Reti di Calcolatori:
Internet, Intranet e Mobile Computing
a.a. 2007/2008

http://www.di.uniba.it/~lisi/courses/reti/reti0708.htm

dott.ssa Francesca A. Lisi
lisi@di.uniba.it

Orario di ricevimento: mercoledì ore 10-12

dib

Sommario della lezione di oggi: Lo strato di trasporto (3/3)

- Servizi e protocolli dello strato di trasporto
- Multiplazione e demultiplazione delle applicazioni
- Trasporto senza connessione: UDP
- Trasporto con connessione: TCP
- Il controllo della congestione nel TCP

Principi di controllo della congestione

Congestione:

- informalmente: "troppe risorse che inviano troppi dati troppo velocemente rispetto alle capacità di manipolazione della rete"
- diversa dal controllo di flusso!
- sintomi:
  - pacchetti persi (buffer overflow nei router)
  - lunghi ritardi (accodamento nei buffer dei router)
- uno fra i primi 10 problemi + importanti nelle reti!
- *Throughput*: velocità di arrivo byte a destinazione

Approcci al controllo della congestione

End-end:

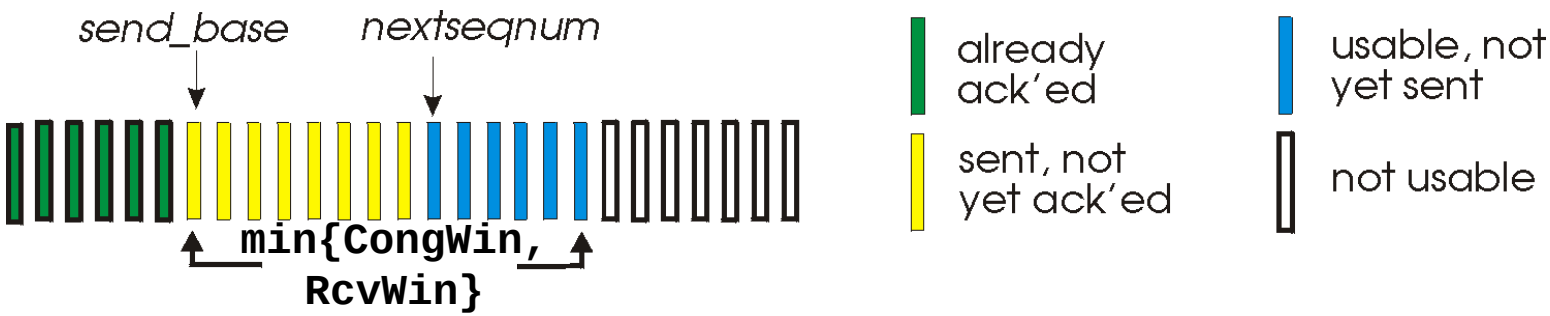
- nessun esplicito feedback dalla rete
- congestione inferita da perdita o ritardo osservata da terminale
- approccio seguito da TCP

Network-assisted:

- i router forniscono feedback ai terminali
  - bit singolo che indica congestione (SNA, DECbit, TCP/IP ECN, ATM)
  - velocità esplicita a cui il mittente dovrebbe inviare

Controllo della congestione del TCP

- La finestra di congestione (variabile **CongWin**) impone un ulteriore limite sul ritmo di invio al mittente:
- **LastByteSent-LastByteAck <= min{CongWin,RcvWin}**



- Supponendo **CongWin** = w, in cui ciascuno dei w segmenti consta di MSS byte inviati in un unico RTT:

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

Controllo della congestione del TCP: principi di base

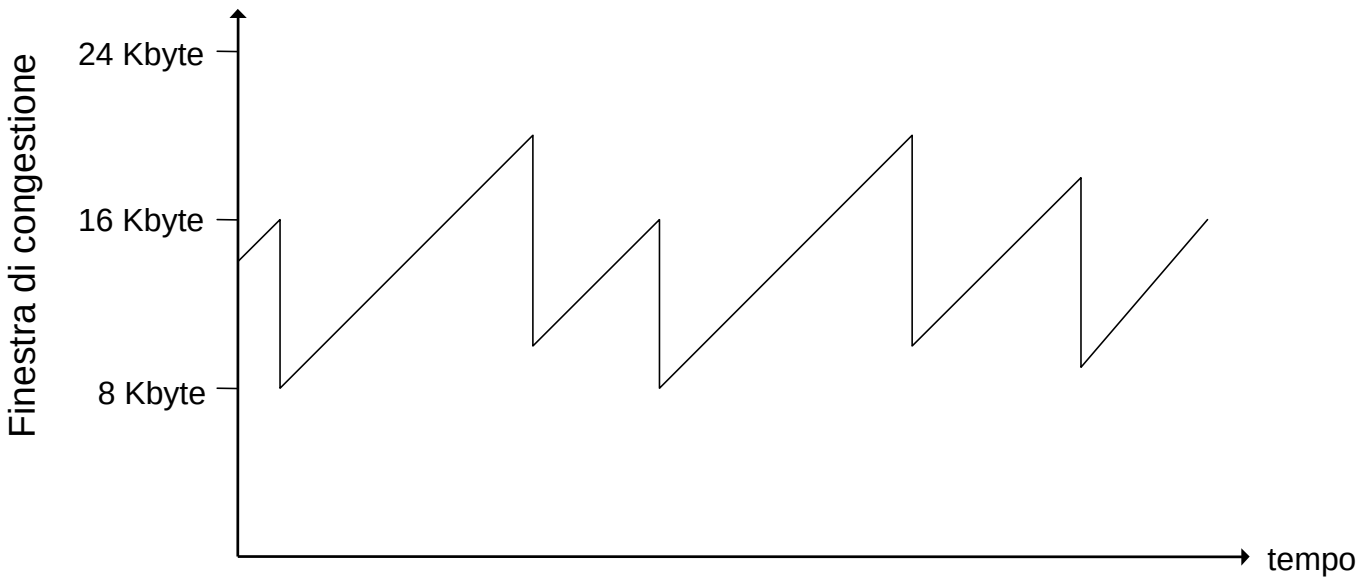
- *Idealmente* il mittente trasmette il più veloce possibile (**CongWin** il più grande possibile) senza perdite
- *In realtà* il mittente trasmette con velocità variabile:
  - *incrementa* **CongWin** fino a causare perdite (congestione)
  - quando si verifica un evento di perdita, *decrementa* **CongWin**
  - torna ad incrementare **CongWin**, etc.

