CPSC 471 Assignment #2

Part 1: UDP Pinger with No Delay and No Loss

1. The UDP pinger uses a while loop and a counter to keep track of the number of pings sent to send exactly 10 pings with the formatted current time, then upon receiving a response from the server, it calculates the difference in the sent time and received time (both in nanoseconds) and transforms it to millisecond

2. To set a timeout value on a UDP socket, we use the socket.settimeot method.
   Ex:

   ```
   client_socket.settimeout(.01)
   ```

3. To run this code, you need to open two terminals. In the first one run `python udppingserver_no_loss.py` (this starts the server), then switch to the second and run `python oc1.py`. Now just wait for the response messages.

4. Client code

```
from socket import *
from time import gmtime, strftime, time_ns
from math import floor
from sys import maxsize


SERVER_ADDRESS = '127.0.0.1'
SERVER_PORT = 12000
ping_count = 0

min_rtt = maxsize
max_rtt = -1
total_rtt = 0
packet_loss = 0

format_as_ms = lambda ns: floor(ns * 10e-4)/100

while ping_count < 10:
      with socket(AF_INET, SOCK_DGRAM) as client_socket:
      #increment count
      ping_count += 1
      # create message
      t = gmtime()
      message = f'Oscar {ping_count} {strftime("%a %b %d %H:%M:%S %Y", t)}'
      # calculate start time in ns
      sent_ns = time_ns()
```

```
        client_socket.settimeout(.01)

        try:
                # send the message
                client_socket.sendto(message.encode(),(SERVER_ADDRESS,
SERVER_PORT))
                # received and print
                received, server_address = client_socket.recvfrom(1024)
                # calculate difference
                received_ns = time_ns()
                difference = format_as_ms(received_ns - sent_ns)
                # print final output string
                print(f'Oscar {ping_count}: server reply: {received.decode()},
RTT = {difference:.2f}')
        except TimeoutError:
                print('Error occurred')
        if difference > max_rtt:
                max_rtt = difference
        if difference < min_rtt:
                min_rtt = difference
        total_rtt += difference
print(f'Min RTT = {min_rtt:.2f}\nMax RTT = {max_rtt:.2f}\nAvg RTT =
{(total_rtt/10):.2f}\nPacket Lost = {(packet_loss/10):.2f} %')
```

Part 2: UDP Pinger with Delays

1. The program creates a udp pinger server socket that randomly generates delays. Originally I chose to use numbers between 10 and 20 in the random int generator to pass to sleep to simulate delays, however this didn't work as I kept getting delays that were larger than 20 ms. So I ramped up the number of pings, and set the max wait to 20 milliseconds so that I could generate RTTs that were below the threshold. I ended up with a randint generator from 9 to 12 due to some of the outliers resulting from potential scheduling decisions from the OS. However ultimately this resulted in no recordings of 20 milliseconds in 500,000 pings.
2. To run this program on your own system, you must run the server first with `python oc2.py`, then test it by starting your client with `oc1.py`.
3. UDP Ping Server

```
from socket import *
from time import sleep
from random import randint


server_socket = socket(AF_INET, SOCK_DGRAM)
server_socket.bind(('', 12000))
```

```
while True:
        # Receive the client packet along with the address it is coming from
        message, address = server_socket.recvfrom(1024)

        # generate random number that will result in 10 -20 ms delays
        n = randint(9, 12)
        sleep(n/10000)

        server_socket.sendto(message, address)
```

Part 3: UDP Pinger with Delays and Packet Losses

1. This program required me to update the client to catch the timeout errors and print a custom message. In order to simulate max loss of 10% packets i set the condition for sending back the message to only send if doing so would the random integer was less than 10 and adding it would not result in greater than a 10% loss.
2. To run this code you just start the server with `python oc3.py` and the clien with `python oc1.py`
3. Server

```
from socket import *
from time import sleep
from random import randint


server_socket = socket(AF_INET, SOCK_DGRAM)
server_socket.bind(('', 12000))
ping_count = 0
loss_count = 0

while True:
        ping_count += 1
        # Receive the client packet along with the addressit is coming from
        message, address = server_socket.recvfrom(1024)

        n = randint(9, 13)
        lost = randint(0, 100)
        sleep(n/10000)

        if lost > 10 and (loss_count + 1) > (ping_count*0.1):
        server_socket.sendto(message, address)
        continue


        loss_count +=1
```

