ESERCIZIO 2: Si considerino due applicazioni in comunicazione tramite una rete di calcolatori, come illustrato in Figura 1. Il router A effettua *load balancing*, per cui i segmenti sperimentano Round Trip Time (RTT) differenti dipendenti dalla sequenza di invio come segue: il primo segmento inviato dal TCP sender sperimenta un RTT pari a 2 s, il secondo segmento un RTT pari a 4 s, poi nuovamente 2 s, poi nuovamente 4 s, e cosi` via per tutti gli altri segmenti. Si supponga che il RTT sia ripartito equamente nelle due tratte di rete dal TCP sender al TCP receiver e viceversa.

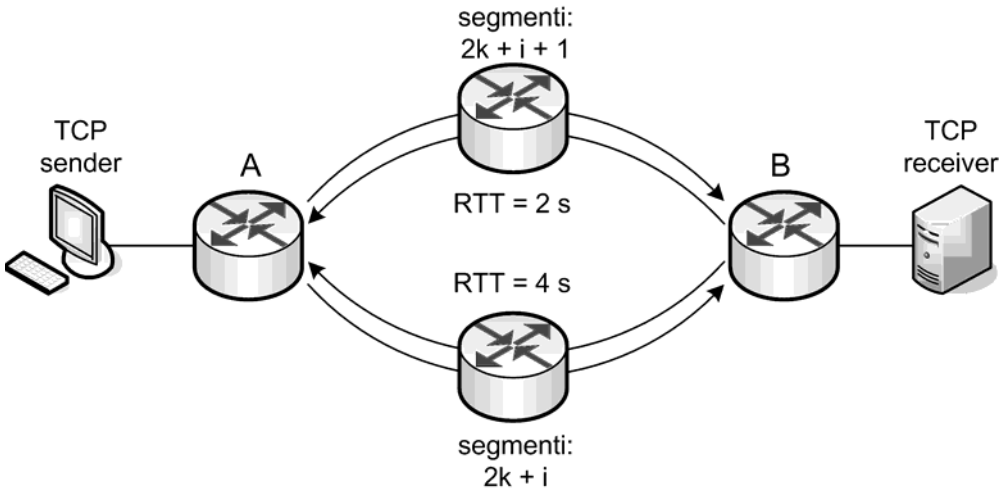

Figure 1 Applicazioni in comunicazione tramite la rete di calcolatori illustrata.

Si supponga che il TCP sender impieghi le seguenti tecniche di controllo della congestione: *slow start*, *congestion avoidance*, *fast retransmit*. Nota: il *fast recovery* **non** sia quindi attivo. Il timeout sia pari a 4 s. 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 1. Sia inizialmente la *congestion window* (CW) pari ad 1 e la *congestion threshold* (CT) infinita. Inoltre, la *advertised window* del ricevitore abbia dimensione infinita.

Il candidato risponda ai seguenti quesiti:

1. Si illustri tramite un diagramma temporale le fasi di invio dei segmenti e di ricezione dei messaggi di riscontro almeno per i segmenti numerati da 1 a 20. In particolare, si evidenzino i valori della CW e della *congestion threshold* (CT).

2. Si supponga ora che, a causa di un malfunzionamento, nessuno dei segmenti inviati sulla tratta di rete con RTT = 4 s venga ricevuto correttamente dal *TCP receiver.* Si calcoli il throughput (in segmenti/s) raggiunto dall'applicazione a regime. Si giustifichi la soluzione proposta tramite un diagramma temporale esemplificativo delle fasi di trasmissione dei segmenti e di ricezione dei riscontri.

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.
- Durante la fase di *slow start*, il *TCP sender* incrementi la *congestion window* di un segmento per ogni riscontro non duplicato ricevuto.
- Al di fuori della fase di congestione, il *TCP sender* invii immediatamente segmenti in maniera tale da avere in volo un numero di segmenti pari alla *effective window.* Invece, durante la fase di congestione, determinata tramite timeout o ricezione di riscontri duplicati, il *TCP sender* invii un segmento alla volta (politica *stop-and-wait*) e non incrementi la CW fino a quando tutti i segmenti in attesa di riscontro vengono indicati come correttamente ricevuti dal *TCP receiver.* N.B.: la ricezione del riscontro che indica al *TCP sender* che tutti i segmenti in volo sono stati ricevuti correttamente provoca **immediatamente** l'uscita dalla fase di congestione, con contestuale incremento della CW di una unita`.
- Il *TCP receiver* memorizzi tutti i segmenti ricevuti fuori ordine in un buffer di dimensione infinita.
- 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.
- Il *fast retransmit* entri in funzione alla ricezione del terzo riscontro duplicato consecutivo, a prescindere del valore della CW.

