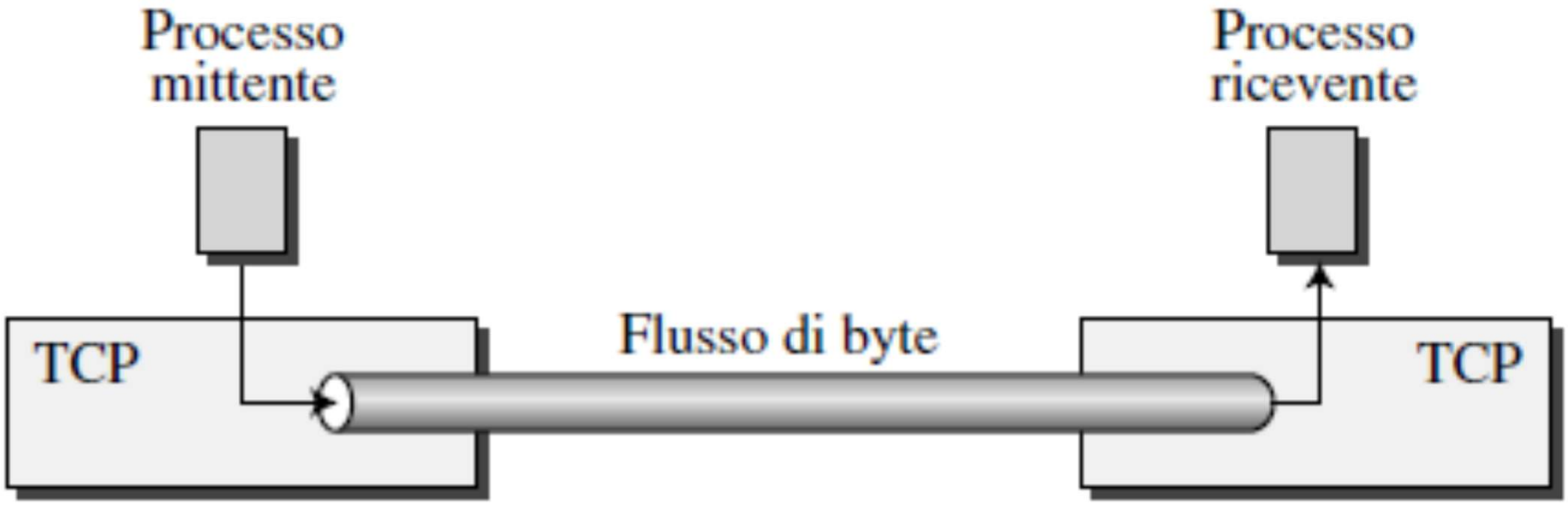
Lo strato di Trasporto
TCP

Corso di
Reti di Calcolatori
AA. 2023-2024

Federica Paganelli

Proprietà del servizio TCP

- Orientamento allo stream
  - Lunghezza di byte indefinita a priori. Il servizio di consegna dello stream nella macchina di destinazione passa esattamente la medesima sequenza di ottetti che il trasmettitore ha passato al servizio di consegna nella macchina origine
  - **TCP vede i dati come un flusso di byte ordinati, ma non strutturati**



Proprietà del servizio TCP

- Orientato alla connessione:
  - I processi effettuano un handshake prima dello scambio dei dati. Invio di informazioni preliminari per preparare lo scambio dei dati
  - **Orientato** perché lo stato della connessione risiede sui punti terminali, non sugli elementi intermedi della rete (ad es. router).
  - La connessione è vista dagli applicativi (USERS) come un circuito dedicato, quindi il TCP è capace di fornire servizi del tipo CONNECTION ORIENTED mentre il protocollo IP su cui appoggia, è in grado di fornire servizi CONNECTION LESS.
- Connessione full-duplex
  - il flusso dati tra due host può avvenire contemporaneamente nelle due direzioni. Le due direzioni sono slegate
  - connessione punto-punto

Proprietà del servizio TCP

Funzioni base per il trasferimento di dati

- capacità di trasferire un flusso continuo di byte
- trasferimento bidirezionale (full duplex)

Multiplexing/demultiplexing

- consente di assegnare una data ***connessione*** ad un particolare processo
  (permette una comunicazione da processo a processo)

Controllo della connessione

- meccanismi di inizio e fine trasmissione
  (controllo di sessione)

Proprietà del servizio TCP

Trasferimento dati ordinato e affidabile

- si intende la capacità di correggere tutti i tipi di errore, quali:
  - dati corrotti
  - segmenti persi
  - segmenti duplicati
  - segmenti fuori sequenza

Controllo di flusso

- evitare di spedire più dati di quanti il ricevitore sia in grado di trattare

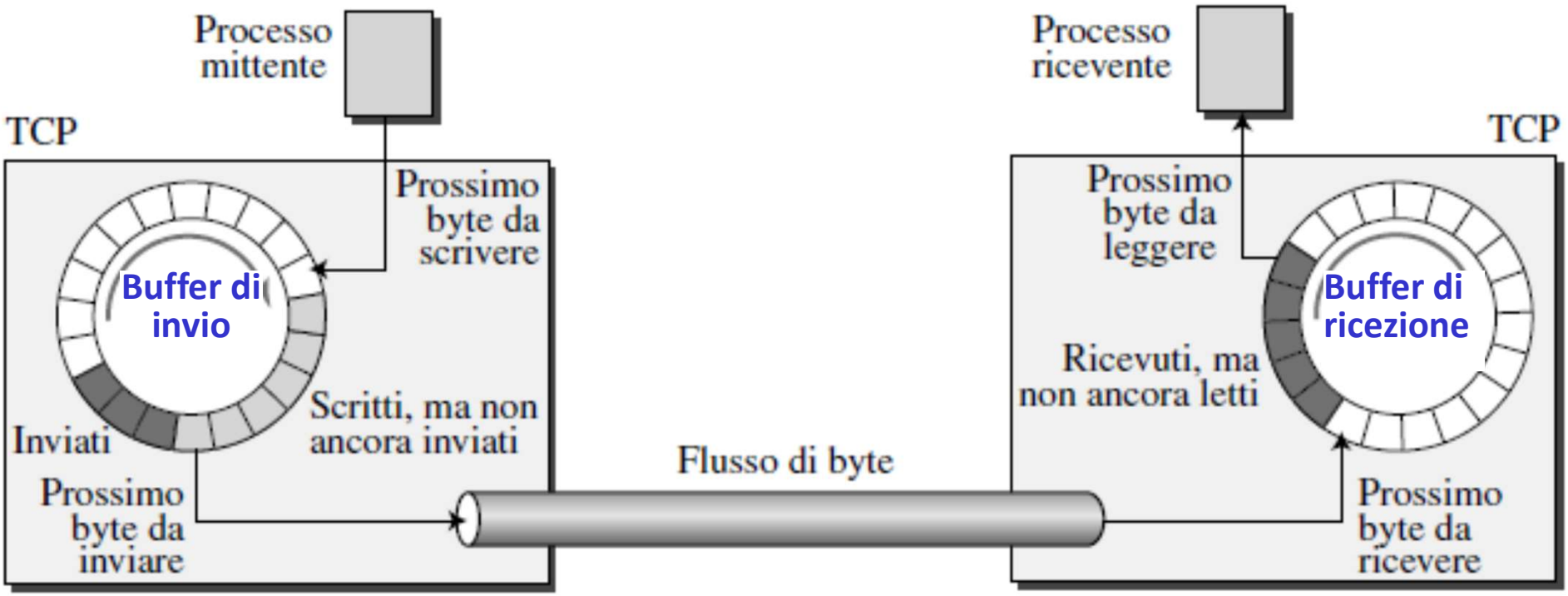
Controllo di congestione

- ha lo scopo di recuperare situazioni di sovraccarico nella rete

Proprietà del servizio TCP

- Trasferimento bufferizzato
  - Il software del protocollo TCP è libero di suddividere il flusso di byte in **segmenti in modo indipendente dal programma applicativo** che li ha generati. Per fare questo è necessario disporre di un BUFFER dove immagazzinare la sequenza di byte. Appena i dati sono sufficienti per riempire un segmento ragionevolmente grande, questo viene trasmesso attraverso la rete.
  - La bufferizzazione consente una riduzione del traffico sulla rete "ottimizzando" in qualche modo il numero di segmenti da trasmettere

Trasferimento bufferizzato

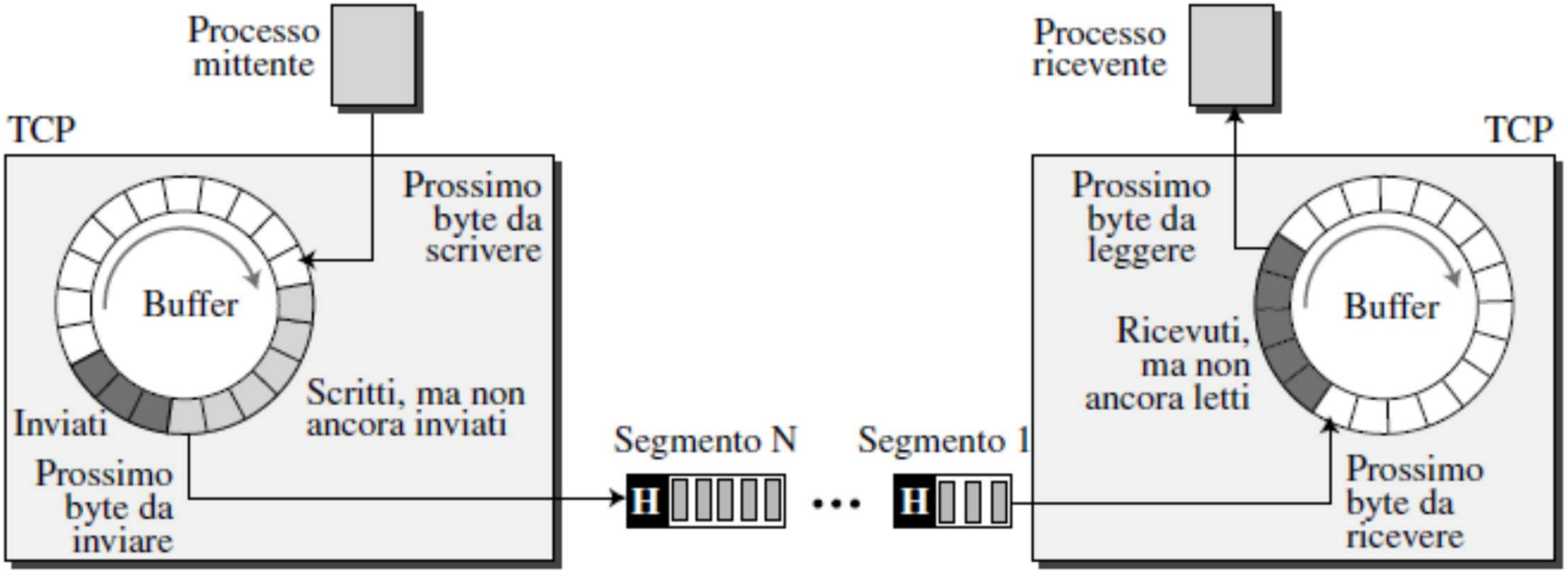


Lato mittente

Lato destinatario

I processi a livello applicativo scrivono e leggono byte nel/dal buffer.
Questo può avvenire a velocità diverse
PS. Connessione TCP è bidirezionale, entrambi i lati avranno buffer di invio e buffer di ricezione

Segmenti TCP



Il flusso di byte viene partizionato in segmenti

- Ogni segmento ha il suo header
- Ogni segmento viene consegnato al livello IP

TCP

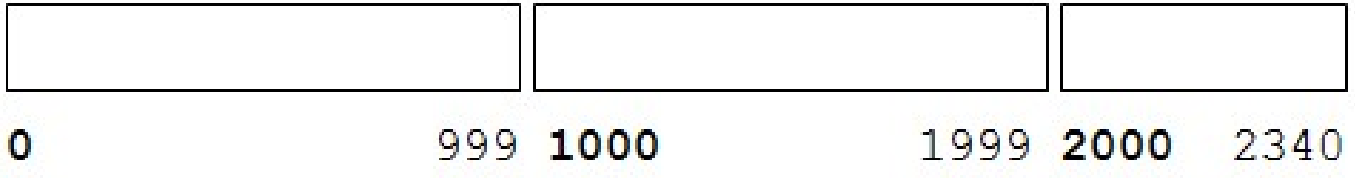
formato segmento

apertura e chiusura della connessione

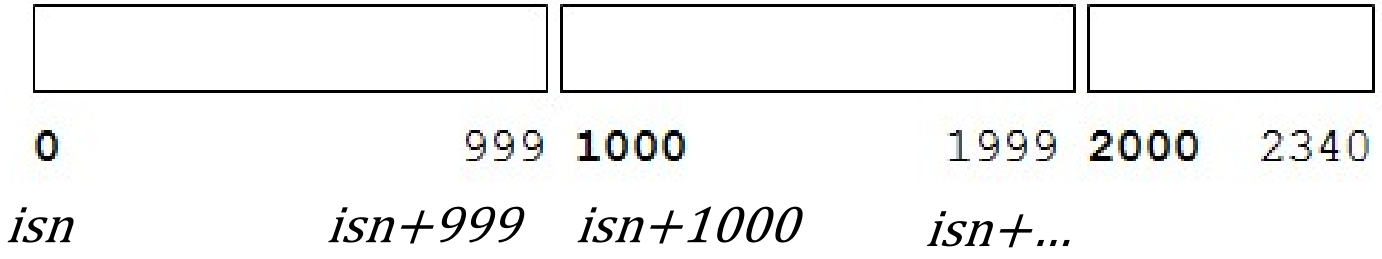
Numeri di sequenza e di riscontro

- TCP numera i byte (anziché i segmenti)
- **Numero di sequenza** associato a un segmento =

  numero (nel flusso) del primo byte (di dati) del segmento



- In genere si parte da un initial sequence number generato in modo casuale (e quindi ≠ 0)

