Reti di Calcolatori I

Prof. Roberto Canonico
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

Corso di Laurea in Ingegneria delle Telecomunicazioni
Corso di Laurea in Ingegneria dell'Automazione

A.A. 2017-2018

Il protocollo TCP

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

Università degli Studi di Napoli Federico II

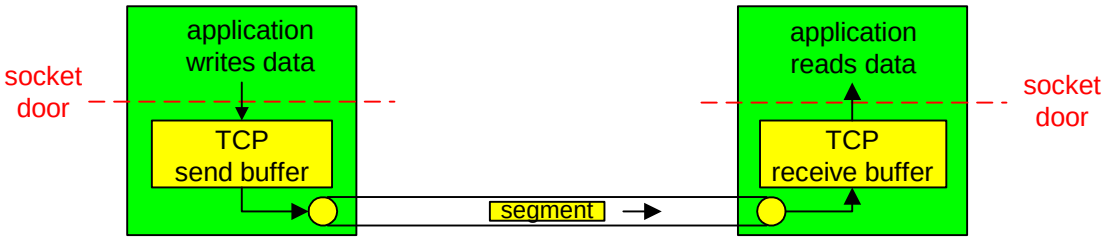
Nota di copyright per le slide COMICS

Nota di Copyright

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

TCP: Transmission Control Protocol

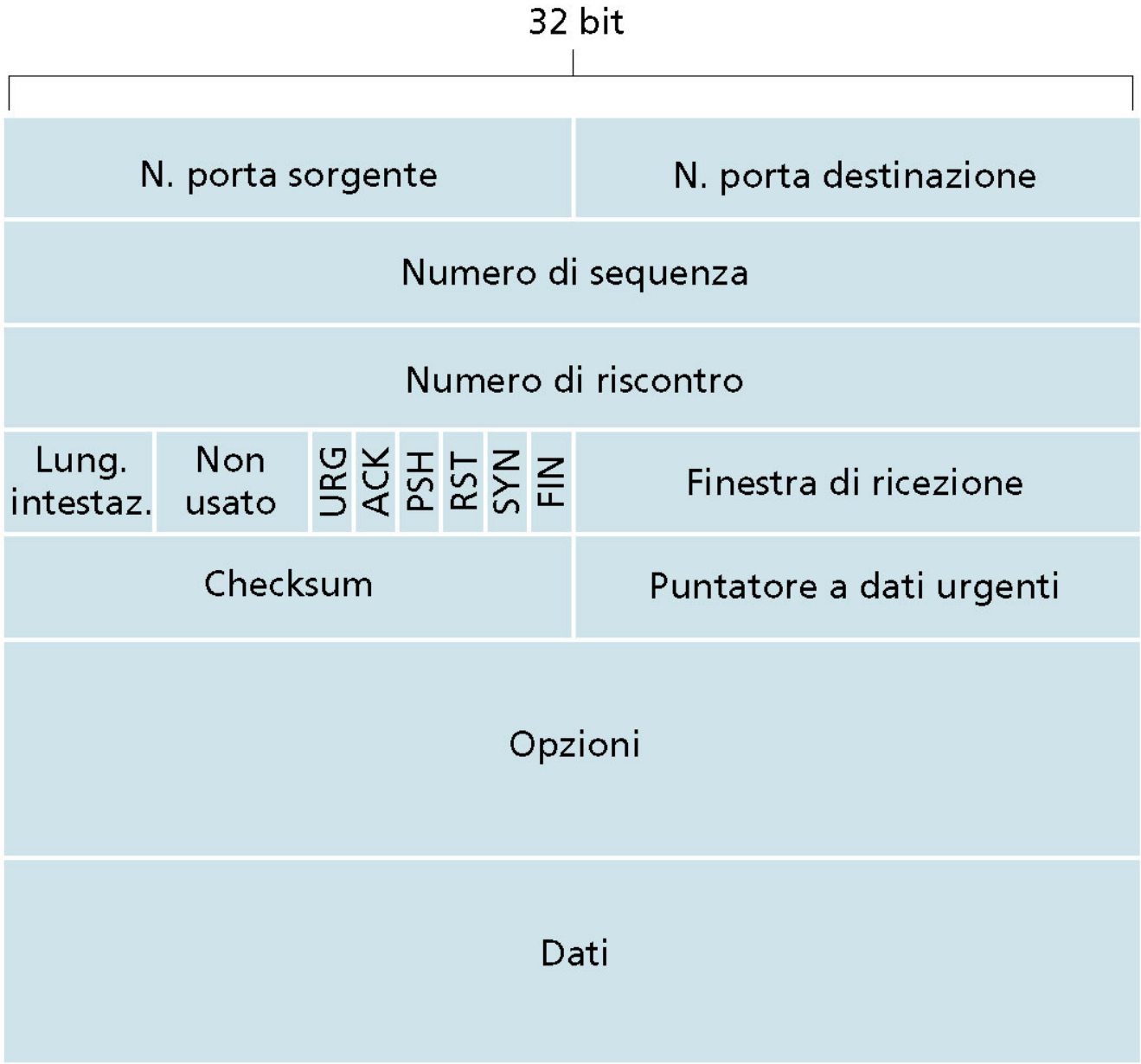
- End-to-end e punto-punto:
  - Una connessione unica tra mittente e ricevente
- Senza errori, sequenza ordinata.
- Pipelined:
  - Controllo di flusso e di congestione impostano la TCP window
- Buffers su mittente e ricevente



- Full duplex data:
  - Flusso di dati bi-direzionale all'interno della stessa connessione
  - *MSS*: Maximum Segment Size
- Connection-oriented:
  - handshaking (scambio di msg di controllo a tre vie) prepara mittente e ricevente prima della comunicazione
- Controllo di flusso:
  - Il mittente non invia più di quanto il ricevente non possa accettare

RFCs: 793, 1122, 1323, 2018, 2581

Struttura del segmento TCP



TCP: PDU

- HLEN:
  - 4 bit, contiene un numero intero che indica la lunghezza dell'intestazione TCP del datagramma in parole da 32 bit. Questa informazione è necessaria perché il campo **opzioni** è di lunghezza variabile
  - Se il campo **opzioni** è vuoto, la la lunghezza è 20 byte (HLEN=5)
- Porta (provenienza/destinazione):
  - Contengono i numeri di porta di protocollo TCP che identificano gli applicativi alle estremità della connessione (mux/demux)

TCP: PDU

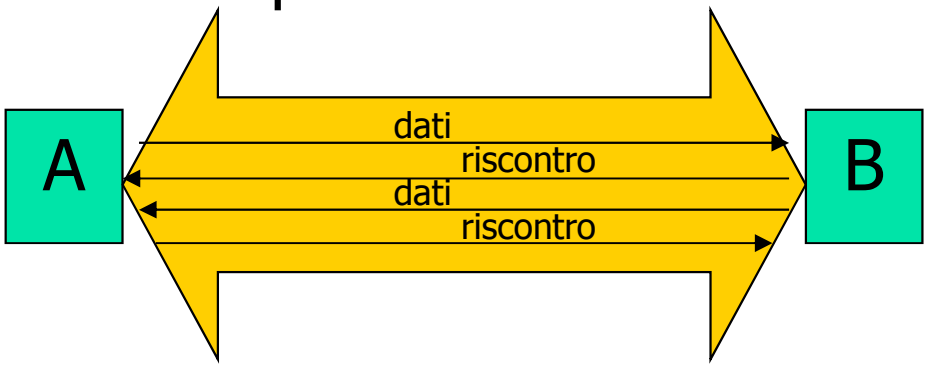
- **Numero di Sequenza:**
  - questo campo identifica, nello stream di byte del trasmettitore, la posizione dei dati nel segmento. Questo valore è riferito alla stream che fluisce nella medesima direzione del segmento, mentre il **Numero di Riscontro** si riferisce allo stream che fluisce nella direzione opposta
- **Numero di riscontro:**
  - Contiene il numero sequenziale del byte successivo a quello correttamente ricevuto dalla destinazione. Tale campo è valido solo nei segmenti di riscontro (cioè in segmenti che hanno ACK=1), e fa riferimento allo stream di dati che fluisce nella direzione opposta a tale segmento
  - Nel calcolo del numero di riscontro, oltre a considerare i bytes contenuti nel payload TCP, bisogna considerare anche la presenza di bytes SYN e FIN inviati, che valgono come un singolo byte

