Reti di Calcolatori

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

Corso di Laurea in Ingegneria Informatica

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

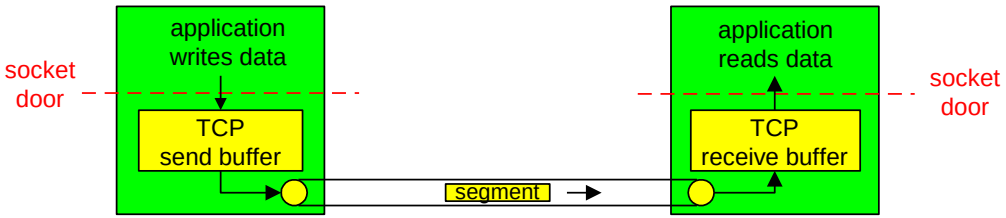
Nota di Copyright

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

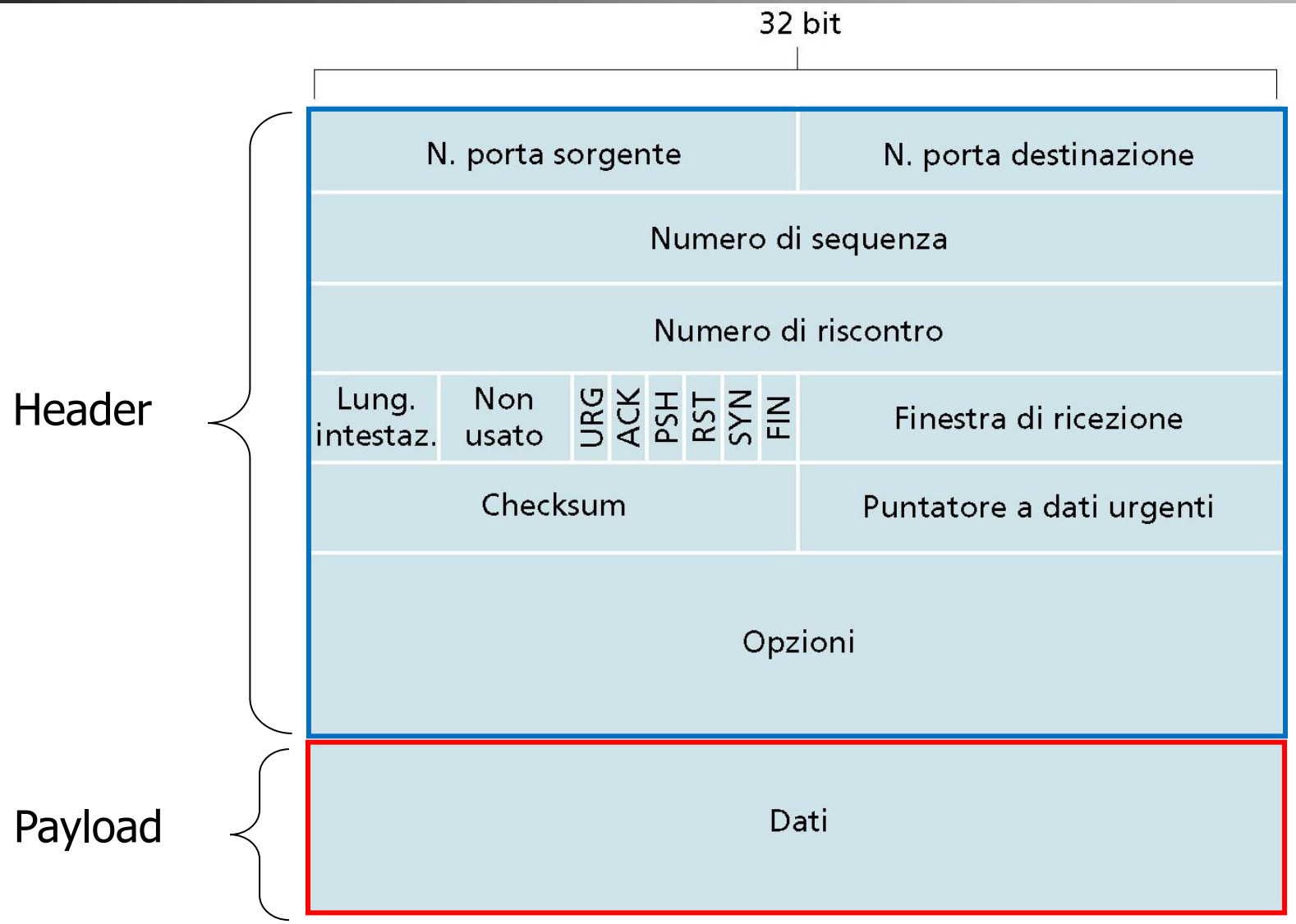
Università degli Studi di Napoli Federico II

TCP: Transmission Control Protocol

- TCP è un protocollo di trasporto connection-oriented che consente la trasmissione bidirezionale affidabile di un flusso di byte tra due endpoint
- TCP garantisce assenza di perdite e rispetto nella sequenza dei dati
- Definito inizialmente in RFC 793 del 1981
- Implementa meccanismi di controllo di flusso e controllo di congestione
- Affidabilità realizzata mediante controllo di sequenza e meccanismi di ritrasmissione
  - Dati sono mantenuti in appositi buffer ad entrambe le estremità dei flussi dei dati
- Le applicazioni interagiscono con TCP tramite API che si avvalgono di punti di accesso al servizio detti *socket*



Struttura di un segmento TCP



TCP: struttura dell'header (1)

- **Numeri di porta sorgente e destinazione**
  - Contengono i numeri di porta di protocollo TCP che identificano gli end-point della connessione (mux/demux)
- **Lunghezza header 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
  - HLEN assume valori compresi tra 5 (minimo) e 15 (massimo)
  - HLEN = 5 → lunghezza header TCP: 20 byte → no **opzioni**
  - HLEN max è 15 → lunghezza massima header TCP: 60 byte

TCP: struttura dell'header (2)

- **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: struttura dell'header (3)

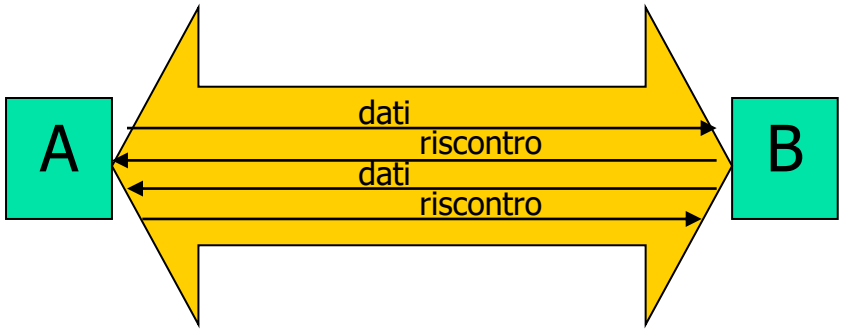
- **Flag**
  - Per identificare il tipo di informazione contenuta nel segmento vengono impiegati i 6 bit di codice:
    - ACK: Il campo riscontro è valido
    - RST: Effettua il reset della connessione
    - SYN: Sincronizza i numeri di sequenza
    - FIN: Il trasmettitore ha raggiunto la fine del suo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

TCP: struttura dell'header (4)

