



Università degli Studi di Bologna  
Scuola di Ingegneria

# Corso di Reti di Calcolatori T

***OSI - Open System Interconnection  
Sistema Standard a livelli***

**Antonio Corradi**

Anno accademico 2023/2024

# **OPEN SYSTEM INTERCONNECTION (OSI)**

---

**OSI** è uno **standard di comunicazione tra sistemi aperti**, che permettere **a sistemi qualunque** di **INTEROPERARE**, ossia di consentire che **sistemi eterogenei possano comunicare ed operare tra loro in modo aperto**

Importante obiettivo laterale per i provider di comunicazione è **la gestione dei sistemi da remoto**

**OSI** è uno **standard per la Gestione dei sistemi remoti**, area detta **Systems management** o **Network management**

**Controllare, coordinare, monitorare** sistemi interconnessi eterogenei per consentirne una gestione efficiente e a distanza **senza porre limiti di località e di coresidenza**

# **OSI COME STANDARD GENERALE**

---

**OSI** nasce con obiettivi di **razionalizzazione** per qualunque tipo di **comunicazione** tra **sistemi diversi**

Architetture di rete diverse proprietarie e protocolli diversi **non erano in grado di interagire in modo sistematico**

**OSI** propone **standard e schemi di progetto astratti, per razionalizzare, inquadrare e abilitare ogni possibile standard di comunicazione** (e guidare le soluzioni)

**OSI è:**

- organizzato a **livelli, con precisi obiettivi e significati**
- **interamente descritto ad oggetti**
- **astratto e senza legami con realizzazioni** (proprietarie o meno)
- uno **scenario ampio di riferimento** per le **soluzioni**

# **OPEN SYSTEM INTERCONNECTION**

---

**OSI come standard di comitati**

**Organizzazioni pubbliche come proponenti e responsabili (comitati internazionali)**

**ISO:** International Organization for Standardization

**IEC:** International Electrotechnical Commission

**CCITT:** International Telegraph and Telephone  
(Consultative Committee)

**Organizzazioni Industriali (per la standardizzazione)**

**ECMA:** European Computer Manufacturers'Association

**IEEE:** Institute of Electrical and Electronics Engineers

**EIA:** Electronic Industries Association

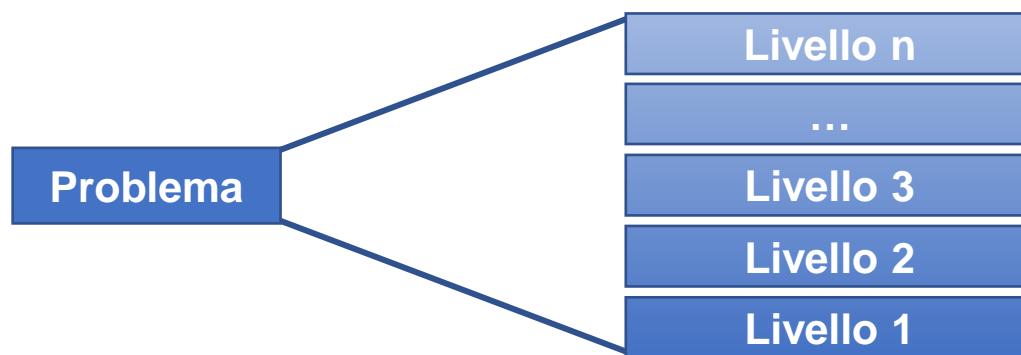
# OSI – ARCHITETTURA

---

**OSI** propone una **precisa architettura di soluzione** per arrivare a **descrivere una comunicazione ICT complessa**

La architettura si basa **sul principio della astrazione** che richiede di **nascondere dettagli e mostrare solo le entità utili** all'utente finale (cliente del livello superiore)

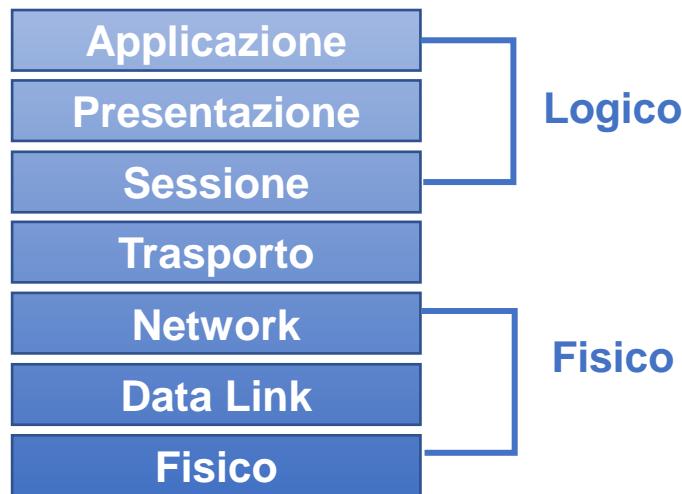
È possibile **risolvere problemi complessi** separando gli ambiti in modo **ordinato e ben identificato** attraverso **una serie di astrazioni (detti livelli)** in modo da **decomporre il problema e affrontare la complessità separatamente**



# OSI – ARCHITETTURA

---

OSI prevede **sette livelli**, uno sopra l'altro: **Fisico, Data Link** (o collegamento), **Network** (o Rete), **Trasporto, Sessione, Presentazione, Applicazione**



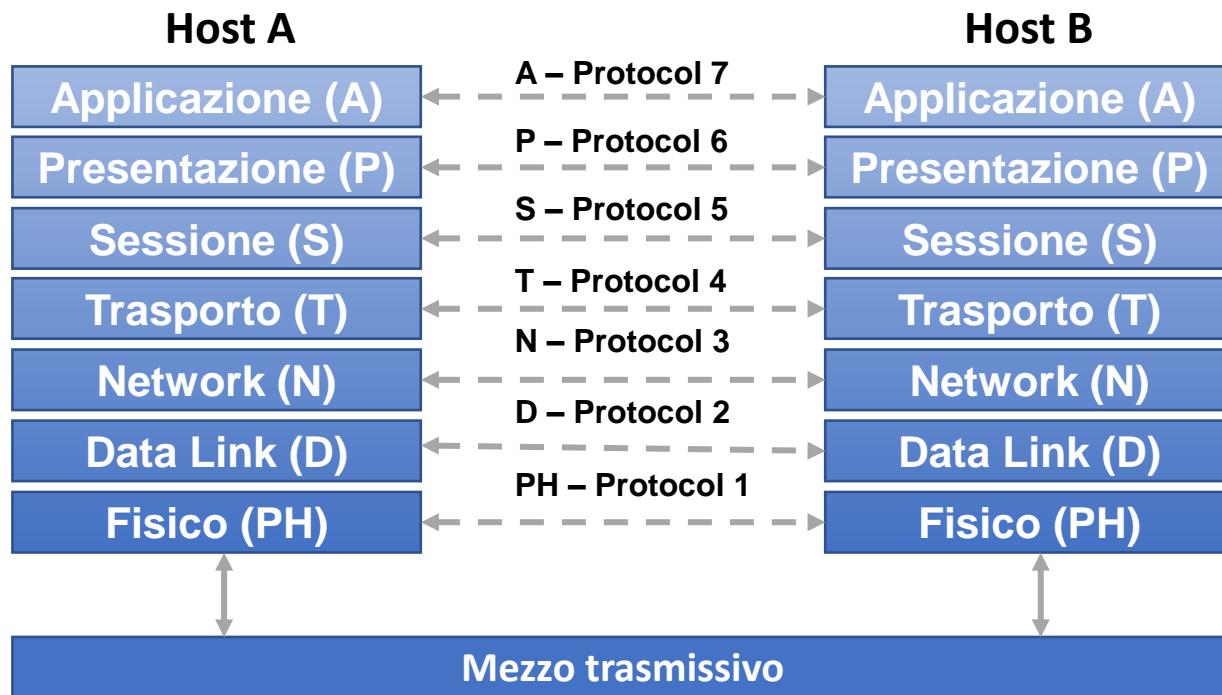
I primi 3 livelli rientrano nella parte **fisica** dell'architettura

Gli ultimi 3 livelli rientrano nella parte **logica** dell'architettura

Il **trasporto** separa le due parti

# OSI – ARCHITETTURA

Ogni livello ha l'obiettivo di **comunicare con il pari**, e lo realizza tramite un **protocollo**, ossia un insieme di passi, che realizza **usando il servizio sottostante**



# OSI – METAFORA...

Pensando a una comunicazione tra due persone che devono comunicare e che sono esperte di un dominio ma hanno modi diversi (docenti di reti), uno italiano e uno brasiliiano che stanno facendo una ideale conference call

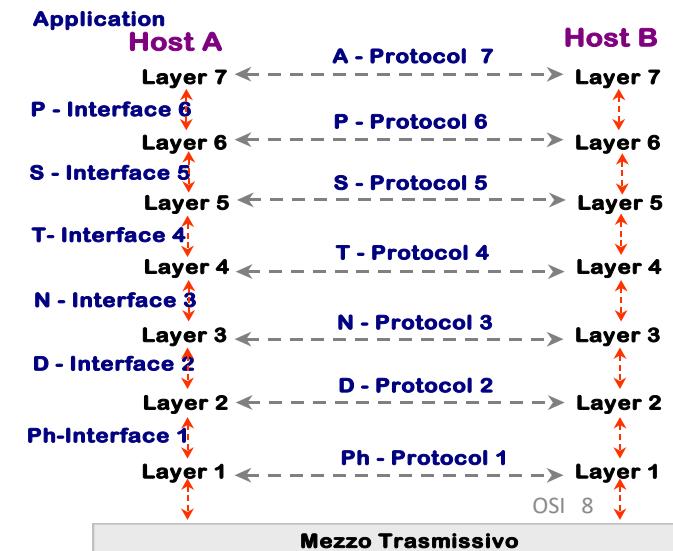
**Applicazione** abilita la loro **comunicazione finale**

**Presentazione** risolve la differenza di lingua (ma anche ICT)

**Sessione** risolve i dialoghi in atto, tenendo conto di quanto si stanno dicendo in diversi modi (video, audio,...)

**Trasporto** prevede connessioni molteplici e diverse ICT

I livelli fisici, li conoscete già  
Network, Data Link, Fisico



# SISTEMI A LIVELLI

---

L'**obiettivo** di un **sistema a livelli** è quello di fornire una migliore **astrazione**, semplificando il punto di vista

**Ogni livello inferiore nasconde al livello superiore dei dettagli non necessari**

**Ogni livello superiore** usa gli strumenti **esposti** da quello inferiore per il suo obiettivo e così via

In generale, **i livelli OSI sono prescrittivi** e non è possibile non considerarli (o **bypassarli**), ma **si devono attraversare in modo ordinato**, solo secondo la gerarchia introdotta

I livelli sono stati introdotti per **ridurre la complessità** tra livelli diversi, ossia la **divisione delle responsabilità e la semplificazione dei compiti**

**Maggiore è la complessità, più opportuni sono i livelli**

# OSI A LIVELLI

---

Ogni sistema a livelli si basa sul principio della **separazione** dei compiti (**delega**) e della **trasparenza** della realizzazione (**astrazione**)

**Partendo dalla applicazione**

**Ogni livello** ha l'obiettivo di **comunicare con il pari**, e lo realizza tramite un **protocollo**, ossia un insieme di passi, che realizza **usando il servizio sottostante**.

**Implementazione**      ⇒      **specifica del protocollo**  
(protocollo realizzato dal livello e non visibile al disopra)

Ogni livello **fornisce un servizio specificato e disponibile al livello superiore**

**Definizione del servizio** ⇒      **semantica del servizio**  
(interfaccia del servizio usata dal livello sovrastante)

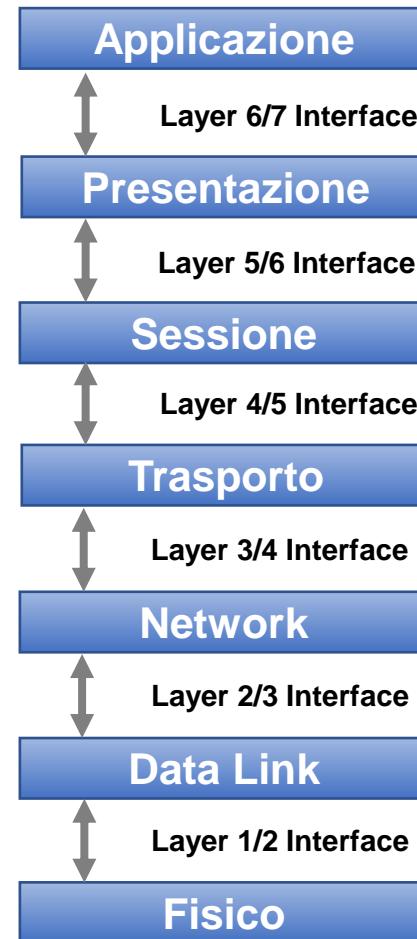
**OSI è uno standard ampio, adottato dalle specifiche internazionali e obbligatorio negli enti di comunicazione relativi**

# I LIVELLI OSI

---

Ogni livello prevede

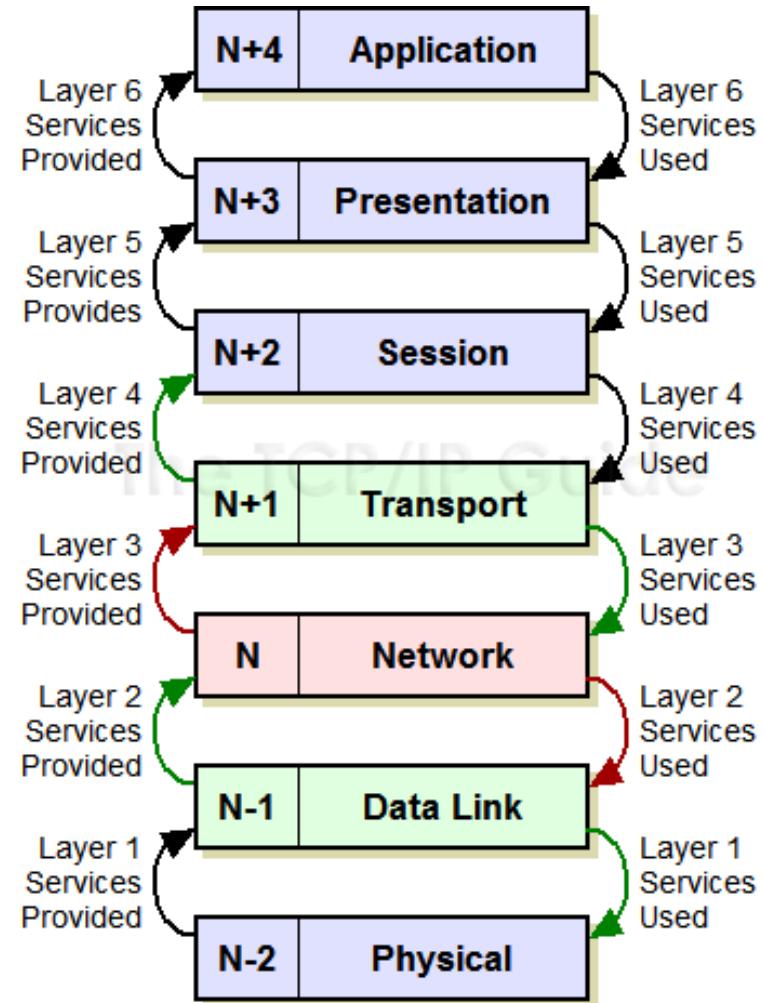
- un **protocollo** da realizzare in modo **definendo lo standard di protocollo di quel livello**
- un **servizio** da richiedere in modo **verticale tramite la interfaccia da parte del livello superiore**



