RETI DI CALCOLATORI – Prima verifica in itinere, a.a. 2009/2010

La prova di verifica è strutturata in due parti: una prima parte formata da semplici quesiti e una seconda parte formata da esercizi. Per ottenere una valutazione sufficiente dell'intera prova è necessario ottenere una valutazione sufficiente della prima parte. Gli elaborati contenenti le risposte ai quesiti della prima parte devono essere consegnati entro i primi 50 minuti della prova.

*Prima parte (**10 punti**)*

Q1. Consideriamo un router R che ha due soli collegamenti, uno con un router A e uno con un router B. Supponiamo che R riceva da A un blocco di N pacchetti, tutti di L bit, e supponiamo che quando R riceve il primo pacchetto di tale blocco la coda che precede il collegamento con B sia vuota e R non stia trasmettendo dati su quel collegamento. Supponendo che la coda che precede il collegamento con B possa contenere N·L bit, determinare – giustificando la risposta – il ritardo medio di accodamento degli N pacchetti del blocco.

Q2. Supponiamo che un server FTP in esecuzione su un host B riceva da un cliente in esecuzione su un host A un segmento S con `sourcePort`=U, `destinationPort`=L, `seqNumber`=X, `ackNumber`=Y, flag `ack` a 1 e come dati la stringa "`RETR v1.PDF`". Indicare i valori dei campi `sourcePort`, `destinationPort`, `seqNumber`, `ackNumber`, eventuali flag a 1 e contenuto della parte dati dei primi due segmenti che B invia ad A dopo avere ricevuto S.

Q3. Supponiamo che un processo applicativo effettui tre chiamate al servizio di trasporto per inviare tre diversi insiemi di dati a un suo pari. Supponiamo che il servizio di trasporto sia realizzato utilizzando una variante V del protocollo *alternating bit* ("ad alternanza di bit") in cui il mittente rispedisce l'ultimo segmento inviato anche quando riceve un riscontro negativo. Assumendo per semplicità che tutti i segmenti arrivino non corrotti a destinazione, che il tempo che intercorre tra l'invio di un pacchetto e la ricezione del pacchetto inviato in risposta ad esso sia costante, e supponendo che la lunghezza dell'intervallo di timeout sia 2/3 RTT per il primo segmento S0, 5/6 RTT per il secondo segmento S1 e 11/12 RTT per il terzo segmento S2, determinare – giustificando la risposta – quante volte V rispedisce i segmenti S0, S1 e S2 in più del protocollo *alternating bit* standard.

Q4. Consideriamo una connessione S stabilita fra due host A e B. Determinare – giustificando la risposta – se è possibile che il TCP di A invii un segmento FIN (relativo a S) e riceva come risposta dal TCP di B un segmento FIN (anch'esso relativo a S) senza che nessun processo applicativo in esecuzione su B abbia richiesto a TCP di chiudere la connessione S.

RETI DI CALCOLATORI – Prima verifica in itinere, a.a. 2009/2010

Seconda parte

E1 (8 punti). (a) Consideriamo un server S che implementa un contatore condiviso. Supponiamo che S serva un solo cliente per volta e che ogni cliente, dopo avere stabilito una connessione TCP con S, possa inviare a S (un numero arbitrario di) richieste di tipo "READ" e "WRITE". Supponiamo che per gestire le richieste dei clienti il server adotti una politica di diritti di accesso basata sugli indirizzi IP dei clienti. Supponiamo per questo che il servizio DNS sia esteso in modo da trattare tuple del tipo <*name*, *value*, "AUTH"> dove *value* indica i diritti associati all'indirizzo IP *name*. Descrivere con un automa a stati finiti il comportamento del server S in modo che: (1) se S riceve una richiesta di tipo"READ", S risponda inviando il valore corrente del contatore; (2) se S riceve una richiesta di tipo "WRITE" e il cliente ha diritti "high", S aggiorni il contatore al valore indicato dal cliente e restituisca la stringa "OK" al cliente; (3) se S riceve una richiesta di tipo "WRITE" e il cliente ha invece diritti "low", S non modifichi il valore del contatore e restituisca la stringa "Not enough rights" al cliente; (4) se S riceve una richiesta di tipo "QUIT" risponda con il messaggio "BYE" e chiuda la connessione.

