Servizi del livello trasporto, protocollo UDP, intro a TCP

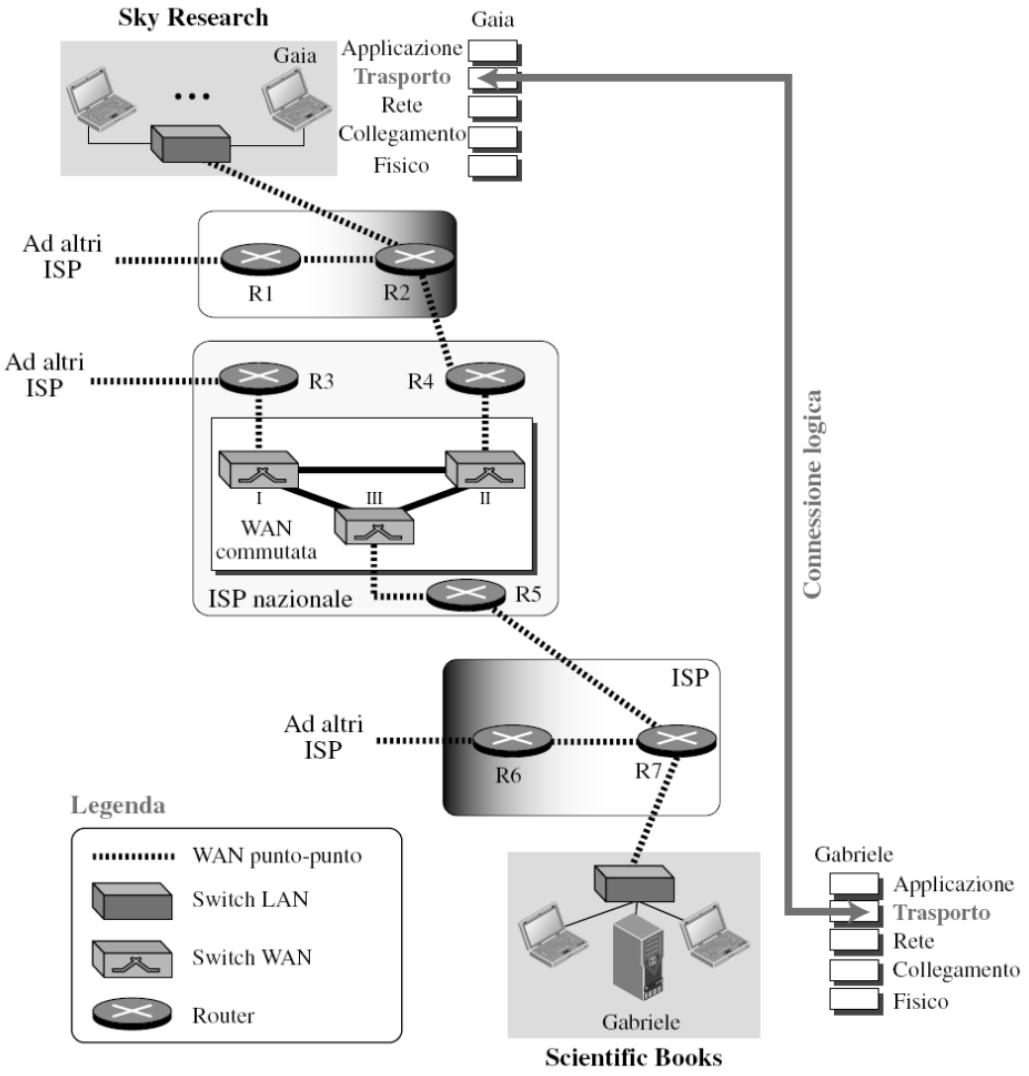
Prof.ssa Gaia Maselli

Parte di queste slide sono state prese dal materiale associato ai libri:

1) B.A. Forouzan, F. Mosharraf – Reti di calcolatori. Un approccio top-down. Copyright © 2013 McGraw-Hill Education Italy srl. Edizione italiana delle slide a cura di Gabriele D'Angelo e Gaia Maselli
2) Computer Networking: A Top Down Approach , 6th edition. All material copyright 1996-2009 J.F Kurose and K.W. Ross, All Rights Reserved

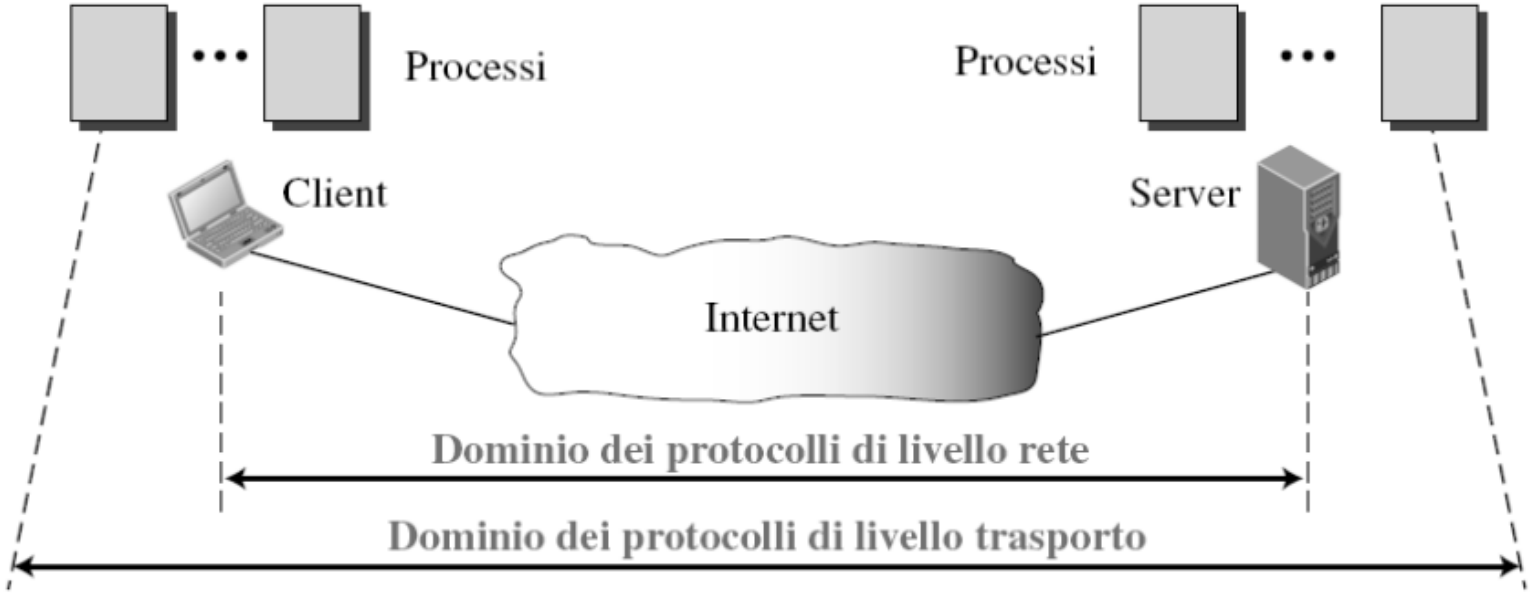
Connessione logica a livello trasporto

- 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: incapsula i messaggi in segmenti e li passa al livello di rete
  - lato ricezione: decapsula i segmenti in messaggi e li passa al livello di applicazione



Relazione tra livello di trasporto e livello di rete

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



Esempio

Analogia con la posta ordinaria:

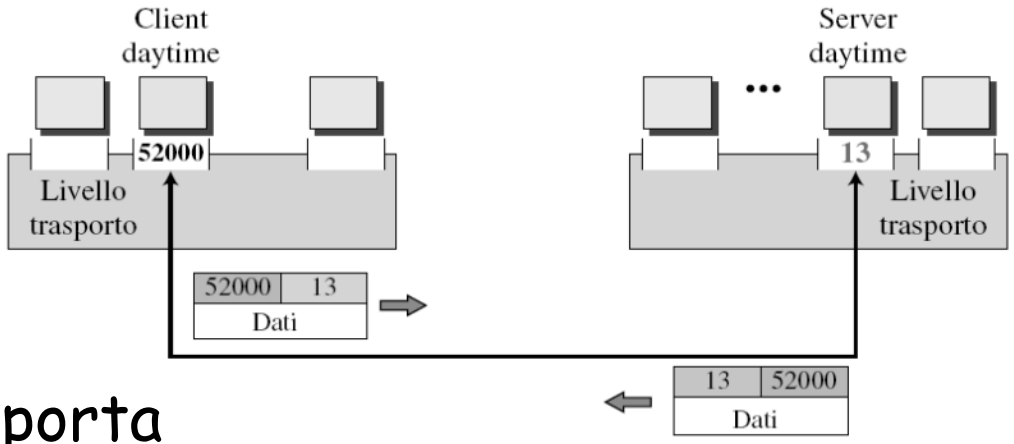
2 persone di un condominio inviano una lettera a 2 persone di un altro condominio consegnandola/ricevendola a/da un portiere

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

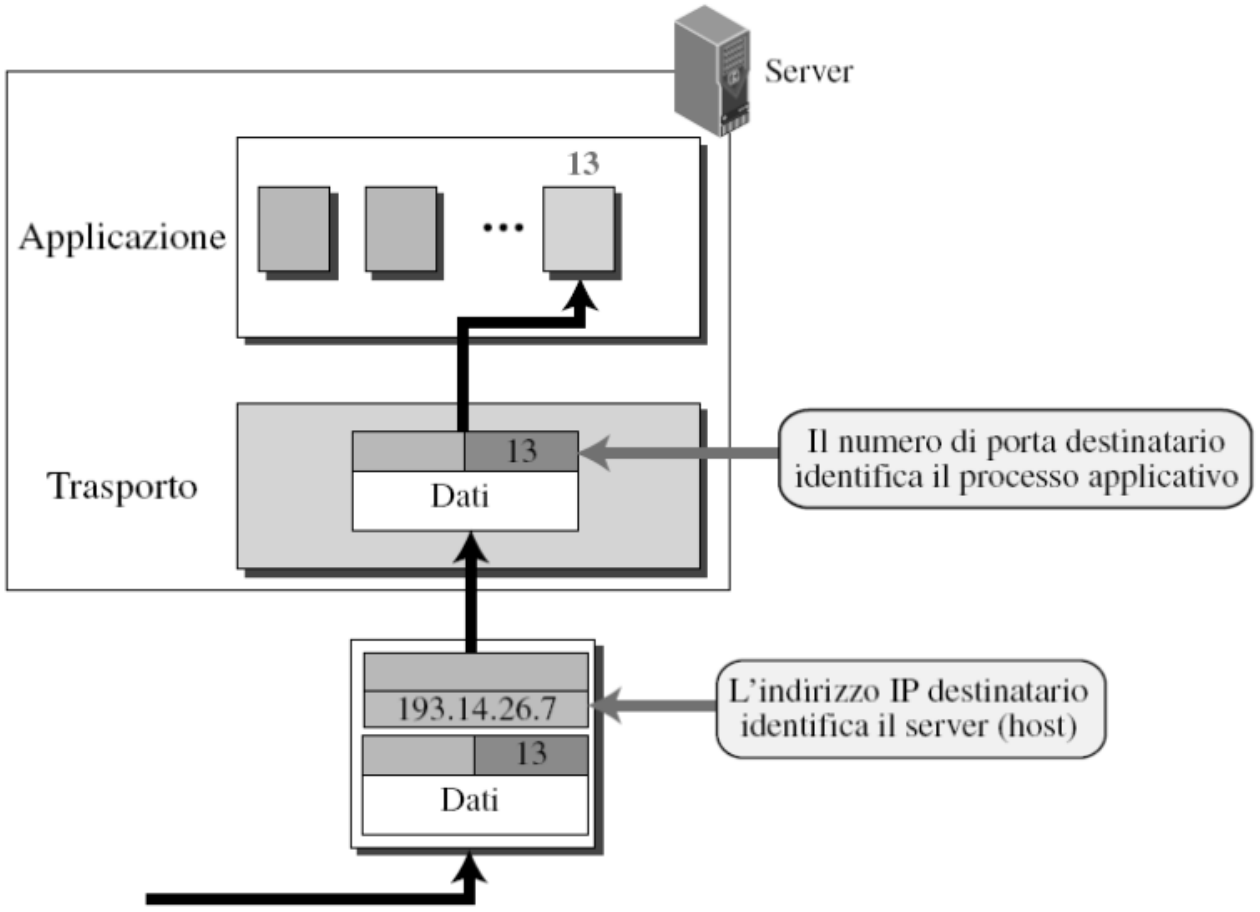
N.B. i portieri svolgono il proprio lavoro localmente, non sono coinvolti nelle tappe intermedie delle lettere (così come il protocollo di trasporto)

