ESERCIZIO 2: Si considerino due applicazioni in comunicazione tramite una rete di calcolatori. Sia il *Round Trip Time* (RTT) tra i due *host* delle applicazioni costante nel tempo e pari a 2 time slot. Si supponga che le applicazioni utilizzino il protocollo di trasporto TCP, nella versione *Reno*, la quale impiega i meccanismi di *Slow start*, *Fast retransmit* e *Fast recovery* per il controllo della congestione. Il timeout sia pari a 3 time slots. Si considerino il tempo di trasmissione dei segmenti, il tempo di trasmissione dei messaggi di riscontro e il tempo di elaborazione del *TCP receiver* e del *TCP sender* trascurabili rispetto al RTT. Per semplicita` si suppongano i segmenti numerati a partire da 0. Sia inizialmente la *congestion window* (CW) pari ad 1. La *Advertised Window* del ricevitore abbia dimensione infinita.

Il candidato illustri tramite un diagramma temporale le fasi di invio dei segmenti e di ricezione dei messaggi di riscontro almeno per i segmenti numerati da 0 a 12. In particolare, si evidenzino i valori della CW e della *congestion threshold* (CT). Si considerino **separatamente** i seguenti casi:

1. Si supponga che il *TCP receiver* riceva correttamente tutti i segmenti tranne il 6. La ritrasmissione di tale segmento venga ricevuta correttamente dal *TCP receiver.*

2. Si supponga che il *TCP receiver* riceva correttamente tutti i segmenti tranne i seguenti: 2, 8. Le ritrasmissioni di tali segmenti vengano ricevute correttamente dal *TCP receiver.*

Inoltre:

3. Per ciascuno dei due casi precedenti il candidato produca un grafico che illustra l'andamento nel tempo del valore della CW.

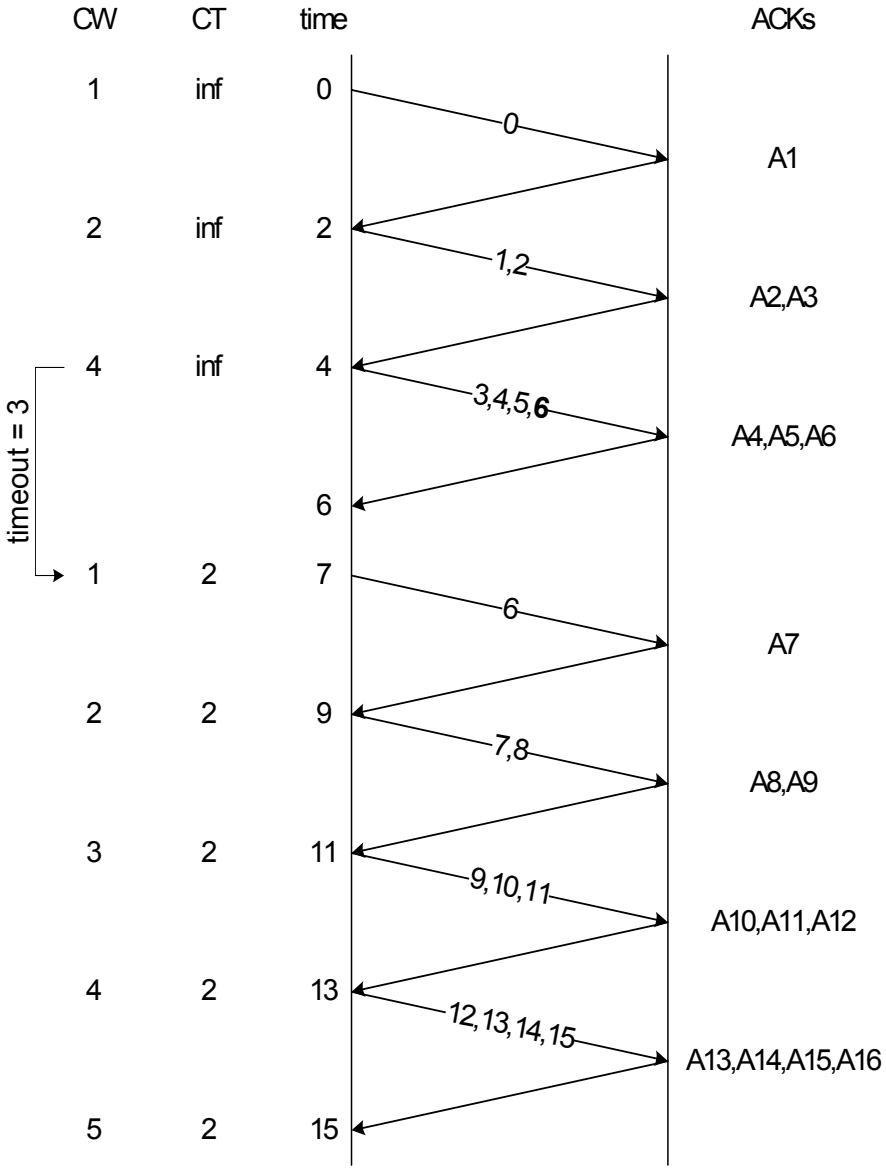
Altre ipotesi:

- L'applicazione mittente generi un numero di segmenti sufficiente a inviare ad ogni istante il massimo numero di segmenti ammessi dal livello trasporto TCP sottostante.
- Il *TCP sender* attenda il riscontro positivo di tutti i segmenti trasmessi nel medesimo gruppo prima di inviare nuovi segmenti.
- Il *TCP receiver* scarti tutti i segmenti ricevuti fuori ordine.
- Il *TCP receiver* invii un riscontro immediato (ACK) per ciascun segmento ricevuto correttamente e in ordine, e un riscontro duplicato (DUP-ACK) immediato per ciascun segmento ricevuto correttamente ma fuori ordine.

