Prova in itinere – 5 Maggio 2016

Cognome	
Nome	
Matricola	

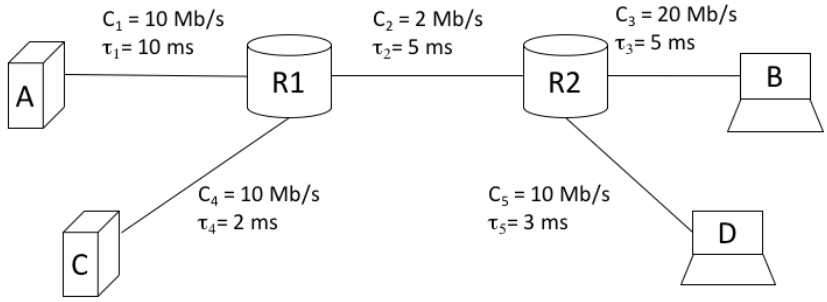
Tempo complessivo a disposizione per lo svolgimento: 1h45m
Usare lo spazio dopo ogni Esercizio/Quesito per la risposta.

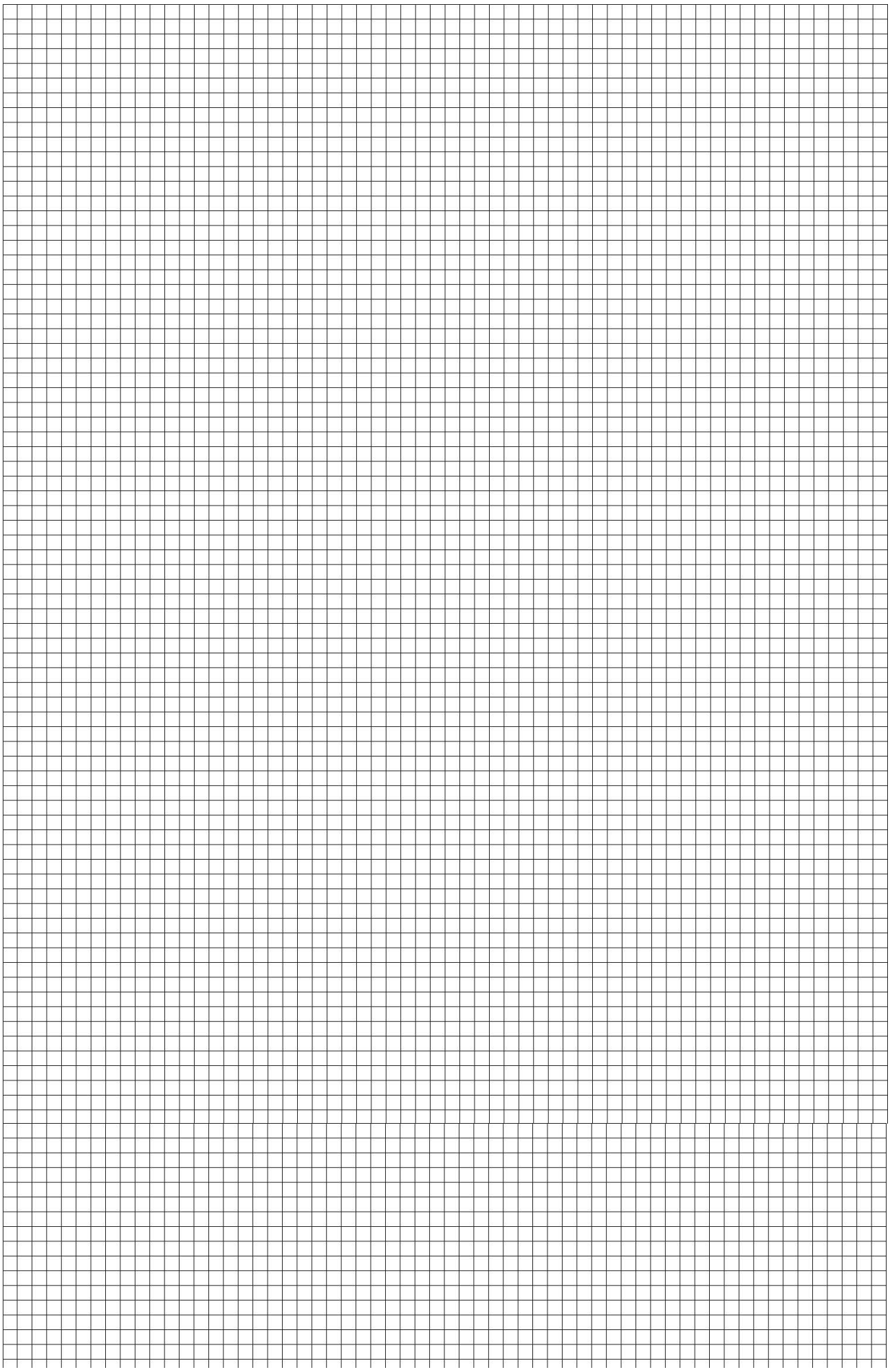
Es1 (9pt)	Es2 (9 pt)	Ques (9 pt)	Lab (6pt)

