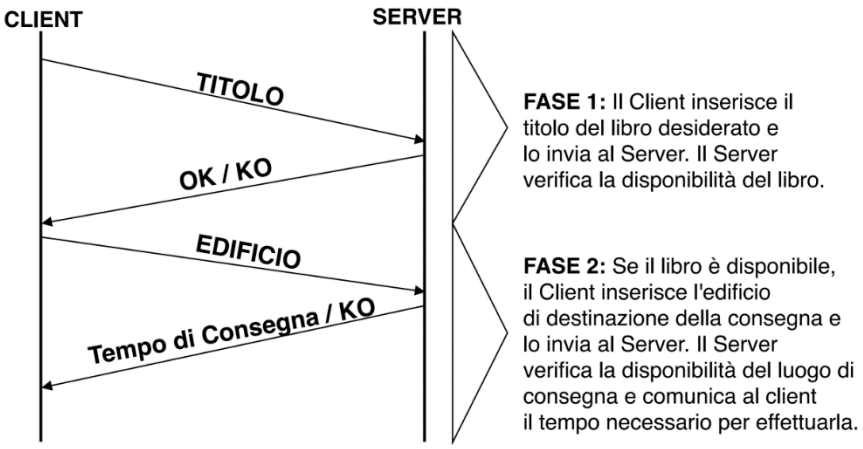
Laboratorio - 17 Aprile 2019

Cognome	
Nome	
Matricola	

Nell'ambito dei progetti di sviluppo del 5G, il Politecnico sta testando il servizio ***DeLibro*** che permette di effettuare prestiti di libri con consegna automatica attraverso self-driving robots. Il codice sottoriportato rappresenta una versione semplificata della richiesta di prenotazione di un libro attraverso DeLibro. L'utente inserisce il titolo del libro che vuole prendere in prestito e l'edificio in cui vuole ritirare il libro, il server risponderà comunicando il tempo necessario per effettuare la consegna del libro richiesto.
Il diagramma in figura mostra il protocollo applicativo.



Script Client:

```
from socket import *
serverAddr = "localhost"
serverPort = 20191
clSocket = socket(AF_INET, SOCK_STREAM)
clSocket.connect((serverAddr, serverPort))
titolo = input("Inserisci il titolo del libro: ")

clSocket._____
_____
while resp.decode('utf-8') != 'OK':
   print(titolo, "non disponibile")
   titolo = input("inserisci il titolo del libro: ")

   clSocket._____
   resp = clSocket.recv(1024)
edificio = input("In quale edificio del Politecnico vuoi ricevere il libro? ")

clSocket._____
resp = clSocket.recv(1024)
if _____:
   print('Consegna non disponibile')
else:
   print('Riceverai il libro', resp.decode('utf-8'))
clSocket.close()
```

Script Server:

```
from socket import *
def libro_disponibile(book):
   return True
serverPort = 20191
edifici_disponibili = ['Edificio 27', 'B8', 'BL27']

_____

_____
serverSocket.listen(1)
while True:
   connSocket, clAddr = serverSocket.accept()
   titolo = connSocket.recv(1024)
   disp = libro_disponibile(titolo.decode('utf-8'))
   while not disp:
      _____
      titolo = connSocket.recv(1024)
      disp = libro_disponibile(titolo.decode('utf-8')
   _____
   edificio_richiesto = connSocket.recv(1024)
   if edificio_richiesto.decode('utf-8') in edifici_disponibili:
      connSocket.send('tra 60 minuti'.encode('utf-8'))
   else:
      _____
   connSocket.close()
```

1) Completare il codice del client e del server. (2 punti)
2) Che protocollo di trasporto viene utilizzato? Da cosa lo si capisce? (1 punto)

3) Scrivere l'output del client quando viene effettuata una corretta prenotazione del libro Reti di Calcolatori e Internet: (3 punti)

```
fir@delibro_client:~$ python3.4 client_esame.py
```

_____

_____

_____

Codice socket programming

UDP client
```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = input('Inserisci lettere:')
clientSocket.sendto(message.encode('utf-8'), (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print (modifiedMessage.decode('utf-8'))
clientSocket.close()
```

UDP server
```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "Il server è pronto ad ricevere"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    print ("Datagram from: ", clientAddress)
    message = message.decode('utf-8')
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage.encode('utf-8'), clientAddress)
```

UDP error management
```
from socket import *
serverName = 'localhost'
serverPort = 12001
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.settimeout(5)
message = input('Input lettere:')
try:
    clientSocket.sendto(message.encode('utf-8'), (serverName, serverPort))
    modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
    # in case of error blocks forever
    print modifiedMessage.decode('utf-8')
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 = input('Input lettere:')
clientSocket.send(sentence.encode('utf-8'))
modifiedSentence = clientSocket.recv(1024)
print (modifiedSentence.decode('utf-8'))
clientSocket.close()
```

TCP server
```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print ('Il server è pronto ad ricevere')
while True:
    connectionSocket,  clientAddress = serverSocket.accept()
    print ('Connesso con: ', clientAddress)
    sentence = connectionSocket.recv(1024)
    sentence = sentence.decode('utf-8')
    capitalizedSentence = sentence.upper()
```

```
    connectionSocket.send(capitalizedSentence.encode('utf-8'))
    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 = input('Inserisci lettere ( . per fermare):')
    clientSocket.send(sentence.encode('utf-8'))
    if sentence == '.':
        break
    modifiedSentence = clientSocket.recv(1024)
    print ('Dal Server: ', modifiedSentence.decode('utf-8'))
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 ('Il server è pronto ad ricevere')
    connectionSocket,  clientAddress = serverSocket.accept()
    print ('Connesso con: ', clientAddress)
    while True:
        sentence = connectionSocket.recv(1024)
        sentence = sentence.decode('utf-8')
        if sentence == '.':
            break
        capitalizedSentence = sentence.upper()
        connectionSocket.send(capitalizedSentence.encode('utf-8'))
    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)
        sentence = sentence.decode('utf-8')
        if sentence == '.':
            break
        capitalizedSentence = sentence.upper()
        connectionSocket.send(capitalizedSentence.encode('utf-8'))
    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 ('Il server è pronto ad ricevere')
    newSocket,  addr = serverSocket.accept()
    thread = Thread(target=handler, args=(newSocket,))
    thread.start()
```