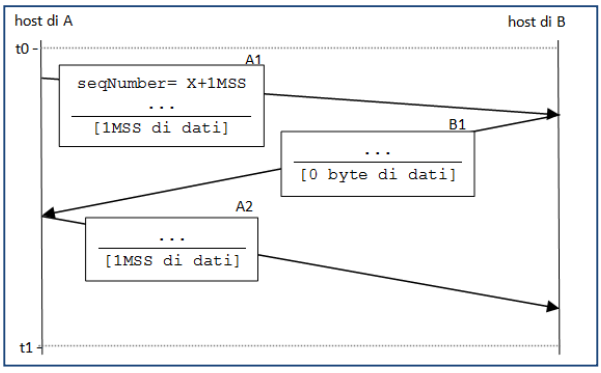
Per descrivere le *azioni* eseguite nelle transizioni dell'automa utilizzare le operazioni `requestPort(int)` per richiedere l'assegnazione di una porta (`requestPort` restituisce 1 se la porta è stata assegnata, 0 altrimenti); `accept(int)` per attendere di ricevere una richiesta di connessione (una volta che la connessione è stata creata, `accept` restituisce un oggetto di tipo `connection`); `send(connection,string)` e `receive(connection)` per inviare e ricevere messaggi su una connessione TCP; `getRemoteIP(connection)` per ottenere l'`IPaddress` dell'host remoto di una connessione; `close(connection)` per chiudere una connessione; `UDPsend(IPaddress,<QNAME,QTYPE>)` per spedire una query DNS con UDP e `UDPreceive()` per ricevere una risposta DNS <*name*,*value*,*type*>. Per semplicità si trascuri la possibilità che i segmenti UDP possano venire persi o corrotti e si assuma che S possa ricevere solo risposte DNS di tipo "AUTH" oppure "NS" e che le risposte di tipo "NS" contengano direttamente l'indirizzo IP (anziché il nome) del server da contattare.

Per descrivere le *condizioni* che definiscono l'eseguibilità delle transizioni dell'automa utilizzare soltanto (oltre agli operatori logici e di confronto) l'evento `startServer()` con cui un processo applicativo avvia il server e le operazioni `reqType(string)` e `reqValue(string)` con cui estrarre rispettivamente tipo ed eventuale valore da una richiesta del cliente.

(b) Descrivere in che modo l'interazione tra cliente e server dell'esercizio precedente può essere implementata utilizzando il protocollo HTTP.

E2 (4 punti). Supponendo di avere a disposizione le operazioni `open(IPAddress,int)`, `send(connection,string)` e `receive(connection)` per aprire una connessione e inviare e ricevere messaggi su una connessione, scrivere uno pseudo-codice che descriva come un cliente POP3 completa la fase di autorizzazione.

E3 (8 punti). Consideriamo la figura a lato in cui un processo applicativo A ha già stabilito una connessione TCP con un suo pari B. Supponiamo che nell'intervallo [t0,t1] non scatti nessun timeout (né in A né in B), che tutti i segmenti indicati nella figura arrivino a destinazione non corrotti e che nessun altro segmento venga spedito o ricevuto nell'intervallo [t0,t1]. Supponiamo inoltre che in t0 sia il TCP di A che quello di B abbiano ciascuno 1MSS di dati "in volo" e si trovino entrambi nello stato di *slow start* e che sempre in t0 il valore di `CongWin` sia 2MSS per il TCP di A e 1MSS per il TCP di B, il valore di `sendBase` sia X per il TCP di A e Y per il TCP di B e che il TCP di A abbia ancora 3MSS di nuovi dati da spedire e che il TCP di B abbia 2MSS di nuovi dati da spedire. Determinare – giustificando la risposta – i valori dei campi `seqNumber`, `ackNumber`, `RcvWin` e i flag a 1 dei segmenti A1, B1 e A2.



