TCP over wireless

Sistemi Wireless, a.a 2011/2012

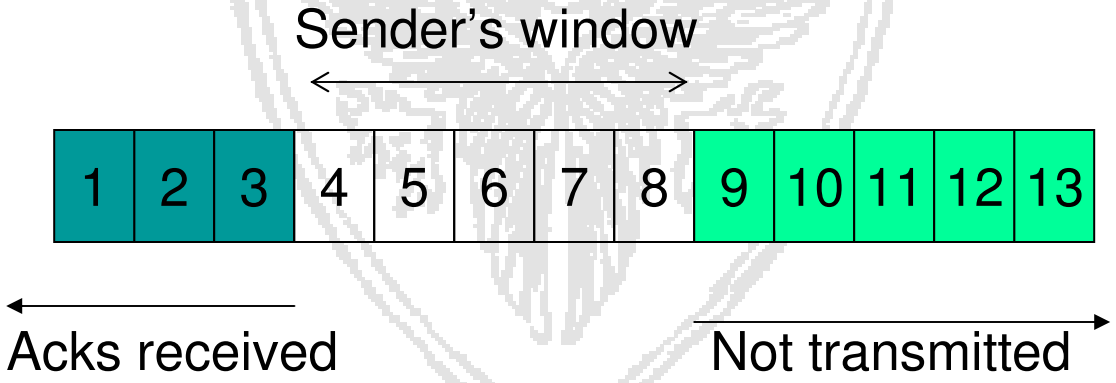
Un. of Rome "La Sapienza"

Chiara Petrioli †, Francesco Lo Presti

† *Department of Computer Science – University of Rome "Sapienza" – Italy*

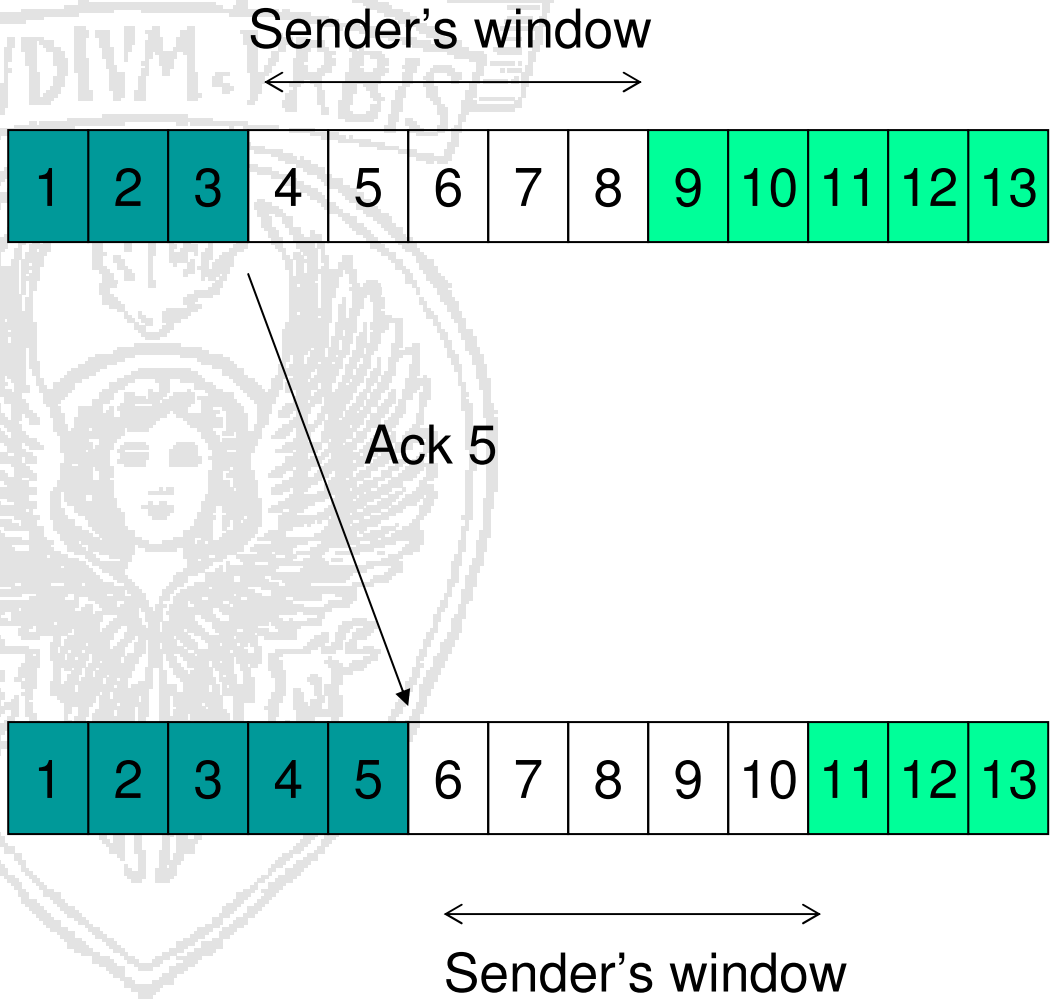
TCP- Window based flow control

- Protocollo basato su *sliding window*
- L'ampiezza della finestra è data dal valore minimo tra
  - receiver's advertised window – determinata dal receiver in base allo spazio disponibile nel buffer
  - congestion window – determinata dal sender, in base al feedback dalla rete



