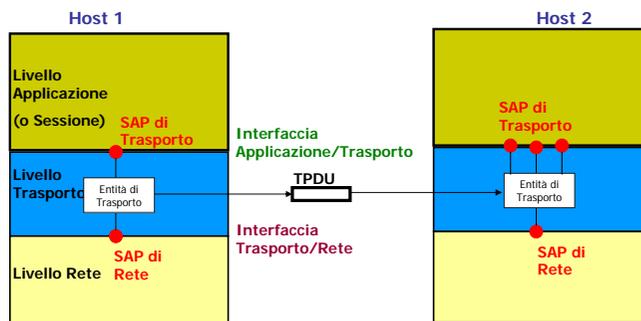


IL LIVELLO TRASPORTO

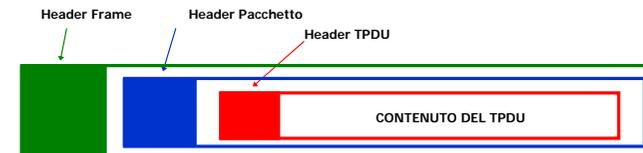
Protocolli TCP e UDP

- Servizi e primitive
- Indirizzamento
- Protocolli di Trasporto
- Livello Trasporto in Internet
 - UDP
 - TCP

- Servizi efficienti ed affidabili per le applicazioni di rete.
- Il software che fornisce i servizi di trasporto è detto Entità di Trasporto.



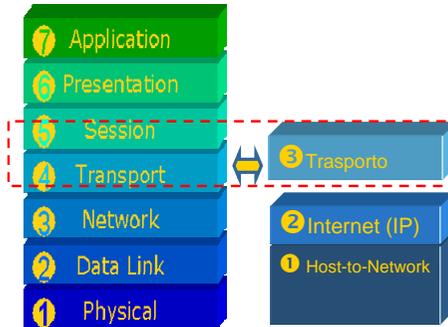
La **TPDU** (**Transport Protocol Data Unit**), spesso denominata anche **segmento**, è l'unità di dati scambiata dal protocollo di trasporto.



Livello TRASPORTO

Funzionalità

1. Indirizzamento dei processi
2. Controllo di flusso
3. Controllo delle connessioni
4. Controllo degli errori
5. Ordinamento
6. Multiplexing
7. Controllo della congestione



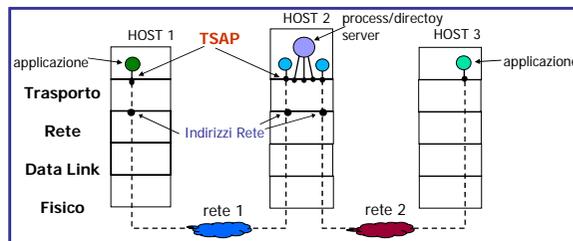
Protocolli di Trasporto

- Sono protocolli end-to-end.
- Gestiscono le connessioni (apertura e chiusura), l'**indirizzamento**, il **controllo di flusso**, il **multiplexing**, l'**ordinamento**, il **controllo degli errori** per un collegamento attraverso una rete.
- La situazione da gestire è più complessa del caso del livello Data Link, perché c'è la rete in mezzo
 - Es, la sottorete ha memoria: pacchetti duplicati!



Indirizzamento e Connessioni

- Un indirizzo di trasporto identifica l'host e la specifica connessione sull'host: **Transport Service Access Point (TSAP)**.



- Se il TSAP non è noto o non è disponibile:
 - Il process server resta in attesa di richieste di connessione, e crea i server che le gestiscono singolarmente.
 - Il directory server fornisce solo l'indirizzo della porta usata per il servizio richiesto (a partire dal nome del servizio)

Protocolli Trasporto di Internet: TCP e UDP

Protocolli di trasporto definiti su rete Internet (su IP)

- **Trasmission Control Protocol (TCP)** definisce un protocollo di trasporto orientato alla connessione
 - progettato per fornire un flusso affidabile end-to-end su una internet inaffidabile (*best effort*).
- **User Data Protocol (UDP)** definisce un protocollo senza connessione
 - permette di inviare datagram IP senza stabilire una connessione
 - si usa per connessioni real-time o per comunicazioni che prevedono solo una richiesta ed una risposta.



