Controllo di flusso in TCP

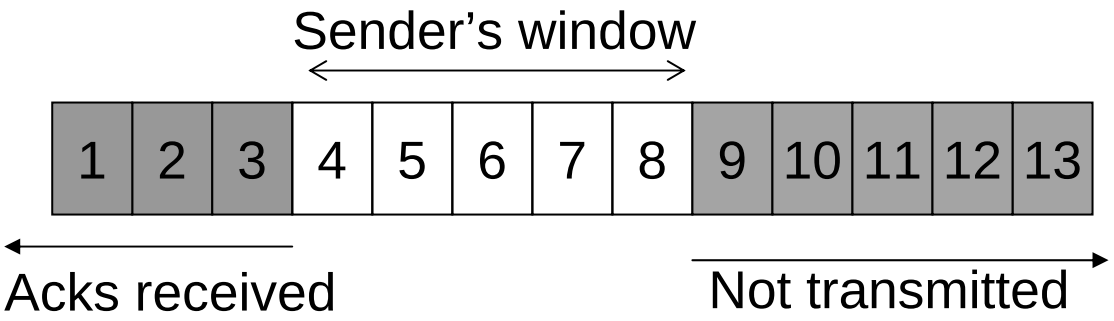
Prof. Ing. Maurizio Casoni

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Modena e Reggio Emilia

Meccanismo a finestra

- La dimensione della finestra viene messa a punto dinamicamente sulla base di informazioni
  - Provenienti dal ricevente (advertised window o AW)
    - Comunicata in modo esplicito al trasmettitore
  - Funzione dello stato di congestione della rete (congestion window o CW)
    - Non c'è in genere comunicazione esplicita di CW e TCP gli attribuisce un valore in base alla percezione di congestione che riceve dalla rete

