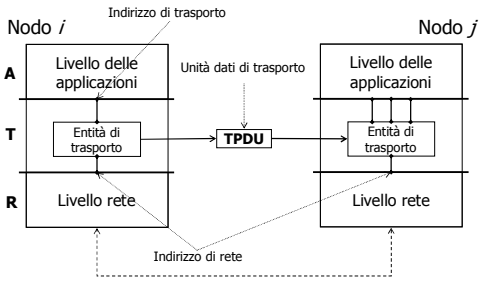
Rete: livello trasporto (TCP/IP)
Parte 1 - Indice

1. Posizionamento nella gerarchia
2. `Socket`
3. Indirizzamento
4. Connessione
5. Controllo di flusso

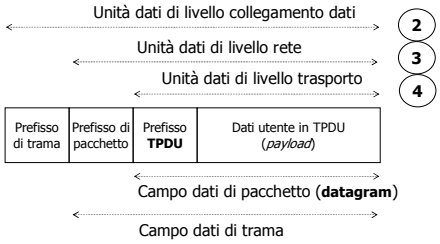
Rete: livello trasporto (TCP/IP)
Posizionamento nella gerarchia - 1



Rete: livello trasporto (TCP/IP)
Posizionamento nella gerarchia - 2

- Posto tra il livello delle **applicazioni** `A` (o sessione, se presente) ed il livello **rete** `R`
  - I suoi servizi sono resi da **entità di trasporto**, che celano al cliente l'esecuzione dei relativi protocolli
  - I clienti di livello `A` in comunicazione tra loro vedono semplicemente un flusso di dati
- Offre ad `A` **più punti di accesso** a servizi di trasporto, mappandoli su un **unico indirizzo** di rete (nello spazio di `R`)
- Contempera l'inaffidabilità del livello rete
  - **Qualità del Servizio** (QoS) è lo sforzo impegnato dal livello trasporto per raggiungere il livello di affidabilità richiesto dai clienti applicativi

Rete: livello trasporto (TCP/IP)
Posizionamento nella gerarchia - 3



Rete: livello trasporto (TCP/IP)
Posizionamento nella gerarchia - 4

- Il livello rete è eterogeneo, e dunque inevitabilmente inaffidabile rispetto alla corretta gestione di flusso (tra nodi), congestione (di *router*), errore
  - Il livello 2 interconnette due *router* agli estremi di un collegamento punto a punto
  - Il livello 4 interconnette due **nodi** (*host*) agli estremi di una connessione virtuale realizzata su una rete senza connessioni

Rete: livello trasporto (TCP/IP)
Socket - 1

- Basate sul modello UNIX BSD
- `Socket` = terminale (***end point***) di comunicazione di livello trasporto
  - `Socket` diversi per comunicazioni con o senza connessione (TCP o UDP)
- Al `socket` di destinazione corrisponde un indirizzo **locale** (**porta**) associato all'indirizzo `IP` del nodo **D**
- A livello implementativo, un `socket` è una **coda** di ricezione con uno stato controllato dal S/O
- Il processo utente associato al `socket` **si blocca** in attesa di comunicazioni in ingresso (se **D**) o di conferma di ricezione (**M**, con connessione)

Rete: livello trasporto (TCP/IP)
Socket - 2

- API di UNIX BSD (1/2)
  - **`socket`**(…) : crea un **`socket`**, restituendo al chiamante l'`ID` del descrittore del *file* speciale corrispondente
  - **`bind`**(…) : associa una porta al **`socket`**, creando così la sua identità di rete
    - Il lato *server* può così pubblicare la propria identità
    - Il cliente usa la propria identità per stabilire il collegamento con il proprio *server*
  - **`connect`**(…) : blocca il chiamante fino allo stabilirsi della connessione richiesta
    - Il cliente vi indica l'indirizzo del *server* ed il proprio **`socket`** locale viene creato automaticamente (senza invocazione di **`bind`**)

Rete: livello trasporto (TCP/IP)
Socket - 3

- API di UNIX BSD (2/2)
  - **`listen`**(…) : dimensiona la coda associata al **`socket`** di lato *server* e pone il chiamante in attesa di richieste
    - Lo stesso *server* può servire più connessioni simultanee
  - **`accept`**(…) : blocca il chiamante fino ad una richiesta di connessione, ricevuta la quale le associa un **`socket`** **identico** a ma **distinto** da quello sul quale la richiesta è pervenuta
    - Un agente del *server* ascolta su **`socket`** iniziale mentre varie istanze di *server* (anche diversi tra loro) lavorano su **`socket`** creati dinamicamente
  - **`send`**(…) / **`receive`**(…) : invia / riceve su un **`socket`**
  - **`close`**(…) : **M** e **D** rilasciano il proprio **`socket`** e dunque la relativa connessione

Rete: livello trasporto (TCP/IP)
Socket - 4

- I **`socket`** BSD hanno un tipo, che specifica:
  - Il **dominio** di indirizzamento
    - **`AF_INET`** denota l'indirizzamento **`IP`** : uso di porta
  - Lo stile di comunicazione (l'unità dati)
    - **`SOCK_DGRAM`** denota l'uso di **`datagram`**
  - Il tipo di comunicazione (il **protocollo**)
    - **`IPPROTO_TCP`** / **`_UDP`** denota i protocolli a noi noti
- I **`socket`** (**`datagram`**) inviano a **`sendto`** (**D**) e ricevono da **`recvfrom`** (**M**)
  - **`send`** e **`receive`** operano su qualunque connessione aperta

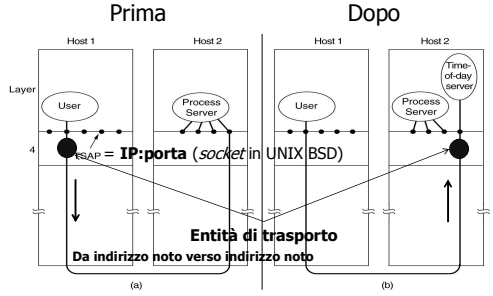
Rete: livello trasporto (TCP/IP)
Indirizzamento - 1

- La comunicazione a livello trasporto richiede conoscenza dell'**indirizzo locale** della controparte **D** = (**`IP`** : porta)
  - Forma di indirizzamento **gerarchica**
- L'indirizzo **D** può essere noto a priori
  - Alcune applicazioni ascoltano su una porta standard fissata, dunque nota a priori ad ogni utente
- Altrimenti un ***name server*** apposito associa il **nome logico** dell'applicazione richiesta al suo indirizzo di livello trasporto

Rete: livello trasporto (TCP/IP)
Indirizzamento - 2

- **M** e **D** sono associati dalle rispettive entità di trasporto ad uno specifico punto di accesso locale (unico nel nodo)
- Alcuni processi **D** (***server***) si pongono in attesa permanente dietro il loro punto di accesso
- Altri processi usano invece un agente (***proxy***) che ascolta il loro vece ed assegna loro dinamicamente un punto di accesso all'arrivo di una richiesta

Rete: livello trasporto (TCP/IP)
Indirizzamento - 3



Rete: livello trasporto (TCP/IP)
Connessione - 1

- La rete **ha memoria**, poiché non cancella automaticamente tutti i duplicati di pacchetti consegnati con successo né le loro conferme
  - Pacchetti duplicati arrivati tardi a destinazione possono esser trattati come nuova conversazione
  - Ogni connessione dovrebbe invece avere inizio e fine **univoci**, che invalidino flussi duplicati
- Problema reso ancor più complesso dal fatto che i nodi possono **perdere traccia** del loro stato di comunicazione

