Livello di trasporto:

Gaia Maselli
maselli@di.uniroma1.it

Livello di trasporto

Obiettivi:

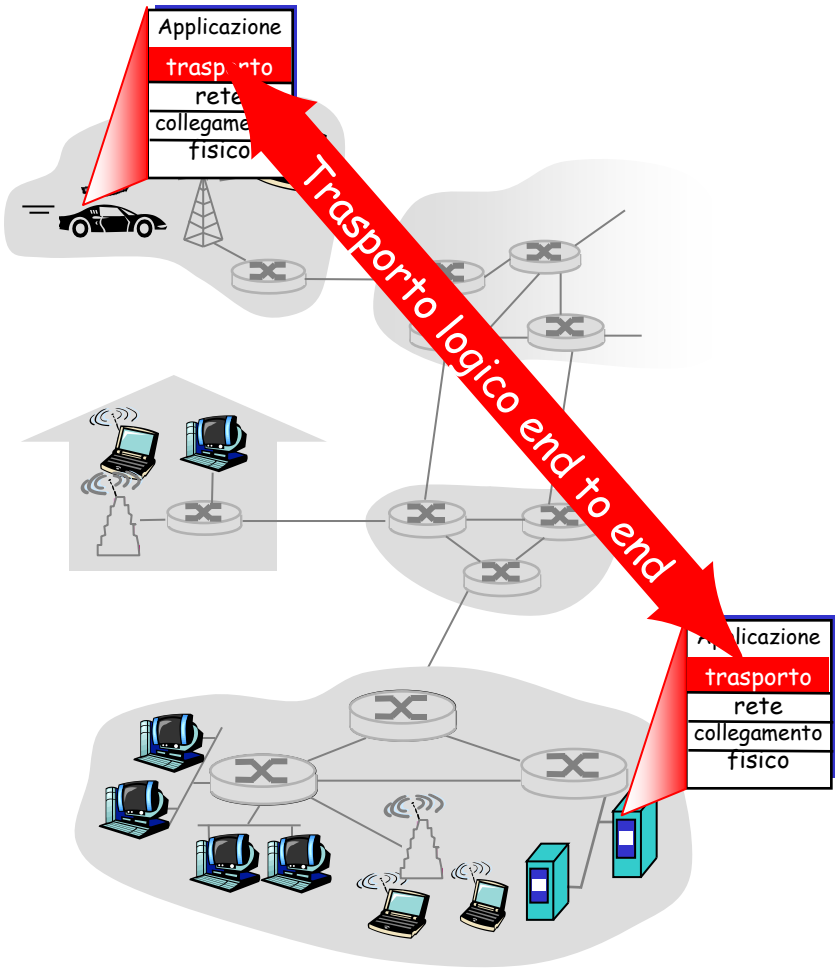
- Capire i principi che sono alla base dei servizi del livello di trasporto:
  - multiplexing/demultiplexing
  - trasferimento dati affidabile
  - controllo di flusso
  - controllo di congestione
- Descrivere i protocolli del livello di trasporto di Internet:
  - UDP: trasporto senza connessione
  - TCP: trasporto orientato alla connessione
  - controllo di congestione TCP

Livello di trasporto

- Servizi a livello di trasporto
- Multiplexing e demultiplexing
- Trasporto senza connessione: UDP

Servizi e protocolli di trasporto

- Forniscono la *comunicazione logica* tra processi applicativi di host differenti
- Comunicazione logica: gli host eseguono i processi come se fossero direttamente connessi (in realtà possono trovarsi agli antipodi del pianeta)
- I protocolli di trasporto vengono eseguiti nei sistemi terminali
  - lato invio: scinde i messaggi in segmenti e li passa al livello di rete
  - lato ricezione: riassembla i segmenti in messaggi e li passa al livello di applicazione



Relazione tra livello di trasporto e livello di rete

- *livello di trasporto:* comunicazione logica tra *processi*
  - si basa sui servizi del livello di rete e li potenzia
- *livello di rete:* comunicazione logica tra *host*
  - si basa sui servizi del livello di collegamento

Analogia con la posta ordinaria:

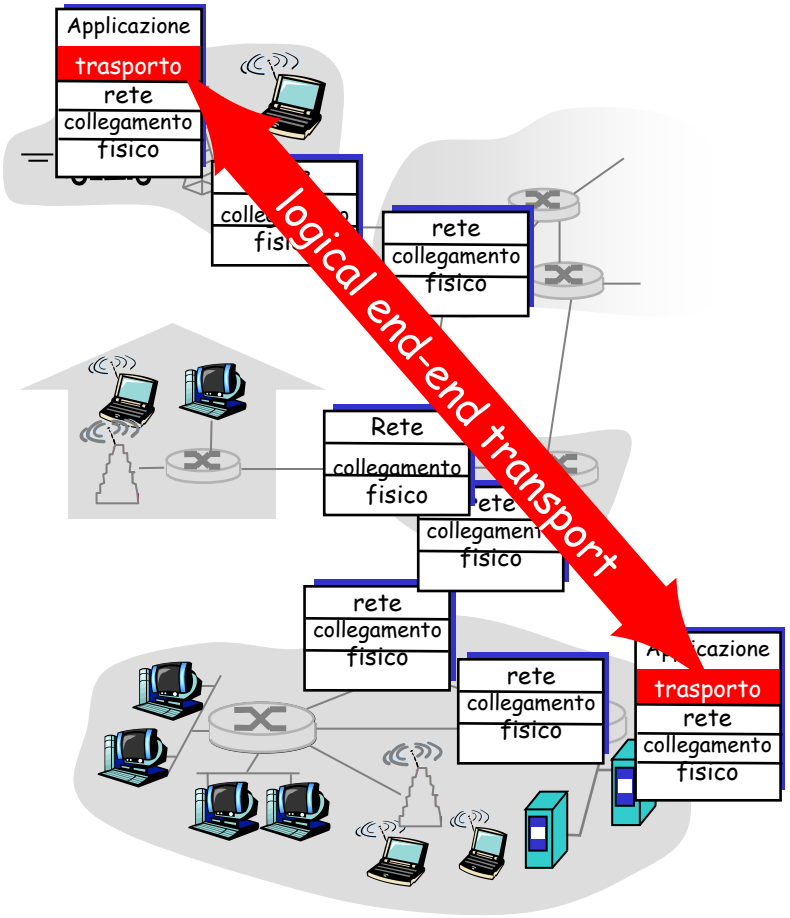
12 ragazzi di un condominio inviano lettere a 12 ragazzi di un altro condominio