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



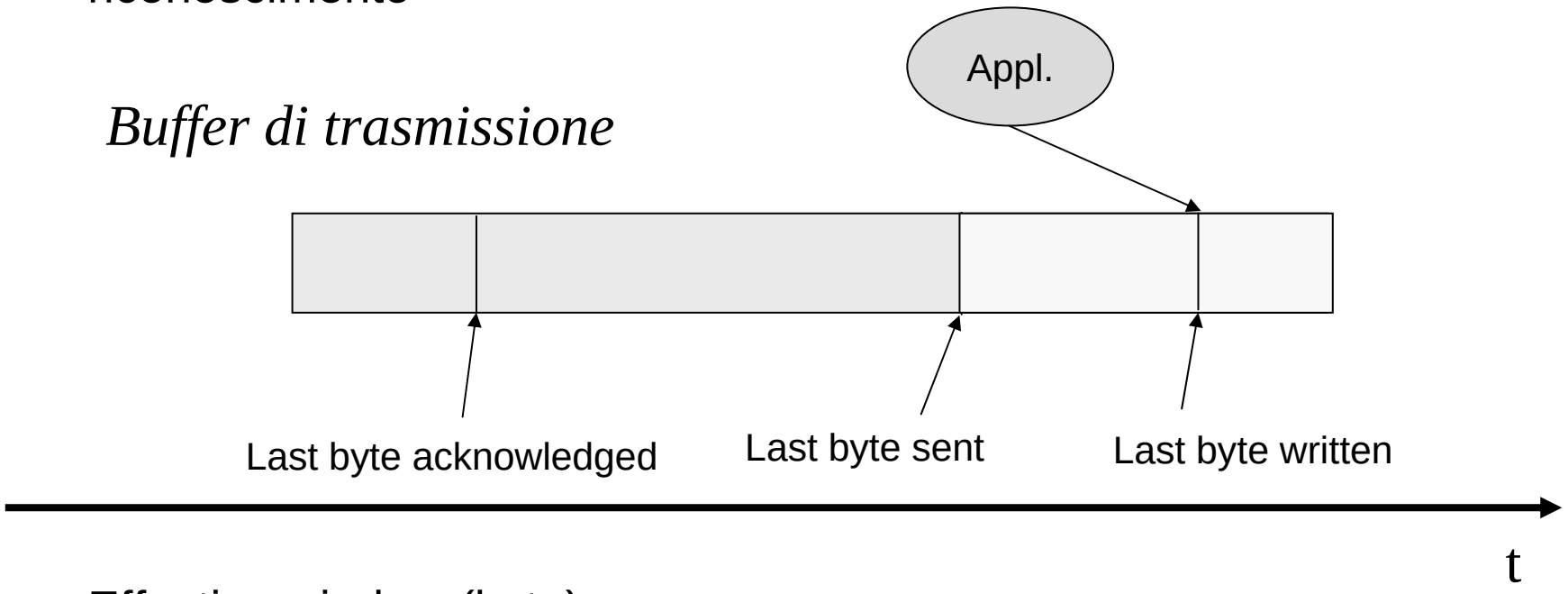
Unità di misura di W

- TCP adotta la numerazione sequenziale dei dati trasmessi per byte
- W può essere misurata
  - In byte (w)
  - In numero di segmenti
    - 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)

$$W \times MSS = w$$

Buffer di trasmissione

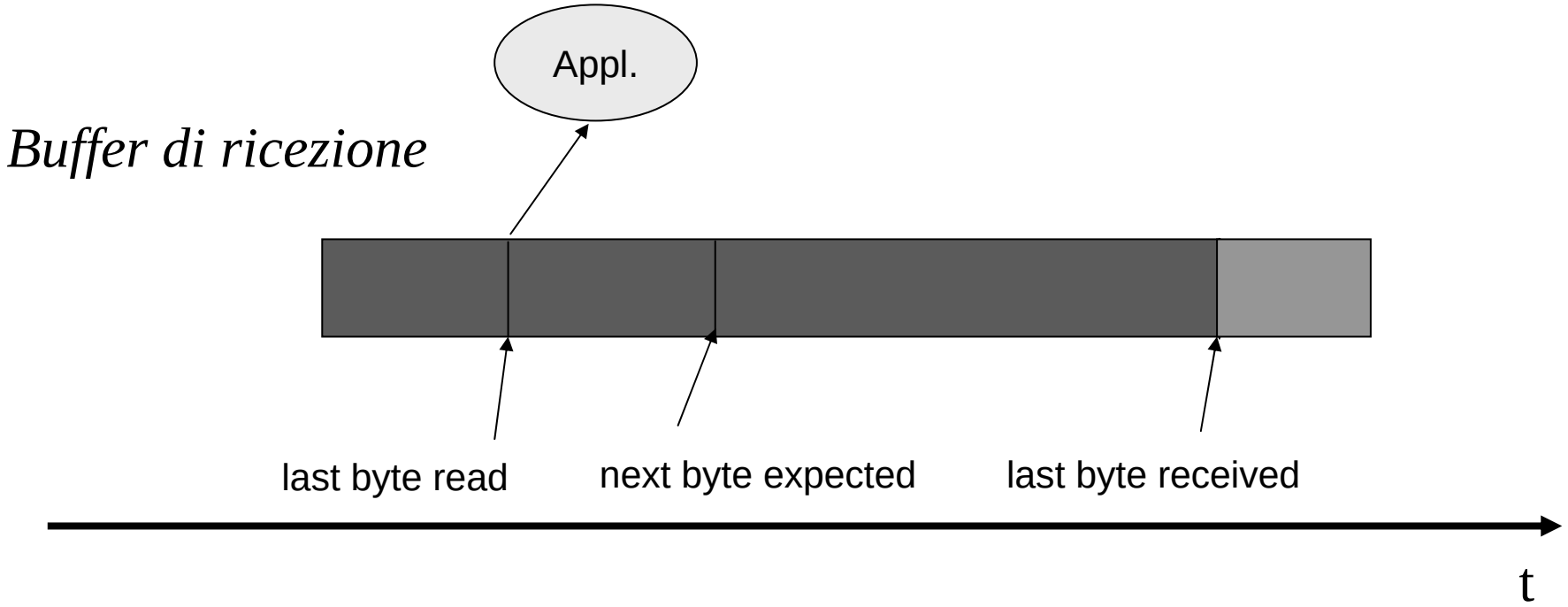
- Il buffer di trasmissione viene riempito dai byte generati dalla applicazione
- Il trasmettitore non puo' avere piu' di AW byte in attesa di riconoscimento



- Effective window (byte)
  - EW= AW-(last byte sent-last byte acknowledged)

