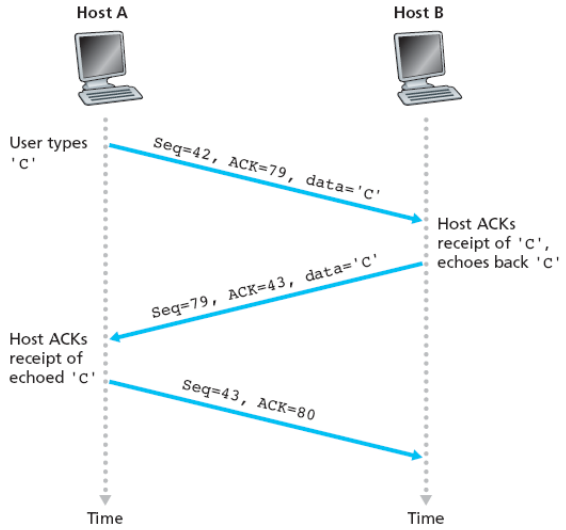
Esercizi (e relative soluzioni)

D: Supponiamo che l'host A stia trasmettendo a B due segmenti su una connessione TCP. Il primo segmento ha numero di sequenza 90 e il secondo 110.

1. Quanti dati si trovano nel primo segmento ?
2. Supponiamo che il primo segmento vada perso, ma il secondo arrivi a B. Quale sarà il numero di acknowledgement che B manda ad A quando riceve il secondo segmento ?

R: (1) 110-90 = 20 bytes. (2) Se B riceve il secondo segmento ma non ha ricevuto il primo l'ack che manda è un ack duplicato che indica quale è il prossimo segmento (in sequenza) che B si aspetta di ricevere e quindi ack number in questo caso è 90.

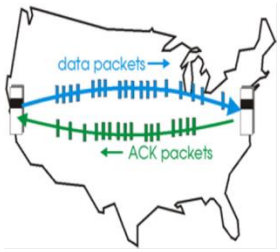
D: Consideriamo l'esempio telnet seguente



Qualche secondo dopo l'immissione da parte dell'utente della lettera 'C', viene immessa la lettera 'R'. Dopo tale immissione, quanti segmenti risultano spediti e quali valori sono posti nei campi numero di sequenza e acknowledgement dei segmenti ?

R: Questo esercizio fa riferimento all'esempio descritto nel libro (pag. 224 e 225 versione italiana). Quando l'host A invierà la lettera 'R' il segmento corrispondente avrà come Seq=43, Ack=80 e data='R', l'host B risponderà con Seq=80, Ack=44, e data='R'.

D: Considerate il seguente scenario:



- trasferimento dati tra due host situati sulle coste est ed ovest degli USA
- Il ritardo di propagazione di one-way è di 15 millisecondi (RTT=30 ms)
- Canale con tasso di trasmissione (R)di 1 Gbps (10^9 bit al secondo)
- Pacchetti di dimensione (L) di 1500 byte (12000 bit) comprensivi di intestazione e dati

Quanto dovrebbe essere grande l'ampiezza della finestra affinché l'utilizzo del canale superi il 98% ?

R: Riferimento per la comprensione di questo esercizio: pag. 205-206 libro di testo.
Il tempo per trasmettere un pacchetto è pari a 12 microsecondi (or 0.012 millisecondi), d_t =1500*8/10^9=12 microsecondi. Per ottenere che l'utilizzazione del canale superi il 98% dobbiamo avere che $util = 0.98 = (d_t \cdot n)/(RTT + d_t) = (0.012n)/30.012$ ed in questo casi si ottiene che n è approssimativamente uguale a 2451 pacchetti.

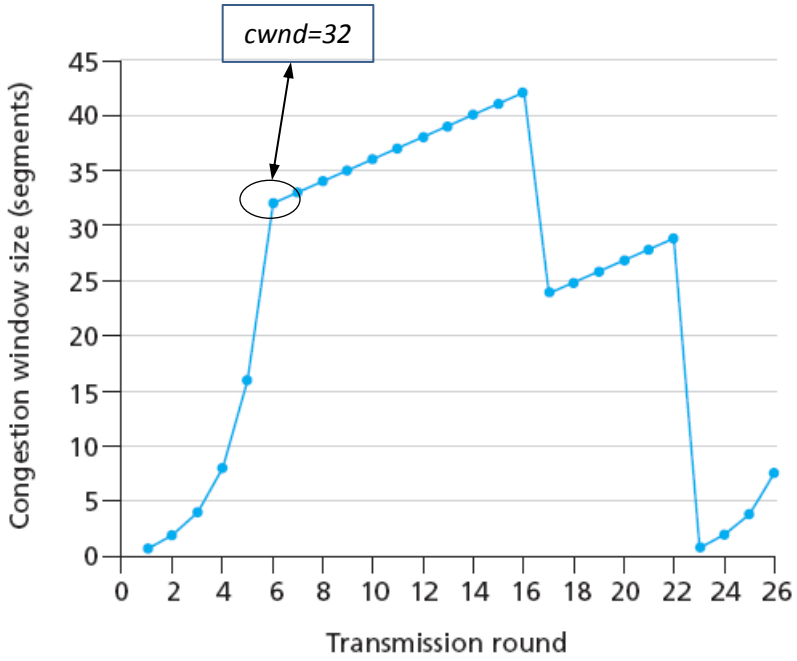
D: Considerate il protocollo GBN con un'ampiezza della finestra mittente pari a 4 e l'intervallo di numeri di sequenza da 0 a 1024. Supponete che all'istante t il successivo pacchetto nella sequenza atteso dal destinatario abbia numero di sequenza k. Ipotizzate che il mezzo non effettui riordino dei messaggio. Rispondete alle domande seguenti:

(1) Quali sono i possibili numeri di sequenza all'interno della finestra del mittente all'istante t ? Giustificate la vostra risposta.
(2) Quali sono tutti i possibili valor dei campo ACK nei possibili messaggi che si propagano all'indietro verso il mittente all'istante t ? Giustificate la vostra risposta.

R: Riferimento Sezione 3.4.3.

(1) Il ricevitore si aspetta di ricevere il pacchetto con sequence number k, quindi gli ultimi ACK inviati sono stati k-1, k-2, k-3, k-4. Se nessuno di questi ACK è stato ricevuto dal server allora la finestra del sender è (k-4, k-3,k-2,k-1). L'ACK con k-5 deve essere stato ricevuto altrimenti il pacchetto k-1 non potrebbe essere stato spedito. Altra possibilità se tutti gli ACK sono stati ricevuti allora il sender avrà una finestra pari a (k,k+1,k+2,k+3). Mettendo insieme tutte queste considerazioni l'insieme delle possibili finestre del sender è formato da: (k-4,k-3,k-2,k-1), (k-3,k-2,k-1,k), (k-2,k-1,k,k+1), (k-1,k,k+1,k+2),(k,k+1,k+2,k+3).
(2) Applicando le stesse considerazione fatte al punto precedente possiamo dire che i possibili valori del campo ACK per i messaggi che si propagano verso il mittente sono k-1, k-2, k-3, k-4.

D: Considerate l'evoluzione della finestra di TCP rappresentata dal grafico. Ipotizzando che sia il protocollo TCP Reno (implementa fast recovery e fast retrasmit) ad avere il comportamento illustrato dalla figura, rispondete alle domande seguenti giustificando sempre la risposta.



1) Identificare gli intervalli di tempo in cui opera lo slow start.
2) Identificate gli intervalli di tempo in cui opera il congestione avoidance.
3) Dopo il 16° turno di trasmissione, la perdita di segmenti viene rilevata da un triplice ACK duplicato oppure da un timeout ?
4) Dopo il 22° turno di trasmissione, la perdita di segmenti viene rilevata da un triplice ACK duplicato oppure da un timeout ?
5) Quale è il valore iniziale di *ssthresh* nel 1° turno di trasmissione ?
6) Quale è il valore iniziale di *ssthresh* nel 18° turno di trasmissione ?
7) Quale è il valore iniziale di *ssthresh* nel 24° turno di trasmissione ?
8) Durante quale turno di trasmissione viene inviato il 70° segmento ?
9) Ipotizzando il rilevamento della perdita di pacchetto dopo il 26° turno tramite un triplice ACK duplicato, quali saranno i valori dell'ampiezza della finestra di congestione e ssthresh ?
10) Supponete che venga usato TCP Tahoe (che implementa solamente il fast retransmit) e che si verifichi un evento di timeout al ventiduesimo round. Quanti pacchetti sono stati inviati dal 17° al 21° turno incluso ?

R: La risposta a questa domanda (multipla) presuppone la conoscenza della descrizione approfondita del controllo di congestione di TCP (sezione 3.7 libro di testo) e delle varie fasi: Slow start, congestion avoidance, differenze tra la versione di TCP che implementa sia *fast retransmit* e sia *fast recovery* (TCP Reno) e versione di TCP che implementa solamente il *fast retransmit* (TCP Tahoe).

1) La fase di TCP slowstart occorre negli intervalli [1,6] e [23,26].
2) La fase di TCP congestion occorre negli intervalli [6,16] e [17,22].

Per rispondere ai quesiti 1) e 2) si deve far riferimento al grafico e dal questo si identificano gli intervalli con "pendenze" tipiche dello SS e del CA.

3) Dopo il 16^0 turno di trasmissione, la perdita di segmenti viene riconosciuta mediante un triplice ACK duplicato. Se ci fosse stato invece un timeout la dimensione della finestra di congestione sarebbe stata ridotta ad 1.
4) Dopo il 22^0 turno di trasmissione, la perdita di segmenti viene riconosciuta mediante un timeout e quindi la dimensione della finestra di congestione sarebbe viene ridotta ad 1.
5) Il valore di ssthresh è inizialmente uguale a 32. Questo può essere dedotto osservando che è in corrispondenza di tale valore della *cwnd* avviene il passaggio tra lo stato SS e quello CA (nella figura viene evidenziato il valore della *cwnd*.
6) In occasione del 18^0 turno il valore della *ssthresh* è uguale a 21. Tale valore si ottiene dalla strategia di aggiornamento utilizzata dai meccanismi di gestione della congestion di TCP. In particolare il occasione del turno 16 *cwnd=42*, in questo istante viene rilevata una perdita di un pacchetto (mediante un triplice ACK duplicato) e questo ha come effetto di aggiornare il valore *ssthresh* alla metà del valore di *cwnd,* quindi *ssthresh=21* a partire dal turno 16 (fino al turno 18 non si verificano eventi che cambiano il valore di *ssthresh*).
7) In occasione del 24^0 turno il valore della *ssthresh* è uguale a 14. Tale valore si ottiene perché una perdita di un pacchetto viene rilevata al turno 22 (mediante timeout) con *cwnd=29.* Per effetto della perdita *ssthresh=cwnd/2=14* (si sceglie il floor). Fino al turno 24 non si verificano eventi che cambiano il valore di *ssthresh*.
8) Durante il turno 1 viene spedito il pacchetto 1, i pacchetti 2-3 vengono spediti durante il turno 2, i pacchetti da 4 a 7 vengono spediti durante il turno 3, i pacchetti da 8 a 15 vengono spediti durante il turno 4, i pacchetti da 16 a 31 vengono spediti durante il turno 5, i pacchetti da 32 a 63 vengono spediti durante il turno 6, i pacchetti da 64 a 96 vengono spediti durante il turno 7. Quindi il pacchetto 70 viene spedito durante il turno n. 7.
9) Il valore *ssthresh* viene assegnato alla metà della cwnd (che al turno 26 è uguale ad 8), quindi *ssthresh*=4. Il valore di cwdn, secondo i meccanismi di TCP viene posto a *ssthresh* + 3 (i tre pacchetti ricevuti che hanno generato i tre ACK duplicate). Quindi *cwnd=7*.
10) Al round 17, viene spedito 1 pacchetto; al round 18, 2 pacchetti; al round 19, 4 pacchetti; al round 20, 8 pacchetti; al round 21, 16 pacchetti. Il totale si ottiene quindi 1+2+4+8+16=31 pacchetti.

D: Perché http, ftp, smtp, e pop3 utilizzano come protocollo di trasporto TCP e non UDP ?
R: Per ragioni di affidabilità!

D: Descrivete il modo in cui il web caching riduce il ritardo di ricezione di un oggetto richiesto. Questo si verificherà per tutti gli oggetti richiesti oppure solo per alcuni di essi ? Perché ?
R: Se l'accesso ad un oggetto richiesto dal client avviene tramite il proxy cache allora si evita di doverlo richiedere al server di origine e quindi non si attende il tempo necessario per contattare tale server. Questo risparmio di tempo è si verificherà solamente per gli oggetti che sono presenti nel proxy cache e non per quelli che non ci sono.

D: Invio di una email all'indirizzo bianchi@di.unito.it: quali Resource Record verranno coinvolti? Elencare il tipo di RR ed il loro utilizzo.
R: Per prima cosa il client che spedisce deve recuperare l'indirizzo IP del server che riceve e per fare questo per prima cerca il resource record MX che contiene il nome del server di posta registrato per il dominio di.unito.it'. Il risultato di questa ricerca è il nome del server su cui è in esecuzione il server di posta elettronica per tale dominio (es. pianeta.di.unito.it). A seguire per ottenere l'indirizzo IP corrispondente a tale nome viene fatta una interrogazione ad DNS per ottenere un RR di tipo A (che contiene l'indirizzo IP corrispondente a pianeta.di.unito.it).

D: Considerate un client HTTP che voglia recuperare un documento web identificato da una determinata URL e che l'indirizzo IP del server non sia noto inizialmente. In questo scenario quali protocolli (sia di livello trasporto e sia di livello applicativo) sono coinvolti ?
R: Una url contiene un nome (a dominio) di questo tipo www.di.unito.it l'interrogazione che viene fatta al DNS ricerca un RR di tipo CNAME (canonical name) che permette di recuperare il nome canonico che corrisponde a www.di.unito.it (es. sun4server.di.unito.it). A seguire viene fatta un'interrogazione al DNS per ottenere un RR di tipo A (che contiene l'indirizzo IP corrispondente a sun4server.di.unito.it).

D: Supponiamo di selezionare, mediante un browser, un collegamento ipertestuale per ottenere una pagina web. L'indirizzo IP della URL relativo non si trova nella cache del DNS locale, e dunque, è necessaria una ricerca DNS per ottenerlo. Supponiamo di visitare n DNS server prima di ricevere l'indirizzo IP dal DNS. Queste visite successive richiedono dei tempi di rtt pari a RTT_1, RTT_2, …, RTT_n. Supponete che la pagina web contenga un solo oggetto consistente in un testo (di pochi caratteri). Denotiamo con RTT_0 il valore del rtt tra l'host locale ed i server web contenente l'oggetto. Quanto tempo intercorre tra l'attimo del clic sul collegamento e la ricezione dell'oggetto stesso da parte del client (assumere che il tempo di trasmissione dell'oggetto sia trascurabile).
R: Il tempo (totale) per ottenere l'indirizzo IP è pari a total amount of time to get the IP address is $RTT_1 + RTT_2 + \cdots + RTT_n$
Una volta recuperato l'indirizzo un tempo pari a RTT_O è necessario per stabilire la connessione TCP, successivamente un altro intervallo di tempo pari a RTT_O è necessario per richiedere e ricevere l'oggetto (di dimensioni piccole). Il tempo totale risulta essere $2RTT_o + RTT_1 + RTT_2 + \cdots + RTT_n$.

D: Considerate la seguente stringa di caratteri ASCII catturata da Wireshark quando il browser ha inviato un messaggio HTTP GET (questo è il messaggio di richiesta inviato dal client). I caratteri *<cr>* e *<lf>* sono il carattere di ritorno a capo e di nuova riga. Rispondete alle seguenti domande, indicando dove, nel messaggio HTTP GET seguente, trovate le risposte.

GET /cs453/index.html HTTP/1.1<cr><lf>Host: gaia.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax) <cr><lf>Accept: ext/xml, application/xml, application/xhtml+xml, text/html;q=0.9, text/plain;q=0.8,image/png,*/*;q=0.5<cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept-Encoding:zip, deflate<cr><lf>Accept-Charset: ISO-8859-1,utf-8;q=0.7,*; q=0.7<cr><lf>Keep-Alive: 300<cr><lf>Connection:keep-alive<cr><lf><cr><lf>

1) Quale è la URL del documento richiesto dal browser ?
2) Quale è la versione di HTTP che sta usando il browser ?
3) Il browser richiede una connessione persistente o non persistente ?
4) Quale è l'indirizzo IP dell'host sul quale è in esecuzione il browser ?
5) Che tipo di browser invia il messaggio ? Perché è necessario che in un messaggio di richiesta HTTP ci sia il tipo di browser ?

R:

(1) gaia.cs.umass.edu//cs453/index.html
(2) 1.1
(3) Persistente
(4) Questa è una domanda 'trabocchetto'. Questa informazione non è presente nel messaggio HTTP ma è possibile recuperarla dal datagram IP.
(5) Mozilla/5.0. Questa informazione è necessaria al server perché potrebbe avere per uno stesso oggetto versioni differenti adatte a diversi tipi di browser.

D: Il testo successivo mostra la risposta mandata dal server dopo aver ricevuto il messaggio HTTP GET della domanda. Rispondete alle seguenti domande, indicando dove, nel messaggio di replica a HTTP GET sottostante, trovate le risposte.

HTTP/1.1 200 OK<*cr*><*lf*>Date: Tue, 07 Mar 2008 12:39:45GMT<*cr*><*lf*>Server: Apache/2.0.52 (Fedora) <cr><lf>Last-Modified: Sat, 10 Dec2005 18:27:46 GMT<*cr*><*lf*>ETag: "526c3-f22-a88a4c80"<*cr*><*lf*>Accept-Ranges: bytes<*cr*><*lf*>Content-Length: 3874<*cr*><*lf*>Keep-Alive: timeout=max=100<*cr*><*lf*>Connection: Keep-Alive<*cr*><*lf*>Content-Type: text/html; charset=ISO-8859-1<*cr*><*lf*><*cr*><*lf*><!doctype html public "-//w3c//dtd html 4.0 transitional//en"><*lf*><html><*lf*><head><*lf*> <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"><*lf*> <meta name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT 5.0; U) Netscape]"><*lf*> <title>CMPSCI 453 / 591 /NTU-ST550A Spring 2005 homepage</title><*lf*></head><*lf*>
<*much more document text following here (not shown)*>

1) Il server è stato capace di trovare il documento ? In quale istante (di tempo) il documento è stato fornito ?
2) Quando è stato modificato l'ultima volta il documento ?
3) Quanti byte ci cono nel documento inviato ?
4) Che cosa sono i primi 5 byte del documento inviato al client ? Il server ha accettato la connessione permanente ?

R:

(1) Lo status code *200* ed il testo *OK* indicano che il server è stato in grado di recuperare il document richiesto. La replica è stata inviata il giorno *Tuesday, 07 Mar 2008 12:39:45 Greenwich Mean Time*.
(2) La data relative all'ultima modifica del document index.html è *Saturday 10 Dec 2005 18:27:46 GMT*.
(3) La dimensione del document è di *3874* bytes.
(4) I primi 5 byte del document sono : *<!doc*. Il server ha accettato la modalità connessione persistente, come indicato dal campo *Connection: Keep-Alive*.

D: Quale è la differenza tra MAIL FROM: in SMTP e From: nel messaggio di posta elettronica stesso ?
R: *MAIL FROM:* in un messaggio dal client SMTP al server identifica il mittente del messaggio di posta. *From:* all'interno di un messaggio di posta stesso non è un messaggio/commando SMTP ma solamente una delle righe nell'intestazione del messaggio stesso.

D: Come si indica la fine del corpo del messaggio in SMTP ? E in HTTP ? Potrebbe HTTP usare lo stesso metodo di SMTP ? Perché ?
R: SMTP usa una riga contenente un . (simbolo di punto) alla fine del body del messaggio.
HTTP utilizza "Content-Length header field" per indicare la lunghezza del body del messaggio.
No, HTTP non può utilizzare lo stesso metodo usato da SMTP perchè il messaggio HTTP potrebbe essere in binario (8 bit per bytes) mentre il body di un messaggio SMTP deve essere in format ASCII a 7 bit (ASCII nvt).

D: Considerate la distribuzione di un file di F bit a N peer, usando un'architettura client-server. Assumete una situazione in cui il server può trasmettere contemporaneamente a più peer, trasmettendo a ciascuno con velocità diverse, finché le velocità combinate (la somma di esse) non superi u_s.

(1) Supponete che $u_s/N \leq d_{min}$. Specificate uno schema di distribuzione che abbia un tempo di distribuzione di $N \cdot F/u_s$.
(2) Supponete che $u_s/N \geq d_{min}$. Specificate uno schema di distribuzione che abbia un tempo di distribuzione di F/ d_{min}.
(3) Concludete che il tempo di distribuzione minimo è in generale dato dal max $\{N \cdot F/u_s, F/ d_{min}\}$.

PS: d_{min} è la banda di download minima (il minimo tra le bande di download di tutti i peer).

R:

(1) Consideriamo uno schema di distribuzione in cui il server invia il file ad ogni client, in parallelo e per ognuno il tasso con cui viene inviato il file è pari a u_s/N. Notare che questo tasso è inferiore al download rate del client per ipotesi ($u_s/N \leq d_{min}$) e quindi ogni client riceverà il file ad un tasso pari a u_s/N. Da questo si deduce che ogni client riceverà l'intero file di dimensione F in un tempo pari a $F/(u_s/N)$ da cui si ricava che il tempo di distribuzione è uguale $N \cdot F/u_s$.
(2) Consideriamo uno schema di distribuzione in cui il server invia ad ogni client il file, in parallel, ad un tasso pari d_{min}. Notare che il tasso aggregato è pari a $N \cdot d_{min}$ e questo per ipotesi è inferiore a tasso del link del server ($u_s/N \geq d_{min}$). Poichè ogni client riceve ad un tasso pari a d_{min}, il tempo necessario ad ogni client per ricevere l'intero file è F/ d_{min}. Questo è il tempo di distribuzione del file.
(3) Dalle considerazioni introdotte nella Sezione 2.6 sappiamo che

$$D_{CS} \geq \max \{NF/u_s, F/d_{\min}\}$$ **(Equazione 1).**

Se assumiamo che $u_s/N \leq d_{\min}$, allora dall'Equazione 1 abbiamo $D_{CS} \geq NF/u_s$. Dalla derivazione (1) abbiamo che $D_{CS} \leq NF/u_s$, combinando questi due risultati si ottiene che:

$D_{CS} = NF/u_s$ quando $u_s/N \leq d_{\min}$. (**Equazione 2**).

Allo stesso modo si può dimostrare che:

$D_{CS} = F/d_{\min}$ quando $u_s/N \geq d_{\min}$ (**Equazione 3**).

Combinando l'Equanzione 2 e l'Equazione 3 si ottiene il risultato richiesto.