



7 – Il Livello di Trasporto

Parte III

Livello di Trasporto

- Introduzione
- Protocollo UDP
- Trasporto affidabile
 - Protocolli di ritrasmissione
 - Controllo di flusso a finestra mobile
- **Protocollo TCP**
 - Generalità
 - Formato
 - Controllo d'errore
 - Controllo di congestione



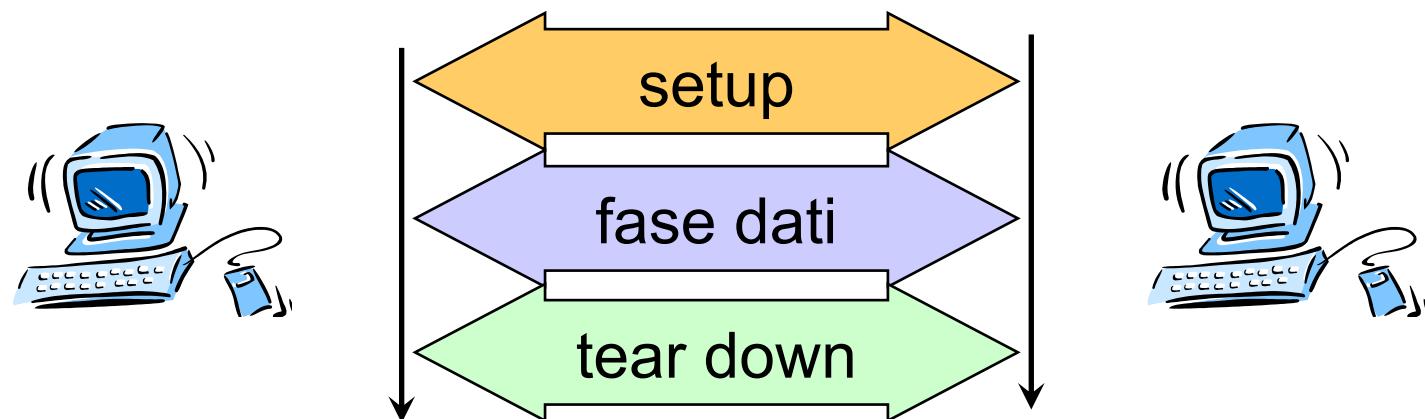
Transmission Control Protocol (TCP)

- Il TCP è un protocollo di trasporto che assicura il trasporto affidabile
 - In corretta sequenza
 - Senza errori/perdite dei dati
- Mediante TCP è possibile costruire applicazioni che si basano sul trasferimento di file senza errori tra host remoti (web, posta elettronica, ecc.)
- E' alla base della filosofia originaria di Internet: servizio di rete semplice e non affidabile, servizio di trasporto affidabile
- Il TCP effettua anche un controllo di congestione *end-to-end* che limita il traffico in rete e consente agli utenti di condividere in modo equo le risorse



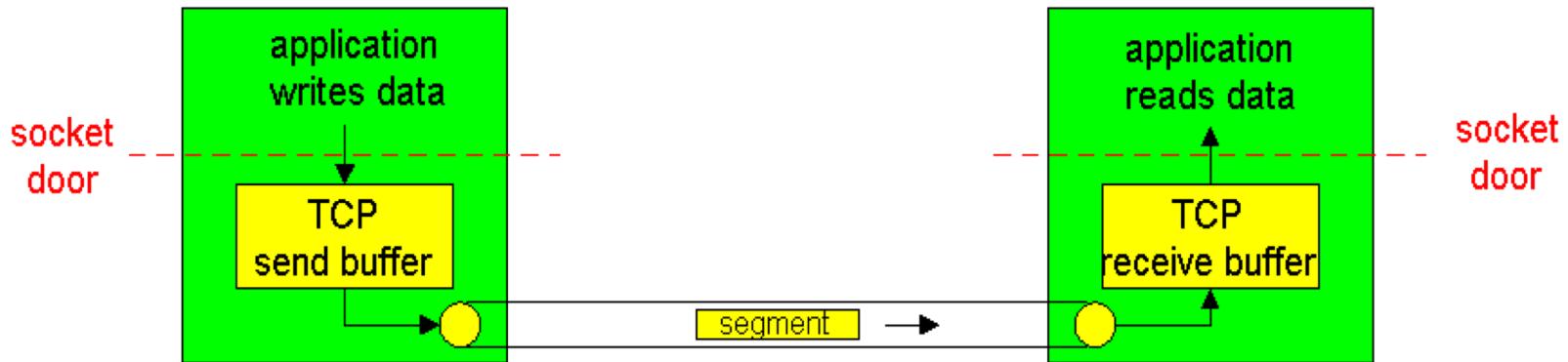
TCP: *connection oriented*

- Il TCP è orientato alla connessione (*connection oriented*):
 - Prima del trasferimento di un flusso dati occorre instaurare una connessione mediante opportuna segnalazione
 - Le connessioni TCP si appoggiano su una rete *connectionless* (datagram)
 - Le connessioni TCP sono di tipo *full-duplex* (esiste sempre un flusso di dati in un verso e nel verso opposto, anche se questi possono essere quantitativamente diversi)



TCP: flusso dati

- Il TCP è orientato alla trasmissione di flussi continui di dati (stream di byte)
- Il TCP converte il flusso di dati in *segmenti* che possono essere trasmessi in IP
- Le dimensioni dei segmenti sono variabili
- L'applicazione trasmittente passa i dati (byte) a TCP e TCP li accumula in un buffer.
- Periodicamente, o quando avvengono particolari condizioni, il TCP prende una parte dei dati nel buffer e forma un segmento
- La dimensione del segmento è critica per le prestazioni, per cui il TCP cerca di attendere fino a che un ammontare ragionevole di dati sia presente nel buffer di trasmissione



