Term Project - Final Report

Course: CSE476 - Mobile Communication Networks

Student: Oğuzhan Agkuş - 161044003

Assignment 1

Server

I have implemented a simple multithreaded web server. I did not use any Python libaries for communication, I used only uses TCP sockets. It is capable of handling HTTP GET requests. I have added multithread functionality after, progress report. Additionally, I write a signal handler for CTRL+C because I want to shutdown my server succesfully.

Libraries, global variables, and signal handler:

```
import sys, signal, socket, threading, datetime

# Set global variables
threads = []
server_socket = None
server_address = ("", 80)

# Signal handler
def signal_handler(signal, frame):
  for thread in threads:
    print(thread.getName())
    thread.join()

  server_socket.close()
  sys.exit(0)
```

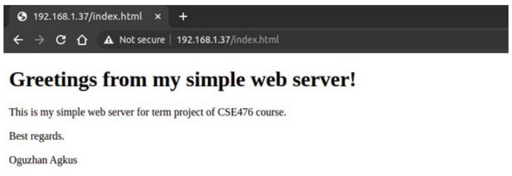
Since I have added multithread functionality, I had to define a thread function. Actually it performs same thing in each iteration of main loop in the previous version. In case of any exception, it will be handled. The threads handle incoming connections will use following function.

```
# Thread function
def client_handler(client_socket, address):
  try:
    request = client_socket.recv(1024).decode()
    items = request.split()

    request_type = items[0]
    if request_type != "GET":
      raise Exception("Method not allowed!")

    filename = items[1]
    file = open(filename[1:])
    data = file.read()
    file.close()

    # Prepare headers
    status_line = "HTTP/1.1 200 OK\r\n"
    header_lines = "Connection: keep-alive\r\n"
    header_lines += "Date: {}\r\n".format(datetime.datetime.now())
    header_lines += "Server: MyServer\r\n"
    header_lines += "Content-Length: {}\r\n".format(len(data))
    header_lines += "Content-Type: text/html\r\n"
    blank_line = "\r\n"
    response_message = status_line + header_lines + blank_line

    # Send headers
    client_socket.send(response_message.encode())

    # Send content of the file
    for i in range(len(data)):
      client_socket.send(data[i].encode())

    print(filename + " sent to " + address[0])

  # Print and response error message
  except Exception as error:
    print("Error occured!", error)
    if (error == "Method not allowed!"):
      client_socket.send("HTTP/1.1 405 METHOD NOT ALLOWED\r\n".encode())
    else:
      client_socket.send("HTTP/1.1 404 NOT FOUND\r\n".encode())

  # Close client socket
  client_socket.shutdown(socket.SHUT_RDWR)
  client_socket.close()
```

Main function accepts new incoming connections and create a new thread for handling these connections. It can be stopped with only CTRL+C.

```
# Main function
def main():
  global threads, server_socket, server_address

  # Prepare socket
  try:
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind(server_address)
    server_socket.listen(10)
  except Exception as error:
    print("Error occured!", error)
    sys.exit(1)

  # Set signal handler
  signal.signal(signal.SIGINT, signal_handler)

  # Main loop
  while True:
    # Accept a connection
    print("Server is ready!")
    client_socket, address = server_socket.accept()
    client_socket.settimeout(5)

    t = threading.Thread(target=client_handler, args=(client_socket, address,))
    threads.append(t)
    t.start()

if __name__ == "__main__":
  main()
```

I create a index.html page for testing my server. It is in the same directory with the code. I tried to connect from different devices (which are in my local network) to my server.

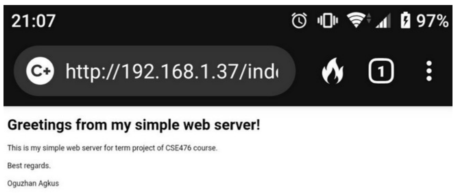


Greetings from my simple web server!

This is my simple web server for term project of CSE476 course.

Best regards.

