Nama : Ivan Fairuz Adinata

NRP : 5025221167

Kelas : Progjar D

LAPORAN TUGAS 1 PROGJAR D

Link Github : https://github.com/mlikiwe/progjar-d-2025-tugas2.git

1. Membuat program Time Server yang nantinya bisa memproses request dari client dan mengembalikan output waktu saat ini kepada client.
   a. Membuka port di port 45000 dengan transport TCP

      Method _init_

```
self.my_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

      Method run

```
self.my_socket.bind(('0.0.0.0',45000))
self.my_socket.listen(1)
```

      Beberapa baris kode di atas berada pada method _init_ dan method run yang berada pada class server. Di method _init_(), kita akan membuka socket dengan perintah socket.AF_INET dan socket.SOCK_STREAM yang menandakan bahwa kita akan menggunakan protokol transport TCP. Selanjutnya, di method run(), kita akan memanggil fungsi bind() dengan argument ('0.0.0.0', 45000) yang akan mengikat socket ke semua interface lokal pada port 45000. 0.0.0.0 akan menjadikan server dapat menerima seluruh koneksi dari IP klien manapun. Selanjutnya, listen(1) mempersiapkan socket untuk menerima koneksi masuk. Meskipun parameternya 1, server nantinya akan bisa menangani banyak client sekaligus karena Ketika nantinya koneksi bertambah, akan dibuatkan thread baru untuk menanganinya.

   b. Server harus dapat melayani request yang concurrent, gunakan contoh.

```
clt = ProcessTheClient(self.connection, self.client_address)
clt.start()
self.the_clients.append(clt)
```

```
class ProcessTheClient(threading.Thread):
    def __init__(self,connection,address):
        self.connection = connection
        self.address = address
        threading.Thread.__init__(self)
```

      Di potongan kode yang atas, dapat dilihat bahwa setiap koneksi dari client di-accept oleh server, maka server akan membuat instance atau memanggil class ProcessTheClient. Class tersebut merupakan subclass dari threading.Thread. Hal ini

memungkinkan untuk setiap koneksi nantinya akan berjalan pada thread yang terpisah, sehingga server kita dapat melayani request dari banyak client sekaligus secara concurrent. Method start di sini akan memanggil method run() yang telah dijalankan sebelumnya dan akan menangani komunikasi dengan client secara independen.

c. Ketentuan request yang dilayani
    i. Diawali dengan string “TIME dan diakhiri dengan karakter 13 dan karakter
    ii. ii. Setiap request dapat diakhiri dengan string “QUIT” yang diakhiri dengan karakter 13 dan 10

```
if (request_string.startswith("TIME") and
request_string.endswith("\r\n")):
        ...
if (request_string.startswith("QUIT") and
request_string.endswith("\r\n")):
        ...
```

Di sini, request yang valid harus diawali dengan kata “TIME” dan diakhiri dengan character carriage return dan new line (“\r\n”). Hal ini disesuaikan dengan standar protokol teks sederhana seperti HTTP. Untuk memeriksanya, kita akan buat pengondisian if, di mana untuk melakukan cek pada request client, kita memakai bantuan method startwith() dan endwith(). Selain itu, untuk mengakhiri request oleh client, request akan diawali dengan “QUIT” dan diakhiri juga oleh (“\r\n”). Syarat tersebut juga kita masukkan ke pengondisian if selanjutnya.

d. Server akan merespon dengan jam dengan ketentuan
    i. Dalam bentuk string (UTF-8)
    ii. Diawali dengan “JAM”
    iii. Berisikan info jam dalam format “hh:mm:ss” dan diakhiri dengan karakter 13 dan karakter 10