TCP: numerazione byte e riscontri

- Il TCP adotta un meccanismo per il controllo delle perdite di pacchetti di tipo **Go-Back-N**
- Sistema di numerazione e di riscontro dei dati inviati
 - TCP numera ogni byte trasmesso, per cui ogni byte ha un numero di sequenza
 - Nell'*header* del segmento TCP è trasportato **il numero di sequenza del primo byte nel segmento stesso**
 - Il ricevitore deve riscontrare i dati ricevuti inviando **il numero di sequenza del prossimo byte che ci si aspetta di ricevere.**
 - Se un riscontro non arriva entro un dato *timeout*, i dati sono ritrasmessi

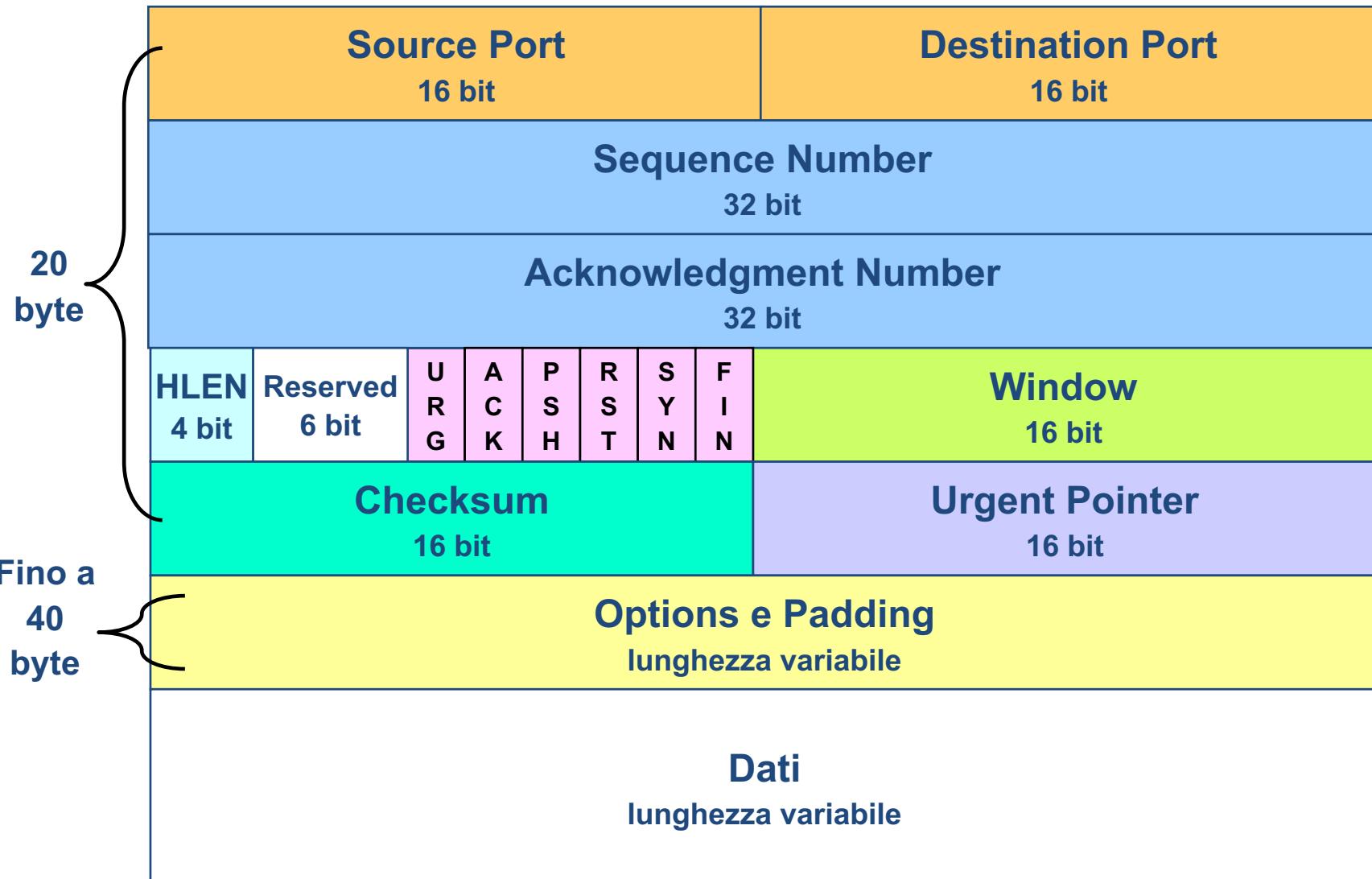


Livello di Trasporto

- Introduzione
- Protocollo UDP
- Trasporto affidabile
 - Protocolli di ritrasmissione
 - Controllo di flusso a finestra mobile
- Protocollo TCP
 - Generalità
 - Formato
 - Controllo d'errore
 - Controllo di congestione



Segmento TCP



Header Segmento TCP I

Source port, Destination port: indirizzi di porta sorgente e porta destinazione di 16 bit

Sequence Number: il numero di sequenza del primo byte nel payload

Acknowledge Number: numero di sequenza del prossimo byte che si intende ricevere (numero valido solo se flag ACK valido)

HLEN (4 bytes words): contiene la lunghezza complessiva dell'header TCP, che DEVE essere un multiplo intero di 32 bit

Reserved 6 bit: non utilizzati

6 “flag” di valore 1 o 0 per funzioni di servizio:

URG è 1 se ci sono dati urgenti

ACK è 1 se il pacchetto è un ACK valido (l'acknowledge num con numero valido)

PSH è 1 quando TX vuole usare il comando di PUSH

RST *reset*, resetta la connessione senza un *tear down*

SYN *synchronize*, usato durante il setup per comunicare numeri di sequenza iniziali

FIN usato per la chiusura esplicita di una connessione



Header Segmento TCP II

Window: contiene il valore della finestra di ricezione come comunicato dal ricevitore al trasmettitore

Checksum: il medesimo di UDP, calcolato in maniera uguale

Options and Padding: da 0 a 40 byte, da riempire (con multipli di 32 bit) per campi opzionale, per es. durante il setup per comunicare il **MAXIMUM SEGMENT SIZE (MSS)** (il valore di default è 536 byte, il valore massimo è 65535 byte).