- due "fasi"
  - partenza lenta
  - prevenzione della congestione
- variabili importanti:
  - **CongWin**: definisce la grandezza in segmenti della finestra di congestione
  - **threshold:** definisce la soglia fra fase di partenza lenta e fase di controllo della congestione

Controllo della congestione del TCP:
Incremento additivo/decremento moltiplicativo

Decremento moltiplicativo:
riduce a metà **CongWin** dopo un evento di perdita

Incremento additivo:
aumenta **CongWin** di 1 MSS a ogni RTT in assenza di eventi di perdita: *sondaggio*

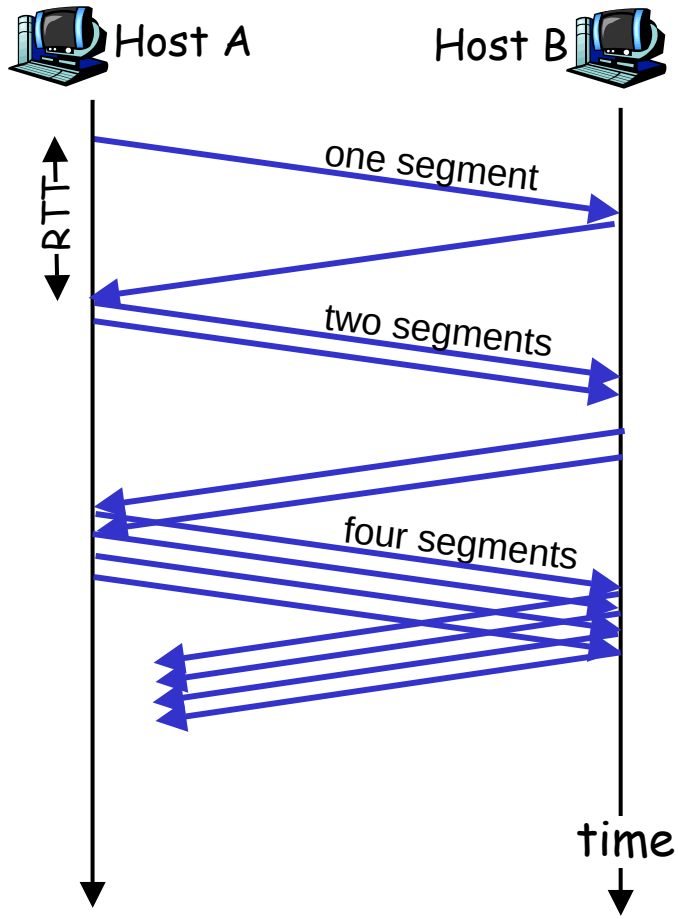


Controllo di congestione AIMD

Controllo della congestione del TCP: partenza lenta

algoritmo

initialize: CongWin = 1
for (each segment ACKed)
    CongWin++
until (loss event OR
    CongWin > threshold)

