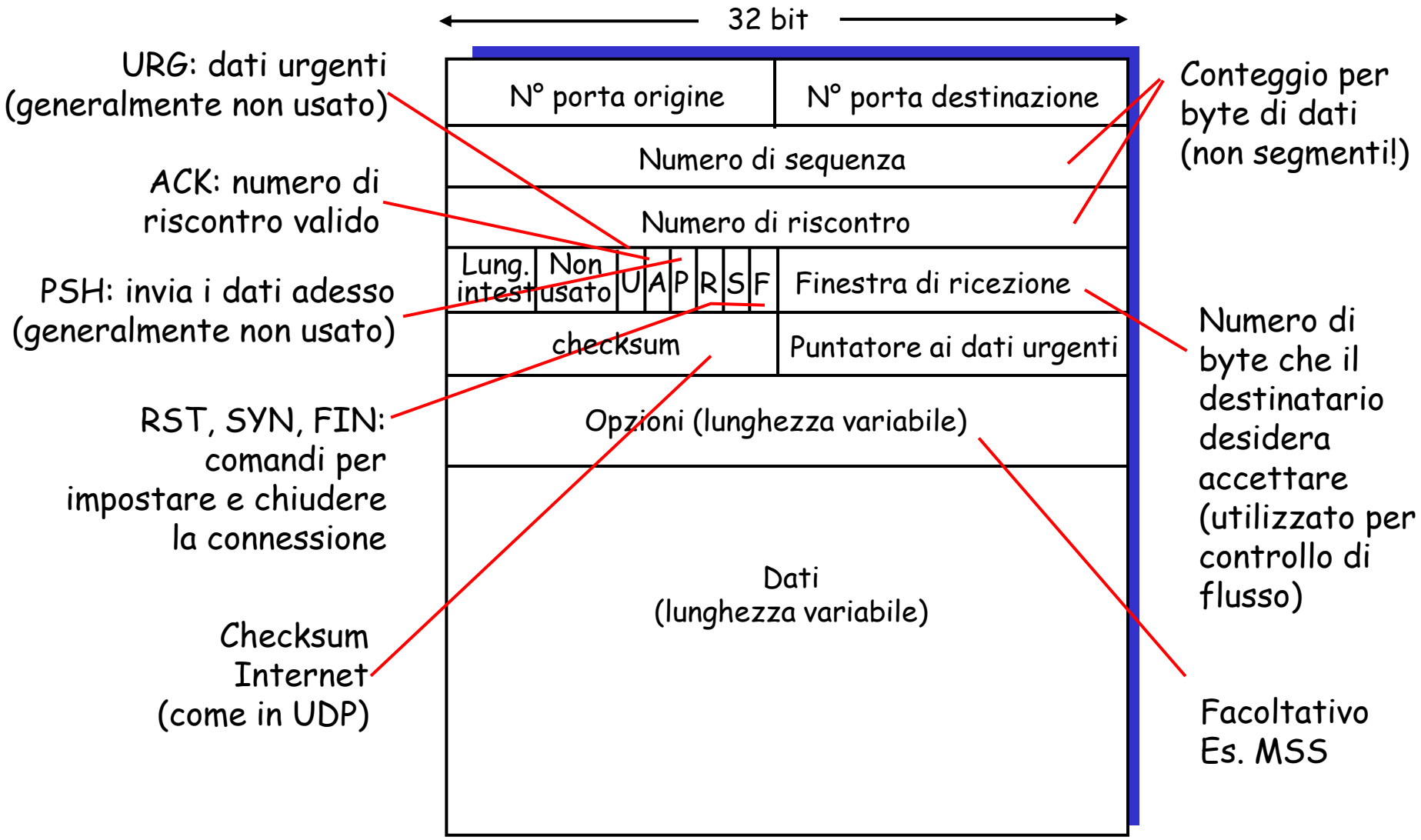
Livello di trasporto: TCP, controllo flusso, controllo congestione

Gaia Maselli
maselli@di.uniroma1.it

Queste slide sono un adattamento delle slide fornite dal libro di testo e pertanto protette da copyright.
All material copyright 1996-2007 J.F Kurose and K.W. Ross, All Rights Reserved

Struttura dei segmenti TCP



Numeri di sequenza e ACK di TCP

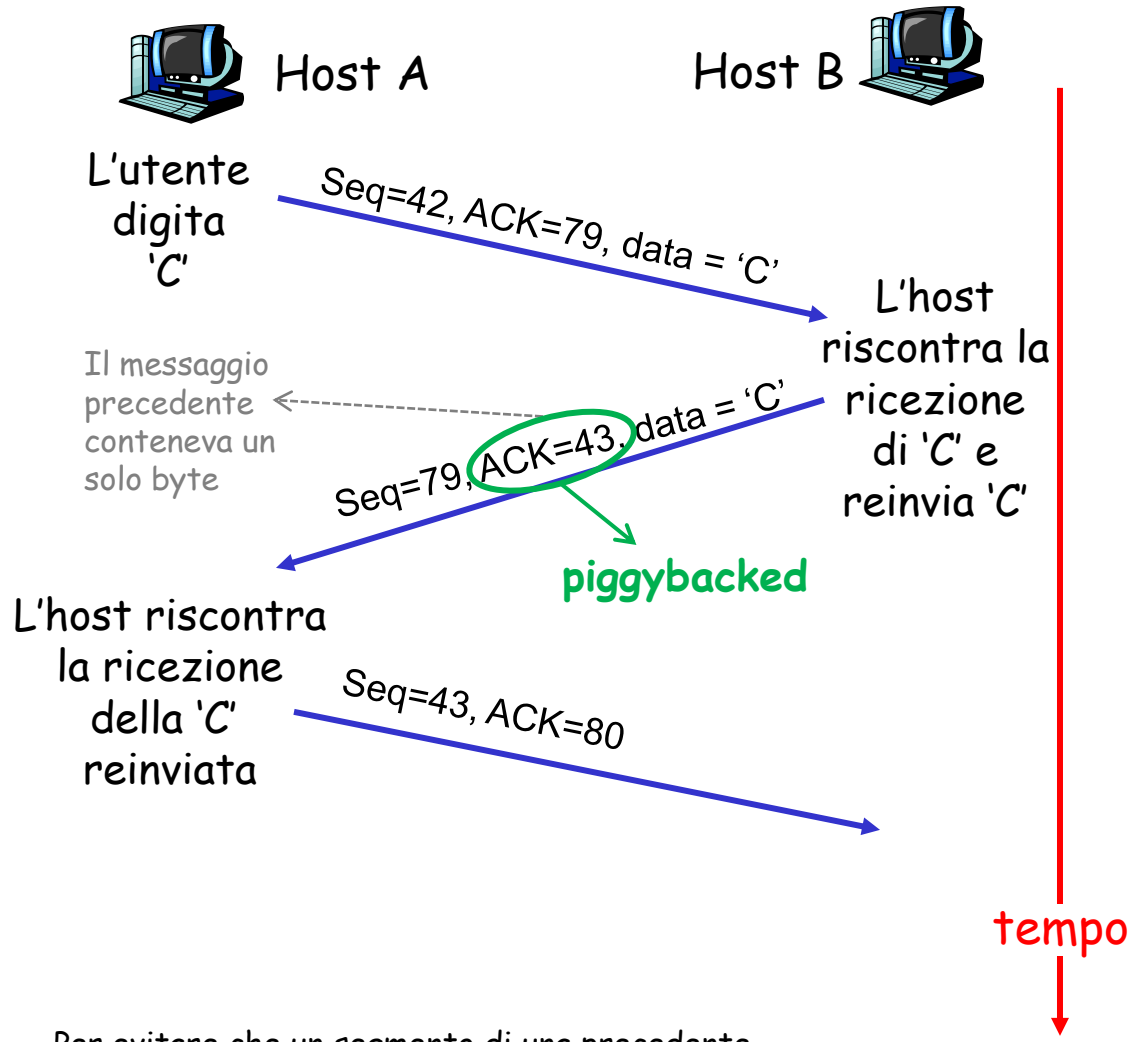
Numeri di sequenza:

