Transmission Control Protocol: TCP

Prof.ssa Gaia Maselli

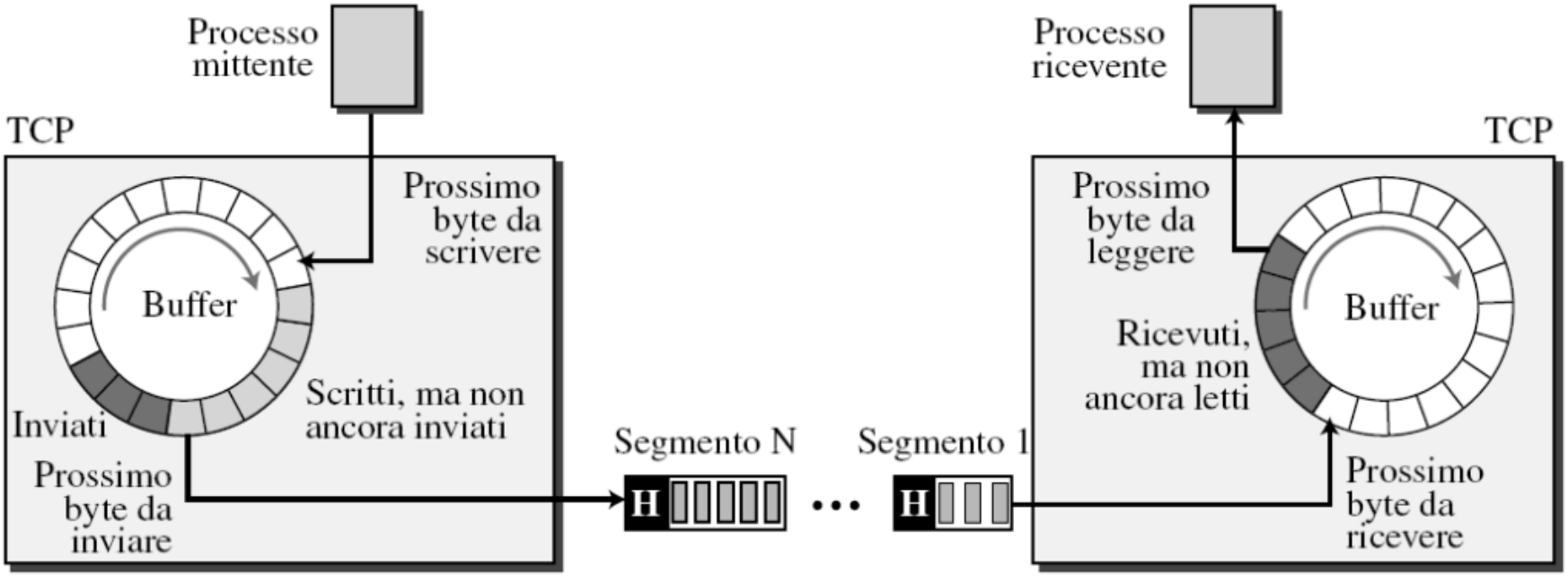
Parte di queste slide sono state prese dal materiale associato ai libri:

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

Protocollo TCP

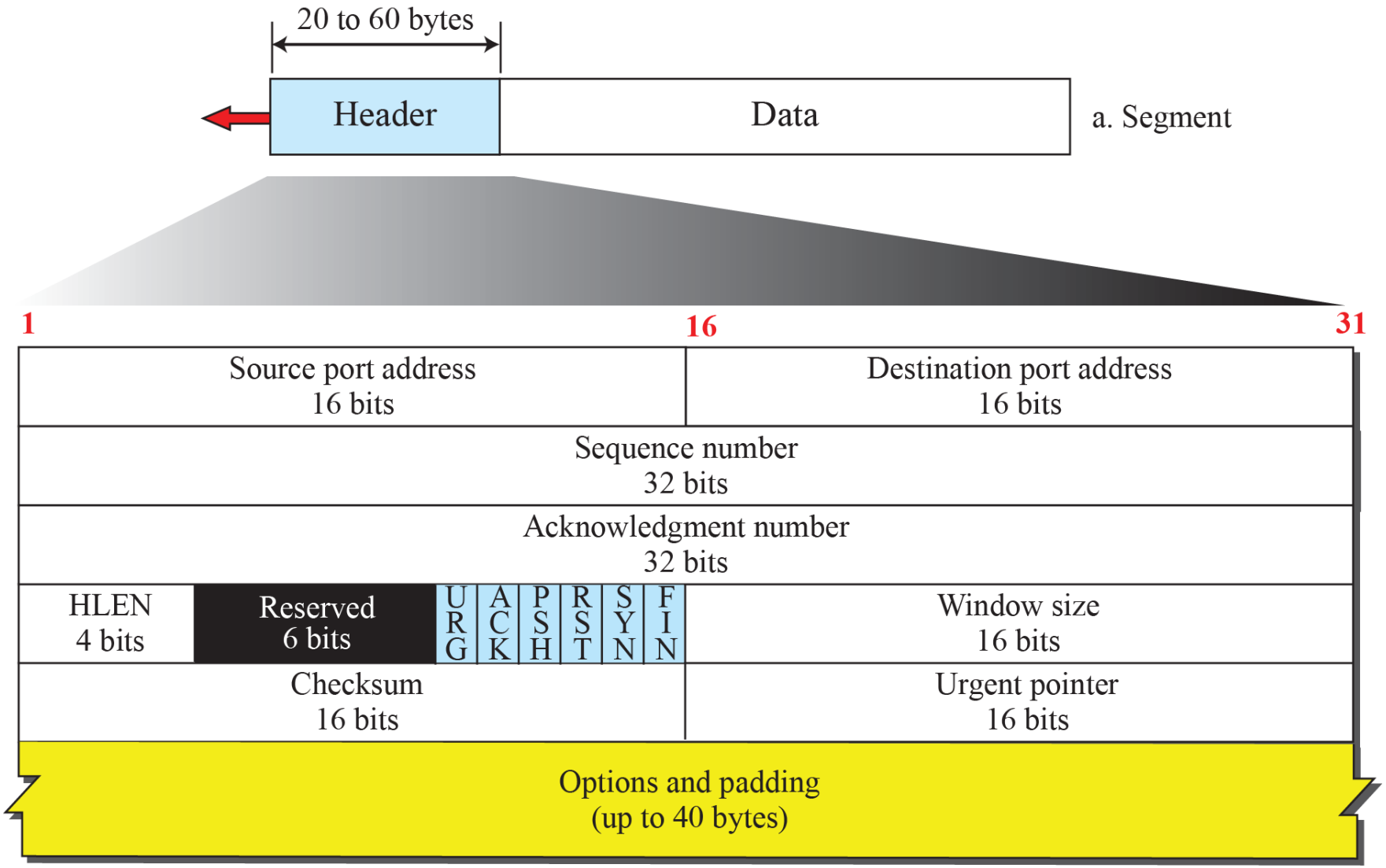
- Protocollo con pipeline
- Bidirezionale (ack in piggybacking)
- Orientato al flusso di dati (stream-oriented)
- Orientato alla connessione
- Affidabile (controllo degli errori)
- Controllo del flusso
- Controllo della congestione

Segmenti TCP

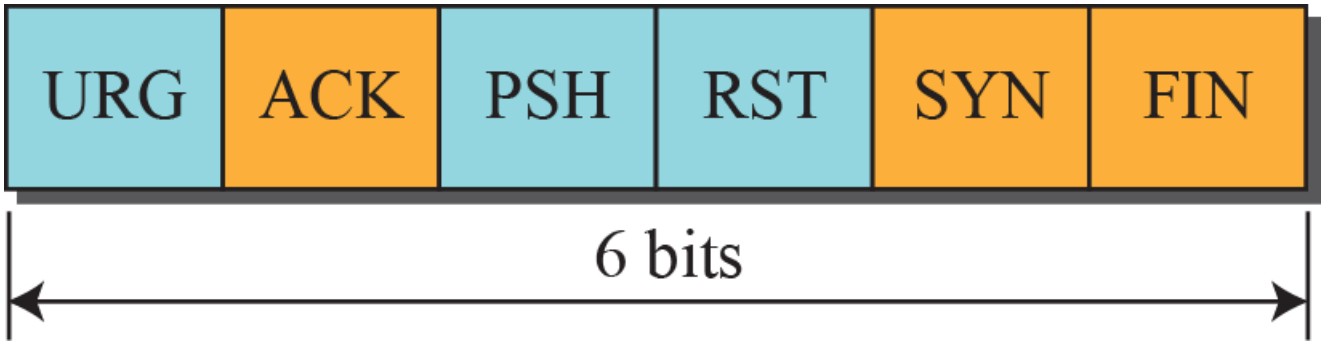


- Il TCP riceve uno stream di byte dal processo (applicazione) mittente
- TCP utilizza il servizio di comunicazione tra host del livello di rete che invia pacchetti
- TCP deve quindi raggruppare un certo numero di byte in ***segmenti***, aggiungere un'intestazione e consegnare al livello di rete per la trasmissione

Struttura dei segmenti



Flag di controllo

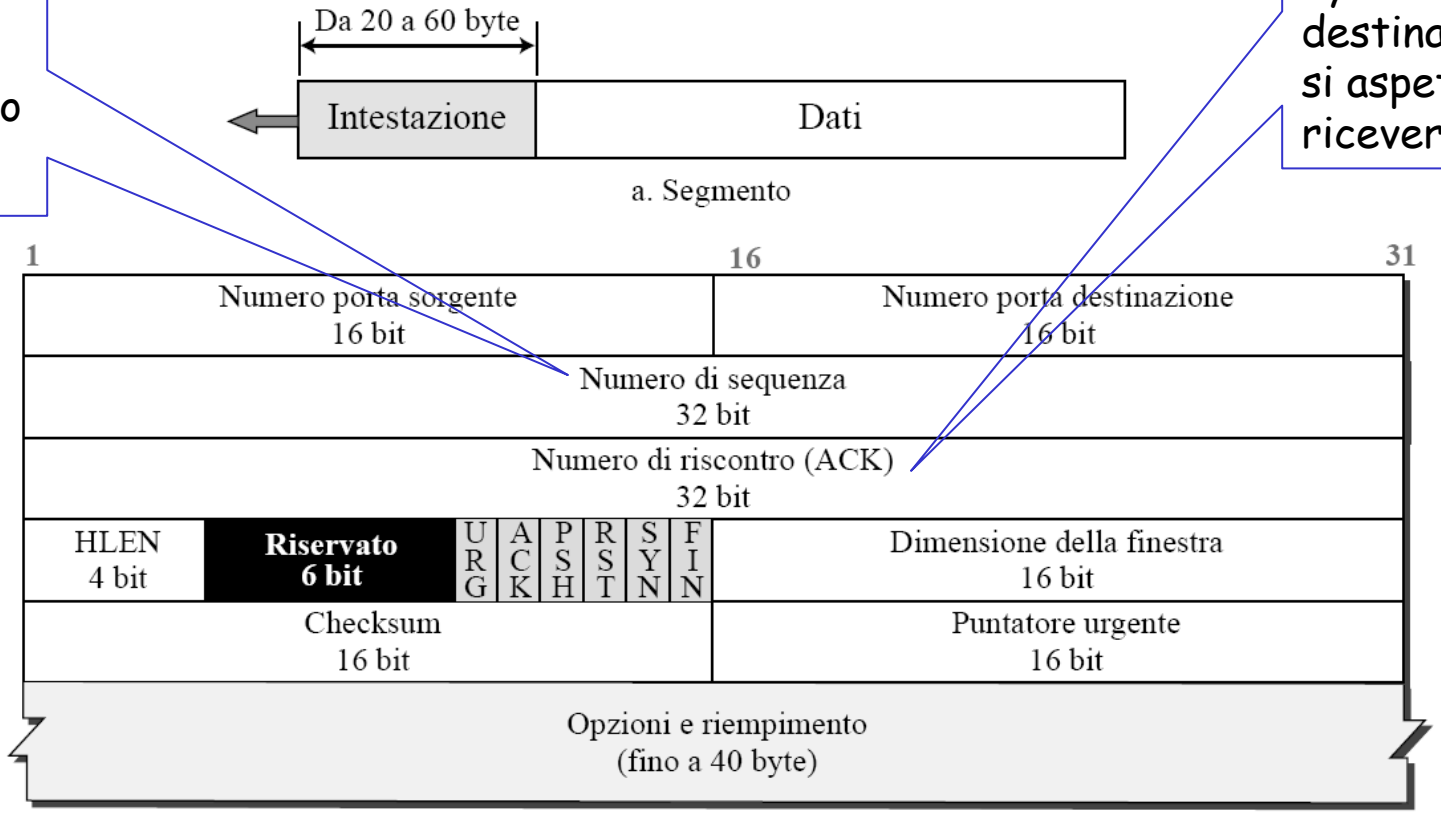


URG: puntatore urgente valido
ACK: riscontro valido
PSH: richiesta di push
RST: azzeramento della connessione
SYN: sincronizzazione dei numeri di sequenza
FIN: chiusura della connessione