Window based flow control

- TCP window flow control è "self-clocking"
- Nuovi dati vengono inviati quando quelli inviati sono stati ricevuti (ack)
- Mantiene "equilibrio"

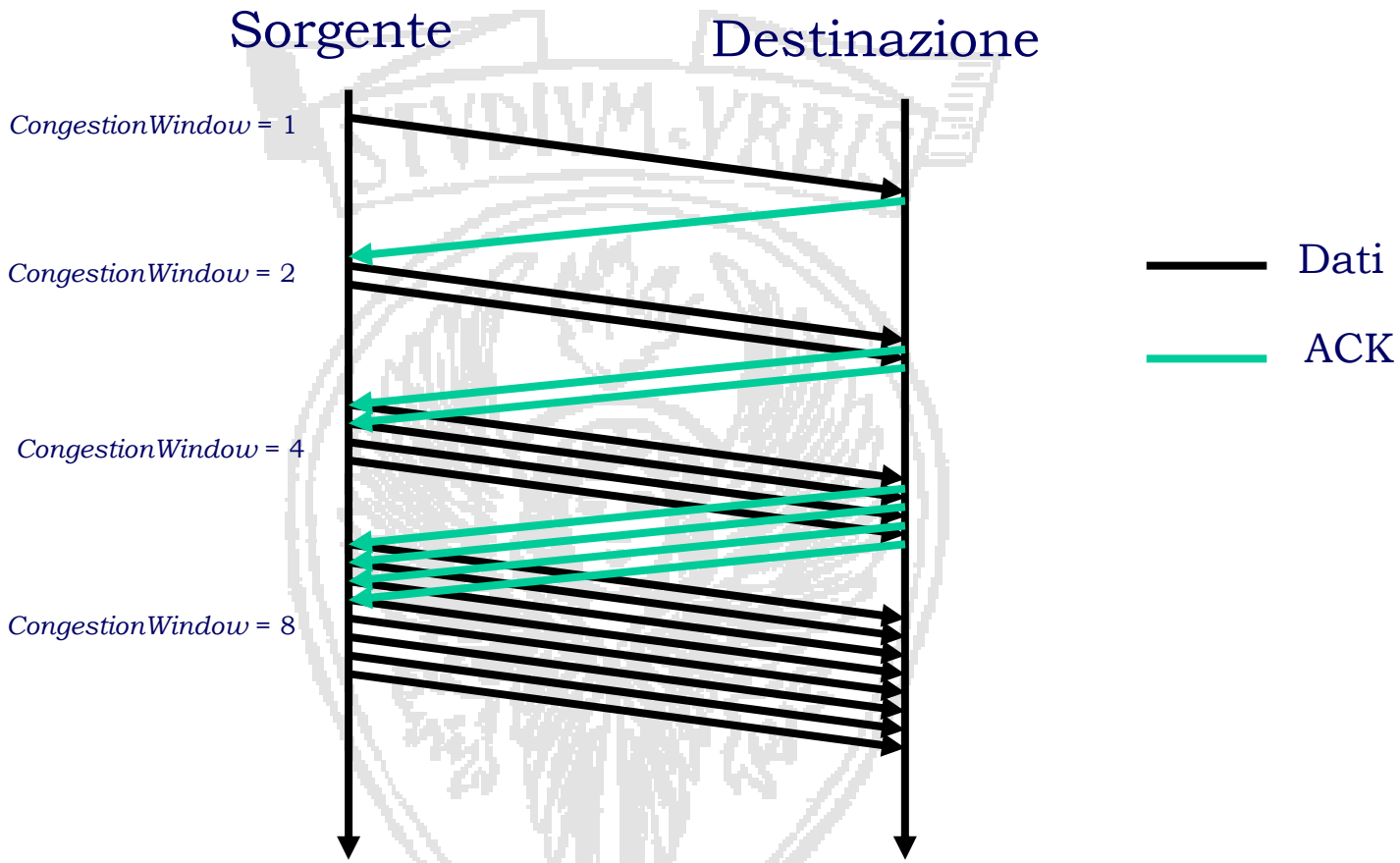


SAPIENZA
UNIVERSITÀ DI ROMA

Meccanismi del TCP

- Slow start
- Congestion avoidance
- Fast retransmit
- Fast recovery

Slow start

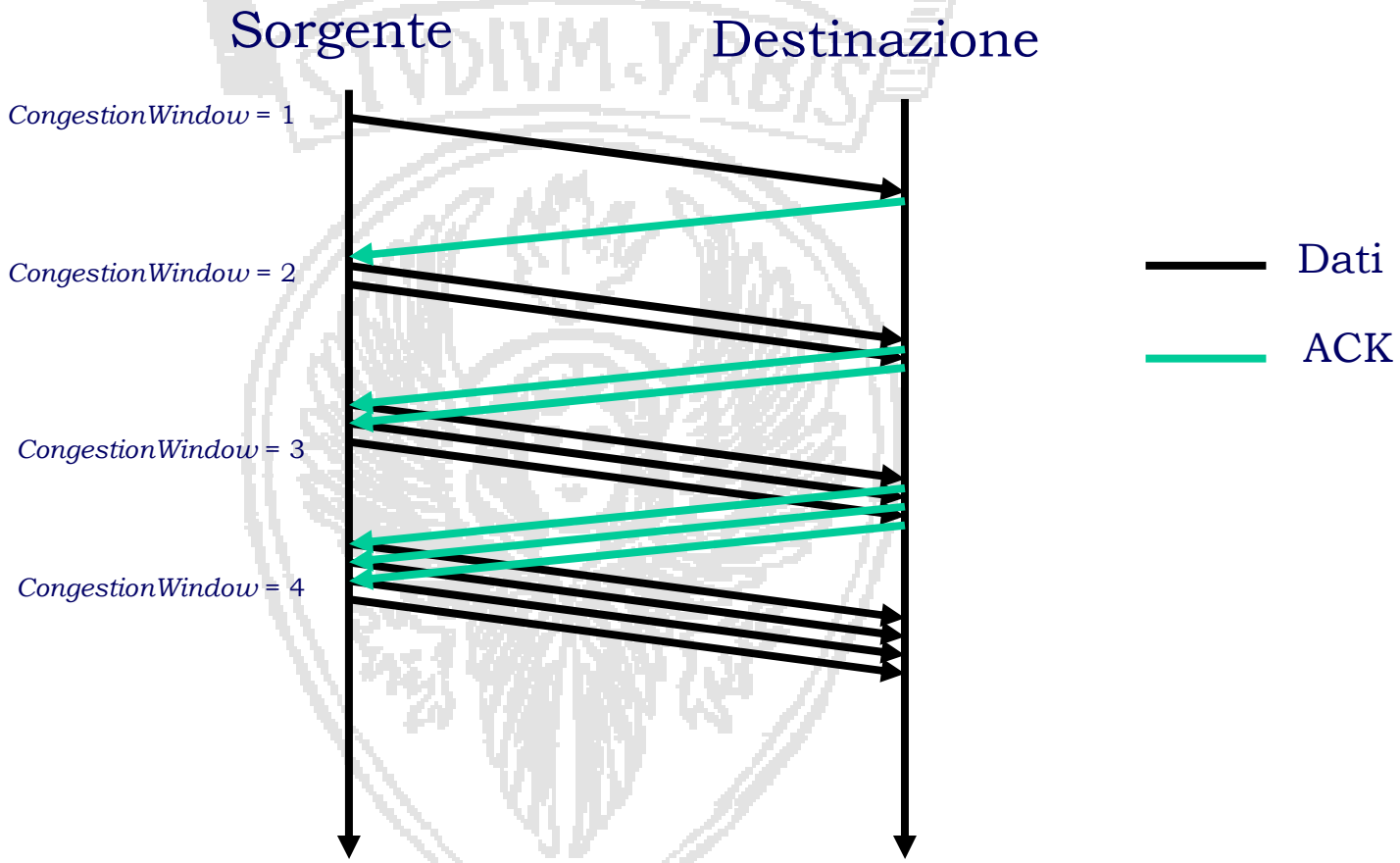


CongestionWindow = CongestionWindow +1

All'arrivo di ogni ACK

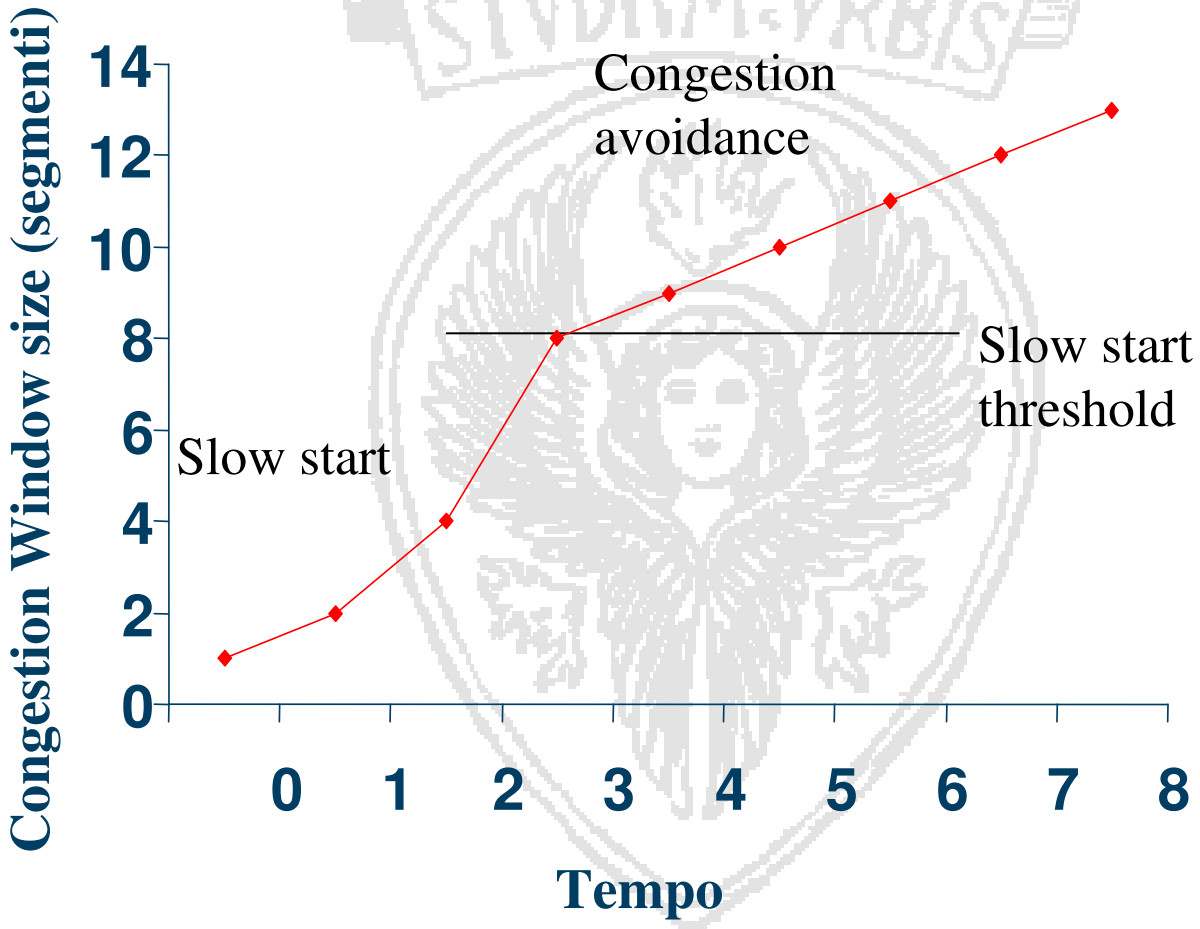
Congestion avoidance

ADDITIVE INCREASE

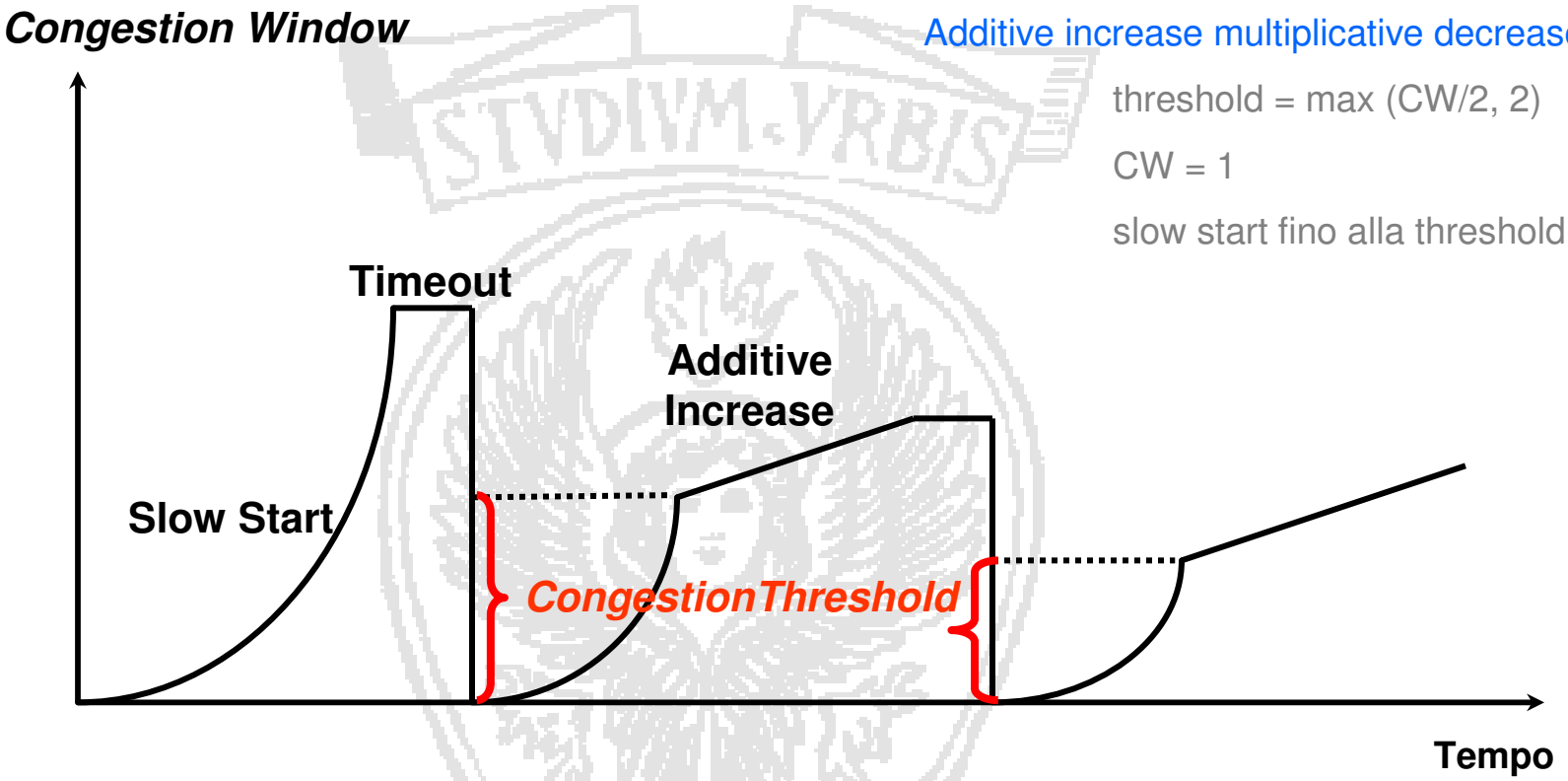


Incremento lineare

Slow start + Congestion Avoidance



Congestion control - timeout



Additive increase multiplicative decrease