Traccia della soluzione

Q1. Se R è la frequenza di trasmissione del collegamento con B, il ritardo di accodamento dell'i-esimo pacchetto del blocco sarà (i-1)L/R. Pertanto il ritardo medio di accodamento degli N pacchetti del blocco sarà:

$$\frac{\sum_{i=0}^{N-1} i\frac{L}{R}}{N} = \frac{(N-1)NL}{2NR} = \frac{(N-1)L}{2R}.$$

Q2. Il primo segmento sarà un riscontro di S e quindi conterrà: `sourcePort`=L, `destinationPort`=U, `seqNumber`=Y[1], flag `ack` a 1 e `ackNumber`=X+Δ, dove Δ è il numero di byte corrispondenti alla stringa "`RETR v1.PDF`". La parte dati del primo segmento conterrà inoltre *"piggybacked"* la risposta FTP inviata dal server in risposta al comando ricevuto.

- Se il server deve inviare il file richiesto[2] il secondo segmento sarà un "SYN" inviato dall'host di B per aprire la connessione dati e quindi conterrà: `sourcePort`=L-1, `destinationPort`=U, `seqNumber`=Z (dove Z è il numero di sequenza scelto dall'host di B per la nuova connessione), flag `syn` a 1 e nessun dato.
- Se invece non deve inviare il file, il server non dovrà iniziare l'handshaking per aprire la connessione dati e spedirà quindi il successivo segmento solo allo scadere del timeout oppure quando riceverà un nuovo segmento da B.

Q3. A causa dei timeout prematuri il protocollo alternating bit rispedirà 1 volta sia S0 che S1 che S2. Anche la variante V rispedirà 1 volta S0 (quando scatta prematuramente il timeout), 1 volta S1 (quando riceve l'ACK duplicato di S0) e 1 volta S2 (quando riceve l'ACK duplicato di S1)[3].

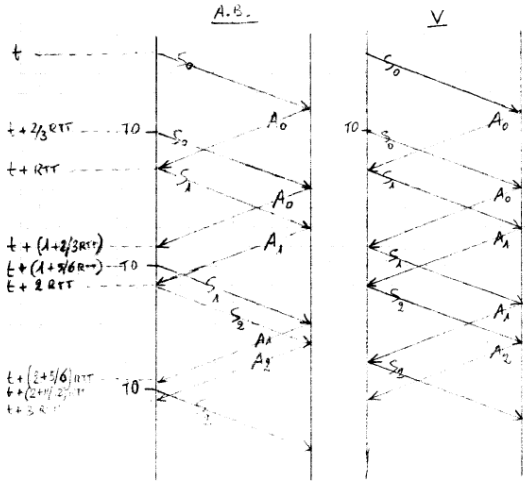
Q4. No, non è possibile che il TCP di B abbia inviato un segmento FIN senza che nessun processo applicativo in esecuzione su B abbia richiesto di chiudere la connessione. Come specificato nella RFC 793*****, un TCP invia infatti un segmento FIN solo quando il "suo" processo applicativo richiede di chiudere la connessione.

> ***Dato che la descrizione fornita nel testo adottato non definisce con chiarezza questa specifica del protocollo né permette di evincerla ragionando, e dato che neppure la RFC 2581 (la cui lettura è stata suggerita come approfondimento) fa riferimento a tale specifica, ai fini della valutazione della prova in itinere tutte le risposte fornite a questo quesito sono state considerate corrette.***

[1] Assumendo che Y-1 sia l'ultimo byte di dati che B ha spedito sulla connessione di controllo.

[2] Ovvero se il server considera valida la richiesta del cliente, possiede il file richiesto e i cliente ha i diritti per scaricarlo.

