

1. Analisis Wireshark Mesin 1 dan Mesin 2 ketika menjalankan program "socket\_info.py"
  - a. Screenshot terminal dan Wireshark Mesin 1

The image shows a terminal window and a Wireshark network traffic capture. The terminal window displays the execution of a Python script named `socket_info.py`. The output shows a timeout of 10.0 seconds and a list of network addresses and socket kinds.

```
(base) jovyan@63fb63c207d5: ~/work$ python3 socket_info.py
timeout : None
timeout : 10.0
[(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>, 6, '', ('103.94.189.4', 80))]
(base) jovyan@63fb63c207d5: ~/work$
```

The Wireshark window shows a capture on interface `any` with a filter `ip.addr == 172.18.0.3 && dns`. The packet list shows four DNS packets. The packet details pane shows the structure of a DNS query for `www.its.ac.id`.

| No.  | Time         | Source       | Destination  | Protocol | Length | Info   |
|------|--------------|--------------|--------------|----------|--------|--|
| 3317 | 86.442450566 | 172.18.0.3   | 192.168.65.7 | DNS      | 75     | Standard query 0x64d1 AAAA www.its.ac.id             |
| 3318 | 86.442450567 | 172.18.0.3   | 192.168.65.7 | DNS      | 75     | Standard query 0x0bde A www.its.ac.id                |
| 3319 | 86.444288063 | 192.168.65.7 | 172.18.0.3   | DNS      | 104    | Standard query response 0x0bde A www.its.ac.id A 103 |
| 3321 | 86.444587508 | 192.168.65.7 | 172.18.0.3   | DNS      | 75     | Standard query response 0x64d1 AAAA www.its.ac.id    |

The packet details pane for the selected packet (Frame 3317) shows the following structure:

- Frame 3317: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface any, id 0
- Linux cooked capture v1
- Internet Protocol Version 4, Src: 172.18.0.3, Dst: 192.168.65.7
- User Datagram Protocol, Src Port: 37511, Dst Port: 53
- Domain Name System (query)

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000  00 04 00 01 00 06 02 42 ac 12 00 03 00 00 08 00 .....B.....
0010  45 00 00 3b c4 15 40 00 40 11 c8 d7 ac 12 00 03 E-;...@_@.....
0020  c0 a8 41 07 92 87 00 35 00 27 ad fd 64 d1 01 00 ..A...5...d...
0030  00 01 00 00 00 00 00 00 03 77 77 77 03 69 74 73 .....www.its
0040  02 61 63 02 69 64 00 00 1c 00 01 .....ac.id.....
```

- b. Screenshot Wireshark Mesin 2

The image shows a terminal window with the same Python script `socket_info.py` being executed. The output is identical to the first screenshot.

```
(base) jovyan@3ef5f6d8c4d6: ~/work$ python3 socket_info.py
timeout : None
timeout : 10.0
[(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>, 6, '', ('103.94.189.4', 80))]
(base) jovyan@3ef5f6d8c4d6: ~/work$
```

The image shows a Wireshark network traffic capture. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for packet capture and analysis. A filter bar at the top shows the filter 'ip.addr == 172.18.0.4 && dns'. The main packet list displays several DNS-related packets:

| No.   | Time         | Source       | Destination  | Protocol | Length | Info   |
|-------|--------------|--------------|--------------|----------|--------|--|
| 13590 | 56.053175179 | 172.18.0.4   | 192.168.65.7 | DNS      | 75     | Standard query 0xff75 AAAA www.its.ac.id             |
| 13591 | 56.053461164 | 172.18.0.4   | 192.168.65.7 | DNS      | 75     | Standard query 0x5c7b A www.its.ac.id                |
| 13592 | 56.055206861 | 192.168.65.7 | 172.18.0.4   | DNS      | 104    | Standard query response 0x5c7b A www.its.ac.id A 103 |
| 13594 | 56.095164164 | 192.168.65.7 | 172.18.0.4   | DNS      | 75     | Standard query response 0xff75 AAAA www.its.ac.id    |
| 18334 | 77.840861106 | 172.18.0.4   | 192.168.65.7 | DNS      | 75     | Standard query 0xfc4a AAAA www.its.ac.id             |
| 18335 | 77.840859271 | 172.18.0.4   | 192.168.65.7 | DNS      | 75     | Standard query 0x2a44 A www.its.ac.id                |
| 18336 | 77.860648896 | 192.168.65.7 | 172.18.0.4   | DNS      | 104    | Standard query response 0x2a44 A www.its.ac.id A 103 |
| 18338 | 77.908267356 | 192.168.65.7 | 172.18.0.4   | DNS      | 75     | Standard query response 0xfc4a AAAA www.its.ac.id    |

Below the packet list, the details pane for packet 13590 is expanded, showing the following information:

- Frame 13590: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface any, id 0
- Linux cooked capture v1
- Internet Protocol Version 4, Src: 172.18.0.4, Dst: 192.168.65.7
- User Datagram Protocol, Src Port: 34333, Dst Port: 53
- Domain Name System (query)

The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII:

