



Il livello trasporto

Il protocollo TCP: Il controllo della congestione Modellazione dei ritardi TCP

Ing. Nadia Ranaldo

1



Approcci al controllo della congestione

- La congestione è una condizione di forte rallentamento causata da un sovraccarico di pacchetti in uno o più router
- Si manifesta con
 - Aumento dei ritardi (tempo di accodamento nei router)
 - Perdita di datagram (code limitate sui router)
- Due sono gli approcci più diffusi al controllo della congestione:
 - Controllo della congestione assistito dal livello rete
 - Controllo della congestione end-to-end

Controllo della congestione end-to-end

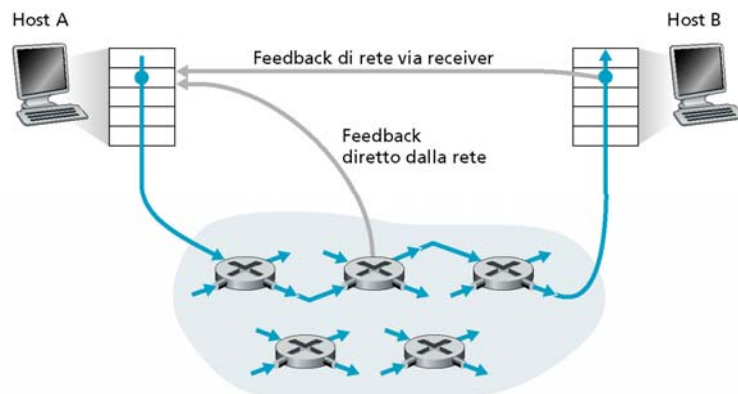
- Non vi è un feed-back esplicito dal livello rete
- La congestione è rilevata dagli host osservando il comportamento della rete
 - (es. perdite di pacchetti o ritardi)
- Approccio seguito dal protocollo TCP

2



Controllo della congestione assistito dal livello rete

- I router forniscono un feed-back esplicito agli host sullo stato di congestione della rete
 - Ad es. un bit che indica presenza di congestione (ATM)
 - Accordo tra mittente e router sulla banda da impiegare



3



Controllo della congestione in TCP (1)

- Controllo end-to-end (nessuna assistenza dalla rete)
- Si rileva congestione allo scadere di un timer o alla ricezione di tre ACK duplicati
- Velocità di trasmissione in funzione della congestione di rete percepita
 - Limitata dalla dimensione della finestra di congestione, **Congwin**
 - Corrispondente a $w = n$. di segmenti
 - La qtà di dati non riscontrati è limitata dal min. tra CongWin e RcvWindow



- In ogni RTT sono inviati w segmenti, ognuno con MSS byte:

$$\text{throughput} = \frac{w * \text{MSS}}{\text{RTT}} \text{ Byte/sec}$$

4

Controllo della congestione in TCP (2)

- L'algoritmo opera in due fasi
 - slow start
 - congestion avoidance
- Le variabili importanti sono:
 - **Congwin**
 - **threshold**: definisce il limite superiore della fase di slow-start e l'inizio della fase congestion avoidance

5

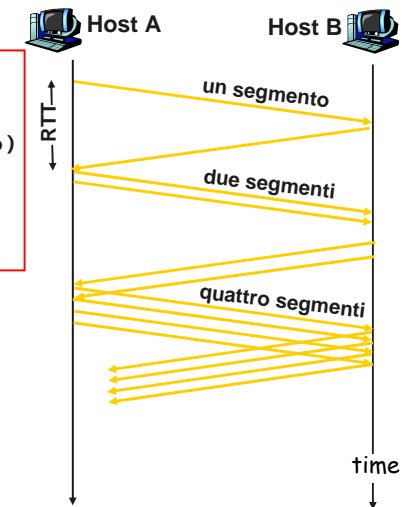
Algoritmo slow start

Algoritmo slowstart

```

inizialmente: Congwin = 1*MSS
for (ogni segmento riscontrato)
    Congwin += MSS
until (perdita o
    CongWin > threshold)
    
```