Struttura dei segmenti TCP

Numero del primo byte di dati contenuto nel segmento

Numero di sequenza del prossimo byte che il destinatario si aspetta di ricevere

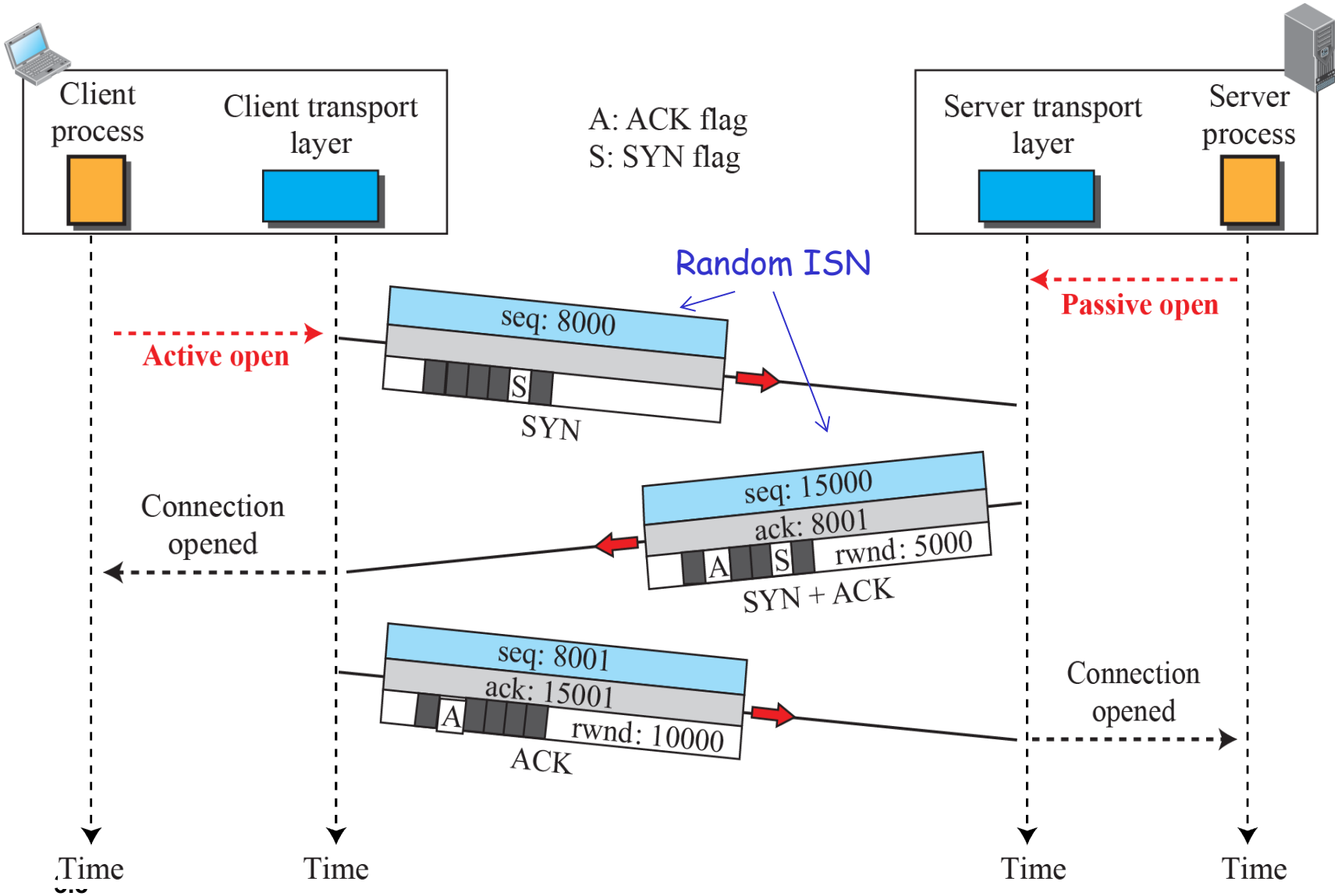


Connessione TCP

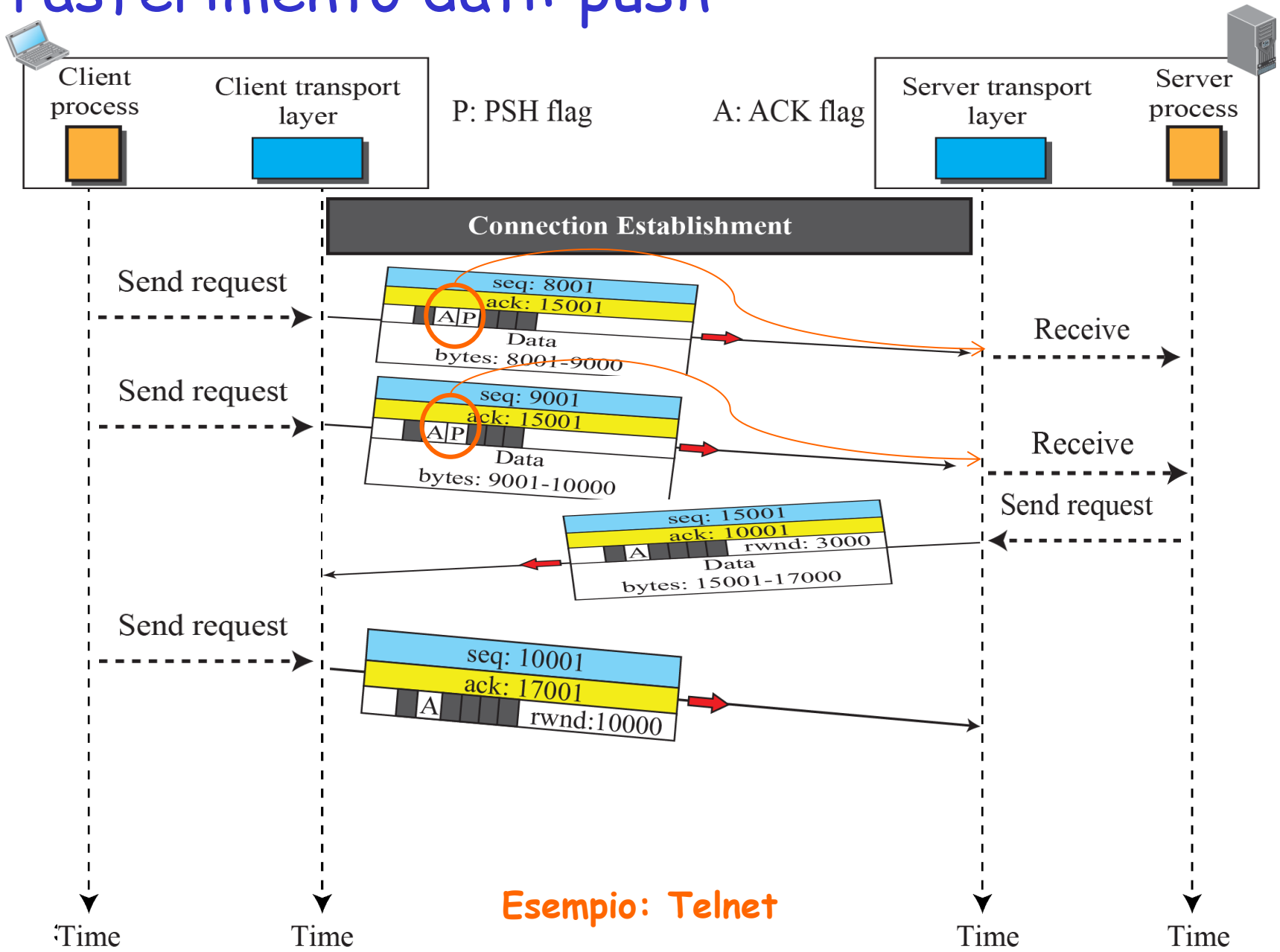
- Percorso virtuale tra il mittente e il destinatario, sopra IP che è privo di connessione

- 3 fasi:
  1. Apertura della connessione
  2. Trasferimento dei dati
  3. Chiusura della connessione

Apertura della connessione: 3 way handshake



Trasferimento dati: push



Esempio: Telnet

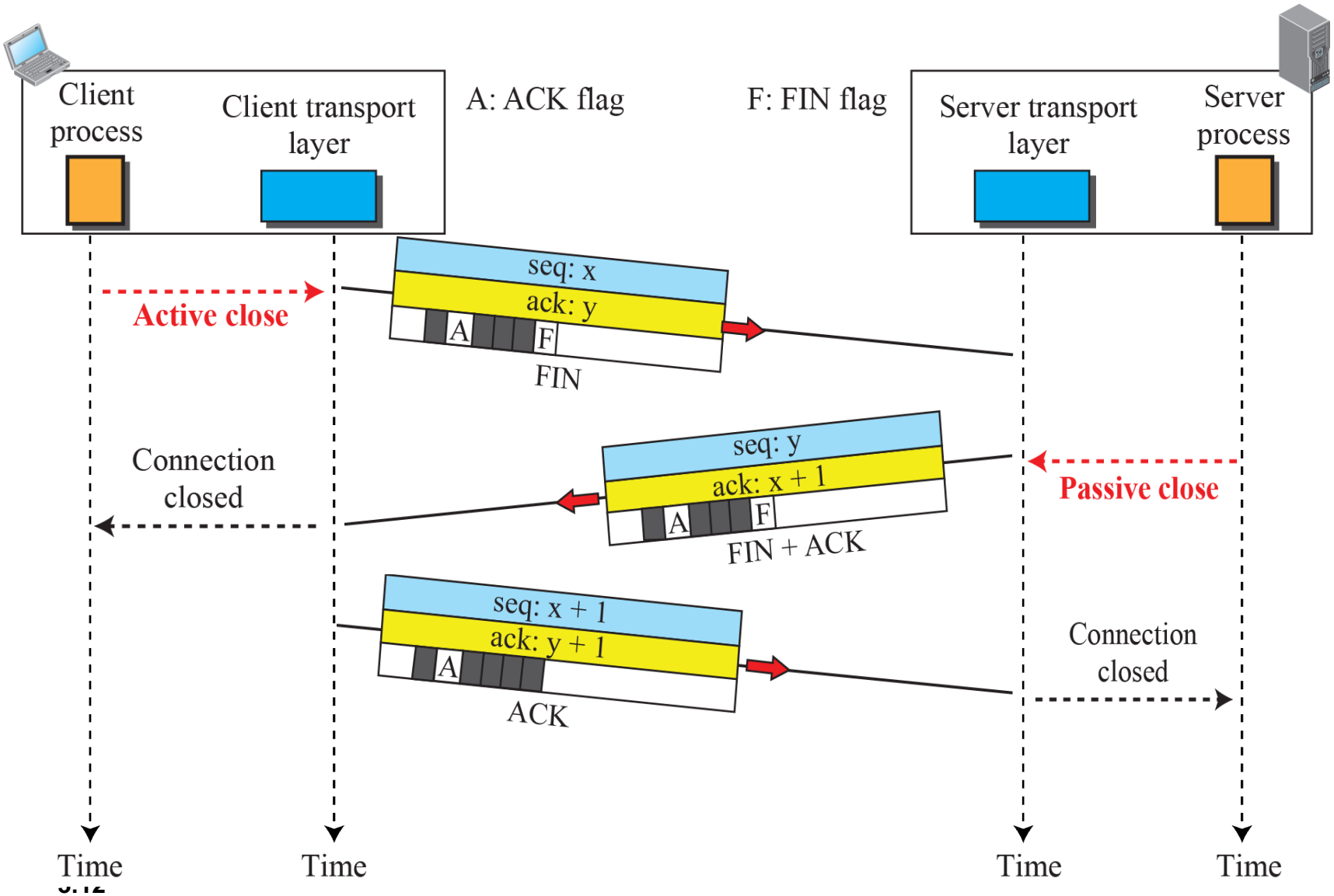
Trasferimento dati: urgent

- Dati URG vengono elaborati subito indipendentemente dalla loro posizione nel flusso
- Flag URG + puntatore urgente (16 bit)
- I dati urgenti vengono inseriti all'inizio di un nuovo segmento, che può contenere dati non urgenti a seguire
- Il campo puntatore nell'intestazione indica dove finiscono i dati urgenti e iniziano quelli normali.
- Esempio: interruzione data transfer

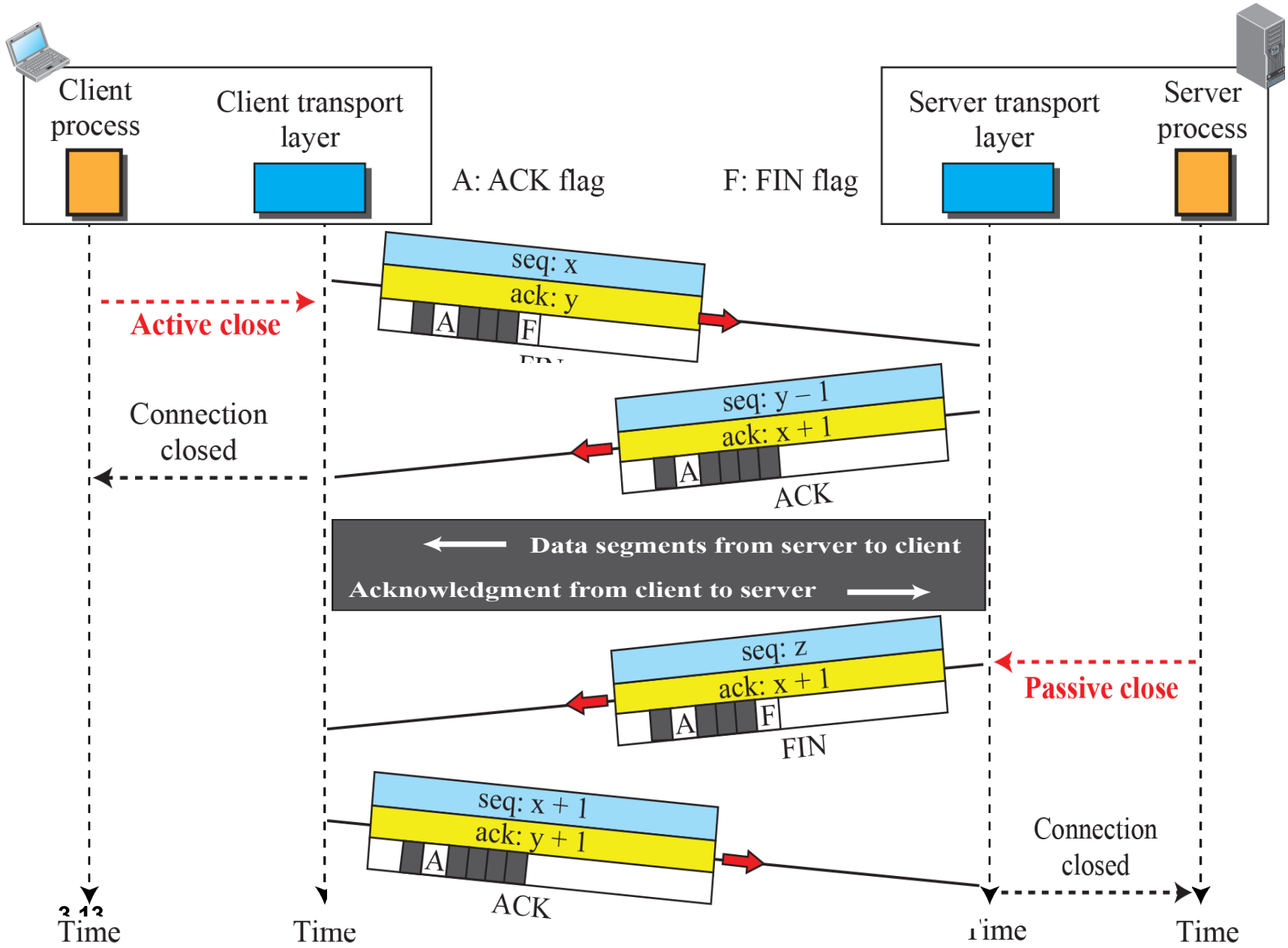
Chiusura della connessione

- Ciascuna delle due parti coinvolta nello scambio di dati può richiedere la chiusura della connessione, sebbene sia solitamente richiesta dal client, oppure timer nel server (se non si ricevono richieste entro un determinato tempo si chiude la connessione)
- Doppio scambio di messaggi FIN e ACK

Chiusura della connessione



Chiusura della connessione: half close



Controllo degli errori

Numeri di sequenza e ACK di TCP

Numeri di sequenza:

