RETI DI CALCOLATORI – prova scritta del 5/09/2017

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)

Q1. Si consideri una rete Ethernet con **topologia a stella**, con velocità di trasmissione effettiva di 92 Mbps, e con velocità di propagazione di $2x10^8$ m/sec. Al tempo t, il nodo A della rete inizia ad inviare un frame di 1104 byte destinato al nodo B. B finisce di ricevere completamente il frame al tempo t+97,75 microsecondi. Qual'è la distanza tra A e B? Giustificare la risposta.

Q2. Sia cwnd(i) l'i-esimo valore assunto dalla variabile cwnd del TCP di un host H. Se cwnd(i) =2,5, quali sono tutti i valori possibili per cwnd(**i-1**), nonché gli stati relativi? Giustificare la risposta.

Q3. Sia cwnd(i) l'i-esimo valore assunto dalla variabile cwnd del TCP di un host H. Se cwnd(i) =2,5, quali sono tutti i valori possibili per cwnd(**i+1**), nonché gli stati relativi? Giustificare la risposta.

Q4. Una rete R di 93 nodi utilizza l'algoritmo distance vector con *split horizon* per determinare i cammini interni ad R. Sia C un router di R che è collegato con altri 8 router della stessa rete. Qual'è il numero *minimo* di righe delle tabelle che C invia ai suoi vicini? Giustificare la risposta.

Q5. In una rete Ethernet con topologia a bus, con bus lungo 200 metri, con velocità di trasmissione effettiva di 16 Mbps, con lunghezza del jamming signal di 48 byte, con slot di attesa pari al tempo di trasmissione di 512 byte, e con velocità di propagazione di $2x10^8$ m/sec, due nodi A e B iniziano simultaneamente al tempo t ad inviare per la seconda volta un frame di 2400 byte. Un terzo nodo C *inizia* ad eseguire il protocollo per trasmettere un frame della stessa lunghezza a partire dal tempo t+80 microsecondi. Qual'è la probabilità che il frame di C venga trasmesso senza mai subire collisioni, se nessun altro nodo della rete vuole trasmettere e se il protocollo adotta il metodo **non** persistente? Giustificare la risposta.

Seconda parte (15 punti)

E1 (7 punti). Descrivere con uno pseudocodice la procedura *mailservercanonico(nome,H)* di un server DNS locale, invocata dal resolver del generico host H, per tradurre il nome simbolico *nome* utilizzando la risoluzione iterativa relativamente al mailserver canonico di un dominio *dom*. Nello pseudocodice, si utilizzino (tra le altre) le procedure/funzioni:

- *UDPreceive(porta,RR)* per ricevere un a risposta*;*
- *UDPsend(ser,porta,RR)* per inviare una richiesta al server "ser";
- *deliver(addr,H)* per restituire al resolver di H l'indirizzo richiesto.

Per semplicità, il server gestisce una sola richiesta alla volta e non effettua caching. Inoltre, si assuma che ogni risposta DNS sia relativa ad una sola richiesta, che i resource record di risposta di tipo *NS* e *CNAME* contengano nel campo *value* l'indirizzo IP (e non il nome) del server opportuno, e che tutti i pacchetti destinati al server locale arrivino correttamente e in tempo. Infine, si supponga di conoscere l'indirizzo IP del primo server da contattare.

E2 (8 punti). Un sistema autonomo utilizza il protocollo DHCP, ed ha 3 server DHCP, ognuno dei quali ha a disposizione 100 indirizzi IP (esclusivi) da assegnare agli host del sistema autonomo che ne fanno richiesta. Descrivere, con un automa a stati finiti, il comportamento del livello applicativo di uno di questi server quando esegue DHCP. Si supponga che il server utilizzi (tra le altre) le variabili:

- *myaddr* che contiene l'indirizzo IP del server stesso,
- *myLT* che contiene il valore del lease time di quel server,
- *tabaddr* tabella hash indirizzabile sia per indice, che con la chiave uguale al campo indir (vedi sotto), di 100 righe, una per ogni indirizzo che può assegnare, con i campi (tra gli altri)
  - *indir* che contiene l'indirizzo IP,
  - *status* che indica lo stato dell'indirizzo: libero, riservato, occupato,
  - *transid* che contiene il valore della transaction id relativa alla richiesta

e (tra le altre) le funzioni/procedure:

- *selectfreeaddr(tabaddr)* funzione che restituisce un indirizzo IP in tabaddr libero, se ci sono indirizzi liberi, e restituisce -1 altrimenti,
- *selectby id(tid)* che restituisce l'indice dell'elemento in tabaddr il cui campo transid è uguale a tid,
- *fhash(addr)* funzione hash per accedere alla tabella in modalità hash,
- *starttimer(ind)* che invoca il gestore del timer (**che non occorre realizzare**) affinchè inserisca nella sua coda delle scadenze una nuova scadenza relativa all'indirizzo ind,
- *timeout(ind)* che viene invocata dal gestore del timer per comunicare al server che il lease time dell'indirizzo ind è scaduto.