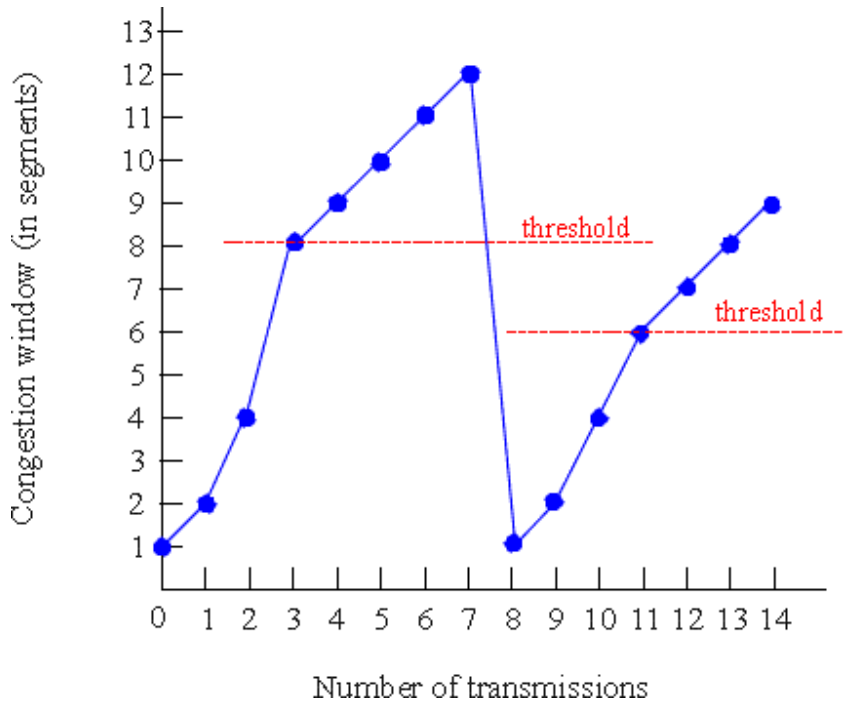


- Crescita esponenziale (per RTT) nella grandezza della finestra (non così lento!)
- evento di perdita: timeout (TCP Tahoe) e/o tre ACK duplicati (TCP Reno)

Controllo della congestione del TCP: prevenzione della congestione

algoritmo

```
/* slowstart is over        */
/* CongWin > threshold */
Until (loss event) {
  every w segments ACKed:
      CongWin++
  }
threshold = CongWin/2
CongWin = 1
perform slowstart[1]
```



1: TCP Reno salta lo slowstart (recupero veloce) dopo tre ACK duplicati

Controllo della congestione del TCP: riassumendo

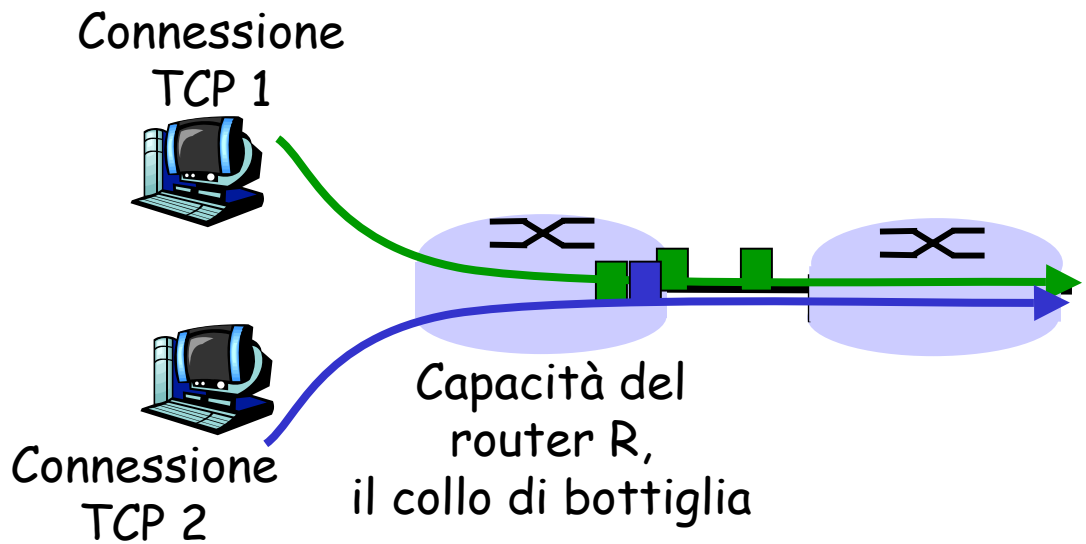
- Quando **CongWin** è sotto la soglia (**Threshold**), il mittente è nella fase di partenza lenta; la finestra cresce in modo esponenziale.
- Quando **CongWin** è sopra la soglia, il mittente è nella fase di congestion avoidance; la finestra cresce in modo lineare.
- Quando si verificano tre ACK duplicati, il valore di **Threshold** viene impostato a **CongWin/2** e **CongWin** viene impostata al valore di **Threshold**.
- Quando si verifica un timeout, il valore di **Threshold** viene impostato a **CongWin/2** e **CongWin** è impostata a 1 MSS.

Controllo della congestione del TCP: eventi/azioni sul lato mittente

Stato	Evento	Azione del mittente TCP	Commenti
Slow Start (SS)	Ricezione di ACK per dati precedente-mente non riscontrati	CongWin = CongWin + MSS, If (CongWin > Threshold) imposta lo stato a "Congestion Avoidance"	CongWin raddoppia a ogni RTT
Congestion Avoidance (CA)	Ricezione di ACK per dati precedente-mente non riscontrati	CongWin = CongWin + MSS * (MSS/CongWin)	Incremento additivo: CongWin aumenta di 1 MSS a ogni RTT
SS o CA	Rilevato un evento di perdita da tre ACK duplicati	Threshold = CongWin/2, CongWin = Threshold, imposta lo stato a "Congestion Avoidance"	Ripristino rapido con il decremento moltiplicativo. CongWin non sarà mai minore di 1 MSS
SS o CA	Timeout	Threshold = CongWin/2, CongWin = 1 MSS, imposta lo stato a "Slow Start"	Entra nello stato "Slow Start"
SS o CA	ACK duplicato	Incrementa il conteggio degli ACK duplicati per il segmento in corso di riscontro	CongWin e Threshold non variano