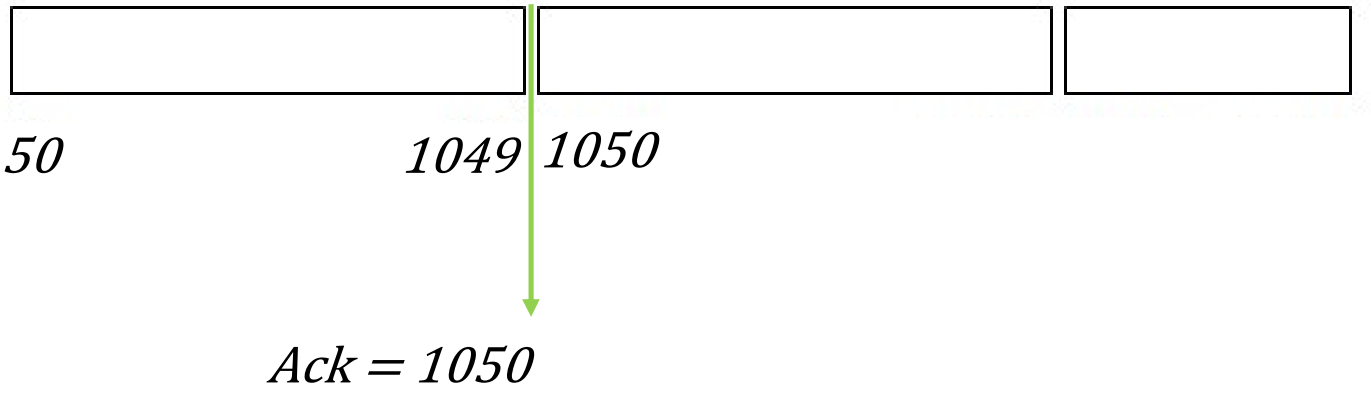


Numeri di sequenza e di riscontro

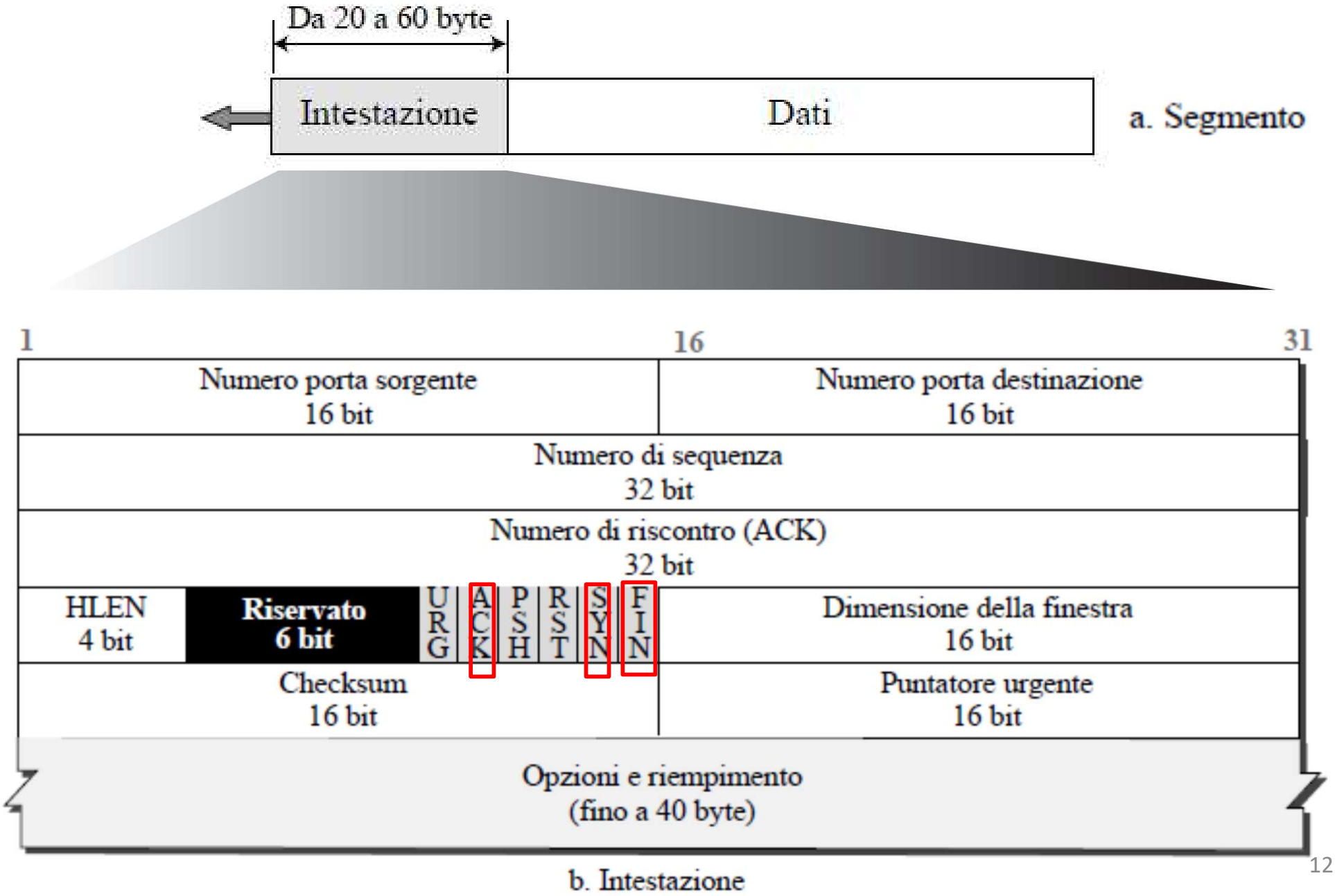
- **Numero di riscontro** =

  numero ultimo byte correttamente ricevuto +1
- Riscontri interpretati come “cumulativi”

  “ACK=y” significa “aspetto il byte y (ho ricevuto tutti i byte fino a y-1 incluso)”



Formato segmenti TCP



Segmento TCP - note

- **Numero di sequenza, Numero di riscontro, Finestra**
  - Essi permettono il flow control, il meccanismo di ritrasmissione ed il riordino dei pacchetti in ricezione, necessari per la struttura stream-based del TCP.
- **Porta (16 bit):** numeri di porta della comunicazione.
- **Numero di sequenza (32bit):** è il numero di sequenza nello stream del primo byte di dati di questo segmento. Se il flag SYN è settato il numero di sequenza è ISN (initial sequence number) e il primo byte di dati è ISN+1.
- **Numero di riscontro (32bit):** se il bit ACK è settato, questo campo contiene il valore del prossimo numero di sequenza che il mittente del segmento si aspetta di ricevere dall'altro host. Una volta che la connessione è stabilita è sempre inviato.
- **Hlen (4bit):** lunghezza dell'header TCP espressa in parole da 4 byte (la lunghezza dell'header può variare tra 20 e 60 byte)

Segmento TCP - campi

- **Bit codice:** sono 6 *flag* e servono per (da sinistra a destra):
  - URG: Il campo *Puntatore Urgente* è significativo e ci sono dati da trasferire in via prioritaria
  - **ACK: Il campo *Numero di Riscontro* contiene dati significativi**
  - PSH: Funzione Push (trasferimento immediato dei dati in un segmento dal trasporto al livello applicativo)
  - RST: Reset della connessione
  - **SYN: Sincronizza il Numero di Sequenza**
  - **FIN: Non ci sono altri dati dal mittente – chiusura della connessione**

Segmento TCP - campi

- **Finestra di ricezione** (16bit): indica il numero di byte di dati a partire da quello indicato nel campo *Numero di Riscontro* che il mittente di questo segmento è in grado di accettare. Serve per il controllo di flusso
- **Checksum (16 bit)**: checksum dell'intero pacchetto (dati, header TCP + parte dell'header IP) per rilevare errori (se i bit del segmento sono stati alterati). Si calcola come per UDP (per TCP è obbligatorio).
- **Opzioni** (facoltativo, lunghezza variabile, max 40 byte): negoziazione di vari parametri: ad es. dimensione massima segmento (MSS), selective acknowledgement supportato e blocchi di dati riscontrati selettivamente. Le opzioni sono sempre multipli di 8 bit e il loro valore è considerato per il calcolo della checksum.

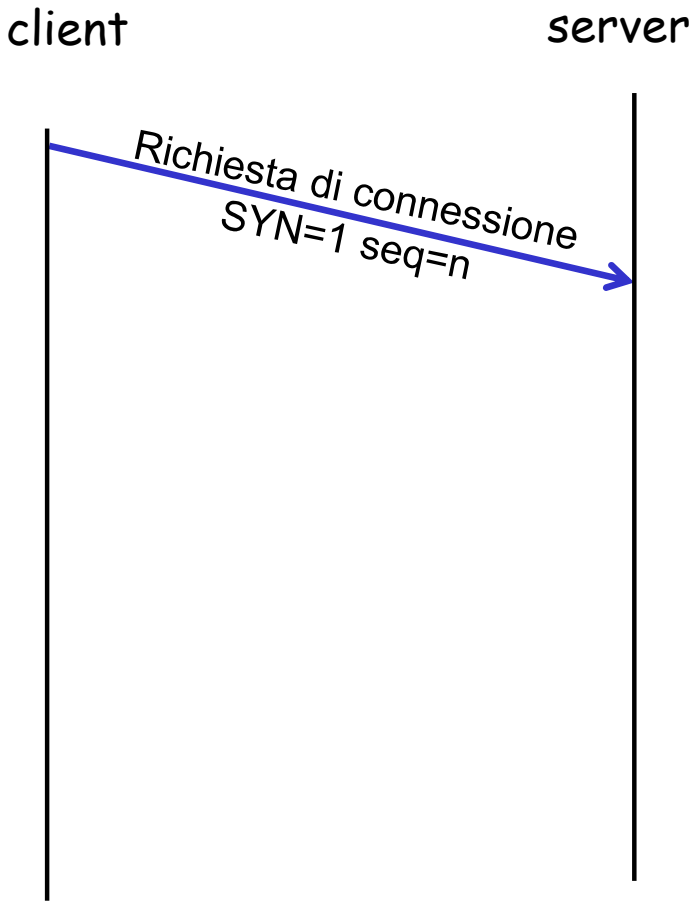
Segmento TCP - campi

- E’ inoltre presente un campo URGENT che permette la trasmissione di dati “fuori banda”, ovvero a priorità maggiore degli altri (la loro gestione però è affidata all’applicazione)
- **Puntatore Urgente** (16 bits): questo campo è un *offset* positivo a partire dal *Numero di Sequenza* del segmento corrente. E' interpretato solo se il bit URG è uguale ad 1. Punta al primo byte di dati **non** urgenti a partire dal *Numero di Sequenza,* e consente di far passare i dati urgenti in testa alla coda di ricezione. Nel segmento contenente dati urgenti **deve** essere presente almeno un byte di dati.
  - Ad esempio se un segmento contiene 400 byte di dati urgenti e 200 byte di dati non urgenti, il puntatore urgente vale 400

Gestione della connessione

- Handshake
- Trasferimento dati
- Chiusura della connessione

Handshake a tre vie



Il client invia una richiesta di connessione a un server TCP

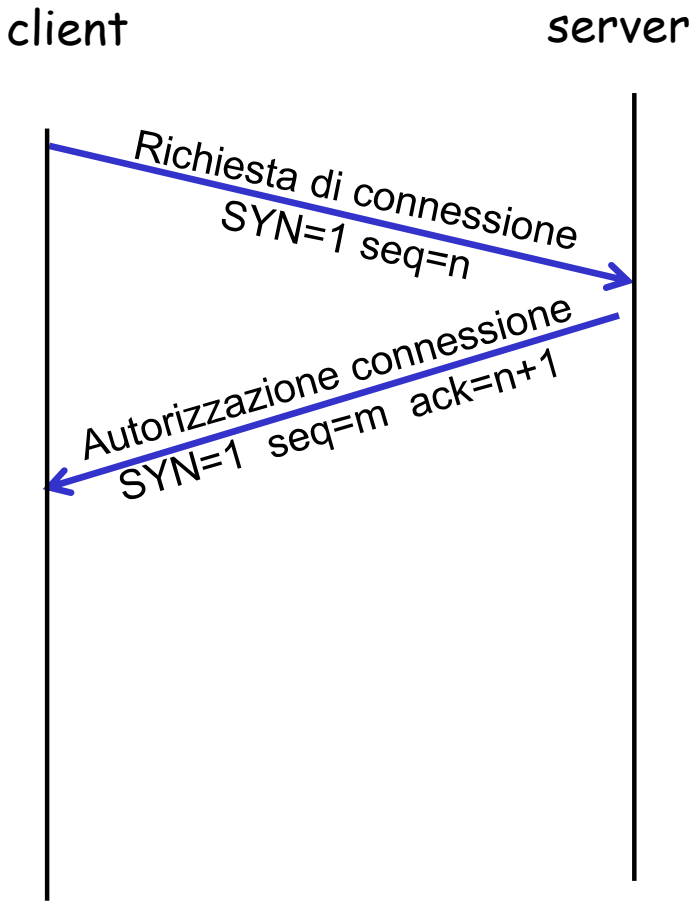
è attivo il bit SYN, il segmento non contiene dati

si trasmette anche un numero di sequenza iniziale (ISN)

Ad esempio:
Flag SYN=1 client_isn = 41

DOPO l'handshaking a livello di trasporto non c'è più distinzione tra client e server

Handshake a tre vie

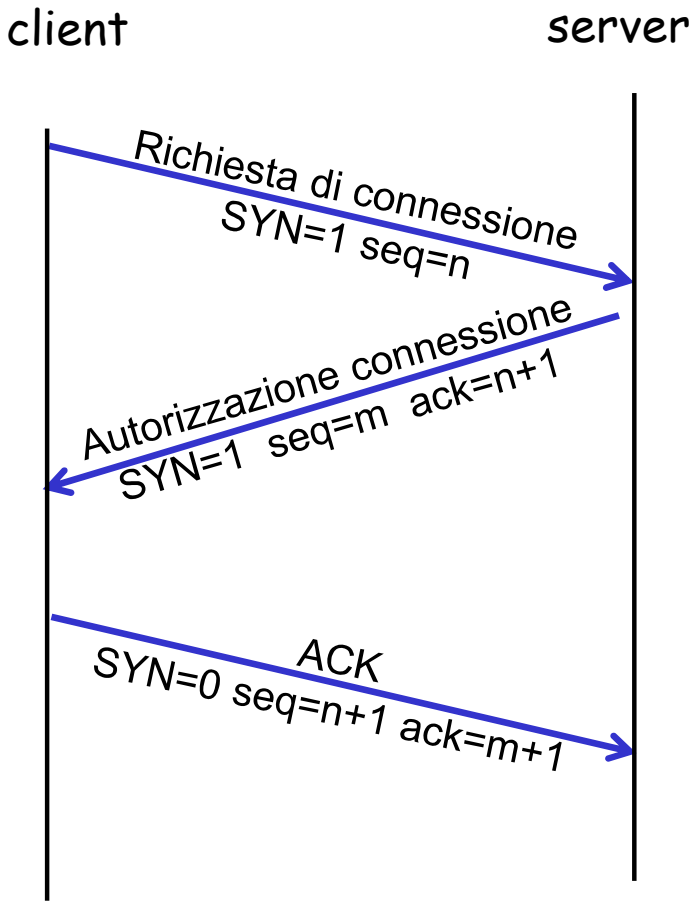


Il server estrae il segmento, alloca i buffer e le variabili TCP per la connessione

invia in risposta un segmento di connessione garantita al client (chiamato SYNACK)

- è attivo SYN, il numero di sequenza è il valore iniziale (es. server_isn = 78)
- è attivo ACK, il server aspetta client_isn+1 (es. 42)
- Esempio SYN=1, ACK = client_isn +1, proprio numero di sequenza iniziale server_isn. Segmento SYNACK

Handshake a tre vie



il client alloca buffer e variabili di connessione

manda un riscontro positivo del messaggio del server.

- SYN è inattivo. Questo segmento può già trasportare dati
- il prossimo dato sarà client_isn+1 (42) ed il client attende server_isn+1 (79)
- SYN=0, riscontro server_isn+1.