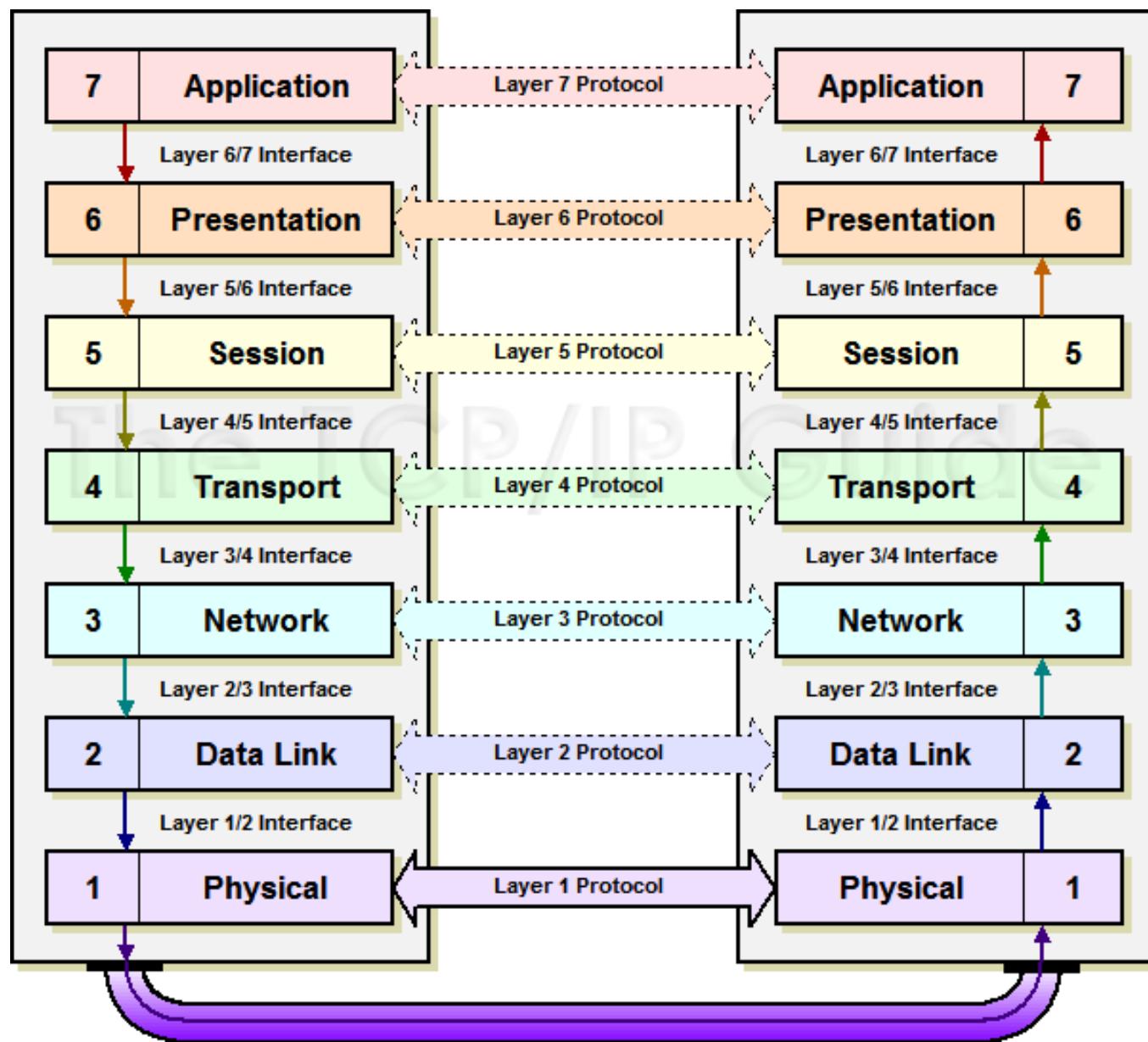
# I LIVELLI OSI

Ogni livello prevede

- un **protocollo** da realizzare in modo **definendo lo standard di protocollo di quel livello**
- un **servizio** da richiedere in modo **verticale tramite la interfaccia da parte del livello superiore**



# I LIVELLI OSI

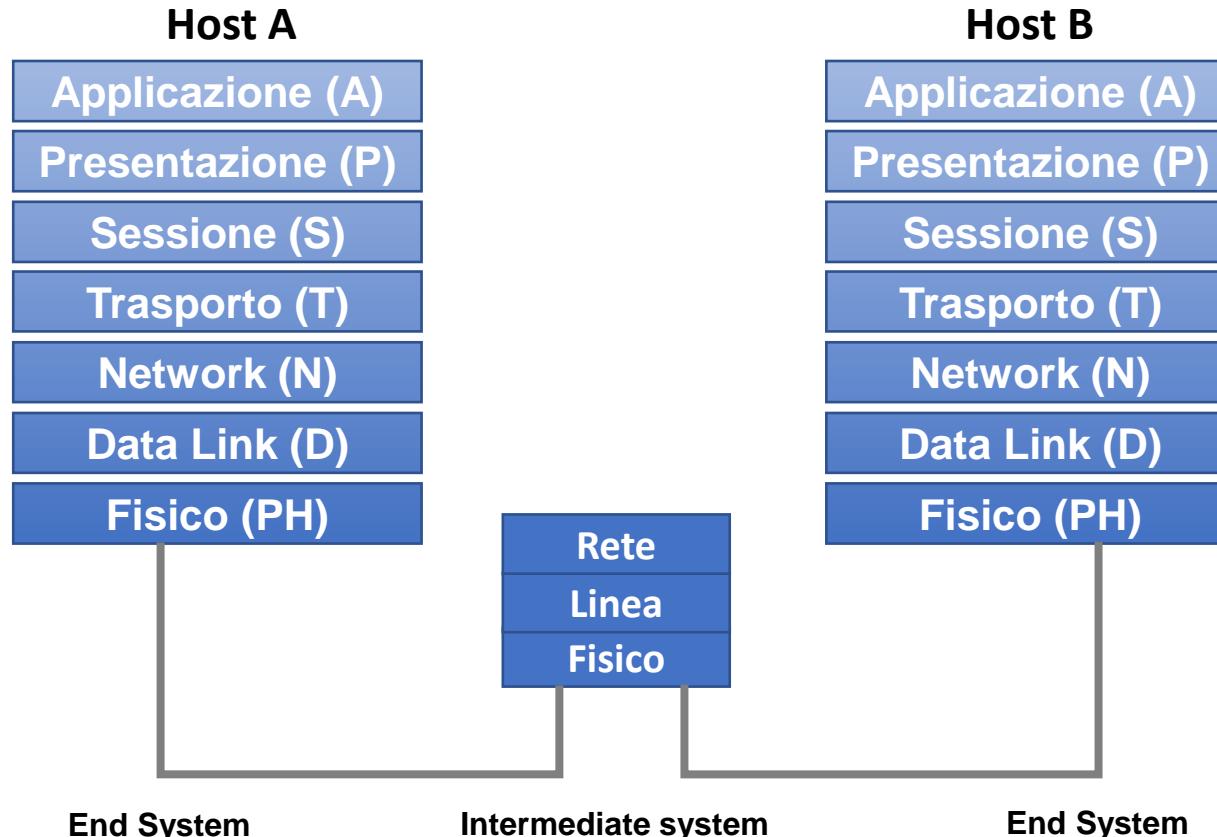


# OSI (VISIONE VERTICALE E ORIZZONTALE)

## Definizione del servizio (**descrizione verticale**)

**definizione astratta dei servizi del livello corrente** disponibile al livello immediatamente superiore

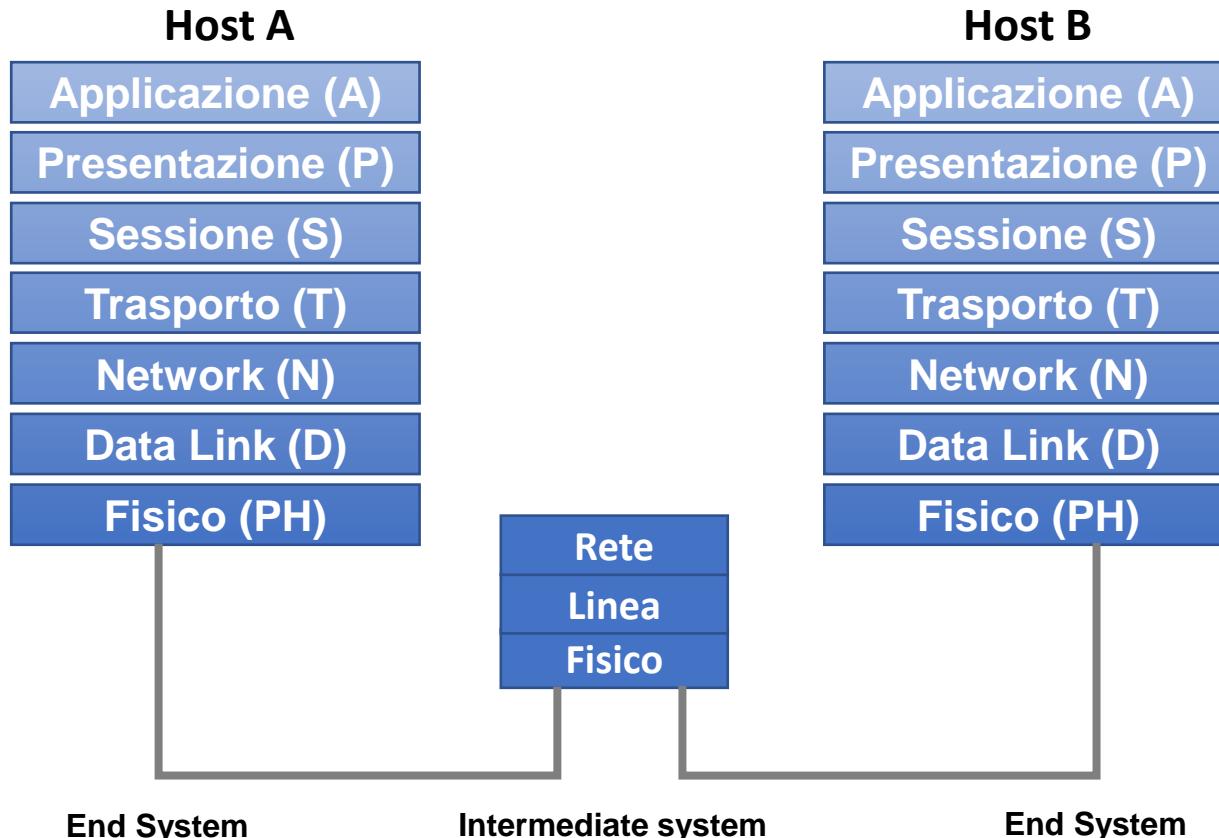
È detta anche **interfaccia** offerta al livello superiore



# OSI (VISIONE VERTICALE E ORIZZONTALE)

## Specifica del protocollo (descrizione orizzontale)

specificata dettagliata di come il livello implementa il servizio tramite **scambio di dati ed informazioni** tra le due realizzazioni dei sistemi comunicanti, ossia come si **realizza il servizio**



# OSI – AZIONI DI COMUNICAZIONE

---

In genere, per una azione di comunicazione, possiamo avere:

## Mittente

Entità che ha la responsabilità di iniziare la comunicazione

## Ricevente

Entità che accetta la comunicazione e poi la sostiene

## Intermediari

Eventuali intermedi che devono partecipare alla comunicazione intesa come risorse per sostenerla

**Il mittente manda dei dati ad un ricevente che può anche rispondere all'invio con un'azione applicativa conseguente**

Ogni azione comporta una comunicazione che passa attraverso i livelli da applicativo a fisico, del mittente e ricevente e *almeno fino al livello di rete per gli intermediari*

# OSI – AZIONI DI COMUNICAZIONE

---

In genere, per una azione di invio informazioni, possiamo avere:

## Mittente

Entità che ha la responsabilità di iniziare la comunicazione

## Ricevente

Entità che accetta la comunicazione e poi la sostiene

## Intermediari

Eventuali intermedi che devono partecipare alla comunicazione intesa come risorse per sostenerla

**Il mittente manda dei dati ad un ricevente che può anche rispondere all'invio con un'azione applicativa conseguente**

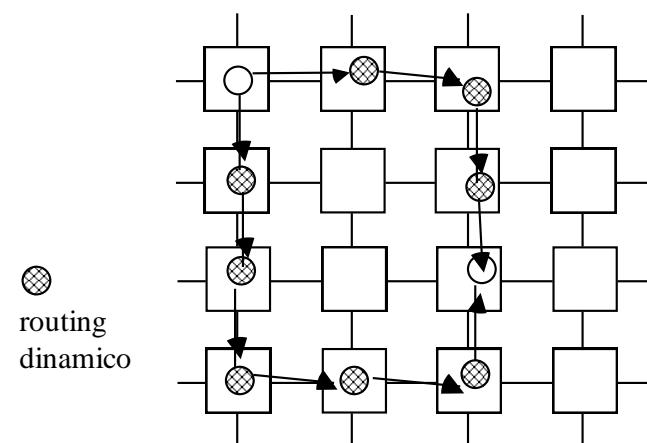
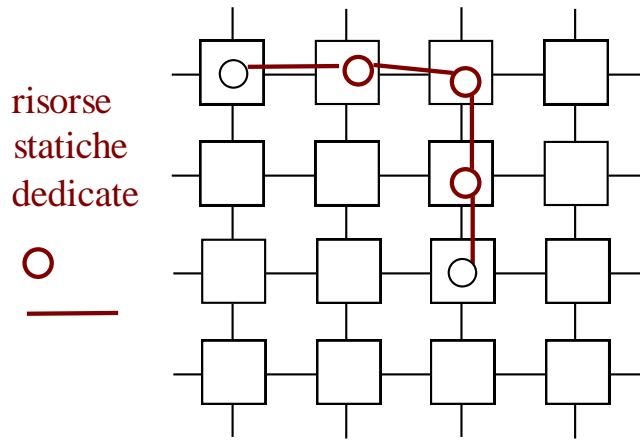
Ogni azione comporta una comunicazione che passa attraverso i livelli da applicativo a fisico, del mittente e ricevente e *almeno fino al livello di rete per gli intermediari*