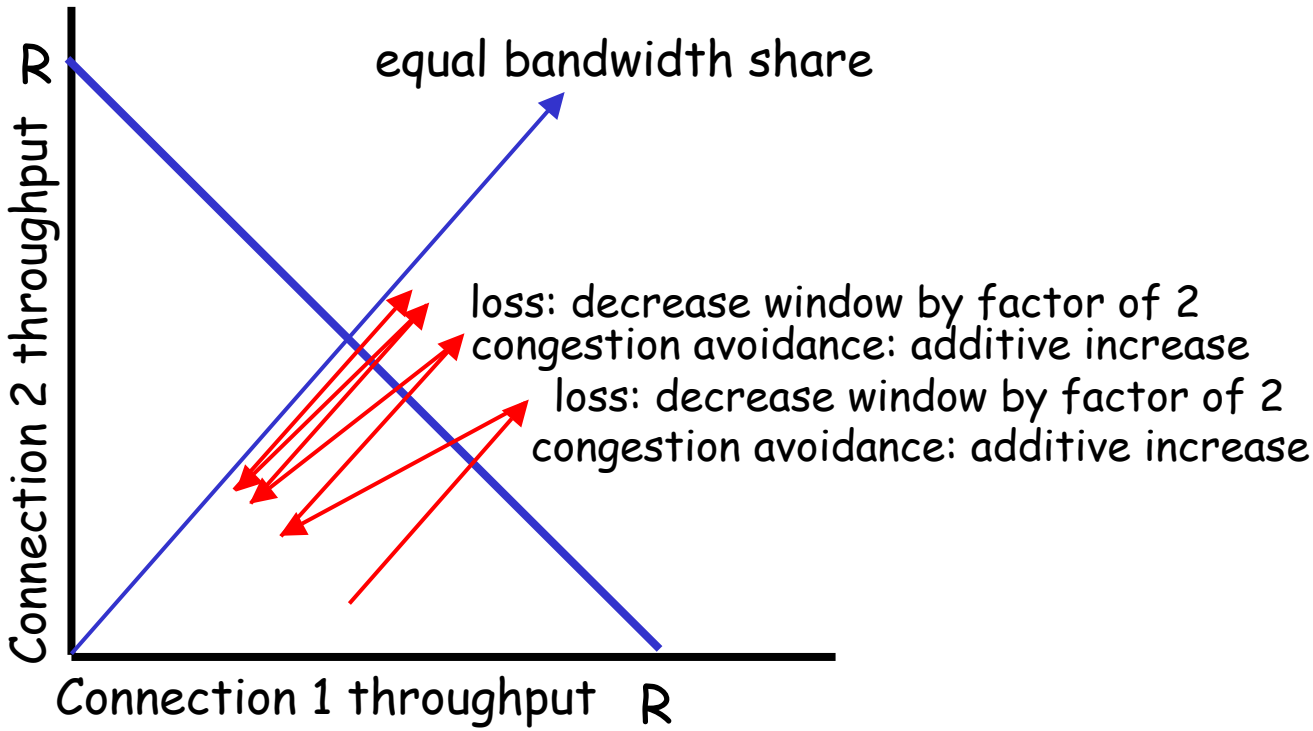
Controllo della congestione del TCP: equità

Definizione: se K sessioni TCP condividono lo stesso collegamento con ampiezza di banda R, che è un collo di bottiglia per il sistema, ogni sessione dovrà avere una frequenza trasmissiva media pari a R/K.



Controllo della congestione del TCP: equità (cont.)

- Due connessioni TCP che condividono link di capacità R



Controllo della congestione del TCP: modellazione della latenza

D: Quanto tempo occorre per ricevere un oggetto da un server WWW dopo avere inviato la richiesta?

Ignorando la congestione, il ritardo è influenzato da:

- Inizializzazione della connessione TCP
- Ritardo nella trasmissione dei dati
- Partenza lenta

Notazioni, ipotesi:

- Unico link tra client e server con velocità R bit/sec
- MSS = S bit
- Oggetto di dimensione O bit
- nessuna ritrasmissione (nessuna perdita né alterazione)

Dimensione della finestra:

- Prima supponiamo che la finestra di congestione sia statica: W segmenti
- Poi supponiamo che la finestra di congestione sia dinamica, per modellare la partenza lenta