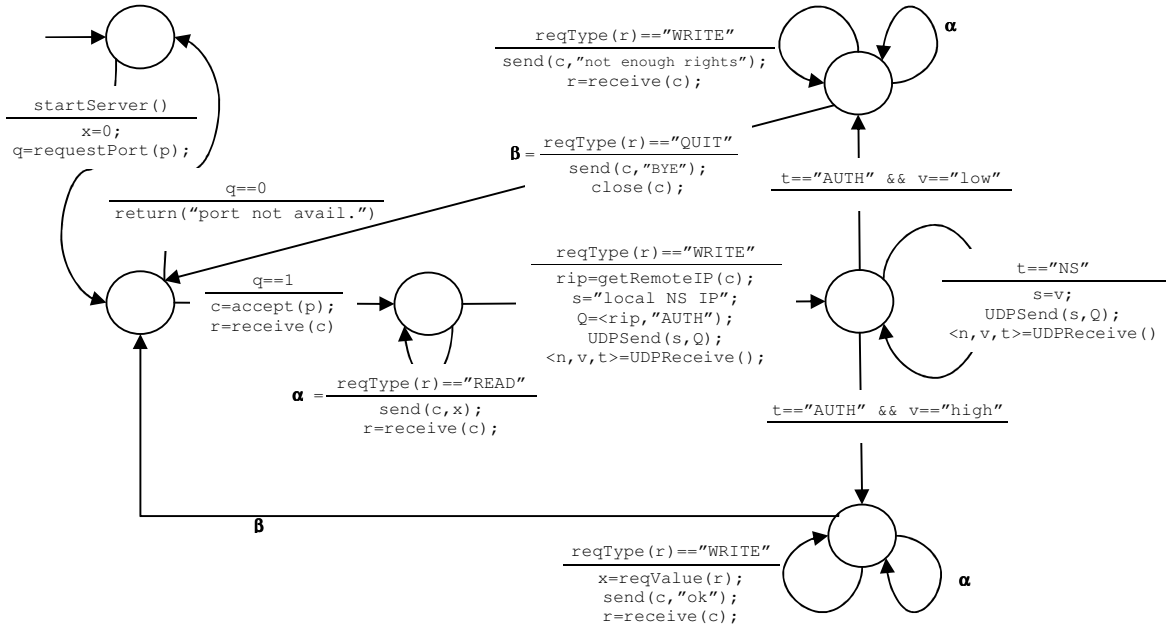
[3] Il comportamento dei protocolli può essere cosi schematizzato:



ovvero:

tempo	*alternating bit*	*variante V*
t	spedisce S0, avvia timer(S0)	spedisce S0, avvia timer(S0)
t + 2/3 RTT	timeout(S0), **rispedisce S0**, avvia timer(S0)	timeout(S0), **rispedisce S0**, avvia timer(S(
t + 1 RTT	riceve A0, ferma timer(S0), invia S1, avvia timer(S1)	riceve A0, ferma timer(S0), invia S1, avvia timer(S1)
t + (1+2/3) RTT	riceve duplicato di A0	riceve duplicato di A0, **rispedisce S1**, avvia timer (S
t + (1+5/6) RTT	timeout(S1), **rispedisce S1**, avvia timer (S1)	
t + 2 RTT	riceve A1, ferma timer(S1), invia S2, avvia timer(S2)	riceve A1, ferma timer(S1), invia S2, avvia timer(S2)
t + (2+2/3) RTT		riceve duplicato di A1, **rispedisce S2**, avvia timer (S
t + (2+5/6) RTT	riceve duplicato di A1	
t + (2+11/12) RTT	timout(S2), **rispedisce S2**, avvia timer(S2)	
t + 3 RTT	riceve A2, ferma timer(S2)	riceve A2, ferma timer(S2)

E1. (a) Assumiamo che il contatore (x) sia inizializzato a 0. Sia p la porta utilizzata dal server S.



