



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)

$$W \text{ MSS} = w$$



# 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

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

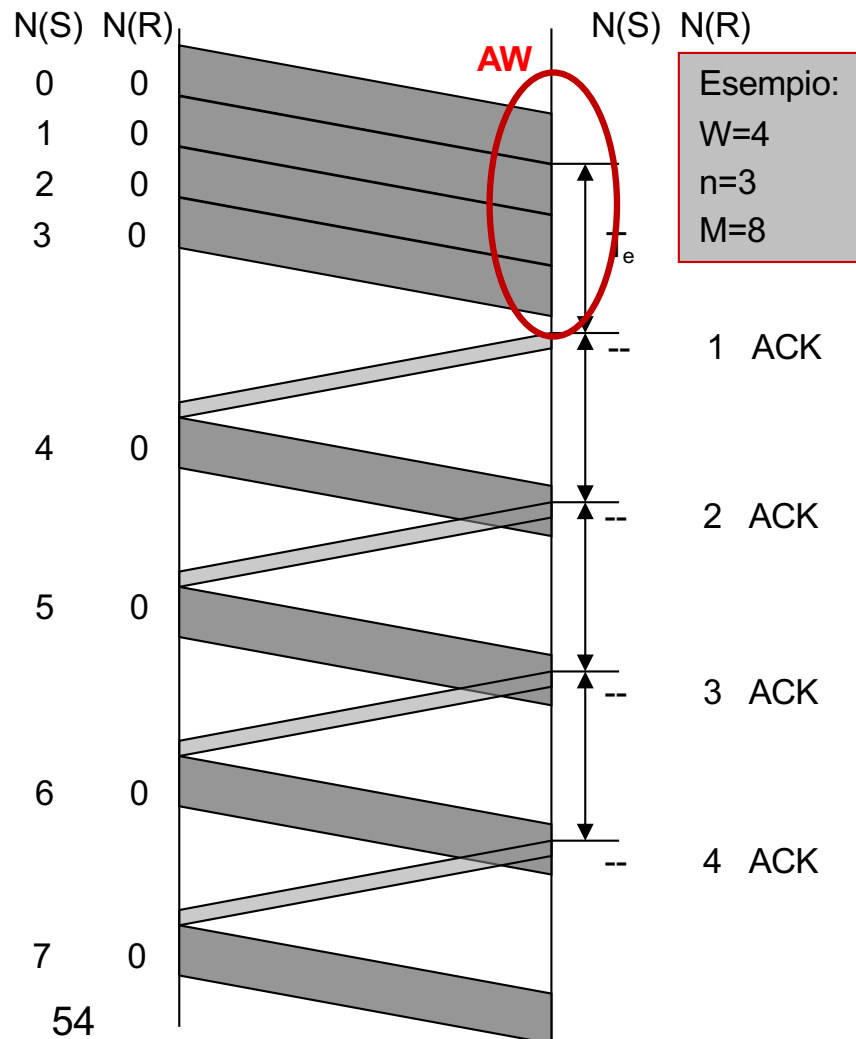


# Motivazioni

- 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

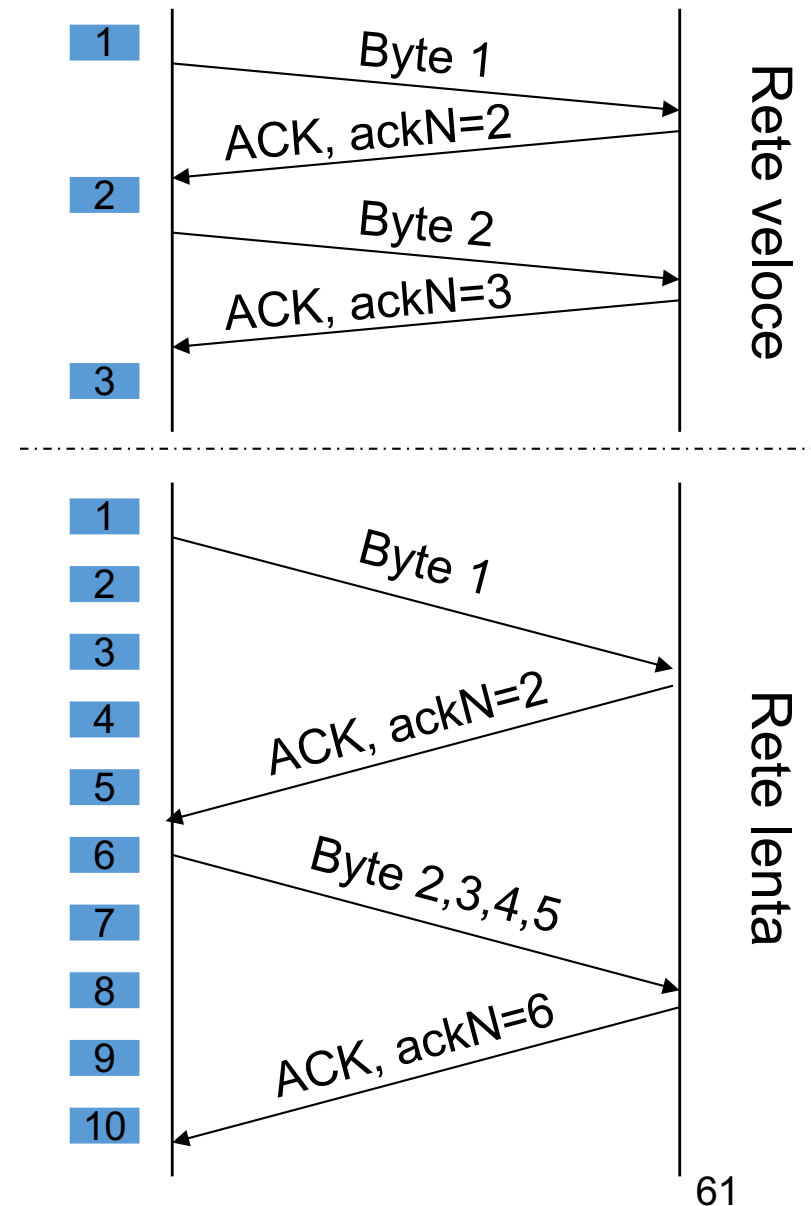


# Tramettitore lento: Algoritmo di Nagle

- 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

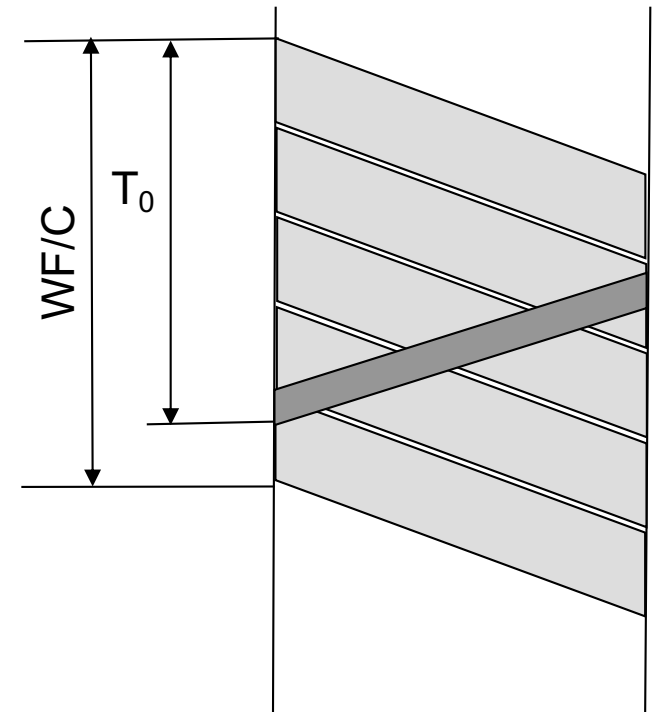


# Efficienza

- 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