# CONNESSIONI e QoS

Modelli a connessione con QoS

- **connessione (OSI)**  
Tutti i messaggi seguono la **stessa strada (route)** per la coppia mittente destinatario **decise staticamente** e **impegnano risorse nei nodi interessati e nei nodi intermedi predeterminati**
- **connessioni solo sugli endpoint (TCP e supporto di IP per routing dinamico datagrammi)**

Nessuna QoS: i messaggi IP possono seguire strada diverse **decise dinamicamente e non impegnano risorse intermedie**



**TCP, basato su IP**, non impegnando risorse intermedie ma solo sul mittente / destinatario, non può garantire QoS

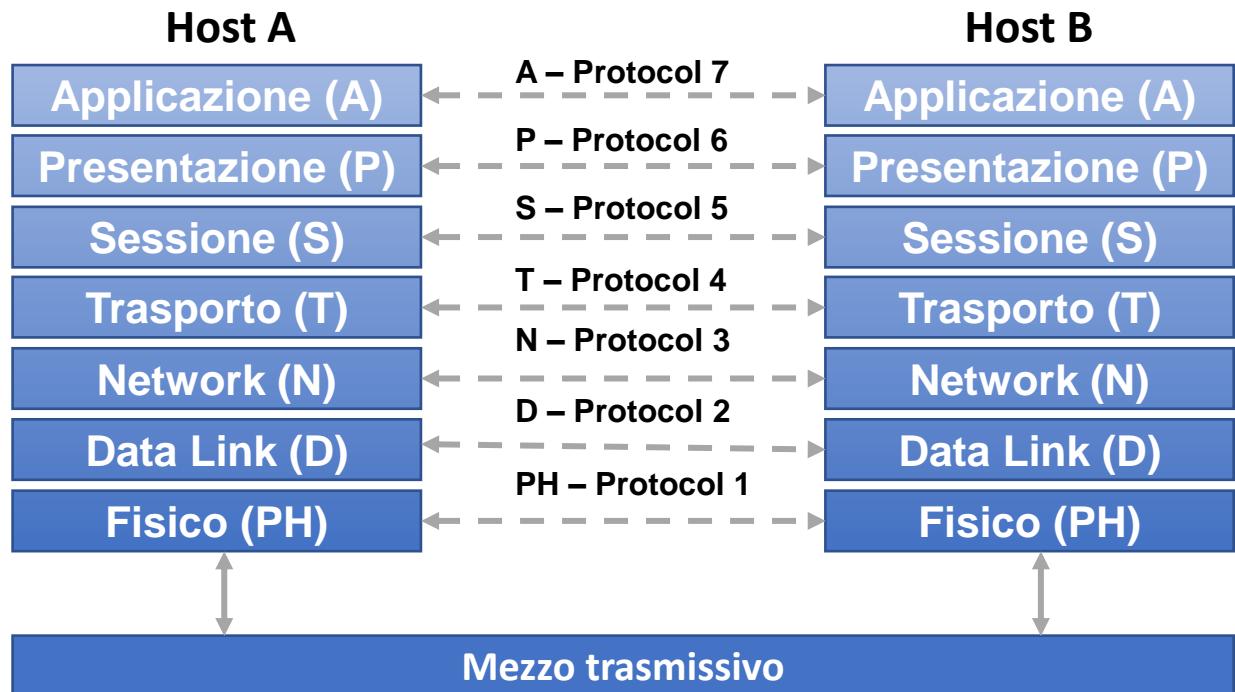
# NOMI ED ENTITÀ IN OSI

Consideriamo una applicazione che vuole comunicare con un'altra (su Host A e Host B)

La **applicazione A** deve indicare quali sono le entità su B, ossia tutti i livelli che deve interessare su B, dal fisico in su, fino al livello di **applicazione B**

Quindi deve conoscere i nomi per tutte le entità del pari

Allo stesso modo,  
**applicazione B**  
deve specificare le  
entità che le  
interessano su A:  
**tutti i nomi per**  
**tutte le entità che**  
**la interessano**



# NOMI ED ENTITÀ IN OSI

---

La **applicazione A** deve identificare il suo corrispondente (**orizzontale**): per trovarlo in un **sistema a livelli**, deve indicare tutti i **livelli** che possono portarlo al suo corrispondente e, se si possono **scegliere più entità per livello, poterle identificare precisamente**

Per arrivare alla **applicazione B** **deve indicare le entità specifiche** per **tutti i livelli** che portano al corrispondente livello pari

Quindi deve potere indicare **per ogni livello le entità** di suo interesse (tra quelle presenti con strategie diverse)

**Applicazione B**      (livello 7)

**Presentazione B**      (livello 6)

**Sessione B**      (livello 5)

**Trasporto B**      (livello 4)

**Rete B**      (livello 3)

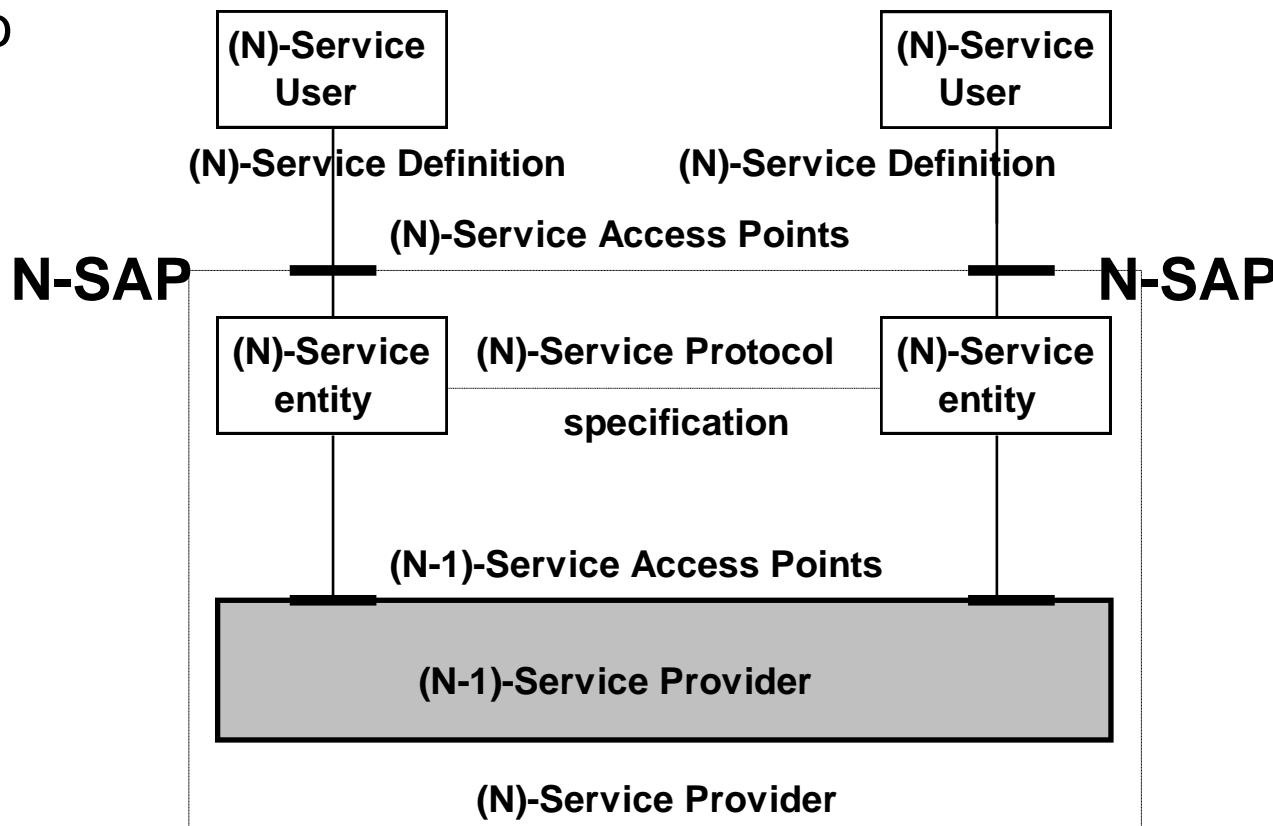
**Data link B**      (livello 2)

**Fisico B**      (livello 1)

# OSI – SAP (SERVICE Access Point)

**SERVICE ACCESS POINT** o **(N)-SAP** identifica il **punto di accesso** che un servizio OSI offre al suo livello superiore.

- **Interfaccia logica** tra una (N-1)-entity ed una (N)-entity
- **API** le funzionalità disponibili ad un SAP
- **(N)-SAP address** un **nome unico** che identifica un singolo punto di accesso



# OSI – NOMI E SAP

---

## NOMI UNICI

Ogni SAP deve avere un **nome unico** per essere identificato

Possono esserci anche più SAP per uno stesso livello

## NOMI delle ENTITÀ

per identificare una **entità remota** si devono nominare **tutti i SAP** di ogni livello fino al livello 1 (il cliente deve nominare tutti i SAP di tutti e 7 i livelli)

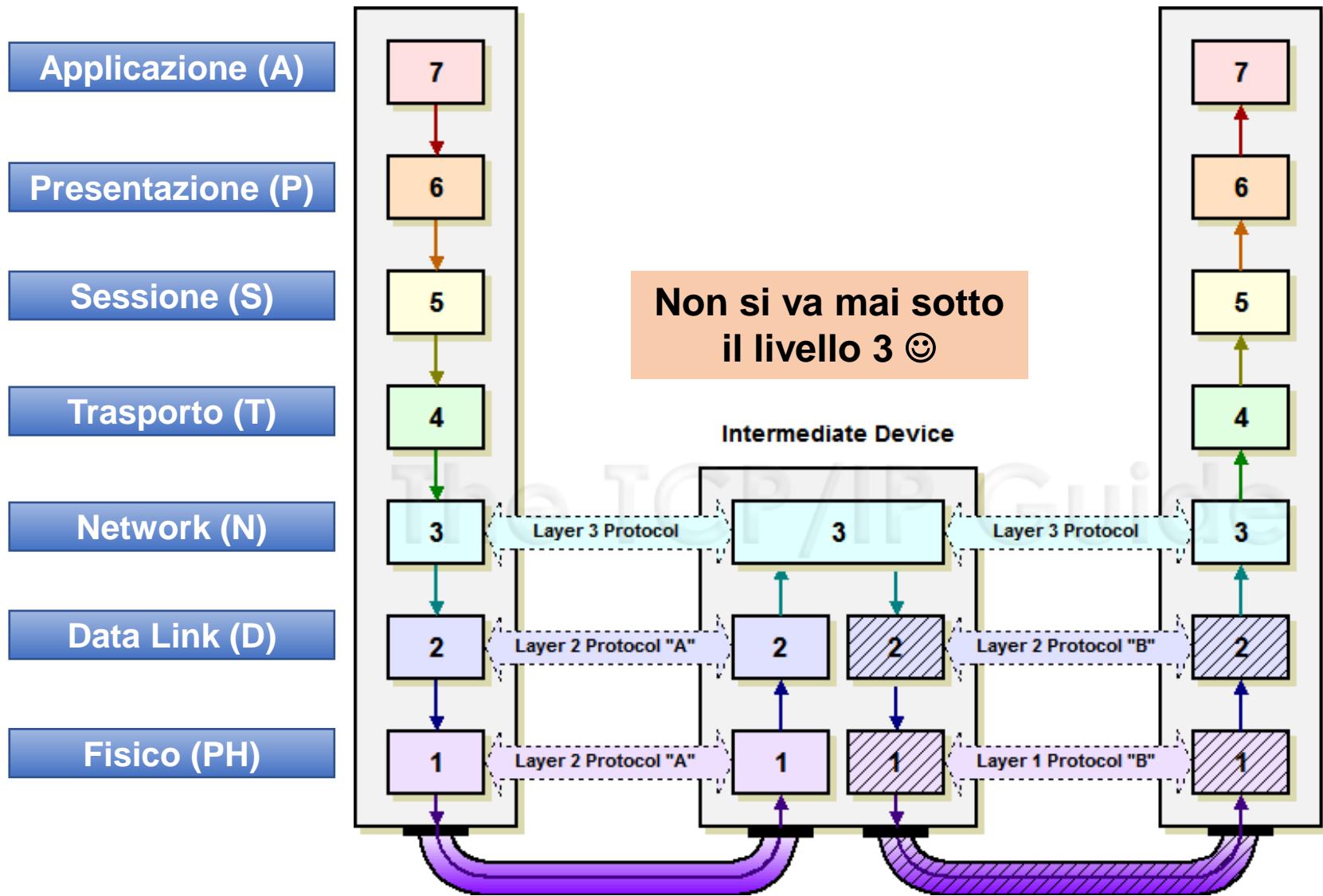
**Scorciatoia: si parte con i nomi di rete che identificano il nodo in modo unico (e da lì in su)**

## Ci si ferma al livello 3 😊

Una entità (un pari) viene identificata con **SAP di rete** e con la **sequenza a stack** dei **nomi** delle **entità SAP sopra** per identificare i livelli superiori (ad esempio: IP, porte, socket, processi, ...)

**In questo modo abbiamo la possibilità di specificare in modo non ambiguo (e abbreviato) ogni entità**

# OSI – MAI SOTTO IL LIVELLO 3



# OSI – PROTOCOLLO STANDARD

---

OSI definisce le **sole specifiche di comunicazione** senza dettare **nessuna specifica a livello locale** né suggerire **alcuna tecnologia di soluzione**

Ad esempio, **non si dice mai come i livelli di protocollo devono essere realizzati**, a processi, procedure, ad oggetti, ecc.