(b) Le richieste "READ" e "WRITE" dei clienti possono essere inviate ad esempio mediante il metodo POST di HTTP (inserendo il tipo della richiesta e l'eventuale valore nel "corpo dell'entità"). Le risposte del server possono essere inviate come messaggi di risposta HTTP.

E2.

```
c = open(msIP,110);                                  //msIP = indirizzo IP del mailserver dell'utente
x = receive(c);                                      //si aspetta di ricevere "+OK POP3 server ready"
if (x != "+OK POP3 server ready")
        return ("server not available");
   else { send(C,"user "+USR);                       //USR = username dell'utente
          x = receive(c);                            //si aspetta di ricevere "+OK"
          if (x != "+ok")
                  return("invalid user name");
              else {  send(c,"pass "+PWD);           //PWD = password dell'utente
                      x = receive(c);                //si aspetta di ricevere "+OK"
                      if (x != "+ok")
                            return("invalid password");
                       else ...
                   }
        }
```

E3. Se A1 contenesse un riscontro "positivo" (per l'host di B) allora A1 dovrebbe contenere A1.ackNumber=Y+1MSS e dovrebbe avere il flag ACK a 1. Ricevuto A1, l'host di B incrementerebbe il valore della sua CongWin e il controllo di congestione gli permetterebbe di inviare 2MSS di nuovi dati. Dato che però B1 non contiene dati, l'host di B dovrebbe essere stato "inibito" dal controllo di flusso, ovvero A1.RcvWin dovrebbe essere 0. Ma se così fosse B1 dovrebbe contenere 1 byte di dati (per il meccanismo di *"zero-window probing"*) e possiamo quindi concludere che A1 *non* può contenere un riscontro positivo[4]. A1 deve quindi contenere un riscontro "negativo" (A1.ackNumber ≤ Y e flag di ACK a 1) oppure non contenere un riscontro (flag di ACK a 0). In entrambi i casi, il meccanismo di controllo di congestione non permette all'host di B di inviare nuovi dati in B1 e non possiamo quindi desumere niente sul valore di A1.RcvWin.
Il segmento B1 sarà un segmento di riscontro, ovvero avrà il flag ACK a 1 (e B1.seqNumber=Y+1MSS). Per determinare i valori dei campi ackNumber e RcvWin di B1 dobbiamo distinguere quattro casi:

(a) se B1.ackNumber=X+2MSS allora in seguito alla ricezione di B1, l'host di A incrementa la sua CongWin a 3 MSS e potrebbe quindi spedire i 2 MSS di nuovi dati che ancora deve spedire. Dato però che l'host di A spedisce solo 1 MSS di dati in A2, esso deve essere stato inibito dal controllo di flusso, ovvero B1.RcvWin=1MSS. In questo caso il segmento A2 conterrà A2.seqNumber=X+2MSS;

[4] Nel caso in cui l'implementazione del TCP dell'host di B adottasse la raccomandazione contenuta nella RFC 1122 di inviare la prima *zero-window probe* solo dopo che la finestra è rimasta a zero per un intero periodo di timeout, allora A1 potrebbe contenere un riscontro positivo ovvero A1 potrebbe contenere ackNumber=Y+1MSS, flag di ACK a 1 e RcvWin=0 (impedendo così all'host di B di spedire nuovi dati quand riceverà A1).

(b) se invece B1.ackNumber=X+1MSS allora – per lo stesso ragionamento descritto nel punto precedente - B1.RcvWin=2MSS e di nuovo A2 conterrà A2.seqNumber=X+2MSS;
(c) se B1.ackNumber=X e non si tratta del terzo riscontro duplicato per X ricevuto dall'host di A, allora B1.RcvWin=3MSS e di nuovo A2 conterrà A2.seqNumber=X+2MSS;
(d) se B1.ackNumber=X e B1 è il terzo riscontro duplicato per X ricevuto dall'host di A, allora A2 sarà una ritrasmissione veloce del segmento più vecchio ancora "in volo" per l'host di A, ovvero A2 conterrà A2.seqNumber=X.