```
0000 00 04 00 01 00 06 02 42 ac 12 00 04 44 99 08 00 .....B....D...
0010 45 00 00 3b e2 70 40 00 40 11 aa 7b ac 12 00 04 E...;p@.@...{...
0020 c0 a8 41 07 86 1d 00 35 00 27 ad fe ff 75 01 00 ..A...5.....u...
0030 00 01 00 00 00 00 00 00 03 77 77 77 03 69 74 73 .....www.its
0040 02 61 63 02 69 64 00 00 1c 00 01 ..ac.id.....
```

The status bar at the bottom indicates: Domain Name System: Protocol, Packets: 75441 · Displayed: 8 (0.0%), Profile: Default.

Pada Docker file terdapat mesin 1, mesin 2, mesin 3 dan mesin 4, namun pada soal ini akan berfokus pada mesin 1 dan mesin 2, pada mesin 1 dan mesin 2. Disini karena saya connect nya menggunakan laptop saya, maka akan menggunakan localhost untuk mengakses mesin nya.

- Mesin 1 = <http://localhost:60001>
- Mesin 2 = <http://localhost:60002>

Dimana kita akan menjalankan file "*socket\_info*" di tiap mesin, dan akan menghasilkan output:

```
timeout : None
timeout : 10.0
[(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>, 6, '', ('103.94.189.4', 80))]
```

Bisa dilihat bahwa awalnya mesin tidak memiliki timeout, namun kemudian terjadi timeout sepuluh detik. Dimana socket tersebut memiliki kriteria:

- IPv4
- TCP
- IP destination = 103.94.189.4
- Port nya = 80

Kemudian kita dapat menjalankan wireshark untuk mesin 1 dan mesin 2, berikut cara mengaksesnya:

- Wireshark(VNC) Mesin 1 = <http://localhost:50001>
- Wireshark(VNC) Mesin 2 = <http://localhost:50002>

#### c. Penjelasan Mesin 1.

Ip address Mesin 1 = 172.18.0.3

Ip address Mesin 2 = 172.18.0.4

No. 3317: Mesin 1 mengirimkan permintaan ke server DNS untuk mencari tahu alamat IP (IPv4) dari situs [www.its.ac.id](http://www.its.ac.id).

No. 3318: Server DNS menjawab permintaan tadi dan memberitahu bahwa alamat IP dari [www.its.ac.id](http://www.its.ac.id) adalah 103.94.189.5.

No. 3319: Setelah mendapatkan alamat IPv4, Mesin 1 juga mencoba mencari tahu apakah situs [www.its.ac.id](http://www.its.ac.id) punya alamat IPv6 (AAAA record).

No. 3321: Server DNS menjawab bahwa tidak ada alamat IPv6 untuk [www.its.ac.id](http://www.its.ac.id).

d. Penjelasan Mesin 2

No. 13590: Mesin 2 mengirim permintaan ke server DNS untuk mencari tahu **alamat IP (IPv4)** dari situs [www.its.ac.id](http://www.its.ac.id).

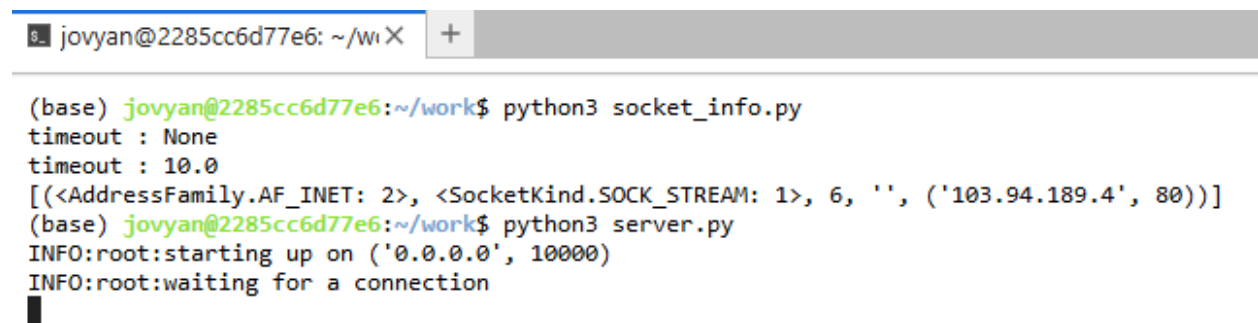
No. 13591: Server DNS membalas bahwa alamat IP dari situs [www.its.ac.id](http://www.its.ac.id) adalah 103.94.189.5.

No. 13592: Setelah itu, Mesin 2 juga mencoba mencari alamat IPv6 dari situs yang sama.

No. 13594: Server DNS menjawab bahwa alamat IPv6 tidak tersedia untuk [www.its.ac.id](http://www.its.ac.id).

2. Jalankan server.py pada mesin 1, dan client.py pada mesin 2. Dimana kita sesuaikan ip address servernya di client.py

a. Screenshot Mesin 1