TCP: PDU

- Bit di Codice:
  - Per identificare il tipo di informazione contenuta nel segmento vengono impiegati i 6 bit di codice:
    - ACK: Il campo riscontro è valido
    - RST: Effettua il reset della connessione
    - SYN: Sincronizza i numeri di sequenza
    - FIN: Il trasmettitore ha raggiunto la fine del suo stream di byte
    - PSH: Questo segmento richiede una “spinta” (a destinazione)
    - URG: Il campo puntatore urgente è valido

Segmenti di riscontro e piggybacking

- L'informazione di riscontro viaggia in normali segmenti TCP identificati dal valore 1 del flag ACK
- Per ogni connessione TCP tra due end-point A e B esistono due flussi dati distinti:
  - quello da A a B e
  - quello inverso da B ad A



- Un segmento inviato dall'host B all'host A:
  - può contenere o meno dati relativi al flusso da B ad A e
  - può contenere o meno un informazione di riscontro relativa al flusso dati da A a B
- Se un segmento contiene sia dati che riscontro, si dice che il riscontro viaggia "a cavalluccio" dei dati (**piggy-backing**)

TCP: PDU

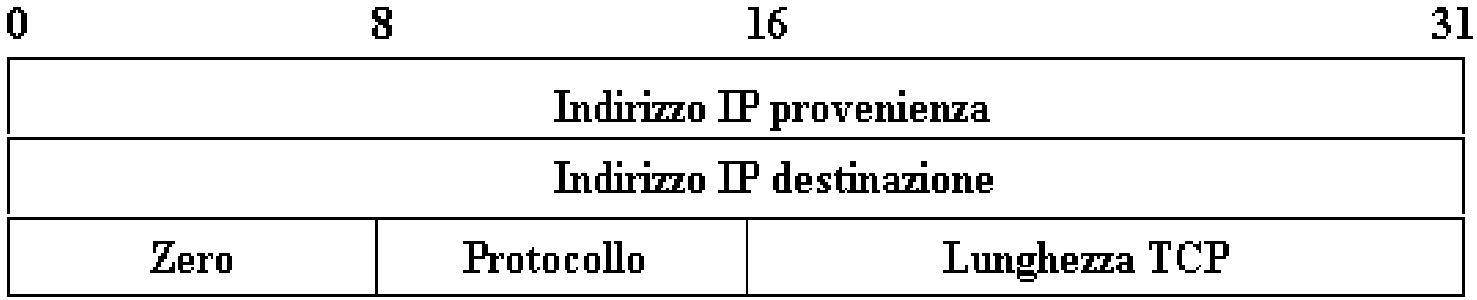
- Finestra:
  - Numero intero senza segno di 16 bit che specifica la dimensione del buffer che il TCP ha a disposizione per immagazzinare dati in arrivo. È utilizzato per la gestione dinamica della dimensione della finestra scorrevole
- Puntatore urgente:
  - Il TCP permette la trasmissione di dati informativi ad alta priorità. Questi devono essere trasmessi il prima possibile, indipendentemente dalla loro posizione nello stream. Questo campo, se valido, conterrà un puntatore alla posizione nello stream, dei dati NON urgenti (ultimo byte dei dati urgenti)

TCP: PDU

- Checksum:
  - Campo di 16 bit contenente un valore intero utilizzato dal TCP della macchina host di destinazione, per verificare l'integrità dei dati e la correttezza dell'intestazione
    - questa informazione è di essenziale importanza perché il protocollo IP non prevede nessun controllo di errore sulla parte dati del frame
    - per il calcolo del valore checksum il TCP aggiunge una pseudointestazione al datagramma, per effettuare così un controllo anche sugli indirizzi IP di destinazione e provenienza

TCP: Pseudoheader per il calcolo checksum

- Pseudoheader creata e posta in testa al datagramma TCP
- Viene inserito in essa un ulteriore byte di zeri per raggiungere un multiplo di 16 bit
- Successivamente viene calcolata la checksum su tutto il messaggio così formato, viene scartata la pseudoheader e passato il datagramma al livello IP
- In fase di ricezione, il livello TCP ricrea la pseudoheader interagendo con l'IP sottostante, calcola la checksum e verifica la correttezza del messaggio ricevuto
- In caso di errore il datagramma viene scartato



TCP: opzioni nell'header

- Maximum TCP payload: durante la fase di connessione, ciascun end-point annuncia la massima dimensione di payload (*MSS – Maximum Segment Size*) che desidera accettare (*MSS announcement*)
- Window Scale: per negoziare un fattore di scala per la finestra; utile per connessioni a larga banda ed elevato ritardo di trasmissione (*long-fat pipes*)
- Selective Repeat: nel caso in cui un segmento corrotto sia stato seguito da segmenti corretti, introduce i NAK (Not AcKnowledge), per permettere al receiver di richiedere la ritrasmissione di quello specifico segmento; è un'alternativa al "go back n", che prevede la ritrasmissione di tutti i segmenti

TCP: Caratteristiche

- Riscontro e ritrasmissione:
  - Consiste nella ritrasmissione di un segmento se non è giunta conferma entro un tempo massimo (time-out)
- Time-Out:
  - Al momento della trasmissione di un segmento, il TCP attiva un timer

TCP: Numeri di sequenza e numeri di riscontro

- TCP vede i dati come un flusso di byte non strutturati, ma ordinati.
- I numeri di sequenza riflettono questa visione: il numero di sequenza per un segmento è quindi il numero nel flusso di byte del primo byte nel segmento.
  - Flusso lungo 500.000 byte, MSS uguale a 1000 byte, primo byte numerato con 0
  - TCP costruisce 500 segmenti, con numeri di sequenza 0, 1000, 2000 …
- Per i numeri di riscontro, vista la natura full-duplex della connessione TCP, si ha che ad es.:
  - A invia e contemporaneamente riceve da B
  - I segmenti B -> A , contengono un numero di sequenza relativo ai dati B -> A
  - **Il numero di riscontro** che A scrive nei propri segmenti è il numero di sequenza del byte successivo che A attende da B (e lo può mandare anche inviando dati a B).
- TCP invia riscontri cumulativi

Numeri di sequenza e ACK di TCP

Numeri di sequenza:

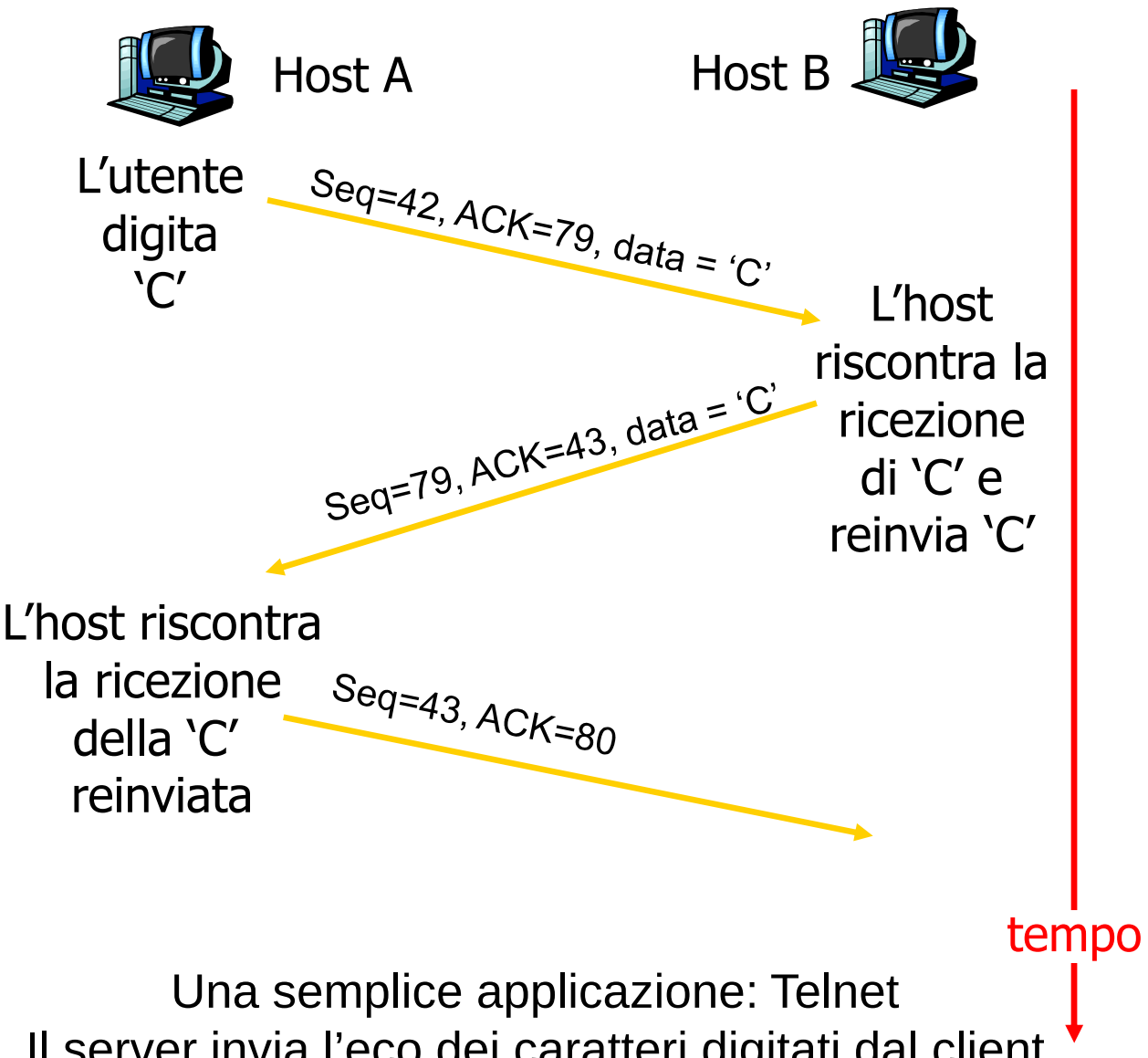
- “numero” del primo byte del segmento nel flusso di byte

