

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:

```
8 class ProcessTheClient(threading.Thread):
9     def __init__(self, connection, address):
10         self.connection = connection
11         self.address = address
12         threading.Thread.__init__(self)
13
14     def run(self):
15         while True:
16             data = self.connection.recv(32)
17             if data:
18                 # get request
19                 request = data.decode()
20                 logging.info(f"Request from client {self.address}: {request}")
21
22                 if request == "TIME":
23                     current_time = time.strftime("%H:%M:%S")
24                     response = f"JAM {current_time}"
25                     self.connection.sendall(response.encode())
26                     logging.info(f"Sent to {self.address}: {response.strip()}")
27                 elif request == "QUIT":
28                     self.connection.close()
29                     break
30                 else:
31                     self.connection.sendall(b"Invalid request")
32                 else:
33                     break
34             self.connection.close()
```

Client:

```
1 import socket
2 import logging
3
4 def main():
5     server_address = ('172.16.16.101', 45000)
6     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7     sock.connect(server_address)
8
9     try:
10         while True:
11             user_input = input("TIME/QUIT: ").strip().upper()
12
13             sock.sendall(user_input.encode())
14
15             if user_input == "QUIT":
16                 logging.info("Connection closed")
17                 break
18
19             # accept response from server
20             response = sock.recv(1024).decode()
21             print(response)
22
23         finally:
24             sock.close()
25
26 if __name__ == "__main__":
27     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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.16.102	172.16.16.101	TCP	74	55188 → 45000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
2	0.000000000	172.16.16.101	172.16.16.102	TCP	74	45000 → 55188 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
3	0.000520009	172.16.16.102	172.16.16.101	TCP	66	55188 → 45000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSv
4	2.098267842	172.16.16.103	172.16.16.101	TCP	74	56990 → 45000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
5	2.098353042	172.16.16.101	172.16.16.103	TCP	74	45000 → 56990 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
6	2.098503641	172.16.16.103	172.16.16.101	TCP	66	56990 → 45000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSv

  

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth1, id 0 Ethernet II, Src: 8a:86:af:53:cf:62 (8a:86:af:53:cf:62), Dst: 3a:8a:00:20:93:3e (3a:8a:00:20:93:3e) Internet Protocol Version 4, Src: 172.16.16.102, Dst: 172.16.16.101 Transmission Control Protocol, Src Port: 55188, Dst Port: 45000, Seq: 0, Len: 0						
---	--	--	--	--	--	--

  

0000	3a 8a 00 20 93 3e 8a 86	af 53 cf 62 08 00 45 00	...	...	S-b-E
0010	00 3c b6 9e 40 00 40 06	0b 32 ac 10 10 66 ac 10	...	...	@-2-f
0020	10 65 d7 94 af c8 1f d4	f1 eb 00 00 00 00 a0 02	...	...	e-6
0030	fa f0 79 1a 00 00 02 04	05 b4 04 02 08 0a a3 05	...	...	y-...
0040	d2 5d 00 00 00 00 01 03	03 07	...	...	]...

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.

8	106.237692279	172.16.16.102	172.16.16.101	TCP	70	55188 → 45000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=4
9	106.237873878	172.16.16.101	172.16.16.102	TCP	66	45000 → 55188 [ACK] Seq=1 Ack=5 Win=65280 Len=0 TSv
10	106.246853631	172.16.16.101	172.16.16.102	TCP	78	45000 → 55188 [PSH, ACK] Seq=1 Ack=5 Win=65280 Len=1
11	106.247043031	172.16.16.102	172.16.16.101	TCP	66	55188 → 45000 [ACK] Seq=5 Ack=13 Win=64256 Len=0 TSv

  

