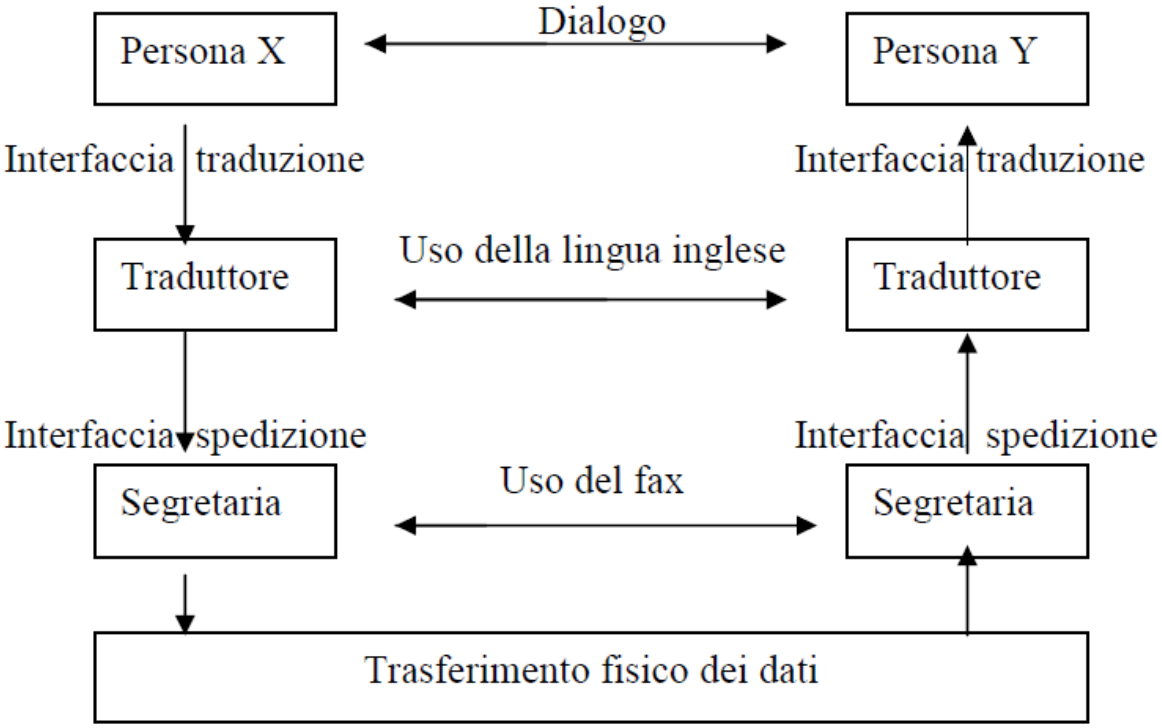
SOFTWARE DI RETE

Software che gestisce la comunicazione tra stazioni.

La comunicazione tra due calcolatori si realizza tramite lo scambio di dati su un canale di comunicazione, esiste quindi un TRASMETTITORE che invia dei dati e un RICEVITORE che riceve i dati sul CANALE prescelto.

Per comprendere meglio i **meccanismi di funzionamento del software di rete** pensiamo alla seguente **analogia** in cui due persone, persona x e persona y, vogliono dialogare tra loro nonostante parlino lingue differenti, rispettivamente cinese e giapponese.
La persona x vuole inviare un messaggio a persona y. Supponiamo che decidano di comunicare via fax e traducendo il messaggio in inglese.
Il messaggio subirà delle trasformazioni:



1) sarà dato in mano ad un traduttore che, tramite le regole di traduzione, lo scriverà in lingua inglese;
2) il messaggio tradotto sarà dato alla segretaria che, utilizzando il fax, farà l'operazione di spedizione
3) il messaggio, così opportunamente trattato, arriverà al mezzo fisico su cui viaggia il fax, ovvero la linea telefonica .

