RETI DI CALCOLATORI – prova di verifica intermedia del 31/10/2017

Prima parte (21 punti)

Matricola_____Cognome_____Nome_____Fila_____Posto_____

Q1. Supponiamo che un router A trasmetta, a partire dal tempo T, un pacchetto di 256 byte a un router B direttamente collegato ad A. Supponiamo che la distanza tra A e B sia di 80 km, che la velocità di propagazione sia di 2×10^8 m/s e che la velocità di trasmissione sia di 2 Mbps. Quanti bit saranno arrivati a B nel momento in cui verrà immesso nel collegamento l'ultimo bit del pacchetto?

SOLUZIONE: tempo di immissione dell'ultimo bit =**$1024x10^{-6}$**; ritardo di propagazione=**$400x10^{-6}$**; bit arrivati=**1248**

Q2. Maria invia (mediante *webmail*) una email a Dario, che la riceve con *webmail*. Indicare il numero minimo possibile di connessioni TCP necessarie per l'invio e la ricezione di cui sopra, nonche' tra chi vengono instaurate tali connessioni.

SOLUZIONE: numero minimo di connessioni TCP: **3** ; le connessioni sono stabilite tra **l'host di Maria** e **il suo mail server**; tra **il mail server di Maria** e **il mail server di Dario** ;tra **il mail server di Dario** e **l'host di Dario** ;

Q3. Un server DNS locale invia una query di tipo A ad un server DNS top level domain e riceve una risposta R di tipo A. La query è stata risolta in modo iterativo o ricorsivo?

SOLUZIONE: la query è stata risolta in modo **iterativo o ricorsivo, non si può distinguere se il tipo della risposta è A.**

Q4. Quali protocolli di livello applicativo, e quando (cioè sotto quali ipotesi), **non** inviano **mai** ack di livello trasporto in piggybacking?

SOLUZIONE: protocollo **DNS** quando **usa UDP** , protocollo **FTP** quando **trasferisce dati**

Q5. Un client C apre una connessione TCP con un server FTP F sul canale di controllo. Indicare i valori contenuti nei campi seq, ack, e rwnd ed i nomi dei flags a true dei segmenti scambiati per l'apertura della connessione, nell'ipotesi che i numeri di sequenza iniziali di C ed F siano 800 e 2300, rispettivamente, e che inizialmente le finestre di ricezione di C ed F siano di 12000 e 7000 byte, rispettivamente.

SOLUZIONE:
primo segmento: seq= **800** ack= **?** Rwnd= **?** flags a true= **SYN**
secondo segmento: seq= **2300** ack= **801** rwnd= **7000** flags a true=**SYN+ACK**
terzo segmento: seq= **800** *(ack non in piggybacking, per FTP)* ack= **2301** rwnd= **12000** flags a true= **ACK**

Q6 e Q7. Al tempo t_0 un server web W ha una connessione TCP aperta con un client C, per cui il valore di cwnd è 6, ed ha nella sua finestra di invio tre segmenti di 1 MSS in volo (con S_f = Y), ed altri 2 MSS di dati nuovi da spedire. Il primo evento successivo a t_0 è la ricezione di un riscontro R, con valore del campo acknumber = X, a seguito del quale invia i dati nuovi in finestra: la fine di questo invio avviene al tempo t_1. Indicare il possibile stato, il valore di rwnd e di ssthreshold al tempo t_0 *(Q6)* e il possibile stato, il valore di rwnd, di ssthreshold e di cwnd al tempo t_1 *(Q7)*.

SOLUZIONE:
-*(Q6)* Lo stato al tempo t_0 può essere *slow start*? **SI** Se si, al tempo t_0 il valore di rwnd è **3MSS** e quello di ssthreshold è **> 6MSS** mentre *(Q7)* al tempo t_1 lo stato è **C.A. o S.S.** *(dipende dalla soglia)*, il valore il valore di rwnd è **>=5MSS-X+Y**, quello di ssthreshold è **<=7MSS se lo stato è C.A.; >7 se lo stato è S.S.** e quello di cwnd è **7** ;

- *(Q6)* Lo stato al tempo t_0 può essere *congestion avoidance*? **SI** Se si, al tempo t_0 il valore di rwnd è **3MSS** e quello di ssthreshold è **6MSS** mentre *(Q7)* al tempo t_1 lo stato è **Congestion Avoidance** , il valore il valore di rwnd è **>=5MSS-X+Y**, quello di ssthreshold è **6MSS** e quello di cwnd è **37/6 MSS** ;

- *(Q6)* Lo stato al tempo t_0 può essere *fast recovery*? **SI** Se si, al tempo t_0 il valore di rwnd è **3MSS** e quello di ssthreshold è **<= 3MSS** *(e >=2MSS)* mentre *(Q7)* al tempo t_1 lo stato è **Congestion Avoidance** , il valore il valore di rwnd è **>=5MSS-X+Y**, *(e >=ssthreshold)* quello di ssthreshold è **<= 3MSS** *(come a t_0)* e quello di cwnd è **uguale ad ssthreshold** ;

Seconda parte (9 punti)

E1. Si consideri una variante del protocollo **Go Back N** in cui la ritrasmissione dei pacchetti non avviene con la regola normalmente utilizzata nel protocollo del materiale didattico, ma avviene utilizzando la tecnica usata in **TCP** versione **Reno**. Per il resto, si comporta come il normale protocollo Go Back N.

Descrivere, mediante un automa a stati finiti che utilizza pseudocodice (come nell'automa [GBN] del materiale didattico) e non con descrizione delle attività a parole, il comportamento del **sender** della variante del protocollo Go Back N sopra descritta. Si supponga che il receiver si comporti correttamente rispetto a tale tecnica.

L'automa ha un unico stato. Si riportano solamente le transizioni. Le modifiche rispetto all'automa di base sono in grassetto.

RDT_send(data)

```
if (nextseqnum < base+N) {
   sndSgm[nextseqnum] = make_segment(nextseqnum,data)
   UDT_send(sndSgm[nextseqnum])
   if (base == nextseqnum)
      start_timer
   nextseqnum++
                                  }
else
   refuse_data(data)
```

Inizializzazione

---

base=1

nextseqnum=1

ndup=0

rcvSgm=UDT_rcv() &&(corrupted(rcvSgm) || ! isACKinWindow(rcvSgm))

NOP

timeout()

UDT_send(sndSgm[base])

start_timer()

rcvSgm=UDT_rcv() && ! corrupted(rcvSgm) && isACKinWindow(rcvSgm)

---

base = getacknum(rcvSgm)

If (base == nextseqnum) stop_timer() else start_timer()

ndup=0

le due seguenti transizioni possono anche essere unificate, utilizzando un *if* per distinguere i due casi sul valore di ndup

rcvSgm=UDT_rcv() && ! corrupted(rcvSgm) **&& getacknum(rcvSgm)<base&&ndup<2**

---

ndup++

rcvSgm=UDT_rcv() && ! corrupted(rcvSgm) **&& getacknum(rcvSgm)<base&&ndup=2**

---

ndup=0

UDT_send(sndSgm[base])

start_timer()