Client

```
from socket import *
from time import gmtime, strftime, time_ns
from math import floor
from sys import maxsize


SERVER_ADDRESS = '127.0.0.1'
SERVER_PORT = 12000
```

```
ping_count = 0

min_rtt = maxsize
max_rtt = -1
total_rtt = 0
packet_loss = 0

format_as_ms = lambda ns: floor(ns * 10e-4)/100
print('UDP Ping Client is Up...\n')


while ping_count < 50:
	with socket(AF_INET, SOCK_DGRAM) as client_socket:
	#increment count
	ping_count += 1
	# create message
	t = gmtime()
	message = f'Oscar {ping_count} {strftime("%a %b %d %H:%M:%S %Y", t)}'
	# calculate start time in ns
	sent_ns = time_ns()

	client_socket.settimeout(.01)

	try:
		# send the message
		client_socket.sendto(message.encode(),(SERVER_ADDRESS,
SERVER_PORT))
		# received and print
		received, server_address = client_socket.recvfrom(1024)
		# calculate difference
		received_ns = time_ns()
		difference = format_as_ms(received_ns - sent_ns)
		# print final output string
		print(f'Oscar {ping_count}: server reply: {received.decode()},
RTT = {difference:.2f}')

		if difference > max_rtt:
		max_rtt = difference
		if difference < min_rtt:
		min_rtt = difference
		total_rtt += difference

	except TimeoutError:
		packet_loss += 1
		print(f'Oscar {ping_count}: timed out, message was lost')

print(f'Min RTT = {min_rtt:.2f}\nMax RTT = {max_rtt:.2f}\nAvg RTT =
{(total_rtt/50):.2f}\nPacket Lost = {(packet_loss/.5):.2f} %')
```

Part 4: UDP Heartbeat Monitor

1. This program uses a few global variables to track the last time and the gap on each incoming message the server will check if last is declared. If so gap is the difference between the current time t and the last time a message was received, then if that gap is above 210 the server stops as the break stops the while true.

The clients each call sleep for a random interval of time between 2 and 25 seconds.

2. Can’t upload media to document. Will zip.
3. Client

```
from socket import *
from time import gmtime, strftime, time, sleep
from random import randint


SERVER_ADDRESS = '127.0.0.1'
SERVER_PORT = 12000
client_no = int(input())

while True:
	with socket(AF_INET, SOCK_DGRAM) as client_socket:
	# create message
	t = gmtime()

	message = f'Oscar client{client_no} heartbeat at {strftime("%a %b
%d %H:%M:%S %Y", t)}'

	delay = randint(2,26)
	sleep(delay)

	# send the message
	client_socket.sendto(message.encode(),(SERVER_ADDRESS,
SERVER_PORT))


	# received and print
	received, server_address = client_socket.recvfrom(1024)

	# print final output string
	print(received.decode())
```

Server

```
from socket import *
from math import floor
from time import time

server_socket = socket(AF_INET, SOCK_DGRAM)
server_socket.bind(('', 12000))
last = -1
gap = 0

while True:
	# Receive the client packet along with the addressit is coming
from
	message, address = server_socket.recvfrom(1024)
	t = floor(time())

	if last > 0:
	gap = t - last
	if gap > 20:
		print('No heartbeat after 20 seconds. Server Quits')
```

```
                break
            print(f'Server received: {message.decode()} Last heartbeat
received {gap} seconds ago.')
            server_socket.sendto(message, address)
            last = t
```