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

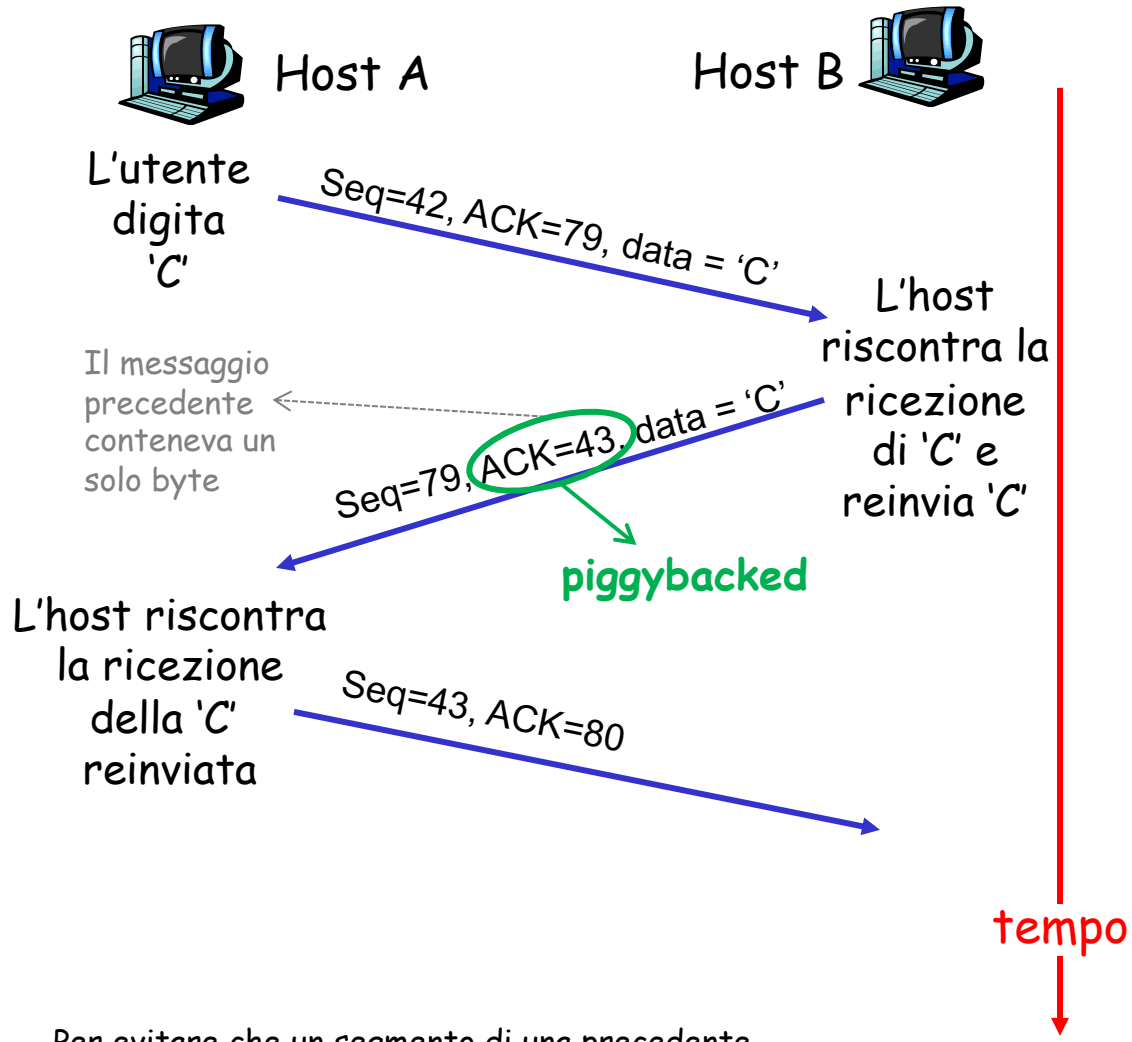
ACK:

- 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 – solitamente il destinatario mantiene i byte non ordinati

Una semplice applicazione Telnet



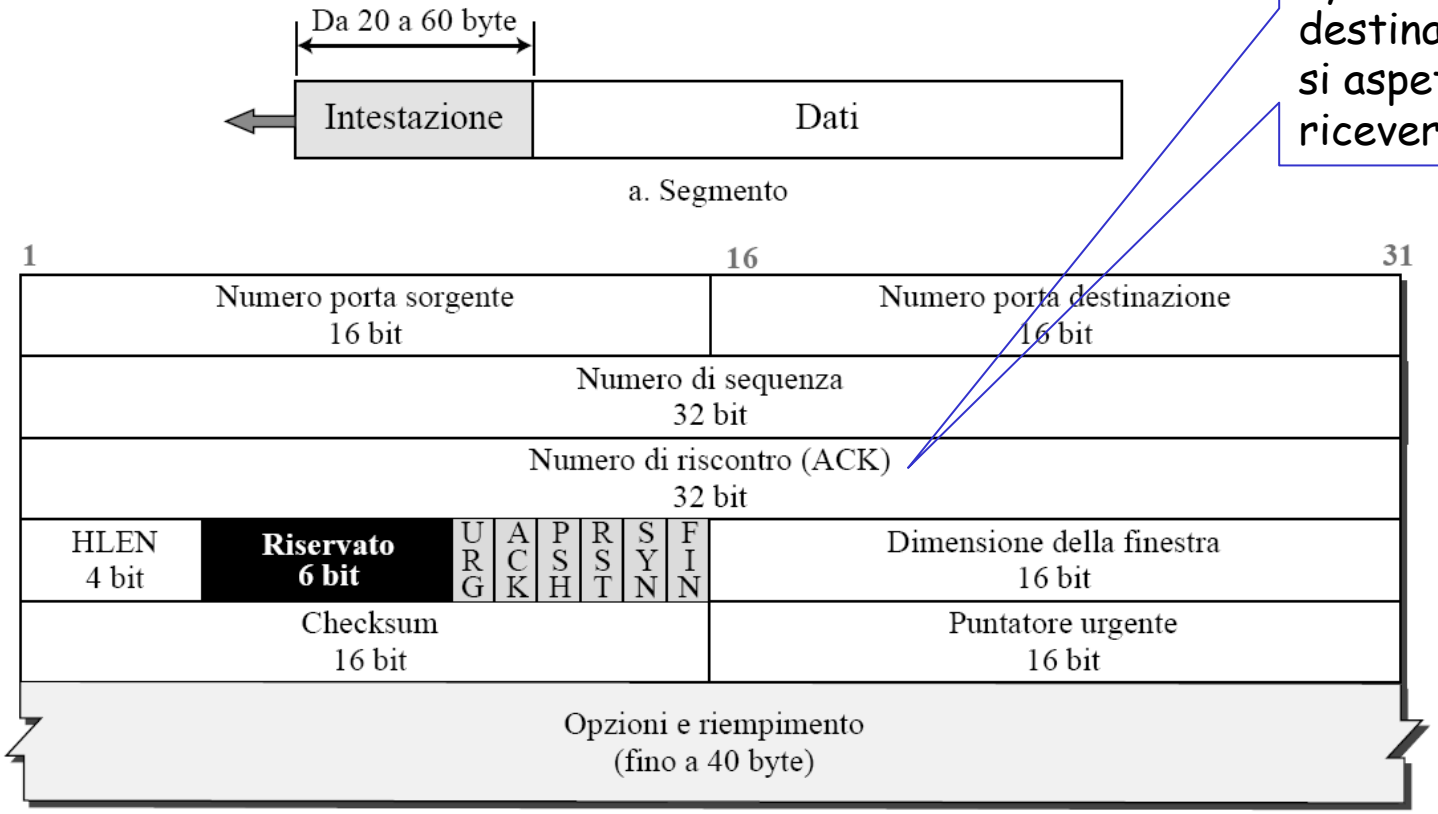
N.B. Il numero di sequenza iniziale è scelto a caso → Per evitare che un segmento di una precedente connessione ancora presente in rete possa essere interpretato come valido per la nuova connessione

TCP: affidabilità (controllo degli errori)

- Checksum
  - Se un segmento arriva corrotto viene scartato dal destinatario
- Riscontri + timer di ritrasmissione (RTO)
  - Ack cumulativi
  - Timer associato al più vecchio pacchetto non riscontrato
- Ritrasmissione
  - Ritrasmissione del segmento all'inizio della coda di spedizione

Perché ACK cumulativi?

Numero di sequenza del prossimo byte che il destinatario si aspetta di ricevere



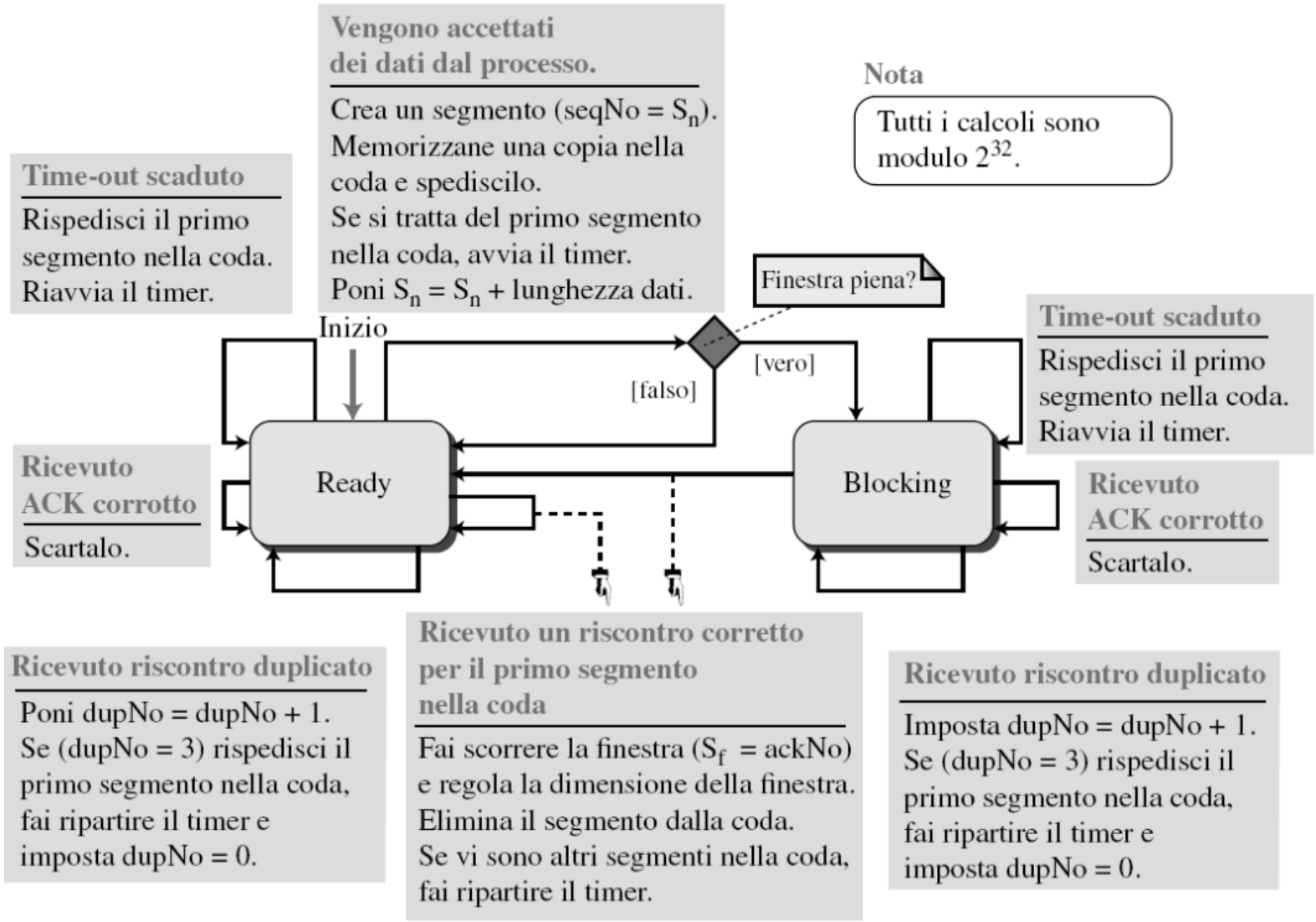
TCP: generazione di ACK [RFC 1122, RFC 2581]

	Evento	Azione
2	Arrivo ordinato di un segmento con numero di sequenza atteso. Tutti i dati fino al numero di sequenza atteso sono già stati riscontrati.	ACK delayed. Attende fino a 500 ms l'arrivo del prossimo segmento. Se il segmento non arriva, invia un ACK.
3	Arrivo ordinato di un segmento con numero di sequenza atteso. Un altro segmento è in attesa di trasmissione dell'ACK (vedi precedente).	Invia immediatamente un singolo ACK cumulativo, riscontrando entrambi i segmenti ordinati.
4	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 (ritrasmissione rapida).
5	Arrivo di un segmento mancante (uno o più dei successivi è stato ricevuto).	Invia immediatamente un ACK
6	Arrivo di un segmento duplicato	Invia immediatamente un riscontro con numero di sequenza atteso

Ritrasmissione dei segmenti

- Quando un segmento viene inviato, una copia viene memorizzata in una coda in attesa di essere riscontrato (finestra di invio)
- Se il segmento non viene riscontrato può accadere che
  - Scade il timer (è il primo segmento all'inizio della coda) ➔il segmento viene ritrasmesso e viene riavviato il timer
  - Vengono ricevuti 3 ack duplicati ➔ ***ritrasmissione veloce*** del segmento (senza attendere il timeout)

FSM mittente



FSM destinatario

Ricevuto un segmento atteso senza errori

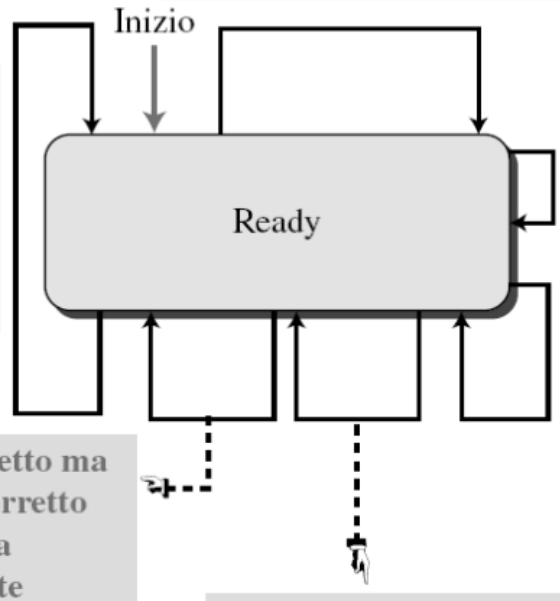
Memorizza il messaggio nel buffer.
$R_n = R_n$ + lunghezza dei dati.
Se il timer di ACK-posticipato è già attivo, interrompilo e invia un riscontro cumulativo.
Altrimenti, fai partire il timer.