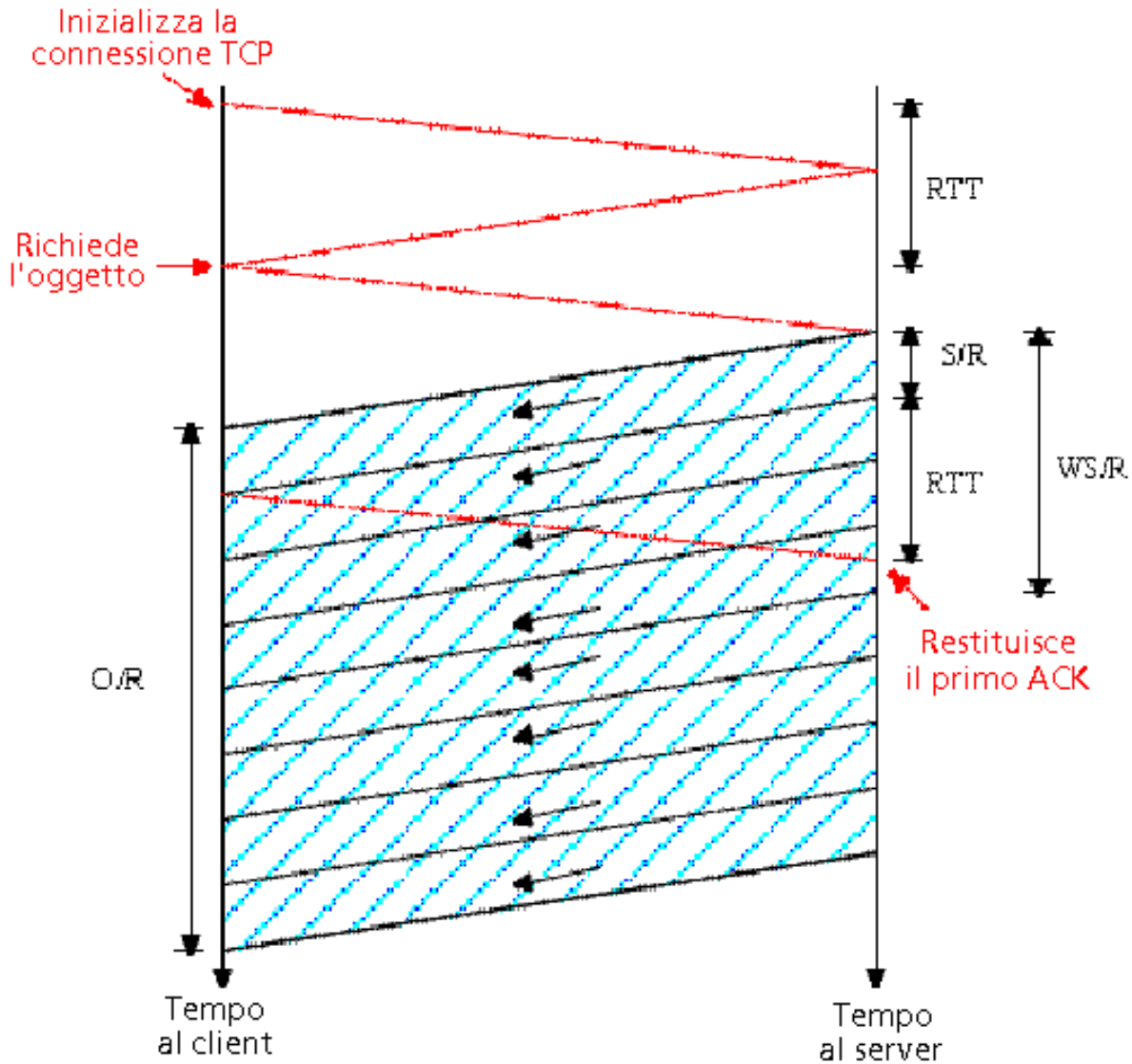
Controllo della congestione del TCP: modellazione della latenza (cont.)

Finestra di congestione statica (I caso):

WS/R > RTT + S/R:

il server riceve un ACK per il primo segmento nella finestra prima di completare la trasmissione della finestra

