Nama : Muhammad Gesang Ridho Widigdo

NRP : 5025221216

Kelas : Pemrograman Jaringan (D)

Link Github : https://github.com/gesangwidigdo/progjar-d-tugas-2

LAPORAN TUGAS PEMROGRAMAN JARINGAN 2

1. Buat program time server
   a) Untuk membuka koneksi di port 45000 dengan transport TCP, dilakukan dengan mengikat server menggunakan fungsi socket.bind() dengan hostnya diatur menjadi 0.0.0.0 agar server dapat menerima koneksi dari semua IP, dan portnya diatur menjadi 45000 seperti di soal. Pada inisialisasi server, digunakan socket.SOCK_STREAM untuk menjamin bahwa program hanya menerima koneksi TCP sehingga server bersifat reliable, urut, dan tidak ada data yang duplikat.

```
37    def __init__(self):
38      self.the_clients = []
39      self.my_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
40      threading.Thread.__init__(self)
41
42    def run(self):
43      self.my_socket.bind(('0.0.0.0',45000))
```

b) Server mendengarkan tiap koneksi baru menggunakan fungsi listen(). Tiap koneksi yang masuk akan ditangani oleh thread baru yang diinisialisasi oleh subclass threading.Thread, sehingga server dapat menangani banyak client sekaligus dalam satu waktu (concurrent) menggunakan multithread (satu thread menangani satu client). Thread melalui objek clt akan menjalankan method run() pada class ProcessTheClient menggunakan clt.start() untuk menerima, memproses, dan membalas permintaan client. Kemudian objek client tersebut disimpan ke array the_clients sebagai daftar client aktif yang ditangani oleh server.

```
36  class Server(threading.Thread):
37    def __init__(self):
38      self.the_clients = []
39      self.my_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
40      threading.Thread.__init__(self)
41
42    def run(self):
43      self.my_socket.bind(('0.0.0.0',45000))
44      self.my_socket.listen(1)
45      while True:
46        self.connection, self.client_address = self.my_socket.accept()
47        logging.warning(f"connection from {self.client_address}")
48
49        clt = ProcessTheClient(self.connection, self.client_address)
50        clt.start()
51        self.the_clients.append(clt)
```

Hasil:

Client 1 (mesin-2 → 172.16.16.102):

```
(base) jovyan@5e01b9ab11c6:~/work/progjar/tugas-2$ python3 client.py
TIME/QUIT:
```

Client 2 (mesin-3 → 172.16.16.103):

```
(base) jovyan@70961f572ca7:~/work/progjar/tugas-2$ python3 client.py
TIME/QUIT:
```

Server (mesin-1 → 172.16.16.101):

```
(base) jovyan@b01fac93c31c:~/work/progjar/tugas-2$ python3 server.py
connection from ('172.16.16.102', 56450)
connection from ('172.16.16.103', 37522)
```

c) Request yang dilayani harus antara TIME atau QUIT dan diakhiri oleh karakter 13 pada ASCII (\r) atau carriage return dan karakter 10 (\n) atau line feed, yaitu "\r\n" yang sama saja artinya seperti baris baru, sehingga tidak perlu ditulis lagi dalam perbandingan. Request yang diterima server didekodekan terlebih dahulu dengan UTF-8, kemudian dibandingkan. Jika request merupakan TIME, maka server akan memproses untuk mendapatkan waktu saat ini dan mengirimkannya ke client. Kemudian client menerima data waktu saat ini sebagai response. Sedangkan jika requestnya adalah QUIT, maka akan menutup koneksi dari client ke server. Jika tidak diantara keduanya, maka server mengirimkan pesan bahwa requestnya invalid.

