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

Esercizio 1 (3 punti). Supponiamo che un browser C debba scaricare da un server remoto S una pagina HTML che referenzia altri 4 oggetti. Stimare il tempo necessario per completare il trasferimento con:

(i) HTTP non persistente, senza connessioni TCP parallele,
(ii) HTTP non persistente, con connessioni TCP parallele,
(iii) HTTP persistente

nei seguenti scenari:

(a) i 4 oggetti risiedono tutti anch'essi in S,
(b) i 4 oggetti risiedono tutti in un server T diverso da S,
(c) 2 oggetti risiedono in T e 2 oggetti risiedono in U, con T e U entrambi diversi da S.

Per semplicità supporre che ciascuno dei 5 file possa essere trasportato con un unico pacchetto IP e considerare costante nel tempo il valore dello RTT_{AB} tra due host A e B.

Esercizio 2 (6 punti).

a) Indicare – esplicitando il significato delle variabili utilizzate – la formula **F** utilizzata da TCP per coordinare i meccanismi di controllo del flusso e controllo di congestione.

b) Considerare un'applicazione A che al tempo t_0 deve inviare 4 MSS di dati a un suo pari B con cui ha già stabilito una connessione TCP. Supporre che in t_0 il valore di CongWin del TCP di A sia 3MSS, che non vi siano segmenti "in volo" contenenti dati spediti da A, e che il valore di RcvWin del TCP di B sia 4MSS. Supporre inoltre che il valore del timeout del TCP di A sia superiore a RTT_{AB}, e che solo 1 dei pacchetti IP spediti dal TCP di A vada perso (mentre tutto gli altri pacchetti scambiati tra A e B arrivano non corrotti e in ordine). Determinare in quanti dei possibili scenari la formula **F** si comporta come una proprietà invariante nel caso in cui il TCP di A si trovi in t_0 nello stato di *congestion avoidance* e nel caso in cui si trovi in t_0 in *slow start*. Giustificare la risposta fornita.

Esercizio 3 (7 punti). Estendere l'algoritmo *distance vector* (senza considerare il meccanismo di *poisoned reverse*) in modo che esso determini per ogni destinazione Y una *rotta primaria* e una *rotta secondaria* da un router X fino a Y. Una *rotta* è rappresentata da una coppia <m,h>, dove m indica il costo complessivo stimato da X a Y, e h indica il primo *hop*.

Scrivere lo pseudo-codice dell'algoritmo esteso eseguito da un router X, indicando:

- con C(Y) il costo del collegamento da X a Y,
- con R1(Y).cost e R1(Y).hop la rotta primaria da X a Y, e
- con R2(Y).cost e R2(Y).hop la rotta secondaria da X a Y.

Esercizio 4 (6 punti).

a) Descrivere con un automa a stati finiti il protocollo token-passing, utilizzando gli eventi:

```
<dest,data>=sendRequest()     //per ricevere una richiesta di spedire i dati data a dest
frame = receiveFrame()        //per indicare la ricezione di un frame
isToken(frame)                //per indicare che un frame contiene il token
```

e le azioni:

```
frame=makeFrame(dest,data)    // per indicare la costruzione di un frame
sendFrame(frame)              //per indicare la spedizione di un frame
forwardToken(frame)           //per indicare l'inoltro del token
data=extractData(frame)       //per estrarre i dati contenuti in un frame
deliverData(data)             //per restituire dati al livello superiore
```

b) Estendere l'automa in modo che transisca in uno stato "RigenerazioneToken" (non specificato, ovvero privo di transizioni uscenti) ogni volta che si rilevi la necessità di rigenerare il token.

Esercizio 5 (4 punti). Se A invia a B il messaggio $<K_B^+(K_A^-(m))$, A> vengono garantite la riservatezza e l'integrità di m? Giustificare la risposta fornita.

Esercizio 6 (4 punti). Consideriamo un rete Gnutella in cui ogni peer è connesso con al più N vicini nella rete di copertura e in cui il valore iniziale del campo di conteggio dei messaggi è K.

a) Determinare il limite superiore L al numero di messaggi di richiesta trasmessi a seguito di una richiesta.
b) Se N=3 quanto deve valere K affinché L sia circa 100?
c) Se K=3 quando deve valere N affinché L sia circa 100?

Traccia della soluzione

Esercizio 1. In generale, per trasferire un file con HTTP da un server B a un host A, e' necessario 1 RTT_{AB} per lo scambio dei primi due messaggi dell'handshake TCP e 1 RTT_{AB} per lo scambio dei due messaggi HTTP. Trascurando l'overhead per la gestione delle connessioni parallele (e supponendo che la versione persistente non usi connessioni TCP parallele) avremo quindi:

	(i)	(ii)	(iii)
(a)	10 RTT_{CS}	4 RTT_{CS}	6 RTT_{CS}
(b)	2 RTT_{CS} + 8 RTT_{CT}	2 RTT_{CS} + 2 RTT_{CT}	2 RTT_{CS} + 5 RTT_{CT}
(c)	2 RTT_{CS} + 4 RTT_{CT} + 4 RTT_{CU}	2 RTT_{CS} + max(2 RTT_{CT} , 2 RTT_{CU})	2 RTT_{CS} + 3 RTT_{CT} + 3 RTT_{CU}

