Università degli Studi di Roma "Tor Vergata"
Laurea in Informatica

Sistemi Operativi e Reti
(modulo Reti)
a.a. 2024/2025

Livello di trasporto

dr. Manuel Fiorelli
manuel.fiorelli@uniroma2.it
https://art.uniroma2.it/fiorelli

Basate sulle slide del libro di testo:
https://gaia.cs.umass.edu/kurose_ross/ppt.php

Livello di trasporto: panoramica

Obiettivi:

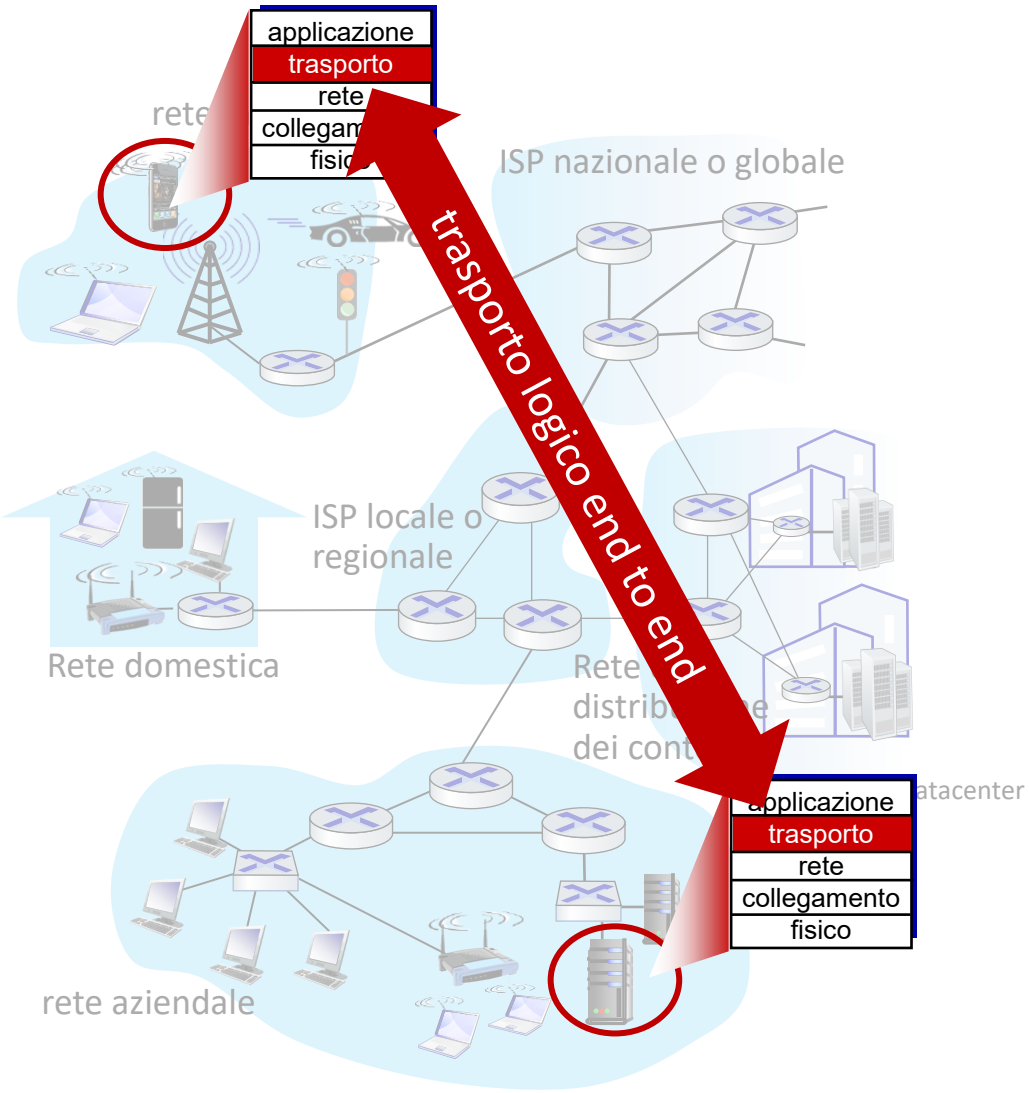
- capire i principi che sono alla base dei servizi del livello di trasporto:
  - multiplexing, demultiplexing
  - trasferimento dati affidabile
  - controllo di flusso
  - controllo della congestione
- conoscere i protocolli del livello di trasporto di Internet:
  - UDP: trasporto senza connessione
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

Livello di trasporto: tabella di marcia

- Servizi a livello di trasporto
- Multiplexing e demultiplexing
- Trasporto senza connessione: UDP
- Principi del trasferimento dati affidabile
- Trasporto orientato alla connessione: TCP
- Principi del controllo della congestione
- Controllo della congestione TCP
- Evoluzione della funzionalità del livello di trasporto

Servizi e protocolli di trasporto

- Forniscono la *comunicazione logica* tra processi applicativi di host differenti
- I protocolli di trasporto vengono eseguiti nei sistemi periferici:
  - lato invio: scinde (se necessario) i messaggi dell'applicazione, combinando ciascuna parte con un'intestazione per creare un *segmento* e lo passa al livello di rete
  - lato ricezione: estrae i dati contenuti del segmento e li passa al livello di applicazione
- I router nel cammino da un host all'altro operano solo sull'intestazione del datagramma, ignorando il segmento incapsulato al suo interno
- Più protocolli di trasporto sono a disposizione delle applicazioni
  - TCP, UDP



Relazione tra livello di trasporto e livello di rete

Analogia domestica:

12 ragazzi a casa di Ann inviano lettere a 12 ragazzi a casa di Bill:

- host = case
- processi = ragazzi
- messaggi delle applicazioni= lettere nelle buste

