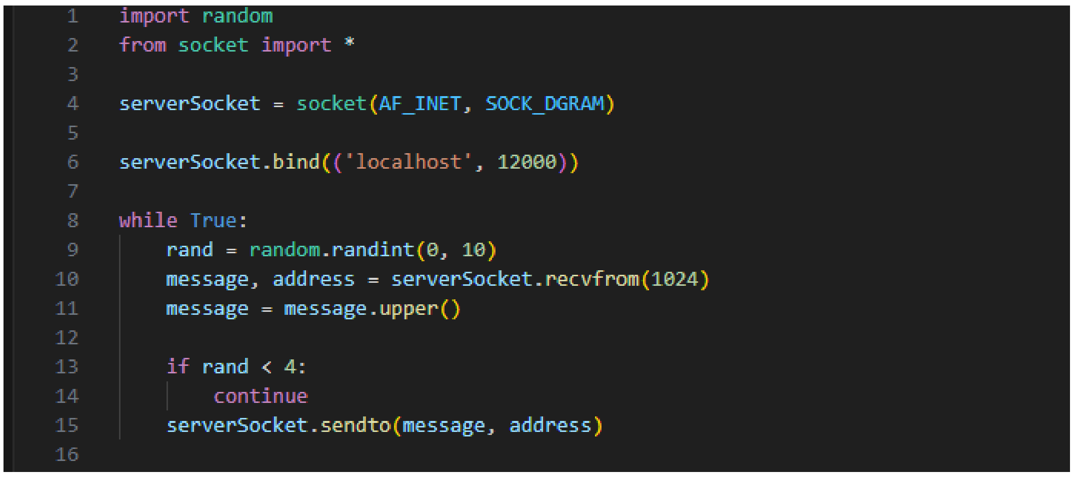
UDP Pinger

1. Server Code

In this assignment, it was asked by us to implement a corresponding client code for the given server code (Figure 1) in Python. About the given server code, it was simulated to lose 30% of the client's packets.



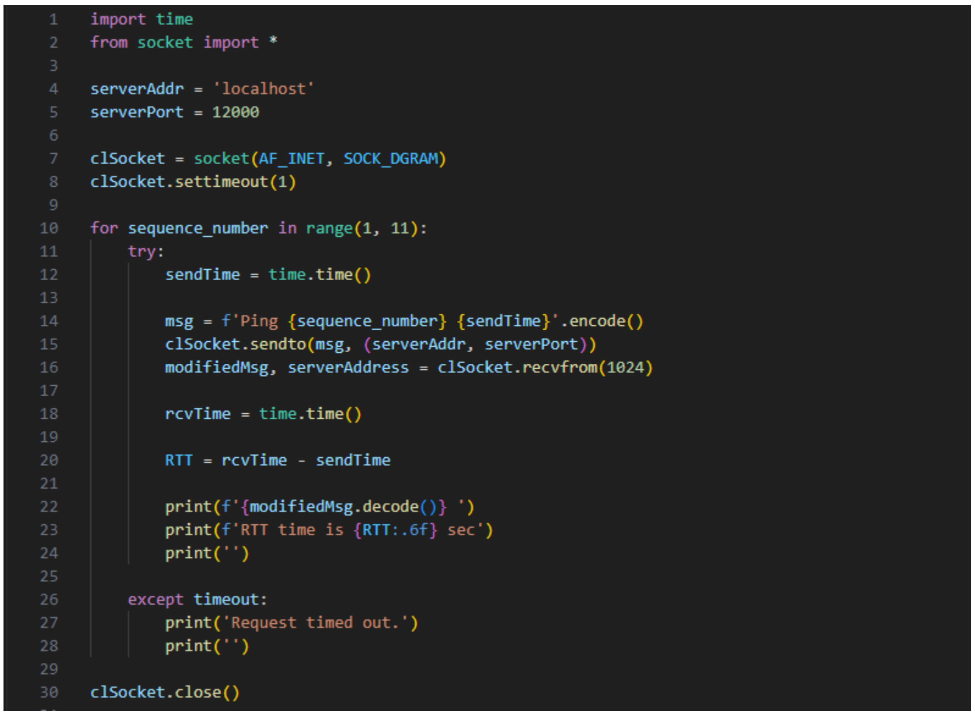
```python
import random
from socket import *

serverSocket = socket(AF_INET, SOCK_DGRAM)

serverSocket.bind(('localhost', 12000))

while True:
    rand = random.randint(0, 10)
    message, address = serverSocket.recvfrom(1024)
    message = message.upper()

    if rand < 4:
        continue
    serverSocket.sendto(message, address)

