



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



## Corso di Fondamenti di Reti di Telecomunicazioni LT - ELE / LM-TLC

### Lo strato di Trasporto

Lezione - 04



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

## Gruppo di Reti e Telematica Applicata



# Controllo di congestione



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



**Il controllo della congestione ha lo scopo di evitare (congestion control) o risolvere (congestion avoidance) eventuali situazioni di sovraccarico nella inter-rete, limitando il traffico offerto alla rete**

## **Difficoltà:**

- ✗ il protocollo IP (protocollo di rete) non possiede alcun meccanismo per rivelare e controllare la congestione**
- ✗ il TCP è un protocollo end-to-end e può rivelare e controllare la congestione solo in modo indiretto**
- ✗ la conoscenza dello stato della rete da parte delle entità TCP è imperfetta a causa dei ritardi (variabili) di rete**
- ✗ le entità TCP che usano la rete non cooperano tra loro, anzi competono per l'uso delle risorse distribuite**



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



- ✘ Il meccanismo “sliding window” per il controllo di flusso di TCP funziona da estremo ad estremo e quindi, in linea di principio, non può essere usato in modo efficiente per il controllo di congestione
- ✘ Cioè il meccanismo funziona per evitare che un trasmettitore veloce sovraccarichi un ricevitore lento, ma non tiene conto della congestione di rete; ovvero il “collo di bottiglia” può essere la rete e non il ricevitore
- ✘ Tuttavia seppure in modo implicito, e con alcune limitazioni, lo schema sliding window di TCP può proteggere, in caso di congestione, sia il destinatario che la rete





CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



**In caso di congestione infatti il controllo di flusso aiuta perchè:**

**Al mittente arriveranno, per una data larghezza di finestra, meno riscontri e quindi saranno emessi meno segmenti**

**Le misure di RTT fatte dal TCP per fissare il timeout permettono (quando la stima è fatta correttamente) di evitare ritrasmissioni inutili che porterebbero ad un aumento della congestione**

**Inoltre, il valore stimato di RTT può essere usato dal TCP ricevente come misura della congestione e quindi può aiutarlo a decidere opportunamente la larghezza della finestra da comunicare al TCP mittente**

**Infine, il meccanismo di ritrasmissione a intervalli crescenti (back-off) coopera per ridurre la congestione**



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



**Il controllo di flusso end-to-end del TCP riesce ad adattare il rate di emissione della sorgente in base al rate di arrivo degli ACK dei segmenti precedenti**

**Il rate di arrivo degli ACK è determinato dal collo di bottiglia nella rete, ovvero nel percorso andata e ritorno tra sorgente e destinazione; il collo di bottiglia può essere il ricevitore o la rete**

**I bottleneck in rete possono essere:**

- logici, causati dalla congestione nei router (nei buffer)
- fisici, causati dalla limitazione della banda nei collegamenti fisici (+ facili da gestire)
- dovuti al ricevitore, per la limitata capacità elaborativa del ricevitore



## **Il controllo di flusso TCP ha funzione auto-sincronizzante (self-clocking): il sender usa gli ACK come “clock” per l’invio di nuovi segmenti nella rete**

- il tasso di generazione dei segmenti dipende dal tasso di ricezione degli ACK e questo dipende a sua volta dal link più lento sul path sorgente destinazione (se il bottleneck è nella rete) o dalla velocità del ricevitore (se il bottleneck è nel ricevitore)

### **Bottleneck nella rete (vedi fig. seguente)**

- L’altezza (spessore) del collegamento tra sorgente e destinazione è proporzionale al data rate; sorgente e destinazione sono su reti ad alta capacità collegate da un link a bassa velocità che fa da bottleneck

(continua...)



