RETI DI CALCOLATORI – prova scritta del 9/02/2017

Per essere ammessi alla prova orale è necessario ottenere una valutazione sufficiente sia della prima parte che dell'intera prova.

Prima parte (15 punti)

Q1. Ugo si trova per la prima volta in una zona di free wifi, e vuole vedere la pagina web https://www.di.unipi.it/didattica. Accende il suo computer ed invia una richiesta per quella pagina. Quali protocolli ed in quale ordine vengono utilizzati dal computer di Ugo da quando viene acceso a quando viene inviata la richiesta? Giustificare la risposta.

Q2. Al tempo t_0 di una connessione TCP già stabilita da un host H, si ha che cwnd= 9 e ssthresh=4, mentre al tempo t_1 ($t_1>t_0$) cwnd=1. Se tra t_0 e t_1 non scade nessun timeout, qual'è una sequenza di eventi che permette questa variazione di valore?

Q3. I percorsi interni ad un sistema autonomo calcolati usando RIP od OSPF sono necessariamente uguali, oppure no? Giustificare la risposta.

Q4. Una rete locale utilizza il protocollo MAC Slotted Aloha in cui la gestione della ritrasmissione dei frames che hanno subito collisioni è uguale a quella del protocollo CSMA/CD, con il metodo di persistenza p-persistente con p=0,8. Nello slot X tre nodi della rete trasmettono simultaneamente un frame: il nodo A per la prima volta, il nodo B per la seconda e il nodo C per la terza. Qual'è la probabilità che nella ritrasmissione successiva questi frames non subiscano collisioni, e che avvenga nello slot X+3 per A e nello slot X+4 per B, ipotizzando che la notifica della collisione avvenga nello slot X+1 e che nessun altro nodo della rete trasmetta frames? Giustificare la risposta.

Q5. Si consideri una rete Ethernet con topologia a bus, lunga 150 metri. La velocità di propagazione dei segnali nel bus è di $2x10^8$ m/sec ed R=12Mbps. Un host H inizia al tempo t ad inviare un frame lungo 256 byte, che subisce collisione. Qual'è il numero massimo di bit del frame che H riesce a trasmettere prima di iniziare ad inviare il jamming signal? Giustificare la risposta.

Seconda parte

E1 (8 punti). Un sistema autonomo S è connesso con internet tramite un router NAT R. In particolare, R è dotato di quattro interfacce di rete, I1, I2, I3 ed I4, le prime tre delle quali sono collegate rispettivamente con i tre sottosistemi del sistema autonomo, denominati SAS1, SAS2, e SAS3 ai quali sono assegnati (rispettivamente) i blocchi di indirizzi privati 192.168.0.0/18, 192.168.64.0/18 e 192.168.128.0/18. La quarta interfaccia di rete, I4, che ha l'indirizzo pubblico 113.114.97.0, è collegata con internet. R svolge funzioni di instradamento sia tra i sottosistemi autonomi (è l'unico router che collega tali sottosistemi) che tra il sistema autonomo ed internet. Descrivere, mediante pseudocodice, il funzionamento di R quando deve inoltrare datagram, sia provenienti dal sistema autonomo che da internet. Per brevità, si considerino **solamente i datagram che provengono dalla interfaccia I1**. Inoltre, si hanno a disposizione le seguenti procedure:

receive(interf,dg) per ricevere il datagramma dg dalla interfaccia interf;

send(interf,dg) per inviare il datagramma dg sulla interfaccia interf;

inblocco(a,b,c) che restituisce true se l'indirizzo a è nel blocco di indirizzo iniziale b e finale c, e false altrimenti;

private_address_error per segnalare un errore sull'indirizzo privato;

nattablook(ind,port) che restituisce (a,b), dove a è un booleano che vale true se è stata trovata una riga con la coppia (ind,port), false altrimenti; se a=true, allora b è l'indice della tabella che contiene quell'elemento, altrimenti il valore non è significativo.

addentrynattable per aggiungere una riga alla tabella nat: restituisce l'indice della riga aggiunta;

freeport che restituisce una porta di R attualmente inutilizzata.

Per semplicità, si supponga che i numeri di porta siano replicati nel campo options del datagramma: *dg.options.sourceport* per la porta sorgente, e *dg.options.destport* per la porta destinazione.

E2 (7 punti). Si vuole usare il protocollo Go Back N come protocollo di finestra scorrevole sia per inviare che per ricevere dati, inviando gli ack in piggybacking, quando possibile. Per fare questo, il receiver, non invia subito gli ack come pacchetti a se stante, ma li passa al sender con il comando *ackinpb(p)* e attende un massimo di TAP istanti: se nel frattempo il sender comunica al receiver di aver inviato l'ack in piggybacking (comunicazione che avviene mediante il comando *acksent()*), allora il receiver non fa niente altro per quell'ack. Altrimenti, allo scadere del timeout, il receiver invia l'ack come pacchetto a se stante, ed informa il sender di tale invio invocando *unsendack()*, a seguito del quale il sender non invia più quel riscontro in piggybacking. Per tutte le altre funzioni, il receiver si comporta normalmente. Descrivere, mediante un automa a stati finiti che utilizza pseudocodice (come nell'automa [GBN] del materiale didattico) e non con descrizione delle attività a parole, il comportamento del receiver della variante del protocollo Go Back N sopra descritta.