- Se non si rileva congestione si cerca di sfruttare meglio la banda disponibile
- Incremento esponenziale (per RTT) della dimensione della finestra
 - CongWin raddoppia a ogni RTT



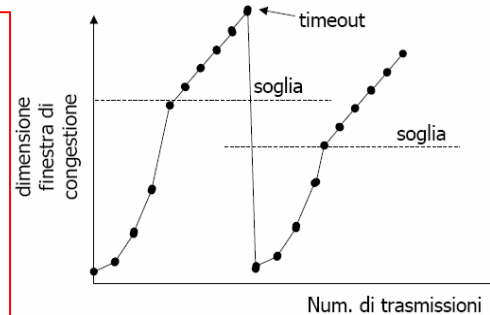
6

Congestion avoidance

Congestion avoidance

```

/* continua */
/* Congwin > threshold */
while (not perdita) {
    ogni w segmenti riscontrati:
        Congwin+=MSS
}
threshold = Congwin/2
Congwin = 1*MSS
esegui slowstart
    
```

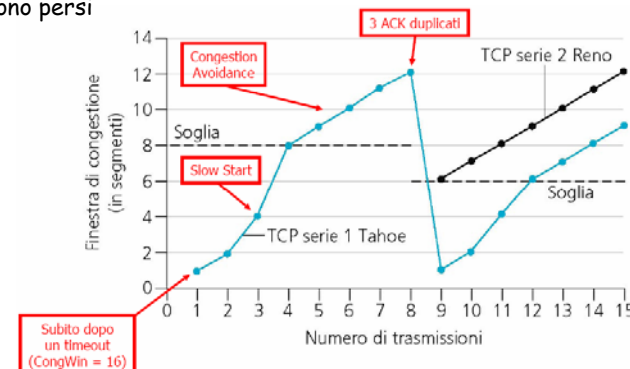


- $CongWin > threshold \Rightarrow$ incremento lineare di CongWin
 - Aumenta di 1 segmento ogni RTT
- Perdita \Rightarrow il valore corrente di CongWin viene riportato a 1 segmento

7

Ripristino rapido

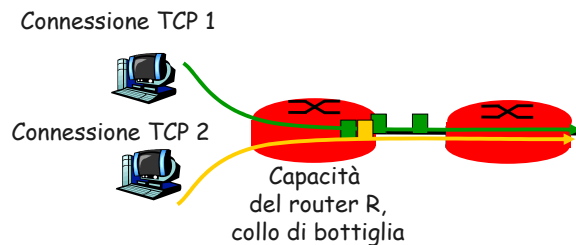
- Se la perdita è segnalata da un timeout si entra nella fase di slow start fino a raggiungere $threshold = CongWin/2$
- Se la perdita è segnalata dalla ricezione di tre ACK duplicati, $threshold = CongWin/2$, $CongWin = threshold$ e si riparte con la fase di congestion avoidance (**ripristino rapido**)
 - **TCP Reno**: la rete ha la capacità di consegnare dei segmenti, anche se alcuni sono persi



8

Fairness (Equità) (1)

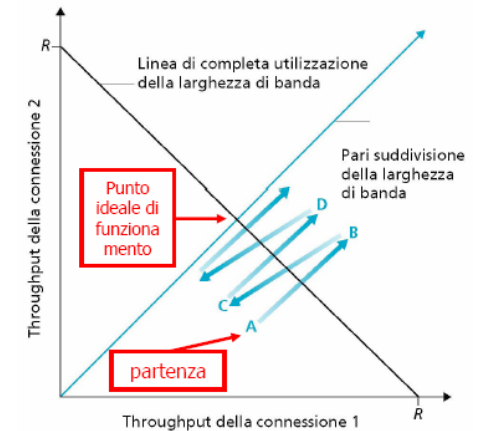
- K connessioni TCP condividono lo stesso router
- R è la capacità del router che costituisce il collo di bottiglia del sistema (tutti gli altri collegamenti non sono congestionati)
 - Un meccanismo di controllo della congestione è equo (*fair*) se garantisce una frequenza trasmissiva media di ciascuna connessione di R/K
- **L'algoritmo di controllo della cong. del TCP garantisce fairness? SI**
- Es. due connessioni con gli stessi valori di MSS e RTT, e quindi ugual finestra di congestione (non si consideri la partenza lenta)



