COLLEGE OF ENGINEERING TRIVANDRUM

NETWORKING PROGRAMMING LAB

Lab Exam Report - Q2B

NIKHIL JOJI
Roll No: 46
TVE16CS047

25 April 2019

Contents

1 Problem 2

1.1 Problem Statement

Write a program to implement a basic publish-subscribe communication model using TCP sockets. Your module should contain code for both server and client. A client can subscribe to one or more channels/topics at a time and it should only receive messages from channels that it has subscribed. Your server must be capable of handling multiple clients at a time. The message to broadcast, i.e (topic, message) is read from the standard input. Note that both server and client should run until explicitly interrupted by the user

2 Theory

2.1 TCP

Transmission Control Protocol is a protocol that is used in networking to communicate between a server and a client. Here there are handshakings done to secure the transmission. 3 way handshake to establish a connection and 2 way handshake to terminate a connection (between server and client). This reduces packet loss and improves transmission of data packets.

2.2 Sockets

Sockets are pathways that allow communication between multiple processes. Sockets are made and they listen on a particular port on a given ip. Sockets facilitate Inter-Process Communication.

2.3 Threads

A thread is a path of execution within a process (program). There can be more than one threads in a process. Threads of a process share memory inside the process, unlike between multiple processes. Threads run simultaneously in memory.

3 Implementation

The program required to have a server that will let many clients to connect simultaneously and listen to the messages that are sent to their corresponding subscribed topics.
Here we have a server that calls a function named client that infinitely listens for connections from clients. If connected the clients send the topics to subscribe and then the corresponding conn objects are added to the list under each topic. The Server has a thread running parallely that is always ready to broadcast the message under a given topic. Once the data to send is available the server sends it to all clients subscribed under the given topic.

The client asks the user to subscribe to any of the 3 topics and then sends the information to the server. After this data is sent is continuously listens to the socket to see if any message is being broadcast under its subscribed topics. The server program named server_Q2B_new.py is run first using the command:

python server_Q2B_new.py

The client program named client_Q2B_new.py is run then on using the command:

python server_Q2B_new.py

Multiple Client programs can be run simultaneously with one server program running.

4 Program

4.0.1 Server Program

```
import sys
import socket
from thread import *
import thread

ip = "127.0.0.1" #ip for socket
port = 6000      #port for socket

news=[]          #lists that store socket ids
sports=[]        #according to subscribed topics
entertainment=[]

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((ip,port))
s.listen(10)

def client(): #function that accepts client connection
        while True:  #infinite loop to keep adding clients
                conn, addr = s.accept()
                conn.send("Connection attempt has reached server")
                topic1 = conn.recv(1024)
                print " has subscribed to " + topic1

                if (topic1 == "news"):
                        news.append(conn)
                elif (topic1 == "sports"):
                        sports.append(conn)
                elif (topic1 == "entertainment"):
```

```
                entertainment.append(conn)

            conn.send("topic1 received")
            topic2 = conn.recv(1024)

            if topic2!='0':
                print " has subscribed to " + topic2

                if (topic2 == "news"):
                    news.append(conn)
                elif (topic2 == "sports"):
                    sports.append(conn)
                elif (topic2 == "entertainment"):
                    entertainment.append(conn)

            conn.send("topic2 received")
            topic3 = conn.recv(1024)

            if topic3!='0':
                print " has subscribed to " + topic3

                if (topic3 == "news"):
                    news.append(conn)
                elif (topic3 == "sports"):
                    sports.append(conn)
                elif (topic3 == "entertainment"):
                    entertainment.append(conn)

            conn.send("topic3 received")

def broadcast():  #function that broadcasts msgs
        while True:   #according to topic
            article = raw_input("Enter article
            to broadcast with topic as first word ")
            #getting topic and data from input
            g = article.split()
            topic = g[0]
            article = article.strip(g[0])
            print article
            if (topic == "news"):
                for x in news:
                    #print x
                    x.send(article)
            elif (topic == "sports"):
                for x in sports:
```