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

## Gruppo di Reti e Telematica Applicata



- Ogni segmento è rappresentato da un rettangolo la cui area è proporzionale al numero di bit, perciò sui link lenti i segmenti si allungano e si abbassano (dimensione orizzontale=tempo)
- Il tempo  $P_b$  è la minima spaziatura tra i segmenti sul link più lento; all'arrivo dei segmenti a destinazione tale spaziatura viene mantenuta anche se aumenta il data rate perché il tempo di interarrivo non cambia, quindi  $P_r = P_b$
- Se la destinazione riscontra i segmenti appena arrivano (tempo di processamento uguale per tutti), allora la spaziatura degli ACK inviati è determinata dalla spaziatura di arrivo dei segmenti, quindi  $A_r = P_r$
- Dato che un time slot  $P_b$  può contenere un segmento dati, potrà a maggior ragione contenere un ACK, quindi  $A_b = A_r$



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

## Gruppo di Reti e Telematica Applicata



### Bottleneck nel ricevitore (vedi fig. seguente)

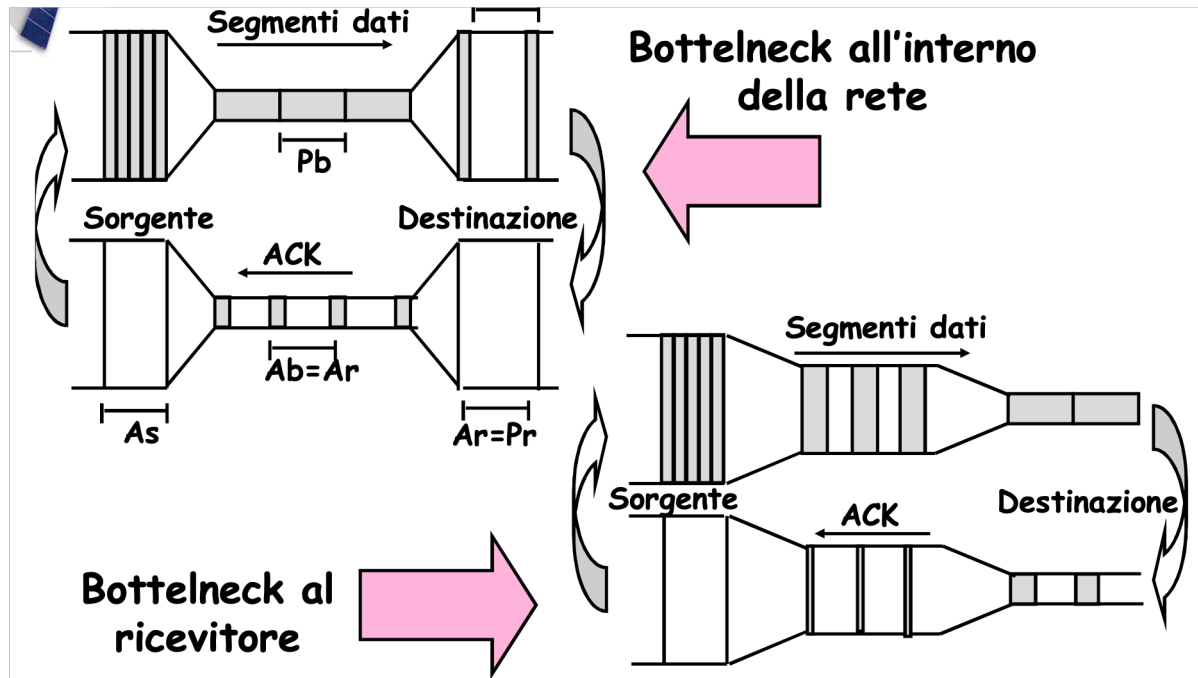
- Il ricevitore può assorbire i segmenti lentamente o per limiti della sua velocità di elaborazione o perché sovraccaricato da segmenti che gli giungono da altre connessioni
- In fig. assumiamo che il link più lento della rete sia relativamente veloce (circa metà del data rate della sorgente), mentre il pipe a destinazione sia stretto
- In tal caso, gli ACK saranno generati alla velocità di assorbimento della destinazione, così i segmenti verranno generati alla velocità con cui possono essere gestiti dalla destinazione



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata





CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



**Nota: la sorgente non ha modo di accorgersi se il tasso di arrivo degli ACK rifletta lo stato della rete (controllo di congestione) o della destinazione (controllo di flusso)**

## **Il controllo di flusso di TCP**

- non è in grado di distinguere il tipo di bottleneck di rete
- non può stabilire il tipo di contromisura più adatta

**TCP utilizza la stima di RTT come misura di congestione, lo scadere del timeout di ritrasmissione è considerato un sintomo di congestione**



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



Nelle implementazioni attuali, si considera lo scadere di un time-out come un sintomo di congestione delle risorse di interconnessione e si usano nuovi algoritmi per porre rimedio a tali situazioni

Algoritmo di Jacobson, Algoritmo di Karn + Backoff esponenziale li abbiamo già descritti, ora vedremo gli algoritmi di Slow Start e Congestion Avoidance, Fast Retransmit e Fast Recovery che agiscono sulla dimensione della finestra del sender

Meccanismo	TCP Berkeley	TCP Tahoe	TCP Reno
Stima varianza RTT	☒	☒	☒
Backoff espon. RTO	☒	☒	☒
Algoritmo di Karn	☒	☒	☒
Slow Start	☒	☒	☒
Congestion Avoidance	☒	☒	☒
Fast Retransmit		☒	☒
Fast Recovery			☒





### Controllo di congestione: Slow Start

**Slow Start (proposto da Jacobson) tende ad evitare l'insorgere di congestione durante la fase di avvio di una connessione, perciò espande gradualmente la finestra del sender**

**Regola l'emissione dei segmenti all'inizio di una connessione e ha lo scopo di raggiungere il ritmo di emissione a regime senza causare congestione**

**Si definisce una Congestion Window (cwnd) (misurata in segmenti) che tende ad aumentare progressivamente**

**La congestion window limita il valore della finestra fino a che questo non sia fissato dalla ricezione degli ACK**



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



L'ampiezza della finestra allowed window (awnd) usata dal sender in segmenti è:

$$\text{awdn} = \min [\text{RW}, \text{cwnd}]$$

**RW:** numero di crediti (in segmenti) concessi nell'ultimo ACK  
(=RCV.WND/MSS)

**cwnd:** congestion window (in segmenti)

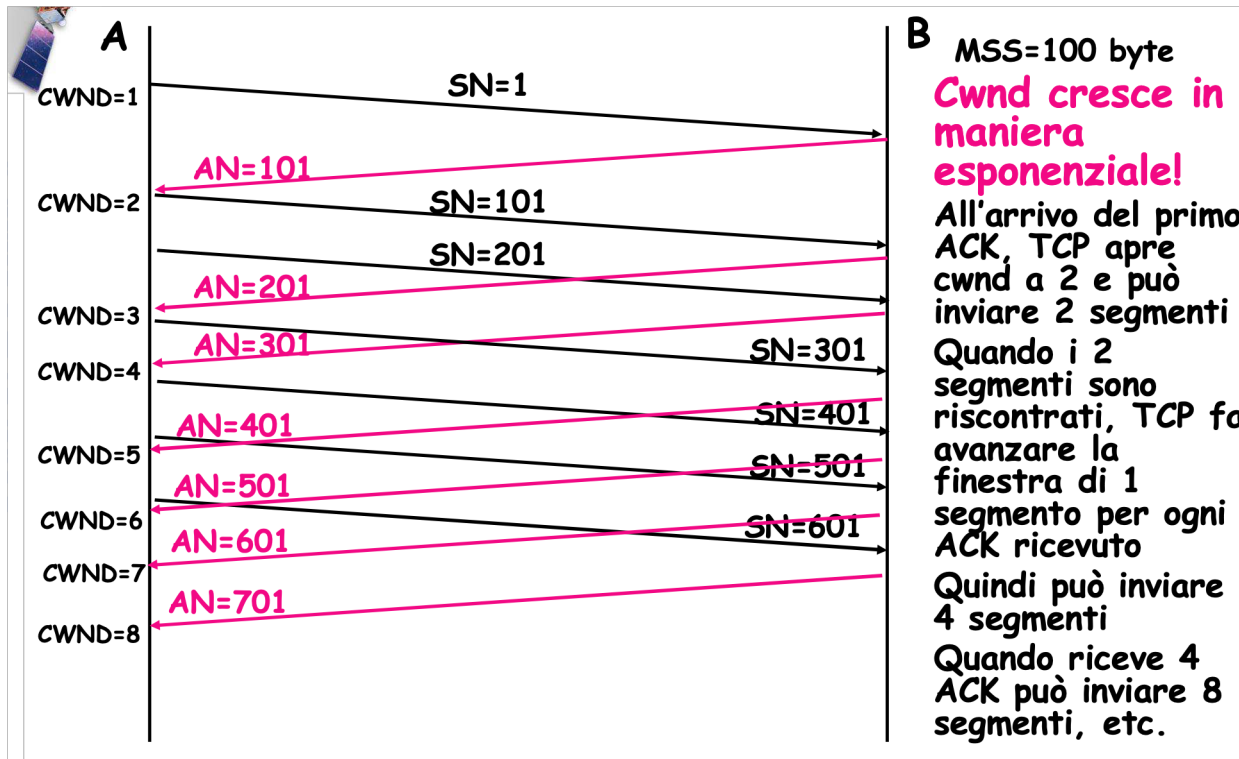
- per il primo segmento (allo start-up della connessione o alla ripartenza dopo un timeout)
  - cwnd=1 (si può partire anche da 2)
- per ogni segmento riscontrato, cwnd è incrementata di 1 segmento fino ad un valore massimo Mwnd
  - cwnd=Min [Mwnd, cwnd+1]