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

ACK:

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

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

Meccanismi adottati da TCP

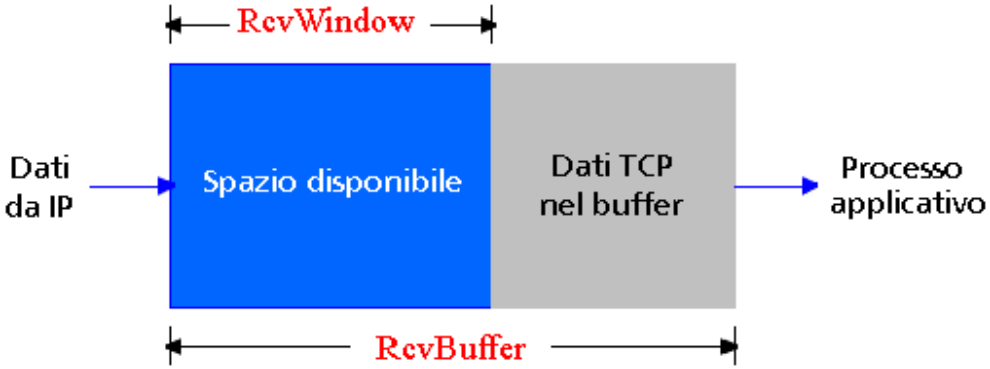
- **Pipeline** (Approccio ibrido tra GBN e Ripetizione Selettiva)
- **Numero di sequenza**: primo byte nel segmento
- **ACK cumulativo** (conferma tutti i byte precedenti a quello indicato) e **delayed** (ritardato, nel caso di arrivo di un pacchetto in sequenza, con precedenti già riscontrati)
- **Timeout** 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 duplicato prima del timeout → ritrasmissione

Livello di trasporto

- Trasporto orientato alla connessione: TCP
  - struttura dei segmenti
  - trasferimento dati affidabile
  - controllo di flusso
  - gestione della connessione
- Principi del controllo di congestione
- Controllo di congestione TCP

TCP: controllo di flusso

- Il lato ricevente della connessione TCP ha un buffer di ricezione:



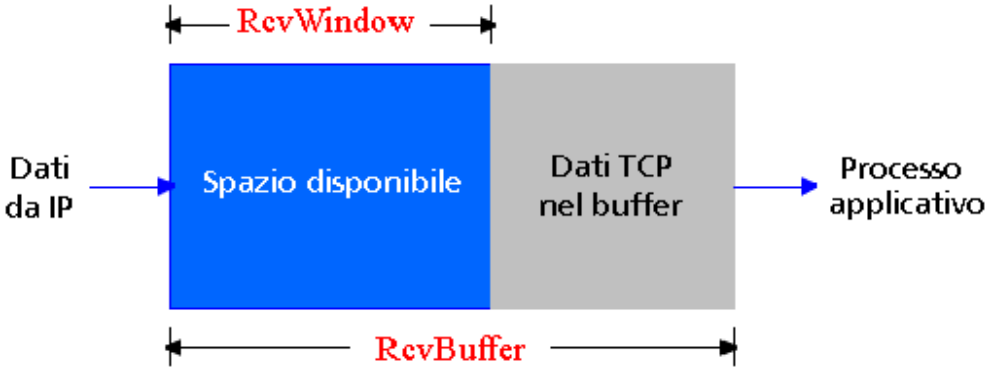
- Il processo applicativo potrebbe essere lento nel leggere i dati dal buffer

Controllo di flusso

Il mittente non vuole sovraccaricare il buffer del destinatario trasmettendo troppi dati, troppo velocemente

- Servizio di corrispondenza delle velocità: la frequenza d'invio deve corrispondere alla frequenza di lettura dell'applicazione ricevente

TCP: funzionamento del controllo di flusso



(supponiamo che il destinatario TCP scarti i segmenti fuori sequenza)

- Spazio disponibile nel buffer

= **RcvWindow**

= **RcvBuffer - [LastByteRcvd - LastByteRead]**

- Il mittente comunica lo spazio disponibile includendo il valore di **RcvWindow** nei segmenti (header TCP)
- Il mittente limita i dati non riscontrati a **RcvWindow**
  - garantisce che il buffer di ricezione non vada in overflow

Livello di trasporto

- Trasporto orientato alla connessione: TCP
  - struttura dei segmenti
  - trasferimento dati affidabile
  - controllo di flusso
  - gestione della connessione
- Principi del controllo di congestione
- Controllo di congestione TCP

Gestione della connessione TCP

Ricordiamo: mittente e destinatario TCP stabiliscono una "connessione" prima di scambiare i segmenti di dati

- inizializzano le variabili TCP:
  - numeri di sequenza
  - buffer, informazioni per il controllo di flusso (per esempio, **RcvWindow**)
- *client:* avvia la connessione
- `Socket clientSocket = new Socket("hostname","portnumber");`
- *server:* contattato dal client
- `Socket connectionSocket = welcomeSocket.accept();`

Handshake a tre vie:

Passo 1: il client invia un segmento SYN al server
- specifica il numero di sequenza iniziale
- nessun dato

Passo 2: il server riceve SYN e risponde con un segmento SYNACK
- il server alloca i buffer
- specifica il numero di sequenza iniziale del server

Passo 3: il client riceve SYNACK e risponde con un segmento ACK, che può contenere dati

