

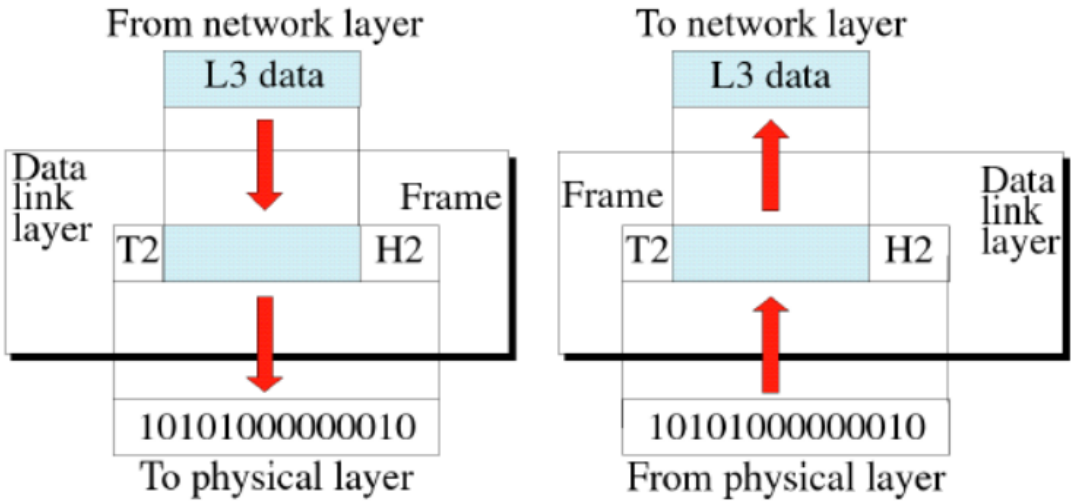
Comunicare tra due nodi

Il layer del collegamento è responsabile della comunicazione **diretta** tra due macchine, che possono essere due router all'interno della rete, o possono essere un host ed un router. In ogni caso parliamo di comunciazione tra **due** interfacce di rete (ad esempio un'interfaccia presente nell'host e l'altra presente nel router) che comunicano in maniera **diretta** attraverso un **link dedicato**.



Host = End System
Router = Nodo Intermedio = Nodo di commutazione

Data link Layer



Questo livello si occupa della trasmissione dei dati (**che riceve dal livello superiore - livello 3**) utilizzando i meccanismi offerti dal livello sottostante (**livello 2 - livello fisico**).
Il livello quindi aggiunge ai dati provenienti dal livello superiore un'intestazione (**header**) ed un **trailer**, ovvero delle informazioni di controllo presenti sia in testa che in coda alla sequenza di bit inviata.

Uno degli obbiettivi principali di questo layer è il **framing**, ovvero la suddivisione della sequenza di bit che inviamo, in cosiddetti **frames**; abbiamo quindi una sequenza di bit **delimitata**, detta proprio frame. L'obbiettivo è quindi fornire dei delimitatori, usati per specificare che una data sequenza di bit inviata sul mezzo fisico, appartiene ad un dato frame e non ad un altro.

Perchè usare i frames? Perchè uno dei problemi affrontati a questo livello è la rilevazione e correzione degli errori; infatti in entrambi i casi abbiamo bisogno di lavorare su una sequenza delimitata di bit: nel caso della rilevazione abbiamo bisogno **di aggiungere informazioni di controllo** che ci consentono di capire se la seq è stata alterata, o meno, e nel caso in cui sia corrotta, richiediamo la ritrasmissione.

Introdurre i delimitatori serve quindi a stabilire cosa bisogna osservare per **capire se sono presenti degli errori**, e sopratutto cosa deve essere ritrasmesso. Nel caso di ritrasmissione sarebbe opportuno avere dei **frames** non troppo grandi, in modo tale da dover ritrasmettere una quantità limitata di dati.

Un'altra responsabilità di questo livello abbiamo la **gestione del traffico**, o meglio **il controllo del flusso**; dobbiamo quindi fare in modo che nella comunicazione tra due macchine, chi spedisce non deve **sovraccaricare** chi riceve. Questo perchè il ricevente avrà sicuramente un buffer, ma se questo viene saturato diventa inutile.

Indirizzamento: l'indirizzamento riguarda diversi livelli, ma in questo livello è affrontato a *basso livello*.

In poche parole

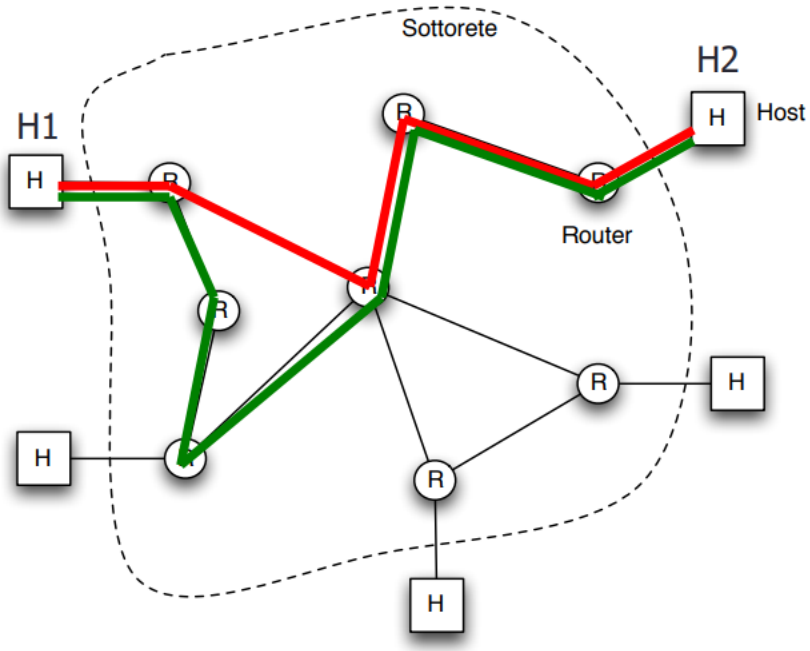
Questo strato espone lo strato superiore al collegamento di trasmissione senza errori. A questo punto, il layer:

- **Frammenta** i dati che arrivano dal lato superiore in **sequenze di bit** con una lunghezza fissa, chiamate **frames**.
- **Invia** la sequenza di frames.