Oguzhan Agkus

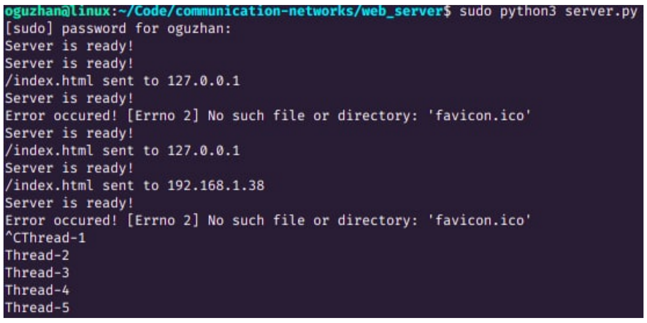


Greetings from my simple web server!

This is my simple web server for term project of CSE476 course.

Best regards.

Oguzhan Agkus



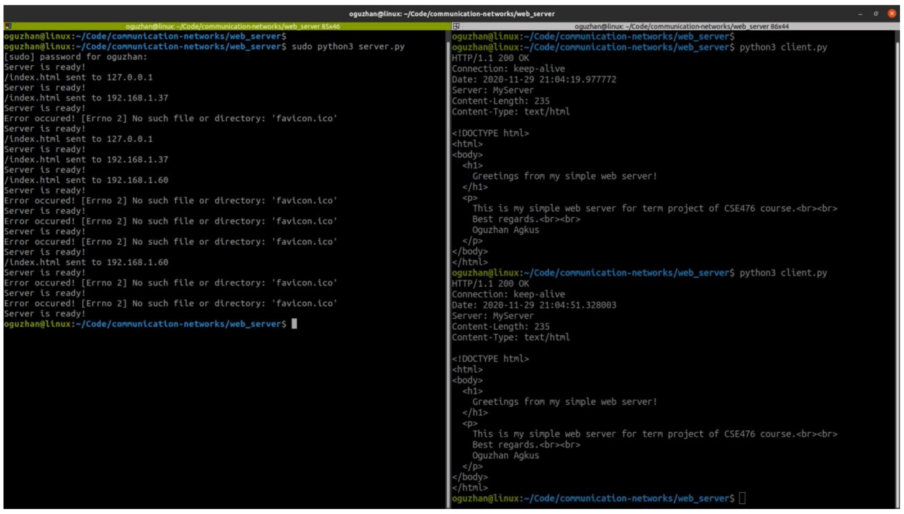
```
oguzhan@linux:~/Code/communication-networks/web_server$ sudo python3 server.py
[sudo] password for oguzhan:
Server is ready!
Server is ready!
/index.html sent to 127.0.0.1
Server is ready!
Error occured! [Errno 2] No such file or directory: 'favicon.ico'
Server is ready!
/index.html sent to 127.0.0.1
Server is ready!
/index.html sent to 192.168.1.38
Server is ready!
Error occured! [Errno 2] No such file or directory: 'favicon.ico'
^CThread-1
Thread-2
Thread-3
Thread-4
Thread-5
```

Client

I implemented optional exercise client code which simulates the browser. It sends request and print the server's response. It is possible to pass arguments to code but it has defaults.

```
import sys, socket

# Default arguments
server_host = "localhost"
server_port = 80
filename = "index.html"

# If any argument passed
if len(sys.argv) == 4:
  server_host = sys.argv[1]
  server_port = int(sys.argv[2])
  filename = sys.argv[3]

# Global variables
server_address = (server_host, server_port)
connection = None

# Prepare socket
try:
  connection = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  connection.connect(server_address)
except Exception as error:
  print("Error occured!", error)
  sys.exit(1)

# Prepare headers
request_line = "GET /" + filename + " HTTP/1.1\r\n"
header_lines = "Host: " + server_host + "\r\n"
header_lines += "Connection: close\r\n"
blank_line = "\r\n"
request_message = request_line + header_lines + blank_line

# Send headers
connection.send(request_message.encode())

# Recieve data
response = ""
while True:
    data = connection.recv(1024).decode()
    if not data:
        break
    response += data

# Print data
print(response)

# Close socket
connection.close()
```

I did not update the code. So, following screenshot is from the previous report.


