RETI DI CALCOLATORI – prova scritta del 30/03/2016

Per essere ammessi alla prova orale è necessario ottenere una valutazione sufficiente della prima parte.

Prima parte (15 punti)

Q1. Un router A trasmette un frame F a un router B su un collegamento diretto il cui ritardo di propagazione è di 4 millisecondi. Indicare – giustificando la risposta – quale è la frequenza di trasmissione del collegamento se la trasmissione del terzo bit di F termina esattamente quando il primo bit di F raggiunge B.

Q2. Indicare – giustificando la risposta – quali sono le chiamate (parametri inclusi) alle API del servizio di livello trasporto eseguite da un server HTTP per ricevere una richiesta da un cliente C da cui non ha ricevuto precedentemente alcuna altra richiesta, nell'ipotesi che C si trovi sotto NAT.

Q3. Indicare – giustificando la risposta – quanti nuovi segmenti può inviare un sender Go-Back-N subito dopo avere ricevuto un riscontro R, se subito prima di ricevere R ha N/2 segmenti spediti e non ancora riscontrati, dove N è la dimensione della finestra.

Q4. Indicare – giustificando la risposta – quali dei seguenti indirizzi IP appartengono alla rete 10.9.7.0/14:

(a) 10.10.7.0
(b) 10.11.7.0
(c) 10.12.7.0
(d) 10.13.7.0
(e) 10.255.7.0

Q5. Indicare a quale valore viene inizializzato il timer NAV di una stazione IEEE 802.11 che riceve un frame RTS.

Seconda parte (15 punti)

E1. Al tempo t0 il TCP di un client SMTP A ha una connessione già stabilita, per la quale:

- ha 3 segmenti full-sized in volo, con S_f=X,
- 3 MSS di nuovi dati da spedire,
- ha come valori delle variabili *cwnd* e *rwnd* 3 MSS e della variabile *ssthresh* 4 MSS, e
- si trova nello stato di *slow start*.

Al tempo t1 il TCP di A riceve un riscontro in conseguenza del quale non invia nuovi dati, e la stessa cosa avviene al tempo t2. Al tempo t3 riceve un riscontro R non duplicato, il cui campo *rwnd* ha valore 2MSS, e subito dopo avere ricevuto R il TCP di A invia al tempo t4 due segmenti, contenenti 1 MSS e 0,5 MSS di dati rispettivamente. Indicare – giustificando la risposta – lo stato e il valore di *cwnd* del TCP di A al tempo t4 e il valore del campo *ackNum* in R.
Supporre che nell'intervallo [t0,t4] non scada alcun timeout.

E2. Consideriamo una versione del protocollo OSPF in cui:

Fase 1 - Quando un nodo viene attivato invia un messaggio OSPF di tipo "hello" a tutti i suoi vicini e quindi attende, per al più tx secondi, di ricevere una risposta di tipo "database description" per inizializzare il proprio LSDB. Se dopo tx secondi non ha ricevuto alcuna risposta di tipo "database description" il nodo invia di nuovo il messaggio "hello".
Fase 2 – Non appena riceve una risposta di tipo "database description", il nodo passa alla fase 2, in cui invia ogni tp secondi un messaggio OSPF di tipo "link-state request" a ciascun suo vicino per continuare ad aggiornare il proprio LSDB nel tempo.

Descrivere, <u>utilizzando un automa a stati finiti</u>, il comportamento sopra descritto di un nodo R, assumendo di disporre dei comandi

```
receive()                    // per ricevere un messaggio,
send_Hello(recipient)        // per inviare un messaggio di tipo "hello",
send_LSrequest(recipient)    // per inviare un messaggio di tipo "link-state request",
startTimer(duration)         // per avviare un timer
```

e assumendo che R memorizzi gli indirizzi dei suoi vicini in un array N[1]...N[V].
Per semplicità supporre che R non riceva messaggi di tipo "hello" né di tipo "link-state request".

Traccia della soluzione

Q1. Sappiamo che $\frac{3\,b}{R} = \frac{1\,b}{R} + \frac{4}{10^3}s$ quindi R=500 b/s.

Q2. Il server HTTP (dopo avere invocato `TCPbind(80)`) dovrà invocare `TCPaccept(80)` per accettare la richiesta di connessione e quindi `TCPreceive(c)` sulla connessione `c` stabilita[1].

Q3. Subito dopo aver ricevuto R il sender potrà inviare N/2+X nuovi segmenti, dove X∈[0,N/2] è il numero[2] di segmenti in volo riscontrati da R.