Buffer di ricezione

- Il processo ricevente legge byte dal buffer di ricezione
- I dati possono arrivare fuori ordine



- Il valore di AW (byte) e' lo spazio disponibile nel buffer del ricevitore di dimensione B
  - AW= B – (last byte received – last byte read)

Attuazione del controllo

- Come un ricevitore lento blocca un trasmettitore piu' veloce
  - Il buffer di ricezione si riempie
    - AW $\Rightarrow$ 0
  - EW $\Rightarrow$ 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

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

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

$$w_{id} \geq RTT*B$$

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

- 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:

$$S = w/RTT$$

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
  - 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

$$AW >> CW \Rightarrow W=CW$$

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

Slow start

All’inizio la W è posta a 1 MSS

- Trasmesso il segmento si ferma ed aspetta l’ACK

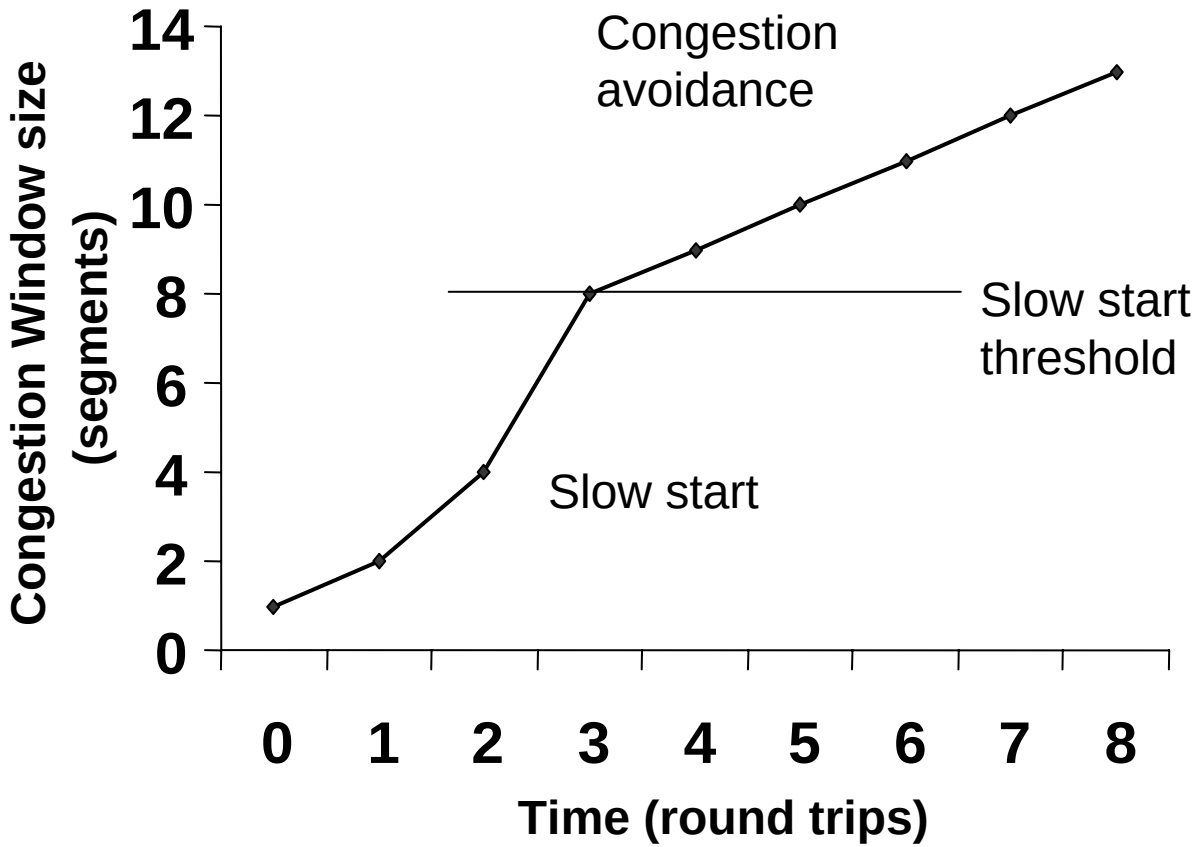
- Se ACK arriva entro il Timeout si pone W=2 MSS e si trasmettono 2 segmenti
- Ad ogni nuovo ACK ricevuto W = W+1
  - Ricevuto l’ACK del terzo segmento risulta W=4
- Slow start viene interrotto quando si raggiunge la Slow Start Threshold (SSThr)
  - All’apertura della connessione un valore tipico per la soglia è SST=64 Kbyte
  - W ha una crescita esponenziale
  - 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 una MTU ad ogni passo fino a raggiungere AW
- Ad ogni nuovo ACK ricevuto
  - W = W+1/W
  - Ricevuti gli ACK di una intera finestra risulta W = W+1
  - Poiché w = W x MSS risulta