Indirizzamento

- La maggior parte dei sistemi operativi è multiutente e multiprocesso
  - Diversi processi client attivi (host locale)
  - Diversi processi server attivi (host remoto)
- Per stabilire una comunicazione tra i due dispositivi è necessario un metodo per individuare:
  - Host locale
  - Host remoto
  - Processo locale
  - Processo remoto
- Host ➔ indirizzo IP
- Processo ➔ numero di porta

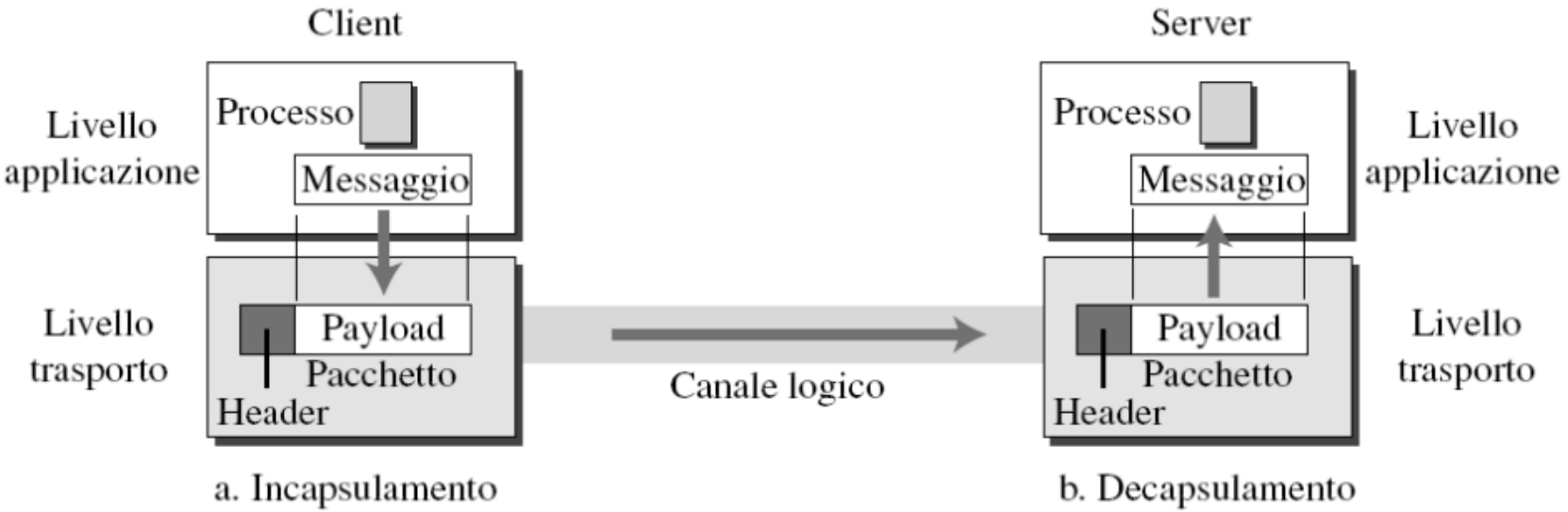


Indirizzi IP vs numeri di porta



Indirizzo IP + porta = socket address

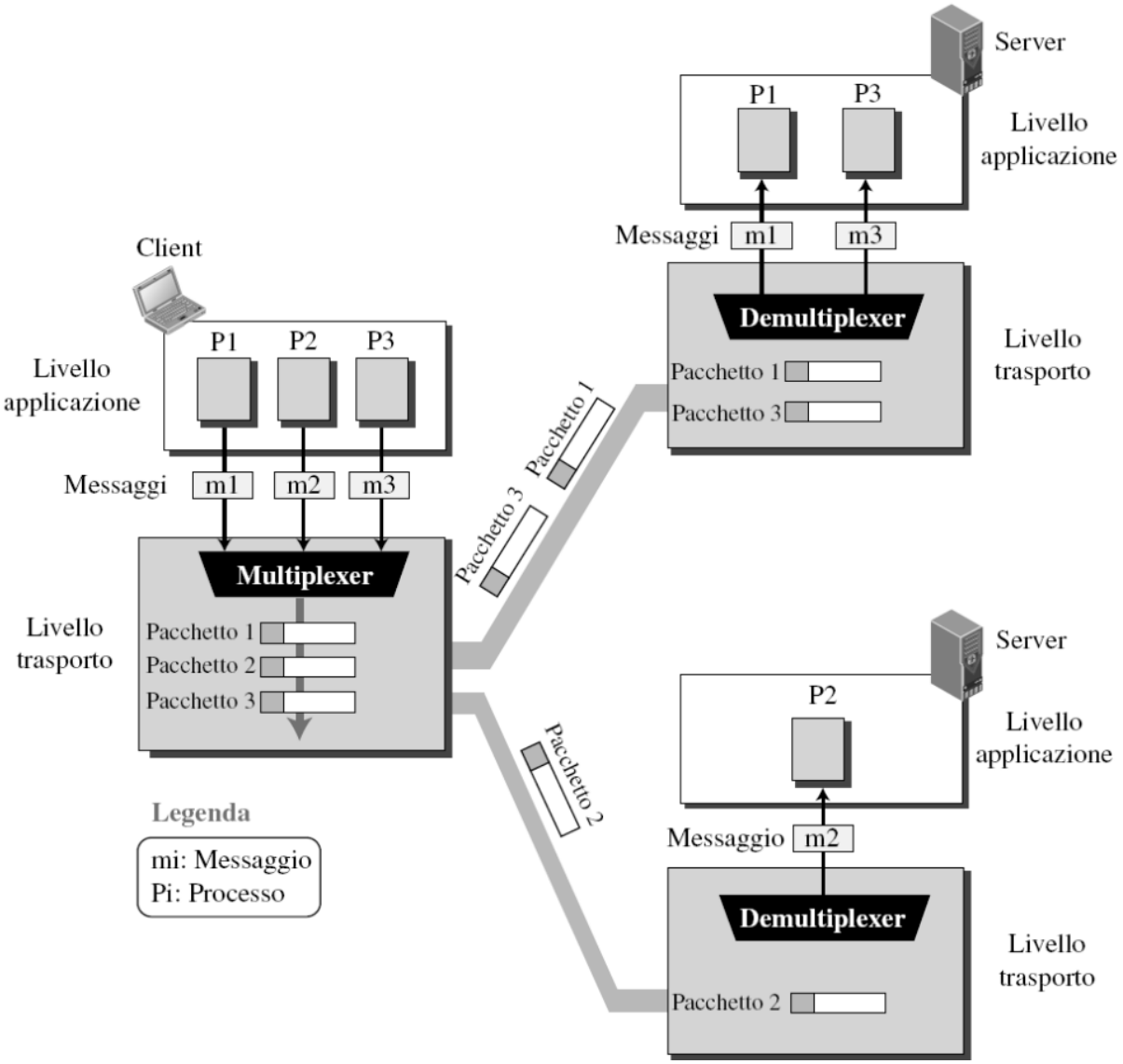
Incapsulamento/decapsulamento



I pacchetti a livello di trasporto sono chiamati *segmenti* (TCP) o *datagrammi utente* (UDP)

Multiplexing/demultiplexing

- 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



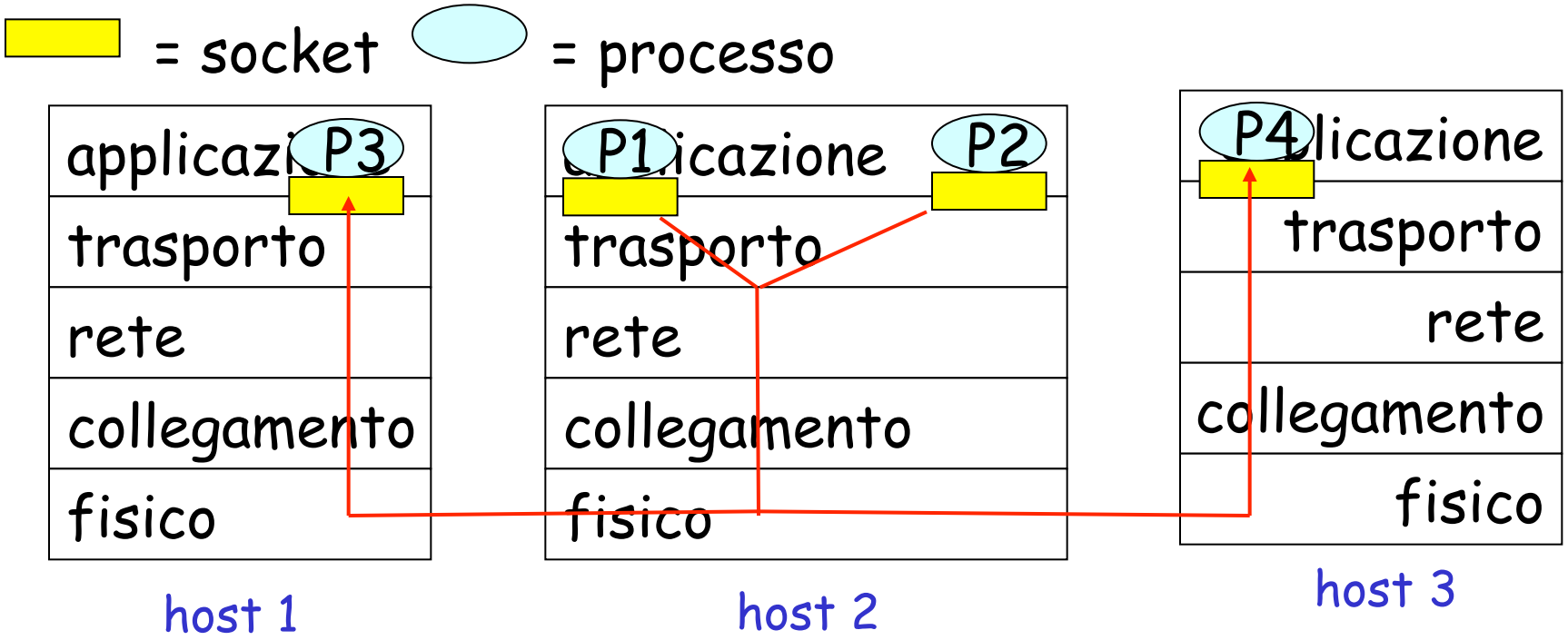
Multiplexing

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

- L'host deve 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)