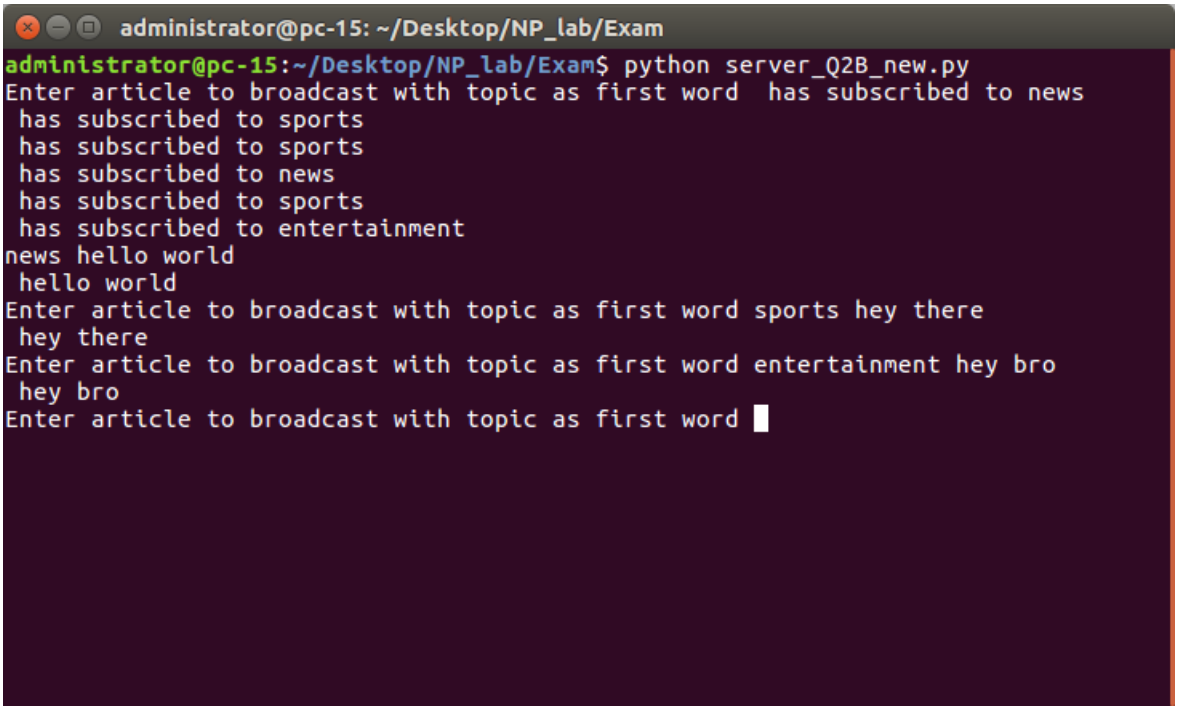
```
                        x.send(article)
                elif (topic == "entertainment"):
                        for x in entertainment:
                                x.send(article)

thread.start_new_thread(broadcast,()) #do broadcast in separate thread
client()

s.close()
```

