Corsi di Laurea in Ingegneria Informatica
Ingegneria delle Telecomunicazioni
Ingegneria dell'Automazione

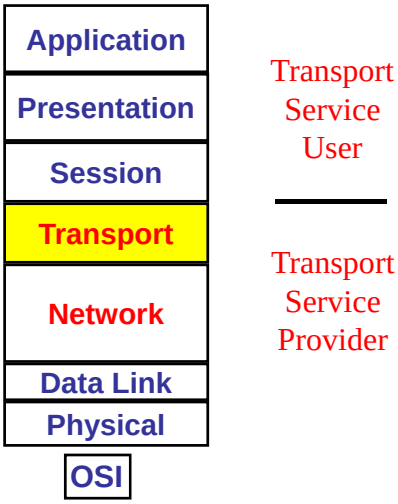
**Corso di Reti di Calcolatori**

Simon Pietro Romano (spromano@unina.it)
Antonio Pescapè (pescape@unina.it)
Giorgio Ventre (giorgio@unina.it)

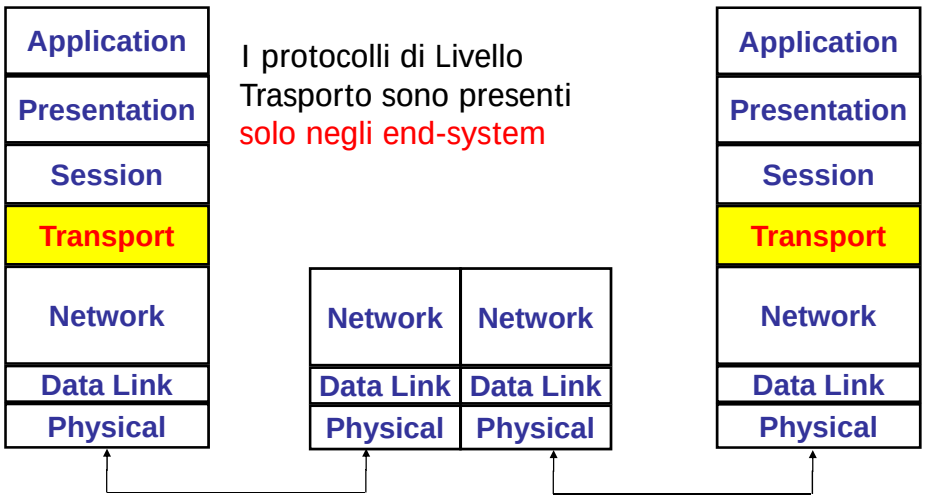
# Il livello trasporto:

# Introduzione e protocollo UDP

## Livello Trasporto



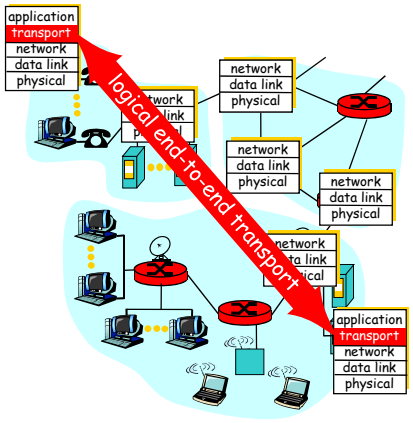
## Livello Trasporto




3

## Servizi e Protocolli del Livello Trasporto

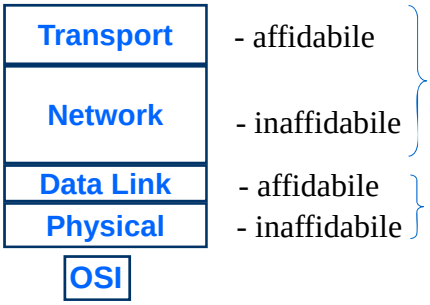
- Offre un canale di comunicazione logica tra applicazioni attive su differenti host
- Differenze tra livello trasporto e livello rete:
  - Network-layer: trasferimento dati tra end-system
  - Transport layer: trasferimento dati tra processi. Naturalmente necessita dei servizi offerti dal livello rete.




