# CAPITOLO 3

# MODELLI DI RIFERIMENTO

# 3. MODELLI DI RIFERIMENTO

Agli inizi degli anni ’70 le case produttrici di hardware e software di reti orientavano la loro produzione in una prospettiva personale, cioè mirata allo sviluppo e alla commercializzazione dei propri prodotti, rendendo praticamente impossibile la comunicazione di sistemi *divers*i.
Per questa ragione, questi sistemi vengono classificati come ***closed syste*m**.
A metà degli anni ’70 l’**International Standards Organization (ISO)** avviò un processo di standardizzazione proponendo un modello di riferimento chiamato ***Open System Interconnection reference model (modello di riferimento OSI)*** il cui obiettivo era quello di permettere la comunicazione tra processi applicativi, residenti su computer di case costruttrici diverse, che dovevano rispettare un dato insieme di regole standard.

## *3.1 Il modello ISO-OSI*.

Quanti strati è opportuno mettere in una architettura a strati?
Fino adesso abbiamo risposto a questa domanda dicendo che una architettura a strati deve contenere tanti strati quanti ne richiede l’applicazione distribuita.
Se, però, si intende realizzare *un* sistema *multi-purpose*, cioè a più obbiettivi, è ovvio che un tale sistema distribuito dovrebbe, in linea di principio, fornire un certo numero di applicazioni e valutare quale è l’intersezione tra le esigenze di tutte queste applicazioni che si intendono realizzare.
C’è stato un momento in cui le grandi case costruttrici, nell'ambito della trasmissione dati, cominciarono a pensare quali potessero essere tutte le funzioni utili per creare un’architettura di base.
I primi ad intraprendere questa impresa, sono stati quelli della IBM, per consentire agli utenti di utilizzare un sistema distribuito IBM con le nuove linee di prodotto che via via la stessa IBM avrebbe tirato fuori, senza costringere l’utente a rinunciare al proprio parco terminali già acquisito. Lo scopo era quello di realizzare la copertura della eterogeneità presente nei singoli prodotti IBM: un obbiettivo sicuramente nobile, ma comunque limitato al vincolo della casa costruttrice.
Questo sforzo, comunque, ha dato vita ad una architettura proprietaria che ha preso il nome di S*ystem Network Architetture*, con 7 strati differenti. In particolare, alcuni di questi strati si riferivano alle funzioni propriamente di comunicazione, cioè relativi alla interconnessione di questi sistemi tra di loro, eventualmente attraverso una rete. Gli altri strati, invece, erano relativi alle classiche funzioni di operazione *end–to-end*, che avevano come obbiettivo quello di nascondere le differenze fra i vari sistemi remoti, emulando reciprocamente il comportamento esterno dell’uno verso l’altro.
A questo punto entra in gioco la Comunità Internazionale che, attraverso l’**ISO**, cercò di portare avanti un progetto, decisamente di più ampio respiro, che coprisse 1’eterogeneità di tutti i sistemi, indipendentemente dalle dimensioni, dalle capacità e soprattutto dalla marca del prodotto.
Questo progetto, chiamato **OSI**, da *Open System Interconnection*, cioè interconnessione di sistemi aperti, consentiva a tutti i sistemi di riconoscersi in questa architettura, a patto di configurarsi come ***aperti*** al mondo esterno (l’obbiettivo di ottenere un progetto più globale entrava subito in una prima contraddizione nascosta nei termini, visto che un sistema era aperto soltanto se ottemperava agli standard OSI, cioè conforme OSI).
L’ISO è un ente di standardizzazione che è nato negli stessi anni in cui è nato un’unione di organismi di standardizzazione, messi su dalle amministrazioni postali e dagli operatori pubblici di rete, chiamato ITU (*International Telecomunication Union*); fino a qualche anno fa, fra i nomi di questi organismi di standardizzazione di tale unione ricordiamo il CCITT, per problemi di trasmissione telegrafica, telefonica e di trasmissione dati, ed il CCIR per la radio propagazione.
In realtà, dal 1993, questi organismi hanno cambiato nome e, ricollegandosi al nome primigenio dell’ITU, il CCITT é diventato ITU-T ed il CCIR è diventato ITU-R.

Oltre ai gestori (dei servizi e delle reti pubbliche), in questa comunità, dobbiamo considerare anche l'utenza e i costruttori, i quali sviluppano i prodotti da immettere nel mercato e che, fra tutti, sono forse quelli più di peso.
La convergenza di queste tre parti del mercato ha portato alla federazione di diversi organismi di standardizzazione in ambito nazionale, che a loro volta erano collegati dall'organismo internazionale **ISO** (*International Standard Organizzation*).
Tanto per avere un'idea, gli organismi nazionali afferenti all'ISO sono
per l'Italia l'**UNI**,
per la Francia la **FNOR**,
per l'Inghilterra la **PSI**,
per l'America 1' **ANSI** (*American National Standard Istitute*),
e così via.
L'ISO poi si è inoltre suddiviso in tanti *Technical Committes* (Comitati Tecnici), ciascuno di questi suddiviso in *Work Groups* (Gruppi di lavoro), e così via.
Uno dei *Technical Committes* dell'ISO ha tirato fuori uno standard che avesse proprio come architettura a strati di riferimento l'ambiente OSI, con tanti standard relativi ai singoli protocolli e ai singoli servizi afferenti agli strati di tale architettura.

L'architettura OSI si occupa dello scambio delle informazioni fra sistemi aperti e non del loro funzionamento interno. Infatti, come visto fino ad ora, l'importante, ai fini della cooperazione e della comunicazione, non è come vengono implementate le entità afferenti ai singoli protocolli dei livelli del sistema, né tantomeno come due moduli appartenenti allo stesso sistema si scambiano le primitive di servizio, ma bensì come si comporta un sistema in termini di sequenze ordinate e coordinate di messaggi da inviare. E' proprio questa **la problematica della visibilità esterna di cui si occupa l'OSI**.
Quindi, ciò che si tende a standardizzare, sono i **protocolli** e i **servizi**:

i **protocolli** perché sono essi che determinano in che modo un formato deve essere riconoscibile verso l'esterno,

i **servizi** perché determinano la corretta evoluzione di sequenze di protocollo, permettendo anche la costruzione dei protocolli stessi.

La Fig. 3.1-1 rappresenta la situazione come si può risolvere il mondo di comunicazione e cooperazione OSI fra due sistemi aperti: all'interno di ciascuno sistema aperto si possono avere tutti gli standard e i funzionamenti che si vogliono, relativi ad applicazioni distribuite, tutti i database, i linguaggi di programmazione, i compilatori, che si desiderano, ma non è questa la problematica di cui l'OSI si occupa.
**All'OSI interessa risolvere soltanto il problema della comunicazione esterna**, ossia di come questi moduli scambiano messaggi con i moduli corrispondenti.