- **Finestra di ricezione**
  - Numero intero senza segno di 16 bit che specifica la dimensione del buffer che il TCP ha a disposizione per immagazzinare dati in arrivo
- **Puntatore ai dati urgenti**
  - Il TCP permette la trasmissione di dati informativi ad alta priorità che devono essere trasmessi il prima possibile
  - Questo campo, se valido (flag URG ad 1), conterrà un puntatore alla posizione nello stream, dei dati NON urgenti (ultimo byte dei dati urgenti)
- **Checksum**
  - Campo di 16 bit contenente un valore intero utilizzato dal ricevitore per verificare l'integrità del segmento ricevuto
    - IP non prevede nessun controllo di errore sulla parte dati del frame
    - In caso di rilevazione di errore il segmento viene scartato

Segmenti di riscontro e piggy-backing

- 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**)

Pseudo-header per il calcolo checksum

- Ai soli fini del calcolo della checksum, TCP aggiunge fittiziamente al segmento effettivo uno pseudo-header costituito come in figura

0	8	16	31
Indirizzo IP provenienza			
Indirizzo IP destinazione			
Zero	Protocollo	Lunghezza TCP	

- In ricezione, TCP ricrea la pseudo-header interagendo con lo strato IP sottostante, calcola la checksum e verifica la correttezza del messaggio ricevuto

Checksum TCP

Mittente:

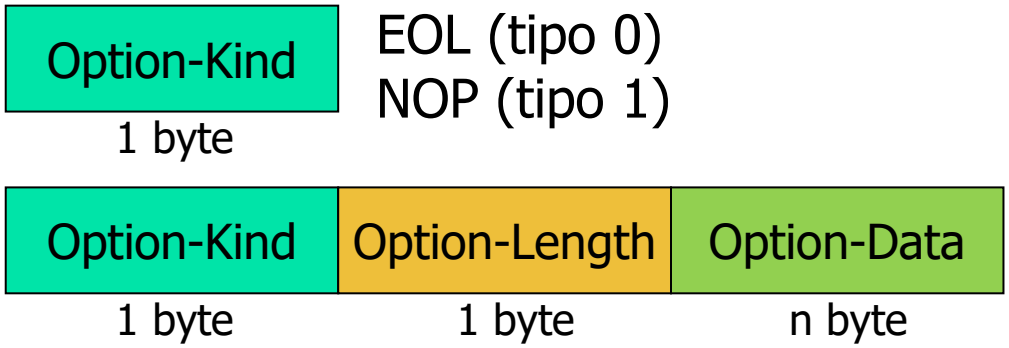
- Tratta il contenuto del segmento (e dello pseudo-header) come una sequenza di interi espressi su 16 bit
- Calcola la ***checksum*** come
  complemento ad 1 della somma in complemento ad 1
  dei «pezzi» da 16 bit che costituiscono il segmento TCP e lo pseudo-header
- Pone il valore di checksum nell'apposito campo del segmento trasmesso

Ricevente:

- Calcola la somma in complemento ad 1 dei campi del segmento ricevuto <u>compresa la checksum</u>
- Se il risultato NON è composto da tutti 1, questo è indicativo di un errore
- Se il risultato è composto da tutti 1, il controllo di checksum è superato

TCP: opzioni nell'header (1)

- Le opzioni sono di lunghezza variabile
- La lunghezza totale dell'header TCP deve essere multipla di 4 byte
- Se necessario si aggiungono zeri di padding dopo l'ultima opzione
- Sono previsti 7 diversi tipi di opzioni: MSS, Window scale, Timestamp, Selective ACK permitted, Selective ACK, NOP, EOL
- Il formato delle opzioni è di due tipi:



EOL (tipo 0)
NOP (tipo 1)

altre opzioni

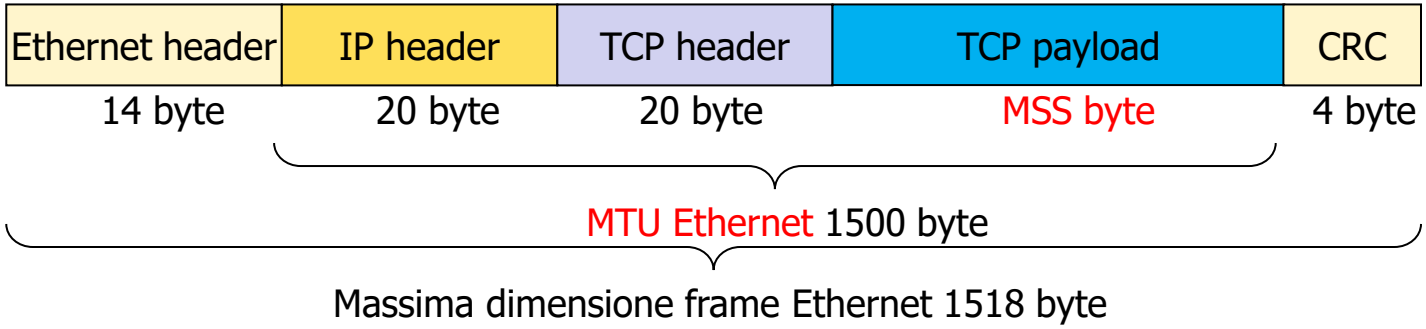
- La maggior parte delle opzioni è consentita solo nella fase di instaurazione della connessione (segmenti con flag SYN=1)

TCP: opzioni nell'header (2)

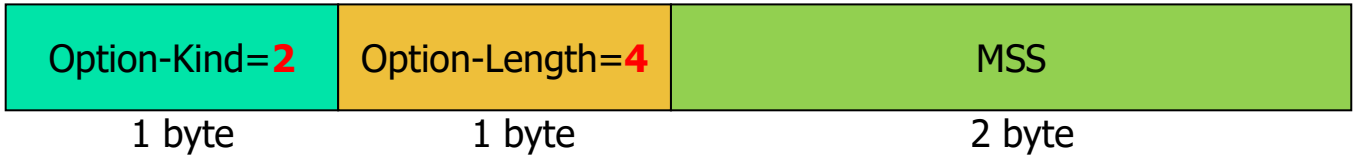
- MSS: durante la fase di connessione, ciascun end-point annuncia la massima dimensione di payload (*MSS – Maximum Segment Size*) che desidera accettare (*MSS announcement*)
- Window Scale: per negoziare un fattore di scala per la finestra; utile per connessioni a larga banda ed elevato ritardo di trasmissione (*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
- Timestamp: utilizzata per aiutare i due endpoint a determinare il RTT (tempo intercorso dalla trasmissione di un segmento alla ricezione del relativo ACK)
- NOP: *No-OPerations*, separa opzioni diverse quando non allineate su multipli di lunghezza di quattro byte
- EOL: *End of Options List*, indica la fine delle opzioni, necessario se la fine delle opzioni non coincide con la fine dell'header TCP

Opzione MSS

- Per MSS in TCP si intende la massima dimensione di payload consentita
- Questo valore è determinato dall'MTU del link sul quale l'host trasmette
- Per un link Ethernet, MTU è 1500 byte e l'MSS è 1460 byte



- L'opzione MSS di TCP consente a ciascuno dei due endpoint di notificare all'altro la massima dimensione di segmento da utilizzare