Frame 8: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface eth1, id 0 Ethernet II, Src: 8a:86:af:53:cf:62 (8a:86:af:53:cf:62), Dst: 3a:8a:00:20:93:3e (3a:8a:00:20:93:3e) Internet Protocol Version 4, Src: 172.16.16.102, Dst: 172.16.16.101 Transmission Control Protocol, Src Port: 55188, Dst Port: 45000, Seq: 1, Ack: 1, Len: 4 Data (4 bytes)						
---	--	--	--	--	--	--

  

0000	3a 8a 00 20 93 3e 8a 86	af 53 cf 62 08 00 45 00	...	...	S-b-E
0010	00 38 b6 a0 40 00 40 06	0b 34 ac 10 10 66 ac 10	...	...	8-@-4-f
0020	10 65 d7 94 af c8 1f d4	f1 ec 7d f4 e9 36 80 18	...	...	e-6
0030	01 f6 79 16 00 00 01 01	08 0a a3 07 71 5a 19 64	...	...	y-...qZ-d
0040	2e d9 54 49 4d 45		...	...	..TIME

  

8	106.237692279	172.16.16.102	172.16.16.101	TCP	70	55188 → 45000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=4
9	106.237873878	172.16.16.101	172.16.16.102	TCP	66	45000 → 55188 [ACK] Seq=1 Ack=5 Win=65280 Len=0 TSv
10	106.246853631	172.16.16.101	172.16.16.102	TCP	78	45000 → 55188 [PSH, ACK] Seq=1 Ack=5 Win=65280 Len=1
11	106.247043031	172.16.16.102	172.16.16.101	TCP	66	55188 → 45000 [ACK] Seq=5 Ack=13 Win=64256 Len=0 TSv

  

Frame 10: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface eth1, id 0 Ethernet II, Src: 3a:8a:00:20:93:3e (3a:8a:00:20:93:3e), Dst: 8a:86:af:53:cf:62 (8a:86:af:53:cf:62) Internet Protocol Version 4, Src: 172.16.16.101, Dst: 172.16.16.102 Transmission Control Protocol, Src Port: 45000, Dst Port: 55188, Seq: 1, Ack: 5, Len: 12 Data (12 bytes)						
--	--	--	--	--	--	--

  

0000	8a 86 af 53 cf 62 3a 8a	00 20 93 3e 08 00 45 00	...	...	S-b-E
0010	00 40 29 da 40 00 40 06	97 f2 ac 10 10 65 ac 10	...	...	@-@-e
0020	10 66 af c8 d7 94 7d f4	e9 36 1f d4 f1 f0 80 18	...	...	f-}-6
0030	01 fe 79 1e 00 00 01 01	08 0a 19 65 cd df a3 07	...	...	y-...e
0040	71 5a 4a 41 4d 20 32 33	3a 34 39 3a 33 36	...	...	qZJAM 23 :49:36



Tampilan server (setelah mesin-3 mengirimkan request):

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.16.102	172.16.16.101	TCP	74	55188 → 45000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
2	0.000000000	172.16.16.101	172.16.16.102	TCP	74	45000 → 55188 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
3	0.000520099	172.16.16.102	172.16.16.101	TCP	66	55188 → 45000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSv
4	2.098267842	172.16.16.103	172.16.16.101	TCP	74	56990 → 45000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
5	2.098353042	172.16.16.101	172.16.16.103	TCP	74	45000 → 56990 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
6	2.098503641	172.16.16.103	172.16.16.101	TCP	66	56990 → 45000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSv
15	106.237693979	172.16.16.102	172.16.16.101	TCP	70	55188 → 45000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=4
16	106.237817278	172.16.16.101	172.16.16.102	TCP	66	45000 → 55188 [ACK] Seq=1 Ack=5 Win=65280 Len=0 TSv
17	106.246779832	172.16.16.101	172.16.16.102	TCP	78	45000 → 55188 [PSH, ACK] Seq=1 Ack=5 Win=65280 Len=1
18	106.247042031	172.16.16.102	172.16.16.101	TCP	66	55188 → 45000 [ACK] Seq=5 Ack=13 Win=64256 Len=0 TSv
23	602.958451730	172.16.16.103	172.16.16.101	TCP	70	56990 → 45000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=4
24	602.958619828	172.16.16.101	172.16.16.103	TCP	66	45000 → 56990 [ACK] Seq=1 Ack=5 Win=65280 Len=0 TSv
25	602.960512308	172.16.16.101	172.16.16.103	TCP	78	45000 → 56990 [PSH, ACK] Seq=1 Ack=5 Win=65280 Len=1
26	602.960722105	172.16.16.103	172.16.16.101	TCP	66	56990 → 45000 [ACK] Seq=5 Ack=13 Win=64256 Len=0 TSv

Frame 23: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface eth1, id 0  
 Ethernet II, Src: 46:f4:0e:37:67:c5 (46:f4:0e:37:67:c5), Dst: 3a:8a:00:20:93:3e (3a:8a:00:20:93:3e)  
 Internet Protocol Version 4, Src: 172.16.16.103, Dst: 172.16.16.101  
 Transmission Control Protocol, Src Port: 56990, Dst Port: 45000, Seq: 1, Ack: 1, Len: 4  
 Data (4 bytes)