Handshake a tre vie

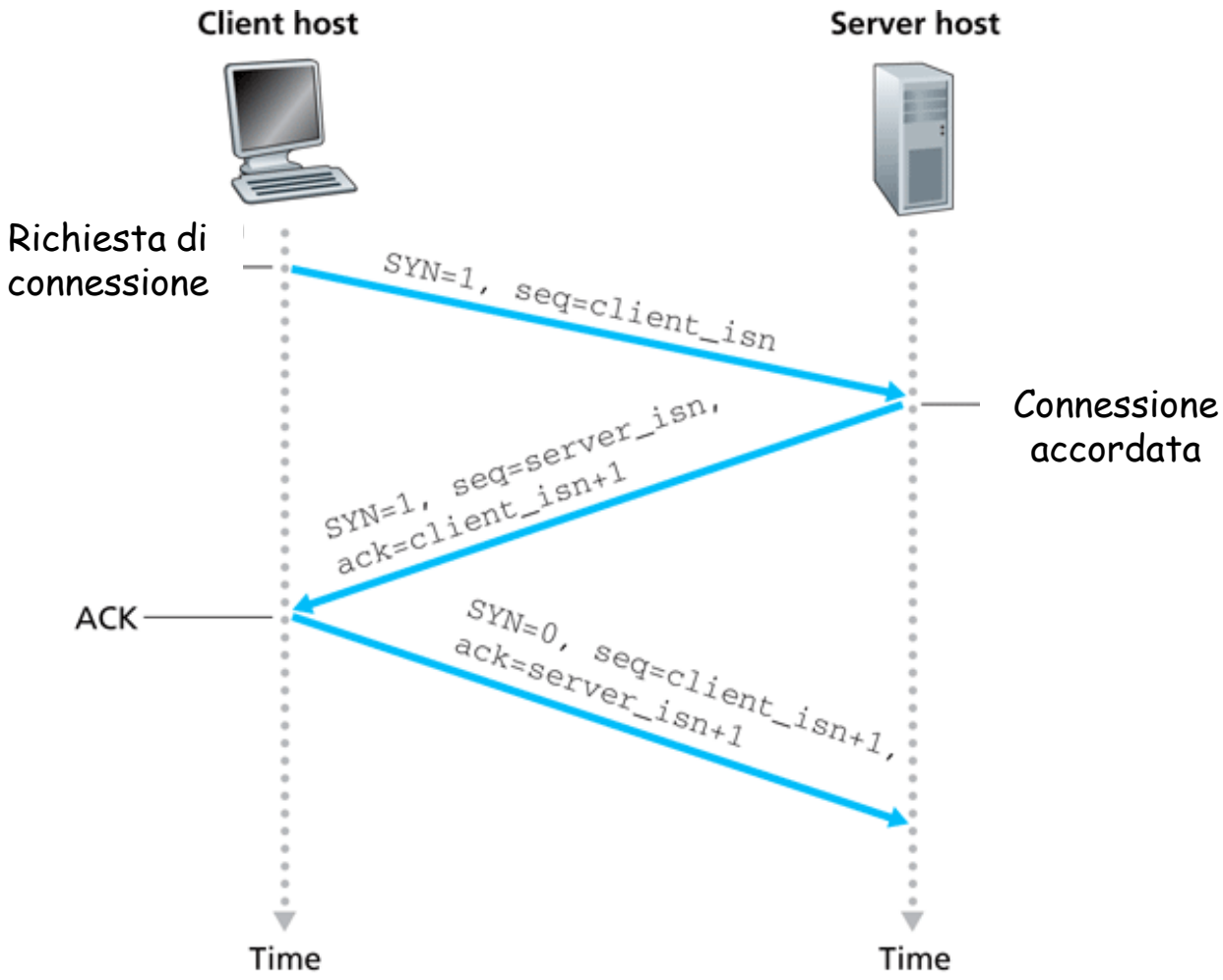


Figure 3.39 ♦ TCP three-way handshake: segment exchange

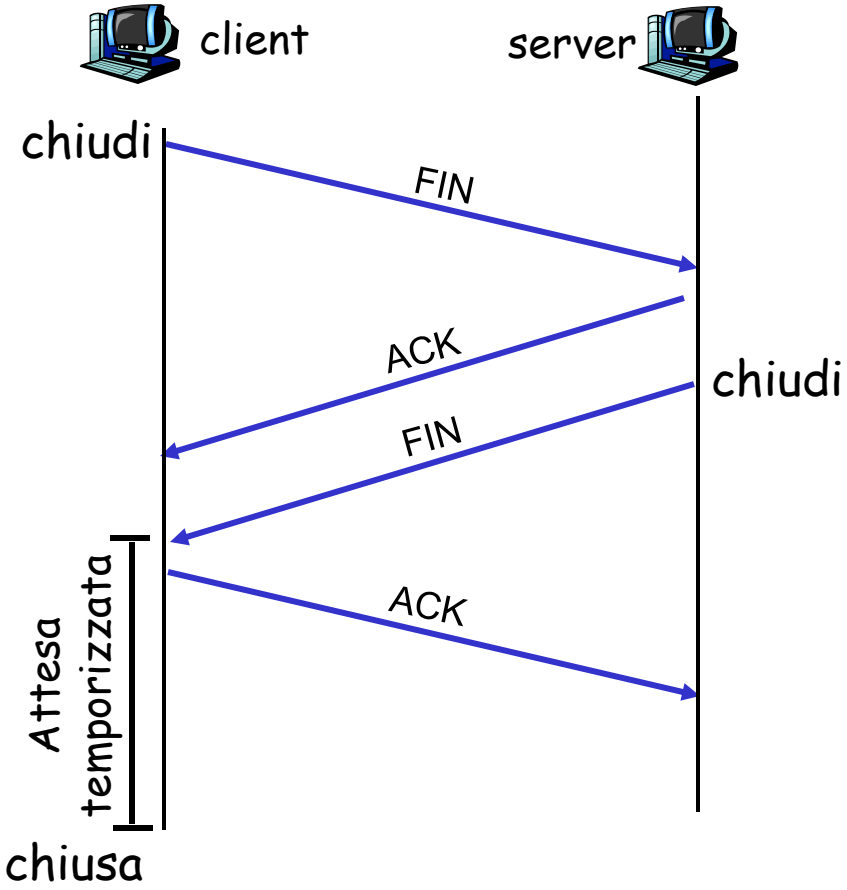
Gestione della connessione TCP (continua)

Chiudere una connessione:

Il client chiude la socket:
`clientSocket.close();`

Passo 1: il client invia un segmento di controllo FIN al server.

Passo 2: il server riceve il segmento FIN e risponde con un ACK. Chiude la connessione e invia un FIN.



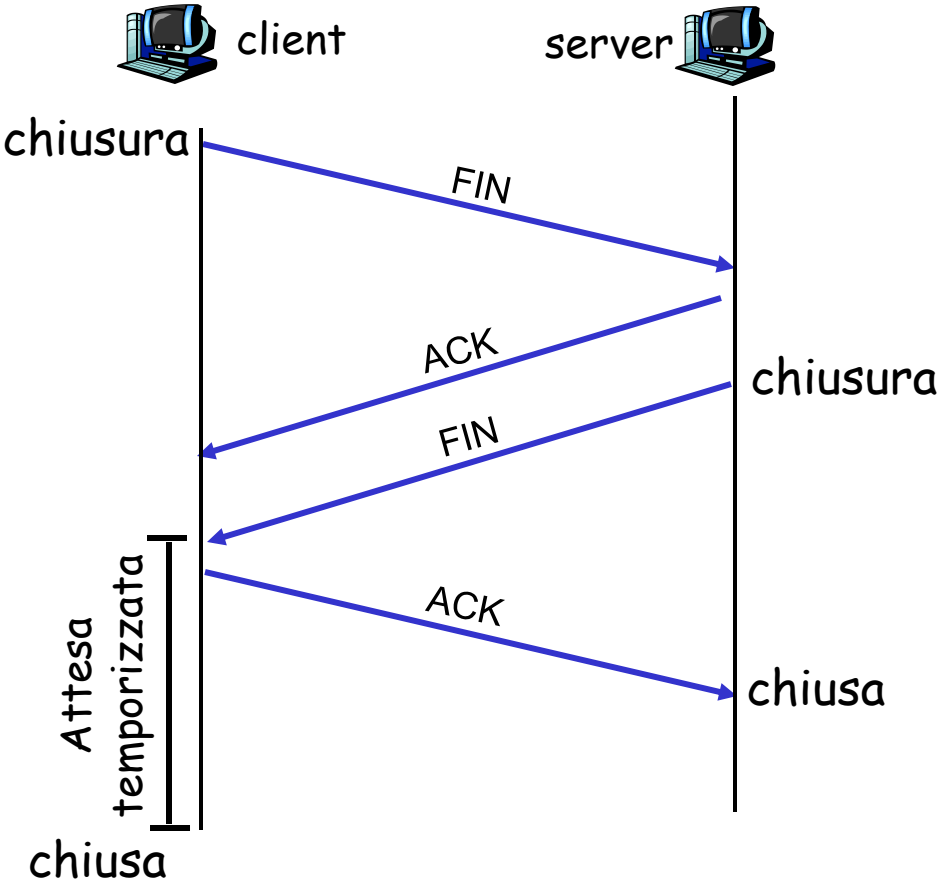
Gestione della connessione TCP (continua)