Nota

Tutti i calcoli sono modulo 2^{32}.

Ricevuta richiesta di trasmissione di k byte di dati dal processo

Consegna i dati.
Fai scorrere la finestra e aggiorna la sua dimensione.



Timer di ACK posticipato scaduto

Invia ACK-posticipato.

Ricevuto un segmento senza errori ma fuori sequenza

Memorizza il segmento se non si tratta di un duplicato.
Invia un riscontro con ackNo uguale al numero di sequenza del segmento atteso (ACK duplicato).

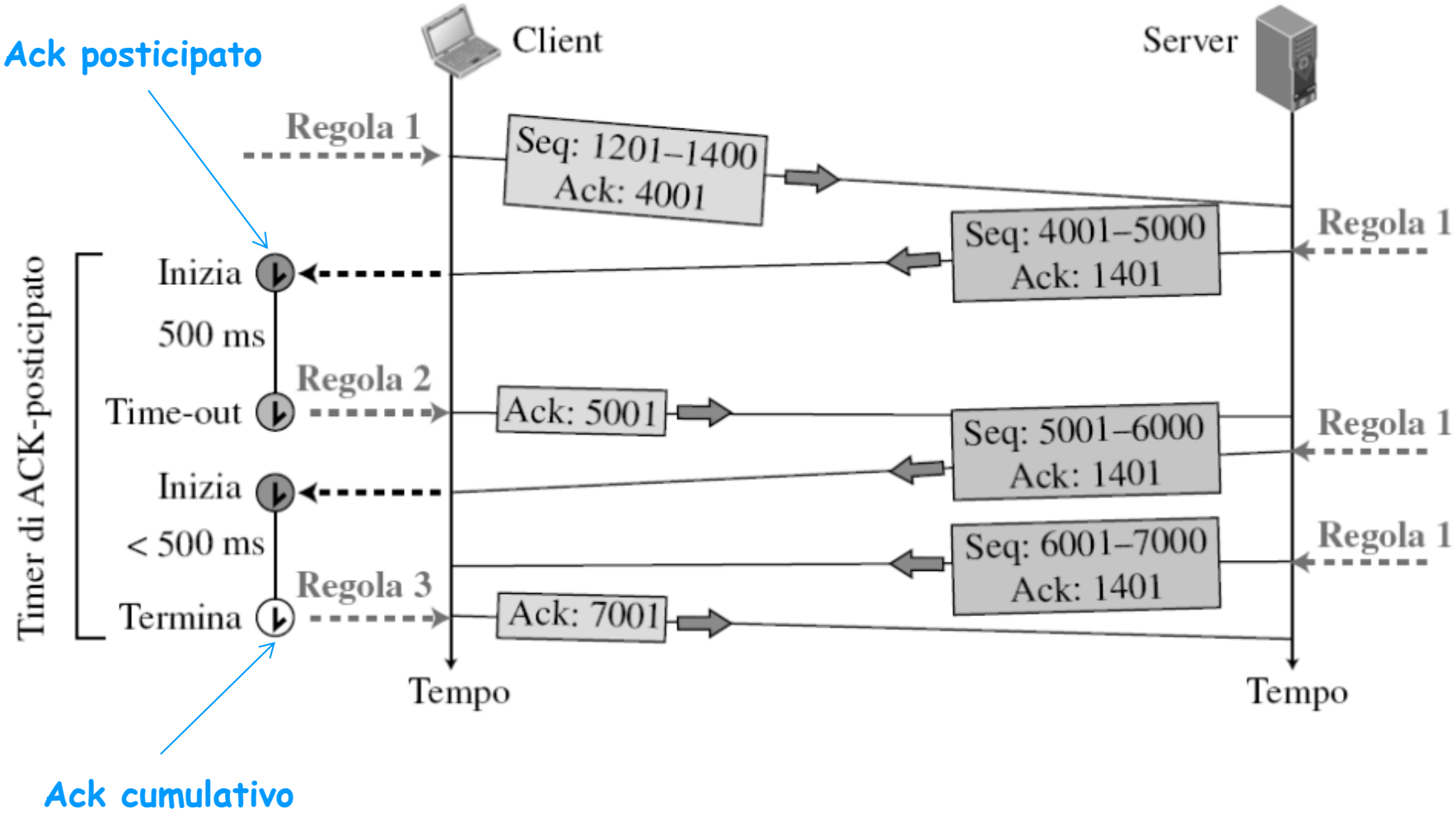
Ricevuto un segmento corretto ma duplicato o un segmento corretto ma con numero di sequenza esterno alla finestra corrente

Scarta il segmento.
Invia un riscontro con ackNo uguale al numero di sequenza del segmento atteso (ACK duplicato).

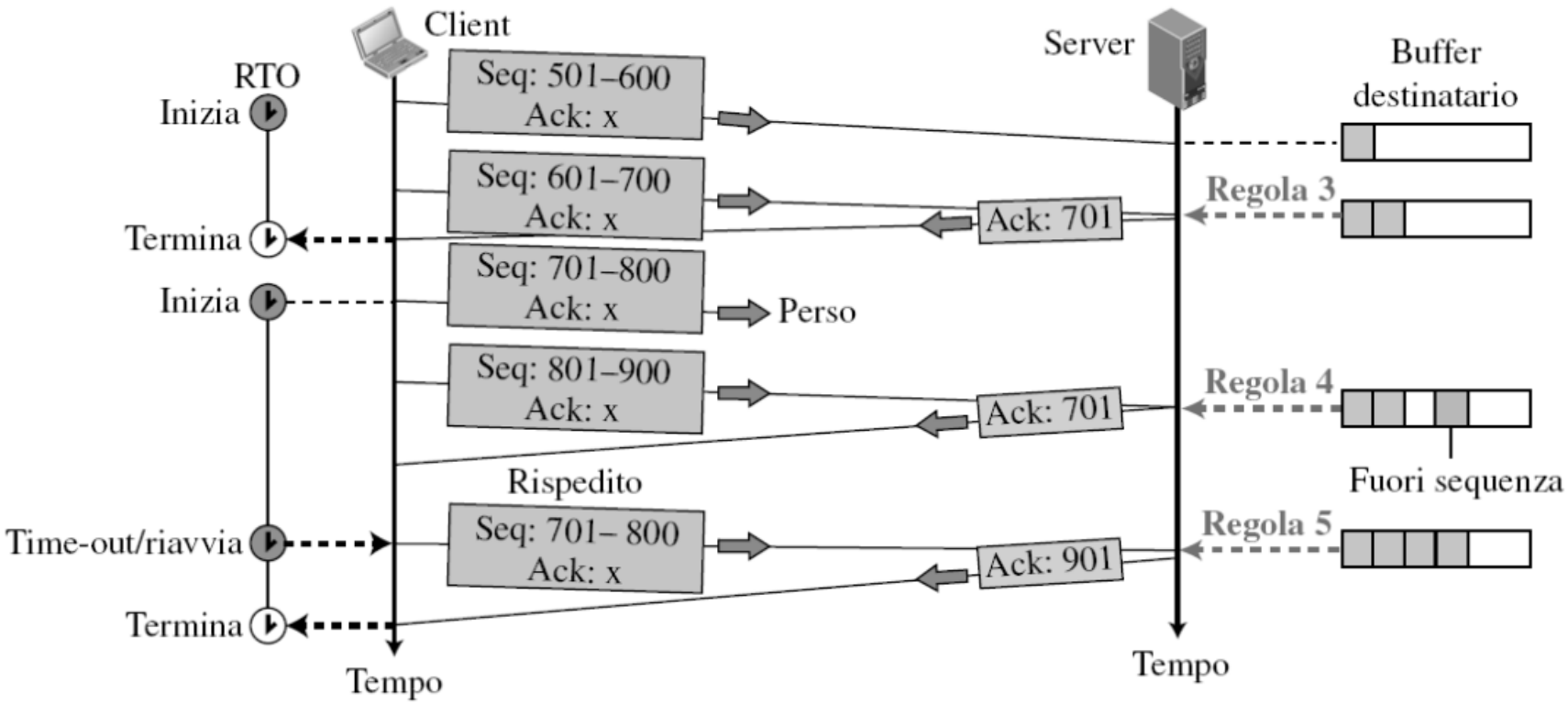
Ricevuto un segmento corrotto

Scarta il segmento.