- processi = ragazzi
- messaggi delle applicazioni = lettere nelle buste
- host = condomini
- protocollo di trasporto = portieri dei condomini
- protocollo del livello di rete = servizio postale

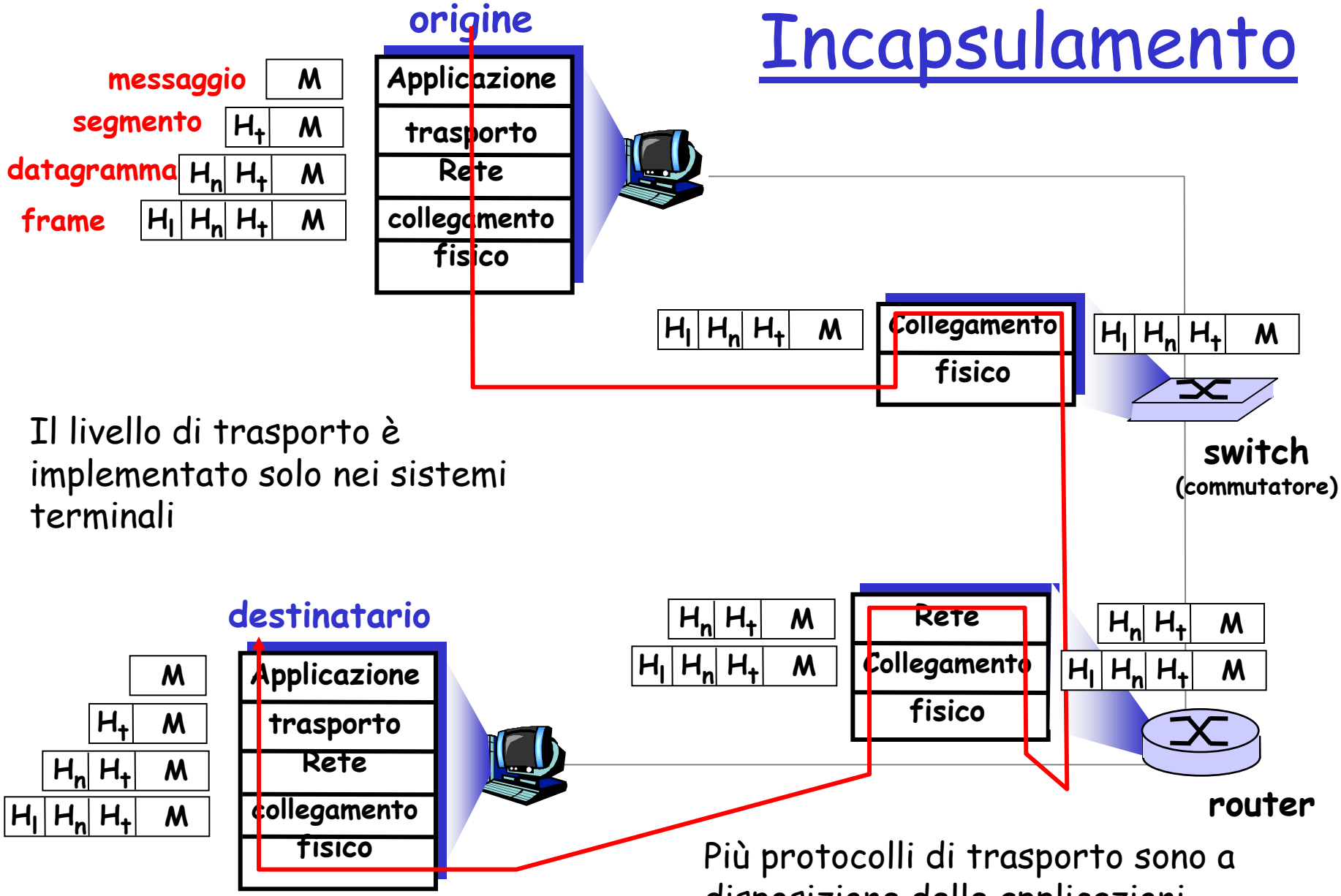
N.B. i portieri svolgono il proprio lavoro localmente, non sono coinvolti nelle tappe intermedie delle lettere