Passo 3: il client riceve FIN e risponde con un ACK.

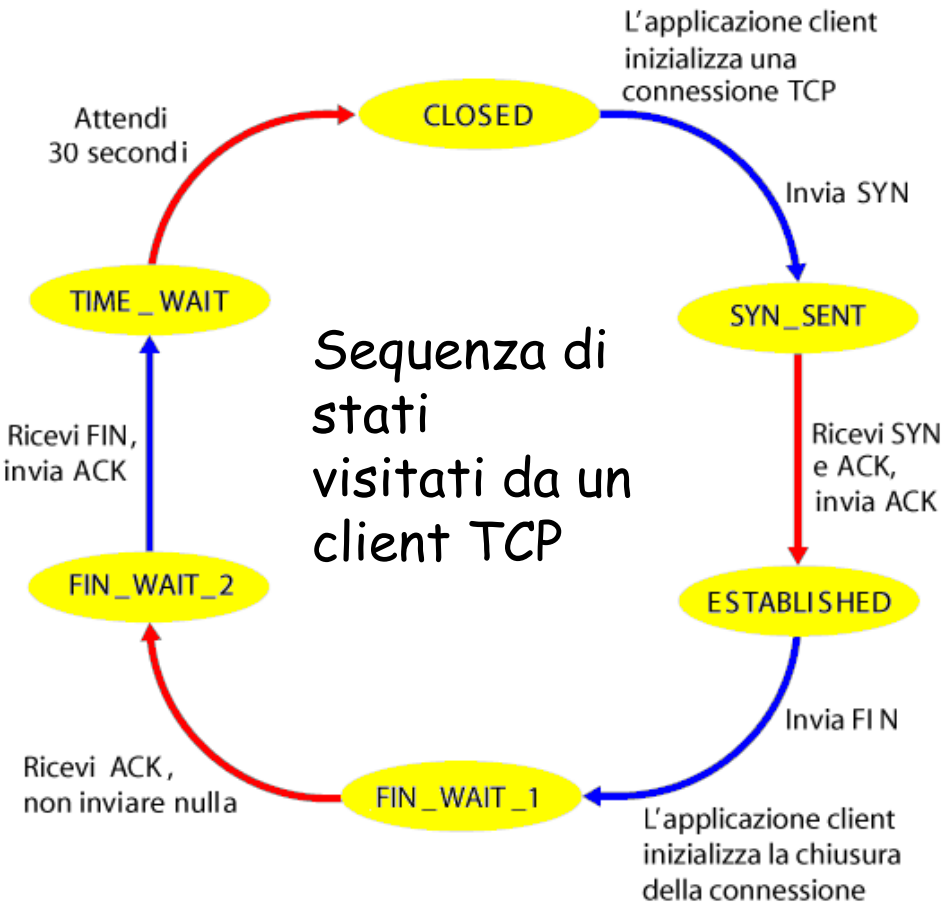
- inizia l'attesa temporizzata - risponde con un ACK ai FIN che riceve

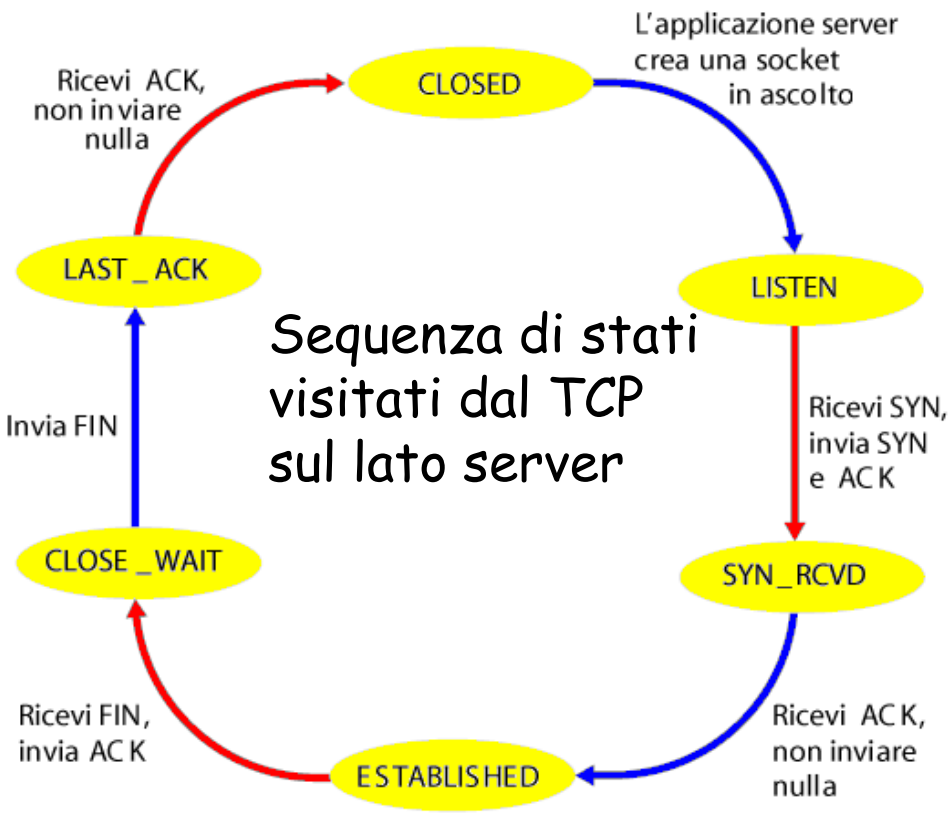
Passo 4: il server riceve un ACK. La connessione viene chiusa.

Nota: con una piccola modifica può gestire segmenti FIN contemporanei.



Gestione della connessione TCP (continua)





Livello di trasporto

- Trasporto orientato alla connessione: TCP
  - struttura dei segmenti
  - trasferimento dati affidabile
  - controllo di flusso
  - gestione della connessione
- Principi del controllo di congestione
- Controllo di congestione TCP

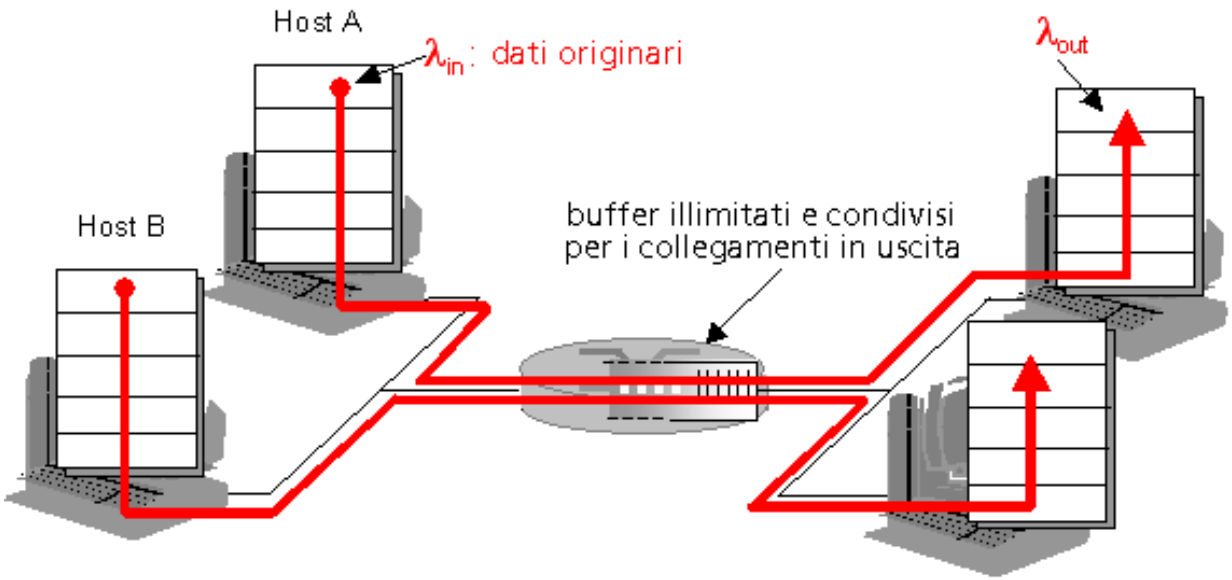
Principi del controllo di congestione

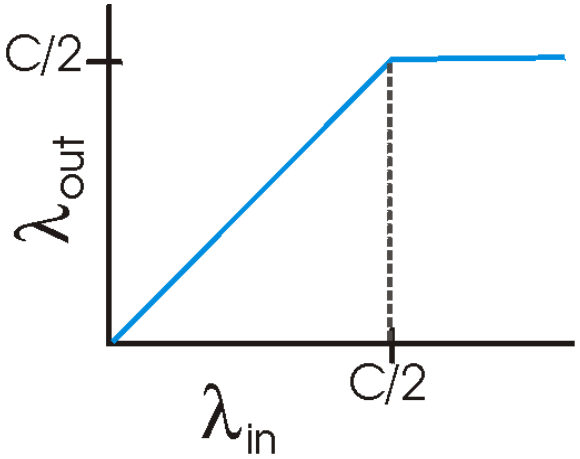
Congestione:

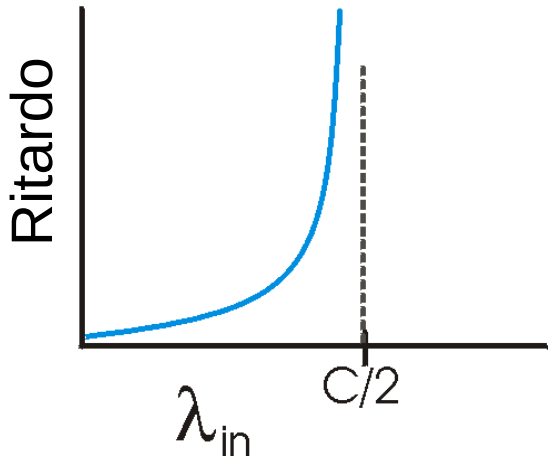
- informalmente: "troppe sorgenti trasmettono troppi dati, a una velocità talmente elevata che la *rete* non è in grado di gestirli"
- differisce dal controllo di flusso!
- sintomi:
  - pacchetti smarriti (overflow nei buffer dei router)
  - lunghi ritardi (accodamento nei buffer dei router)
- tra i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

Cause/costi della congestione: scenario 1

- due mittenti, due destinatari
- un router con buffer illimitati
- nessuna ritrasmissione



