Esercizio 3

Si consideri una connessione TCP aperta tra 2 host A e B. L'host A invia segmenti dati all'host B, il quale, invece di riscontrare ogni singolo segmento al suo arrivo, esegue il seguente algoritmo (*delayed ack*) all'arrivo di un segmento:

- se B non ha nessun segmento nel buffer, inserisce il segmento nel buffer ed attiva un *ACK timeout* di 150 ms
- se ha già almeno un segmento nel buffer, inserisce il segmento nel buffer senza modificare l' *ACK timeout*
- allo scadere dell'*ACK timeout*, sulla base dei segmenti contenuti nel buffer invia il *minimo* numero *necessario* di ACK e toglie dal buffer i segmenti riscontrati.
- Lo scatto dell'*ACK timeout* viene gestito *prima* di controllare se vi sono segmenti in arrivo. Quindi, se un segmento arriva nello stesso istante in cui scatta l'*ACK timeout*, l'ACK inviato non lo riscontra.

Si supponga che il tempo di propagazione su ciascun verso della connessione sia di 50 ms, e che i tempi di trasmissione dei segmenti e degli ACK siano trascurabili. Si supponga inoltre che la finestra di flow control di B sia di dimensione infinita. Si supponga che A, nei limiti consentiti dal controllo di congestione, invii un segmento ogni 50 ms.

Il candidato

1) Supponendo che non vi siano errori e che il *retransmission timeout* su A non scatti mai, disegni un diagramma temporale dei segmenti $j = 1,...,10$ inviati dall'host A, evidenziando:
    a. La dimensione della Congestion Window su A.
    b. La lista dei segmenti in attesa di ACK su A
    c. La lista dei segmenti ricevuti da B e non ancora riscontrati
2) Calcoli il RTT misurato per i segmenti 1, 2, 4, 7, 8, e proponga un valore per il *retransmission timeout* su A sulla base di tale osservazione.

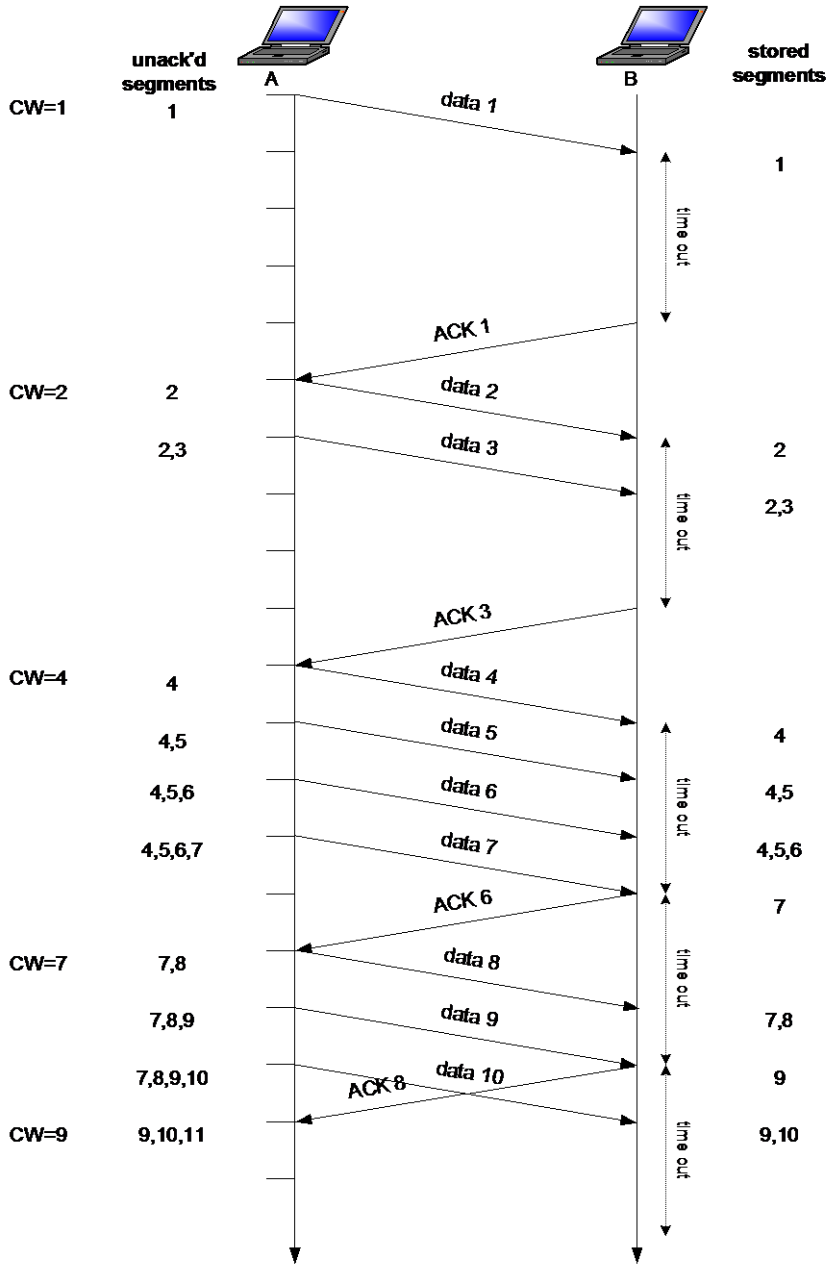
Supponiamo che il segmento n.4 vada perso, e che il *retransmission timeout* di A non scatti. Il candidato:

3) Descriva quale deve essere il comportamento dell'host B allo scattare del successivo *ACK timeout* affinché il *fast retransmit* possa essere attivato.
4) Sotto tale ipotesi, calcoli a quale istante l'ACK che riscontra il segmento n.4 viene correttamente ricevuto da A.
5) Proponga una modifica all'algoritmo del *delayed ack*, che, senza modificarne la filosofia di fondo, consenta al meccanismo di *fast retransmit* di essere attivato più velocemente.