oguzhan@linux: ~/Code/communication-networks/web_server
oguzhan@linux: ~/Code/communication-networks/web_server 85x46
oguzhan@linux: ~/Code/communication-networks/web_server 86x44

```
oguzhan@linux:~/Code/communication-networks/web_server$
oguzhan@linux:~/Code/communication-networks/web_server$ sudo python3 server.py
[sudo] password for oguzhan:
Server is ready!
/index.html sent to 127.0.0.1
Server is ready!
/index.html sent to 192.168.1.37
Server is ready!
Error occured! [Errno 2] No such file or directory: 'favicon.ico'
Server is ready!
/index.html sent to 127.0.0.1
Server is ready!
/index.html sent to 192.168.1.37
Server is ready!
/index.html sent to 192.168.1.60
Server is ready!
Error occured! [Errno 2] No such file or directory: 'favicon.ico'
Server is ready!
Error occured! [Errno 2] No such file or directory: 'favicon.ico'
Server is ready!
Error occured! [Errno 2] No such file or directory: 'favicon.ico'
Server is ready!
/index.html sent to 192.168.1.60
Server is ready!
Error occured! [Errno 2] No such file or directory: 'favicon.ico'
Server is ready!
Error occured! [Errno 2] No such file or directory: 'favicon.ico'
Server is ready!
oguzhan@linux:~/Code/communication-networks/web_server$
```

```
oguzhan@linux:~/Code/communication-networks/web_server$
oguzhan@linux:~/Code/communication-networks/web_server$ python3 client.py
HTTP/1.1 200 OK
Connection: keep-alive
Date: 2020-11-29 21:04:19.977772
Server: MyServer
Content-Length: 235
Content-Type: text/html

<!DOCTYPE html>
<html>
<body>
  <h1>
    Greetings from my simple web server!
  </h1>
  <p>
    This is my simple web server for term project of CSE476 course.<br><br>
    Best regards.<br><br>
    Oguzhan Agkus
  </p>
</body>
</html>
oguzhan@linux:~/Code/communication-networks/web_server$ python3 client.py
HTTP/1.1 200 OK
Connection: keep-alive
Date: 2020-11-29 21:04:51.328003
Server: MyServer
Content-Length: 235
Content-Type: text/html

<!DOCTYPE html>
<html>
<body>
  <h1>
    Greetings from my simple web server!
  </h1>
  <p>
    This is my simple web server for term project of CSE476 course.<br><br>
    Best regards.<br><br>
    Oguzhan Agkus
  </p>
</body>
</html>
oguzhan@linux:~/Code/communication-networks/web_server$
```

Assignment 2

UDP Pinger

I have implemented a ping server and client which uses UDP instead of ICMP. Actually I modified the server code. I think it has some extra and missing parts. But I did not change the main purpose of the server. It accetps connections, reads data, simulates data loss and sends back the incoming data.

```
import sys, signal, socket, random

# Set global variables
server_address = ("", 12000)
server_socket = None

# If CTRL+C occurs
def signal_handler(signal, frame):
  server_socket.close()
  sys.exit(0)

# Prepare UDP socket
try:
  server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
  server_socket.bind(server_address)
except Exception as error:
  print("Error occured!", error)
  sys.exit(1)

# Set signal handler
signal.signal(signal.SIGINT, signal_handler)

# Main loop
while True:
  message, address = server_socket.recvfrom(1024)
  print(message.decode())

  # Simulate package loss
  loss = random.randint(0, 10)
  if loss < 4:
    continue

  server_socket.sendto(message, address)
```