- Inizia lo scambio dati, SYN=0

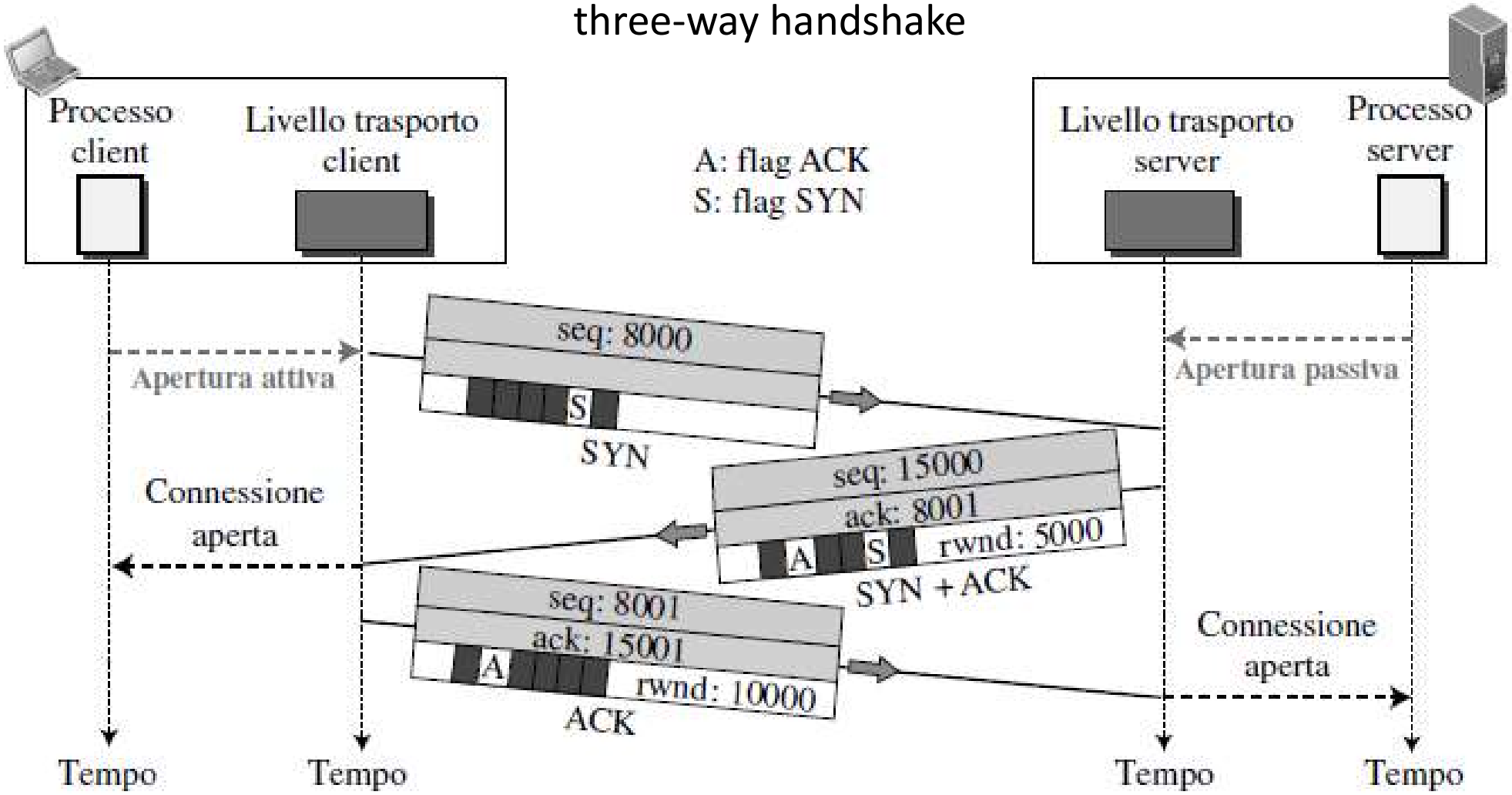
DOPO l'handshaking a livello di trasporto
non c'è più distinzione tra client e server

Handshake a tre vie

- I primi segmenti non hanno carico utile.
- All'arrivo del primo segmento il server inizializza due buffer (memorie di scambio) e le variabili, necessari per il controllo del flusso e della congestione.
- All'arrivo del riscontro del primo segmento il client alloca due buffer e le variabili, necessari per il controllo del flusso e della congestione.
- Alla ricezione del terzo segmento la connessione è instaurata.

Apertura connessione

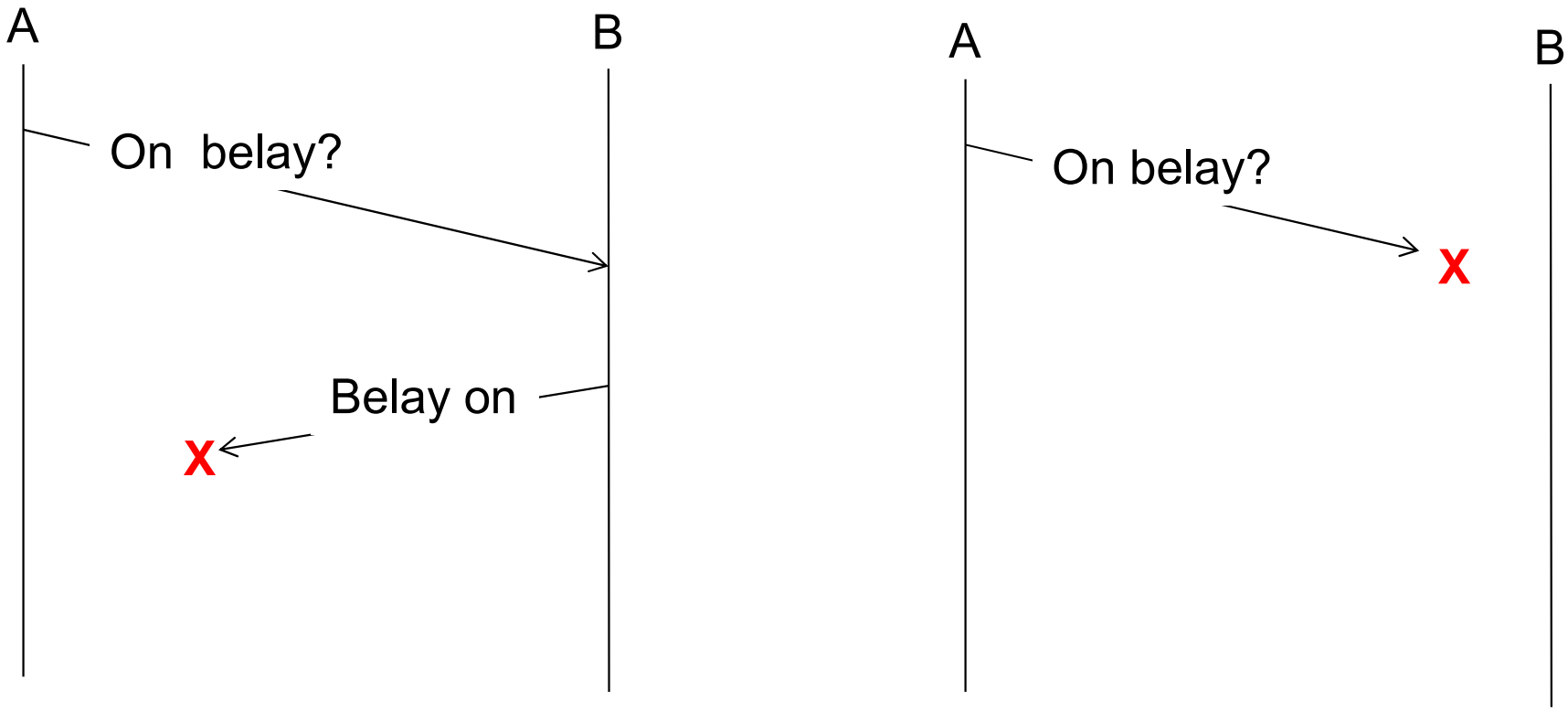
three-way handshake



SYN e SYN+ACK non contengono dati utente ma consumano un numero di sequenza

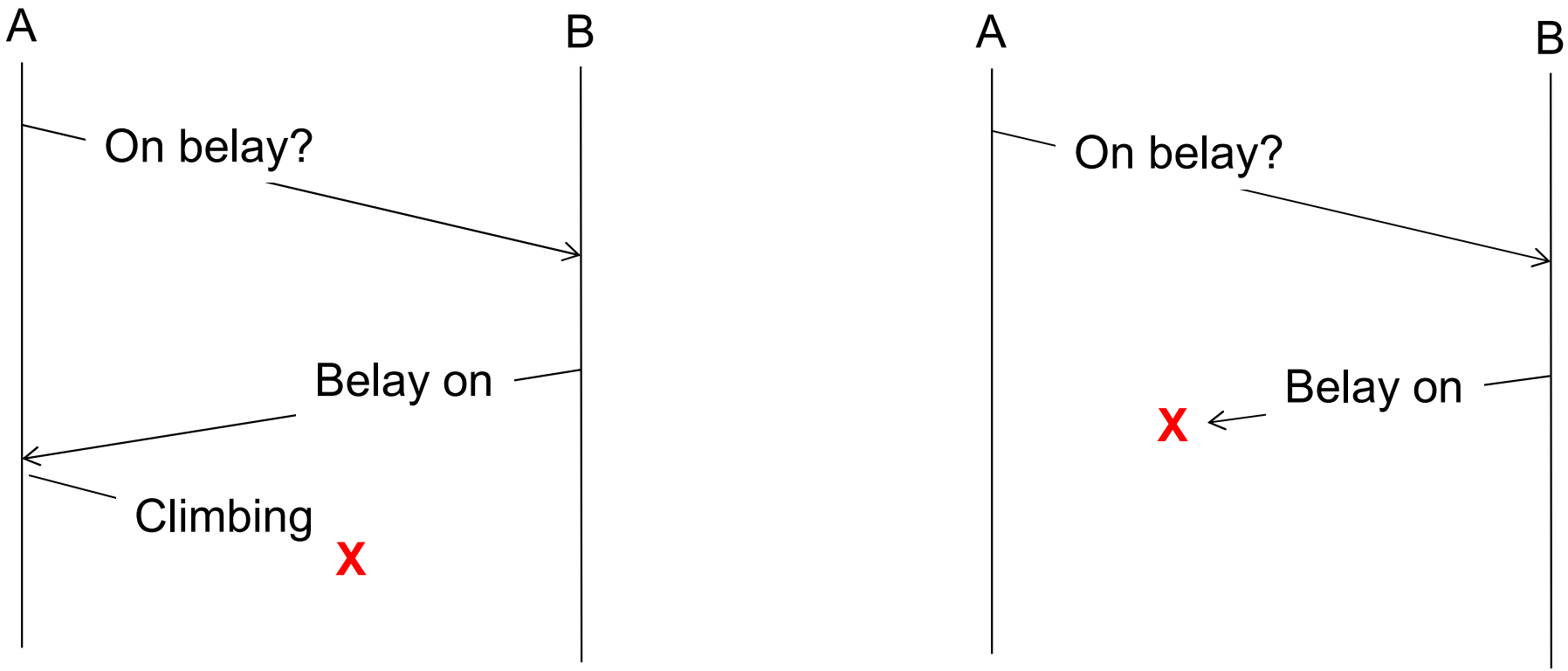
Importanza «terza via»

Importanza «terza via»



Situazioni indistinguibili per A

Importanza «terza via»

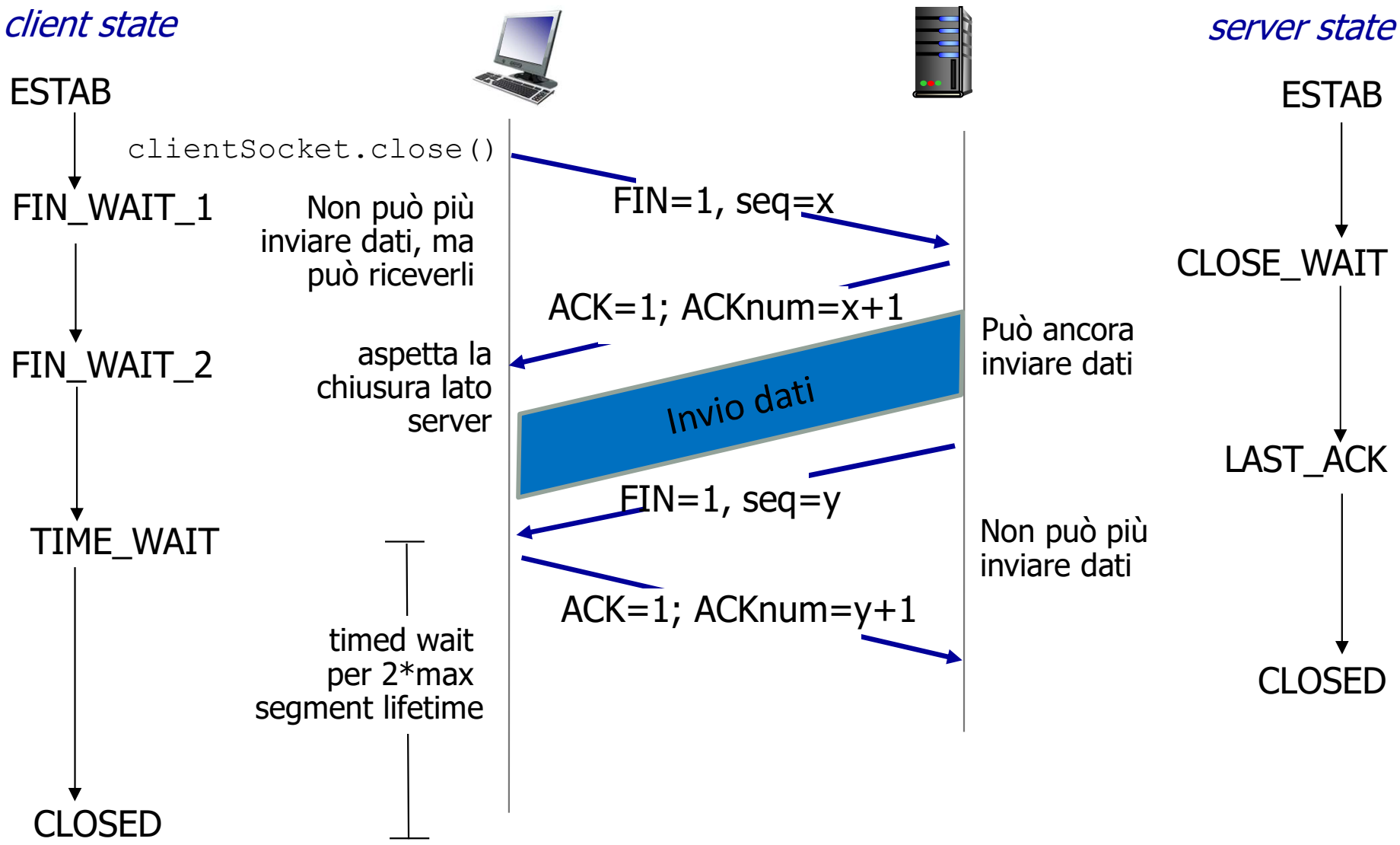


Situazioni indistinguibili per B !