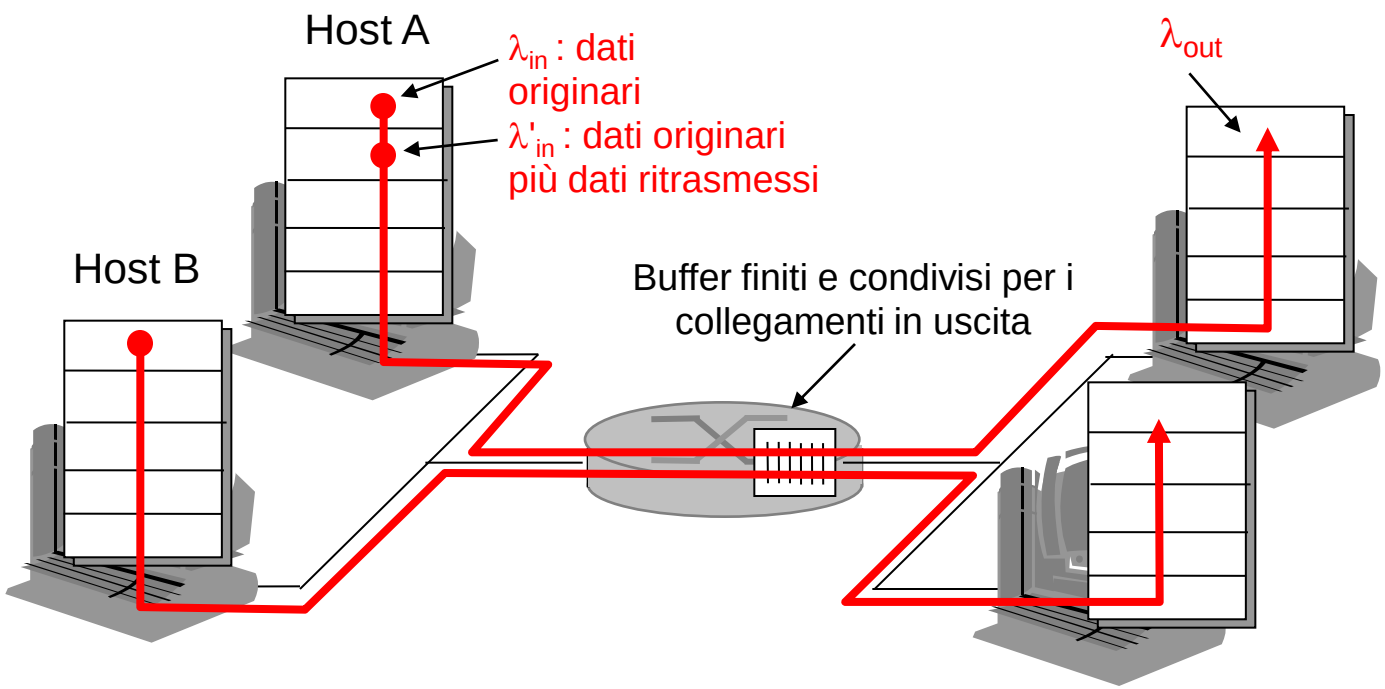




- grandi ritardi se congestionati
- throughput massimo

Cause/costi della congestione: scenario 2

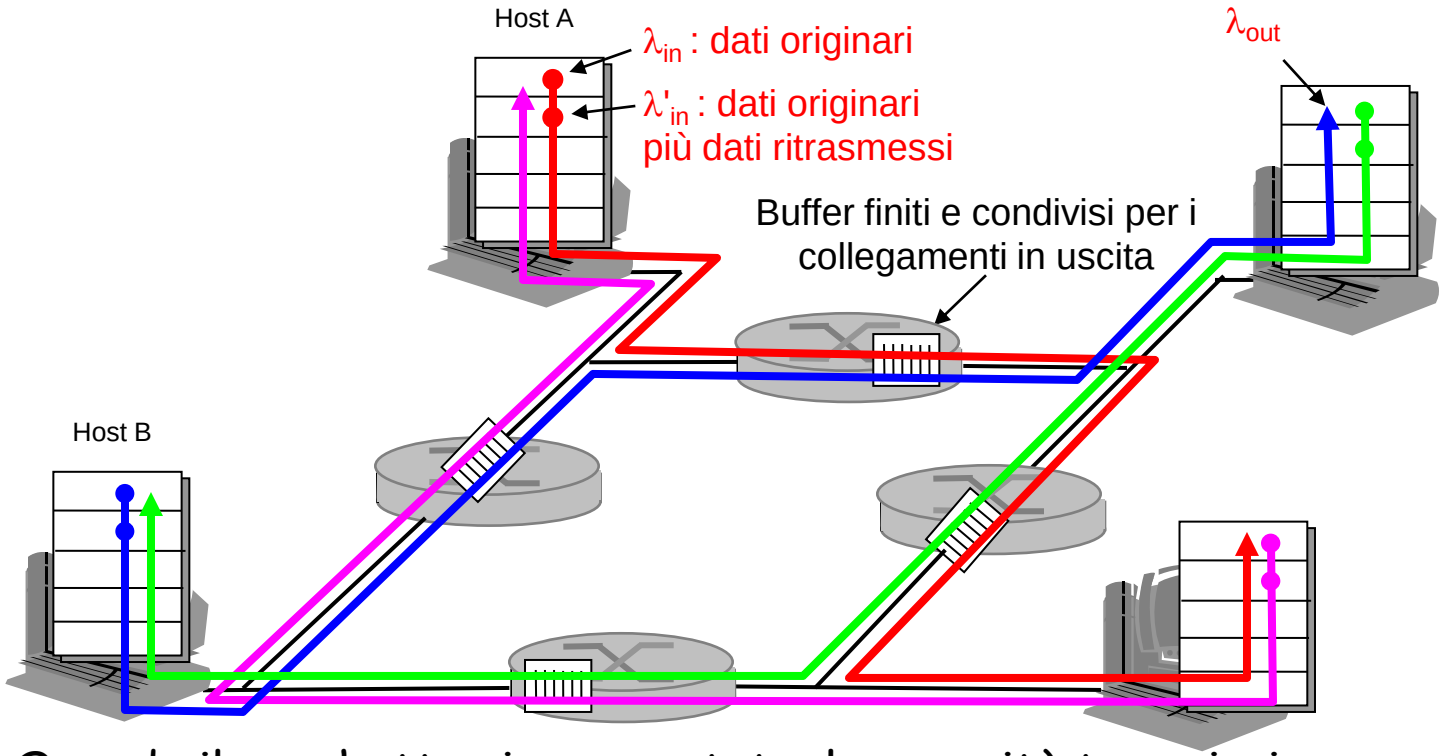
- un router, buffer *finiti*
- il mittente ritrasmette il pacchetto perduto



Cause/costi della congestione: scenario 3

- Quattro mittenti
- Percorsi multihop
- timeout/ritrasmissione

D: Che cosa accade quando λ_{in} e λ'_{in} aumentano?



- Quando il pacchetto viene scartato, la capacità trasmissiva utilizzata sui collegamenti di upstream per instradare il pacchetto risulta sprecata!

Approcci al 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

Livello di trasporto

- Trasporto orientato alla connessione: TCP
  - struttura dei segmenti
  - trasferimento dati affidabile
  - controllo di flusso
  - gestione della connessione
- Principi del controllo di congestione
- Controllo di congestione TCP

Problematiche

1. Come può il mittente limitare la frequenza di invio del traffico sulla propria connessione?
2. Come percepisce il mittente la congestione sul percorso che porta alla destinazione?
3. Quale algoritmo dovrebbe essere usato per variare la frequenza di invio in funzione della congestione end-to-end?

1. Finestra di congestione

- La frequenza di invio viene controllata dalla ***finestra di congestione***
- `CongWin`
- La quantità di dati non riscontrati non può eccedere il minimo tra i valori `CongWin` (e `RcvWindow`)

- Il mittente limita la trasmissione:

  `LastByteSent-LastByteAcked`

  $\leq$ `CongWin`

- Approssimativamente:

$$\text{Frequenza d'invio} = \frac{\text{CongWin}}{\text{RTT}} \text{ byte/sec}$$

- **`CongWin`** è una funzione dinamica della congestione percepita

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

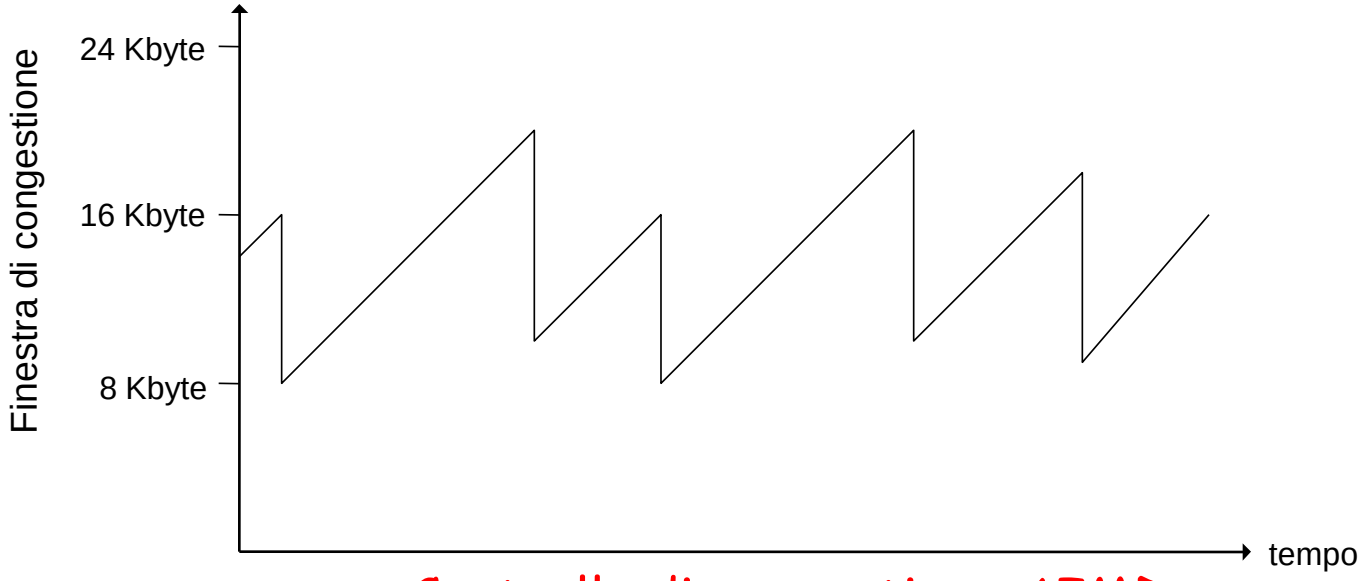
- Algoritmo di controllo della congestione TCP
  1. Incremento additivo e decremento moltiplicativo (AIMD)
  2. Slow start (partenza lenta)
  3. Reazione ai timeout

