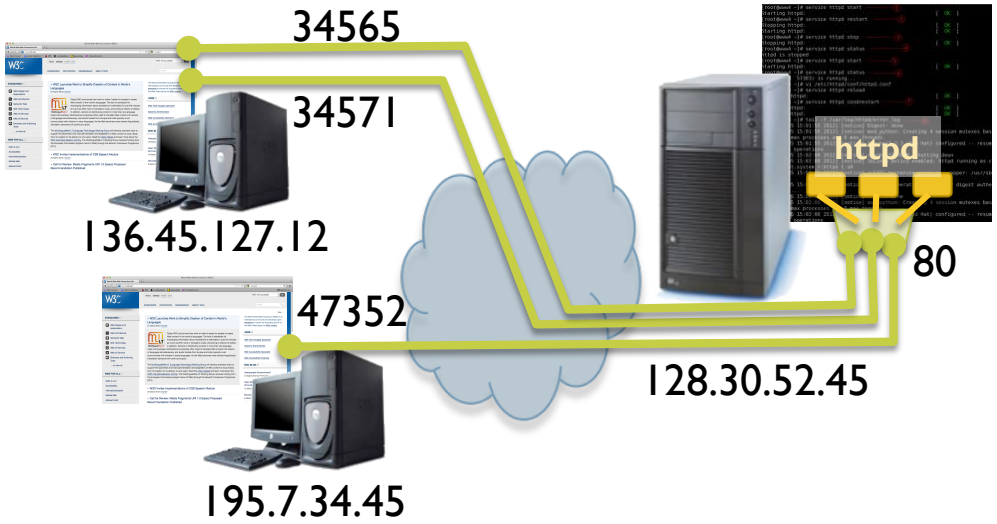
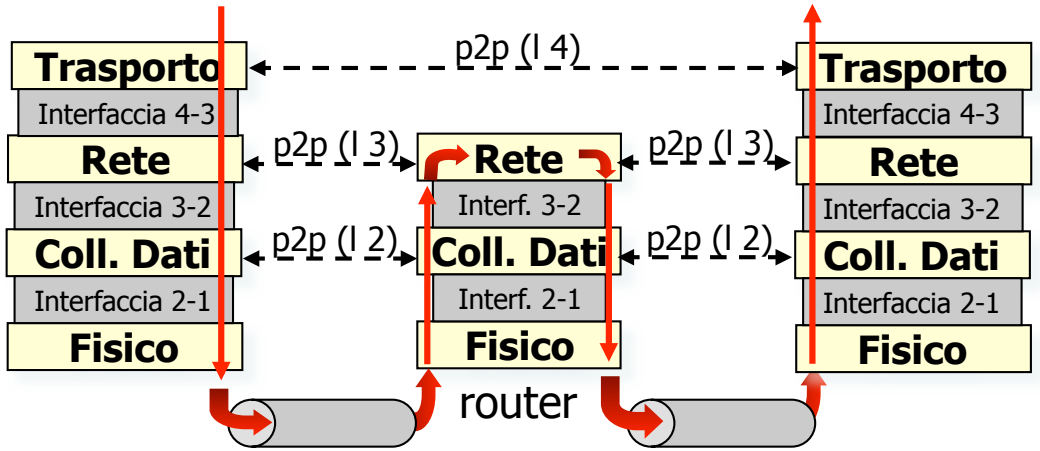


Reti di Calcolatori

Transport Layer & TCP/UDP

Il livello di trasporto



- Fornisce un servizio per la comunicazione logica fra processi applicativi in host su rete
  - E' una funzionalità del sistema operativo disponibile come libreria API con cui si possono implementare meccanismi di scambio dati fra applicazioni (socket API)
  - Presuppone un modello client-server
  - Gestisce l'indirizzamento verso dei processi (indirizzo di porta) e il multiplexing/demultiplexing dei messaggi (indirizzo di socket)

Servizio di trasporto

- Il livello di trasporto può offrire più modalità per lo scambio dati fra due applicazioni
  - Servizio per stream dati (con connessione)
    - fornisce un canale affidabile su cui scrivere o da cui leggere dati
    - dal punto di vista del programmatore è come un file
    - deve gestire il controllo degli errori, la perdita di pacchetti, i numeri di sequenza per l'invio del flusso dati come sequenza di datagram e il controllo di flusso per un collegamento attraverso una rete inaffidabile
    - deve nascondere il problema della capacità di memorizzazione della rete (un pacchetto può essere memorizzato in un router e consegnato dopo un certo ritardo)
  - Servizio datagram (senza connessione)
    - fornisce il meccanismo di indirizzamento verso le applicazioni (porte)
    - replica le funzionalità di consegna del servizio di consegna dei pacchetti (datagram) del livello di rete
    - può aggiungere il controllo a rilevazione di errore (con checksum)

Tipi di servizi e QoS

- Le modalità con connessione (stream) e senza connessione (datagram) sono modalità di base
  - Disponibili per lo sviluppo di applicazioni su rete Internet (TCP/UDP)
  - In generale potrebbero essere sviluppate tipologie di trasporto con funzionalità/garanzie aggiuntive, come ad esempio
    - servizio con flusso dati garantito
    - servizio con ritardo dati massimo limitato
  - Un'applicazione potrebbe richiedere la tipologia di servizio di trasporto in base al tipo di esigenze richieste
    - La conoscenza del livello di qualità del servizio (Quality of Service) richiesto può essere utilizzata per ottimizzare l'uso delle risorse di rete
    - Le richieste possono non essere accettate se il sistema non è in grado di soddisfarle (col carico attuale)
    - Il tipo di servizi che si possono offrire dipendono anche dalle caratteristiche del livello di rete (es. gestione del ritardo massimo)