1- 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)

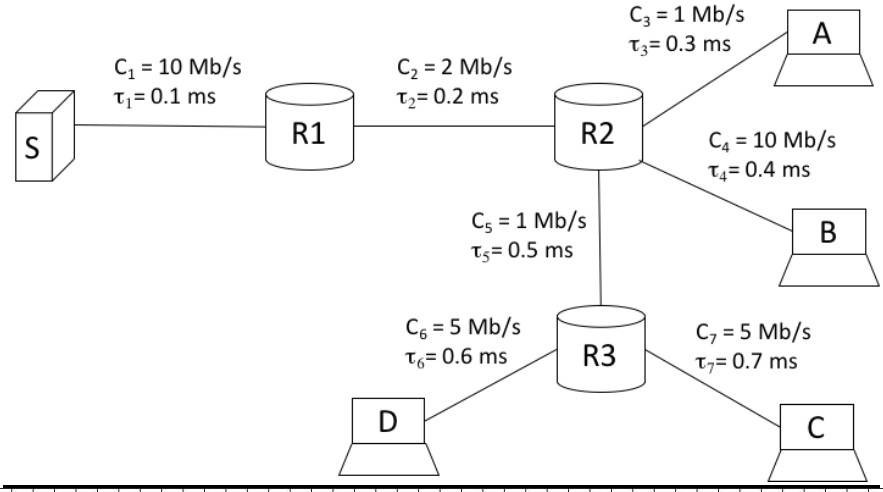
b) Si assuma A sia un server http e B un client http. Occorre trasferire un documento di 100 kB base e 9 immagini di 5 MB in presenza di 1 flusso interferente tra C e D. Si calcoli il tempo necessario assumendo il RTT calcolato nel punto precedente e un ritmo medio di trasmissione pari al valore di condivisione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





2 - Esercizio (8 punti)

In una rete a commutazione di pacchetto al tempo t=0 sono presenti 8 pacchetti in S diretti rispettivamente alle seguenti destinazioni: A, A, B, B, C, C, D, D. Calcolare il tempo di ricezione di ciascuno dei pacchetti assumendo che i pacchetti abbiamo le seguenti dimensioni: pacchetti verso A, L_A=1250B; pacchetti verso B, L_B=250B; pacchetti verso C, L_C=1250B; pacchetti verso D, L_C=1250B.