$$T_0 = \text{MSS}/B + \text{RTT} = 20.8 \text{ ms}$$

$$\text{MSS}/B = 0.8 \text{ ms}$$

W = 26 segmenti di dimensione MSS

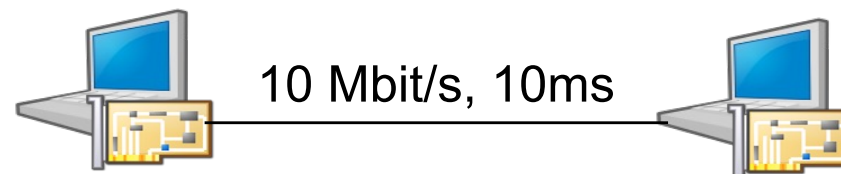
- Prodotto banda ritardo

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

- Pertanto

$$B \text{ RTT} / \text{MSS} = 25 \text{ segmenti}$$

$$W_{id} \geq B \text{ RTT} / \text{MSS}$$



# 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:

$$S = w / RTT$$



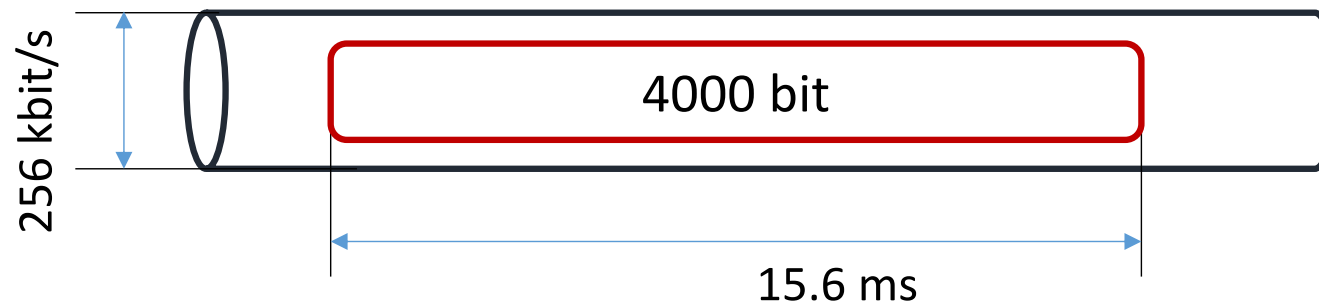
# Nel mondo reale : la rete!

