5.1.4 Appello del 25/07/2011

ESERCIZIO 1: La Figura 1.1 illustra un client collegato ad un server mediante una linea punto-punto esente da errori (*BER* = 0), di rate *R* bps ed *RTT* = 12 msec. Supponiamo che il client utilizzi il TCP/IP per effettuare il download

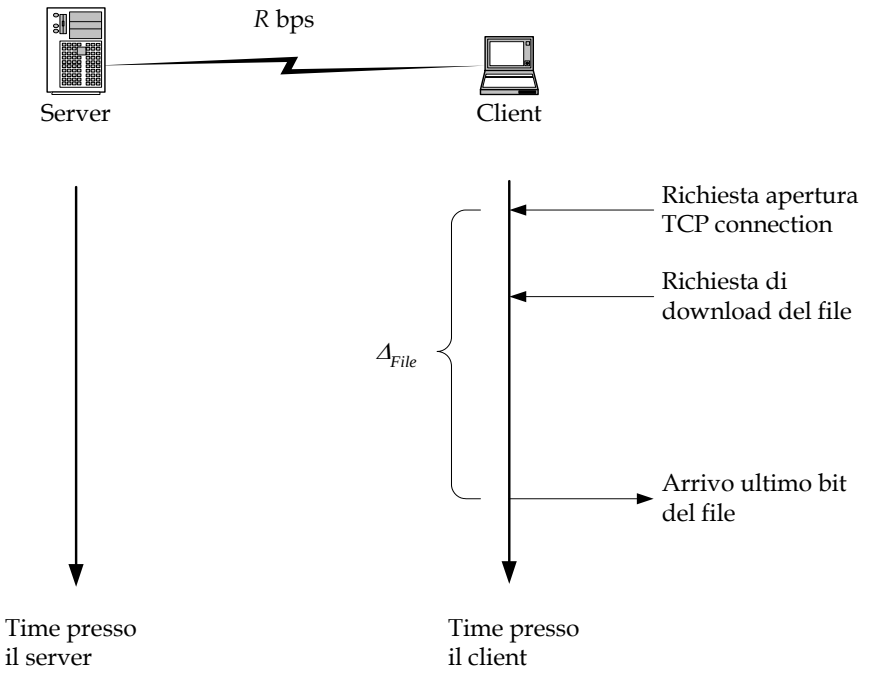

Figura 1.1: Collegamento tra Client e Server

di un file di dimensione pari ad *F* bits (multiplo intero di segmenti di dimensione *MSS*) residente sul server. Dopo aver iniziato la procedura di apertura della connessione di trasporto, il client richiede il download del file tramite un messaggio applicativo che invia al server appena ne ha l'opportunità. Al fine di semplificare i calcoli richiesti supponiamo che tutti i segmenti dati abbiano la stessa dimensione $L = MSS = 1000$ bits e tempo di trasmissione $m = L/R = 2$ msec. I segmenti di *ACK*, *SYN* ed il messaggio applicativo con cui il client richiede il file al server hanno tempi di trasmissione trascurabili. Supponiamo che la finestra di flow control del TCP sia infinita ed indichiamo con Δ_{File} il tempo di trasferimento del file (v. Figura 1.1), definito come intervallo di tempo che intercorre tra l'istante in

cui il client richiede l'apertura della connessione di trasporto e quello in cui l'ultimo bit del file viene ricevuto dal client. Supponiamo che il TCP aggiorni il valore della *Congestion Window* (*CW*) quando riceve l'*ACK* cumulativo di tutti i segmenti compresi nella *CW* medesima. Il candidato, relativamente ai meccanismi di *Slow Start* e *AIMD*:

1. calcoli analiticamente il numero *K* di *Congestion Window* necessario per trasmettere il file ed esprima *F* in funzione di *K*;
2. calcoli analiticamente Δ_{File} ed il throughput γ;
3. valuti numericamente le precedenti espressioni nel caso in cui *F* consti di 7 segmenti.