- Se l'altro endpoint non ha trasmesso l'opzione MSS nel segmento SYN, RFC1122 (*Requirements for Internet Hosts*) prescrive che MSS sia posto a 536 byte

TCP: affidabilità della consegna dati

- Ottenuta tramite:
  - 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
- Il campo «numero di sequenza» di un segmento TCP indica il numero di sequenza progressivo 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:

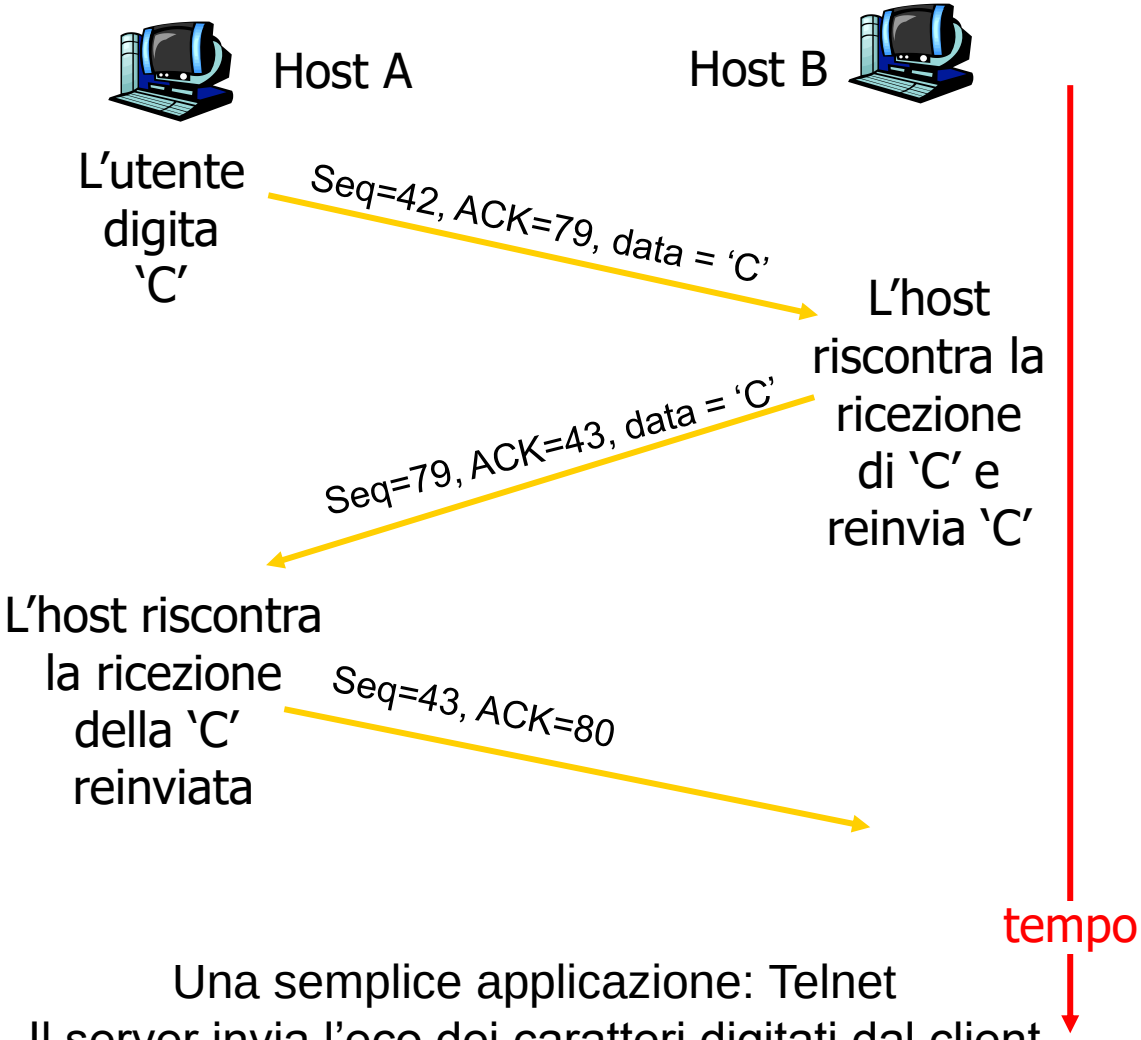
- "numero" del primo byte del segmento nel flusso di byte

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

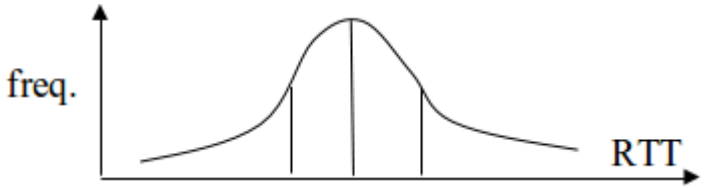
TCP: Caratteristiche del riscontro cumulativo

- Nel datagramma di riscontro la destinazione comunica quale byte dello stream si aspetta di ricevere successivamente
  - Il riscontri specificano 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