ACK:

- numero di sequenza del prossimo byte atteso dall’altro lato
- ACK cumulativo

D: come gestisce il destinatario i segmenti fuori sequenza?

- R: la specifica TCP non lo dice – dipende dall’implementatore



Una semplice applicazione: Telnet
Il server invia l’eco dei caratteri digitati dal client

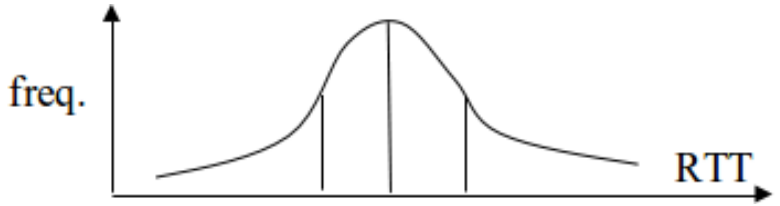
TCP: Caratteristiche del riscontro cumulativo

- Nel datagramma di riscontro la destinazione comunica quale byte dello stream si aspetta di ricevere successivamente:
  - I riscontri specificano sempre il numero sequenziale del primo byte non ancora ricevuto
    - » Esempio: in uno stream di 1000 byte segmentato in blocchi di 100 byte, partendo da 0, il primo riscontro conterrà il numero sequenziale 100
  - Con questo metodo di riscontro *cumulativo* si ha il vantaggio che la perdita di un riscontro non blocca la trasmissione se confermato dal riscontro successivo

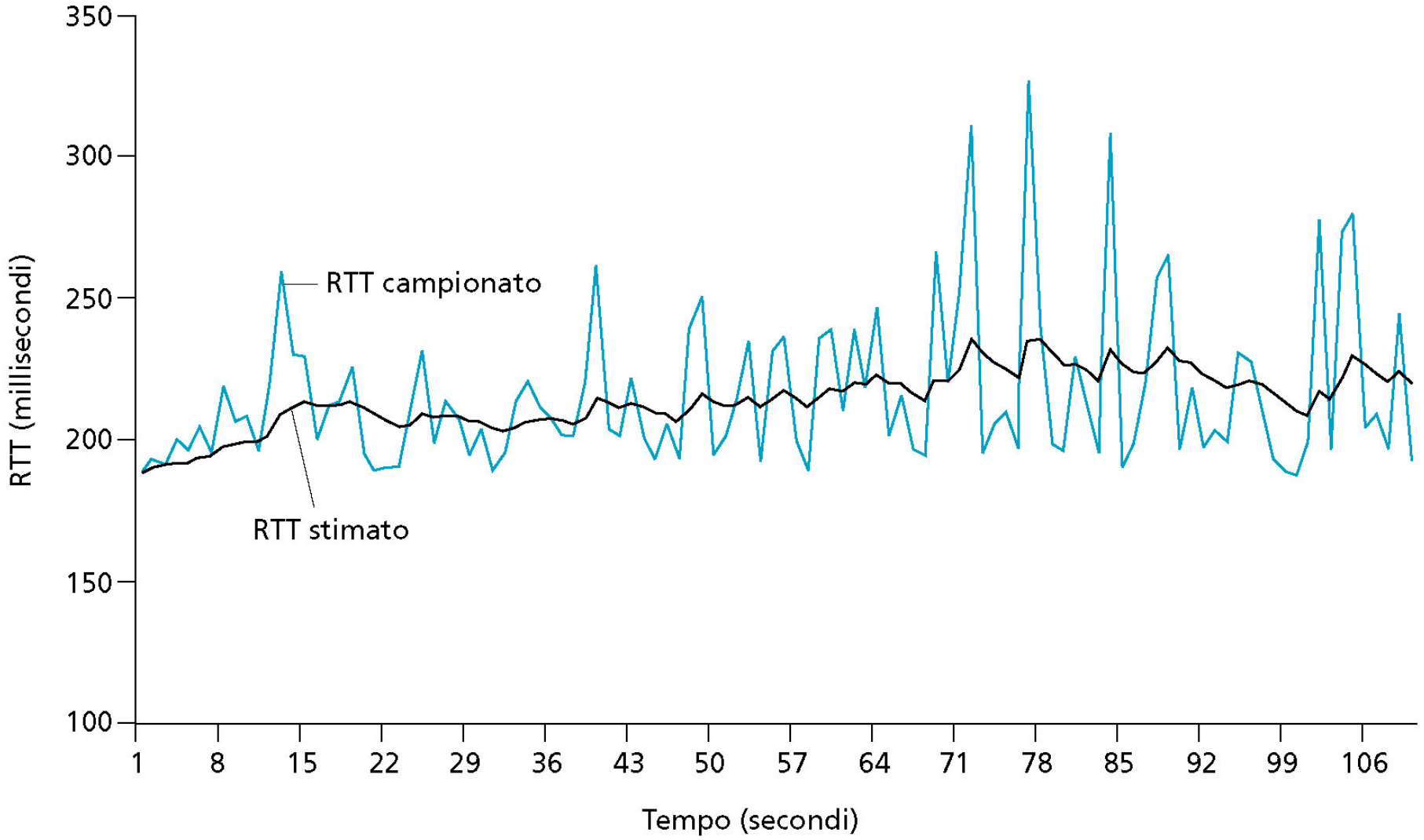
Round Trip Time e Timeout

Domanda: A quale valore impostare il timeout?

- Di sicuro maggiore di RTT (Round Trip Time)
- N.B: RTT varia nel tempo
  - Se timeout è scelto troppo breve:
    - timeout prematuro
      - ritrasmissioni ridondanti
      - scarsa efficienza
  - Se timeout è scelto troppo lungo:
    - scarsa efficienza nella gestione delle ritrasmissioni
- Problema: stimare RTT corrente
  - A causa della variabilità, occorre considerare anche la varianza



RTT campionato vs RTT stimato



SampleRTT: tempo misurato dalla trasmissione del segmento fino alla ricezione di ACK, ignorando le ritrasmissioni (se ne sceglie uno ad ogni istante di tempo).