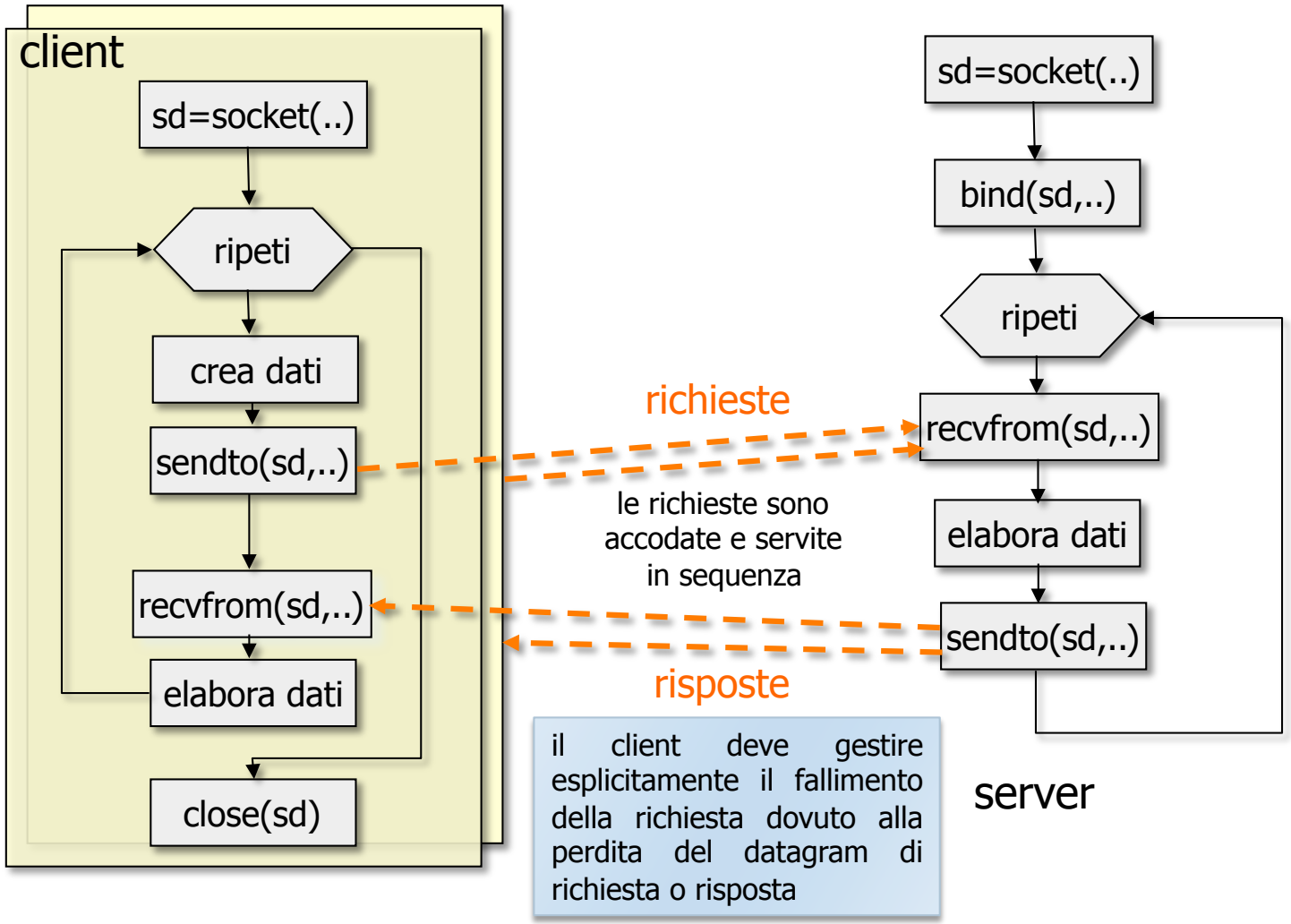
Modello client-server e API - 1

- L'interfaccia API del servizio di trasporto assume il modello client-server
  - L'agente server attiva una procedura che rimane in attesa di richieste da parte di un agente client (LISTEN)
    - Il server effettua una richiesta al Sistema Operativo (SO) per la creazione di un end-point (server socket) specificando il tipo di servizio richiesto (stream/dgram)
    - Il processo server specifica la porta su cui si metterrà in ascolto
      - La porta deve essere non già in uso e nota al processo client
    - La procedura crea le strutture dati a livello di SO per la gestione della comunicazione col livello di servizio richiesto (buffer, code di richiesta di connessione, strutture per il multiplexing di più canali di comunicazione,...)
    - Per il servizio stream il server rimane in attesa di accettare una richiesta di connessione (ACCEPT)
    - Per il servizio datagram rimane in attesa dell'arrivo di un pacchetto datagram da un client (RECVFROM)

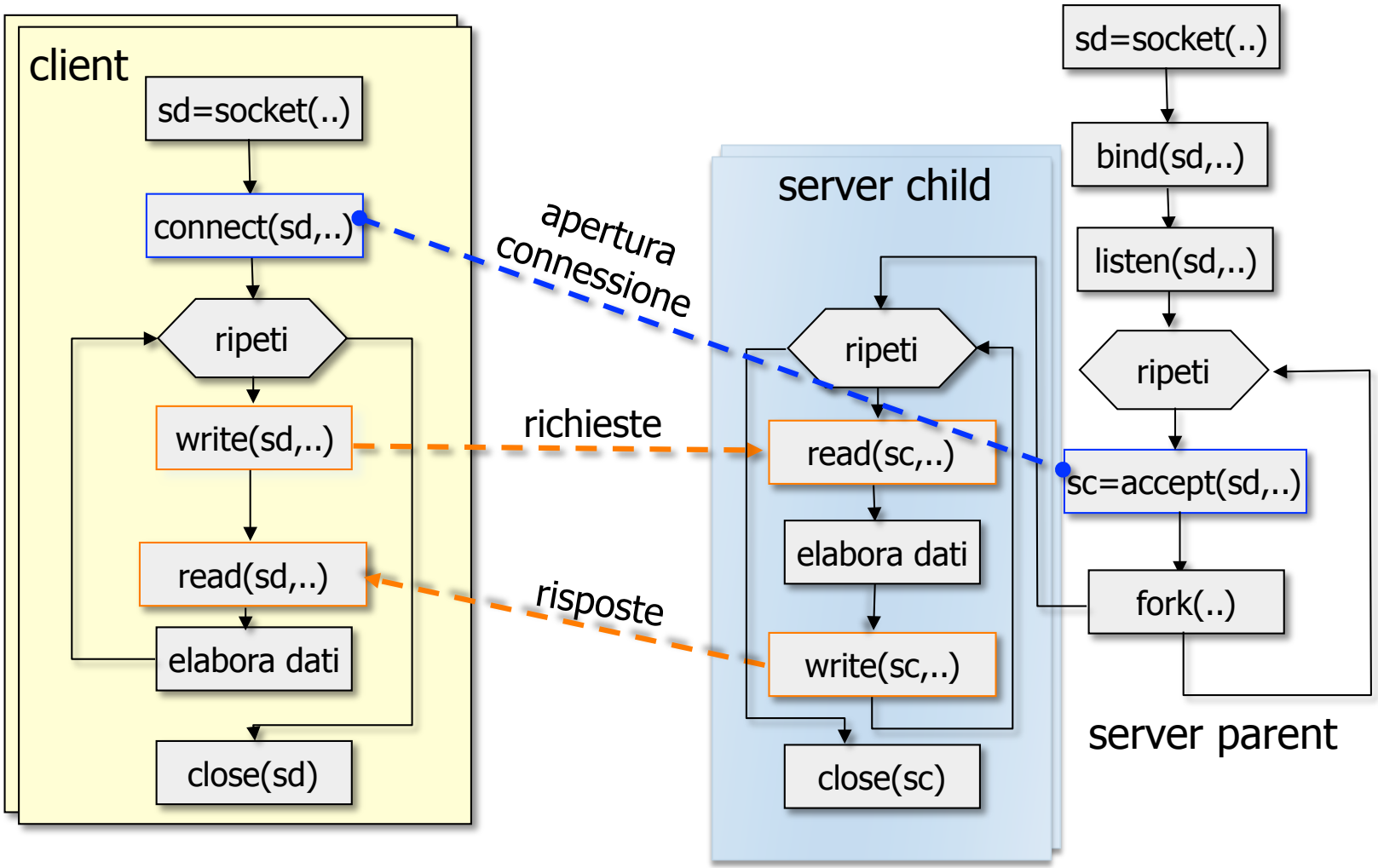
Modello client-server e API - 1

- Dopo che il server ha inzializzato l'end-point e si è messo in attesa
  - Il processo server è bloccato in attesa di un evento generato da un client
    - Richiesta di connessione/invio di un datagram
  - L'agente client invia una richiesta al server usando il suo indirizzo di rete e di trasporto (IP+porta)
    - Può essere una richiesta di connessione (CONNECT) o l'invio di un messaggio datagram (SENDTO) in base al tipo di servizio usato
  - Per i servizi stream con connessione, client e server si scambiano dati leggendo/scrivendo (SEND/RECEIVE) sul canale (socket)
    - Al termine dello scambio la connessione viene chiusa (DISCONNECT)