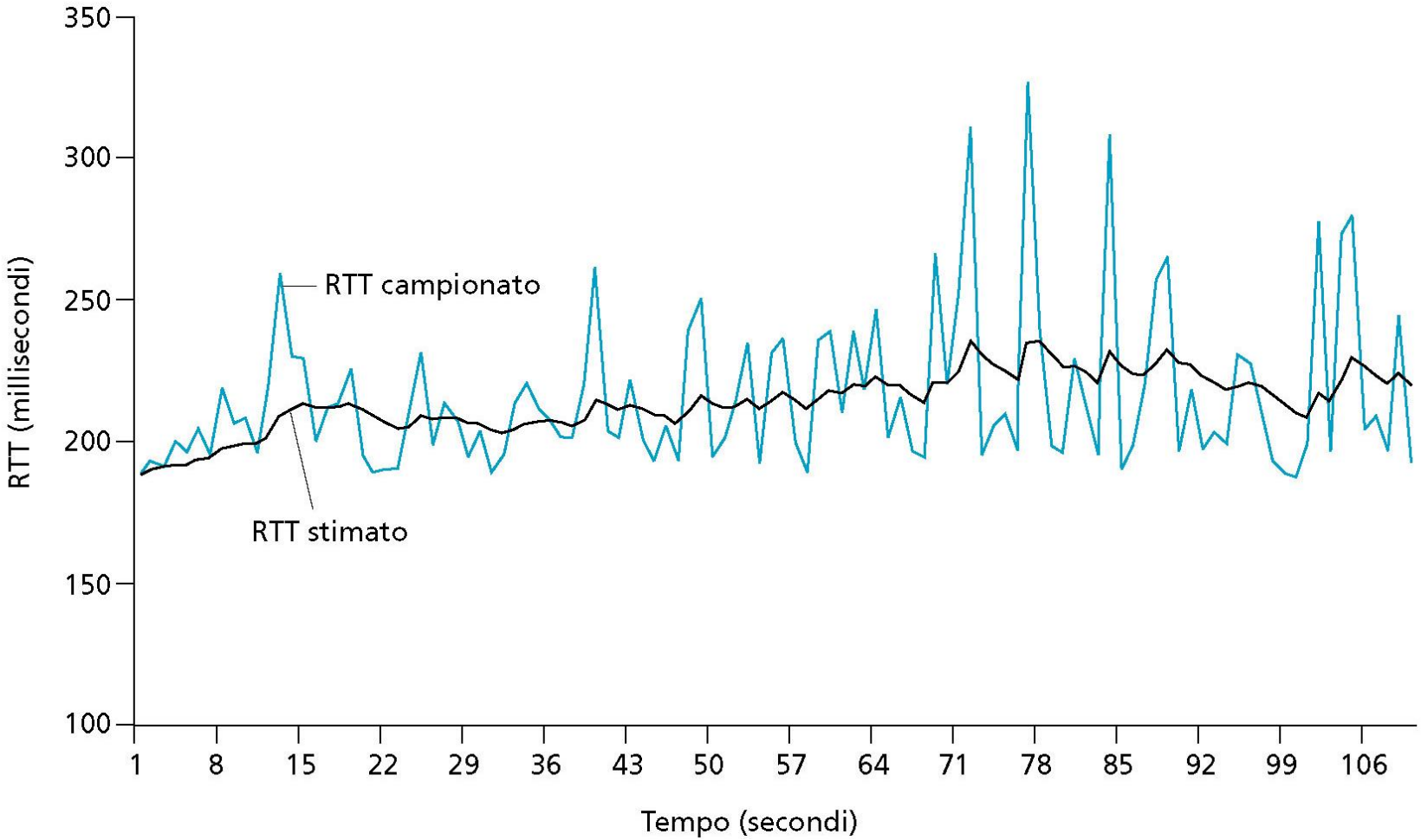
Round Trip Time e Timeout

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

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

- **Stima della deviazione di RTT**:

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

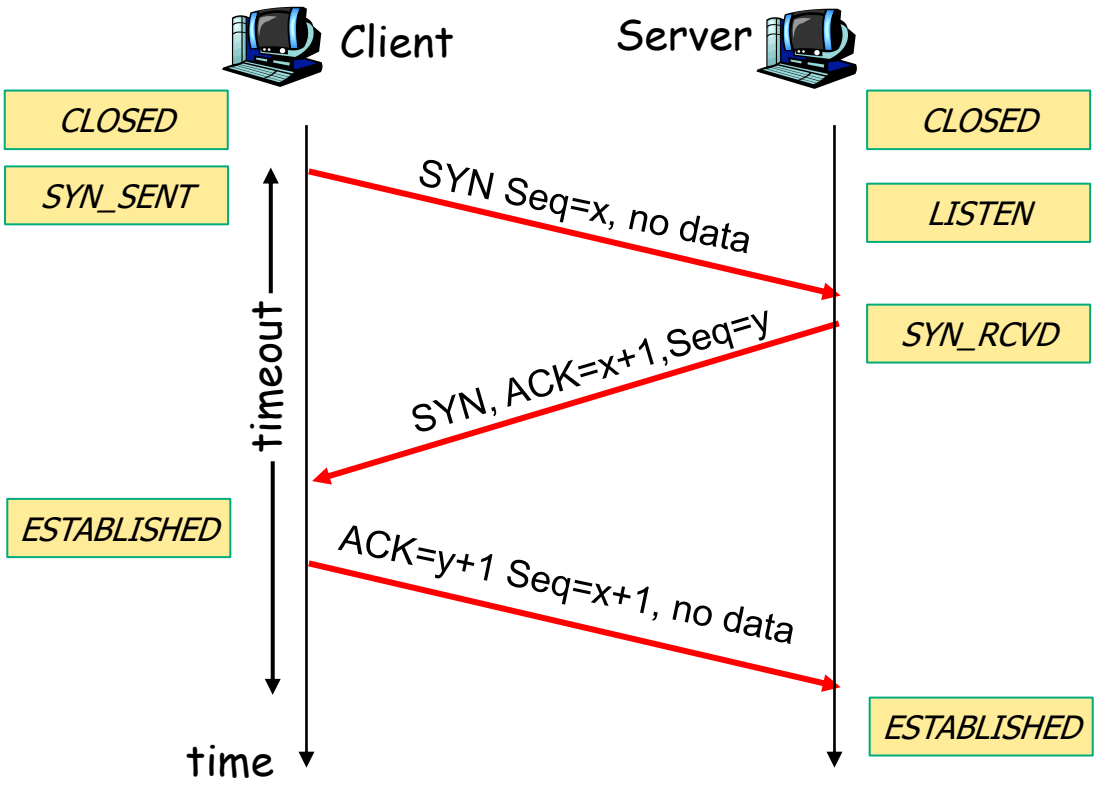
  - Valore raccomandato per β: 0,25

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

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

TCP connection establishment (1)

- Procedura detta *three-way-handshake*
- Il client prende l'iniziativa inviando il primo segmento



TCP connection establishment (2)

Nella fase three-way-handshake i due endpoint allocano i buffer per i dati e concordano sui valori iniziali dei numeri di sequenza da utilizzare per gli stream dati in entrambi i versi

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

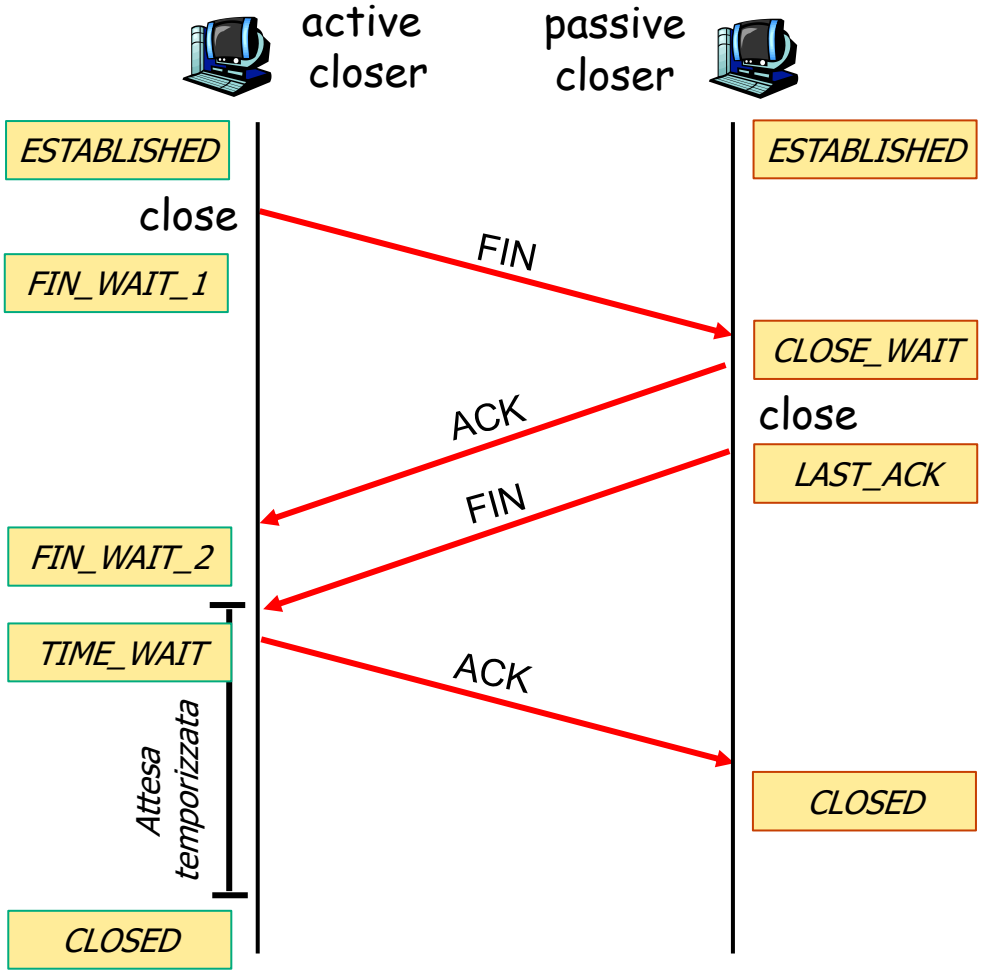
- Connessione instaurata

Nella fase di three-way-handshake inoltre:

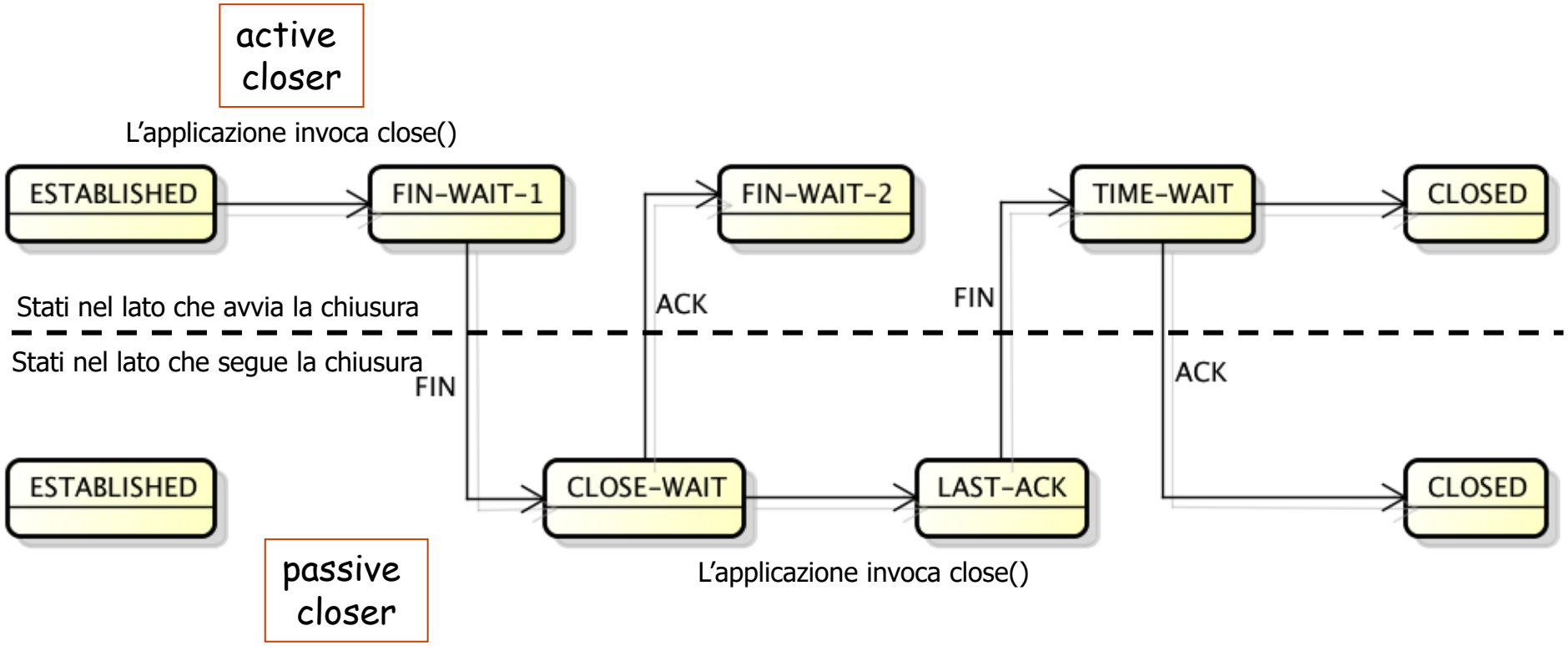
- le due parti apprendono il valore iniziale di Recv Window
- utilizzano le opzioni TCP per determinare il valore di MSS, RTT, ecc.

Connection termination

- Procedura detta *four-way-handshake*
- La parte che manifesta per prima la volontà di chiudere la connessione è detta *active closer*, l'altra è il *passive closer*
  - L'*active closer* può essere il client oppure il server a seconda dei protocolli applicativi
- Se non ha altri dati da inviare, il *passive closer* può inviare ACK e FIN nello stesso segmento
- L'*active closer* attende in uno stato TIME_WAIT (nel caso in cui l'ultimo ACK vada perso, e riceva un ulteriore FIN dal *passive closer*)
- La durata della permanenza nello stato TIME_WAIT dipende dal sistema operativo (tipicamente 120s)