Per questo motivo, gli aspetti della progettazione di questo livello riguardano:

- **Framing**: ovvero l'introduzione di **delimitatori** alla *raw sequence* di bits
    - Selezione dei delimitatori
- **Gestione degli errori alla destinazione**: rilevamento con un'eventualre **ritrasmissione o correzione**.
- **Gestione del traffico:** per evitare un **overflow al ricevente**.
- **Controllo al *medium access***
- **Addressing**

Comunicare tra due Hosts

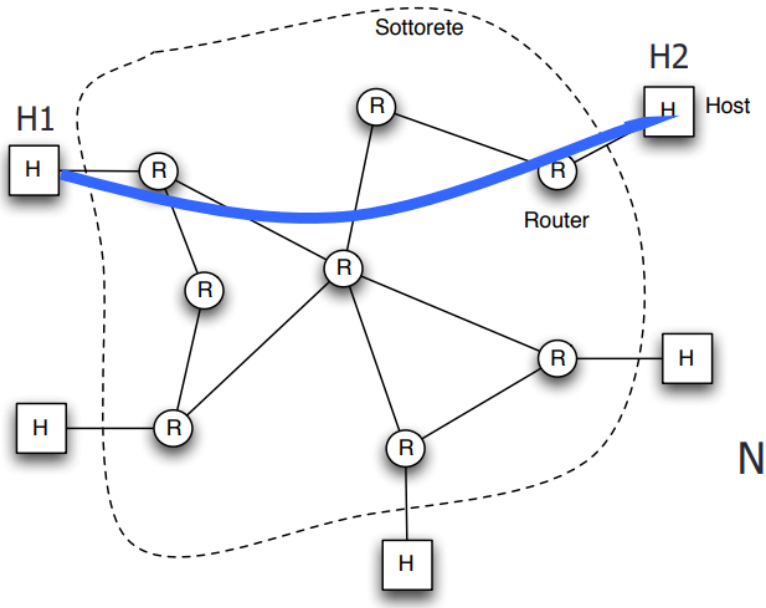


Host = End System
Router = Nodo Intermedio = Nodo di commutazione

Al di sopra del collegamento, per consentire la comunicazione tra host, abbiamo la necessità che i nodi intermedi lungo il percorso, si attivino per costruire dei percorsi. Nella figura si nota il problema da affrontare: l'host per comunicare deve usare di link, scelti di volta in volta dai **nodi intermedi**. Vediamo in colori diversi le alternative possibili per collegare H1 ad H2.

0:18 lezione 3

Canale Logico tra due hosts

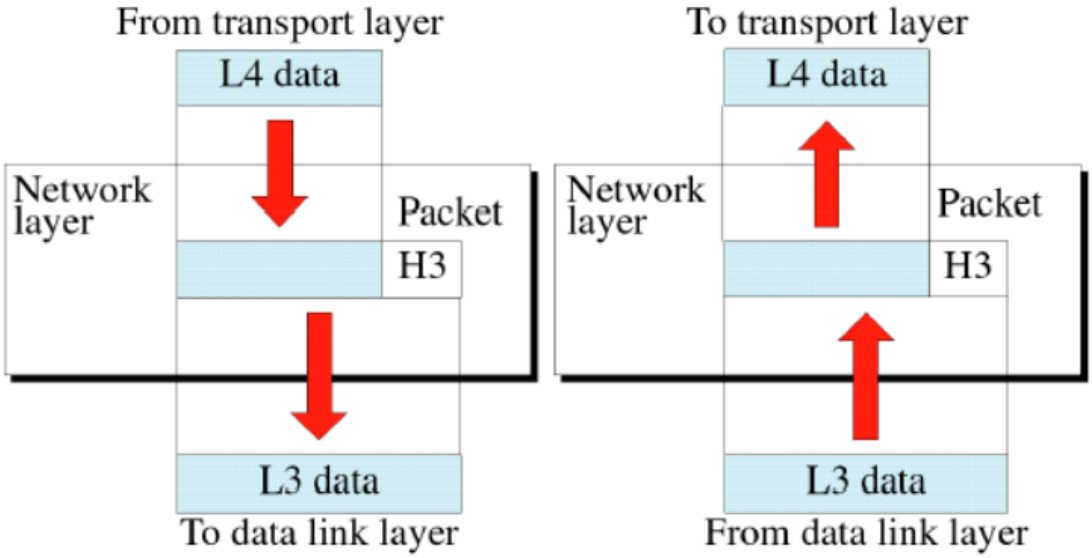


Network level

Host = End System
Router = Nodo Intermedio = Nodo di commutazione

L'obbiettivo del livello 3 è di costruire un canale logico, mappato sul link della rete, che consenta la comunicazione tra due host. Nell'immagine si vede il canale logico, che fisicamente potrebbe essere uno (o entrambi) dei percorsi visti nell'immagine precedente.

Network Layer



Questo livello è particolarmente importante per consentire il funzionamento di una rete. I dati provenienti dal lvl 4 vengono estesi con un'intestazione specifica e la PDU (protocol data unit) di livello 3 viene affidata al livello sottostante (collegamento).

Addressing

Una responsabilità di questo lvl è sicuramente **l'indirizzamento**, da non confondere con l'indirizzamento del lvl 2: Mentre il lvl 2 utilizza l'indirizzo per contattare l'host direttamente connesso (link dedicato, ovvero ai capi del collegamento ci sono solo i 2 host che devono comunicare), nel caso del livello di rete, dobbiamo prevedere come **indirizzo, l'indirizzo dell'host che vogliamo raggiungere**. Quindi l'indirizzo che H1 inserirà all'interno del pacchetto da inviare, sarà l'indirizzo di H2; sarà un indirizzo **non modificato** durante tutto il percorso che il pacchetto seguirà all'interno della rete.