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*

➡ multiplexing/demultiplexing

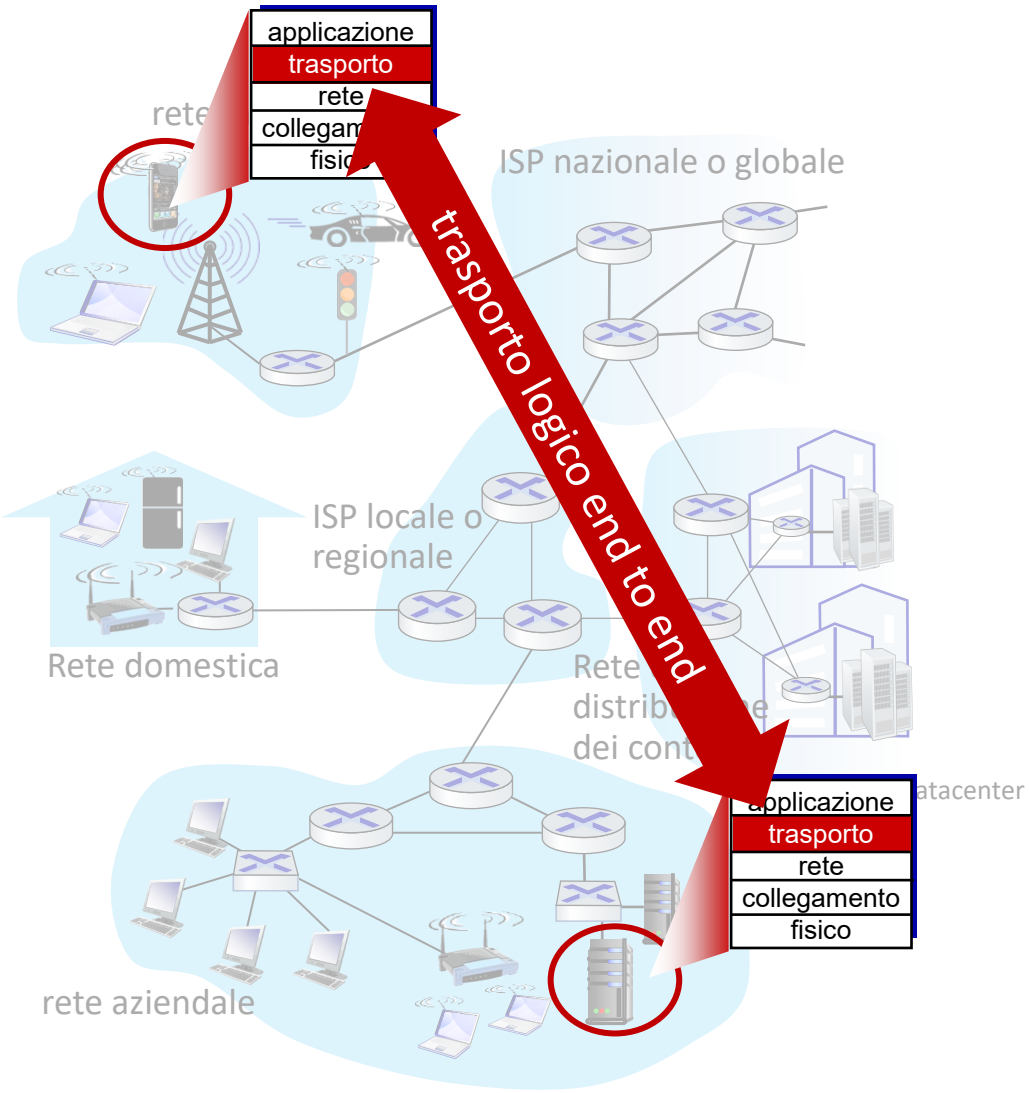
Analogia domestica:

12 ragazzi a casa di Ann inviano lettere a 12 ragazzo a casa do Bill:

- host = case
- processi = ragazzi
- messaggi delle applicazioni= lettere nelle buste
- protocollo di trasporto = Ann e Bill che raccolgono e distribuiscono la posta tra i fratelli
- protocollo a livello di rete = servizio postale (compresi i postini)

Protocolli del livello di trasporto in Internet

- **UDP:** User Datagram Protocol
  - inaffidabile, consegne senz'ordine
  - estensione "senza fronzoli" di IP: solo comunicazione tra processi e controllo degli errori
- **TCP:** Transmission Control Protocol
  - comunicazione tra processi affidabile, consegne nell'ordine originario
  - controllo di flusso
  - controllo della congestione
  - instaurazione della connessione
- servizi *non* disponibili (impossibile realizzarli a livello di trasporto date le caratteristiche del servizio di rete di Internet):
  - garanzie su ritardi
  - garanzie su banda



Capitolo 3: tabella di marcia

- Servizi a livello di trasporto
- Multiplexing e demultiplexing
- Trasporto senza connessione: UDP
- Principi del trasferimento dati affidabile
- Trasporto orientato alla connessione: TCP
- Principi del controllo della congestione
- Controllo della congestione TCP
- Evoluzione della funzionalità del livello di trasporto

