

**Corso di Laurea in Ingegneria Informatica**



**Corso di Reti di Calcolatori**

**Roberto Canonico ([roberto.canonico@unina.it](mailto:roberto.canonico@unina.it))**

**Giorgio Ventre ([giorgio.ventre@unina.it](mailto:giorgio.ventre@unina.it))**

**Il livello trasporto: introduzione  
Il protocollo UDP**

**I lucidi presentati al corso sono uno strumento didattico  
che NON sostituisce i testi indicati nel programma del corso**



## Nota di Copyright

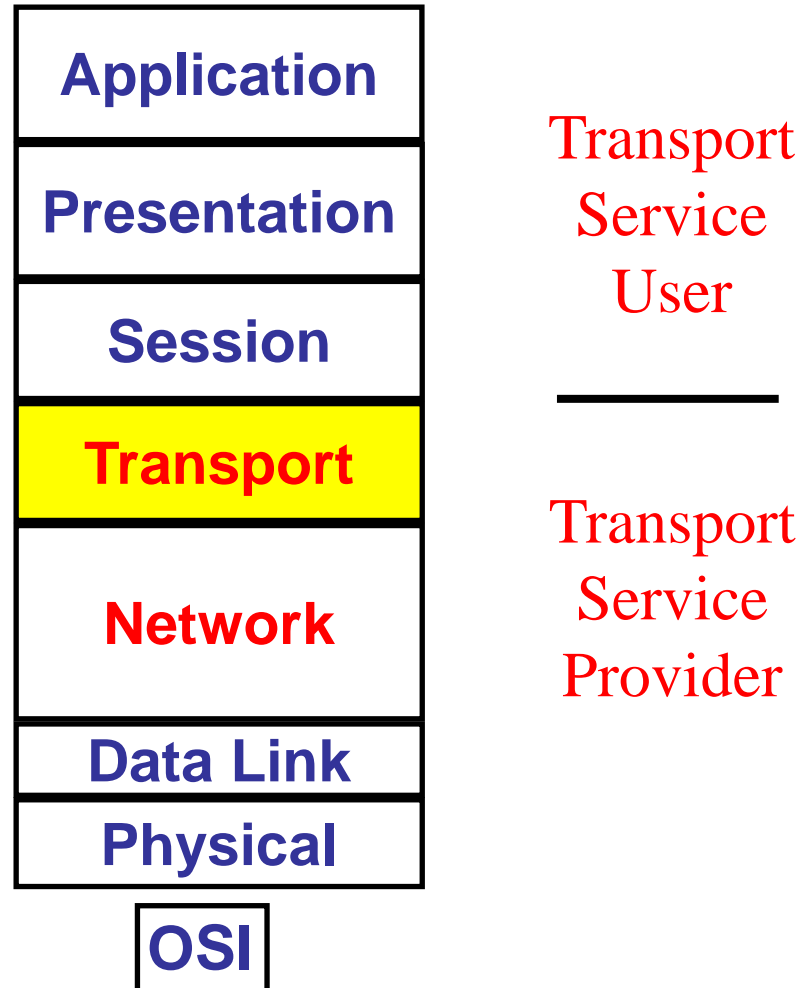
Questo insieme di trasparenze è stato ideato e realizzato dai ricercatori del Gruppo di Ricerca COMICS del Dipartimento di Informatica e Sistemistica dell'Università di Napoli Federico II. Esse possono essere impiegate liberamente per fini didattici esclusivamente senza fini di lucro, a meno di un esplicito consenso scritto degli Autori. Nell'uso dovranno essere esplicitamente riportati la fonte e gli Autori. Gli Autori non sono responsabili per eventuali imprecisioni contenute in tali trasparenze né per eventuali problemi, danni o malfunzionamenti derivanti dal loro uso o applicazione.

### Autori:

Simon Pietro Romano, Antonio Pescapè, Stefano Avallone,  
Marcello Esposito, Roberto Canonico, Giorgio Ventre