Servizi e porte

- La divisione tra porte note, assegnate e dinamiche è la stessa che per UDP
- Alcuni delle applicazioni più diffuse:

22	SSH
21	FTP signalling
20	FTP data
23	telnet
25	SMTP
53	DNS
80	HTTP
110	POP
143	IMAP



Livello di Trasporto

- Introduzione
- Protocollo UDP
- Trasporto affidabile
 - Protocolli di ritrasmissione
 - Controllo di flusso a finestra mobile
- Protocollo TCP
 - Generalità
 - Formato
 - Controllo d'errore
 - Controllo di congestione



Controllo d'errore

- Il meccanismo di controllo d'errore del TCP serve a recuperare pacchetti persi in rete
- La causa principale della perdita è l'*overflow* di una delle code dei *router* attraversati a causa della congestione
- Il meccanismo di ritrasmissione è di tipo *Go-back-N* con *Timeout*
- La finestra di trasmissione (valore di N) dipende dal meccanismo di controllo di flusso e di congestione
- L'orologio per la ritrasmissione di un segmento viene inizializzato al momento della trasmissione e determina la ritrasmissione quando raggiunge il valore del *Timeout*



Gestione del Time-Out

- Uno dei problemi è stabilire il valore ottimo del timeout:
 - Timeout troppo breve, il trasmettitore riempirà il canale di ritrasmissioni di segmenti,
 - Timeout troppo lungo impedisce il recupero veloce di reali errori
- Il valore ottimale dipende fortemente dal ritardo in rete (rete locale o collegamento satellitare?)
- Il TCP calcola dinamicamente un valore opportuno per il timeout stimando il RTT (*Round Trip Time*)



Livello di Trasporto

- Introduzione
- Protocollo UDP
- Trasporto affidabile
 - Protocolli di ritrasmissione
 - Controllo di flusso a finestra mobile
- Protocollo TCP
 - Generalità
 - Formato
 - Controllo d'errore
 - Controllo di congestione



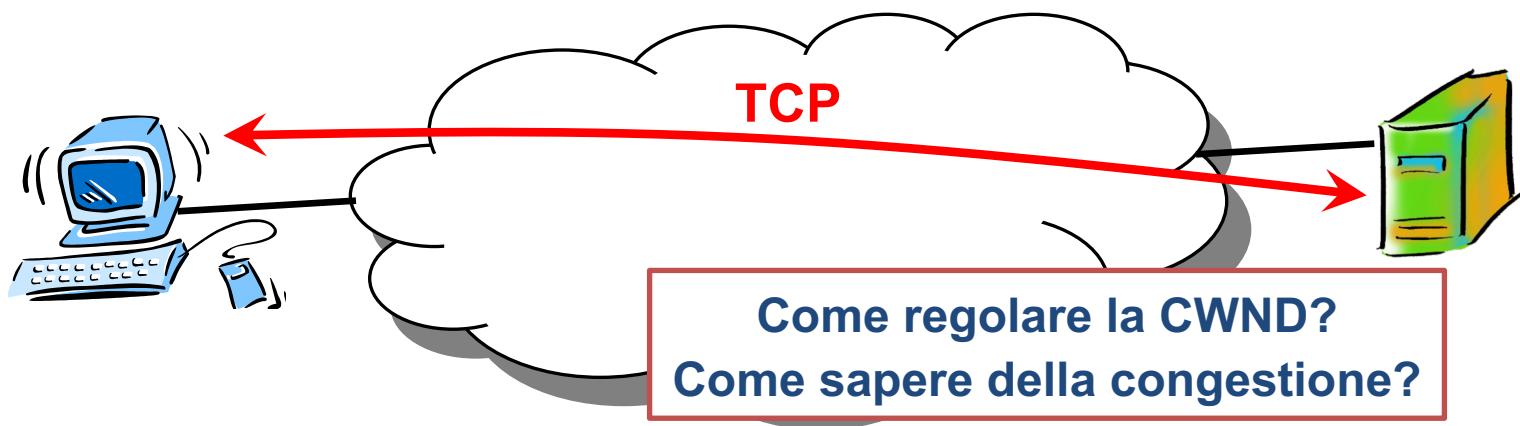
Controllo di congestione

- Il controllo di flusso
 - Dipende **solo** dalla “capacità” del ricevitore
 - Non è sufficiente ad evitare la congestione nella rete
- Nella rete INTERNET attuale non ci sono meccanismi sofisticati di controllo di congestione a livello di rete (come ad esempio meccanismi di controllo del traffico in ingresso)
- Il controllo di congestione è delegato al TCP!!!
 - Se il traffico in rete porta a situazioni di congestione il TCP deve ridurre velocemente il traffico in ingresso
- Essendo il TCP implementato solo negli *host* il controllo di congestione è di tipo *end-to-end*