Stima di RTT e valore del timeout

- **Stima di RTT**: media esponenziale pesata dei campioni (*EWMA: Exponential Weighted Moving Average*)
  - L'influenza di un singolo campione sul valore della stima decresce in maniera esponenziale, e si da più importanza a campioni recenti.
  - Valore tipico per α: 0.125

$$\mathtt{EstimatedRTT} = (1-\alpha) * \mathtt{EstimatedRTT} + \alpha * \mathtt{SampleRTT}$$

- **Stima della deviazione di RTT**:

$$\mathtt{DevRTT} = (1-\beta) * \mathtt{DevRTT} + \beta * |\mathtt{SampleRTT}-\mathtt{EstimatedRTT}|$$

  - Valore raccomandato per β: 0,25

- **Valore del timeout:** EstimatedRTT più un "margine di sicurezza" proporzionale alla variabilità della stima effettuata
  - variazione significativa di **EstimatedRTT** → margine più ampio:

$$\mathtt{Timeout} = \mathtt{EstimatedRTT} + 4*\mathtt{DevRTT}$$

TCP Connection Management

Mittente e Ricevente concordano l'apertura della connessione prima di inviare i dati

---

Impostare le variabili del TCP:

- Numeri di sequenza
- Allocare i buffer, impostare un valore iniziale della **`RcvWindow`**

Three way handshake:

Passo 1: client invia segmento di controllo TCP SYN al server

- Specifica il 1° seq #

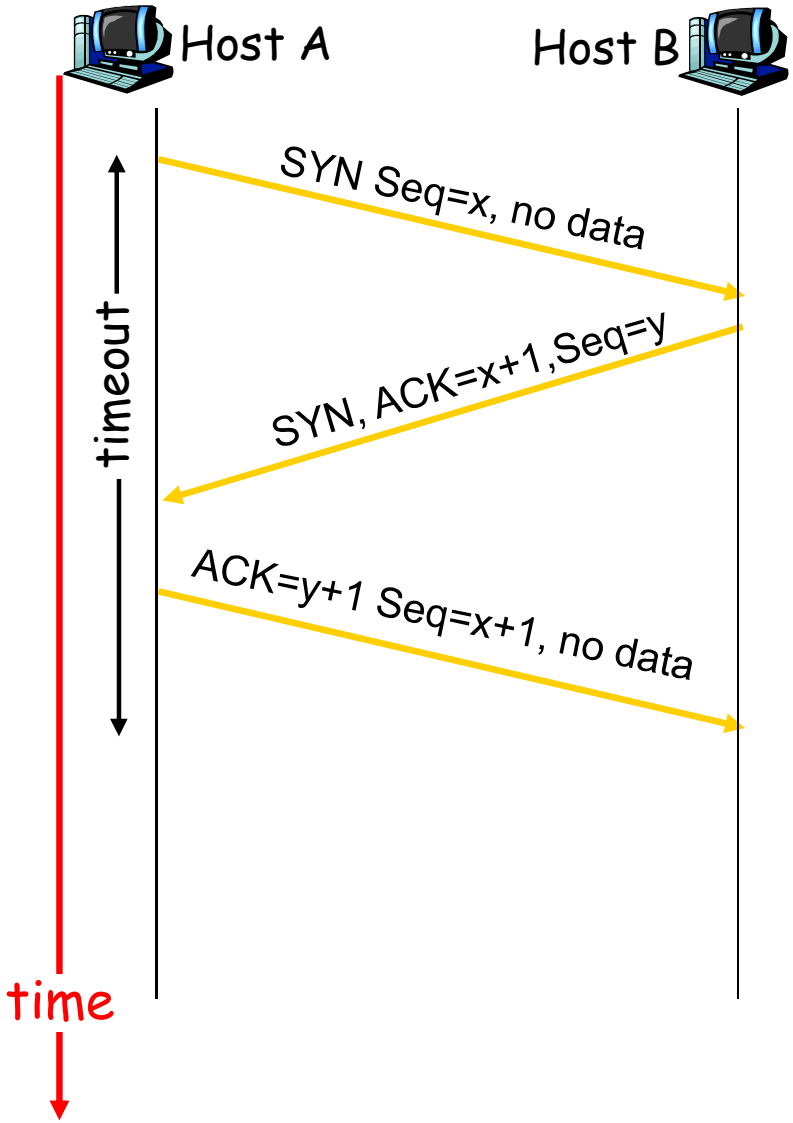
Passo 2: server riceve SYN, risponde con segmento di controllo SYN/ACK

- ACK del SYN ricevuto
- Alloca buffer
- Specifica il 1°seq. # per la connessione server→client

Passo 3: client riceve SYN/ACK, invia ACK al server

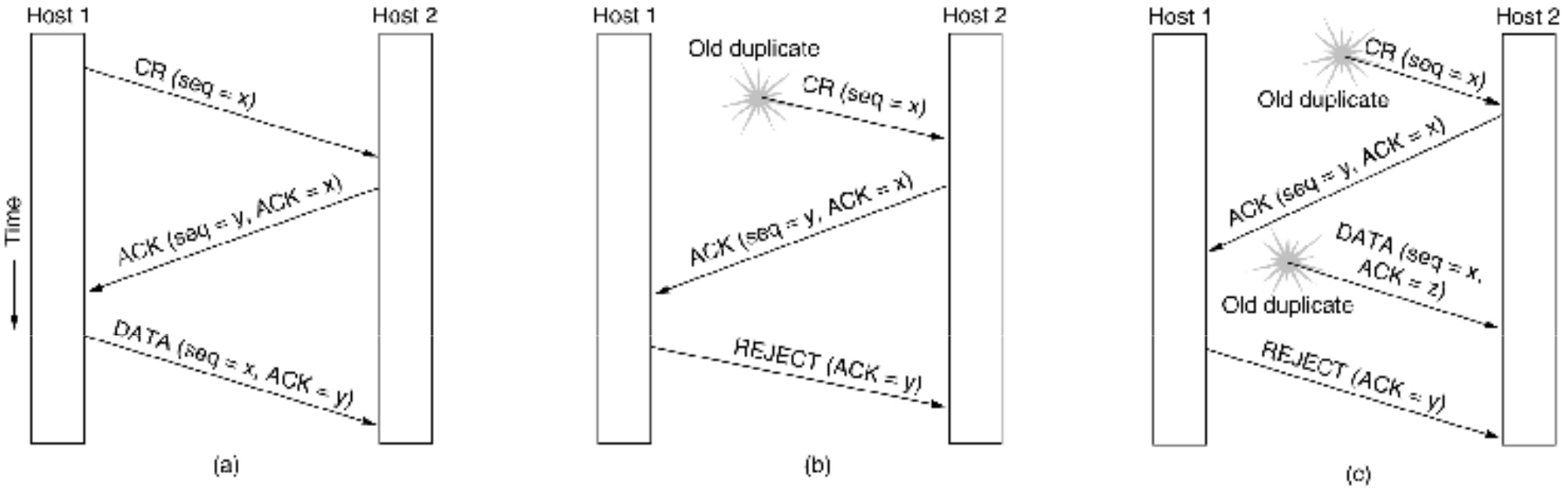
- Connessione instaurata

Three way handshake



Three way handshake: scenari patologici

- (a) – caso normale di three-way handshake
- (b) – arrivo di un CR duplicato per connessione già stabilita
- (c) – arrivo di CR e segmento dati da connessione già stabilita



Il three-way-handshake consente di gestire correttamente anche questi casi "patologici"

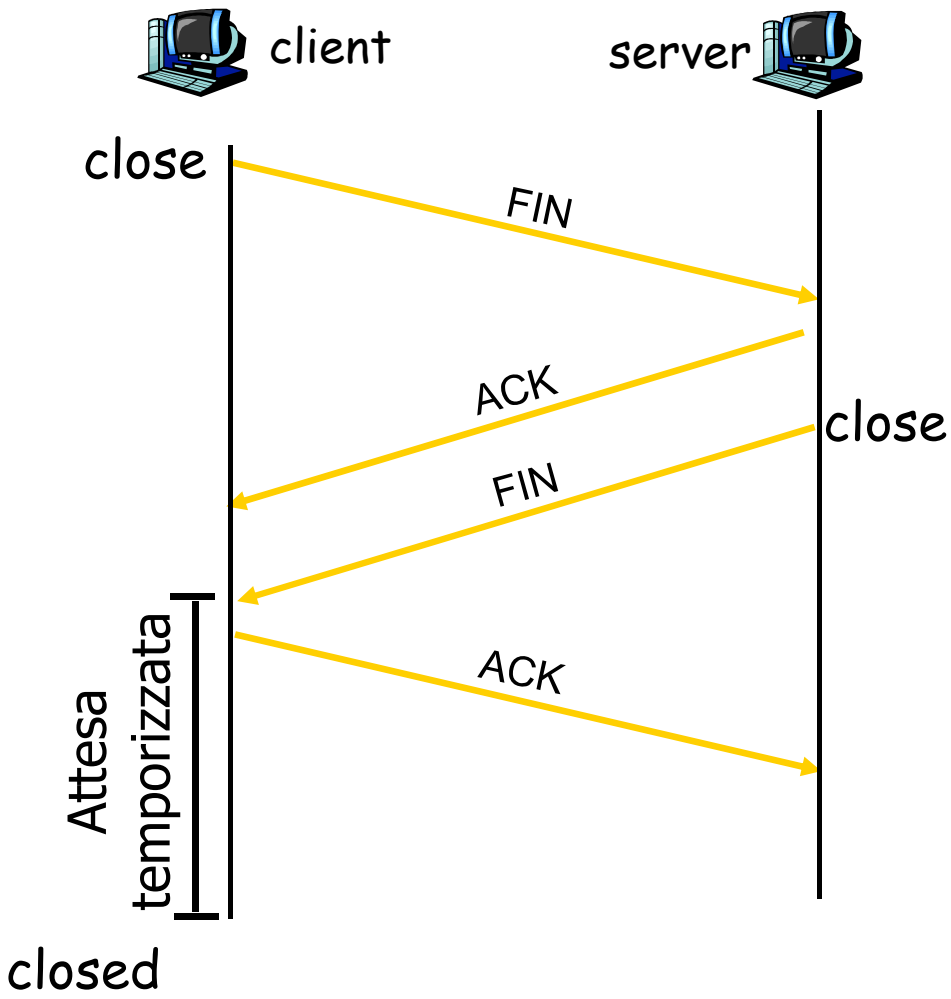
CR sta genericamente per Connection Request

Chiusura della connessione

Una delle due parti (ad esempio il client) decide di chiudere la connessione ...