Client-server datagram



Client-server stream

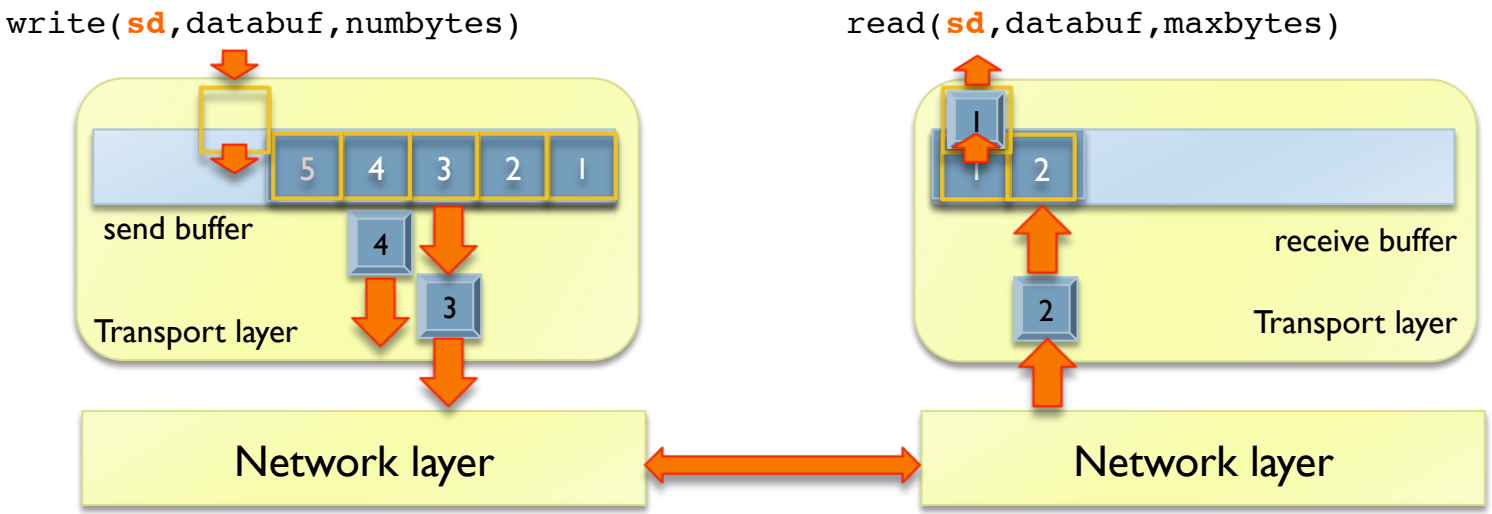


TCP e UDP

- Protocolli di trasporto definiti su rete Internet (su IP)
  - Trasmission Control Protocol (TCP)
    - protocollo di trasporto orientato alla connessione
    - definito in RFC 793, RFC 1122 e RFC 1323
    - progettato per fornire un flusso affidabile end-to-end su una rete internet non affidabile (consegna bets-effort)
    - include meccanismi per il controllo di flusso e della congestione
  - User Datagram Protocol (UDP)
    - protocollo senza connessione che invia messaggi in datagram IP indipendenti
    - descritto in RFC 768
  - L'indirizzo di trasporto è il numero di porta a 16 bit
    - TCP e UDP usano due spazi di porte distinti
    - La differenza è definita da un campo dell'intestazione IP che indica il protocollo usato

Servizio stream con TCP

- TCP implementa un trasferimento affidabile di un flusso dati
  - Il flusso è scomposto in datagram
  - I datagram sono inviati al destinatario usando un servizio di rete non affidabile
    - I datagram possono essere persi
    - I datagram posso arrivare in un ordine diverso da quello di spedizione
    - Il contenuto dei datagram si può corrompere

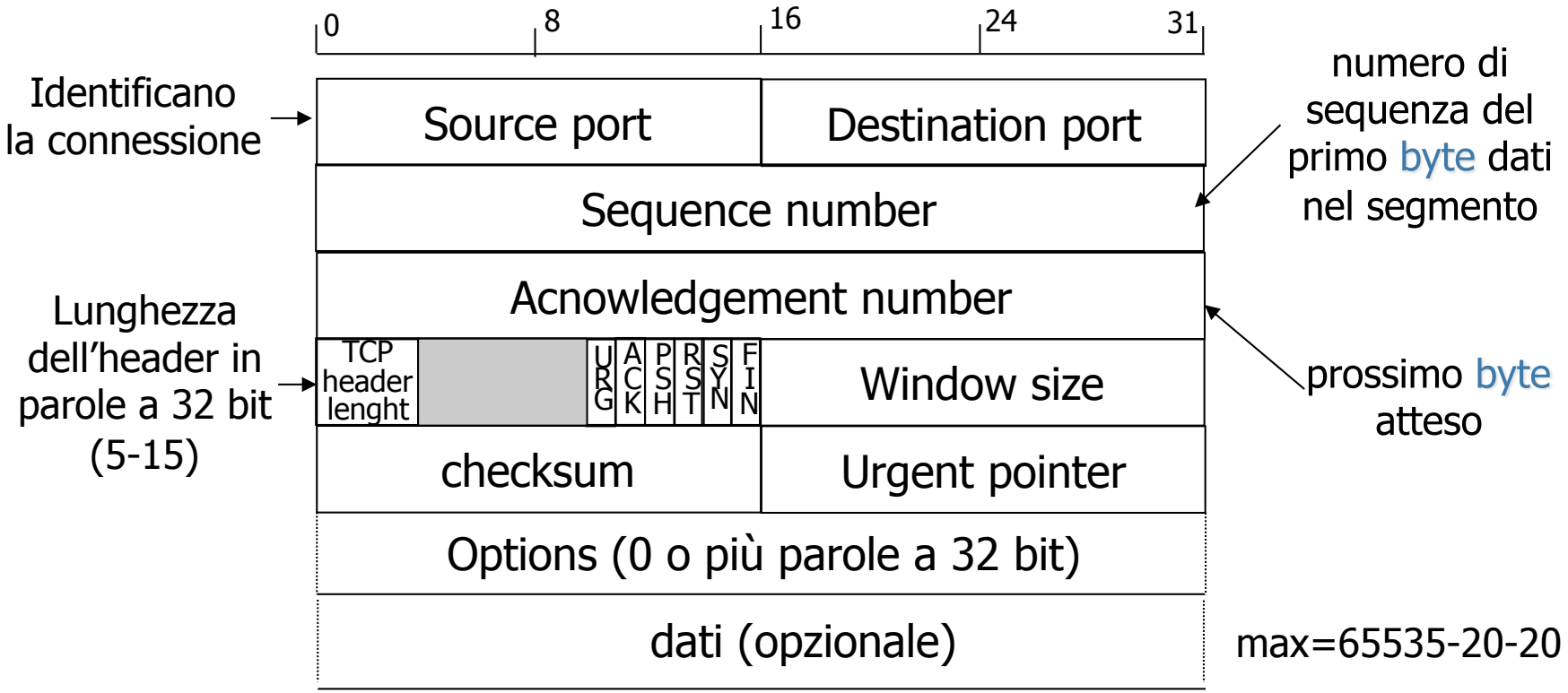


Affidabilità del servizio

- **Perdita dei pacchetti**
  - Per garantire l'arrivo di tutti i pacchetti viene adottato un meccanismo basato sul riscontro (ack) della ricezione
    - Il ricevente invia un ack in risposta alla corretta ricezione di un pacchetto
    - Il mittente ritrasmette il pacchetto se non riceve l'ack prima dello scadere di un timeout (ritrasmissione – Automatic Repeat Request)
      - Il mittente rimanda il pacchetto fino a che non è sicuro della sua corretta ricezione
      - Il meccanismo gestisce sia la perdita del pacchetto che del suo ack
      - Dopo un certo numero di ritrasmissioni fallite si genera un errore
- **Ordinamento e duplicazione**
  - Il protocollo prevede la numerazione dei pacchetti e dei relativi ack
    - Il numero d'ordine identifica univocamente i dati nella connessione e permette di riconoscere pacchetti duplicati (si scartano)
- **Controllo di errore**
  - Un codice a rilevazione di errore permette di scartare i pacchetti corrotti (sono ritrasmessi quando scade il timeout del mittente)

I segmenti TCP

- Ogni segmento ha un header fisso di 20 byte più eventuali parti opzionali seguito da 0 o più byte di dati



TCP flag bits - 1

- Nel segmento TCP sono presenti 6 bit di flag
  - URG
    - Indica che l'Urgent Pointer specifica la posizione di dati urgenti a partire dal numero di sequenza attuale (es. pressione di CTRL-C per interrompere il programma remoto)
  - ACK
    - Indica se il campo Acknowledgement number è valido
  - PSH
    - Indica dati di tipo PUSH ovvero si richiede di consegnare subito i dati senza bufferizzarli (la funzione read esce)
  - RST
    - Richiesta di reinizializzazione di una connessione diventata instabile
    - Viene anche usato per rifiutare un segmento non valido o l'apertura di una connessione

