Esercizi per la Preparazione dell'orale di Reti Gennaio 2019 Livello Trasporto

1 Esercizio

Si mostrino i pacchetti scambiati (mostrando in particolare i valori del flag SYN e FIN, ed il contenuto dei campi SEQ ed ACK) durante una connessione TCP, in cui il client spedisce 800 byte ed il server spedisce 30 byte. Si assuma $MSS = 370$ byte con un header di 20 byte. Si includano i pacchetti di apertura e chiusura della connessione

2 Esercizio

Si consideri una connessione TCP tra due host A e B. Al tempo t=0 i valori di $EstimatedRTT$ e di $DevRTT$ dell'host A sono rispettivamente 0,120 s e 0,033 s. Supponiamo che, a partire dal tempo t, A invii la seguente sequenza di segmenti lungo la connessione TCP allhost B e che riceva i seguenti segmenti di ACK:

Segmenti inviati (in secondi)		Segmenti ricevuti (in secondi)	
S1	t=0,010	ACK S1	t=0,100
S2	t=0,015	ACK S2	t=0,210
S3	t=0,140	ACK S3	t=0,330
S4	t=0,190	ACK S4	t=0,350
S5	t=0,200	ACK S5	t=0,355
S6	t=0,210	ACK S6	t=0,360

Si chiede di calcolare i valori di Estimated RTT e di DevRTT all'host A per ogni segmento di ACK ricevuto, assumendo che il parametro usato per calcolare la media esponenziale ponderata Estimated RTT sia $\alpha = 0,125$, e che il parametro β usato per calcolare la media esponenziale ponderata DevRTT sia $\beta = 0,250$. Completare la seguente tabella:

ACK ricevuto	RTT campionato	Estimated RTT	Deviazione campionata	DevRTT
ACK S1				
ACK S2				
ACK S3				
ACK S4				
ACK S5				
ACK S6				

3 Esercizio

Si considerino due host A (mittente) e B (destinatario) che comunicano tramite il protocollo di trasporto TCP su un canale che ha un ritardo di propagazione costante e pari a 10 msec (conseguentemente il RTT di 20 msec). A livello di rete A e B utilizzano il protocollo IP, mentre il livello di collegamento sottostante ha un MTU pari a 1.500 bytes. Pertanto ogni segmento TCP può trasportare fino a 1.460 bytes di dati. Per semplicità si assuma inoltre che i tempi di trasmissione, di elaborazione e di accodamento siano trascurabili, e che il timeout del trasmittente sia fissato a 40 msec. Al tempo t=0, la finestra di trasmissione di A è vuota, come pure la finestra di ricezione di B. Inoltre, l'ultimo byte riscontrato nel flusso di dati da A a B ha numero di sequenza 10.000, mentre l'ultimo byte riscontrato nel flusso di dati da B ad A ha numero di sequenza 9.000. A partire da questo istante, il livello di trasporto di A riceve dall'applicazione la richiesta di invio a B di un blocco di 5.000 bytes di dati. Assumendo per semplicità che:

- i controlli di flusso e di congestione siano disabilitati,
- la finestra di trasmissione venga mantenuta costante dal trasmettitore per tutto il tempo necessario a trasferire il flusso di dati (e che questa sia sufficientemente ampia da contenere tutti i bytes da trasmettere),
- B mandi un riscontro immediatamente dopo la ricezione di ogni segmento dati,
- non ci siano perdite di dati in trasmissione,
- la trasmissione di un segmento richieda un tempo ϵ (molto piccolo),

utilizzare la seguente tabella per mostrare l'evoluzione nel tempo del protocollo:

t	Evento (su A o B)	N. sequenza nel segmento inviato	N. riscontro nel segmento inviato	Sendbase	NextSeqNum
0					

4 Esercizio

Si considerino gli effetti dell'uso di slow start su una linea con RTT di 10 msec. e nessuna congestione. La finestra di ricezione $24KB$ e la massima dimensione di segmento $2KB$. Quanto tempo necessario prima che possa essere inviata una finestra intera, se la slow start threshold $32KB$?

5 Esercizio: $Stop\&Wait$ modificato

Due host comunicano tramite una rete di calcolatori inaffidabile, che può ritardare o perdere i pacchetti a causa di fenomeni di congestione, ma che non li può corrompere. Per comunicare i due host utilizzano un protocollo di tipo $Stop\&Wait$ che, oltre al normale funzionamento, prevede un ulteriore meccanismo per mitigare gli effetti della congestione. Questo meccanismo opera come segue:

- per ogni pacchetto trasmesso, il mittente misura il tempo $estimated-RTT$ che è intercorso tra l'istante di spedizione e la ricezione dell'ACK. Se $estimated-RTT > sogliaRTT$ (dove $sogliaRTT$ è un parametro preconfigurato del mittente) allora il mittente non accetta ulteriori pacchetti da spedire per un tempo pari a $Idle$ (anche questo un parametro preconfigurato del protocollo).
- inoltre, quando scatta un timeout che segnala la mancata ricezione di un ACK, il pacchetto viene ritrasmesso solo dopo un tempo pari a $Ritrasm$ (nel frattempo non accetta altri pacchetti da trasmettere dal livello applicazione).

Scrivere l'automa a stati finiti per il mittente del protocollo descritto sopra. Per semplicità, nella figura seguente, si riportano i soli stati *Wait for call 0 from Above* e *Wait for ACK0*. Si chiede di aggiungere gli stati necessari con le relative transizioni e di ignorare le transizioni in uscita dagli stati *Wait for ACK1* e *Wait for call 1 from above*.
Utilizzare le seguenti funzioni:

- $Rdt\_rcv(pkt)$: per ricevere un pacchetto dal livello di rete
- $Udt\_send(pkt)$: per spedire il pacchetto pkt
- $Rdt\_send(data)$: per ricevere un dato da trasferire dal livello applicazione
- $Start\_timer(x)$: per impostare il timer al tempo x
- $Stop\_timer()$: per fermare il timer

6 Quesiti

Riguardo al protocollo UDP, indicare quali delle seguenti affermazioni sono vere e quali false

- è eseguito dai router della rete

- non provvede a riordinare eventuali dati fuori sequenza prima di consegnarli all'applicazione
- garantisce una velocità minima di trasmissione
- normalmente impiegato da tutte le applicazioni per le quali l'integrità dei dati fondamentale
- prima dell'invio dei dati provenienti dall'applicazione stabilisce una connessione
- un protocollo di livello trasporto

7 Quesiti

Riguardo al protocollo HTTP, indicare quali delle seguenti affermazioni sono vere e quali false

- questo specifica uno scambio di messaggi testuali tra browser e web server
- le richieste HTTP sono costituite da una linea di stato, eventuali header, una linea vuota ed eventualmente un corpo
- il metodo GET viene impiegato dal browser per inviare una pagina
- il server specifica l'esito della richiesta mediante un numero