```

Figure 1: Server Code

2. Client Code

For client code (Figure 2), we implemented a code that sends 10 ping messages to the server using UDP. But because of UDP being an unreliable protocol, we implemented client code to wait up to one second for a reply, this way if client does not receive any replies within one second, it will be assumed that the packet was lost during the transmission across the network and will print out the output "Request timed out". If a reply is received in less than one second of time, the client will see the ping number, received message, and the RTT calculation of that specific ping number as the output for every 10 ping.

When it comes to RTT calculation, we subtract the send time of the message from the received time, as it can be seen in the client code below.



```python
import time
from socket import *

serverAddr = 'localhost'
serverPort = 12000

clSocket = socket(AF_INET, SOCK_DGRAM)
clSocket.settimeout(1)

for sequence_number in range(1, 11):
    try:
        sendTime = time.time()

        msg = f'Ping {sequence_number} {sendTime}'.encode()
        clSocket.sendto(msg, (serverAddr, serverPort))
        modifiedMsg, serverAddress = clSocket.recvfrom(1024)

        rcvTime = time.time()

        RTT = rcvTime - sendTime

        print(f'{modifiedMsg.decode()} ')
        print(f'RTT time is {RTT:.6f} sec')
        print('')

    except timeout:
        print('Request timed out.')
        print('')

clSocket.close()
```

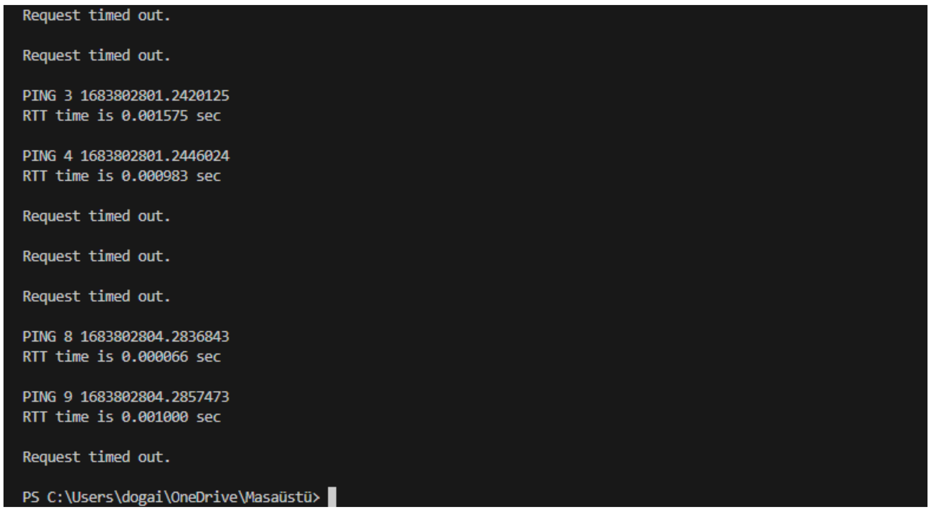
Figure 2: Client Code

To briefly explain the implemented code above, we can say that firstly we define the IP address and the port number of our computer to be able to make a connection between the client and the server. Then we create our socket and we set the timeout value to 1, for the reason explained previously, with the usage of function settimeout(). After that, we created a for loop with the range of 10 to be able to see all 10 pings that we should be printed as output. Inside the for loop, we clarify the received message format and then receive the message from the server in the defined format. While receiving the message, we keep the received time and send time values to be able to calculate the RTT value later. After we are done with the calculations we print out the ping numbers and modified messages along side RTT time values of these ping numbers. Lastly, in case of timeout and/or packet loss, we print out the "Request timed out" message as output for that timed out ping number.