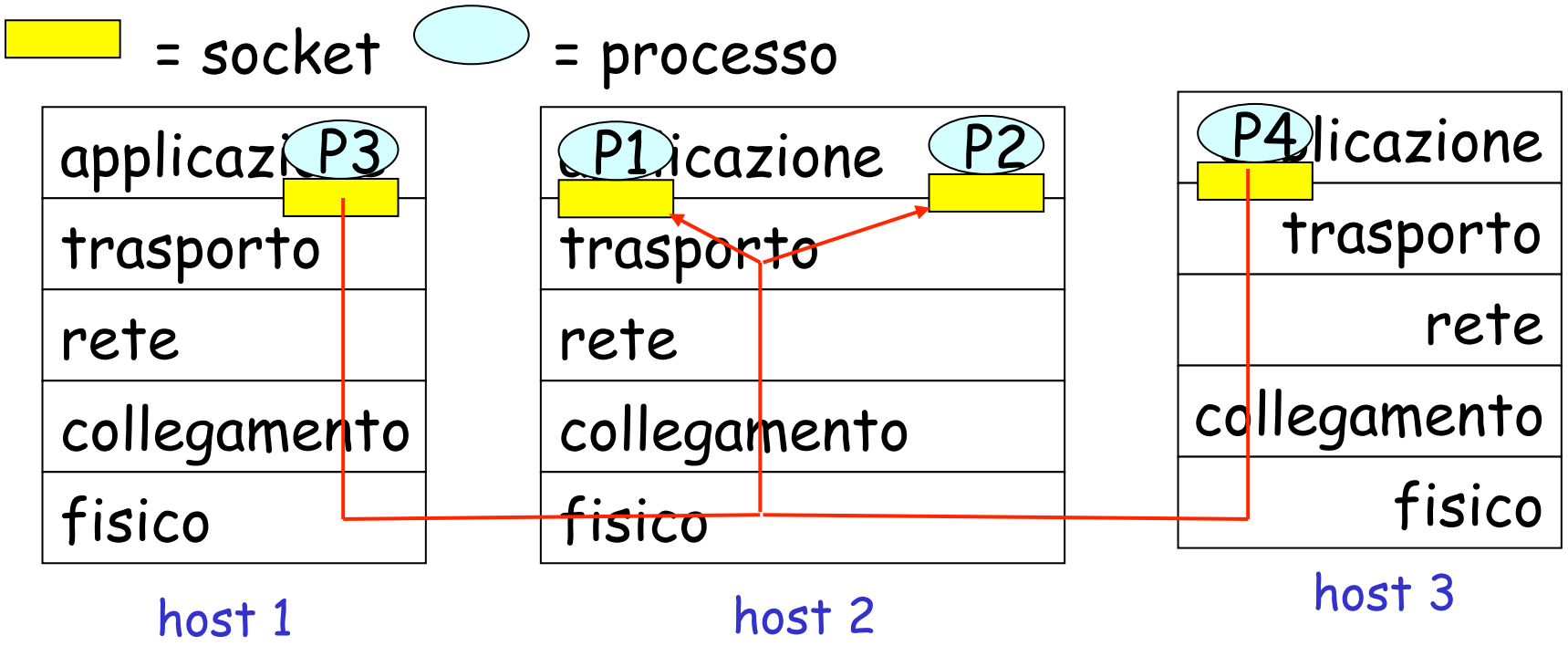
Demultiplexing

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



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 al processo appropriato

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

Struttura del pacchetto TCP/UDP

esempio

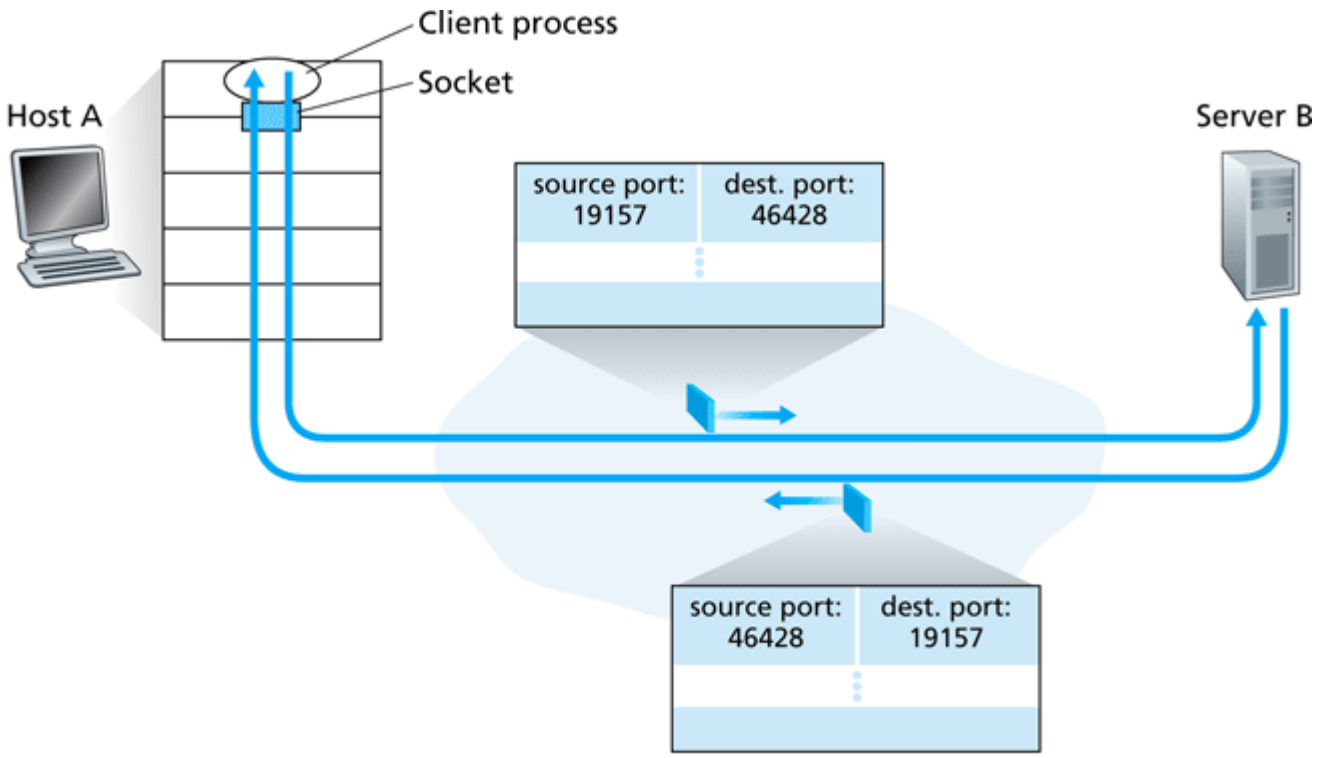
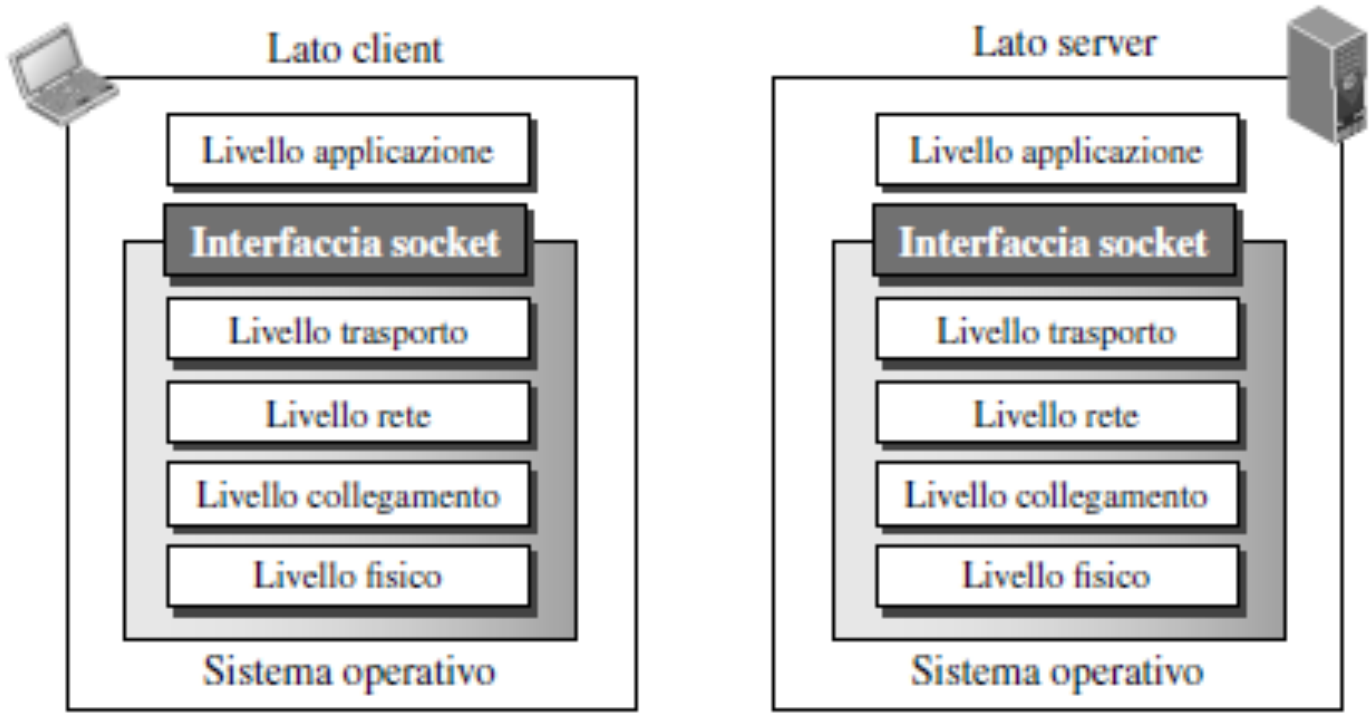


Figure 3.4 ♦ The inversion of source and destination port numbers

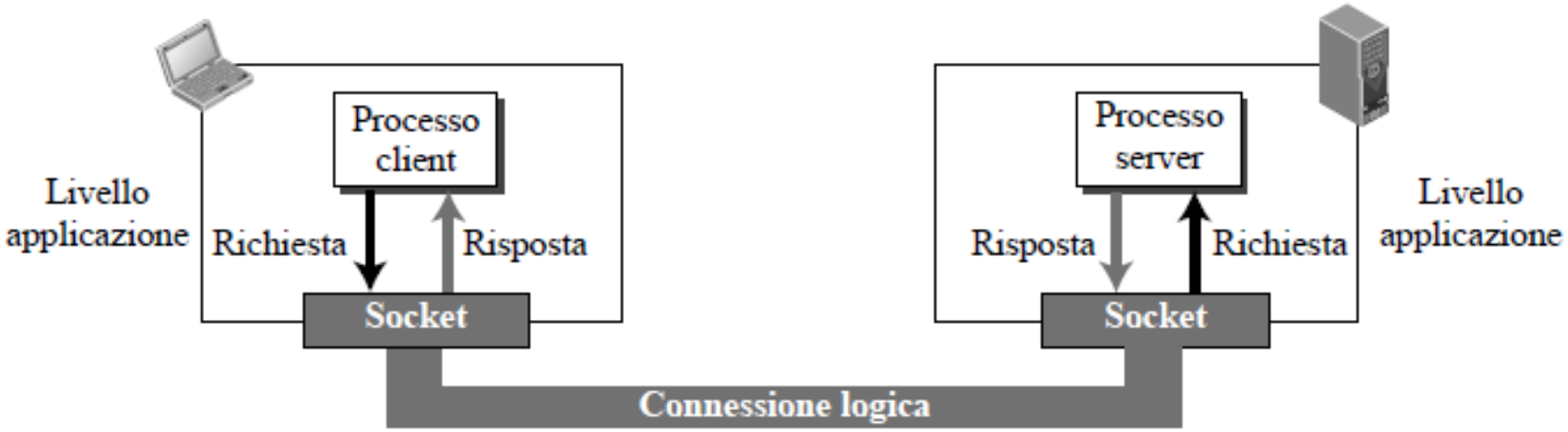
API di comunicazione

- Socket

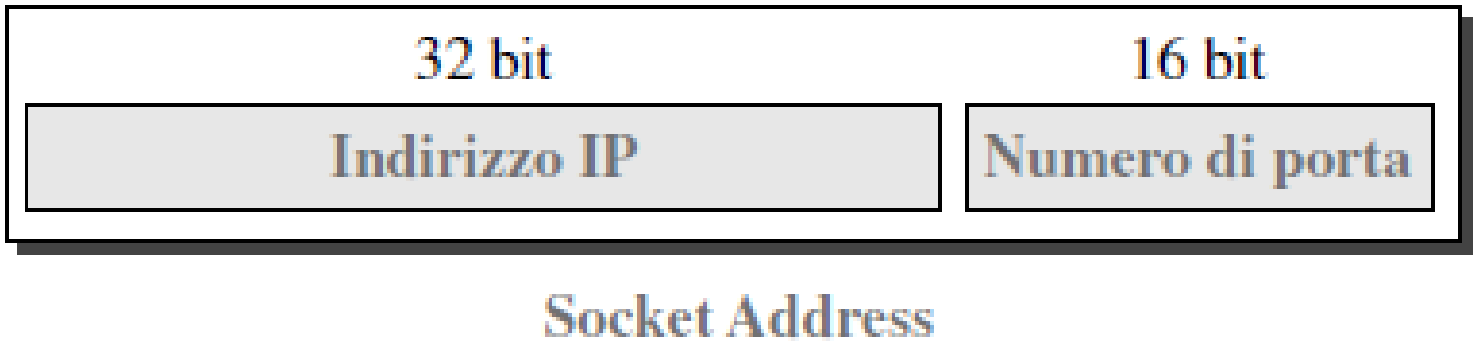


Comunicazione tra processi: socket

- Appare come un terminale o un file ma non è un'entità fisica
- Astrazione
- Struttura dati creata e utilizzata dal programma applicativo
- Comunicare tra un processo client e un processo server significa comunicare tra due socket create nei due lati di comunicazione



Un socket address



Individuare i socket address

- L'interazione tra client e server è bidirezionale
- È necessaria quindi una coppia di indirizzi socket: *locale* (mittente) e *remoto* (destinatario)
- L'indirizzo locale in una direzione è l'indirizzo remoto nell'altra direzione
- Come vengono definiti/individuati questi indirizzi?

Individuare i socket lato server

- Il server ha bisogno di un socket address locale (server) e uno remoto (client) per comunicare
- Socket address locale: fornito dal sistema operativo
  - Conosce l'indirizzo IP del computer su cui il server è in esecuzione
  - Il numero di porta è noto al server perché assegnato dal progettista (numero well known o scelto)
- Socket address remoto
  - è il socket address locale del client che si connette
  - Poiché numerosi client possono connettersi, il server non può conoscere a priori tutti i socket address, ma li trova all'interno del pacchetto di richiesta
- N.B. il socket address locale di un server non cambia (è fissato e rimane invariato), mentre il socket address remoto varia ad ogni interazione con client diversi (anche con stesso client su connessioni diverse)

Individuare i socket lato client

- Il client ha bisogno di un socket address locale (client) e uno remoto (server) per comunicare
- Socket address locale: fornito dal sistema operativo
  - Conosce l'indirizzo IP del computer su cui il client è in esecuzione
  - Il numero di porta è assegnato temporaneamente dal sistema operativo (numero di porta effimero o temporaneo – non utilizzato da altri processi)
- Socket address remoto
  - Numero di porta noto in base all'applicazione (http porta 80)
  - Indirizzo IP fornito dal DNS (Domani Name System)
  - Oppure porta e indirizzo noti al programmatore quando si vuole verificare il corretto funzionamento di un'applicazione (esempi sul libro di testo)

Servizi dei protocolli di trasporto Internet

Servizio di TCP:

- *Orientato alla connessione:* è richiesto un setup fra i processi client e server
  - *Somiglia al **sistema telefonico**:* come per eseguire una telefonata, l'utente deve stabilire una connessione, usarla e quindi rilasciarla.
  - Funziona come un tubo: il trasmettitore vi spinge oggetti (bit) a una estremità e il ricevitore li prende dall'altra. L'ordine è conservato ovvero i bit arrivano nella sequenza con cui sono stati trasmessi.
- *trasporto affidabile* fra i processi d'invio e di ricezione
- *controllo di flusso:* il mittente non vuole sovraccaricare il destinatario
- *controllo della congestione:* "strozza" il processo d'invio quando le rete è sovraccaricata
- *non offre:* temporizzazione, garanzie su un'ampiezza di banda minima, sicurezza (alcune possono essere implementate, Es. SSL)

Servizi dei protocolli di trasporto Internet

Servizio di UDP:

- ***Senza connessione:*** non è richiesto alcun setup fra i processi client e server
- trasferimento dati inaffidabile fra i processi d'invio e di ricezione
  - *Somiglia al **sistema postale**:* Ogni messaggio è instradato attraverso il sistema postale in modo indipendente dagli altri
  - E' possibile che due messaggi mandati alla stessa destinazione arrivino in tempi diversi
- non offre: setup della connessione, affidabilità, controllo di flusso, controllo della congestione, temporizzazione né ampiezza di banda minima e sicurezza

D: Perché esiste UDP?

Applicazioni Internet: protocollo a livello applicazione e protocollo di trasporto

Applicazione	Protocollo a livello applicazione	Protocollo di trasporto sottostante
Posta elettronica	SMTP [RFC 2821]	TCP
Accesso a terminali remoti	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Trasferimento file	FTP [RFC 959]	TCP
Multimedia in streaming	HTTP (es. YouTube) RTP [RFC 1889]	TCP o UDP
Telefonia Internet	SIP, RTP, proprietario (es. Skype)	Tipicamente UDP

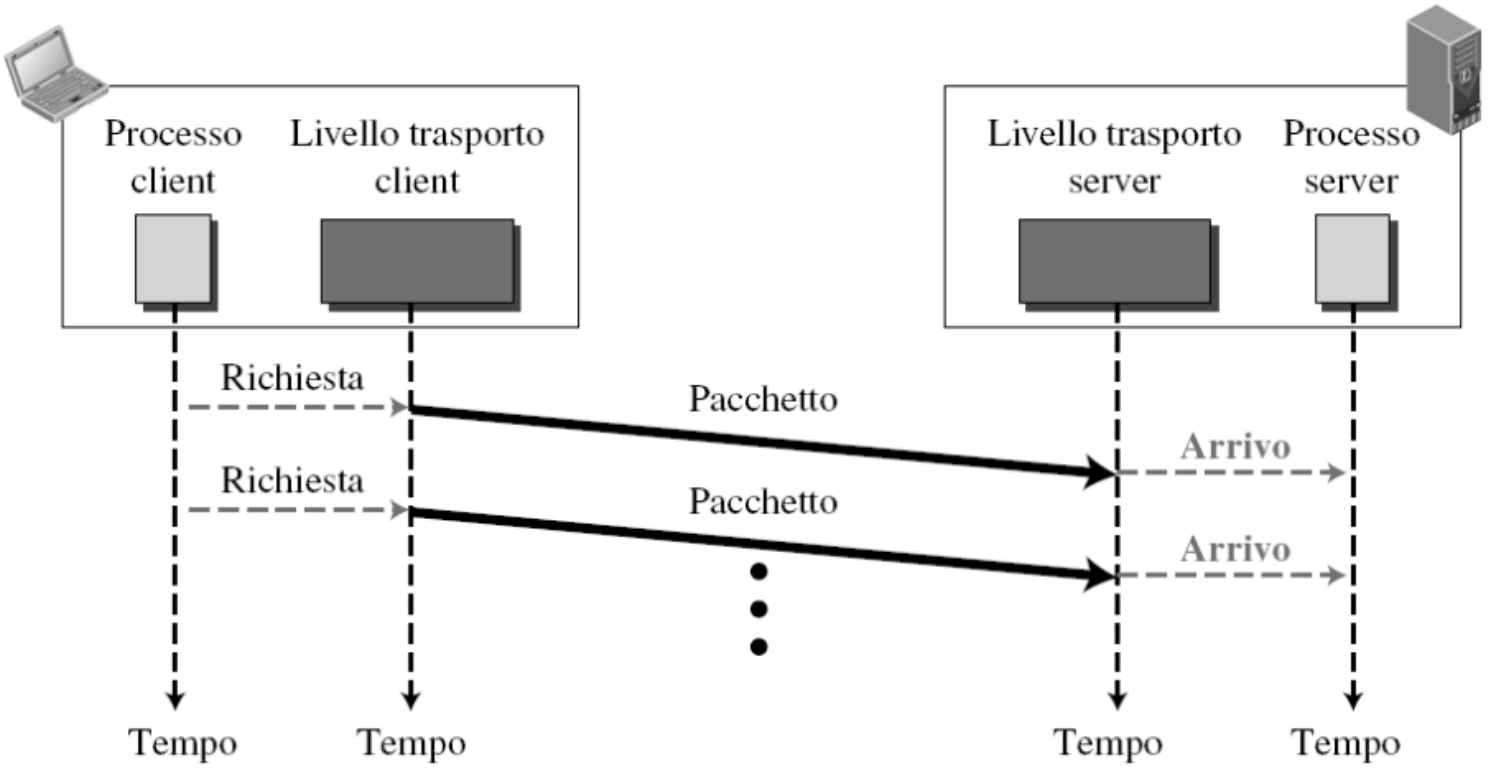
Protocollo UDP
(User Datagram Protocol)

User Datagram Protocol (UDP)[RFC 768]

- Protocollo di trasporto inaffidabile e privo di connessione
- Fornisce i servizi di
  - Comunicazione tra processi utilizzando i socket
  - Multiplexing/demultiplexing dei pacchetti
  - Incapsulamento e decapsulamento
    - Datagrammi indipendenti, non numerati
- Non fornisce alcun controllo di flusso, errori (eccetto checksum), congestione

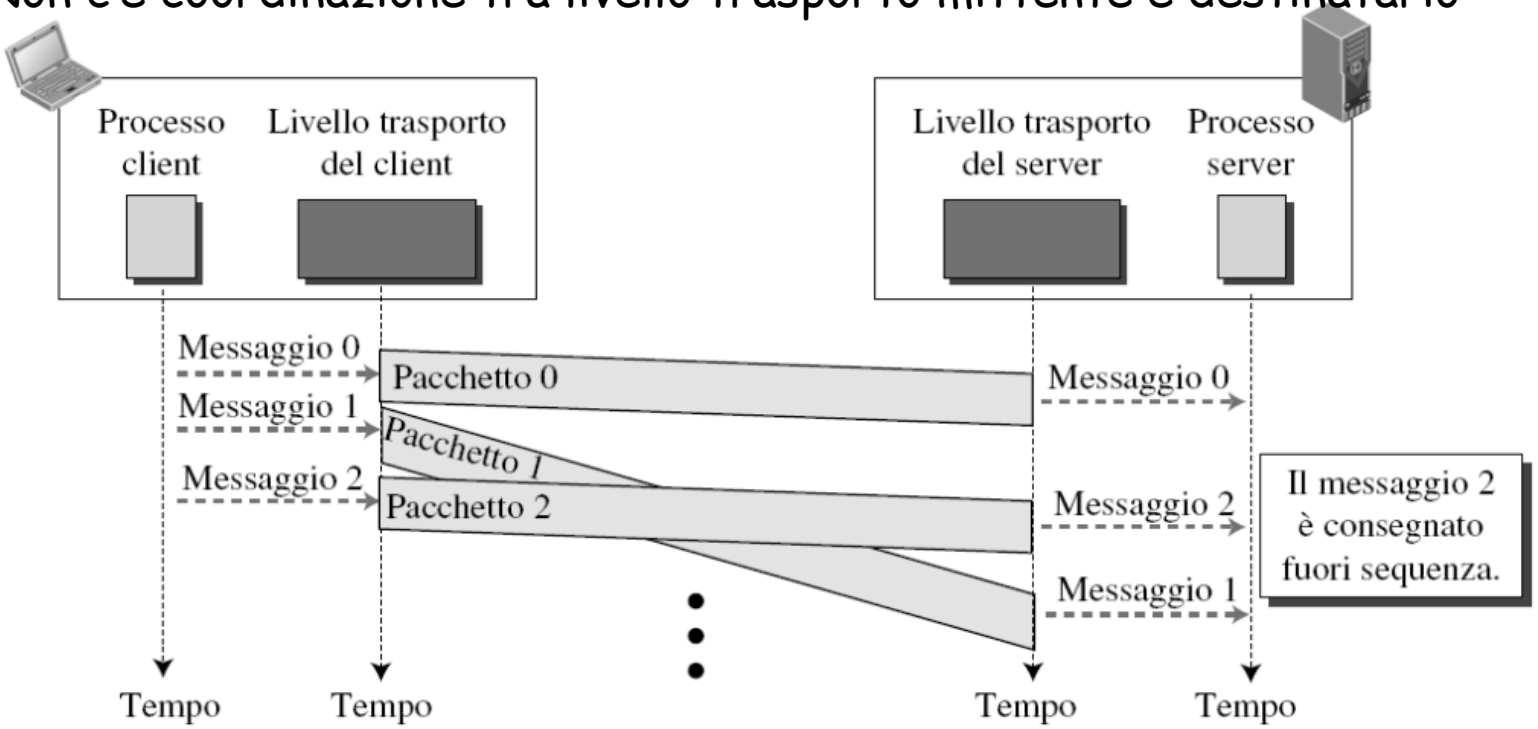
Diagramma di comunicazione

- Il mittente invia pacchetti uno dopo l'altro senza pensare al destinatario (può inviare dati a raffica perché non c'è controllo di flusso né di congestione)



Servizio connectionless

- Il mittente deve dividere i suoi messaggi in porzioni di dimensioni accettabili dal livello di trasporto, a cui consegnarli uno per uno
- Ogni pacchetto è ***indipendente*** dagli altri (la sequenza di arrivo può essere diversa da quella di spedizione)
- Non c'è coordinazione tra livello trasporto mittente e destinatario



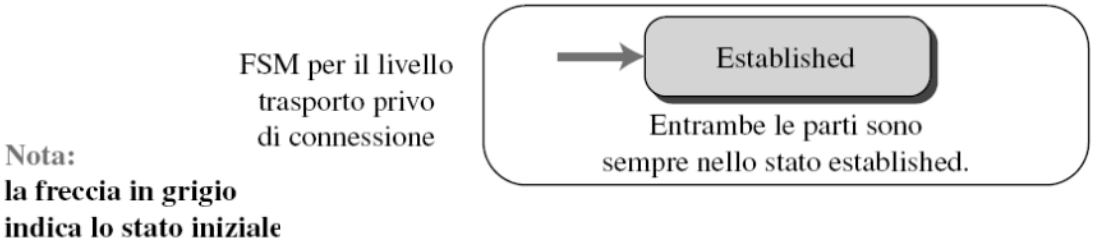
- Non è possibile implementare efficacemente controllo di flusso, controllo degli errori, controllo della congestione

Rapprensentazione mediante FSM (Finite state machine)

- Il comportamento di un protocollo di trasporto può essere rappresentato da un automa a stati finiti
- L'automa rimane in uno *stato* fin quando non avviene un *evento* che può modificare le stato dell'automa (transizione di stato) e fargli compiere un'*azione*

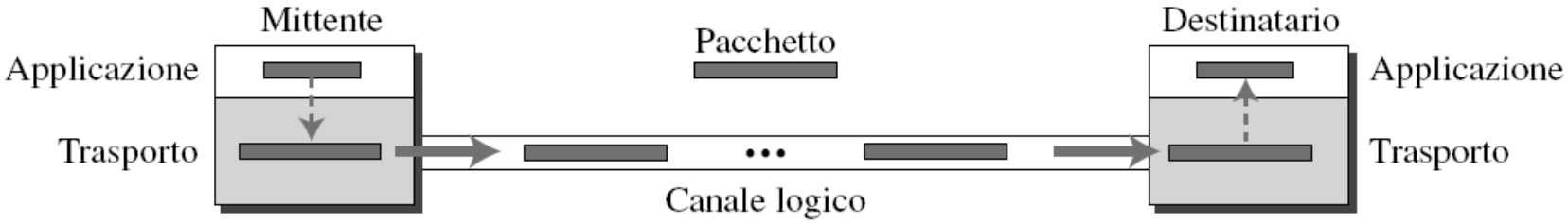
$$\frac{\textit{evento}}{\textit{azione}}$$

RAPPRESENTAZIONE DI UN SERVIZIO SENZA CONNESSIONE

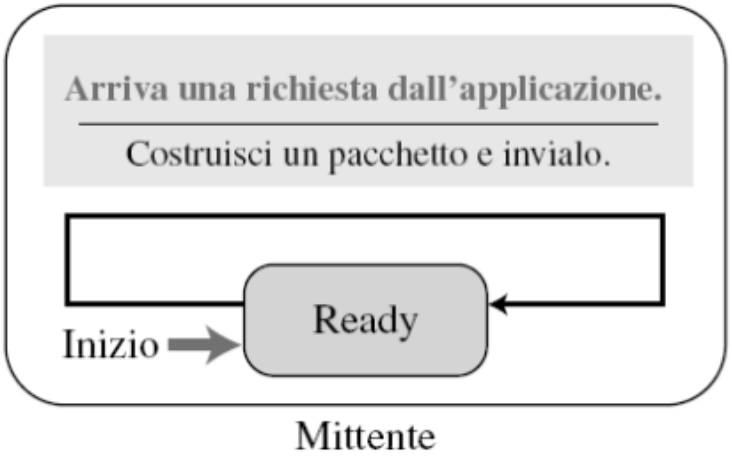


Protocollo UDP

- Protocollo molto semplice

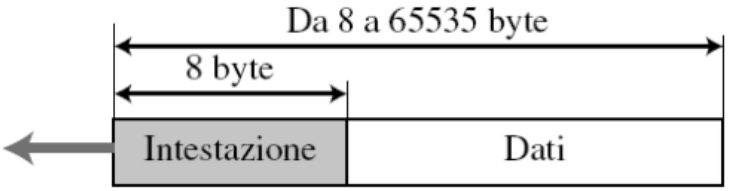


- FSM per UDP



Datagrammi UDP

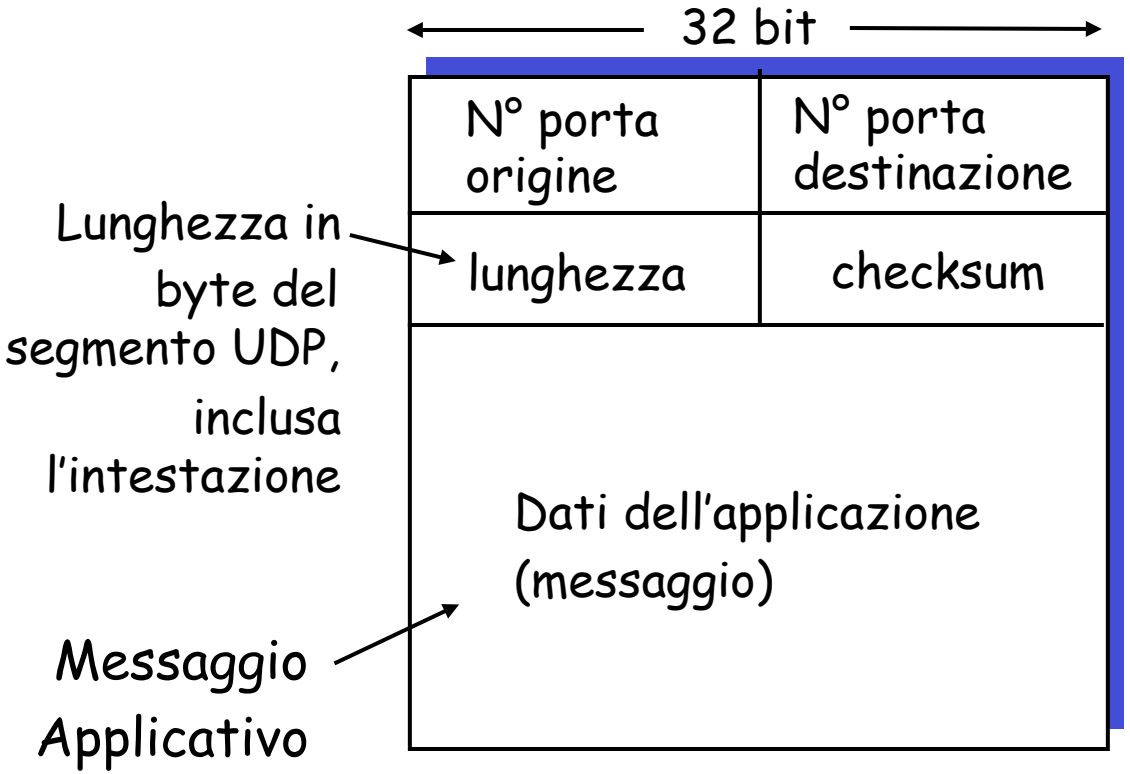
- Non vi è alcun flusso di dati
- Il processo mittente non può inviare un flusso di dati e aspettarsi che UDP lo suddivida in datagrammi correlati
- I processi devono inviare richieste di dimensioni sufficientemente piccole per essere inserite ciascuna in un singolo datagramma utente
- Solo i processi che usano messaggi di dimensioni inferiori a 65507 byte (65535 – 8 byte di intestazione UDP e 20 byte di intestazione IP) possono utilizzare il protocollo UDP



a. Datagramma utente UDP.