RISOLUZIONE

1. Il diagramma temporale di invio dei segmenti e ricezione dei riscontri e` illustrato in Figura 2. Siccome il *TCP receiver* memorizza i segmenti fuori ordine, il riordino dei segmenti da parte della rete non provoca l'arresto della CW, che cresce tuttavia in maniera piu` lenta rispetto allo *slow start* in RTT costante. Tuttavia, nel momento in cui la CW diventa tale per cui il numero di segmenti trasmessi sulla tratta di rete a RTT = 4 s e` maggiore o uguale di tre ($t$ = 14 s, nell'esempio), allora il meccanismo di *fast retransmit* induce (erroneamente) congestione sul *TCP sender*, il quale riporta immediatamente la CW a 1 (il *fast recovery* non e` utilizzato per ipotesi) e ritrasmette il segmento che e` considerato mancante. All'istante t = 18 s, comunque, tutti i segmenti in volo vengono riscontrati, per cui il *TCP sender* esce dalla fase di congestione e incrementa la CW.

2. Il diagramma temporale di invio dei segmenti e ricezione dei riscontri e` illustrato in Figura 3. Si noti come il sistema si rigeneri all'istante t = 12 s riproponendosi identico rispetto all'istante t = 2 s. Possiamo quindi con-

siderare l’intervallo [2 s, 12 s] come il periodo di rigenerazione, all’interno del quale calcolare il throughput dall’applicazione, che risulta quindi essere di 0.2 segmenti/s.

Figura 2: Soluzione domanda 1.

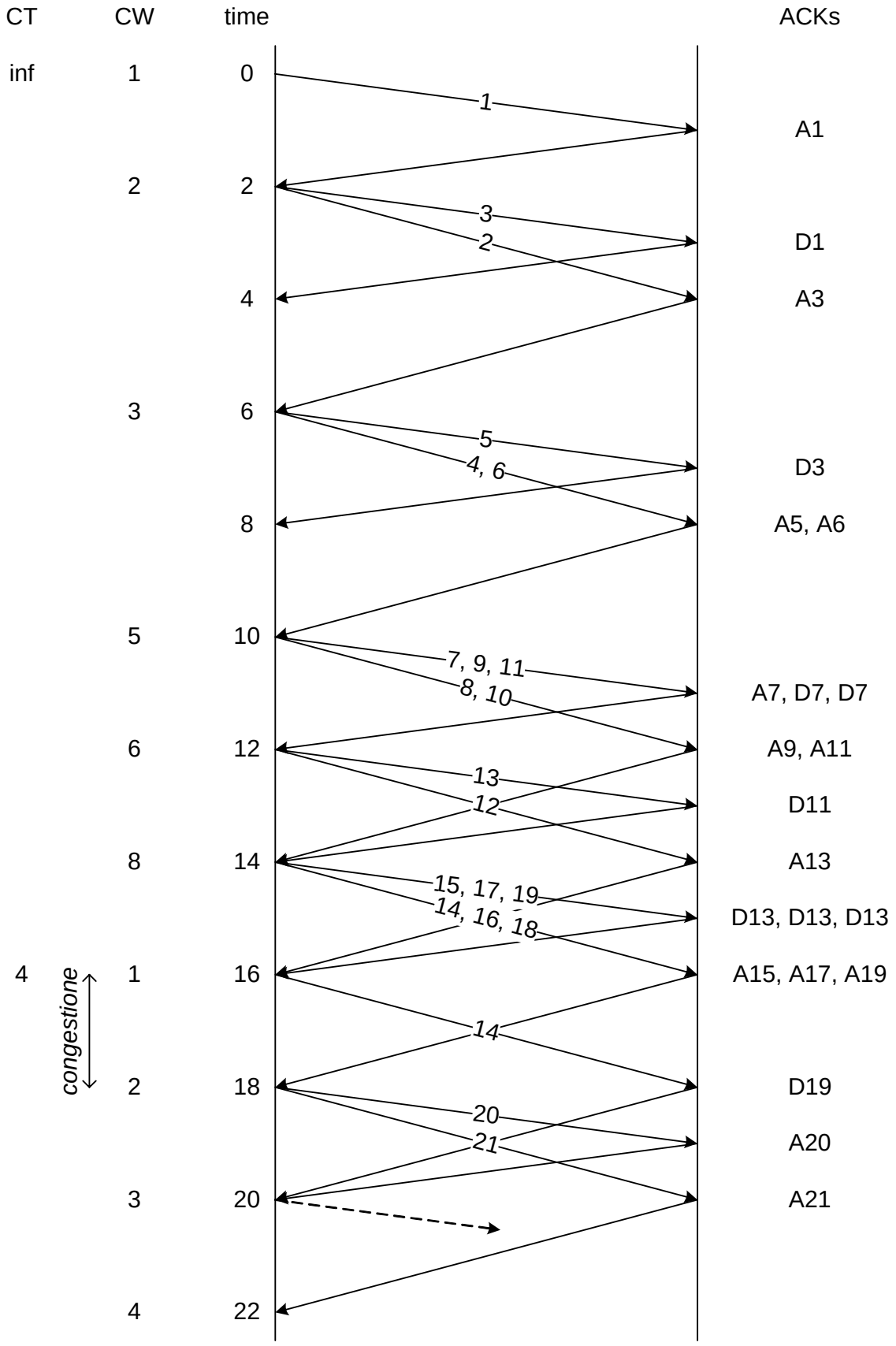

Figura 3: Soluzione domanda 2.