TCP: chiusura della connessione

- Client e server chiudono ciascuno il loro lato della connessione
  - Invio di segmento TCP con bit FIN = 1
- Ciascuno risponde al FIN ricevuto con un ACK
  - Quando viene ricevuto un FIN, l'ACK può essere combinato con il proprio FIN
- È possibile anche lo scambio simultaneo di FIN

TCP: chiusura della connessione



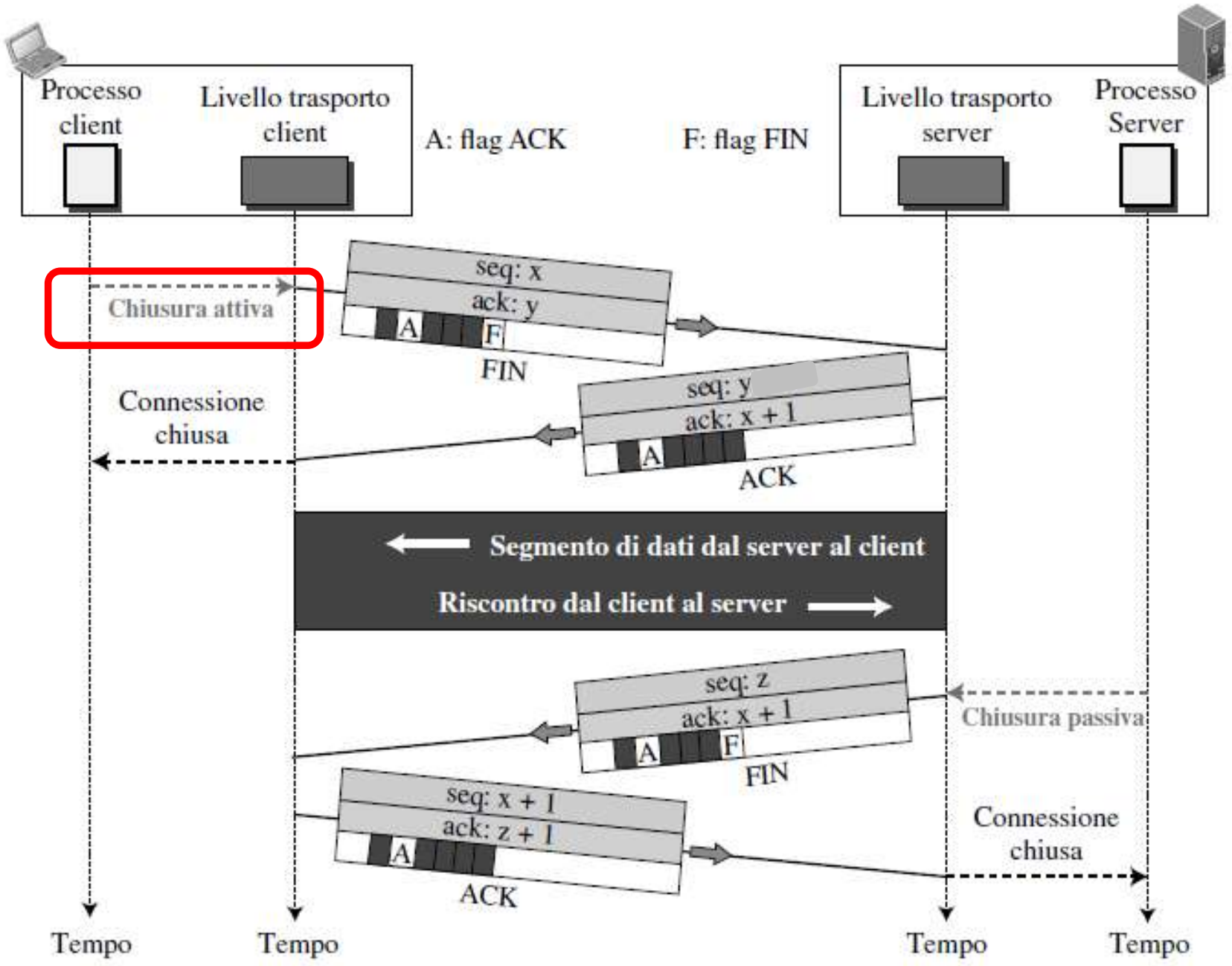
NB. MSL è 2 minuti secondo RFC, questo valore può variare
Per verificare su sistemi Linux `sysctl net.inet.tcp.msl`

Stato TIME-WAIT

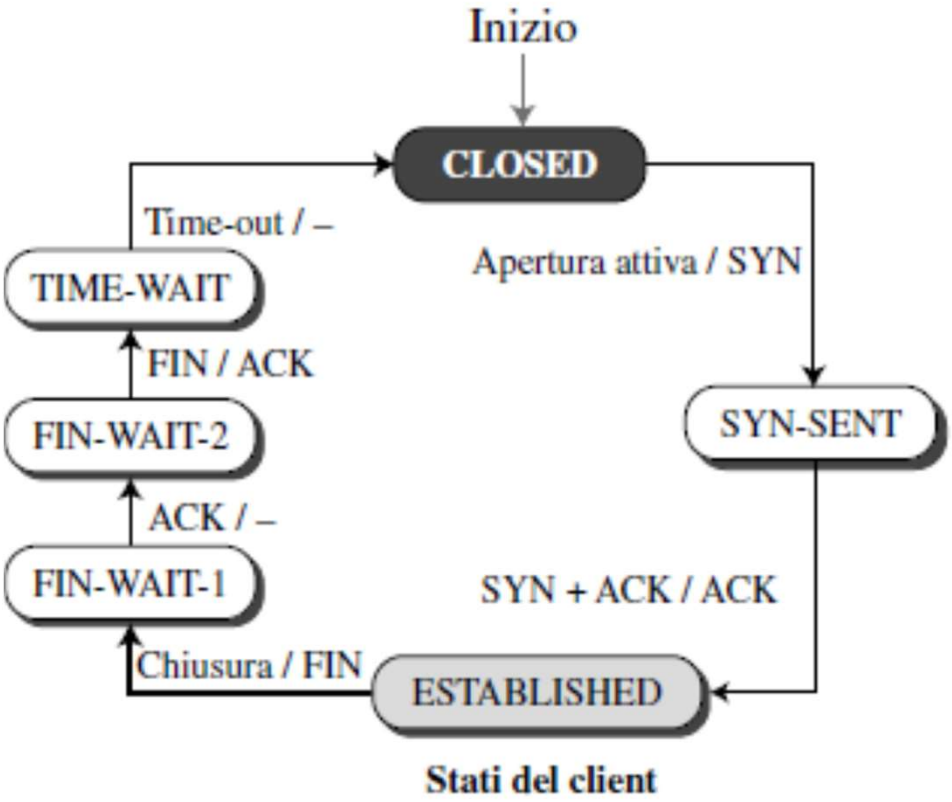
- TIME_WAIT è lo stato finale in cui il capo di una connessione che esegue la chiusura attiva resta prima di passare alla chiusura definitiva della connessione
  - due volte la MSL (Maximum Segment Lifetime).
  - La MSL è la stima del massimo periodo di tempo che un pacchetto IP può vivere sulla rete; questo tempo è limitato perché ogni pacchetto IP può essere ritrasmesso dai router un numero massimo di volte (detto hop limit).
  - Ogni implementazione del TCP sceglie un valore per la MSL (RFC 793 2 minuti, Linux 30 o 60 secondi).
- Lo stato TIME_WAIT viene utilizzato dal protocollo per due motivi principali:
  - implementare in maniera affidabile la terminazione della connessione in entrambe le direzioni.
    - Se l'ultimo ACK della sequenza viene perso, chi esegue la chiusura passiva manderà un ulteriore FIN, chi esegue la chiusura attiva deve mantenere lo stato della connessione per poter reinviare l'ACK
  - consentire l'eliminazione dei segmenti duplicati in rete.

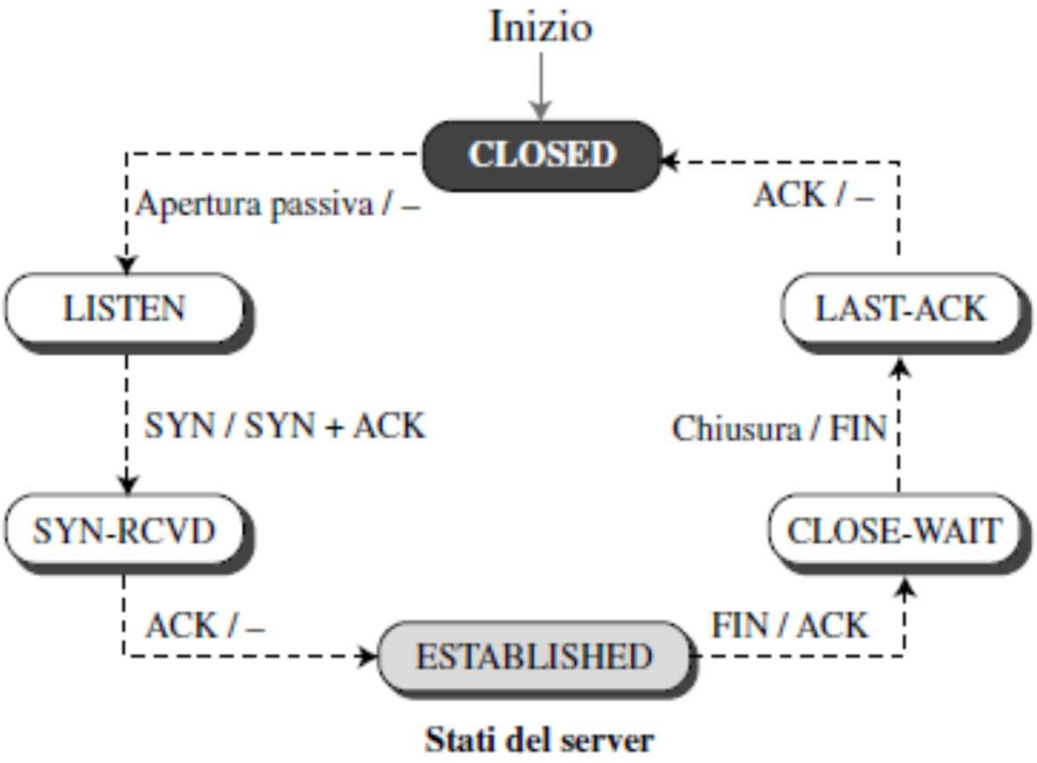
Half-close

Uno dei due processi può smettere di inviare dati mentre sta ancora ricevendo dati