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata





CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



In regime di slow start: al tempo 0 viene trasmesso 1 segmento di lunghezza MSS, al tempo RTT 2 segmenti MSS, al tempo 2RTT 4 segmenti MSS e così via; al tempo  $k \cdot \text{RTT}$  sono spediti  $2^k$  segmenti che corrispondono a  $2^k \cdot \text{MSS}$  byte

In un tempo pari a  $N \cdot \text{RTT}$ , assumendo che si mantenga valida la condizione  $\text{CW} < \text{RW}$  con  $\text{CW} = \text{MSS} \cdot 2^k$ , il numero totale di byte trasmessi è pari a:

$$\begin{aligned} \sum_{k=1..N} (2^k \cdot \text{MSS}) &= \text{MSS} \cdot \sum_{k=1..N} 2^k = \\ &= \text{MSS} \cdot \frac{1-2^{N+1}}{1-2} = \text{MSS} \cdot (2^{N+1}-1) \text{ [byte]} \end{aligned}$$



## Gruppo di Reti e Telematica Applicata

Supponiamo che  $RW = m * MSS$  e che la rete non si congestioni, il regime di slow start finisce quando  $CW > RW$ , ovvero:

$$2^n * MSS > m * MSS$$

$$2^n > m$$

$$n > \log_2 m$$

dove  $n$  rappresenta il numero di RTT dopo il quale il numero di byte trasmessi è costante, cioè il numero di RTT necessari affinché  $CW$  raggiunga il valore di regime; definiamo  $n_{inf}$  è il numero intero immediatamente inferiore a  $\log_2 m$ :

$$n_{inf} = \lfloor \log_2 m \rfloor$$



## Gruppo di Reti e Telematica Applicata

Quindi, in regime di slow start il TCP trasmette un numero di byte pari a:

$$MSS * (2^{n_{inf}+1}-1) = MSS * (2^{\lfloor \log_2 m \rfloor +1}-1)$$

Per  $n > n_{inf}$  ad ogni RTT saranno trasmessi  $m * MSS$  byte, per cui in un tempo pari a  $h * RTT$  verranno trasmessi  $h * m * MSS$  byte

In generale il numero di byte trasmessi dal TCP in  $L * RTT$  secondi, nel caso in cui  $RW = m * MSS$ , è dato da:

$$\begin{aligned} \text{se } L \leq n_{inf} &\Rightarrow MSS * (2^{L+1}-1) \\ \text{se } L > n_{inf} &\Rightarrow MSS * (2^{\lfloor \log_2 m \rfloor +1}-1) + (L-n_{inf}) * m * MSS \end{aligned}$$