Struttura dei datagrammi UDP

- 4 campi di 2 byte



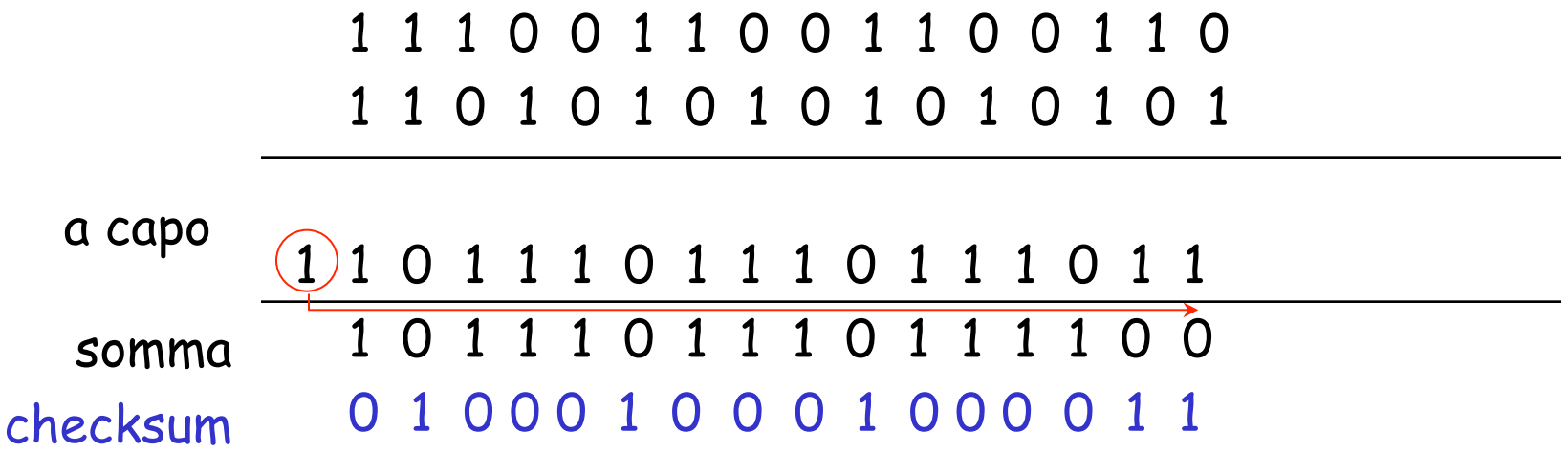
Checksum UDP

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

Mittente	*Ricevente*
1. Il messaggio viene diviso in "parole" da 16 bit.	

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



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 che ha richiesto il servizio DNS
- 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

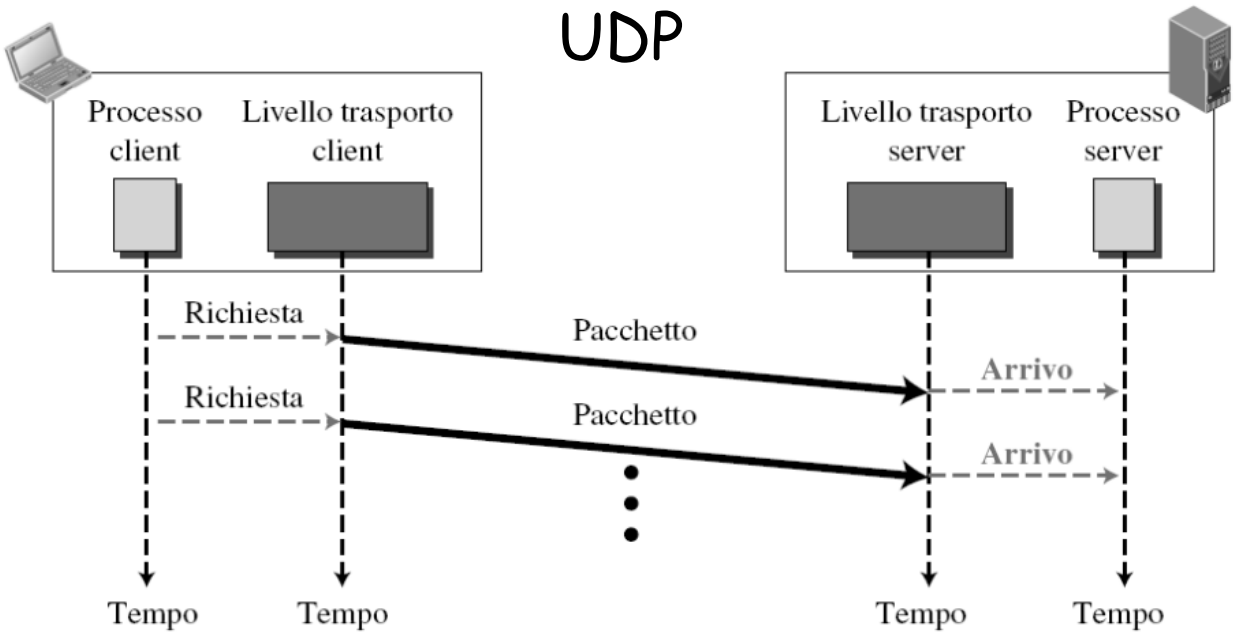
UDP: ulteriori informazioni

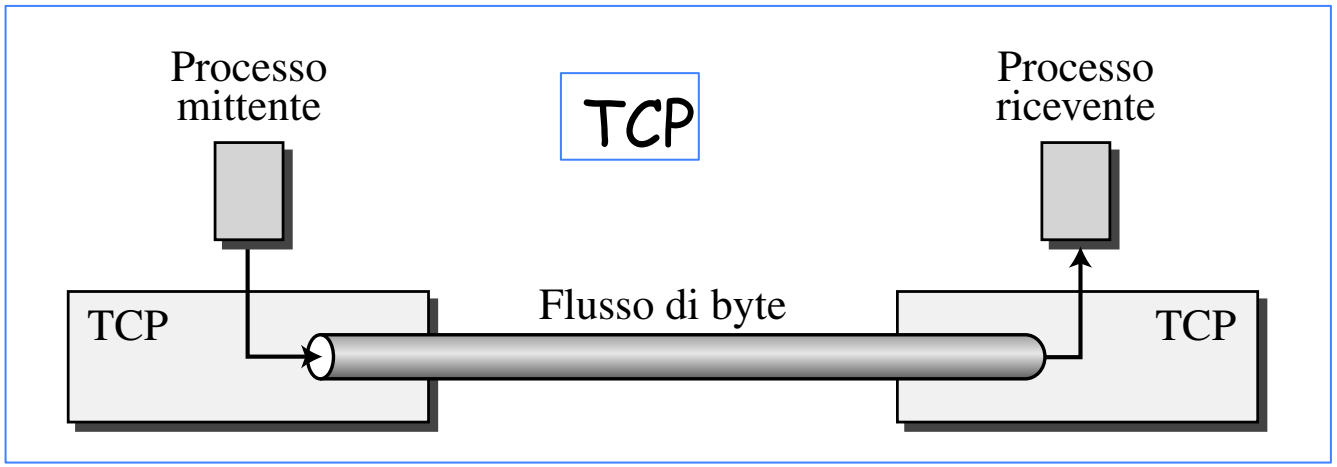
- Utilizzato spesso nelle applicazioni multimediali
  - Tollera perdite di pacchetti (limitate)
  - sensibile alla frequenza
- Altri impieghi di UDP
  - SNMP