Q4. Un indirizzo IP appartiene alla rete 10.9.7.0/14 se i suoi 14 bit più significativi coincidono con il prefisso della rete. Quindi solo (a) e (b) appartengono alla rete 10.9.7.0/14:

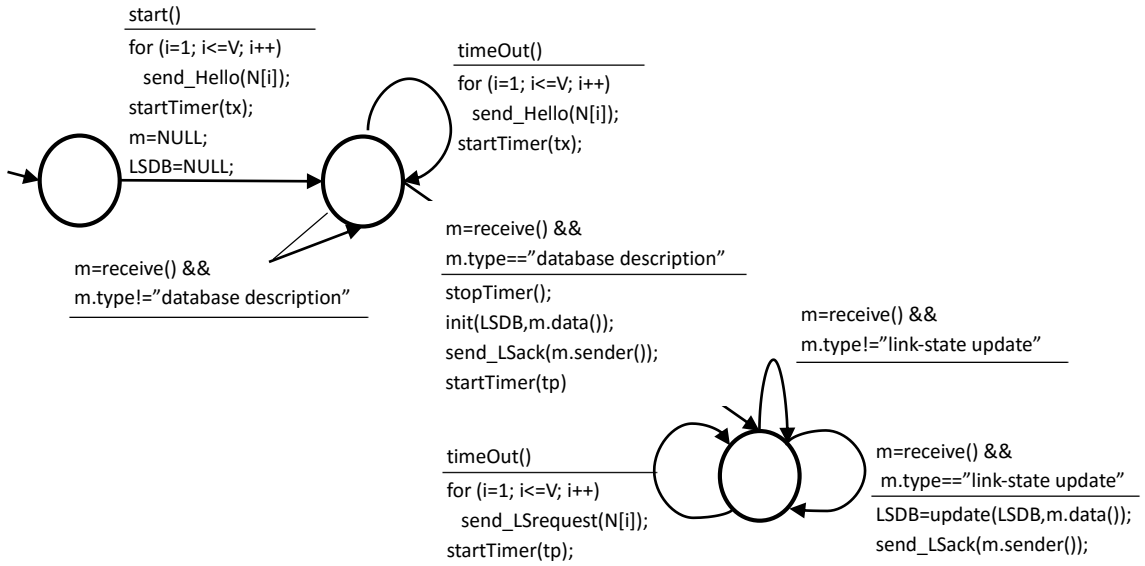
rete *00001010 . 00001001 . 00000111 . 00000000*
(a) *00001010 . 00001010 . 00000111 . 00000000*
(b) *00001010 . 00001011 . 00000111 . 00000000*
(c) *00001010 . 00001100 . 00000111 . 00000000*
(d) *00001010 . 00001101 . 00000111 . 00000000*
(e) *00001010 . **111111**11 . 00000111 . 00000000*

Q5. Al valore contenuto nel campo D (*Duration*) del frame RTS ricevuto, con cui la stazione che ha inviato il frame RTS indica il tempo in cui occuperà il canale per completare la trasmissione del frame di dati e ricevere il relativo riscontro.

E1. Siano R1 e R2 i riscontri ricevuti in t1 e t2 e analizziamo i vari casi possibili.

- R1 non può essere un riscontro non duplicato. (Se lo fosse, dopo averlo ricevuto TCP passerebbe in *congestion avoidance* con *cwnd*=4 MSS e, dopo avere ricevuto R2 e R, *cwnd*≥4 MSS e quindi in t4 min(*cwnd*,*rwnd*)= 2 MSS e TCP non invierebbe in alcun caso solo 1,5 MSS di nuovi dati.)
- Se R1 è un riscontro duplicato ricevuto per la terza volta allora dopo averlo ricevuto TCP passa in *fast recovery* con *ssthresh*=1,5 MSS e *cwnd*=4,5 MSS. A questo punto:
  - Se R2 è un riscontro duplicato ricevuto non per la terza volta, allora dopo averlo ricevuto TCP rimane *in fast recovery* e *cwnd* diventa 5,5 MSS. Dopo avere ricevuto R TCP passa quindi in *congestion avoidance*, *cwnd* diventa 1,5 MSS e R.ackNum deve quindi avere come valore X+3 MSS.
  - R2 non può essere un riscontro non duplicato. (Se lo fosse, allora dopo averlo ricevuto TCP passerebbe in *congestion avoidance* con *cwnd*=1,5 MSS e, dopo avere ricevuto R, *cwnd* assumerebbe un valore maggiore di 1,5 MSS e TCP non invierebbe in alcun caso solo 1,5 MSS di nuovi dati.)
- Se R1 è un riscontro duplicato ricevuto non per la terza volta:
  - R2 non può essere un riscontro duplicato ricevuto non per la terza volta. (Se lo fosse allora dopo avere ricevuto R *cwnd*=4 MSS e min(*cwnd*,*rwnd*)= 2 MSS e quindi TCP non invierebbe solo 1,5 MSS di nuovi dati.)
  - Se R2 è un riscontro duplicato ricevuto per la terza volta, allora dopo averlo ricevuto TCP passa in *fast recovery* con *ssthresh*=1,5 MSS e *cwnd*=4,5 MSS. Quindi, dopo avere ricevuto R, TCP passa in *congestion avoidance* con *cwnd*=1,5 MSS e R.ackNum deve quindi avere valore X+3 MSS.
  - R2 non può essere un riscontro non duplicato. (Se lo fosse allora dopo avere ricevuto R2 il TCP passerebbe in *congestion avoidance* con *cwnd*=4 MSS e dopo avere ricevuto R *cwnd*>4 MSS e min(*cwnd*,*rwnd*)= 2 MSS e quindi TCP non invierebbe solo 1,5 MSS di nuovi dati.)

E2.



[1] In Java, per esempio:

```
ServerSocket ss = new ServerSocket(80);
Socket c = ss.accept();
Scanner input = new Scanner(c.getInputStream());
String request = input.nextLine();
```

[2] X=R.ackNum-base se R.ackNum≥base, X= R.ackNum+buffer.size()-base altrimenti.