Server:

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

    def run(self):
        while True:
            data = self.connection.recv(32)
            if data:
                # get request
                request = data.decode()
                logging.info(f"Request from client {self.address}: {request}")

                if request == "TIME":
                    current_time = time.strftime("%H:%M:%S")
                    response = f"JAM {current_time}"
                    self.connection.sendall(response.encode())
                    logging.info(f"Sent to {self.address}: {response.strip()}")
                elif request == "QUIT":
                    self.connection.close()
                    break
                else:
                    self.connection.sendall(b"Invalid request")
            else:
                break
        self.connection.close()
```

Client:

```python
import socket
import logging

def main():
    server_address = ('172.16.16.101', 45000)
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect(server_address)

    try:
        while True:
            user_input = input("TIME/QUIT: ").strip().upper()

            sock.sendall(user_input.encode())

            if user_input == "QUIT":
                logging.info("Connection closed")
                break

            # accept response from server
            response = sock.recv(1024).decode()
            print(response)

    finally:
        sock.close()

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

d) Saat server menerima request “TIME\r\n”, server harus mengirim response berupa waktu saat ini ke client. Pada server, digunakan fungsi time.strftime() dengan parameter berupa format "%H:%M:%S" yang akan mengembalikan format waktu “hh:mm:ss”. Format waktu yang didapat kemudian dimasukkan ke dalam format data yang akan dikirim, yaitu “JAM {current_time}” agar sesuai dengan protocol response pada soal. Data tersebut diencode ke dalam bentuk UTF-8 menggunakan fungsi encode() sebelum dikirim ke client menggunakan fungsi sendall().

```
# get request
request = data.decode()
logging.info(f"Request from client {self.address}: {request}")

if request == "TIME":
  current_time = time.strftime("%H:%M:%S")
  response = f"JAM {current_time}"
  self.connection.sendall(response.encode())
  logging.info(f"Sent to {self.address}: {response.strip()}")
```

Hasil saat run:

Client 1 (mesin-2 → 172.16.16.102):

```
(base) jovyan@5e01b9ab11c6:~/work/progjar/tugas-2$ python3 client.py
TIME/QUIT: TIME
JAM 22:51:35
TIME/QUIT: TIME
JAM 22:51:38
TIME/QUIT: TIME
JAM 22:51:41
TIME/QUIT:
```

Client 2 (mesin-3 → 172.16.16.103):

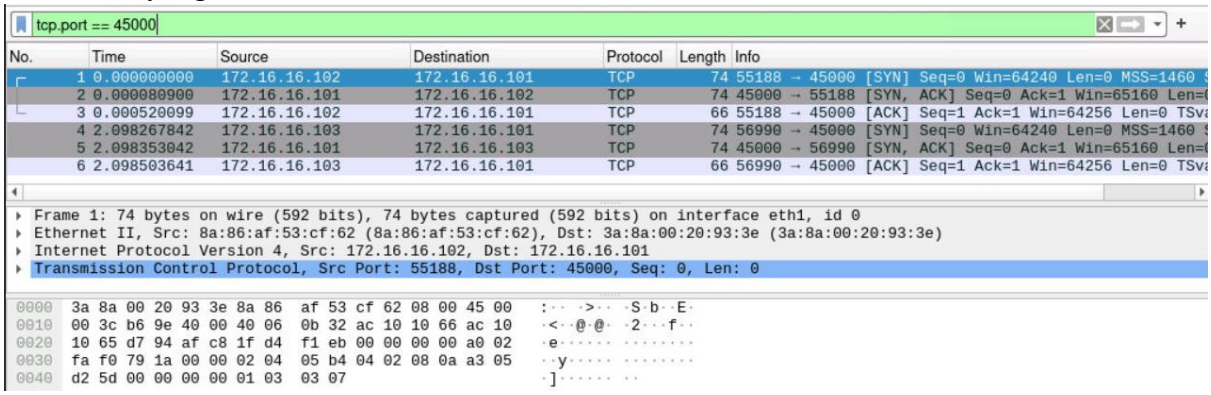
```
(base) jovyan@70961f572ca7:~/work/progjar/tugas-2$ python3 client.py
TIME/QUIT: TIME
JAM 22:51:36
TIME/QUIT: TIME
JAM 22:51:40
TIME/QUIT: TIME
JAM 22:51:43
TIME/QUIT:
```

Server (mesin-1 → 172.16.16.101):

