RETI DI CALCOLATORI – Primo appello, a.a. 2009/2010

La prova è strutturata in due parti. Per ottenere una valutazione sufficiente dell'intera prova è necessario ottenere una valutazione sufficiente della prima parte. In accordo con quanto deliberato dal Consiglio di Facoltà il 15.12.2009, ogni studente può partecipare a tutti i cinque appelli previsti e consegnare al più quattro prove scritte.

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

Q1. Supponiamo che un router A debba inviare un pacchetto di 2n bit al suo vicino B. Indicare, giustificando la risposta, quanto deve essere il ritardo di propagazione sul collegamento tra A e B affinché i primi n bit del pacchetto siano già arrivati a B quando l'ultimo bit del pacchetto viene immesso sul collegamento.

Q2. Consideriamo un'implementazione di TCP in cui ogni volta che il valore di CongWin viene modificato esso non viene più incrementato fino a quando il pari non ha effettivamente ricevuto CongWin nuovi byte. Specificare come può essere realizzata tale implementazione in modo che il valore di CongWin (al più) raddoppi a ogni RTT nello stato di slow start e incrementi di (al più) 1 MSS nello stato di congestion avoidance.

Q3. Consideriamo un sistema autonomo i cui router utilizzano il protocollo distance vector e supponiamo che il vettore delle distanze del router R contenga i valori $D_R(V1)=x$, $D_R(V2)=2x$ e $D_R(E)=x+y$, dove V1 e V2 sono gli unici due vicini di R, e che gli ultimi due advertisement ricevuti da R contenessero i valori $D_{V1}(E)=y$ e $D_{V2}(E)=2y$. Indicare in quale caso R aggiorna la sua distanza $D_R(E)$ se riceve un advertisement da V2 con cui V2 annuncia che la sua nuova distanza $D_{V2}(E)$ per E è y/2.

Q4. Indicare, giustificando la risposta, per quale motivo l'utilizzo di un MAC non è sufficiente per realizzare la firma digitale di un documento.

Seconda parte

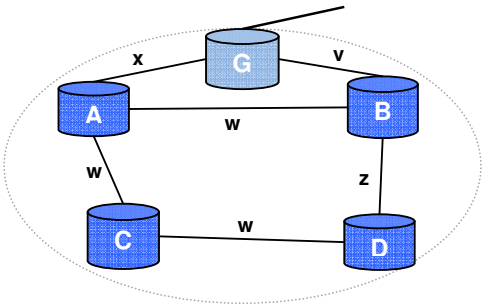
E1 (6 punti). Consideriamo un'applicazione che utilizza UDP per realizzare un trasferimento (non affidabile) unidirezionale di dati in cui il mittente utilizza numeri di sequenza per numerare ogni gruppo di dati che invia con UDP. Il ricevente mostra all'utente i dati ricevuti rispettando l'ordine con cui sono stati spediti dal suo pari e tollerando la perdita di al più due gruppi consecutivi di dati. Quando il ricevente riceve l'i-esimo blocco di dati risponde al suo pari

- con un messaggio ACK(i) se può mostrare i dati ricevuti all'utente, altrimenti
- con un messaggio NEED(x) indicando così che l'ultimo blocco che è riuscito a mostrare all'utente è l' x-1 esimo.

Specificare, con un automa a stati finiti, il comportamento del lato mittente dell'applicazione assumendo che utilizzi una finestra di dimensione W e ignorando per semplicità la necessità di verificare l'integrità dei dati trasmessi. Esplicitare con opportuni commenti il significato di eventi, azioni e variabili utilizzati.

E2 (6 punti). (a) Supponiamo che al tempo t_0 il TCP di un processo applicativo A abbia 1 MSS di dati "in volo" (sia X il valore della variabile sendBase del TCP di A in t_0) e che, pur essendo il suo valore di CongWin pari a 3MSS, il TCP di A non spedisca gli altri 2MSS di nuovi dati che ancora deve spedire. Supponiamo che nell'intervallo $[t_0, t_1)$ il TCP di A non rilevi nessun evento e che al tempo t_1 il TCP di A riceva dal suo pari un segmento S. Indicare, giustificando la risposta, quali valori deve contenere S affinché il TCP di A possa spedire entrambi i 2MSS di nuovi dati subito dopo avere ricevuto S. **(b)** Rispondere al precedente quesito considerando lo scenario in cui il valore di CongWin del TCP di A in t_0 sia pari a 6MSS (anziché a 3MSS).

E3 (4 punti). Consideriamo l'area OSPF illustrata di lato. Determinare, giustificando la risposta, un insieme di vincoli che permettano di assegnare costi ai collegamenti nell'area in modo da garantire che il traffico in uscita dall'area non passi per B (ad eccezione ovviamente di quello generato da B). Mostrare un esempio di assegnazione dei costi che soddisfi tali vincoli.



E4 (4 punti). Consideriamo una variante di CSMA/CD in cui l'algoritmo di attesa esponenziale ("exponential backoff") è sostituito da un algoritmo di attesa lineare in cui il tempo di attesa dopo una collisione cresce probabilisticamente in modo lineare anziché esponenziale al crescere del numero di tentativi di rispedizione effettuati. Supponendo che due sole stazioni debbano spedire dati e che esse collidano al primo tentativo, determinare – per entrambi i protocolli – quale è la probabilità che entrambi i nodi riescano a spedire effettuando ciascuno al più altri 3 tentativi.