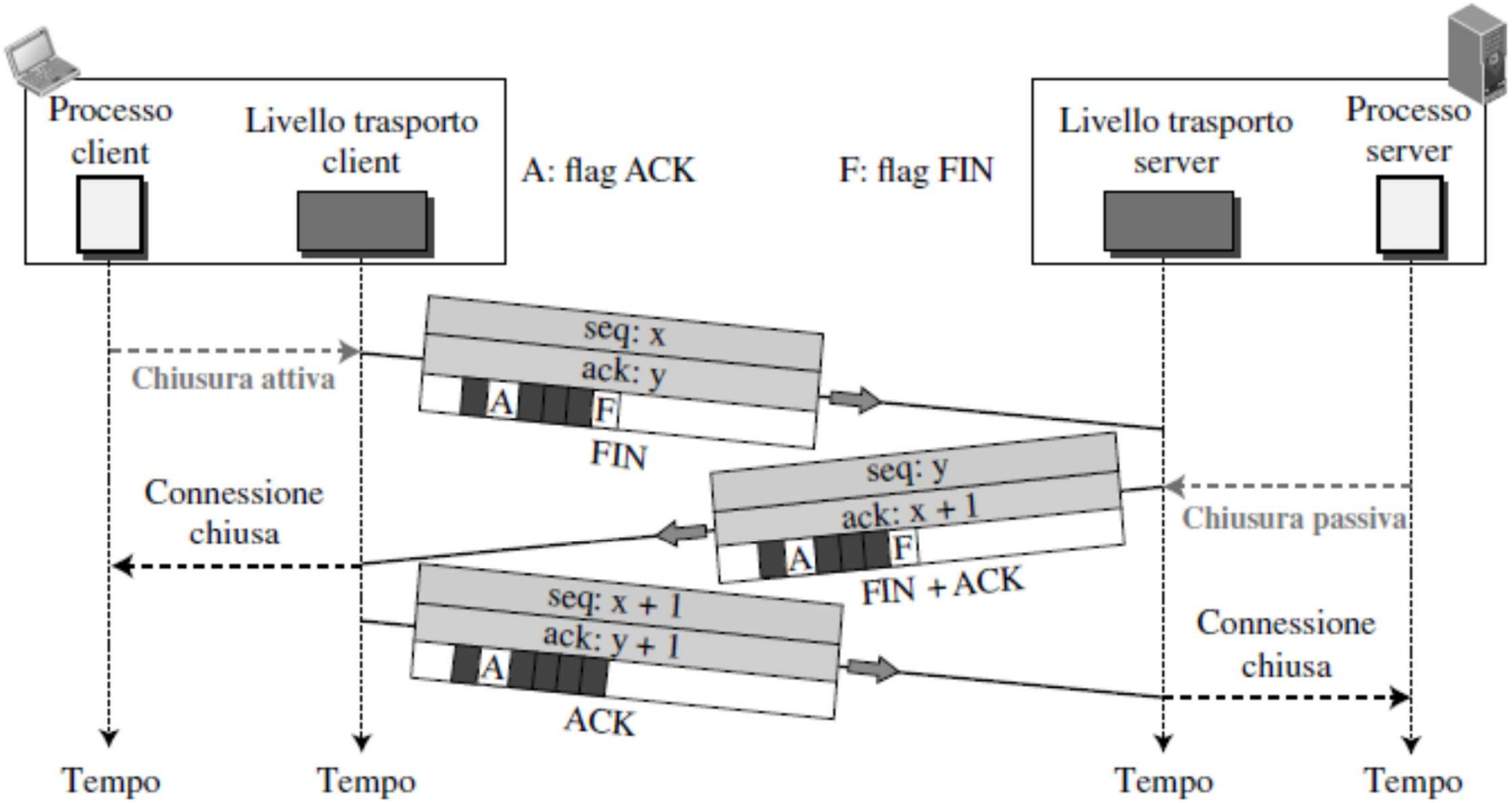


ASF per chiusura half-close



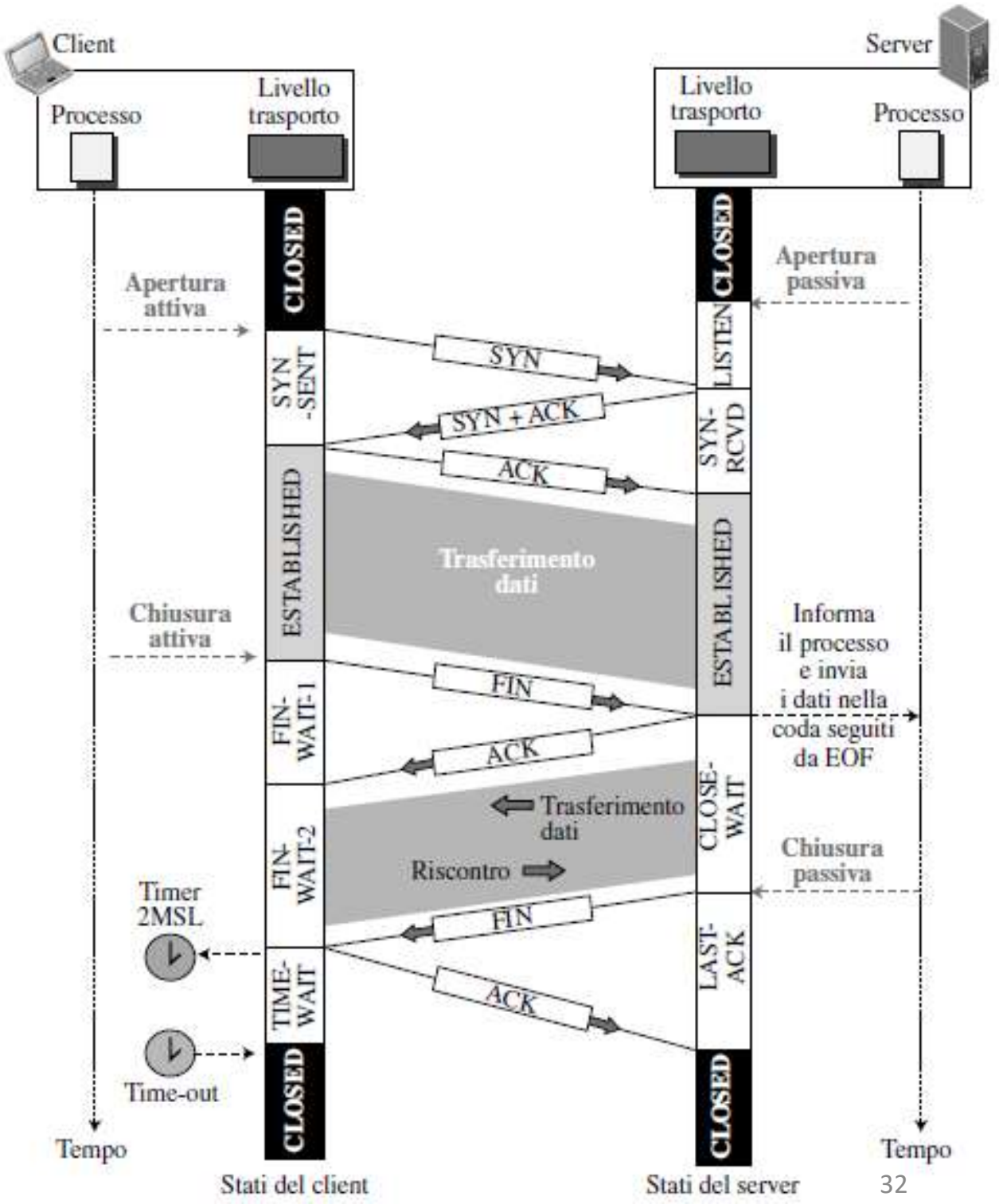


Chiusura connessione con three-way handshake

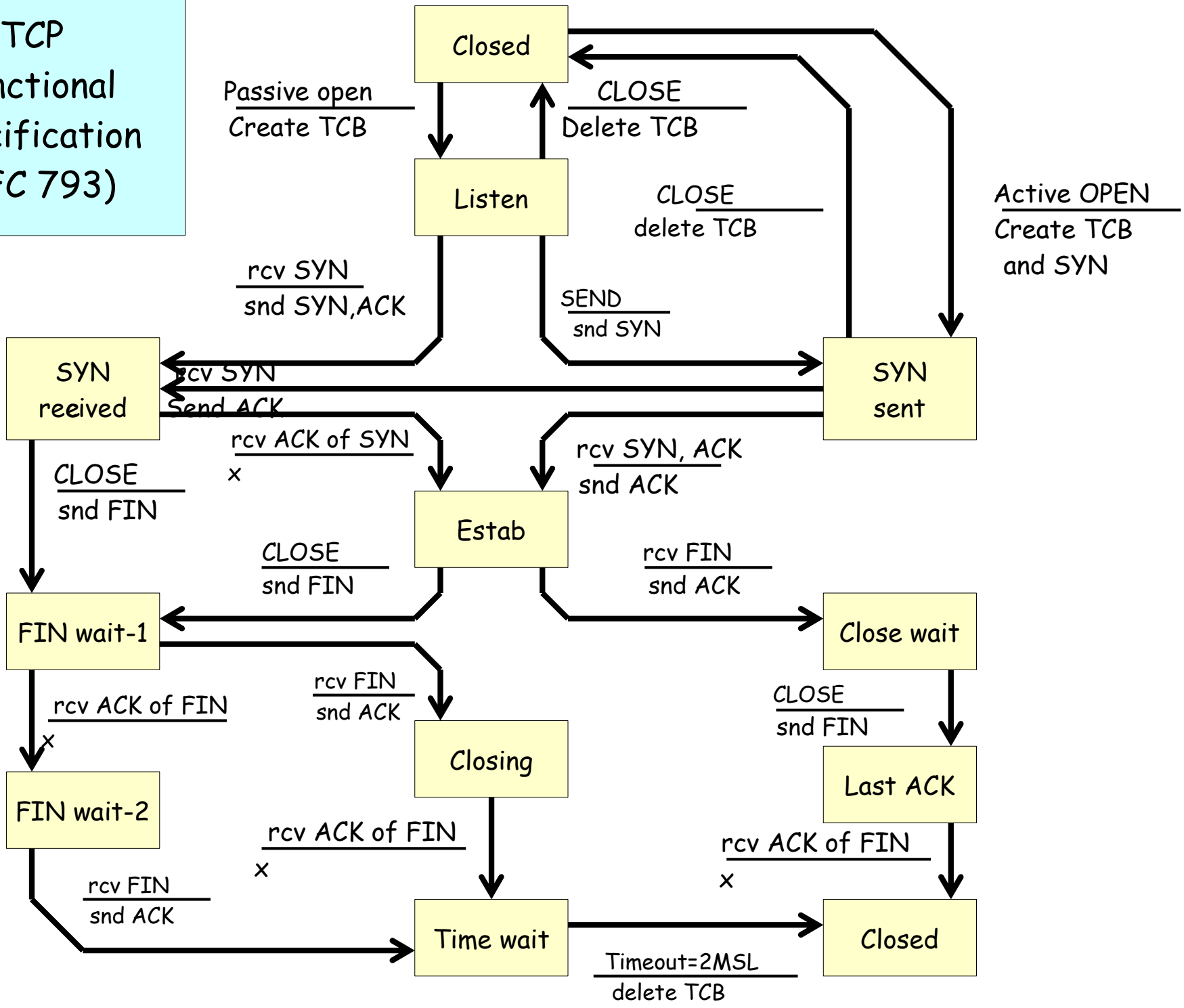


- Un FIN che non trasporta dati consuma comunque un numero di sequenza
- Idem per FIN+ACK

Apertura e chiusura connessione



TCP Functional Specification (RFC 793)



TCP: stati (1/2) – RFC 793

- LISTEN - represents waiting for a connection request from any remote TCP and port.
- SYN-SENT - represents waiting for a matching connection request after having sent a connection request.
- SYN-RECEIVED - represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.
- ESTABLISHED - represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.
- FIN-WAIT-1 - represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

TCP: stati (2/2) - RFC 793

- FIN-WAIT-2 - represents waiting for a connection termination request from the remote TCP.
- CLOSE-WAIT - represents waiting for a connection termination request from the local user.
- CLOSING - represents waiting for a connection termination request acknowledgment from the remote TCP.
- LAST-ACK - represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).
- TIME-WAIT - represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.
- CLOSED - represents no connection state at all.

TCP

Trasferimento dati affidabile
controllo di flusso
controllo di congestione

Corso di
Reti di Calcolatori
AA. 2023-2024

Federica Paganelli

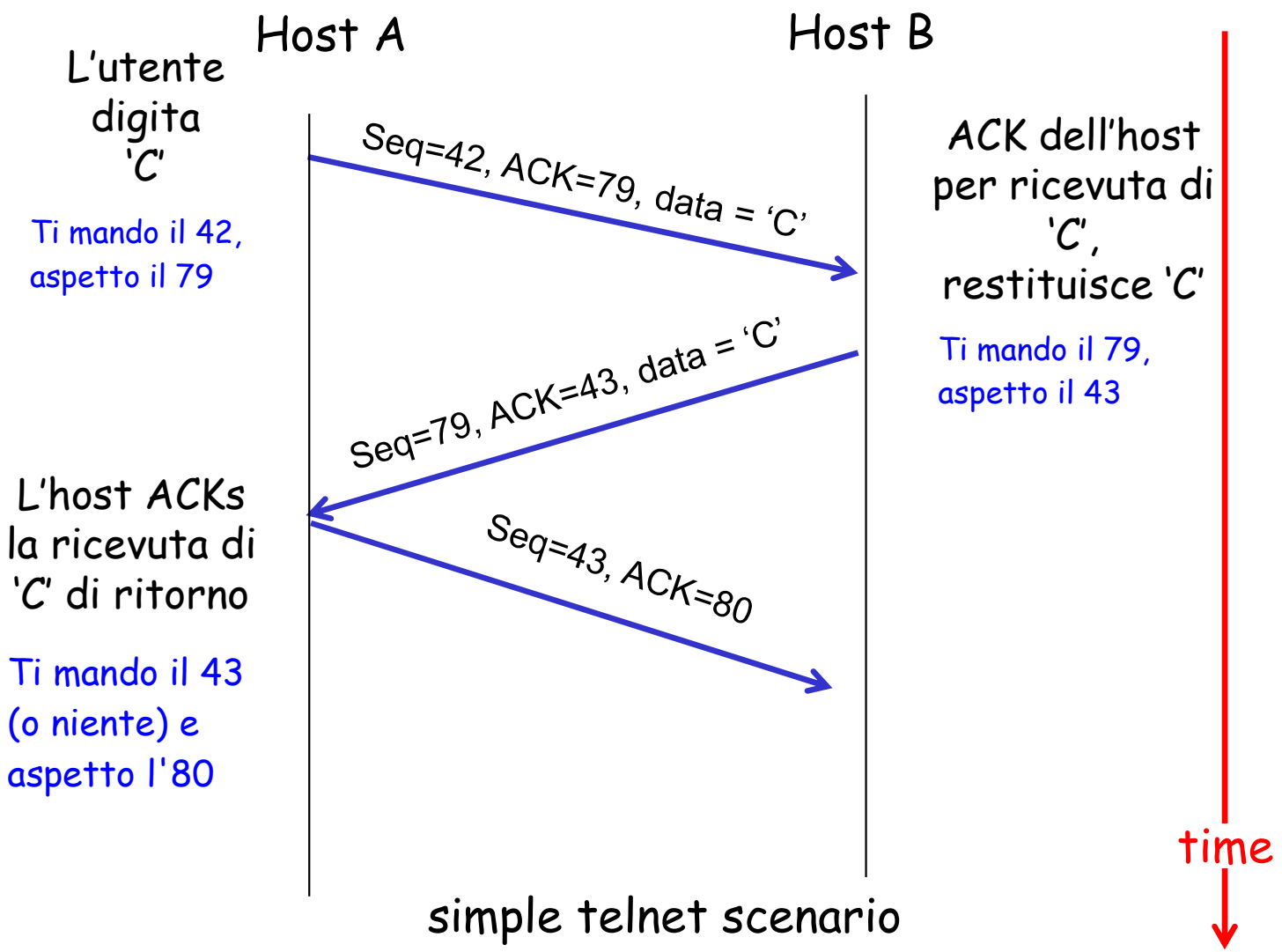
TCP: trasferimento dati affidabile

- Un segmento può essere smarrito o corrotto
- TCP crea un servizio di trasferimento dati affidabile sul servizio inaffidabile di IP
- Checksum: controllo obbligatori, i segmenti corrotti vengono scartati
- Riscontri
  - **Numero di sequenza** di un segmento è il numero del primo byte del segmento nel flusso di byte. N.B. i numeri di sequenza si applicano ai byte, non ai segmenti trasmessi
  - **Numero di riscontro**: il numero di sequenza del byte che l'host attende dall'altro.
  - **Riscontro cumulativo:** si effettua il riscontro dei byte fino al primo byte mancante nel flusso
  - Timer

Sequenza e riscontro
Esempio: Telnet su TCP

Viene inviato il carattere "C" da A verso B

Nella direzione opposta si rimanda quanto ricevuto



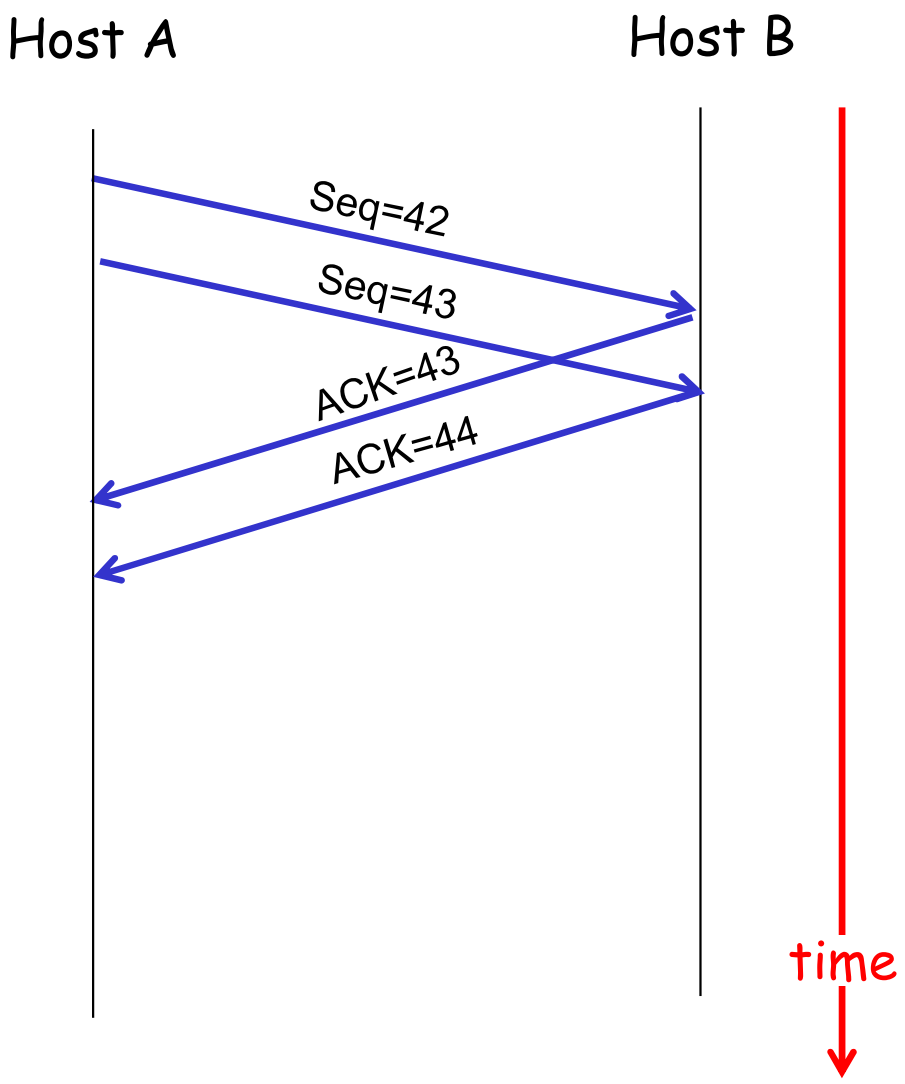
NB. Esempio con invio di 1 byte di dati

Kurose,Fig. 3.31 (3.a ed.)