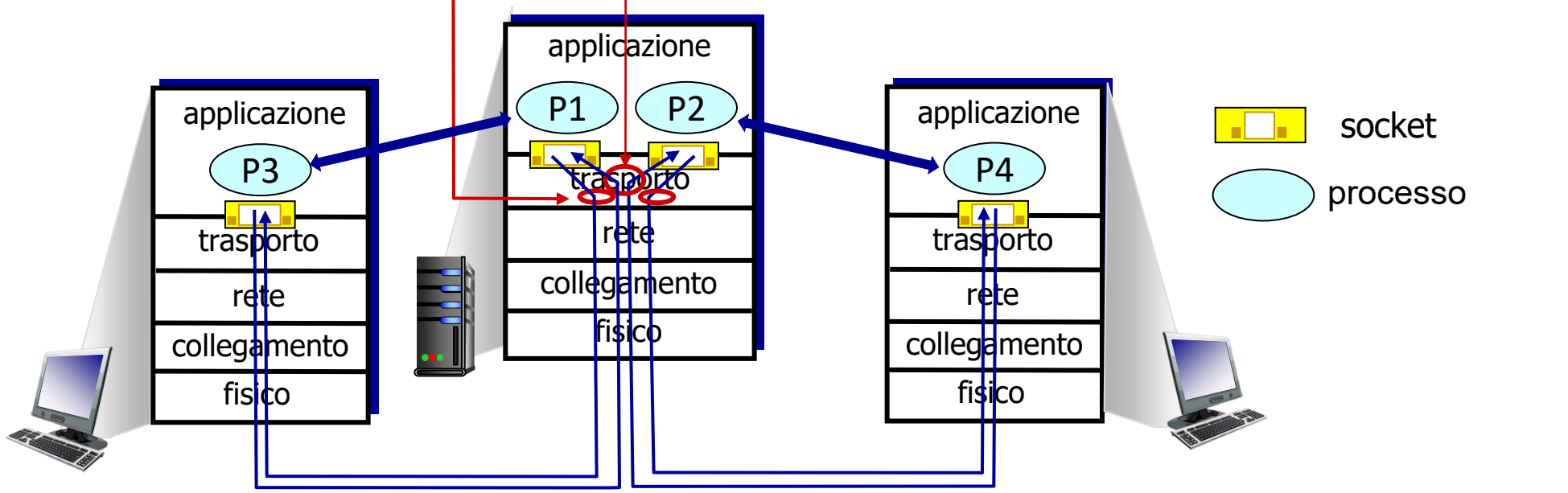
Multiplexing/demultiplexing

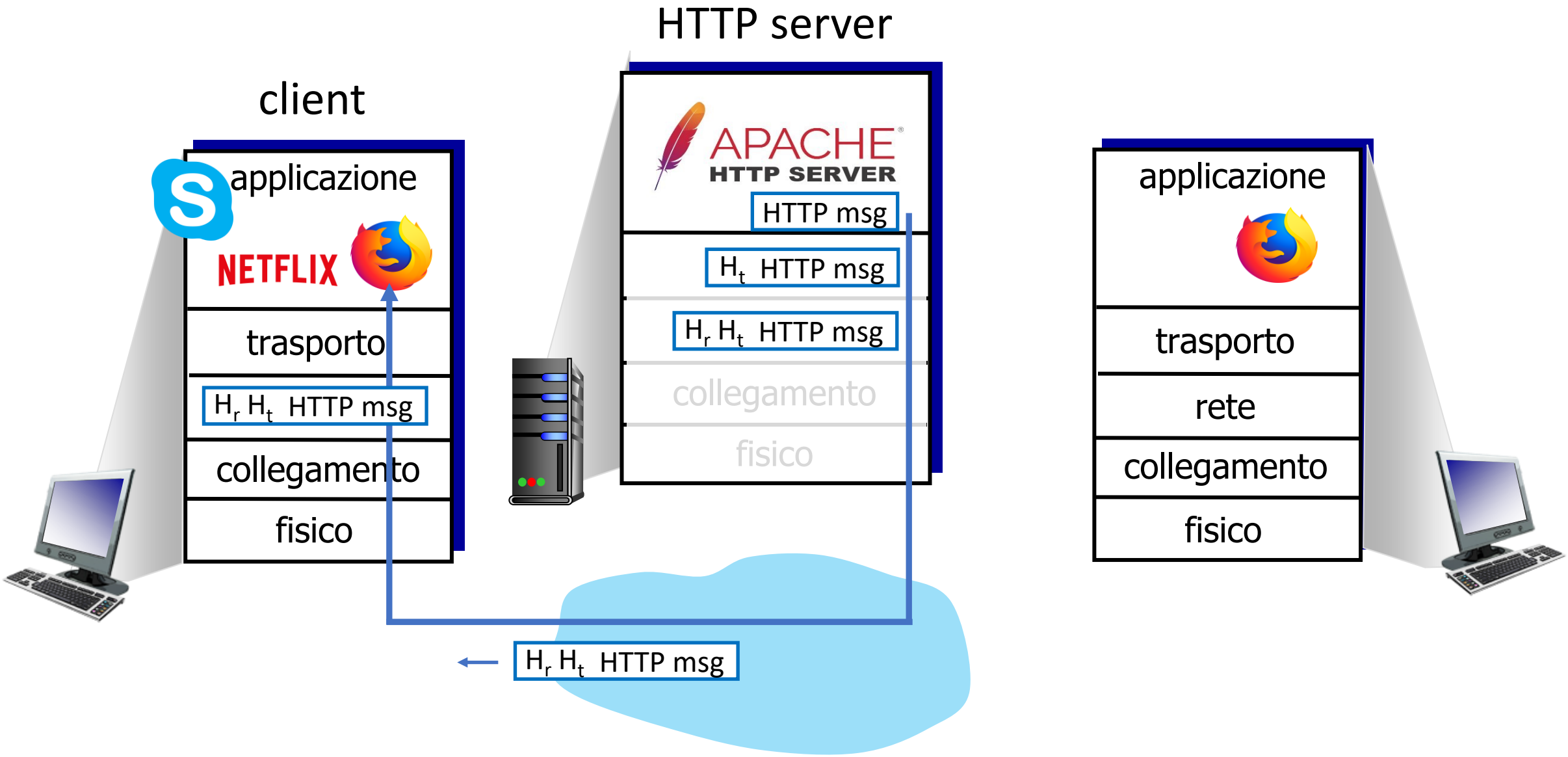
Multiplexing lato mittente:

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

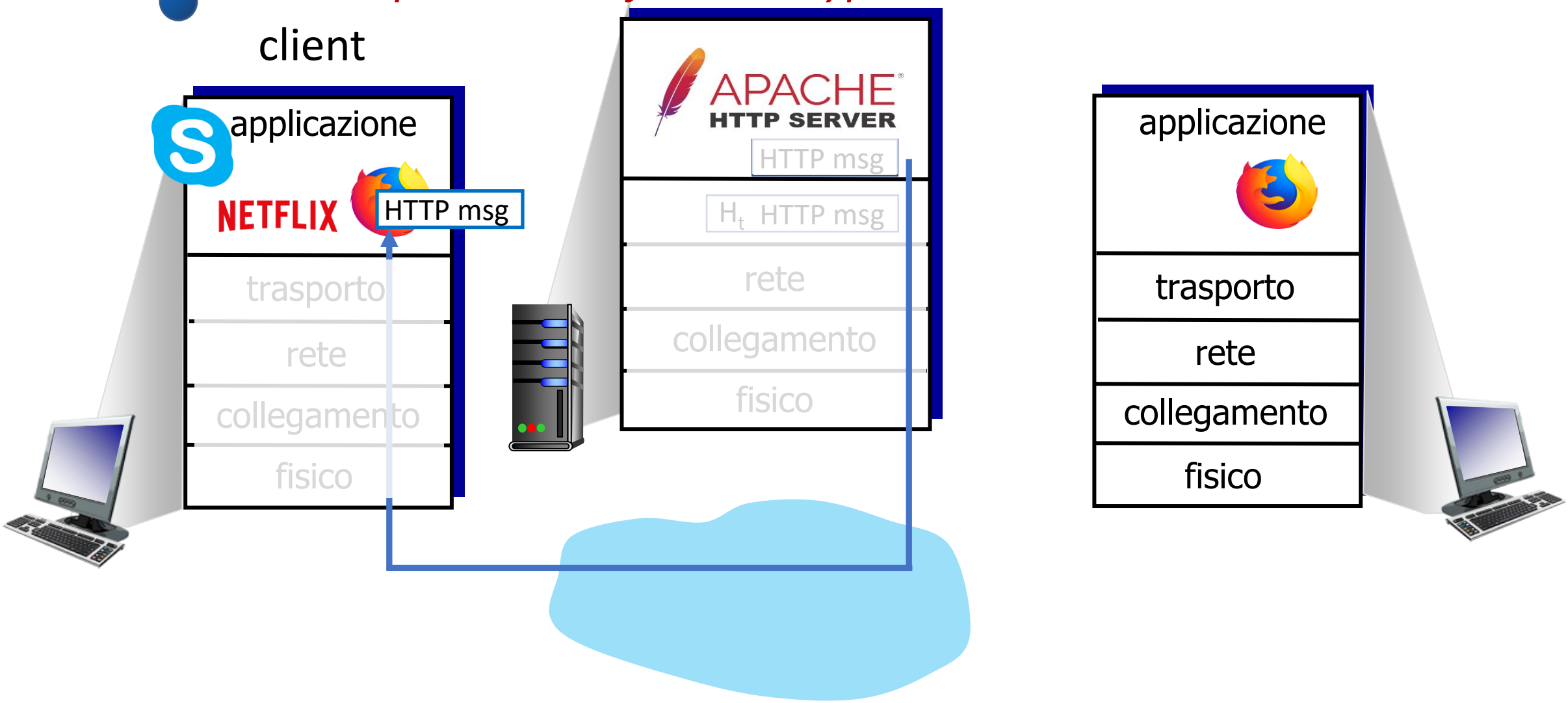
demultiplexing lato ricevente:

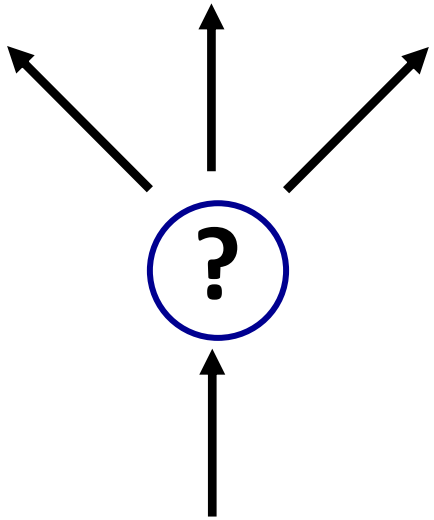
utilizzare le informazioni nell'intestazione per consegnare i segmenti ricevuti alla socket corretta