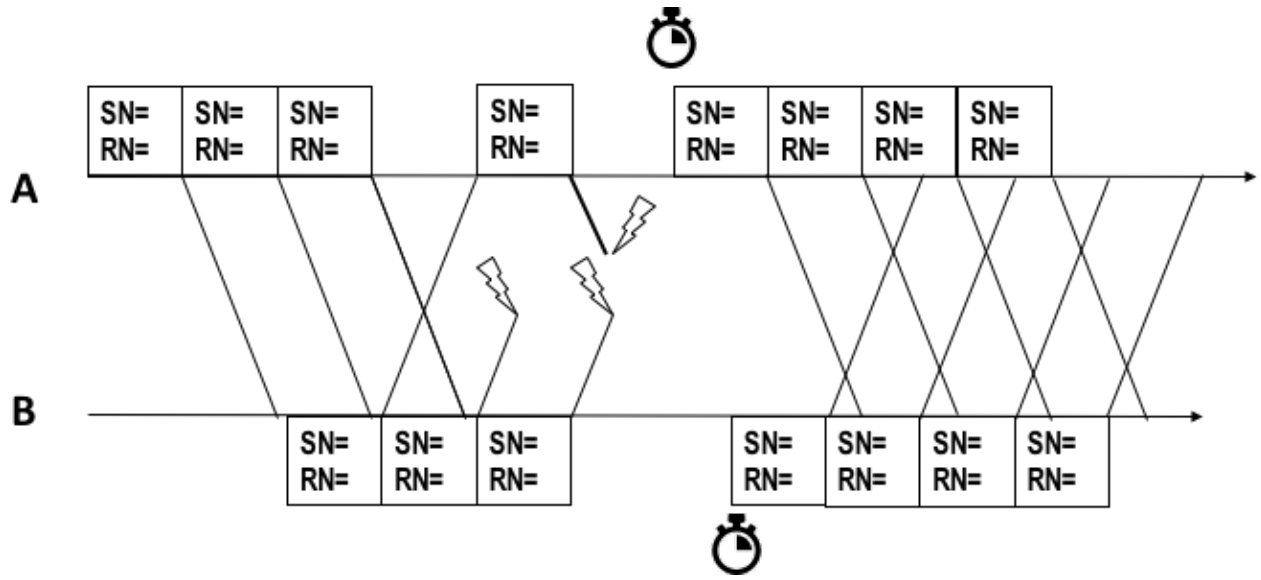
Quesiti (9 punti)

Q1

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

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

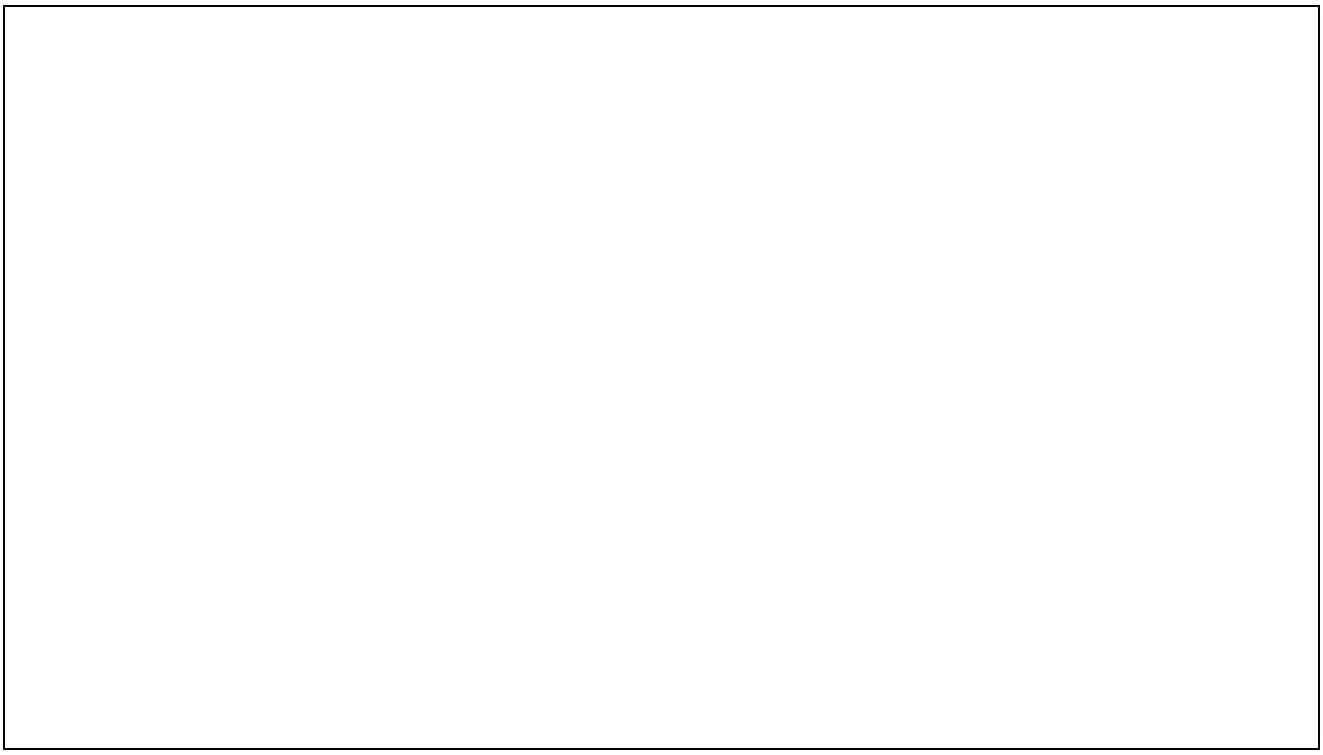
Da un host viene eseguito il comando dig due volte consecutive come riportato sotto. Si illustri il significato delle risposte ottenute.

```
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
```



