Elson Jian (918147110)
Jacob Dominick (917503904)

Chat GPT Session Link
Submitted files:
client.py, server.py
client_gpt.py, server_gpt.py

server.py:
```
import socket
import time

# specify host and port to receive messages on
HOST = '127.0.0.1'
PORT = 5500

# create a new datagram socket
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as server_socket:
# bind this socket to OS
server_socket.bind((HOST, PORT))

while True:
print("UDP server listening on {}:{}".format(HOST, PORT))
# data -> message, addr -> (client_addr, client_port)
data, addr = server_socket.recvfrom(1024)
data_size = len(data)
if data:
start_time = time.time()
end_time = time.time()

# receive data until size exceeds 100 kb
while data_size < 102400:
data, addr = server_socket.recvfrom(1024)
data_size += len(data)
print(f"Received {len(data)} bytes from {addr}")
end_time = time.time()

# throughput = bytes / (RTT / 2)
throughput = data_size / (end_time - start_time) / 1024 # [KB/sec]
server_socket.sendto(str(throughput).encode(), addr)
print("Sent throughput to client")

server_socket.close()
```

client.py:
```
import socket
import time

# specify server host and port to connect to
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 5500

# maximum transmission unit
MTU = 1024

# open a new datagram socket
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as client_socket:
client_socket.settimeout(5) # set timeout to 5 seconds
# craft message body
message = "a" * 102400 # 100 KB string

# send message
for i in range(0, len(message), MTU):
# .encode() converts to bytes
segment = message[i:i + MTU]
client_socket.sendto(segment.encode(), (SERVER_HOST, SERVER_PORT))

time.sleep(1)
# receive return message
response, addr = client_socket.recvfrom(MTU)
print("{:.2f} KB/sec".format(float(response.decode())))

client_socket.close()
```

Output:
The throughput varied a lot each time the code was ran but in general, it was in the range of 60,000 to 150,000 KB/sec.

In this implementation, the server measures the throughput completely independently of the client.py script. It does this by measuring the time it takes to receive 100 KB of data. But unfortunately, this value of 100 KB is hard-coded as the check to stop the timer and calculate throughput. So if a message of size <100KB is sent to the server, it would be stuck in a loop forever and the client would never receive a response.

I think this solution is adequate for the scope of this project, but in the future, receiving messages of varying sizes should be taken into consideration.