RETI DI CALCOLATORI – Sesto appello a.a. 2008/09

E1 [6 punti]. Descrivere con un automa a stati finiti il comportamento di un server che offre la possibilità ai clienti di intercambiare aforismi. Il server attende di ricevere richieste di connessioni TCP sulla porta 212121 e permette a una coppia di clienti simultaneamente collegati di inviare e di ricevere un aforisma ciascuno. Una volta che lo scambio dei due aforismi è avvenuto, il server torna ad attendere di ricevere nuove richieste di connessioni. Assumiamo che ciascun cliente cerchi di inviare e di ricevere solo un aforisma, mentre non sappiamo quale cliente invierà per primo il suo aforisma. Per la descrizione dell'automa utilizzare gli eventi:

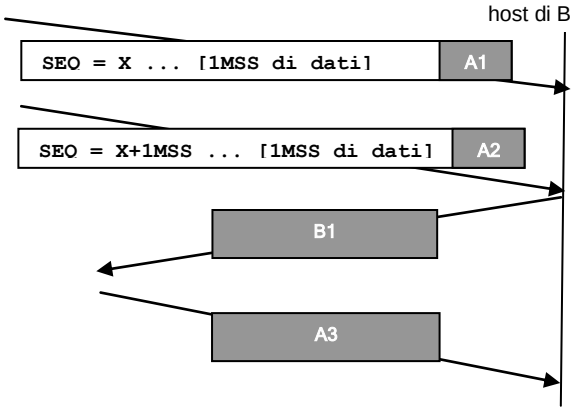
```
c=accept(212121)        //per indicare l'accettazione di una richiesta di connessione
s=receive(c)            //per indicare la ricezione di un messaggio su una connessione
```

e le azioni:

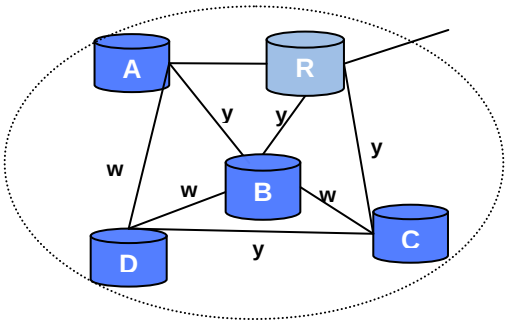
```
send(c,s)               //per spedire un messaggio su una connessione
close(c)                //per chiudere una connessione
```

E2 [6 punti]. Considerare una variante del protocollo di ripetizione selettiva SR in cui allo scadere di un timeout, anziché ritrasmettere solo il pacchetto P a cui il timer era associato, vengono ritrasmessi anche tutti gli eventuali pacchetti più "vecchi" di P spediti e non ancora riscontrati. Assumendo che il buffer del mittente sia rappresentato da un vettore **`Data`** (per memorizzare i dati spediti) e da un vettore **`Acked`** (per tenere traccia dei riscontri ricevuti), entrambi di dimensione **`bufferSize`**, oltreché dai cursori **`sendBase`** e **`nextSeqNum`**, descrivere con uno pseudo-codice il comportamento del mittente: (a) nella fase di inizializzazione; (b) quando riceve una richiesta di trasmissione di dati, e (c) quando scade un timeout.
Nello pseudo-codice modellare esplicitamente la circolarità del buffer.

E3 [7 punti]. Considerare un'applicazione B che ha già stabilito una connessione TCP con un suo pari. Specificare, motivando la risposta, il valore del campo *ACK* del segmento *B1* e del campo *SEQ* del segmento *A3*, supponendo che tutti i segmenti indicati nella figura a lato arrivino a destinazione non corrotti e che B non abbia dati da spedire.



E4 [3 punti]. Considerare l'area OSPF illustrata di lato, in cui R è l'unico router di confine. Indicare, giustificando la risposta, quale valore deve essere assegnato al collegamento A-R affinché tutti i pacchetti provenienti dall'esterno dell'area e destinati a D attraversino soltanto il router A (oltre ovviamente a R e a D), sapendo che w<y.



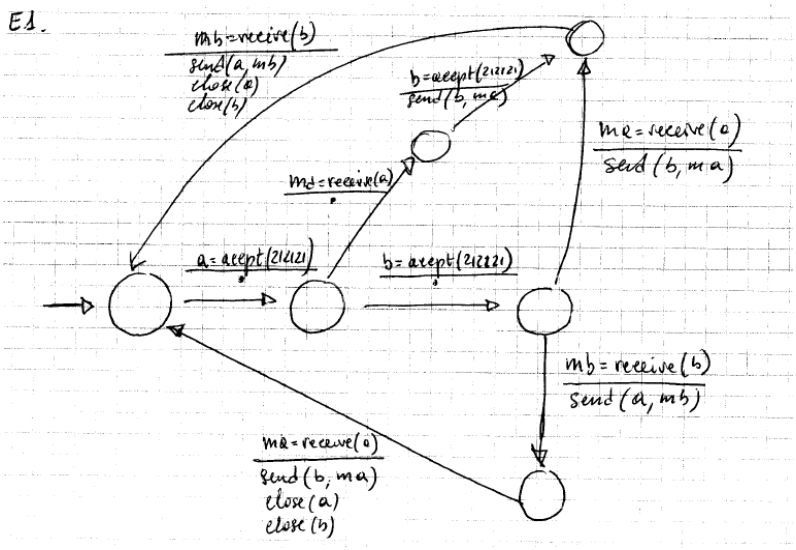
E5 [4 punti]. Indicare, giustificando la risposta, quanti pacchetti vengono spediti complessivamente per trasmettere un pacchetto in broadcast In un grafo fortemente connesso composto da N nodi nei casi in cui venga utilizzato: (a) flooding controllato con numeri di sequenza; (b) inoltro su percorso inverso (*RPF*); (c) un albero di copertura (*spanning tree*) precedentemente costruito.