Da ora in poi analizziamo soltanto lo *Slow Start*. Supponiamo però che gli *ACK* vengano trasmessi a livello di segmento (ovvero trasmissione di un *ACK* per ogni segmento ricevuto) e che il valore della *CW* venga incrementato di 1 *MSS* ogni qual volta il server riceve un *ACK*. Il candidato:

4. illustri graficamente l'evoluzione del sistema fino a quando il numero di segmenti contenuti nella *CW* consente al server la trasmissione continua dei segmenti;
5. calcoli, nel caso generale, il valore *k* relativo alla *k*.ma *CW* in corrispondenza della quale il server può trasmettere continuamente segmenti al client (senza dover cioè attendere un *ACK* da parte del client);

NOTA. Per semplificare i calcoli di cui al punto 2 precedente, il candidato assuma che la dimensione di *F* sia tale da riempire l'ultima *CW*, ovvero *CW*(*K*).

RISOLUZIONE

1. Distinguiamo i due meccanismi per il controllo della congestione. Prima di fare ciò osserviamo che il client richiede il trasferimento del file al server mediante un mesaggio applicativo inoltrato sul *terzo ramo* del three-way-handshake. Questa è la prima opportunità che il client ha per segnalare informazioni al server.

- **Slow Start**

Secondo il meccanismo dello *Slow Start* (v. Figura 1.2) la *k*.ma *CW* o *CW*(*k*) comprende un numero di segmenti pari a $CW(k) = 2^{k-1}$, $\forall k \geq 1$. Di

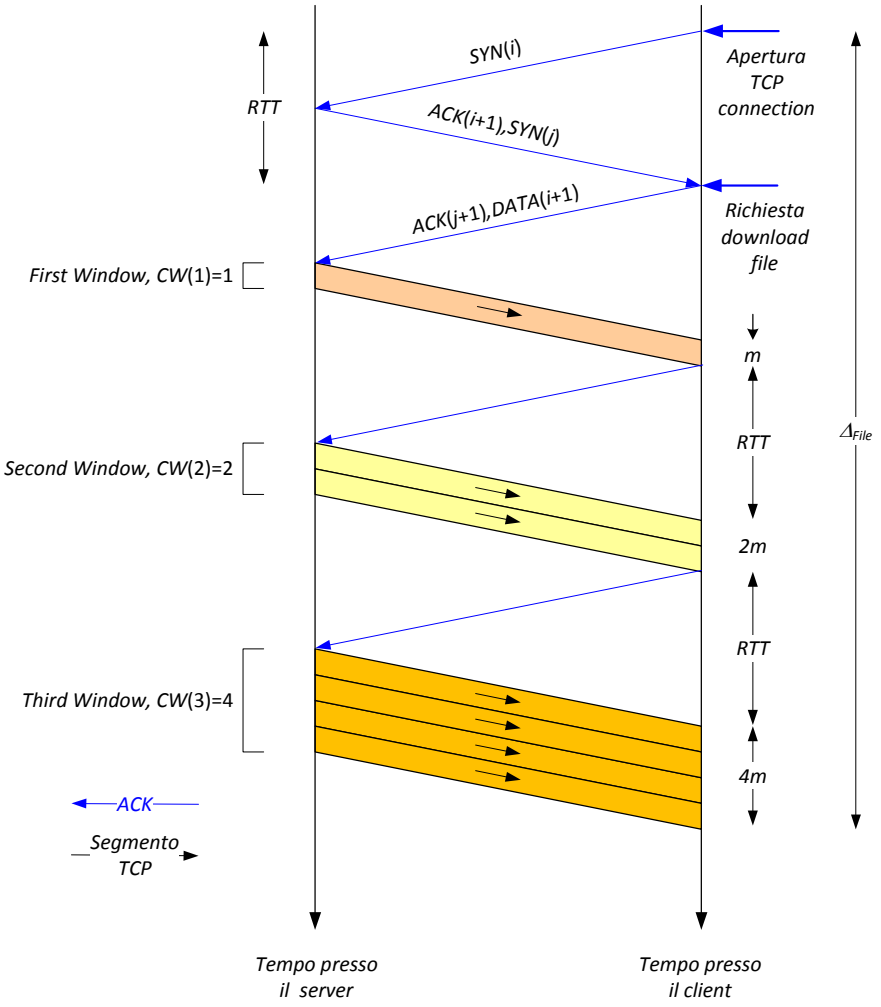

Figura 1.2: Evoluzione temporale secondo lo *Slow Start*

conseguenza, per calcolare K basta imporre la condizione:

$$\sum_{h=1}^{K} 2^{h-1} \geq F/L \tag{1.1}$$

dove F/L rappresenta il numero di segmenti di cui è composto il file. Quindi

$$\sum_{h=1}^{K} 2^{h-1} = \sum_{j=0}^{K-1} 2^{j} = \frac{1-2^{K}}{1-2} = 2^{K}-1 \tag{1.2}$$

Sostituendo (1.2) in (1.1) si ottiene

$$2^K - 1 \geq F/L \tag{1.3}$$

ovvero

$$2^K \geq 1 + F/L \tag{1.4}$$

Calcolando il logaritmo in base 2 di entrambi i membri

$$K = \lceil log_2(1 + F/L) \rceil \tag{1.5}$$

La (1.4) può essere riscritta nel modo seguente:

$$F \leq (2^K - 1)L \tag{1.6}$$

che esprime, appunto, la dimensione del file in funzione della massima dimensione (*K*) della *Congestion Window.*

- **AIMD**

Secondo il meccanismo AIMD (v. Figura 1.3) la *k*.ma *CW* o *CW*(*k*) comprende $CW(k) = k, \forall k \geq 1$ segmenti. Di conseguenza, per calcolare *K* basta imporre la condizione:

$$\sum_{h=1}^{K} h \geq F/L \tag{1.7}$$

dove *F/L* rappresenta il numero di segmenti di cui è composto il file. Sostituendo