Controllo di congestione TCP: incremento additivo e decremento moltiplicativo (AIMD)

Approccio: aumenta il tasso trasmissivo sondando la rete, fino a quando non si verifica una perdita

Incremento additivo: aumenta **CongWin** di 1 MSS a ogni RTT in assenza di eventi di perdita

Decremento moltiplicativo: riduce a metà **CongWin** dopo un evento di perdita



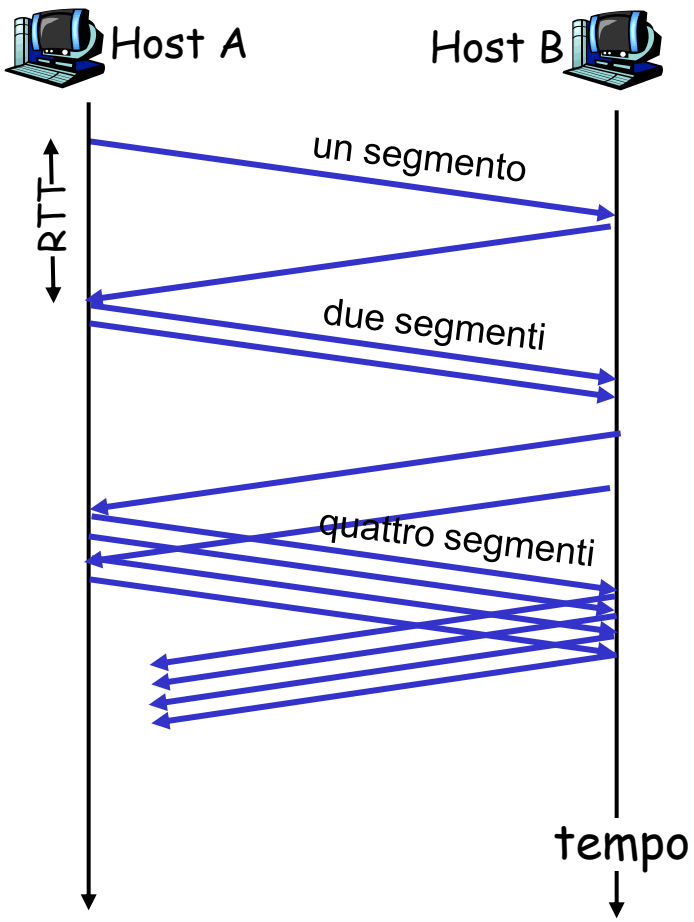
Controllo di congestione AIMD

Partenza lenta

- Quando si stabilisce una connessione, **`CongWin`** = 1 MSS
  - Esempio: MSS = 500 byte RTT = 200 msec
  - Frequenza iniziale = 20 kbps
- La larghezza di banda disponibile potrebbe essere >> MSS/RTT
  - Consente di raggiungere rapidamente una frequenza d'invio significativa
- Quando inizia la connessione, la frequenza aumenta in modo esponenziale, fino a quando non si verifica un evento di perdita

Partenza lenta (continua)

- Quando inizia la connessione, la frequenza aumenta in modo esponenziale, fino a quando non si verifica un evento di perdita:
  - raddoppia **`CongWin`** a ogni RTT
  - ciò avviene incrementando **`CongWin`** per ogni ACK ricevuto
- Riassunto: la frequenza iniziale è lenta, ma poi cresce in modo esponenziale



Affinamento

- Dopo 3 ACK duplicati (TCP Reno):
  - **`CongWin`** è ridotto a metà (*fast recovery*)
  - la finestra poi cresce linearmente
- Ma dopo un evento di timeout:
  - **`CongWin`** è impostata a 1 MSS (*slow start*);
  - poi la finestra cresce in modo esponenziale
  - fino a un valore di **Threshold** (soglia), poi cresce linearmente (*congestion avoidance*)

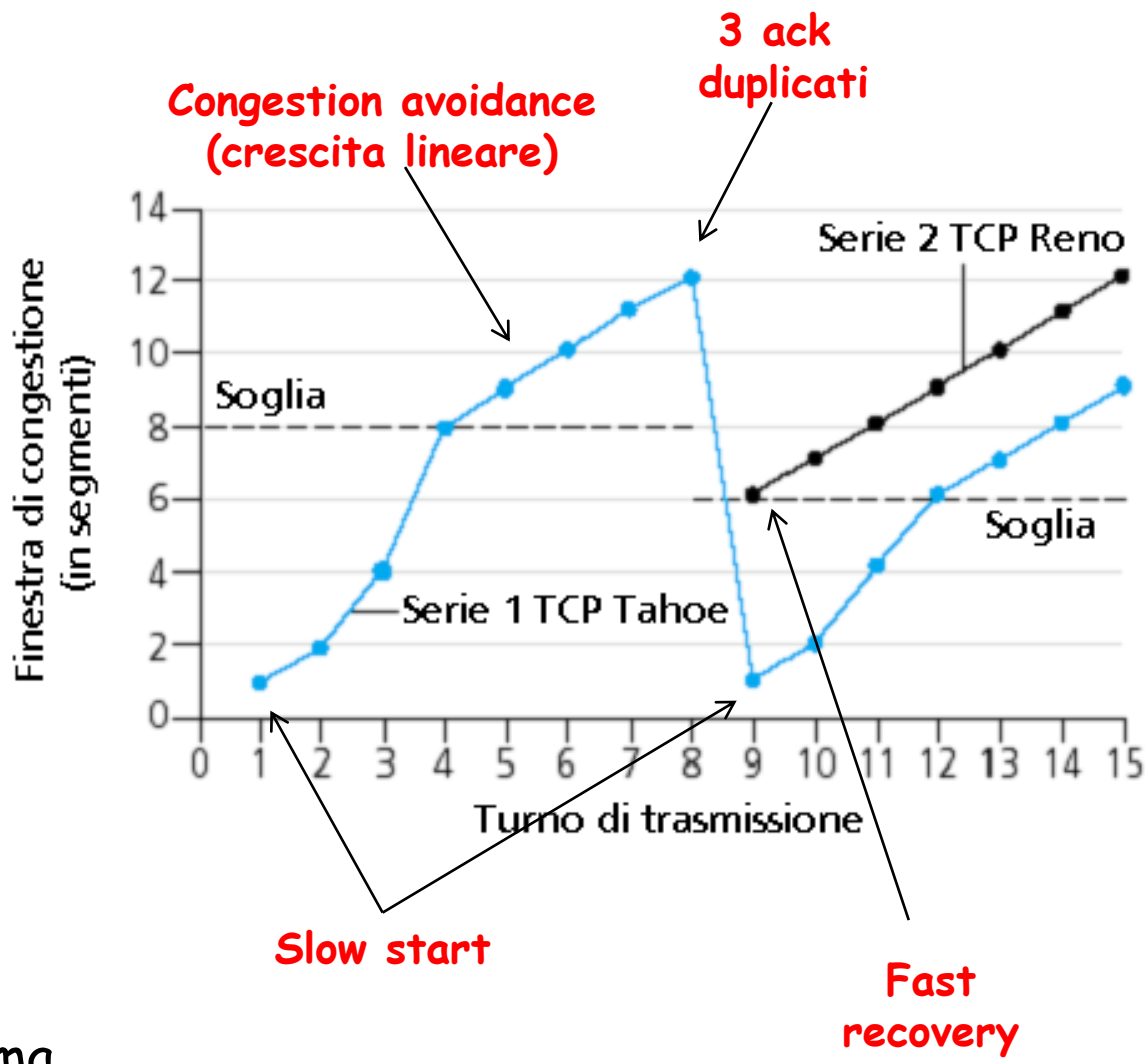
Filosofia:

- 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)

Affinamento (continua)

D: Quando l'incremento esponenziale dovrà diventare lineare?

R: Quando **CongWin** raggiunge 1/2 del suo valore prima del timeout.



Implementazione:

- Soglia variabile
- In caso di evento di perdita, la soglia è impostata a 1/2 di CongWin, appena prima dell'evento di perdita

Riassunto: il controllo della congestione TCP

- Quando **`CongWin`** è sotto la soglia (**`Threshold`**), il mittente è nella fase di partenza lenta; la finestra cresce in modo esponenziale.
- Quando **`CongWin`** è sopra la soglia, il mittente è nella fase di congestion avoidance; la finestra cresce in modo lineare.
- Quando si verificano tre ACK duplicati, il valore di **`Threshold`** viene impostato a **`CongWin/2`** e **`CongWin`** viene impostata al valore di **`Threshold`**.
- Quando si verifica un timeout, il valore di **`Threshold`** viene impostato a **`CongWin/2`** e **`CongWin`** è impostata a 1 MSS.

Controllo di congestione del mittente TCP

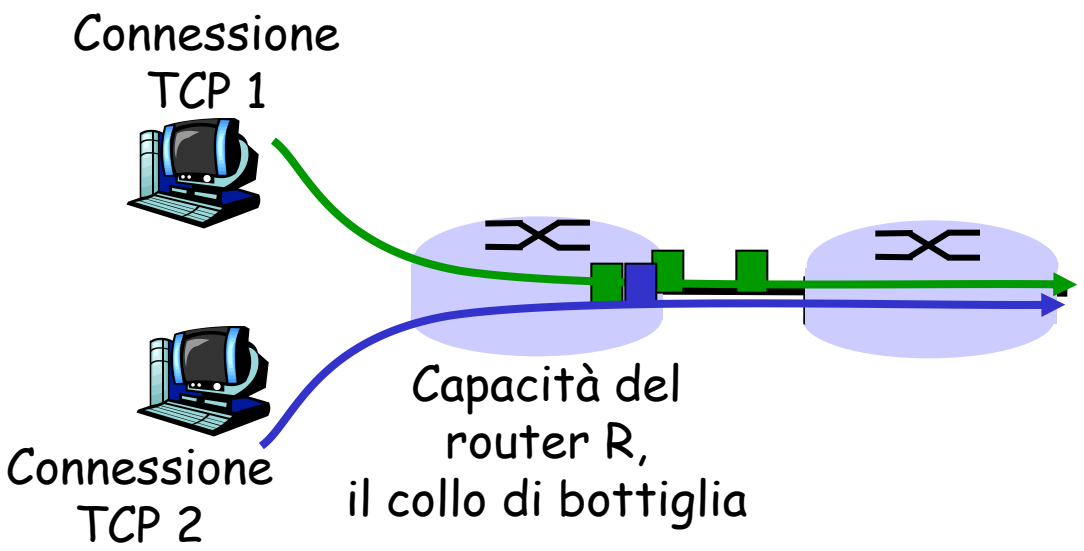
Stato	Evento	Azione del mittente TCP	Commenti
Slow Start (SS)	Ricezione di ACK per dati precedente-mente non riscontrati	CongWin = CongWin + MSS, If (CongWin > Threshold) imposta lo stato a "Congestion Avoidance"	CongWin raddoppia a ogni RTT
Congestion Avoidance (CA)	Ricezione di ACK per dati precedente-mente non riscontrati	CongWin = CongWin + MSS * (MSS/CongWin)	Incremento additivo: CongWin aumenta di 1 MSS a ogni RTT
SS o CA	Rilevato un evento di perdita da tre ACK duplicati	Threshold = CongWin/2, CongWin = Threshold, imposta lo stato a "Congestion Avoidance"	Ripristino rapido con il decremento moltiplicativo. CongWin non sarà mai minore di 1 MSS
SS o CA	Timeout	Threshold = CongWin/2, CongWin = 1 MSS, imposta lo stato a "Slow Start"	Entra nello stato "Slow Start"
SS o CA	ACK duplicato	Incrementa il conteggio degli ACK duplicati per il segmento in corso di riscontro	CongWin e Threshold non variano

Throughput TCP

- Qual è il throughput medio di TCP in funzione della dimensione della finestra e di RTT?
  - Ignoriamo le fasi di partenza lenta
- Sia W la dimensione della finestra quando si verifica una perdita.
- Quando la finestra è W, il throughput è W/RTT
- Subito dopo la perdita, la finestra si riduce a W/2, il throughput a W/2RTT.
- Throughout medio: 0,75 W/RTT

Equità di TCP

Equità: se K sessioni TCP condividono lo stesso collegamento con ampiezza di banda R, che è un collo di bottiglia per il sistema, ogni sessione dovrà avere una frequenza trasmissiva media pari a R/K.



Equità

Equità e UDP

- Le applicazioni multimediali spesso non usano TCP
  - non vogliono che il loro tasso trasmissivo venga ridotto dal controllo di congestione
- Utilizzano UDP:
  - immettono audio/video a frequenza costante, tollerano la perdita di pacchetti
- Area di ricerca: TCP friendly

Equità e connessioni TCP in parallelo

- Nulla può impedire a un'applicazione di aprire connessioni in parallelo tra 2 host
- I browser web lo fanno
- Esempio: un collegamento di frequenza R che supporta 9 connessioni;
  - Se una nuova applicazione chiede una connessione TCP, ottiene una frequenza trasmissiva pari a R/10
  - Se la nuova applicazione chiede 11 connessioni TCP, ottiene una frequenza trasmissiva pari a R/2 !

Riassunto

- principi alla base dei servizi del livello di trasporto:
  - multiplexing, demultiplexing
  - trasferimento dati affidabile
  - controllo di flusso
  - controllo di congestione
- implementazione in Internet
  - UDP
  - TCP

Prossimamente:

- lasciare la "periferia" della rete (livelli di applicazione e di trasporto)
- entrare nel "cuore" della rete