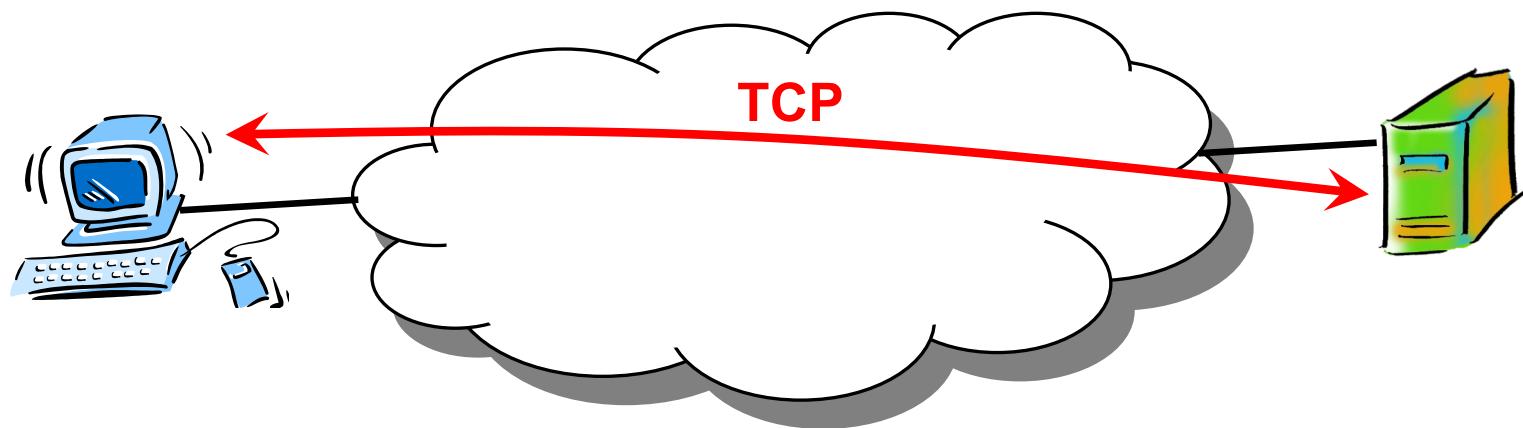
Controllo di congestione

- Il modo più naturale per controllare il ritmo di immissione in rete dei dati per il TCP è quello di regolare la finestra di trasmissione
- Il trasmettitore mantiene una *Congestion Window* (CWND) che varia in base agli eventi che osserva (ricezione ACK, timeout)
- Il trasmettitore non può trasmettere più del minimo tra RCVWND (Receive Window – spazio del buffer in ricezione disponibile per ricevere nuovi dati) e CWND



Controllo di congestione

- L'idea base del controllo di congestione del TCP è quella di interpretare la perdita di un segmento, segnalata dallo scadere di un timeout di ritrasmissione, come un evento di congestione
- La reazione ad un evento di congestione è quella di ridurre la finestra (*CWND*)



Slow Start & Congestion Avoidance

- Il valore della finestra CWND viene aggiornato dal trasmettitore TCP in base ad un algoritmo
- Il modo in cui avviene l'aggiornamento dipende dalla fase (o stato) in cui si trova il trasmettitore
- Esistono due fasi fondamentali:



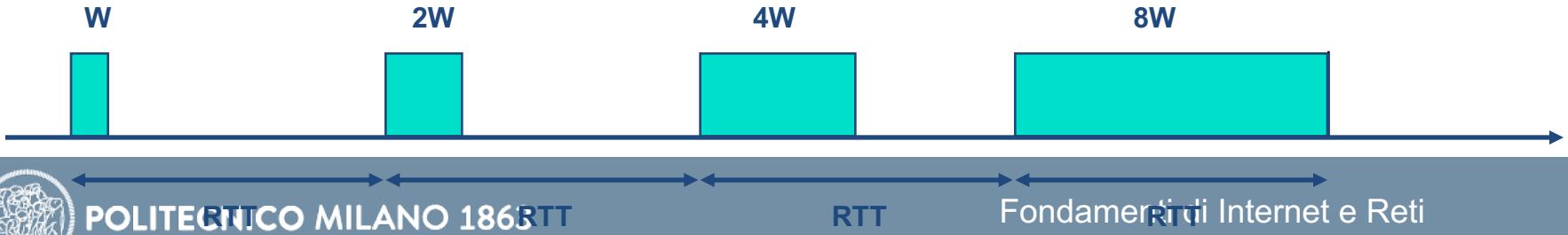
- Slow Start
- Congestion Avoidance

- La variabile STHRESH è mantenuta al trasmettitore per distinguere le due fasi:
 - ➔ se $CWND < STHRESH$ si è in *Slow Start*
 - ➔ se $CWND > STHRESH$ si è in *Congestion Avoidance*