9

Fairness (Equità) (2)

- Obiettivo è ottenere throughput aggregato vicino all'intersezione tra la bisettrice e la linea di massimo utilizzo della banda
- Si suppone che ad un certo istante le due connessioni realizzino i throughput corrispondenti al punto A
- Non si verificheranno perdite ($\ll R$) e quindi le due connessioni aumenteranno la loro finestra di 1 MSS per RTT
- Superato R (punto B) si può verificare la perdita di un pacchetto, quindi le finestre verranno dimezzate (punto C)
- La banda aggregata può fluttuare lungo la linea di pari suddivisione e c'è convergenza verso di essa, indipendentemente dal punto di partenza delle due connessioni



10

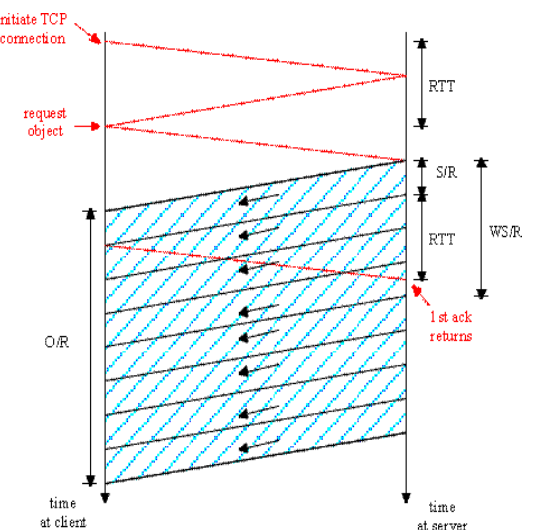
Modellazione dei ritardi TCP

- Quanto tempo occorre per ricevere un oggetto da un server Web dopo aver inviato una richiesta? (Latenza)
- Supponendo che non ci sia congestione (no perdite o ritrasmissioni), la latenza è influenzata da:
 - Setup della connessione
 - Slow start
 - Tempo di trasmissione dell'oggetto
- Notazioni
 - O: dimensione in byte dell'oggetto da trasferire
 - S: Massima dimensione dei segmenti in bit (MSS)
 - R: velocità di trasmissione dal server al client in bps
- **In assenza di vincoli sulla finestra di congestione** la latenza è:
 - Un RTT per i primi due segmenti
 - Nel terzo segmento può essere inserita la richiesta
 - Dopo 2 RTT il client inizia a ricevere

11

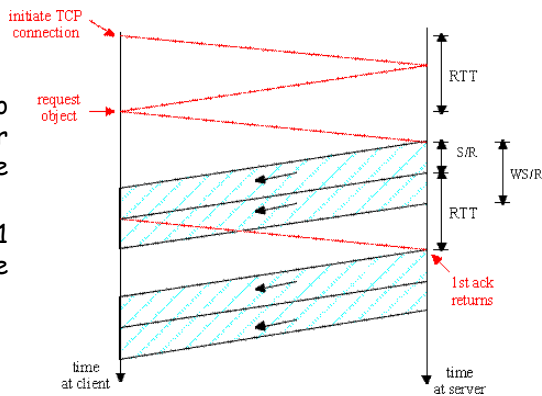
Finestra di congestione statica (1)

- Caso di finestra statica di dimensione W
- Il server non può avere più di W segmenti senza riscontro
- Dopo aver inviato W segmenti, trasferisce un segmento per ogni riscontro ricevuto dal client
- Primo caso:
 - $WS/R > RTT + S/R$
- Il server riceve il riscontro per il 1° segmento prima di completare la trasmissione della finestra (W segmenti)
- **Latenza = 2 RTT + O/R**