Arrivato a destinazione, il fax subirà il percorso inverso:

1) viene interpretato correttamente dal dispositivo di ricezione, passato alla segretaria che effettua la stampa,

2) la stampa viene passata al traduttore che, utilizzando le regole della lingua inglese, lo traduce e lo passa alla persona Y.

Da questo esempio possiamo osservare che:

- la COMUNICAZIONE è ORGANIZZATA in più LIVELLI O STRATI;
- ogni strato per comunicare con lo strato corrispondente della destinazione (COMUNICAZIONE VIRTUALE), deve stabilire una serie di regole e convenzioni, che possiamo chiamare **protocolli**.
  Le informazioni contenute all'interno dei protocolli possono essere ad esempio:
  - l'indirizzo del mittente
  - l'indirizzo del destinatario
  - la struttura del messaggio, ad esempio l'inizio e la fine
  - la natura del messaggio, ad esempio la lingua
  - la lunghezza del messaggio

  ecc…
- ogni strato mette a disposizione per lo strato superione dei **servizi** necessari alla trasmissione del messaggio, ad esempio il traduttore si mette a disposizione della persona x di svolgere il servizio di traduzione.

La stessa cosa avviene nella comunicazione tra due stazioni:

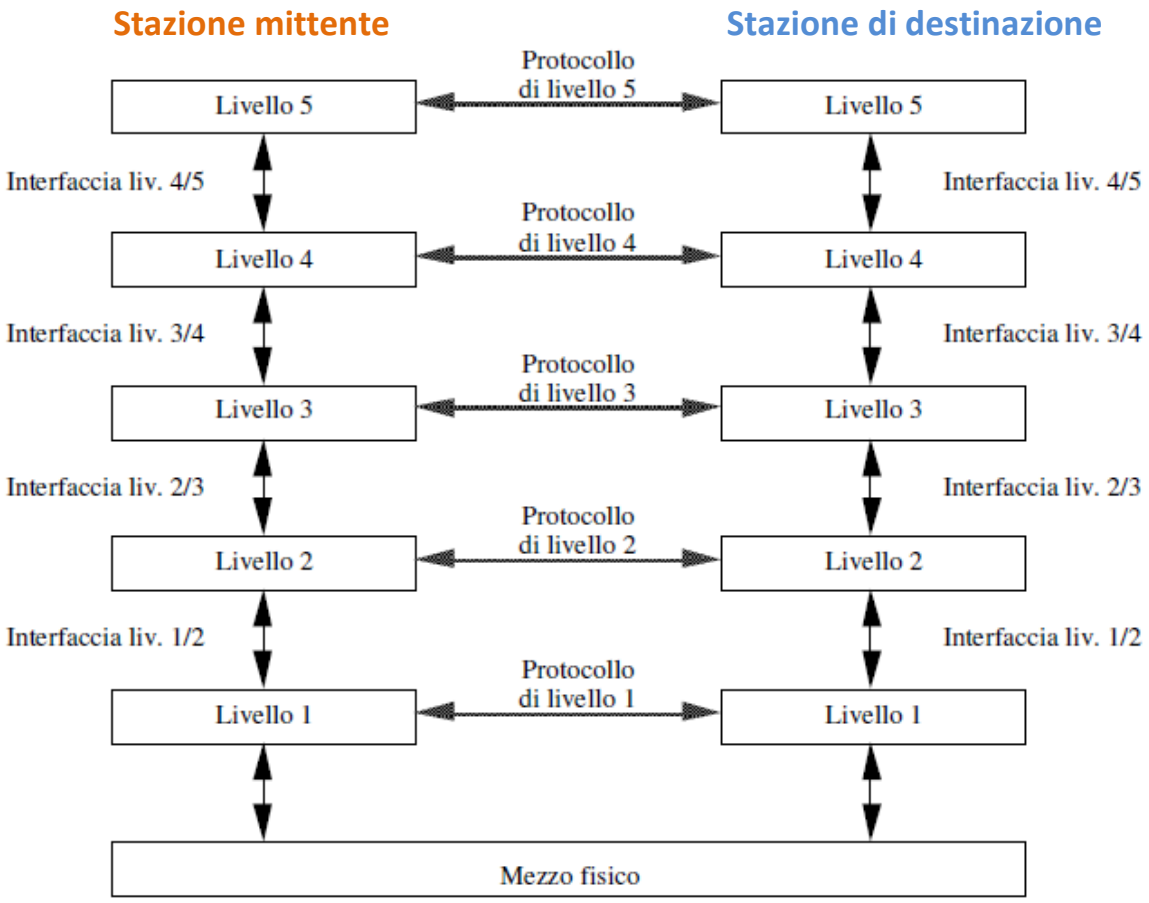
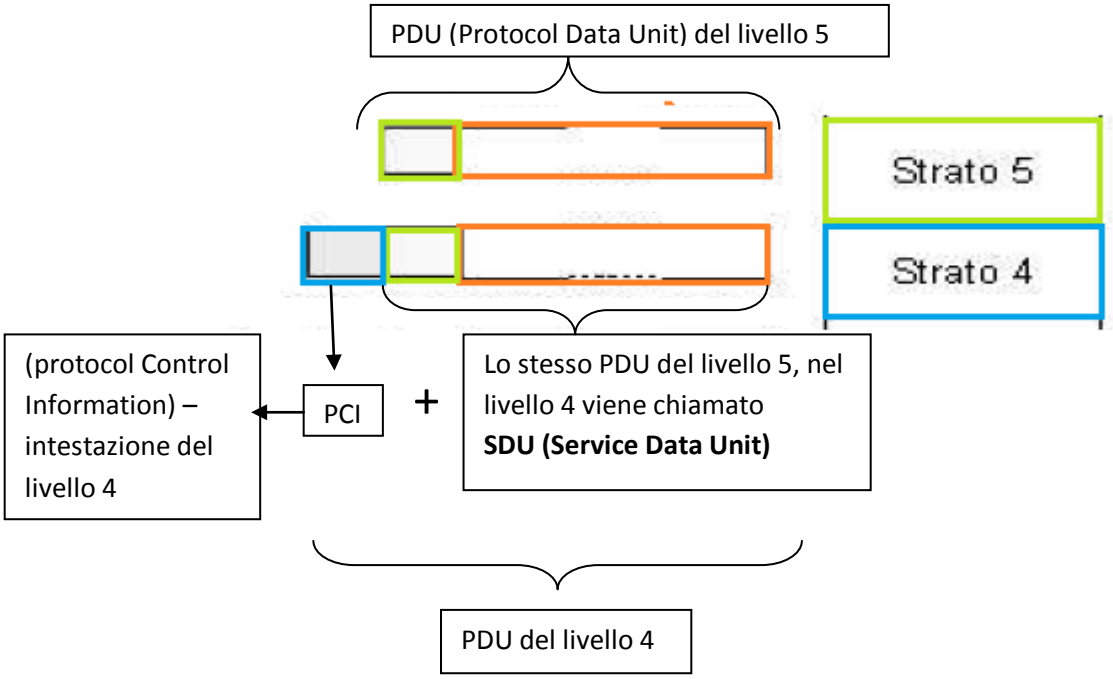


