RETI DI CALCOLATORI – prova scritta del 19/1/2018

Per essere ammessi alla prova orale è necessario ottenere una valutazione sufficiente della prima parte e una votazione totale di 15 o superiore.

Prima parte (15 punti)

Matricola_____Cognome_____Nome_____

Q1. Al tempo t, un host H ha attivi solamente tre protocolli di livello applicativo: il browser, con 5 schede contemporaneamente aperte, la ricezione di due file (da due siti diversi) mediante FTP, e la ricerca di un indirizzo IP mediante il suo resolver DNS. Quante connessioni TCP sono aperte al tempo t in H?

RISPOSTA: Le connessioni TCP contemporaneamente aperte al tempo t sono **5+4+0=9**

Q2. E' possibile che in un browser ci siano due cookies identici?

RISPOSTA: SI, **perché i cookies vengono assegnati dai server (che non si coordinano) e non dai browser**
oppure NO, perchè_____

Q3. Un client C ha una connessione TCP aperta con un server FTP F sul canale dati. Indicare i valori contenuti nei campi seq, ack, ed i nomi dei flags a true dei segmenti scambiati *per la chiusura della connessione*, se questa avviene mediante *half-close*, nell'ipotesi che i numeri di sequenza utilizzati da C ed F nei segmenti immediatamente prima dell'inizio della chiusura (tutti lunghi 4096 byte) siano 44500 e 952300 (tale segmento di F è inviato dopo l'invio da parte di C del suo ultimo segmento di cui sopra), rispettivamente, che F debba inviare (per terminare l'invio del file richiesto) due segmenti di 4096 byte nel momento in cui riceve la richiesta di chiusura da parte di C, e che il FIN di F *non* sia inviato in piggybacking.

RISPOSTA:
primo segmento: seq= **48596** ack= **956396** flags a true= **FIN+ACK**
secondo segmento: seq= **956395** ack= **48597** flags a true= **ACK**
terzo segmento: seq= **964588** ack= **48597** flags a true= **FIN+ACK**
quarto segmento: seq= **48597 (oppure 48596)** ack= **964589** flags a true= **ACK**

Q4. Si consideri un sistema autonomo composto da 47 router, di cui 5 sono router di frontiera. Quante sessioni BGP (sia e-BGP che i-BGP) vengono in totale instaurate nel sistema autonomo?

RISPOSTA: nel sistema autonomo vengono instaurate in totale **(47x46)/2+5=1086** sessioni BGP

Q5. Una rete locale in cui sono attivi 3 nodi, A, B e C, utilizza il protocollo slotted aloha 1-persistente e la tecnica binary exponential backoff per i periodi di attesa prima di ritrasmettere i frame che hanno subito collisioni. Nello slot X-2 i frame trasmessi dai tre nodi (quelli di A e B trasmessi per la prima volta, quello di C per la terza) subiscono una collisione. Successivamente, nello slot X, A e B trasmettono il loro frame per la seconda volta. Qual'è la probabilità che la ritrasmissione successiva dei due frame che subiscono collisione nello slot X avvenga senza collisioni, e che la trasmissione del frame di C avvenga (sempre senza collisioni) nello slot X+3? Si supponga che i nodi ricevano la notifica della collisione nello slot successivo a quello della collisione.

RISPOSTA: la probabilità richiesta è: **(1/4)x(1/4)x3x2x(1/8)=3/64**

Seconda parte

E1 (9 punti). Descrivere con uno pseudocodice il comportamento di TCP Reno quando riceve un riscontro non duplicato. Si usino (tra le altre) le seguenti variabili e funzioni/procedure:
- *SWND[]* finestra che contiene i dati da inviare o in volo
- *makesegment(A,B,C)* che costruisce un segmento di C byte (a partire dal byte B) di dati memorizzati nella variabile A
- *send(S)* per inviare il segmento S.

Descrivere il contenuto o la funzionalità delle altre variabili e funzioni/procedure eventualmente utilizzate.

SOLUZIONE:

```
stoptimer(); // per evitare che scada durante l'esecuzione del codice
Sf=S.ack; //Sf è il puntatore al primo byte non riscontrato della finestra; S è il segmento ricevuto con il nuovo ack
rwnd=S.rwnd;  //aggiorno il valore di rwnd usando quello appena ricevuto
if stato==slowstart              //stato è la variabile che indica lo stato attuale di TCP: si aggiornano cwnd e stato
   { cwnd++;
     if cwnd >= ssthresh
          { stato=congestionavoidance};
   }
else if stato=congestionavoidance
        {  cwnd=cwnd+(1/cwnd)};
       else if stato=fastrecovery
                {   cwnd=ssthresh;
                    stato=congestionavoidance;
                 };
while  (min(cwnd,rwnd)>Sn-Sf)&&(Sn<lastbyteinwindow)    // Si inviano nuovi dati, se del caso: Sn è il puntatore
                                  //all'ultimo byte inviato, lastbyteinwindow è il puntatore all'ultimo byte memorizzato e
                                  //non ancora inviato, min calcola il minimo dei suoi parametri
         { size=min(MSS,lastbyteinwindow-Sn,cwnd-(Sn-Sf),rwnd-(Sn-Sf));  //size è la dimensione dei dati nuovi da
                                                                          //spedire
           SEG=makesegment(SWND,Sn,size); //makesegment costruisce un segmento con i dati da Sn+1 a size di SWND
           send(SEG);
           Sn=Sn+size;
          };
if Sn>Sf {starttimer()}; //se ci sono segmenti in volo si fa ripartire il timer per il più vecchio
```

E2 (6 punti). Descrivere con uno pseudocodice il comportamento di un router R di un sistema autonomo che usa OSPF come protocollo di routing, quando riceve un messaggio M *link-state update* da un vicino V, che contiene, nel campo *M.costo* il nuovo costo di un collegamento, nel campo *M.link* la coppia di nodi estremi di tale collegamento, e nel campo *M.seqnum* il numero di sequenza dell'update. La variabile *LastSN[L]* contiene il numero di sequenza più alto dei link-state update finora ricevuti e relativi al link L (serve per non considerare updates vecchi). Tale numero di sequenza è un intero non negativo: il suo valore è inizializzato a 0. Inoltre, sia *C[]* l'array che contiene i costi aggiornati dei collegamenti. Infine, il calcolo dei nuovi cammini minimi viene effettuato dalla procedura *recomputeSP* che riceve come parametro C, e che non occorre realizzare. Descrivere il contenuto o la funzionalità delle altre variabili e funzioni/procedure eventualmente utilizzate. Per semplicità, i vicini di R sono rappresentati dai numeri interi tra 1 ed N.

SOLUZIONE:

```
if LastSN[M.link]<M.seqnum  //else è un update già considerato e va ignorato
   { LastSN[M.link]=M.seqnum;
     for i=1 to N do   //invia l'update a tutti i vicini tranne V
          {  if i!=V
                {send(i,M)};
           };
      C[M.link]=M.costo;  //aggiorna il costo del link
       shortestpaths=recomputeSP( C );  // ricalcola i cammini minimi
    }
```