Passo 1: client invia un segmento di controllo TCP con FIN alto al server

Passo 2: server riceve FIN, risponde con ACK. Quindi chiude la connessione inviando FIN al client



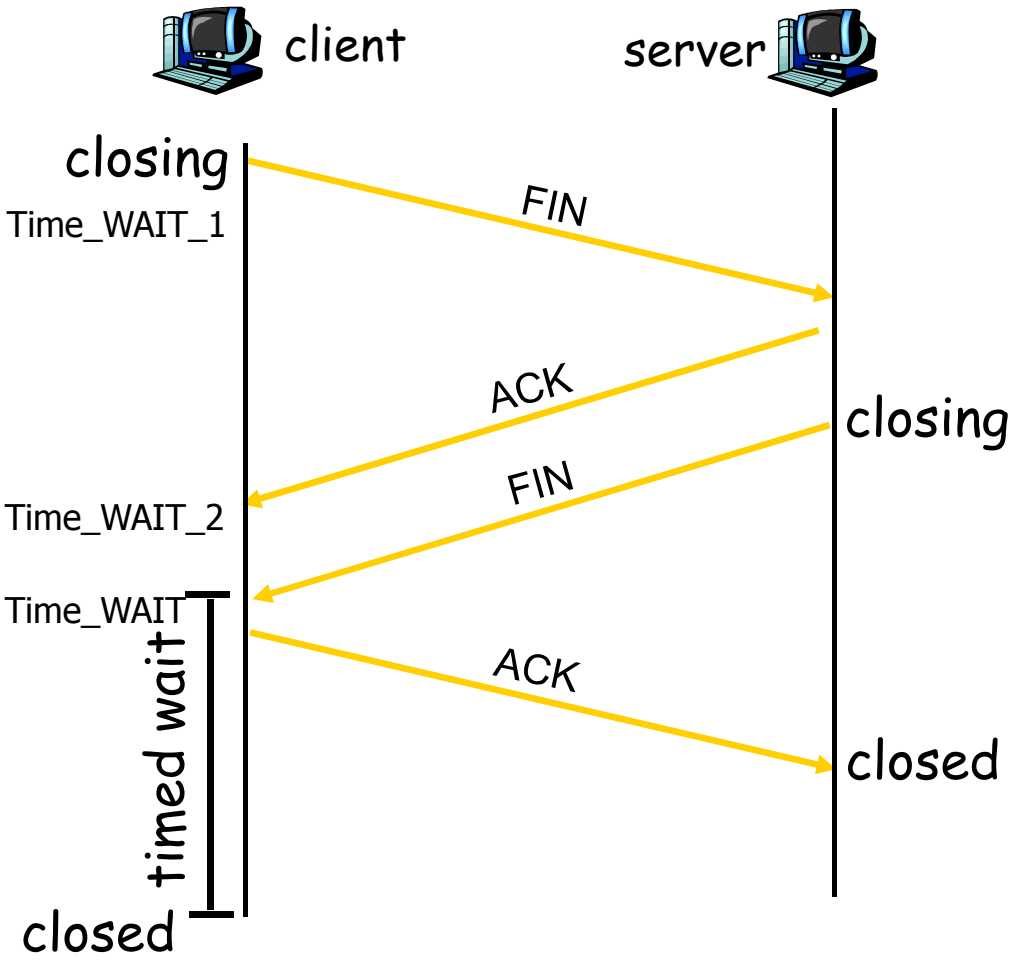
Chiusura della connessione

Passo 3: client riceve FIN, risponde con un ACK

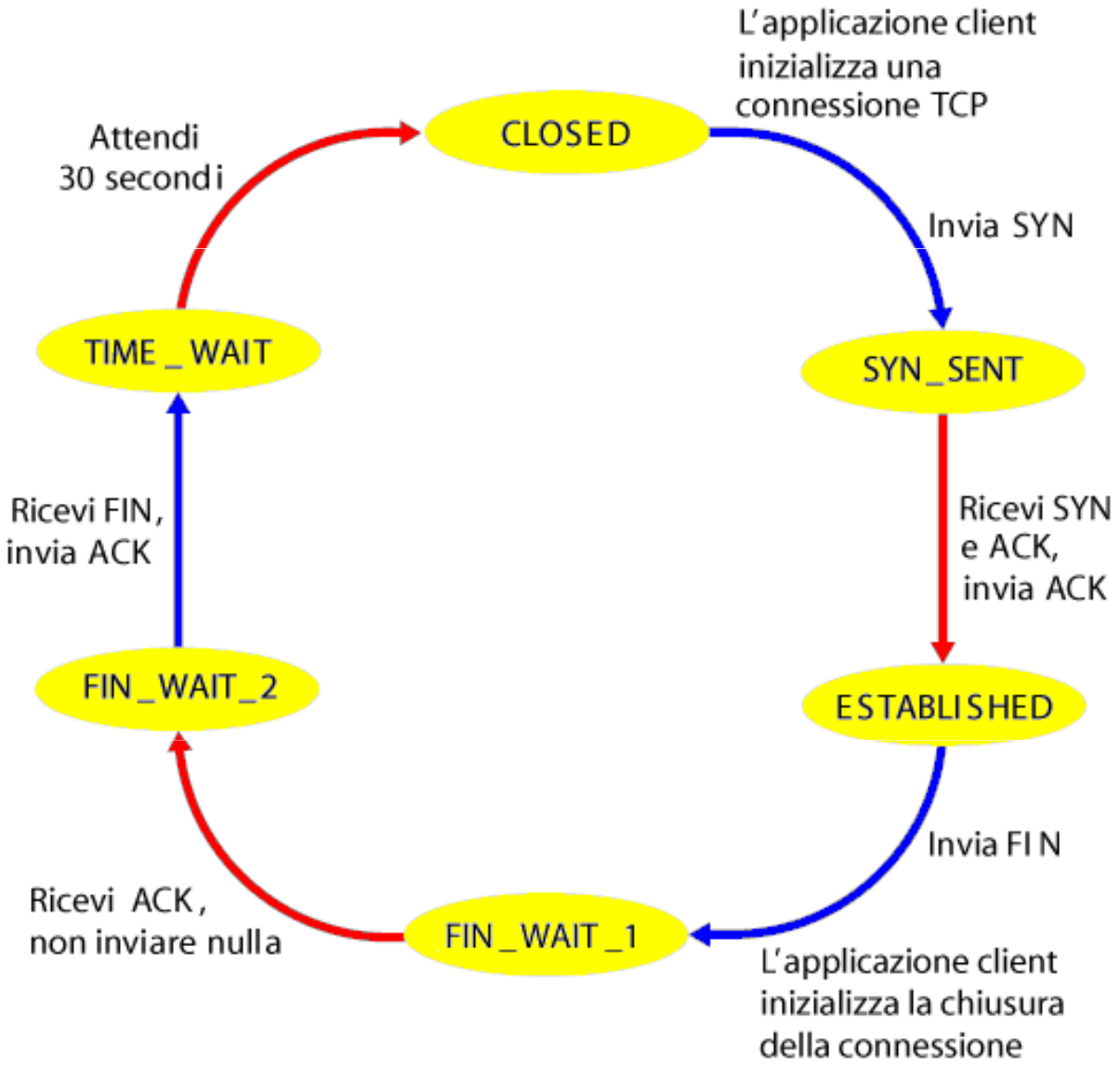
- Attende in uno stato TIMED_WAIT (nel caso in cui l'ultimo ACK vada perso, e riceva un ulteriore FIN dal server)

Passo 4: server, riceve ACK. Chiude la connessione.

