Soluzione dell'esercizio 2 (TCP) dell'esame del 16 giugno 2015

La soluzione di questo esercizio per quanto riguarda la parte di perdita di pacchetti è data assumendo l'algoritmo di FAST RECOVERY, ossia seguendo la macchina a stati in Figura 3.52 a pagina 275 della versione inglese del libro di testo (Kurose-Ross, 6th edition, si veda l'ultima pagina della soluzione). Questa non è l'unica soluzione corretta. Come visto a lezione esistono diversi algoritmi utilizzati da TCP per gestire la perdita di pacchetti. Nell'esame specificate qual è l'algoritmo che state utilizzando, a meno che non venga esplicitamente detto quale utilizzare.

1. La consegna avviene in maniera indiretta poiché sorgente e destinazione appartengono a sottoreti diverse.
2. 1460 Byte, derivanti dalla dimensione massima del payload Ethernet e tolti 20 Byte di header IP e 20 Byte di header TCP.
3. Stiamo considerando un trasferimento dati tramite HTTP e la porta di ascolto standard per questo servizio applicativo è la 80. Il client quindi dovra aprire una connessione TCP impostando come destination port la porta 80. Come source port dovrà scegliere una porta a caso con un valore superiore a 1023 (porte standard riservate). Il client HTTP apre la connessione con conseguente scambio dei pacchetti TCP di handshake, ossia SYN (client verso server), SYN+ACK (server verso client), ACK (client verso server) (si veda il punto 5.).
4. TCP invierà una serie di pacchetti di dimensione massima (MSS) più l'ultimo pacchetto per concludere l'invio di tutti i dati. L'applicazione deve inviare 16600 Byte (16300 di file più i 300 per il comando PUT/POST di HTTP). Dato che MSS = 1460 Byte:
   - 16600 / 1460 = 11 pacchetti di dimensione MSS più
   - un pacchetto di dimensione 16600 - 1460 * 11 = 540 Byte
5. La Fig. 1 mostra lo scambio di pacchetti inclusi quelli per l'apertura e la chiusura della connessione. In rosso sono mostrati i round trip time, in rosso scuro metà round trip time (tempo di propagazione dovuto all'invio dell'ultimo ACK per la chiusura) e in verde i tempi di trasferimento dei pacchetti dati. Tali durate sono pari alla dimensione del segmento a cui fanno riferimento diviso per la velocità di trasmissione (verde chiaro: 1500 B / 4800000 b/s = 2.5 ms, verde scuro: 580 B / 4800000 b/s = 1 ms (circa)). Come scritto nel testo, consideriamo trascurabile la dimensione di pacchetti ACK e simili (FIN dal server) e dunque non li contiamo (anche se tenerne conto è semplice). In totale la trasmissione dura circa 37.5 ms:
   - 1 RTT per l'apertura della connessione (2ms)
   - 1 RTT per l'attesa dell'ACK del primo segmento dati (2ms)
   - 11 frame dati da 1500B più 1 da 580B (28.5ms)
   - 1 RTT per l'attesa dell'ACK dell'ultimo segmento dati (2ms)
   - 1.5 RTT per la chiusura della connessione (3ms)
6. La Fig. 2 mostra invece lo scambio di pacchetti nel caso in cui i segmenti 4 e 9 vengano persi. Per quanto riguarda il segmento 4, al momento dell'invio la finestra di congestione è impostata a 4 segmenti, dunque il client invia in sequenza i segmenti 4 (perso), 5, 6, e 7. I segmenti 5, 6, e 7 causano l'invio di 3 ACK duplicati recanti l'ultimo segmento ricevuto correttamente, ossia il 3. Alla ricezione del 3o ACK duplicato, scatta il meccanismo di FAST RECOVERY che causa la ritrasmissione immediata del segmento 4. SSHTHR diventa metà CWND, e CWND diventa SSHTHR più 3. Questo consente l'invio del segmento 8 immediatamente dopo la ritrasmissione del segmento 4. Quando il server riceve il segmento 4 invia al client l'ACK A7, informandolo quindi della corretta ricezione dei segmenti fino a quel punto. TCP quindi entra in congestion avoidance, partendo da CWND = SSHTHR = 2. Il client invia i segmenti 9 e 10: il 9 viene perso, mentre il 10 causa l'invio dell'ACK duplicato A8. Non essendoci pero' altri ACK

duplicati il client si accorge della perdita allo scadere del timeout. Il client quindi imposta SSHTHR a 1 e CWND a 1 e riparte con slow start. Alla ricezione dell'ACK A10 il client aumenta CWND a 2 e invia gli ultimi due segmenti. In questo caso, la trasmissione ha una durata di 166.5 ms (circa)

