RETI DI CALCOLATORI – prova scritta del 15/07/2014

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

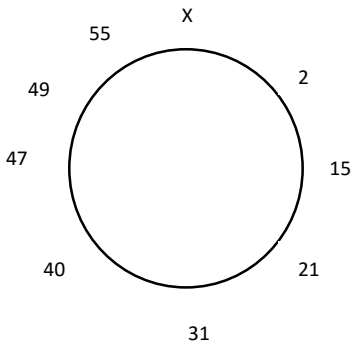
Prima parte (10 punti)

Q1. Un router A trasmette in modo continuato dati su un collegamento lungo 10 km con un router B, la cui velocità di propagazione è $2 \cdot 10^8$m/s. Determinare –giustificando la risposta– quale è la frequenza di trasmissione se al più 500 bit possono essere simultaneamente presenti nel collegamento.

Q2. Indicare –giustificando la risposta– quanti sono i byte di dati trasportati da un segmento UDP la cui intestazione (in esadecimale) è 0632000D002CE217.

Q3. Un router IPv4 R deve inoltrare su un collegamento con MTU a 500 byte un datagram IP che ha ricevuto, la cui intestazione non contiene opzioni e che contiene un segmento TCP di 980 byte. Indicare –giustificando la risposta– i valori dei campi *offset* e *length* dei frammenti inviati da R.

Q4. Considerare l'anello Chord a lato, che utilizza identificatori a 6 bit ed è formato dai nodi con identificatori 2, 15, 21, 31, 40, 47, 49, 55 e X, con X≠55 e X≠2. Indicare –giustificando la risposta– a quale nodo il nodo 47 inoltra la query relativa all'identificatore 1, a seconda del valore di X.



Seconda parte

E1 (5 punti). Descrivere <u>con un automa a stati finiti</u> il comportamento di un semplice server FTP che consente ai suoi clienti di eseguire solo i comandi LIST e QUIT dopo avere effettuato un accesso autenticato. Il server consente inoltre di inviare al più tre volte la password per ogni tentativo di login. Nella descrizione dell'automa utilizzare gli eventi

USER u *PASS p* *LIST* *QUIT*

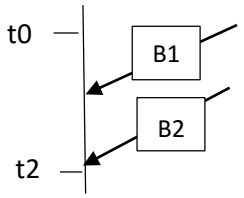
per indicare le richieste dei clienti.

E2 (5 punti). Descrivere <u>con uno pseudo-codice</u> il modo con cui un name server locale riceve e risolve una query di tipo A. Supporre di avere a disposizione le operazioni:

```
UDPsend(localPort, remoteIP, remotePort, DNSmessage)
<remoteIP, remotePort, DNSmessage> UDPreceive(localPort)
DNSmessage lookUpCache (DNSmessage)  //per cercare in cache il risultato di una query
storeInCache (DNSmessage)            //per aggiornare la cache
```

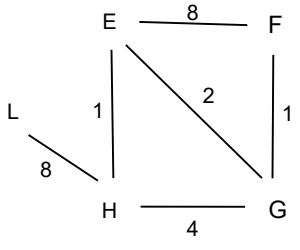
Per semplicità supporre che il name server risolva una sola richiesta alla volta, nessun pacchetto IP vada perso e ogni risposta DNS consista di una sola tupla `<name,value,type>` nella sezione "Answer" e di al più una tupla `<name,value,type>` nella sezione "Additional".

E3 (6 punti). Al tempo t0 il TCP di un processo applicativo A si trova nello stato di *slow start*, ha due segmenti *full-sized* in volo, il più vecchio dei quali ha numero di sequenza X, il valore di *ssthresh* è 3 MSS e il valore di *cwnd* è C MSS. Indicare –giustificando la risposta– quanti nuovi dati potrebbe spedire il TCP di A dopo avere ricevuto B2, nell'ipotesi che B2 sia un riscontro non duplicato e che il valore di *rwnd* contenuto in B2 sia (C+1,6)MSS.



E4 (4 punti). Considerare la rete a lato i cui router utilizzano *distance vector* con *poisoned reverse*. Indicare il contenuto del vettore delle distanze calcolato da E, (delle ultime copie) dei vettori che E ha ricevuto dai suoi vicini e della tabella *nextHop* del router E:

(a) una volta che la rete ha raggiunto lo stato di quiescenza al tempo t e
(b) se, subito dopo t, E determina che il collegamento EH è caduto.



Traccia della soluzione

Q1. I bit che possono essere simultaneamente presenti nel collegamento sono quelli che il router riesce a trasmettere mentre il segnale si propaga nel canale, ovvero $R \cdot d_{prop}$ = 500 b e quindi $R = 500\ b \cdot \frac{2 \cdot 10^8 m/s}{10^4 m}$ =10 Mbps.

Q2. La lunghezza totale del segmento UDP è definita nel terzo byte dell'intestazione, ovvero dal terzo gruppo di quattro cifre esadecimali (002C), ed è quindi di 44 byte. I dati trasportati dal segmento sono 36 byte, ovvero la differenza tra la lunghezza totale del segmento e la lunghezza dell'intestazione.

Q3. Per inoltrare il datagram di 1000 byte ricevuto (980 byte di dati e 20 byte di intestazione). R dovrà creare 3 frammenti: il primo con *offset*=0 e *length*=500, il secondo con *offset*=60 e *length*=500, il terzo con *offset*=120 e *length*=40.

