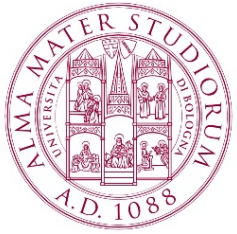


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Controllo di flusso in TCP

Controllo di flusso

- Le velocità di trasmettitore e ricevitore possono essere molto diverse
  - Il trasmettitore non deve saturare il ricevitore
- Analogamente al caso dello strato 2
  - Si utilizza un meccanismo a finestra scorrevole
- Quale deve essere la dimensione della finestra?
  - Deve essere dimensionata in modo congruente con le memoria di trasmissione e ricezione
  - Il trasmettitore conosce le dimensioni della propria memoria ma non conosce quelle delle memoria di ricezione
- Il ricevitore deve comunicare al trasmettitore le dimensioni della sua memoria di ricezione
  - Nell'intestazione del pacchetto TCP è contenuto il campo advertised window (AW)

Finestra di trasmissione e di ricezione

- Ricevitore e trasmettitore possono memorizzare segmenti
  - W_T = finestra di trasmissione
    - Insieme di segmenti inviabili da trasmettitore senza ricevere conferme di ricezione
  - W_R = finestra di ricezione
    - Insieme di segmenti memorizzabili fuori sequenza al ricevitore
- M = spazio di numerazione (in TCP 2^{32})
- Se $W_T = W_R$ allora deve essere $W_T + W_R \leq M$
  - $W_T \leq 2^{31}$
  - $W_R \leq 2^{31}$

Unità di misura di W

- TCP adotta la numerazione sequenziale dei dati trasmessi per byte
- W può essere misurata
  - In byte (w)
  - In numero di segmenti (W)
    - In questo caso si deve indicare quale lunghezza si assume per i segmenti
- Normalmente W viene misurata in segmenti di dimensione massima (full sized segments)

$$W\ MSS = w$$

Dimensionamento di W

- W (o w) viene messa a punto dinamicamente sulla base di informazioni
  - Provenienti dal ricevente (advertised window o AW)
    - Comunicate in modo esplicito al trasmettitore
  - Correlate allo stato di congestione della rete (congestion window o CW)
    - Desunte dal trasmettitore in base al comportamento del canale
- AW e CW sono funzione del tempo
- In un generico istante di tempo la connessione imposta

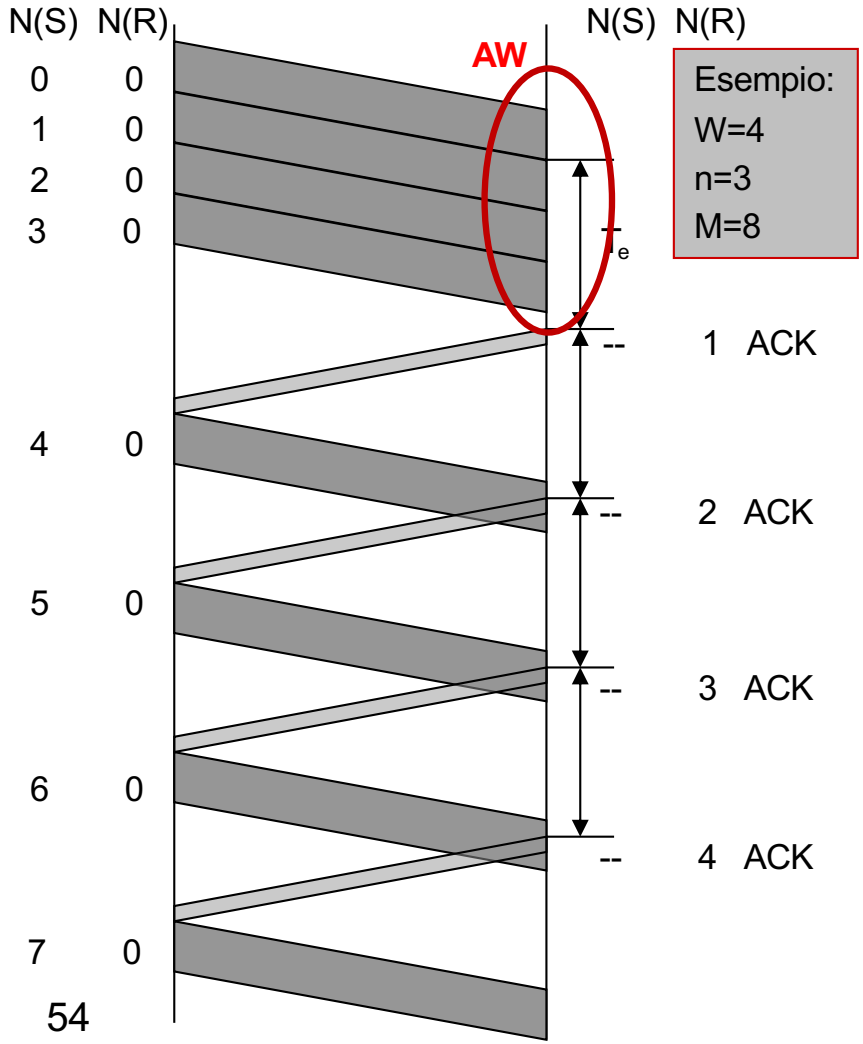
$$W = \min[AW, CW]$$

Motivazioni

- Il protocollo a finestra ha l'obiettivo di garantire il controllo di flusso
  - Impedire che una sorgente congestioni il sistema di comunicazione impedendo il corretto trasferimento dei dati
- Nello strato 4 un'errato dimensionamento di W può congestionare:
  - Il ricevitore
    - Per prevenire questo viene utilizzata AW
  - I nodi intermedi della rete
    - Per prevenire questo viene utilizzata CW
- Impostare la finestra al minimo fra AW e CW rappresenta la soluzione più conservativa

Controllo di flusso: ruolo di AW

- Riprendiamo l'esempio di trasmettitore veloce e ricevitore lento



- Con AW il ricevitore indica al trasmettitore la dimensione del suo buffer
- In questo modo si è certi che possa ricevere l'intera prima finestra di dati

Attuazione del controllo con AW

- Come un ricevitore lento blocca un trasmettitore più veloce
  - Il buffer di ricezione si riempie
    - AW = 0
  - W = 0
    - Il trasmettitore blocca la trasmissione
- Ripresa della trasmissione
  - Il processo ricevente legge dal buffer
    - AW > 0
  - Arrivano gli ACK
    - Si libera il buffer di trasmissione
    - Viene ricevuto AW > 0
  - Il processo trasmittente ricomincia a trasmettere

Deadlock

Trasmittente

- Invia messaggi
- Riceve un messaggio con AW=0
- Sospende l'invio dei dati

Ricevente

- Il buffer di ricezione si riempie
- Invia un messaggio con AW=0
- Non ha altri messaggi da trasmettere

A questo punto il protocollo è in deadlock

- Il trasmittente non può inviare dati poiché AW=0
- Il ricevente non ha dati da inviare quindi non ha modo di comunicare AW>0

TCP prevede che sia sempre possibile inviare un segmento di 1 byte anche se AW=0

Persist timer e window probe

- Il trasmettitore riceve ACK fino al byte X ma contenente AW=0
- Fa partire il "persist timer"
  - PT =1,5 sec per un normale collegamento LAN
  - Quando PT scade si invia un segmento di 1 byte
    - seqN=X+1
  - Il ricevitore deve rispondere
    - Invia ACK con ackN=X+2 e AW>0
      - La trasmissione riprende
    - Invia ACK con ackN=X+1 e AW=0
      - Non ha spazio nel buffer di ricezione perciò non può ricevere il byte X+1
      - PT = 2PT e si ricomincia ad attendere
    - Il massimo valore di PT viene fissato a 60 sec

Problemi

- Questo meccanismo può dare luogo a inefficienze in casi particolari
  - Trasmissione "lenta"
    - Ad esempio l'applicazione trasmette un carattere per volta
  - Ricezione "lenta"
    - Ad esempio l'applicazione è lenta ad accettare i dati, legge un byte alla volta e comunica una dimensione di finestra molto piccola

Ricevitore lento: Silly window syndrom

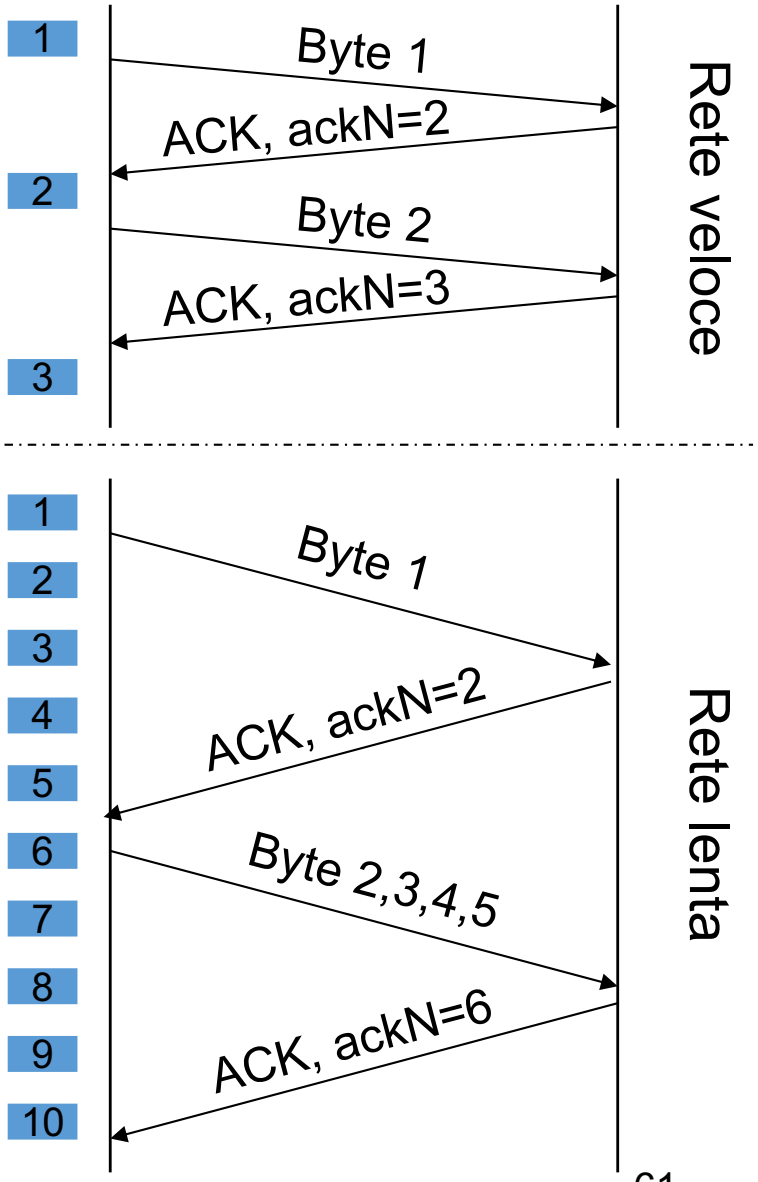
- In caso di applicazione ricevente lenta
  - Il buffer di ricezione si riempie $\Rightarrow$ AW=0
  - L'applicazione legge un byte e trasmette AW=1
  - Il trasmettitore manda un segmento di un byte
  - Il buffer di ricezione si riempie $\Rightarrow$ AW=0
- Viene trasmesso un byte alla volta
  - Qualunque sia la velocità della rete il throughput risulta dell'ordine di un byte per RTT
- Soluzione:
  - Il ricevitore non può aumentare AW a meno che
    - Il nuovo valore di AW sia almeno pari a MSS
    - Il nuovo valore di AW sia almeno pari a metà del buffer di ricezione

Tramettitore lento: Algoritmo di Nagle

- Applicazione trasmittente lenta
  - Passa a TCP un carattere per volta (Telnet, Rlogin)
  - Vengono trasmessi dei "tinygram", ossia segmenti di un solo byte
    - Ciascun byte richiede almeno 40 byte di header più 40 byte di ACK
    - L'overhead per byte risulta essere molto elevato (efficienza 1/81)
- Si deve aumentare la dimensione del messaggio
- Soluzione (algoritmo di Nagle)
  - Il trasmettitore trasmette un nuovo segmento solo se è vera una delle seguenti condizioni
    - Il segmento è di dimensioni pari a MSS
    - Il segmento è di dimensioni almeno pari a metà del valore di AW
    - Non vi sono ACK pendenti ed è possibile trasmettere tutto ciò che è in attesa nel buffer di trasmissione

Effetto dell'algoritmo di Nagle

- Si può avere un solo segmento pendente per il quale non si è ricevuto ACK
  - Più veloci arrivano gli ACK più velocemente si trasmette
- Ethernet
  - RTT tipico dell'ordine di 10 ms
    - Oltre 60 caratteri al sec.
    - Un carattere per tinygram
- Rete geografica
  - RTT dell'ordine dei sec.
    - Numerosi caratteri per tinygram



Disabilitare l'algoritmo di Nagle

- L'algoritmo di Nagle tende a ritardare i dati nel buffer di trasmissione
  - Per alcune applicazioni questo potrebbe non essere accettabile
    - X-windows: i movimenti del mouse devono essere trasmessi in tempo reale
- È possibile disabilitare l'algoritmo

Controllo di congestione

- W è limitato superiormente da AW o da CW
- Come viene determinata CW?
- TCP cerca di adattare la dimensione della finestra alle condizioni di congestione della rete
- Idea base:
  - se si verifica congestione in rete si rallenta la trasmissione
    - Quando si verifica una perdita si riduce W
    - Quando gli ACK arrivano correttamente W viene aumentata

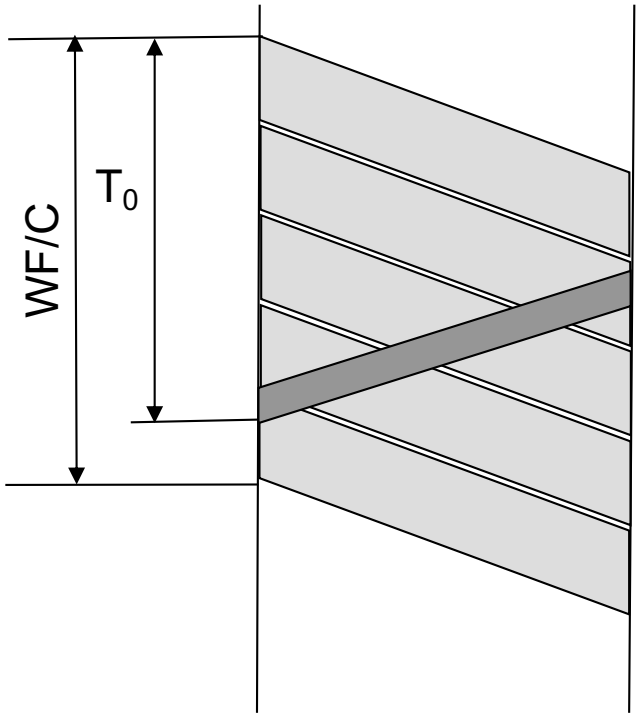
Efficienza

- Abbiamo già visto che l'efficienza ottima in assenza di errori richiede

$$W F \geq C T_0$$

- Nella letteratura sul TCP si fa solitamente riferimento al cosiddetto ***prodotto banda-ritardo***
- Di fatto il prodotto banda-ritardo è una stima veloce di $C T_0$

- Questo è possibile solo se AW non pone un vincolo più stringente



Esempio

- Caso di studio:
  - TCP A invia dati a TCP B
  - Capacità del canale B = 10 Mbit/s
  - Ritardo di propagazione T = 10 ms
  - Round trip time RTT = 20 ms
  - Tempo di elaborazione trascurabile
  - MSS = 1000 byte = 8000 bit