- 1 RTT per l'apertura della connessione (2ms)
- 1 RTT per l'attesa dell'ACK del primo segmento dati (2ms)
- 13 frame dati da 1500B (2 ritrasmissioni) più 1 da 580B (33.5ms)
- 1 RTT dovuto alla perdita del segmento 4 (attesa del 3o ACK duplicato) (2ms)
- 1 RTO dovuto alla perdita del segmento 9 (120ms)
- 1 RTT dovuto all'attesa dell'ACK dopo la ritrasmissione del segmento 9 (2ms)
- 1 RTT per l'attesa dell'ACK dell'ultimo segmento dati (2ms)
- 1.5 RTT per la chiusura della connessione (3ms)

7. Per trasferire la stessa quantità di dati, nel primo caso sono serviti 37.5ms, mentre nel secondo caso 166.5ms. La velocità di trasferimento media per trasferire 16300B (contiamo i 300 B di HTTP PUT come overhead) è stata dunque di circa 3.5 Mbps nel primo caso e di 0.78 Mbps nel secondo. L'efficienza è data dal rapporto fra la velocità misurata a livello applicativo e la velocità di trasferimento a livello fisico. Nel primo caso è data da 3.5 Mbps / 4.8 Mbps ~= 73%, mentre nel secondo da 0.78 Mbps / 4.8 Mbps ~= 16%.

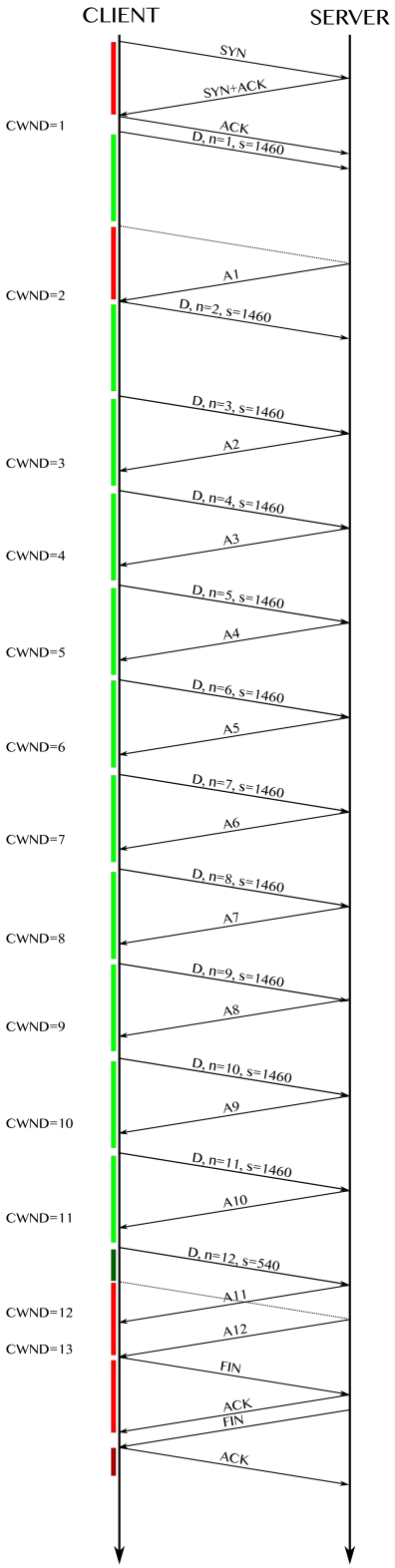

Fig 1: Scambio di pacchetti senza perdite.