Nota: se non ha altri dati da inviare al client, il server può inviare ACK e FIN contemporaneamente all'interno dello stesso segmento



Sequenza tipica degli stati nel client



Sequenza tipica degli stati nel server

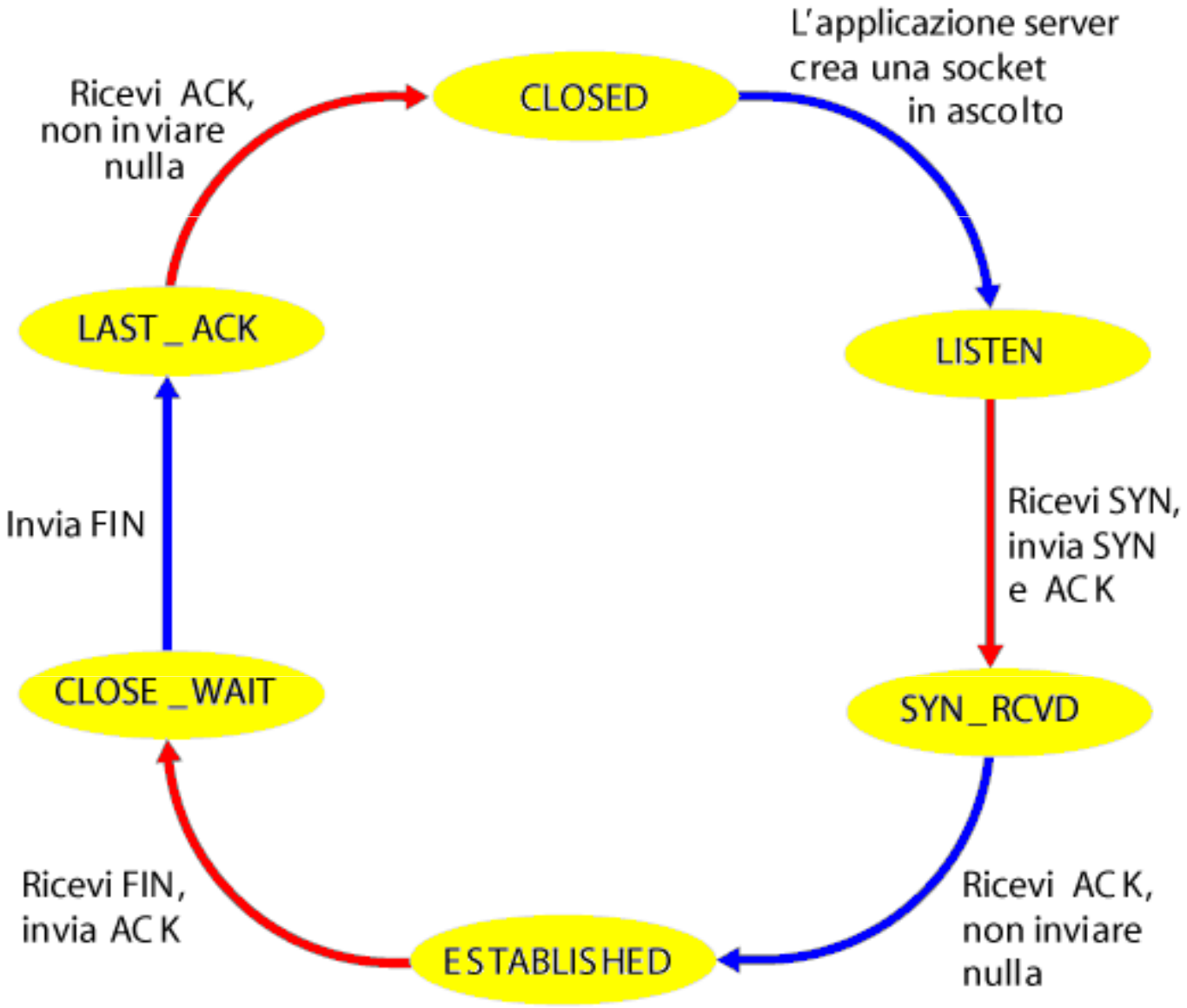
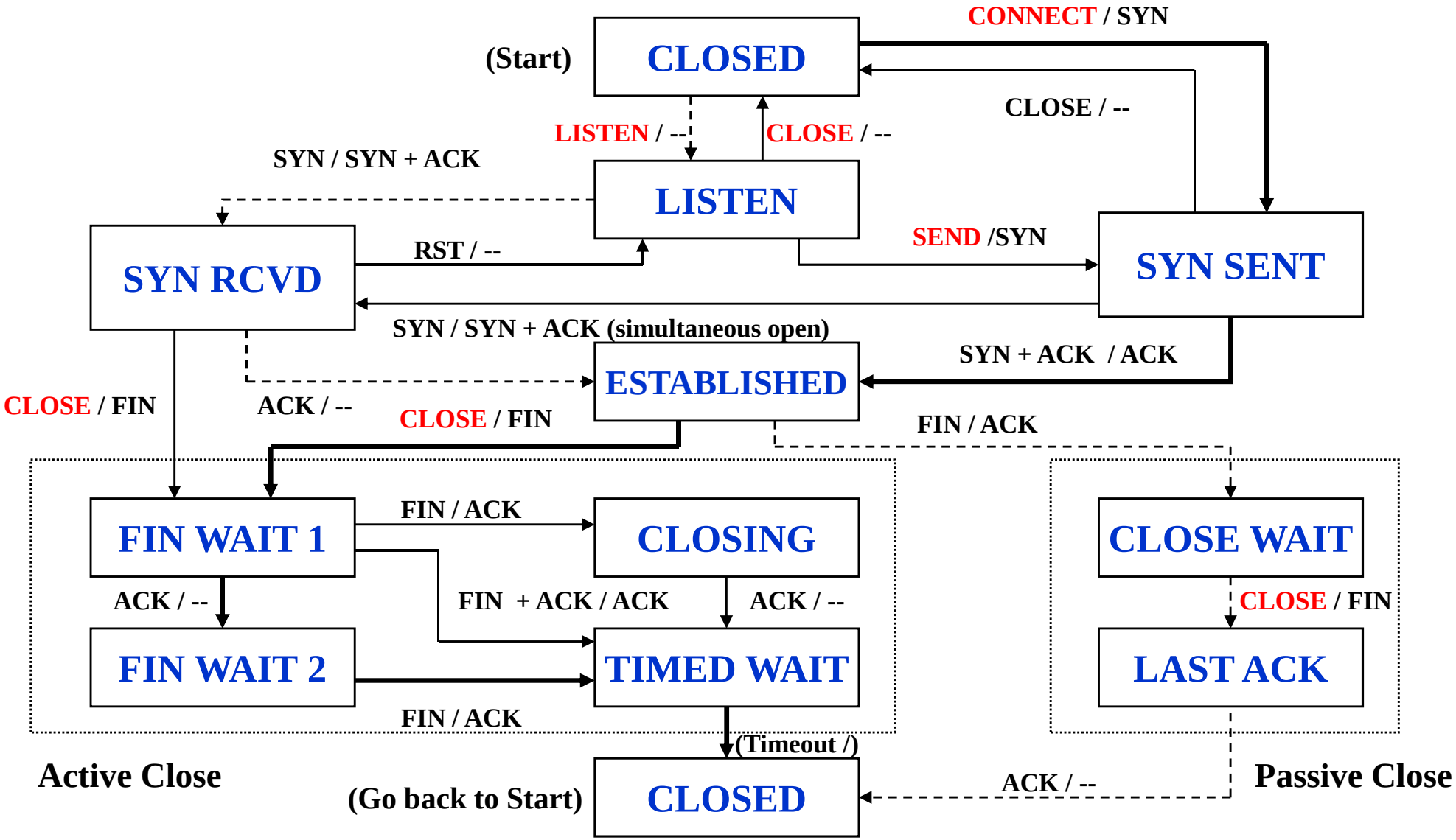
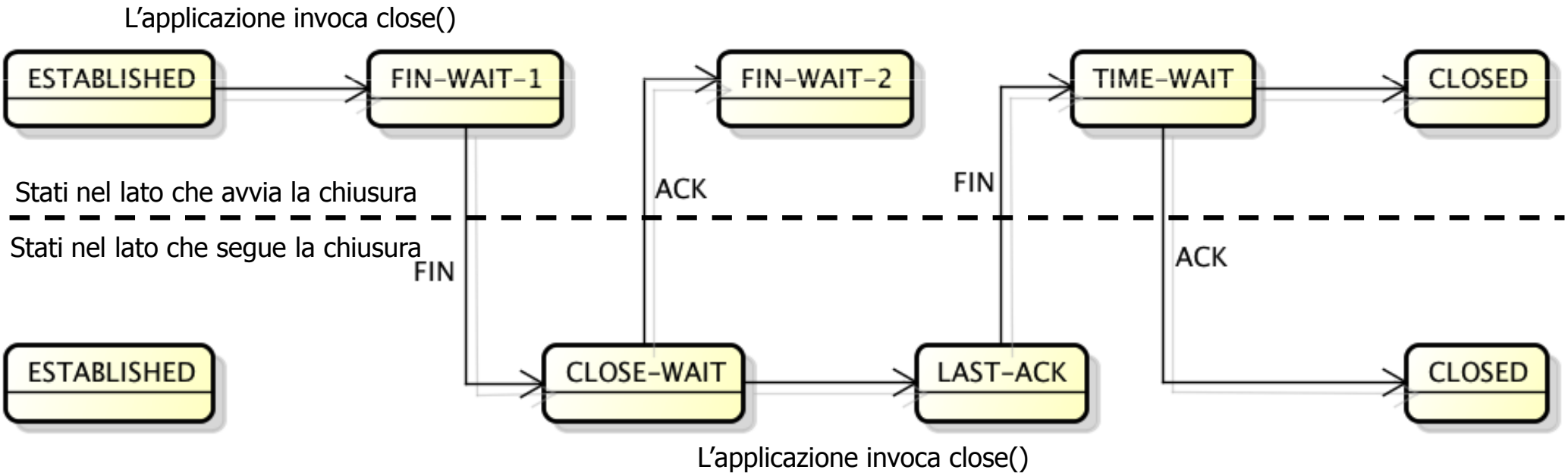


