CPSC 471-01 Project 2

Team: Luc Dang (solo)

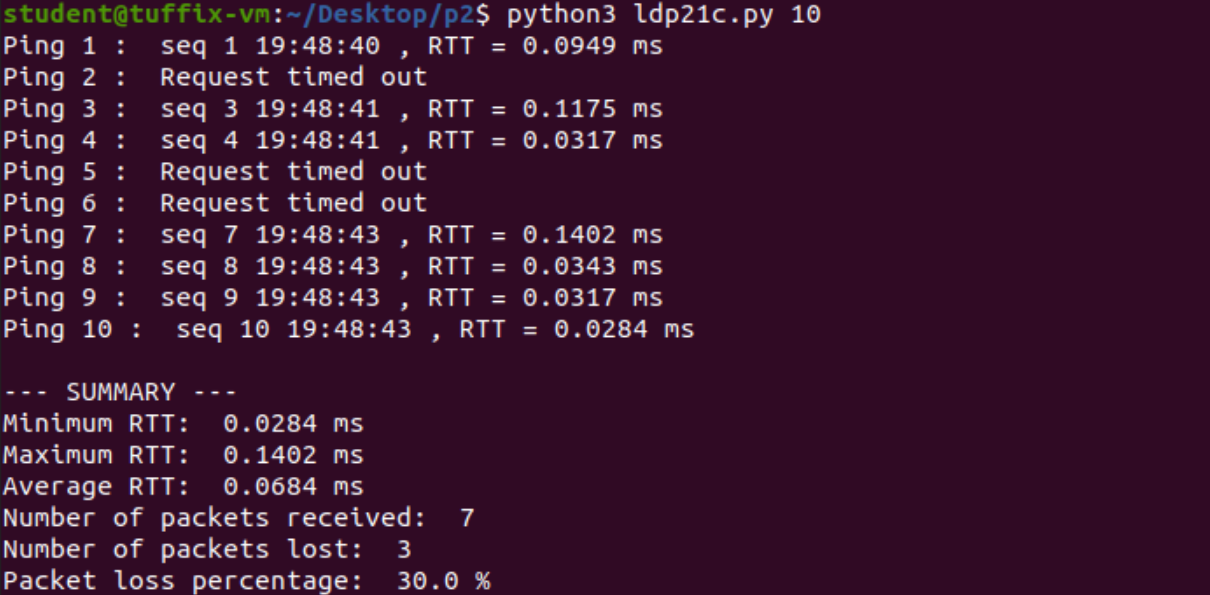
# Part 1 – UDP Pinger

## Instructions:

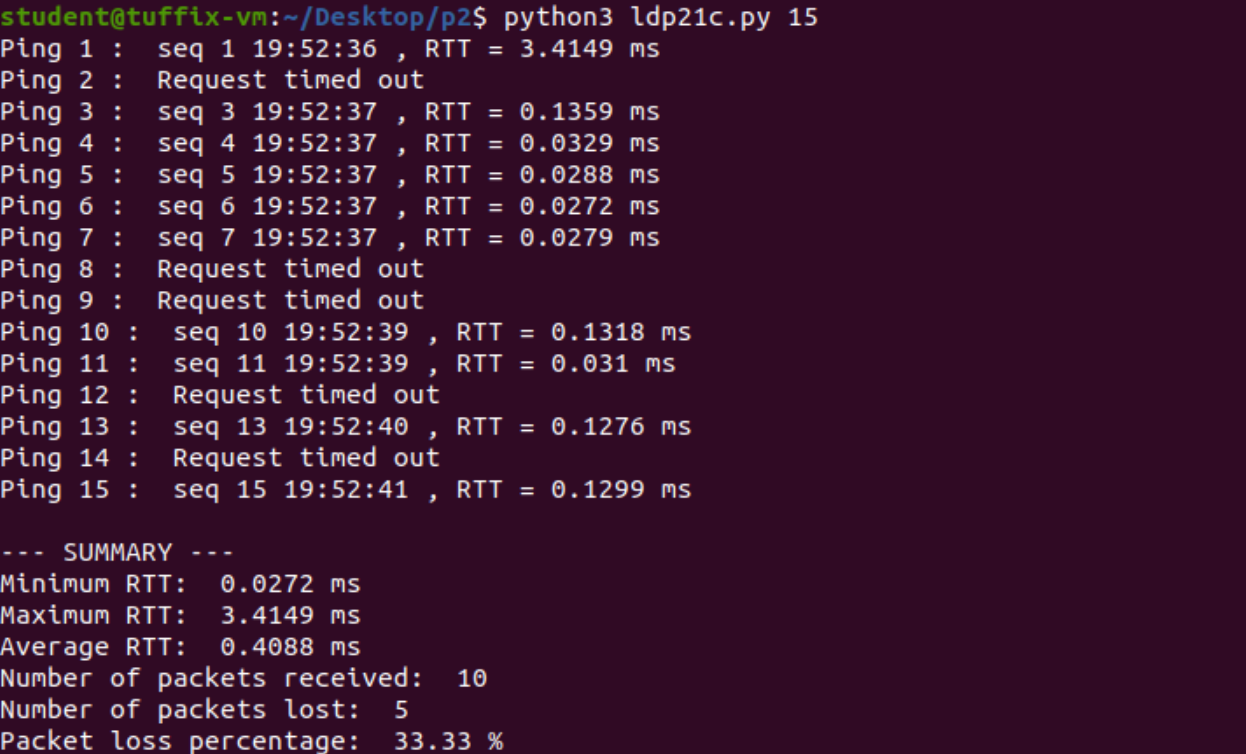
1. On a UNIX system open the system terminal and navigate to the appropriate directory containing the *udppingserver.py* and *ldp21c.py* files.
2. In the terminal, run the server program first with the command “*python3 udppingserver.py”.*
3. Open a new tab in the terminal, or a separate terminal window, and run the client program with the command *“python3 ldp21c.py NUM\_PINGS”* where NUM\_PINGS is the desired number of pings*.*