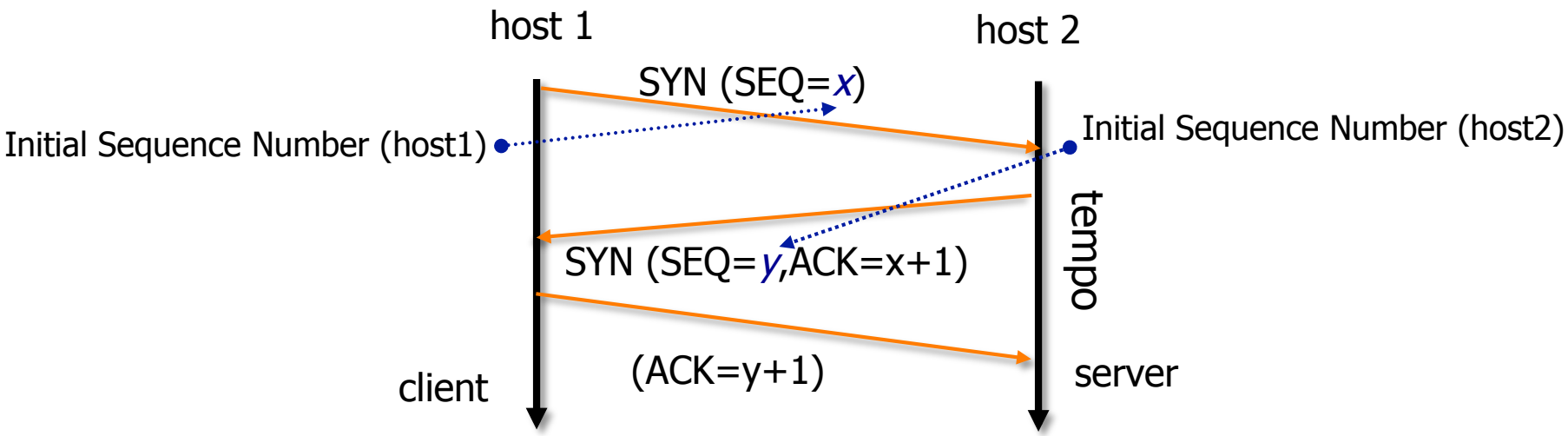
TCP flag bits 2

- SYN
  - Viene utilizzato per creare connessioni
  - La richiesta di connessione è caratterizzata da SYN=1 e ACK=0
  - La risposta di connessione contiene un ack e quindi ha SYN=1 e ACK=1
  - Individua i segmenti TCP CONNECTION REQUEST e CONNECTION ACCEPTED
- FIN
  - Viene utilizzato per chiudere una connessione (il mittente non ha altri dati da spedire)

- I flag ACK, SYN, FIN, RST sono usati nelle procedure di gestione della connessione
- I flag URG, PSH sono usati per richieste particolari nello scambio dati

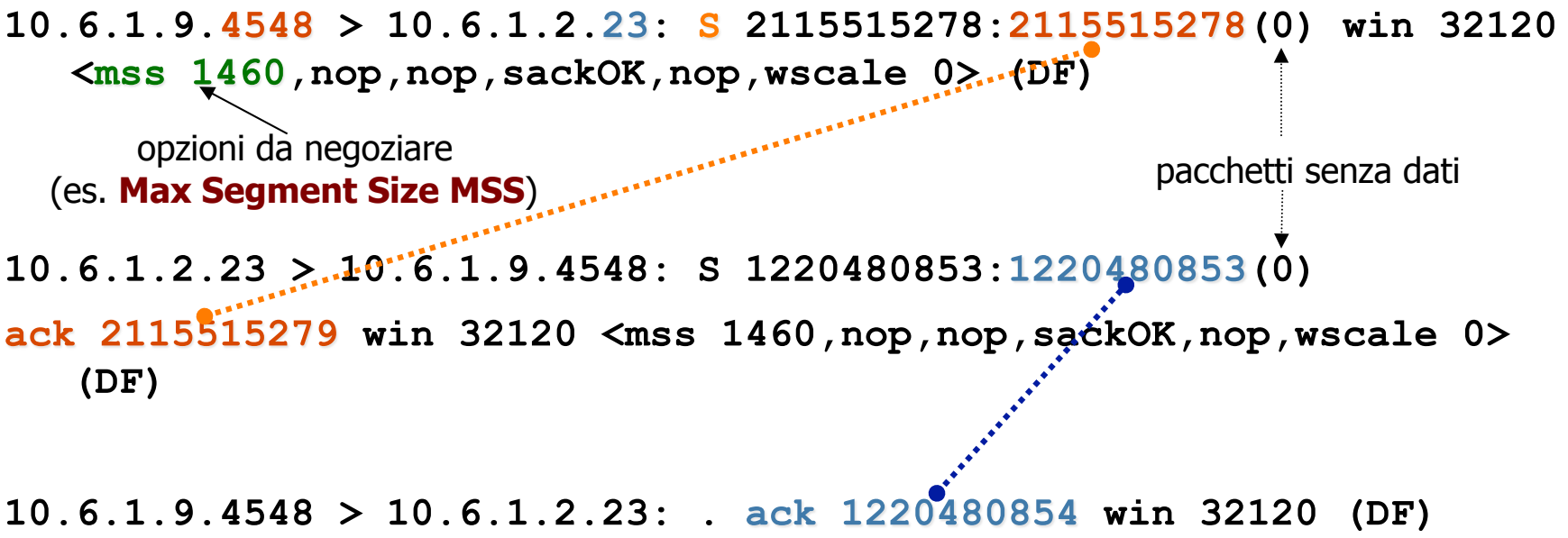
Apertura della connessione

- Si utilizza un protocollo 3-way handshake
  - Sincronizza la numerazione dei pacchetti (valore iniziale)
    - La specifica richiede che il numero iniziale sia casuale in modo da evitare che si possano scambiare segmenti relativi a due connessioni distinte con gli stessi parametri e aperte a breve distanza di tempo
    - I segmenti possono essere memorizzati nella rete ma in ogni caso la loro vita non può eccedere la Maximum Segment Lifetime (MSL) di 2 minuti
  - Il TCP ricevente invia un segmento di rifiuto della connessione (RST) se non esiste un processo in attesa sulla porta destinazione



Un esempio di connessione

- Connessione Telnet fra da 10.6.1.9 a 10.6.1.2 catturata con **`tcpdump`** *
  - porta client 4548 - porta server 23 (telnet)



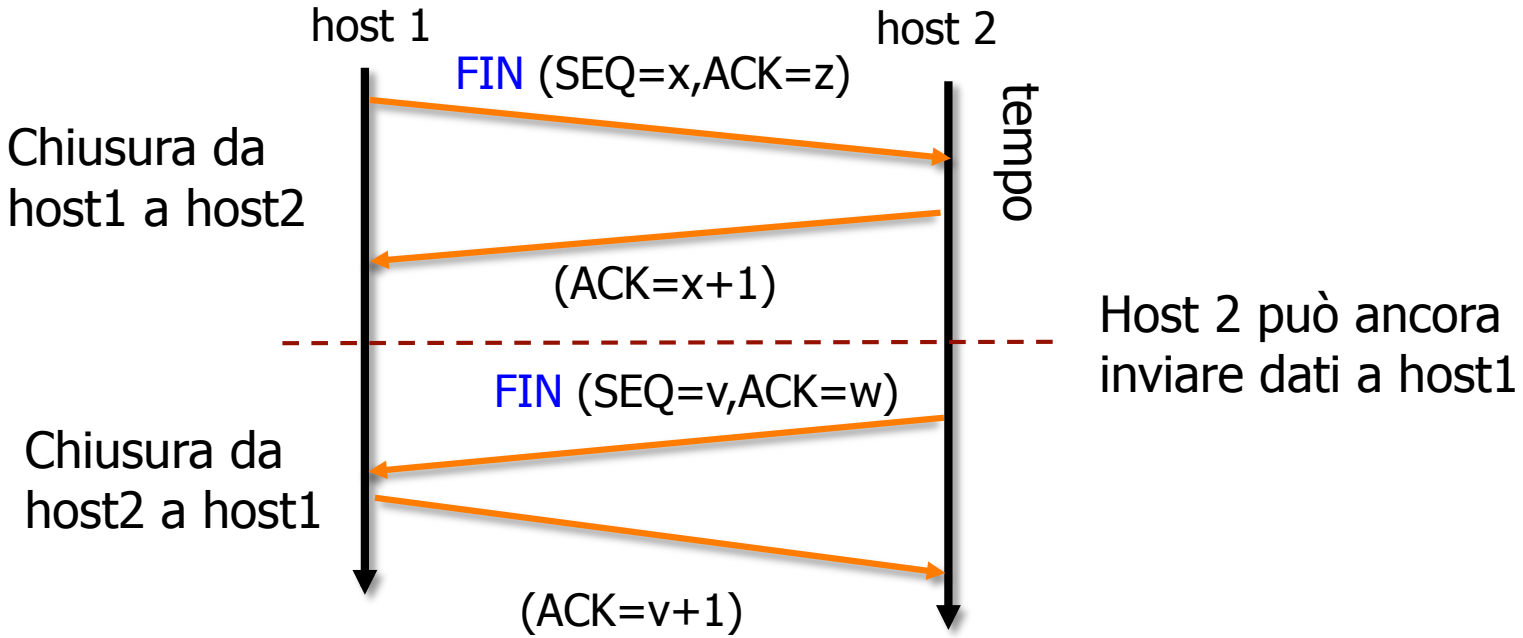
* `tcpdump -S -n -t \(dst 10.6.1.2 and src 10.6.1.9\) or \(dst 10.6.1.9 and src 10.6.1.2\)`