Diagramma degli stati del TCP



Sequenze di stati in fase di chiusura



TCP: trasferimento dati affidabile

- TCP crea un servizio di trasferimento dati affidabile sul servizio inaffidabile di IP
- Pipeline dei segmenti
- ACK cumulativi
- TCP usa un solo timer di ritrasmissione

- Le ritrasmissioni sono avviate da:
  - eventi di timeout
  - ACK duplicati
- Inizialmente consideriamo un mittente TCP semplificato:
  - ignoriamo gli ACK duplicati
  - ignoriamo il controllo di flusso e il controllo di congestione

TCP: eventi del mittente

Dati ricevuti dall'applicazione:

- Crea un segmento con il numero di sequenza
- Il numero di sequenza è il numero del primo byte del segmento nel flusso di byte
- Avvia il timer, se non è già in funzione (pensate al timer come se fosse associato al più vecchio segmento non riscontrato)
- Intervallo di scadenza: `TimeOutInterval`

Timeout:

- Ritrasmette il segmento che ha causato il timeout
- Riavvia il timer

ACK ricevuti:

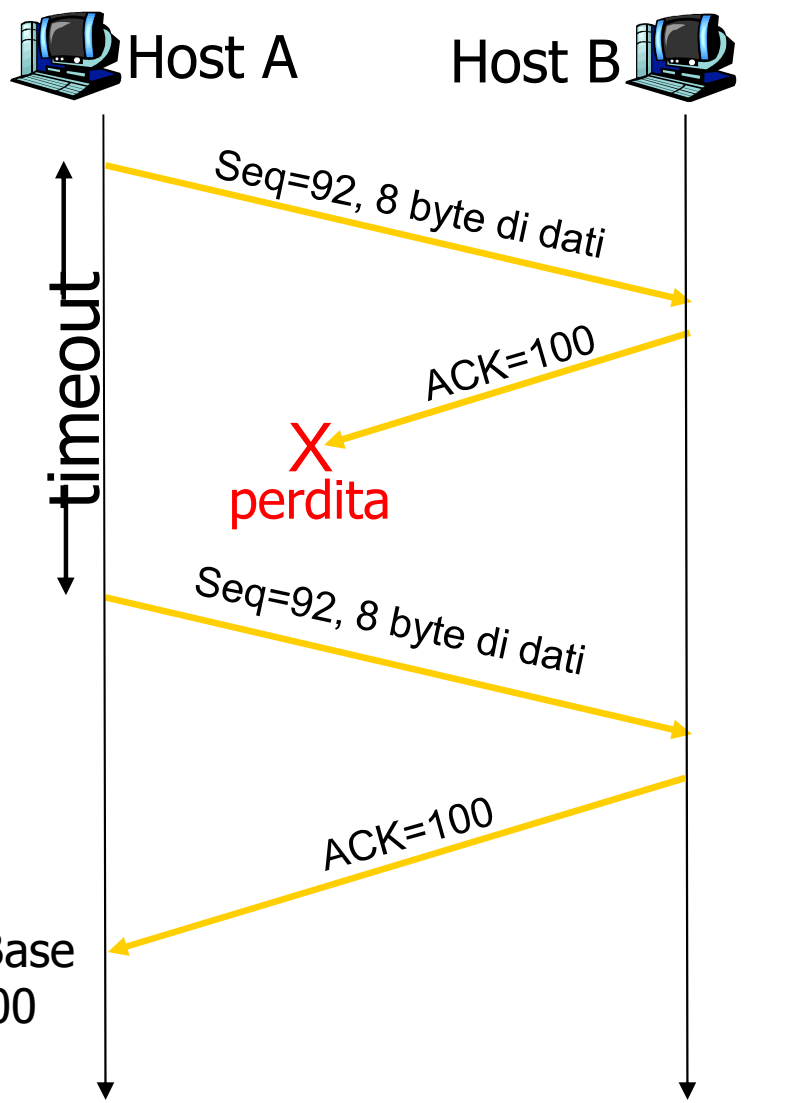
- Se riscontra segmenti precedentemente non riscontrati
  - aggiorna ciò che è stato completamente riscontrato
  - avvia il timer se ci sono altri segmenti da completare

Un sender TCP semplificato

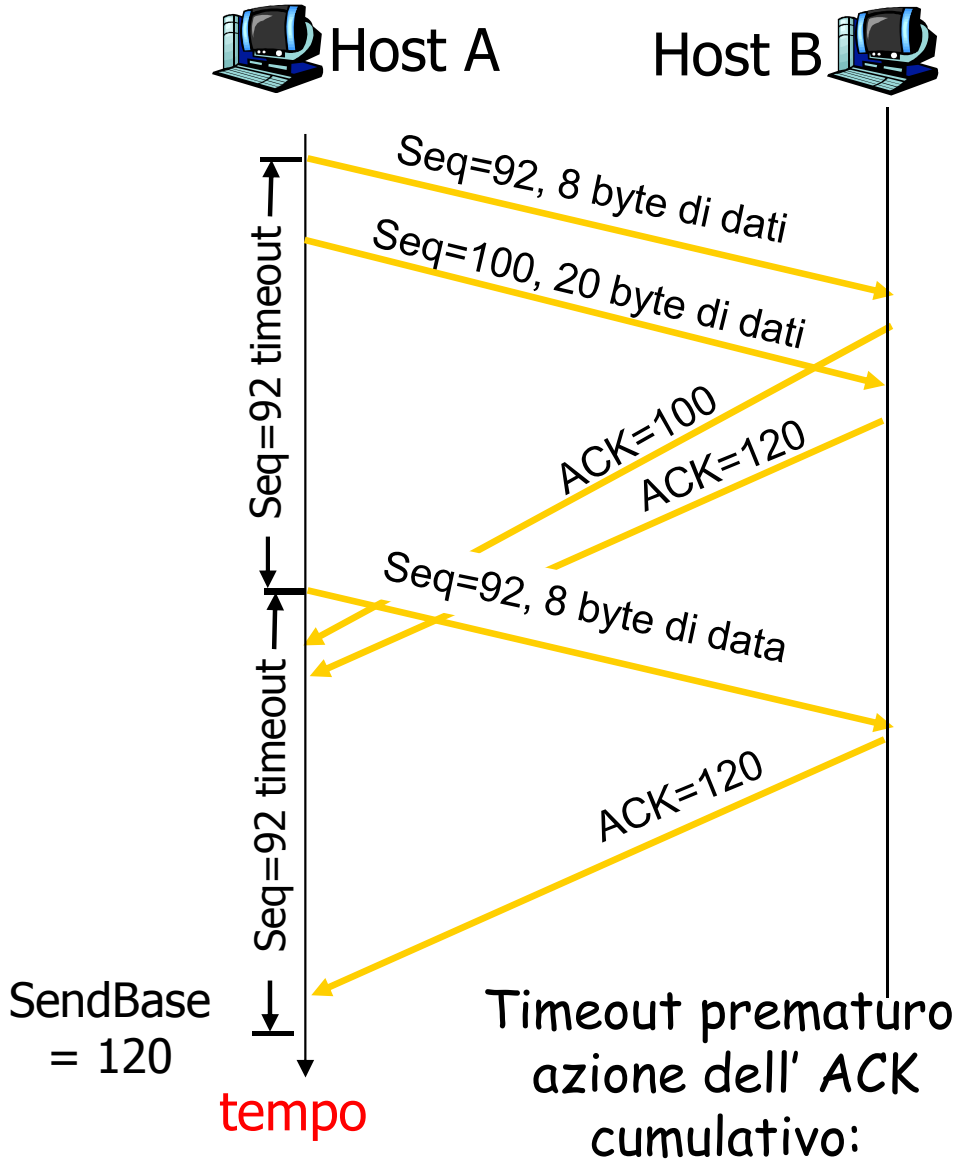
/* Si è assunto che il sender non sia limitato dal controllo di flusso o di congestione del TCP, che i dati da sopra siano di dimensioni inferiori all'MSS e che il trasferimento dei dati avvenga in una sola direzione.*/

```
NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop (forever) {
   switch(event)

      event: data received from application above
         create TCP segment with sequence number NextSeqNum
         if (timer currently not running)
            start timer
         pass segment to IP
         NextSeqNum=NextSeqNum+length(data)
         break;

      event: timer timeout
         retransmit not-yet-acknowledged segment with
            smallest sequence number
         start timer
         break;

      event: ACK received, with ACK field value of y
         if (y > SendBase) {
            SendBase=y
            if (there are currently any not-yet-acknowledged
               segments)
               start timer
            }
            break;

   } /* end of loop forever */
```