3. Requested Outputs

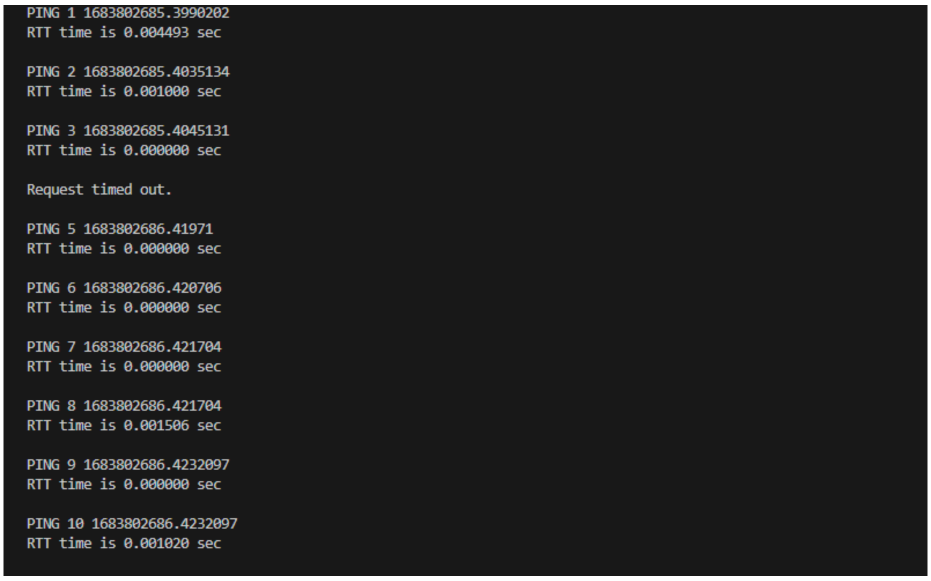
Lastly, we run the server and client code at the same time to be see the output values which can be seen in the following:

In the first output screenshot (Figure 3), for ping numbers 3, 4, 8, and 9 message is being received and according to received time and send time of these messages their RTT values is being calculated. For ping numbers 1, 2, 5, 6, 7, and 10 "Request timed out" is being printed as the output as a result of their receiving time being longer than one. (There are more output screenshots after the first explained one and the same explanation above can be done for each of them as well.)



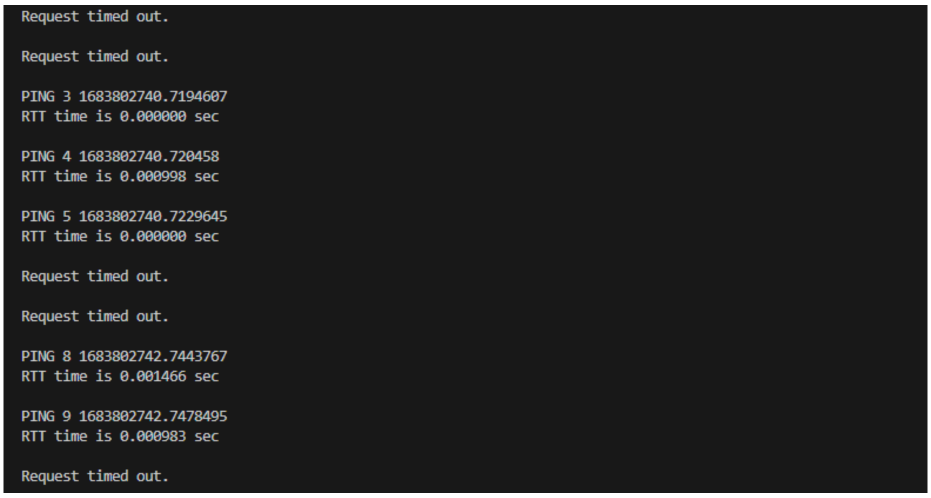
```
Request timed out.

Request timed out.

PING 3 1683802801.2420125
RTT time is 0.001575 sec

PING 4 1683802801.2446024
RTT time is 0.000983 sec

Request timed out.

Request timed out.

Request timed out.

PING 8 1683802804.2836843
RTT time is 0.000066 sec

PING 9 1683802804.2857473
RTT time is 0.001000 sec

Request timed out.

PS C:\Users\dogai\OneDrive\Masaüstü>
```