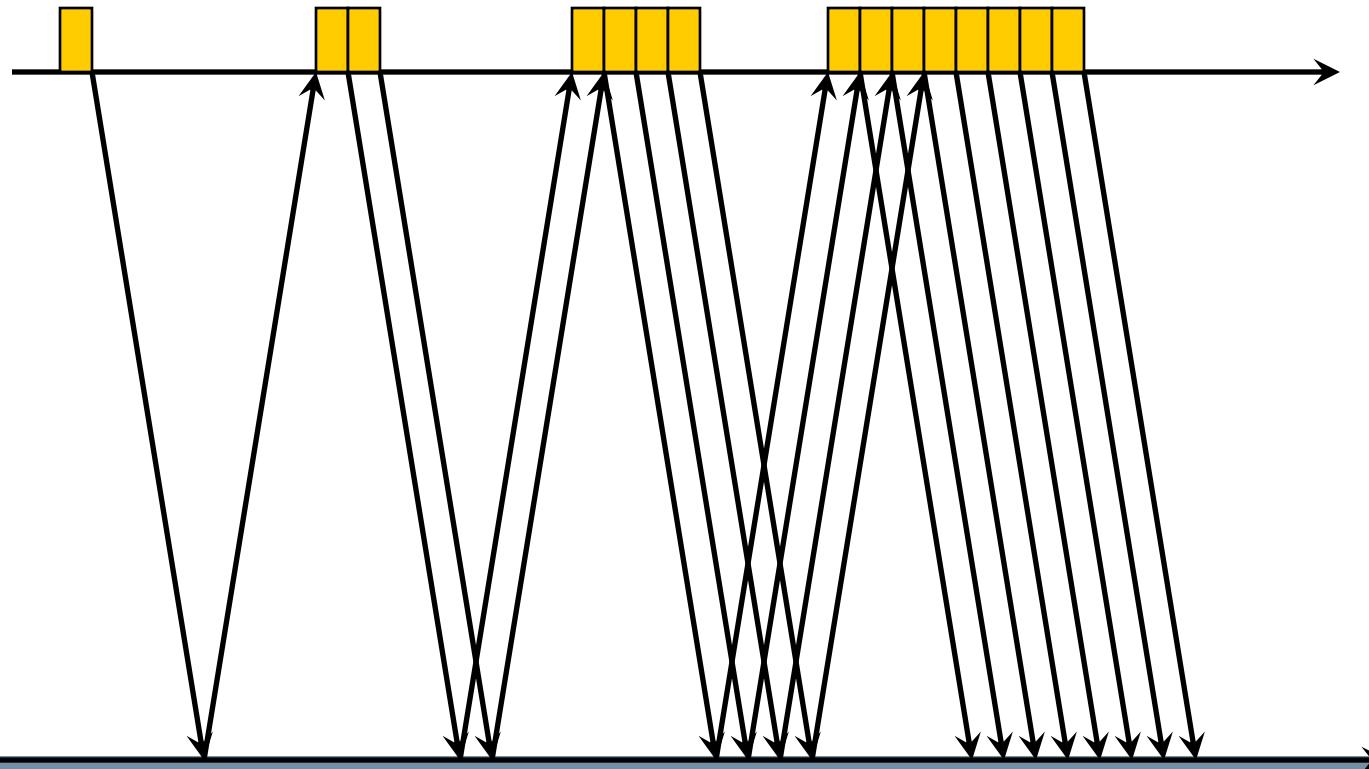
Slow Start

- All'inizio, il trasmettitore pone la CWND a 1 segmento (MSS) e la SSTHRESH ad un valore di *default* molto elevato
- Essendo CWND < SSTHRESH si parte in *Slow Start*
- *In Slow Start:*
 - La CWND viene incrementata di 1 per ogni ACK ricevuto
- Si invia un segmento e dopo RTT si riceve l'ACK, si pone CWND a 2 e si inviano 2 segmenti, si ricevono 2 ACK, si pone CWND a 4 e si inviano 4 segmenti, ...



Slow Start

- Al contrario di quanto il nome faccia credere l'incremento della finestra avviene in modo esponenziale (raddoppia ogni RTT)



Slow Start

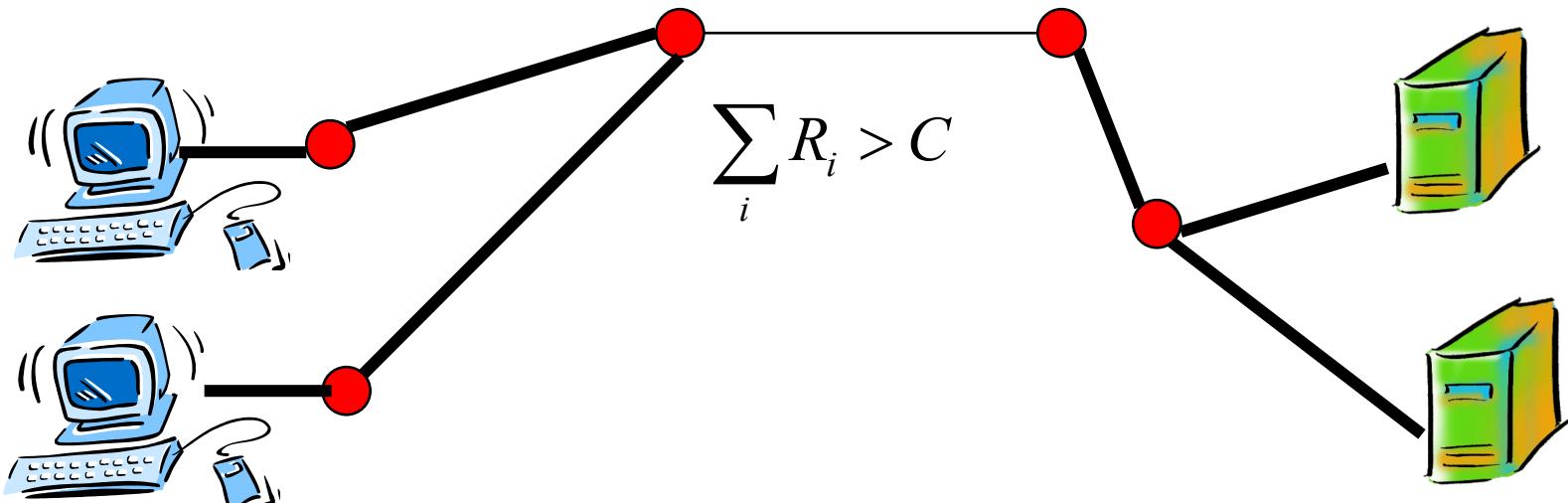
- L'incremento può andare avanti fino
 - Primo evento di congestione
 - Fino a che CWND < SSTHRESH
 - CWND < RCWND
- Insieme alla finestra aumenta il ritmo (o rate) di trasmissione che può essere stimato come:

$$R = \frac{CWND}{RTT} \text{ [bit/s]}$$



Evento di Congestione

- Un evento di congestione si verifica quando il ritmo di trasmissione porta in congestione un link sul percorso in rete verso la destinazione
- Un link è congestionato quando la somma dei ritmi di trasmissione dei flussi che lo attraversano è maggiore della sua capacità



Evento di congestione

- Scade un timeout di ritrasmissione
 - Il TCP reagisce ponendo SSTHRESH uguale alla metà dei “byte in volo” (byte trasmessi ma non riscontrati); più precisamente
 - E ponendo CWND a 1MSS
- Si noti che di solito i “byte in volo” sono con buona approssimazione pari a CWND



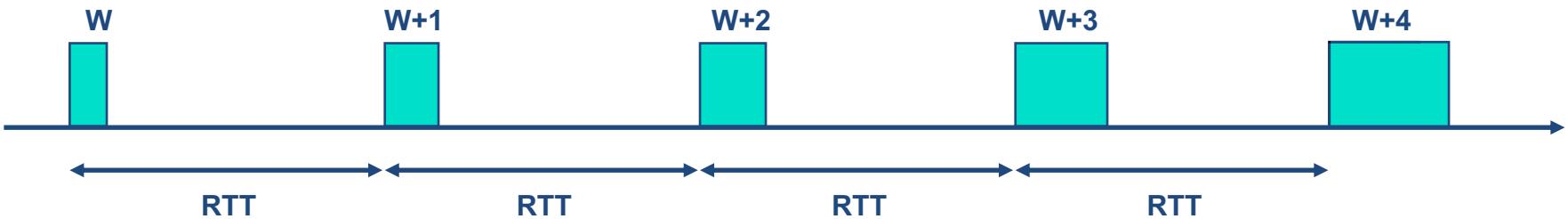
Evento di congestione

- Come risultato:
 - CWND è minore di SSTHRESH e si entra nella fase di Slow Start
 - Il trasmettitore invia un segmento e la sua CWND è incrementata di 1 ad ogni ACK
- Il trasmettitore trasmette tutti i segmenti a partire da quello per cui il timeout è fallito (politica Go-Back-N)
- Il valore a cui è posta la SSTHRESH è una stima della finestra ottimale che eviterebbe futuri eventi di congestione



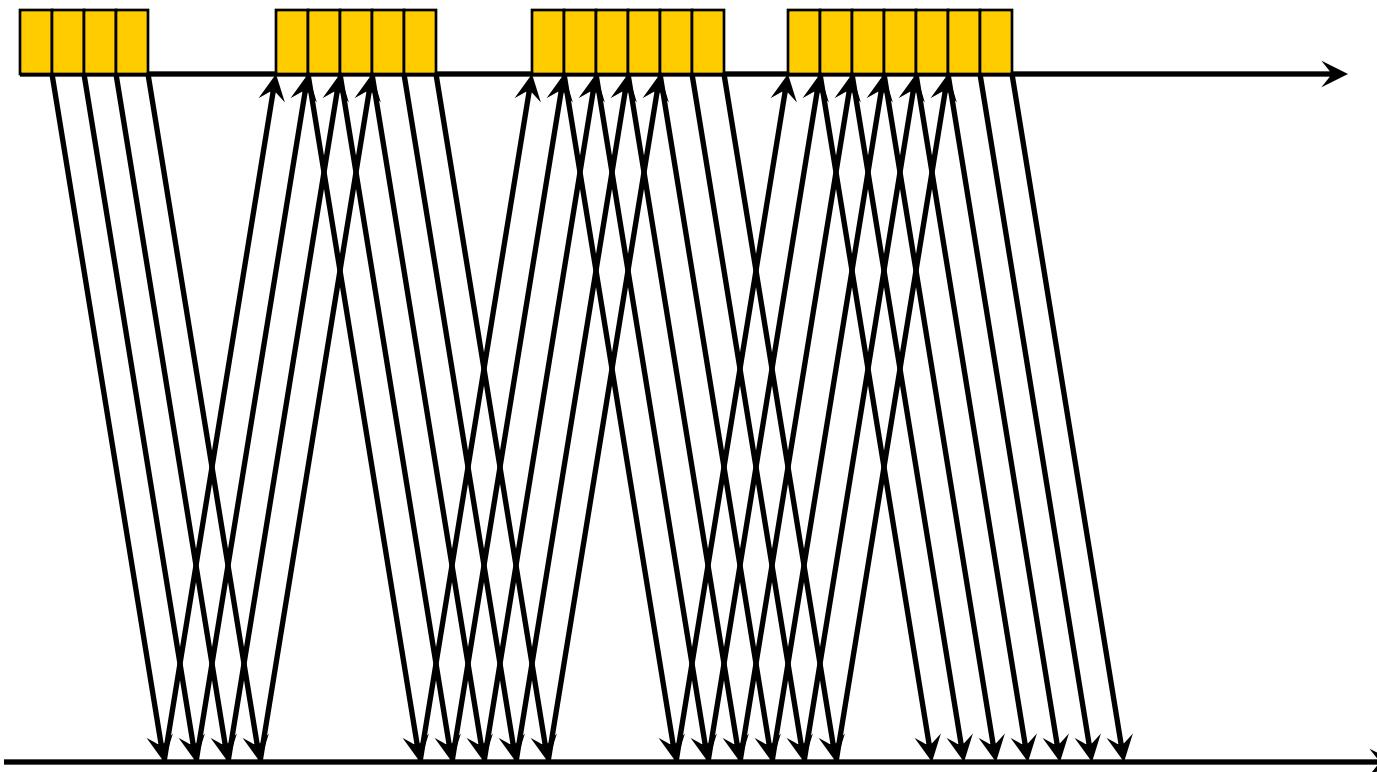
Congestion Avoidance

- Lo slow start continua fino a che CWND diventa grande come SSTHRESH e poi parte la fase di *Congestion Avoidance*
- Durante il *Congestion Avoidance*:
 - Si incrementa la CWND di $1/\text{CWND}$ ad ogni ACK ricevuto
- Se la CWND consente di trasmettere N segmenti, la ricezione degli ACK relativi a tutti gli N segmenti porta la CWND ad aumentare di 1 segmento
- In Congestion Avoidance si attua un incremento lineare della finestra di congestione