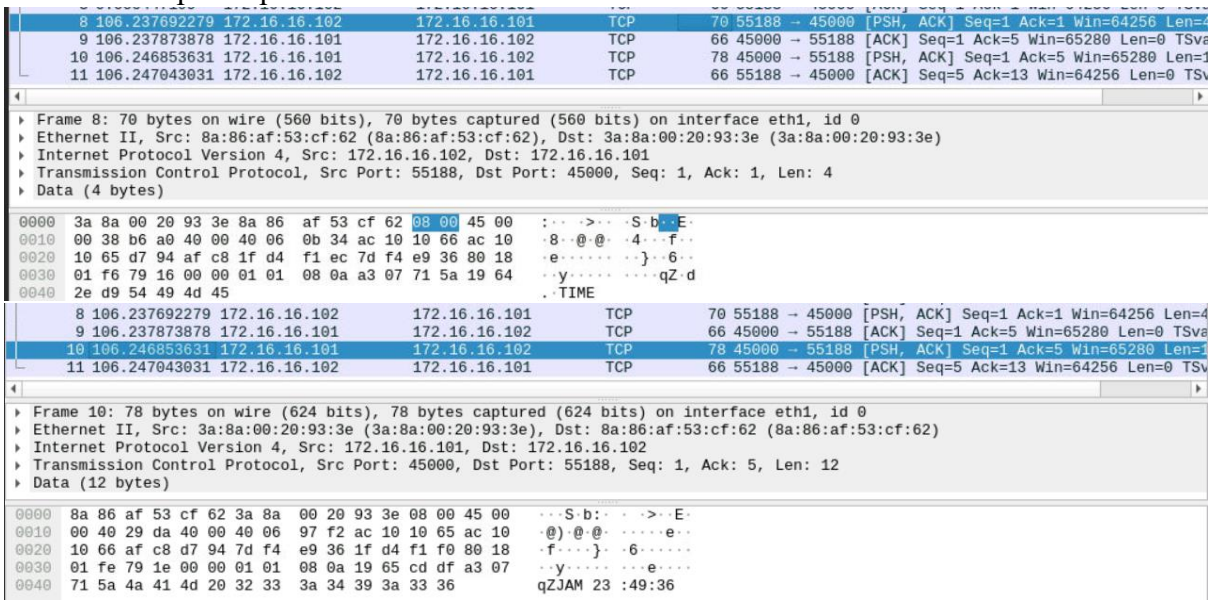
```
(base) jovyan@b01fac93c31c:~/work/progjar/tugas-2$ python3 server.py
connection from ('172.16.16.102', 56450)
connection from ('172.16.16.103', 37522)
Request from client ('172.16.16.102', 56450): TIME
Sent to ('172.16.16.102', 56450): JAM 22:51:35
Request from client ('172.16.16.103', 37522): TIME
Sent to ('172.16.16.103', 37522): JAM 22:51:36
Request from client ('172.16.16.102', 56450): TIME
Sent to ('172.16.16.102', 56450): JAM 22:51:38
Request from client ('172.16.16.103', 37522): TIME
Sent to ('172.16.16.103', 37522): JAM 22:51:40
Request from client ('172.16.16.102', 56450): TIME
Sent to ('172.16.16.102', 56450): JAM 22:51:41
Request from client ('172.16.16.103', 37522): TIME
Sent to ('172.16.16.103', 37522): JAM 22:51:43
```

