# *Fondamenti di Internet e Reti*

#### Proff. A. Capone, M. Cesana, I. Filippini, G. Maier

**Prova in itinere – 5 Maggio 2016**

|  |  |
| --- | --- |
| **Cognome** |  |
| **Nome** |  |
| **Matricola** |  |

##### Tempo complessivo a disposizione per lo svolgimento: 1h45m

**Usare lo spazio dopo ogni Esercizio/Quesito per la risposta**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Es1 (9pt)** | **Es2 (9 pt)** | **Ques (9 pt)** | **Lab (6pt)** |
|  |  |  |  |

## - Esercizio (10 punti)

a) Una connessione TCP tra l’*host* A e l’*host* B nelle rete in figura è caratterizzata dai seguenti parametri: lunghezze di *header* e *ack* trascurabili, *link* bidirezionali simmetrici, *MSS* = 1250 B, *RCWND* >> *CWND, SSTHRESH* = 8 MSS.

Si calcoli il tempo necessario a trasferire un file di 30 kB.

Si ripeta il calcolo assumendo un file di 55 kB.

*(nota la soluzione è la stessa se si scambiano di posto il link 1 con il link 2)*

b) Si assuma A sia un server http e B un client http. Occorre trasferire un documento base html di Lhtml = 100 kB base e 9 immagini di Lobj = 5 MB in presenza di 1 flusso interferente tra C e D. Si calcoli il tempo necessario assumendo il RTT calcolato nel punto precedente e un ritmo medio di trasmissione (Rhtml e Robj) pari al valore di condivisione equa delle risorse (capacità del link 2 diviso il numero di flussi che lo attraversano) nel caso di connessione http persistente e non persistente (con trasmissione in parallelo delle immagini).



a)

caso 1)

slow start: (1) – (2) – (4) – (8)

cong. avoidance: (9)

caso 2)

slow start: (1) – (2) – (4) – (8)

cong. avoidance: (9)

continua (link 2): (20)

b)

Persistente:

Non-persistente:

## 2 - Esercizio (8 punti)

In una rete a commutazione di pacchetto al tempo t=0 sono presenti 6 pacchetti in S diretti rispettivamente alle seguenti destinazioni: A, A, B, B, C, D. Calcolare il tempo di ricezione di ciascuno dei pacchetti assumendo che i pacchetti abbiamo le seguenti dimensioni: pacchetti verso A, LA=1250B; pacchetti verso B, LB=250B; pacchetti verso C, LC=1250B; pacchetti verso D, LC=1250B.

## 

## Quesiti (9 punti)

**Q1**

Si illustri la sindrome della finestra di Silly in TCP (lato trasmettitore e lato ricevitore) e come viene risolto.

**Silly window syndrome - lato ricevitore:**

**- Il ricevitore svuota lentamente il buffer di ricezione**

**- Invia segmenti con finestra molto piccola**

**- Il trasmettitore invia segmenti corti con molto overhead**

**Soluzione (algoritmo di Clark)**

**Il ricevitore “mente” al trasmettitore indicando una finestra nulla sino a che il suo buffer di ricezione non si è svuotato per metà o per una porzione almeno pari al MSS**

**Silly window syndrome - lato trasmettitore:**

**- L’applicazione genera dati lentamente**

**- Invia segmenti molto piccoli man mano che vengono prodotti**

**Soluzione (algoritmo di Nagle)**

**- Il TCP sorgente invia la prima porzione di dati anche se corta**

**- Gli altri segmenti vengono generati e inviati solo se**

**a) Il buffer d’uscita contiene dati sufficienti a riempire un MSS**

**b) oppure, quando si riceve un acknowledgement per il segmento precedente.**

**Q2**

Si completi la figura in accordo alle regole del protocollo Go-back-N. Si inseriscano i valori di SN ed RN, si indichino gli istati di accettazione delle trame corrette e in sequenza.