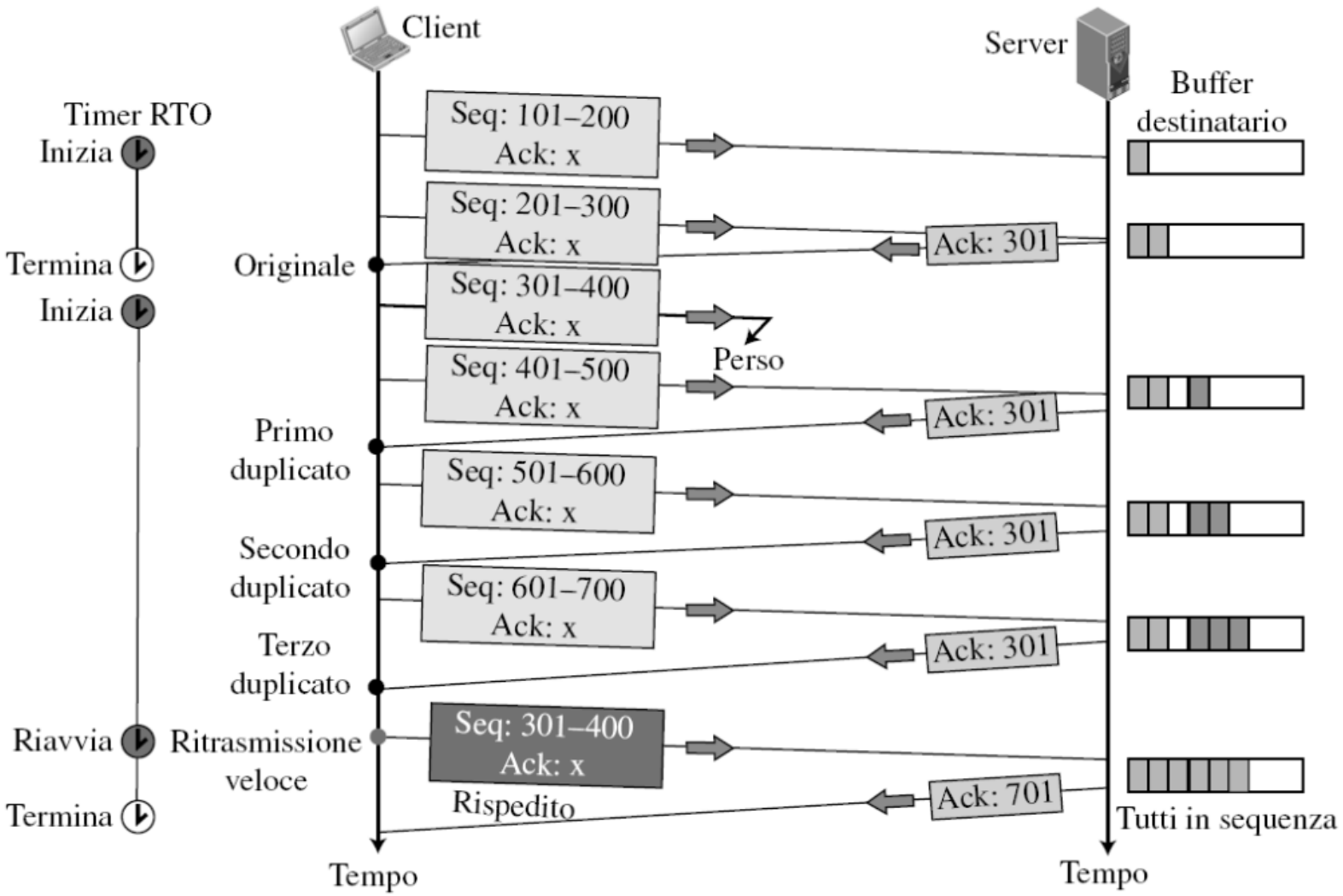
Normale operatività



Segmento smarrito



Ritrasmissione rapida



Ritrasmissione rapida: 3 ack duplicati

Esistono segmenti non riscontrati quindi avvia timer

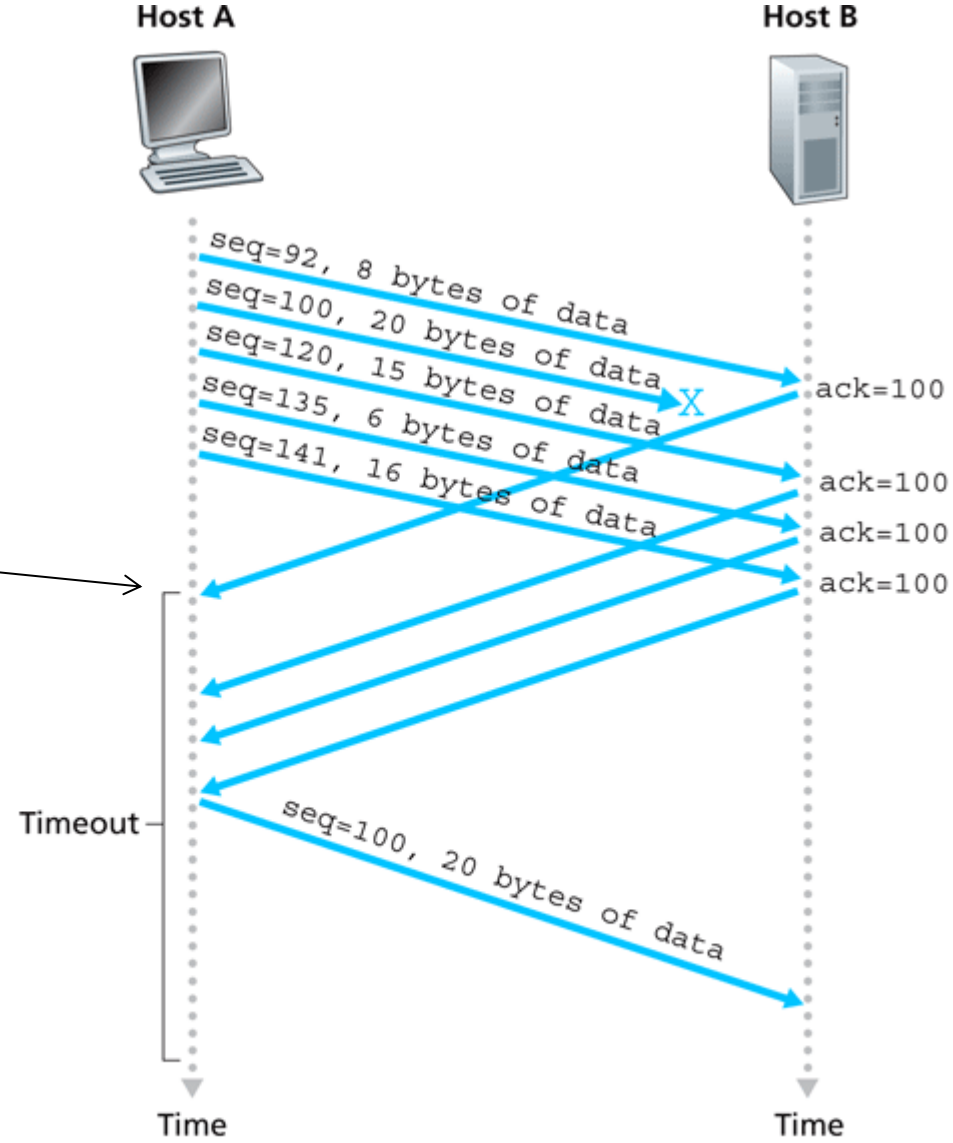
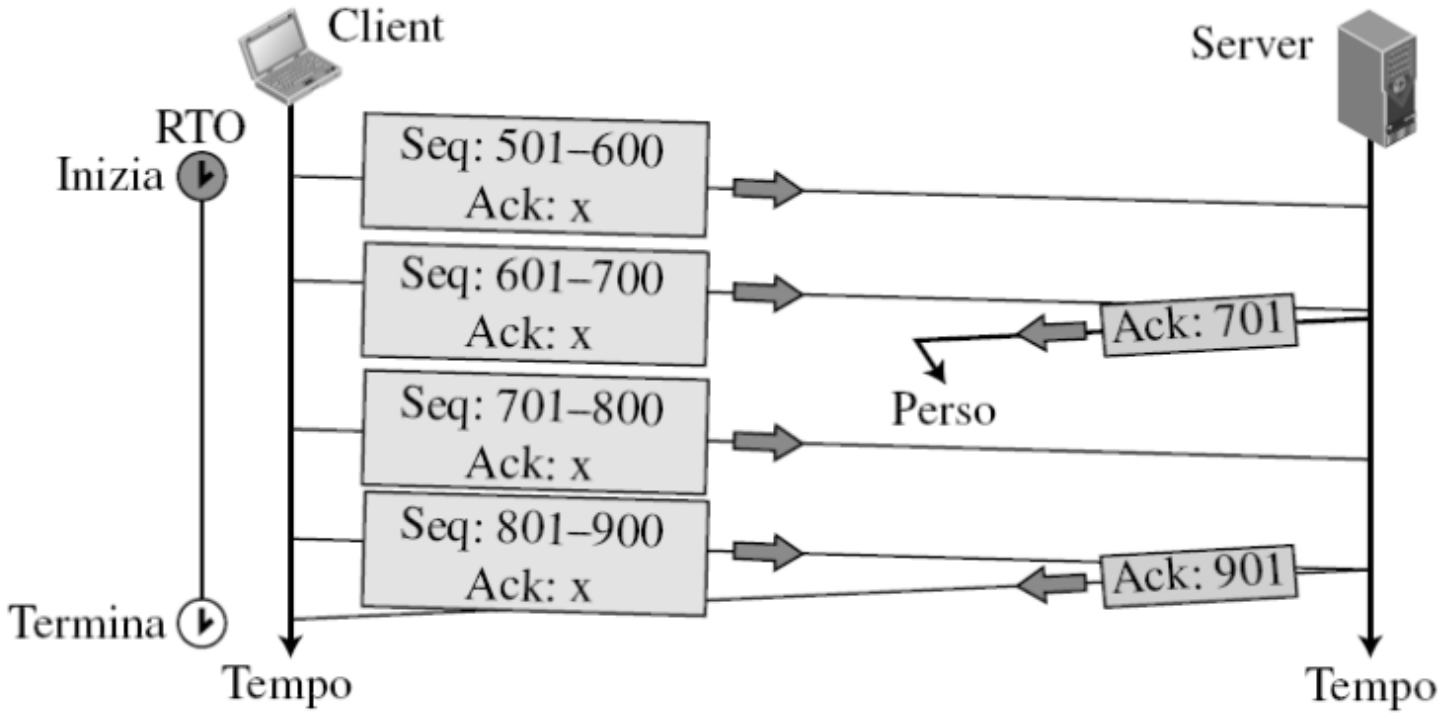
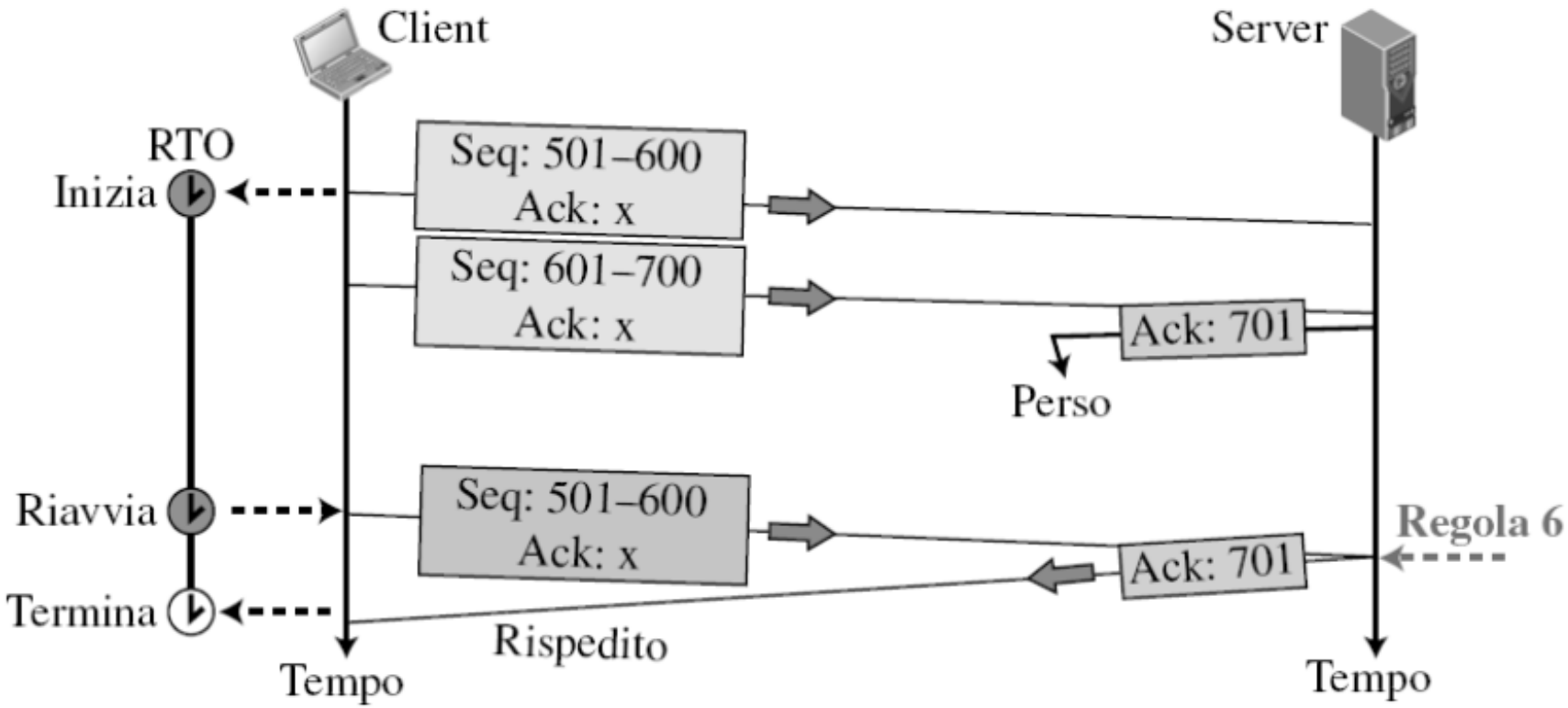


Figure 3.37 ♦ Fast retransmit: retransmitting the missing segment before the segment's timer expires.

Riscontro smarrito senza ritrasmissione



Riscontro smarrito con ritrasmissione



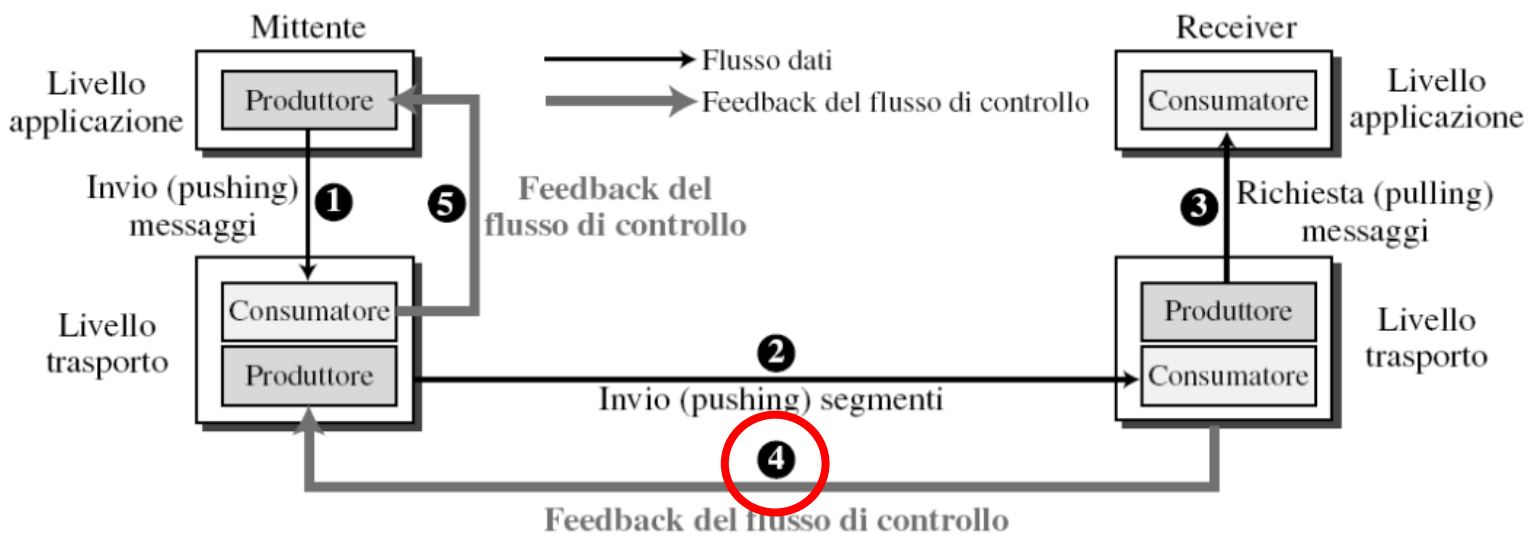
Riassunto su meccanismi adottati da TCP

- **Pipeline** (Approccio ibrido tra GBN e Ripetizione Selettiva)
- **Numero di sequenza**: primo byte nel segmento
- **ACK cumulativo** (conferma tutti i byte precedenti a quello indicato) e **delayed** (posticipato, nel caso di arrivo di un pacchetto in sequenza, con precedenti già riscontrati)
- **Timeout** basato su RTT: **unico** timer di ritrasmissione (associato al più vecchio segmento non riscontrato). Quando arriva una notifica intermedia, si riavvia il timer sul più vecchio segmento non riscontrato
- **Ritrasmissione**
  - **Singola**: solo il segmento **non riscontrato** (non i successivi)
  - **Rapida**: al 3 ACK duplicati prima del timeout → ritrasmissione

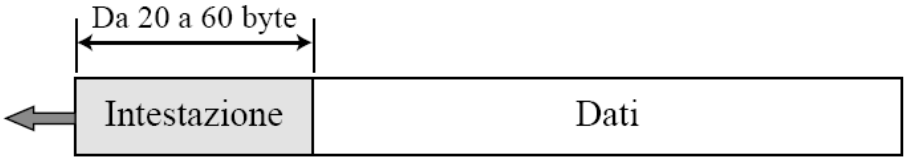
Controllo del flusso

Controllo del flusso

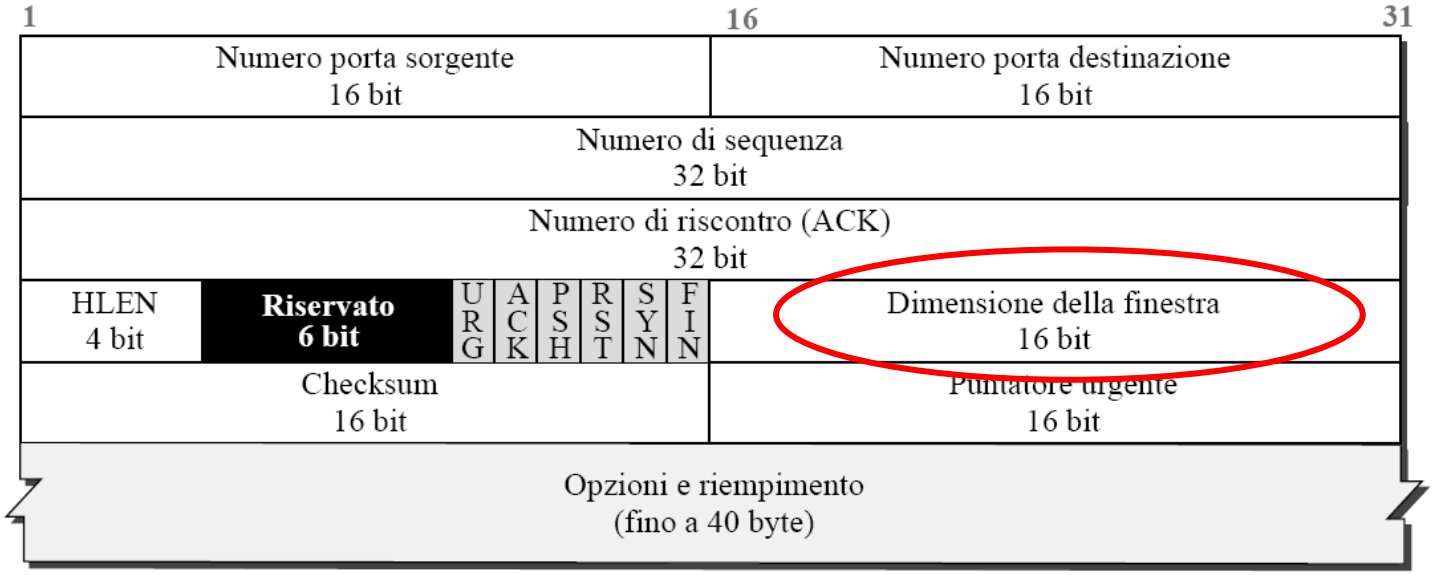
- Obiettivo: Il mittente non vuole sovraccaricare il buffer del destinatario trasmettendo troppi dati, troppo velocemente (bilanciare velocità di invio con velocità di ricezione a livello di processi)
- Feedback esplicito del destinatario: comunica al mittente lo spazio disponibile includendo il valore di RcvWindow (RWND) nei segmenti (header TCP)



