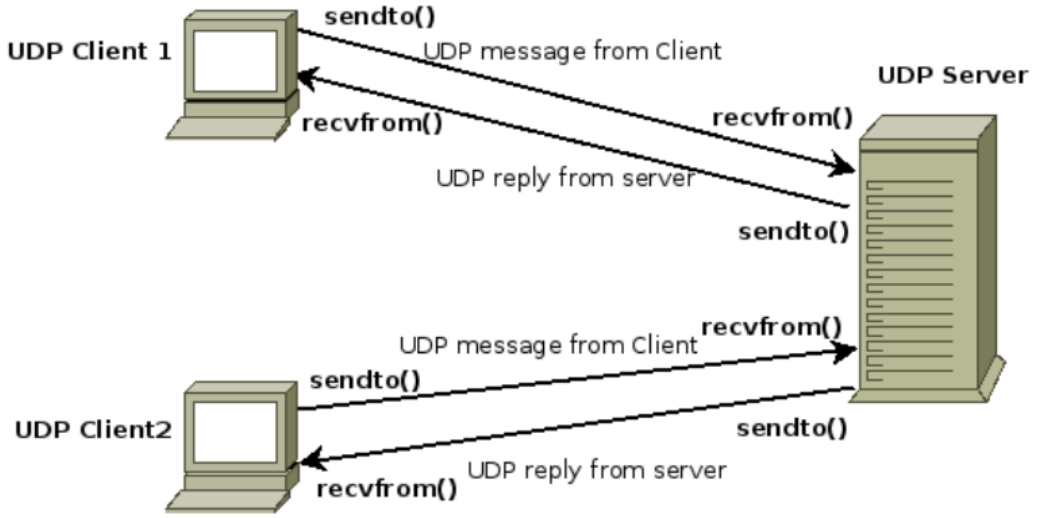
CO223 - Lab 05
Socket Programming in Python II

UDP



UDP or User Datagram Protocol is connection-less protocol which is suitable for applications that require efficient communication that doesn't have to worry about packet loss.

UDP is the abbreviation of User Datagram Protocol. In communications using UDP, a client program sends a message packet to a destination server wherein the destination server also runs on UDP.

Sample UDP Server Code

```
import socket

localIP = "127.0.0.1"

localPort = 20001

bufferSize = 1024

msgFromServer ='Hello UDP Client'

bytesToSend = str.encode(msgFromServer)

# Create a datagram socket

UDPServerSocket = socket.socket(family=socket.AF_INET,
```

```
type=socket.SOCK_DGRAM)

# Bind to address and ip

UDPServerSocket.bind((localIP, localPort))

print("UDP server up and listening")

# Listen for incoming datagrams

while (True):
    bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)

    message = bytesAddressPair[0]

    address = bytesAddressPair[1]

    clientMsg = "Message from Client: {}".format(message)
    clientIP = "Client IP Address: {}".format(address)

    print(clientMsg)
    print(clientIP)

    # Sending a reply to client

    UDPServerSocket.sendto(bytesToSend, address)
```

Sample UDP Client Code

```
import socket

msgFromClient = 'Hello UDP Server'

bytesToSend = str.encode(msgFromClient)

serverAddressPort = ("127.0.0.1", 20001)

bufferSize = 1024

# Create a UDP socket at client side

UDPClientSocket = socket.socket(family=socket.AF_INET,
type=socket.SOCK_DGRAM)

# Send to server using created UDP socket

UDPClientSocket.sendto(bytesToSend, serverAddressPort)
```

```
msgFromServer = UDPClientSocket.recvfrom(bufferSize)

msg ="Message from Server {}".format(msgFromServer[0])

print(msg)
```

Lab Exercise

Task 01

1. Create a simple UDP server and UDP client program. Observe the output behaviour. (You can use the given sample codes as reference)
2. If a client sends multiple messages/requests before waiting for the reply from the server, what will happen?
3. Explain the behaviour of the network when million requests are sent to the server from the client.

Task 02

1. Modify the code in Task 01 to create multiple servers which listens and communicate with a single client.
2. Is the above possible? Explain your answer.

Task 03

3. What will happen when Connect() is used in UDP connection in Socket Programming?
4. Modify the code to use Connect() and try sending and receiving messages using send(), recv() and sendto(), recvfrom() commands and explain your observations.

Task 04

1. Create a Server which would return a number sent by the client.
2. Client would send *n* number of requests (sequential numbers) before start listening to the replies & printing those out (note: *n* could be a global variable or an input to the program when running)
3. Keep increasing *n* to see when the packet drop (missing numbers) starts. Explain the reason for the observation.

4. Next, change the Receive Buffer Size (reference: `socket.setsockopt()` https://docs.python.org/3/library/socket.html and which should lead you to https://manpages.debian.org/bullseye/manpages/socket.7.en.html) and test again. Explain the differences in the observation of this task with the previous results. What would be the reason for this?

Submission

Create a report renamed as **E18XXX_Report.pdf** (XXX is your E Number).

The report should include,

- Answers for the lab tasks 1 -4 with the necessary explanations, screenshots of the codes segments and the output
- Include comments in the code to explain

Submit a zipped folder named **E18XXX_Lab05.zip**including the report and the .py files.