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:~
```

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()
```

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()
```

Codice esercizi laboratorio

UDP client

```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message, (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```

UDP server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    print "Datagram from: ", clientAddress
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

UDP error management

```
from socket import *
serverName = 'localhost'
serverPort = 12001
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.settimeout(5)
message = raw_input('Input lowercase sentence:')
try:
    clientSocket.sendto(message, (serverName, serverPort))
    modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
    # in case of error blocks forever
    print modifiedMessage
except error, v:
    print "Failure"
    print v
finally:
    clientSocket.close()
```

TCP client

```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

TCP server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
```

```
print 'The server is ready to receive'
while True:
    connectionSocket,  clientAddress = serverSocket.accept()
    print "Connection form: ", clientAddress
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP client persistent

```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
while True:
    sentence = raw_input('Input lowercase sentence ( . to stop):')
    clientSocket.send(sentence)
    if sentence == '.':
        break
    modifiedSentence = clientSocket.recv(1024)
    print 'From Server:', modifiedSentence
clientSocket.close()
```

TCP server persistent

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
while True:
    print 'The server is ready to receive'
    connectionSocket,  clientAddress = serverSocket.accept()
    print "Connection form: ", clientAddress
    while True:
        sentence = connectionSocket.recv(1024)
        if sentence == '.':
            break
        capitalizedSentence = sentence.upper()
        connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP auto client

```
from socket import *
import time
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
for a in range(100):
    clientSocket.send('A')
time.sleep(1)
clientSocket.send('.')
#clientSocket.recv(1024)
clientSocket.close()
```

TCP auto server

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
```

```
while True:
    print 'The server is ready to receive'
    connectionSocket,  clientAddress = serverSocket.accept()
    print "Connection form: ", clientAddress
    while True:
        sentence = connectionSocket.recv(1024)
        if sentence == '.':
            break
        print len(sentence)
#        connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP server thread

```
from socket import *
import thread
def handler(connectionSocket):
    while True:
        sentence = connectionSocket.recv(1024)
        if sentence == '.':
            break
        capitalizedSentence = sentence.upper()
        connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
while True:
    print 'The server is ready to receive'
    newSocket,  addr = serverSocket.accept()
    thread.start_new_thread(handler, (newSocket,))
```