Controllo del flusso



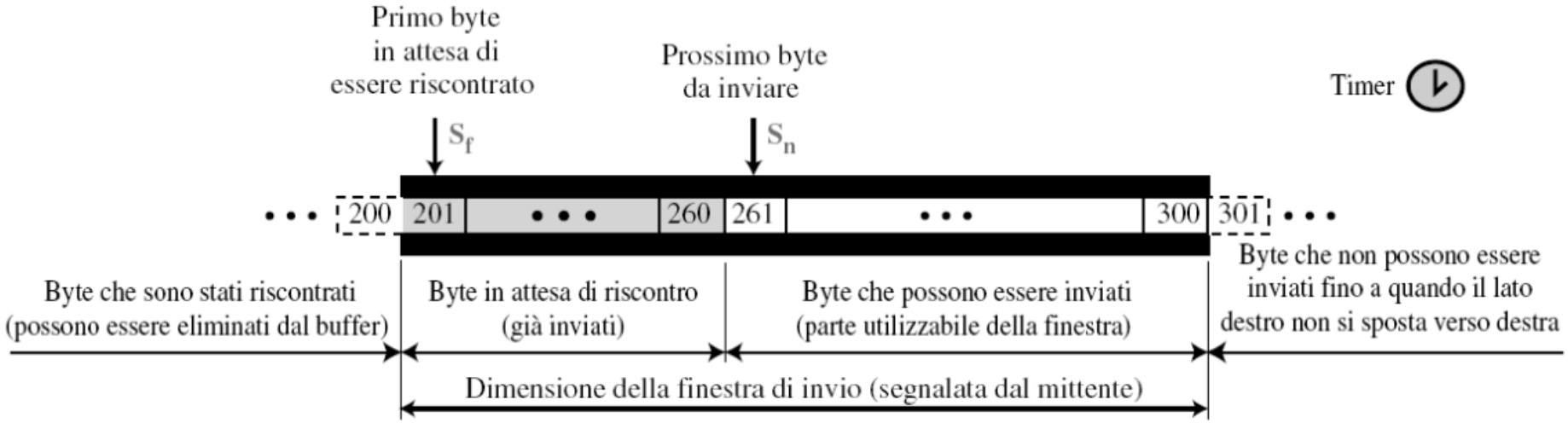
a. Segmento



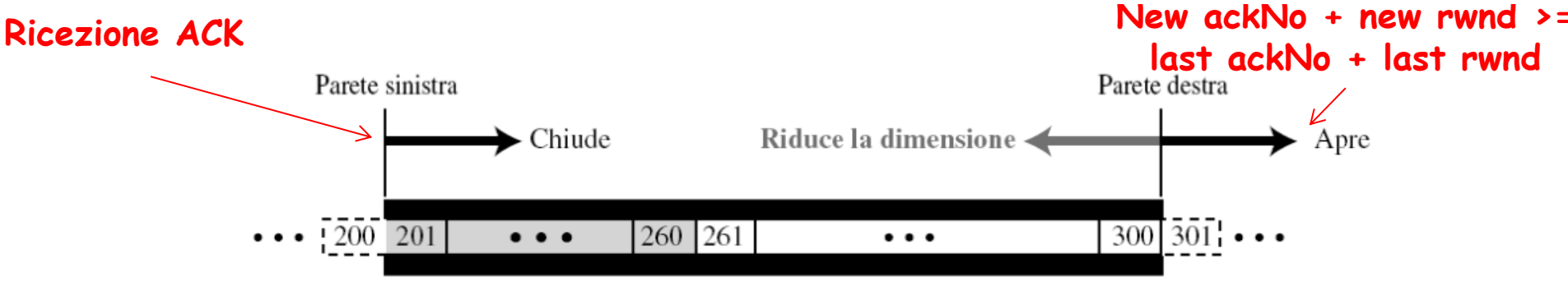
b. Intestazione

Finestra di invio

- L'apertura, chiusura e riduzione della finestra di invio sono controllate dal destinatario

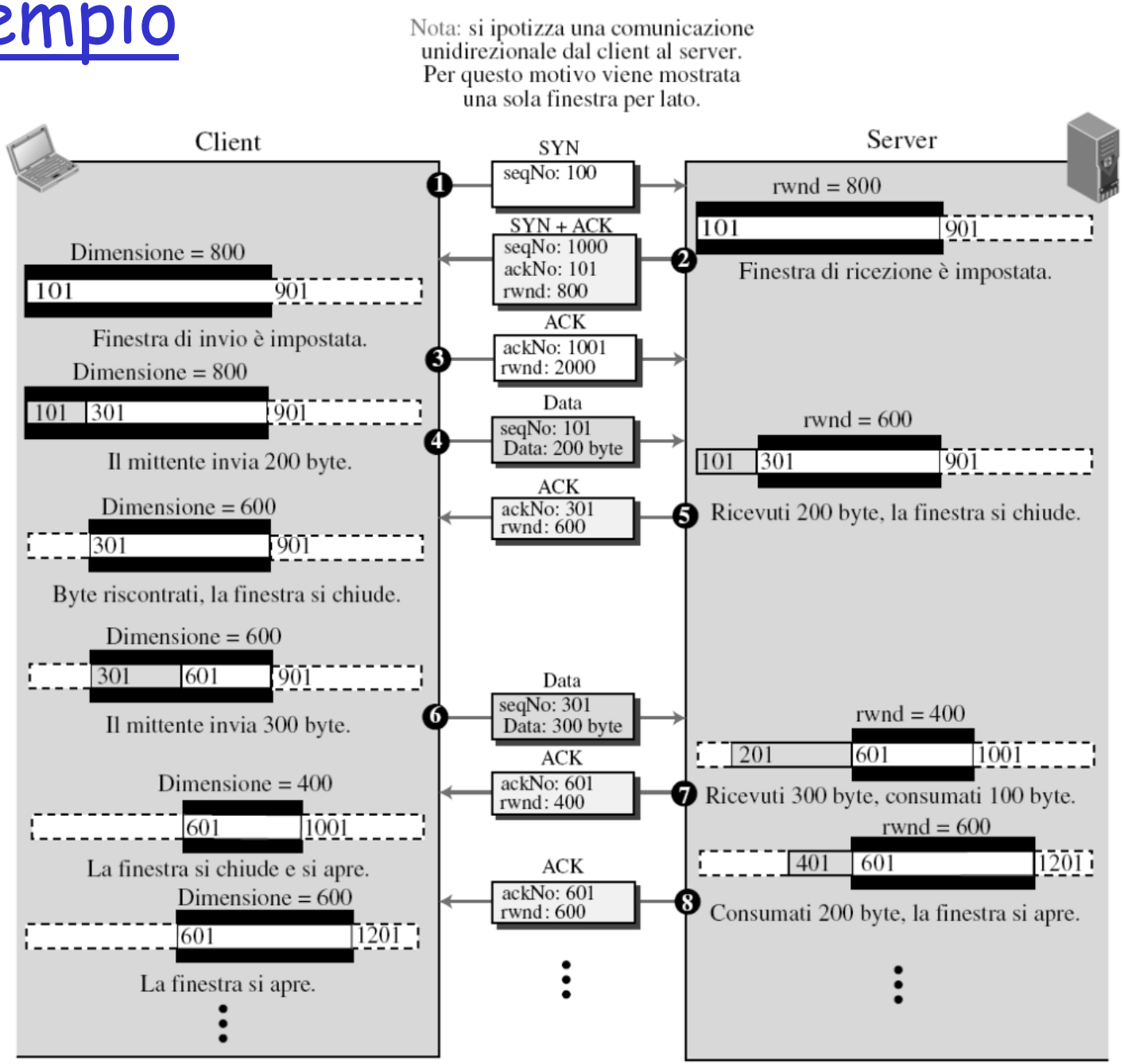


a. Finestra di invio



b. Apertura, chiusura e riduzione della dimensione della finestra di invio.

Esempio


Nota: si ipotizza una comunicazione unidirezionale dal client al server. Per questo motivo viene mostrata una sola finestra per lato.
Client
Server
SYN
seqNo: 100
rwnd = 800
101
901
Finestra di ricezione è impostata.
SYN + ACK
seqNo: 1000
ackNo: 101
rwnd: 800
Dimensione = 800
101
901
Finestra di invio è impostata.
ACK
ackNo: 1001
rwnd: 2000
Dimensione = 800
101
301
901
Il mittente invia 200 byte.
Data
seqNo: 101
Data: 200 byte
rwnd = 600
101
301
901
ACK
ackNo: 301
rwnd: 600
Ricevuti 200 byte, la finestra si chiude.
Dimensione = 600
301
901
Byte riscontrati, la finestra si chiude.
Dimensione = 600
301
601
901
Il mittente invia 300 byte.
Data
seqNo: 301
Data: 300 byte
rwnd = 400
201
601
1001
ACK
ackNo: 601
rwnd: 400
Ricevuti 300 byte, consumati 100 byte.
Dimensione = 400
601
1001
La finestra si chiude e si apre.
rwnd = 600
401
601
1201
ACK
ackNo: 601
rwnd: 600
Consumati 200 byte, la finestra si apre.
Dimensione = 600
601
1201
La finestra si apre.

Controllo della congestione

Controllo della congestione

Congestione:

- informalmente: "troppe sorgenti trasmettono troppi dati, a una velocità talmente elevata che la rete non è in grado di gestirli"
- differisce dal controllo di flusso!
- Sintomi:
  - Pacchetti smarriti (overflow nei buffer dei router)
  - Lunghi ritardi (accodamento nei buffer dei router)
- Tra i dieci problemi più importanti del networking!

Controllo della congestione vs controllo del flusso

- Con il controllo del flusso la dimensione della finestra di invio è controllata dal destinatario tramite il valore rwnd che viene indicato in ogni segmento trasmesso nella direzione opposta

➔La finestra del ricevente non viene mai sovraccaricata con i dati ricevuti

- I buffer intermedi (nei router) possono comunque congestionarsi!
- Un router riceve dati da più mittenti
- Non vi è congestione agli estremi ma vi può essere congestione nei nodi intermedi
- La perdita di segmenti comporta la loro rispedizione, aumentando la congestione
- La congestione è un problema che riguarda IP ma viene gestito da TCP

Controllo della congestione

I due principali approcci al controllo della congestione:

Controllo di congestione end-to-end:

- nessun supporto esplicito dalla rete
- la congestione è dedotta osservando le perdite e i ritardi nei sistemi terminali
- metodo adottato da TCP

Controllo di congestione assistito dalla rete:

- i router forniscono un feedback ai sistemi terminali
  - un singolo bit per indicare la congestione (TCP/IP ECN)
  - comunicare in modo esplicito al mittente la frequenza trasmissiva

Problematiche

1. Come può il mittente limitare la frequenza di invio del traffico sulla propria connessione?
2. Come può il mittente rilevare la congestione?
3. Quale algoritmo dovrebbe essere usato per limitare la frequenza di invio in funzione della congestione end-to-end?

1. Finestra di congestione

- Per controllare la congestione si usa la variabile CWND (congestion window) che insieme a RWND definisce la dimensione della finestra di invio
- CWND: relativa alla congestione della rete
- RWND: relativa alla congestione del ricevente (flusso mittente-destinatario)

Dimensione della finestra = min(rwnd, cwnd)

2. Rilevare la congestione

- Evento di perdita
  - **ACK duplicati** e **timeout** possono essere intesi come eventi di perdita (danno indicazione dello stato della rete)
- Reazione in base allo stato della rete
  - Se ACK arrivano in sequenza e con buona frequenza → si può inviare e incrementare la quantità di segmenti inviati
  - Se ACK duplicati o timeout → è necessario ridurre la finestra dei pacchetti che si spediscono senza aver ricevuto riscontri
- TCP è auto-temporizzante: reagisce in base ai riscontri che ottiene

3. Controllo della congestione

- Idea di base: incrementare il rate di trasmissione se non c'è congestione (ack), diminuire se c'è congestione (segmenti persi)

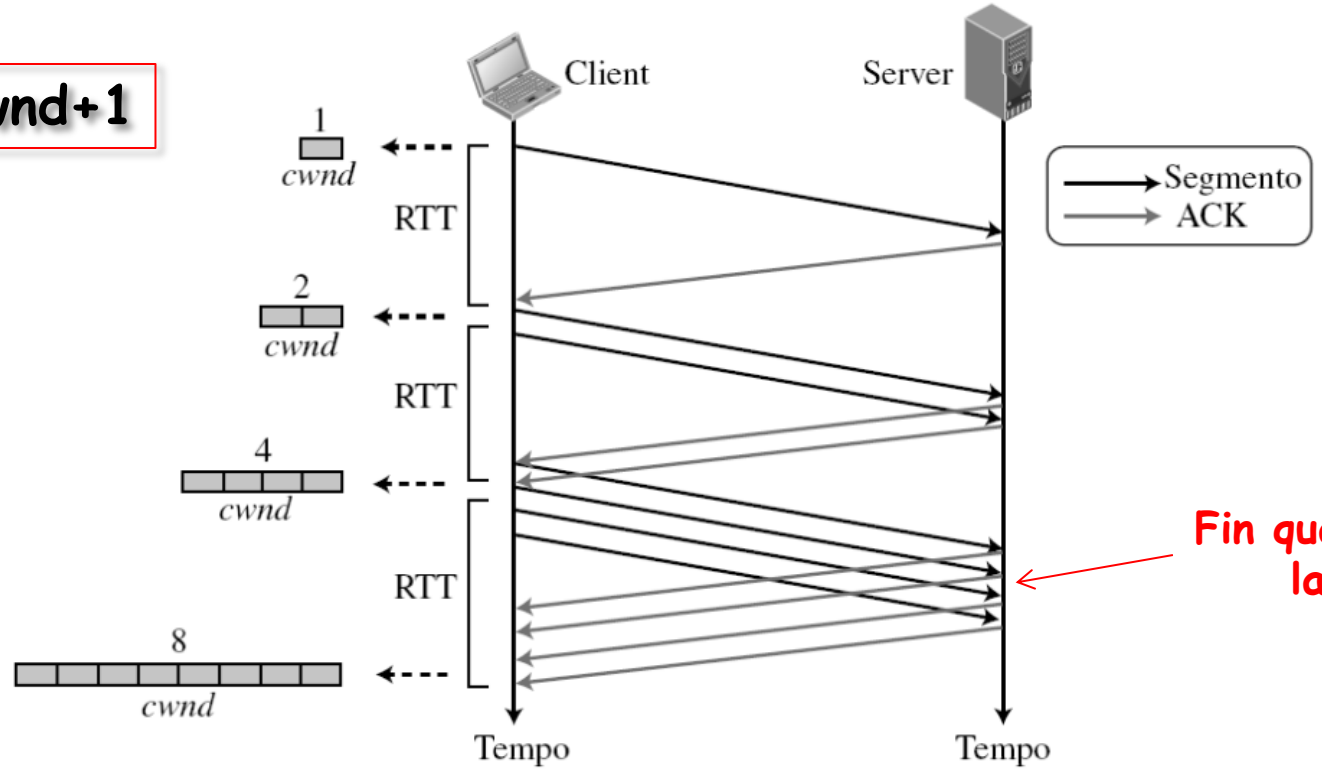
L'algoritmo di controllo della congestione si basa su tre componenti

1. Slow start
2. Congestion avoidance
3. Fast recovery

Slow start: incremento esponenziale

- Cwnd inizializzata a 1 MSS
- Poiché la banda disponibile può essere molto maggiore, slow start incrementa di 1MSS la cwnd per ogni segmento riscontrato

cwnd=cwnd+1



Fin quando cresce la cwnd?