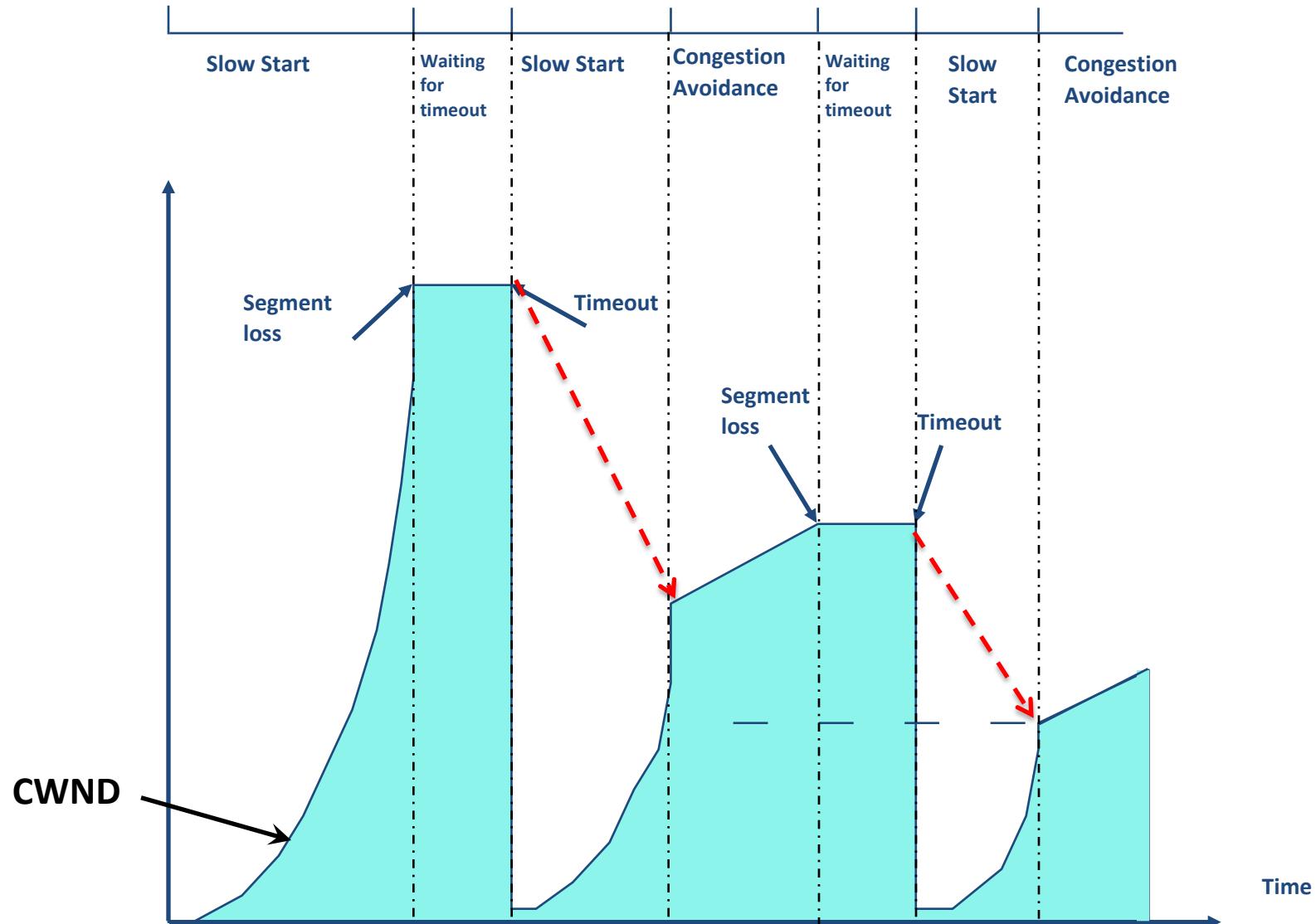


Congestion Avoidance

- Dopo aver raggiunto STHRESH la finestra continua ad aumentare ma molto più lentamente

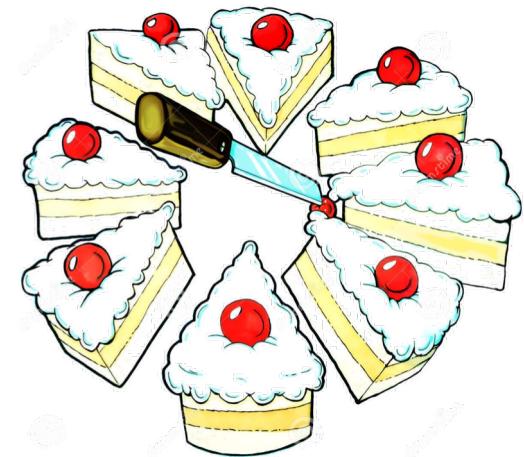


Esempio di funzionamento



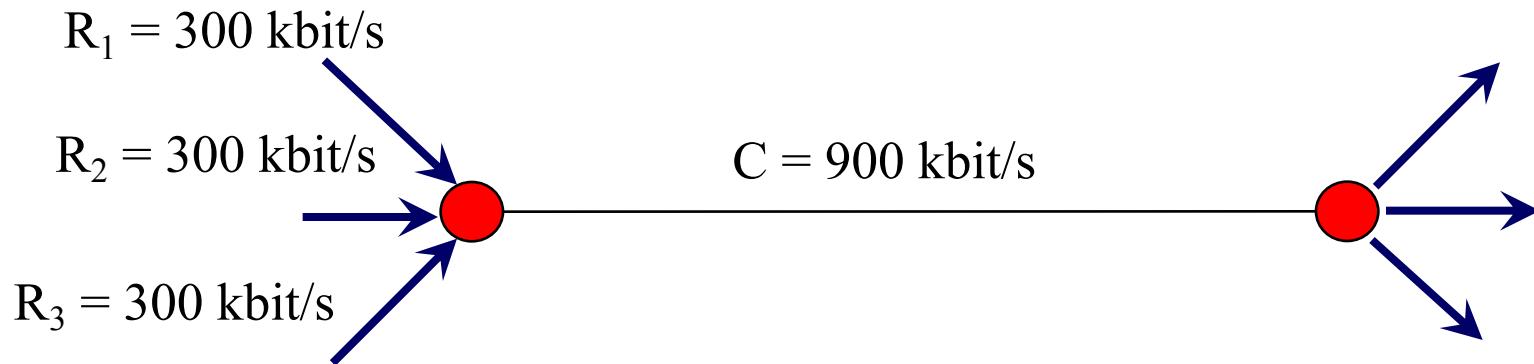
Condivisione equa delle risorse

- In condizioni ideali il meccanismo di controllo del TCP è in grado di
 - Limitare la congestione in rete
 - Consentire di dividere in modo equo la capacità dei link tra i diversi flussi
- Le condizioni ideali sono alterate tra l'altro da
 - Differenti RTT per i diversi flussi
 - Dimensione dei Buffer