Caratteristiche di TCP

- Orientamento alla connessione
- Comunicazione punto-punto (no multicast)
- Totale affidabilità (senza perdite, duplicazioni, inversioni)
- Comunicazione full-duplex
- Trasmissione di flussi di byte (informazioni non strutturate)
- Instaurazione affidabile della connessione (sulla rete potrebbero viaggiare duplicati di richieste di connessioni)
- Rimozione non traumatica della connessione (consegna di tutti i dati inviati prima della chiusura)
- Controllo di flusso (non sommergere il ricevitore!)
- Controllo della congestione
- Eventuale Multiplexing



Funzionamento di TCP

Trasmissione

- Riceve un flusso di dati dall'applicazione.
- Li organizza in unità lunghe al massimo 64KB.
- Spedisce **segmenti** di dati incapsulandoli in datagrammi IP.

Ricezione

- Estrae i segmenti dai datagrammi IP.
- Ricostruisce il flusso di byte originale nella sequenza corretta.

E' necessaria la ritrasmissione dei dati non ricevuti ed il riordinamento dei dati pervenuti in ordine errato.



Le Porte: i TSAP del TCP

- Per connettersi ad un servizio specifico su un server si deve conoscere il numero di porta su cui il processo server accetta le connessioni.
- Le porte da 0 a 1023 sono dette **porte ben note** (well-known ports) e corrispondono a servizi standard.
- Ad esempio:
 - la **porta 21** di TCP corrisponde al servizio **FTP** (File Transfer Protocol).
 - la **porta 80** di TCP corrisponde al servizio **HTTP** (Hypertext Transfer Protocol) utilizzato per il Web
 - per la lista completa <http://www.iana.org/assignments/port-numbers>
 - un servizio "standard" può anche essere attivato su una porta diversa (es. HTTPS sulla porta 443).
- In Unix la lista dei servizi e delle porte è nel file `/etc/services`.

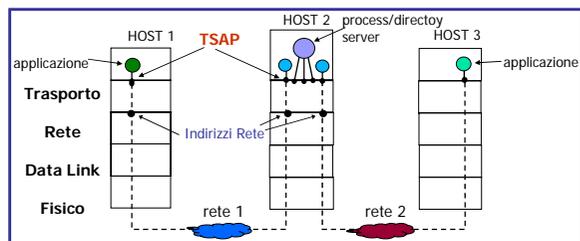


Le Porte del Client

- Il client definisce la porta di ogni sua connessione utilizzando numeri in genere elevati e scelti in modo da essere unici sull'host.
- Ad esempio nella richiesta di connessione ad un server HTTP si può avere:

```
client port 18426
server port 80
```
- Le connessioni TCP sono punto-a-punto e full duplex.

Process server e directory server



- Meccanismi usati nel caso il client non conosca la porta usata dal server :
 - Il directory server fornisce solo l'indirizzo della porta usata per il servizio richiesto (a partire dal nome del servizio)
- ... oppure la porta non è ancora stata creata
 - Il process server resta in attesa di richieste di connessione su più porte non ancora create, e crea i server che gestiscono singolarmente tali connessioni.

Primitive per l'accesso al servizio TCP: i socket

- Il concetto di **socket** è stato introdotto su UNIX BSD (Berkeley): primitive implementate come system call
- Ogni socket è caratterizzato da un indirizzo composto dall'**indirizzo IP** dell'host e da un numero locale a 16 bit (**porta**)
- Per ottenere un servizio TCP si deve creare esplicitamente una connessione fra un socket della macchina richiedente (**client**) ed un socket della macchina ricevente (**server**).
- Una volta attivato, un socket è utilizzato come un file: la **send** corrisponde ad una scrittura su uno stream, la **receive** ad una lettura.
- Le connessioni sono identificate tramite gli identificatori dei socket sui due lati.