Q4. Analizziamo i vari casi possibili:

(a) se 55<X<63 i finger del nodo 47 sono

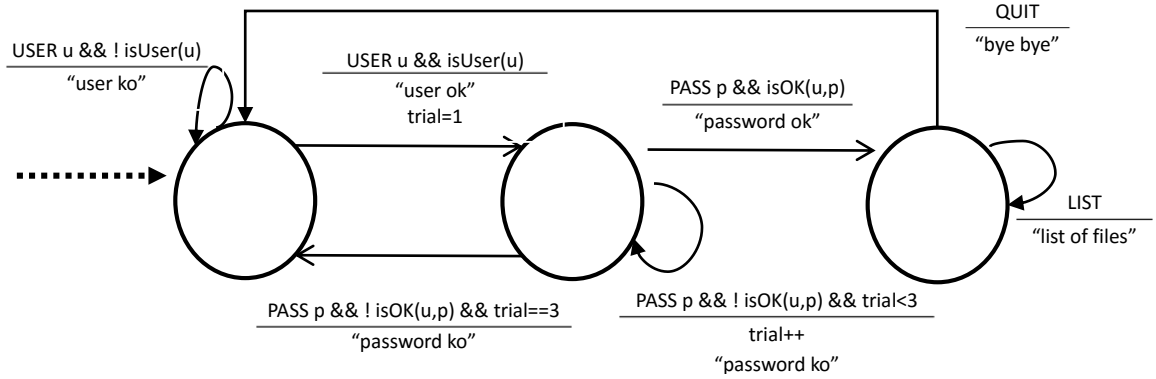
(b) se X=63 o X=0 i finger del nodo 47 sono

(c) se X=1 i finger del nodo 47 sono

49	49	55	55	2	15
49	49	55	55	X	15
49	49	55	55	X	15

Il nodo 47 inoltra quindi la query relativa all'identificatore 1 al nodo 55 nei casi (a) e (c), ovvero se 55<X<63 o X=1.
Il nodo 47 inoltra invece la query relativa all'identificatore 1 al nodo X nel caso (b), ovvero se X=63 o X=0.

E1. Indichiamo con isUser(u) la funzione che verifica se u è un utente del server e con isOK(u,p) la funzione che verifica se p è la password di u. Per semplicità indichiamo con "descrizione messaggio" ogni messaggio inviato dal server al cliente.



E2.
```
/* Indichiamo con m.answer() e m.additional() le tuple contenute nella sezione "Answer" e
"Additional" di un messaggio DNS. */

nsIP = NULL;      //IP server a cui inoltrare q
<rIP, rPort, q> = UDPreceive(53);
iterate = 1;
m = lookUpCache(q);
if m==NULL
   then
        nsIP = "IP server a cui inoltrare q";
   else {
         <n,v,t> = m.answer();
         if t == "NS"
            then {
                  <n2,v2,t2> = m.additional();
                  nsIP = v2;
                 }
            else if t == "A"
                    then {
                          UDPsend(53, rIP, rPort, m);
                          iterate = 0;
                          }
        }
if iterate==1
   then {
         do {
             UDPsend(53, nsIP, 53, q);
             <ip,p,m> = UDPreceive(53);
             storeInCache(m);
             <n,v,t> = m.answer();
             if t="NS"
                then {
                      <n2,v2,t2> = m.additional();
                      nsIP = v2;
                     }
             }
         while t!= "A";
         UDPsend(53, rIP, rPort, m);
        }
```

E3. Distinguiamo i vari casi possibili*:

(1) Se B1.ackN=X

(1.a) se B2.ackN=X+1MSS allora $cwnd_{t2}$=(C+1)MSS e quindi in t2 TCP potrebbe spedire al più min((C+1)MSS,(C+1,6)MSS)-1MSS = C MSS di nuovi dati.

(1.b) se B2.ackM=X+2MSS allora $cwnd_{t2}$=(C+1)MSS e quindi in t2 TCP potrebbe spedire al più min((C+1)MSS,(C+1,6)MSS) = (C+1)MSS di nuovi dati.

(2) Se B1.ackN=X+1 MSS allora B2.ackN=X+2 MSS.

(2.a) se (C+1)MSS≥3MSS allora $cwnd_{t2}$=(C+1+$\frac{1}{C+1}$)MSS e quindi in t2 TCP potrebbe spedire al più min((C+1+$\frac{1}{C+1}$)MSS,(C+1,6)MSS) = (C+1+$\frac{1}{C+1}$)MSS dato che C≥1MSS.

(2.b) se (C+1)MSS<3MSS allora $cwnd_{t2}$=(C+2)MSS e quindi in t2 TCP potrebbe spedire al più min((C+2)MSS,(C+1,6)MSS) = (C+1,6)MSS di nuovi dati.

* Se consideriamo anche la possibilità che scatti un timeout subito dopo la ricezione di B2 allora il TCP di A potrebbe spedire 1 MSS di nuovi dati se B2.ackN=X+2MSS, nessun nuovo dato se B2.ackN=X+1MSS.

E4.

(a)

	D_E	D_E	D_G	D_H
F	3	/	1	Inf
G	2	1	/	Inf
H	1	4	Inf	/
L	9	12	Inf	8

dest	*nextHop*
F	G
G	G
H	H
L	H

(b)

	D_E	D_E	D_G	D_H
F	3	/	1	Inf
G	2	1	/	Inf
H	12	4	Inf	/
L	20	12	Inf	8

dest	*nextHop*
F	G
G	G
H	F
L	F