# Livello Trasporto





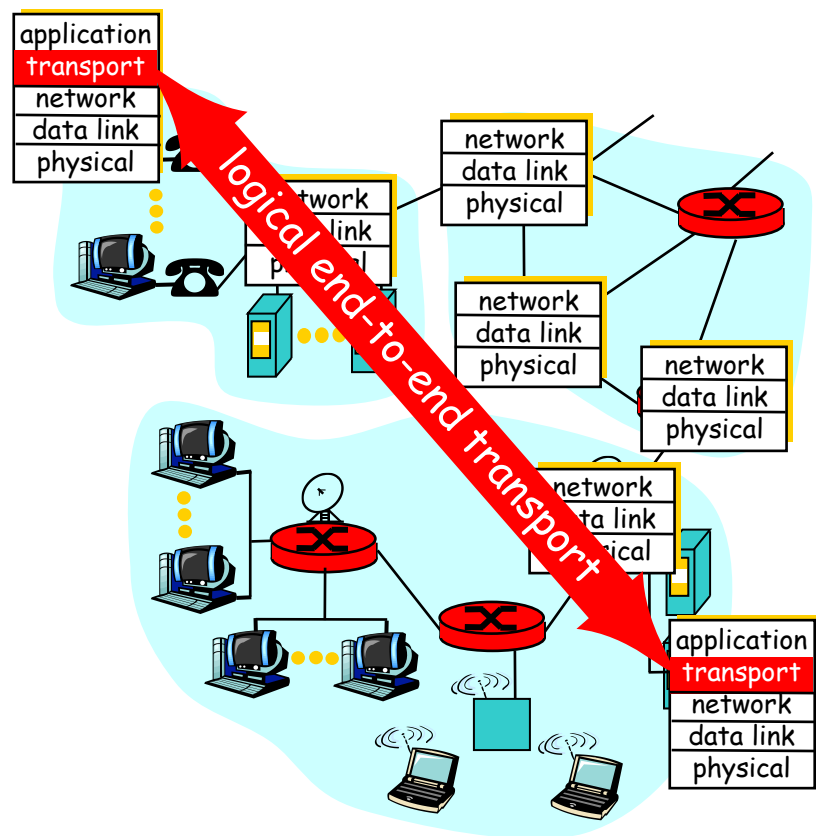
# Livello Trasporto





# Servizi e Protocolli del Livello Trasporto

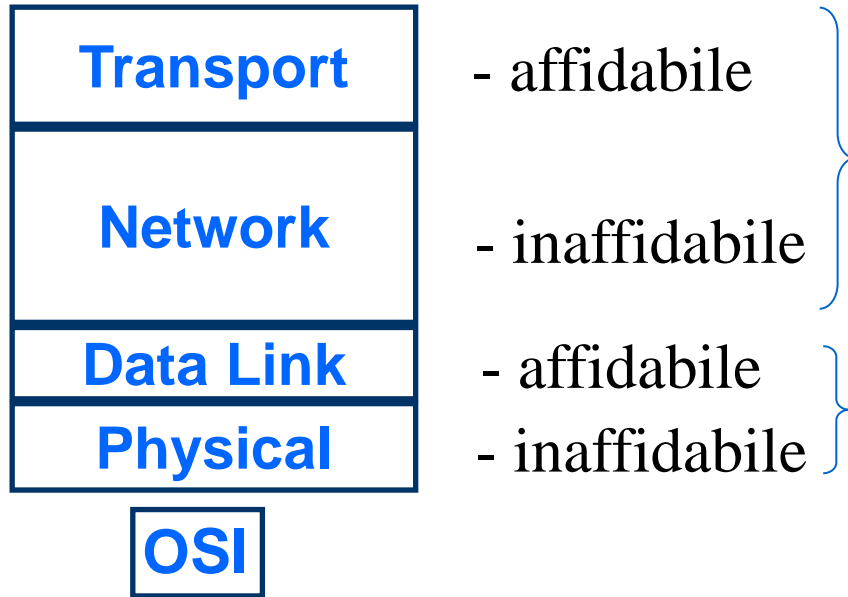
- Offre un canale di **comunicazione logica** tra applicazioni attive su differenti host
- Differenze tra livello trasporto e livello rete:
  - **Network-layer**: trasferimento dati tra end-system
  - **Transport layer**: trasferimento dati tra processi. Naturalmente necessita dei servizi offerti dal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