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

## Gruppo di Reti e Telematica Applicata



Nelle condizioni in cui il time-out non scatti mai e il ricevitore dia sempre la massima RW disponibile, il tempo impiegato per trasmettere un file di B byte è  $L \cdot RTT$ , con L intero:

se  $B \leq MSS \cdot (2^{n_{inf}} + 1 - 1)$ , cioè B è più piccolo del numero di byte che si possono trasmettere in regime di slow start:

$$B = MSS \cdot (2^{L+1} - 1) \quad \Rightarrow \quad L = \lceil \log_2(1 + B/MSS) - 1 \rceil$$

se  $B > MSS \cdot (2^{n_{inf}} + 1 - 1)$ , cioè la trasmissione avverrà parte in slow start parte in regime, in regime il numero di byte da trasmettere è:

$$B - MSS \cdot (2^{\lfloor \log_2 m \rfloor + 1} - 1) = (L - n_{inf}) \cdot m \cdot MSS$$
$$L = \lfloor \log_2 m \rfloor + \left\lceil \frac{B - MSS \cdot (2^{\lfloor \log_2 m \rfloor + 1} - 1)}{m \cdot MSS} \right\rceil$$



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

## Gruppo di Reti e Telematica Applicata



# Congestion Avoidance

**Regola l'ampiezza della finestra in caso di congestione durante la connessione (congestion avoidance)**

**Il meccanismo è innescato in caso di timeout (ritrasmissione) e consente di controllare il flusso di una sorgente per**

- ✓ **consentire l'esaurimento della congestione**
- ✓ **evitare un sovraccarico della rete**

**Usare la procedura di slow start in caso di congestione potrebbe essere troppo aggressiva perché la crescita della finestra è esponenziale**





CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



Procedura di congestion avoidance allo scadere di un timeout

Dimezzare il valore della cwnd

$$Cwnd = cwnd / 2$$

Ad ogni ACK si incrementa linearmente cwnd:

$$cwnd += 1 / cwnd$$

Inviare  $\min(RW, cwnd)$

Tuttavia, la procedura di congestion avoidance è usata in combinazione con la procedura di slow start nel modo seguente



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



## Procedura di slow start+congestion avoidance allo scadere di un timeout

- fissare una soglia per il passaggio da slow start a congestion avoidance uguale alla metà del valore corrente della congestion window

$$\text{ssthresh} = \text{cwnd} / 2$$

(in effetti,  $\text{ssthresh} = \min(\text{cwnd}/2, \text{RW})$ )

- fissare  $\text{cwnd}=1$  ed eseguire la procedura slow start finchè  $\text{cwnd} < \text{ssthresh}$  ; in questa fase,  $\text{cwnd}$  è incrementato di 1 per ogni ACK ricevuto (apertura esponenziale)
- se  $\text{cwnd} \geq \text{ssthresh}$ , parte la fase di congestion avoidance,  $\text{cwnd}$  è incrementato di uno ogni round trip delay ( $\text{cwnd} += 1/\text{cwnd}$ ) (apertura lineare)