Routing

Un altro compito di questo strato è quello del **routing**, ovvero la scelta del percorso ottimale da usare per spedire il pacchetto. La scelta del link non è banale, perchè la scelta errata di esso potrebbe produrre dei percorsi più lunghi del dovuto, o addirittura **cicli**, che fanno rimanere il pacchetto all'infinito all'interno della rete.

Gestione delle congestioni

Quando ragioniamo a livello di rete, il problema del traffico si concretizza in maniera diversa: è possibile a livello di rete che un router sia raggiunto da dati provenienti da altri router, ed a causa di picchi, è possibile che il router non riesca a smaltire il traffico in arrivo. Quindi i buffer associati ai link di ingresso vengono saturati a causa dell'attività contemporanea di altri router che inviano

molti dati ad un unico router. Questa situazione viene chiamata **congestione**, ovvero la saturazione di alcuni **buffers del router**, che impediscono quindi la ricezione.
Questo è un altro problema che deve essere risolto o in maniera **preventiva**, ovvero cercando di evitare che questa situazione si verifichi, oppure nel momento in cui si verifica, dobbiamo gestire il problema.

Frammentazione

Un altro problema da risolvere a questo strato è quello della **frammentazione dei pacchetti**, in **frames** che devono essere costruiti dal **livello del collegamento dati**.
I router che sono responsabili dell'istradamento dei pacchetti si trovano ad operare con link diversi, e ciascuno di questi link **non è detto sia caratterizzato dalla stessa tecnologia trasmissiva**, quindi è possibile che il **livello 2 - livello del collegamento** di un dato link, sia diverso da quello di un altro link.

Creazione di connessioni network

Quando parliamo di reti **orientate alla connessione**, parliamo di reti che prima di una trasmissione stabiliscono una connessione tra host: vedi il capitolo della lezione precedente.

Quindi, le due entità comunicanti, prima della trasmissione chiedono alla rete delle **risorse** per poter comunicare; la rete si predispone alla comunicazione, in alcuni casi **riservando delle risorse** per quella data comunicazione.

Data conversion

Nei casi in cui la rete lavori con protocolli diversi (diversi tipi di link) una responsabilità della rete è quella di realizzare **l'adattamento** tra le parti diverse della rete: un router potrebbe quindi dover adattare un certo protocollo ad un altro protocollo. Un router che fa questo viene chiamato **gateway**, o **router multi protocollo**, che è in grado di adattare il pacchetto in arrivo (caratterizzato da un certo formato), ad un altro formato.

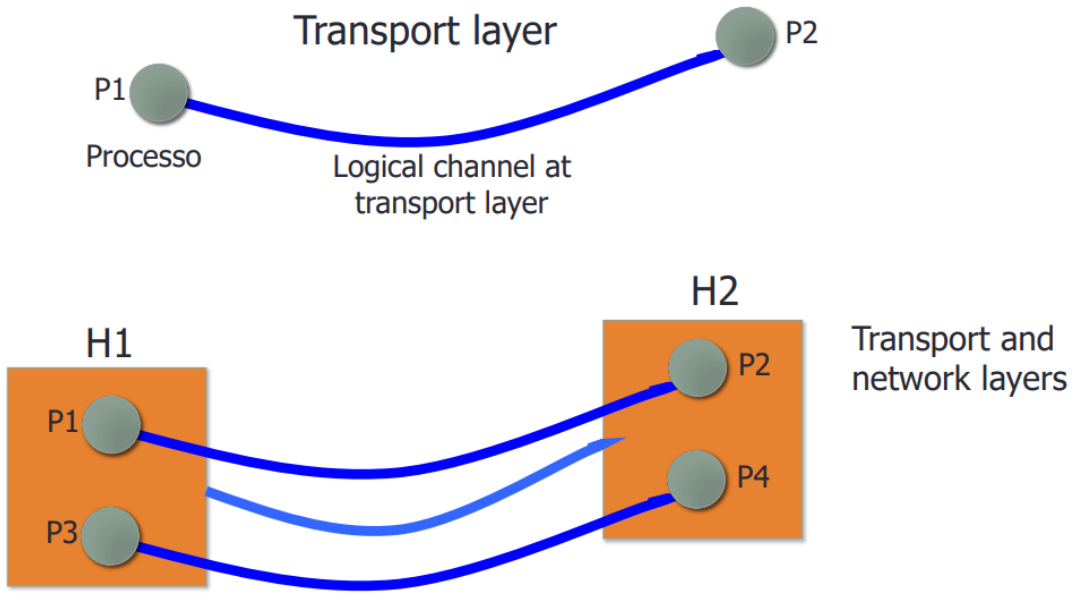
🏁 0:32 lez 3p

In poche parole

Questo layer è responsabile di gestire le **sottoreti** composte da ***switching nodes***; A questo punto i seguenti aspetti sono importanti:

- **Addressing**
- **Routing**: ovvero la selezione del **percorso ottimale** per assicurarsi la consegna delle informazioni
    - Selezione del percorso **statica**.
    - Selezione del percorso **dinamica, per ogni pacchetto**.
- **Gestione delle congestioni**
- **Frammentazione**: serve per adattare la grandezza dei pacchetti a tecnologie diverse
- **Creazione di connessioni network**
- **Conversione dei dati:** per gestire il routing tra due reti con diverse caratteristiche:
    - **Traduzione degli indirizzi**
    - **Adattamento del protocollo** attraverso il gateway.

Canale logico tra due processi



Il livello successivo, ovvero quello di trasporto, vuole consentire la comunicazione tra due entità logiche di esecuzione, tipicamente **processi**. L'obbiettivo è quindi quello di consentire la comunicazione tra due processi attraverso un canale logico leggermente rispetto a quello precedente.

Infatti, il livello 3 consente la comunicazione **tra host** ed il livello 4 (questo, livello di trasporto) consente la comunicazione tra **processi**.

E' possibile che tra due host ci siano **più coppie di entità logiche comunicanti**; questo è un esempio di multiplazione, proprio perchè consentiamo la creazione di canali logici tra processi, pur usando un unico link fisico.

Strato di trasporto

Questo è il primo livello che vediamo che viene detto **end-to-end**, ed ha come responsabilità la costruzione del canale logico visto nell'immagine precedente.