Sequenza e riscontro
Pipeline

Pipeline:

- il mittente può inviare più segmenti senza attendere il riscontro
- Permette di aumentare la produttività



eventi lato mittente

- TCP riceve i dati dall'applicazione
- Incapsula i dati in uno o più segmenti e assegna numero di sequenza
- Avvia il timer di ritrasmissione (timeout di ritrasmissione - RTO)
  - Il timer viene avviato se non è già in funzione per un qualche altro segmento

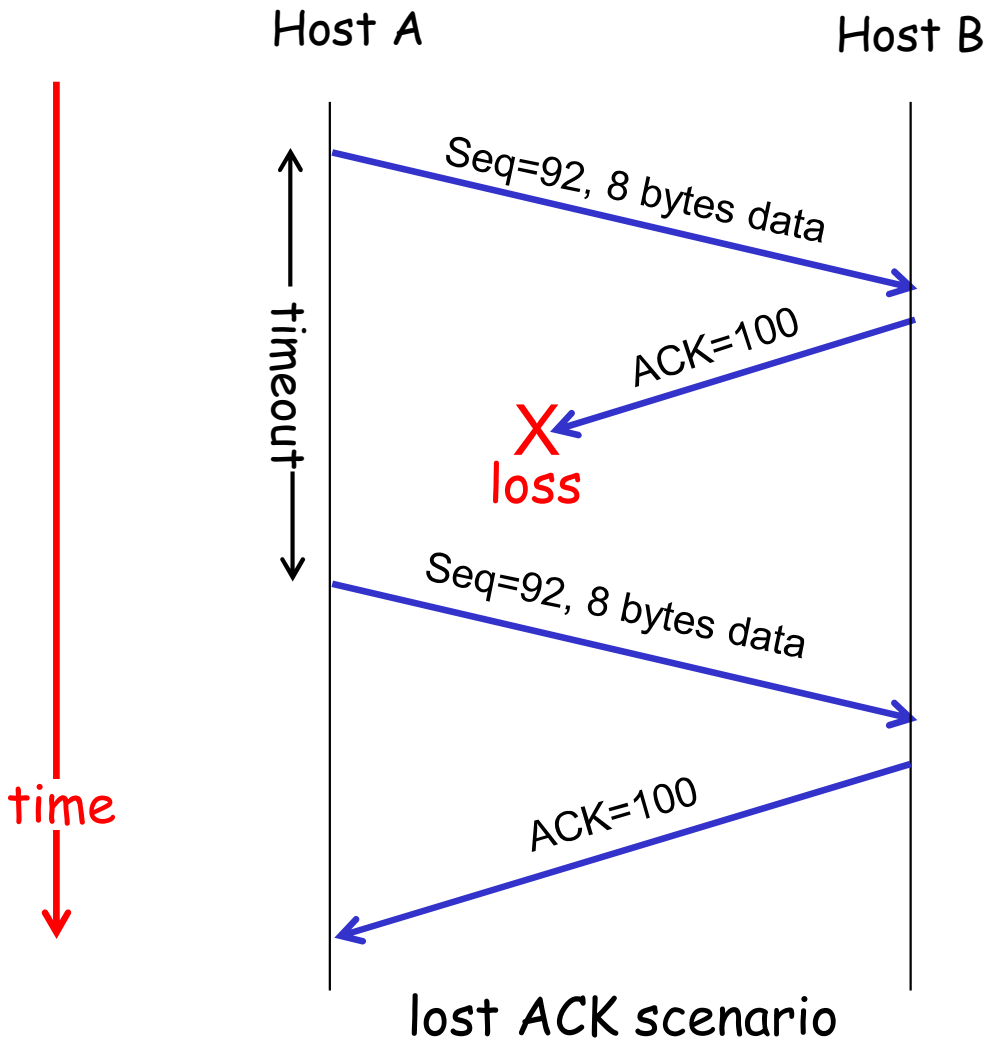
eventi lato mittente

- **Ritrasmissione** dei segmenti in caso di:
  - Timeout
  - Ricezione di tre ACK duplicati
- Timeout
  - Ritrasmette il segmento che non è stato riscontrato e che ha causato il timeout
  - Il timer viene riavviato
- Ack duplicato
  - Se il mittente riceve tre ACK duplicati, il segmento successivo a quello riscontrato è andato perso. Ritrasmissione veloce (*fast retransmission*), prima della scadenza del timer.

TCP: trasferimento dati affidabile

- **Segmenti fuori sequenza**
  - I dati possono arrivare fuori sequenza ed essere temporaneamente memorizzati dall'entità TCP destinataria
  - Il TCP non dice come il destinatario deve gestire i pacchetti fuori sequenza, dipende dall'implementazione
- Nelle versioni più recenti si implementa la Selective ACK (SACK)
  - i pacchetti ricevuti fuori sequenza vengono memorizzati
  - riscontro di pacchetti fuori sequenza e duplicati inviato in OPTIONS

Ritrasmissione dovuta a riscontro perso



A, trascorso un timeout senza riscontro, ritrasmette

Kurose, Fig. 3.34 (3.a ed.)

Lato destinatario

(1) Tutti i segmenti inviati per trasmettere dati includono ACK

(2) Se destinatario non ha dati da inviare e riceve segmento «in ordine» ritarda invio ACK di 500ms a meno che non riceva nuovo segmento

(3) Se destinatario riceve segmento atteso e precedente non è stato riscontrato allora invia immediatamente ACK

Se destinatario riceve (4) segmento fuori sequenza oppure (5) «mancante» («buco» in una sequenza) oppure (6) duplicato allora invia immediatamente un segmento ACK (indicando prossimo numero atteso)

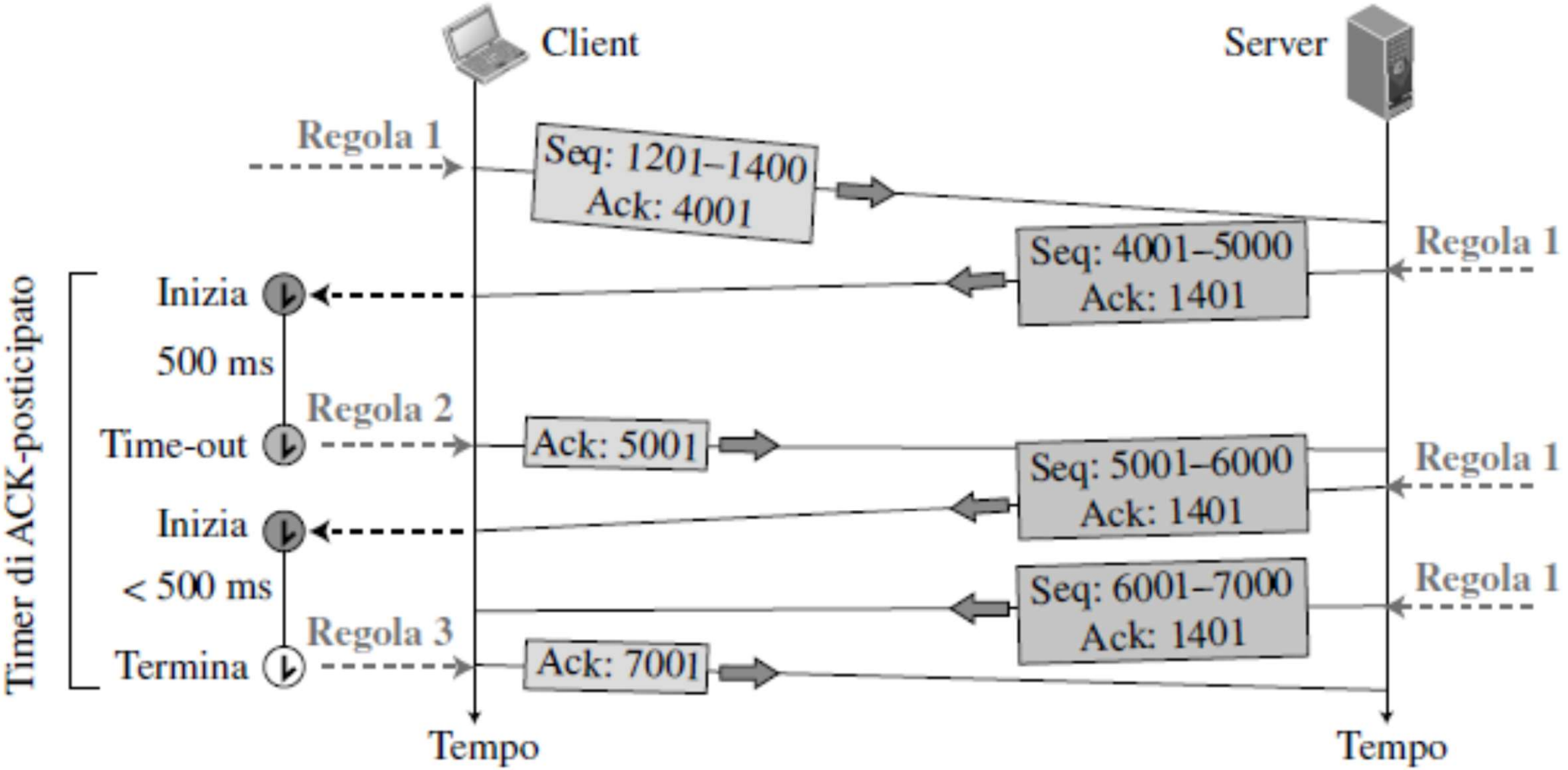
Operatività normale

(1) Tutti i segmenti contenenti dati includono ACK

(2) Se destinatario non ha dati da inviare e riceve segmento «in ordine» ritarda invio ACK di 500ms a meno che non riceva nuovo segmento

(3) Se destinatario riceve segmento atteso e precedente non è stato riscontrato allora invia immediatamente ACK

Se destinatario riceve (4) segmento fuori sequenza oppure (5) «mancante» oppure (6) duplicato allora invia immediatamente un segmento ACK (indicando prossimo numero atteso)