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

Verso TCP

Diagramma di flusso





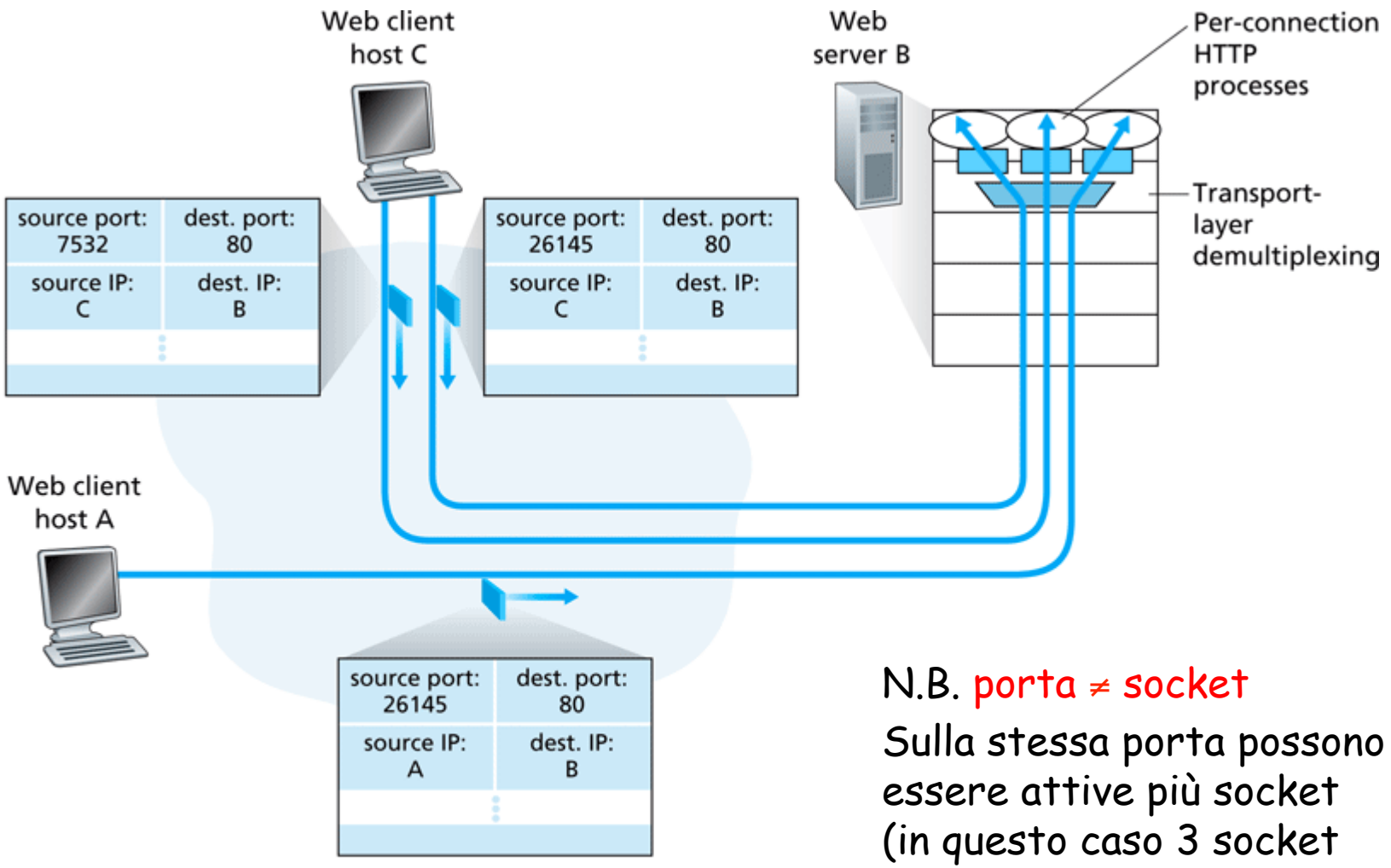
Servizi del TCP

- Comunicazione tra processi
  - Indirizzamento mediante numero di porta
- Incapsulamento/decapsulamento (frammenti)
- Multiplexing/demultiplexing
- Trasporto orientato alla connessione
- Controllo di flusso
- Controllo degli errori
- Controllo della congestione

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

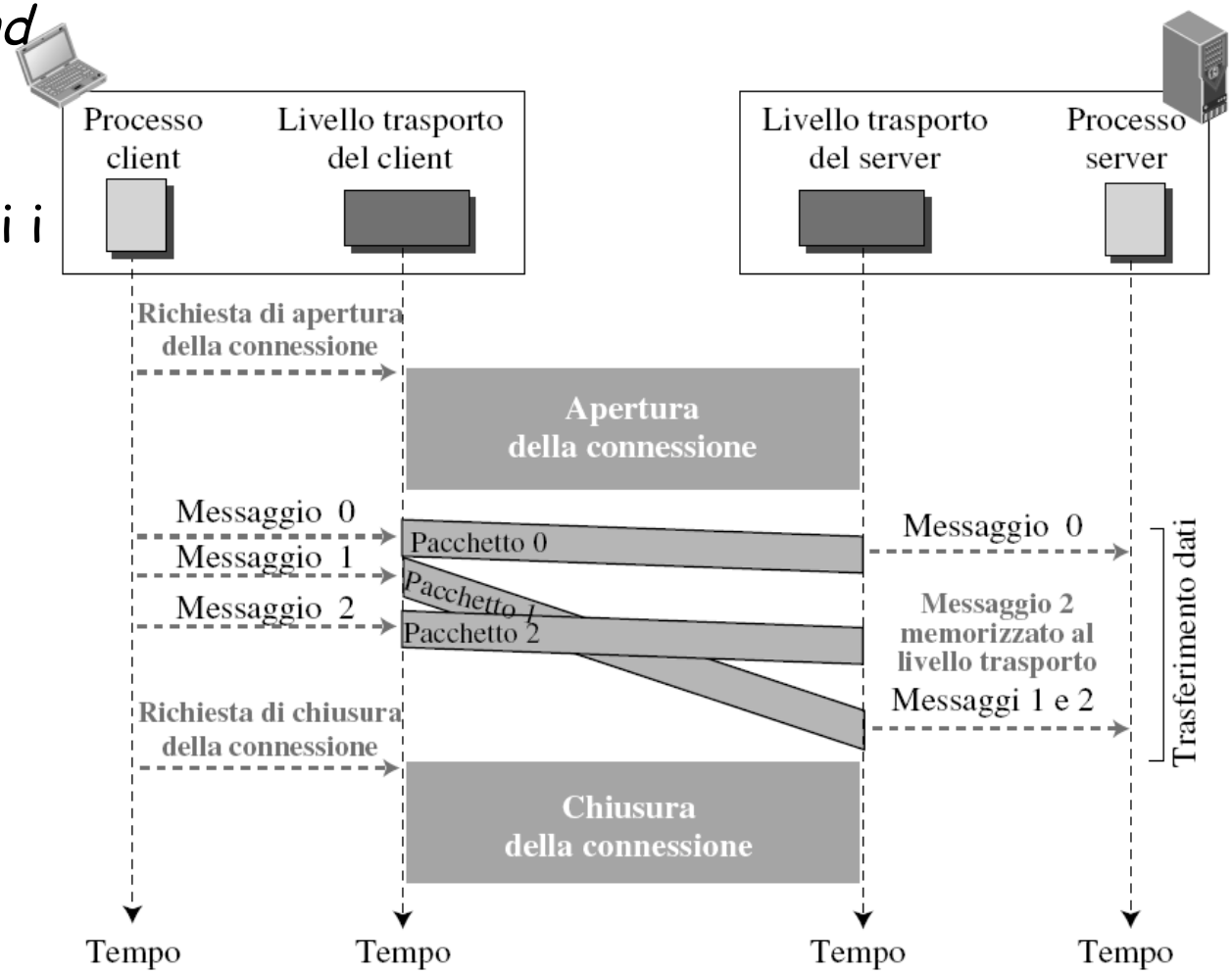


N.B. porta ≠ socket

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

Servizio connection oriented

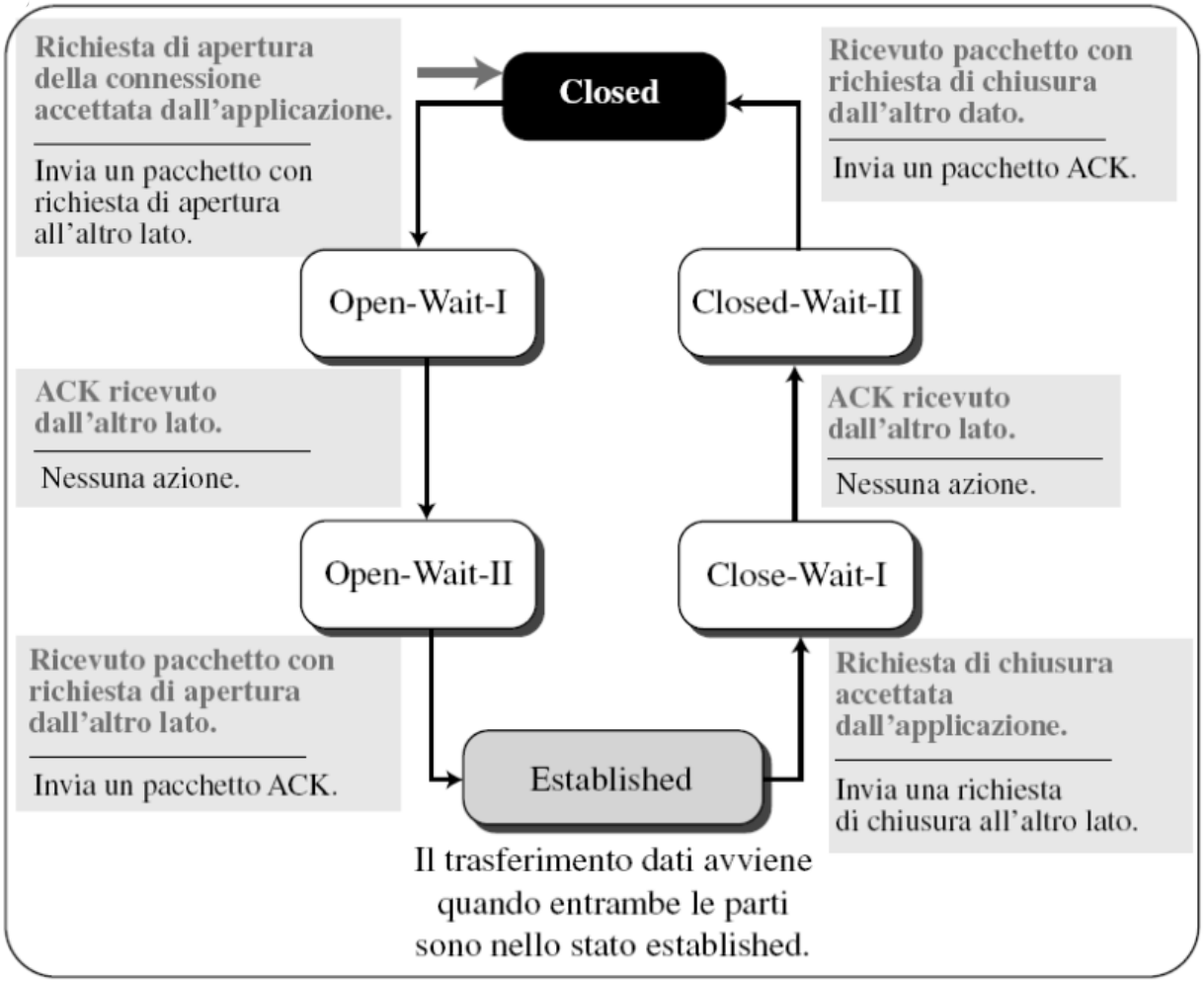
- Servizio *end-to-end*
- Viene stabilita una *connessione logica* prima di scambiarsi i dati
- E' possibile implementare controllo di flusso, controllo degli errori e controllo della congestione



Rapprensentazione mediante FSM (Finite state machine)