Traccia della soluzione

Q1. 97,75= tempo di trasmissione+tempo di propagazione. Siccome 1104 byte= 8832 bit e R=92Mbps, il tempo di trasmissione è di 8832/92=96 microsecondi. Quindi, il tempo di propagazione è di 1,75 microsecondi, e quindi la distanza tra A e B è di $1{,}75 \times 2 \times 10^2$ = 350 metri.

Q2. Siccome cwnd è sempre intero in slow start, e in fast recovery cwnd è > 3, il TCP si trova in congestion avoidance. Il valore di cwnd(i-1) poteva essere 2 ed aver ricevuto un ack non duplicato, sempre nello stato di congestion avoidance, oppure poteva essere in fast recovery, con ssthresh=2.5 ed aver ricevuto un nuovo ack: in questo caso il valore di cwnd(i-1) poteva essere X.5 con X intero e maggiore di 4.

Q3. Il valore di cwnd(i+1) può essere 5/2+2/5=2.9 se rimane in congestion avoidance e riceve un ack non duplicato, oppure 1 se scade il time out e passa in slow start, oppure 5/4+3=17/4 se riceve 3 ack duplicati e passa in fast recovery.

Q4. Split horizon prevede che la tabella inviata da un router ad un suo vicino contenga solamente una riga per ogni destinazione raggiungibile passando per un router diverso da quello a cui sta inviando la tabella. Quindi, se tutte le destinazioni sono raggiungibili solamente passando per un vicino V, C invierà una tabella vuota a V.

Q5. La modalità non persistente, prevede che un nodo inizi a trasmettere solo se trova il canale libero, altrimenti aspetta un tempo casuale e ripete il protocollo. Quindi, due nodi collidono solamente se ascoltano il canale prima che arrivi loro la trasmissione dell'altro nodo. Nel nostro caso, se C ascolta il canale al tempo t, lo trova libero ed inizia a trasmetter , può avere collisione con A o B solo se questi hanno iniziato a trasmettere nell'intervallo (t-1, t+1) (il tempo è espresso in microsecondi): infatti, nel nostro caso il ritardo di propagazione è di 1 microsecondo. Il tempo di trasmissione del jamming signal è di 48/2=24 microsecondi, ed ogni slot di attesa (T_{fr}) è lungo 512/2=256 microsecondi, mentre il tempo di trasmissione del frame di A e B è 2400/2=1200 microsecondi. Quindi, C ascolterà il canale ai tempi t+80+X256, con X che può assumere valori non negativi (a seconda dei casi di collisione/attesa di A e B), mentre A e B lo faranno ai tempi t+Y25+Z256 (con Y e Z che possono assumere valori positivi): non può mai capitare che C collida con A o B, e quindi la probabilità richiesta è 1.

E1. I campi classe e TTL dei resource record non vengono utilizzati e quindi sono ignorati nella soluzione.

```
{nonA==true;
//prepara la prima richiesta
 RR.type==MX;
 RR.domname==nome;
 server==firstserver; //firstserver è il primo server da contattare
 UDPsend(server, 53, RR)
 UDPreceive (53,M); //per ricevere il resource record M
 while (nonA)
  {
    if M.type=NS //se la risposta è NS si deve iterare la richiesta
                  { server==M.value;
                    UDPsend(server, 53, RR);
                    UDPreceive (53,M);
                  }
   else  if (M.type=A) || (M.type=CNAME) //se la risposta contiene l'indirizzo richiesto
                                  {nonA==false;
                                   deliver(M.value,H);
                                   }
  }
```

E2. L'automa ha un solo stato. Le transizioni finiscono in quell'unico stato, e sono:

DHCPDISCOVER(M)

```
y==selectfreeaddr(tabaddr);
if y>0
        {    DHCPOFFER.transid==M.transid;
             DHCPOFFER.leasetime==mylt;
             DHCPOFFER.clientaddr==NIL;
             DHCPOFFER.youraddr==y;
             DHCPOFFER.serveraddr==myaddr;
              tabaddr(fhash(M.clientaddr)).status==riservato;
              tabaddr(fhash(M.clientaddr)).transid==M.transid;
             UDPsend(255.255.255.255, sourceport=67, destport=68, DHCPOFFER);
        }
```

DHCPREQUEST(M)

```
if  M.serveraddr=myaddr
        {    DHCPACK.transid==M.transid;
             DHCPACK.leasetime==mylt;
             DHCPACK.clientaddr==NIL;
             DHCPACK.youraddr==M.clientaddr;
             DHCPACK.serveraddr==myaddr;
             UDPsend(255.255.255.255, sourceport=67, destport=68, DHCPACK);
             starttimer(M.clientaddr);
             tabaddr(fhash(M.clientaddr)).status==occupato;
        }
else { j==selectbyid(M.transid);
        tabaddr(j).status==libero;}
```

timeout(addr)

```
tabaddr(addr).status==libero;
```