- Con riferimento alla slide precedente

T0 = MSS/B+RTT = 20.8 ms

MSS/B = 0.8 ms

W = 26 segmenti di dimensione MSS

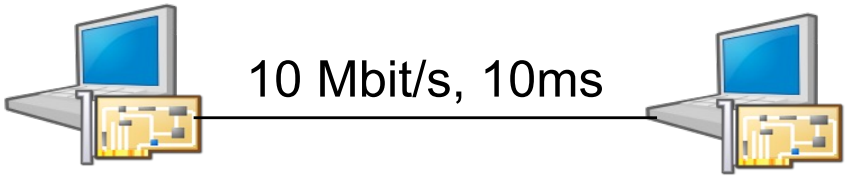
- Prodotto banda ritardo

$$B\ RTT = 20\ 10^{-3}\ 10\ 10^{6} = 200\ 10^{3}\ bit$$

  - Pertanto

$$B\ RTT\ /\ MSS = 25\ segmenti$$

$$Wid \geq B\ RTT\ /\ MSS$$



CW ideale

- Avendo a disposizione una banda B (byte/sec)
  - Il massimo throughput si ottiene quando il protocollo a finestra non limita la velocità di scambio dei dati

$$w_{id} = RTT*B$$

$$W_{id} = w_{id}/MSS$$

- In questo caso si utilizza al 100% la capacità disponibile nella tratta trasmettitore/ricevitore
  - Se $w < w_{id}$: si spreca banda
  - Se $w > w_{id}$: è necessario accodare nei router intermedi
    - cresce il ritardo e potenzialmente anche la perdita
- Il massimo throughput (byte/sec) vale circa:

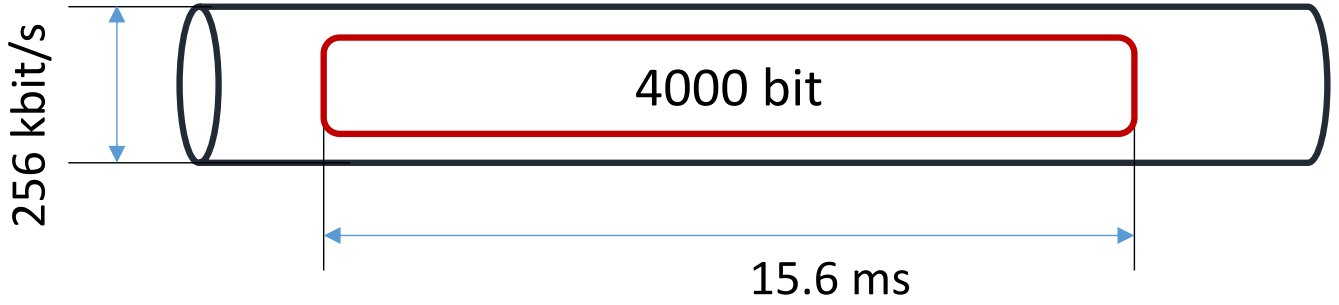
$$S = w/RTT$$

Nel mondo reale : la rete!

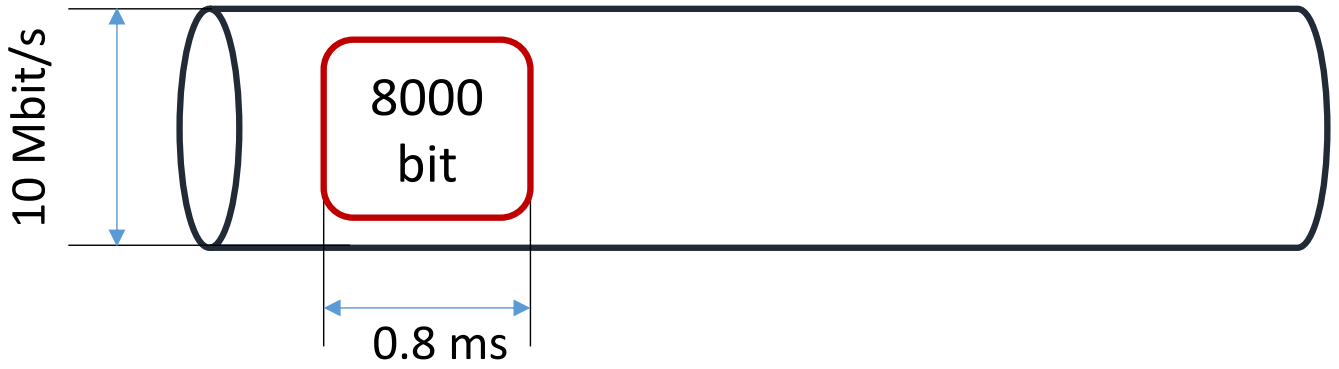
- Al momento dell'instaurazione della connessione:
  - La banda disponibile B è incognita
  - Il percorso che verrà seguito da dati e quindi il RTT è incognito
- Durante il trasferimento dei dati
  - Il canale di comunicazione è condiviso con altri utenti (rete a pacchetto) quindi la banda realmente disponibile varia continuamente
  - La rete può modificare il percorso dei dati pertanto il RTT può modificarsi
  - Lungo il percorso i pacchetti vengono accodati nei nodi di rete quindi il ritardo di propagazione può modificarsi anche segmento per segmento
- In conclusione … il TCP dovrebbe poter «indovinare» i valori di banda e ritardo per poter dimensionare correttamente CW

La rappresentazione di Jackobson

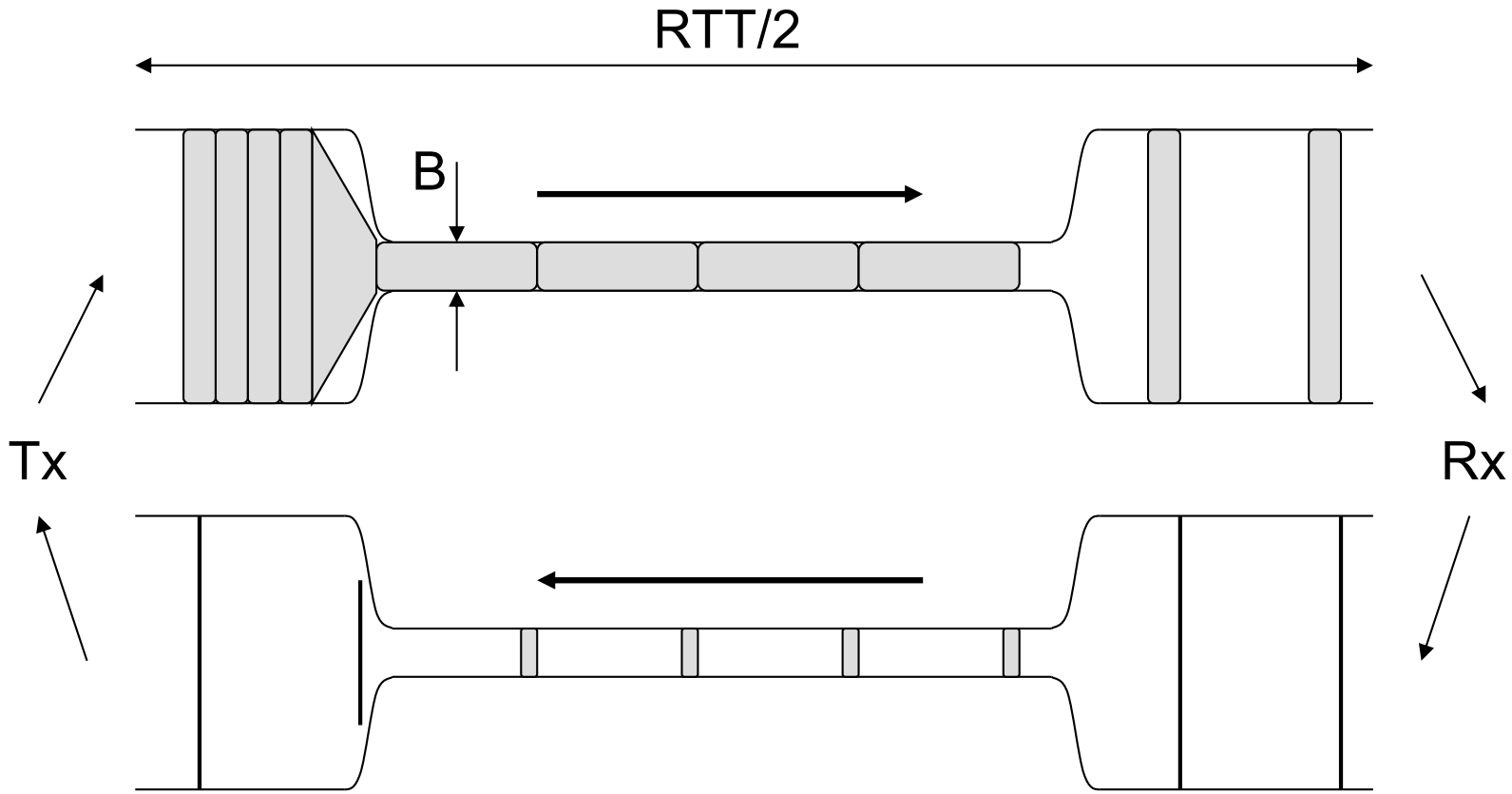
- Diagramma bidimensionale
  - Orizzontalmente si indica il tempo
  - Verticalmente di indica la banda disponibile
    - Esempio: MSS=4000bit, B= 256Kbit/s