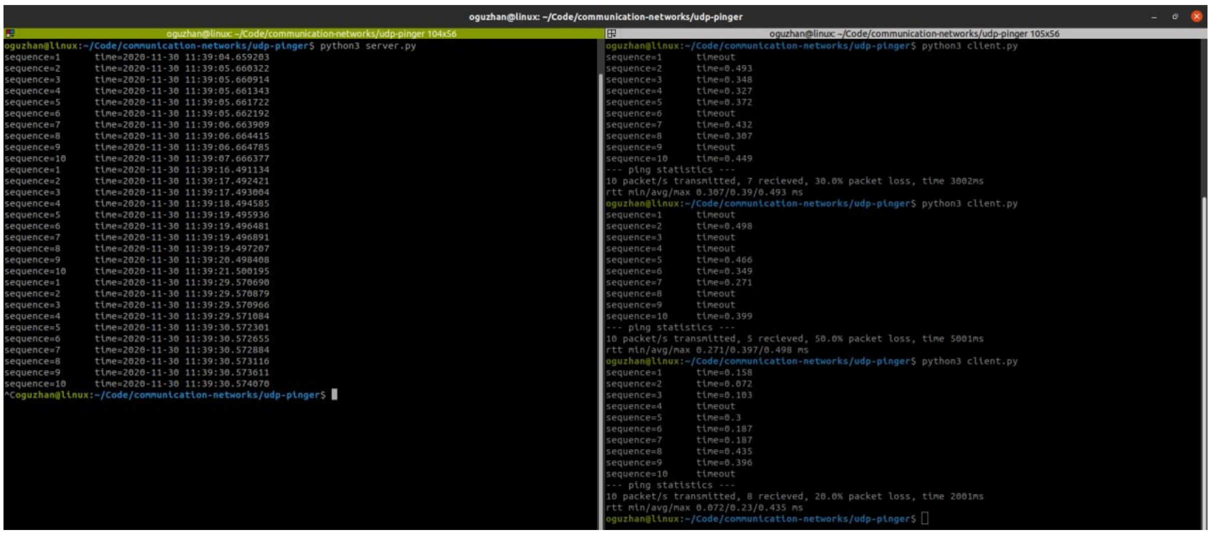
The client sends 10 packets. It has 1 second timeout. The sending sequence cannot be interrupted.

```
import sys, signal, socket, socket, datetime

# Set global variables
server_address = ("", 12000)
client_socket = None
packet_count = 10
rtt_table = []
lost = 0

# If CTRL+C occurs
def signal_handler(signal, frame):
  pass

# Prepare UDP socket
try:
  client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  client_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
  client_socket.settimeout(1)
except Exception as error:
  print("Error occured!", error)
  sys.exit(1)

# Set signal handler
signal.signal(signal.SIGINT, signal_handler)

# Sequential package sending
for i in range(1, packet_count + 1):
  temp = datetime.datetime.now()
  message = "sequence={}\ttime={}".format(i, temp)
  client_socket.sendto(message.encode(), server_address)
  try:
    reply_message, temp_address = client_socket.recvfrom(1024)
    duration = datetime.datetime.now() - temp
    rtt_table.append(duration)
    print("sequence={}\ttime={}ms".format(i, duration.microseconds / 1000))
  except socket.timeout:
    print("sequence={}\ttimeout".format(i))
    lost += 1

# Close socket
client_socket.close()
```

Also I calculate the ping statistics:

```
# Calculate statistics
total = datetime.timedelta()
minimum = datetime.timedelta()
maximum = datetime.timedelta()
average = datetime.timedelta()

if len(rtt_table):
  total = sum(rtt_table, datetime.timedelta())
  minimum = min(rtt_table)
  maximum = max(rtt_table)
  average = total / len(rtt_table)

time = lost * 1000 + total.microseconds / 1000

# Print statistics
print("--- ping statistics ---")
print("{} packet/s transmitted, {} recieved, {}% packet loss, time {}ms".format(packet_count, packet_count - lost, lost / packe
print("rtt min/avg/max {}/{}/{} ms".format(minimum.microseconds / 1000, average.microseconds / 1000, maximum.microseconds / 100
```

An example output is below. The left side is server and the right side is client. Also I show RTT values in ms. Because second is too big unit to measure delays, especially in local network.