**Anzi, si evita qualunque ricopertura con termini che suggeriscano una soluzione specifica**

La **organizzazione a SAP è astratta** e potrebbe essere applicata alla modellazione anche di molti altri sistemi

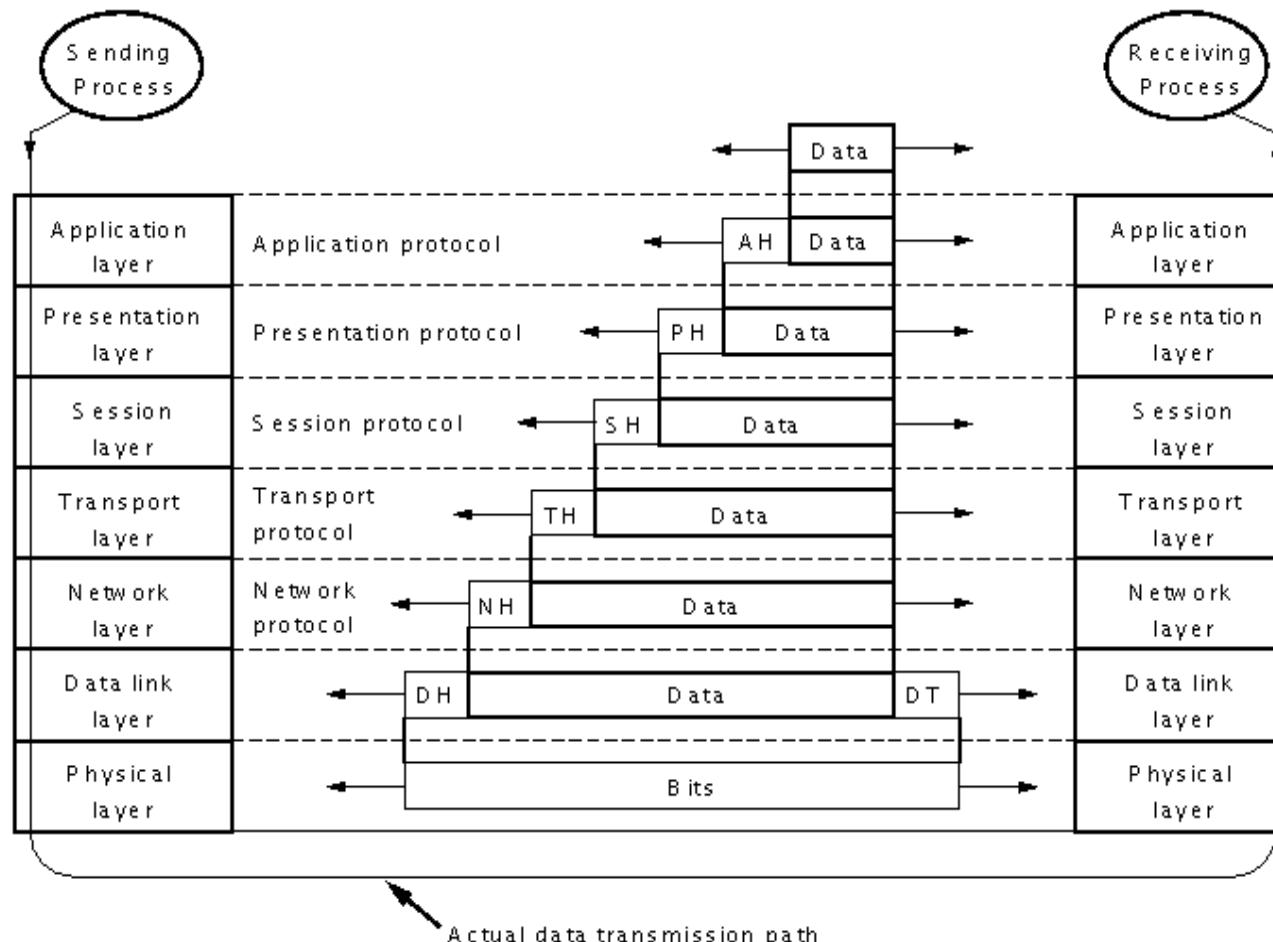
Per ogni livello, sono possibili e riconosciute:

- implementazioni a **procedure**
- implementazioni a **processi**
- implementazioni ancora più **parallele**

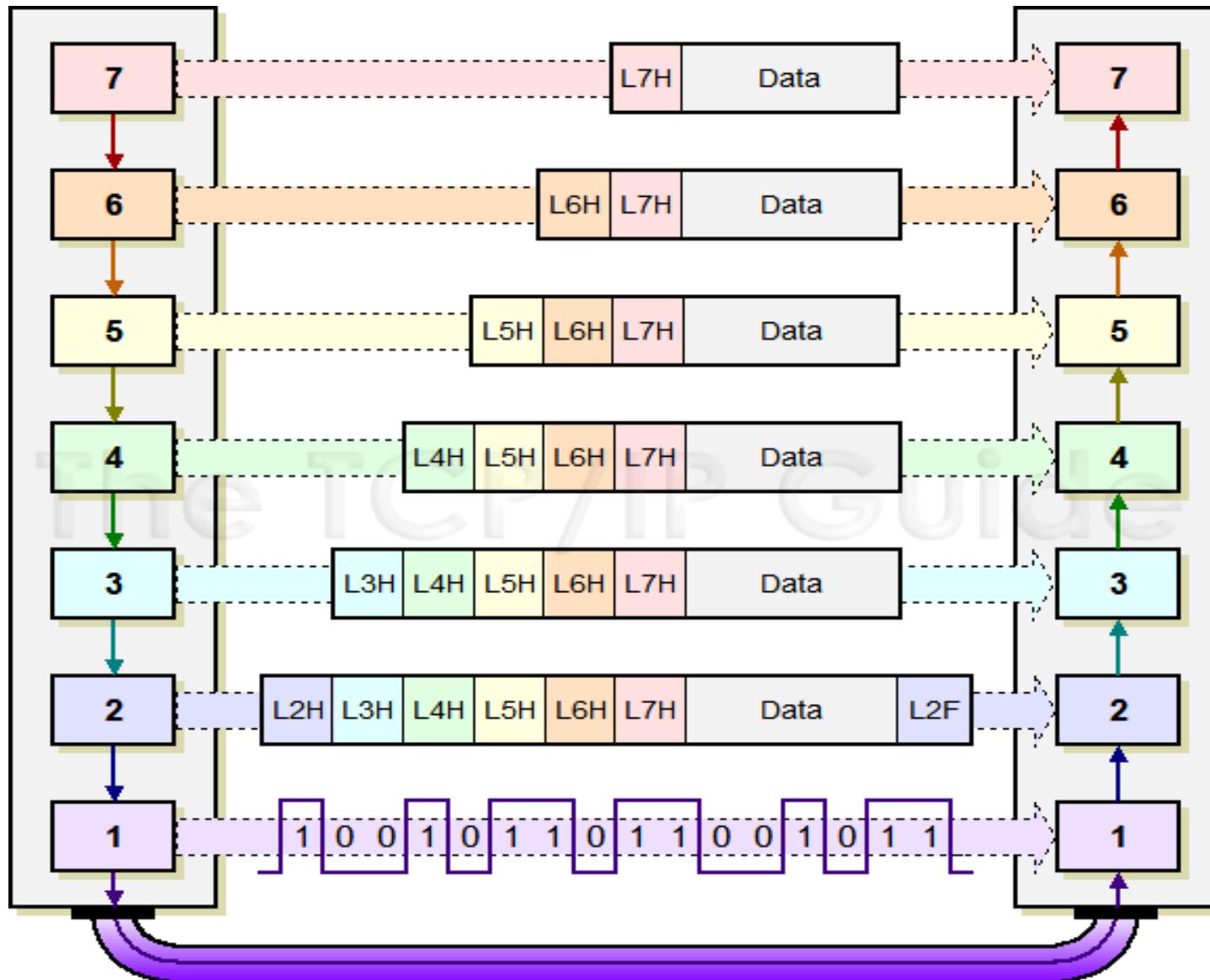
Non si parla mai di processi, procedure, ma si usano termini neutri come **attività, flusso di informazioni**

# OSI – COMUNICAZIONE

Su **iniziativa del mittente**, ogni livello **introduce le proprie esigenze specifiche a quelle del dato** e le passa al SAP sottostante (**fino al livello fisico, dopo il quale si comincia a risalire fino all'Application corrispondente**)



# OSI – ACCUMULO HEADER



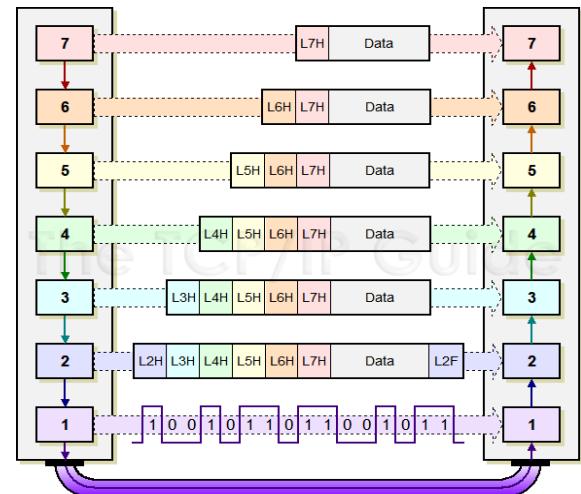
# OSI – MECCANISMI

OSI ha affrontato la sfida di **definire un ‘prodotto’ a lungo ciclo di vita** (idealmente **infinito**)

Mentre i protocolli reali e le architetture reali cambiano in modo anche veloce, **è necessario definire standard che invece rimangano a lungo**

Per garantire una copertura stabile della comunicazione, **OSI specifica solo meccanismi di comunicazione** e lascia le **politiche di uso non standardizzate** alla decisione locale.

Ogni livello viene descritto in termini di **puro meccanismo**, ossia di **standard di messaggi scambiati** e non di come le **singole azioni sono integrate a livello locale**  
(che può cambiare in modo libero)



# OSI – FORMATI

---

Si standardizzano i **formati dei messaggi scambiati tra i diversi livelli**, considerando sia i messaggi scambiati per chiedere un **servizio** ad un SAP, sia i messaggi scambiati per realizzare un **protocollo**

- **SDU (SERVICE DATA UNIT o INTERFACE DU, IDU)**
- **PDU (PROTOCOL DATA UNIT)**

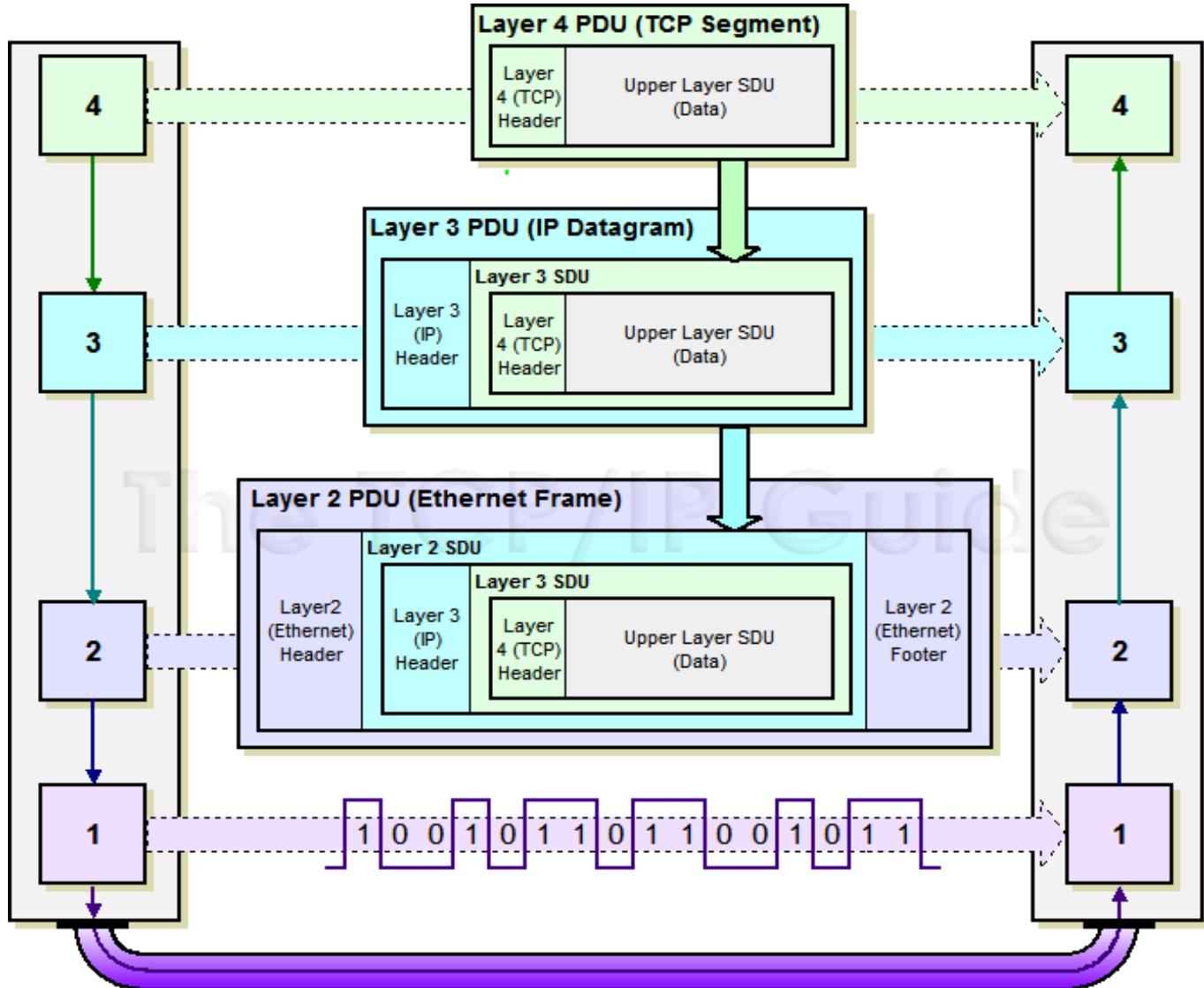
che descrivono i messaggi che permettono di **chiedere il servizio** e di **specificare il protocollo**

Tipi di data unit che sono standardizzati:

- **SDU della sessione**, per richiedere i servizi
- **IDU come richieste** portate alla interfaccia
- **PDU del protocollo**, con cui si realizza operazioni in orizzontale

# OSI – COMUNICAZIONE SUI LIVELLI

**PDU (Protocol Data Unit)**  
fornito dal protocollo  
diventa  
**SDU (Service Data Unit o Interface Data Unit)**  
per il livello sottostante



# OSI – CONNESSIONE

---

**CONNECTIONLESS:** Ogni unità di dati è trasferita in modo indipendente dalle altre unità essendo autocontenuta (senza ordine)

Nessuna **qualità del servizio** e nessuna negoziazione

Lo scambio di informazioni tra i due pari avviene **senza storia e senza nessun concetto di negoziazione**

**CONNECTION-ORIENTED:** Si stabilisce una **connessione** tra entità pari che devono comunicare, con caratteristiche della connessione negoziate durante la fase iniziale: si supportano **messaggi multipli**

In modalità connection-oriented la comunicazione tra due utenti di pari livello avviene in tre fasi:

1. **apertura della connessione**
2. **trasferimento di dati sulla connessione**
3. **terminazione della connessione**

Il servizio **connection-oriented** di un livello deve fornire le opportune funzionalità per le tre fasi con la richiesta **qualità del servizio**

# OSI – MODALITÀ

---

## CONNECTIONLESS

adatta per **dati occasionali** e **senza qualità** della comunicazione: un messaggio mandato dopo può arrivare prima di un precedente

**Costo limitato ma scarse garanzie**

## CONNECTION-ORIENTED

Il pari con iniziativa deve prima provvedere alla connessione che spesso porta anche a stabilire **quali sono gli intermediari per la connessione stessa (OSI)**

**Su una connessione costi più elevati, ma maggiori garanzie**

la comunicazione può avere luogo in modo bidirezionale sulla connessione e si ottiene qualità, come l'ordinamento dei messaggi, il coordinamento delle risorse, ...