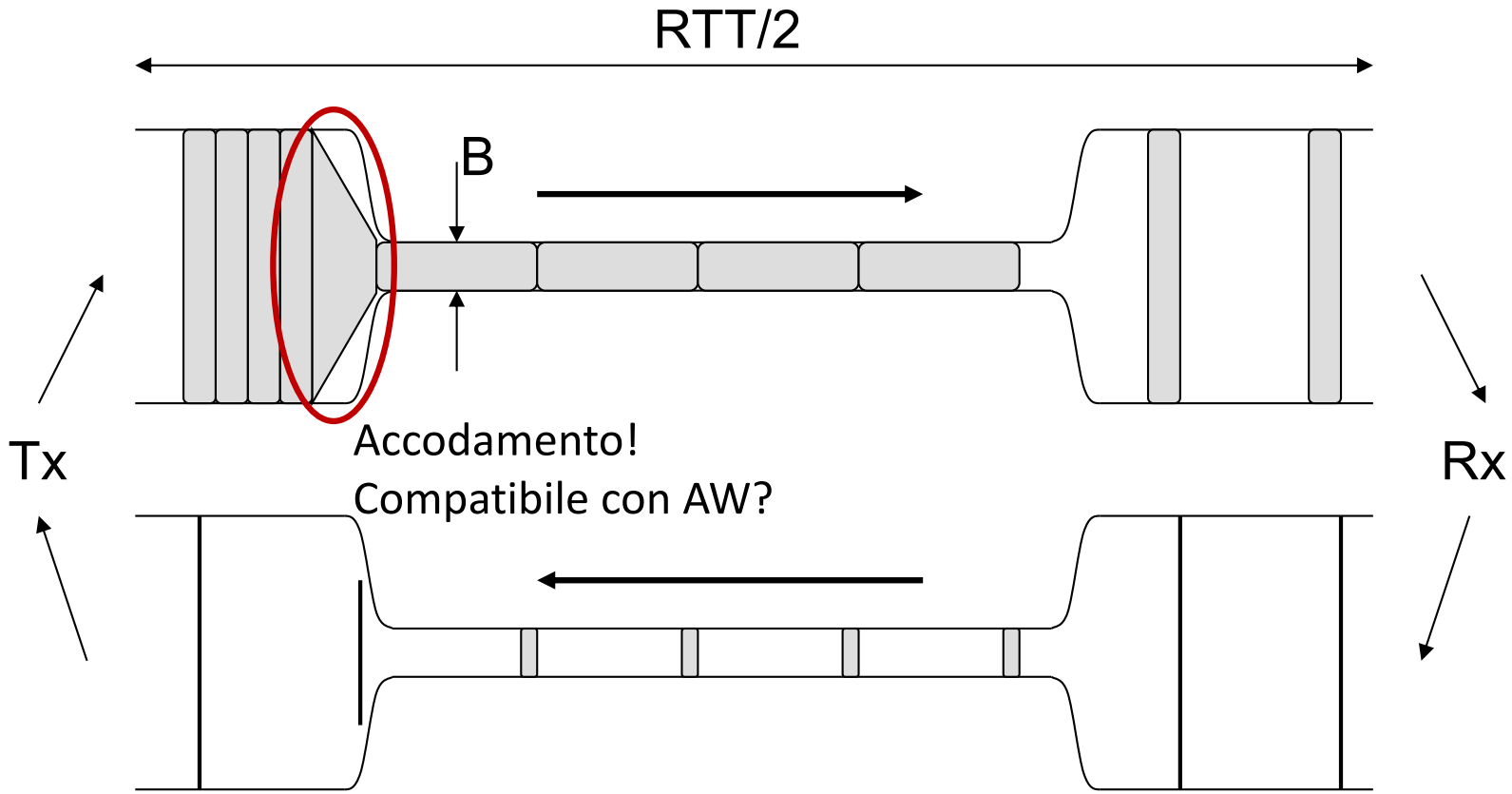
    - Esempio: MSS = 8000 bit, B = 10 Mbit/s



Nel mondo reale: la rete!



Nel mondo reale: la rete!



Problemi

- Al momento dell'instaurazione della connessione TCP la banda disponibile B è incognita
  - A quale valore si deve impostare CW?
- La banda disponibile B può cambiare durante la connessione
  - CW va adattata dinamicamente alla banda disponibile
- Sono definite due fasi che corrispondono a diverse dinamiche di CW
  - Slow start
    - Per raggiungere velocemente un W prossimo a W_{id}
  - Congestion avoidance
    - Per far si che W sia il più prossimo possibile a W_{id} durante la connessione

Ipotesi di partenza

- Trasmettitore e ricevitore sono correttamente configurati
  - Non ci sono problemi di silly window syndrome
  - I buffer di trasmissione e ricezione sono abbastanza grandi per le necessità della connessione
  - Le applicazioni non determinano stagnazione dei dati nei buffer di ricezione

$$AW >> CW \text{ quindi } W = CW$$

- W viene determinata dai meccanismi di controllo della congestione del TCP

Slow start

- All' inizio della connessione

$$w \leq 2*MSS \text{ e } W \leq 2$$

- Per ogni ACK ricevuto senza scandenza di RTO

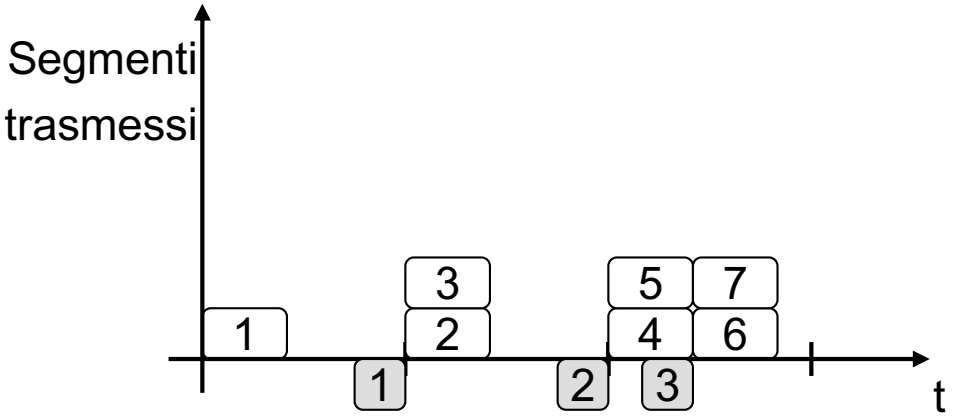
$$W = W+1 \ (w \leq w+MSS)$$

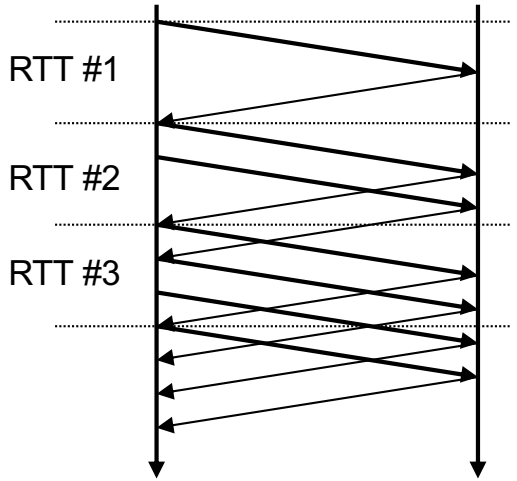
- Slow start termina quando
  - Si verifica congestione (no ACK in RTO)
  - w > ssthr (slow start threshold)
    - ssthr all' apertura della connessione può essere configurato ad un valore arbitrariamente alto (uguale a AW oppure a 64 Kbyte)
    - SSTHR = ssthr/MSS limite per W
- Se w = ssthr si può usare Slow Start o Congestion Avoidance

Dinamiche di Slow Start

- RTT approssimativamente costante
  - Si ipotizza una situazione di rete abbastanza stazionaria
  - L'evoluzione di W avviene per tempi multipli di RTT
- W ha una crescita esponenziale in Slow Start
  - Al termine di ogni RTT la finestra è raddoppiata
- La fase di Slow start dura approssimativamente

$$T_{ss} = RTT \log_2 SSTHR$$





Congestion avoidance

- Si passa da una crescita esponenziale ad una crescita lineare
- w viene incrementata di un MSS per ogni RTT
  - Fino a quando si verifica congestione oppure si raggiunge AW
- Implementazione dell'incremento:
  - Ricevuto ACK i-esimo

$$W = W + 1/W$$

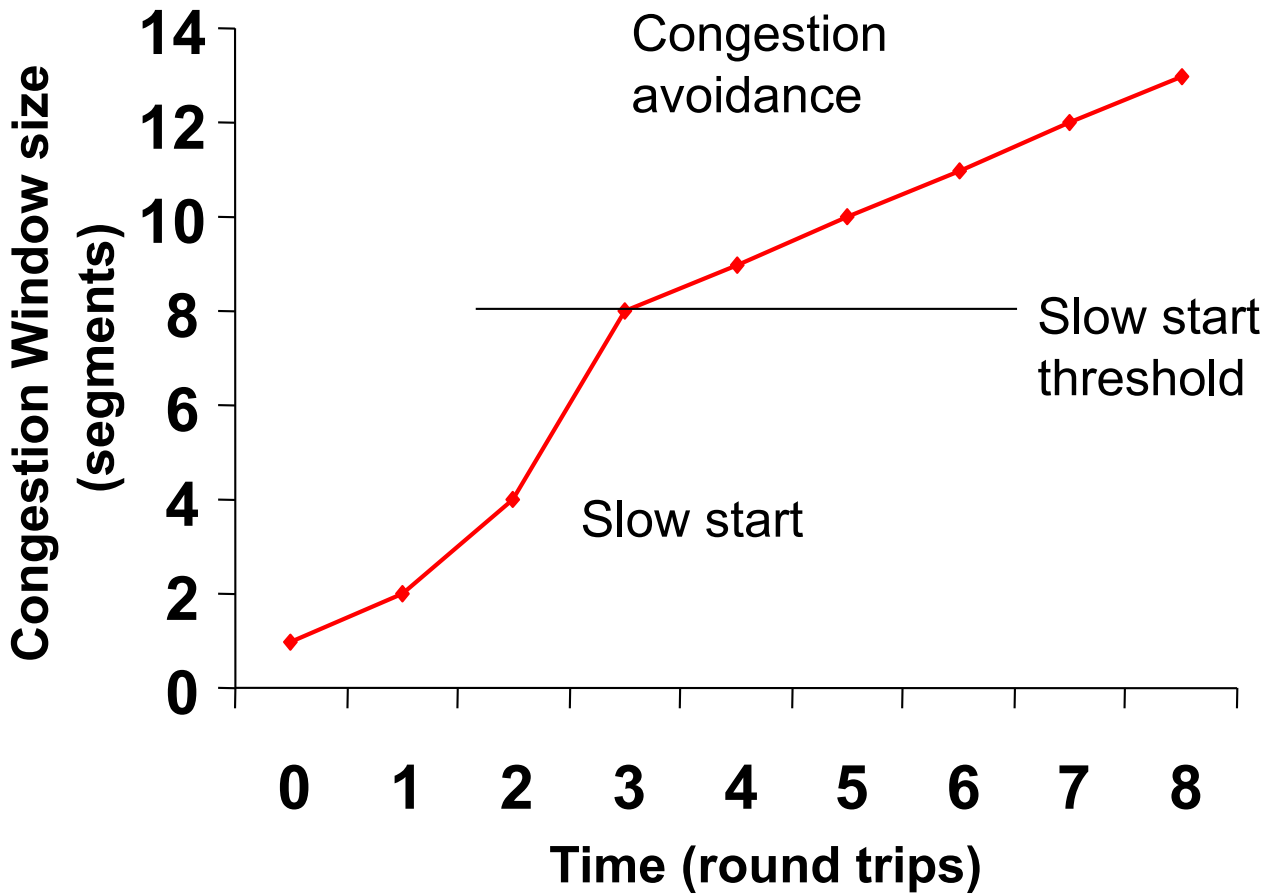
  - Ricevuti gli ACK di una intera finestra risulta $W \approx W+1$
  - Normalmente si implementa il calcolo in byte

$$w = w + MSS^2/w$$

Il calcolo reale

- Si ipotizzi W = CW = 4
  - Vengono trasmessi 4 segmenti
  - Ricevuto il primo ACK risulta
    - W = 4+1/4 = 4,25
  - Ricevuto il secondo ACK risulta
    - W = 4,25+1/4,25 = 4,49
  - …
  - Ricevuto il quarto ACK, ossia terminato il RTT dell'intera finestra
    - W = 4,92
      - Non è esattamente W = W+1 dopo la ricezione di tutti gli ACK della finestra
      - La crescita di W non è esattamente lineare
- Per semplicità ipotizzeremo la crescita di W strettamente lineare nel tempo

Esempio di evoluzione della finestra



Alcune definizioni

- *Loss Window* (LW)
  - Quando scade un RTO il trasmettitore ritiene perso un segmento
  - Il segmento deve essere ritrasmesso
  - Si pone CW ≤ LW (tipicamente LW = 1)
- *Flightsize* = quantità di byte trasmessi ma non confermati
  - È la quantità di dati presenti in rete
  - Non è necessariamente uguale a W
    - Dipende da dove si è arrivati nella trasmissione di una finestra