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

Esempio di evoluzione della finestra



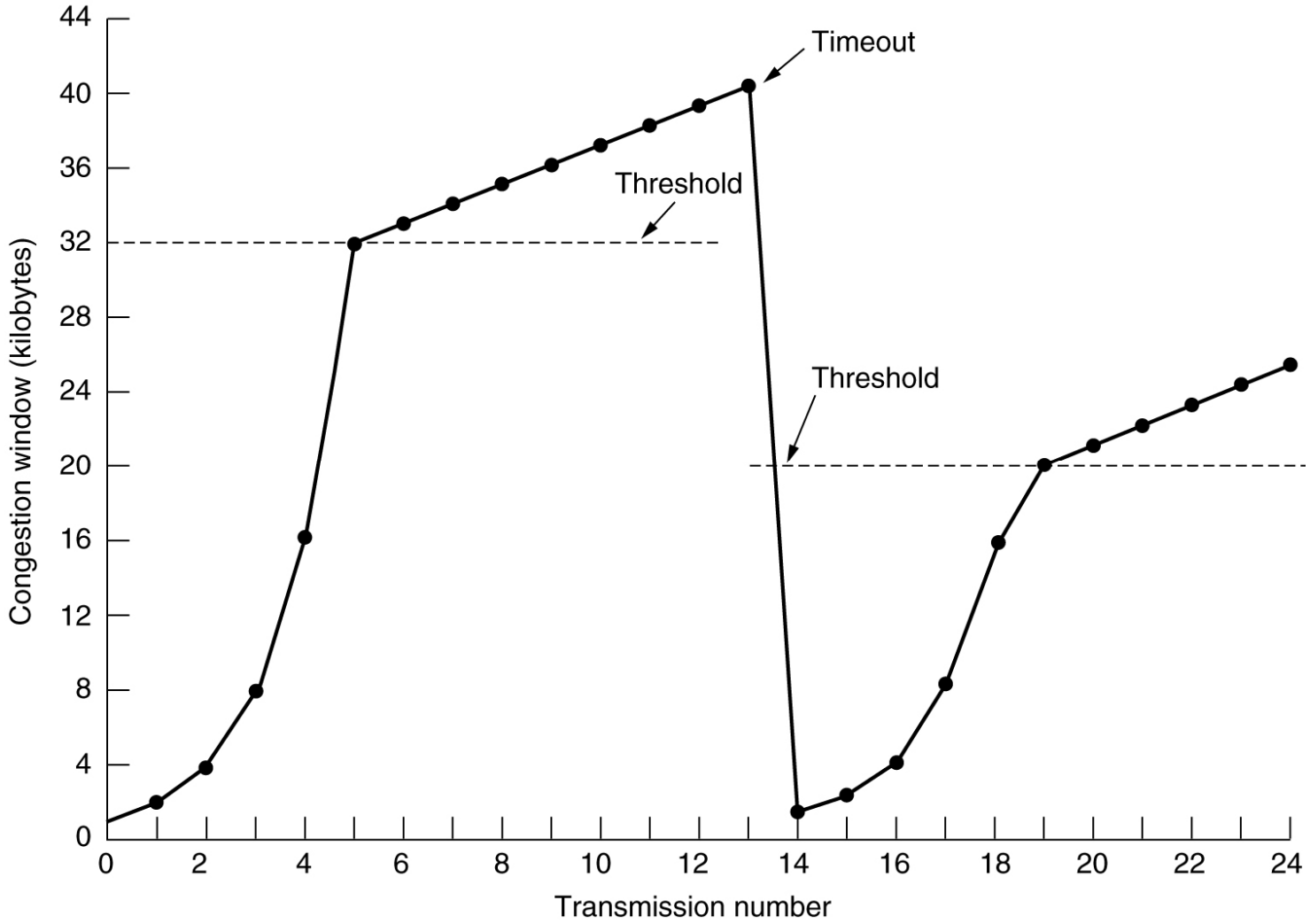
In caso di congestione

- Un segmento non viene riscontrato
  - Scade il Time out
  - Viene preso 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
- TCP in Congestion Avoidance
  - Riparte lo Slow Start con W = 1
  - Si impone SSThr = SSThr/2