Controllo del flusso

Abbiamo, anche in questo livello, lo stesso problema di **controllo del flusso** che avevamo nel livello precedente: un processo potrebbe inviare dati troppo velocemente rispetto a quelli che un altro può ricevere; anche in questo caso potrebbe essere necessario prevedere dei meccanismi per regolare l'attività del mittente, facendo in modo che il processo mittende abbassi il suo **throughput.**

Potremmo avere un'alterazione dei pacchetti inviati, quindi il ricevente deve essere in grado di "accorgersi" di un'eventuale perdita di dati.

Perchè ha senso prevedere un meccanismo di rilevamento degli errori al livello di trasporto se il canale logico prevede comunque che ciascun link fisico sia affidabile?

Potremmo avere errori di diversa natura, ovvero errori non generati dai link che separano i diversi host; la ragione più importante, assumendo che lato software non ci siano dei bugs, è che lato hardware potrebbero verificarsi degli errori di codifica, e quindi trasmettendo dei pacchetti codificati male, ma trasmessi bene, il ricevente successivo non avrebbe idea di aver ricevuto un pacchetto sbagliato.

Frammentazione dei dati

i dati provenienti dai livelli più alti vengono frammentatie poi riassemblati alla ricezione.

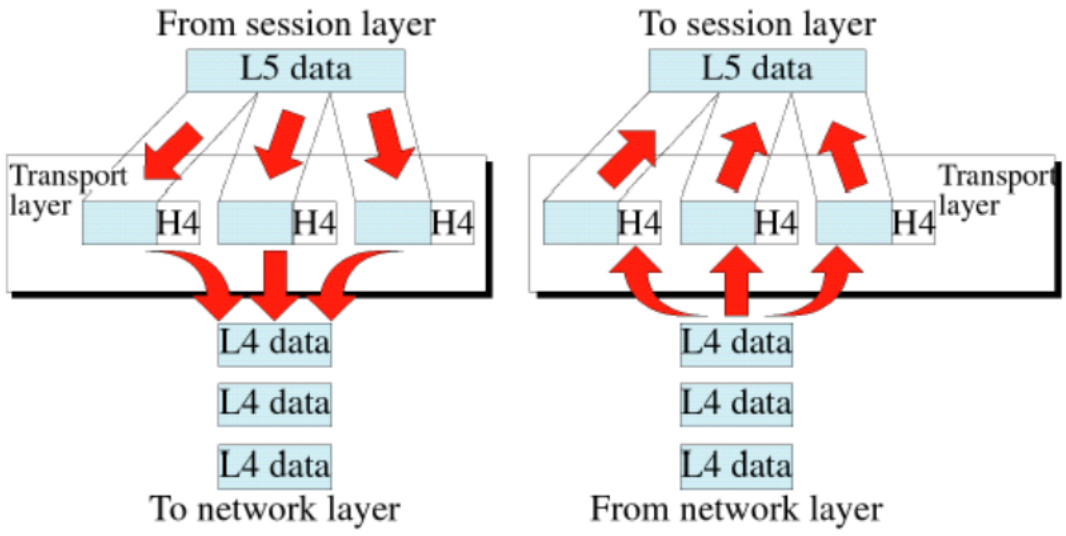
In poche parole

E' il primo strato di tipo **end-to-end, quindi sender to receiver**; la comunicazione tra **entità peer** a questo livello è eseguita **senza intermediari**.

Lo strato è responsabile della **frammentazione** dei dati che arrivano dallo strato superiore in pacchetti (**segmenti**) per essere trasmessi in maniera affidabile attraverso il **network layer**.

Gli aspetti di progettazione riguardano:

- **Frammentazione e riassemblaggio**
- **Controllo del flow end-to-end**
- **Gestione degli errori end-to-end**
- **Modelli di servizio per la comunicazione end-to-end**



Session Layer

A questo punto possiamo immaginare le entità comunicanti come delle unità logiche, quindi dei **processi**, e non delle macchine.

Gestione del dialogo

Gestire il dialogo significa fare in modo, qualora ci siano dei **malfunzionamenti transitori della rete**, il dialogo tra due entità non deve essere **ripreso da 0** (nel momento in cui la rete viene ripristinata).
Quindi un concetto importante introdotto a questo livello, è il concetto di **sessione**: per sessione intendiamo un **intervallo temporale** durante il quale si realizza il dialogo tra due o più entità comunicanti.

Attivare una sessione significa **iniziare il dialogo**; durante questo intervallo le unità si scambiano delle **unità di dialogo**, ad esempio è possibile che queste entità si scambino **dei files**, che sono un esempio di unità di dialogo.
Fare in modo che il dialogo non debba essere riavviato completamente, significa **memorizzare delle informazioni di controllo** che sono utilizzate dagli interlocutori per capire **cosa si sono scambiati fino a quell'istante**.

Punti di sincronizzazione

Per gestire il dialogo, e fare quindi in modo che la conversazione non vada persa in caso di problemi, vengono usati dei meccanismi di sincronizzazione:

- **Maggiori**: sono usati per delimitare le unità di dialogo (ad esempio i files), quindi ricevere un punto di sincronizzazione maggiore significa aver ricevuto un'unità di dialogo che precede il punto di sincronizzazione; quel punto di sync viene memorizzato così nel caso errore i due interlocutori saprebbero esattamente da che punto riprendere la conversazione.
- **Minori**: Per evitare di ritrasmettere l'intero file a seguito di un malfunzionamento verificatosi durante la trasmissione del file, vengono utilizzati dei punti di sync detti "minori"; sono scambiati dopo un certo numero di byte per fare in modo che in caso di un malfunzionamento si debba ritrasmettere solo quel numero di byte.

Quindi - Importante

Trasmettere un file, ovvero una sequenza di byte **significativa**, non significa poter incapsulare il file in un unico pacchetto ed inviarlo, ma questo viene spezzato in tanti blocchi, ognuno dei quali viene incapsulato in un pacchetto il quale viene poi inviato.

E' possibile quindi che il singolo blocco venga perso, ma garantisce che il file nella sua interezza venga recapitato. E' molto raro vedere un intero file trasmesso in un unico pacchetto.