RAPPRESENTAZIONE DI UN SERVIZIO ORIENTATO ALLA CONNESSIONE

FSM per il livello trasporto orientato alla connessione



Controllo di flusso

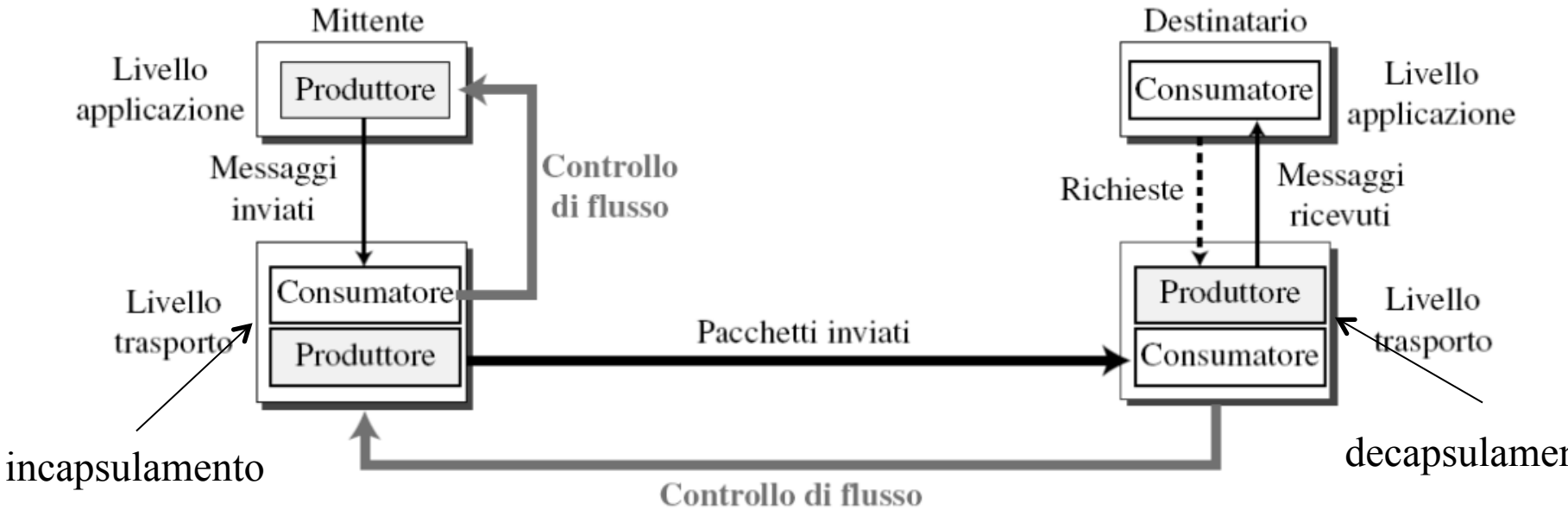
- Quando un'entità produce dati che un'altra entità deve consumare, deve esistere un *equilibrio fra la velocità di produzione e la velocità di consumo dei dati*
- Se velocità di produzione > velocità di consumo

  ➔ il consumatore potrebbe essere sovraccaricato e costretto a eliminarne alcuni
- Se velocità di produzione < velocità di consumo

  ➔ il consumatore rimane in attesa riducendo l'efficienza del sistema
- ***Il controllo di flusso è legato alla prima problematica per evitare di perdere dati***

Controlli di flusso a livello trasporto

- 4 entità: processo mittente, trasporto mittente, trasporto destinatario, processo destinatario
- 2 casi di controllo di flusso



Controllo di flusso

- Soluzione per realizzare controllo di flusso?
  - **Buffer** (insieme di locazioni di memoria che possono contenere pacchetti)
  - La comunicazione delle informazioni di controllo del flusso può avvenire inviando segnali *dal consumatore al produttore*
  - Il livello trasporto del mittente segnala al livello applicazione di sospendere l'invio di messaggi quando ha il buffer saturo: quando si libera spazio nel buffer segnala al livello applicazione che può riprendere l'invio di messaggi.
  - Il livello trasporto del destinatario segnala al livello trasporto del mittente di sospendere l'invio di messaggi quando ha il buffer saturo: quando si libera spazio nel buffer segnala al livello trasporto mittente che può riprendere l'invio di messaggi.

Controllo degli errori

- Poiché il livello di rete è inaffidabile, è necessario implementare l'*affidabilità* al livello di trasporto
- Per avere un servizio di trasporto affidabile è necessario implementare un *controllo degli errori*
  - Rilevare e scartare pacchetti corrotti
  - Tenere traccia dei pacchetti persi e gestirne il rinvio
  - Riconoscere pacchetti duplicati e scartarli
  - Bufferizzare i pacchetti fuori sequenza finche' arrivano i pacchetti mancanti
- Il controllo degli errori coinvolge solo i livelli trasporto mittente e destinatario (i messaggi scambiati tra livelli sono esenti da errori)
- Il livello trasporto del destinatario gestisce il controllo degli errori segnalando il problema al livello trasporto del mittente

Controllo degli errori

- Il mittente deve sapere quali pacchetti ritrasmettere e il destinatario deve saper riconoscere pacchetti duplicati e fuori sequenza
- Numerazione dei pacchetti con numero di sequenza (campo all'interno dell'header)
- Numerazione sequenziale
- Poichè il numero di sequenza deve essere inserito nell'intestazione del pacchetto, occorre specificarne la dimensione massima
  - Se l'intestazione prevede m bit per il numero di sequenza, questi possono assumere i valori da 0 a 2^m-1
  - Esempio m=4, numeri di sequenza da 0 a 15
  - I numeri di sequenza sono considerati in modulo 2^m

Controllo degli errori

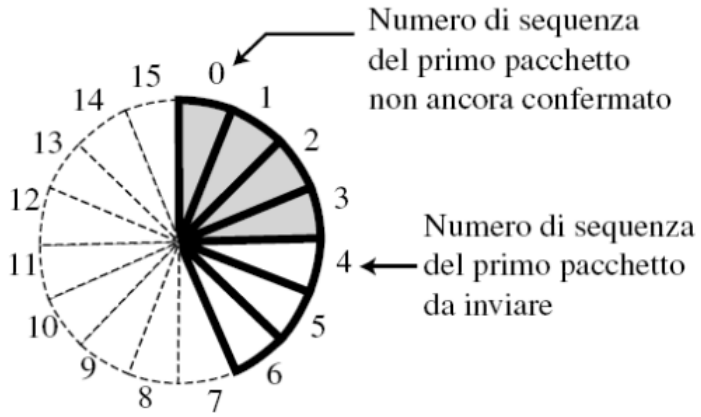
- Il numero di sequenza è utile al destinatario per capire
  - La sequenza di pacchetti in arrivo
  - Pacchetti persi
  - Pacchetti duplicati
- ma come può il mittente capire che un pacchetto è andato perso?
- Numero di riscontro (*acknowledgment*, *ack*, o *conferma*) permette di notificare al mittente la corretta ricezione di un pacchetto
- Il destinatario può scartare i pacchetti corrotti e duplicati

Integrazione del controllo degli errori e controllo di flusso

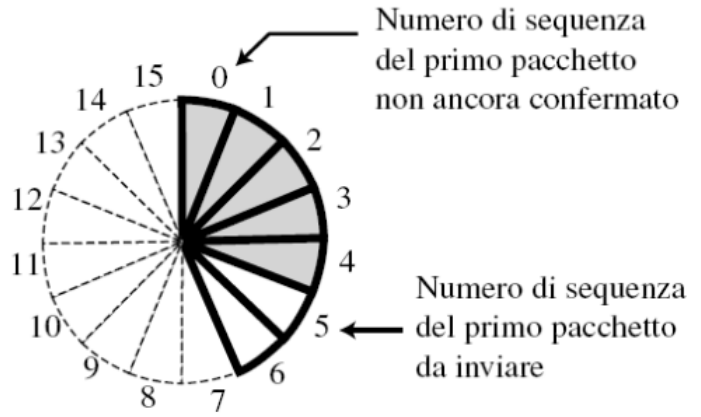
- Controllo di flusso richiede due buffer (mittente e destinatario)
- Controllo degli errori richiede numero di sequenza e ack
- Combinazione dei due meccanismi mediante buffer numerato (presso mittente e destinatario)
- Mittente:
  - Quando prepara un nuovo pacchetto usa come numero di sequenza il numero (x) della prima locazione libera nel buffer
  - Quando invia il pacchetto ne memorizza una copia nella locazione x
  - Quando riceve un ack di un pacchetto libera la posizione di memoria che era occupata da quel pacchetto
- Destinatario:
  - Quando riceve un pacchetto con numero di seq. y, lo memorizza nella locazione y fin quando il livello applicazione è pronto a riceverlo
  - Quando passa il il pacchetto y al livello applicazione invia ack al mittente

- Poiché i numeri di sequenza sono calcolati in modulo 2^m, possono essere rappresentati con un cerchio
- Il buffer viene rappresentato con un insieme di settori, chiamati **finestra scorrevole** o **sliding windows**, che in ogni istante occupano una parte del cerchio

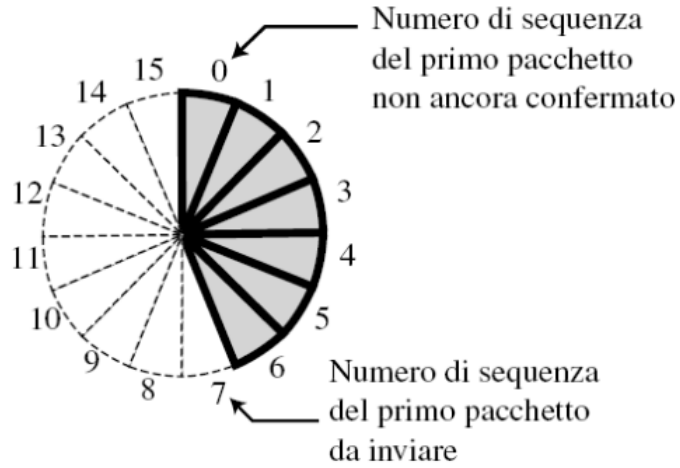
Finestra di dimensione 7 presso mittente



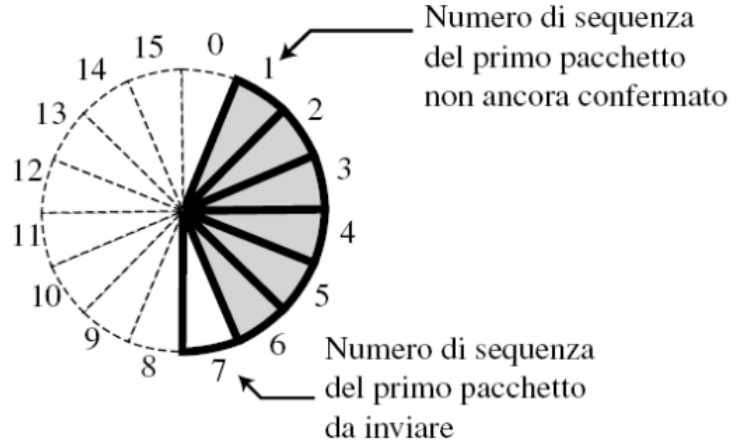
a. Sono stati inviati quattro pacchetti.



b. Sono stati inviati cinque pacchetti.



c. Sono stati inviati sette pacchetti e la finestra è completa.



d. Il pacchetto 0 è stato confermato, la finestra scorre di una posizione.

Rappresentazione lineare della finestra scorrevole

- La finestra scorrevole è solo una rappresentazione lineare, nella realtà si usano variabili per contenere i numeri di sequenza del pacchetto successivo da inviare e dell'ultimo pacchetto inviato
- In genere la finestra scorrevole viene rappresentata linearmente

a. Sono stati inviati quattro pacchetti.

b. Sono stati inviati cinque pacchetti.

c. Sono stati inviati sette pacchetti e la finestra è completa.

d. Il pacchetto 0 è stato confermato, la finestra scorre di una posizione.

Controllo della congestione

- Nella commutazione a pacchetto, la congestione avviene se il *carico* della rete (numero di pacchetti inviati alla rete) è superiore alla *capacità* della rete (numero di pacchetti che la rete può gestire)
- *Controllo della congestione*: meccanismi e tecniche per controllare la congestione mantenendo il carico della rete al di sotto della sua capacità
- Perché congestione?
  - Se router e switch non riescono a elaborare i pacchetti alla stessa velocità con cui arrivano, le code si sovraccaricano e avviene la congestione