Alcuni scenari di rilievo - 1

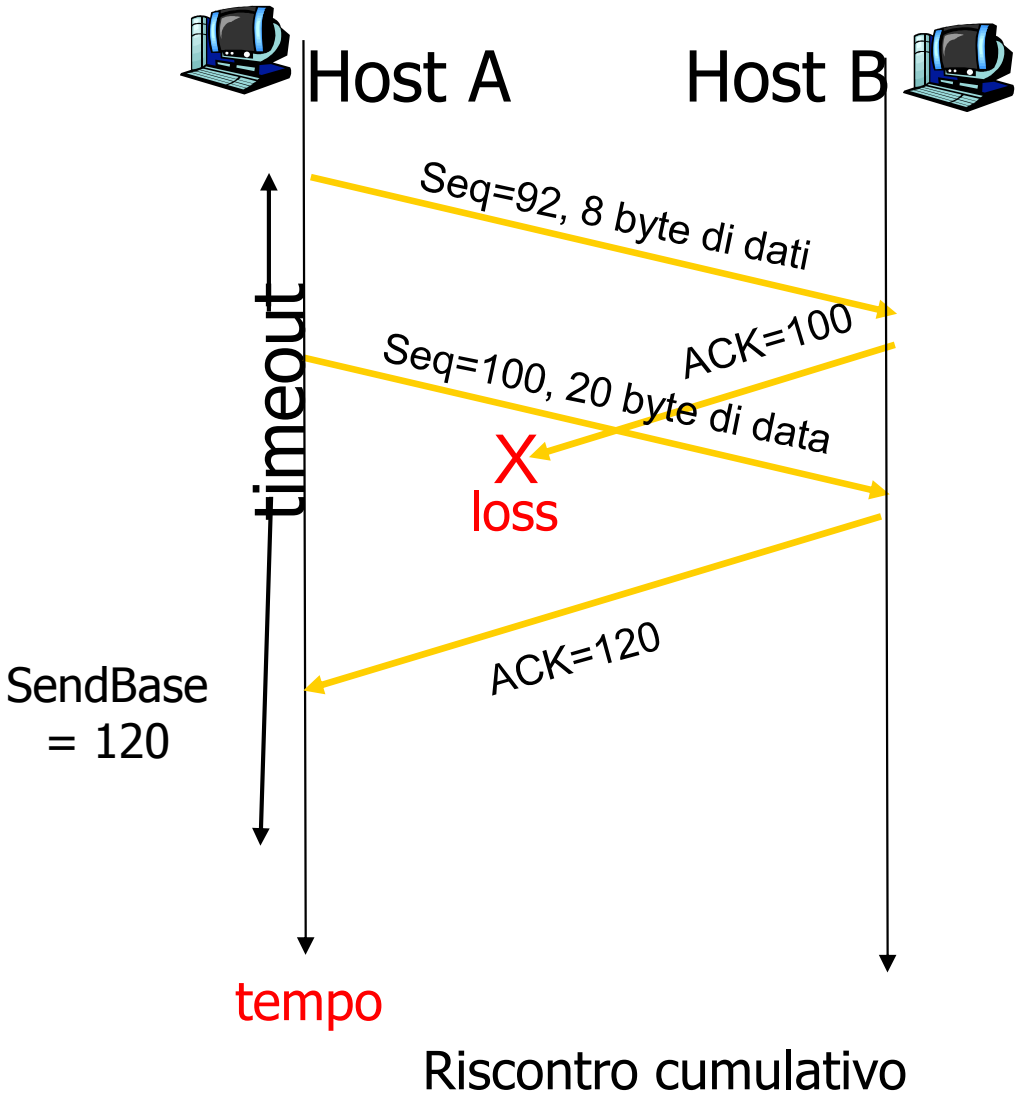


Ritrasmissione dovuta ad un riscontro perso



Timeout prematuro, azione dell' ACK cumulativo: il segmento 100 non è ritrasmesso

Alcuni scenari di rilievo - 2



Riscontro cumulativo

Il riscontro cumulativo evita la ritrasmissione del primo segmento…

Modifiche tipiche del TCP - 1

- Raddoppio dell'intervallo di timeout:
  - Allo scadere di un timeout:
    - si imposta il prossimo intervallo al doppio del valore precedente (invece di usare la stima di RTT)
      - Crescita esponenziale degli intervalli dopo ogni ritrasmissione
  - Quando il timer viene riavviato (ricezione di un ACK o di nuovi dati dall'applicazione):
    - l'intervallo di timeout viene nuovamente configurato in funzione dei valori più recenti di `EstimatedRTT` e `DevRTT`

- Fornisce una forma limitata di controllo della congestione:
  - Il mittente, nel caso supponga una situazione di congestione (perdita di un segmento), ritrasmette ad intervalli sempre più lunghi.

Modifiche tipiche del TCP - 2

- Ritrasmissione veloce:
  - ACK duplicati:
    - Consentono di rilevare la perdita di un pacchetto prima del timeout
      - un receiver che rileva un “buco” nei segmenti ricevuti (ricezione di un segmento con numero di sequenza maggiore di quello atteso):
        - » invia un nuovo riscontro per l’ultimo byte di dati che ha ricevuto correttamente
      - poiché il mittente spesso manda molti segmenti contigui, se uno di tali segmenti si perde, ci saranno molti ACK duplicati contigui:
        - » un sender che riceve tre ACK duplicati per gli stessi dati assume che il segmento successivo a quello riscontrato tre volte è andato perso ed effettua, quindi, una ritrasmissione prima della scadenza del timeout

TCP: generazione di ACK [RFC 1122, RFC 2581]

Evento nel destinatario	Azione del ricevente TCP
Arrivo ordinato di un segmento con numero di sequenza atteso. Tutti i dati fino al numero di sequenza atteso sono già stati riscontrati.	ACK ritardato. Attende fino a 500 ms l'arrivo del prossimo segmento. Se il segmento non arriva, invia un ACK.
Arrivo ordinato di un segmento con numero di sequenza atteso. Un altro segmento è in attesa di trasmissione dell'ACK.	Invia immediatamente un singolo ACK cumulativo, riscontrando entrambi i segmenti ordinati.
Arrivo non ordinato di un segmento con numero di sequenza superiore a quello atteso. Viene rilevato un buco.	Invia immediatamente un ACK (duplicato), indicando il numero di sequenza del prossimo byte atteso.
Arrivo di un segmento che colma parzialmente o completamente il buco.	Invia immediatamente un ACK, ammesso che il segmento cominci all'estremità inferiore del buco.