Rete: livello trasporto (TCP/IP)
Connessione - 2

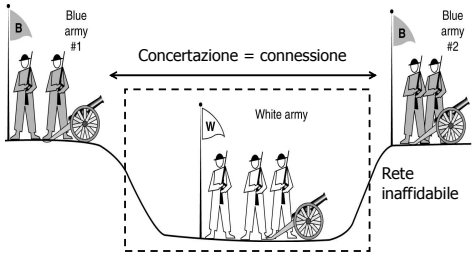
- I pacchetti in viaggio sulla rete accumulano ritardi variabili
  - Pacchetti dati, conferme, pacchetti di controllo
- Servono tecniche che offrano garanzie di inizializzazione consistente a **M** e **D**
- ***Three-way handshake***
  - **M** → **D** (`rc,x`) : **richiesta di connessione**, con emissione di TPDU **numerati** a partire da `x`
  - **D** → **M** (`cc,y,x`) : **conferma di connessione** (`rc,x`), con emissione di TPDU **numerati** a partire da `y`
  - **M** → **D** (`dati,y`) : invio del primo TPDU dati, numerato `x`, con conferma dell'indice `y` di **D**

Rete: livello trasporto (TCP/IP)
Rilascio

- La terminazione di una connessione può avvenire in modo asimmetrico o simmetrico
  - **Asimmetrico**: come nella conversazione telefonica; il rilascio di una parte distrugge la connessione
    - Può comportare **perdita di dati** → non desiderabile
  - **Simmetrico**: il rilascio deve avvenire da ambo i lati della connessione (anche in modo asincrono)
    - Il **`socket`** richiede rilascio **simmetrico**
    - Può lasciare la connessione **aperta** all'infinito
      - **Problema dei 2 eserciti**
    - I lati delle connessioni che non portino dati validi entro **intervalli fissati** sono autonomamente rilasciati dal rispettivo possessore (approccio collaborativo)

Rete: livello trasporto (TCP/IP)
Illustrazione del problema (senza soluzione)

Attacco coordinato = rilascio simmetrico



Rete: livello trasporto (TCP/IP)
Controllo di flusso - 1

- Problematica analoga a quella incontrata al livello 2 (collegamento dati)
  - Protocolli come **`SWP`** e **`SRP`** richiedono ai 2 lati della connessione di trattenere trame in memoria
    - I *router* connettono relativamente **poche linee**, il che richiede *buffer* di ampiezza accettabile
    - Nodi **M** e **D** possono ospitare **numerosissimi flussi** di conversazione, il che richiede molta più memoria
- Se la rete è inaffidabile (caso **`Internet`**) **M** e **D** devono salvare i pacchetti inviati, secondo le regole del protocollo adottato

Rete: livello trasporto (TCP/IP)
Controllo di flusso - 2

- Allocazione **statica** di memoria pacchetti
  - Un insieme di *buffer* di dimensione fissa ciascuno assegnato ad 1 TPDU
    - Enorme spreco di memoria causato dall'esigenza che i *buffer* siano dimensionali alla massima ampiezza di TPDU
- Allocazione **dinamica**
  - 1 solo *buffer* circolare ∀ connessione
    - Spreca memoria se la connessione ha poco traffico, altrimenti ha buona resa

Rete: livello trasporto (TCP/IP)
Controllo di flusso - 3

- **Allocazione dinamica**
  - **`SWP`** e **`SRP`** legano la gestione dei *buffer* all'arrivo di conferme di ricezione; questo non conviene in ambiente di rete
  - Meglio che **M** richieda a **D** un'ampiezza di *buffer* dimensionata alle sue aspettative di traffico e **D** indichi, all'atto di attivazione della connessione, la propria disponibilità corrente
    - Lo spazio reso disponibile da **D** si riduce progressivamente ad ogni TPDU emesso da **M,**
    - In questo modo **M** può adattare il suo flusso in uscita alle capacità ricettive di **D**

Rete: livello trasporto (TCP/IP)
Controllo di flusso - 4

- **Allocazione dinamica**
  - Un meccanismo di controllo che si basi **solo** sulla capacità ricettiva di **D** assume (**sbagliando**) che la rete abbia capacità trasmissiva infinita
  - Ad **M non conviene** emettere troppi pacchetti senza conferma
    - Capacità di assorbimento della rete = ***n*** pacchetti / *sec.*
    - Tempo medio tra emissione di TPDU ed arrivo di conferma = ***p*** *sec.*
    - Con 1 *buffer* di dimensione = $n \times p$ pacchetti, **M** può operare a massima velocità (rispetto alla rete) avendo però il *buffer* sempre mediamente pieno