Protocolli del livello di trasporto in Internet

- Il compito del servizio di trasporto è la consegna **process-to-process** ovvero "tra processi in esecuzione tra sistemi terminali diversi"
- Affidabile, consegne nell'ordine originario (TCP)
  - controllo di congestione
  - controllo di flusso
  - setup della connessione
- Inaffidabile, consegne senz'ordine: UDP
  - estensione senza fronzoli del servizio di consegna best effort
- Servizi non disponibili:
  - garanzia su ritardi
  - garanzia su ampiezza di banda



Incapsulamento



Il livello di trasporto è implementato solo nei sistemi terminali

Più protocolli di trasporto sono a disposizione delle applicazioni Internet: TCP e UDP

Livello di trasporto

- Servizi a livello di trasporto
- Multiplexing e demultiplexing
- Trasporto senza connessione: UDP

Come il servizio di trasporto da host a host fornito dal livello di rete possa diventare un servizio di trasporto da processo a processo per le applicazione in esecuzione sugli host

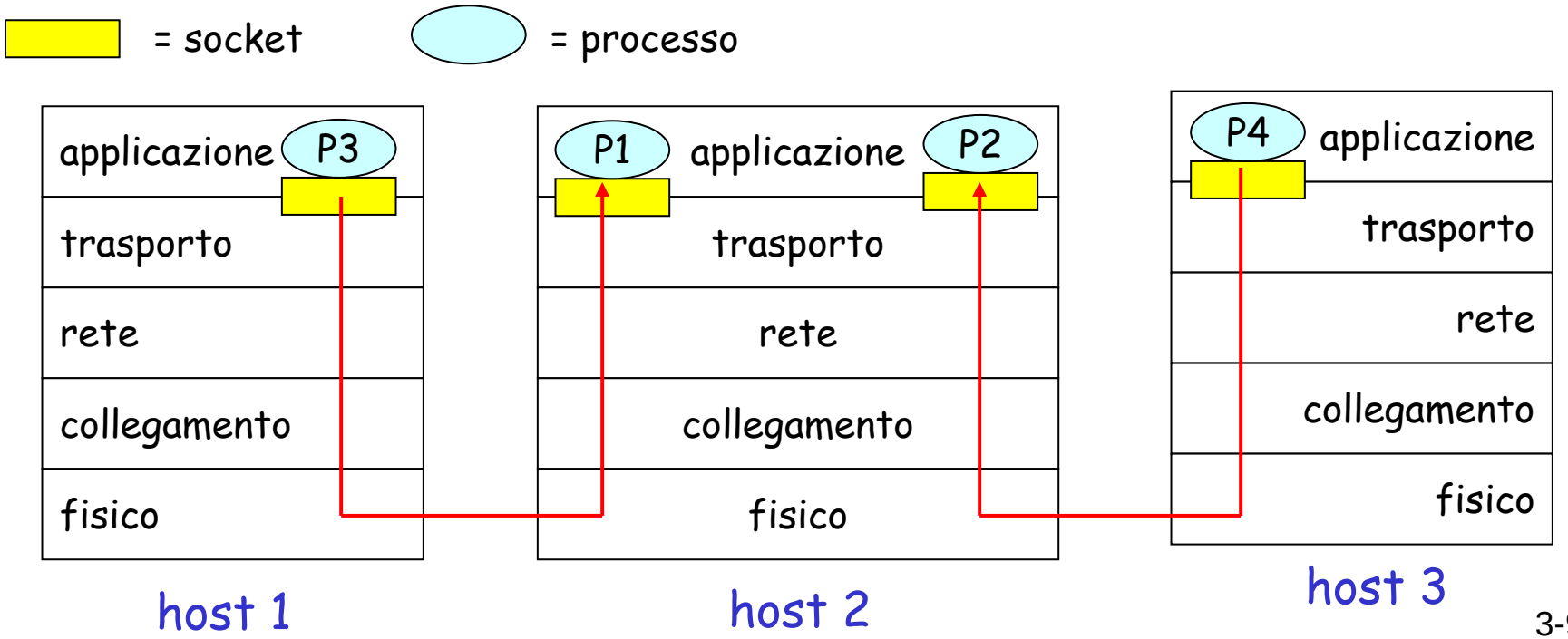
Demultiplexing

Esempio: su un host ci sono due processi in esecuzione: P1= FTP, P2= HTTP

- Quando il livello di trasporto dell'host riceve i dati dal livello di rete sottostante, deve indirizzare i dati a uno di questi processi. Quale? Come?
- Informazioni all'interno dell'header

Demultiplexing nell'host ricevente:

consegnare i segmenti ricevuti alla socket appropriata



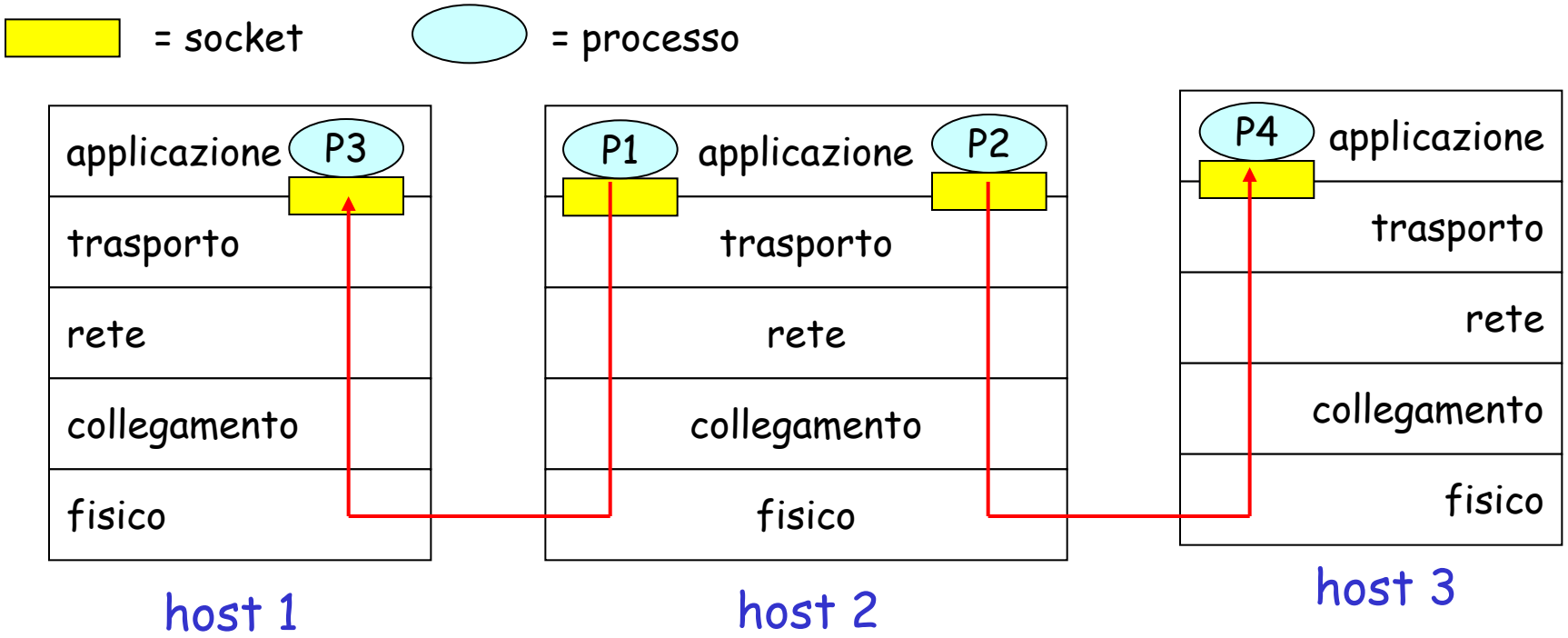
Multiplexing

Esempio: su un host ci sono due processi in esecuzione: P1= FTP, P2= HTTP

- L'host deve anche raccogliere i dati in uscita da queste socket e passarli al livello di rete

Multiplexing nell'host mittente:

raccogliere i dati da varie socket, incapsularli con l'intestazione (utilizzata poi per il demultiplexing)



Esempio dei condomini

- I portieri effettuano un'operazione di
  - Multiplexing quando raccolgono le lettere dai condomini (mittenti) e le imbucano
  - Demultiplexing quando ricevono le lettere dal postino , leggono il nome riportato su ciascuna busta e consegnano ciascuna lettera al rispettivo destinatario

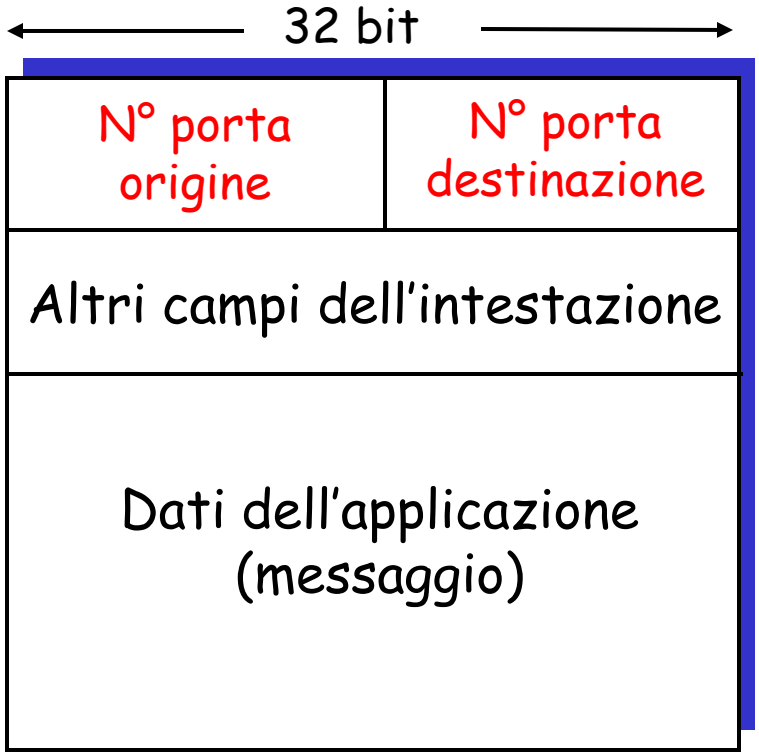
Come funziona il demultiplexing

L'host riceve i datagrammi IP

- ogni datagramma ha un indirizzo IP di origine e un indirizzo IP di destinazione
- ogni datagramma trasporta 1 segmento a livello di trasporto
- ogni segmento ha un numero di porta di origine e un numero di porta di destinazione

L'host usa gli indirizzi IP e i numeri di porta per inviare il segmento alla socket appropriata