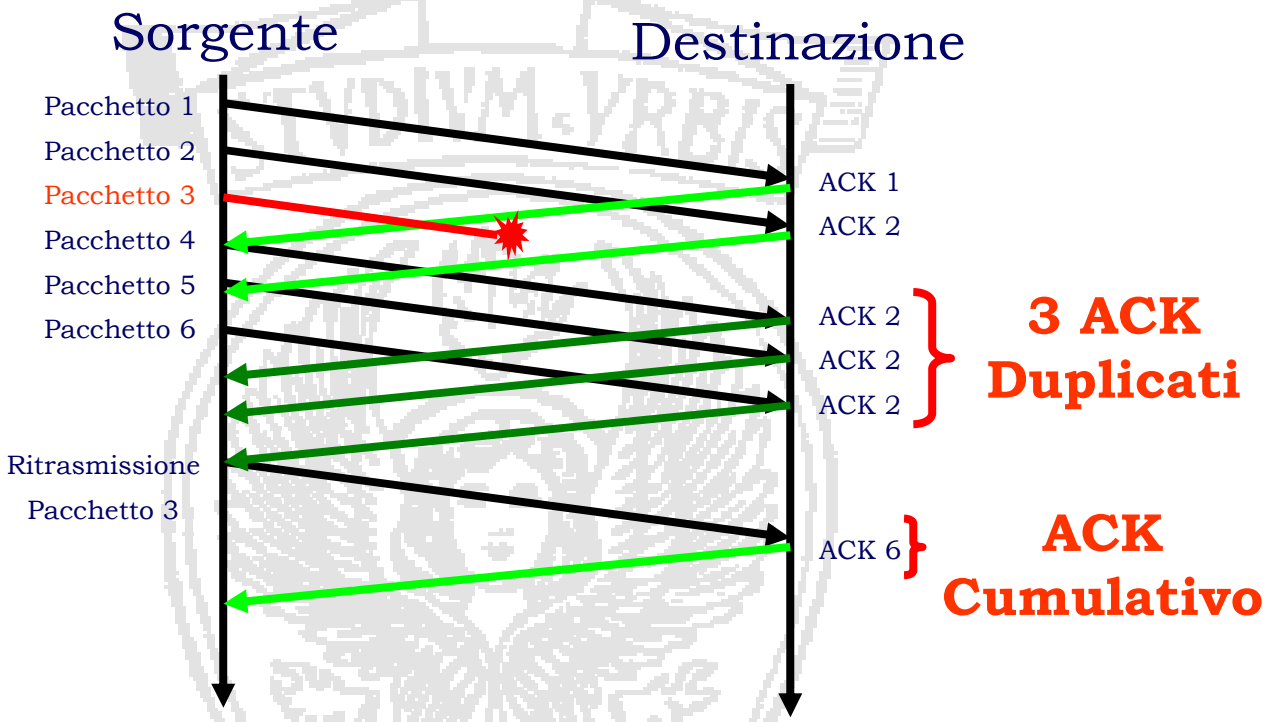
threshold = max (CW/2, 2)

CW = 1

slow start fino alla threshold

$$CongestionThreshold = \text{Max}(CongestionWindow/2, 2)$$

Fast retransmit (ack duplicati)



- Fast retransmit non aspetta lo scadere del timeout
  - Attivo quando la sorgente riceve multipli (>= 3) ack duplicati (prima del timeout)

SAPIENZA
UNIVERSITÀ DI ROMA

Timeout VS Fast retransmit

Differente dal timeout : slow start segue timeout

- timeout -> pacchetti non arrivano più a destinazione
- fast retransmit -> un pacchetto viene perso, ma i successivi arrivano a destinazione
- Non c'è necessità di eseguire slow start, perchè si è persa una piccola quantità di pacchetti