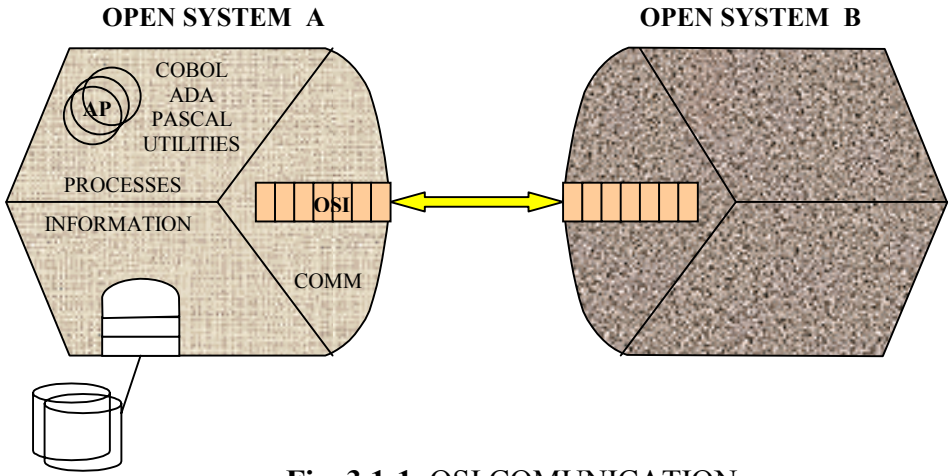


**Fig. 3.1-1**: OSI COMUNICATION

Vediamo di descrivere in dettaglio i **7 livelli** che ha previsto **l'architettura OSI**.
La motivazione ufficiale di questa scelta è che 7 sono stati i livelli ritenuti utili, ma la reale motivazione ha due valenze:

la **prima** di carattere egoistico, visto che si è partiti dai 7 livelli dell'architettura SNA già esistente per poter scardinare il processo di monopolio avviato dalle case costruttrici e

la **seconda** di carattere politico, per non mettere fuori standard, inglobando il pregresso senza sconfessarlo.

**Il principio fondamentale del modello OSI** (in linea con l'SNA, differente dall'OSI in altri aspetti sostanziali) è stato quello di considerare una **netta separazione tra**

**le funzionalità per la cooperazione** (*end-to-end*) **e**

**1e funzionalità per la comunicazione**, cioè l'interconnessione fisica tra un sistema elaborativo e un sistema di telecomunicazione, come può essere una linea trasmissiva o anche un *feeder*.

In Fig. 3.1-2 si può immediatamente notare che, in ciascun sistema, gli strati dell'architettura sono stati suddivisi in **funzioni dipendenti dalla rete** (***Network depended functions***, cioè funzioni di comunicazione) e **funzioni orientate alle applicazioni** (***Application oriented functions***).

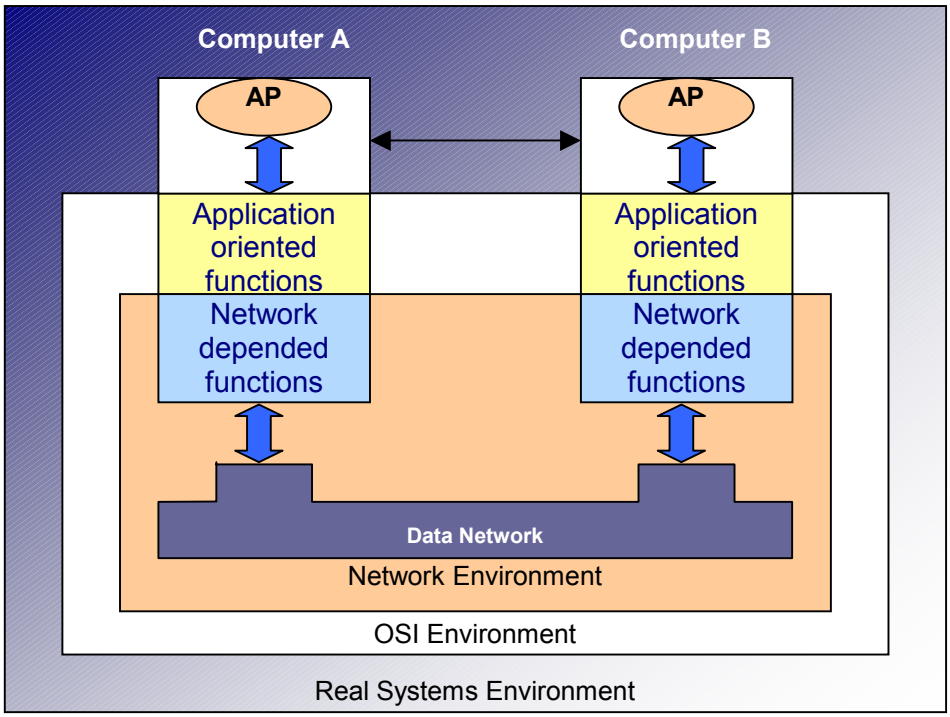


**Fig. 3.1-2**: Suddivisione tra le funzioni di comunicazione e le funzioni orientate alle applicazioni in un'architettura a strati

Questo porta alla definizione di tre **ambienti** operativi, come mostra la Fig. 3.1-2 e 3.1-3(a):

- **il *Real System Environment* (ambiente del sistema reale)** rappresenta i processi utente che necessitano la comunicazione con sistemi remoti. All'interno troviamo
- **l'*OSI Environment* (ambiente OSI),** rappresentato dall'insieme dei 7 strati dell'architettura OSI, si occupa dei protocolli e degli standard per la comunicazione di sistemi aperti; in esso è a sua volta racchiuso
- **il *Network Environment* (ambiente di rete)**, il quale rappresenta l'interfaccia verso la rete di ogni sistema, occupandosi dei protocolli e degli standard relativi ai differenti tipi di sotto rete. Infatti l'ambiente di rete deve, coinvolgendo le funzionalità dei dispositivi di *Network* nei vari nodi della rete stessa (rettangolo tratteggiato che interessa i primi 3 strati), sia ottemperare alle regole protocollari, che effettuare le operazioni di *Front End Processor*.

Il resto dell'***OSI Environment*** si riferisce a tutte quelle funzionalità e a tutti quei protocolli che hanno una interazione logica *end-to-end* con gli elementi di rete.
Nel caso in cui un gestore di rete pubblica vuole inserire dentro la rete un elaboratore che offre dei servizi di utilità generali per l'utenza, è opportuno configurare l'elaboratore come un computer che, oltre alle funzionalità di comunicazione, abbia funzionalità di cooperazione; naturalmente tali funzionalità non attengono al comportamento del sistema di telecomunicazione, ma è il gestore che vuole offrire un servizio diverso all'utenza, all'interno del sistema di telecomunicazione.

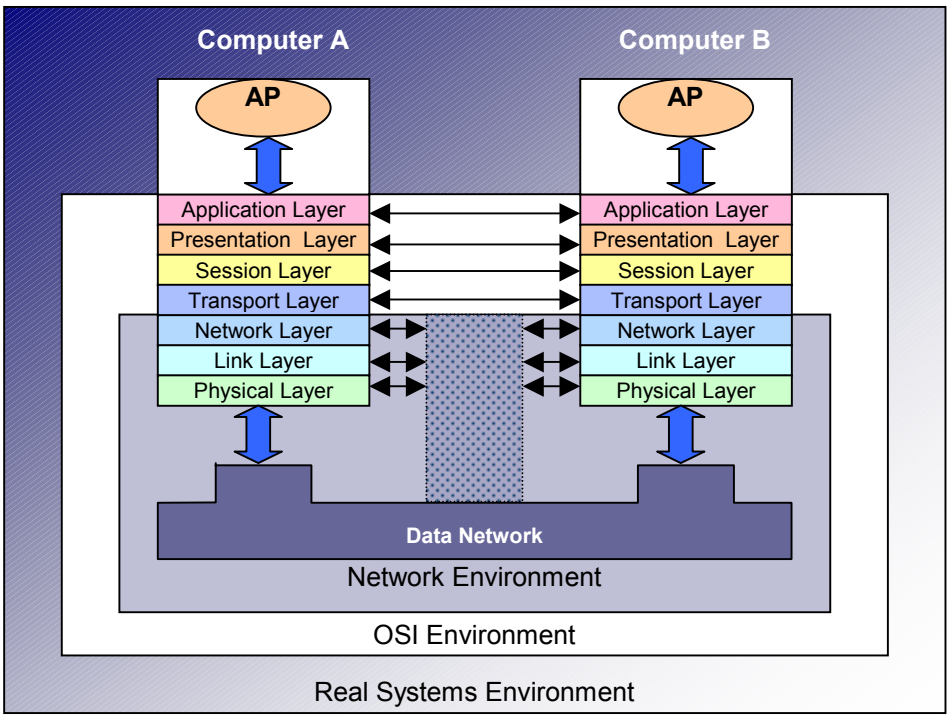


**Fig. 3.1-3(a)**: Architettura OSI

Partendo dal livello più basso, i nomi dei 7 strati, o livelli, dell'architettura OSI sono:

**Livello Fisico** *(Physical Layer)*
**Livello di Linea** *(Data Link Layer - DLL)*
**Livello di Rete** *(Network Layer)*
**Livello di Trasporto** *(Transport Layer)*
**Livello di Sessione** *(Session Layer)*
**Livello di Presentazione** *(Presentation Layer)*
**Livello di Applicazione** *(Application Layer).*

Analizziamo le caratteristiche funzionali di ciascuno di questi livelli, mostrati in Fig. 3.1-3(b).

### *Livello Fisico (Physical Layer)*

Il punto di partenza per poter realizzare un colloquio tra applicazioni distribuite su sistemi eterogenei è il **mezzo di comunicazione** (*Data Comunication Network*), il quale deve essere visto come un canale pronto a trasportare dei segnali elettrici, ottici o radio, a cui bisogna assegnare una valenza di informazione (è facile pensare ai bit).
Per inserire un segnale, visto come sequenza di bit, all'interno di un mezzo trasmissivo è necessario utilizzare un dispositivo d'interfaccia fisica, rappresentata in qualsiasi computer da una porta seriale. Questo significa che si debbono andare a stabilire le caratteristiche meccaniche con cui può

realizzarsi l'interconnessione tra un sistema di calcolo, che ospita l'applicazione, ed il canale che viene utilizzato come mezzo trasmissivo.
Oltre al connettore bisogna anche decidere quali sono le caratteristiche elettriche dei segnali (livello di tensione, forma del fronte d'onda, etc) che dovranno trasportare l'informazione, per poterli distinguere dal segnale rumore, che evidentemente dovrà essere eliminato.

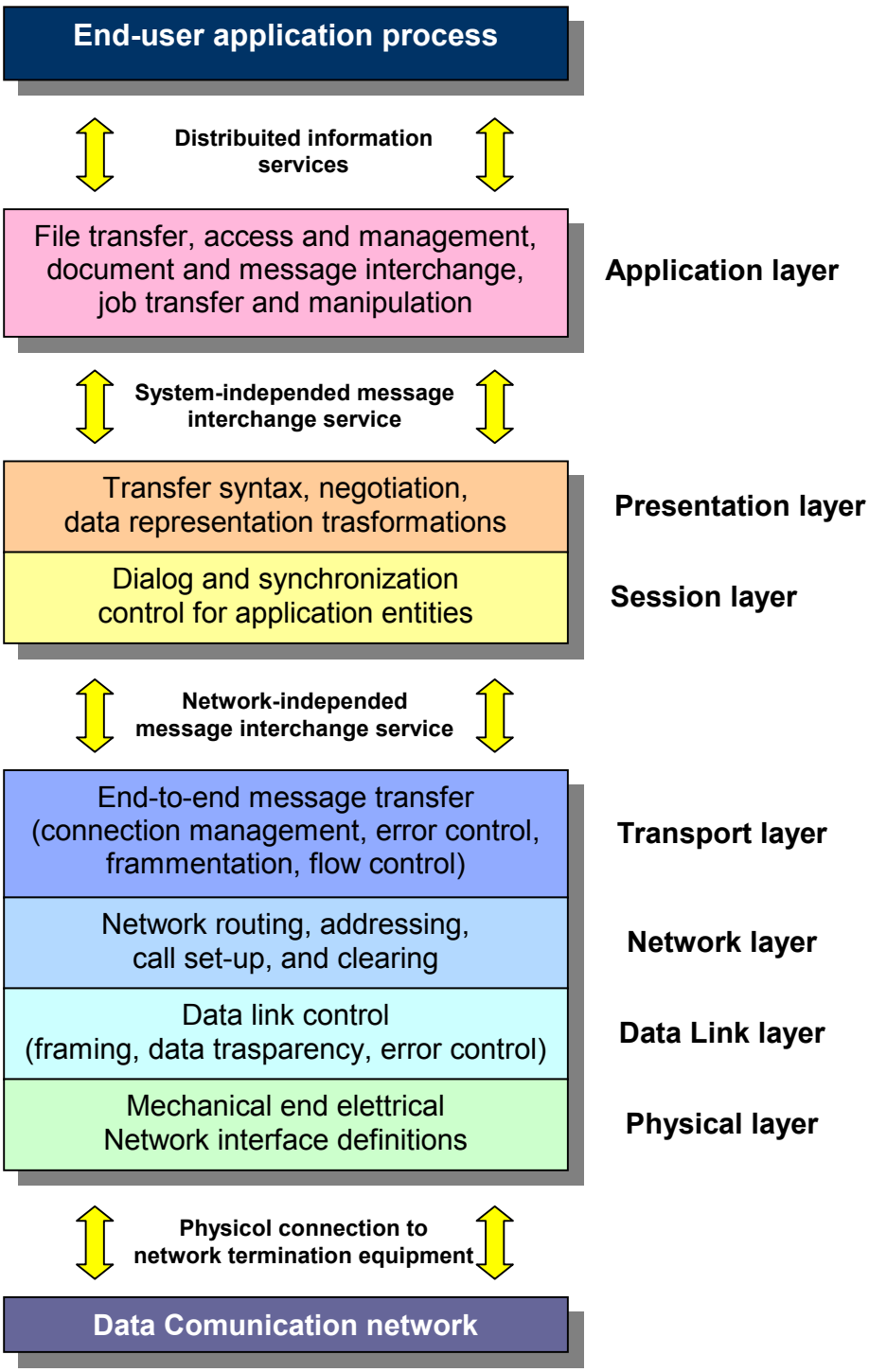


**Fig. 3.1-3(b)**: Caratteristiche funzionali dei 7 livelli dell'architettura OSI

Si tratta di elementi base che occorre necessariamente ritrovare nell'apposito standard e che fanno parte del protocollo a livello fisico, progettato per interconnettere la stazione dati con quel particolare mezzo di comunicazione.

Se si osserva un connettore seriale, tipo quelli presenti in ogni PC, si nota la presenza di tanti pin; in realtà se il collegamento è pin-to-pin, detto anche "*circuito d'interfaccia*" (insieme dei due pin uniti tra di loro), in esso viaggeranno, bidirezionalmente, dei segnali elettrici che hanno la valenza di bit dati.
Tutti gli altri pin servono per trasportare dei segnali elettrici che hanno valenza di bit di controllo, cioè non rappresentanti il flusso informativo, bensì un flusso di controllo che, per esempio, serve ad attivare il circuito di interfaccia stesso, cioè per attivare la connessione a livello fisico.
Oltre a precisare, in modo funzionale, quale è il significato di ciascun *circuito di interfaccia* (orientato al trasporto di bit dati o di controllo), si dovranno anche indicare delle procedure che interpretino la valenza dei segnali elettrici a secondo dei *circuiti di interfaccia* che attraversano.
Ecco la necessità di avere, anche a livello fisico, un protocollo inteso come insieme di regole sintattiche. Se il messaggio è il singolo bit 0 e 1, non è possibile pensare di inserire un campo di controllo. Infatti, in questo caso, il campo di controllo è sostituito dall'idea spaziale di *circuito di interfaccia* su cui verrà convogliato un determinato bit; cioè l'identificativo del significato del bit è connesso al *circuito di interfaccia* che viene attraversato; mentre la sequenza dei messaggi che realizzano un protocollo, in questo caso, è sostituita dalla sequenza semantica con cui i vari bit attraversano i vari *circuiti di interfaccia*.
Nel momento in cui si riescono a realizzare tutte queste funzioni, che in Fig. 3.1-3(b) sono indicate come "*Mechanical and electrical network interface definition*", allora si sarà realizzato un servizio di comunicazione di livello fisico che dà all'utente la possibilità di trasferire dei bit da una parte all'altra di un collegamento fisico.