Alcuni commenti

- In questo modo TCP si adatta dinamicamente alle variazioni di capacità della rete, tendendo ad 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”

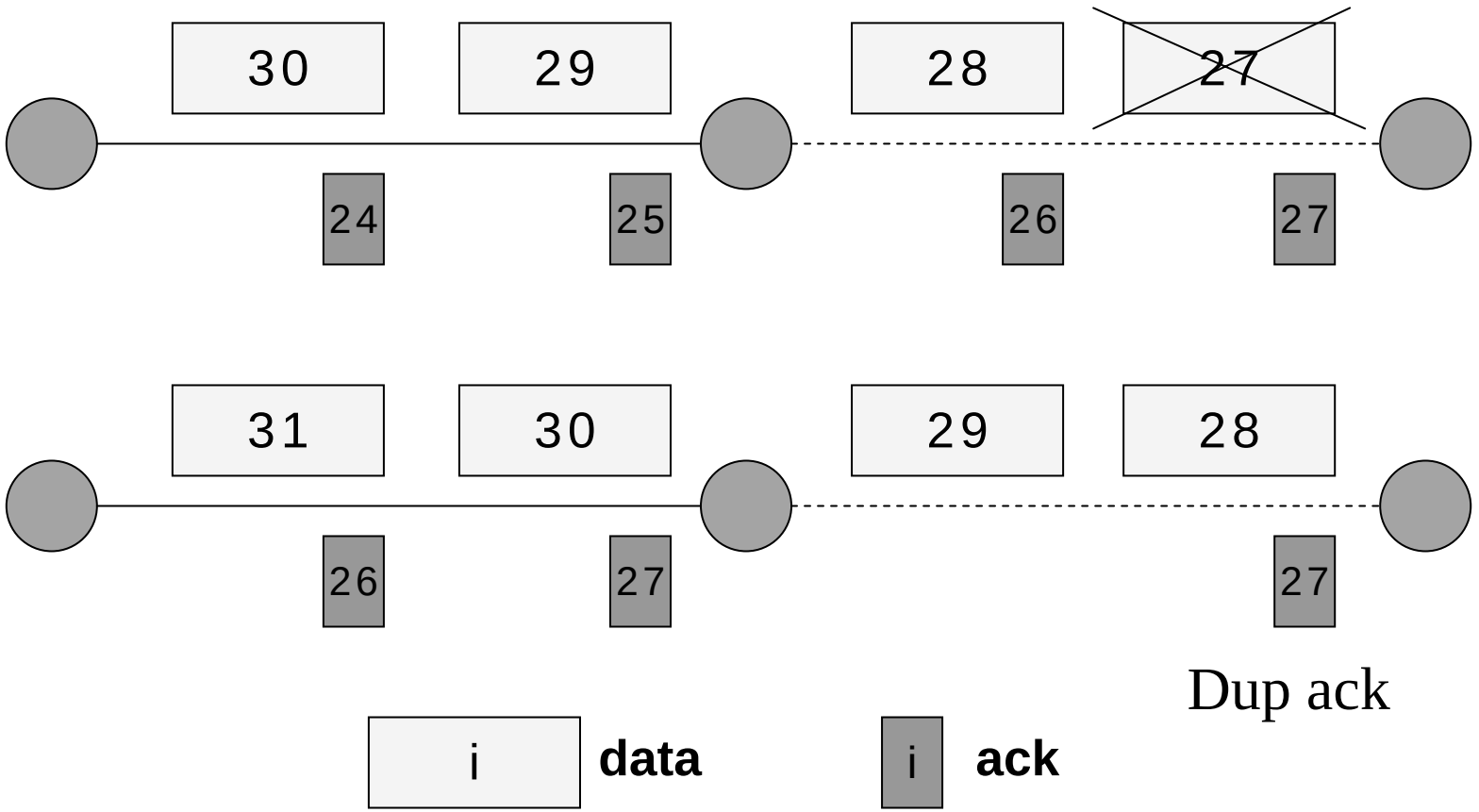
AIMD Congestion control

- 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
    - $r_{i+1} = r_i + c$ $\quad c << r_{max}$
- Multiplicative-decrease
  - Decremento la velocità (dimensione della finestra) in modo moltiplicativo
    - Se si rivela una situazione di congestione
    - $r_{i+1} = a \times r_i$ $\quad a < 1$