E6 [4 punti]. Consideriamo il protocollo descritto a lato per lo scambio di una chiave di sessione tra due entità di rete A e B con l'ausilio di un server S.

(1) *A invia a B il messaggio:* $< M, A, B, K_{AS}(Na, M, A, B) >$
(2) *B invia a S il messaggio:* $< M, A, B, K_{AS}(Na, M, A, B), K_{BS}(Nb, M, A, B) >$
(3) *S invia a B il messaggio:* $< M, K_{AS}(Na, K_{AB}), K_{BS}(Nb, K_{AB}) >$
(4) *B invia a A il messaggio:* $< M, K_{AS}(Na, K_{AB}) >$

Supponiamo che il server S generi la chiave di sessione K_{AB}, che A condivida con il server la chiave simmetrica K_{AS} e che B condivida con il server la chiave simmetrica K_{BS}. Indichiamo con Na, Nb le *nonce* generate da A e B, rispettivamente, e con M l'identificatore della sessione. Determinare se il protocollo garantisce o meno che lo scambio della chiave di sessione avvenga correttamente, giustificando il perché nel primo caso oppure descrivendo un possibile attacco nel secondo caso.

TRACCIA DELLA SOLUZIONE



E2.
```
//inizializzazione
    sendBase=0;
    nextSeqNum=0;

//evento: riceve (dal livello superiore) la richiesta di trasmettere i dati D
    if ((sendBase<=nextSeqNum && nextSeqNum-sendBase<N) ||
        (sendBase>nextSeqNum && bufferSize-sendBase+nextSeqNum<N)) {
            Data[nextSeqNum]=makeSegment(nextSeqNum,D);
            Acked[nextSeqNum]=0;
            UDTsend(Data[nextSeqNum]);
            startTimer(nextSeqNum);
            nextSeqNum=(nextSeqNum+1) mod bufferSize
    }
//evento: scade il timeout del pacchetto x
    i=sendBase;
    do {
          if (Acked[i]==0) {
              UDTsend(Dati[i]);
              startTimer(i);
          }
          i=(i+1) mod bufferSize;
       }
    while (i!=x)
```

E3. Dato che l'host di B attende prima di inviare il riscontro di A1, ciò significa che A1 è un "arrivo ordinato" di un segmento col numero di sequenza "atteso". Il campo ACK di B1 conterrà quindi il valore X+2MSS, trattandosi di un riscontro cumulativo. Distinguiamo a questo punto due possibili casi[1]:

(a) A3 è stato spedito dal pari *dopo* che esso ha ricevuto B1. In tale caso A3 non può essere una ritrasmissione e deve quindi necessariamente contenere nuovi dati. Pertanto il campo SEQ di A3 conterrà il valore X+2MSS.

(b) A3 è stato spedito dal pari *prima* che esso ricevesse B1.

(b1) Se A3 contiene nuovi dati (ovvero non è una ritrasmissione) il suo campo SEQ conterrà il valore X+2MSS come in (a).

(b2) Se invece A3 è una ritrasmissione allora il suo campo SEQ conterrà un valore Y, dove Y è il numero di sequenza del segmento più vecchio spedito dal pari di B e non ancora riscontrato.

E4. Sia x il costo del collegamento A-R. Affinché i pacchetti provenienti dall'esterno destinati a D attraversino solo A deve valere che *min (x+w, x+y+w, x+2y+w) < min (y+w,2y+w,y+2w,2y)* ovvero *x+w < min (y+w,2y)* ovvero (dato che w<y) $x < y$.

E5. (a) Nel caso di flooding controllato con numeri di sequenza, il sorgente invia N-1 pacchetti. Ognuno degli altri N-1 nodi inoltrerà quindi il pacchetto ricevuto a N-2 suoi vicini – i quali non inoltreranno di nuovo il pacchetto. In totale verranno quindi complessivamente spediti $(N-1)^2$ pacchetti. (b) Nel caso di RPF il sorgente invierà N-1 pacchetti e ogni altro nodo invierà una sola volta il pacchetto ricevuto a N-2 suoi vicini. Anche in questo caso verranno quindi complessivamente spediti $(N-1)^2$ pacchetti. (c) Nel caso di ST verranno invece spediti soltanto N-1 pacchetti , uno per ogni arco dell'albero.

E6. Il protocollo descritto è suscettibile di attacchi del tipo "man-in-the-middle". Un intruso I può infatti inserirsi nel passo (1) del protocollo nel modo seguente:

(1') I intercetta il messaggio di A per B: < M, A, B, K_{AS}(Na, M, A, B) >

(1'') I inoltra a B il messaggio di A intercettato

e quindi di nuovo nel passo (4):

(4') I interecetta il messaggio di B per A: < M, K_{AS}(Na, K_{AB}) >

(4'') I invia ad A il messaggio: < M, K_{AS}(Na, M, A, B) >

e in questo modo A interpreterà la stringa (M,A,B) come la chiave di sessione e I potrà quindi decifrare i messaggi inviati da A a B. *(AB)*

[1] Alternativamente possiamo considerare i due casi: (a) A3 contiene nuovi dati. In tale caso il campo SEQ di A3 conterrà il valore X+2MSS. (b) A3 è una ritrasmissione (ovvero B1 è arrivato a destinazione dopo lo scadere del timeout). In tale caso il campo SEQ di A3 conterrà un valore Y, dove Y è il numero di sequenza del segmento più vecchio spedito dal pari di B e non ancora riscontrato.