import sys, time

from socket import \*

# Initialization Section

# Set host, port and timeout

host = '127.0.0.1'

port = 12000

timeout = 1 # in second

seq = 1

# create dgram udp socket

try:

s = socket(AF\_INET, SOCK\_DGRAM)

s.settimeout(1)

except socket.error:

print('Failed to create socket')

sys.exit()

# Ping for 10 times using a while loop

min\_rtt = 1

max\_rtt = 0

losses = 0

total\_rtt = 0

while seq <= 10:

t = time.localtime()

message = f"Ping {seq} {time.asctime(t)}"

seq += 1

# Put the rest in a try/except block to handle any possible timeouts

try:

# YOU FILL IN: Set a variable for the time sent (hint: time.time() is nice)

start = time.time()

# YOU FILL IN:Send the UDP packet with the ping message

s.sendto(message.encode(), (host, port))

# YOU FILL IN: Receive the server response and address; user recvfrom(1024). Read the documentation!

# Pay attention to the format of the returned parameters.

data = s.recvfrom(1024)

reply = data[0]

addr = data[1]

# YOU FILL IN: Set a variable for the time received

finish = time.time()

# YOU FILL IN:Print the server response as an output as described earlier: PING FROM ADDRESS: MESSAGE

# Notice the message that is returned is a byte string, not a normal UTF-8 string

# To get a normal string use the mystring.decode() standard function

# Also, print only the IPv4 address. I'm not interested in the port number.

print(f"Reply from {addr[0]}: {reply.decode()}")

# YOU FILL IN: calculate and print the round trip time

rtt = finish-start

print(f"RTT: {rtt}")

if rtt < min\_rtt:

min\_rtt = rtt

if rtt > max\_rtt:

max\_rtt = rtt

total\_rtt += rtt

except Exception as e:

losses += 1

# Server does not respond. Assume the packet is lost

# Assume the packet is lost

print ("Request timed out.")

continue

print(f'\n--- {addr[0]} ping statistics ---')

print(f'10 packets transmitted, {10-losses} received, {losses/10 \* 100}% packet loss, time {total\_rtt\*1000:.0f}ms')

print(f'rtt min/avg/max = {min\_rtt\*1000:.3f}/{(total\_rtt/(10-losses))\*1000:.3f}/{max\_rtt\*1000:.3f} ms')

# Close the client socket

s.close()

Reply from 127.0.0.1: PING 1 TUE SEP 15 00:12:11 2020

RTT: 0.0005924701690673828

Request timed out.

Reply from 127.0.0.1: PING 3 TUE SEP 15 00:12:12 2020

RTT: 0.0010123252868652344

Reply from 127.0.0.1: PING 4 TUE SEP 15 00:12:12 2020

RTT: 0.0007503032684326172

Reply from 127.0.0.1: PING 5 TUE SEP 15 00:12:12 2020

RTT: 0.0011098384857177734

Reply from 127.0.0.1: PING 6 TUE SEP 15 00:12:12 2020

RTT: 0.0013206005096435547

Reply from 127.0.0.1: PING 7 TUE SEP 15 00:12:12 2020

RTT: 0.0005893707275390625

Reply from 127.0.0.1: PING 8 TUE SEP 15 00:12:12 2020

RTT: 0.0010259151458740234

Reply from 127.0.0.1: PING 9 TUE SEP 15 00:12:12 2020

RTT: 0.00023055076599121094

Reply from 127.0.0.1: PING 10 TUE SEP 15 00:12:12 2020

RTT: 0.00017762184143066406

--- 127.0.0.1 ping statistics ---

10 packets transmitted, 9 received, 10.0% packet loss, time 7ms

rtt min/avg/max = 0.178/0.757/1.321 ms

Reply from 129.82.44.96: PING 1 TUE SEP 15 00:15:14 2020

RTT: 0.00030112266540527344

Reply from 129.82.44.96: PING 2 TUE SEP 15 00:15:14 2020

RTT: 0.00022029876708984375

Request timed out.

Reply from 129.82.44.96: PING 4 TUE SEP 15 00:15:15 2020

RTT: 0.0003249645233154297

Reply from 129.82.44.96: PING 5 TUE SEP 15 00:15:15 2020

RTT: 0.00035762786865234375

Request timed out.