Campo n° porta: 16 bit con valori da 0 a 65535 (fino a 1023, *well known-port numer*)



Struttura del segmento TCP/UDP

esempio

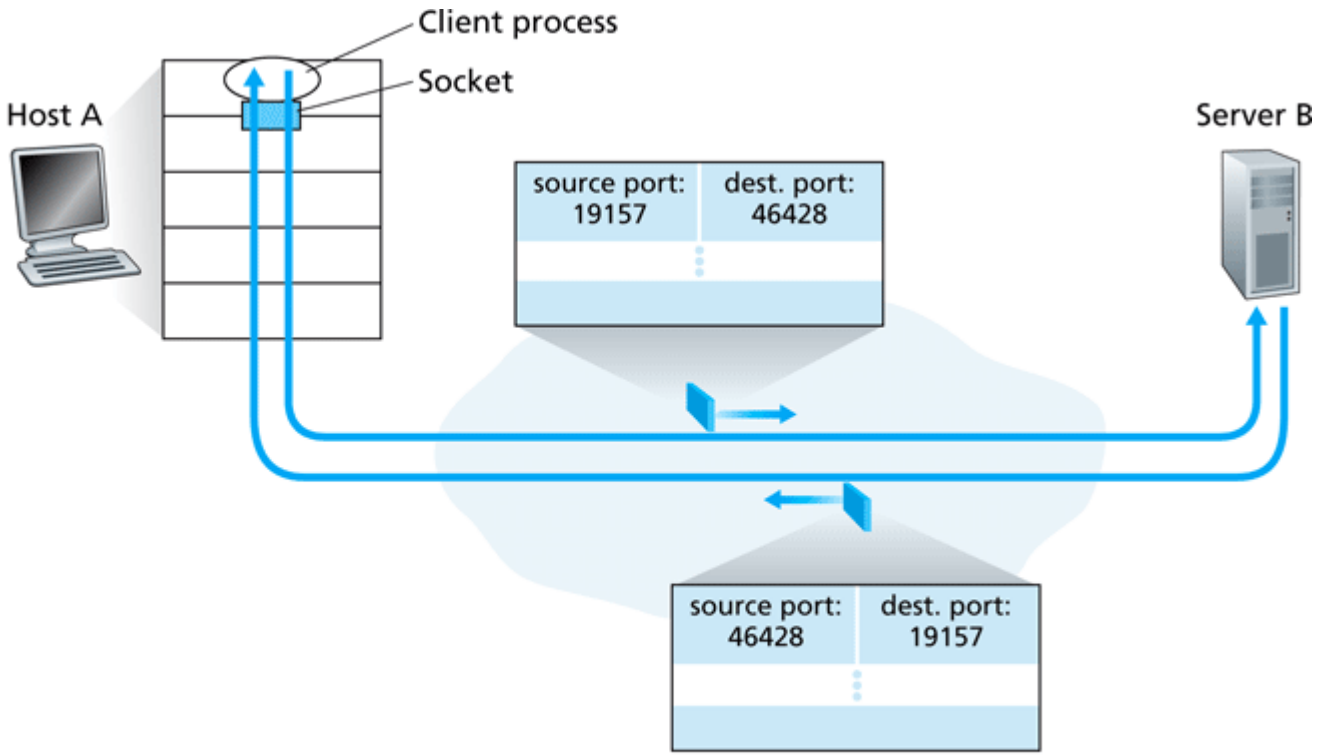


Figure 3.4 ♦ The inversion of source and destination port numbers

Demultiplexing senza connessione

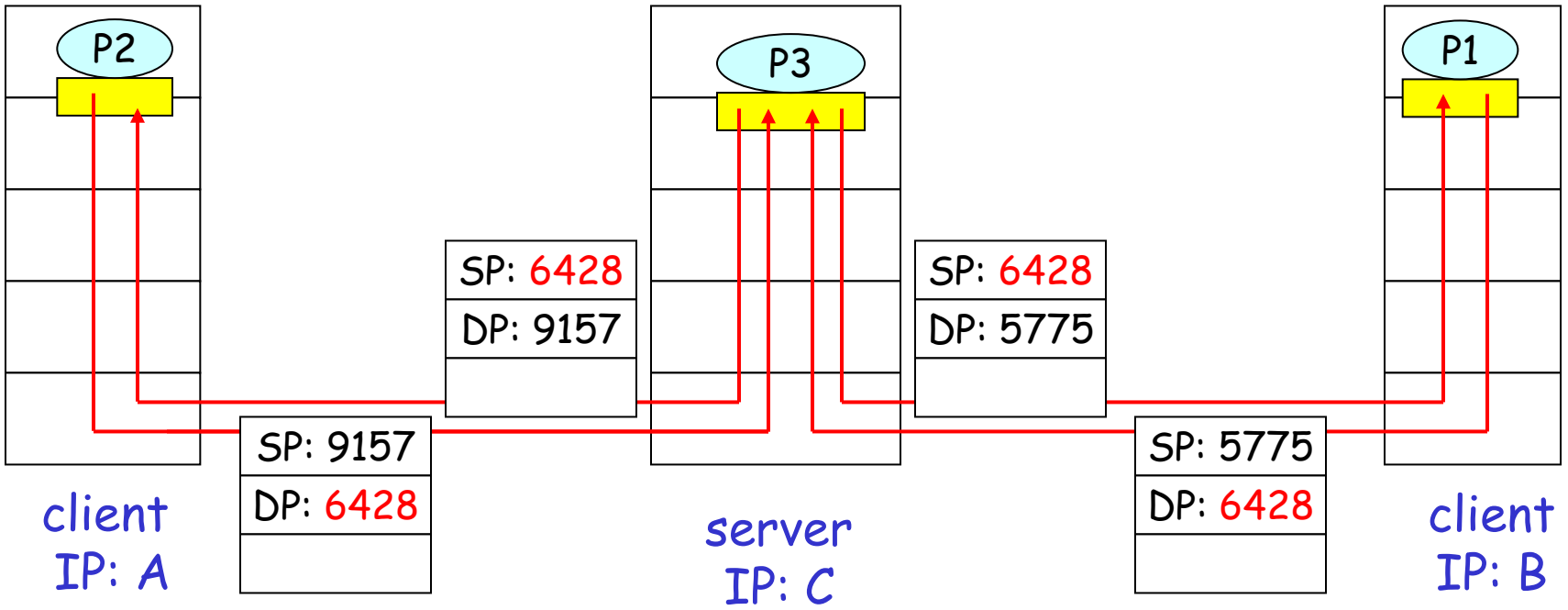
- Crea le socket con i numeri di porta:
- `DatagramSocket mySocket1 = new DatagramSocket(12534);`
- La socket UDP è identificata da 2 parametri:

  (indirizzo IP di destinazione, numero della porta di destinazione)

- Quando l'host riceve il segmento UDP:
  - controlla il numero della porta di destinazione nel segmento
  - invia il segmento UDP alla socket con quel numero di porta
- Datagrammi IP con indirizzi IP di origine e/o numeri di porta di origine differenti, ma con lo stesso IP e porta di destinazione vengono inviati alla stessa socket

Demultiplexing senza connessione (cont.)

```
DatagramSocket serverSocket = new DatagramSocket(6428);
```

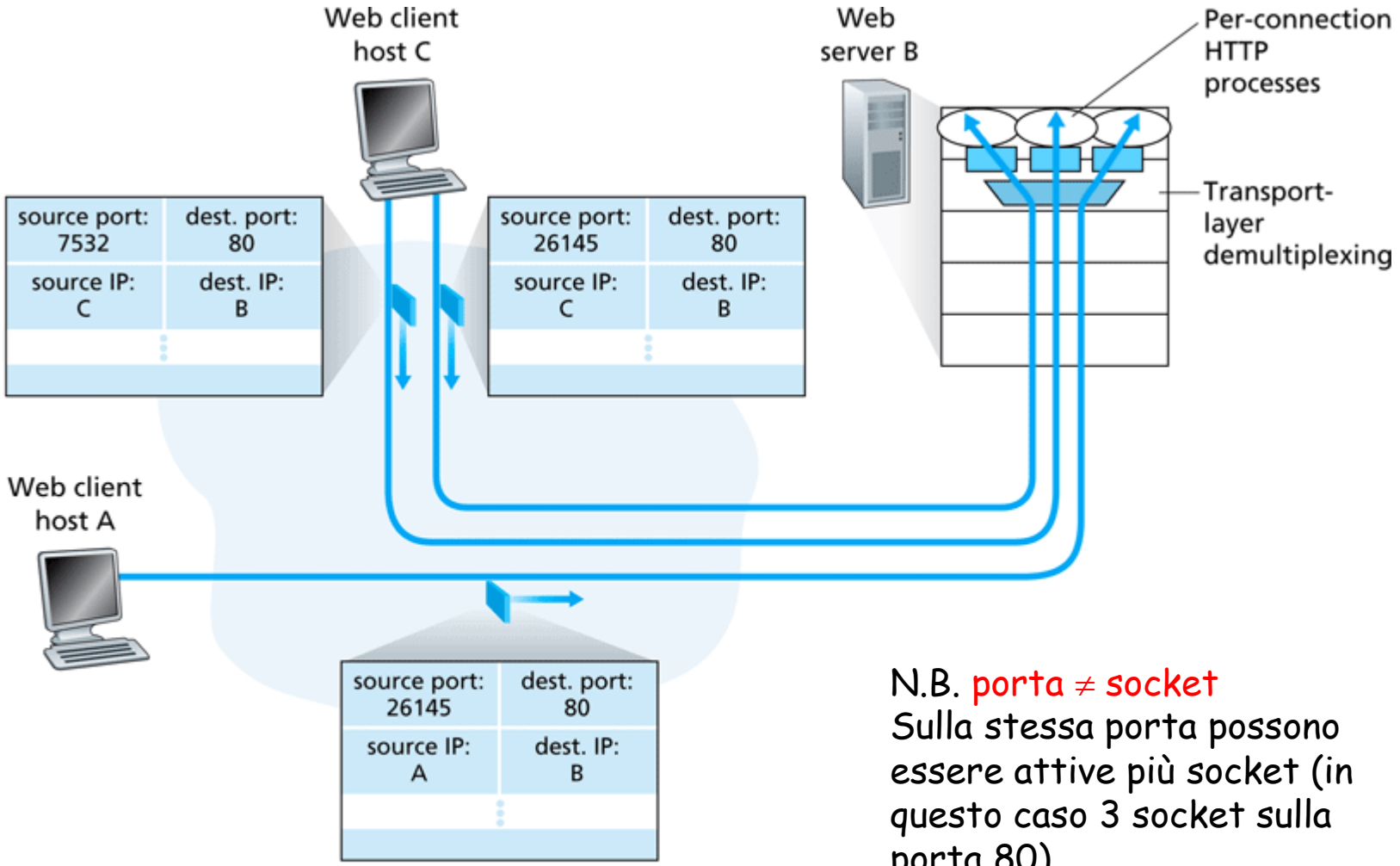


SP fornisce "l'indirizzo di ritorno"