4.0.2 Client Program

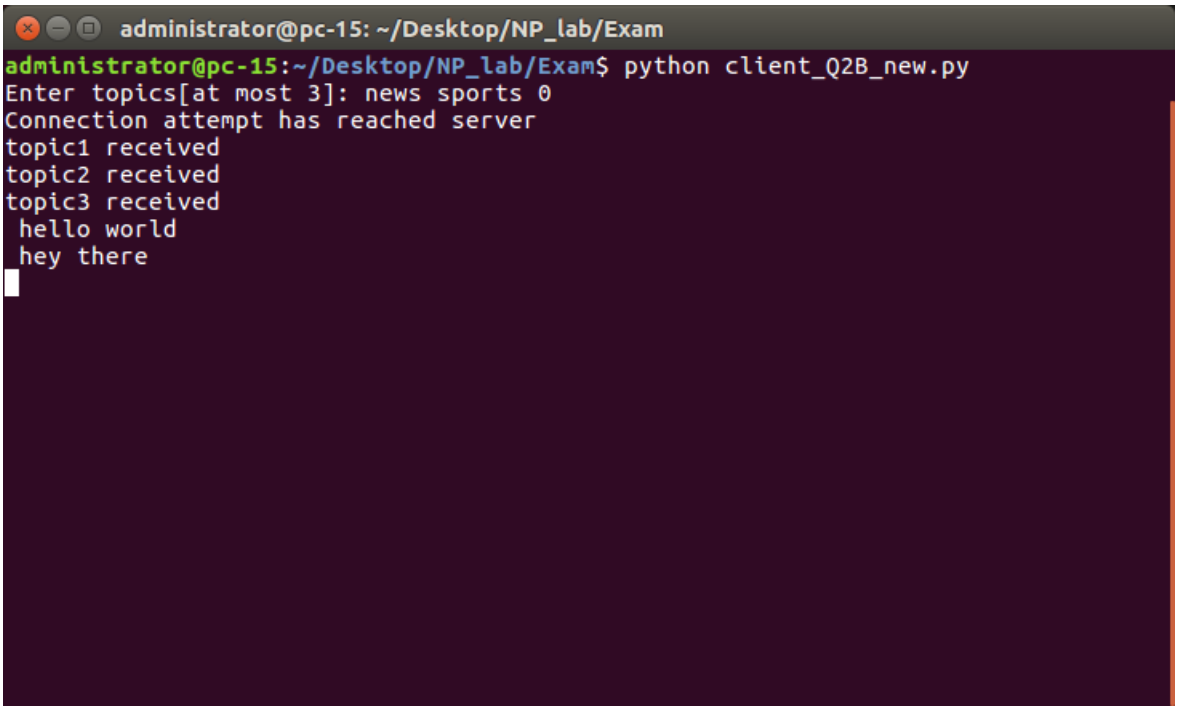
```
import sys
import socket
from thread import *
import thread

ip = "127.0.0.1"  #ip for socket
port = 6000       #port for socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((ip,port))
#connecting to server through socket
topic = raw_input("Enter topics[at most 3]: ")
g = topic.split()

print s.recv(1024)
s.send(g[0])

print s.recv(1024)
s.send(g[1])

print s.recv(1024)
s.send(g[2])

print s.recv(1024)

while True:  #listening for msg infinitely from server
            data = s.recv(1024)
            print  data
#s.close()
```