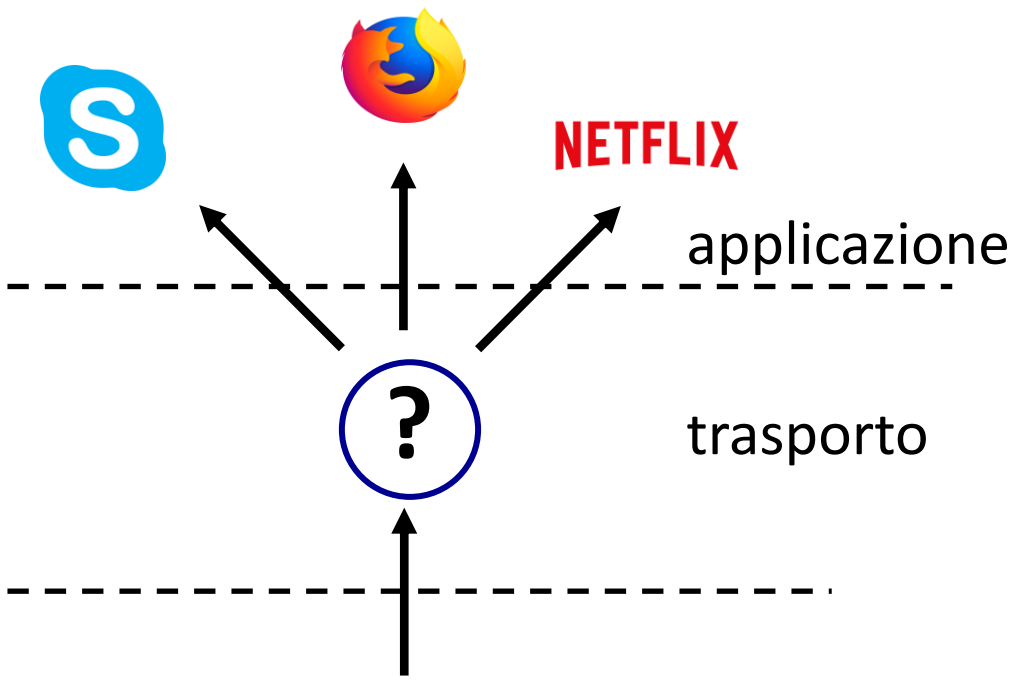

client
HTTP server
applicazione
NETFLIX
trasporto
H_r H_t HTTP msg
collegamento
fisico
APACHE
HTTP SERVER
HTTP msg
H_t HTTP msg
H_r H_t HTTP msg
collegamento
fisico
applicazione
trasporto
rete
collegamento
fisico
H_r H_t HTTP msg

D: come ha fatto il livello di trasporto a sapere di dover consegnare il messaggio al processo del browser Firefox piuttosto che a quello di Netflix o di Skype?