****

**Q3**

Da un host viene eseguito il comando dig due volte consecutive come riportato sotto. Si illustri il significato delle risposte ottenute.

host:~ acapone$ dig www.yahoo.com +noall +answer

; <<>> DiG 9.8.3-P1 <<>> www.yahoo.com +noall +answer

;; global options: +cmd

www.yahoo.com. 24 IN CNAME fd-fp3.wg1.b.yahoo.com.

fd-fp3.wg1.b.yahoo.com. 53 IN A 46.228.47.115

fd-fp3.wg1.b.yahoo.com. 53 IN A 46.228.47.114

host:~ acapone$ dig www.yahoo.com +noall +answer

; <<>> DiG 9.8.3-P1 <<>> www.yahoo.com +noall +answer

;; global options: +cmd

www.yahoo.com. 217 IN CNAME fd-fp3.wg1.b.yahoo.com.

fd-fp3.wg1.b.yahoo.com. 59 IN A 46.228.47.114

fd-fp3.wg1.b.yahoo.com. 59 IN A 46.228.47.115

L’indirizzo simbolico www.yahoo.com corrisponde all’alias fd-fp3.wg1.b.yahoo.com e quest’ultimo a due indirizzi IPv4 46.228.47.114/115 che vengono restituiti in modo alternato per effettuare un bilanciamento di carico.

## Laboratorio (6 punti)

**Q1**

Si voglia mandare il messaggio in figura collegandosi manualmente tramite telnet al server smtp.polimi.it. Si indichino i comandi da inviare.

host:~ telnet smtp.polimi.it 25

HELO polimi.it

…

MAIL FROM: studente@polimi.it

…

RCPT TO: antonio.capone@polimi.it

…

DATA

…

From: studente@polimi.it

To: antonio.capone@polimi.it

Subject: Primo compitino

Questo è il primo compitino di FIR.

Speriamo bene.

.

…

QUIT

…

**Q2**

Si consideri i seguenti codici Python per client e server. Il codice contiene un errore. Quale? Come lo si può correggere modificando il server?

**Client**

from socket import \*

Name = 'ilmioserver.gratis'

Port = 15001

Socket = socket(AF\_INET, SOCK\_STREAM)

Socket.connect((Name, Port))

for count in range(1,11):

Socket.send('Ho una domanda')

risposta = Socket.recv(1024)

print 'From Server:', risposta

Socket.send('.')

Socket.close()

**Server**

from socket import \*

sPort = 15001

sSocket = socket(AF\_INET, SOCK\_STREAM)

sSocket.bind(('', sPort))

sSocket.listen(1)

while True:

cSocket, clientAddress = sSocket.accept()

domanda = cSocket.recv(1024)

if domanda == 'Ho una domanda':

risposta = 'Non ho risposte'

cSocket.send(risposta)

cSocket.close()

**Il client invia per dieci volte la domanda ma il server chiude la connessione dopo aver inviato la prima risposta.**

**Possibile modifica:**

from socket import \*

sPort = 15001

sSocket = socket(AF\_INET, SOCK\_STREAM)

sSocket.bind(('', sPort))

sSocket.listen(1)

while True:

cSocket, clientAddress = sSocket.accept()

while True:

domanda = cSocket.recv(1024)

if domanda == 'Ho una domanda':

risposta = 'Non ho risposte'

else:

break

cSocket.send(risposta)

cSocket.close()

**Q3**

Si consideri il server in python scritto sotto. Vengono aperti in parallelo una dopo l’altra 4 connessioni da client verso il server. Come vengono gestiti e cosa vedono i quattro client assumendo che ciascuno di essi cerchi di inviare prima una stringa ‘xxx’ e poi, attendendo 10 s, una stringa ‘.’ ?

from socket import \*

sPort = 15001

sSocket = socket(AF\_INET, SOCK\_STREAM)

sSocket.bind(('', sPort))

sSocket.listen(2)

client = 1

while True:

cSocket, clientAddress = sSocket.accept()

while True:

domanda = cSocket.recv(1024)

if domanda == '.':

break

risposta = ''

for count in (0,client):

risposta = risposta + domanda + ' '

cSocket.send(risposta)

cliente = client + 1

cSocket.close()

**Il primo client viene servito subito e riceve una risposta uguale alla stringa inviata ‘xxx’ e poi la connessione viene chiusa.**

**Il secondo cliente viene servito dopo il primo e riceve come risposta due volte la stringa ‘xxx xxx’ e poi la connessione viene chiusa.**

**Il terzo cliente viene servito dopo il secondo e riceve come risposta tre volte la stringa ‘xxx xxx xxx’ e poi la connessione viene chiusa.**

**La connessione del quarto client viene rifiutata.**