Finestra di congestione statica (2)

- Secondo caso:
- $WS/R < RTT + S/R$
- Il server riceve il riscontro per il 1° segmento dopo aver completato la trasmissione di W segmenti
- Il server rimane in stallo K-1 volte dove $K = n$. di finestre che copre O
- $K = O/WS$



$$\text{latenza} = 2RTT + O/R + (K-1)[S/R + RTT - WS/R]$$

13

Finestra di congestione dinamica (1)

- Slow Start:
- La finestra di congestione raddoppia ogni RTT
- K = numero di finestre che copre l'oggetto:

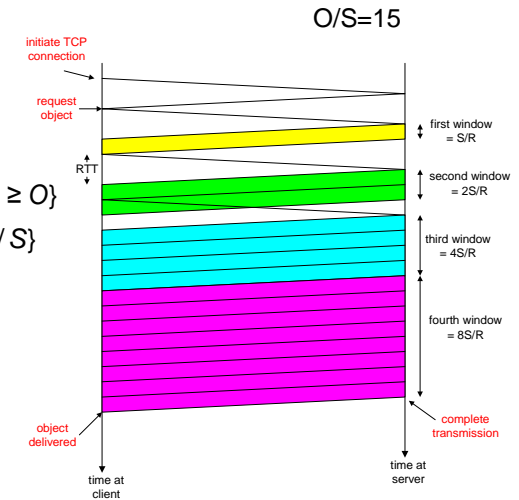
$$K = \min\{k : 2^0 S + 2^1 S + \dots + 2^{k-1} S \geq O\}$$

$$= \min\{k : 2^0 + 2^1 + \dots + 2^{k-1} \geq O/S\}$$

$$= \min\{k : 2^k - 1 \geq \frac{O}{S}\}$$

$$= \min\{k : k \geq \log_2(\frac{O}{S} + 1)\}$$

$$= \log_2(\frac{O}{S} + 1)$$



14

Finestra di congestione dinamica (2)

- Dopo aver trasmesso i segmenti di una finestra, il server può andare in stallo, in attesa di riscontro
- Dopo la trasmissione della k-esima finestra, la durata dello stallo è
 - $S/R + RTT$ è l'istante in cui riceve il 1° riscontro
 - $2^{k-1}S/R$ è il tempo per trasmettere la finestra k-esima

$$\frac{S}{R} + RTT - 2^{k-1} \frac{S}{R}$$

- Se tale quantità risulta negativa, deve essere posta a zero nel calcolo della latenza complessiva

$$\text{latenza} = \frac{O}{R} + 2RTT + \sum_{p=1}^P \text{idleTime}_p$$

$$= \frac{O}{R} + 2RTT + \sum_{k=1}^P \left[\frac{S}{R} + RTT - 2^{k-1} \frac{S}{R} \right]$$

15

Finestra di congestione dinamica (3)

- Con una derivazione analoga a K si ottiene che il numero di volte Q in cui il server andrebbe in stallo se l'oggetto contenesse un numero infinito di segmenti è:

$$Q = \max\left\{k : RTT + \frac{S}{R} - \frac{S}{R} 2^{k-1} \geq 0\right\} = \max\left\{k : 2^{k-1} \leq 1 + \frac{RTT}{S/R}\right\}$$

$$= \max\left\{k : k \leq \log_2\left(1 + \frac{RTT}{S/R}\right) + 1\right\} = \left\lfloor \log_2\left(1 + \frac{RTT}{S/R}\right) \right\rfloor + 1$$

$$\text{Latency} = 2RTT + \frac{O}{R} + P \left[RTT + \frac{S}{R} \right] - (2^P - 1) \frac{S}{R}$$

$$P = \min\{Q, K-1\}$$

- Lo slow start incide in modo significativo sulla latenza quando la dimensione dell'oggetto è relativamente piccola e RTT relativamente grande

16