RETI DI CALCOLATORI - prova scritta 03/11/2015

Per essere ammessi alla prova orale è necessario ottenere una valutazione sufficiente sia della prima parte che dell'intera prova scritta.

Prima parte (12 punti)

Q1. Un server DNS locale invia una query di tipo A e riceve una risposta R di tipo NS. Indicare – giustificando la risposta – se la query è stata risolta in modo iterativo o ricorsivo, e se R può contenere tuple di tipo A.

Q2. Un client C ha stabilito una connessione TCP con un server web S per scaricare una pagina web che consiste di tre oggetti. Al tempo t, subito dopo avere inviato la richiesta per il terzo oggetto, l'host di C invia a S un segmento con il flag FIN a true. Indicare – giustificando la risposta – il tempo minimo necessario al TCP di C per chiudere definitivamente la connessione supponendo che la dimensione del terzo oggetto sia 1,5 MSS, che RTT sia costantemente 700 msec, che il *maximum segment lifetime* sia 1100 msec, che tutti i segmenti vengano ricevuti corretti ed in ordine, e che il valore di *cwnd* del TCP di S sia 1MSS quando esso riceve il FIN inviato dall'host di C. Trascurare i tempi di preparazione e di trasmissione dei segmenti e assumere che S abbia già a disposizione tutti i dati da inviare.

Q3. Indicare – giustificando la risposta – se, e in che modo, frame Ethernet possono essere trasportati in pacchetti ARP o viceversa.

Q4. Consideriamo una rete locale che utilizza il protocollo MAC Slotted Aloha p-persistente, con p=0,75, e supponiamo che un host A di tale rete trasmetta per la prima volta un frame durante lo slot S, rilevando una collisione. Indicare – giustificando la risposta – se la probabilità che A riesca a trasmettere quel frame con successo nello slot S+4 senza rilevare altre collisioni è superiore a 1/16.

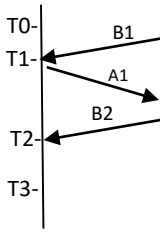
Seconda parte (18 punti)

E1 (6 punti). Consideriamo un'applicazione ITV client-server che permette ai client di eseguire il download di programmi televisivi. ITV utilizza UDP e realizza a livello application un controllo del flusso. I client inviano messaggi di tipo `GET(fileName)`, per richiedere il download di un file, e di tipo `WSIZE(size)`, per comunicare la quantità di nuovi dati che possono ricevere. Descrivere con uno pseudocodice il comportamento di un server ITV utilizzando le operazioni

```
UDPsend(localPort, remoteIP, remotePort, message)
<remoteIP, remotePort, message> UDPreceive(localPort)
```

per l'invio e la ricezione di messaggi e dati.

E2 (6 punti). Al tempo T0 il TCP di un host A ha già stabilito una connessione con il TCP di un altro host B, ha 2 MSS di dati in volo (spediti in due segmenti *full-sized*), 2 MSS di dati non ancora spediti, il numero di sequenza del segmento più vecchio in volo è X, la dimensione della finestra di congestione *cwnd* è 6 MSS e il valore di *ssthresh* è 5,5 MSS. Al tempo T1>T0 riceve da B un riscontro B1 non duplicato e non contenente dati e in conseguenza di ciò invia un segmento A1 contenente 1 MSS di dati. Al tempo T2>T1 riceve da B un altro riscontro B2. Sapendo che nell'intervallo [T0,T3] non si verifica alcun altro evento oltre quelli sopra menzionati, indicare –giustificando la risposta– i possibili valori dei campi *ackNum* e *rwnd* contenuti in B1 e B2 e di *cwnd* al tempo T3.



E3 (6 punti). Consideriamo una rete che contiene due server DHCP. Supponiamo che un host che desideri richiedere l'assegnazione di un indirizzo IP esegua il protocollo DHCP ma debba attendere di ricevere un'offerta da entrambi i server per poi scegliere l'indirizzo con il *lease time* più alto. Se l'host non riceve un'offerta da entrambi i server entro un tempo TO ripete il protocollo dall'inizio. Descrivere utilizzando un automa a stati finiti il comportamento del demone di livello application di tale protocollo in esecuzione negli host, utilizzando:

- l'evento **`getIP`**`()` per la richiesta al demone,
- le operazioni

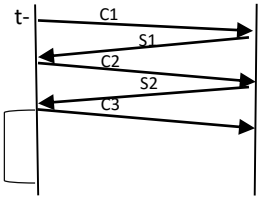
  ```
  UDPsend(localPort, remoteIP, remotePort, message)
  <remoteIP, remotePort, message> UDPreceive(localPort)
  ```

  per l'invio e la ricezione di messaggi DHCP,
- l'operazione **`buildDHCPmsg`**`(type,tid,lt,ca,yia,sa)` per costruire un messaggio DHCP e i metodi `m.type`, `m.tid`, `m.lt`, `m.ca`, `m.yia`, `m.sa` per accedere ai campi corrispondenti di un messaggio DHCP `m`.

