**PSG COLLEGE OF TECHNOLOGY**

**DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES**

**COMPUTER NETWORKS LAB**

20XC46 COMPUTER NETWORKS LAB

&

### 20XW46 COMPUTER NETWORKS AND TCP/IP LAB

20pw35-Swetha Muralidharan

1. Build a simple client-server system, where you use the client to chat with a dummy server. The protocol between the client and server is as follows.

* The server is first started on a known port.
* The client program is started (server IP and port are provided on the command line).
* The client connects to the server, and then asks the user for input. The user types his message on the terminal (e.g., "Hi", "Bye", "How are you"). The user's input is sent to the server via the connected socket.
* The server reads the user's input from the client socket. If the user has typed "Bye" (without the quotes), the server must reply with "Goodbye". For any other message, the server must reply with "OK".
* The client then reads the reply from the server, and checks that it is accurate (either OK" or "Goodbye").
* If the user had typed "Bye", and the server replied with a "Goodbye" correctly, the client quits. Otherwise, the client asks the user for the next message to send to the server.

Sample output:

Connected to server

Please enter the message to the server: Hello

Server replied: OK

Please enter the message to the server: Hi

Server replied: OK

Please enter the message to the server: Bye

Server replied: Goodbye

**Client :**

import socket

c = socket.socket()

try:

c.connect((socket.gethostname(),9999))

name = input("Enter your name: ")

c.send(name.encode())

while True:

s = input("Enter your message: ")

c.send(s.encode())

reply = c.recv(1024).decode()

print(f"Server replied: {reply}")

if reply == "Goodbye":

break

c.close()

except ConnectionRefusedError:

print("No connection could be made because the target machine actively refused it")

**Server:**

import socket

import time

s = socket.socket()

host = socket.gethostname()

s.bind((host,9999))

s.listen(1)

print('waiting for connections!')

c,addr = s.accept()

name = c.recv(1024).decode()

print(f"connected with {name}")

while True:

msg = c.recv(1024).decode()

if msg:

if msg == "bye":

c.send("Goodbye".encode())

else:

c.send("OK".encode())

else:

print(f"Connection closed for {name}")

c.close()

break

s.close()

2. The problem is to implement a client - server user-level application using sockets . Server accepts strings from clients and replies with reverse strings. For example, when client sends “IITHYD”, Server replies with “DYHTII”. Both server and client have to output both sending & receiving strings on the terminal. The server and client processes should be run on different machines.

**Client**

import socket

c = socket.socket()

try:

c.connect((socket.gethostname(),9999))

name = input("Enter your name: ")

c.send(name.encode())

while True:

s = input("Enter your message: ")

if s == "quit":

break

c.send(s.encode())

reply = c.recv(1024).decode()

print(f"Server replied: {reply}")

c.close()

print(f'{name} closed connection')

except ConnectionRefusedError:

print("No connection could be made because the target machine actively refused it")

**Server**

import socket

import time

s = socket.socket()

host = socket.gethostname()

s.bind((host,9999))

s.listen(1)

print('waiting for connections!')

c,addr = s.accept()

name = c.recv(1024).decode()

print(f"connected with {name}")

while True:

msg = c.recv(1024).decode()

if msg:

c.send(msg[::-1].encode())

else:

print(f"Connection closed for {name}")

c.close()

break

3.Design an echo server .Echo server is a server that echoes back all data it receives to a client that sent it.