## Screen captures:

### a. A sequence consisting of 10 pings:



### b. A sequence consisting of 15 pings:



## Python code listing:

**# ldp21c.py**

**import time**

**import sys**

**from socket import \***

**# UDP socket**

**clientSocket = socket(AF\_INET, SOCK\_DGRAM)**

**clientSocket.settimeout(1)**

**serv\_addr = ('', 45678)**

**RTTarr = []**

**try:**

**numPings = int(sys.argv[1])**

**except IndexError:**

**print("\nIncorrect usage. Use: 'python3 ldp21c.py NUM\_PINGS' where NUM\_PINGS is the desired number of pings.\n")**

**for x in range(numPings):**

**sendTime = time.time()**

**message = ('seq ' + str(x+1) + " " + str(time.strftime("%H:%M:%S"))).encode()**

**clientSocket.sendto(message, serv\_addr)**

**try:**

**data, server = clientSocket.recvfrom(1024)**

**recvTime = time.time()**

**rtt = (recvTime - sendTime) \* 1000**

**RTTarr.append(rtt)**

**print("Ping", str(x+1) ,": ", data.decode(), ", RTT =",round(rtt, 4), "ms")**

**except timeout:**

**print("Ping", str(x+1) ,": ", "Request timed out")**

**print()**

**print("--- SUMMARY ---")**

**print("Minimum RTT: ", round(min(RTTarr), 4), "ms")**

**print("Maximum RTT: ", round(max(RTTarr), 4), "ms")**

**print("Average RTT: ", round((sum(RTTarr)/len(RTTarr)), 4), "ms")**

**print("Number of packets received: ", len(RTTarr))**

**print("Number of packets lost: ", numPings - len(RTTarr))**

**print("Packet loss percentage: ", round( ((numPings - len(RTTarr))/numPings\*100), 2), "%")**

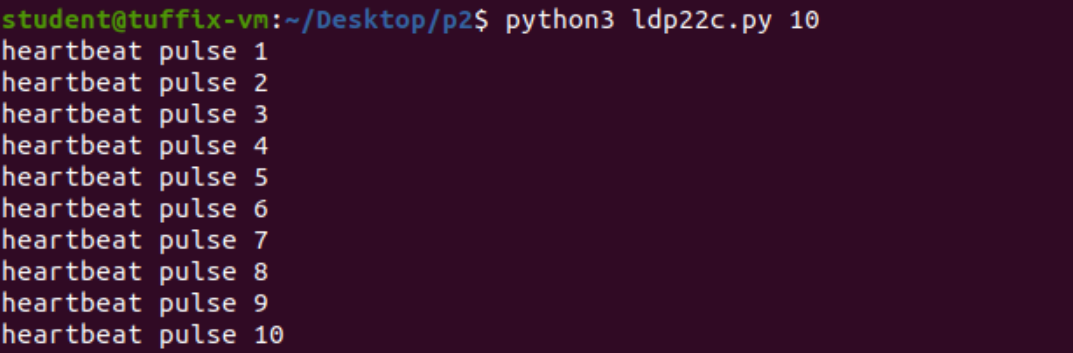
# Part 2 – UDP Heartbeat Monitor

## Instructions:

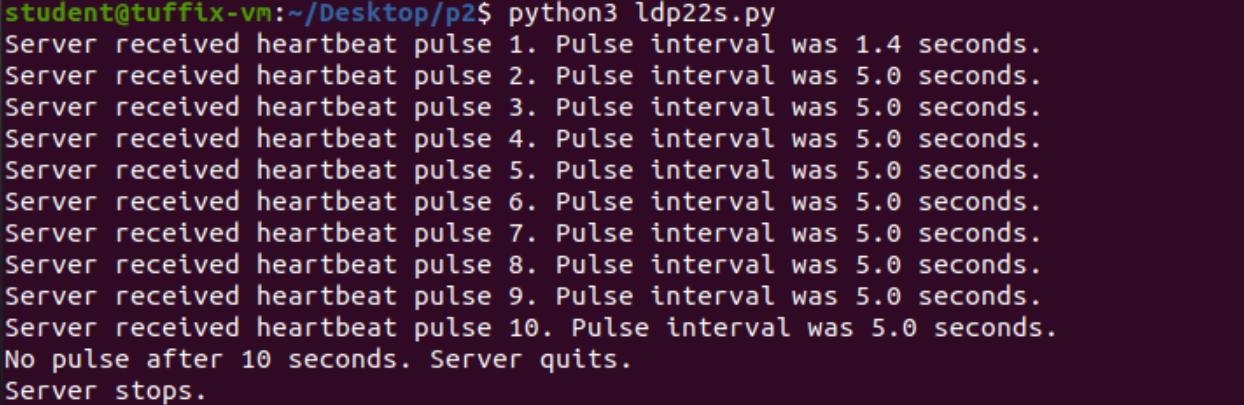
1. On a UNIX system open the system terminal and navigate to the appropriate directory containing the *ldp22s.py* and *ldp22c.py* files.
2. In the terminal, run the server program first with the command “*python3 ldp22s.py”.*