### *Livello di Linea (Link Layer)*

Il servizio fornito dal livello fisico, se pur essenziale, permette solo, in maniera nuda e cruda, di trasferire sequenze di bit da una parte all'altra del collegamento, senza curarsi di risolvere i problemi legati all'interferenza: i bit trasferiti possono essere alterati dalle interferenze presenti in ogni sistema di telecomunicazioni fisico e nell'ambiente che circonda il sistema di comunicazione fisico.
Inoltre il livello fisico permette il trasferimento di sequenze di bit, senza che esso riesca ad attribuirvi un significato: per il livello fisico il messaggio è il singolo bit, mentre, nell'accezione comune quando si parla di messaggio, si intende una unità dati significativa, con una certa logica, costituita da un insieme di bit.
Ecco perché nasce la necessità di utilizzare un altro strato, che realizzi appunto un protocollo di linea.
Per ***protocollo di linea*** s'intende un protocollo che ha come obbiettivo quello di nobilitare il servizio di comunicazione fisico con un valore aggiunto.
Questo valore aggiunto consiste, innanzitutto, nel definire in modo univoco, all'interno della sequenza di bit trasmessa dal livello fisico, dove inizia e dove finisce un messaggio di senso compiuto, per poter effettuare le operazioni che il protocollo intende svolgere sul messaggio.
Dunque una volta stabilito quale è il messaggio, il protocollo di linea deve fornire un insieme di procedure che permettano di capire se tale messaggio è stato inquinato da errori; in tal caso il protocollo deve introdurre degli schemi attraverso cui le due stazioni dati possano ritrasmettere il messaggio e quindi recuperare gli effetti indesiderati dovuti all'errore.
In altre parole questa nobilitazione porta il servizio di trasferimento di bit, ad un livello superiore, trasformandolo in un servizio di trasferimento di messaggi di tipo affidabile sul link: per "**messaggi affidabili**" si intende dei messaggi liberati dagli errori.
Analizziamo adesso come, il livello *Data Link* riesce a realizzare una comunicazione affidabile. Esso spezzetta i dati provenienti dal *Network layer* in delle ***data Frame*** (la cui lunghezza può variare dalle centinaia alle migliaia di byte), li spedisce in maniera sequenziale e controlla le ***acknowledgement frame*** spedite dal ricevente. Il *Link Layer* si deve anche prevedere che la linea di comunicazione potrebbe distruggere o modificare il contenuto delle frame a causa di disturbi elettromagnetici, quindi deve supportare dei meccanismi di ritrasmissione delle frame perse o

distorte. Inoltre potrebbe accadere che al ricevente arrivi più volte la stessa frame se viene persa una acknowledgement frame, quindi bisogna anche prevedere dei meccanismi per il riconoscimento di pacchetti duplicati.
Il livello di linea, oltre al **controllo d'errore**, fornisce tante altre funzionalità altrettanto importanti: una di queste è la **trasparenza dei dati**.
Si supponga di avere la necessità di trasmettere una sequenza di dati attraverso una sequenza di bit a cui, il protocollo, attribuisce un significato di controllo. In questo caso bisogna trovare il modo di mascherare opportunamente tali dati, di trasformarli, e nello stesso tempo permettere che essi possano essere ricostruiti tali e quali com'erano, quando vengono consegnati all'utente destinatario.
Il livello di linea si occupa anche di **politiche di arbitraggio**.
Finora abbiamo parlato della comunicazione tra due stazioni dati, ma nel momento in cui si ha una linea multipunto, oppure, peggio ancora, un mezzo di comunicazione ad *accesso comune* (come quello delle reti locali), bisogna prevedere delle politiche di arbitraggio per evitare che si presenti il caso in cui tutte le stazioni tentino insieme di trasmettere. Magari si può anche ammettere che tutte le stazioni tentino di trasmettere insieme, ma nel momento in cui questo accade, bisogna trovare delle strategie e degli algoritmi attraverso cui le stazioni riconoscano questo evento e ordinatamente cominciano a trasmettere una per volta. In particolare questo compito è svolto da un sotto livello del DLL (Data Link Layer) detto ***MAC* (*Medium Access Control*)**.
Per impedire, all'interno della rete, che un nodo molto veloce inondi di frame un nodo più lento, nel *Link Layer* deve essere inserita la funzionalità del ***flow control*** (controllo di flusso), in modo tale da evitare di esaurire i buffer in cui memorizzare i dati in ingresso (con conseguente perdita di frame ricevute successivamente).

### *Livello di Rete (Network Layer)*

Non abbiamo ancora considerato quell'insieme di funzionalità che permettono, ad esempio, ad un utente di una rete telefonica o di una rete Internet, di segnalare un indirizzo fisico o simbolico di rete, per riuscire a connettersi con un qualsiasi utente collegato a questa rete, senza preoccuparsi di tutto quello che è l'insieme delle funzioni di instradamento e di commutazione che permettono alla rete di creare un canale di transito *end-to-end* tra i due utenti interessati. Questo importante compito che svolge il livello di rete è detto ***routing***: ovviamente in semplici reti di tipo ***broadcast*** (reti in cui tutti i nodi collegati ricevono il dato trasmesso, e solo il destinatario del messaggio processa il messaggio ricevuto) questa funzionalità non è presente.
Se inoltre si considera che non tutte le reti utilizzano lo stesso protocollo di linea, si capisce che è necessario trovare un comune denominatore, cioè un protocollo che permetta di presentare i dati nella *forma* (ad esempio la lunghezza dei messaggi) che la singola rete si aspetta, creando un servizio che sia indipendente, per l'utente e per le stazioni *end-to-end*, dalle caratteristiche della singola rete che viene utilizzata. Tale funzionalità è detta ***internetworking***.
Di tutte queste funzioni di **instradamento** (*network routing*), **indirizzamento** (*addressing*), **internetworking** e delle operazioni di **apertura e di chiusura** (***set-up***) di un circuito di una connessione di rete *end-to-end*, se ne occupa il **livello di rete** (*Network Layer*), come mostra la Fig. 3.1-3. In altre parole, il livello di rete permette, attraverso opportuni protocolli, il colloquio due-a-due tra una stazione d'utente ed un elemento di rete (ad esempio un nodo), tra un elemento di rete ed un altro elemento di rete, tra quest'ultimo elemento di rete ed un altro elemento di rete, …, ed infine tra un elemento di rete ed una stazione dati: così la visibilità che viene data all'utente è quella di un canale unico, un collegamento di rete, attraverso cui le due stazioni d'utente possono parlare.

### *Livello di Trasporto (Transport Layer)*

In una rete di tipo magliata, se per il protocollo che si utilizza, ogni messaggio viaggia secondo una strada (*rout*), che può essere differente da messaggio a messaggio, nasce il rischio che un messaggio arrivi prima di un altro messaggio emesso precedentemente.