Da trasporto da *host a host* a trasporto da *processo a processo*

4

## Servizi del Livello Trasporto - 1




Isolare i livelli superiori dai problemi dovuti all'uso di differenti tecnologie di rete e dalle loro (eventuali) imperfezioni

Il livello rete offre un servizio inaffidabile, quindi:

Il livello trasporto deve rimediare:

- aumentare l'efficienza
- aumentare l'affidabilità

In particolare:

- controllo degli errori
- sequenza ordinata
- controllo di flusso
- controllo di congestione

5

## Servizi del Livello Trasporto - 2

- I protocolli di Livello Trasporto sono realizzati al di sopra del Livello Rete, quindi è necessario gestire:
  - apertura della connessione (setup)
  - memorizzazione dei pacchetti all'interno della rete
  - un numero elevato di connessioni …
    - Multiplexing e Demultiplexing

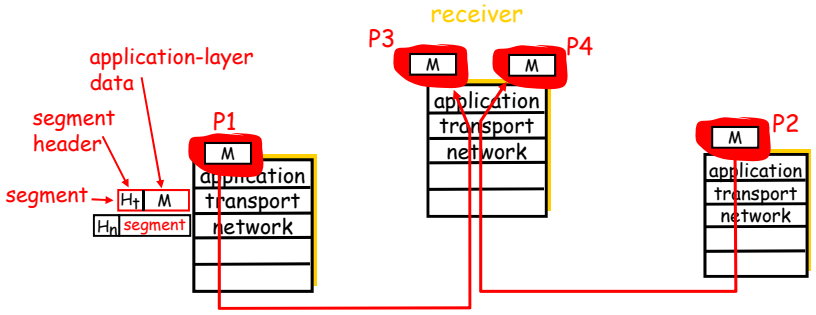
6

# Multiplexing e Demultiplexing - 1

*segment* – dati che sono scambiati tra processi a livello trasporto

Demultiplexing: inoltrare i segment ricevuti al corretto processo cui i dati sono destinati

TPDU: transport protocol data unit



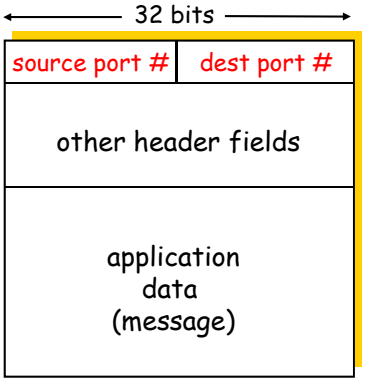



# Multiplexing e Demultiplexing - 2

Multiplexing: Raccogliere i dati provenienti dalle applicazioni, imbustare i dati con un header approriato (per il de-multiplexing)

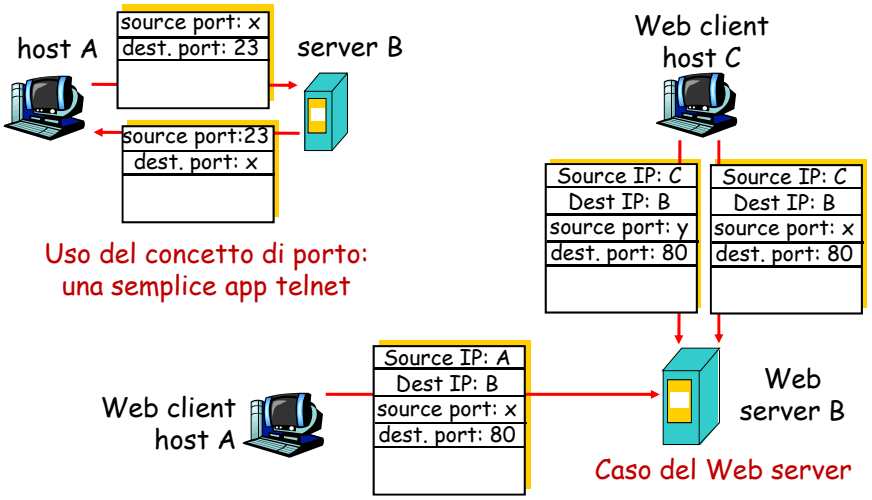
multiplexing/demultiplexing:

- Realizzato attraverso la coppia <indirizzo IP, numero di porto>
  - source, dest port # è presente in ogni segmento
  - numeri di porto "well-known" per applicazioni particolari




TCP/UDP segment format

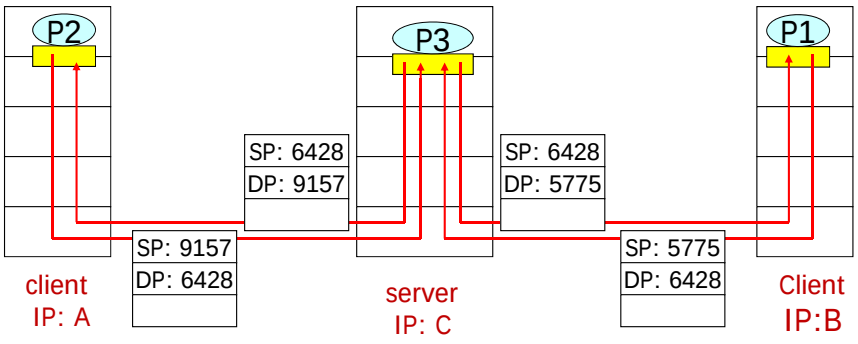
## Multiplexing e Demultiplexing: esempi




9

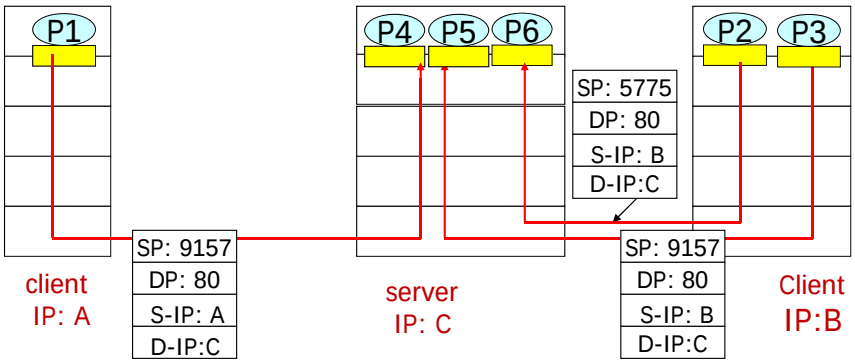
## Connectionless demux

Su P3 c'è un processo in ascolto sul porto UDP 6428.



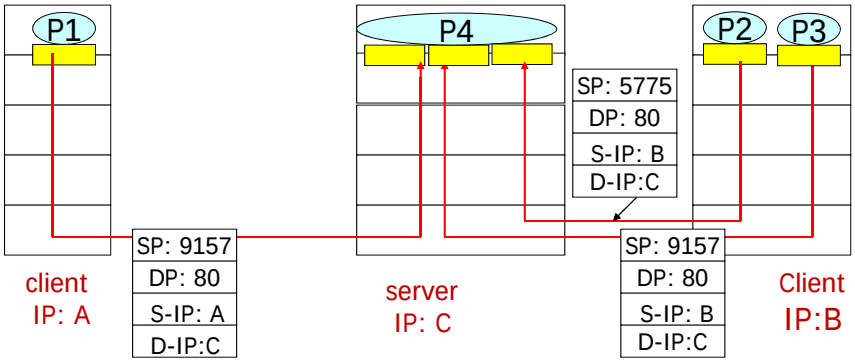

Il SP fornisce il "return address"

## Connection-oriented demux 1/2

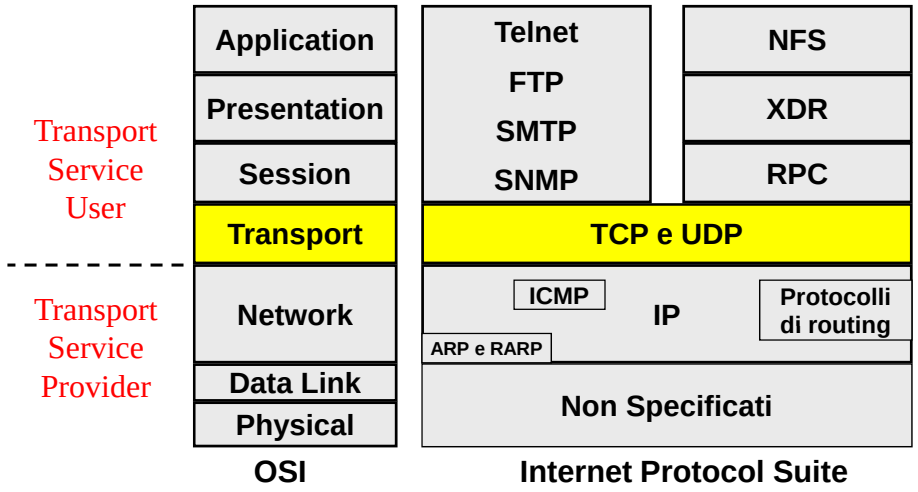



Osservazione su HTTP persistente e non persistente...

## Connection-oriented demux: Threaded Web Server



## I protocolli TCP e UDP






## UDP: User Datagram Protocol [RFC 768]

- Aggiunge poco ad IP:
  - servizio “best effort”:
    - i pacchetti UDP possono:
      - subire perdite
      - giungere a destinazione in ritardo, o non arrivare affatto
      - giungere a destinazione non ordinati
  - *servizio connectionless:*
    - non è prevista una fase di inizializzazione
    - ogni segmento UDP è inviato indipendentemente dagli altri
      - *Domanda: C’è qualcuno in ascolto?*

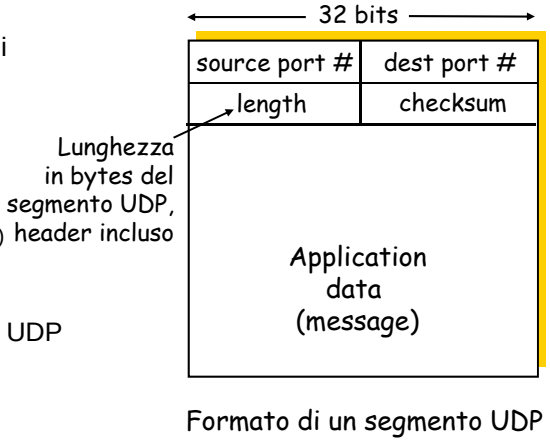
## UDP: User Datagram Protocol - 2

- Perché è stato introdotto UDP?
  - non è necessaria la fase di inizializzazione (setup) che introduce delay
    - Es: DNS è basato su UDP
  - semplice: sender e receiver non devono conservare informazioni di stato
  - intestazione di dimensioni contenute:
    - basso overhead
  - controllo della congestione assente:
    - le applicazioni possono inviare alla velocità desiderata
      - utile per alcune applicazioni
      - rischioso per la rete



## UDP: User Datagram Protocol - 3

- Ampiamente usato per applicazioni multimediali:
  - tolleranti alle perdite di pacchetti
  - sensibili ai ritardi
- Altre applicazioni:
  - DNS
  - NFS (Network File System)
  - SNMP (Simple Network Management Protocol)
- Domanda: si può rendere UDP affidabile?
  - Gestione degli errori
  - Conferma di avvenuta ricezione




Formato di un segmento UDP

## Checksum UDP

Obiettivo: rilevare gli “errori” (bit alterati) nel segmento trasmesso

Mittente:

- Tratta il contenuto del segmento come una sequenza di parole da 16 bit
- ***checksum***: somma (complemento a 1) i contenuti del segmento
- Il mittente pone il valore della checksum nel campo checksum del segmento UDP

Ricevente:

- calcola la checksum del segmento ricevuto
- controlla se la checksum calcolata è uguale al valore del campo checksum:
  - No - errore rilevato
  - Sì - nessun errore rilevato. *Ma potrebbero esserci errori nonostante questo?* Altro più avanti …

## Incapsulamento di segmenti UDP