2. Analisis Wireshark

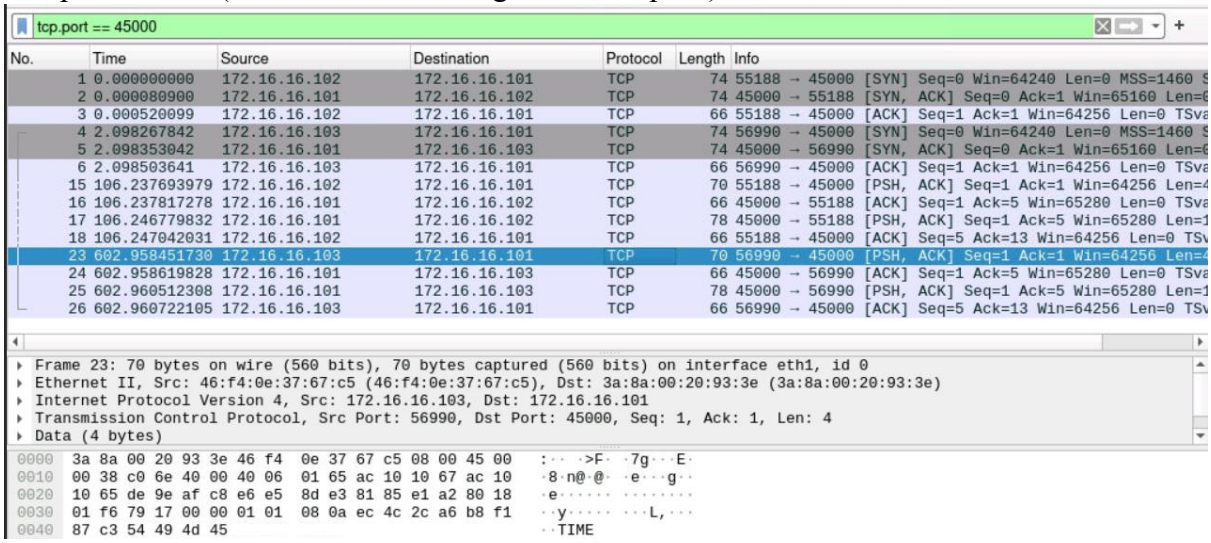
Pada saat client pertama kali mengirim permintaan koneksi, client dan server melakukan 3-way handshake, hal ini dapat dilihat pada wireshark pada urutan flag [SYN → SYN, ACK → ACK]. Client di mesin-2 (172.16.16.102) dengan port acak 55188 dan client di mesin-3 (172.16.16.103) dengan port acak 56990 mengirim request koneksi ke server (172.16.16.101) dengan port 45000. Berikut adalah tampilan request handshake yang diterima oleh server di wireshark



Saat client mengirim request TIME, packet request tersebut dikirim menggunakan flag [PSH, ACK] yang menunjukkan bahwa data langsung dikirim ke server. Kemudian server mengonfirmasi bahwa data telah diterima menggunakan flag [ACK], dan mengirim response ke client menggunakan flag [PSH, ACK] yang berisi data waktu yang telah diformat. Pada mesin-2, client mengirim request TIME pada no 8, dan menerima response pada no 10.

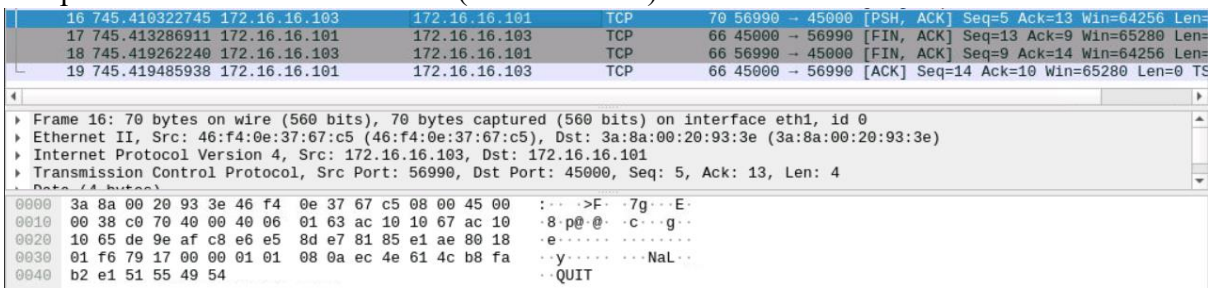


Tampilan server (setelah mesin-3 mengirimkan request):



Kemudian, saat client mengirimkan request QUIT, maka server akan mengirim packet dengan flag [FIN, ACK], yaitu flag untuk mengakhiri koneksi, kemudian dibalas oleh client dengan flag yang sama. Misal client di mesin-3 mengirim request QUIT:

Tampilan wireshark client mesin-3 (lihat no 16-19):



Tampilan wireshark server mesin-1 (lihat no 31-34):