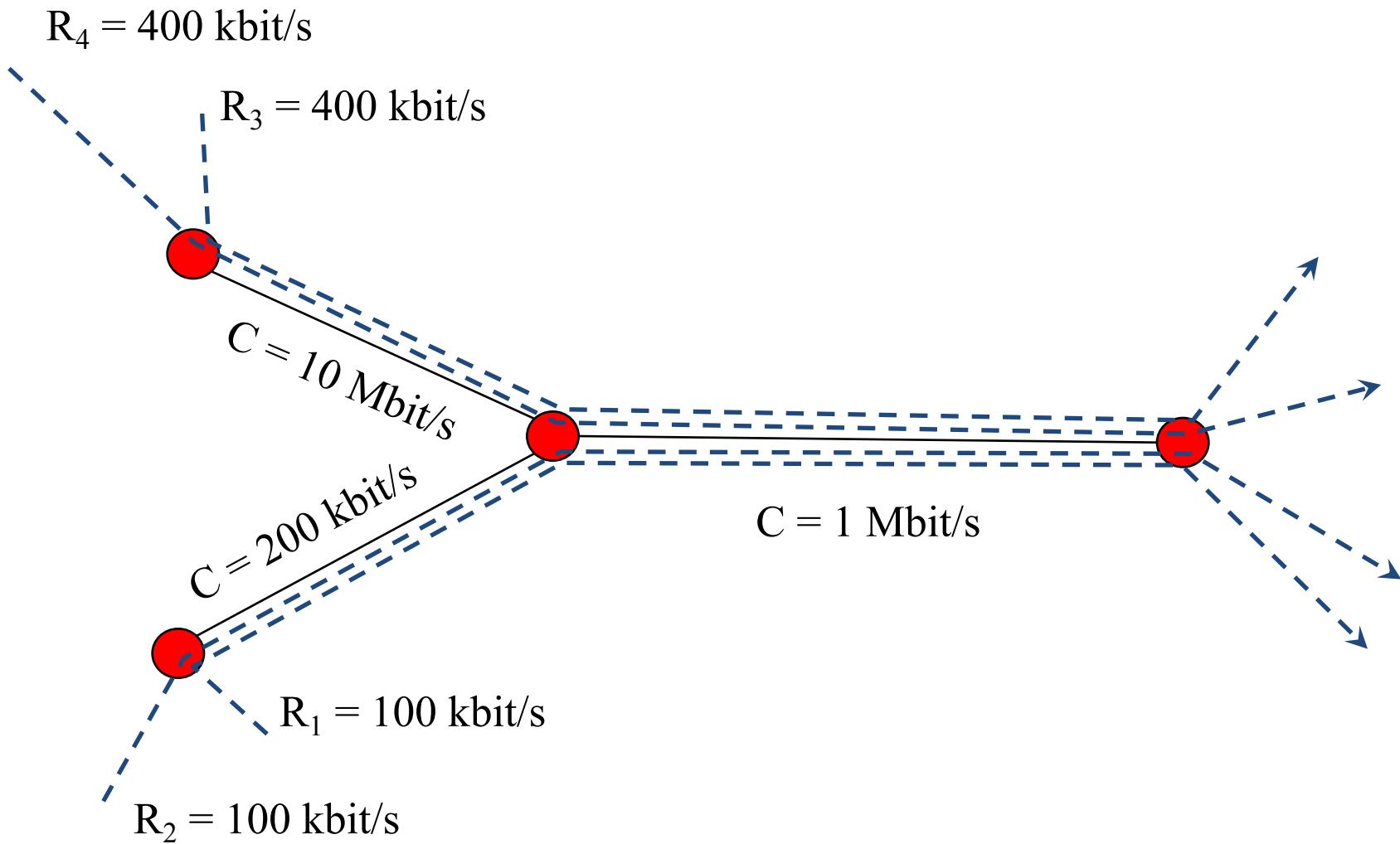


Condivisione equa delle risorse

- I valori dei rate indicati sono solo valori medi e valgono solo in condizioni ideali
- Il ritmo di trasmissione in realtà cambia sempre e in condizioni non ideali la condivisione può non essere equa

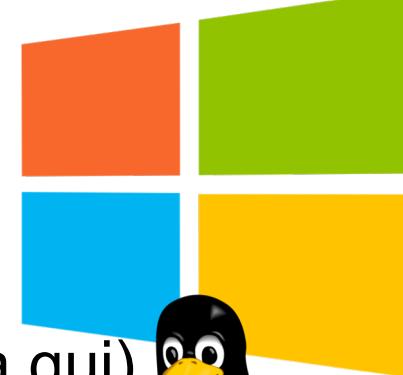


Condivisione equa delle risorse



TCP e sistemi operativi

- Esistono diverse versioni di protocollo TCP
- TCP è implementato nel sistema operativo
- Versione base è TCP **Tahoe** (versione presentata qui)



Famiglia Windows

per esempio **Reno / New Reno**

Famiglia Linux

Altro approccio al Livello 4 di trasporto. Per esempio **BIC** per grandi prodotti banda ritardo o **CUBIC** (non guarda gli ACK)

Famiglia MacO

prima protocolli proprietari, per esempio **MacTCP**. Poi **CUBIC**