Figura 1-12: Dialogo fra peer entity

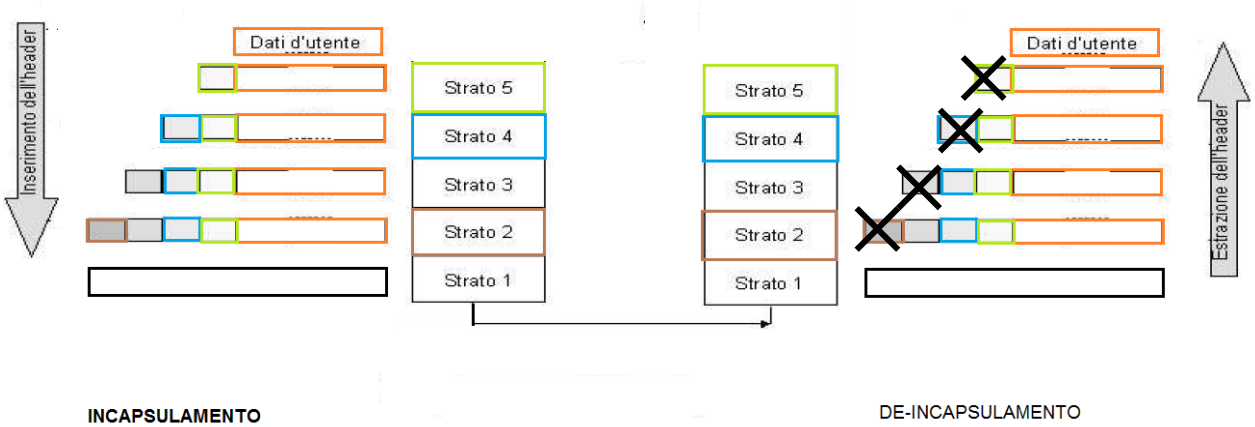
- Il livello N su un host conduce una conversazione con il livello N di un altro host. Le regole e le convenzioni che gestiscono la conversazione prendono il termine di **protocollo di livello n.**

- Questa comunicazione tra entità di pari livello è **VIRTUALE** e avviene tramite lo scambio di pacchetti chiamati **PDU (Protocol Data Unit)**, composti da una parte di **dati** e da una **intestazione** (header) specifica del livello.

- Ogni protocollo descrive il formato del PDU del relativo livello.



- In realtà la comunicazione, cioè il trasferimento dei pacchetti non avviene direttamente tra due entità di pari livello, ma ogni livello del mittente passa le informazioni al livello sottostante, fino ad arrivare al livello 1 dove l'informazione viene trasmessa mediante un mezzo fisico. Il dato quindi passa da ogni livello a quello superiore sino a raggiungere il livello N del destinatario.



- ✓ Ogni livello tranne l'ultimo aggiunge un'intestazione (header) che contiene delle informazioni di controllo (ad esempio la dimensione del messaggio, priorità, indirizzo mittente e destinatario …) e a volte un **trailer** a fine messaggio. (processo detto INCAPSULAMENTO)

- ✓ un livello potrebbe dover *frammentare* un messaggio in unità più piccole, aggiungendo un'intestazione ad ogni frammento;