$$\sum_{h=1}^{K} h = \frac{K(K+1)}{2} \tag{1.8}$$

nella (1.7) si ottiene

$$\frac{K(K+1)}{2} \geq F/L \tag{1.9}$$

ovvero

$$K^2 + K - \frac{2F}{L} \geq 0 \tag{1.10}$$

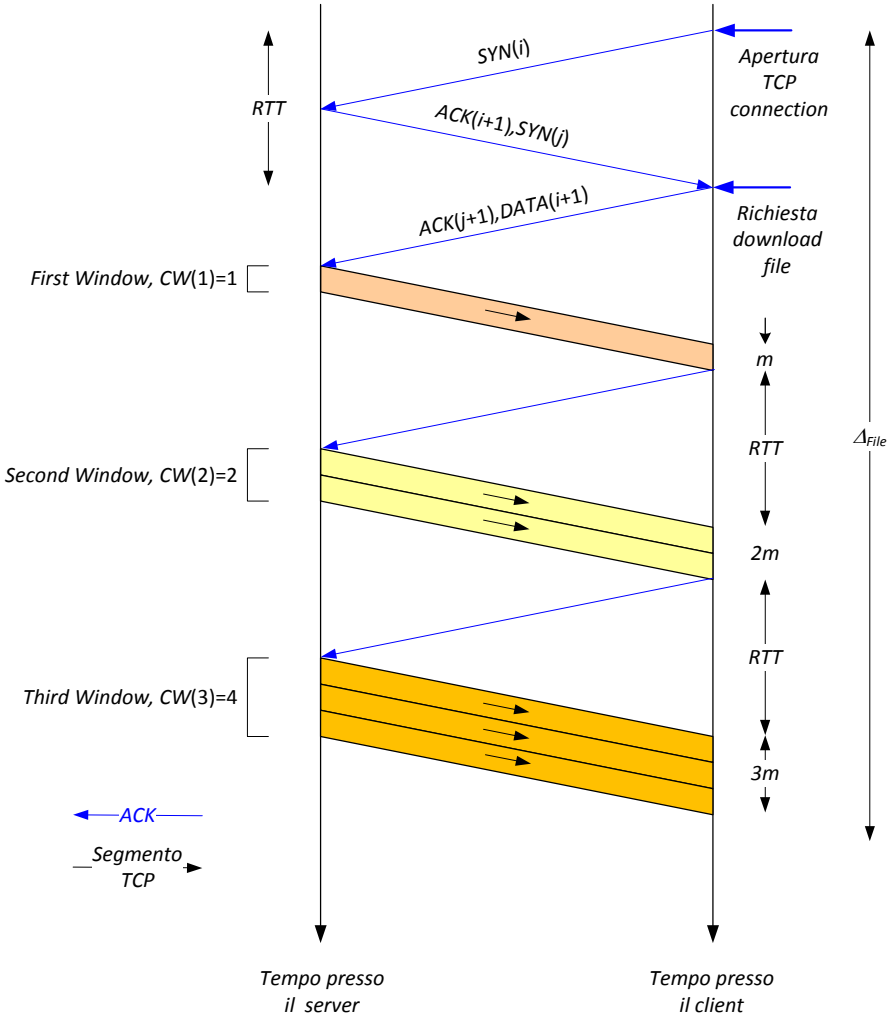

Figura 1.3: Evoluzione temporale secondo AIMD

La soluzione ingegneristicamente accettabile è:

$$K = \left\lceil \frac{-1 + \sqrt{1 + 8(F/L)}}{2} \right\rceil \tag{1.11}$$

La (1.9) può essere riscritta nel modo seguente:

$$F \leq \frac{LK(K+1)}{2} \tag{1.12}$$

che esprime, appunto, la dimensione del file in funzione della massima dimensione della *Congestion Window.*

2. Distinguiamo i due casi

- **Slow Start**

Come si evince dalla Figura 1.2, Δ_{File} è costituito dalla somma dei due *RTT* del three-way-handshake con i tempi dovuti alla trasmissione dei segmenti appartenenti alle *CW* che seguono. Come sappiamo, la *k*.ma *CW* consta di $CW(k) = 2^{k-1}$, $\forall k \geq 1$ segmenti, per trasmettere ciscuno dei quali sono necessari $m = L/R$ msec. Ricevuto l'ultimo bit dell'ultimo segmento in una *CW*, c'è bisogno di attendere *RTT* msec prima di ricevere il primo bit del primo segmento della *CW* successiva. Quest'ultimo contributo viene a mancare per $CW(K)$. Di conseguenza:

$$\Delta_{File} = 2RTT + \sum_{j=1}^{K-1} [2^{j-1}m + RTT] + 2^{K-1}m$$

$$= 2RTT + \sum_{j=1}^{K-1} 2^{j-1}m + \sum_{j=1}^{K-1} RTT + 2^{K-1}m$$

$$= 2RTT + \frac{1-2^{K-1}}{1-2}m + (K-1)RTT + 2^{K-1}m$$

Dopo alcuni passaggi algebrici perveniamo al risultato

$$\Delta_{File} = (K+1)RTT + (2^K - 1)m \tag{1.13}$$

Dalla (1.6) e dalla (1.13) si ottiene:

$$\gamma = \frac{(2^K - 1)L}{(K+1)RTT + (2^K - 1)m} \tag{1.14}$$

Quando la dimensione del file cresce all'infinito anche *K* tende all'infinito per cui è semplice verificare che la (1.14) tende ad *R*.

- **AIMD**

Come si evince dalla Figura 1.3, Δ_{File} è costituito dalla somma dei due *RTT* del three-way-handshake con i tempi dovuti alla trasmissione dei seg-

menti appartenenti alle *CW* che seguono. Come sappiamo, la *k*.ma *CW* consta di $CW(k) = k,\ \forall k \geq 1$ segmenti, per trasmettere ciascuno dei quali sono necessari $m = L/R$ msec. Ricevuto l'ultimo bit dell'ultimo segmento in una *CW,* c'è bisogno di attendere *RTT* msec prima di ricevere il primo bit del primo segmento della *CW* successiva. Quest'ultimo contributo viene a mancare per $CW(K)$. Di conseguenza:

$$\Delta_{File} = 2RTT + \sum_{j=1}^{K-1} [jm + RTT] + Km$$

$$= 2RTT + \sum_{j=1}^{K-1} jm + \sum_{j=1}^{K-1} RTT + Km$$

$$= 2RTT + \frac{K(K-1)}{2}m + (K-1)RTT + Km$$

Dopo alcuni passaggi algebrici perveniamo al risultato

$$\Delta_{File} = (K+1)\left(RTT + \frac{K}{2}m\right) \tag{1.15}$$

Dalla (1.12) (1.12) e dalla (1.15) si ottiene:

$$\gamma = \frac{\frac{LK(K+1)}{2}}{(K+1)\left(RTT + \frac{K}{2}m\right)} \tag{1.16}$$