latenza = 2RTT + O/R



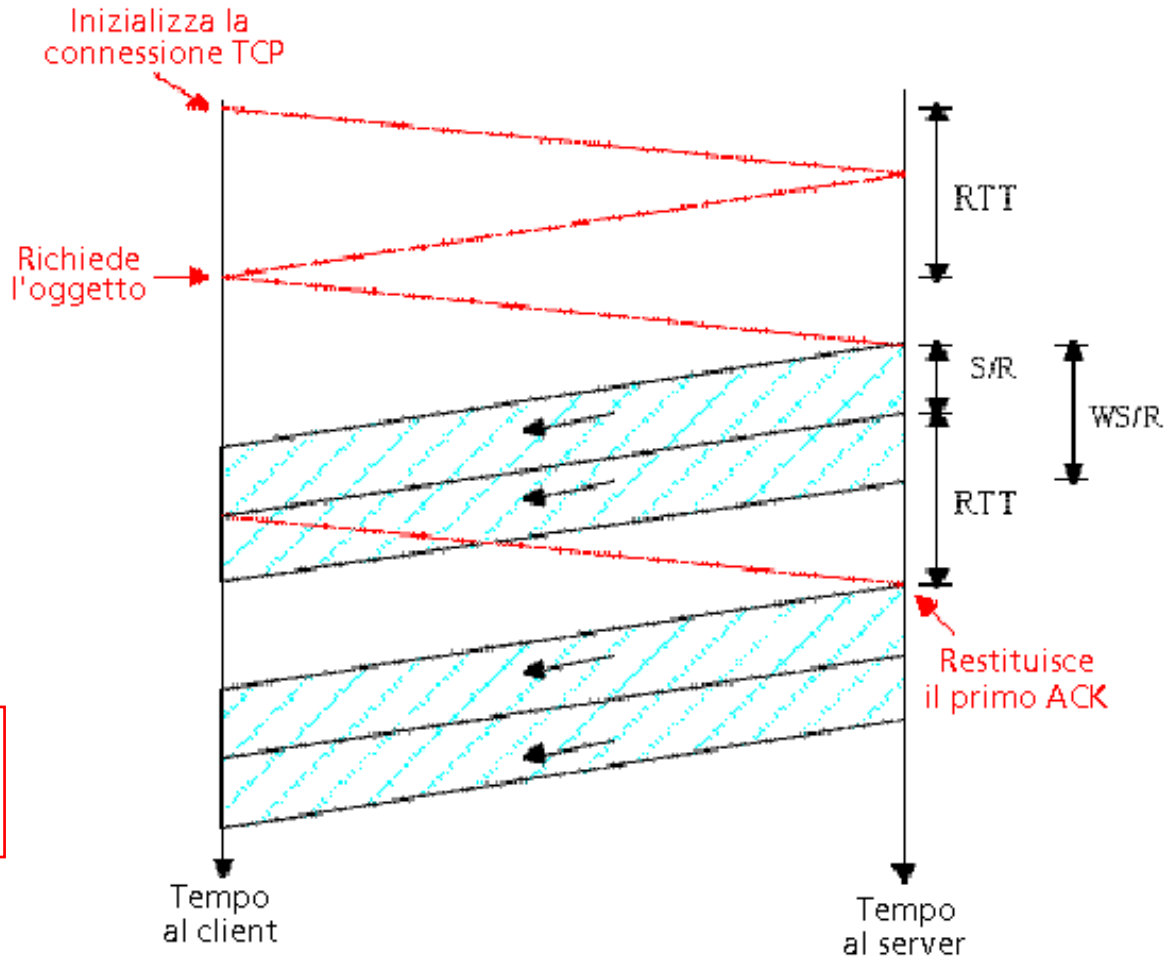
Controllo della congestione del TCP: modellazione della latenza (cont.)

Finestra di congestione statica (II caso):

WS/R < RTT + S/R:

il server trasmette tanti segmenti quanti ne consente la dimensione della finestra prima che il server riceva un riscontro per il primo segmento nella finestra

latenza = 2RTT + O/R + (K-1)[S/R + RTT - WS/R]



Controllo della congestione del TCP: modellazione della latenza (cont.)

Finestra di congestione dinamica

Supponiamo che la finestra cresca secondo la partenza lenta

Dimostreremo che il ritardo per un oggetto è:

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

dove P è il numero di volte in cui il server entra in stallo:

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

- dove Q è il numero di volte in cui il server andrebbe in stallo se l'oggetto contenesse un numero infinito di segmenti.

- e K è il numero di finestre che coprono l'oggetto.

Controllo della congestione del TCP: modellazione della latenza (cont.)

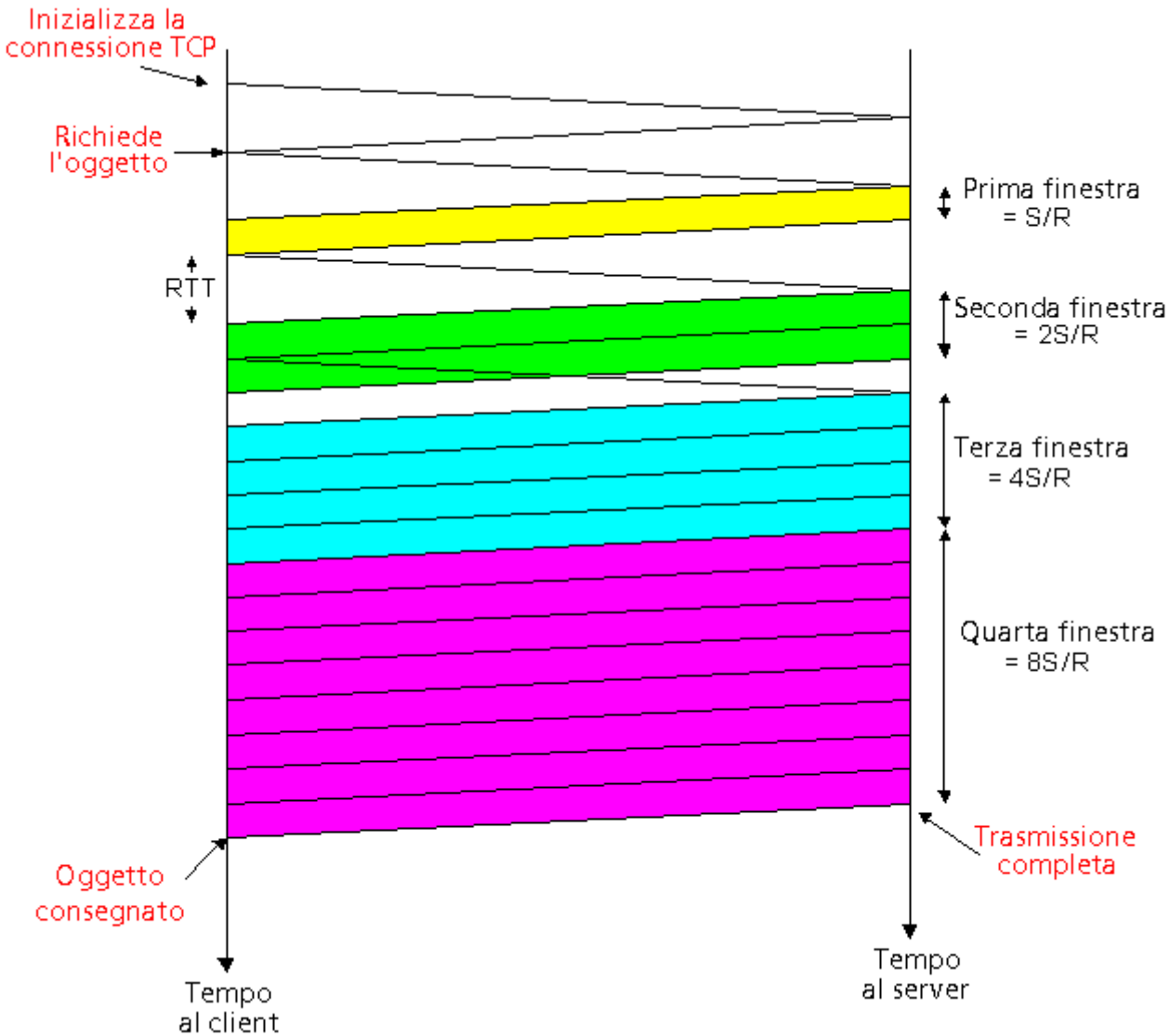
Componenti della latenza:

- 2 RTT per inizializzare la connessione ed inviare la richiesta
- O/R per trasmettere l'oggetto
- periodo di stallo del server a causa della partenza lenta

Stalli del server:
P = min{K-1,Q} volte

Esempio:

- O/S = 15 segmenti
- K = 4 finestre
- Q = 2
- P = min{K-1,Q} = 2

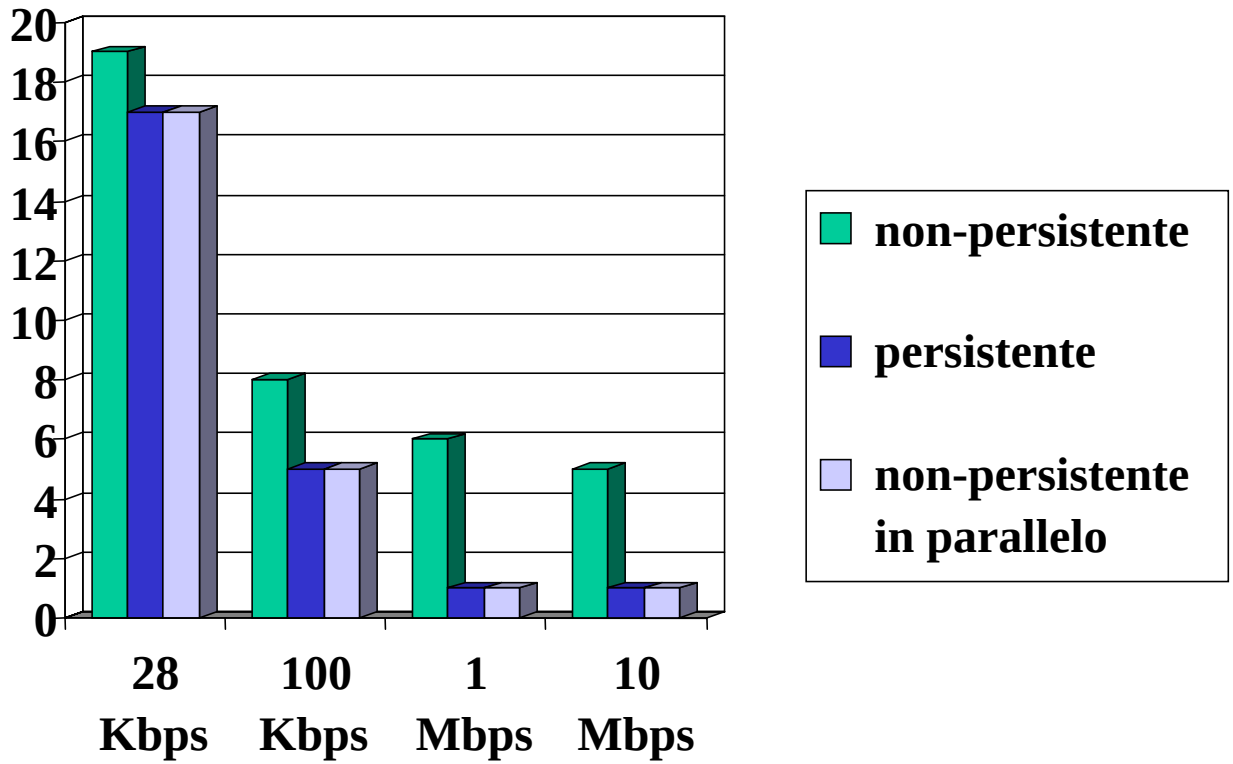


Controllo della congestione del TCP: modellazione della latenza HTTP

- Supponiamo che la pagina web sia formata da:
  - *1* pagina HTML di base (di dimensione *O* bit)
  - *M* immagini (ciascuna di dimensione *O* bit)
- HTTP non persistente:
  - *M+1* connessioni TCP in serie
  - *Tempo di risposta = (M+1)O/R + (M+1)2RTT + somma degli stalli*
- HTTP persistente:
  - *2 RTT* per la richiesta e per ricevere il file HTML di base
  - *1 RTT* per la richiesta e per ricevere M immagini
  - *Tempo di risposta = (M+1)O/R + 3RTT + somma degli stalli*
- HTTP non persistente con X connessioni in parallelo
  - Supponiamo M/X intero
  - Una connessione TCP per il file di base
  - M/X gruppi di connessioni parallele per le immagini
  - *Tempo di risposta = (M+1)O/R + (M/X + 1)2RTT + somma degli stalli*