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



# 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



# 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

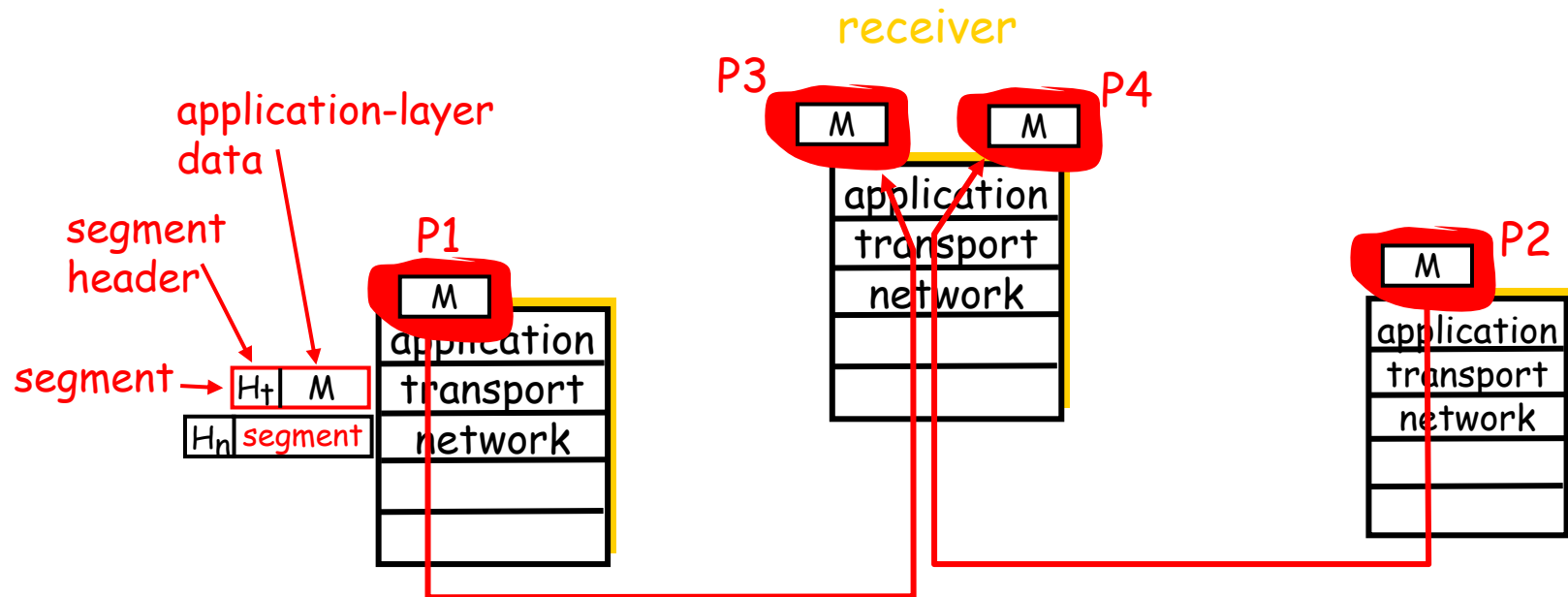


# Multiplexing e Demultiplexing - 1

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

TPDU: transport protocol data unit

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







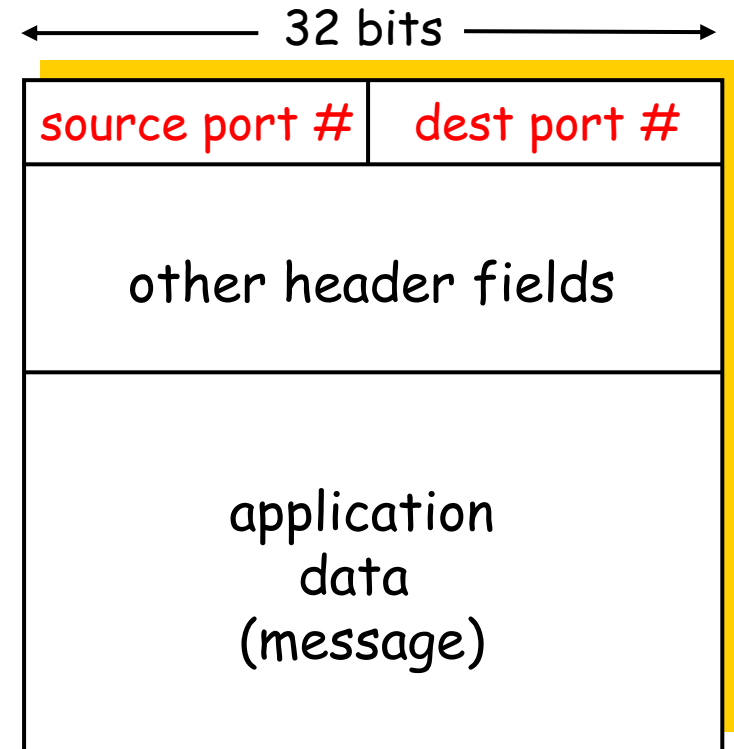
# Multiplexing e Demultiplexing - 2

## Multiplexing:

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

multiplexing/demultiplexing:

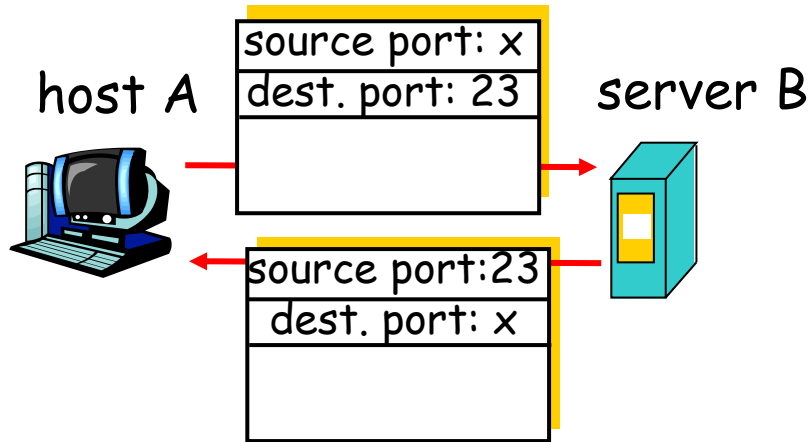
- Realizzato attraverso la coppia **<indirizzo IP, numero di porto>**
  - source, dest port # è presente in ogni segmento
  - numeri di porto “**well-known**” per applicazioni particolari



TCP/UDP segment format



# Multiplexing e Demultiplexing: esempi



Uso del concetto di porto:  
una semplice app telnet



Web client  
host A

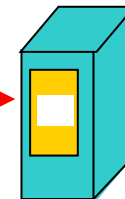
Source IP: A
Dest IP: B
source port: x
dest. port: 80