Sequence Number Wrap Around

- Lo spazio dei numeri di sequenza è finito [0,2^{32}-1]
  - Si ripresentano gli stessi numeri di sequenza dopo 4Gb
- Può essere un problema quando
  - Si trasferiscono flussi dati più lunghi di 4Gb
  - La velocità del trasferimento è tale che la numerazione si ripresenta prima dello scadere del Maximum Segment Lifetime
    - Per una linea a 1.2Gbps il Wrap Around si ha in circa 28s
- Protection Against Wrapped Sequence (PAWS)
  - Viene utilizzata l'opzione Timestamp che aggiunge un ulteriore campo a 32 bit che permette di individuare l'ordine con cui sono stati generati i pacchetti
  - In base al Timestamp è possibile ordinare correttamente i segmenti anche in presenza di reinizializzazione dei numeri di sequenza

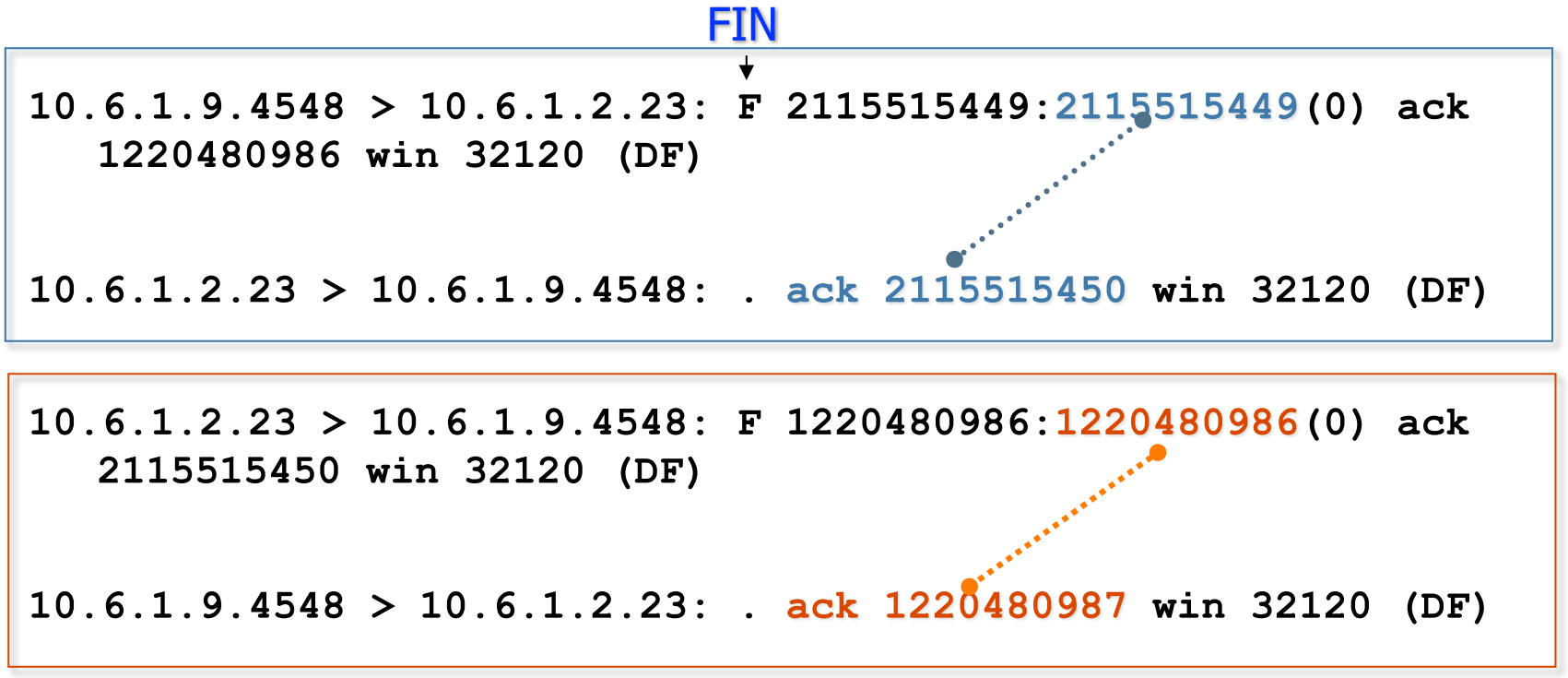
Chiusura della connessione

- La connessione è full-duplex e le due direzioni devono essere chiuse indipendentemente
- Se l'ack di un messaggio FIN si perde, l'host mittente chiude comunque la connessione dopo un timeout



Un esempio di chiusura

- Chiusura Telnet da 10.6.1.9
  porta client 4548 - porta server 23 (telnet)



Tipologie di flusso dati

- Il flusso di dati ha caratteristiche che dipendono dall'applicazione
  - Flussi a bassa velocità
    - I dati da trasmettere sono prodotti da un utente (pochi byte/s)
    - E' richiesta interattività, ovvero l'utente deve avere il feedback sui input con una bassa latenza
    - I segmenti inviati portano pochi dati perché non si possono usare tecniche di bufferizzazione su lunghi intervalli di tempo
    - Si devono adottare teniche per massimizzare il payload
  - Flussi ad alta velocità
    - I dati sono il prodotto di un'elaborazione
    - I dati possono essere inviati alla massima velocità perché il buffer di invio è "sempre" pieno
    - Occorre regolare il flusso per evitare di saturare la capacità del ricevente (riempire il buffer di ricezione) o della rete (riempire le code dei router)

Flussi di dati interattivi

- Una connessione interattiva è caratterizzata da una **bassa frequenza dei dati** da trasmettere
  - I dati sono generati dall'azione di un utente umano
    - Sessione di terminale remoto (es. telnet o ssh)
  - L'applicazione è interattiva
    - Il sistema di trasmissione non può introdurre latenze che riducono la percezione dell'interattività
    - Non si possono accumulare i dati ma occorre inviare segmenti piccoli
  - Il carico utile (**payload**) dei segmenti TCP è di pochi byte
    - Il 90% dei segmenti telnet porta circa 10 byte
    - Nel caso limite si ha un segmento per ogni carattere battuto
      - In genere si ha anche un echo del carattere battuto da parte del destinatario con un ulteriore segmento con un solo carattere

Esempio: sessione telnet

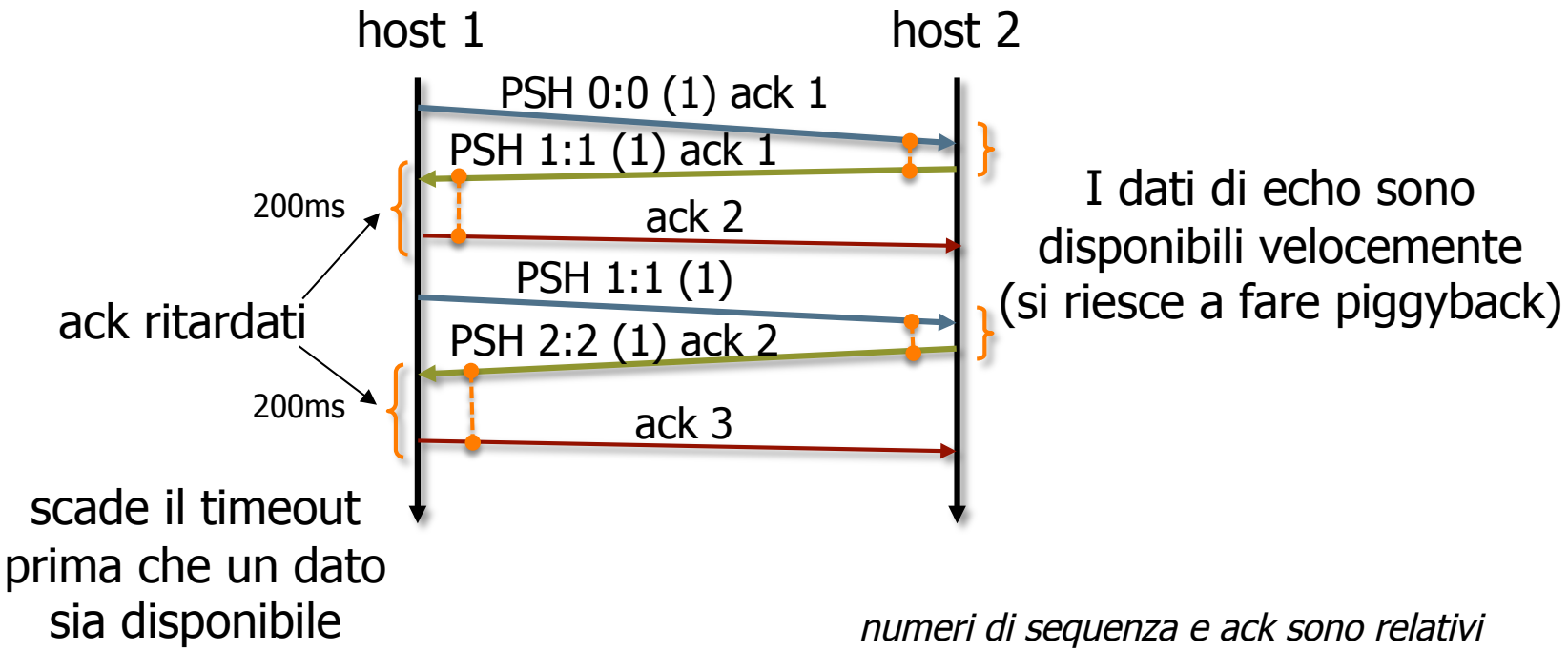
- Quando l'utente batte un tasto in una sessione telnet
  - Segmento dal client col carattere battuto
    - 20IP+20TCP+1byte = 41byte
  - Segmento di ack dal server al client
    - 20IP+20TCP = 40 byte
  - Segmento di echo dal server
    - 20IP+20TCP+1byte = 41byte
  - Segmento di ack dal client
    - 20IP+20TCP = 40 byte

si possono unire...

- In totale si userebbero 162 (122) byte in 4 (3) segmenti TCP per 1 carattere!!

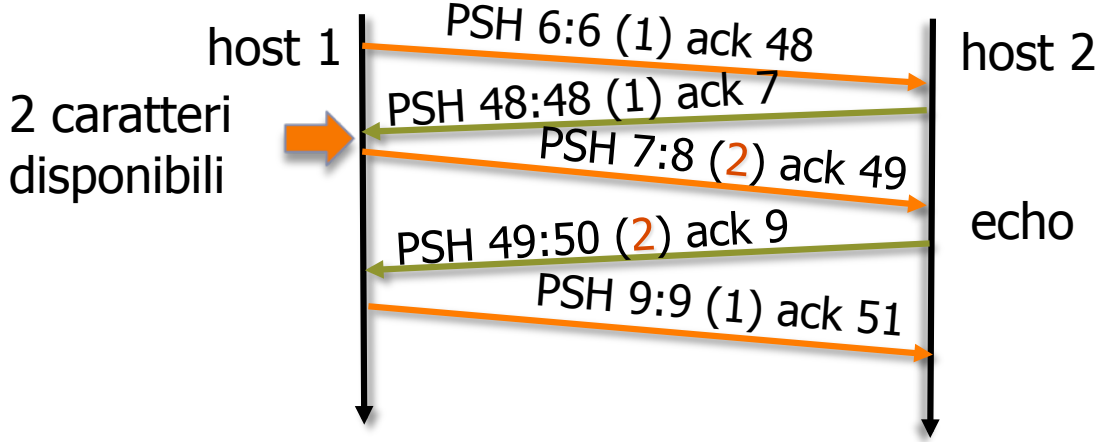
Ack ritardati

- Normalmente il TCP non invia un ack istantaneamente ma ritarda l'invio sperando di avere dati da spedire con l'ack (ACK piggyback)
  - Molte implementazioni usano un ritardo di 200ms



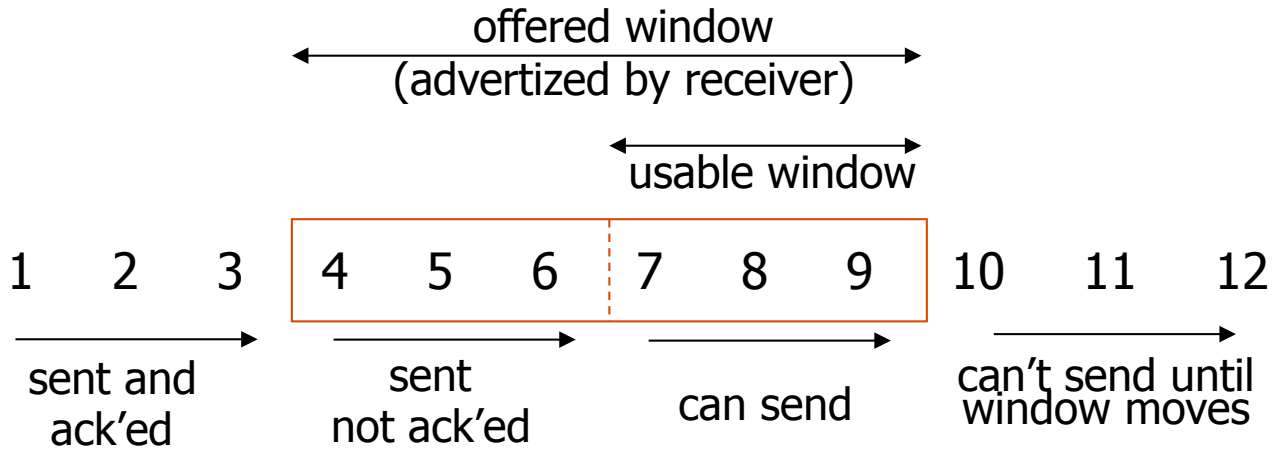
Algoritmo di Nagle (1984)

- Per aumentare il payload si possono bufferizzare i dati generati dall'applicazione per un certo tempo
  - Si può utlizzare un timer fisso (che valore?)
  - Nagle ha proposto un semplice algoritmo auto-temporizzato
    - Si accumulano i dati fino a che non si riceve l'ack per il segmento inviato in precedenza
    - Se non si sta attendendo un ack o sono disponibili dati per la Maximum Segment Size (MSS) negoziata per la connessione si inviano, i dati disponibili
    - Crea una latenza mascherata dal ritardo di trasmissione
    - Può (e deve) essere disabilitato per alcuni casi (es. mouse in Xwindows)



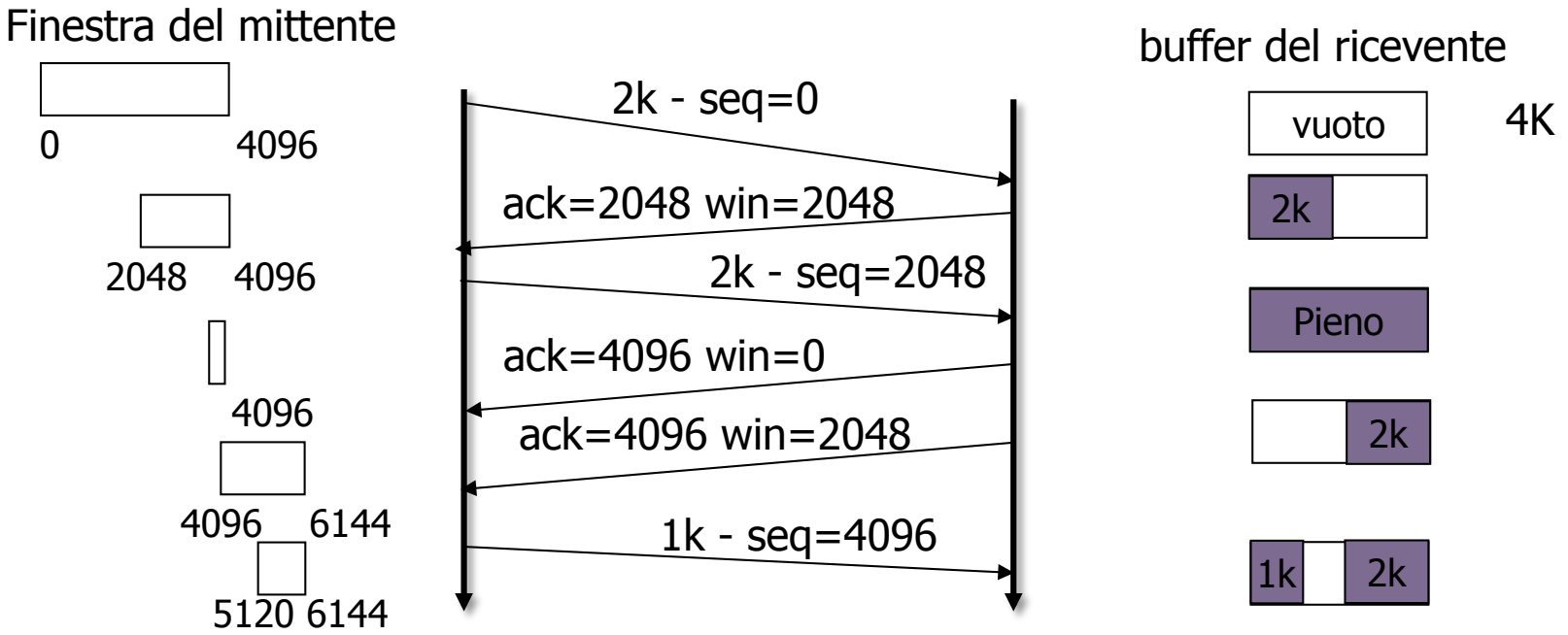
Regolazione del flusso di dati

- Viene utilizzato un protocollo a finestra scorrevole (sliding window) con dimensione variabile
  - Il ricevente indica la dimensione della finestra (quantità dei dati) che può gestire in un dato momento
  - La dimensione dipende dallo spazio disponibile nel buffer di ricezione
  - L'evoluzione della dimensione dipende dalla capacità dell'applicativo destinatario di elaborare e quindi leggere i dati dal buffer di ricezione
- La finestra di dati trasmissibili ancora senza aspettare l'ack è ottenuta dall'ampiezza della finestra e dal numero dell'ultimo byte ricevuto
  - Il mittente può continuare a inviare dati senza attendere l'ack fino a che non esaurisce la finestra disponibile



Finestra scorrevole

- Se il ricevente è saturo indica una finestra 0 e il mittente non può trasmettere dati
  - Il mittente può inviare periodicamente un segmento di un byte per forzare il destinatario a indicare il prossimo byte atteso e l'ampiezza della finestra (per non rimanere in attesa infinita se si perdono pacchetti)
  - Non conviene notificare subito la disponibilità della finestra se è piccola (silly window syndrome)



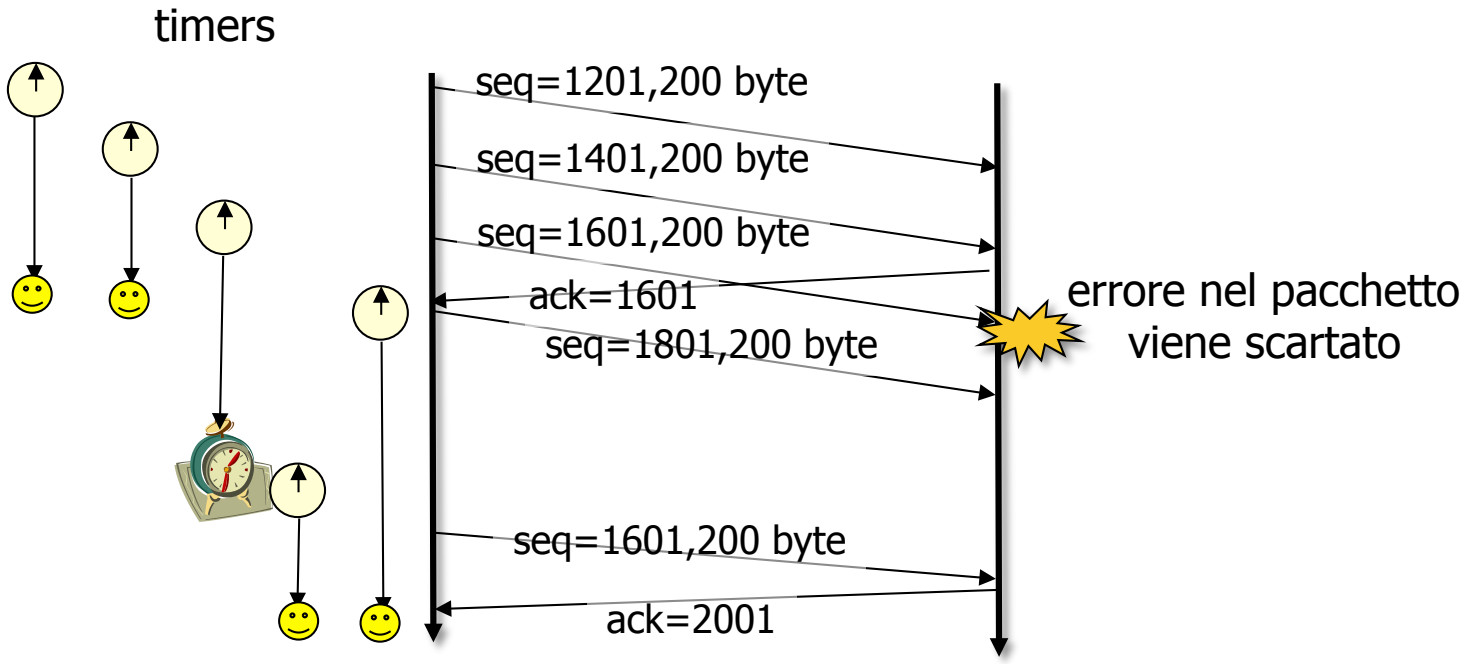
Un esempio di trasmissione

- Connessione Telnet fra da 10.6.1.9 a 10.6.1.2 catturata con tcpdump porta client 4548 - porta server 23 (telnet)
  - La finestra offerta è di circa 32k e pari alla dimensione del buffer di ricezione

```
10.6.1.9.4548 > 10.6.1.2.23: P 2115515279:2115515306(27) ack 1220480854 win 32120
10.6.1.2.23 > 10.6.1.9.4548: . ack 2115515306 win 32120
10.6.1.2.23 > 10.6.1.9.4548: P 1220480854:1220480866(12) ack 2115515306 win 32120
10.6.1.9.4548 > 10.6.1.2.23: . ack 1220480866 win 32120
10.6.1.2.23 > 10.6.1.9.4548: P 1220480866:1220480905(39) ack 2115515306 win 32120
10.6.1.9.4548 > 10.6.1.2.23: P 2115515306:2115515443(137) ack 1220480905 win 32120
10.6.1.2.23 > 10.6.1.9.4548: P 1220480905:1220480908(3) ack 2115515443 win 32120
10.6.1.9.4548 > 10.6.1.2.23: P 2115515443:2115515446(3) ack 1220480908 win 32120
10.6.1.2.23 > 10.6.1.9.4548: . ack 2115515446 win 32120
```

Gestione della perdita di segmenti

- Il mittente ha un timeout di attesa dell'ack per i pacchetti spediti
  - Se il timeout scade il pacchetto viene ritrasmesso
  - Permette di gestire la perdita di pacchetti (non arrivati o arrivati con errori)