TRACCIA DELLA SOLUZIONE

Q1. La query è stata risolta in modo iterativo dato che il server locale ha ricevuto una risposta di tipo NS. Se la query fosse stata risolta in modo ricorsivo il server locale avrebbe infatti ricevuto una risposta di tipo A.
La risposta R include (nella sezione "Additional") per ogni tupla di tipo NS (contenuta nella sezione "Answer") almeno una tupla di tipo A contenente un indirizzo IP del nome del server indicato nella tupla di tipo NS.

Q2. Il TCP di C riceve al tempo t + RTT il riscontro S1 del segmento FIN da lui inviato. Se S1 trasporta in piggybacking il primo MSS dei dati del terzo oggetto, il TCP di C invierà* un riscontro C2 per tali dati e quindi il TCP di S invierà un segmento S2 con il flag FIN a true contenente la seconda parte dei dati del terzo oggetto. Il TCP di C invierà un riscontro C3 di S2 e considererà chiusa la connessione dopo avere atteso 2MSL, ovvero al tempo t + 2 RTT + 2 MSL = t + 3600 msec.



Q3. I pacchetti ARP vengono trasportati nella parte dati dei frame Ethernet specificando ARP nel campo Type del preambolo, se l'host che esegue ARP appartiene a una rete Ethernet. Il viceversa non può invece avvenire dato che ARP è un servizio a livello di rete.

Q4. Sia con K_M il numero di slot da attendere scelto da A dopo avere rilevato una collisione nello slot M e sia R_N il numero scelto da A all'inizio dello slot N per la p-persistenza. A trasmette nello slot S+4 senza rilevare collisioni negli slot [S+1,S+3] solo se (K_S=0 $\wedge$ R_{S+1}>0.75 $\wedge$ R_{S+2}>0.75 $\wedge$ R_{S+3}>0.75 $\wedge$ $R_{S+4}\leq$0.75) oppure (K_S=1 $\wedge$ R_{S+2}>0.75 $\wedge$ R_{S+3}>0.75 $\wedge$ $R_{S+4}\leq$0.75). Quindi A può trasmettere con successo nello slot S+4 con probabilità minore uguale a $\left(\frac{1}{2}\times\frac{1}{4}\times\frac{1}{4}\times\frac{1}{4}\times\frac{3}{4}\right)+\left(\frac{1}{2}\times\frac{1}{4}\times\frac{1}{4}\times\frac{3}{4}\right)=\frac{3}{2^9}+\frac{3}{2^7}=\frac{15}{2^9}$ quindi minore di $\frac{1}{2^4}=\frac{2^5}{2^9}=\frac{32}{2^9}$.

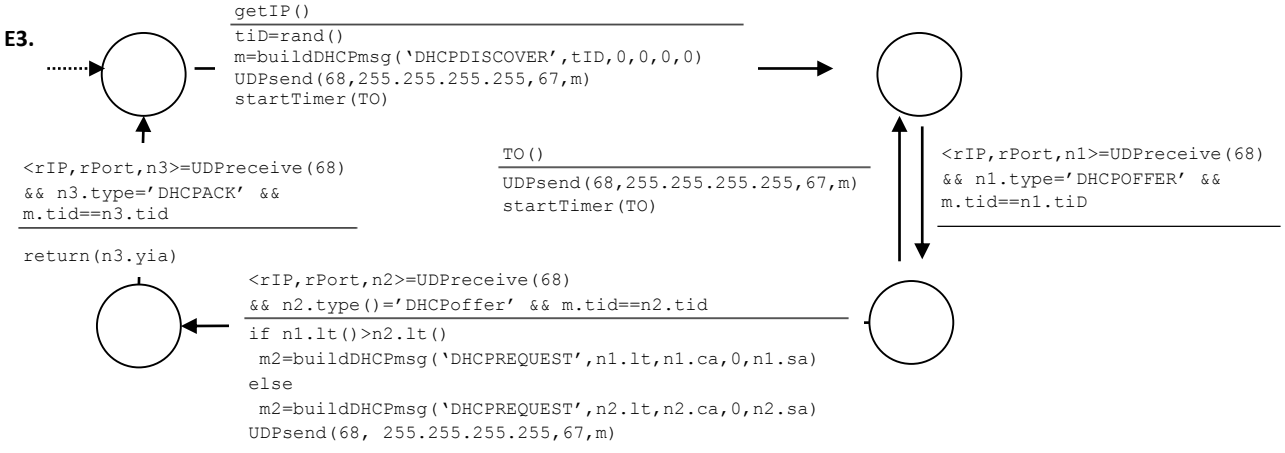
E1.
```
while true do {
   <rIP,rP,m>=UDPreceive(9090);
   if m==GET(f)
      then {
        <rIP2,rP2,m2>=UDPreceive(9090);
        if (rIP2==rIP && rP2==rP && m2==WSIZE(n))
            then {
             fSize=fileSize(f); sent=0;
             while (sent<fSize) do {
               if n>0
                then {
                 m3=buildMessage(sent,f);
                 UDPsend(9090,rIP,rP,m3);
                 n=n-sizeData(m3);
                 sent=sent+sizeData(m3);
                }
                else {
                 <rIP2,rP2,m2>=UDPreceive(9090);
                 if (rIP2==rIP && rP2==rP && m2==WSIZE(n2))
                  then n=n2;
                }
             }
            }
      }
}
```