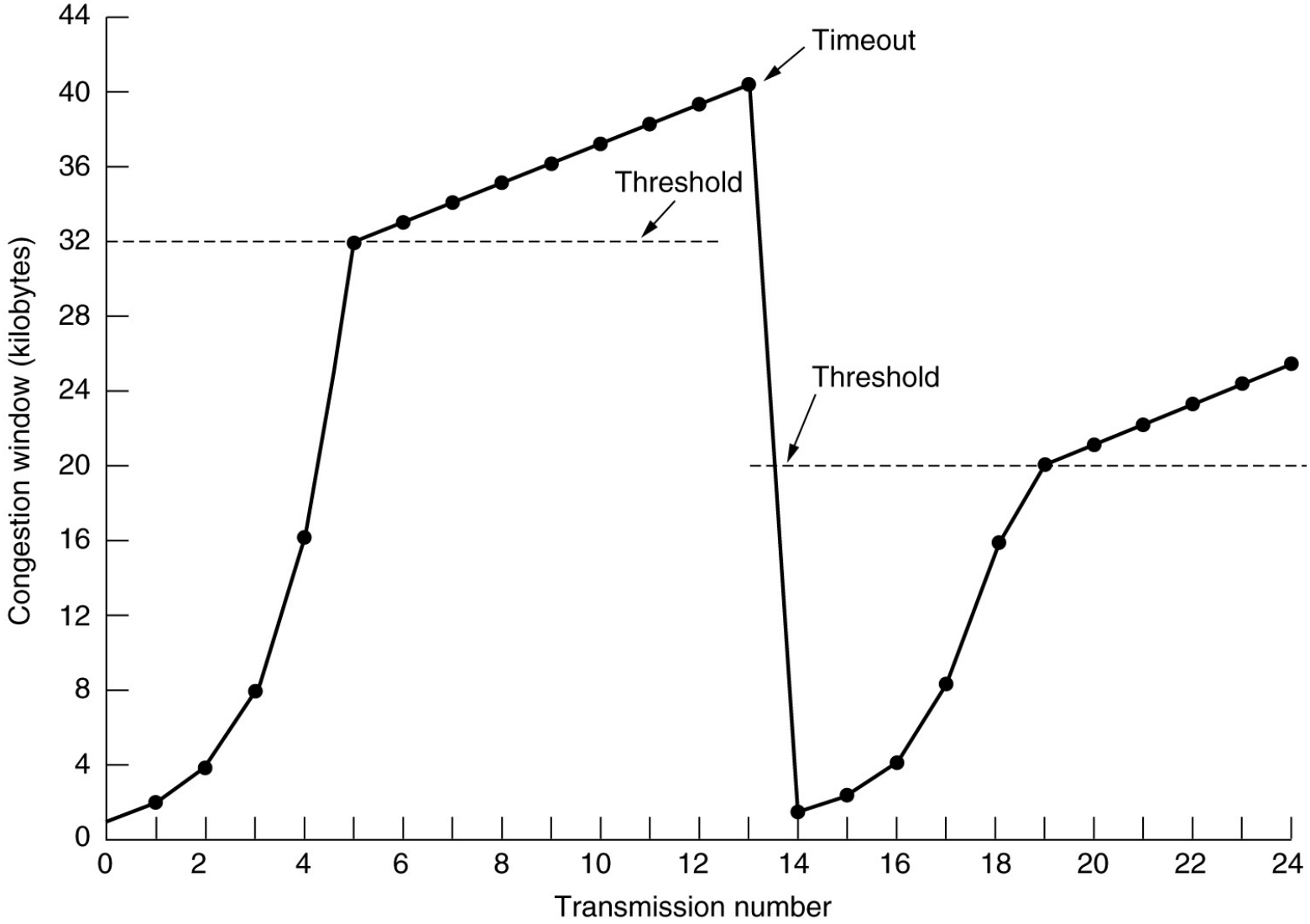
Scade RTO

- RTO scade quando un segmento non viene riscontrato oppure il relativo ACK non giunge in tempo utile
  - Tale evento viene interpretato come indicatore di rete congestionata
    - Con una buona stima del RTT il time out scaduto è (quasi) sempre dovuto a perdita del segmento
    - Con una tecnologia di trasmissione affidabile la perdita è (quasi) sempre dovuta a saturazione delle code nei router
- TCP in Slow Start
  - Si riparte da capo ponendo W = 1
  - Si impone ssthr = max(W/2, 2 MSS)
    - Oppure ssthr = max(Flightsize/2, 2 MSS)
- TCP in Congestion Avoidance
  - Termina la fase di Congestion Avoidance e riparte lo Slow Start
  - Si impone ssthr = max(W/2, 2 MSS)
    - Oppure ssthr = max(Flightsize/2, 2*MSS)

Alcuni commenti

- TCP cerca di
  - Adattarsi dinamicamente alle variazioni di capacità della rete
  - Occupare tutta la banda disponibile (protocollo greedy)
- Recentemente sono state proposte nuove implementazioni di TCP, con una gestione più sofisticata dello slow start e della soglia T, che risultano più efficienti
- Se la rete è molto inaffidabile (per esempio una rete radio) non si può attribuire ogni perdita di pacchetto a congestione e occorrono algoritmi speciali

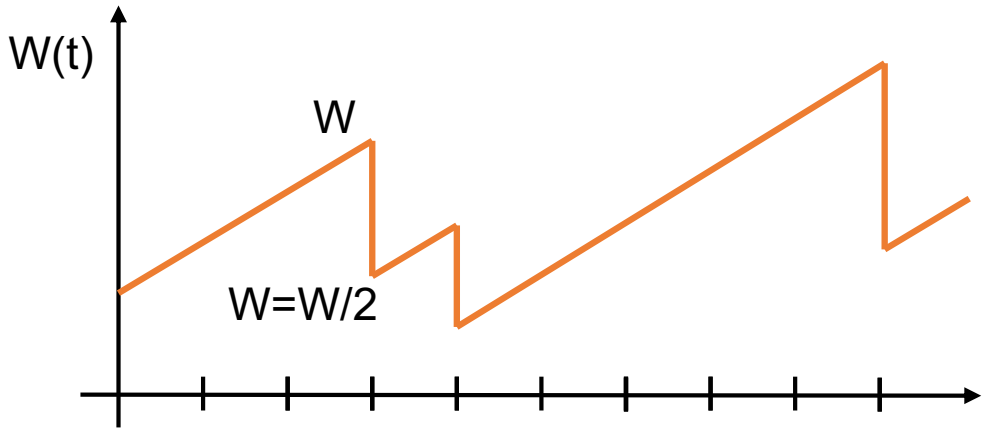
Esempio di evoluzione della CW



Da A.S. Tanenbaum, "Reti di Calcolatori"

Alcune aprossimazioni

- T_{ss} : durata della fase di slow start
- T_{ca} : durata della fase di Congestion Avoidance
- Se la rete è abbastanza stabile:
  - $T_{ss} << T_{ca}$
- In prima approssimazione si può dire che la connessione sia composta da una successione di fasi CA
  - Durante CA la finestra cresce a tasso costante
  - Ad gni perdita di segmento W si dimezza



Esempio

MSS = 1000 byte = 8000 bit
RTT = 50 ms = 50 10^{-3}
W(i) = 24
W(i+1) = 25

RTT numero i -> inviato in rete 24*8000 = 192 Kbit -> 192/50 *10^{6} = 3,8 Mbit/s

RTT numero i+1 -> 25*8000 = 200 Kbit/s -> 4 Mbit/s

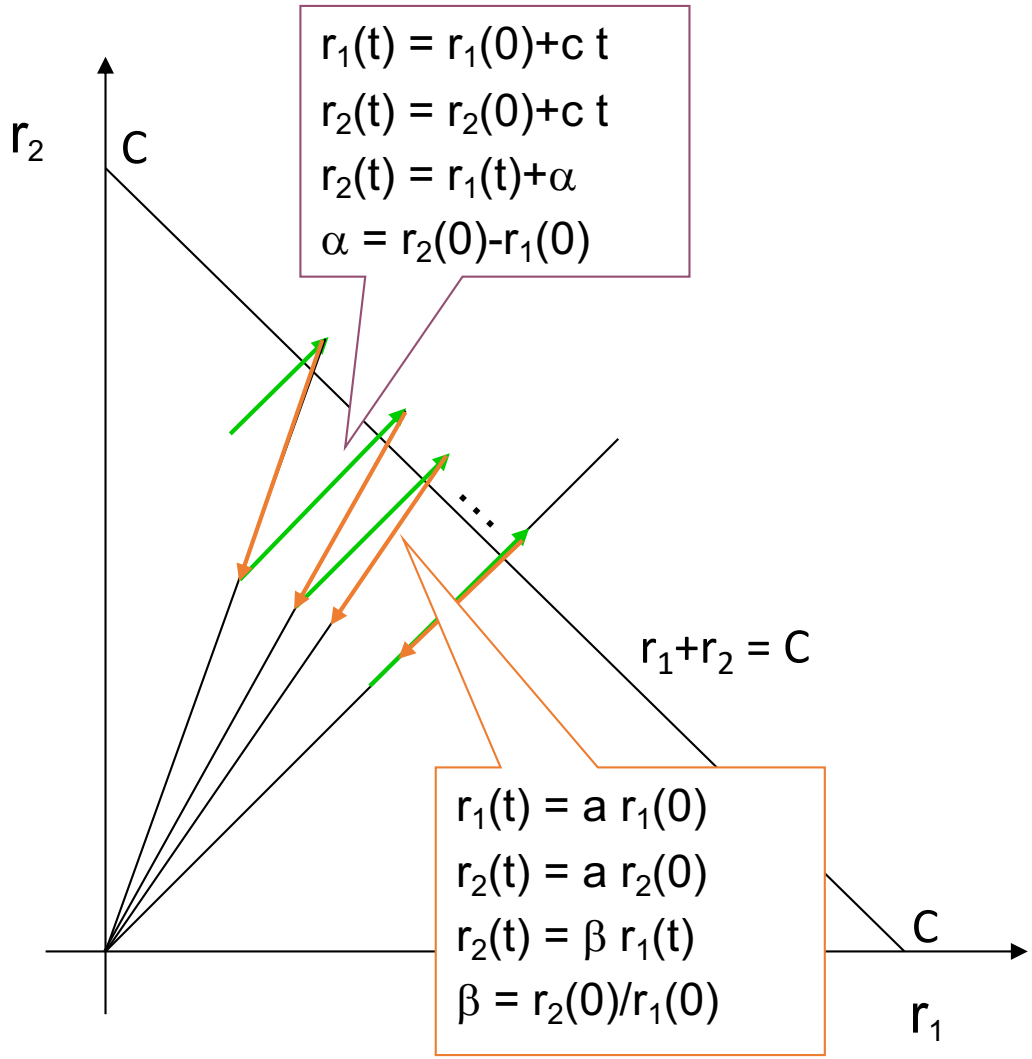
RTT numero i+2 -> 26*8000 = 208 Kbit -> 4,2 Mbit/s