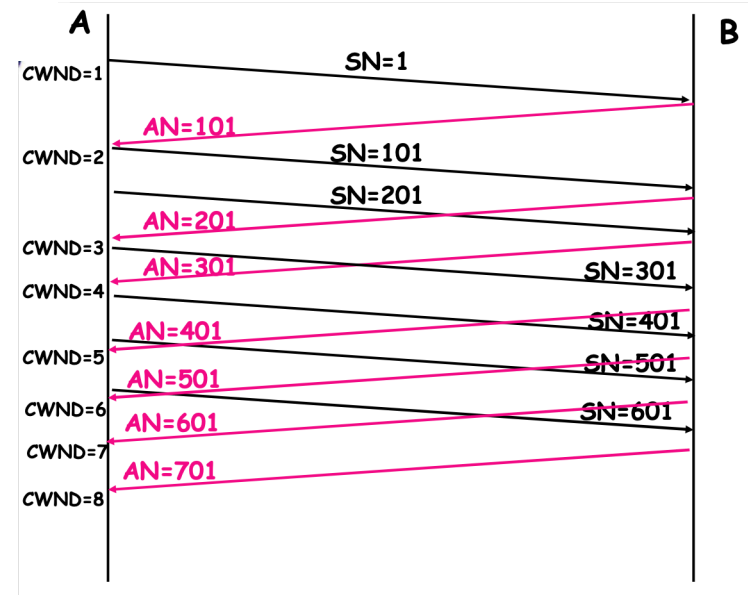
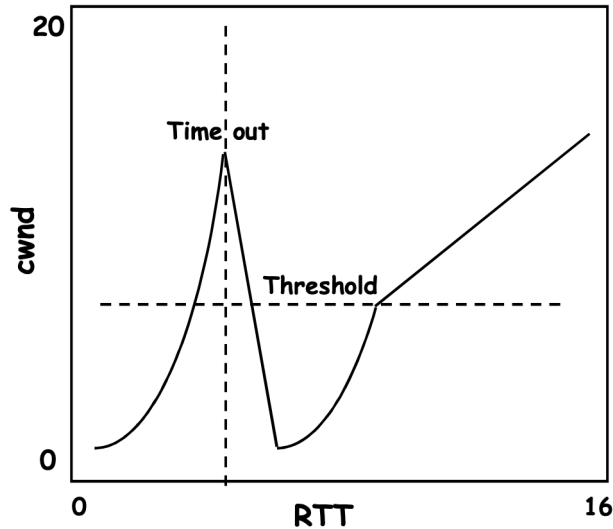
CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



## Congestion Avoidance

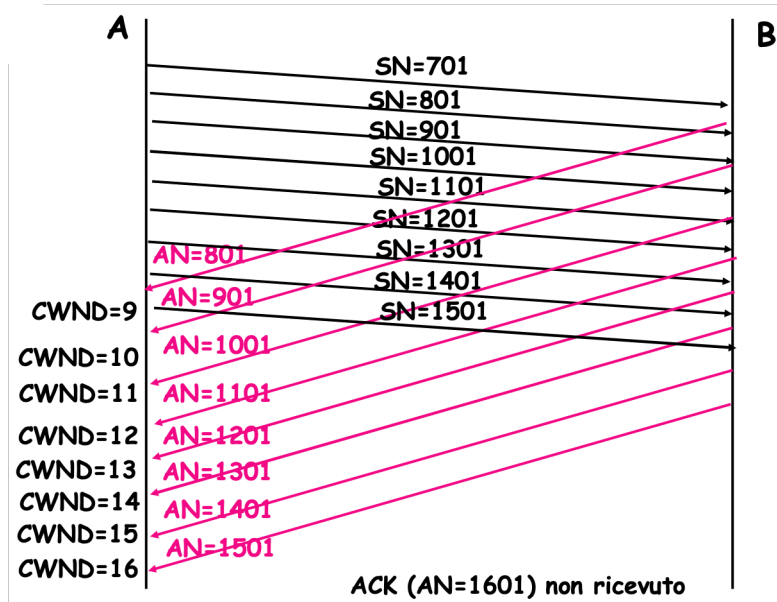




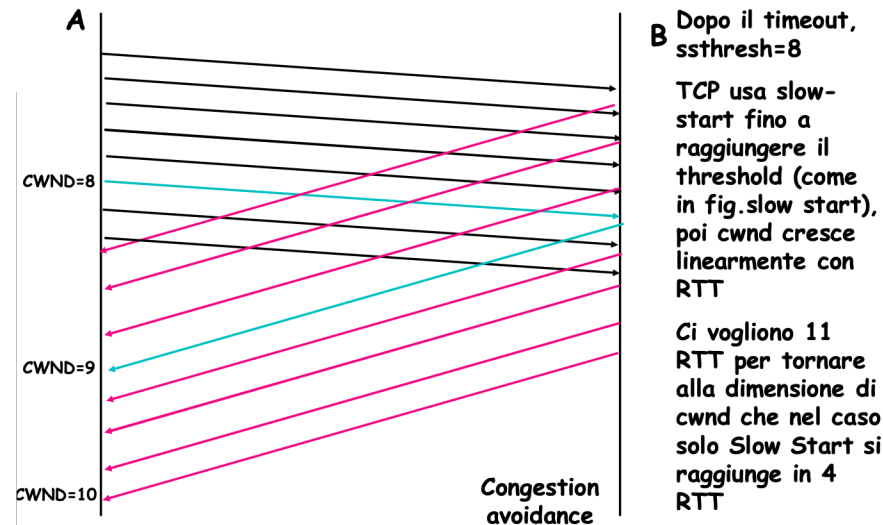
CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