- Fast retransmit in genere è seguita dalla fase di Fast recovery

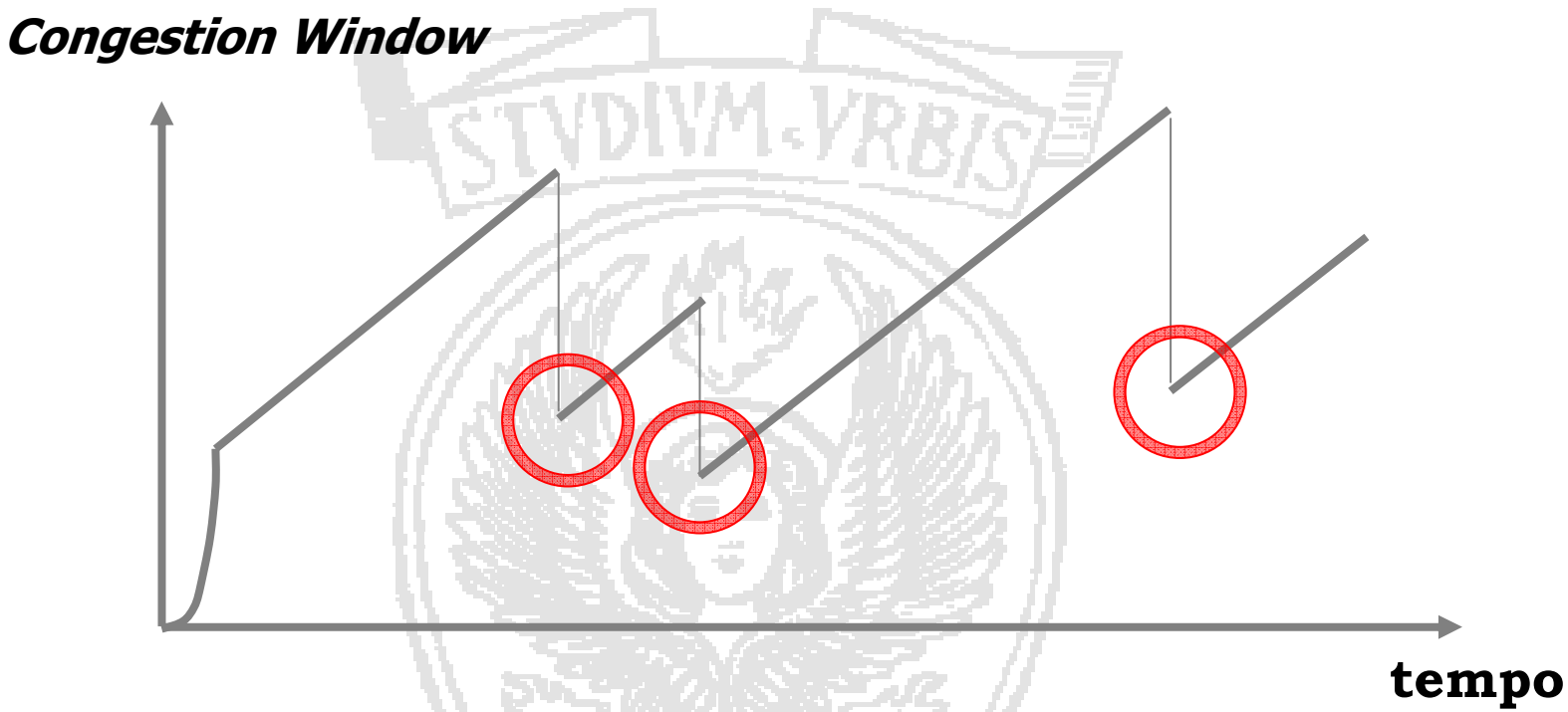
Fast Recovery

- ssthresh = cwnd/2
- cwnd = ssthresh + number of dupacks

- Ritrasmette il pacchetto perso (fast retransmit) e attende l'ack
- Quando riceve un nuovo ack (non di un pacchetto ritrasmesso) :
  - cwnd = ssthreh
  - esegue congestion avoidance

Congestion window dimezzata

Fast recovery



Stato di Congestione Leggera:
la rete è congestionata, ma alcuni pacchetti arrivano a destinazione. Con 3 DUPACK viene rilevata la perdita del pacchetto, si effettua il **Fast Retransmit** e si entra nella fase di **Fast Recovery.**

Livello di trasporto in ambito wireless

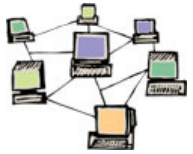
- La soluzione naturale sarebbe usare TCP
- Tuttavia sorgono dei problemi
  - TCP assume che quando si verifica un time-out o si perdono pacchetti è per via di congestione nei router della rete
    - ✓ Assume link affidabili (tipici dei link fissi)
    - ✓ In ambito wireless perdita di pacchetti o time-out possono anche essere dovuti a
      - » errori nella ricezione dei pacchetti (e.g. per fenomeni di fading)
      - » collisioni
      - » mobilità dei nodi che porta a temporanee disconnessioni

→ In questo diverso scenario è opportuno diminuire fortemente il trasmission rate quando avviene una perdita?

- ✓ Banda trasmissiva risorse preziosa, vorremmo usarla il piu' possibile senza congestionare la rete→ non opportuno se loss sporadici

→ Eterogeneità dei link wireless può portare a soluzioni diverse

- ✓ Satellite, reti cellulare o basate su AP (access Point), Reti Ad Hoc



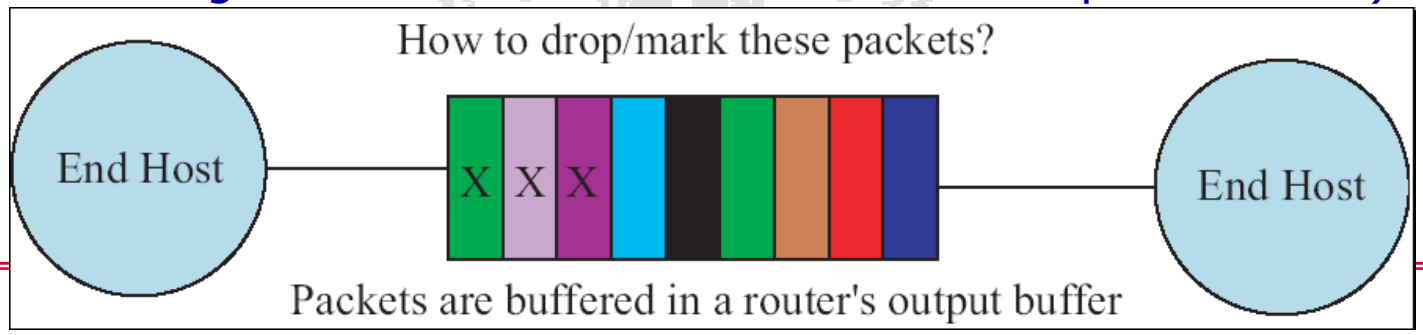
Vari approcci

- Problemi legati alla comunicazione sul canale radio possono essere risolti a livello link layer
  - ✓ Es. FEC, ARQ
  - ✓ Problema: mancanza di sinergia tra livello di trasporto e link-layer
    - – es. introduzione di ritardi per risolvere a livello link layer collissioni e perdita di pacchetti potrebbe far scattare time-out
- Approcci che 'separano' la parte della connessione fino al punto di accesso alla rete fissa e la parte di connessione dall'AP all'utente fisso
  - ✓ Non si mantiene end-to-end il protocollo
  - ✓ Richiesto intervento da 'nuove entità' in rete
- Meccanismi proposti per distinguere tra
  - ✓ perdita di pacchetti per congestione
  - ✓ perdita da canale radio etc…
  - ✓ diversi meccanismi di congestion control usati a seconda dei casi
  - ✓ Questi approcci possono a loro volta essere completamente end-to-end oppure prevedere un livello di supporto dai router della rete