```

0000  3a 8a 00 20 93 3e 46 f4 0e 37 67 c5 08 00 45 00  :...>F..7g...E.
0010  00 38 c0 6e 40 00 40 06 01 65 ac 10 10 67 ac 10  :8.n@...c...g..
0020  10 65 de 9e af c8 e6 e5 8d e3 81 85 e1 a2 00 18  :e.....
0030  01 f6 79 17 00 00 01 01 08 0a ec 4c 2c a6 b8 f1  :.y.....L...
0040  87 c3 54 49 4d 45                                :..TIME
  
```

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):

16	745.410322745	172.16.16.103	172.16.16.101	TCP	70	56990 → 45000 [PSH, ACK] Seq=5 Ack=13 Win=64256 Len=4
17	745.413286911	172.16.16.101	172.16.16.103	TCP	66	45000 → 56990 [FIN, ACK] Seq=13 Ack=9 Win=65280 Len=0 TSv
18	745.419262240	172.16.16.103	172.16.16.101	TCP	66	56990 → 45000 [FIN, ACK] Seq=9 Ack=14 Win=64256 Len=0 TSv
19	745.419485938	172.16.16.101	172.16.16.103	TCP	66	45000 → 56990 [ACK] Seq=14 Ack=10 Win=65280 Len=0 TSv

Frame 16: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface eth1, id 0  
 Ethernet II, Src: 46:f4:0e:37:67:c5 (46:f4:0e:37:67:c5), Dst: 3a:8a:00:20:93:3e (3a:8a:00:20:93:3e)  
 Internet Protocol Version 4, Src: 172.16.16.103, Dst: 172.16.16.101  
 Transmission Control Protocol, Src Port: 56990, Dst Port: 45000, Seq: 5, Ack: 13, Len: 4  
 Data (4 bytes)

```

0000  3a 8a 00 20 93 3e 46 f4 0e 37 67 c5 08 00 45 00  :...>F..7g...E.
0010  00 38 c0 70 40 00 40 06 01 63 ac 10 10 67 ac 10  :8.p@...c...g..
0020  10 65 de 9e af c8 e6 e5 8d e7 81 85 e1 ae 80 18  :e.....
0030  01 f6 79 17 00 00 01 01 08 0a ec 4e 61 4c b8 fa  :.y.....NaL...
0040  b2 e1 51 55 49 54                                :..QUIT
  
```

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

No.	Time	Source	Destination	Protocol	Length	Info
5	2.098353042	172.16.16.101	172.16.16.103	TCP	74	45000 → 56990 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
6	2.098503641	172.16.16.103	172.16.16.101	TCP	66	56990 → 45000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSv
15	106.237693979	172.16.16.102	172.16.16.101	TCP	70	55188 → 45000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=4
16	106.237817278	172.16.16.101	172.16.16.102	TCP	66	45000 → 55188 [ACK] Seq=1 Ack=5 Win=65280 Len=0 TSv
17	106.246779832	172.16.16.101	172.16.16.102	TCP	78	45000 → 55188 [PSH, ACK] Seq=1 Ack=5 Win=65280 Len=1
18	106.247042031	172.16.16.102	172.16.16.101	TCP	66	55188 → 45000 [ACK] Seq=5 Ack=13 Win=64256 Len=0 TSv
23	602.958451730	172.16.16.103	172.16.16.101	TCP	70	56990 → 45000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=4
24	602.958619828	172.16.16.101	172.16.16.103	TCP	66	45000 → 56990 [ACK] Seq=1 Ack=5 Win=65280 Len=0 TSv
25	602.960512308	172.16.16.101	172.16.16.103	TCP	78	45000 → 56990 [PSH, ACK] Seq=1 Ack=5 Win=65280 Len=1
26	602.960722105	172.16.16.103	172.16.16.101	TCP	66	56990 → 45000 [ACK] Seq=5 Ack=13 Win=64256 Len=0 TSv
31	747.508428787	172.16.16.103	172.16.16.101	TCP	70	56990 → 45000 [PSH, ACK] Seq=5 Ack=13 Win=64256 Len=4
32	747.511254954	172.16.16.101	172.16.16.103	TCP	66	45000 → 56990 [FIN, ACK] Seq=13 Ack=9 Win=65280 Len=0 TSv
33	747.517335683	172.16.16.103	172.16.16.101	TCP	66	56990 → 45000 [FIN, ACK] Seq=9 Ack=14 Win=64256 Len=0 TSv
34	747.517431381	172.16.16.101	172.16.16.103	TCP	66	45000 → 56990 [ACK] Seq=14 Ack=10 Win=65280 Len=0 TSv

Frame 31: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface eth1, id 0  
 Ethernet II, Src: 46:f4:0e:37:67:c5 (46:f4:0e:37:67:c5), Dst: 3a:8a:00:20:93:3e (3a:8a:00:20:93:3e)  
 Internet Protocol Version 4, Src: 172.16.16.103, Dst: 172.16.16.101  
 Transmission Control Protocol, Src Port: 56990, Dst Port: 45000, Seq: 5, Ack: 13, Len: 4  
 Data (4 bytes)

```

0000  3a 8a 00 20 93 3e 46 f4 0e 37 67 c5 08 00 45 00  :...>F..7g...E.
0010  00 38 c0 70 40 00 40 06 01 63 ac 10 10 67 ac 10  :8.p@...c...g..
0020  10 65 de 9e af c8 e6 e5 8d e7 81 85 e1 ae 80 18  :e.....
0030  01 f6 79 17 00 00 01 01 08 0a ec 4e 61 4c b8 fa  :.y.....NaL...
0040  b2 e1 51 55 49 54                                :..QUIT
  
```