Si ha dunque bisogno, ai fini di una ricostruzione esatta dei dati d'utente, a livello *end-to-end*, di rimuovere tutte le cause di possibile errore e di ricostruire in ricezione il flusso originario (qualcosa di simile al protocollo di *Data Link* visto per il singolo collegamento dati).
E' il **Livello di Trasporto** (*Transport Layer*) che si occupa del **trasferimento affidabile di messaggi *end-to-end***, con relativa **gestione di connessione**, **controllo di errore** e di **controllo di flusso** (il ***flow control*** deve intervenire nel caso in cui un utente invia dati in rete troppo velocemente rispetto ad un ricevitore o rispetto ad un lento smaltimento dei dati da parte della rete, provocato da temporanei problemi di traffico o di guasto), ed eventualmente anche della **frammentazione**.
Quest'ultima operazione è necessaria nel momento in cui gli utenti volessero trasferire in rete messaggi di elevata ampiezza, senza alcuna limitazione: se dunque la rete non può imporre agli utenti l'ampiezza massima del messaggio trasferibile, deve prevedere un meccanismo che frammenti opportunamente i messaggi in trasmissione e li ricomponga in ricezione prima di consegnarli all'utente destinatario.
Per il fatto che è un protocollo *end-to-end*, e quindi non vede elementi di rete, in alcuni libri di testo il *Transport Layer* viene visto come un protocollo di cooperazione.
Altri testi, invece, sono più del parere che esso risolve ancora delle funzioni di comunicazione e che, come mostra la Fig. 3.1-3, sia 1'ultimo degli strati relativi alle funzionalità di comunicazione; per cui esso, insieme ai primi 3 strati, realizza un servizio di scambio di messaggi realmente *network independed*, a *livello end-to-end*.
Per concludere, il *livello di trasporto*, in base alla **qualità del servizio** (***QOS quality of service***) messa a disposizione dal *Network Layer*, può presentare *5 classi di servizio (classes of services)*:
*la classe 0* provvede alle funzionalità base per creare una connessione e trasferire dati, mentre
*la classe 4* provvede una connessione affidabile e prevede delle procedure per il *flow control*.

### *Livello di Sessione (Session Layer)*

Per risolvere il problema della cooperazione tra sistemi eterogenei non basta il *Transport Layer*, in quanto ci sono da risolvere altre problematiche.
Ad esempio se consideriamo la comunicazione tra 2 persone, può avvenire che ciascuno dei due interlocutori parli senza ascoltare l'altro.
Per questo motivo, nella maggior parte delle applicazioni, sono necessari dei meccanismi che trasformino un collegamento *full-duplex* (che garantisce il principio di trasporto bidirezionale di messaggi in qualsiasi momento) in un **servizio di tipo *two way alternate***, cioè alternato a due: in pratica le entità devono comunicare una per volta rispettando il proprio turno.
Per far questo, i calcolatori di una rete di telecomunicazioni devono passarsi un *token* (gettone), e soltanto chi ha il *token* in mano può trasmettere dei dati, secondo una evoluzione coerente. Devono inoltre essere previsti dei meccanismi di richiesta del *token*.
Questo concetto è di estrema importanza. Si pensi, infatti, al caso in cui un calcolatore sta trasmettendo dei file su un sistema di database distribuito: bisogna essere certi, prima di permettere che qualche altro calcolatore cominci a trasmettere, che il primo calcolatore abbia terminato di mandare i suoi dati, altrimenti non si avrà alcuna garanzia di consistenza.
Un altro problema importante e l'esigenza della **sincronizzazione** del dialogo.
Si consideri un colloquio tra un terminale *page-mode* ed un terminale *scroll-mode*.
Il *page-mode* è un terminale che avanza a pagine, mentre lo *scroll-mode* è il tipico terminale in cui man mano che si scrive, c'è uno *scrolling*, cioè scompare la prima linea in alto e ne appare un'altra in basso.
In particolare si supponga che il terminale *page-mode* invia una pagina di dati per volta allo *scroll-mode*, e si supponga che ad un certo punto si verifichi un errore, per cui nasce il problema di effettuare la ritrasmissione.
Se lo *scroll-mode* aveva ricevuto correttamente i dati fino a metà della prima pagina, non ha senso che il *page-mode* ritrasmetta i dati relativi all'intera pagina, in quanto così sul terminale *scroll-mode*

verrà ripetuta la parte di testo corretta; avviene, in pratica, una disincronizzazione e quindi non si ha più l'interezza della pagina.
In questi casi è estremamente importante dare la possibilità alle applicazioni (e non al meccanismo di comunicazione) di scegliere e di fissare autonomamente dei punti di sincronizzazione, all'interno delle cosiddette "attività o unità dialogo".
In questo modo nel momento in cui si dovesse verificare un errore (caso di crash), la ritrasmissione ricomincerebbe dall'ultimo punto di sincronizzazione.
Tali funzionalità sono esplicate dallo strato chiamato *Session Layer*, visto che organizza le attività in sessioni con un ben preciso significato, almeno dal punto di vista dell'applicazione.

### *Livello di Presentazione (Presentation Layer)*

Inoltre quando si considerano sistemi di calcolo che lavorano con delle strutture dati, che possono essere legate, da una parte alle caratteristiche del sistema e dall'altra all'applicazione, è importante che il sistema trasmittente faccia riconoscere all'applicazione con cui si sta comunicando, il formato della propria struttura dati: cioè se due calcolatori "non parlano la stessa lingua", i due sistemi non riusciranno mai a comunicare. E' proprio il *Presentation Layer* che si occupa della **sintassi** e della **semantica** delle informazioni da trasferire.
Prima dell'attuale unifomizzazione, ai tempi in cui nasceva l'OSI, c'erano ancora PC che lavoravano con sequenze di 8 bit, altri con sequenze di 16 bit, fino a sequenze strane di 36 bit.
In una situazione del genere era necessario effettuare una presentazione iniziale dei dati, affinché ogni sistema potesse spiegare che tipo di sequenza stava inviando, in modo da consentire al sistema destinatario di ricomporla secondo il formato originario.
Ma al di là di questo problema, oggi superato, se si vuole mandare una struttura dati complessa, quale può essere un record di tipo multimediale, che contiene il formato testo, il formato immagine e il formato suono, bisogna, per consegnare l'informazione nella sua interezza, fare precedere tale struttura dati da una intestazione in cui si indichi, al proprio corrispondente, quanti sono i campi che compongono la struttura dati e da quanti bit è composto ciascun campo.
In questo modo il sistema che riceve questi dati, leggendo l'intestazione, sa come e dove deve riconoscere le informazioni che gli sono pervenute.
Per superare l'ostacolo della diversità di strutture dati, prima di trasmetterle bisogna che siano filtrate attraverso una **sintassi di riferimento**, che sia negoziata e concordata da entrambi i sistemi di comunicazione: così, quando le strutture dati, opportunamente codificate, arrivano a destinazione, verranno prima decodificate, secondo la sintassi di trasferimento, e dopo rielaborate secondo la sintassi locale, relativa alle strutture dati dell'ambiente locale.
Il *Presentation Layer* si occupa appunto di fornire questa sintassi di trasferimento ad entrambi i sistemi remoti.
Col passare del tempo il *Session Layer* ed il *Presentation Layer* hanno perso via via di importanza, in quanto si è deciso di utilizzare un'unica sintassi di trasferimento standard. Dal momento in cui si è implementata un'unica sintassi di trasferimento, con le regole di codifica basate sulla notazione cosiddetta ASNl, Il *Presentation Layer* non è comunque scomparso, in quanto si deve prevedere in ciascun sistema aperto un codificatore-decodificatore basato sull'ASNI.
Un'altra funzione svolta dal livello di presentazione è il **criptaggio** dei dati.
Passiamo all'ultimo livello dell'architettura OSI.