Demultiplexing orientato alla connessione

- La socket TCP è identificata da 4 parametri:
  - indirizzo IP di origine
  - numero di porta di origine
  - indirizzo IP di destinazione
  - numero di porta di destinazione
- L'host ricevente usa i quattro parametri per inviare il segmento alla socket appropriata
- Un host server può supportare più socket TCP contemporanee:
  - ogni socket è identificata dai suoi 4 parametri
- I server web hanno socket differenti per ogni connessione client
  - con HTTP non-persistente si avrà una socket differente anche per ogni richiesta dallo stesso client

Demultiplexing orientato alla connessione (continua)



N.B. porta ≠ socket
Sulla stessa porta possono essere attive più socket (in questo caso 3 socket sulla porta 80).

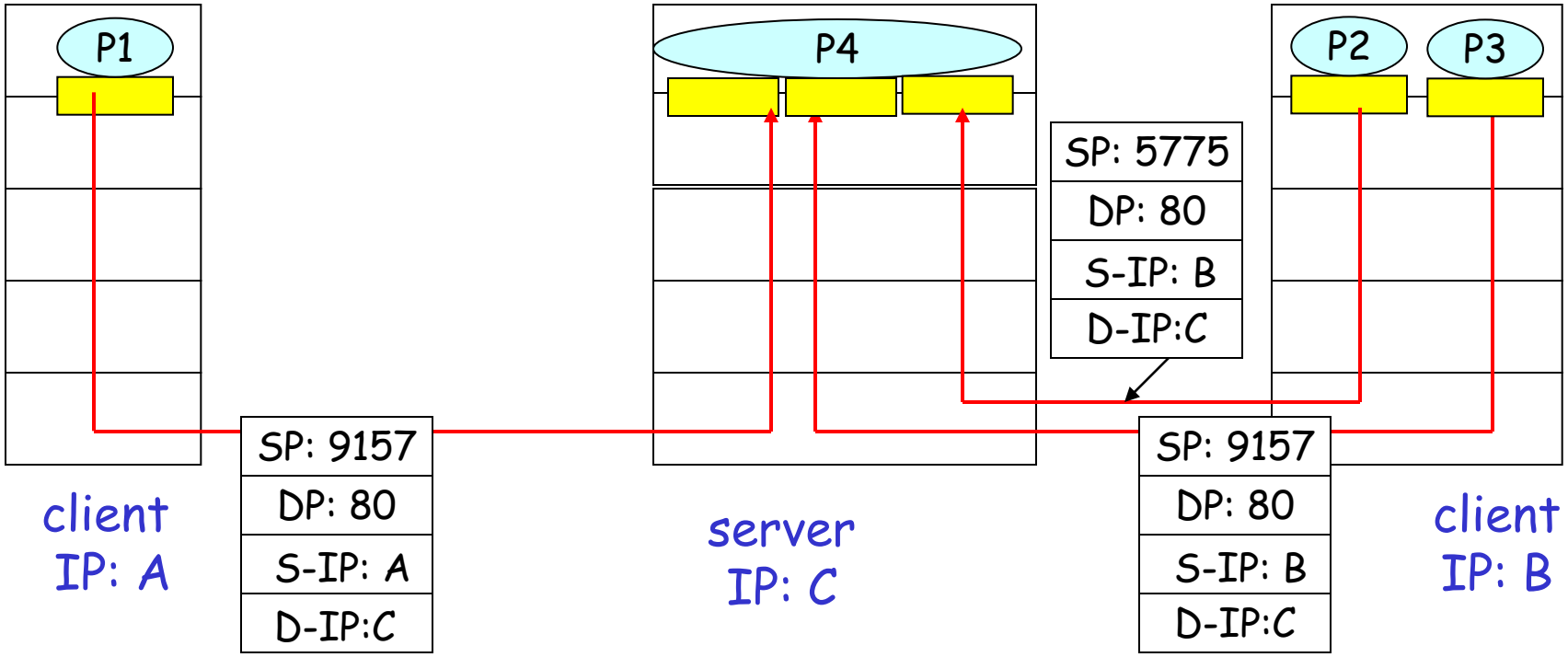
In Java

- Client crea una connessione con il server
  - `Socket clientSocket = new Socket("hostname", 6789);`
  - 6789 è la porta di destinazione, quella di origine non viene specificata (è assegnata automaticamente e inserita nella richiesta)
- Server:
  - `ServerSocket welcomeSocket = new ServerSocket(6789);`
  - Si mette in attesa di richieste dai client
  - Ricevuta la richiesta del client crea una nuova connessione
  - `Socket connectionSocket = welcomeSocket.accept();`
  - Torna in attesa di richieste dal client

- Per ogni richiesta che riceve stabilisce una nuova connessione creando una nuova socket ma la porta rimane la stessa (6789)

N.B. Java è solo un esempio… Altri linguaggi (es. C) forniscono librerie per l'utilizzo delle socket (implementano l'interfaccia con il livello di trasporto)

Demultiplexing orientato alla connessione: thread dei server web



Capitolo 3: Livello di trasporto

- Servizi a livello di trasporto
- Multiplexing e demultiplexing
- Trasporto senza connessione: UDP

UDP: User Datagram Protocol [RFC 768]

- Protocollo di trasporto "senza fronzoli"
- Servizio di consegna "a massimo sforzo" (best effort), i segmenti UDP possono essere:
  - perduti
  - consegnati fuori sequenza all'applicazione
- *Senza connessione:*
  - no handshaking tra mittente e destinatario UDP
  - ogni segmento UDP è gestito indipendentemente dagli altri

Perché esiste UDP?