Slow start: incremento esponenziale

Se arriva un riscontro, *cwnd* = *cwnd* + 1

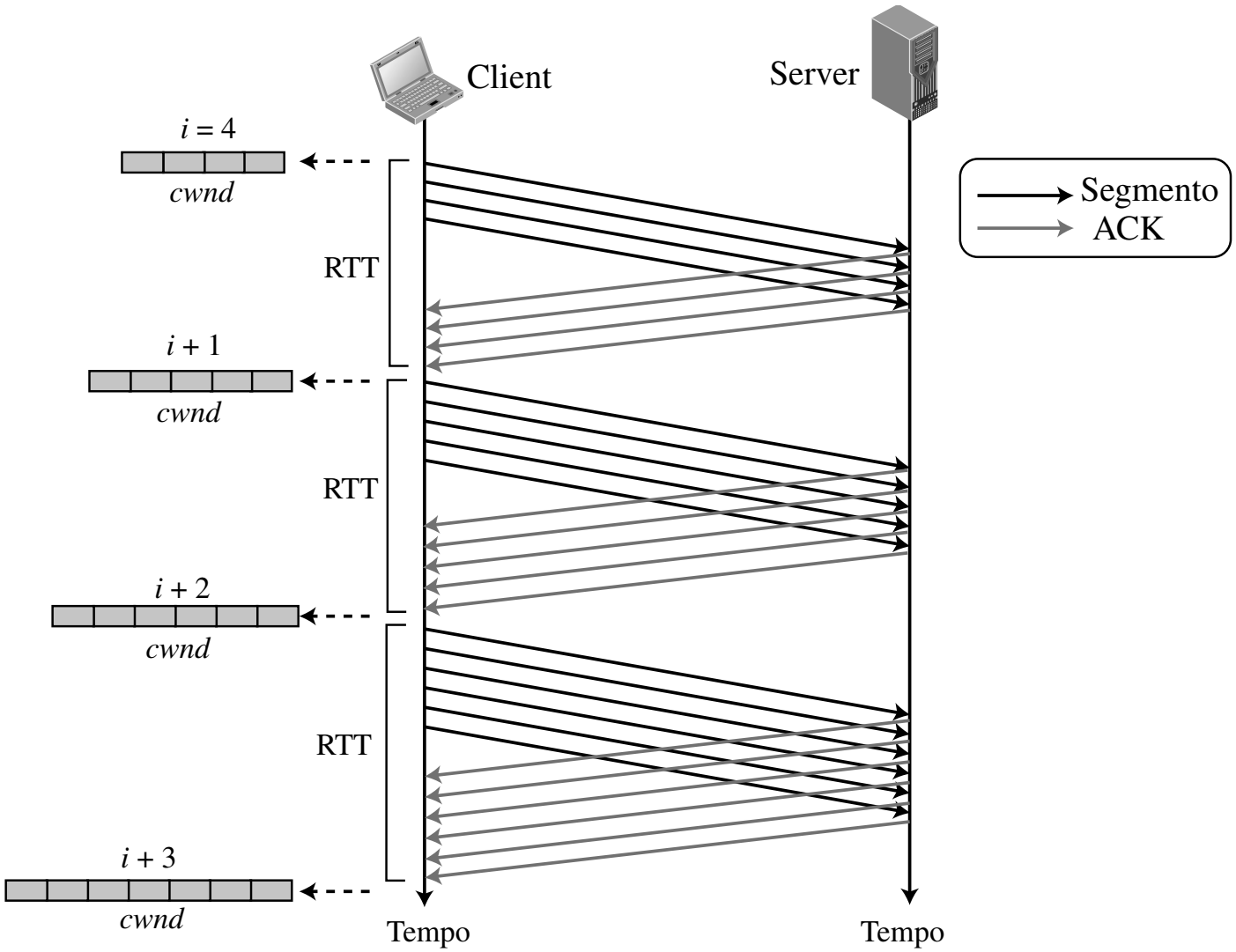
Inizio	$\rightarrow$	$cwnd = 1 \rightarrow 2^0$
Dopo 1 RTT	$\rightarrow$	$cwnd = cwnd + 1 = 1 + 1 = 2 \rightarrow 2^1$
Dopo 2 RTT	$\rightarrow$	$cwnd = cwnd + 2 = 2 + 2 = 4 \rightarrow 2^2$
Dopo 3 RTT	$\rightarrow$	$cwnd = cwnd + 4 = 4 + 4 = 8 \rightarrow 2^3$

La dimensione della finestra di congestione nell'algoritmo slow start viene aumentata esponenzialmente fino al raggiungimento di una soglia (ssthresh).

Congestion avoidance

- La cwnd cresce finchè non viene perso un pacchetto (in tal caso si pone ssthreshold = cwnd/2) o finchè non raggiunge la ssthresh (slow start threshold)
- Si arresta slow start e inizia congestion avoidance:
  - Incremento lineare (ogni volta che viene riscontrata l'intera finestra di segmenti, si incrementa di 1 la cwnd)
  - Congestion avoidance incrementa linearmente finchè non si rileva congestione (timeout o 3 ack duplicati)
  - Al timeout ssthreshold=cwnd/2 and cwnd=1

Congestion avoidance



Congestion avoidance: additive increase

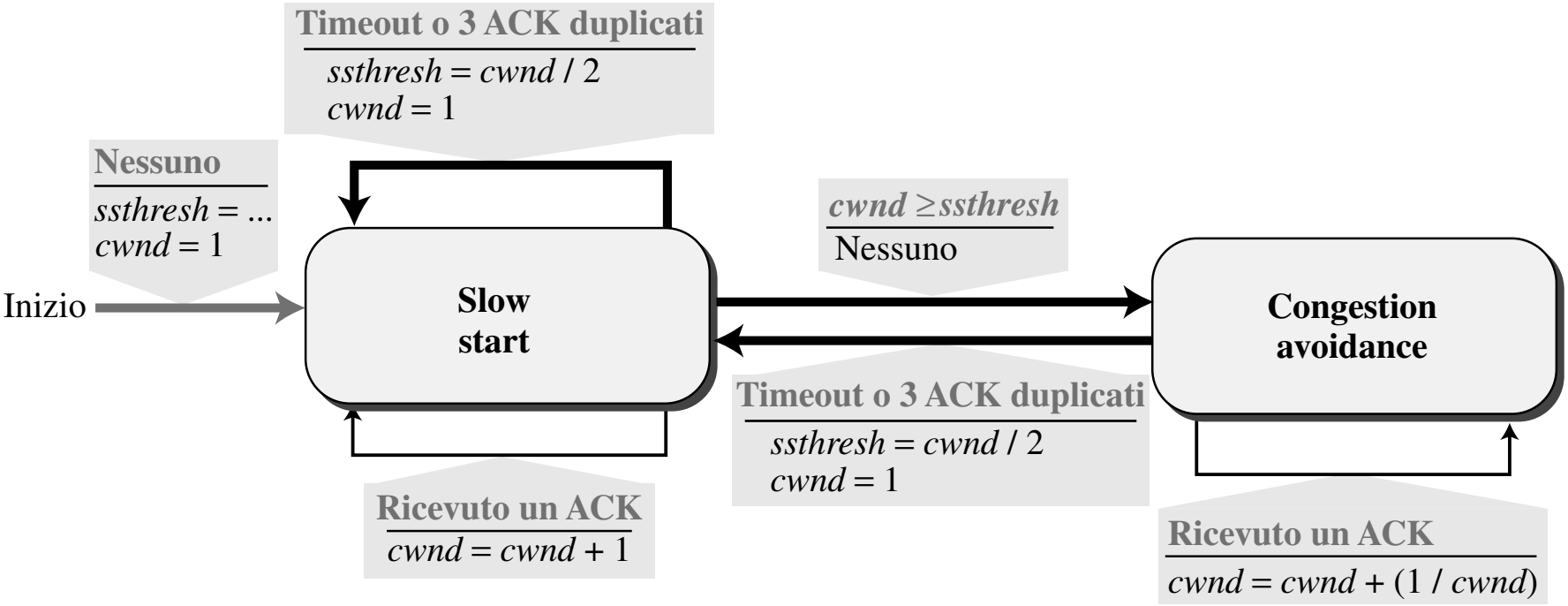
Se arriva un riscontro, *cwnd* = *cwnd* + (1 / *cwnd*)

Inizio	$\rightarrow$	$cwnd = i$
Dopo 1 RTT	$\rightarrow$	$cwnd = i + 1$
Dopo 2 RTT	$\rightarrow$	$cwnd = i + 2$
Dopo 3 RTT	$\rightarrow$	$cwnd = i + 3$

Con l'algoritmo congestion avoidance, la dimensione della finestra di congestione viene aumentata linearmente fino alla rilevazione della congestione.

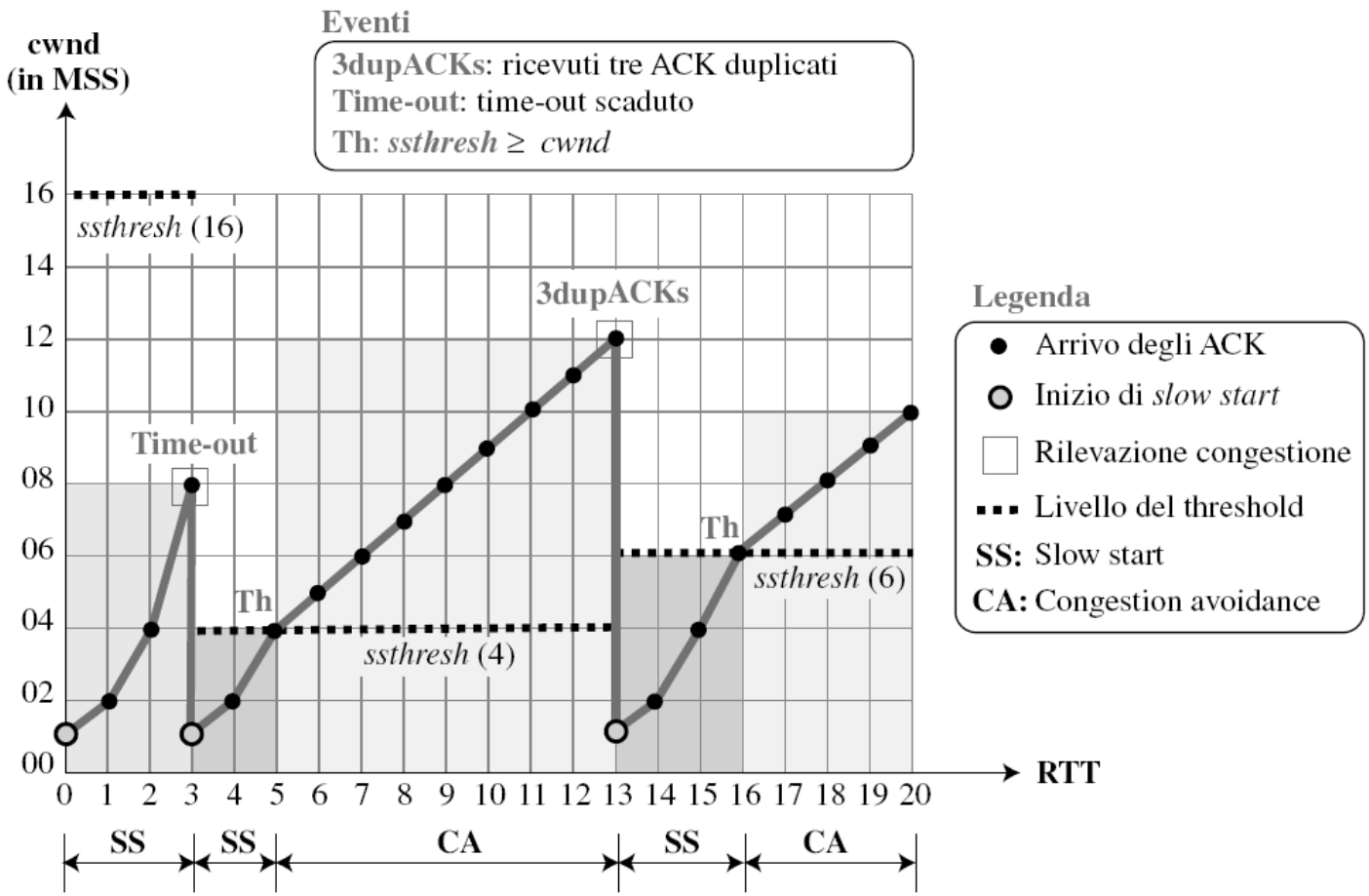
Versioni TCP

FSM TCP Taho


Timeout o 3 ACK duplicati
ssthresh = cwnd / 2
cwnd = 1
Nessuno
ssthresh = ...
cwnd = 1
Inizio
Slow start
cwnd ≥ssthresh
Nessuno
Congestion avoidance
Timeout o 3 ACK duplicati
ssthresh = cwnd / 2
cwnd = 1
Ricevuto un ACK
cwnd = cwnd + 1
Ricevuto un ACK
cwnd = cwnd + (1 / cwnd)

TCP Taho

- TCP Tahoe: considera timeout e 3 ack duplicati come congestione e riparte da 1 con ssthresh=cwnd/2



Affinamento

Filosofia:

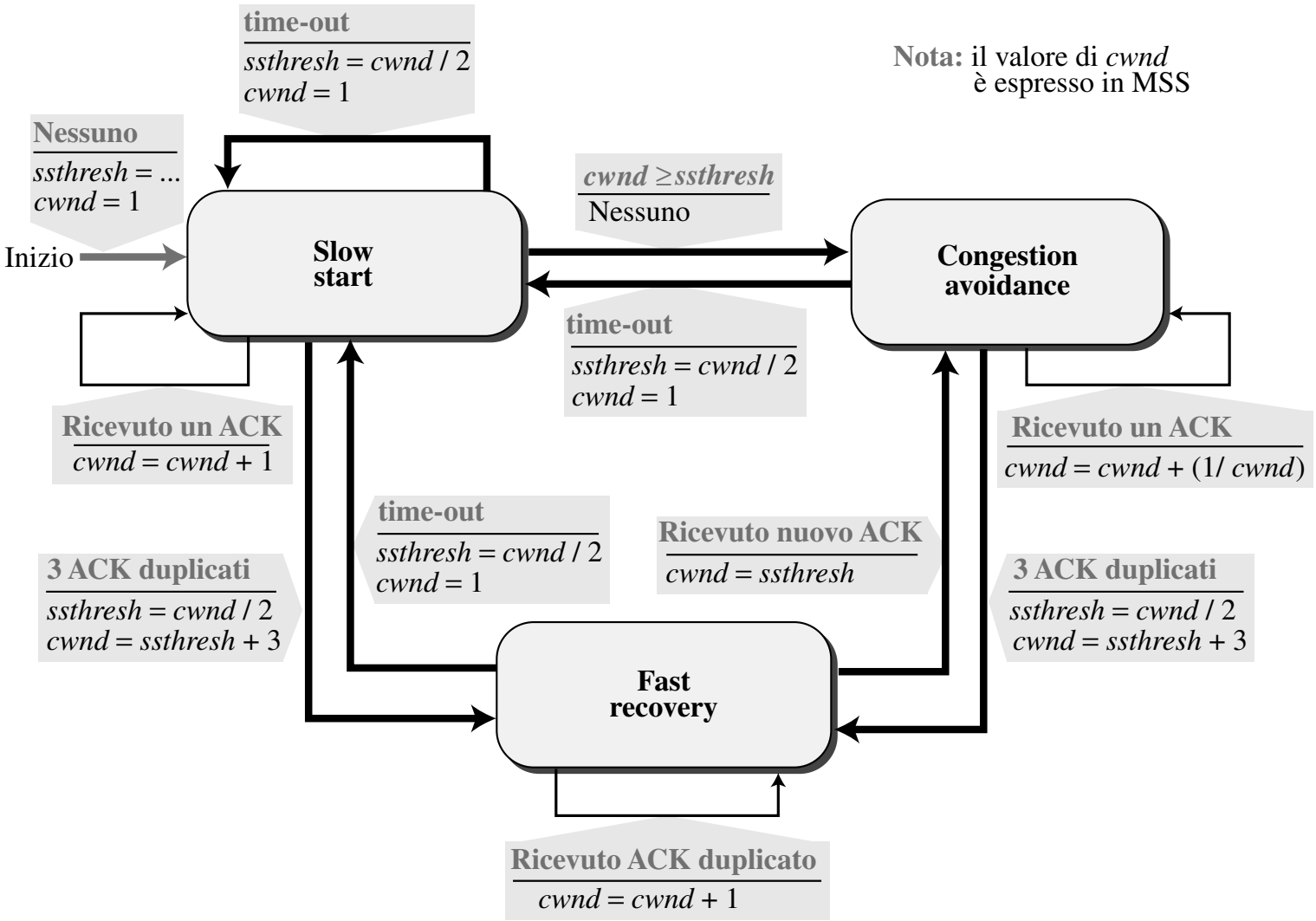
- 3 ACK duplicati indicano la capacità della rete di consegnare qualche segmento (3 pacchetti oltre quello perso sono arrivati)
- un timeout prima di 3 ACK duplicati è "più allarmante" (non sono arrivati nemmeno i pacchetti seguenti)

- Si può quindi distinguere i due tipi di congestione e reagire in maniera più appropriata e meno drastica nel caso dei 3 ack duplicati
- Fast recovery: incrementa linearmente perché indica congestione leggera

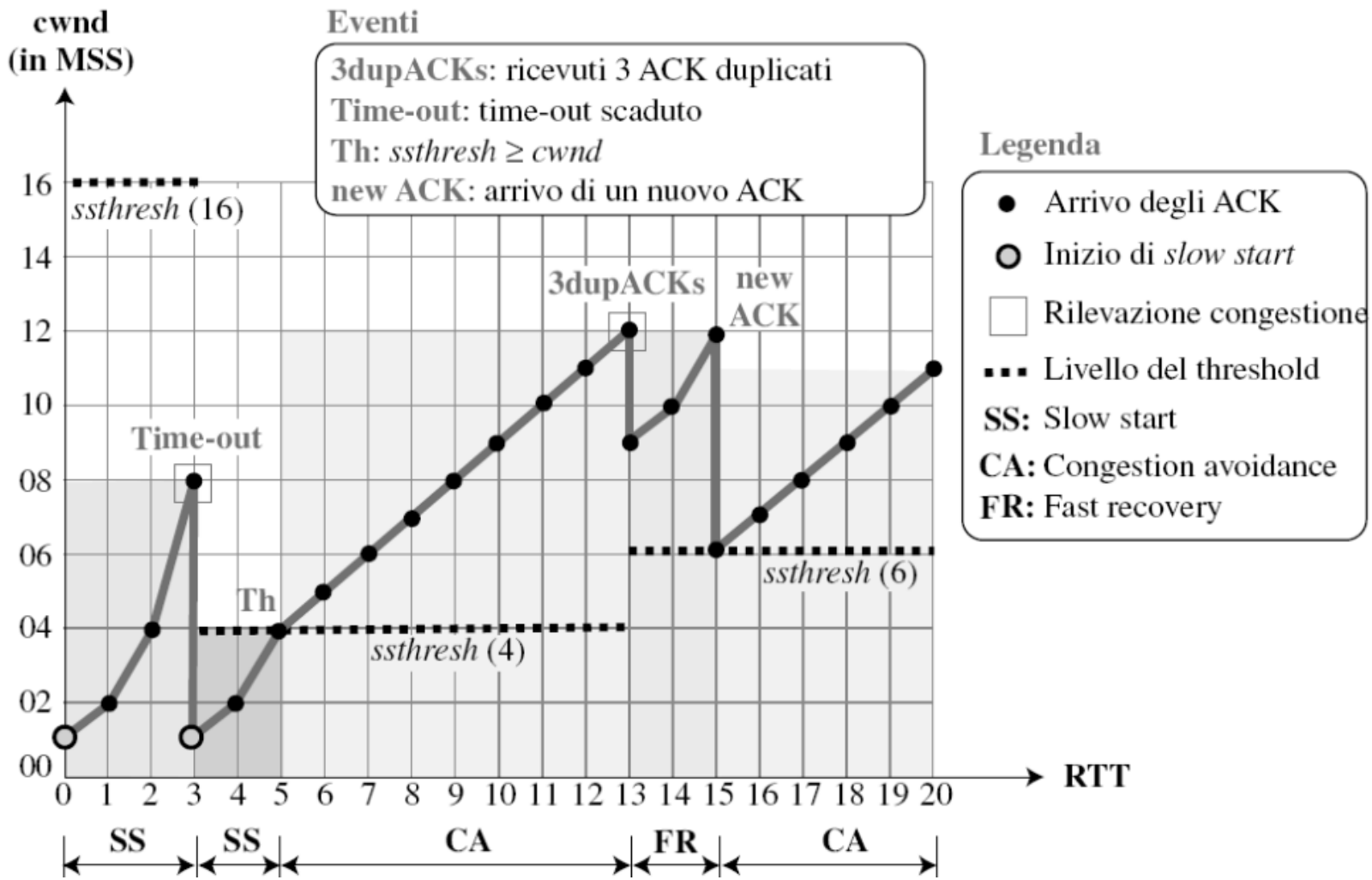
TCP Reno

- Timeout: congestione importante
  - riparte da 1
- 3 ack duplicati: congestione lieve
  - applica fast recovery a partire da ssthreshold + 3

TCP Reno


time-out
ssthresh = cwnd / 2
cwnd = 1
Nota: il valore di cwnd è espresso in MSS
Nessuno
ssthresh = ...
cwnd = 1
Inizio
Slow start
cwnd ≥ssthresh
Nessuno
Congestion avoidance
time-out
ssthresh = cwnd / 2
cwnd = 1
Ricevuto un ACK
cwnd = cwnd + 1
Ricevuto un ACK
cwnd = cwnd + (1/ cwnd)
time-out
ssthresh = cwnd / 2
cwnd = 1
Ricevuto nuovo ACK
cwnd = ssthresh
3 ACK duplicati
ssthresh = cwnd / 2
cwnd = ssthresh + 3
3 ACK duplicati
ssthresh = cwnd / 2
cwnd = ssthresh + 3
Fast recovery
Ricevuto ACK duplicato
cwnd = cwnd + 1

TCP Reno


cwnd
(in MSS)
Eventi
3dupACKs: ricevuti 3 ACK duplicati
Time-out: time-out scaduto
Th: ssthresh ≥ cwnd
new ACK: arrivo di un nuovo ACK
Legenda
Arrivo degli ACK
Inizio di slow start
Rilevazione congestione
Livello del threshold
SS: Slow start
CA: Congestion avoidance
FR: Fast recovery
ssthresh (16)
3dupACKs
new ACK
Time-out
Th
ssthresh (4)
ssthresh (6)
16
14
12
10
08
06
04
02
00
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
RTT
SS
SS
CA
FR
CA

TCP: tempo di andata e ritorno e timeout

D: come impostare il valore del timeout di TCP?

- Più grande del tempo di andata e ritorno della connessione (RTT)
  - ma RTT varia
- Troppo piccolo: timeout prematuro
  - ritrasmissioni non necessarie
- Troppo grande: reazione lenta alla perdita dei segmenti

D: come stimare RTT?

- **SampleRTT**: tempo misurato dalla trasmissione del segmento fino alla ricezione di ACK
  - ignora le ritrasmissioni
  - Un solo **SampleRTT** per più segmenti trasmessi insieme
- **SampleRTT** varia a causa di congestione nei router e carico nei sistemi terminali, quindi occorre una stima "più livellata" di RTT
  - media di più misure recenti, non semplicemente il valore corrente di **SampleRTT**

TCP: tempo di andata e ritorno e timeout

$$\mathbf{EstimatedRTT_{t+1} = (1 - \alpha)*EstimatedRTT_t + \alpha*SampleRTT_{t+1}}$$

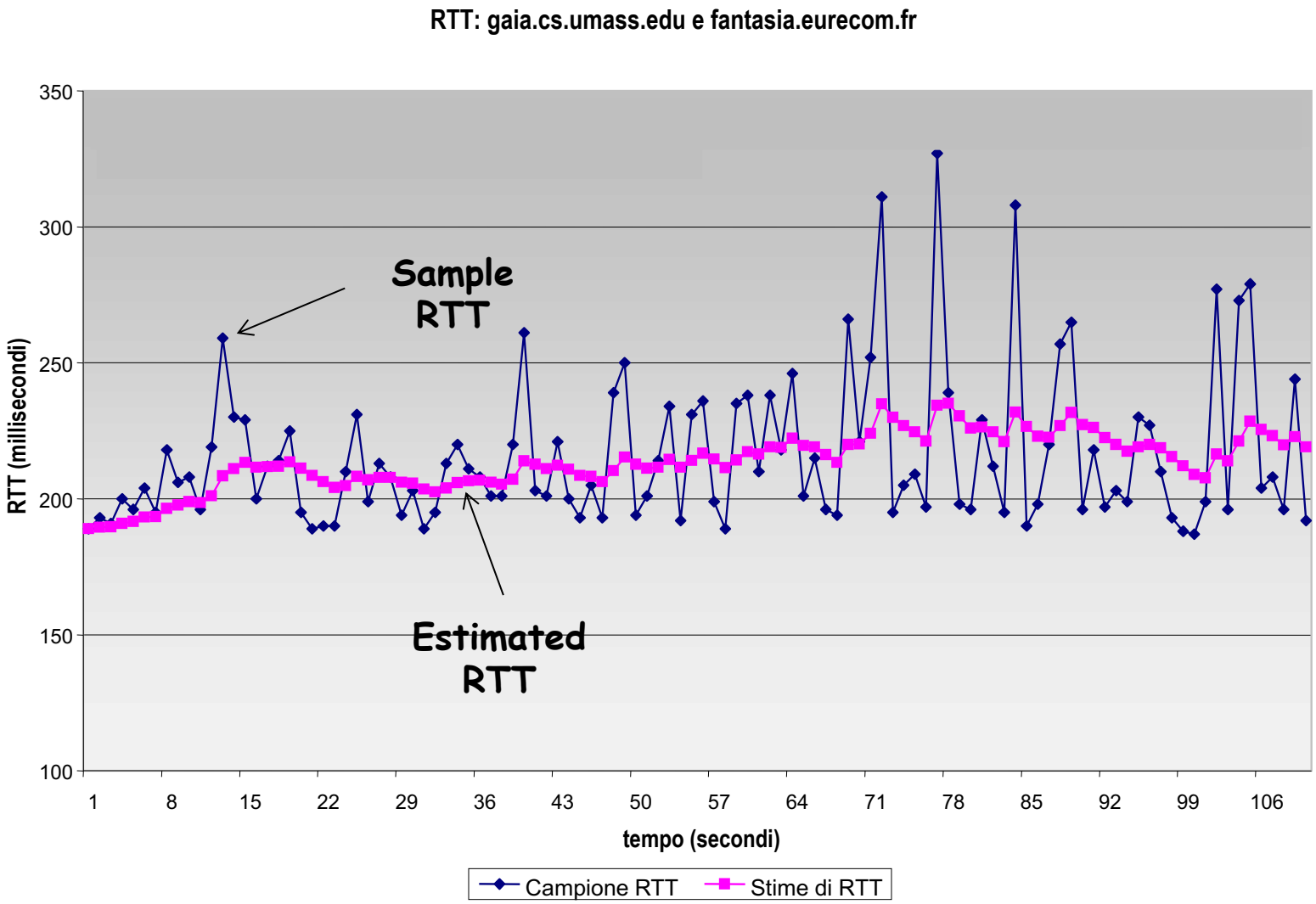
- Media mobile esponenziale ponderata
- L'influenza delle misure passate decresce esponenzialmente
- Valore tipico: $\alpha = 0{,}125$

ovvero

$$\mathbf{EstimatedRTT_{t+1} = 0.875*EstimatedRTT_t + 0.125*SampleRTT_{t+1}}$$

con tale valore ($\alpha = 0{,}125$) si assegna minore peso alle misure recenti che a quelle più vecchie

Esempio di stima di RTT:



TCP: tempo di andata e ritorno e timeout

Impostazione del timeout

- **EstimatedRTT** più un "margine di sicurezza"
  - grande variazione di **EstimatedRTT** -> margine di sicurezza maggiore
- Stimare innanzitutto di quanto SampleRTT si discosta da EstimatedRTT:

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

(tipicamente, β = 0,25)

Poi impostare l'intervallo di timeout:

- Si imposta un valore iniziale pari a 1 secondo.
- Se avviene un timeout si raddoppia
- Appena viene ricevuto un segmento e aggiornato **EstimatedRTT**, si usa la formula

$$\mathbf{TimeoutInterval = EstimatedRTT + 4*DevRTT}$$