Nel seguito vediamo delle soluzioni che fanno riferimento a quest'ultima categoria

Soluzione 1: ECN (explicit congestion notification)

- Fenomeni di congestione sono ben rappresentanti dal fatto che le code di alcuni dei router si caricano eccessivamente
  - Drop tail scheme (usato da TCP): quando sono arrivata a riempire il buffer di coda scarto i pacchetti → dropping fornisce info usate da TCP
  - Difetti dell'approccio standard:
    - ✓ Puo' caricare troppo comunque la rete, global synchronization
    - ✓ Non controlla i queuing delays
    - ✓ Unfairness (se ho un misto di pacchetti TCP e UDP l'effetto del congestion control di TCP ha effetto solo sui pacchetti TCP)



Random Early Discard (RED)

- Prevede che i pacchettipossano essere scartati con probabilità non nulla anche prima che la coda del nodo sia completamente riempita
  - Stima la average queue size
  - Droppa pacchetti con prob p_b
    - ✓ Riduce buffer overflow, fair (chi piu' Produce traffico ha piu' dropping)
    - ✓ Da indicazione sul livello di congestione
  - La stess… n uscita con una
  - Probabil… di congestione → ECN

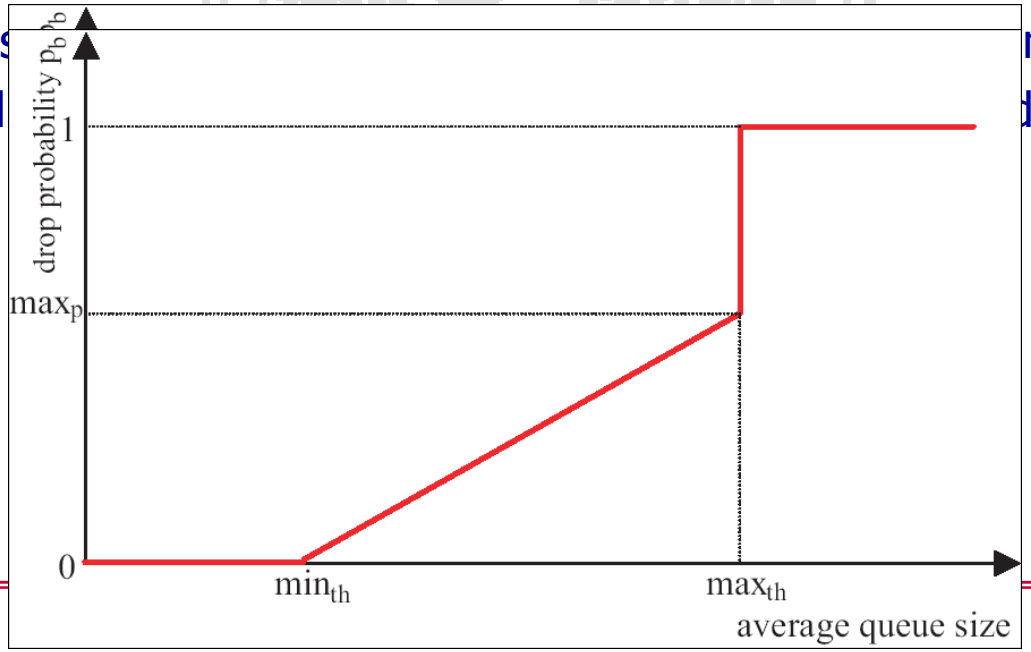
$$avg_q = (1 - w_q) \times avg_q + w_q q.$$

where

q is the instantaneous queue size,

avg_q is the average queue size,

w_q is the the time constant of the lowpass filter



ECN: marca il pacchetto Con probabilità p_b quando la dim della coda è tra min_{th} e min_{th}

ECN

- Quando riceve un pacchetto marcato il ricevitore setta un flag nell'acknowledgment che invia al sender, informandolo della congestione
- La ricezione di ACK con flag settato porta a effettuare congestion avoidance
- Coinvolge router e end host (e richiede modifiche)
  - Router
    - ✓ se un pacchetto è ECN capable e c'è congestione viene marcato altrimenti droppato secondo le regole di RED (se un pacchetto appartiene ad un flusso ECN capable è q.sa che nella fase di set up della connessione deve essere stabilita)
  - End host: lo stack viene modificato
    - ✓ Aggiunta di info nell'header TCP
      - » ECN-Echo flag (ECE) settato dal ricevitore nell'ACK per indicare presenza di congestione
      - » Congestion Window Reduced (CWR) flag settato dal sender per informare il ricevitore del fatto che la dimensione della finestra di congestione è stata modificata e si è risposto all'ECE
      - » ECN Echo flag viene settato negli ACK dal ricevitore fino a quando il ricevitore non riceve un pacchetto con CWR flag settato.