Figure 3: First Output Screenshot



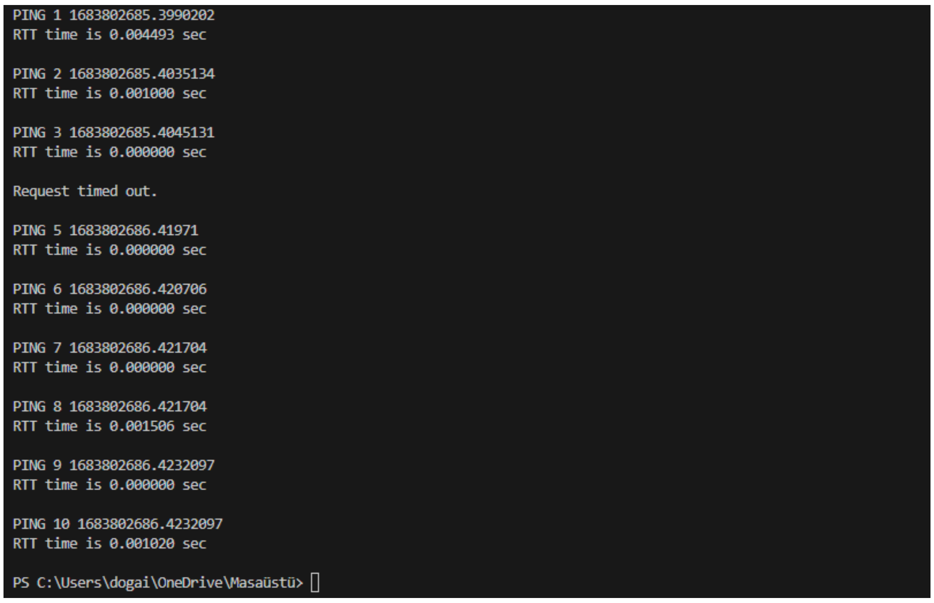
```
PING 1 1683802685.3990202
RTT time is 0.004493 sec

PING 2 1683802685.4035134
RTT time is 0.001000 sec

PING 3 1683802685.4045131
RTT time is 0.000000 sec

Request timed out.

PING 5 1683802686.41971
RTT time is 0.000000 sec

PING 6 1683802686.420706
RTT time is 0.000000 sec

PING 7 1683802686.421704
RTT time is 0.000000 sec

PING 8 1683802686.421704
RTT time is 0.001506 sec

PING 9 1683802686.4232097
RTT time is 0.000000 sec

PING 10 1683802686.4232097
RTT time is 0.001020 sec
```

Figure 4: Second Output Screenshot



```
Request timed out.

Request timed out.

PING 3 1683802740.7194607
RTT time is 0.000000 sec

PING 4 1683802740.720458
RTT time is 0.000998 sec

PING 5 1683802740.7229645
RTT time is 0.000000 sec

Request timed out.

Request timed out.

PING 8 1683802742.7443767
RTT time is 0.001466 sec

PING 9 1683802742.7478495
RTT time is 0.000983 sec

Request timed out.
```

Figure 5: Third Output Screenshot



```
PING 1 1683802685.3990202
RTT time is 0.004493 sec

PING 2 1683802685.4035134
RTT time is 0.001000 sec

PING 3 1683802685.4045131
RTT time is 0.000000 sec

Request timed out.

PING 5 1683802686.41971
RTT time is 0.000000 sec

PING 6 1683802686.420706
RTT time is 0.000000 sec

PING 7 1683802686.421704
RTT time is 0.000000 sec

PING 8 1683802686.421704
RTT time is 0.001506 sec

PING 9 1683802686.4232097
RTT time is 0.000000 sec

PING 10 1683802686.4232097
RTT time is 0.001020 sec

PS C:\Users\dogai\OneDrive\Masaüstü>
```

Figure 6: Fourth Output Screenshot