Sample output:

Enter the message to be sent: This is Client

Message from Server: This is Client

Do you want to continue: Y

Enter the message to be sent: Connection Established

Message from Server: Connection Established

Do you want to continue: n

**Server**

import socket

import time

s = socket.socket()

host = socket.gethostname()

s.bind((host,9999))

s.listen(1)

print('waiting for connections!')

c,addr = s.accept()

name = c.recv(1024).decode()

print(f"connected with {name}")

while True:

msg = c.recv(1024).decode()

if msg:

c.send(msg.encode())

else:

print(f"Connection closed for {name}")

c.close()

break

**Client**

import socket

c = socket.socket()

try:

c.connect((socket.gethostname(),9999))

name = input("Enter your name: ")

c.send(name.encode())

while True:

s = input("Enter the message to be sent: ")

if s == "quit":

break

c.send(s.encode())

reply = c.recv(1024).decode()

print(f"Message from Server: {reply}")

c.close()

print(f'{name} closed connection')

except ConnectionRefusedError:

print("No connection could be made because the target machine actively refused it")

4. The Binary Encoding Server: Write a server program for TCP using Python to do the following: a. Server returns the binary value of the text sent by the client. Example: for a text string “comnetsii”, the client should receive “01100011 01101111 01101101 01101110 01100101 01110100 01110011 01101001 01101001”

**Server**

import socket

import time

s = socket.socket()

host = socket.gethostname()

s.bind((host,9999))

s.listen(1)

print('waiting for connections!')

c,addr = s.accept()

name = c.recv(1024).decode()

print(f"connected with {name}")

while True:

msg = c.recv(1024).decode()

if msg:

reply = ' '.join(format(ord(x), 'b') for x in msg)

c.send(reply.encode())

else:

print(f"Connection closed for {name}")

c.close()

break

**client**

import socket

c = socket.socket()

try:

c.connect((socket.gethostname(),9999))

name = input("Enter your name: ")

c.send(name.encode())

while True:

s = input("Enter the message to be sent: ")

if s == "quit":

break

c.send(s.encode())

reply = c.recv(1024).decode()

print(f"Message from Server: {reply}")

c.close()

print(f'{name} closed connection')

except ConnectionRefusedError:

print("No connection could be made because the target machine actively refused it")

5. Design an client server program using TCP where the server returns the Time to the client.

Sample output:

$ python server.py &

Got a connection from ('127.0.0.1', 54597)

$ python client.py

The time got from the server is Sun Dec 19 17:19:31 2021

**Client**

import socket

c = socket.socket()

try:

c.connect((socket.gethostname(),9999))

name = input("Enter your name: ")

c.send(name.encode())

while True:

reply = c.recv(1024).decode()

print(reply)

c.close()

print(f'{name} closed connection')

break

except ConnectionRefusedError:

print("No connection could be made because the target machine actively refused it")

**Server**

import socket

from datetime import datetime

def get\_time():

now = datetime.now()

dt\_string = now.strftime("%d/%m/%Y %H:%M:%S")

time = f"Current Time: {dt\_string}"

return time

s = socket.socket()

host = socket.gethostname()

s.bind((host,9999))

s.listen(1)

print('waiting for connections!')

c,addr = s.accept()

name = c.recv(1024).decode()

print(f"connected with {name}")

while True:

c.send(get\_time().encode())

print(f"Connection closed for {name}")

c.close()

break

6. Send objects via sockets using pickle

**Client**

import socket

import pickle

c = socket.socket()

try:

c.connect((socket.gethostname(),9999))

name = input("Enter your name: ")

c.send(name.encode())

while True:

s = input("Enter the message to be sent: ")

if s == "quit":

break

data\_string = pickle.dumps(s)

c.send(data\_string)

reply = c.recv(1024).decode()

print(f"Message from Server: {reply}")

c.close()

print(f'{name} closed connection')

except ConnectionRefusedError:

print("No connection could be made because the target machine actively refused it")

Server

import socket

import pickle

s = socket.socket()

host = socket.gethostname()

s.bind((host,9999))

s.listen(1)

print('waiting for connections!')

c,addr = s.accept()

name = c.recv(1024).decode()

print(f"connected with {name}")

while True:

msg = c.recv(1024)

if msg:

data\_variable = pickle.loads(msg)

c.send(data\_variable.encode())

else:

print(f"Connection closed for {name}")

c.close()

break