E2. Osserviamo che in TO il TCP di A si trova nello stato di congestion avoidance dato che $ssthresh_{T0} < cwnd_{T0} < ssthresh_{T0}+3$. Subito dopo avere ricevuto B1, TCP rimane in congestion avoidance e cwnd diventa (6+1/6) MSS = 37/6 MSS. Analizziamo i vari casi possibili:

- Se B1.ackNum=X+2MSS allora, dopo avere ricevuto B1, min(cwnd,B1.rwnd)=1MSS dato che viene spedito 1 solo MSS di nuovi dati, quindi B1.rwnd=1MSS.
  - Se B2.ackNum=X+3MSS allora, dopo avere ricevuto B2, min(cwnd,B2.rwnd)=0 MSS dato che non vengono spediti nuovi dati, quindi B2.rwnd=0MSS e $cwnd_{T3}$ = (37/6+6/37) MSS =1405/222 MSS.
  - Se B2 è un riscontro duplicato il valore di B2.rwnd non è significativo, dato che B2 viene scartato, e $cwnd_{T3}$ = 37/6 MSS.
- Se B1.ackNum=X+1MSS allora, dopo avere ricevuto B1, min(cwnd,B1.rwnd)=2MSS dato che viene spedito 1 solo MSS di nuovi dati, quindi B1.rwnd=2MSS.
  - Se B2.ackNum=X+3MSS allora, dopo avere ricevuto B2, min(cwnd,B2.rwnd)=0 MSS dato che non vengono spediti nuovi dati, quindi B2.rwnd=0MSS e $cwnd_{T3}$ = 1405/222 MSS.
  - Se B2.ackNum=X+2MSS allora, dopo avere ricevuto B2, min(cwnd,B2.rwnd)$\leq$1MSS dato che non vengono spediti nuovi dati, quindi B2.rwnd$\leq$1MSS e $cwnd_{T3}$ = 1405/222 MSS.
  - Se B2 è un riscontro duplicato il valore di B2.rwnd non è significativo, dato che B2 viene scartato, e $cwnd_{T3}$ = 37/6 MSS.

* Suppponiamo per semplicità che il TCP di A non ritardi l'invio dei riscontri quando riceve segmenti "in ordine".

E3.


getIP()
tiD=rand()
m=buildDHCPmsg('DHCPDISCOVER',tID,0,0,0,0)
UDPsend(68,255.255.255.255,67,m)
startTimer(TO)
<rIP,rPort,n3>=UDPreceive(68)
&& n3.type='DHCPACK' &&
m.tid==n3.tid
return(n3.yia)
TO()
UDPsend(68,255.255.255.255,67,m)
startTimer(TO)
<rIP,rPort,n1>=UDPreceive(68)
&& n1.type='DHCPOFFER' &&
m.tid==n1.tiD
<rIP,rPort,n2>=UDPreceive(68)
&& n2.type()='DHCPoffer' && m.tid==n2.tid
if n1.lt()>n2.lt()
m2=buildDHCPmsg('DHCPREQUEST',n1.lt,n1.ca,0,n1.sa)
else
m2=buildDHCPmsg('DHCPREQUEST',n2.lt,n2.ca,0,n2.sa)
UDPsend(68, 255.255.255.255,67,m)