RISOLUZIONE

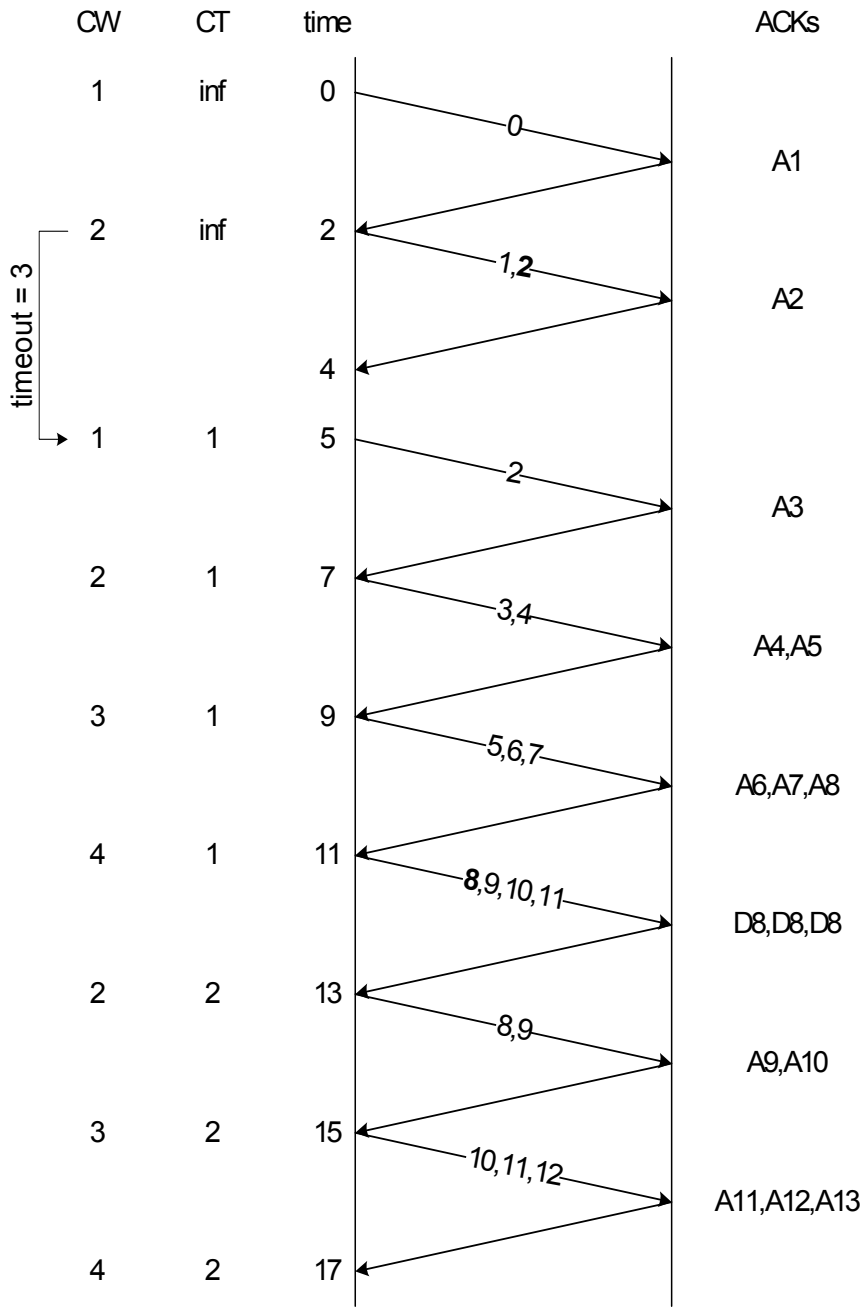
1. Il diagramma temporale di invio dei segmenti e ricezione dei riscontri e` illustrato in Figura 1. All'istante t = 7 il *TCP sender* rileva congestione in seguito al timeout dovuto alla mancata ricezione del riscontro del segmento 6. La successiva transmissione dei segmenti e` quindi governata dal meccanismo *Slow start*. A partire dall'istante t = 9, tuttavia, il valore della CW eguaglia quello della CT, per cui la fase di incremento esponenziale della CW termina. Inizia, invece, la fase di incremento lineare della CW, denominata *Congestion avoidance*.

Figura 1:



2. Il diagramma temporale di invio dei segmenti e ricezione dei riscontri e` illustrato in Figura 2. All'istante t = 5 il *TCP sender* rileva congestione in base al timeout dovuto alla mancata ricezione del riscontro del segmento 2. L'evoluzione temporale e` dunque simile a quella relativa al punto 1. All'istante t = 13, tuttavia, il *TCP sender* rileva congestione in base al meccanismo di *Fast retransmit*, in quanto tre riscontri duplicati vengono ricevuti consecutivamente. La CW e la CT vengono quindi entrambe impostante alla meta` del valore precedente alla congestione, e la trasmissione e` regolata dal meccanismo di *Congestione avoidance*, cioe` incremento lineare della CW.

Figura 2:



3. I diagrammi temporali del valore della CW nei casi 1 e 2, rispettivamente, sono riportati in Figura 3.

Figura 3: