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

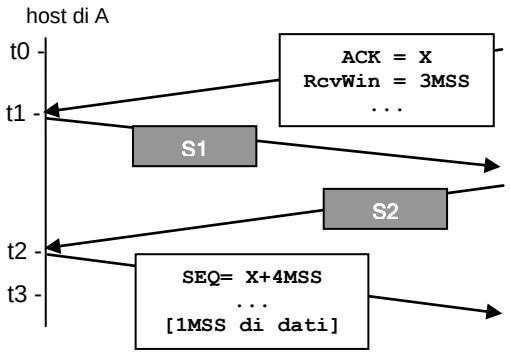
E1 (4 punti). Consideriamo un messaggio di posta elettronica con mittente **`alice@unipi.it`**, destinatario **`bob@mit.edu`** e testo del messaggio "**`Happy new year!`**". Indicare il contenuto dei primi due segmenti TCP inviati dal mailserver **`smtp.unipi.it`** per ricevere tale messaggio, specificando di entrambi i segmenti sia il contenuto della parte dati che i valori dei campi numero di porta origine, numero di porta destinazione, numero di sequenza, numero di riscontro ed eventuali bit del campo flag a 1.

E2 (4 punti). Consideriamo un'applicazione che utilizza UDP per trasferire dati da un server a un cliente. Scrivere il codice del lato cliente dell'applicazione supponendo di avere a disposizione le operazioni:

<d, n> = UDP_rcv() // per indicare la ricezione dell' *n*-esimo segmento contenente i dati *d*
UDP_send(<"ACK",m>) // per indicare l'invio di un riscontro selettivo per l'*m*-esimo segmento
UDP_send(<"stop">) // per indicare l'invio di un messaggio per interrompere il trasferimento
show(d) // per mostrare i dati *d* all'utente

in modo che il lato cliente, senza utilizzare buffer di ricezione, mostri all'utente (alcuni de)i dati ricevuti – rispettando l'ordine dei segmenti - fino a quando la percentuale di dati persi non superi il 25%.

E3 (6 punti). Consideriamo un'applicazione A che ha già stabilito una connessione TCP con un suo pari. Supponiamo che al tempo t_0 il valore della finestra di congestione **`CongWin`** dell'host di A sia pari a *4MSS*, gli indici **`sendBase`** e **`nextSeqNum`** valgano rispettivamente *X* e *X+4MSS* e che, sempre al tempo t_0, l'host di A debba spedire 2 MSS di (nuovi) dati. Supponiamo inoltre che nell'intervallo [t0,t3] non scada nessun timeout e che al tempo t3 l'host di A non spedisca ulteriori segmenti (fino al verificarsi di nuovi eventi).



Specificare, motivando la risposta:

- il valore del campo **`SEQ`** del segmento S1,
- il valore dei campi **`ACK`** e **`RcvWin`** del segmento S2,
- i valori delle variabili **`CongWin`**, **`sendBase`** e **`nextSeqNum`** al tempo t3.

Supporre, per semplicità, che tutti i segmenti contenenti dati spediti dall'host di A prima di t0 contengano 1MSS di dati.

E4 (6 punti). Specificare il modo in cui un router R determina l'insieme N dei suoi vicini che appartengono all'albero di copertura ("spanning tree") costruito per supportare routing broadcast. Considerare sia il caso in cui R invia che quello in cui riceve un messaggio di adesione.

E5 (4 punti). Consideriamo una rete locale di N nodi, in cui X nodi devono spedire 3 frame ciascuno, e confrontiamo il tempo complessivo richiesto per spedire tali frame nel caso in cui il protocollo MAC utilizzato sia TDM oppure token-passing. Sia T_F il tempo necessario per spedire un frame e sia T_T il tempo necessario per spedire il token.

(a) Se i nodi X che devono spedire frame di dati sono un decimo di tutti i nodi N, quanto deve essere al più T_T affinché token-passing impieghi meno tempo di TDM per spedire tutti i 3X frame di dati?
(b) Se il tempo necessario per spedire un frame è il quadruplo del tempo necessario per spedire il token, quante devono essere almeno le X stazioni affinché TDM impieghi meno tempo di token-passing per spedire tutti i 3X frame di dati?

E6 (6 punti).

(a) Illustrare in che modo SSL realizza l'autenticazione di un server, esplicitando il contenuto dei messaggi scambiati e le azioni eseguite sui due lati. Spiegare perché tale protocollo di autenticazione è indenne da attacchi del tipo "person-in-the-middle".
(b) Spiegare per garantire quale proprietà SSL introduce numeri di sequenza nella parte dati dei segmenti e in che modo tale meccanismo contribuisce a garantire tale proprietà.

Traccia della soluzione

E1. Il primo segmento inviato dal mailserver sara' un segmento di tipo "SYNACK" contenente:

portaOrigine = 25
portaDestinazione = P
SEQ = Y
ACK= X+1
bit SYN e ACK a 1
DATI: nessun dato spedito

dove P e' il numero di porta del cliente e Y e X sono i numeri di sequenza inziali scelti dal server smtp.unipi.it e dal client, rispettivamente. Il secondo segmento conterra' invece:

portaOrigine = 25
portaDestinazione = P
SEQ = Y+1
ACK= X+1
DATI: 220 smtp.unipi.it

E2.

```
expected=1;                    //numero di sequenza atteso
shownSegments=0;               //numero di segmenti i cui dati sono gia stati mostrati a utente
do {
 <d,n>=UDP_rcv();              //riceve un nuovo segmento
 UDP_send(<"ACK",n>);          //invia riscontro corrispondente
 if (n >= expected) {          //se segmento ricevuto in ordine
       show(d);                //mostra dati all'utente
       shownSegments++;
       expected=n+1;
 }                             //se segmento ricevuto non in ordine lo scarta
}
while ( (shown/(expected-1)) >= 0.75 );      //controlla percentuale dati persi
UDP_send(<"STOP">);
```

E3. Dato che al tempo t0 l'host di A non puo' spedire nuovi dati, il segmento S1 che viene spedito al tempo t1, dopo avere ricevuto un riscontro negativo (ACK=X), e' una "ritrasmissione veloce" del pacchetto piu' vecchio ancora in volo. Quindi il **valore del campo SEQ del segmento S1 e' X** e al tempo t1 TCP dimezza il valore di CongWin e passa nello stato di congestion avoidance.
Dato che al tempo t2 l'host di A inviera' nuovi dati, il segmento S2 deve contenere un riscontro positivo, a seguito della cui ricezione (ovvero in t2) il valore di CongWin diventa 2,5MSS.
Consideriamo adesso i possibili valori del campo ACK del segmento S2, sapendo che deve trattarsi di un riscontro positivo.
Notiamo che il valore del campo ACK di S2 deve essere maggiore di X+2MSS, altrimenti in t2 vi sarebbero ancora "in volo" piu' di 2 MSS di dati e quindi, per il controllo di congestione, TCP non potrebbe inviare inviare 1MSS di nuovi dati.
Sfruttando l'ipotesi semplificativa che tutti i segmenti contenenti dati spediti dall'host di A prima di t0 contengano 1MSS di dati, abbiamo solo due casi possibili:

in S2		al tempo t3		
ACK	RcvWin	sendBase	nextSeqNum	CongWin
X+4MSS	1MSS	X+4MSS	X+5MSS	2,5MSS
X+3MSS	2MSS	X+3MSS		

E4. Nel caso in cui sia R a inviare il messaggio di adesione indirizzato al nodo "centrale" C, l'insieme N viene inizializzato con il vicino di R che è il prossimo hop per C nella tabella di indirizzamento unicast di R.
Nel caso in cui R riceva un messaggio di adesione dal collegamento con un suo vicino V, R aggiungerà V all'insieme N. Inoltre se R non aveva ancora inviato il suo messaggio di adesione, R aggiungerà a N anche il suo vicino che é il prossimo hop per C nella tabella di indirizzamento unicast di R.

E5. Il tempo complessivo richiesto per trasmettere i 3X frame di dati e' $3NT_F$ per TDM e $3NT_T+3XT_F$ per token passing.
(a) Dobbiamo determinare il valore di T_T tale che:

$$3NT_F > 3NT_T+3/10NT_F$$

ovvero

$$T_T < 9/10T_F.$$

(b) Dobbiamo determinare il valore di X tale che:

$$12NT_T < 3NT_T+12XT_T$$

ovvero

$$X > 3/4\ N.$$

[Per ottenere una valutazione piu' esatta dovremmo confrontare nel caso (a) il caso ottimo di TDM con il caso pessimo di token-passing, ovvero risolvere la disequazione $(2N+X)\ T_F > 3NT_T+3XT_F$ ottenendo $T_T < 3/5T_F$. Dualmente nel caso (b) dovremmo risolvere la disequazione $3NT_F < (2N+X)T_T+3XT_F$ ottenendo $X > 10/13\ N$.]

E6. (a) Il protocollo di autenticazione utilizzato da SSL prevede l'invio da parte del client di un messaggio "SSL hello" contenente un nonce e l'invio da parte del server di un messaggio di risposta contenente un certificato (contenente la chiave pubblica K_S^+ del server) e il nonce del server. Il cliente verifica l'identità del server analizzando il certificato, ovvero verificando che la chiave K_S^+ ricevuta appartiene effettivamente S. L'attendibilità di tale verifica dipende ovviamente dall'attendibilità della certification authority che ha rilasciato il certificato. L'uso di certificati (attendibili) permette quindi al client di verificare l'identità del server, inibendo così la possibilità di attacchi del tipo "man-in-the-middle" da parte di intrusi che riescano a inserirsi nello scambio di messaggi e cerchino di sostituirsi al server.
(b) SSL introduce nel MAC numeri di sequenza (uno per ogni nuovo record SSL inviato) per garantire l'integrità dei messaggi, in particolare per evitare che un intruso "in-the-middle" possa alterare -senza che gli interessati se ne accorgano - l'ordine dei segmenti TCP scambiati, modificando i valori dei campi del preambolo TCP relativi ai numeri di sequenza utilizzati da TCP.

(AB)