Tempo di risposta HTTP (in secondi)

RTT = 100 msec, O = 5 Kbyte, M = 10 e X = 5

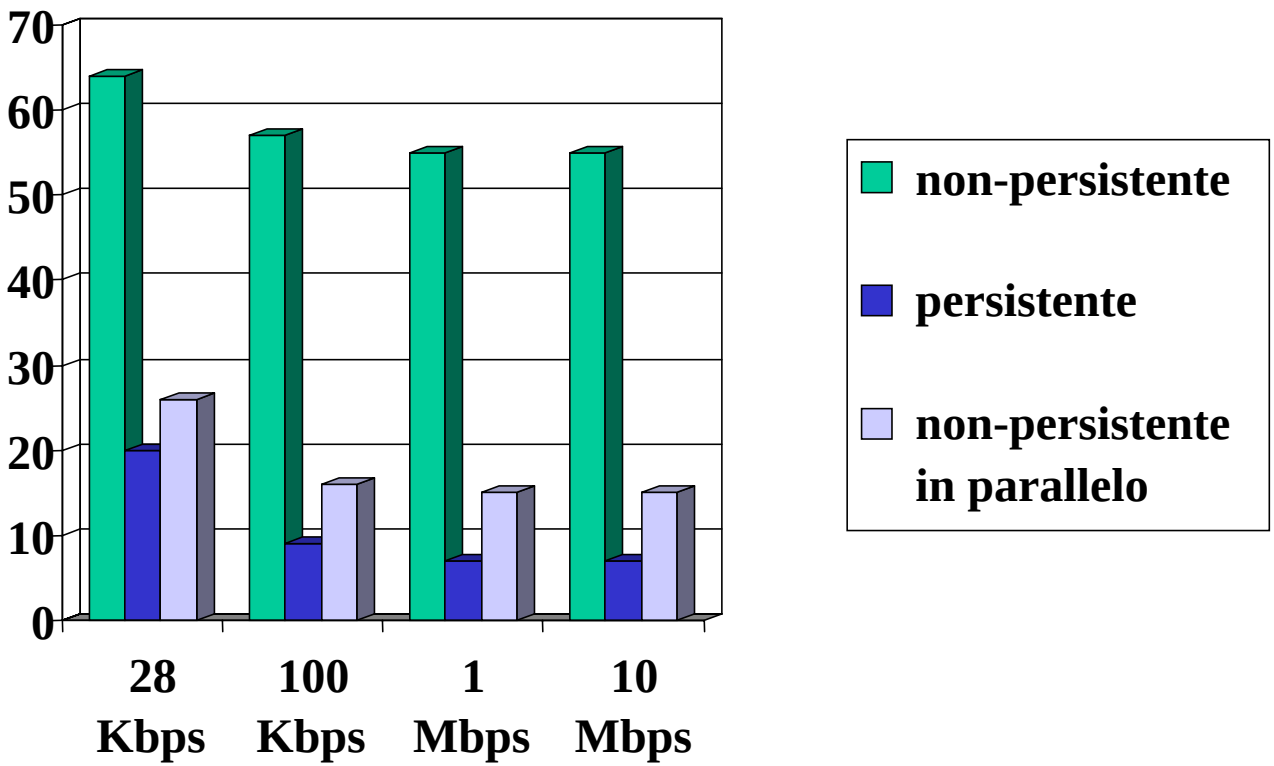


Se l'ampiezza di banda è limitata, la connessione e il tempo di risposta sono dominati dal tempo di trasmissione.

Le connessioni persistenti apportano soltanto un modesto miglioramento sulle connessioni parallele.

Tempo di risposta HTTP (in secondi)

RTT = 1 sec, O = 5 Kbyte, M = 10 e X = 5



Per RTT più grandi, il tempo di risposta è dominato dalla inizializzazione della connessione TCP e dai ritardi per partenze lente. Le connessioni persistenti adesso apportano un miglioramento significativo: in particolare nelle reti con un valore elevato del prodotto ritardo•ampiezzadibanda.

Sommario dello strato di trasporto

- Principi sottostanti i servizi dello strato di trasporto:
  - multiplexing/demultiplexing
  - Trasferimento affidabile dei dati
  - Controllo di flusso
  - Controllo di congestione
- istanziazione ed implementazione in Internet
  - UDP
  - TCP

Prossima volta:

- lasceremo la soglia della rete (strati applicazione/trasporto)
- entreremo nel cuore della rete (strato di rete)