### *Livello di Applicazione (Application Layer).*

Se due sistemi, con l'architettura fin qui costruita, riescono a comunicare ed a capirsi, nel momento in cui uno è interessato ad applicazioni di un certo tipo, mentre l'altro è interessato ad applicazioni di tutt'altro tipo, non ha senso la cooperazione e la comunicazione, in quanto ognuno di loro sarà volto a risolvere problematiche diverse. In pratica, affinché un dialogo tra 2 entità sia costruttivo, è necessario un **universo del discorso comune**.

Per risolvere tale questione si sono individuate delle applicazioni universali, come il *File Transfer*, l'accesso a database, ecc. (che le applicazioni proprietarie, prima o poi, finiscono con l'utilizzare), in cui si sono inseriti quei protocolli di cooperazione che permettono agli utenti di conoscere i propri universi, in modo da intendersi su quali saranno i dati e le operazioni con cui lavoreranno, al fine di trasferire i dati stessi.
L'*Application Layer*, essendo il livello più alto dell'architettura OSI, deve, quindi, offrire ai programmi utente un insieme di funzionalità che, a causa dell'enorme diversità, sono raggruppate in entità dette **ASE** (***Application Service Element***), cioè elemento di servizio applicativo). Infatti, come mostra la Fig. 3.1-4, a differenza degli altri strati, l'*Application Layer* è composto da tanti *Application Service Element*, che mettono a disposizione delle particolari funzionalità utilizzate dall'applicazione vera e propria, per accedere alle risorse remote.
Il vantaggio di tale approccio è che i programmatori di applicazioni possono invocare direttamente il servizio fornito dell'ASE, senza avere la necessità di crearsi proprie routines per risolvere i problemi più comuni.

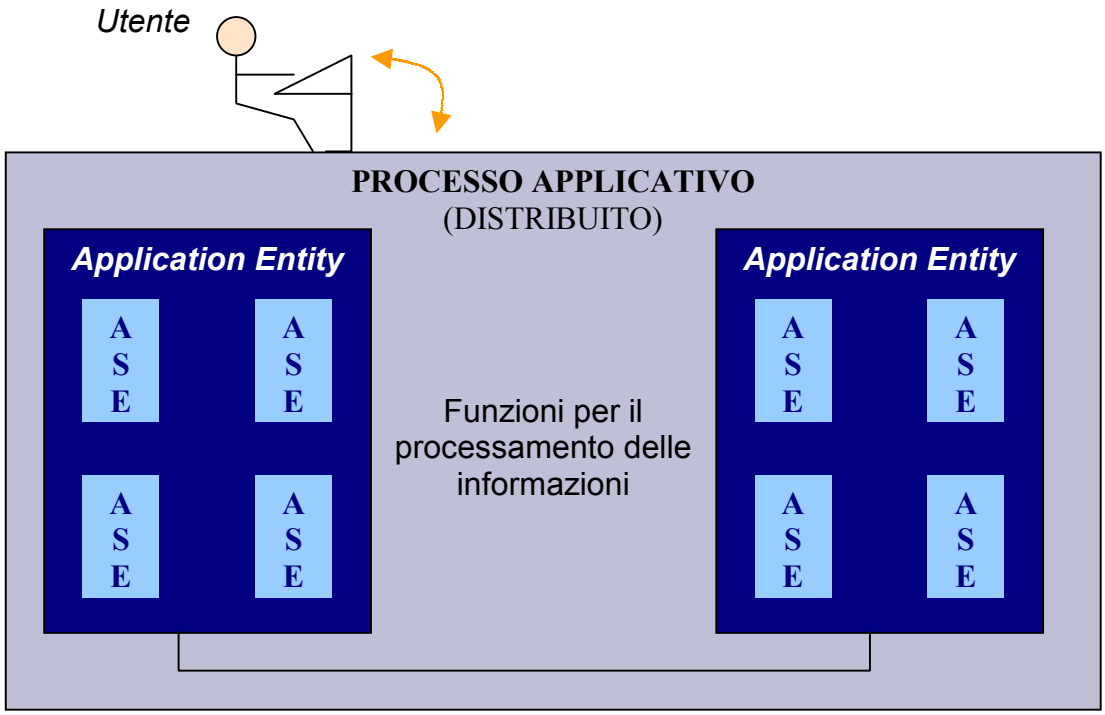


**Fig. 3.1-4**: *Application Layer* composto da più ASE (*Application Service Element*)

Quello che ci si aspetta da una rete di calcolatori in un sistema telematico è la possibilità di condividere delle risorse, cioè di fare in modo che ciascun utente abbia a disposizione un ambiente globale (sistema globale) in cui poter lavorare, esattamente come lavorerebbe nel suo ambiente (sistema locale). Ovviamente, a differenza del sistema locale, il sistema telematico gli consente di lavorare anche su risorse remote e distribuite.
Da questo punto di vista l'interazione sarà di tipo *client-server*, per cui ogni processo applicativo (o anche l'utente finale), lavora con delle risorse che sono i server, visti come dei moduli che permettono l'accesso alle risorse vere e proprie.
Questo significa che un utente che si trova in un ambiente locale, per operare secondo il modello *client-server*, deve usare delle interazioni di tipo *Request-Confirm*, che abbiamo già analizzato parlando delle primitive di servizio.
La Fig. 3.1-4 mostra il vero modello di servizio, in cui l'utente accede ad un *Service Provider* globale con cui, in qualche misura, esegue delle primitive di *Request* e *Confirm*.

Si può pensare che la risorsa di cui ha bisogno l'utente sta in una grande scatola nera che è il processo applicativo distribuito: se la risorsa di cui ha bisogno l'utente è contenuta nel sistema dell'utente (sistema locale) o in un sistema remoto, questo a lui non importa.
Se però la risorsa desiderata dovesse trovarsi in un sistema remoto, allora bisogna che gli sia permesso aprire un canale in cui utilizzare risorse hardware e software per accedere alla risorsa remota: **questo canale (finestra sul mondo) è proprio ciò che viene offerto dai 7 livelli dell'architettura OSI, con i relativi protocolli di servizio**.

## *3.2 Il Modello TCP/IP*

La realizzazione degli standard OSI è stata frutto di un lavoro immane durato 10÷15 anni, nell'ambito dell'ISO,. che ha portato alla redazione di una grande quantità di protocolli e di standard.
Oggi, praticamente, di tutto questo lavoro è rimasto poco, ma la cosa importante è che il modello ISO-OSI rappresenta l'unico modello di architettura a strati che possa far capire bene la necessita di progettare i protocolli e servizi secondo un paradigma di riferimento, poco applicativo, ma certamente altamente formativo.
Per il resto, quella che si è fatta strada non è l'architettura OSI, ma l'architettura TCP/IP.

Vediamo come nasce l'**architettura TCP/IP**.
Mentre i comitati dell'ISO discutevano su come era meglio realizzare i protocolli dell'OSI, c'erano molti ricercatori che si preoccupavano di produrre sistemi che funzionassero praticamente, verso l'obbiettivo **I*nternet***, cioè l'obbiettivo di **interconnettere** le tante reti esistenti sul mercato: reti ARPANET (progetto sponsorizzato dal *DoD* (*Department of Defense*) e sviluppato da varie università americane), reti locali, le *packet* radio, reti satellite, ecc.
Grazie al contributo di questi ricercatori si è giunto alla progettazione di un *protocollo di rete*, (associato al *Network Layer*, ovvero al *Internet Layer* in questo modello di riferimento) che è stato chiamato **IP (*Interconnection Protocol*)**.
Successivamente, a questo, si è ritenuto opportuno aggiungere due *protocolli di trasporto* (mentre l'ISO pensava a 5 protocolli di trasporto a fronte di classi differenziate di servizio) di cui uno era un protocollo *orientato alla connessione* e l'altro era un protocollo *non orientato alla connessione*.
La progettazione del **protocollo UDP non orientato alla connessione** è stata molto semplice, visto che si poggiava sul protocollo **IP**, il quale è anch'esso un *protocollo orientato alla connessione* in cui non occorre, preliminarmente, aprire una connessione di rete per permettere il trasferimento dei messaggi nella rete stessa.
Il **protocollo TCP orientato alla connessione** permette, invece, la visibilità di un collegamento *end-to-end* all'utente.