TCP timeout e ritrasmissione

- TCP utilizza un timeout di attesa dell'ack dopo di che provvede alla ritrasmissione dei dati
- Il problema è determinare il valore del timeout migliore (i ritardi possono essere molto variabili nel tempo sulla rete)
  - Se il timeout è troppo piccolo si fanno ritrasmissioni inutili
  - Se il timeout è troppo elevato si avranno ritardi di trasmissione

  - Si utilizza un algoritmo di stima del migliore timeout basato sulla misura del Round-Trip Time (RTT)

Stima del timeout

- Per ogni connessione si tiene una stima di RTT, aggiornandola per ogni pacchetto con

$$RTT_i = \alpha\, RTT_{i-1} + (1-\alpha)\, T_{rtt}(pkt_i)$$

- Si stima poi la deviazione media

$$D_i = \alpha\, D_{i-1} + (1-\alpha)\, |RTT_i - T_{rtt}(pkt_i)|$$

- E si sceglie

$$timeout = RTT + 4*D$$

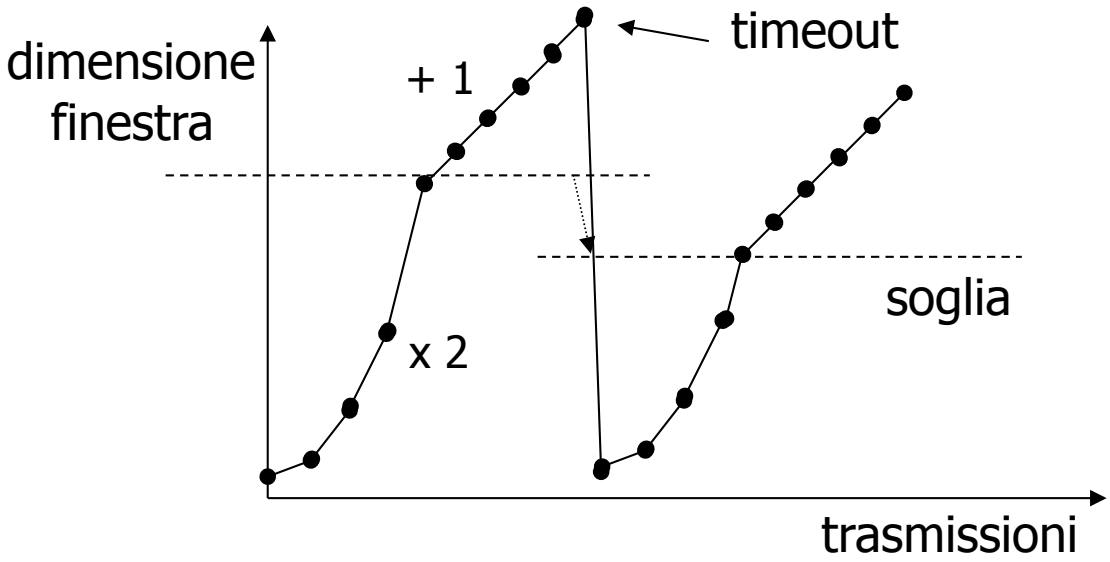
- Ci sono altre soluzioni (es. algoritmo di Karn – nella stima si usano solo i pacchetti non ritrasmessi)

Controllo di congestione

- Il TCP adatta la velocità di trasmissione alla capacità della rete
  - I pacchetti possono essere distrutti dai router se si riempiono i buffer di trasmissione/ricezione
  - Per i pacchetti distrutti nella rete non si riceve l'ack e quindi vengono ritramessi
  - La ritrasmissione è un potenzale fattore per incrementare ulteriormente la congestione
- Si utilizza una finestra di congestione per il mittente
  - La finestra effettiva è la minima fra quella di congestione (definita dalla rete) e quella di ricezione (definita dallo stato del buffer del ricevente)

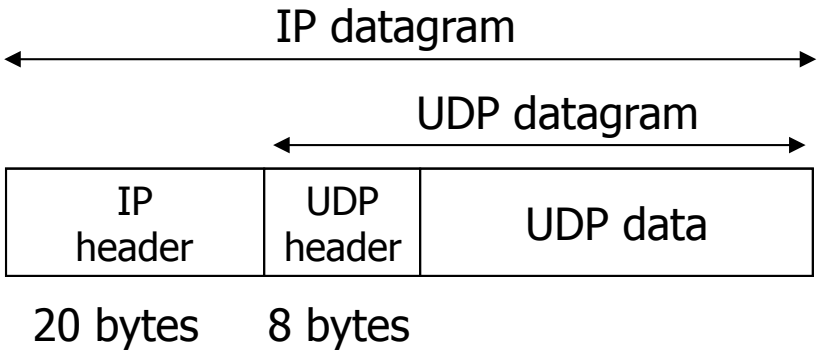
Finestra di congestione

- La dimensione della finestra di congestione è adattata dinamicamente
  - Cresce se non scade il timeout di trasmissione
  - Viene ridotta se scade il timeout di ritrasmissione

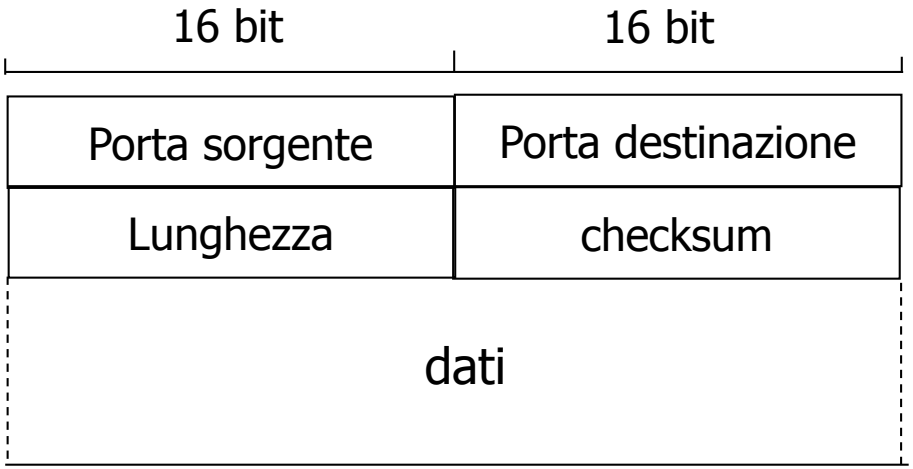


UDP

- Ogni operazione di output produce esattamente un datagram UDP che comporta l'invio di un datagram IP
  - UDP non garantisce affidabilità di consegna
  - UDP implementa il controllo di integrità con un codice a rilevazione di errore (checksum)
  - Se il datagram eccede la MTU (Maximum Transfer Unit) della rete fisica, esso viene frammentato dal livello di rete
  - Richiede meno overhead di una connessione TCP
    - Ha un header di dimensioni minori (8 byte invece di 20)
    - Non richiede la procedura di connessione
    - Non prevede l'invio di un ack



Header UDP



- La **porta destinazione** individua il processo destinatario (socket)
- La **porta sorgente** (insieme all'indirizzo IP) fornisce le informazioni necessarie ad inviare una eventuale risposta
- La **lunghezza in byte** comprende sia i dati che l'header (≥8)
  - Considerando l'header IP (20 byte) un pacchetto UDP può contenere fino a 65507 byte = 65535-20-8
- Il **checksum** comprende anche uno pseudoheader che contiene le informazioni IP (indirizzi IP, tipo di protocollo, dimensione)

Uso di UDP

- Implementazione esplicita del protocollo di scambio dati
  - Perdita della richiesta
    - L'applicazione che invia una richiesta deve prevedere il caso in cui si verifica un errore e la risposta non arriva
    - Ritrasmissione della richiesta allo scadere di un timeout di attesa
  - Associazione richiesta-risposta
    - Identificativo della richiesta da inviare nella risposta
    - La rete non garantisce che le risposte arrivino nello stesso ordine delle richieste
  - Controllo sull'invio/ricezione dei dati
    - Nessuna bufferizzazione. I datagram UDP sono inviati direttamente al livello di rete
    - Nessun controllo di flusso e di congestione. I datagram sono inviati alla velocità con cui sono generati
    - Nessun ack per ogni pacchetto inviato. L'ack è implicito nella ricezione di una risposta
      - L'ack può non essere necessario per le applicazioni in cui è tollerata la perdita di dati (es. streaming in tempo reale)