oguzhan@linux: ~/Code/communication-networks/udp-pinger
oguzhan@linux: ~/Code/communication-networks/udp-pinger 104x56
oguzhan@linux: ~/Code/communication-networks/udp-pinger 105x56

```
oguzhan@linux:~/Code/communication-networks/udp-pinger$ python3 server.py
sequence=1      time=2020-11-30 11:39:04.659203
sequence=2      time=2020-11-30 11:39:05.660322
sequence=3      time=2020-11-30 11:39:05.660914
sequence=4      time=2020-11-30 11:39:05.661343
sequence=5      time=2020-11-30 11:39:05.661722
sequence=6      time=2020-11-30 11:39:05.662192
sequence=7      time=2020-11-30 11:39:06.663909
sequence=8      time=2020-11-30 11:39:06.664415
sequence=9      time=2020-11-30 11:39:06.664785
sequence=10     time=2020-11-30 11:39:07.666377
sequence=1      time=2020-11-30 11:39:16.491134
sequence=2      time=2020-11-30 11:39:17.492421
sequence=3      time=2020-11-30 11:39:17.493004
sequence=4      time=2020-11-30 11:39:18.494585
sequence=5      time=2020-11-30 11:39:19.495936
sequence=6      time=2020-11-30 11:39:19.496481
sequence=7      time=2020-11-30 11:39:19.496891
sequence=8      time=2020-11-30 11:39:19.497207
sequence=9      time=2020-11-30 11:39:20.498408
sequence=10     time=2020-11-30 11:39:21.500195
sequence=1      time=2020-11-30 11:39:29.570690
sequence=2      time=2020-11-30 11:39:29.570879
sequence=3      time=2020-11-30 11:39:29.570966
sequence=4      time=2020-11-30 11:39:29.571084
sequence=5      time=2020-11-30 11:39:30.572301
sequence=6      time=2020-11-30 11:39:30.572655
sequence=7      time=2020-11-30 11:39:30.572884
sequence=8      time=2020-11-30 11:39:30.573116
sequence=9      time=2020-11-30 11:39:30.573611
sequence=10     time=2020-11-30 11:39:30.574070
^Coguzhan@linux:~/Code/communication-networks/udp-pinger$
```

```
oguzhan@linux:~/Code/communication-networks/udp-pinger$ python3 client.py
sequence=1      timeout
sequence=2      time=0.493
sequence=3      time=0.348
sequence=4      time=0.327
sequence=5      time=0.372
sequence=6      timeout
sequence=7      time=0.432
sequence=8      time=0.307
sequence=9      timeout
sequence=10     time=0.449
--- ping statistics ---
10 packet/s transmitted, 7 recieved, 30.0% packet loss, time 3002ms
rtt min/avg/max 0.307/0.39/0.493 ms
oguzhan@linux:~/Code/communication-networks/udp-pinger$ python3 client.py
sequence=1      timeout
sequence=2      time=0.498
sequence=3      timeout
sequence=4      timeout
sequence=5      time=0.466
sequence=6      time=0.349
sequence=7      time=0.271
sequence=8      timeout
sequence=9      timeout
sequence=10     time=0.399
--- ping statistics ---
10 packet/s transmitted, 5 recieved, 50.0% packet loss, time 5001ms
rtt min/avg/max 0.271/0.397/0.498 ms
oguzhan@linux:~/Code/communication-networks/udp-pinger$ python3 client.py
sequence=1      time=0.158
sequence=2      time=0.072
sequence=3      time=0.103
sequence=4      timeout
sequence=5      time=0.3
sequence=6      time=0.187
sequence=7      time=0.187
sequence=8      time=0.435
sequence=9      time=0.396
sequence=10     timeout
--- ping statistics ---
10 packet/s transmitted, 8 recieved, 20.0% packet loss, time 2001ms
rtt min/avg/max 0.072/0.23/0.435 ms
oguzhan@linux:~/Code/communication-networks/udp-pinger$
```

UDP Heartbeat

Additionally, I implemented a heartbead application. The client sends signals periodically. These signals are following by the server. Server tracks the time difference between these signals. If there is delay, it is reported. Also if there is no signal for a while, the server tought the client has stopped.

Heartbeat client:

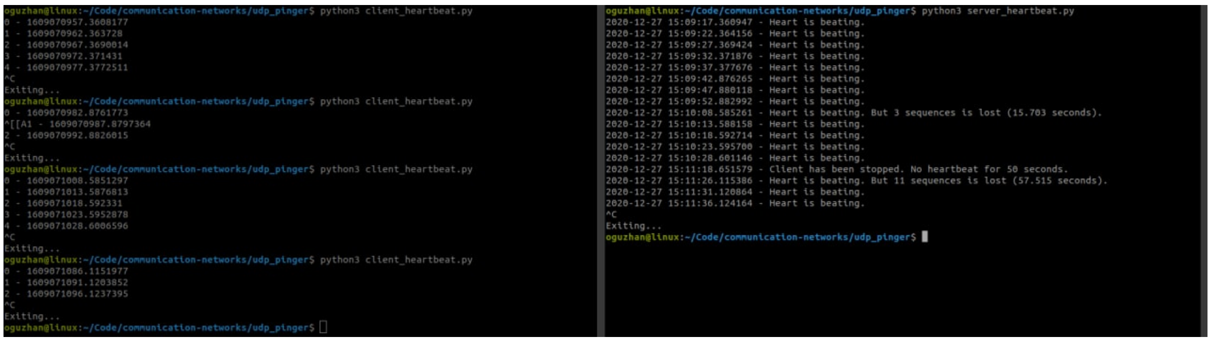
```
import sys, signal, socket
import time

# Set global variables
server_address = ("", 13000)
client_socket = None
interval = 5
sequence = 0
```

```
# If CTRL+C occurs
def signal_handler(signal, frame):
  print("\nExiting...")
  client_socket.close()
  sys.exit(0)

# Prepare UDP socket
try:
  client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  client_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
  client_socket.settimeout(1)
except Exception as error:
  print("Error occured!", error)
  sys.exit(1)

# Set signal handler
signal.signal(signal.SIGINT, signal_handler)

while True:
  epoch_time = time.time()
  message = str(sequence) + " - " + str(epoch_time)
  client_socket.sendto(message.encode(), server_address)

  print(message)

  sequence += 1
  time.sleep(interval)
```

Heartbeat server:

```
import sys, signal, socket
import datetime

# Set global variables
server_address = ("", 13000)
server_socket = None
previous = 0
period = 5
timeout = period * 10
timeout_period = 0

# If CTRL+C occurs
def signal_handler(signal, frame):
  print("\nExiting...")
  server_socket.close()
  sys.exit(0)

# Prepare UDP socket
try:
  server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
  server_socket.bind(server_address)
  server_socket.settimeout(timeout)
except Exception as error:
  print("Error occured!", error)
  sys.exit(1)

# Set signal handler
signal.signal(signal.SIGINT, signal_handler)

# Main loop
while True:
  try:
    message, address = server_socket.recvfrom(1024)
    temp_time = float(message.decode().split(" - ")[1])
    difference = round(temp_time - previous, 3)

    print(datetime.datetime.now(), "-", end=" ")
    if previous != 0 and difference > period + 0.1:
      missing = difference / period
      print("Heart is beating. But", int(missing), "sequences is lost ({} seconds).".format(difference))
    else:
      print("Heart is beating.")

    previous = temp_time
    timeout_period = 0
  except Exception as error:
    timeout_period += 1
    print(datetime.datetime.now(), "-", end=" ")
    print("Client has been stopped. No heartbeat for {} seconds.".format(timeout * timeout_period))
```

An example output is below. The left side is client and the right side is server. The client sends epoch time in every 5 seconds. The server listens incoming signals from the client. If the signal is reaching properly it reports as "Heart is beating.", else it reports the suspicious situation. If the signal did not recieve for a while, it means the client has been stopped.



```
oguzhan@linux:~/Code/communication-networks/udp_pinger$ python3 client_heartbeat.py
0 - 1609070957.3608177
1 - 1609070962.363728
2 - 1609070967.3690014
3 - 1609070972.371431
4 - 1609070977.3772511
^C
Exiting...
oguzhan@linux:~/Code/communication-networks/udp_pinger$ python3 client_heartbeat.py
0 - 1609070982.8761773
^[[A1 - 1609070987.8797364
2 - 1609070992.8826015
^C
Exiting...
oguzhan@linux:~/Code/communication-networks/udp_pinger$ python3 client_heartbeat.py
0 - 1609071008.5851297
1 - 1609071013.5876813
2 - 1609071018.592331
3 - 1609071023.5952878
4 - 1609071028.6006596
^C
Exiting...
oguzhan@linux:~/Code/communication-networks/udp_pinger$ python3 client_heartbeat.py
0 - 1609071086.1151977
1 - 1609071091.1203852
2 - 1609071096.1237395
^C
Exiting...
oguzhan@linux:~/Code/communication-networks/udp_pinger$ 
```

```
oguzhan@linux:~/Code/communication-networks/udp_pinger$ python3 server_heartbeat.py
2020-12-27 15:09:17.360947 - Heart is beating.
2020-12-27 15:09:22.364156 - Heart is beating.
2020-12-27 15:09:27.369424 - Heart is beating.
2020-12-27 15:09:32.371876 - Heart is beating.
2020-12-27 15:09:37.377676 - Heart is beating.
2020-12-27 15:09:42.876265 - Heart is beating.
2020-12-27 15:09:47.880118 - Heart is beating.
2020-12-27 15:09:52.882992 - Heart is beating.
2020-12-27 15:10:08.585261 - Heart is beating. But 3 sequences is lost (15.703 seconds).
2020-12-27 15:10:13.588158 - Heart is beating.
2020-12-27 15:10:18.592714 - Heart is beating.
2020-12-27 15:10:23.595700 - Heart is beating.
2020-12-27 15:10:28.601146 - Heart is beating.
2020-12-27 15:11:18.651579 - Client has been stopped. No heartbeat for 50 seconds.
2020-12-27 15:11:26.115386 - Heart is beating. But 11 sequences is lost (57.515 seconds).
2020-12-27 15:11:31.120864 - Heart is beating.
2020-12-27 15:11:36.124164 - Heart is beating.
^C
Exiting...
oguzhan@linux:~/Code/communication-networks/udp_pinger$ 
```