AIMD Congestion control

- L'algoritmo di Congestion Avoidance viene anche detto di incremento additivo e decremento moltiplicativo
  - Se r è la quantità di dati inviata dal trasmettitore
- Additive-increase
  - Aumento la velocità (dimensione della finestra) in modo additivo
    - Se la rete non evidenzia congestione al passo i-esimo
    - $r(t_{i+1}) = r(t_i) + c$ $\quad c << r_{max}$
- Multiplicative-decrease
  - Decremento la velocità (dimensione della finestra) in modo moltiplicativo
    - Se si rivela una situazione di congestione
    - $r(t_{i+1}) = a \times r(t_i)$ $\quad a < 1$

Condivisione della risorsa

- Nel lungo termine AIMD permette di avere un'equa distribuzione della banda disponibile
  - Due connessioni si trovano a condividere la banda
  - Entrambe tentano di occupare tutta la banda disponibile
  - Si realizzano situazioni di congestione



Delayed ACK

- Se tutti i messaggi correttamente ricevuti generano un ACK
  - In un RTT trasmetto W messaggi
  - Ricevo W ACK
  - Ogni RTT incremento la finestra di

    #ACK * 1/W = W * 1/W = 1
- Se il protocollo utilizza i delayed ACK
  - In un RTT trasmetto W messaggi
  - Ricevo un ACK ogni due messaggi
  - Ogni RTT incremento la finestra di

    #ACK * 1/W = ½ W * 1/W = ½

RTT

- L'incremento di CW è funzione del RTT
  - W = W+1 approssimativamente ogni RTT
- Due connessioni che sperimentano diversi RTT aumentano in modo diverso le proprie finestre
  - Si tende a favorire connessioni con RTT brevi su connessioni con RTT lunghi

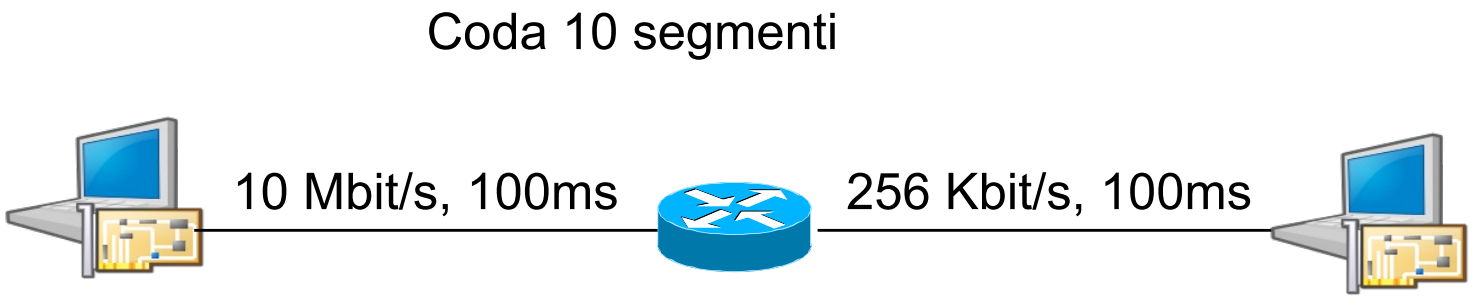
Esempio

- Perché l'utilizzo di Slow Start e congestion avoidance migliorano il comportamento del protocollo?
- Caso di studio
  - Prodotto banda ritardo

$$B\ RTT = 400\ 10^{-3}\ 256\ 10^{3} = 100000 \text{ bit}$$

  - Con MSS = 1000 byte = 8000 bit

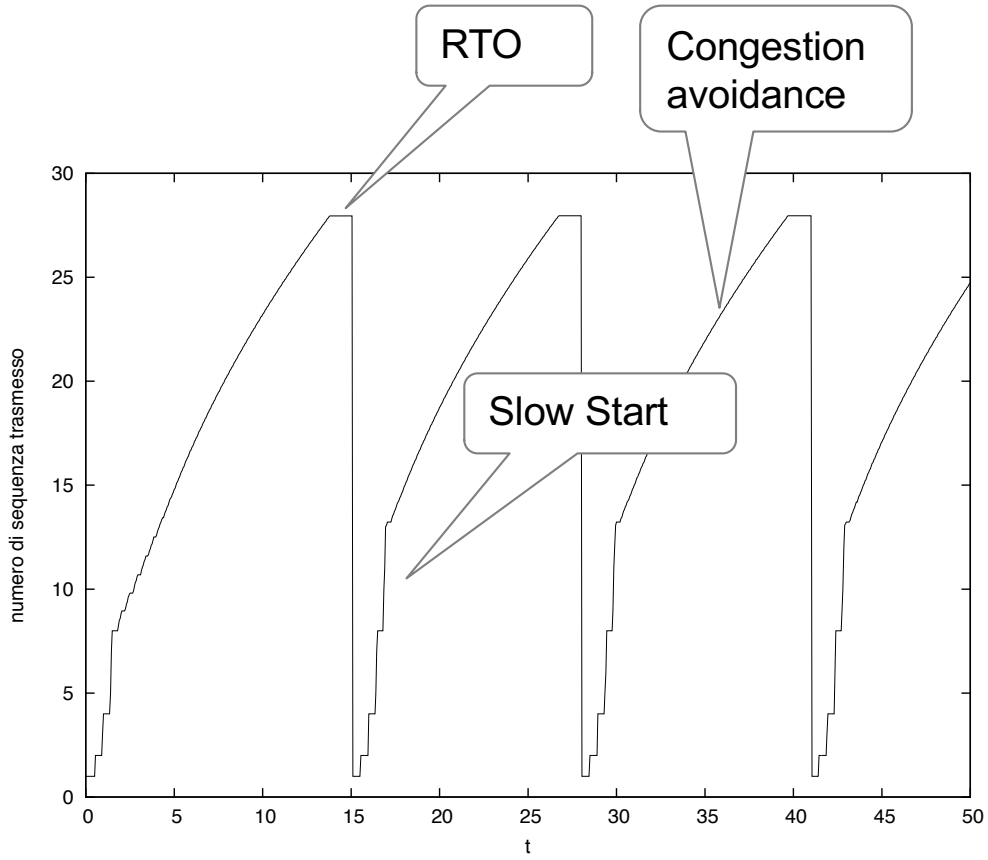
$$Wid \geq B\ RTT / MSS = 12{,}8 \approx 13 \text{ segmenti}$$



Andamento di CW (AW = 64)