```
if (request_string.startswith("TIME") and
request_string.endswith("\r\n")):
    from datetime import datetime
    now = datetime.now()
    waktu = now.strftime("%H:%M:%S")
    balas=f"JAM {waktu}\r\n"
```

Respon dari server akan diberikan dalam bentuk string UTF-8, dengan ketentuan di awali dengan kata “JAM” lalu diikuti informasi jam saat ini dengan format “hh:mm:ss”. Untuk mengirim response balik sesuai ketentuan, kita akan memanfaatkan dungsi datetime.now() yang didapatkan dari package datetime.

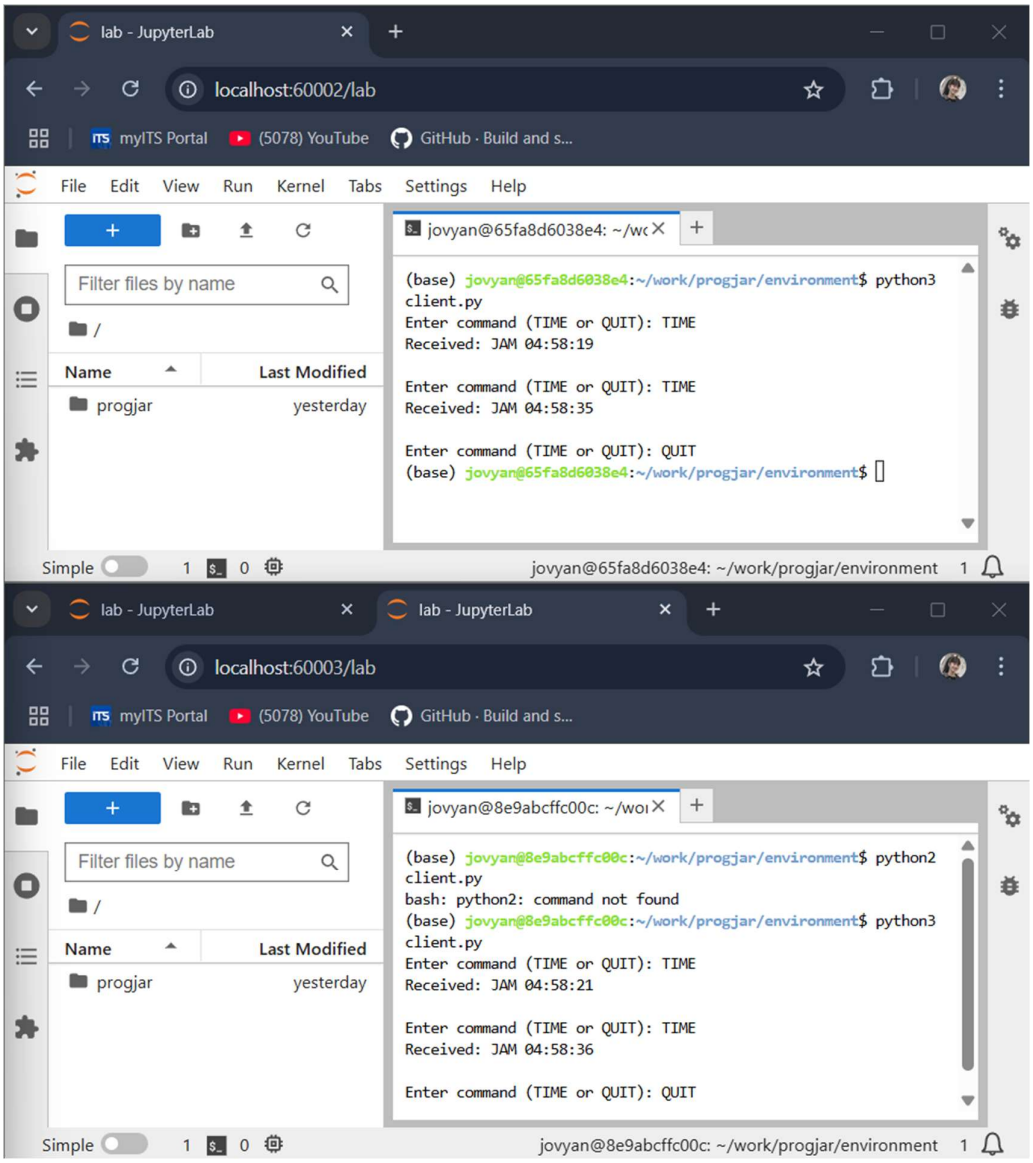
Selanjutnya, kita pakai bantuan strftime() dengan parameter “%H:%M:%S”. Dengan method/fungsi tersebut, kita dapat melakukan formatting datetime sesuai ketentuan. Selanjutnya, sesuai ketentuan juga, di akhir response, kita menambahkan carriage return dan new line (“\r\n”). Dengan itu, server akan dapat melakukan response balik kepada client dengan memberi informasi jam saat ini sesuai ketentuan.

2. Capture Hasilnya pada wireshark

Untuk membuktikan bahwa program time server dapat bekerja secara concurrent, kita akan menjalankan client.py pada mesin-2 dan mesin-3 secara bersamaan. Client.py akan kita buat dengan simple, di mana akan membuka socket dan melakukan koneksi ke ip mesin-1 sebagai server dan berjalan pada port yang sama yakni 45000. Selanjutnya, terdapat pengondisian pengiriman message untuk di response oleh server sebagai berikut:

```
if message == "TIME":
    client_socket.sendall(f"TIME\r\n".encode('utf-8'))
    response = client_socket.recv(32)
    print("Received:", response.decode('utf-8'))
elif message == "QUIT":
    client_socket.sendall(f"QUIT\r\n".encode('utf-8'))
    break
```

Kita akan jalankan server.py di mesin-1 sebelum menjalankan kedua client secara bersamaan. Setelah kedua client berhasil connect, kita akan mencoba mengirim pesan secara bergantian dari mesin-1 dan mesin-2 seperti gambar di bawah



Hasil capture di wireshark

Source	Destination	Protocol	Length	Info
a2:55:45:6c:df:b7	Broadcast	ARP	42	Who has 172.16.16.101? Tell 172.16.16.102
66:b4:e8:80:15:02	a2:55:45:6c:df:b7	ARP	42	172.16.16.101 is at 66:b4:e8:80:15:02
172.16.16.102	172.16.16.101	TCP	74	36754 → 45000 [SYN] Seq=0 Win=64240 Len=0 M
172.16.16.101	172.16.16.102	TCP	74	45000 → 36754 [SYN, ACK] Seq=0 Ack=1 Win=65
172.16.16.102	172.16.16.101	TCP	66	36754 → 45000 [ACK] Seq=1 Ack=1 Win=64256 L
66:b4:e8:80:15:02	a2:55:45:6c:df:b7	ARP	42	Who has 172.16.16.102? Tell 172.16.16.101
a2:55:45:6c:df:b7	66:b4:e8:80:15:02	ARP	42	172.16.16.102 is at a2:55:45:6c:df:b7
6a:ed:71:b7:72:7e	Broadcast	ARP	42	Who has 172.16.16.101? Tell 172.16.16.103
66:b4:e8:80:15:02	6a:ed:71:b7:72:7e	ARP	42	172.16.16.101 is at 66:b4:e8:80:15:02
172.16.16.103	172.16.16.101	TCP	74	33732 → 45000 [SYN] Seq=0 Win=64240 Len=0 M
172.16.16.101	172.16.16.103	TCP	74	45000 → 33732 [SYN, ACK] Seq=0 Ack=1 Win=65
172.16.16.103	172.16.16.101	TCP	66	33732 → 45000 [ACK] Seq=1 Ack=1 Win=64256 L
66:b4:e8:80:15:02	6a:ed:71:b7:72:7e	ARP	42	Who has 172.16.16.103? Tell 172.16.16.101
6a:ed:71:b7:72:7e	66:b4:e8:80:15:02	ARP	42	172.16.16.103 is at 6a:ed:71:b7:72:7e

Source	Destination	Protocol	Lengtl	Info
172.16.16.102	172.16.16.101	TCP	72	36754 → 45000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=6 T
172.16.16.101	172.16.16.102	TCP	66	45000 → 36754 [ACK] Seq=1 Ack=7 Win=65280 Len=0 TSval=
172.16.16.101	172.16.16.102	TCP	80	45000 → 36754 [PSH, ACK] Seq=1 Ack=7 Win=65280 Len=14
172.16.16.102	172.16.16.101	TCP	66	36754 → 45000 [ACK] Seq=7 Ack=15 Win=64256 Len=0 TSval
172.16.16.103	172.16.16.101	TCP	72	33732 → 45000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=6 T
172.16.16.101	172.16.16.103	TCP	66	45000 → 33732 [ACK] Seq=1 Ack=7 Win=65280 Len=0 TSval=
172.16.16.101	172.16.16.103	TCP	80	45000 → 33732 [PSH, ACK] Seq=1 Ack=7 Win=65280 Len=14
172.16.16.103	172.16.16.101	TCP	66	33732 → 45000 [ACK] Seq=7 Ack=15 Win=64256 Len=0 TSval

172.16.16.101	172.16.16.102	TCP	66 45000 → 36754 [FIN, ACK] Seq=2
172.16.16.102	172.16.16.101	TCP	66 36754 → 45000 [FIN, ACK] Seq=1
172.16.16.101	172.16.16.102	TCP	66 45000 → 36754 [ACK] Seq=30 Ack
172.16.16.102	172.16.16.101	TCP	66 36754 → 45000 [ACK] Seq=20 Ack
172.16.16.103	172.16.16.101	TCP	72 33732 → 45000 [PSH, ACK] Seq=1
172.16.16.103	172.16.16.101	TCP	66 33732 → 45000 [FIN, ACK] Seq=1
172.16.16.101	172.16.16.103	TCP	66 45000 → 33732 [FIN, ACK] Seq=2
172.16.16.103	172.16.16.101	TCP	66 33732 → 45000 [ACK] Seq=20 Ack

Di paling atas terdapat ARP ke broadcast untuk memberi tahu server mana mesin-2 dan mesin-3, setelah itu mesin-2 dan mesin-3 dengan mesin-1 akan melakukan handshake tiga arah. Ketika sudah berhasil terkoneksi, maka pada mesin-2 dan mesin-3 akan dapat mengirimkan pesan “TIME” ke mesin-1. Pada beberapa packet setelahnya, terjadilah pertukaran data, di mana mesin-2 dan mesin-3 mengirimkan message TIME, dan dibalas waktu saat ini oleh mesin-1.Ketika sudah selesai, mesin-2 dan mesin-3 akan mengirimkan message QUIT untuk mengakhiri percakapan. Hal itu dapat dilihat dari traffic packet paling bawah, keduanya berkomunikasi dengan mengirimkan FIN, ACK yang digunakan untuk mengakhiri percakapan dan memutus koneksi.