5 Output

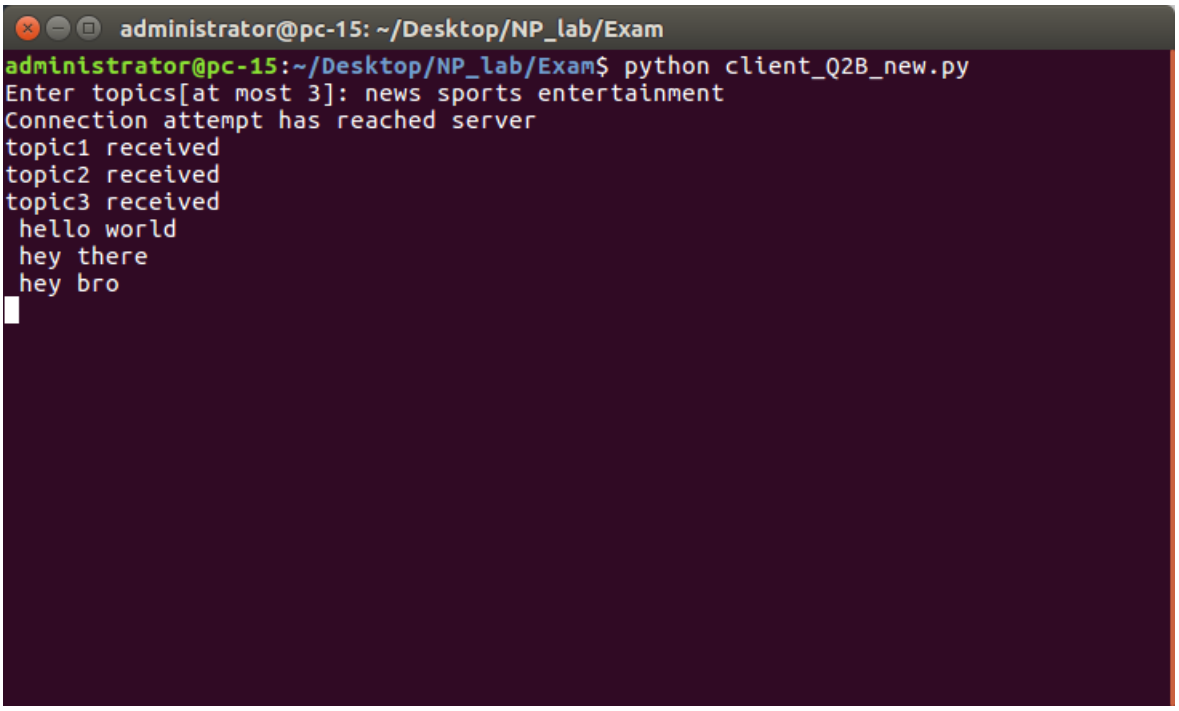
5.1 Server



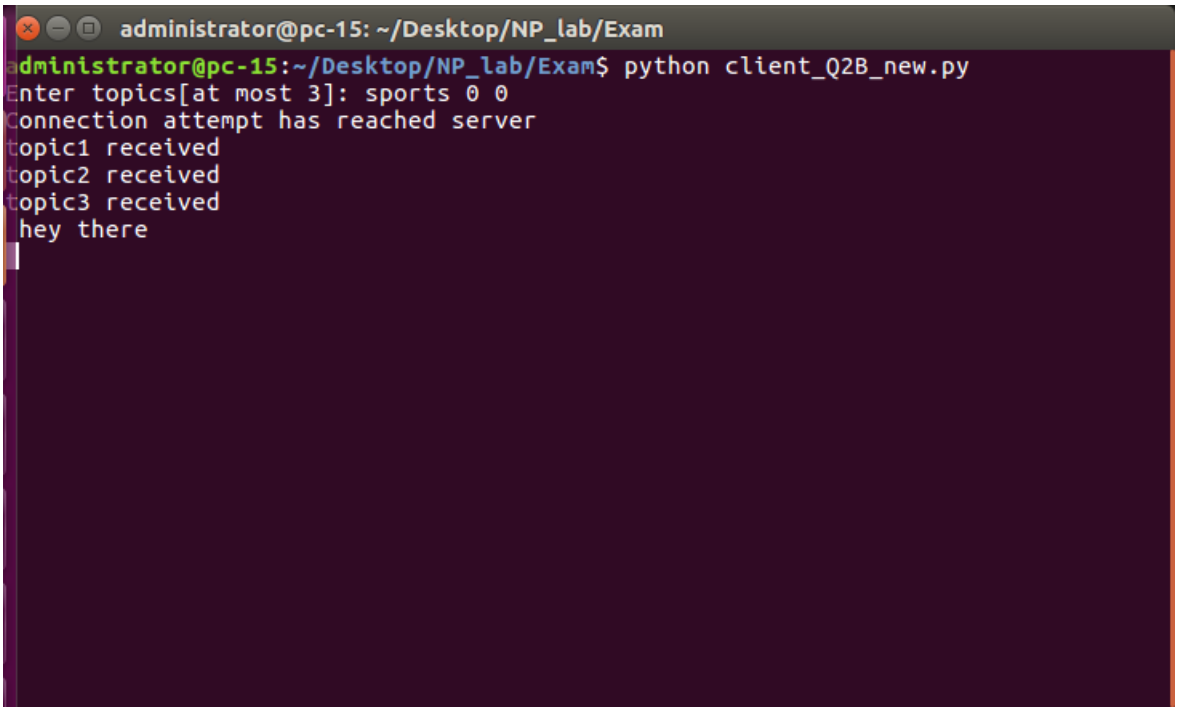
5.2 Client 1



5.3 Client 2



5.4 Client 3



6 Result

A Program to implement a basic publish-subscribe communication model using TCP sockets was written in python and run on Ubuntu 16.04 LTS system.