Primitive Socket

Sono utilizzate dai programmi per aprire e chiudere connessioni, e per inviare i dati. Sotto: **Primitive Socket di UNIX BSD**

Primitiva	Significato	Uso sul
SOCKET	Crea un socket e restituisce il suo descrittore	C / S
BIND	Assegna una porta locale al socket	Server
LISTEN	Il socket si mette in attesa di richieste di connessione; a tal scopo gestisce una coda	Server
ACCEPT	Se la coda è vuota, attende una richiesta di connessione; altrimenti gestisce la prima richiesta in coda	Server
CONNECT	Richiede una connessione alla controparte	Client
SEND	Invia dati alla controparte	C / S
RECEIVE	Riceve dati dalla controparte	C / S
CLOSE	Rilascia la connessione	C / S

Sequenze di primitive socket

Client	Server semplice	Server "multiplo"
<ul style="list-style-type: none"> ■ Socket ■ Connect ■ (Send / Receive)* ■ Close 	<ul style="list-style-type: none"> ■ Socket ■ Bind ■ Listen ■ Accept ■ (Send / Receive)* ■ Close 	<ul style="list-style-type: none"> ■ Socket ■ Bind ■ Listen ■ Accept ■ Fork <ul style="list-style-type: none"> ■ (Send / Receive)* ■ Close ■ Exit

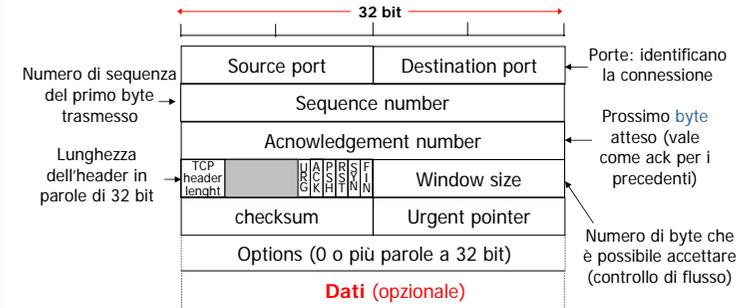
Attenzione: i metodi java delle classi *Socket* e *ServerSocket* sono definiti ad un livello di astrazione superiore

Multiplexing

- normalmente, una connessione network è usata per ciascuna connessione transport
- Downward multiplexing
 - per ottenere un alto throughput: molte connessioni network per una singola connessione transport
- Upward multiplexing
 - Se esiste una sola connessione network oppure è alto il costo di una connessione network
 - una singola connessione network è usata per molte connessioni transport

Il Segmento TCP

- Ogni **segmento TCP** ha un header fisso di 20 byte più eventuali dati opzionali seguiti da 0 o più byte di dati.
- La suddivisione dei dati in segmenti fatta dal mittente può variare lungo la rete: flusso di byte e non flusso di messaggi



I Flag TCP

Nel segmento TCP sono presenti 6 bit di flag:

ACK

Indica se il campo **Acknowledgement number** è valido.

SYN

Viene utilizzato per creare connessioni. La richiesta di connessione è caratterizzata da SYN=1 e ACK=0. La risposta di connessione contiene un ack e quindi ha SYN=1 e ACK=1. Corrispondono alle primitive astratte **CONNECTION REQUEST** e **CONNECTION ACCEPTED**.

RST

Richiesta di re-inizializzazione di una connessione diventata instabile. Viene anche usato per rifiutare un segmento non valido o l'apertura di una connessione.

FIN

Viene utilizzato per chiudere una connessione (il mittente non ha altri dati da spedire).

I Flag TCP

URG

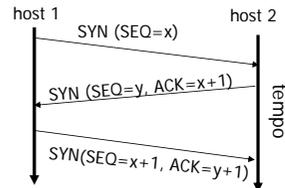
Se il bit URG è 1, Urgent Pointer indica la posizione, a partire dal numero di sequenza attuale, dei **dati urgenti** (es. pressione di CTRL-C per interrompere il programma remoto).

PSH

Indica dati di tipo **PUSH**, ovvero si richiede di consegnare subito i dati senza bufferizzarli.