Assignment 3

Mail Client

I implemented a mail client which uses only sockets. It sends STMP commands over a TCP socket. To send a mail, you need to login on your mail server. I defined two mail servers: Gmail and Office 365. You can choose one of the mor add yours. Also you should define your username and password for the server. The reciever and mail informations are globally defined variables. There is no user input part. I have some handler functions to close socket and STMP session securely. Also I added the functionality to send images. I used MIME protocol to send images. Because we need to seperate pure text and base64 encoded image data. Actually MIME is very simple protocol. A MIME object can contain diffrent types of multimedia items, they are seperated with a spesific key. I believe that it is easy to implement but it is not purpose of this assignment.

```
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
import os, sys, signal, socket, ssl, time, base64

# Set your authentication information
username = "******"
password = "******"

# Choose your mail server
gmail = ("smtp.gmail.com", 587)
office = ("smtp.office365.com", 587)
mail_server = gmail

# Set email details
reciever = "agkusoguzhan@gmail.com"
subject = "Test Mail From My Client"
message = "This is my simple SMTP client for term project of CSE476 course.\n\nBest regards.\n\nOguzhan Agkus\n\n"
attachment_file = "./1.png"

# Global declaration for using in the functions below
client_socket = None
content = None

# After connection established, if server returns unexpected reply then call this function
def exit_handler(message = None, exit_code = 0):
  if message:
    print(message)

  client_socket.send("QUIT\r\n".encode())
  client_socket.close()
  sys.exit(exit_code)

# If CTRL+C occurs
def signal_handler(signal, frame):
  exit_handler()
```

Connect and login to server:

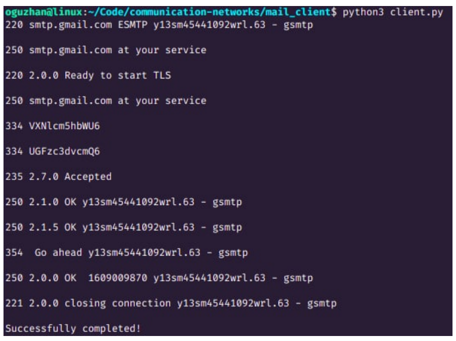
```
# Establish TCP connection
try:
  client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  client_socket.connect(mail_server)
except Exception as error:
  print("Error occured!", error)
  sys.exit(1)

# Set signal handler
signal.signal(signal.SIGINT, signal_handler)

# Get first data
data = client_socket.recv(1024).decode()
print(data)
if data[:3] != "220":
  exit_handler("Reply is not 220 after connection established!", 1)

# Send HELO command
client_socket.send("HELO MYSERVER\r\n".encode())
data = client_socket.recv(1024).decode()
print(data)
if data[:3] != "250":
  exit_handler("Reply is not 250 after hello!", 1)

# Send STARTTLS command, if encryption is possible, start a new session
client_socket.send("STARTTLS\r\n".encode())
data = client_socket.recv(1024).decode()
print(data)
if data[:3] == "220":
  context = ssl._create_stdlib_context(certfile=None, keyfile=None)
  client_socket = context.wrap_socket(client_socket)

  client_socket.send("HELO MYSERVER\r\n".encode())
  data = client_socket.recv(1024).decode()
  print(data)
  if data[:3] != "250":
    exit_handler("Reply is not 250 after TLS connection!", 1)

# Send AUTH command
client_socket.send("AUTH LOGIN\r\n".encode())
data = client_socket.recv(1024).decode()
print(data) # It says send username

# Send username first, encode it with base64
client_socket.send(base64.b64encode(username.encode()))
client_socket.send("\r\n".encode())
data = client_socket.recv(1024).decode()
print(data) # It says send password

# Send password then, encode it with base64
client_socket.send(base64.b64encode(password.encode()))
client_socket.send("\r\n".encode())
data = client_socket.recv(1024).decode()
print(data) # Accepted or failed
if data[:3] != "235":
  exit_handler("Authentication failed!", 1)
```