3. Open a new tab in the terminal, or a separate terminal window, and run the client program with the command *“python3 ldp22c.py NUM\_PINGS”* where NUM\_PINGS is the desired number of pings*.*

## Screen captures:

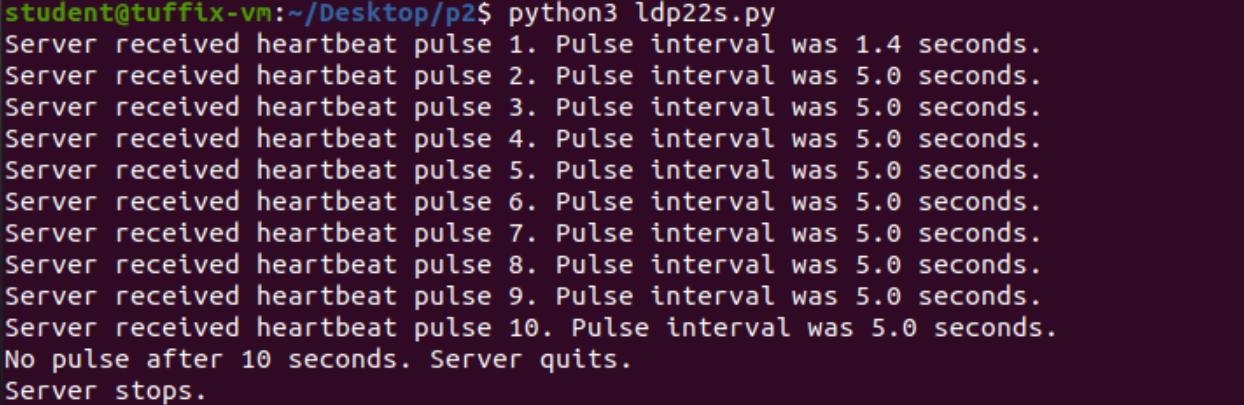
### a. Client sending heartbeat pings every 5 seconds:



### b. Server printing received heartbeat pings and time interval:



### c. Server detects absence of client heartbeat and quits:



## Python code listing:

### a. Client program:

**# ldp22c.py**

**import time**

**import sys**

**from socket import \***

**# UDP socket**

**clientSocket = socket(AF\_INET, SOCK\_DGRAM)**

**serv\_addr = ('', 45678)**

**RTTarr = []**

**try:**

**numPings = int(sys.argv[1])**

**except IndexError:**

**print("\nIncorrect usage. Use: 'python3 ldp22c.py NUM\_PINGS' where NUM\_PINGS is the desired number of pings.\n")**

**for x in range(numPings):**

**message = ("heartbeat pulse " + str(x+1)).encode()**

**clientSocket.sendto(message, serv\_addr)**

**print(message.decode())**

**time.sleep(5)**

### b. Server program:

**# ldp22s.py**

**import time**

**import sys**

**from socket import \***

**# Create a UDP socket**

**serverSocket = socket(AF\_INET, SOCK\_DGRAM)**

**# timeout after 10 seconds**

**serverSocket.settimeout(10)**

**# Assign IP address and port number to socket**

**serverSocket.bind(('', 45678))**

**interval = 0**

**while True:**

**prevTime = time.time()**

**try:**

**message, address = serverSocket.recvfrom(1024)**

**recvTime = time.time()**

**interval = recvTime - prevTime**

**print("Server received " + message.decode() + ". Pulse interval was " + str(round(interval, 1)) + " seconds.")**

**except timeout:**

**print("No pulse after 10 seconds. Server quits.\nServer stops.")**

**serverSocket.close()**

**sys.exit()**