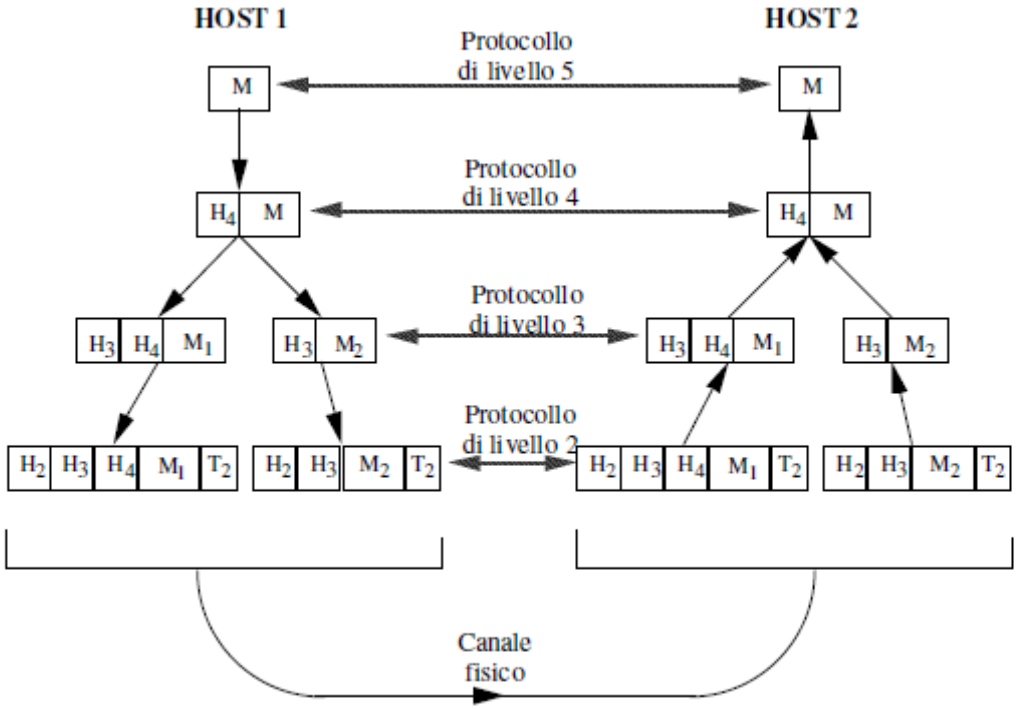


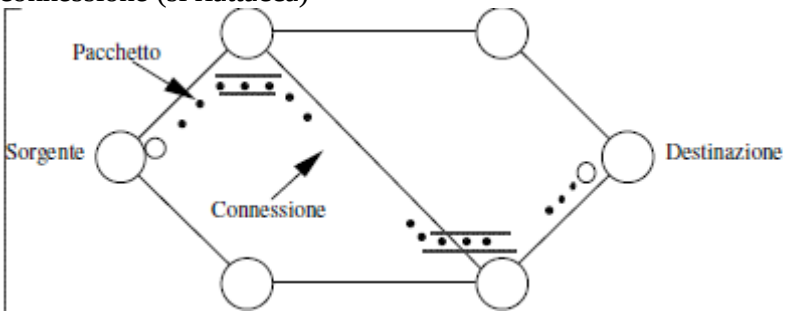
Figura 1-14: Flusso dell'informazione fra peer entity

- ✓ quando il messaggio arriva alla stazione di destinazione il messaggio risale lungo i livelli: ogni livello elimina l'header e trailer del corrispondente livello e poi passa il resto al livello superiore (DE-INCAPSULAMENTO)
- ✓ Il trasferimento dei dati da un livello superiore al livello sottostante viene realizzato da **SERVIZI** offerti dal livello sottostante
- ✓ SCOPO di un livello è di offrire servizi a quello superiore.

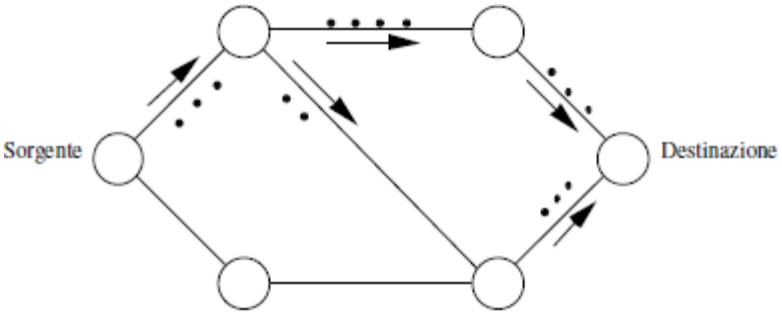
TIPI di servizi.
Un servizio offerto da un livello a quello superiore può essere:

- ✓ ***connection oriented o orientato alla connessione***, come una telefonata:

dopo aver stabilito una connessione (si alza la cornetta e si compone il numero) si scambiano i dati (si parla) e poi si rilascia la connessione (si riattacca)



✓ ***connectionless o privo di connessione,*** come una lettera: due lettere che devono raggiungere la stessa destinazione possono seguire percorsi diversi e arrivare in modo non ordinato o non arrivare; i servizi senza connessione sono chiamati servizi d**atagram**