- 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

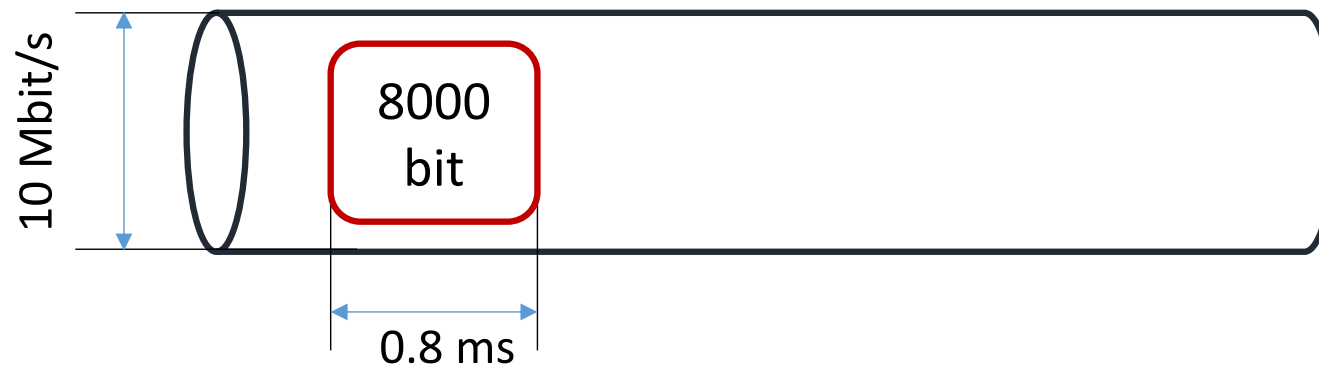
# La rappresentazione di Jakobson



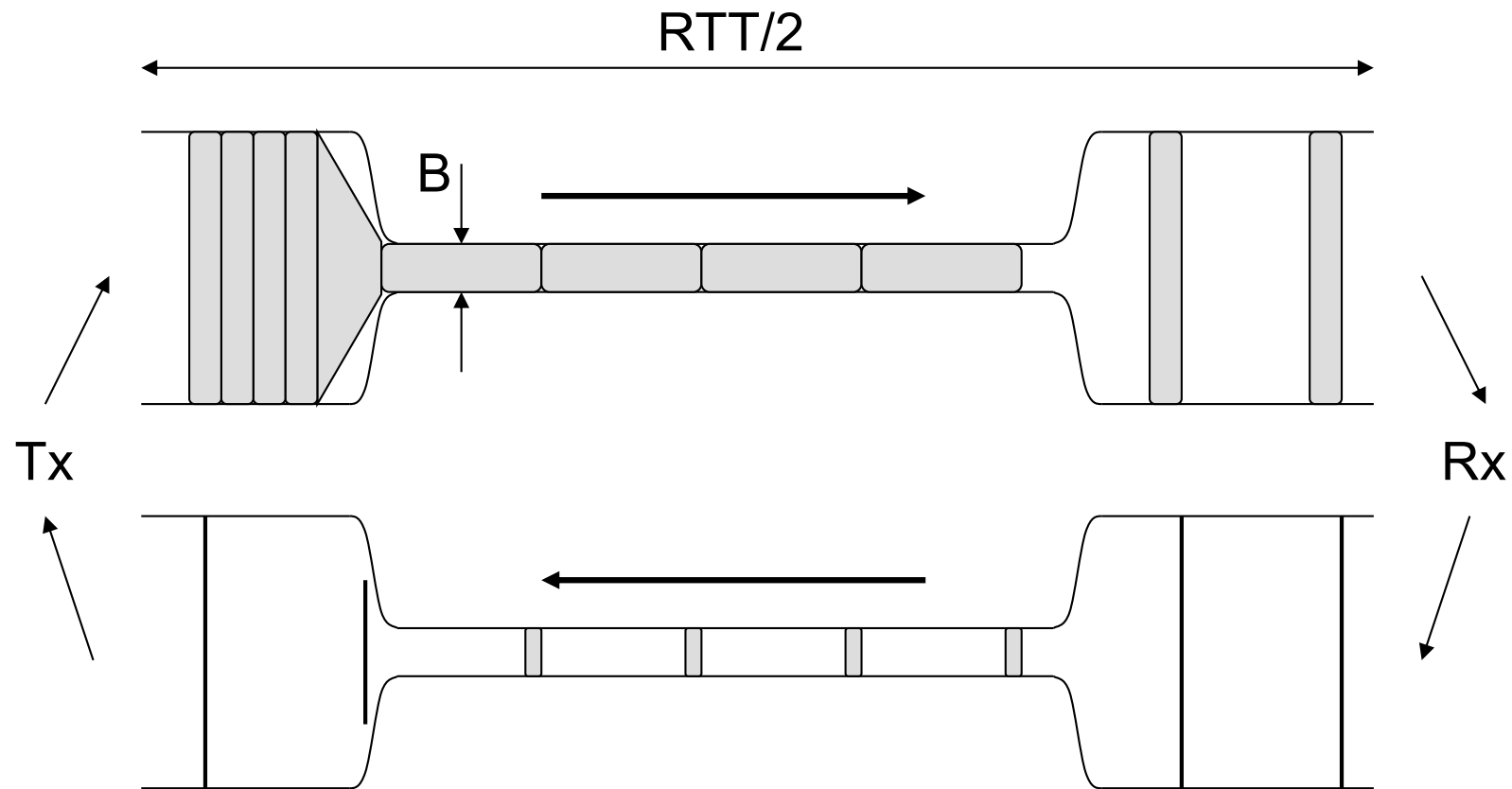
- Diagramma bidimensionale
  - Orizzontalmente si indica il tempo