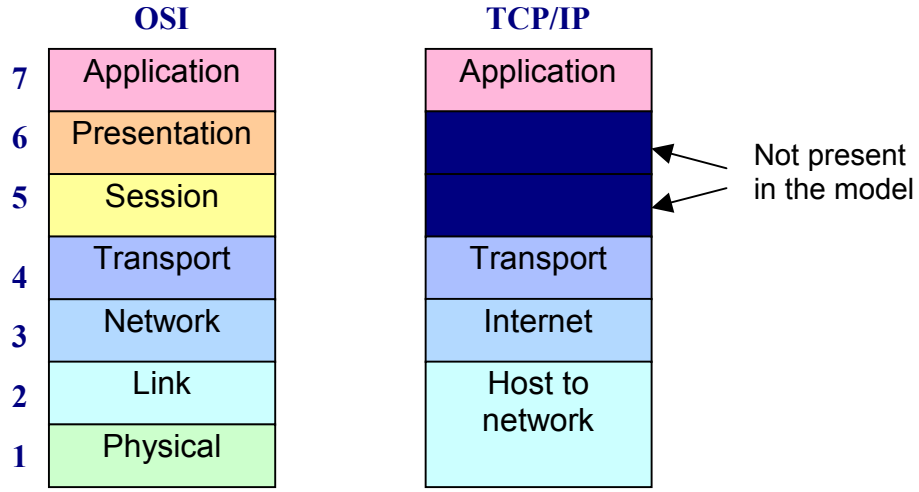


**Fig. 3.2-1**: differenze tra l'architettura OSI e l'architettura TCP/IP.

Fondamentalmente i protocolli **IP** e **TCP** sono stati i due cardini di tutto il lavoro: infatti l'architettura che ne nacque si chiamò **TCP-IP**.
Inoltre questi protocolli raggiunsero, ben presto, un alto grado di efficienza e di popolarità visto che erano anche disponibili a costo nullo.

Come si vede dalla Fig. 3.2-1 e 3.2-2, non esistono più i livelli di *Session* e *Presentation* dell'architettura OSI, mentre i livelli di *Data Link* e *Physical* sono stati unificati nel livello *Host-to-Network*.

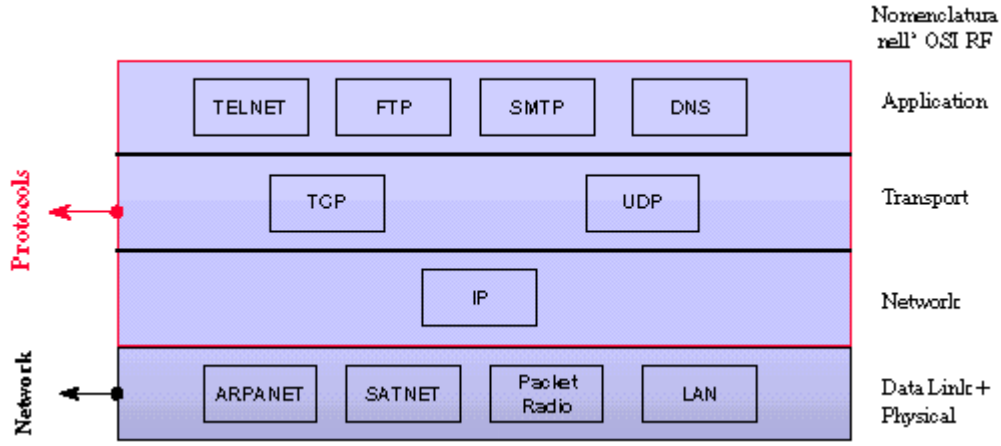


**Fig. 3.2-2**: Protocolli e reti presenti nel primo modello dell'architettura TCP/IP

Vediamo di descrivere le funzioni svolte da ciascuno dei *livelli* del modello di riferimento TCP/IP.

### *Host-to-Network Layer*

Questo livello, che sta sotto l'*Internet Layer*, non è in realtà specificato rigorosamente dal modello di riferimento TCP/IP, in quanto il protocollo che si utilizza viene definito e varia da *host* a *host* e da rete a rete.

### *Internet Layer*

L'obiettivo di interconnettere più reti e di ottenere un alto grado di robustezza, intesa come la capacità di funzionamento anche in caso di guasto di nodi intermedi, ha portato alla scelta di un *livello di rete* a *commutazione di pacchetto* di tipo *connectionless*.
Infatti tale livello di rete *Internet Layer* (corrispondente al *Network Layer* dell'architettura OSI), attraverso il protocollo **IP (*Internet Protocol*)**, effettua la spedizione di pacchetti in ogni rete e in un qualsiasi nodo di destinazione.
Ma visto che la comunicazione è di tipo *connectionless* l'ordine di arrivo dei pacchetti potrebbe differire dall'ordine con cui sono stati spediti, per cui, sarà compito del livello superiore *Transport* ricomporre correttamente l'intero messaggio.

### *Transport Layer*

Le funzionalità di questo livello sono svolte dai due protocolli **TCP (*Transmission Control Protocol*)** e **UDP (*User Datagram Protocol*)** sopra citati.

- Il **protocollo TCP** è basato su una comunicazione affidabile a circuito virtuale; in pratica il TCP permette ad un utente di inviare dati in modo affidabile, ricostruendo a destinazione, con la corretta sequenza, tutti i pacchetti che compongono l'intero messaggio inviato. Inoltre il TCP effettua il *controllo di flusso* (*flow control*) per evitare che una sorgente di messaggi veloce saturi il buffer di un ricevente lento.

- Il **protocollo UDP** è invece basato su una comunicazione a datagramma inaffidabile: non effettua il *flow control* e il riassemblaggio dei pacchetti. Questo giustifica il perché è utilizzato in applicazioni di tipo *client-server*, che necessitano del passaggio di piccole quantità di dati, e in applicazioni in cui è preferita la velocità all'accuratezza delle informazioni (ad esempio quando vengono trasmesse sequenze audio o video).

### *Application Layer*

Come si vede dalla Fig. 3.2-1 e dalla Fig. 3.2-2, subito sopra al *livello di trasporto* vi è quello di *applicazione*. In esso si sono inseriti alcuni applicativi ad alto livello che permettono di risolvere problemi concreti.

Uno di questi applicativi è il **TELNET** (*terminale virtuale*) che eseguiva il *Login* remoto, cioè l'operazione per cui un utente, in qualunque luogo della Terra si trovasse, purché avesse a disposizione un sistema connesso alla rete *internet*, fornendo la propria *Login* e la propria *password*, poteva avere l'accesso alla rete e quindi lavorare dal terminale in cui si trovava come se stesse lavorando sul proprio PC. In questo modo ogni utente poteva lavorare su una macchina remota.