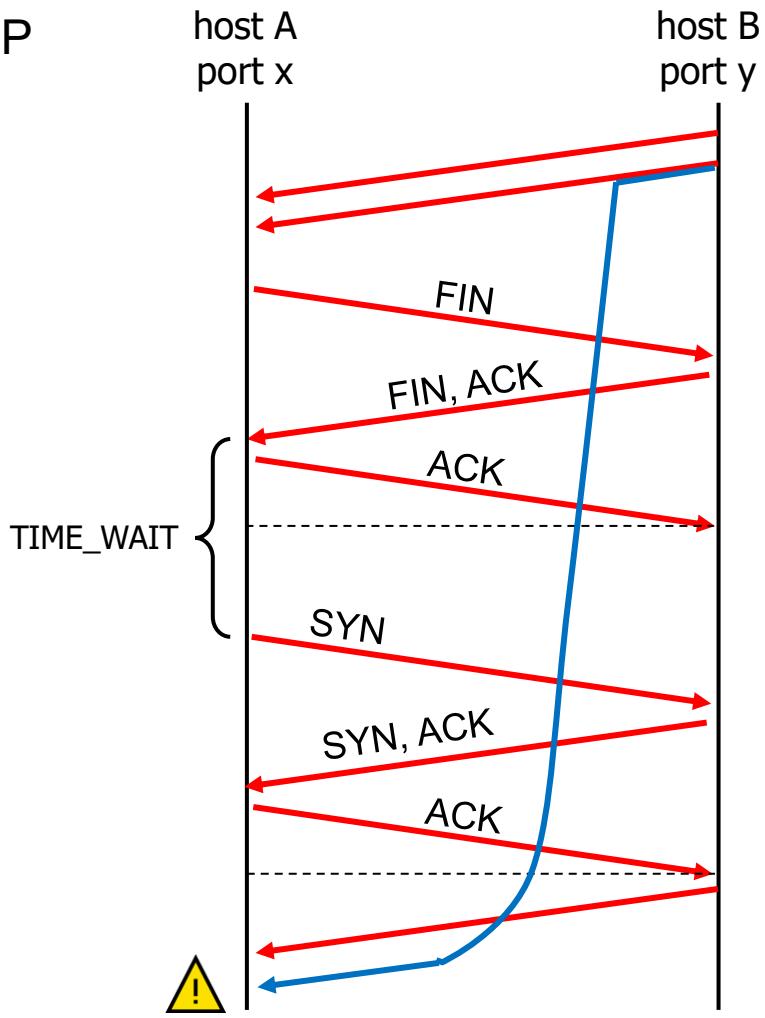


Sequenze di stati in fase di chiusura

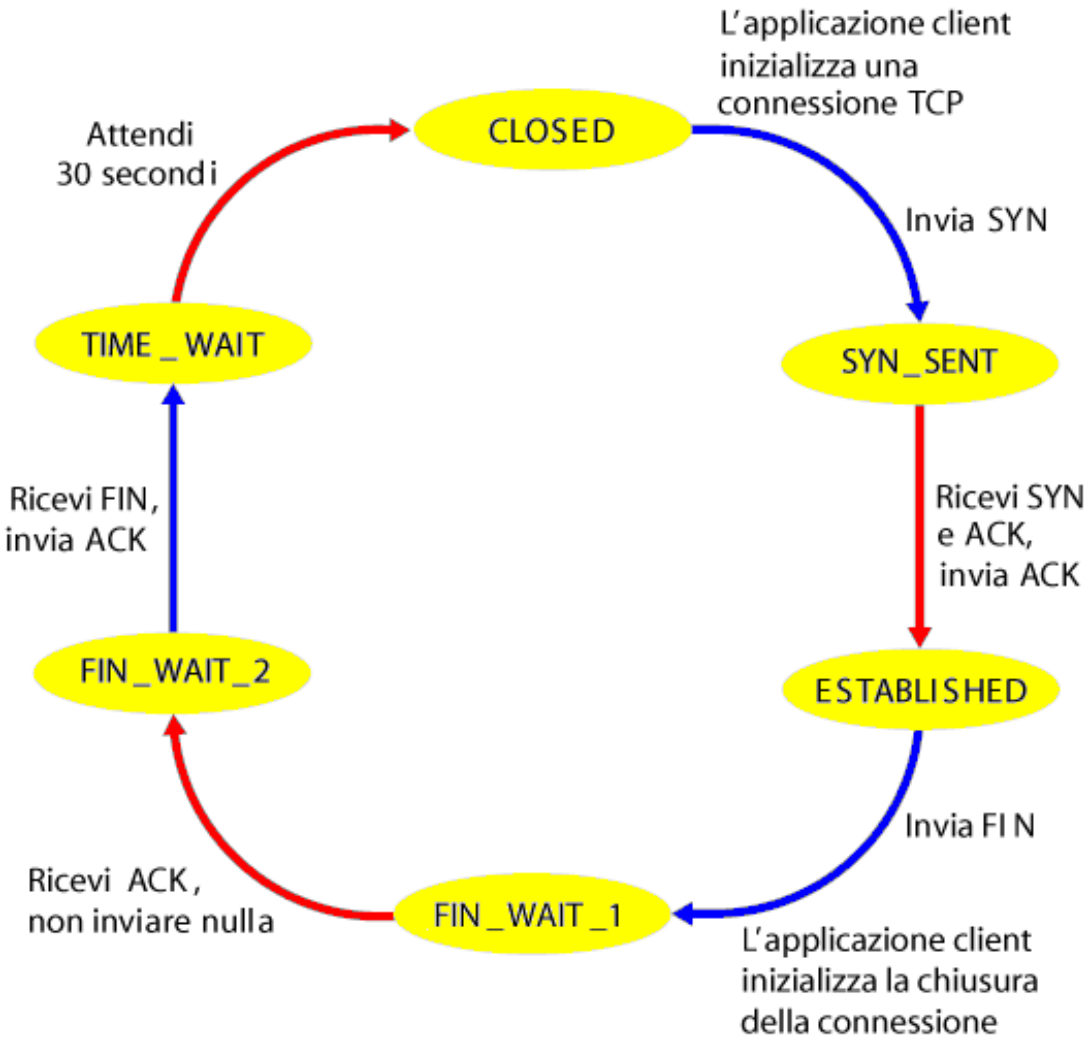


Necessità dello stato TIME_WAIT

- Lo stato TIME_WAIT è raggiunto dall'endpoint che ha preso l'iniziativa di chiudere la connessione TCP (*active closer*)
- Finché l'endpoint (A,x) rimane in TIME_WAIT non è possibile la creazione di una nuova connessione tra gli stessi endpoint (A,x) e (B,y)
- Ciò serve ad evitare uno scenario come in figura
- Un segmento dati duplicato prodotto da una precedente connessione tra (A,x) e (B,y) il cui numero di sequenza iniziale rientri nella finestra di ricezione corrente sarebbe erroneamente ritenuto un segmento «valido» per la nuova connessione