## Congestion Avoidance: esempio





## Fast Retransmit

**Migliora le prestazioni se si perde un singolo segmento**

- ✗ **velocizza la ritrasmissione del segmento perso**
- ✗ **evita la ritrasmissione dei segmenti successivi già ricevuti con successo**

**Assunzioni su cui si basa la procedura (proposta da Jacobson nel '90) sono**

- ✗ **il ricevitore che riceve un segmento fuori sequenza emette immediatamente un ACK per l'ultimo segmento ricevuto in ordine; e continua a ripetere quest'ACK per ogni segmento successivo finché non riceve il segmento fuori sequenza**
- ✗ **quindi il TCP ricevente genera un ACK cumulativo per tutti i segmenti ricevuti in ordine nel frattempo**



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



Se una sorgente TCP riceve un ACK duplicato, questo può significare (1) che il segmento che segue quello riscontrato è stato ritardato e arriverà fuori sequenza, o (2) il segmento si è perso. Per essere sicuro che il caso in esame è il (2) e non l'(1) J. suggerì:

- ✗ la ricezione di tre ACK duplicati per lo stesso segmento è sintomo che il segmento successivo è perso
- ✗ in questo caso è molto probabile che il segmento sia perso e quindi conviene ritrasmetterlo senza aspettare la scadenza del timeout

La ritrasmissione del segmento inizia non appena sono ricevuti quattro ACK (3 duplicati e uno normale) del segmento precedente anche se il timeout non è scaduto



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



## Fast Recovery

Quando il TCP usa il fast retransmit per ritrasmettere un segmento, assume che il segmento sia perso anche se il timeout non è ancora scaduto, quindi deve prendere misure per combattere la congestione usando qualche meccanismo simile a slow start/congestion avoidance

La tecnica di Fast Recovery proposta da Jacobson è associata alla procedura di fast retransmit

- ✗ l'arrivo di ACK multipli assicura che i segmenti sono ricevuti abbastanza regolarmente, quindi la procedura normale di slow start/cong. avoidance potrebbe essere troppo conservativa



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

# Gruppo di Reti e Telematica Applicata



**Rispetto alla procedura normale di slow start/congestion avoidance, il fast retransmit evita la fase iniziale di slow start, cioè:**

- ✕ Si ritrasmette il segmento perso (fast retransmit)**
- ✕ Si dimezza la cwnd**
- ✕ Si procede aumentando linearmente la cwnd ad ogni ACK**





CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

## Gruppo di Reti e Telematica Applicata



La procedura è la seguente

- ✗ quando sono stati ricevuti tre ACK duplicati
- si pone:

$$\text{sstresh} = \text{cwnd} / 2$$

- viene ritrasmesso il segmento perduto
- per tener conto dei segmenti già ricevuti (nella cache del ricevitore con numeri di sequenza successivi a quello ritrasmesso) si pone:

$$\text{cwnd} = \text{sstresh} + 3$$



CULTURE

Calabria University Laboratory on  
Telecommunication Research  
Engineering

## Gruppo di Reti e Telematica Applicata



ogni volta che arriva un ulteriore ACK duplicato (per lo stesso segmento), il valore di `cwnd` viene incrementato di 1 e trasmesso (se possibile) un segmento (tiene conto di eventuali altri Ack duplicati in viaggio nella rete)

quando viene ricevuto un ACK (riscontro cumulativo del segmento perso più altri)

– si pone:

$$\text{cwnd} = \text{ssthresh}$$

e si entra nella fase di congestion avoidance