La connessione non significa **necessariamente** impegno di risorse su eventuali nodi intermedi necessariamente (vedi **Internet**)

# OSI – QUALITÀ

---

OSI considera il servizio come caratterizzato da attributi che ne costituiscono la **Qualità di Servizio (QoS)** e permettono una scelta

Ogni servizio deve fare i conti con la qualità logicamente richiesta e le possibilità reali di risorse richieste e disponibili

## Proprietà ed esempi di Servizi

**affidabilità** della comunicazione e **sequenza** dei flussi di dati

<b>connessione</b>	<b>affidabile</b>	<b>non affidabile</b>
--------------------	-------------------	-----------------------

<b>non connessione</b>	<b>affidabile</b>	<b>non affidabile</b>
------------------------	-------------------	-----------------------

**Esempi:**

*datagramma* senza connessione *non affidabile*

*connessioni affidabili* sequenze di messaggi

Ma anche richieste diverse: solo **garanzia di sequenza** di dati che si possono anche perdere, anche dati che **richiediamo che siano ricevuti**, sequenze di byte che richiediamo **ricevuti in modo unico**,

...

# OSI – PRIMITIVE E FORME

---

Le entità pari cooperano tramite primitive per implementare le funzionalità del livello cui appartengono

Primitive base tipiche sono: **data** per trasmettere contenuto, e **connect, disconnect** per aprire chiudere la connessione

Quattro possibili forme per una primitiva

- **Request** ⇒ il service user richiede un servizio (una azione)
- **Indication** ⇒ il service provider indica al service user che è stato richiesto un servizio (segnalazione di evento)
- **Response** ⇒ il service user specifica la risposta alla richiesta di servizio (una azione)
- **Confirm** ⇒ il service provider segnala la risposta alla richiesta di servizio (segnalazione di evento)

**POSSONO ESSERE PRESENTI ANCHE TUTTE LE FORME**  
(nel caso detto sincrono)

# OSI – PRIMITIVE E FORME

Nel dialogo tra pari si possono utilizzare le forme  
**S-CONNECT.request**

Sintassi: **Nome primitiva. (Punto) Tipo primitiva**

**Primitiva asincrona**

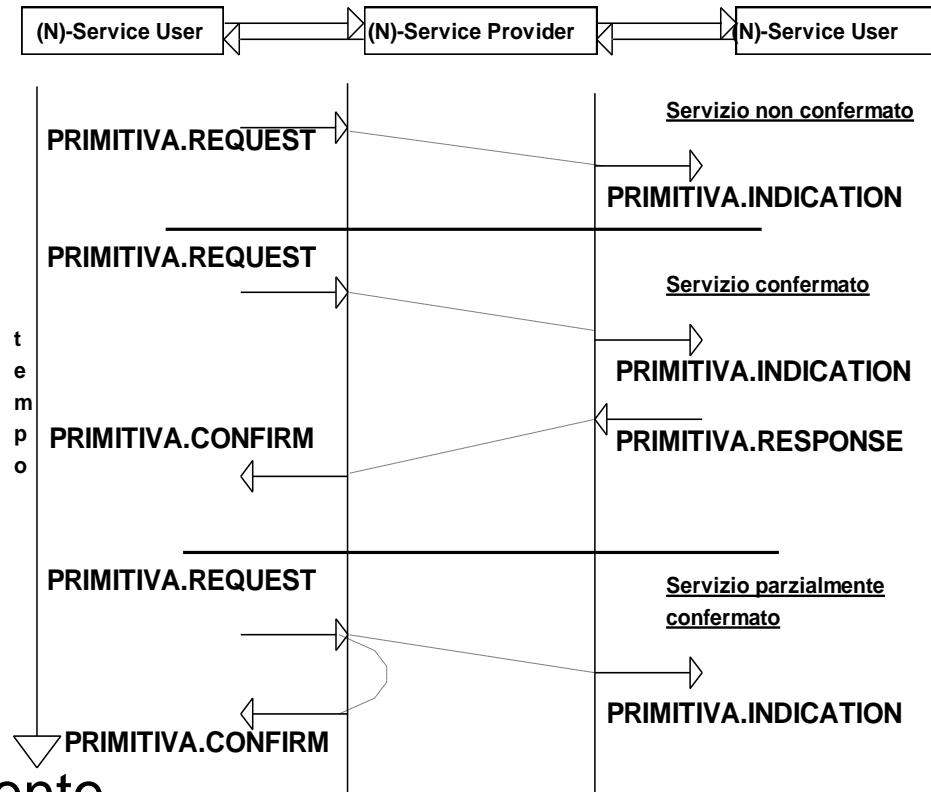
nessuna conferma al cliente

**Primitiva sincrona**

(risultato al cliente) **con conferma**  
**e azione al servitore**

**Primitiva asincrona bloccante**  
**solo conferma al cliente**

Si noti l'evento non richiesto dall'utente  
stimolato dalla comunicazione (la parte indication)



# SEQUENZA DELLE PRIMITIVE

---

Tipica sequenza con connessione attivata dall'iniziatore

<b>CONNECT</b>	apertura della connessione con negoziazione
<b>DATA</b>	invio dati
<b>DISCONNECT</b>	chiusura della connessione

Per la **CONNECT**

<b>CONNECT . request</b>	- Richiesta di stabilire una connessione
<b>CONNECT . indication</b>	- Segnale al chiamato
<b>CONNECT . response</b>	- Il chiamato accetta/rifiuta di connessione
<b>CONNECT . confirm</b>	- Conferma al chiamante della connessione

Per la trasmissione dei messaggi **DATA**

<b>DATA . request</b>	- Invio dati
<b>DATA . indication</b>	- Segnala dati con la qualità della connessione

# OSI vs TCP/IP INTERNET

---

**connessione OSI** tipicamente con impegno intermedi e QoS  
**connessione TCP/IP** solo best effort e impegno solo endpoint

Il modello di **standard OSI** ha determinato un catalogo di descrizione della possibili realizzazioni e dei meccanismi che sono necessari in una comunicazione, in modo astratto, NON una **realizzazione specifica** (che non è mai standardizzata)

Quindi le **implementazioni possono essere molto semplificate**

Il modello di **standard TCP/IP** adotta invece la idea della realizzazione di **ogni singolo standard** per la parte di meccanismi, e parte anche da un loro uso in una **implementazione specifica accettata dalla comunità di utenti (Request For Comments RFC)**

Ovviamente, TCP/IP si ispira fortemente al modelli OSI, semplificando le proposte e proponendole alla comunità di utenti

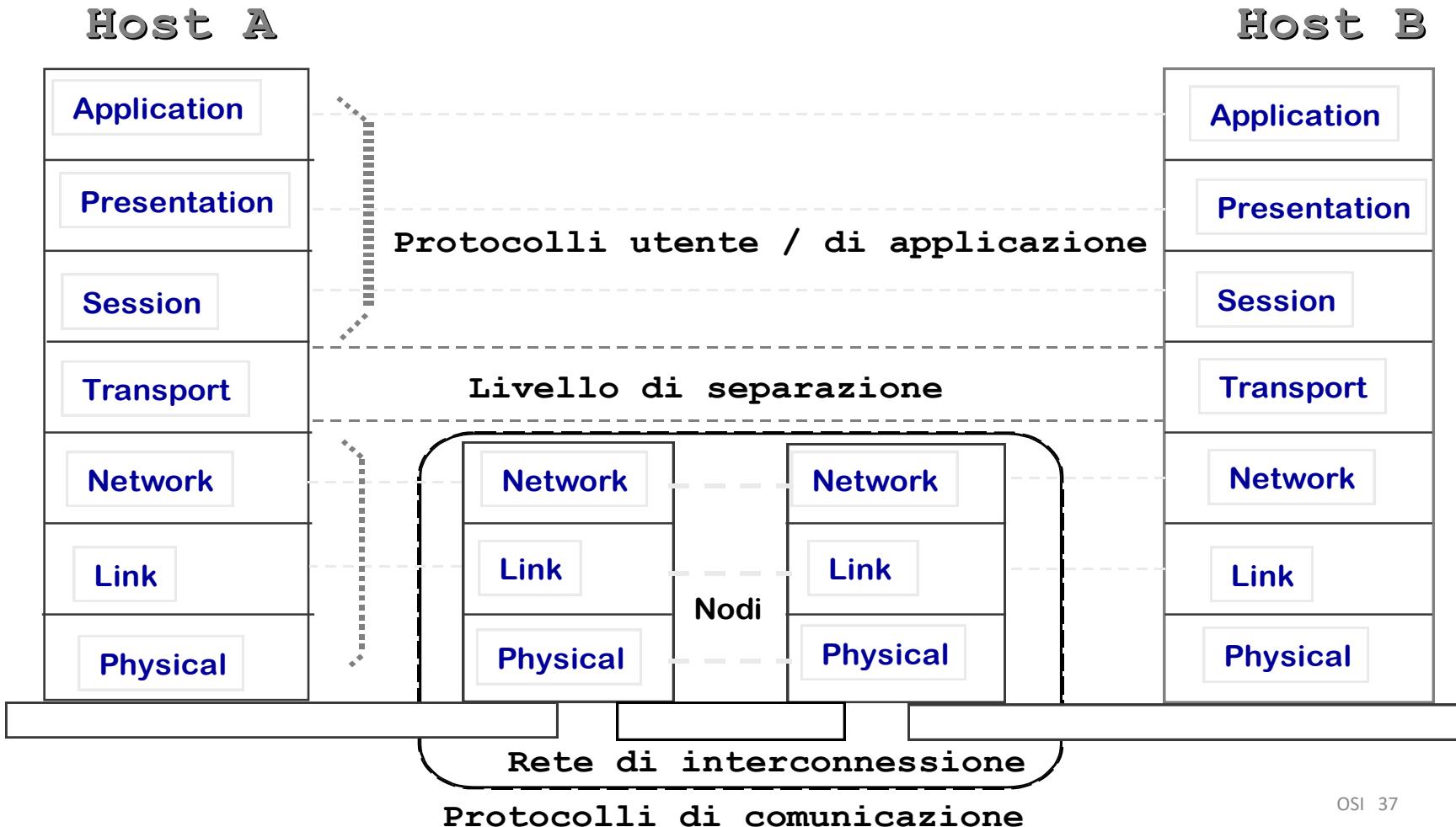
# OSI – LIVELLI E TRASPORTO

Comunicazione

i livelli fino sotto il trasporto

Applicazione

i livelli sopra il trasporto



# OSI – LIVELLI E COMPITI

#	Layer Name	Key Responsibilities	Data Type Handled	Scope	Common Protocols and Technologies
1	Physical	Encoding and Signaling; Physical Data Transmission; Hardware Specifications; Topology and Design	Bits	Electrical or light signals sent between local devices	(Physical layers of most of the technologies listed for the data link layer)
2	Data Link	Logical Link Control; Media Access Control; Data Framing; Addressing; Error Detection and Handling; Defining Requirements of Physical Layer	Frames	Low-level data messages between local devices	IEEE 802.2 LLC, Ethernet Family; Token Ring; FDDI and CDDI; IEEE 802.11 (WLAN, Wi-Fi); HomePNA; HomeRF; ATM; SLIP and PPP
3	Network	Logical Addressing; Routing; Datagram Encapsulation; Fragmentation and Reassembly; Error Handling and Diagnostics	Datagrams / Packets	Messages between local or remote devices	IP; IPv6; IP NAT; IPsec; Mobile IP; ICMP; IPX; DLC; PLP; Routing protocols such as RIP and BGP
4	Transport	Process-Level Addressing; Multiplexing/Demultiplexing; Connections; Segmentation and Reassembly; Acknowledgments and Retransmissions; Flow Control	Datagrams / Segments	Communication between software processes	TCP and UDP; SPX; NetBEUI/NBF
5	Session	Session Establishment, Management and Termination	Sessions	Sessions between local or remote devices	NetBIOS, Sockets, Named Pipes, RPC
6	Presentation	Data Translation; Compression and Encryption	Encoded User Data	Application data representations	SSL; Shells and Redirectors; MIME
7	Application	User Application Services	User Data	Application data	DNS; NFS; BOOTP; DHCP; SNMP; RMON; FTP; TFTP; SMTP; POP3; IMAP; NNTP; HTTP; Telnet

# OSI – LIVELLI BASSI

---

I 3 Livelli inferiori OSI (**Fisico, Data Link, Rete**) sono detti **fisici**

Un Livello intermedio: **Trasporto**

Tutti questi livelli forniscono un **meccanismo trasparente per il trasporto**, con funzioni base che includono:

- **controllo degli errori** dovuti al rumore o altra causa
- **controllo di flusso dei dati**
- **modelli di indirizzamento** per identificare end system (naming)
- per rete le **strategie di routing** per trasferire i dati (internetworking)

## Protocolli

**livello fisico**

ripetitore protocollo

**RS232**

**livello data link**

bridge protocolli

**Ethernet, HDLC, PPP**

# OSI – LIVELLO NETWORK

**Livello di RETE tiene conto dei nodi intermedi tra due pari**

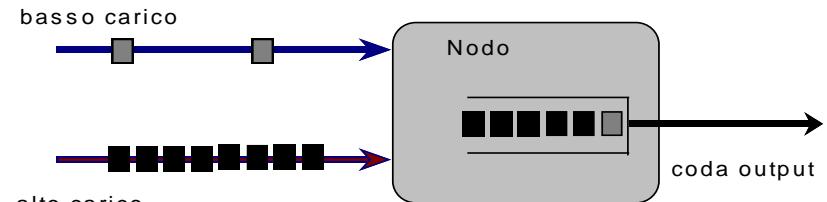
Impossibilità di controllare direttamente il cammino da un qualunque mittente ad un qualunque destinatario

**Il livello di network si occupa delle diverse realizzazioni di routing tra reti diverse oltre a definire il sistema di nomi delle entità**

**Obiettivo:** passaggio delle informazioni **interferendo meno possibile** sul comportamento locale

**COMPITI del LIVELLO di RETE**

- **Indirizzamento** (vedi nomi di IP)
- **Controllo di flusso** tra due pari
- **Controllo di congestione** nel sistema intero