Send mail details:

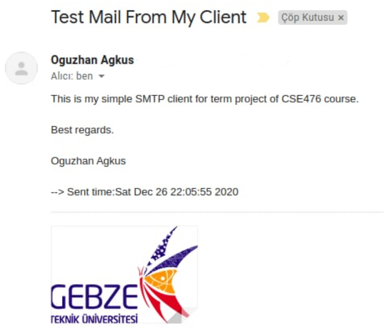
```
# Send MAIL FROM command
client_socket.send("MAIL FROM:<{}>\r\n".format(username).encode())
data = client_socket.recv(1024).decode()
print(data)

# Send RCPT TO command
client_socket.send("RCPT TO:<{}>\r\n".format(reciever).encode())
data = client_socket.recv(1024).decode()
print(data)

# Send DATA command
client_socket.send("DATA\r\n".encode())
data = client_socket.recv(1024).decode()
print(data)

# Timestamp to compare outgoing with incoming mail
timestamp = "--> Sent time: " + time.asctime()
```

```
# Fill content
content = MIMEMultipart()
content["From"] = "Oguzhan Agkus"
content["To"] = reciever
content["Subject"] = subject

text = MIMEText(message + timestamp)
content.attach(text)

# Attach image
if attachment_file != None:
  try:
    with open(attachment_file, "rb") as image_file:
      image_data = image_file.read()
      image = MIMEImage(image_data, name=os.path.basename(attachment_file))
      content.attach(image)
  except Exception as error:
    print("Cannot add image to mail: ", error)

# Convert MIME object to string
content = content.as_string()

# Send content
client_socket.send(content.encode())
client_socket.send("\r\n.\r\n".encode())
data = client_socket.recv(1024).decode()
print(data)

# Send QUIT command
client_socket.send("QUIT\r\n".encode())
data = client_socket.recv(1024).decode()
print(data)

client_socket.close()

print("Successfully completed!")
```

Example output:



```
oguzhan@linux:~/Code/communication-networks/mail_client$ python3 client.py
220 smtp.gmail.com ESMTP y13sm45441092wrl.63 - gsmtp

250 smtp.gmail.com at your service

220 2.0.0 Ready to start TLS

250 smtp.gmail.com at your service

334 VXNlcm5hbWU6

334 UGFzc3dvcmQ6

235 2.7.0 Accepted

250 2.1.0 OK y13sm45441092wrl.63 - gsmtp

250 2.1.5 OK y13sm45441092wrl.63 - gsmtp

354  Go ahead y13sm45441092wrl.63 - gsmtp

250 2.0.0 OK  1609009870 y13sm45441092wrl.63 - gsmtp

221 2.0.0 closing connection y13sm45441092wrl.63 - gsmtp

Successfully completed!
```



Test Mail From My Client Çöp Kutusu ×

Oguzhan Agkus
Alıcı: ben

This is my simple SMTP client for term project of CSE476 course.

Best regards.

Oguzhan Agkus

--> Sent time:Sat Dec 26 22:05:55 2020