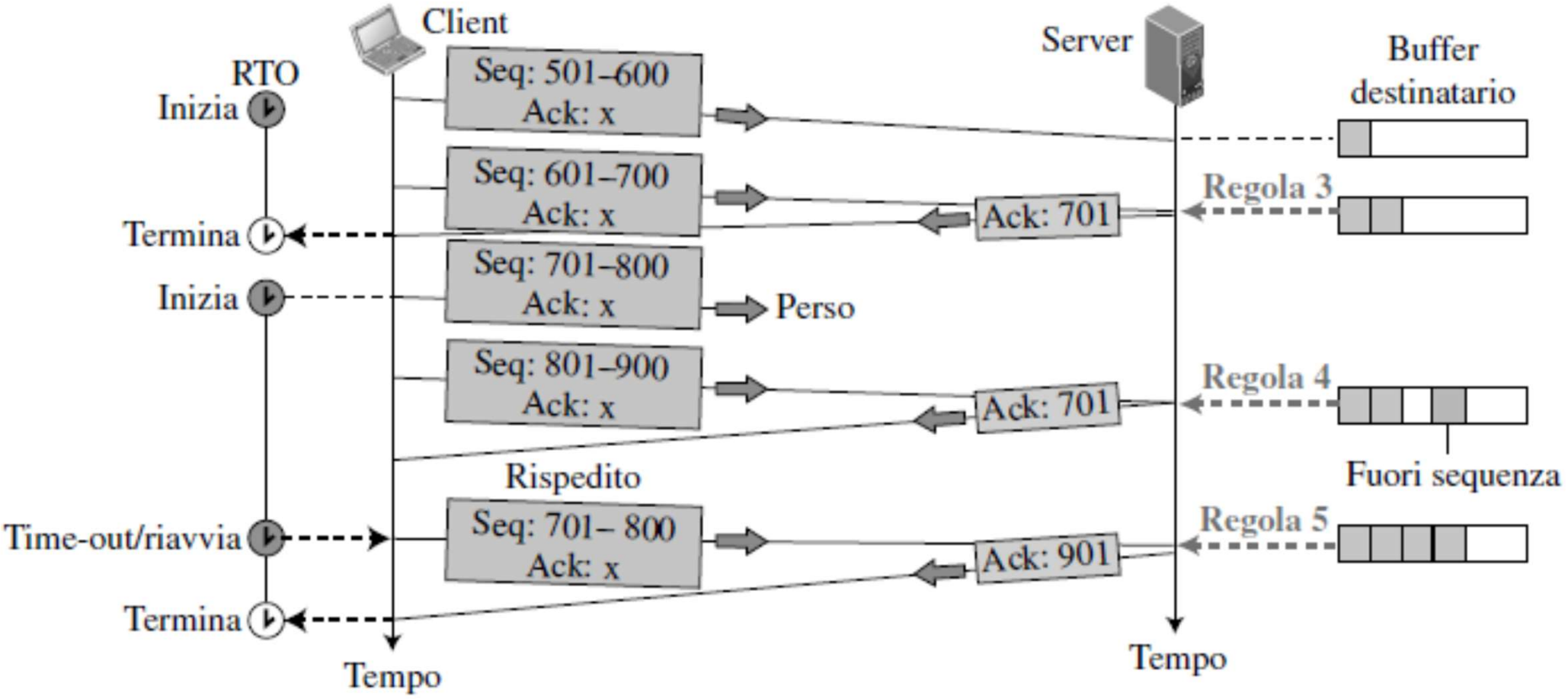
Segmento smarrito

(1) Tutti i segmenti contenenti dati includono ACK

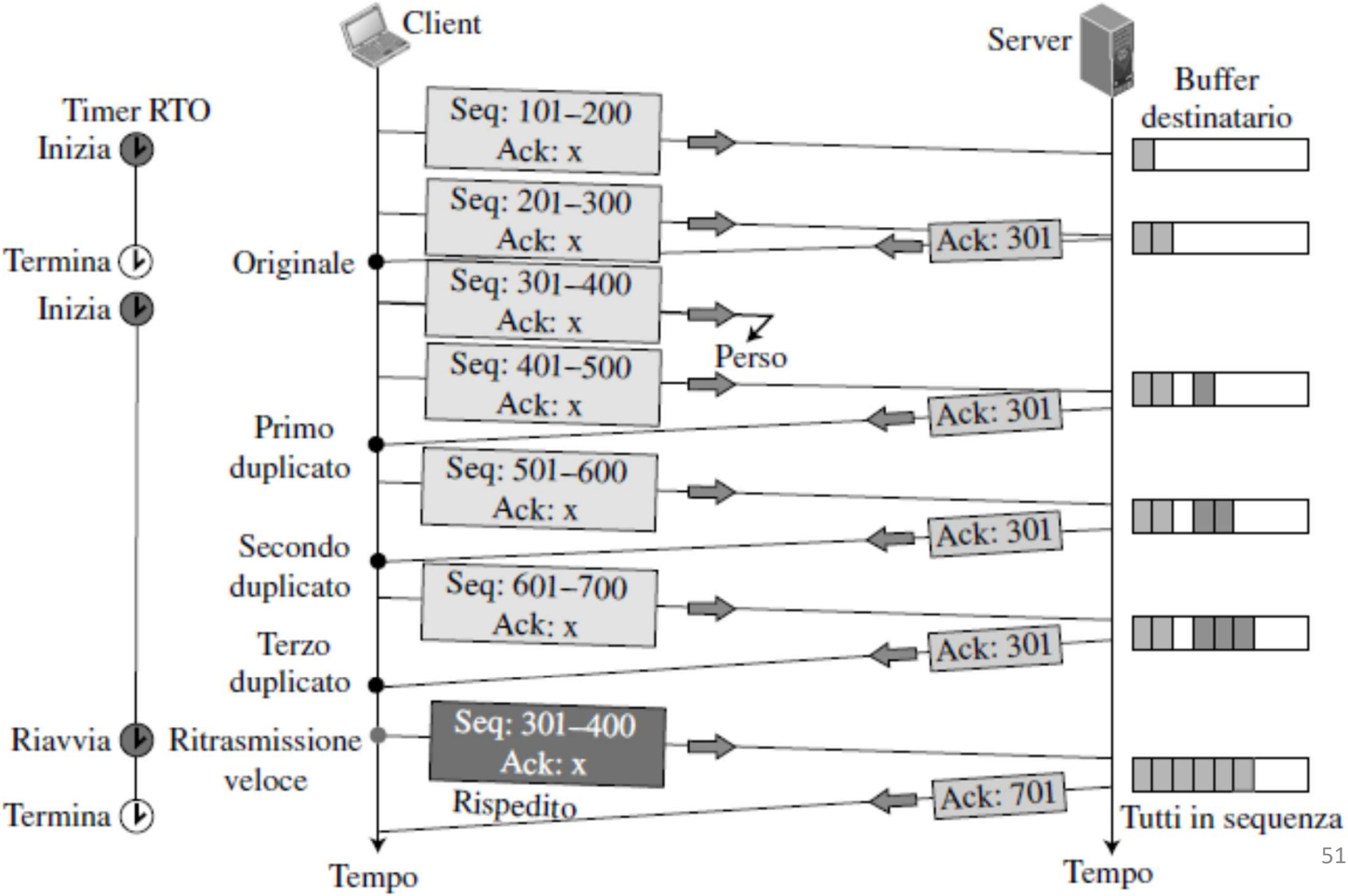
(2) Se destinatario non ha dati da inviare e riceve segmento «in ordine» ritarda invio ACK di 500ms a meno che non riceva nuovo segmento

(3) Se destinatario riceve segmento atteso e precedente non è stato riscontrato allora invia immediatamente ACK

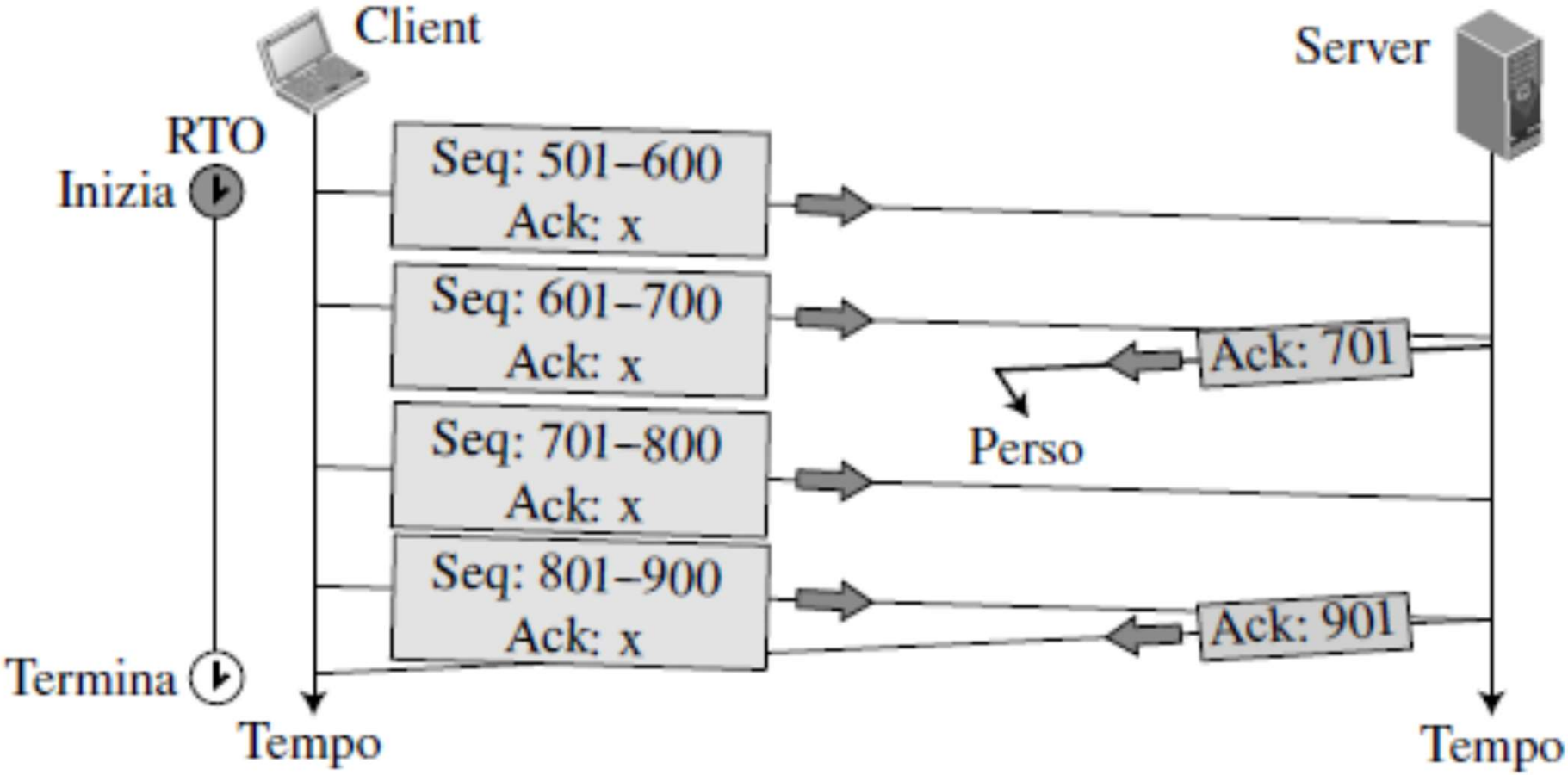
Se destinatario riceve (4) segmento fuori sequenza oppure (5) «mancante» oppure (6) duplicato allora invia immediatamente un segmento ACK (indicando prossimo numero atteso)



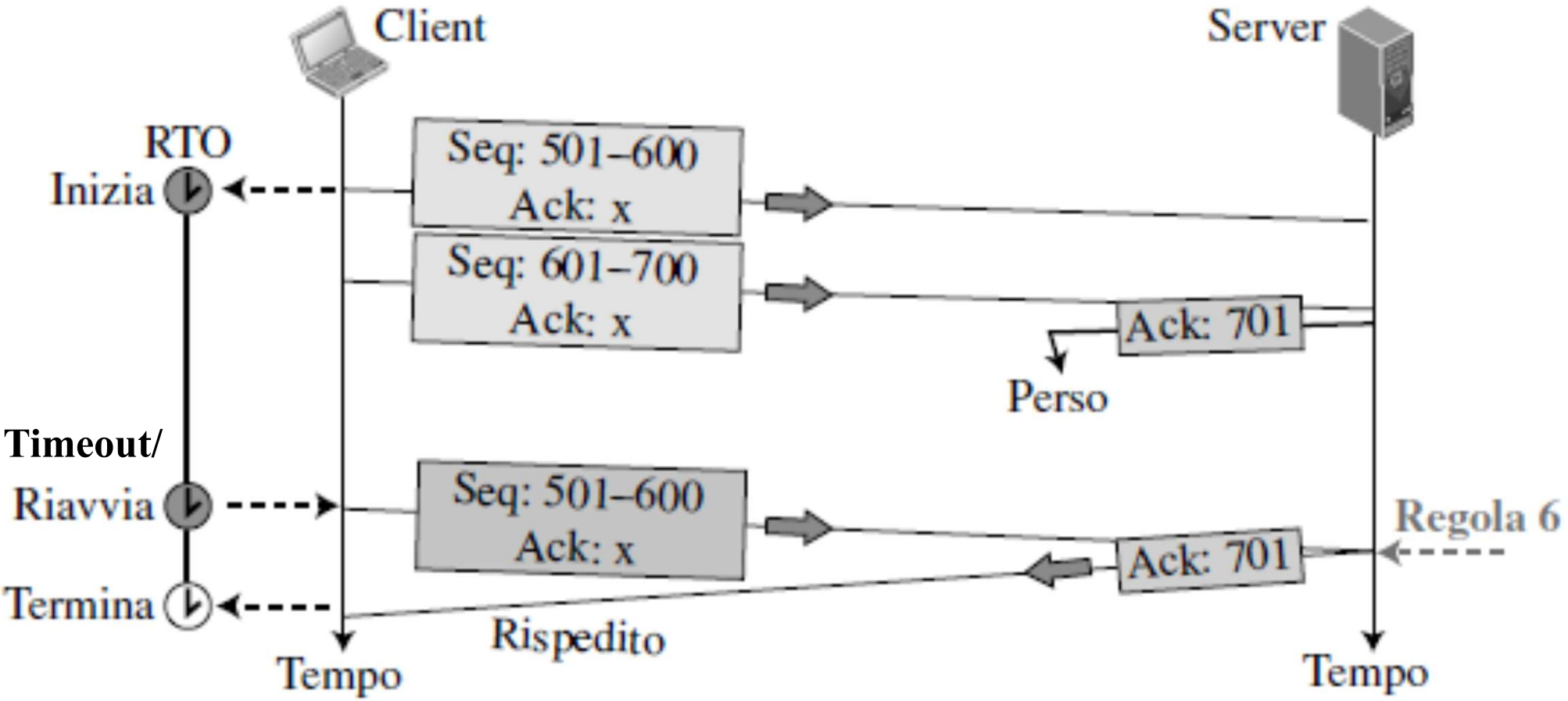
Ritrasmissione veloce



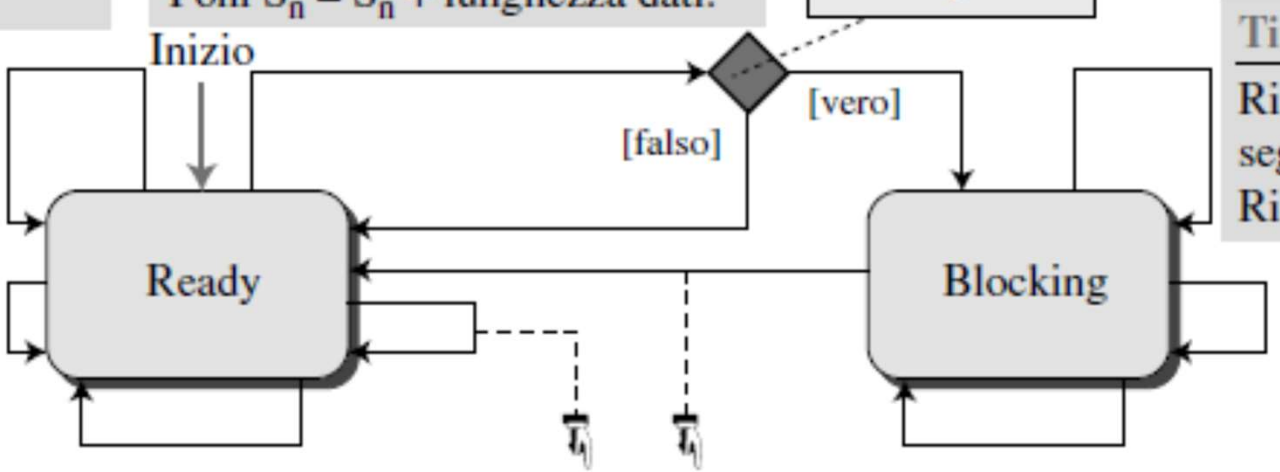
Riscontro smarrito



Riscontro perso corretto da ritrasmissione



FSM semplificata per il mittente TCP



Vengono accettati dei dati dal processo.

Crea un segmento (seqNo = S_n).
Memorizzane una copia nella coda e spediscilo.
Se si tratta del primo segmento nella coda, avvia il timer.
Poni $S_n = S_n$ + lunghezza dati.

Time-out scaduto

Rispedisci il primo segmento nella coda.
Riavvia il timer.

Ricevuto ACK corrotto

Scartalo.

Time-out scaduto

Rispedisci il primo segmento nella coda.
Riavvia il timer.

Ricevuto ACK corrotto

Scartalo.

Ricevuto riscontro duplicato

Poni dupNo = dupNo + 1.
Se (dupNo = 3) rispedisci il primo segmento nella coda, fai ripartire il timer e imposta dupNo = 0.

Ricevuto un riscontro corretto per il primo segmento nella coda

Fai scorrere la finestra (S_f = ackNo) e regola la dimensione della finestra.
Elimina il segmento dalla coda.
Se vi sono altri segmenti nella coda, fai ripartire il timer.

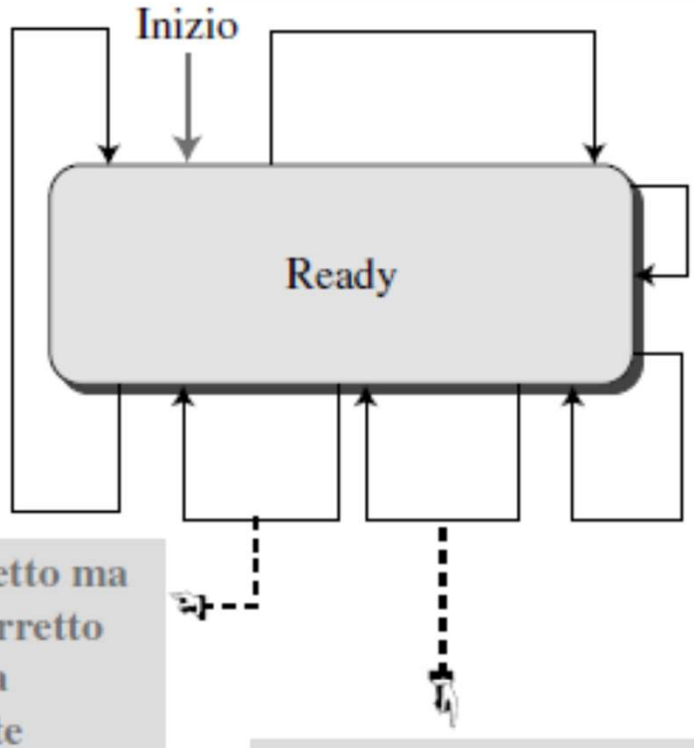
Ricevuto riscontro duplicato

Imposta dupNo = dupNo + 1.
Se (dupNo = 3) rispedisci il primo segmento nella coda, fai ripartire il timer e imposta dupNo = 0.