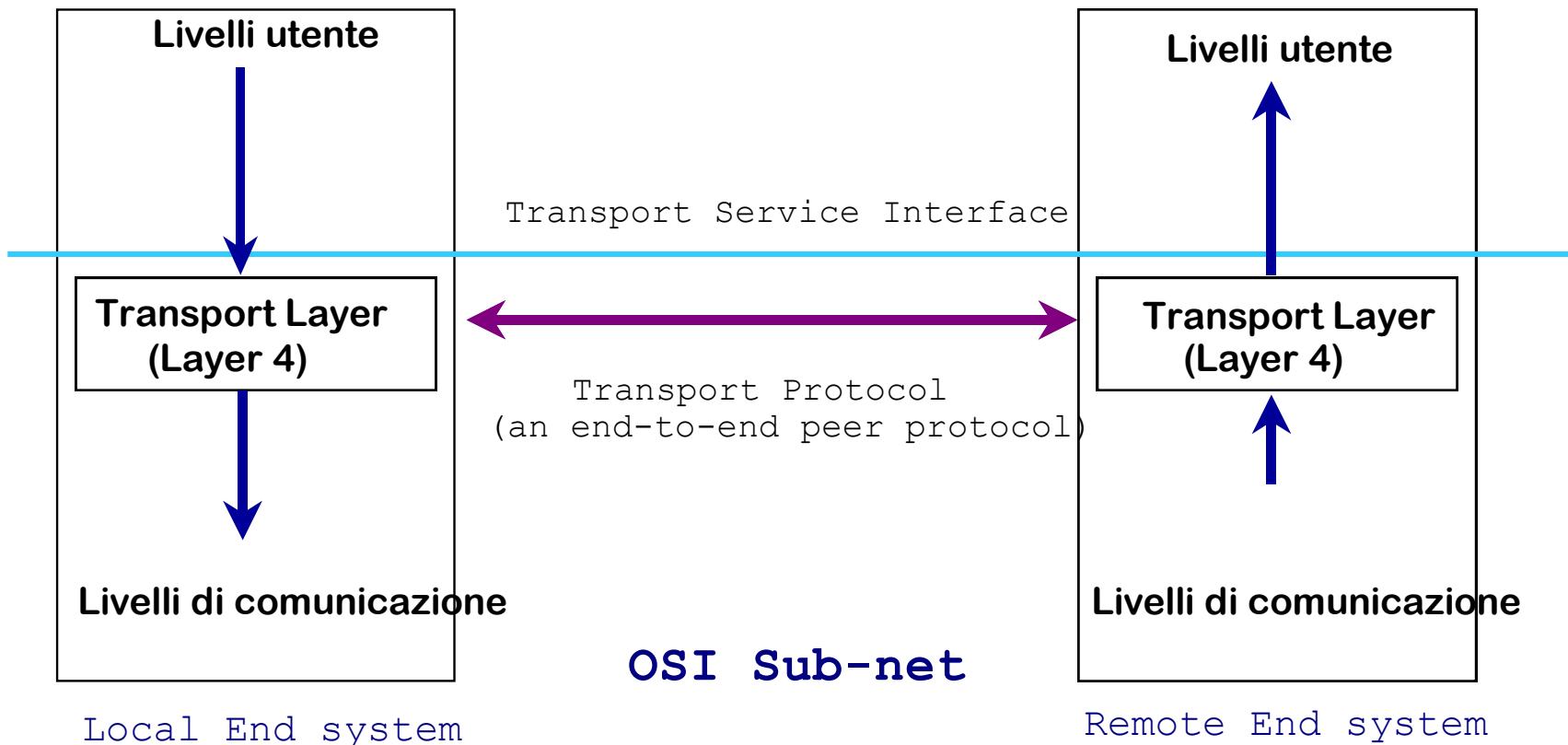


**OBIETTIVI:** migliorare efficienza ed evitare ingiustizia, deadlock

Principio di separazione: i nodi intermedi devono potere interagire solo per le funzionalità necessarie e non essere toccati ai livelli applicativi

# OSI – LIVELLI E TRASPORTO

Il trasporto separa i livelli applicativi da quelli fisici  
Applicazione si localizza sopra al trasporto



# OSI – LIVELLI E TRASPORTO

---

il Trasporto comincia a considerare la struttura dei nodi partecipanti, in particolare gli endpoint

Ogni livello ha le **proprie entità** e **SAP (anche più di una)**

Se focalizziamo solo **T** e **N**, un nodo potrebbe avere molte **T-SAP di trasporto** e una **sola N-SAP di rete**: in questo caso le applicazioni devono specificare quali enti sono coinvolti per ogni comunicazione

**Lo stesso discorso vale per ogni *livello superiore***: un pari che vuole comunicare deve specificare tutta la pila di SAP proprie e anche quelle che permettono di arrivare all'altro in modo non ambiguo

Anche gli intermedi devono essere considerati per la connessione con qualità e devono essere negoziati

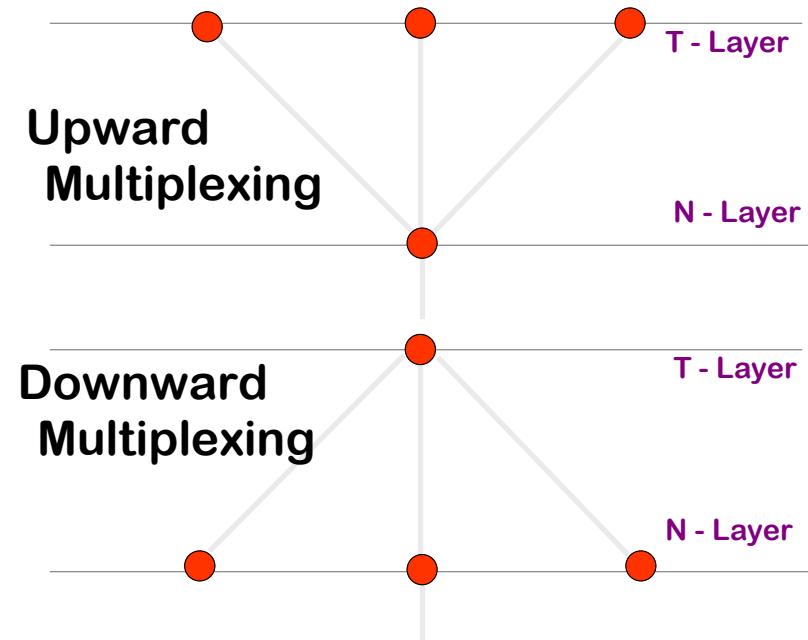
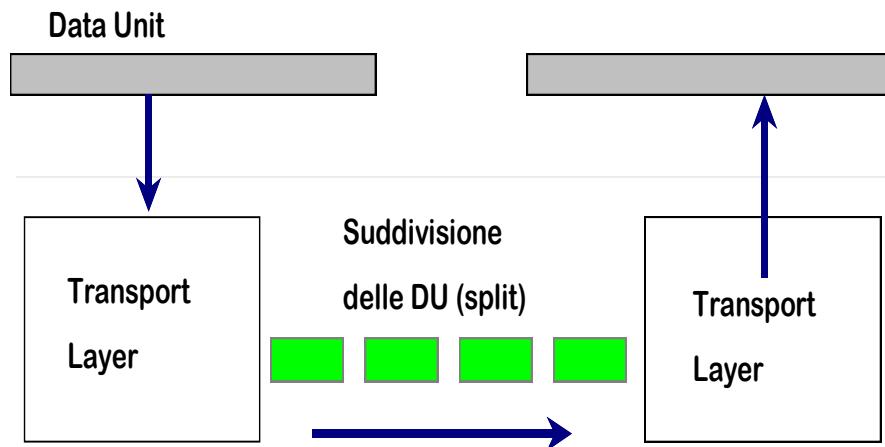
La primitiva **CONNECT** mette in gioco **molte entità su tutti i nodi interessati alla connessione e mantiene le risorse impegnate**

# OSI – LIVELLI E TRASPORTO

## Livello T - Funzioni possibili

Il trasporto può **spezzare** il dato e **ricomporlo** dopo averlo portato suddiviso fino al pari

Il trasporto può lavorare **unendo** o **decomponendo** (SAP) flussi di trasporto rispetto a quelli di rete (**Multiplexing / Demultiplexing**)



# LIVELLO DI TRASPORTO - CONNESSIONE

---

Trasporto come **livello end-to-end per separare i livelli** relativi alla comunicazione da quelli più vicini alla applicazione

**Obiettivo** - spedizione di dati sul canale di connessione con correttezza, con certi tempi di risposta, e con una certa qualità di servizio, su richieste dal livello S superiore

## MODALITÀ CONNECTION ORIENTED

- apertura e terminazione di una connessione - CONNECT/ DIS...
- trasferimento di dati normali e **privilegiati (expedited)** – DATA, ...

I **dati expedited** sono soggetti ad un controllo di flusso separato che permette l'invio di messaggi di controllo anche se il servizio per i dati normali è bloccato

In generale, **ogni primitiva di servizio** per un dato livello prevede un certo insieme di parametri

Anche MODI NON CONNESSI (vedi INTERNET)

# PRIMITIVE LIVELLO TRASPORTO

---

## Livello di Trasporto agisce su base end-to-end

Le primitive del servizio di Trasporto sono poche (4) e semplici, con relativamente pochi parametri significativi

<u>primitiva</u>	<u>tipo di servizio</u>	<u>parametri di servizio</u>
T-CONNECT	<b>servizio confermato</b>	<b>indirizzo del chiamante e del chiamato, opzione per l'uso di dati privilegiati, qualità di servizio e dati d'utente.</b>
T-DATA	<b>servizio non confermato</b>	<b>dati di utente</b>
T-EXPEDITED-DATA	<b>servizio non confermato</b>	<b>dati di utente</b>
T-DISCONNECT	<b>servizio non confermato</b>	<b>ragione della terminazione, dati d'utente</b>

# LIVELLO SESSIONE

---

Sul trasporto che lavora da nodo a nodo, la prima esigenza della sessione è il **supporto al dialogo**

Un dialogo può avere **molte dimensioni possibili** (scambio testo tipo chat, scambio file, video in real-time, ecc. ecc.) e anche molte **specifiche differenziate di qualità** (scaricamento di molti frame alla volta, senza ripartire da zero, garanzia di persistenza, dialogo uno alla volta, ... )

**LA SESSIONE** considera e determina i meccanismi per il dialogo tra entità diverse, tenendo in conto le possibilità tra due pari che comunicano in modo anche vario ed eterogeneo

Nella sessione, il **dialogo** può

- essere **bidirezionale**
- essere **molteplice** e strutturato in **attività separate** e diverse
- considerare le **risorse impegnate**
- avere garanzie di **correttezza e affidabilità**

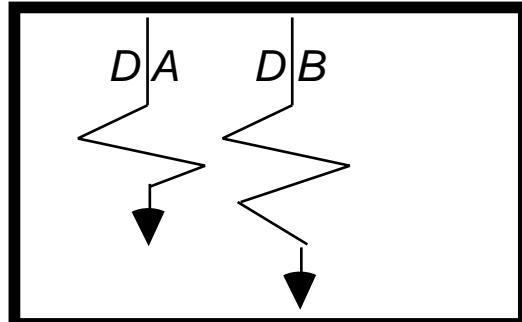
# LIVELLO SESSIONE - DIALOGO

Il **dialogo** trae vantaggio da un **supporto ad hoc** di funzioni che siano capaci di **garantire i protocolli e i meccanismi per il migliore supporto**

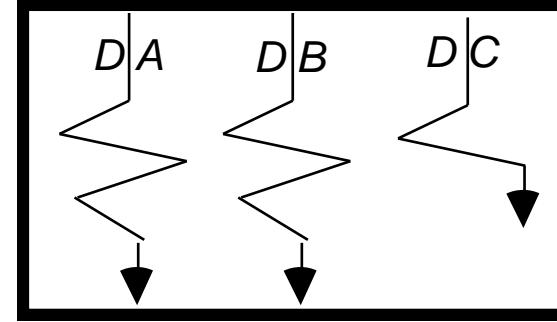
LA SESSIONE standardizza una serie di componenti standard per avere a disposizione tutte le funzionalità necessarie secondo i requisiti accettati

Nella sessione, avremo **altre primitive oltre alla DATA** e diversi supporti a tipi di dati ad-hoc definiti qui

Si pensi alla sessione come un insieme di **dialoghi diversi** tra due utenti



Applic A



Applic B

# SERVIZI DI SESSIONE - SINCRONICITÀ

---

Il **livello di Sessione** offre servizi analoghi a quelli del livello di Trasporto, ed ogni livello:

- **apertura della connessione e sua terminazione**
- **trasferimento dati**
  - si possono avere fino a **quattro tipi** di dato diversi oltre ai **dati normali**

Servizi aggiunti e specializzati per:

- **gestione dell'interazione**  
modalità di dialogo half-duplex, full-duplex o simplex
- **sincronizzazione**  
inserimento dei punti di sincronizzazione (**checkpoint**) e gestione delle eccezioni
- **autorizzazioni** alla azione

I punti di **sincronizzazione sono nuovi dati**, come pure le **autorizzazioni** messe in gioco negli stati della comunicazione

# PRIMITIVE LIVELLO SESSIONE

---

Il livello di Sessione coordina il dialogo basandosi sul servizio offerto organizzato in **unità funzionali**, ognuna legata ad un **insieme di primitive e parametri**

Il numero delle unità funzionali cresce per i livelli verso l'applicazione  
Servizio di Sessione offre **58 primitive** raggruppate in **diverse unità funzionali (14)**

uso di connessione, con QoS negoziata, e semantica più complessa

**Il Dialogo viene strutturato** attraverso

- azioni di **controllo**, come sincronizzazione, trasferimento di autorizzazioni, ecc.
- **attività**: il dialogo è diviso in attività indipendenti che si possono gestire (iniziare, sospendere, cancellare, ecc.)
- **eccezioni**: è possibile notificare eccezioni ai servizi corrispondenti

Ogni pari può richiedere il livello di servizio adatto alle esigenze sue e della invocazione

# UNITÀ FUNZIONALI DI SESSIONE

---

Nel livello di sessione possono essere presenti molti servizi, raggruppati in **unità funzionali**, ad esempio:

- **Kernel**: composto dalla sola unità funzionale Kernel
- **Basic Combined**: composto dal sottoinsieme Kernel e dall'unità funzionale Half Duplex o Duplex
- **Half Duplex**: possibilità di invio dati in base al possesso di token
- **Typed data**: possibilità di invio dati anche fuori turno
- **Basic Synchronized**: composto dal sottoinsieme Basic Combined e dalle unità funzionali Minor, Major Synchronize, Typed Data, Negotiated Release, e Resynchronize
- **Basic Activity**: composto da Basic Synchronized (modalità half-duplex e punti di sincronizzazione minore) e dalle unità Activity Management ed Exception Report

- ...

# SINCRONIZZAZIONE DELLA SESSIONE

---

Possibilità di intervenire automaticamente sul dialogo tra pari

- Trasmissione di un file da due ore bloccata dopo un'ora ⇒  
Si riprende dal risultato del trasferimento precedente
- Se si verificano errori nella comunicazione ⇒ roll-back
- Nel trasferimento di molti MByte, se crash ⇒ si ricomincia (?)