Inoltre un servizio può essere:

✓ ***affidabile*** o ***servizio con conferma***, se non vengono mai persi dati; normalmente si realizza usando messaggi di conferma di avvenuta ricezione

✓ ***non affidabile*** o ***servizio senza conferma*** se non è garantita la consegna dei dati

Si possono quindi avere le seguenti combinazioni:

✓ **servizio orientato alla connessione e affidabile**, necessario per il trasferimento di un file (i dati devono arrivare tutti ed in ordine)

✓ **servizio orientato alla connessione e non affidabile**, indicato per la trasmissione di voce o filmati in tempo reale; è preferibile infatti un servizio inaffidabile per non subire il ritardo della conferma

✓ **servizio non connesso non confermato**, utilizzabile quando non è importante se qualche messaggio si perde

✓ **servizio non connesso confermato**, in cui dopo l'invio si attende un messaggio di conferma della ricezione

Il livello superiore chiede un servizio al livello sottostante attraverso alcune **primitive** definite nell'**interfaccia** posta tra i due livelli adiacenti.

Un esempio di primitive può essere:

Primitiva	Significato
`request()`	si chiede al servizio di fare qualcosa
`indication()`	si viene avvertiti, dal servizio, di qualche evento
`response()`	si vuole rispondere ad un evento
`confirm()`	la risposta che si attendeva è arrivata

Il servizio connect usato per stabilire la connessione è un **servizio confermato** e quindi vengono usate tutte e quattro le primitive, ad esempio:

- A invia una connect.request per la richiesta di connessione
- B riceve una connect.indication che avvisa il livello della richiesta
- B invia una connect.response per accettare o rifiutare la connessione
- A riceve una connect.confirm con la risposta (accettazione orifiuto)

In modalità connessa due peer entity concordano il trasferimento di una PDU, quindi i pacchetti vengono trasmessi in ordine e se la connessione cade può essere riavviata. Se il servizio è anche affidabile, i pacchetti vengono anche numerati, e, per ogni pacchetto ricevuto, viene inviato dal destinatario un pacchetto di avvenuta ricezione chiamato **ACK** *(Acknowledge)*. Con questa modalità è possibile gestire eventuali errori di trasmissione e anche correggerli.

In modalità non connessa il protocollo non necessita della fase di creazione della connessione e il pacchetto, corredato con l'indirizzo di destinazione, viene inviato come pacchetto a se stante. Con questa modalità il protocollo è generalmente più efficiente, perché non necessita della fase di creazione della connessione ma non può correggere eventuali errori di trasmissione.

Il servizio data usato per il trasferimento dati può essere sia confermato sia non confermato. In caso di non confermato avremo:

- A invia una data.request per inviare dati
- B riceve una data.indication che avvisa dell'arrivo dei dati

Servizi e protocolli sono spesso confusi, ma sono concetti ben distinti.

Servizio	insieme di operazioni primitive che un livello offre al livello superiore. Come tali operazioni siano implementate non riguarda il livello superiore.
Protocollo	insieme di regole che governano il formato ed il significato delle informazioni (messaggi, frame, pacchetti) che le peer entity si scambiano fra loro. Le entità usano i protocolli per implementare i propri servizi.

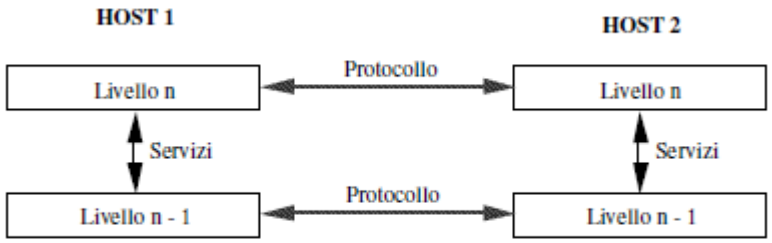


Figura 1-19: Relazione fra protocolli e servizi