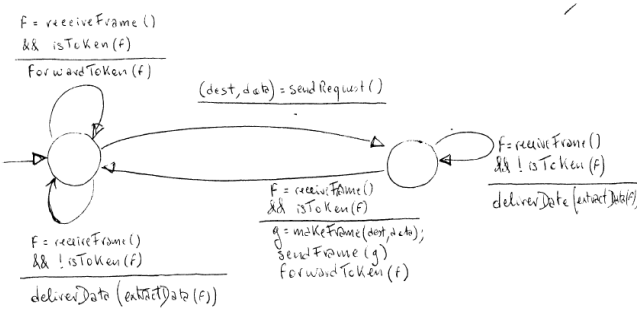
Esercizio 2. a) $F = (lastByteSent - lastByteAcked \leq min(CongWin,RcvWin))$
dove *lastByteSent* e *lastByteAcked* indicano rispettivamente il numero dell'ultimo byte spedito e riscontrato, e *CongWin* e *RcvWin* indicano la dimensione rispettivamente della finestra di congestione e della finestra di ricezione (quest'ultima ricevuta dal pari).
b) Per il controllo di congestione, al tempo t_0 il TCP di A puo' spedire tre segmenti contenenti ciascuno 1MSS di dati. Se uno di tali pacchetti andra' perso, allo scadere del relativo timeout il TCP di A ridurra' la dimensione della sua CongWin a 1MSS, pur essendovi piu' di 1MSS di dati ancora "in volo". Se invece il pacchetto che va perso e' il quarto spedito dal TCP di A, allo scadere del timeout vi sara' soltanto 1MSS di dati ancora"in volo" e quindi la formula F non sara' violata. In conclusione quindi, indipendentemente dallo stato (CA o SS) del TCP di A in t_0, la formula F si comporta come una proprieta' invariante solo in 1 dei 4 possibili scenari.

Esercizio 3. Indichiamo con $\boldsymbol{D_V} = [D_V(y_1),...,D_V(y_N)]$ il vettore delle distanze ricevuto dal nodo V.

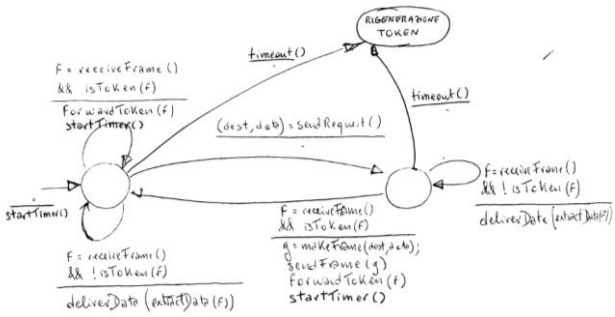
```
// Inizializzazione
per ogni destinazione Y:
R2(Y).cost=inf;
if Y vicino di X
  then R1(Y).cost=C(Y); R1(Y).hop=Y;
  else R1(Y).cost=∞;
per tutti i vicini W di X:
  per tutte le destinazioni Y:
   DW(Y) = ∞
per ogni vicino W di X:
  invia a W [R1(Y1).cost, ..., R1(YN).cost]
```

```
//loop
"quando cambia costo di un collegamento
 o riceve un nuovo vettore da un vicino"
  per ogni destinazione Y
    sia W vicino di X t.c. C(W)+DW(Y)=minV{C(V)+ DV(Y)};
    R1(Y).cost= C(W)+DW(Y); R1(Y).hop=W;
    sia Z vicino di X t.c. C(Z)+DZ(Y)=minV≠W{C(V)+ DV(Y)};
    R2(Y).cost= C(Z)+DZ(Y); R2(Y).hop=Z;
  if qualche valore di R1().cost è cambiato
     then invia a tutti i vicini
          [R1(Y1).cost, ..., R1(YN).cost]
```

Esercizio 4. (a)



(b)



Esercizio 5. La riservatezza di m e' garantita dall'utilizzo della chiave pubblica K_B^+ di B, ovviamente in funzione del meccanismo utilizzato da A per reperire K_B^+. Se K_B^+ e' infatti la chiave pubblica di B, solo B dovrebbe essere in grado di decifrare il messaggio. L'integrita' di m non e' invece garantita dato che ne' m ne' un suo hash sono replicati nel messaggio.

Esercizio 6. a) L = N + N(N-1) + N(N-1)^2 + ... + N(N-1)^K = $N\sum_{i=0}^{K}(N-1)^i$ = N ((N-1)$^{K+1}$-1)/(N-2).

(b) Se N=3 abbiamo che L= 3(2^{K+1}-1) e quindi per K=4 otteniamo L=93.
(c) Se K=3 abbiamo che L= N((N-1)^4-1)/(N-2) e quindi per N=4 otteniamo L=160. (AB)