Web client  
host C



Source IP: C
Dest IP: B
source port: y
dest. port: 80

Source IP: C
Dest IP: B
source port: x
dest. port: 80



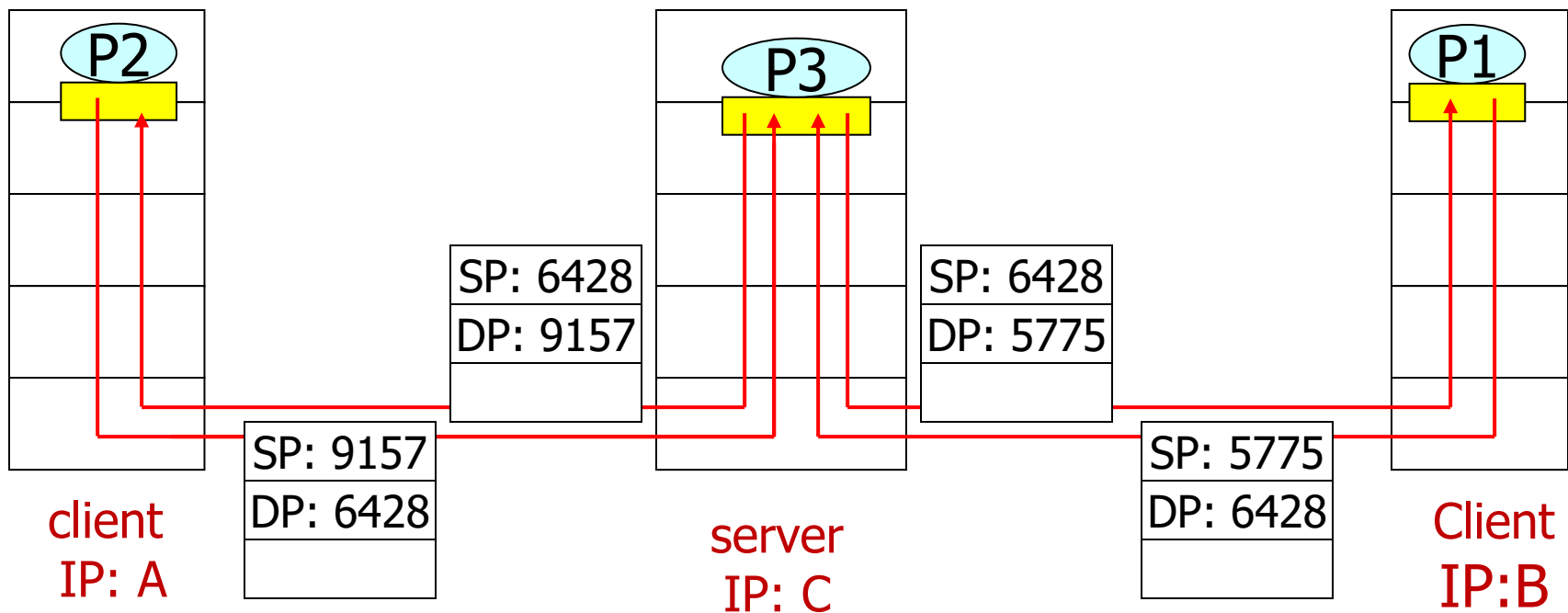
Web  
server B

Caso del Web server



# Connectionless demux

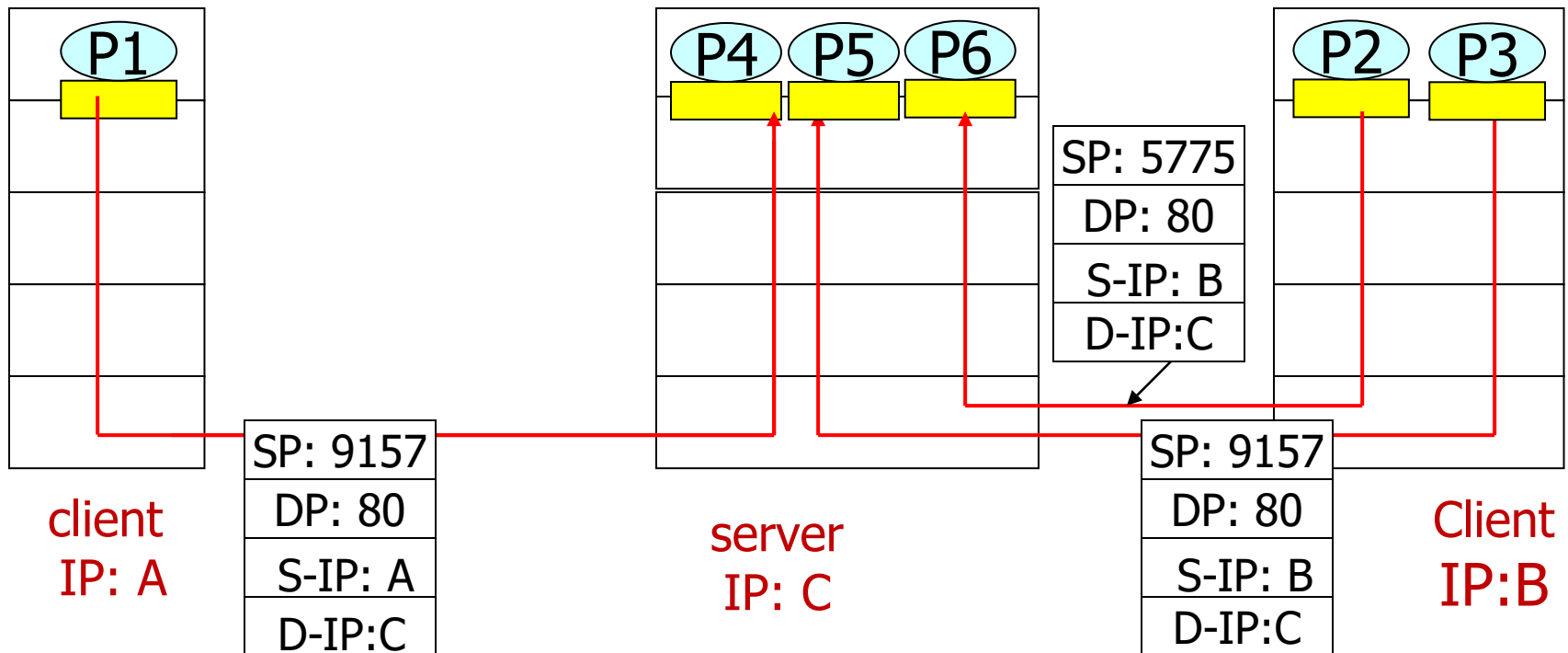
Su P3 c'è un processo in ascolto sul porto UDP 6428.



Il SP fornisce il "return address"