Traccia della soluzione

Q1. 1. IEEE 802.11 per l'associazione all'access point; 2. DHCP, UDP, IP ed IEEE 802.11 per ottenere un indirizzo IP; 3. DNS, UDP, IP, ARP, IEEE802.11 per poter comunicare con il router della WLAN ed avere l'indirizzo IP del server web; 4. HTTPS, SSL/TLS, TCP, IP, IEEE 802.11 per aprire la connessione con il server web e poi per inviare la GET.

Q2. Dati i valori iniziali di cwnd e ssthresh, al tempo t_0 il TCP in questione si trovava in stato di fast recovery (FR). Se riceve un riscontro non duplicato, passa in congestion avoidance (CA) ponendo cwnd=ssthresh=4. Poi, riceve 3 duplicati di quell'ACK e passa in FR con cwnd=5 e ssthresh=2. Quindi riceve un ack nuovo e torna in CA con cwnd=ssthresh=2. Riceve di nuovo 3 duplicati di quell'ACK e passa in FR con cwnd=4 e ssthresh=1. Quindi riceve un ack nuovo e torna in CA con cwnd=ssthresh=1.

Q3. No, in genere sono diversi perché i due protocolli utilizzano metriche diverse per calcolare il costo dei percorsi: RIP utilizza sempre hop-count, mentre OSPF utilizza una metrica decisa dall'amministratore di rete, non necessariamente hop-count.

Q4. Sia K(N) il valore di attesa generato dal nodo N. La probabilità che la ritrasmissione successiva di A sia nello slot X+3 è 1/5x4/5x1/2 se viene generato K(A)=0, oppure 4/5x1/2 se K(A)=1. Per il nodo B, ritrasmissione nello slot X+4: 1/5x1/5x4/5x1/4, se K(B)=0 oppure 1/5x4/5x1/4 se K(B)=1 oppure 4/5x1/4 se K(B)=2. Per il nodo C, questo non deve trasmettere né nello slot X+3 né nello slot X+4. La probabilità che trasmetta in X+3 è:1/5x4/5x1/8 se viene generato K(C)=0 oppure 4/5x1/8 se K(C)=1. Quella che trasmetta in X+4 è: 1/5x1/5x4/5x1/8 se K(C)=0 oppure 1/5x4/5x1/8 se K(C)=1 oppure 4/5x1/8 se K(C)=2. Quindi, per A si ha che la probabilità che trasmetta nello slot X+3 è 2/25+2/5; per B che trasmetta nello slot X+4 è 1/125+1/25+1/5 e per il nodo C è: 1-(1/50+1/10 + 1/250+1/50+1/10). Pertanto, la probabilità richiesta è 12/25x31/125x189/250=70308/781250=0,0899942.

Q5. H si accorge della collisione (e quindi inizia ad inviare il jamming signal) quando riceve il primo bit trasmesso dall'altro host (quello con cui ha collisione). Al massimo, questo avviene dopo 2 volte il ritardo di propagazione, t_{prop}. Siccome $t_{prop}=1590/2x10^8=0,75x10^{-6}$ sec., H si accorge della collisione dopo $1,5x10^{-6}$ sec. (al massimo), durante i quali ha trasmesso $1,5x10^{-6}$ x$12x10^6$=18 bit.

E1.

```
receive(I1,dg);
if inblocco(dg.inddest,192.168.64.0,192.168.127.255)
        {send(I2,dg)}
        else if inblocco(dg.inddest,192.168.128.0,192.168.191.255)
                {send(I3,dg)}
                else if inblocco(dg.inddest,192.168.192.0,192.168.255.255)
                        {private_address_error}
                        else {  (exist,i)=nattablelook(dg.indsource,dg.options.sourceport);
                                if ! exist
                                        {  i=addentrynattable;
                                           nattable[i].privateaddr=dg.indsource;
                                           nattable[i].privateport=dg.options.sourceport;
                                           nattable[i].newport=freeport;
                                           nattable[i].externaladdr=dg.inddest;
                                           nattable[i].externalport=dg.options.destport;
                                        }
                                dg.indsource=113.114.97.0;
                                dg.options.sourceport=nattable[i].newport;
                                send(I4,dg);
                             }
```

E2.

Due stati servono per gestire un ack alla volta in modo semplice (altrimenti bisogna usare dei vettori per sapere lo stato dei vari ack pendenti: potrebbero arrivare più segmenti dati in TAP istanti).