Sequenza tipica degli stati nel client



Sequenza tipica degli stati nel server

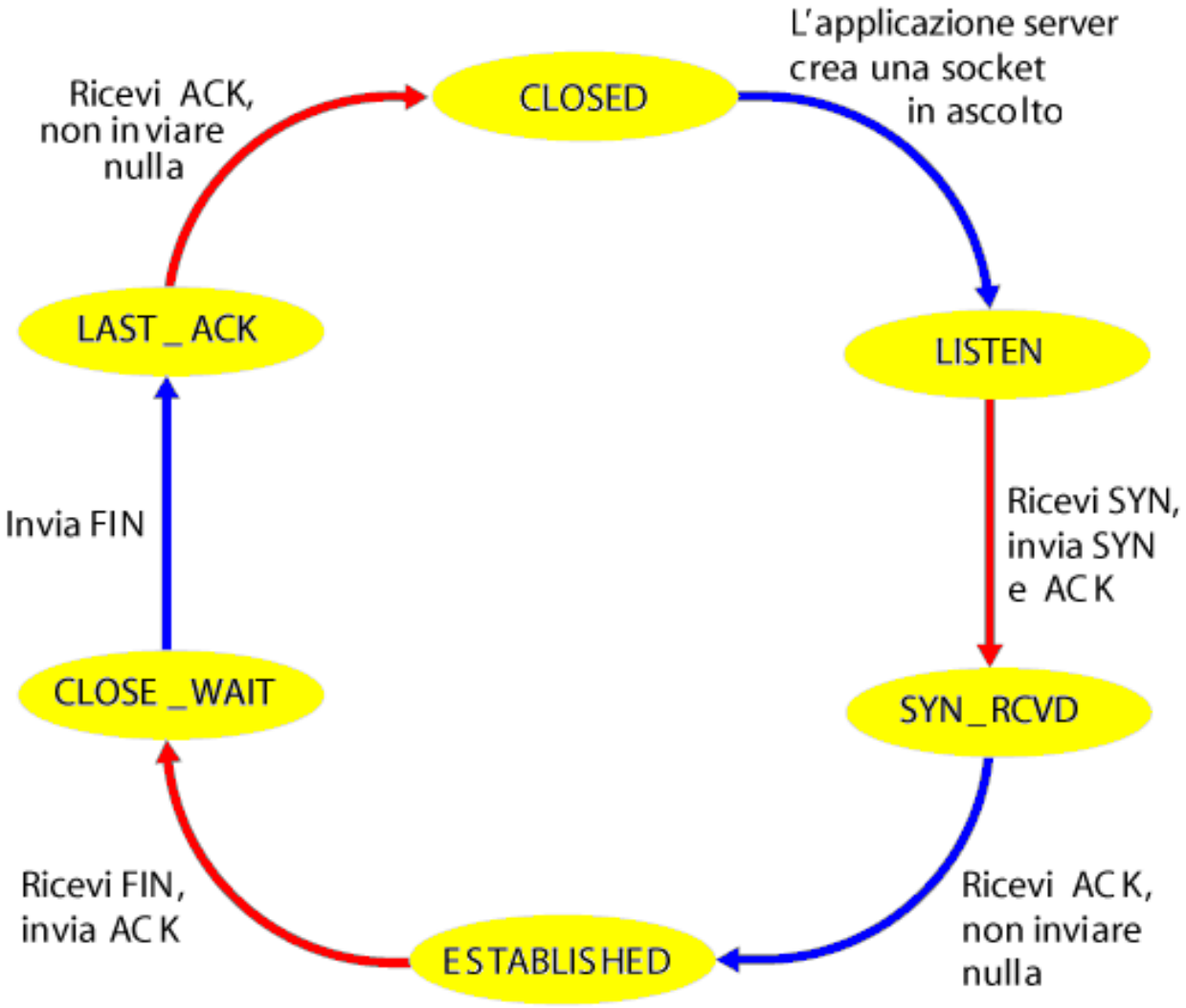
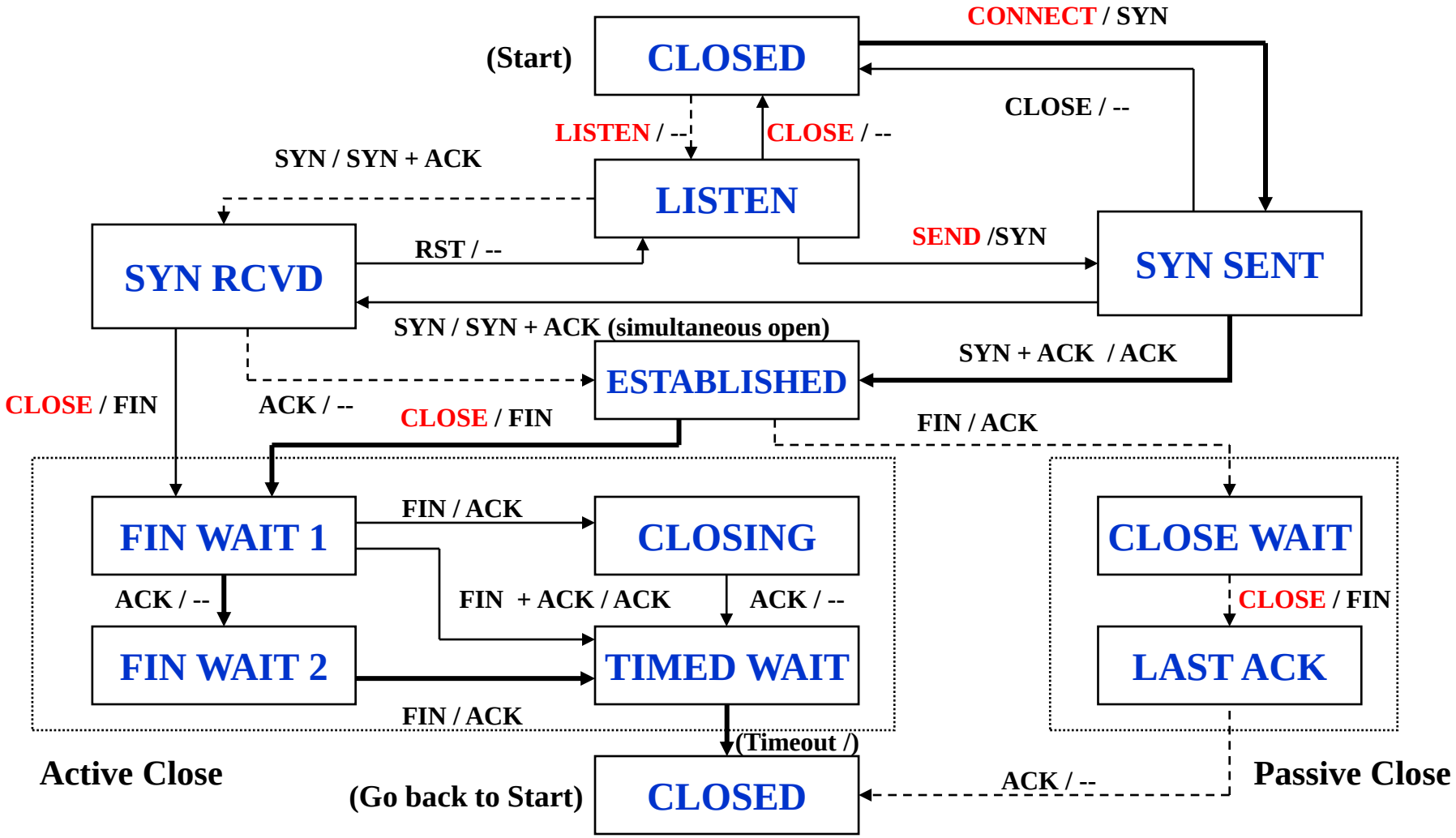


Diagramma degli stati completo del TCP



TCP: trasferimento dati affidabile

- TCP crea un servizio di trasferimento dati affidabile sul servizio inaffidabile di IP
- ACK cumulativi
- <u>TCP usa un solo timer di ritrasmissione</u>
- 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

Università degli Studi di Napoli Federico II

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:

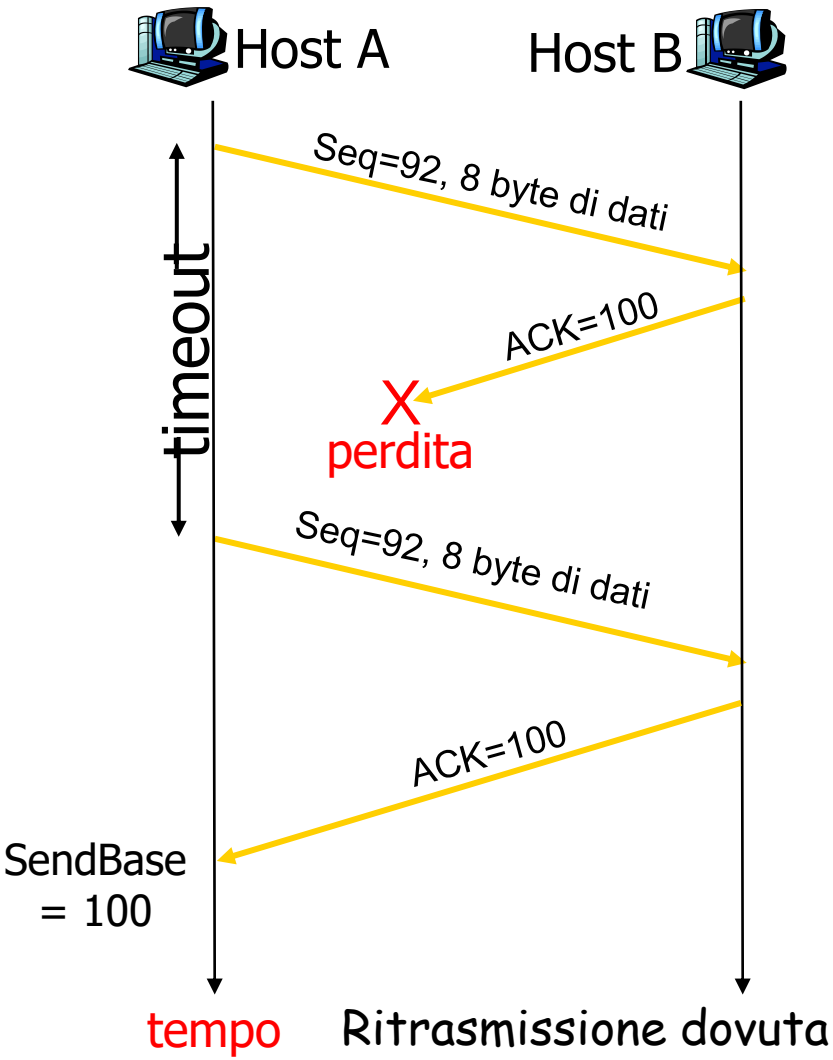
- Se riscontra segmenti precedentemente non riscontrati
  - aggiorna ciò che è stato completamente riscontrato
  - avvia il timer se ci sono altri segmenti da completare

