Name : Nimna Wijedasa

UCID : 30146042

*from* socket *import* \*

*import* threading

*def* handle\_client(connectionSocket):

*try*:

*# Receive the HTTP request message*

message = connectionSocket.recv(*1024*).decode()

*# Extract the filename from the request*

filename = message.split()[*1*][*1*:]

print('Filename:', filename)

*# Open the requested file*

*with* open(filename, "rb") *as* f:

outputdata = f.read()

*# Send HTTP response headers*

response\_headers = "HTTP/1.1 200 OK*\r\n\r\n*"

connectionSocket.send(response\_headers.encode())

*# Send the content of the requested file to the client*

connectionSocket.sendall(outputdata)

*except* IOError:

*# Send response message for file not found (404 Not Found)*

not\_found\_response = "HTTP/1.1 404 Not Found*\r\n\r\n*File not found"

connectionSocket.send(not\_found\_response.encode())

*# Close the client socket*

connectionSocket.close()

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

*# Prepare a server socket*

serverPort = *6789*

serverSocket.bind(('0.0.0.0', serverPort))

serverSocket.listen(*5*)

print('The server is ready to receive')

*while* *True*:

*# Establish the connection*

print('Ready to serve...')

connectionSocket, addr = serverSocket.accept()

print('Connected by', addr)

*# Create a new thread to handle the client connection*

client\_thread = threading.Thread(target=handle\_client, args=(connectionSocket,))

client\_thread.start()

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

*import* socket

*import* time

*# Server address (replace with the actual server's IP address and port)*

server\_address = ('10.9.183.59', *12000*)

*# Number of pings to send*

num\_pings = *10*

*# Create a UDP socket*

client\_socket = socket.socket(socket.*AF\_INET*, socket.*SOCK\_DGRAM*)

*# Initialize variables for RTT statistics*

min\_rtt = float('inf')

max\_rtt = *0*

total\_rtt = *0*

packets\_lost = *0*

*for* sequence\_number *in* range(*1*, num\_pings + *1*):

*# Get the current time*

send\_time = time.time()

*# Prepare the ping message*

ping\_message = *f*'Ping *{*sequence\_number*}* *{*send\_time*}*'

*# Send the ping message to the server*

client\_socket.sendto(ping\_message.encode(), server\_address)

*# Set a timeout for receiving the response (1 second)*

client\_socket.settimeout(*1*)

*try*:

*# Receive the response from the server*

response, server\_address = client\_socket.recvfrom(*1024*)

*# Calculate the round-trip time (RTT)*

receive\_time = time.time()

rtt = receive\_time - send\_time

*# Update RTT statistics*

total\_rtt += rtt

min\_rtt = min(min\_rtt, rtt)

max\_rtt = max(max\_rtt, rtt)

*# Print the response message and RTT*

print(*f*'Response from *{*server\_address*}*: *{*response.decode()*}*')

print(*f*'Round-trip time (RTT): *{*rtt*:.6f}* seconds')

*except* socket.timeout:

*# Handle a timeout (packet loss)*

print(*f*'Request timed out for sequence number *{*sequence\_number*}*')

packets\_lost += *1*

*# Calculate the average RTT*

average\_rtt = total\_rtt / num\_pings

*# Calculate the packet loss rate*

packet\_loss\_rate = (packets\_lost / num\_pings) \* *100*

*# Print statistics*

print(*f*'*\n*Ping statistics for *{*server\_address[*0*]*}*:')

print(*f*'Packets: Sent = *{*num\_pings*}*, Received = *{*num\_pings - packets\_lost*}*, Lost = *{*packets\_lost*}* (*{*packet\_loss\_rate*:.2f}*% loss)')

print('Approximate round-trip times in milliseconds:')

print(*f*'Minimum = *{*min\_rtt \* *1000:.6f}*ms, Maximum = *{*max\_rtt \* *1000:.6f}*ms, Average = *{*average\_rtt \* *1000:.6f}*ms')

*# Close the socket*

client\_socket.close()

A computer screen shot of a computer program

Description automatically generated