  - Verticalmente di indica la banda disponibile
    - Esempio:  $MSS=4000\text{bit}$ ,  $B=256\text{Kbit/s}$



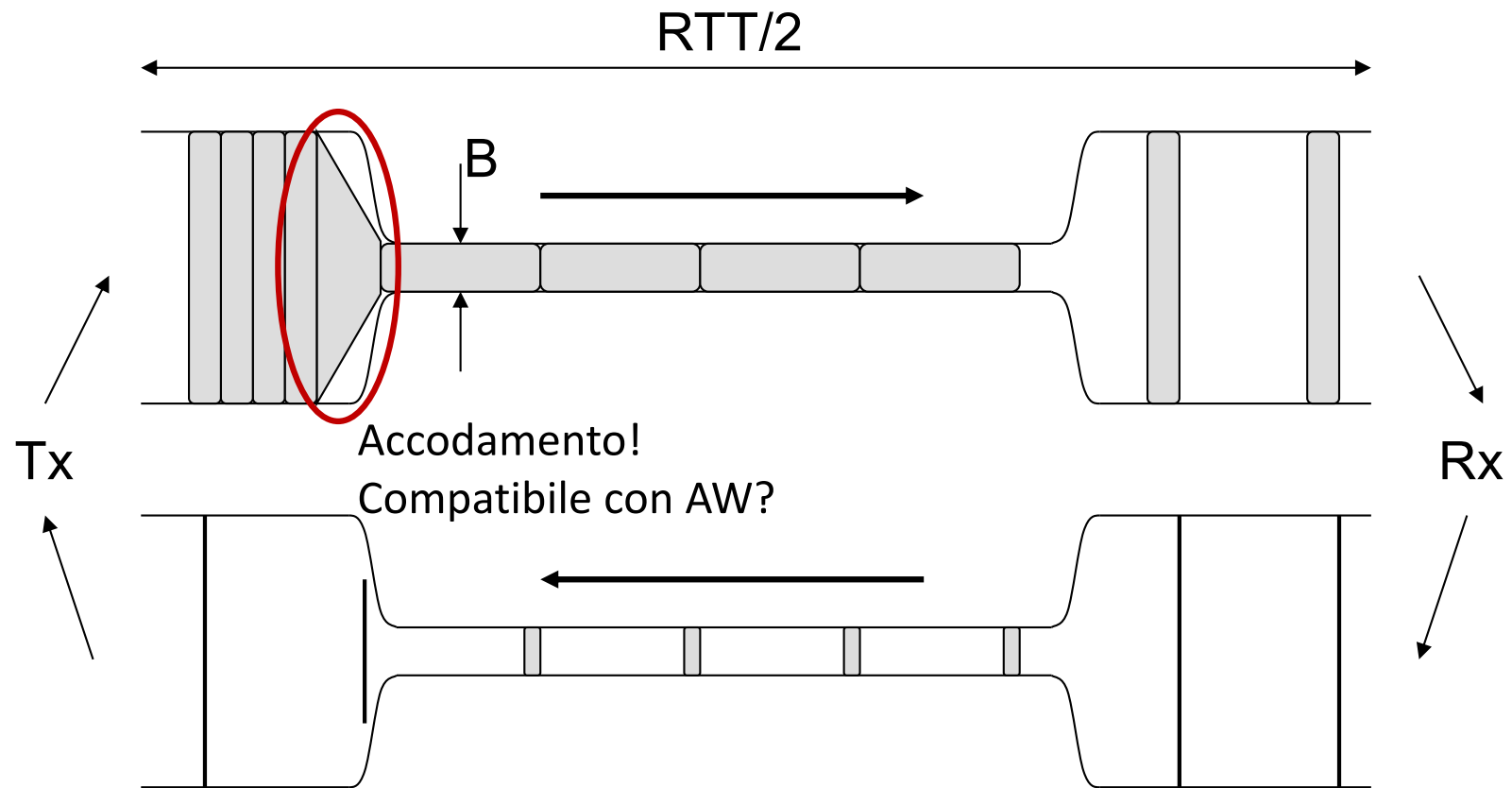
- Esempio:  $MSS = 8000 \text{ bit}$ ,  $B = 10 \text{ Mbit/s}$