In poche parole

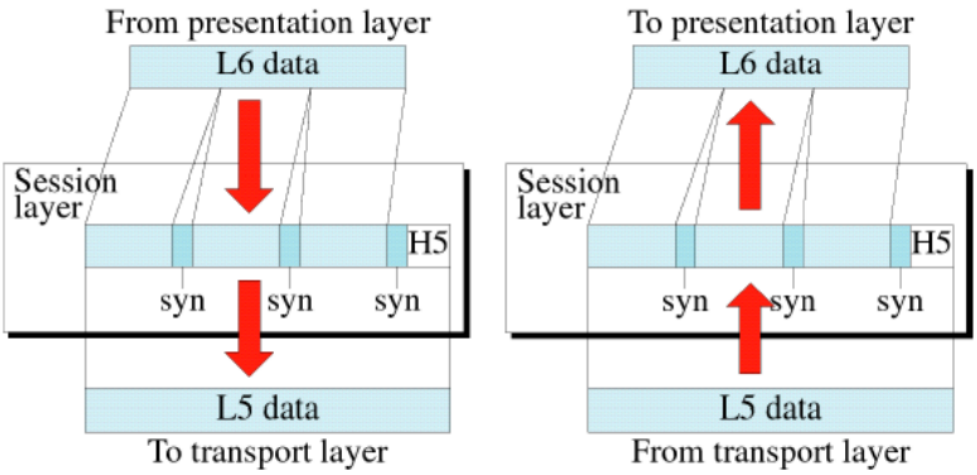
Questo strato è responsabile per il **dialogo** tra processi usando il concetto di **sessione**. Questo strato riduce le responsabilità dello **strato applicazione** riguardanti le funzioni specifiche per il trattamento dei dati.

Le funzioni principali offerte da questo strato sono:

- **Dialog management:**
    - **Setup della sessione** con le entità peer
    - Gestione del dialogo: l'abilità di recuperare un dialogo dopo un'interruzione.
    - **Tear-down** della sessione
- **Sincronizzazione:**
    - Adozione di **punti di sincronizzazione** durante il data-flow

Abbiamo due tipi diversi di **punti di sincronizzazione**:

- **Major - maggiore**: permette lo scambio di unità **indipendenti** (ad esempio files)
- **Minor - minore:** introduce delimitatori all'interno delle unità di dialogo per permettere il recupero quando alcuni dati non vengono correttamente ricevuti.



Presentation layer

Questo livello è ancora una volta un livello **non strettamente necessario**, ma lo è nel momento in cui le due unità che vogliamo far comunicare sono **eterogenee**, ovvero **sono processi in esecuzione su macchine diverse, con HW diverso e con SO diversi.**

Questo livello introduce la **rappresentazione astratta dell'niformazione da scambiare**; si intende dire che l'informazione che si vuole scambiare deve essere caratterizzata in modo indipendente dalla sua rappresentazione concreta con l'HW usato: se i due processi vogliono scambiarsi un intero, devono dichiarare di voler scambiare un intero; successivamente questo intero verrà rappresentato in diverse modalità.

L'obbiettivo di questo lvl è quindi garantire l'interoperabilità, ovvero la corretta interpretazione del contenuto scambiato, tenendo presente che l'informazione scambiata è **rappresentabile localmente** in modo diverso; ad esempio gli interi possono essere rappresentati in maniere diverse a seconda del sistema ed architettura.

In poche parole

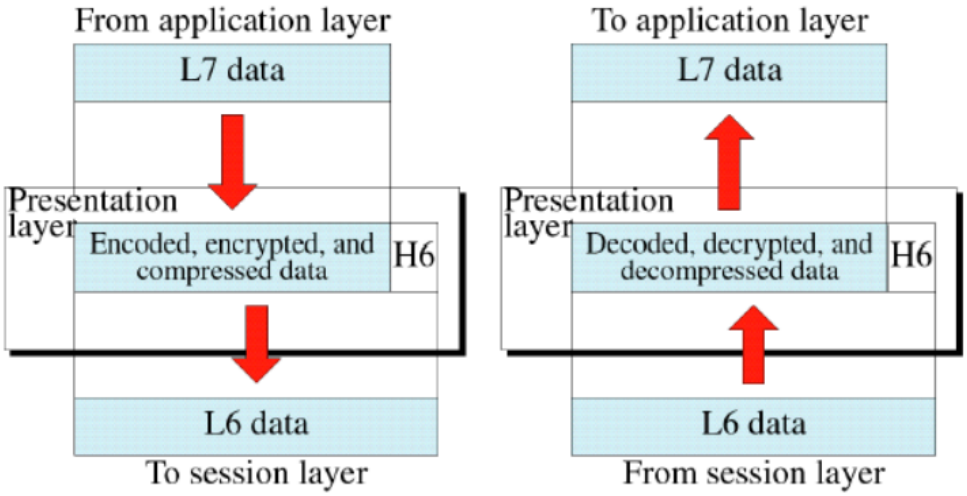
Permette scambio di dati **interoperabili** attraverso la definizione di un **formato comune** per rappresentare i dati. A differenza dei layers inferiori, si riferisce alla **sintassi e semantica** dei dati trasmessi.

Per interpretare correttamente i dati, essi vengono rappresentati in una **maniera astratta**; consideriamo tre sintassi:

- **Astratta**: definizione formale dei dati scambiati
- **Concreta locale:** rappresentazione locale dei dati
- **Trasferimento**: rappresentazione dei dati durante i trasferimenti

A questo strato vengono *addressati* anche i seguenti aspetti:

- **Encryption - cifratura**
- **Compression - compressione**



Application Layer

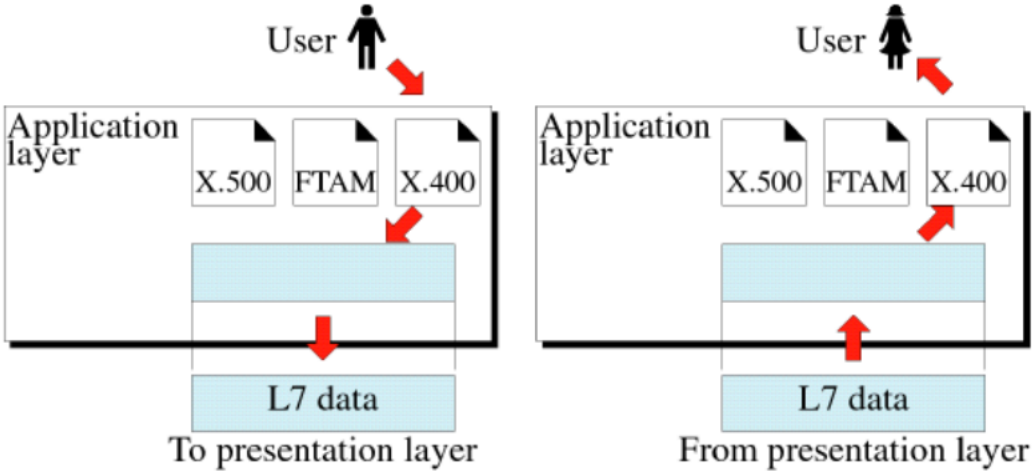
Questo livello è il livello in cui vengono definiti i protocolli applicativi, da non confondere con le applicazioni: questo livello non gestisce come un'applicazione legge i dati dallo Stdin, ecc. ma si interessa di **come è strutturato il messaggio applicativo**.

Nel caso del trasferimento di file, il protocollo applicativo si interessa di come il file debba essere trasferito, non di come il file debba essere letto da FS, ecc.

Fornisce protocolli che vengono **usati direttamente dalle applicazioni**. Non è corretto identificare un'applicazione come parte del layer applicazione.

I protocolli a questo strato implementano operazioni basilari come:

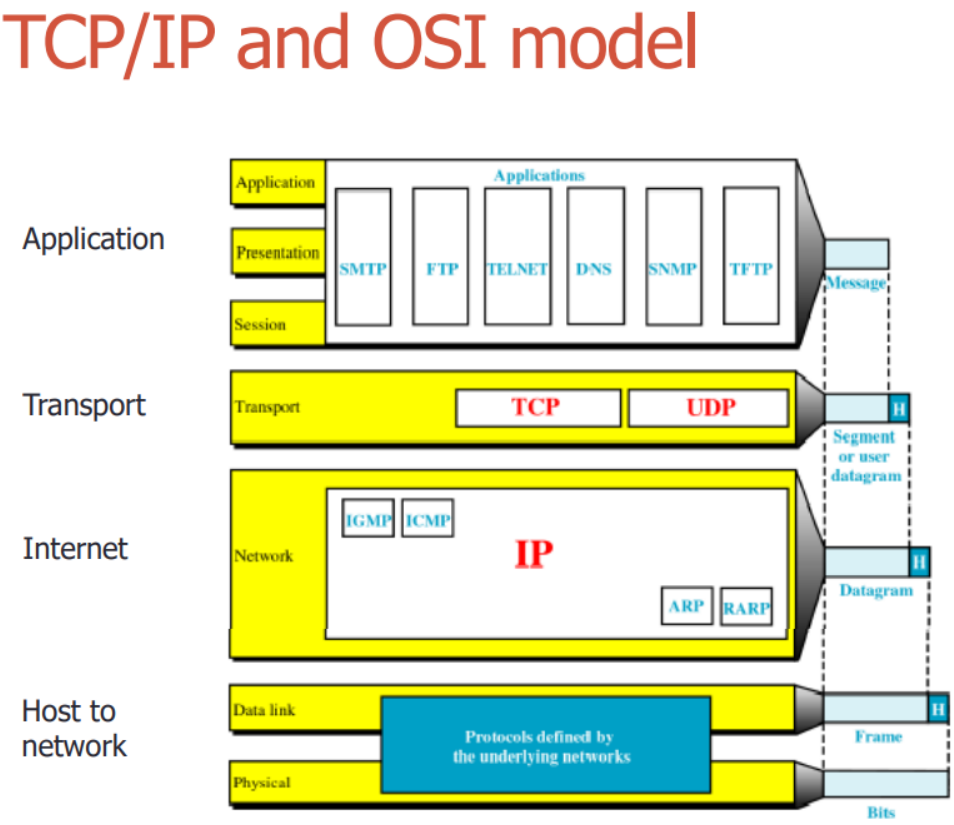
- **Trasferimento files**
- **E-mail**
- **Terminali virtuali**
- **Gestione dei processi remoti**



Architettura TCP/IP

Questa architettura caratterizza il funzionamento della **rete internet**, infatti viene anche chiamata **internet protocol suite**; serve appunto una suite protocolli che consente il funzionamento della rete.

E' un'architettura che nasce prima dal "basso", da un progetto americano ed universitario; siccome fu finanziato dal ministero della difesa aveva come requisiti importanti l'elevata flessibilità e l'elevata **fault tolerance**; doveva quindi resistere a guasti, prodotti anche da possibili attacchi.



Come si vede nell'immagine, i lvl che caratterizzano l'architettura TCP/IP sono **inferiori a 7**; I livelli specificati sono infatti (dal basso verso l'alto):

- **Internet**
- **Trasporto**
- **Applicazione**

Bonus:

- **Host to network:** E' un modello di adattamento; non vi è una specifica ben definita, è infatti un livello che dice in che modo il **livello internet** può sfruttare le tecnologie trasmissive messe a disposizione dal livello 2.

C'è una corrispondenza tra i modelli TCP/IP ed OSI: in particolare il livello **internet** (TCP/IP) ha una corrispondenza significativa con il livello **network** del modello OSI; per il livello internet compare il **protocollo principale IP**, accompagnato da diversi protocolli minori:

- IGMP
- ICMP
- ARP
- RARP

Il livello di trasporto ha una corrispondenza significativa con il livello di trasporto del modello OSI; anche in questo livello abbiamo due protocolli importanti: **TCP ed UDP**. Sono due protocolli di trasporto, e che consentono di realizzare la comunicazione processo-processo, e quindi il canale logico.
Hanno però un funzionamento diverso: offrono servizi diversi, in particolare il protocollo TCP è orientato ai **flussi** mentre il protocollo UDP è orientato ai **datagram**.
In altre parole, con un protocollo TCP possiamo inviare sequenze di byte **senza preoccuparci di delimitarle** a livello applicativo mentre con UDP i byte da inviare **devono essere delimitati** in unità chiamate **datagram**; ognuno di questi datagram è caratterizzato da informazioni di controllo.