- Nessuna connessione stabilita (che potrebbe aggiungere un ritardo)
- Semplice: nessuno stato di connessione nel mittente e destinatario
- Intestazioni di segmento corte
- Senza controllo di congestione: UDP può sparare dati a raffica

DNS usa UDP

- Quando vuole effettuare una query, DNS costruisce un messaggio di query e lo passa a UDP
- L'entità UDP aggiunge i campi di intestazione al messaggio e trasferisce il segmento risultante al livello di rete, etc.
- L'applicazione DNS aspetta quindi una risposta
- Se non ne riceve tenta di inviarla a un altro server dei nomi oppure informa l'applicazione
- La semplicità della richiesta/risposta (molto breve) motiva l'utilizzo di UDP, che risulta più veloce
  - Nessuna connessione stabilita
  - Nessuno stato di connessione
  - Intestazioni di pacchetto più corte

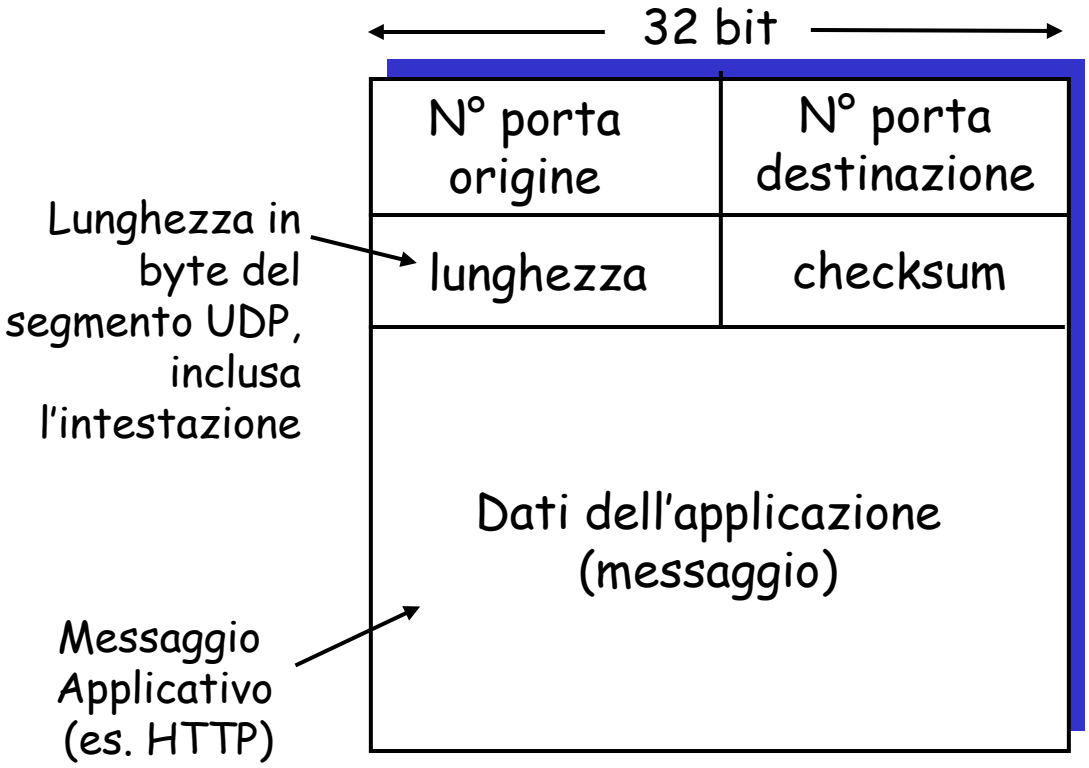
- UDP è utilizzato anche perché consente un controllo più sottile a livello di applicazione su quali dati sono inviati e quando

UDP: ulteriori informazioni

- Utilizzato spesso nelle applicazioni multimediali
  - tollera piccole perdite
  - sensibile alla frequenza
- Altri impieghi di UDP
  - SNMP

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

Struttura pacchetto UDP



Checksum UDP

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

Mittente:

- Tratta il contenuto del segmento come una sequenza di interi da 16 bit
- checksum: complemento a 1 della somma di tutte le parole di 16 bit del segmento
- Il mittente pone il valore della checksum nel campo checksum del segmento UDP

Ricevente:

- calcola la le parole ricevute con la checksum del segmento ricevuto
- Risultato
  - = 1111111111111111→ nessun errore rilevato. (*Ma potrebbero esserci errori nonostante questo?* Lo scopriremo più avanti …)
  - ≠ 1111111111111111 → errore

Esempio di checksum

- Nota
  - Quando si sommano i numeri, un riporto dal bit più significativo deve essere sommato al risultato
- Esempio: sommare due interi da 16 bit