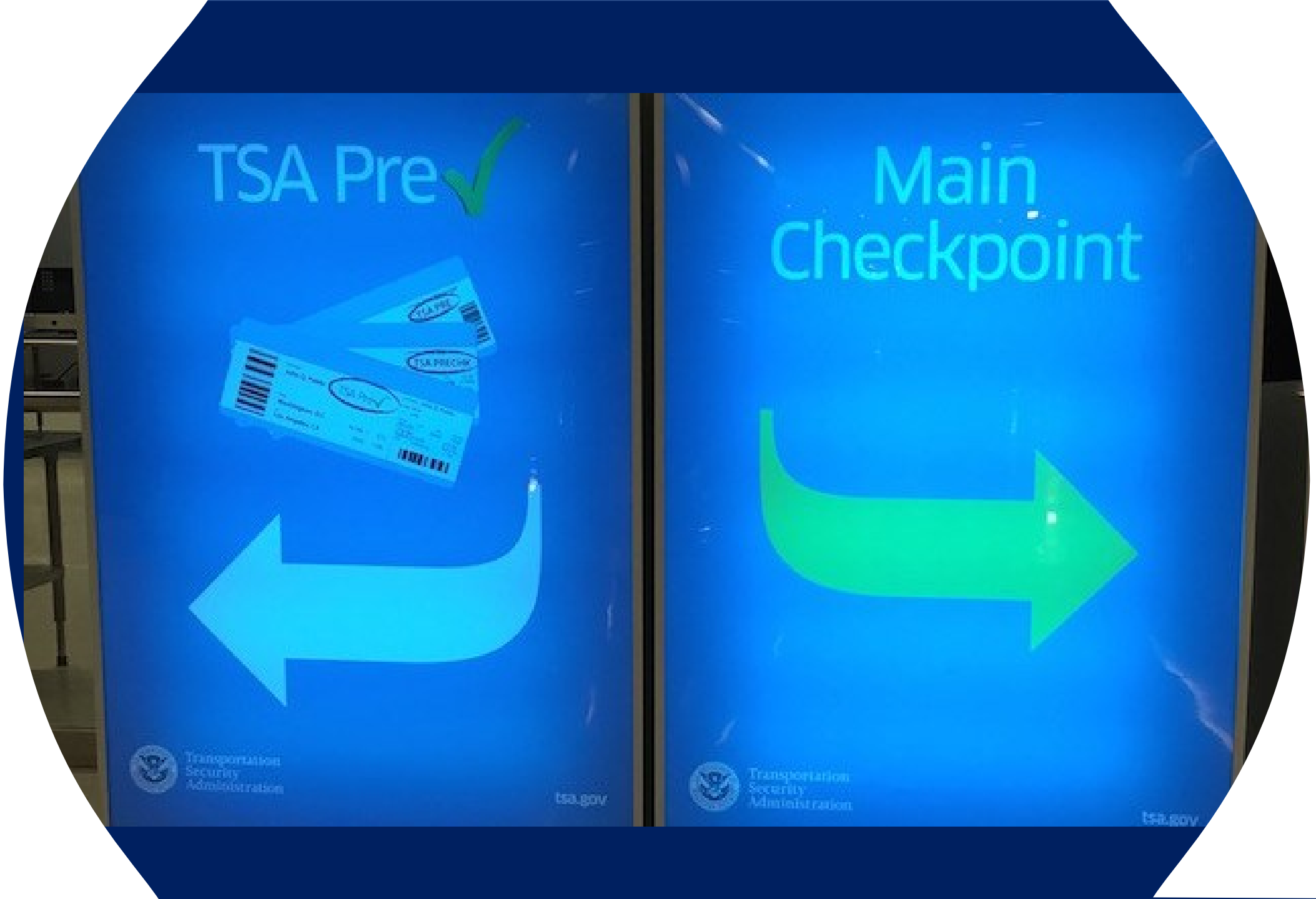
de-multiplexing

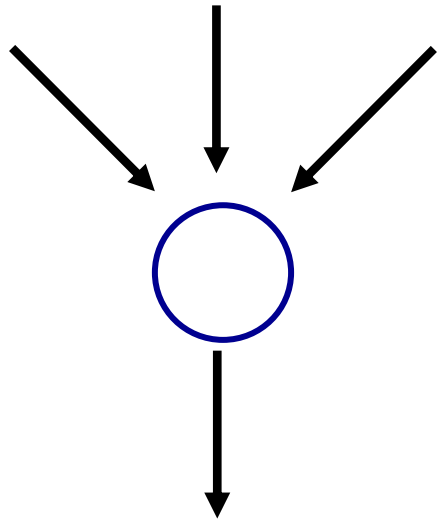


de-multiplexing

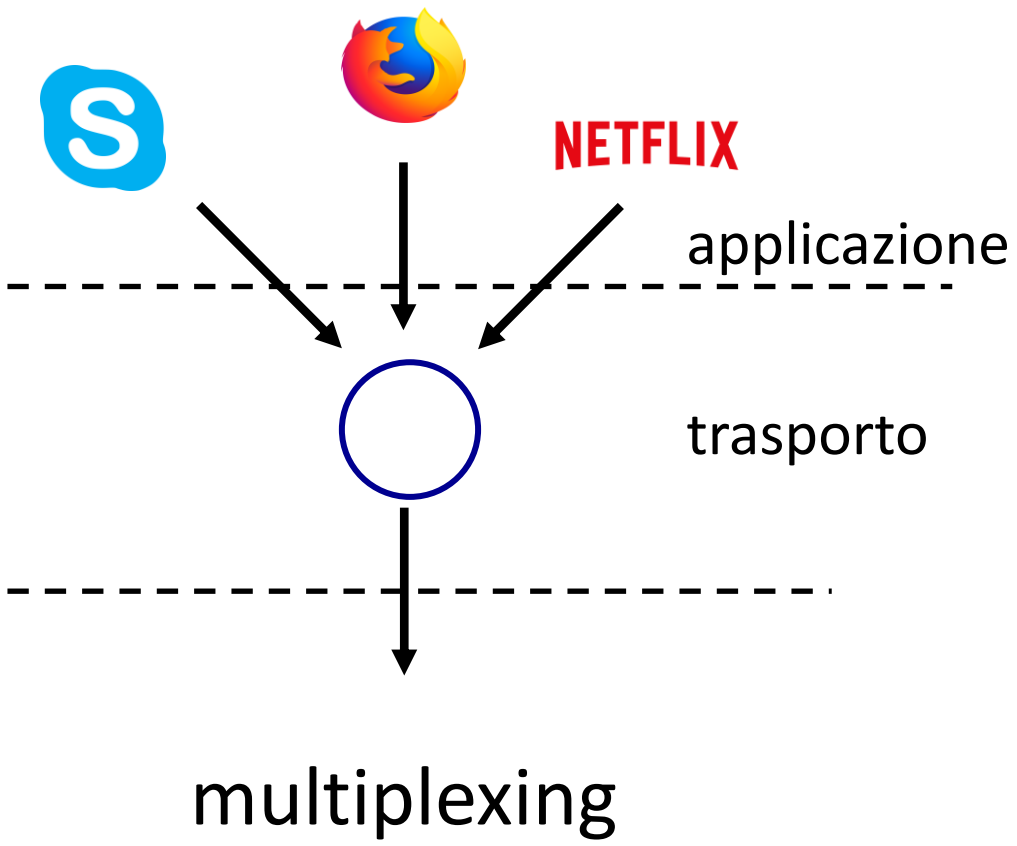
Demultiplexing

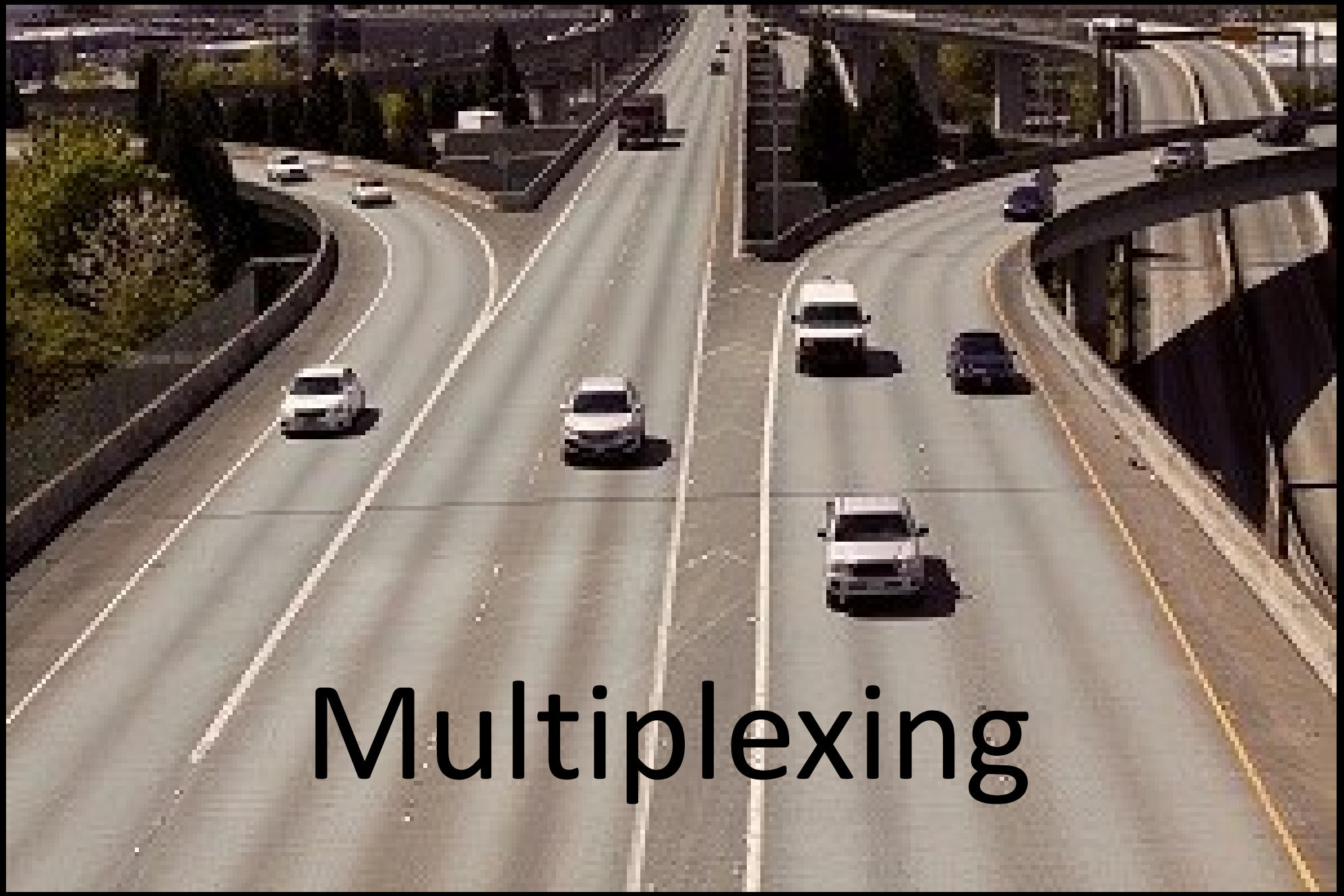

AIRFRANCE
ECONOMY
SKYTEAM
AIRFRANCE
SKY
PRIORITY™
SKYTEAM


TSA Pre
Transportation Security Administration
tsa.gov
Main Checkpoint
Transportation Security Administration
tsa.gov



multiplexing


NETFLIX
applicazione
trasporto
multiplexing


Multiplexing

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 porta* per inviare il segmento alla socket appropriata

formato dei segmenti TCP/UDP

Demultiplexing senza connsessione

- quando si crea una socket, si può specificare il numero di porta:

```
mySocket = socket(AF_INET, SOCK_DGRAM)