# 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 sì 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

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

- **$W$  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 scadenza di RTO

$$W = W+1 \text{ (} w \leq w+MSS \text{)}$$

- Slow start termina quando

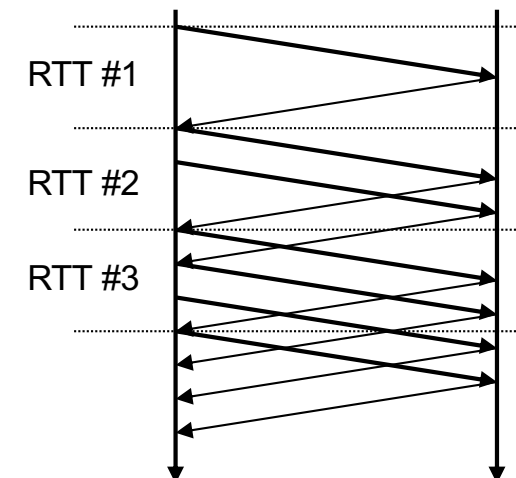
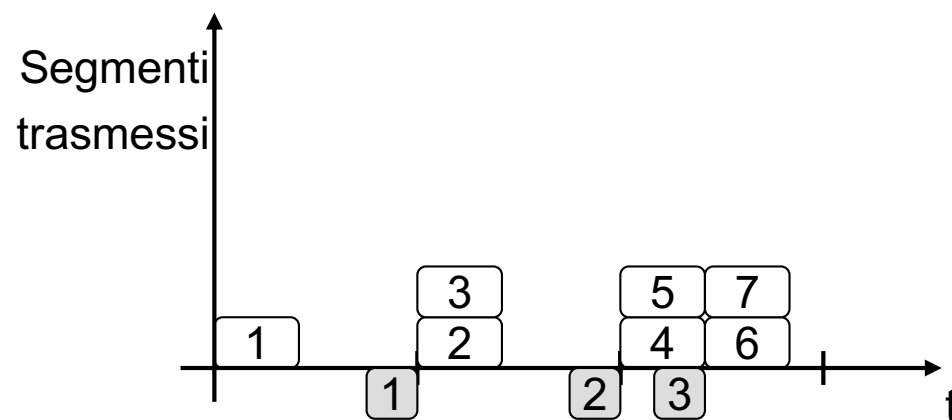
- Si verifica congestione (no ACK in RTO)
- $w > ssth$  (slow start threshold)
  - $ssth$  all' apertura della connessione può essere configurato ad un valore arbitrariamente alto (uguale a  $AW$  oppure a 64 Kbyte)
  - $SSTHR = ssth/MSS$  limite per  $W$

- Se  $w = ssth$  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
$$w = w + \text{MSS}^2/w$$

# 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