Reply from 129.82.44.96: PING 7 TUE SEP 15 00:15:16 2020

RTT: 0.0003590583801269531

Reply from 129.82.44.96: PING 8 TUE SEP 15 00:15:16 2020

RTT: 0.0003104209899902344

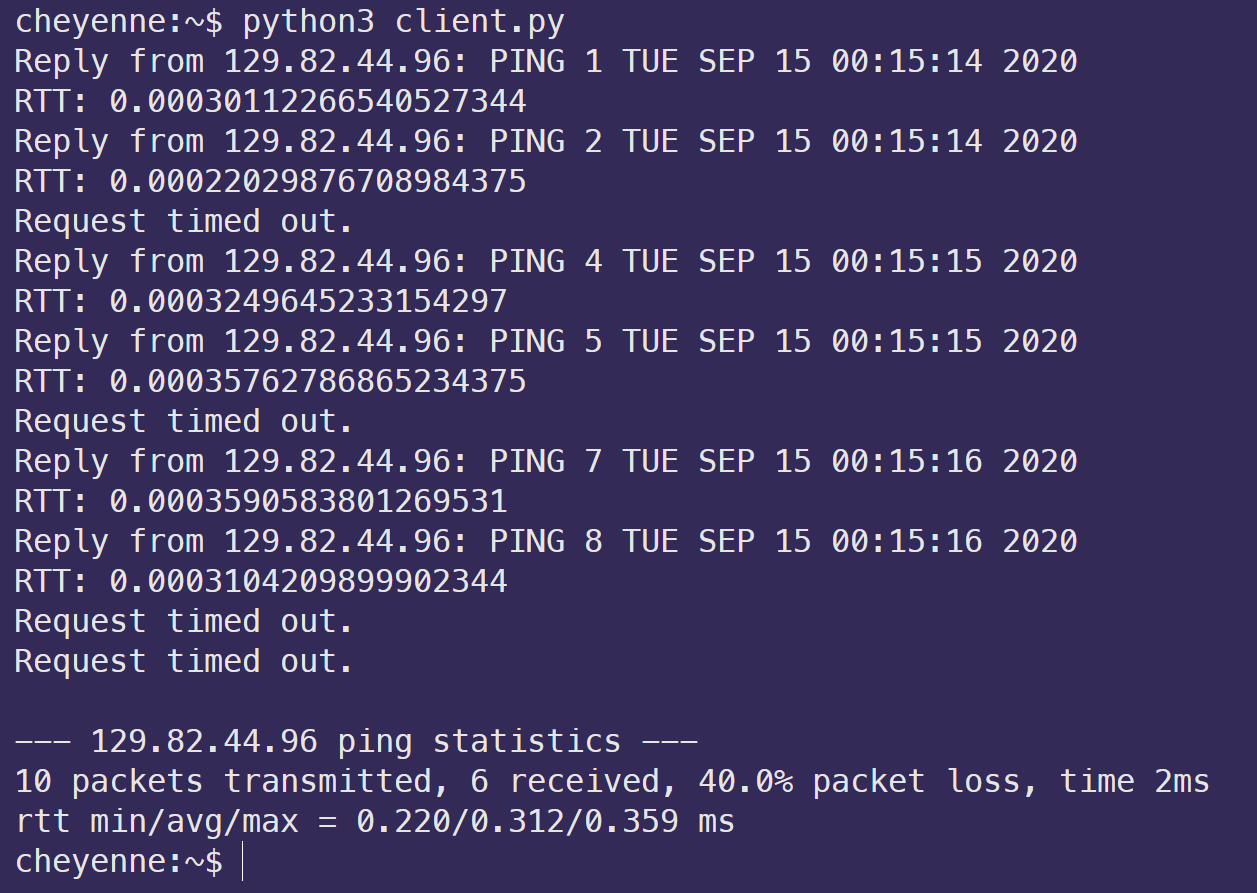
Request timed out.

Request timed out.

--- 129.82.44.96 ping statistics ---

10 packets transmitted, 6 received, 40.0% packet loss, time 2ms

rtt min/avg/max = 0.220/0.312/0.359 ms



It took approximately twice as long for a packet to perform a round-trip on my local machine compared to the CS lab machines. This may be due to me running the experiment within the Windows Subsystem for Linux.