Altri applicativi inseriti da subito sono stati:

l'**FTP** (*File Transfer Protocol*), che permette di scambiare o di copiare file presenti nei *server* della rete,

l'**SMTP** (*Simple Mail Transfer Protocoll*) che serve per gestire i servizi di posta elettronica e

il **DNS** (*Domain Name Server*) che serve a fornire il cosiddetto servizio di directory (che sta per "elenco telefonico"), cioè un servizio che permette all'utente, fornendo un indirizzo simbolico (per esempio il proprio indirizzo di posta elettronica), di avere automaticamente riconsegnato indietro l'indirizzo fisico della rete, al quale corrisponde la porta del suo PC.

Oltre a questi servizi forniti dall'architettura TCP/IP, se ne sono sviluppati altri grazie all'evoluzione che tale architettura ha subito in questi ultimi anni. Uno tra questi è

l'**HTTP**, un protocollo usato per caricare le pagine del *World Wide Web*.

### *Confronto tra l'architettura OSI e TCP/IP*

Il motivo del successo del TCP/IP a spese dell'architettura OSI, sta proprio nell'obiettivo, da parte dell'architettura OSI, di pretendere di risolvere l'universalità dei problemi di rete, attraverso una serie di moduli protocollari, che è risultato troppo ambizioso e pesante (si pensi a quale deve essere l'*overhead* di un ambiente *Connect Oriented* esteso a tutti gli strati).
Circa 20 anni fa, quando il lavoro sul *Transfer Protocol* dell'OSI era stato completato, fu fatta una prima prova mettendo su una rete ETHERNET a 10 Mbit/s una implementazione del *Transfer Protocol* fatto dall'IBM (non una ditta qualunque); si è visto che il ***throughput*** (massimo volume di traffico che si riesce a trasferire nell'unità di tempo) da l0 Mbit/s nominali di ETHERNET (che poi erano 10 Mbit/s di fatto perché, utilizzando due sole stazioni che comunicano tra di loro, di collisioni non ce ne possono essere) si è arrivati ad 1,75 Mbit/s, cioè c'è stata una perdita secca di *throughput* soltanto per l'implementazione del *Transfer Protocol*.
In effetti dopo sono state apportate delle modifiche che hanno consentito dei miglioramenti, ma questo primo esperimento che abbiamo ricordato è un chiaro indicatore di che cosa comporta realizzare una architettura OSI.
Al contrario, una architettura molto snella, come l'TCP/IP, funziona chiaramente meglio.
Inoltre la formulazione dell'obbiettivo iniziale dell'OSI di far sparire tutte le architetture proprietarie, è stata una politica infelice , in quanto si andava contro gli interessi delle grandi case costruttrici, come l'IBM, la Digital, ecc.

L'OSI, accortasi dell'insofferenza delle aziende produttrici, ha cercato subito di correggere il tiro ribadendo che non si intendeva rendere inutili tutte le architetture proprietarie, ma si intendeva, piuttosto, creare un elemento unificatore, nel senso che i sistemi OSI dovevano essere destinati a diventare sistemi *gateway* (*gateway* letteralmente significa "porta di attraversamento", ma lo si deve intendere come l'anello per tutte le altre realtà del mercato).
In altre parole, date l'architettura A e B, l'OSI non intendeva eliminarle, ma intendeva aggregare all'architettura A l'architettura OSI, in modo tale che essa fosse visibile come OSI dall'esterno e viceversa. Per cui ogni casa costruttrice, con la propria architettura, poteva fare tutto ciò che voleva, ma nel momento in cui si doveva interfacciare verso qualche altra architettura, lo doveva fare tramite OSI.
Esiste, comunque, una regola pratica, di cui l'OSI non ha tenuto conto, quando si lavora su aspetti tecnici: nel processo lavorativo, c'è una prima fase di grosso sforzo di ricerca (negli ambienti universitari, negli ambienti di produzione) per trovare la soluzione tecnica migliore; dopo segue una fase di assestamento e solo, successivamente, davanti alle possibili soluzioni raggiunte, le case costruttrici decidono di investire su quella che sembra la soluzione tecnica migliore.
L'OSI è arrivato troppo tardi in questo processo lavorativo, cioè quando l'investimento si era già fatto, quindi è chiaro che è stato messo fuori mercato.
E' per questo che l'OSI rimane uno standard *de iure*, ma non uno standard *de facto*, mentre il TCP/IP è una architettura ormai consolidata il cui organismo di standardizzazione è il cosiddetto **IETF** (*Internet Engineerinig Task Force*), a cui collaborano, in modo volontario, tutti quelli che si occupano di *Internet*.
Il metodo con cui tale organismo procede nella ricerca, non è quello di emanare delle proposte di standard che devono essere votate, bensì ognuno, nel proprio ambiente accademico e di sviluppo, non appena trova una buona soluzione, prima la implementa e poi mette in linea (*on line*) su Internet il cosiddetto RFC (*Request For Comment*).
L'RFC è un documento, che rimanda ad una implementazione software che chiunque si può scaricare dalla rete, mettere sul proprio sistema e verificarlo. A questo punto, in modo estremamente trasparente, arrivano i commenti, e se l'implementazione sembra efficiente, diventa, di fatto, uno standard dell'IETF.
In pratica il TCP/IP è una bella collezione di protocolli, che non può essere chiamata "architettura", visto che non ha niente di un modello di riferimento.
Infatti, anche se il TCP/IP e l'OSI Reference Model hanno molte cose in comune, bisogna comunque mettere in evidenza che mentre nell'OSI i protocolli sono stati realizzati successivamente al modello di riferimento, nel TCP/IP prima sono stati realizzati i protocolli e da questi è nato il modello di riferimento.
Come conseguenza, nell'OSI vengono perfettamente definiti e differenziati i concetti di servizio, interfaccia e protocollo, mentre nel TCP/IP la separazione non è così netta. L'implementazione dei protocolli OSI può essere modificata in base all'evoluzione tecnologica.
Ad esempio non è possibile spiegare il concetto di *layering* (stratificazione) attraverso il TCP/IP. Infatti mentre nell'architettura OSI è stato possibile definire con molta chiarezza il concetto di interfaccia, intesa come la superficie reale di separazione verso cui viene offerto un *servizio di comunicazione*, nell'architettura TCP/IP non è possibile parlare di una interfaccia che divide il *livello di rete* (caratterizzato dal protocollo **IP**) dall'insieme di reti reali (che viene chiamato *Host to Network*); si può dire semplicemente che l'IP vedrà, in un sistema locale, tutto il resto del mondo, attraverso una certa interfaccia.
Dunque nel TCP/TP, non c'è nessun *layering* e quindi nessun aiuto alla progettazione completa e ordinata di protocolli per la rete sottostante, cioè per tutto quello che c'è al di sotto di Internet.
E' per questo motivo che probabilmente, tra una ventina di anni, di Internet non rimarrà molto.
Sicuramente si salveranno i protocolli IP e TCP e magari altri protocolli importanti, ma il metodo che si utilizzerà per produrli, potrebbe portare una sorta di anarchia nella compatibilità tra i diversi sistemi.

Come abbiamo visto, l'OSI prevede sia la comunicazione *connection oriented* che la comunicazione *connectionless* nel *Network Layer* e solamente la comunicazione *connection oriented* nel *Transport Layer*; al contrario, nel TCP/IP è prevista solo la tecnica *connectionless* nel *Network Layer* ed entrambe le tecniche nel *Transport Layer*.
Ma il grosso difetto del modello TCP/IP è che il livello più basso (*host-to-network*) non è un vero e proprio livello. In realtà esso è un'interfaccia tra il *Network Layer* e il *Data Link*, che svolge sia le funzioni del *Physical Layer* che del *Data Link Layer*, i quali hanno compiti completamente diversi ed è quindi assurdo inglobarli in un unico livello.