Segmenti fuori sequenza

- Quando un segmento viene perso il ricevitore riceve uno o più segmenti fuori sequenza
  - Il ricevitore *dovrebbe* inviare immediatamente un ACK duplicato per ogni segmento fuori ordine
    - L'ACK duplicato è riferito all'ultimo segmento ricevuto correttamente
- La perdita di un segmento genera ACK duplicati
  - Questo può essere utilizzato come indicazione di congestione unitamente al time-out
- Sono stati proposti ulteriori algoritmi per sfruttare al meglio questa situazione
  - Fast retransmit
  - Fast recovery

Ack duplicati: esempio



Fast retransmit

- La ricezione di 3 ACK duplicati indica la perdita del segmento successivo a quello confermato dagli ACK
  - 3 ACK con uguale ackN
  - Sono stai ricevuti i byte fino a ack(N-1), manca il segmento successivo
- TCP ritrasmette il segmento mancante senza attendere la scadenza del timeout
  - Si ritrasmette il segmento di numero seqN=ackN
  - Si fa partire la procedura di Fast recovery

Fast recovery

- Gli ACK duplicati indicano che il ricevitore sta comunque ricevendo dei segmenti
  - La rete continua a consegnare segmenti a meno di quello mancante che non è giunto a destinazione
    - Questo può non essere vero se la rete duplica grandi quantità di segmenti
- Si assume che la perdita del segmento sia un evento puntuale
  - Non c'è ragione di ripartire con lo slow start
    - Si spera che la ritrasmissione del segmento mancante sia sufficiente
    - Ritrasmesso il segmento mancante si prosegue in congestion avoidance

Algoritmo di Fast recovery

- Si riduce la soglia di CW
  - SSThr = SSThr/2
- Si pone W = SSThr+3
  - Si tiene conto che almeno 3 segmenti successivi al mancante sono stati ricevuti in quanto sono partiti gli ACK duplicati
- Si continua con la trasmissione
  - Si rispetta il limite di W
  - Ad ogni ACK ricevuto ed ancora duplicato si pone W = W+1
- Quando arriva l'ACK per il pacchetto perduto finisce la fase di fast recovery
  - Si riparte con congestion avoidance ponendo W = SSThr

Alcuni commenti

- Fast retransmit/fast recovery cercano di sfruttare la capacità di TCP di memorizzare i segmenti fuori sequenza
  - SST viene ridotta per riadattarsi alle condizioni di carico della rete tramite una nuova fase di congestion avoidance
  - Durante il Fast recovery si *gonfia* W di tanto quanti sono i segmenti ricevuti fuori sequenza (che hanno causato gli ACK duplicati)
  - Terminata la fase di fast recovery W viene *sgonfiata* per tornare al valore previsto da congestion avoidance

Modelli matematici del TCP

- Un modello analitico del TCP deve catturare due processi fondamentali:
  - Dinamica della finestra di trasmissione W(t)
  - Il processo di perdita che si genera all'interno della rete
- Espressione della velocità di trasmissione dei pacchetti da parte della sorgente TCP:

$$X(t)=W(t)/RTT$$

  - Nell'ipotesi che RTT rimanga sostanzialmente costante per tutto il periodo di osservazione

Un’ipotesi di base

- Per semplificare il calcolo ipotizziamo che
  - Le dinamiche del TCP non modificano significativamente lo stato delle code nei router
    - I ritardi non si modificano e RTT è circa costante
  - Il tempo si suddivide in unità di dimensione RTT
    - L’intera finestra (batch) viene trasmessa all’inizio del periodo
    - Al termine del periodo arriva la conferma di ricezione e si può iniziare a trasmettere la finestra successiva