Fast retransmit, fast recovery

SAPIENZA
UNIVERSITÀ DI ROMA

Estensione di ECN per TCP over wireless

- Nel caso di ricezione di duplicate ACK si distingue tra i due casi
  - ✓ 3 DUPACK contiene indicazione di congestione
    - congestion avoidance
  - ✓ 3 DUPACK senza indicazione di congestione
    - approccio meno conservativo
      - » faster recovery
      - » la dimensione della finestra può essere diminuita ma non si effettua congestion avoidance (in modo da effettuare un incremento veloce della dimensione della finestra nel caso di perdita sporadica di pacchetti)

SAPIENZA
UNIVERSITÀ DI ROMA

TCP Westwood

- Example of an end-to-end solution
  - ✓ We want to fully maintain the end to end paradigm
  - ✓ No change needed in the TCP/IP stack (apart from inclusion of the new protocol) and header, no collaboration and changes needed in network
- Improvement of TCP Reno for both wireless and wired networks with advantages expecially in wireless networks
- The idea is to discriminate the cause of packet loss (e.g., congestion, wireless channel)
  - ✓ 'Reno' halves the congestion window after three duplicate ACKs
  - ✓ 'Westwood' tries to estimate the available bandwidth, tuning the slow start threshold and congestion window accordingly (*faster recovery*)

SAPIENZA
UNIVERSITÀ DI ROMA

TCP Westwood: how it works

- If I received an ACK at time t_k, ACKING a data segment d_k, this is an indication the available bandwidth is at least $b_k=d_k/(t_k-t_{k-1})$
- We use a filter to estimate an average available bandwidth

$$\hat{b}_k = \frac{\frac{2\tau}{t_k - t_{k-1}} - 1}{\frac{2\tau}{t_k - t_{k-1}} + 1}\hat{b}_{k-1} + \frac{b_k + b_{k-1}}{\frac{2\tau}{t_k - t_{k-1}} + 1} \quad (1)$$

- $\hat{b}_k$ =Avg bandwidth(k)
- If ACKs have a fixed interarrival rate with value around $\tau/10$ this maps to
- *Avg bandwdith (k)= .9 Avg bandwith(k-1)+.1*(b_k+b_{k-1}/2)*

SAPIENZA
UNIVERSITÀ DI ROMA

TCP Westwood: how it works

- If I received an ACK at time t_k, ACKING a data segment d_k, this is an indication the available bandwidth is at least $b_k=d_k/(t_k-t_{k-1})$
- We use a filter to estimate an average available bandwidth

$$\hat{b}_k = \frac{\frac{2\tau}{t_k - t_{k-1}} - 1}{\frac{2\tau}{t_k - t_{k-1}} + 1}\hat{b}_{k-1} + \frac{b_k + b_{k-1}}{\frac{2\tau}{t_k - t_{k-1}} + 1} \quad (1)$$

- $\hat{b}_k$ =Avg bandwidth(k)

•In reality: variable interarrival times

•When t_k-t_{k-1} increases the weight of the 'old' samples decreases

•Requires a sampling every $\tau/2$ –or a sampling frequency of $2/\tau$ (to filter out only components above $1/\tau$)←Nyquist. If an ACK does not arrive within this timeframe

it is assumed that $b_k=0$→ if no ACK arrives the avg bandwidth exponentially decreases

Challenges

- Duplicate ACK
  - ✓ means an out of order packet has been received
    - this segment should count for sake of bandwidth estimation
    - however I don't know which of the transmitted packets have been received
      - » Cannot use an exact (but an average) segment size
- Delayed ACKs
  - ✓ an ACK is sent back for every in sequence received segment OR if a 200ms timeout expires after reception of the last segment
    - For bandwidth estimation
    - How to we identify and deal with delayed ACKs?

Come si calcola il d_k

```
PROCEDURE AckedCount
cumul_ack=current_ack_SQN - last_ack_SQN
if (cumul_ack==0) /*duplicate ACK*/
        accounted_for=accounted_for+1;
        cumul_ack=1;
endif

if (cumul_ack >1)
        if (accounted_for*seg_size >= cumul_ack)
                accounted_for = accounted_for – cumul_ack/seg_size;
                cumul_ack=0;

        else if (accounted_for*seg_size < cumul_ack)
                cumul_ack=cumul_ack-accounted_for*seg_size;
                accounted_for=0;
        endif
endif
last_ack_SQNO=current_ack_SEQNO;
acked=cumul_ack;
return (acked);
```

Si considerano i DUPACK

Conta i segmenti non ACKed ma che
Sono già stati considerati per la stima della banda

Segmenti arrivati sono già stati considerati per la stima della banda

Stima del d_k
Usato per la stima corrente del b_k

Congestion control in Westwood

```
if nDUPACKs received
    ssthreshold=(BWE*RTTmin)/seg_size;
    if (cwin >ssthresh)
        cwin=ssthresh;
    endif
endif
```

Caso duplicate ACKs

Banda stimata

Normalizza per riportarsi in num segmenti

Si setta la dimensione della finestra ad un valore Che dipende dalla stima della banda disponibile

→Ci si aspetta che in situazioni di non congestione si diminuisca di una entità minore la dimensione della finestra di congestione

→ Sei entra una fase di congestion avoidance

Congestion control in Westwood

```
if course timeout expires
    ssthreshold=(BWE*RTTmin)/seg_size;
    if (ssthresh<2)
          ssthresh=2;
    endif
    cwin=1;
endif
```

Caso timeout

Banda stimata

Normalizza per riportarsi in num segmenti

Si setta la dimensione della finestra ad 1 e si parte con Slow start ma si è meno conservativi/più accurati nel settare ssthresh