I punti di sincronizzazione sono punti per introdurre checkpoint nel dialogo e consentire di articolarlo in modo organizzato

Due tipi previsti:

## 1. punti di sincronizzazione maggiore

il mittente attende (modo **sincrono bloccante**) che il ricevente confermi

## 2. punti di sincronizzazione minore

**invio con conferma o meno**, il mittente può continuare a spedire dati o punti di sincronizzazione. Il **ricevente non deve segnalare** subito la ricezione di un punto minore, e la conferma di un **punto minore conferma anche tutti i punti precedenti**

# PUNTI DI SINCRONIZZAZIONE

---

I punti di sincronizzazione usati per ri-sincronizzarsi verso uno stato definito dai punti stessi in caso di recovery

In caso di punti maggiori, attesa fino ad alla .confirm

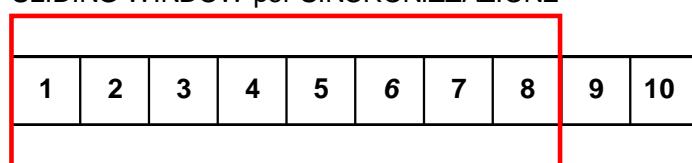
In caso di punti minori, accumulo ed eventualmente anche attesa con un attesa massima limitata

Tipicamente, si può negoziare il numero di punti di sincronizzazione minore che sono in attesa di conferma

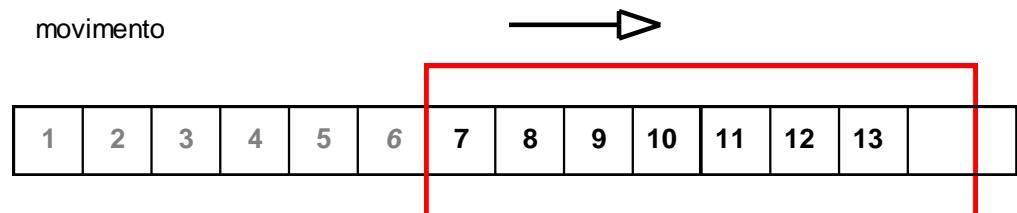
Si determina la dimensione di una finestra che scorre (**sliding window**) di punti non confermati

Al riempimento della finestra, il mittente deve aspettare conferma per procedere con altri punti di sincronizzazione

SLIDING WINDOW per SINCRONIZZAZIONE



movimento



NUOVA SLIDING WINDOW

# SERVIZI DI SESSIONE - TOKEN

---

**Strutturazione e sincronizzazione del dialogo attraverso oggetti astratti detti **token**, intesi come gettoni di autorizzazione**

Un solo utente possiede il token in ogni momento ed ha il diritto di uso di un insieme di servizi di Sessione

La primitiva **S-CONNECT** per stabilire la connessione **permette di negoziare anche i token**

ad esempio, l'utente che richiede la connessione e l'utente che la accetta indicano i servizi da implementare

Vari tipi di token distinti come diritto di:

- **data token**: spedire i dati in Half Duplex
- **release token**: richiedere la terminazione
- **synchronize minor token**: creare punto di sincronizzazione minore
- **synchronize major token**: creare punti maggiori

L'intersezione degli insiemi di requisiti determina la S-connessione

# SERVIZI DI SESSIONE - GESTIONE

---

I punti di sincronizzazione di un dialogo possono essere usati nel recovery per ritrovare uno stato significativo in modo automatico o negoziato

In caso di recovery, si prevedono anche strategie diverse  
Quelle previste consentono

1. **abbandono:** reset della comunicazione  
L'utente può decidere di ripeterla
2. **ripristino:** la comunicazione è riportata nello stato precedente  
L'ultimo punto di **sincronizzazione maggiore** confermato (anche con raffinamenti)
3. **ripristino diretto dall'utente:** la comunicazione è riportata in uno stato arbitrario senza controllo delle conferme "mancanti" di punti di sincronizzazione  
È compito delle applicazioni decidere in modo coordinato su uno stato da cui ricominciare la trasmissione

# LIVELLO DI PRESENTAZIONE

---

**La codifica delle informazioni non è univoca e ogni pari può usare codifiche diverse**

Il livello di Presentazione offre i servizi per consentire **una corretta interoperabilità sui dati**, ad esempio superare il problema della codifica dei dati dei diversi pari, da mandare e da ricevere

Il livello di presentazione **norma gli strumenti e i protocolli** necessari per una corretta gestione dei **dati eterogenei nei sistemi aperti**

La presentazione quindi affronta tutto il problema della **rappresentazione dei dati**, e delle **differenze naturali tra i sistemi che comunicano**,  
**ma anche dei casi di necessità di codifiche ad hoc** per compressione dei dati (**efficienza**) o crittografia (**sicurezza**)

# LIVELLO DI PRESENTAZIONE - DATI

---

**Le informazioni su nodi diversi possono essere eterogenee per molte ragioni, e a diversi livelli**

**Ogni pari può usare codifiche diverse dipendentemente dalla architettura sistema, dai sistemi operativi considerati, dai linguaggi, dalla applicazioni specifiche, ....**

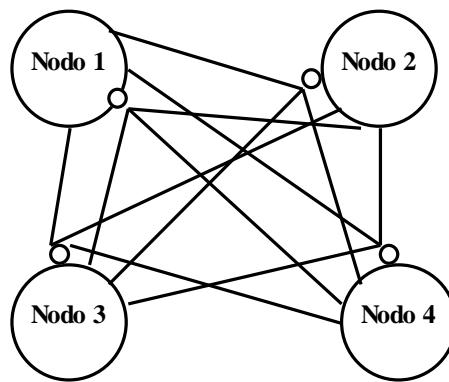
I dati devono essere scambiati dopo un accordo tra i pari che superi gli eventuali problemi di eterogeneità

- ambienti di programmazione diversi                   **HTML, XML, ,...**
- linguaggi di programmazione diversi               **C,C++,C#, ADA, Java,...**
- sistemi operativi diversi                              **UNIX , VMS, WINXX**
- architetture diverse                                      **ARM, RISC, ...**

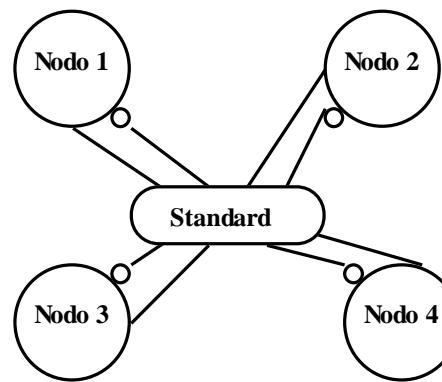
La presentazione quindi come livello per la gestione della rappresentazione e l'accordo sui dati

# RAPPRESENTAZIONI DATI DIVERSE

---



Sono necessarie 12 funzioni  
di conversione del formato di dati



Sono necessarie 8 funzioni  
di conversione del formato di dati

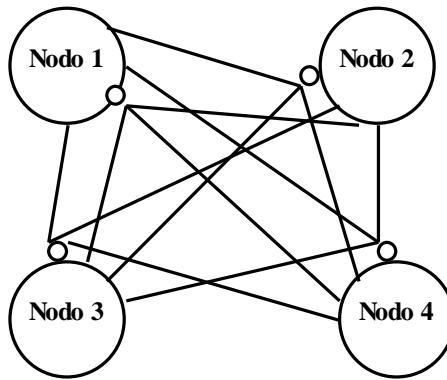
**In caso di disomogeneità dei dati, per la comunicazione tra nodi eterogenei, ci sono due soluzioni:**

1. dotare ogni nodo di **tutte le funzioni di conversione** possibili per ogni possibile rappresentazione dei dati
2. concordare un **formato comune di rappresentazione dei dati**: ogni nodo possiede le funzioni di conversione da/per questo formato

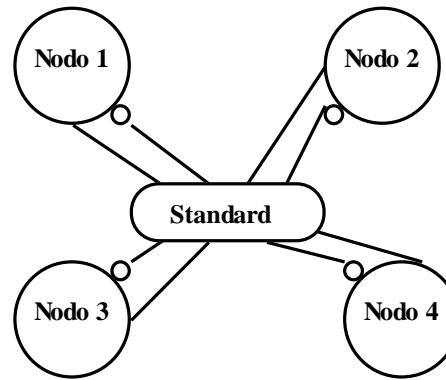
La prima  $\Rightarrow$  **elevata performance**

La seconda  $\Rightarrow$  **implementazione di un minore numero di funzioni di conversione**

# STANDARD PER FORMATI ETEROGENEI



Sono necessarie 12 funzioni  
di conversione del formato di dati



Sono necessarie 8 funzioni  
di conversione del formato di dati

**Con N nodi eterogenei che trasformazioni? Per le comunicazioni**

Nel **primo caso** le funzioni di conversione sono  $N*(N-1)$

Nel **secondo** sono N dal formato locale ed N verso il formato comune

Nel primo caso ogni volta una trasformazione

Nel secondo ogni volta due trasformazioni

**Si usa sempre e solo la seconda**

# MOLTI CASI DIVERSI E PROTOCOLLI ...

---

**SE c'è un completa uniformità del FORMATO DATI**

Cioè tutti usano lo **stesso formato** e nessuna disomogeneità

⇒ **non si fanno trasformazioni** (vedi l'uso di **Java JVM**)

**ALTRIMENTI bisogna tenerne conto nel progetto**

Se abbiamo **ETEROGENEITÀ** e molti **formati diversi?**

**SE C'È ACCORDO STABILITO... ⇒ (parte concreta dell'accordo)**

**Che rappresentazione standard usiamo  
e trasformazione dal formato del pari all'altro**

(vedi stringhe nelle diverse architetture big-endian vs. little-endian

In Internet **big-endian**)

**MEGLIO** - in questo caso useremo tipicamente un **unico formato intermedio standard** e le **trasformazioni** saranno fatte **due volte** per ogni comunicazione, dal *formato locale allo standard* e *dallo standard al locale*

# MOLTI CASI DIVERSI E PROTOCOLLI...

---

**Se non solo** abbiamo **ETEROGENEITÀ** e **molti formati diversi**  
**Ma abbiamo anche situazioni dinamiche**, in cui, non avendo  
preparato una comunicazione e un protocollo predefinito, **non sappiamo come dialogare con gli altri**

**Dobbiamo prima decidere cosa comunicare e dopo in che modo farlo usando il protocollo stesso che possiamo definire**

**SE NON C'È ACCORDO... ⇒ (parte astratta dell'accordo)**

**... su cosa dobbiamo dire, dobbiamo negoziarlo**

**Come accordarsi? Usando un linguaggio comune**

**Bisogna determinare un linguaggio comune per tentare l'accordo**

**SE NON C'È ACCORDO... ⇒ (parte concreta dell'accordo)**

**...ma sappiamo cosa dire, ci dobbiamo accordare sul formato dei dati standard e della trasformazione specifica dei contenuti di interesse**

**Bisogna usare un linguaggio di descrizione dei dati**

# ESEMPI DI ACCORDO ASTRATTO

---

Uso del linguaggio comune noto a entrambi (accordo concreto per decidere chi fa echo)

In caso di telnet, si è definito un **linguaggio ad-hoc** per l'accordo sulle specifiche del video standard

«*Chi fa l'echo?*»

«*Lo fai tu?*»

«*Io no, Fallo tu!*»

«*No, tu! ...*»

E se non ci fosse un linguaggio comune, si ricorre ad uno standard noto ad entrambi per coordinare un linguaggio comune concreto (**accordo astratto**)

«*Cosa mi dici?*»

«*File HTML con nome, cognome*»

«*preferirei cognome e nome*»

«*OK*»

parliamo dei componenti **1.3.6.1.2.1.4** in albero X.500

# **DATI E FORMATI USATI (CASO CONCRETO)**

---

In ogni caso dobbiamo anche chiederci come viaggiano i dati per discorsi di QoS ed efficienza

- come **pura rappresentazione (valori)**
- anche con dei **descrittori del dato (descrizione e valore)** che producono ridondanza e garanzie di qualità

**Solo i valori ⇒ efficienza, ma anche descrizione ⇒ affidabilità**

**SE C'È ACCORDO ⇒ TRASFORMAZIONI E RIDONDANZA**

Possiamo usare **diversi gradi di attributi dei dati** a secondo del costo associato alla comunicazione (impegno di banda)

I dati possono prevedere

**valore**

**lunghezza del valore, valore  
tipo, lunghezza campo valore, valore**

# PRESENTAZIONE: IL CASO ASTRATTO

---

Necessità di accordarsi (**in astratto**) e definire

- **il contesto di comunicazione**
- **il soggetto della comunicazione**
- **la semantica delle informazioni**
- **le informazioni vere e proprie**

per poi comunicare in **concreto (fase di accordo presente)**

Il contesto potrebbe definire di cosa stiamo parlando (magazzino), il soggetto di cosa (parti dell'oggetto), per poi dare significato (tipo di attributi) e specifica di ciascuno

**Il livello di presentazione stabilisce come negoziare e definire una base comune**

Spesso l'accordo può essere ottenuto solo con **protocolli detti di negoziazione, ossia a molte fasi** (con durata non predefinita e anche lunga nei casi peggiori)

# PROTOCOLLI DI NEGOZIAZIONE

---

**Protocolli con numero di fasi non predicibile, spesso determinato da eventi verificatisi durante il protocollo stesso**

**BIDDING (Contract Net)** tra sender e receiver si prevedono molte fasi, almeno 5 anche ripetute

1. il sender fa un **broadcast della propria esigenza**
2. i receiver fanno una **offerta** (bid)
3. il sender **sceglie** tra i bid dei receiver
4. il receiver **accoglie** l'ok definitivo (contract)
5. accordo

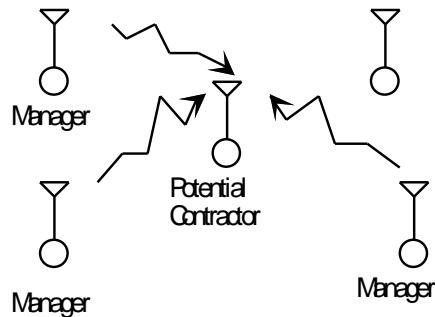
**Non ci sono prenotazioni delle fasi: alla fase 4, il receiver può rifiutare e si riparte da 3 (o peggio da 1)**

**SELEZIONE molto FLESSIBILE ma COSTOSA**

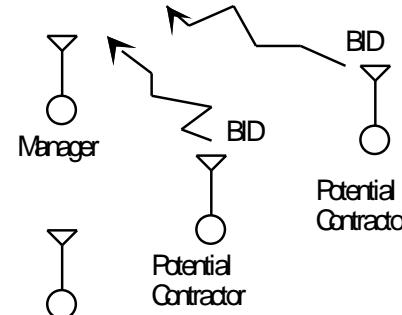
# BIDDING (CONTRACT NET)