FSM semplificata per il lato destinatario TCP

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.



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.

TCP: calcolo del timeout (1)

- Il tempo di timeout (RTO) è fondamentale per il funzionamento di TCP
- Deve essere maggiore di RTT (Round Trip Time)
  - RTT: tempo trascorso da quando si invia un segmento a quando se ne riceve il riscontro
- Viene calcolato analizzando gli RTT dei segmenti non ritrasmessi (Sample RTT, stimato per un segmento trasmesso – non per ogni invio)

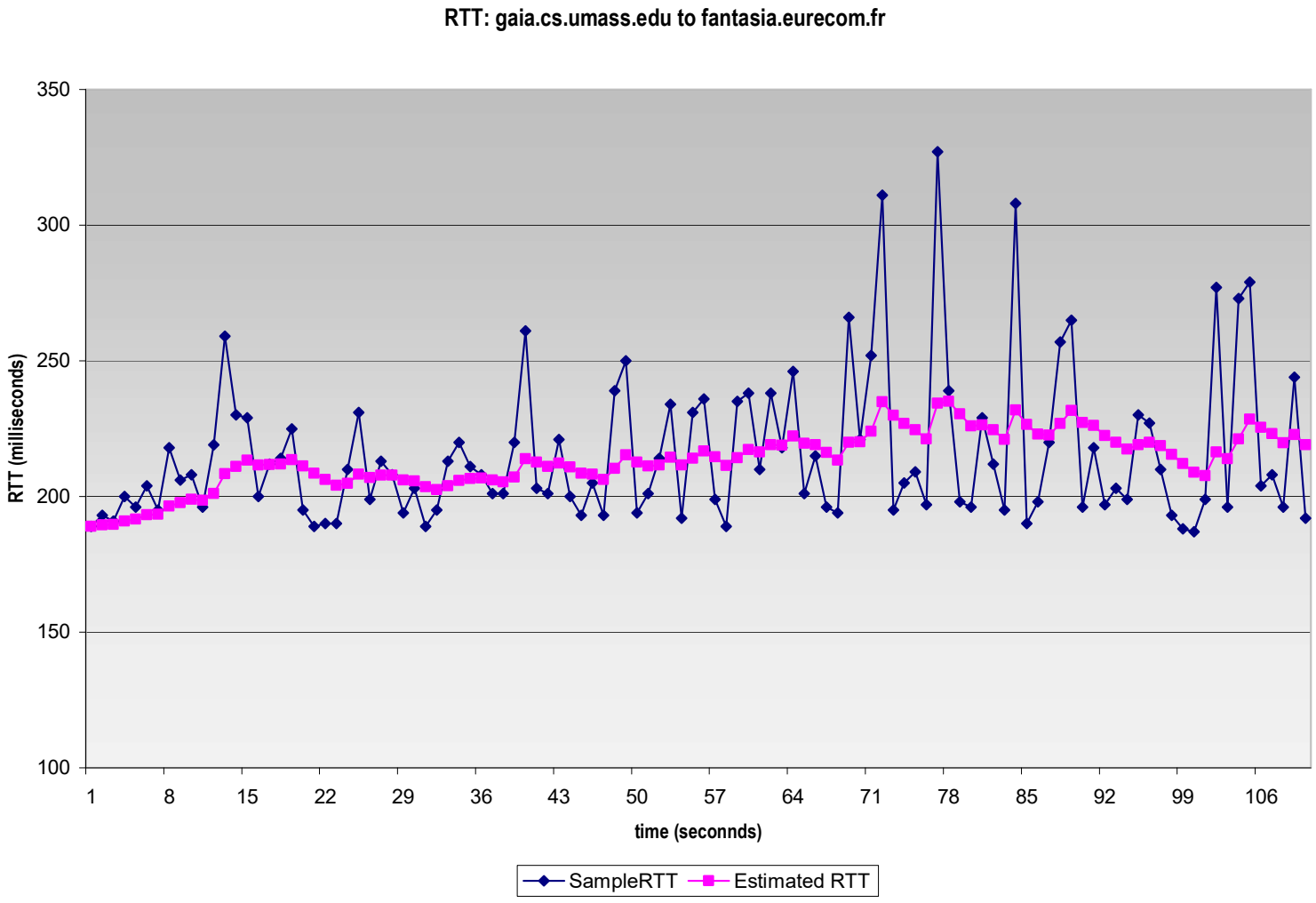
Estimated RTT = (1 – α) * EstimatedRTT + α * Sample RTT

- SampleRTT può fluttuare. Si considera EstimatedRTT: combinazione dei precedenti valori di EstimatedRTT con il nuovo valore SampleRTT
- Il valore di **α** viene posto a 1/8 in modo da rendere via via meno importanti gli RTT dei pacchetti più vecchi (RFC 2988)

Estimated RTT = 0,875 * EstimatedRTT + 0,125 * Sample RTT

exponential weighted moving average (EWMA)

TCP: calcolo del timeout (1)



TCP: calcolo del timeout (2)

- Oltre al valore RTT stimato è necessario anche una stima della variabilità di RTT data dalla seguente formula

$$RTT_{DEV} = (1-\beta)\, RTT_{DEV} + \beta\, |RTT_{SAMPLE} - RTT_{ESTIMATEd}|$$

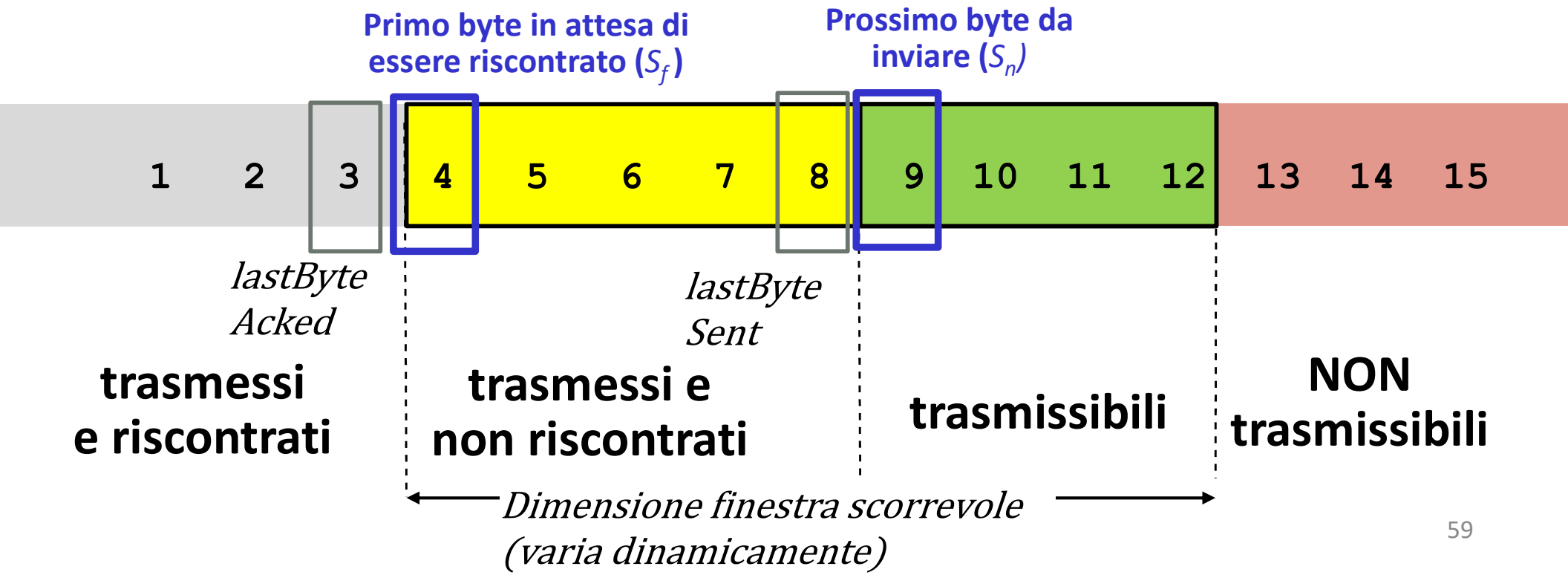
- Stima di quanto SampleRTT si discosta da EstimatedRTT
- Il valore di β viene posto a 1/4 (RFC 2988)
- Una volta ottenuti questi valori, il timeout viene normalmente calcolato come

$$\mathbf{RTO = RTT_{ESTIMATED} + 4\, RTT_{DEV}}$$

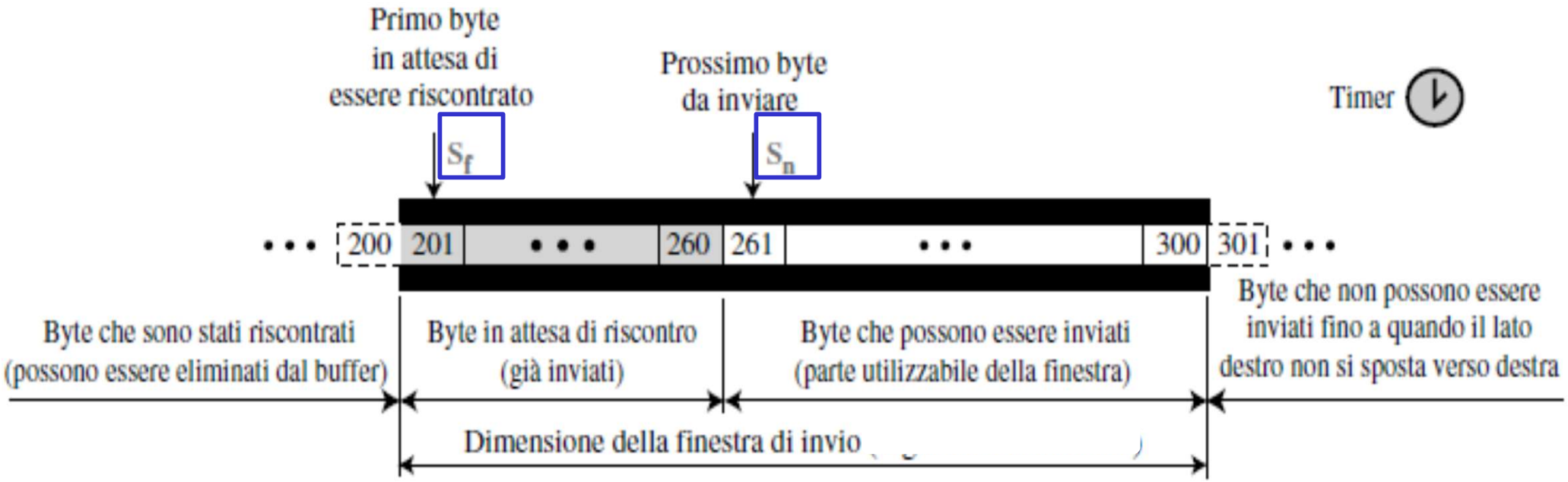
- In molte implementazioni, dopo un errore (es. ACK non ricevuto) si raddoppia il timeout: si tratta di un primo meccanismo di controllo della congestione

Finestra di trasmissione (TCP)

- I dati inviati dal processo a livello applicativo sono mantenuti nel buffer di invio
- La trasmissione dei dati si basa sulla finestra di trasmissione (*sliding window*).
  - finestra sovrapposta sulla sequenza da trasmettere
  - negoziata dinamicamente
  - viene fatta avanzare alla ricezione di un ACK



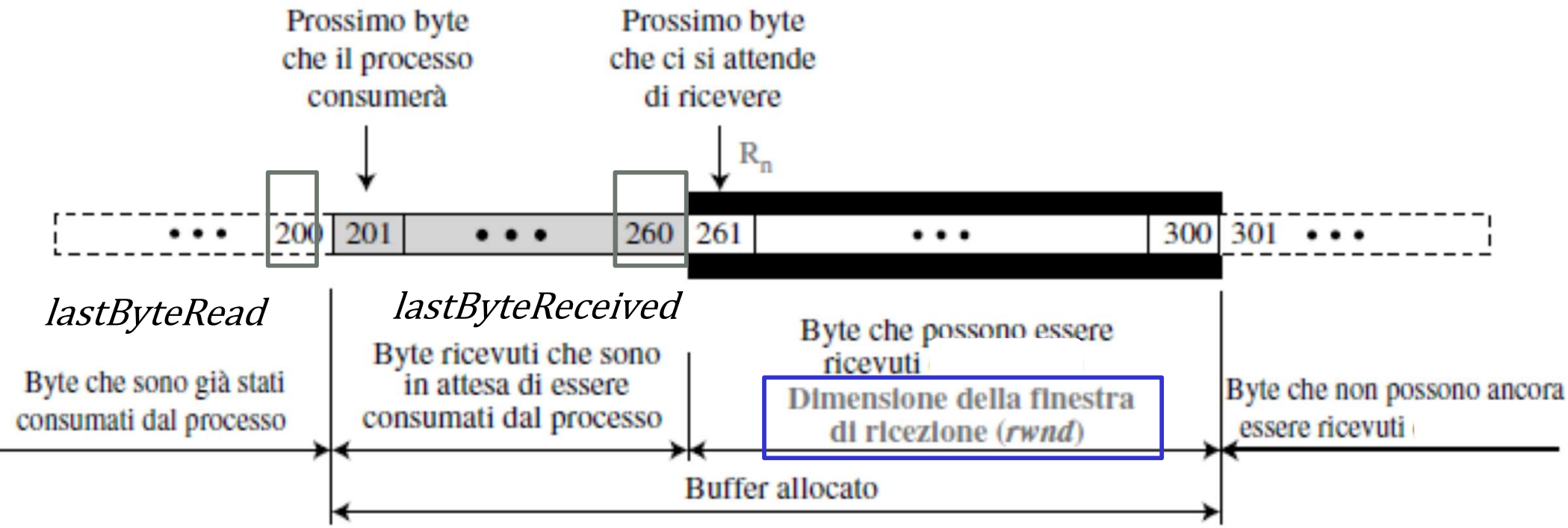
Finestra di trasmissione



S_f (SendFirst): numero di sequenza del primo byte in attesa di essere riscontrato (chiamato SendBase sul Kurose, SND.UNA nelle RFC)

S_n (Send Next): prossimo byte da inviare *(*prossimo numero di sequenza da inviare, chiamato NextSeqNum sul Kurose, SND.NXT nelle RFC)

Finestra di ricezione



R_n : Receive next (RCV.NXT nelle RFC)