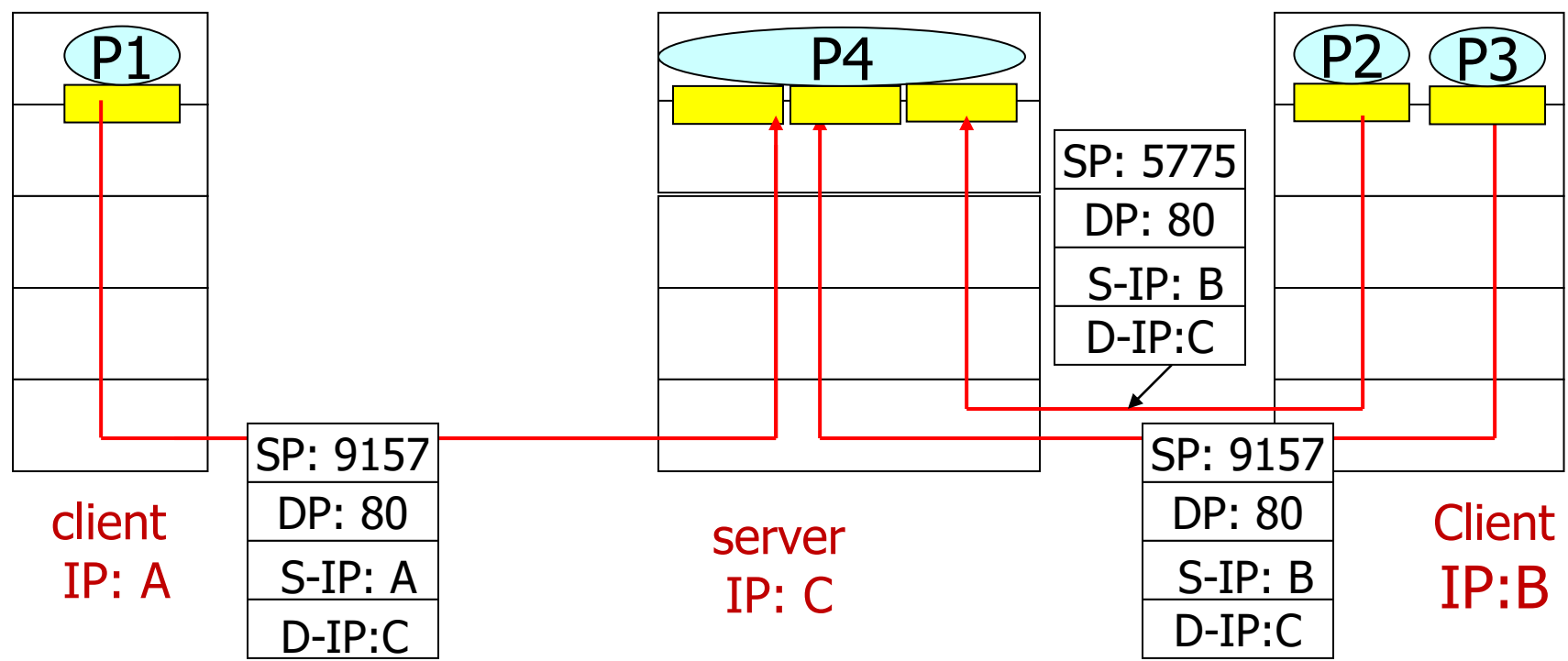
# Connection-oriented demux 1/2



Osservazione su HTTP persistente e non persistente...