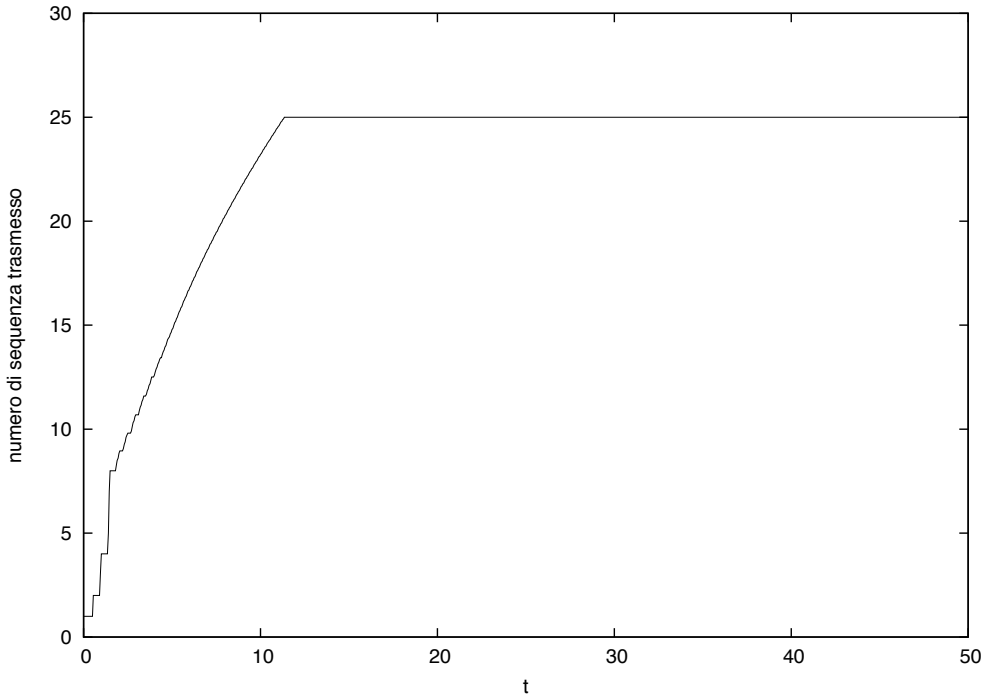
- CW cresce quasi linearmente
  - Fino a quando?
    - Raggiunge AW
    - Si verifica una congestione con perdita di segmenti

- Nell'esempio si verifica la congestione
  - Si riparte di nuovo in slow start
  - ssthr = W/2
  - Si verifica un andamento circa periodico

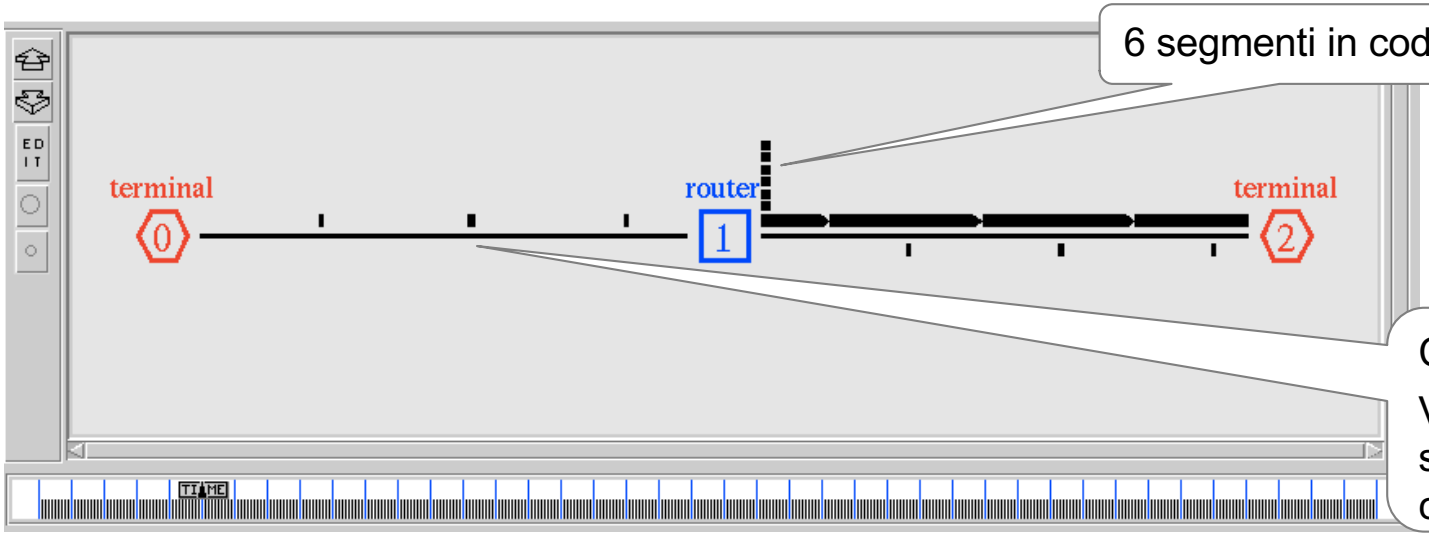


Se AW = 25

- AW pone un limite a W prima che CW possa determinare una congestione
  - W si stabilizza a AW
  - Viene utilizzata la massima capacità al ricevitore
  - W > 13 quindi viene utilizzata anche tutta la capacità di rete



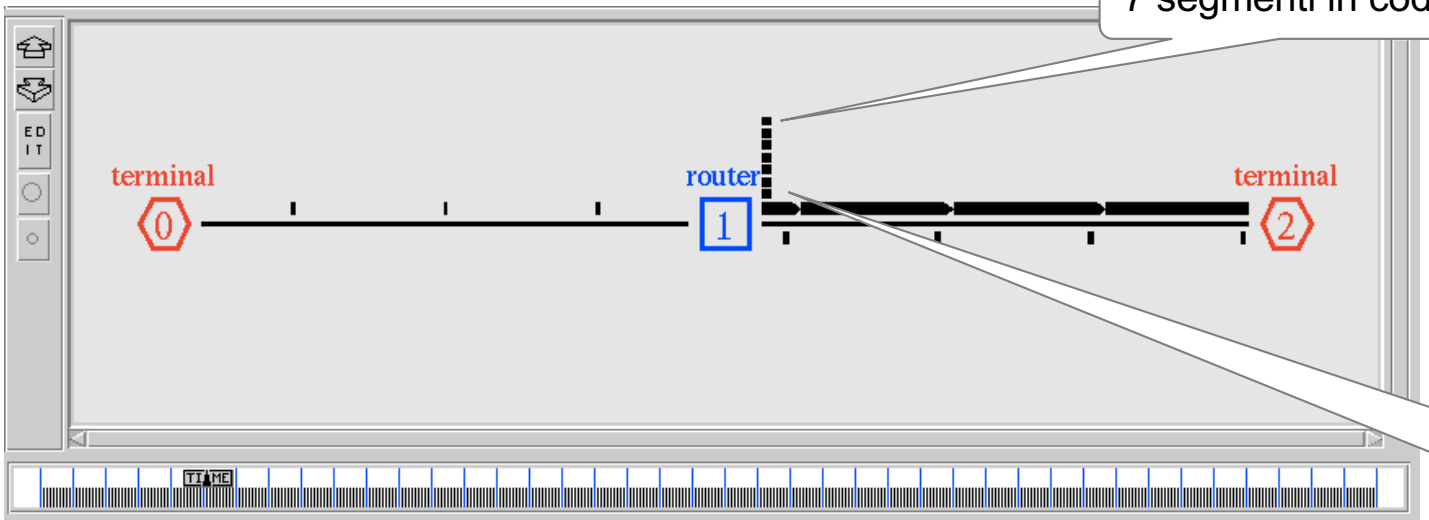
Come si determina la congestione in CA



6 segmenti in coda

CW aumenta di 1

Vengono inviati due segmenti in contemporanea



7 segmenti in coda

La coda nel nodo si allunga di un segmento

Se AW = W_{id} = 13