CS425

Assignment IV - Socket Programming

Course Instructor :

Dr. Amitangshu Pal

Submitted By :

Nishant Roshan
200643

Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur
April 19, 2023

Question

Please write a socket programming for UDP client server application with 1 server and 1 client. You can reuse the code shown in the class and can do some small changes as needed.

Solution:

Ans: The following is the server and client code:

client.py server.py

server.py > ...

```python
from socket import *

server_port = 12000
server_IP = 'localhost'   #127.0.0.1

# create a socket object
server_sock = socket(AF_INET,SOCK_DGRAM)

# bind the socket to a specific address and port
server_sock.bind(server_IP,server_port)

print(f"Server listening on {server_IP}:{server_port}")

while True:
    #receive data from client
    message, address = server_sock.recvfrom(1024)   # buffer size is 1024 bytes
    print(f"Connection from {address} is established")
    print(f"Received message:{message.decode()}")

    modified_message = message.decode().upper()

    # send a response back to the client
    server_sock.sendto(modified_message.encode(),address)

```

Figure 1:

In the above code, the server listens on port 12000 for incoming messages from the client. When it receives a message, it prints the message to the console and sends a response back to the client with the message in uppercase of the message received.

client.py server.py

client.py > ...

```python
from socket import *

server_port = 12000
server_IP = 'localhost'   #127.0.0.1

# create a socket object
client_sock = socket(AF_INET, SOCK_DGRAM)

# send a message to the server
message =  input('Input lowercase sentence:')
client_sock.sendto(message.encode(), (server_IP, server_port))

# receive the server's response
modified_message, address = client_sock.recvfrom(1024)   # buffer size is 1024 bytes
print(f"Received message: {modified_message.decode()}")

# close the socket
client_sock.close()
```

Figure 2:

In the above code, the client sends to the server the message it receives from the command line and then waits to receive a response. When it receives a response from the server, it prints the message to the console.

Thank You