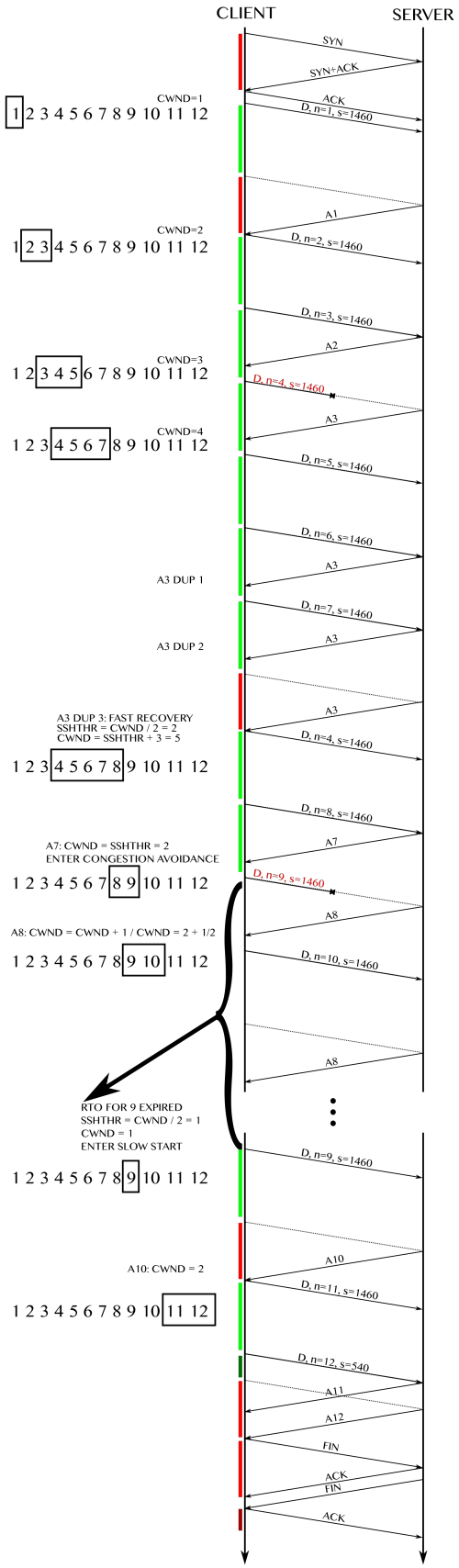

Fig 2: Scambio di pacchetti nel caso di perdita dei segmenti 4 e 9.

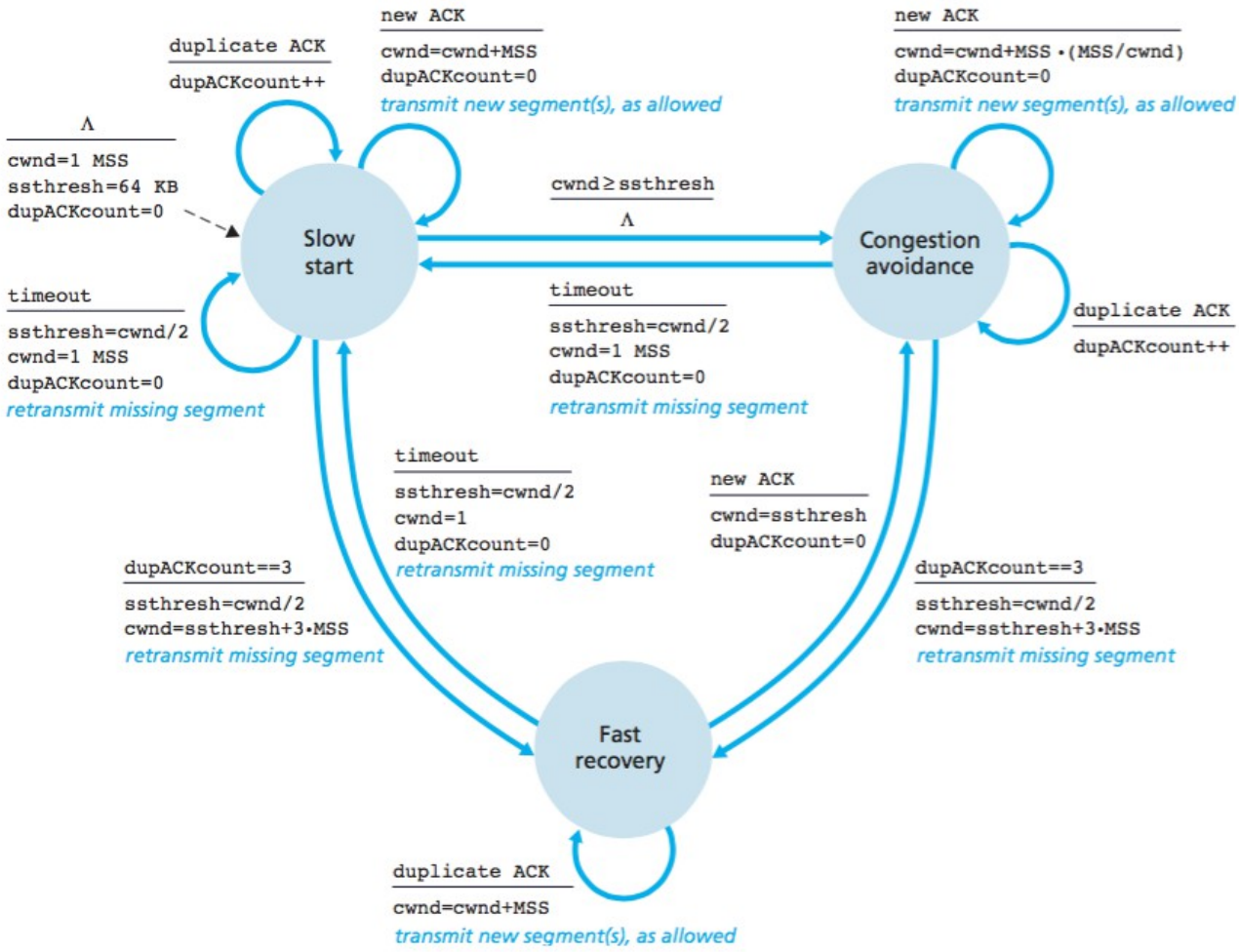

Figure 3.52 ♦ FSM description of TCP congestion control