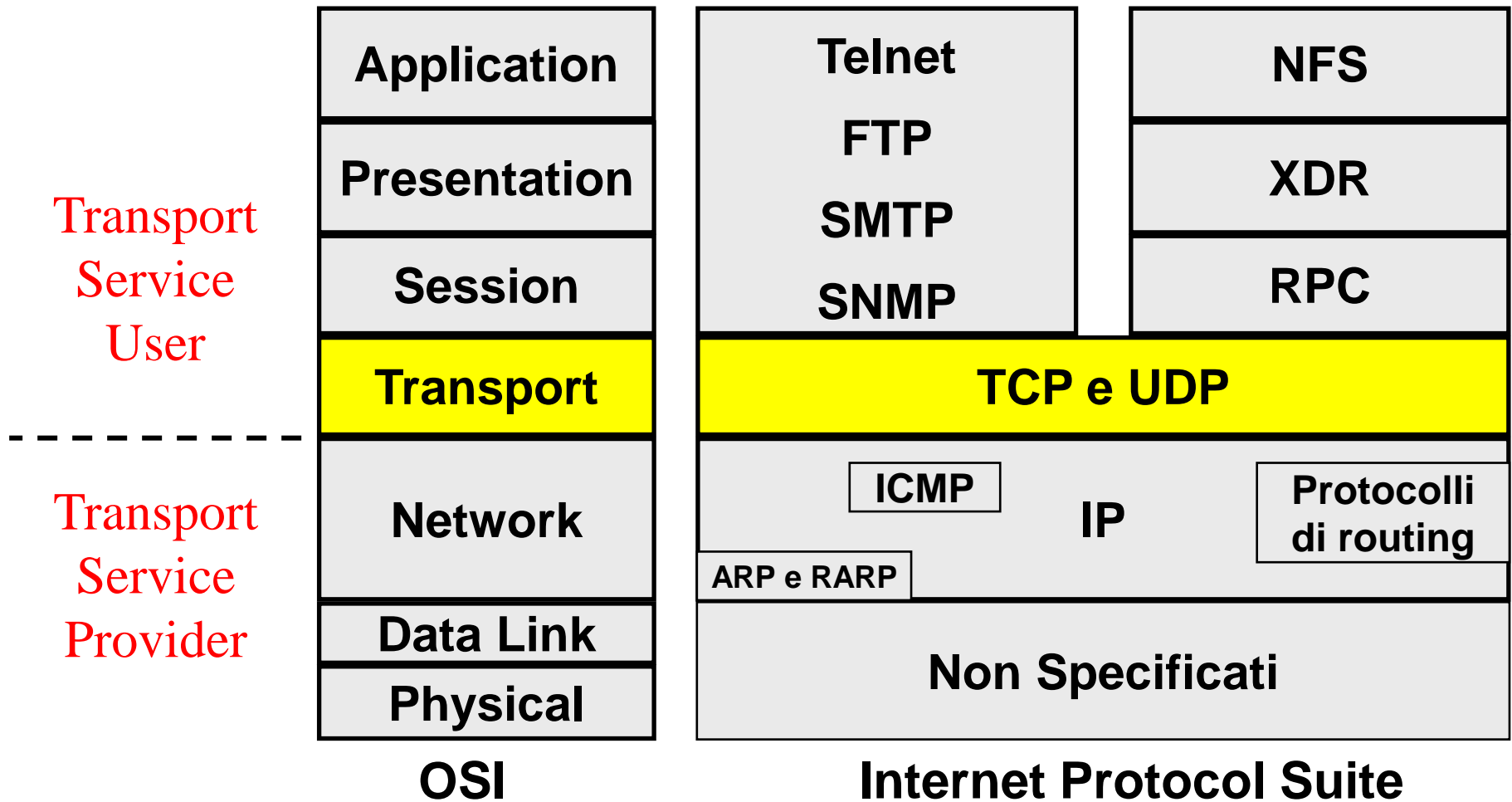


# Connection-oriented demux: Threaded Web Server





# 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

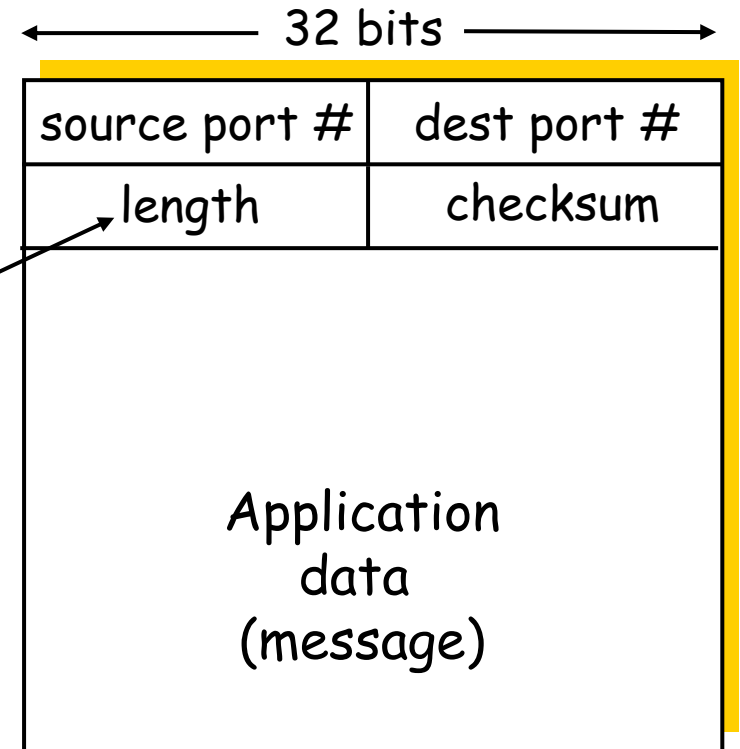




# UDP: User Datagram Protocol - 3

- Ampiammente 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

Lunghezza  
in bytes del  
segmento UDP,  
header incluso



Formato di un segmento UDP



# Checksum UDP

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

## Mittente:

- Tratta il contenuto del segmento come una sequenza di interi espressi su 16 bit
- **checksum:** complemento ad 1 della somma (in complemento ad 1) dei numeri da 16 bit che costituiscono il segmento UDP
  - Incluso uno pseudoheader che include gli indirizzi IP sorgente e destinazione
- Il mittente pone il valore della checksum nel campo checksum del segmento UDP