Il protocollo TCP garantisce affidabilità, mentre UDP no.

Per quanto riguarda il livello **applicazione** del modello TCP/IP, vengono proposti 3 livelli del modello OSI: applicazione, presentazione, sessione. Non vi è quindi una specifica per i livelli OSI. Ad esempio: un protocollo del livello applicazione è **l'FTP**, che è un protocollo per il trasferimento dei files; se vogliamo che l'applicazione per il trasferimento dei file garantisca la gestione del dialogo (possibilità di tenere traccia dei file scambiati, ecc) deve essere definito nel momento in cui definiamo il protocollo.

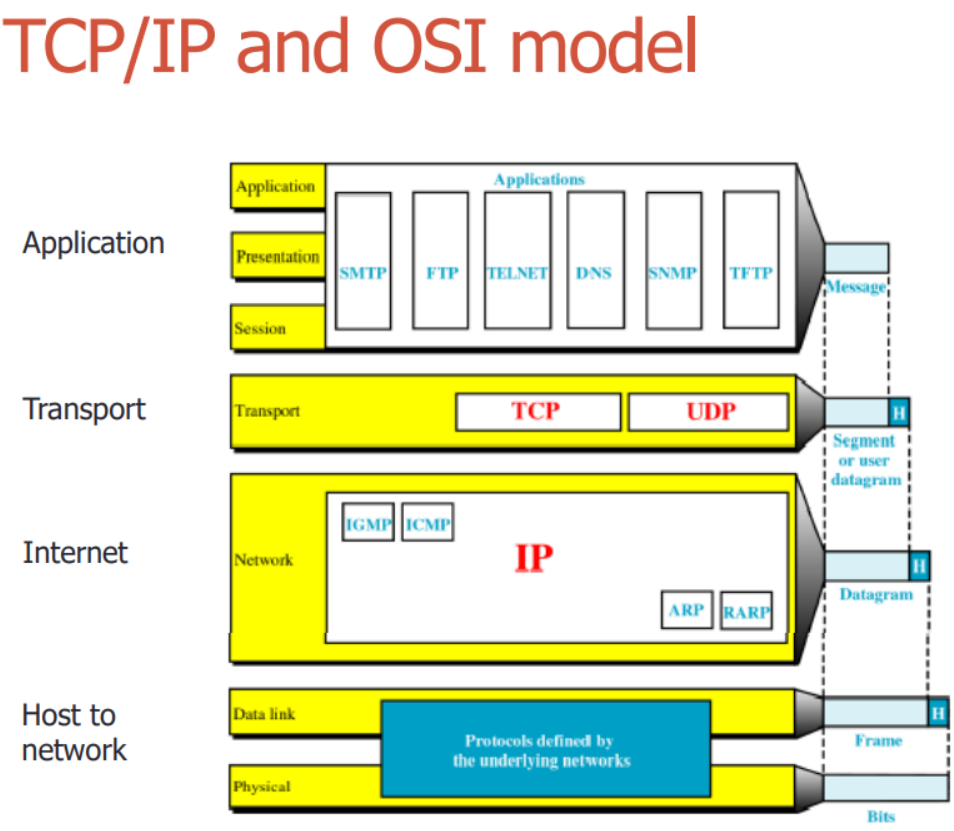
in poche parole

L'architettura è nota con i nomi **Internet Protocol Suite** e **Architettura TCP/IP**, dai due protocolli che la caratterizzano:

Non è un **modello** siccome essa include **protocolli reali** che sono specificati nel **RFCs - Request For Comments**.

Requisiti Stabili

Flessibilità estrema e **fault tolerance**, interconnessione di diverse reti. A sinistra (giallo) vediamo il modello OSI ed a destra il modello TCP/IP



Strato "host to network"

Questo strato non è specificato dall'architettura; vengnono utilizzati **software ed hardware** già esistenti. L'unica assunzione è **l'abilità di un host di inviare pacchhetti provenienti dallo strato network (network layer)**

Strato "internet"

E' lo **strato fondamentale dell'architettura**; le responsabilità di questo strato sono:

- Permettere ad un host di iniettare pacchetti nella rete
- Effettuare il routing dei pacchetti con un modello **best effort**
- Ogni pacchetto può seguire un **percorso diverso** per arrivare a destinazione.

Il servizio di comunicazione è chiamato **connectionless best-effort datagram**; viene inoltre definito un protocollo specifico: **IP (Internet Protocol)**.

Abbiamo però due problemi principali:

- **Addressing**
- **Routing**

Strato "trasporto"

Abilita la conversazione tra **due processi**; la comunicazione è eseguita tra grazie a **due protocolli:**

- **TCP (Transmission Control Protocol)**
    - E' un protocollo **orientato alla connessione ed affidabile** (tutti i pacchetti arrivano a destinazione preservando l'ordine di trasferimento)
    - Frammenta il flusso dallo strato superiore in diversi pacchetti (segmenti) che vengono passati al **network layer**.
    - Alla destinazione, i pacchetti vengono riassemblati.
- **UDP (User Datagram Protocol)**
    - E' un protocollo **inaffidabile e senza connessione**
        - I pacchetti potrebbero quindi arrivare in diverso ordine o addirittura non arrivare
    - Permette l'utilizzo all'**Application layer** un protocollo semplice ed efficiente **fortemente basato sull'IP**.

Strato "applicazione"

Nell'architettura TCP/IP gli strati **sessione e presentazione** sono mancanti. Questo strato è costruito direttamente al di sopra dello **strato di trasporto**, che implementa tutti i protocolli ad alto livello he vengono usati dalle applicazioni.