***Nota**: per risolvere i punti 1 e 3 si consiglia di fare uso di carta quadrettata.*

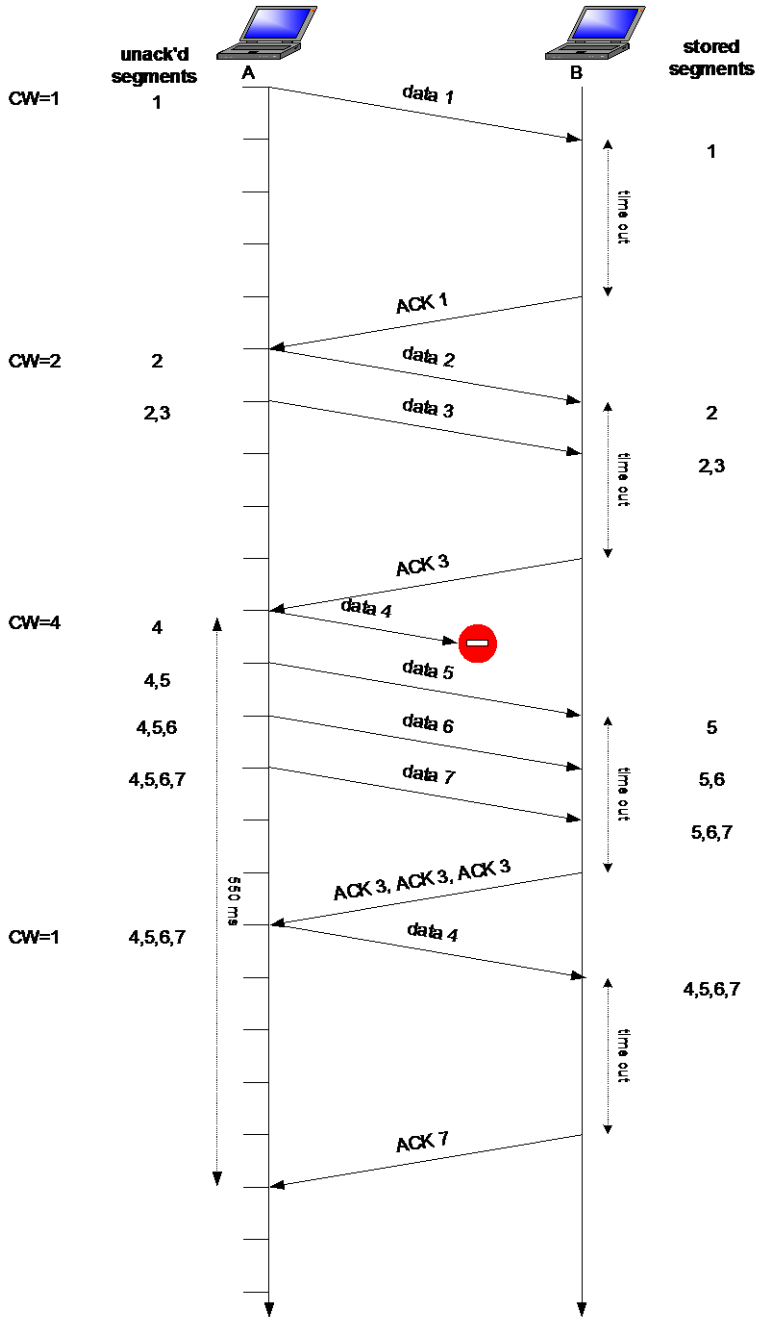
Risoluzione

1) Il un diagramma dei segmenti $j = 1,2,...,10$ inviati dall'host A è il seguente.



2) Il RTT associato ai segmenti 1, 2, 4 e 7 è pari a 250ms. Infatti, questi arrivano quando il buffer del ricevitore è vuoto. Pertanto, l'ACK che li riscontra deve includere tutto il tempo dell'*ACK timeout*. Non così per il segmento 8, che arriva quando l'*ACK timeout* è già stato attivato. Il RTT ad esso relativo è di 150ms. Visto che ogni segmento che arriva nello stesso istante in cui scatta l'*ACK timeout* avrà comunque un RTT di 250ms, un valore consono per il *retransmission timeout* è di 500 ms.

3) Nel caso in cui il segmento 4 vada perso, l'host B deve inviare un numero di ACK 3 pari al numero di segmenti fuori sequenza che ha ricevuto. Se, infatti, invia un solo ACK 3, il *fast retransmit* non può essere attivato.
4) Sotto l'ipotesi del punto 3, il diagramma è il seguente:



Il *fast retransmit* viene attivato perché il receiver invia un ACK per ogni segmento fuori sequenza ricevuto (3, in questo caso). Il fast retransmit avviene dopo 300ms dalla prima trasmissione del segmento n. 4, e l'ACK che riscontra il segmento n.4 (ACK 7) è ricevuto dopo 550ms.

Nota: visto che all'atto dell'invio dei 3 ACK 3 il buffer è non vuoto, alcuni studenti hanno risolto l'esercizio considerando che l'*ACK timeout* venisse fatto ripartire immediatamente. La soluzione è stata considerata valida.

5) Per rendere più efficiente l'attivazione del *fast retransmit*, B deve inviare *subito* un ACK quando riceve un segmento fuori sequenza, indipendentemente dall'*ACK timeout*. In questo caso, il tempo di recupero dell'errore è tagliato di 50ms. Peraltro, alcune implementazioni di TCP che sfruttano il *delayed ACK* si regolano esattamente in questo modo.