mySocket.bind(('',9157))
```

oppure, soprattutto per il lato client, lasciare che il sistema operativo le assegni un numero di porta *effimero*

- quando si crea il datagramma da inviare alla socket UDP, si deve specificare
  - indirizzo IP di destinazione
  - numero di porta di destinazione
- il segmento viene passato al livello di rete

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/UDP con lo *stesso indirizzo IP e numero di porta di destinazione*, ma indirizzi IP e/o numeri di porta di origine differenti vengono inviati alla *stessa socket* sull'host ricevente

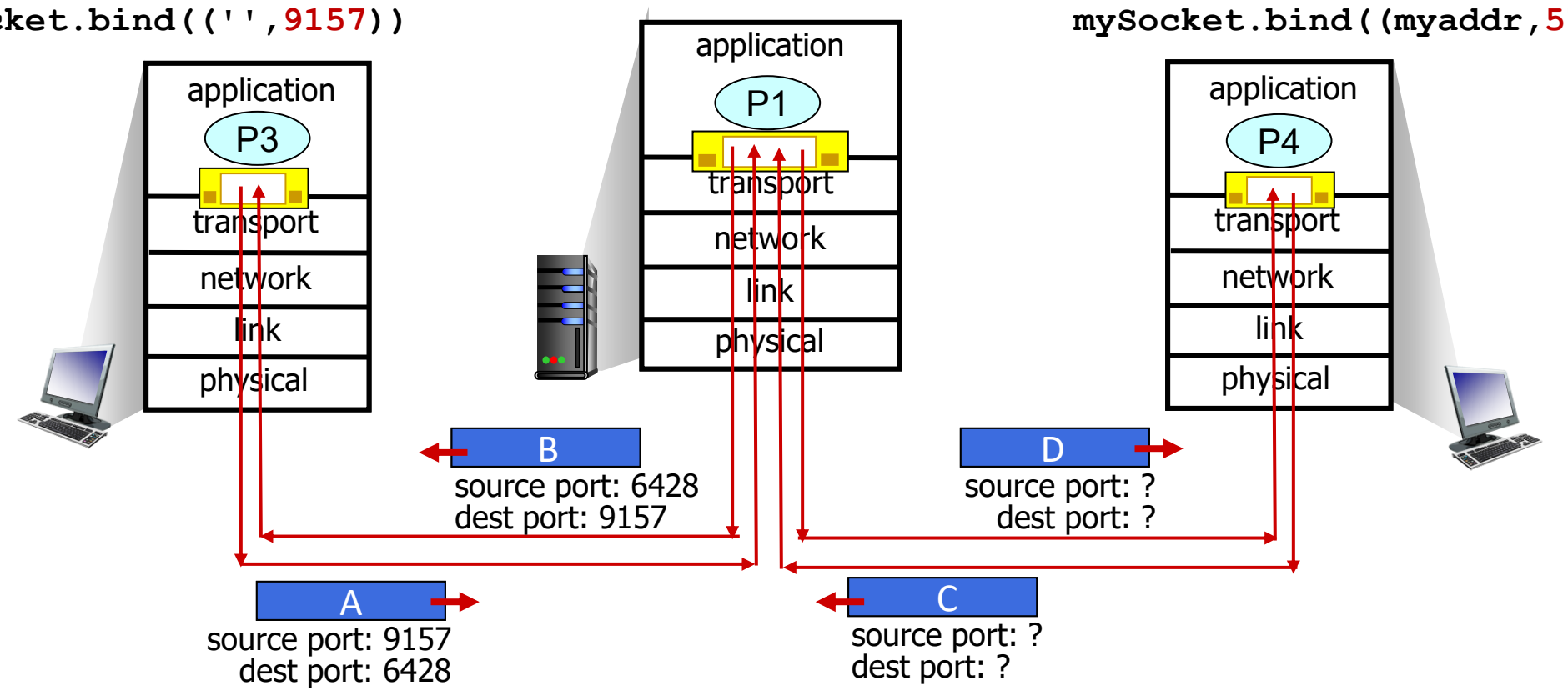
Indirizzo IP e numero di porta di origine servono come "indirizzo di ritorno" per una eventuale risposta