Dopo alcune manipolazione algebriche si perviene al seguente risultato:

$$\gamma = \frac{L}{(2RTT)/K + m} \tag{1.17}$$

Anche in questo caso, quando la dimensione del file cresce all'infinito *K* tende all'infinito per cui è semplice verificare che la (1.17) tende ad *R*.

3. Nel caso in esame otteniamo i seguenti valori numerici:

- **Slow Start**

$$K = log_2(1 + F/L) = log_2(1 + 7) = 3 \tag{1.18}$$

$$\gamma = \frac{(2^K - 1)L}{(K+1)RTT + (2^K - 1)m} \tag{1.19}$$

$$= \frac{(2^3 - 1)10^3}{(3+1)12 + (2^3 - 1)2} \approx 112Kbps \tag{1.20}$$

- **AIMD**

$$K = \left\lceil \frac{-1 + \sqrt{1 + 8(F/L)}}{2} \right\rceil = \left\lceil \frac{-1 + \sqrt{1 + 8 \times 7}}{2} \right\rceil = 4 \tag{1.21}$$

$$\gamma = \frac{L}{(2RTT)/K + m} \tag{1.22}$$

$$= \frac{10^3}{(2 \times 12)/4 + 2} = 125Kbps \tag{1.23}$$

Dai precedenti calcoli sembra che il throughput dello *Slow Start* sia minore di quello dell'*AIMD*. Questa apparente contraddizione deriva dalle assunzioni fatte per il calcolo del throughput, ovvero che la parte terminale del file riempia completamente l'ultima *CW*. Di conseguenza, disponendo di un file costituito da 7 segmenti, con lo *Slow Start* si riesce a trasmetterlo con tre *CW* mentre con *AIMD* c'è bisogno di 4 *CW* e quindi, in base alle assunzioni fatte, il throughput calcolato si riferisce ad un file lungo 10 segmenti.

4. Come illustrato in Figura 1.4 l'evoluzione del sistema cambia profondamente rispetto al caso in cui si utilizza *l'ACK* cumulativo. Infatti, a partire dalla quarta finestra di congestione, cioè $CW(4) = 8$, la linea di collegamento tra client e server risulta permanentemente occupata. Ovviamente in Figura 1.4 le lunghezze dei segmenti che rappresentatno *m* ed *RTT* sono state scelte in modo da rispettare il rapporto $RTT/m = 12/2 = 6$. Da notare che, relativamente alla quarta finestra di congestione, l'*ACK* del primo segmento viene ricevuto dal server quando questo inizia la trasmissione dell'ottavo segmento che, ovviamente, è compreso nella finestra di congestione $CW(4) = 8$.

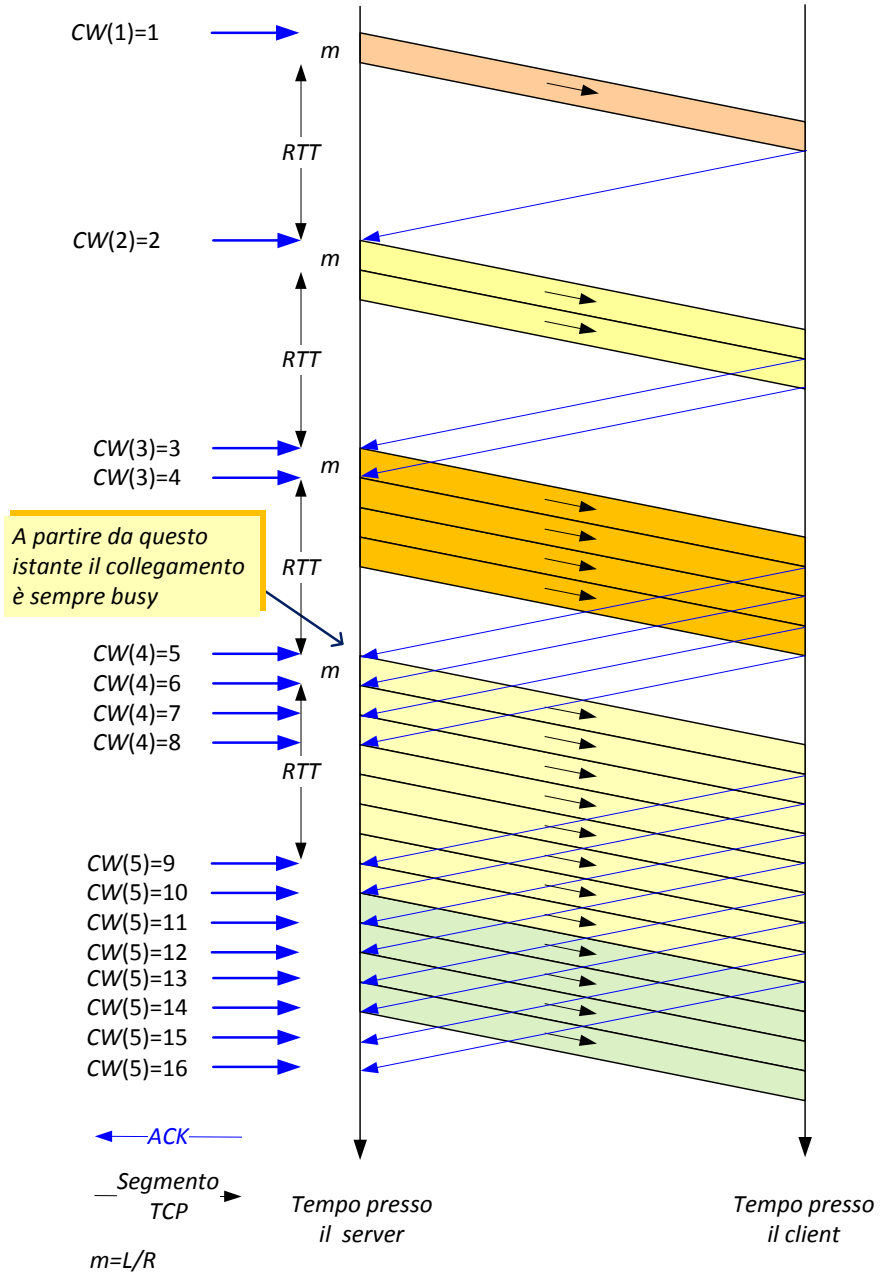

Figura 1.4: Evoluzione del sistema

5. Osservando la Figura 1.4 si può dedurre facilmente che nel caso generale la linea che collega il client ed il server risulterà permanentemente busy solo

quando la finestra di congestione contiene un numero di segmenti maggiore di $(RTT + m)/m$, ossia maggiore del numero di segmenti che possono essere trasmessi nell'ntervallo temporale $RTT + m$. In altri termini, quando arriva l'*ACK* del primo segmento di una finestra di congestione, il client deve continuare a trasmettere segmenti senza interruzioni. Questo è possibile se la *k*.ma *Congestion Window* soddisfa il seguente vincolo:

$$\frac{RTT + m}{m} \leq CW(k) \tag{1.24}$$

ovvero

$$\frac{RTT + m}{m} \leq 2^{k-1} \tag{1.25}$$

Prendendo il logaritmo in base 2 di entrambi i membri della (1.25) si ottiene la seguente espressione generale

$$k \geq \left\lceil log_2\left(1 + \frac{RTT}{m}\right) \right\rceil + 1 \tag{1.26}$$

Utilizzando i dati del problema

$$k \geq \lceil log_2(7) \rceil + 1 = \lceil 2,81 \rceil + 1 \; = \; 4 \tag{1.27}$$

ricaviamo per *k* lo stesso valore (non poteva che essere cosi'!) che abbiamo ottenuto analizzando il diagramma di Figura 1.4.