Un sender TCP semplificato

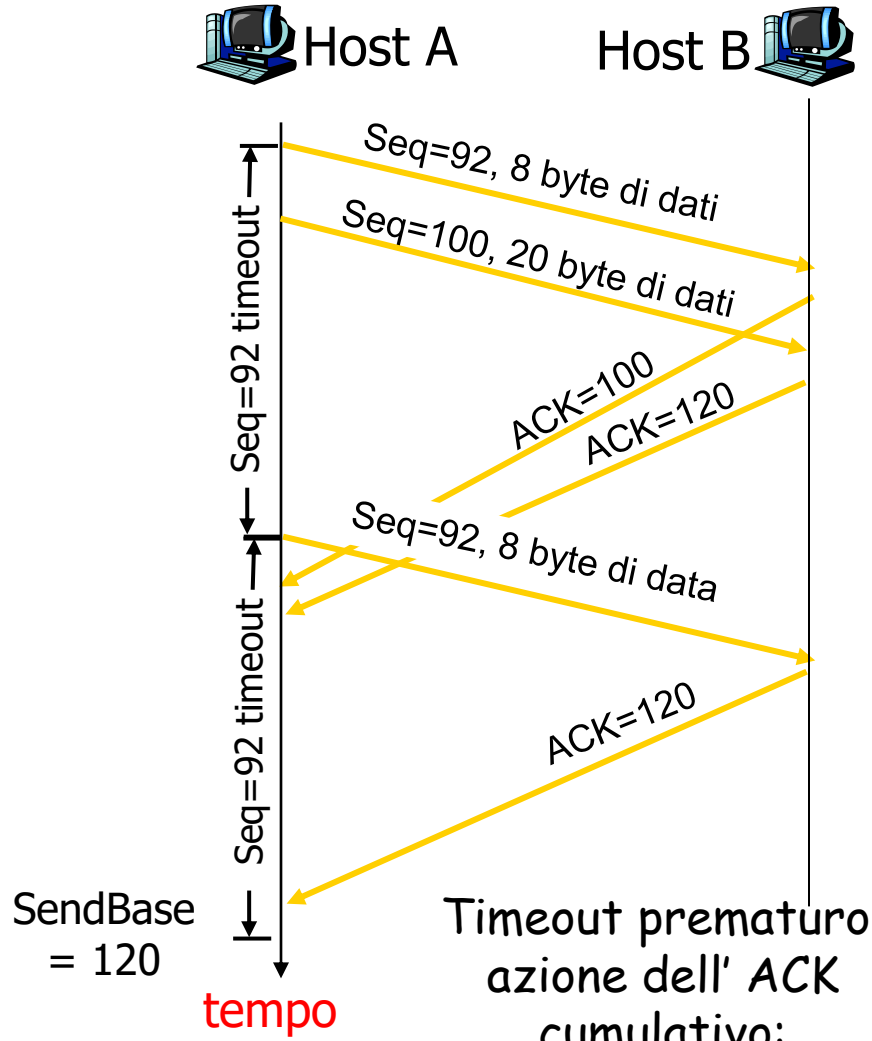
/* Si è assunto che il sender non sia limitato dal controllo di flusso o di congestione del TCP, che i dati da sopra siano di dimensioni inferiori all'MSS e che il trasferimento dei dati avvenga in una sola direzione.*/

```
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

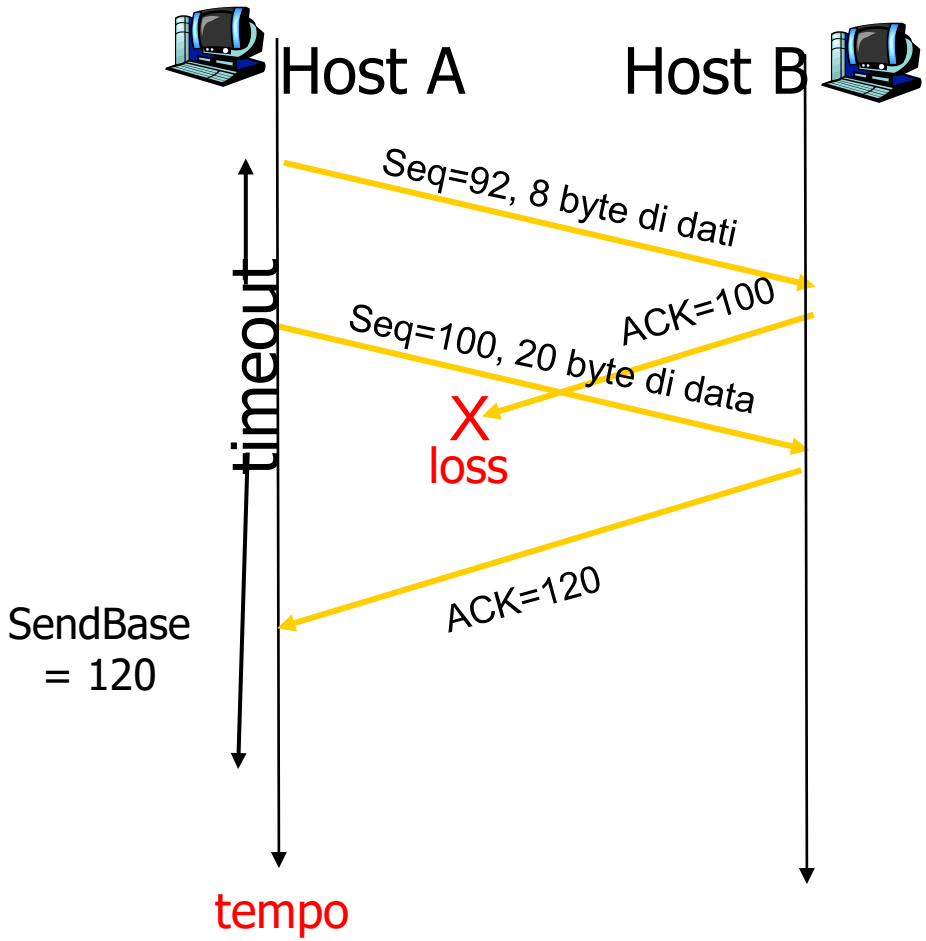


Ritrasmissione dovuta ad un riscontro perso



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, 2581, 5681]

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.