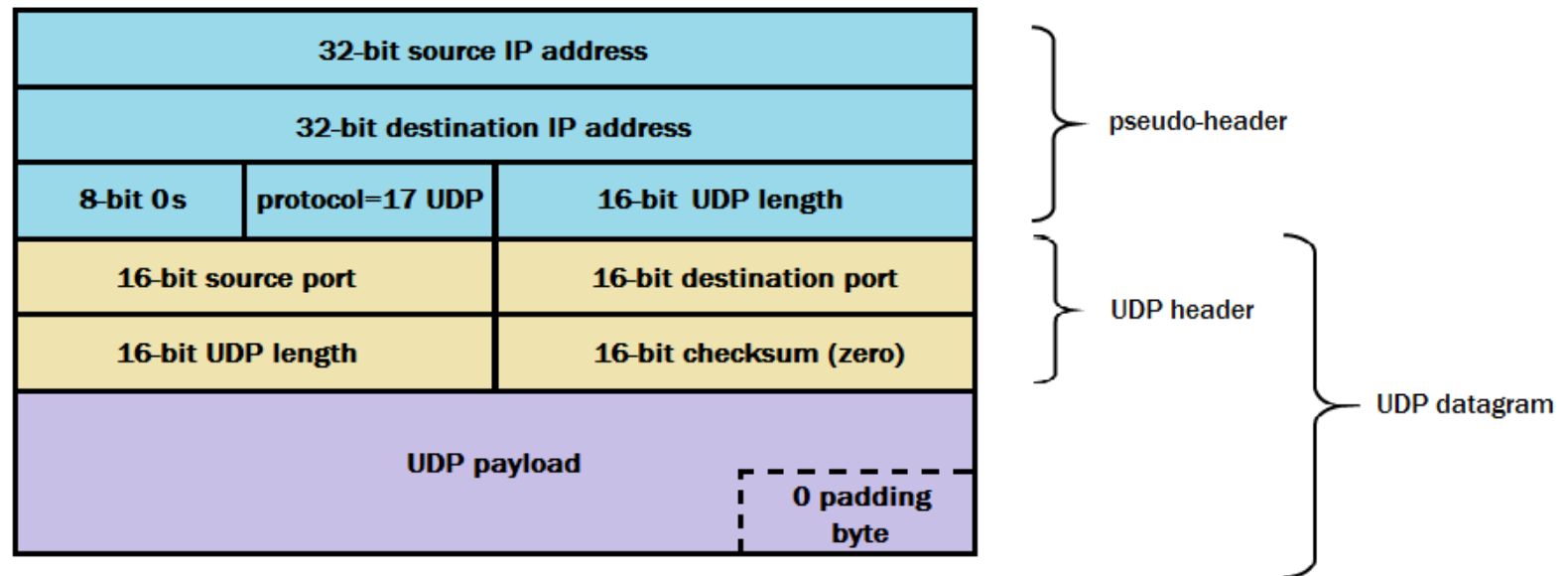
## Ricevente:

- calcola la somma in complemento ad 1 dei campi del segmento ricevuto compresa la checksum
- Risultato composto da tutti 1 ?
  - NO: errore!
  - SI: nessun errore rilevato ...
    - ... il che non vuol dire che non vi siano stati errori ...



# UDP checksum: pseudo-header

- La checksum viene calcolata antepoendo al datagramma UDP uno pseudo-header che contiene gli indirizzi IP sorgente e destinazione e la lunghezza del datagramma



- Lo pseudo-header (come dice il nome) viene aggiunto ai soli fini del calcolo della checksum, non viene trasmesso



# UDP checksum: calcolo in C (1/2)

```
unsigned short int udp_checksum(const void *buf, int len, in_addr_t src_addr, in_addr_t dest_addr)
{
    const unsigned short int *buf=buf;

    unsigned short int *ip_src=(void *)&src_addr, *ip_dst=(void *)&dest_addr;
    unsigned short int sum;
    size_t length=len;

    // Calculate the sum
    sum = 0;
    while (len > 1)
    {
        sum += *buf++;
        if (sum & 0x80000000)
            sum = (sum & 0xFFFF) + (sum >> 16);
        len -= 2;
    }

    // Add the padding if the packet lenght is odd
    if ( len & 1 )
        sum += *((unsigned char *)buf);

    // Add the pseudo-header
    ...
}
```



# UDP checksum: calcolo in C (2/2)

```
...  
// Add the pseudo-header  
sum += *(ip_src++);  
sum += *ip_src;  
  
sum += *(ip_dst++);  
sum += *ip_dst;  
  
sum += htons(IPPROTO_UDP);  
sum += htons(length);  
  
// Add the carries  
while (sum >> 16)  
    sum = (sum & 0xFFFF) + (sum >> 16);  
  
// Return the one's complement of sum  
return ( (unsigned short int)(~sum) );  
}
```