I primi protocolli erano:

- **Telnet**: terminale virtuale
- **FTP** - **File Transfer Protocol**: file transfer
- **SMTP - Simple Mail Transfer Protocol** e **POP - Post Office Protocol**: email

Altri protocolli sono:

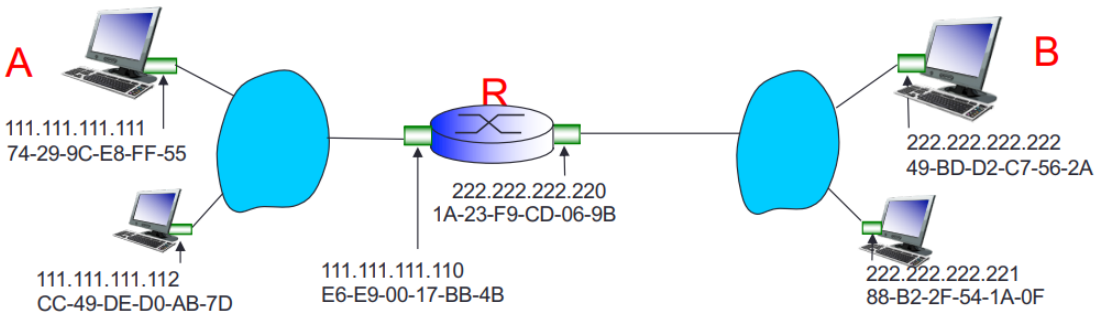
- **DNS - Domain Name Service**: mappare tra nomi host ed indirizzi IP
- **NNTP - Network News Transfer Protocol**: newsgroup
- **HTTP - HyperText Transfer Protocol**: World Wide Web

Strati ed incapsulazione: riassumendo

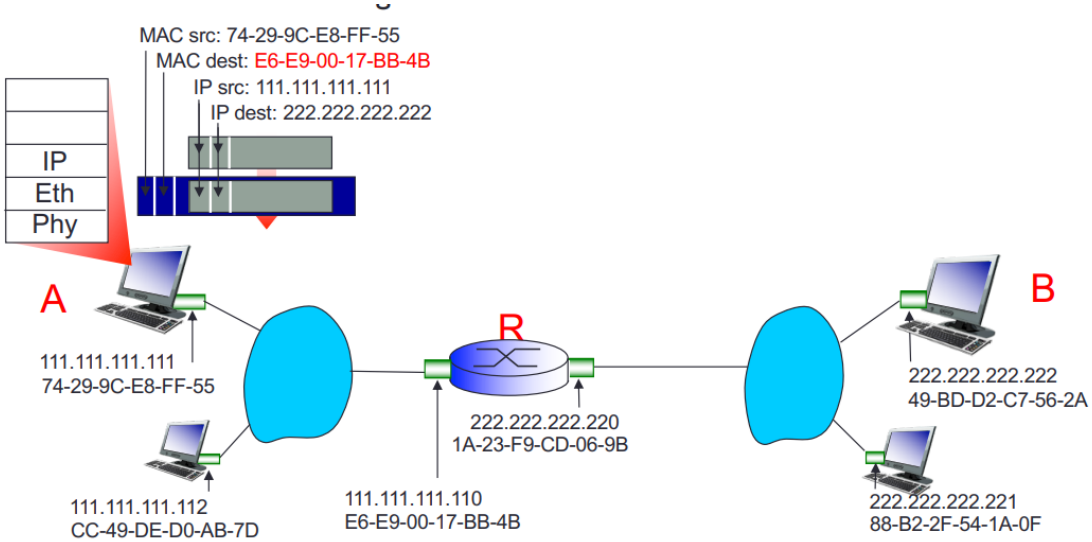
Comunicare tra due LANs

Percorso da A a B via R

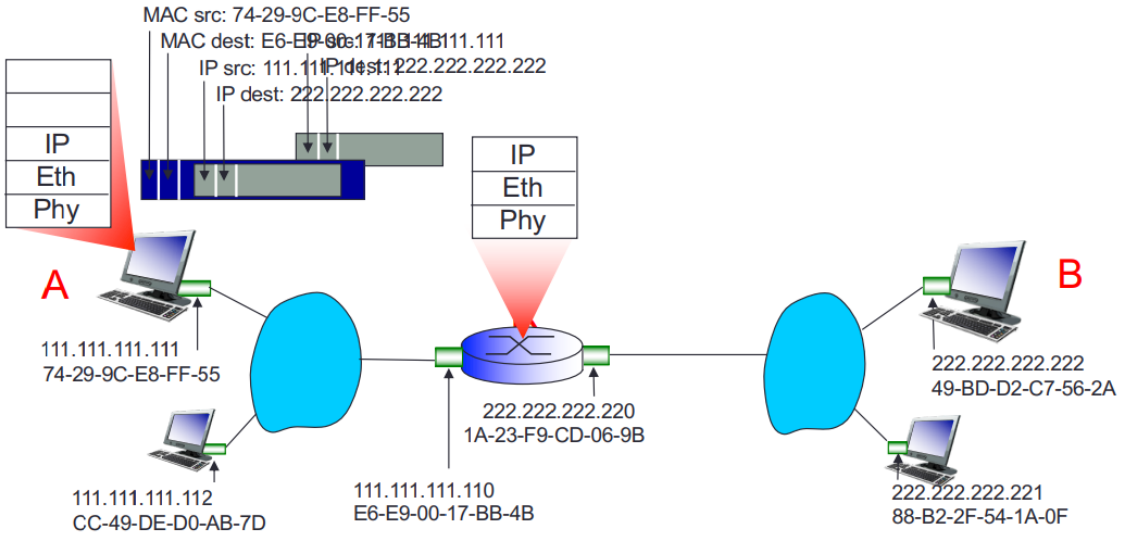
- Assumiamo che A conosca l'indirizzo di B
- Assumiamo che A conosca l'indirizzo del primo router R
- Assumiamo che A conosca l'indirizzo fisico di R



- A crea un datagram con l'indirizzo sorgente A, destinazione B.
- A crea un frame con l'indirizzo fisico di R come destinazione, il frame contiene il datagram da consegnare a B



- Il frame viene inviato da A ad R
- Il frame viene ricevuto da R; il datagram è estratto e passato all'IP



- R inolta il datagram con l'indirizzo sorgente A e la destinazione B
- R crea un frame con l'indirizzo fisico di B come destinazione; il frame contiene il datagram da consegnare a B