TRACCIA DELLA SOLUZIONE

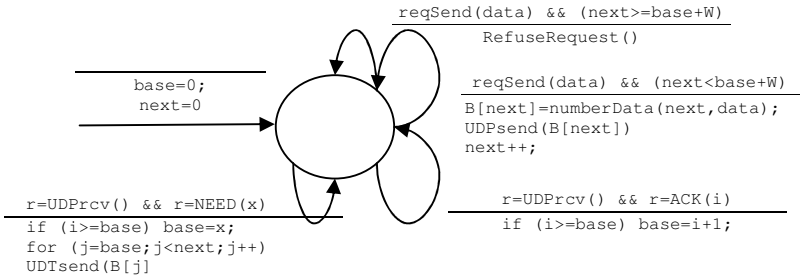
Q1. Il tempo necessario per immettere tutti i bit sul collegamento è $d_{trasm} = 2n/R$ (dove R è la frequenza di trasmissione del canale), mentre il tempo necessario affinché l'n-esimo bit del pacchetto raggiunga B è $n/R + d_{prop}$. Affinché i primi n bit del pacchetto siano già arrivati a B quando l'ultimo bit del pacchetto viene immesso sul collegamento deve quindi valere: $d_{trasm} > d_{trasm}/2 + d_{prop}$ ovvero $d_{prop} < d_{trasm}/2$.

Q2. L'implementazione può semplicemente utilizzare una variabile ackedBytes – da reinizializzare a 0 ogni volta che viene modificato il valore di CongWin -, incrementarla di (Y-sendBase) ogni volta che riceve un riscontro con AckNum Y per dati che considerava ancora "in volo" (ovvero Y>sendBase) e non appena ackedBytes>=CongWin raddoppiare o incrementare di 1MSS il valore di CongWin in funzione dello stato (slow start o congestion avoidance) in cui si trova.

Q3. R aggiorna la sua distanza per E solo se C(R,V2)+y/2 < x+y, dove C(R,V2) indica il costo del collegamento tra R e V2.

Q4. Non è sufficiente perché non può essere garantita la non ripudiabilità del messaggio. Per poter verificare l'integrità di un messaggio m, il destinatario deve infatti conoscere la chiave di autenticazione s utilizzata dal mittente per generare il MAC H(m+s) di m. Non è quindi possibile garantire che il messaggio esteso <m, H(m+s)> sia stato effettivamente firmato dal mittente (e non dal destinatario).

E1. Supponiamo che il mittente utilizzi un buffer *B* per memorizzare i dati spediti e in attesa di riscontro, la variabile *base* per indicare il gruppo di dati più vecchio ancora in volo e la variabile *next* per indicare il prossimo numero di sequenza utilizzabile. Indichiamo con *reqSend* l'operazione con cui l'utente chiede di inviare nuovi dati, con *UDPsend* e *UDPrcv* le operazioni offerte da UDP, con *numberData* l'operazione che numera un gruppo di dati e con *refuseRequest* la risposta all'utente.



E2. (a) Vi sono due casi possibili:

(i) S contiene il riscontro di dati che il TCP di A considerava ancora in volo, ovvero S.AckNum=X+1MSS, e S.RcvWin≥2MSS. In tal caso controllo di flusso e congestione permetteranno al TCP di A di spedire 2MSS di nuovi dati subito dopo avere ricevuto S.

(ii) S non riscontra nessuno dei dati che il TCP di A considera ancora in volo, ovvero S.AckNum≤X, ma S.RcvWin≥3MSS. In tal caso il TCP di A può spedire 2MSS di nuovi dati purché S non sia il terzo ACK duplicato ricevuto per X.

(b) I casi possibili sono gli stessi anche nel secondo scenario. L'unica differenza è che nel secondo caso (ii) S può anche essere il terzo ACK duplicato ricevuto per X (in tal caso il TCP di A rispedirà il segmento con numero di sequenza X oltre a spedire idue nuovi segmenti).

E3. Dobbiamo garantire che:

- A inoltri a G e non a B i pacchetti con destinazione esterna all'area,
- C inoltri sul percorso AG i pacchetti con destinazione esterna all'area, e
- D inoltri sul percorso CAG i pacchetti con destinazione esterna all'area.

Ne derivano i seguenti vincoli:

(a1) $x < w + v$
(a2) $x < 2w + z + v$ //sussunto da (a1)
(c1) $w + x < w + z + v$ //sussunto da (d1)
(c2) $w + x < 2w + z + x$ //soddisfatto da ogni w e z ≥0
(c3) $w + x < 2w + z$ //coincide con (a1)

(d1) $2w + x < z + v$
(d2) $2w + x < z + w + x$
(d3) $2w + x < 3w + z$ //coincide con (a1)

ovvero

(a1) $x < w + v$
(d1) $2w + x < z + v$
(d2) $w < z$

Qualunque assegnazione dei costi che soddisfi i vincoli (a1), (d1) e (d2) garantisce quindi che il traffico in uscita non passi per B. Un esempo di assegnazione è: x=1, v=2, w=1, z=3.

E4. Nel caso di CSMA/CD dopo che si è verificata la i-esima collisione un nodo attende K·512bit, con K valore intero scelto in modo casuale nell'intervallo $[0,2^i-1]$, prima di tentare nuovamente la spedizione. La probabilità che i due nodi riescano a spedire con al più altri 3 tentativi è quindi: $1 – (1/2 \cdot 1/4 \cdot 1/8) = 63/64$. Se nella variante considerata dopo che si è verificata la i-esima collisione K viene scelto in modo casuale nell'intervallo [0,2i-1], la probabilità che i due nodi riescano a spedire con al più altri 3 tentativi diventa: $1 – (1/2 \cdot 1/4 \cdot 1/6) = 47/48$.