## Protocolli a fasi multiple

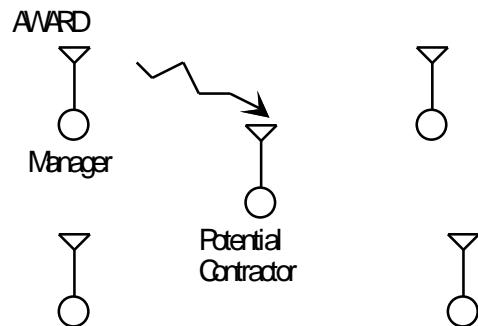
### 1. Announce (richiesta)



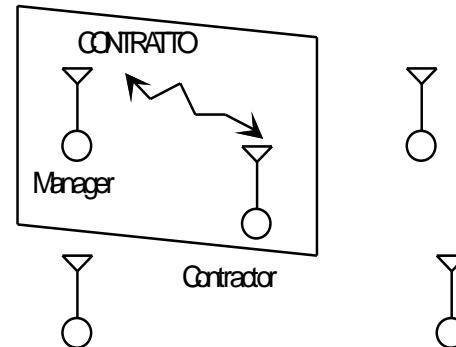
### 2. Fase di Bidding



### 3. Scelta del Bidder (award)



### 4. Contratto finale



# PROTOCOLLI DI PRESENTAZIONE

---

Il livello di presentazione definisce:

- un **linguaggio astratto di specifica (parte astratta accordo)** (**ASN.1 Abstract Syntax Notation**) per casi dinamici
  - e un **linguaggio concreto di descrizione (parte concreta accordo)** (**BER Basic Encoding Rules**) usato estensivamente
- 

- definendo un contesto di comunicazione
- distinguendo informazioni in forma astratta e concreta
- permettendo di specificare i **dati in modo astratto e le informazioni di controllo in modo indipendente dalla forma concreta** (se è il caso, e solo se necessario)
- consentendo di definire la **forma comune di rappresentazione concreta dei dati** (non coincidente con la precedente astratta e richiedendo le usuali trasformazioni da forma locale a forma comune)

# BER - LINGUAGGI DI PRESENTAZIONE

**BER - sempre necessario**  
**(BER Basic Encoding Rules)**

**sempre usato**

Triple Tag-Length-Value:  
codifica a discesa ricorsiva

```
address.source = "Suna"  
address.destination = "Decb"  
length = 3  
data = 'x', 'y', 'z'
```

<u>Primitive types</u>	<u>BER</u>
BOOLEAN	
INTEGER	02
OCTETSTRING	04
IA5String	16

<u>Constructor types</u>	
SEQUENCE	30
SEQUENCE OF	
SET	
SET OF	
CHOICE	



```
30 22
  30 12
    16 04 'S' 'u' 'n' 'a'
    16 04 'D' 'e' 'c' 'b'
  02 01 03
  04 03 01 02 03
```

# ASN.1 – LINGUAGGI DI PRESENTAZIONE

---

**ASN.1 - usato in caso di bisogno per descrivere dati  
(ASN.1 Abstract Syntax Notation) se si deve creare l'accordo**

## Primitive types

BOOLEAN  
INTEGER  
OCTETSTRING  
IA5String

```
Address ::= SEQUENCE {  
    addr_src    IA5String,  
    addr_dst    IA5String  
}
```

## Constructor types

SEQUENCE  
SEQUENCE OF  
SET  
SET OF  
CHOICE

```
Pdu ::= SEQUENCE {  
    pdu_ad      Address,  
    pdu_len     INTEGER,  
    pdu_data    OCTETSTRING (SIZE 1024)  
}
```

**In molti casi, si descrivono dati e si negozia quali  
(confrontare con XML e HTML: si usa sempre la rappresentazione  
dei dati XML con il contenuto HTML?)**

# ASN.1 – LINGUAGGI DI PRESENTAZIONE

---

## ASN.1 Example Types

```
PersonnelRecord ::= SET {  
    Name,  
    title [0] VisibleString,  
    children [1] IMPLICIT SEQUENCE OF  
        ChildInformation DEFAULT {}  
}
```

```
ChildInformation ::= SET {  
    Name, dateOfBirth [0] Date}
```

```
Name:: = SEQUENCE {  
    givenName VisibleString,  
    familyName VisibleString}
```

# USO DEI PROTOCOLLI LIVELLO 6

---

In un caso generale, possiamo avere bisogno di **ASN.1 Abstract Syntax Notation**, per accordarci su cosa vogliamo comunicare e fare una negoziazione **BER Basic Encoding Rules**, per stabilire quale sia il formato comune dei dati da trasferire (numero di trasformazioni lineari)

- Nel caso **non abbiamo un accordo**, procediamo e definire con **ASN.1** che dati ci interessano (possiamo aiutarci con **X.500**)

Questa fase non serve se lo sappiamo a priori o lo abbiamo già negoziato

- In tutti i casi, **BER** ci consente di definire quale sia il formato dei dati standard a cui fare riferimento nella comunicazione

Questo formato standard è quello da cui si trasforma da ogni *rappresentazione locale*, nei due versi (le due trasformazioni necessarie)

# LIVELLO DI APPLICAZIONE

---

Il livello di Applicazione è il livello che si interfaccia con l'utente finale della comunicazione in base al modello OSI

## Obiettivo Astrazione

**nascondere la complessità dei livelli sottostanti coordinando le applicazioni distribuite**

Il livello applicativo OSI standard definisce un **insieme di servizi indipendenti dal sistema** ed ambienti standard a programmi di utente o ad utenti (ISO 9545):

**Message Handling System**

**MHS**

**Directory service**

**X.500**

**System Management**

**X.700**

**Common Management Information Service & Protocol**

**CMISE & CMIP**

**File Transfer, Access and Management**

**FTAM**

**Virtual Terminal Standard**

**VT**

**Distributed Transaction Processing**

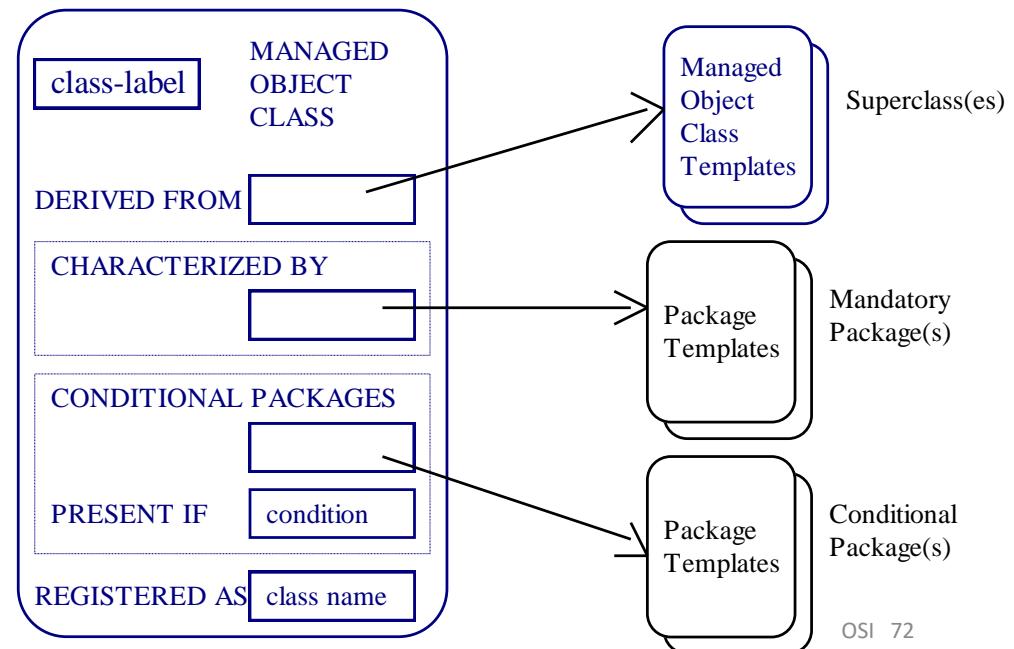
**DTP**

# LIVELLO DI APPLICAZIONE

OSI adotta un approccio particolare basato **sul modello ad Oggetti per la specifica delle applicazioni**

- **Uso di template e package per definire gli oggetti**
- Pura **ereditarietà statica** tra astrazioni
- **Oggetti** da manipolare come interfaccia ed espressi attraverso l'uso di **package** (anche condizionali)

Si noti la **unicità dei nomi**  
come presupposto di base per  
l'accordo e il coordinamento  
**NOMI UNICI** come servizio  
(X.500)



# LIVELLO DI NOMI STANDARD

## Servizi X.500

Il servizio di directory consente di **collocare e classificare ogni entità di interesse** (ogni dispositivo noto) in base al **contenuto degli attributi** in un sistema gerarchico di conoscenza molto accessibile (24/7)

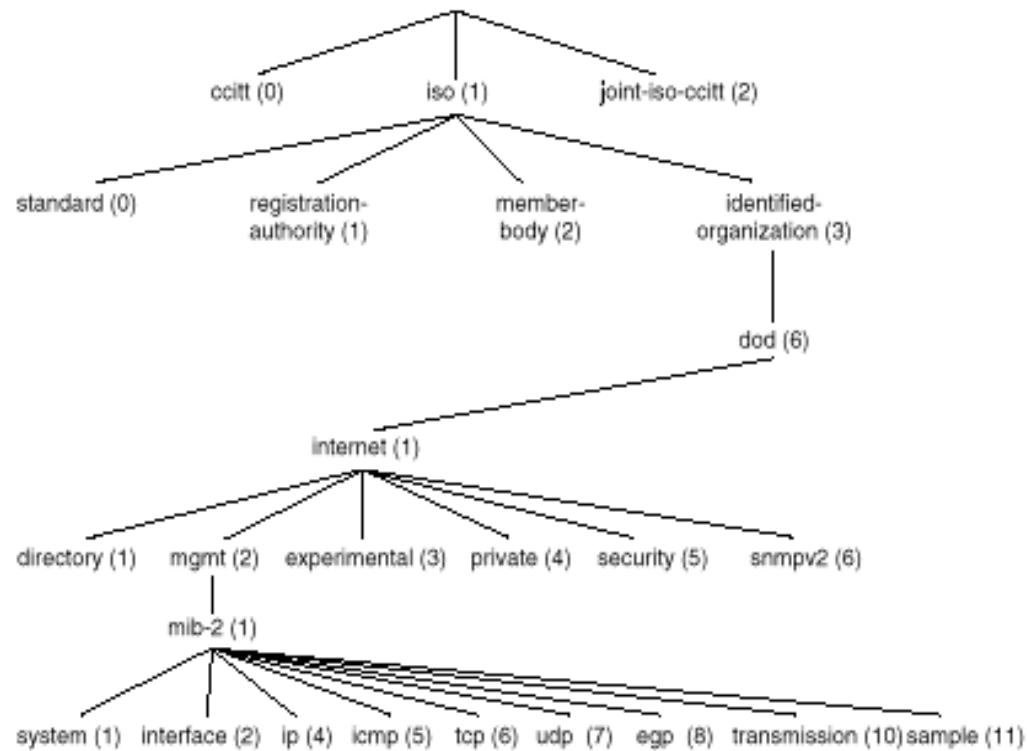
Spesso si riferisce una entità attraverso il suo **identificatore nella gerarchia unica specificato come sequenza di decisioni e scelte**

1.3.6.1.2.1.4

Il sottoalbero per la descrizione del protocollo IP per gestione

Negli altri casi, si negoziano le proprietà con ASN.1

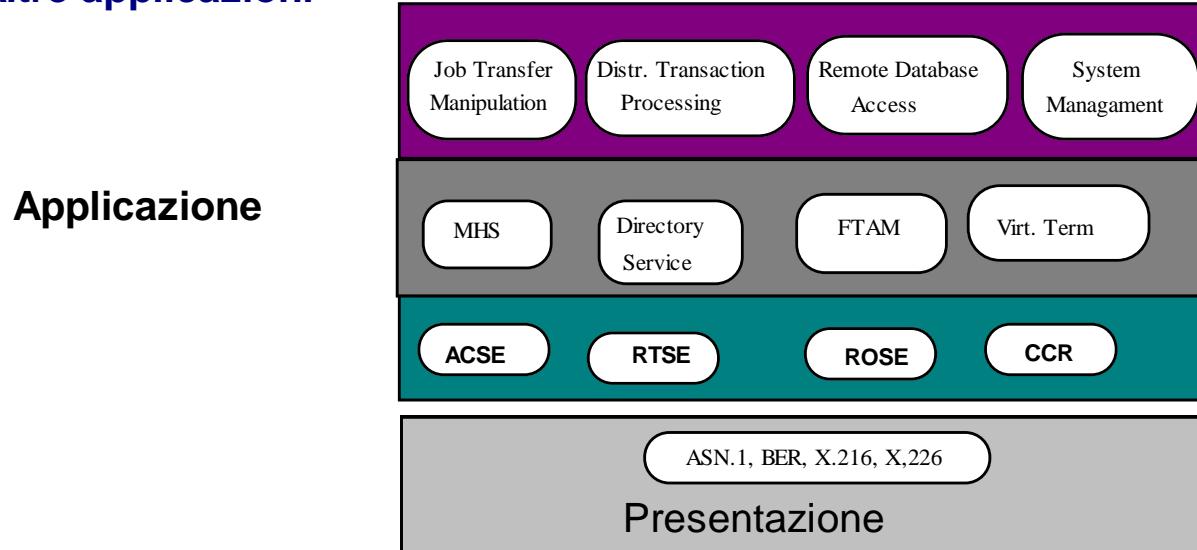
Per il directory, si veda la parte sui sistemi di nomi



# APPLICAZIONE A LIVELLI

Applicazione a sua volta come insieme di livelli e di strumenti

## Altre applicazioni



Alcuni strumenti sono a livello di base rispetto agli altri

**ACSE** (Association Control Service Element) di base per ogni servizio

**RTSE** (Reliable Transfer Service Element) per ottenere servizi affidabili

**ROSE** (Remote Operation Service Element) per operazioni remote

**CCR** (Commitment Concurrency and Recovery) per azioni multiple coordinate

# OSI E INTERNET

---

## Confronto OSI e TCP/IP, aldilà del numero dei livelli

### completezza di OSI

uso di Object-Orientation

interfaccia

implementazione

progetto completo

interesse in standard

definizione ampia e

aperta

**Qualità di servizio**

**Connessione OSI**

### limiti TCP/IP

descrizione approssimata negli RFC  
protocolli e implementazione  
insieme

implementazione solamente  
prodotti

progetto in crescita

**Internet in-segue le specifiche OSI (appena può)**