Apertura della Connessione

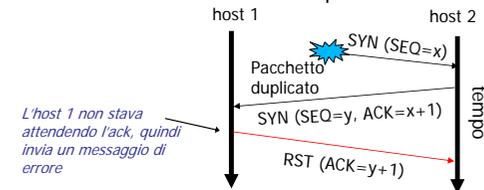
- Si utilizza un protocollo **3-way handshake**
- Le due parti devono riconoscere i rispettivi numeri di sequenza



- Se il TCP ricevente non verifica la presenza di un processo in attesa sulla porta destinazione manda un segmento di rifiuto della connessione (RST).

Gestione dei Pacchetti Duplicati

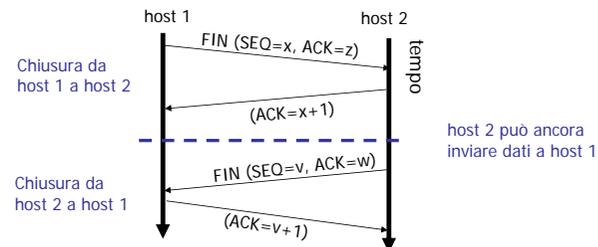
- I pacchetti (nello specifico le richieste di connessione) possono essere "memorizzati" e ricomparire nella rete.



- La numerazione è fatta con un orologio locale (tick=4 μs)
- L'intervallo dei numeri di sequenza (32 bit) garantisce che non venga riutilizzato lo stesso numero di sequenza prima di qualche ora
- A causa del **time to live** dei pacchetti IP, segmenti con lo stesso numero non possono coesistere sulla rete

Chiusura della Connessione

- La connessione è full-duplex e le due direzioni devono essere chiuse indipendentemente.



- Se l'ack di un messaggio FIN si perde l'host mittente chiude comunque la connessione dopo un timeout.

TCP: ack e ritrasmissioni

- TCP deve garantire la consegna dei dati ed il loro corretto ordinamento.
- A tal scopo, i dati sono ordinati tramite **numeri di sequenza**.
- Dopo aver inviato un segmento di dati, il trasmittente attende un **ack** da parte del ricevente.
- Viene utilizzato un **timeout** per limitare l'attesa dell'ack: se il timeout scade, il trasmittente provvede alla ritrasmissione dei dati.
- Il ricevente ha il compito di riordinare i dati; per poter accettare dati non in ordine, mantiene un apposito buffer.
- La ricezione di un ack avente un dato numero di sequenza vale come "ricevuta" per tutti i dati con numero di sequenza inferiore trasmessi in precedenza.

TCP: timeout

- Il problema è determinare il valore del timeout migliore (i ritardi possono essere molto variabili nel tempo sulla rete)
 - Se il timeout è troppo piccolo si faranno ritrasmissioni inutili
 - Se il timeout è troppo elevato si avranno ritardi di trasmissione eccessivi

TCP: timeout

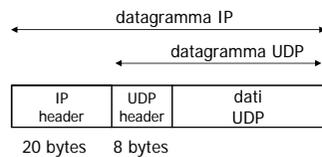
- Il problema è determinare il valore del timeout migliore (i ritardi possono essere molto variabili nel tempo sulla rete)
 - Se il timeout è troppo piccolo si faranno ritrasmissioni inutili
 - Se il timeout è troppo elevato si avranno ritardi di trasmissione eccessivi



- Si utilizza un algoritmo di stima del migliore timeout basato sulla misura del **Round-Trip Time (RTT)**.

UDP: Trasporto senza Connessione

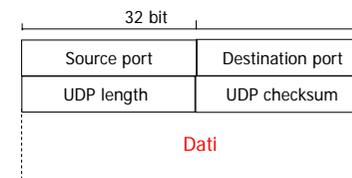
- Ogni datagramma UDP è incapsulato in un pacchetto IP.



- UDP non garantisce affidabilità di consegna.
- Viene utilizzato da:
 - Applicazioni real time
 - Applicazioni request-response (es. interrogazione di un database)
 - Applicazioni di monitoring
 - ...

Header UDP

- Header di un datagramma UDP: soli 8 byte, dimensione fissa.



- Le **porte** UDP sono indipendenti da quelle TCP.
- La **lunghezza** in byte comprende sia i dati che l'header.
- Il **checksum** controlla la correttezza dei dati: non sempre conviene usarlo.