Demultiplexing senza connsessione

```
mySocket =
 socket(AF_INET,SOCK_DGRAM)
mySocket.bind(('',6428))
```

```
mySocket =
 socket(AF_INET, SOCK_DGRAM)
mySocket.bind(('',9157))
```

```
mySocket =
  socket(AF_INET, SOCK_DGRAM)
mySocket.bind((myaddr,5775))
```

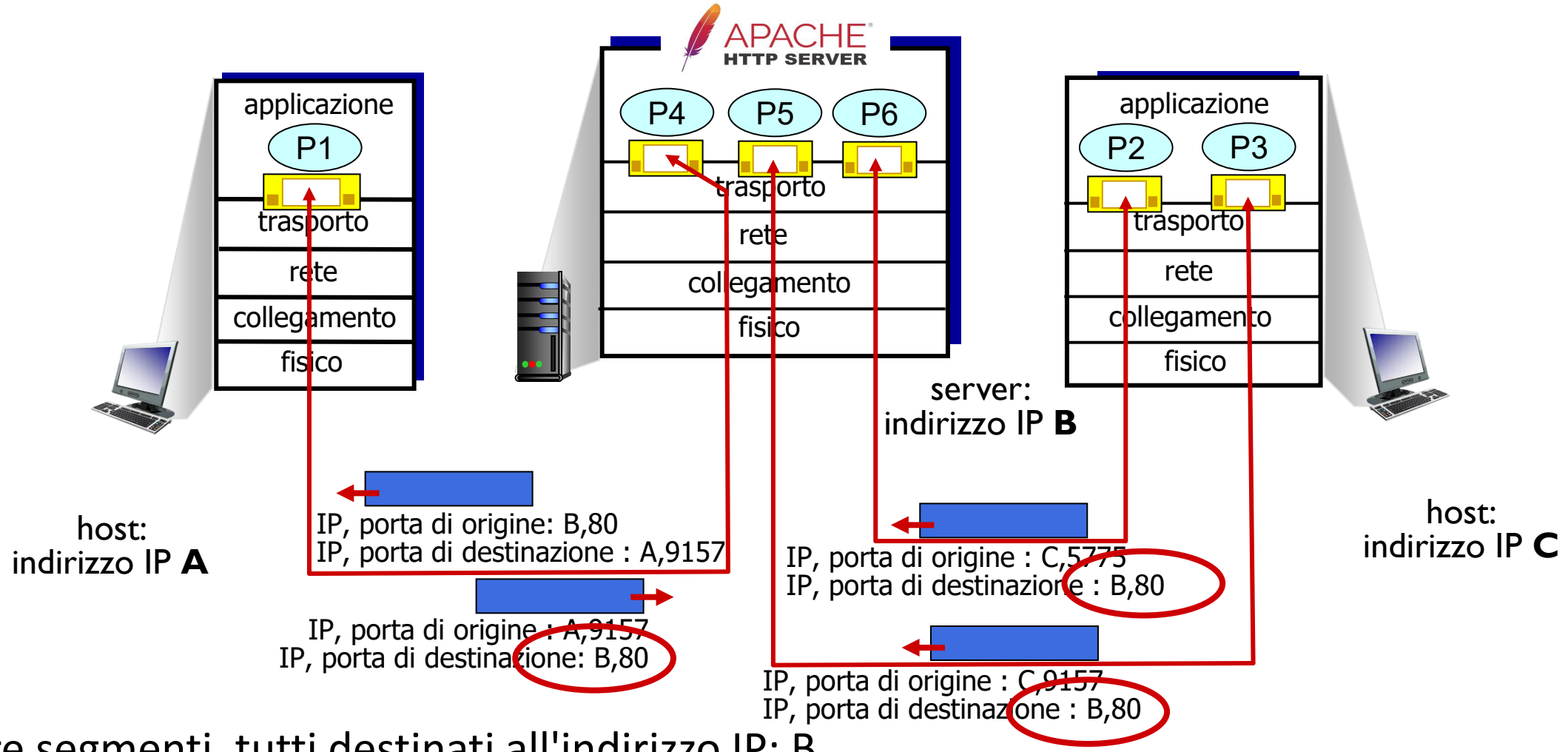


Demultiplexing orientato alla connessione

- la socket TCP è identificata da quattro parametri:
  - indirizzo IP di origine
  - numero di porta di origine
  - indirizzo IP di destinazione
  - numero di porta di destinazione
- demux: il lato ricevente usa i *quattro valori (quadrupla)* per inviare il segmento alla socket appropriata

- Un host server crea una *socket passiva* specificando un numero di porta
- La socket passiva viene usata per accettare le richieste di connessione, per ciascuna delle quali verrà creata una nuova *socket connessa* (con la medesima porta e indirizzo IP locale, ma diversa porta e indirizzo remoto, discriminando pertanto le socket connesse di client diversi)

Demultiplexing orientato alla connessione



Tre segmenti, tutti destinati all'indirizzo IP: B,
porta di destinazione: 80: ne viene fatto il demultiplexing verso socket *differenti*

Riassunto

- Multiplexing, demultiplexing: basato sui valori dei campi dell'intestazione del segmento o del datagramma
- **UDP:** demultiplexing usando (solo) il numero di porta e indirizzo IP di destinazione
- **TCP:** demultiplexing usando la quadrupla di valori: indirizzi di origine e di destinazione, e numeri di porta
- Multiplexing/demultiplexing avviene a *tutti* i livelli (ogni volta che entità diverse vogliono usare i servizi del protocollo di livello inferiore)

Nella creazione di una socket abbiamo usato `''` (che in Python equivale a '0.0.0.0' nel caso di IPv4) per indicare qualunque indirizzo IP dell'host; tuttavia, avremmo potuto specificare uno in particolare.