```
jovyan@2285cc6d77e6: ~/work$ python3 socket_info.py
timeout : None
timeout : 10.0
[(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>, 6, '', ('103.94.189.4', 80))]
(base) jovyan@2285cc6d77e6: ~/work$ python3 server.py
INFO:root:starting up on ('0.0.0.0', 10000)
INFO:root:waiting for a connection
```

ip.addr == 172.18.0.2 && ip.addr == 172.18.0.3

| No.  | Time          | Source     | Destination | Protocol | Length | Info   |
|------|---------------|------------|-------------|----------|--------|--|
| 8573 | 104.272195134 | 172.18.0.3 | 172.18.0.2  | TCP      | 76     | 43396 → 10000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460   |
| 8574 | 104.272207136 | 172.18.0.2 | 172.18.0.3  | TCP      | 76     | 10000 → 43396 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 |
| 8575 | 104.272234848 | 172.18.0.3 | 172.18.0.2  | TCP      | 68     | 43396 → 10000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TS   |
| 8576 | 104.272344537 | 172.18.0.3 | 172.18.0.2  | TCP      | 114    | 43396 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=  |
| 8577 | 104.272347582 | 172.18.0.2 | 172.18.0.3  | TCP      | 68     | 10000 → 43396 [ACK] Seq=1 Ack=47 Win=65152 Len=0 T   |
| 8578 | 104.272770652 | 172.18.0.2 | 172.18.0.3  | TCP      | 100    | 10000 → 43396 [PSH, ACK] Seq=1 Ack=47 Win=65152 Le   |
| 8579 | 104.272801076 | 172.18.0.3 | 172.18.0.2  | TCP      | 68     | 43396 → 10000 [ACK] Seq=47 Ack=33 Win=64256 Len=0    |
| 8580 | 104.272870436 | 172.18.0.2 | 172.18.0.3  | TCP      | 82     | 10000 → 43396 [PSH, ACK] Seq=33 Ack=47 Win=65152 L   |
| 8581 | 104.272875490 | 172.18.0.3 | 172.18.0.2  | TCP      | 68     | 43396 → 10000 [ACK] Seq=47 Ack=47 Win=64256 Len=0    |
| 8582 | 104.272922610 | 172.18.0.3 | 172.18.0.2  | TCP      | 68     | 43396 → 10000 [FIN, ACK] Seq=47 Ack=47 Win=64256 L   |
| 8584 | 104.273210825 | 172.18.0.2 | 172.18.0.3  | TCP      | 68     | 10000 → 43396 [FIN, ACK] Seq=47 Ack=48 Win=65152 L   |

Frame 8581: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0  
Linux cooked capture v1  
Internet Protocol Version 4, Src: 172.18.0.3, Dst: 172.18.0.2  
Transmission Control Protocol, Src Port: 43396, Dst Port: 10000, Seq: 47, Ack: 47, Len: 0

```
0000  00 00 00 01 00 06 02 42 ac 12 00 03 83 a3 08 00  ....B.....
0010  45 00 00 34 ff fb 40 00 40 06 e2 9e ac 12 00 03  E..4...@.@...
0020  ac 12 00 02 a9 84 27 10 25 8b b2 f4 6e bd 6a c6  ....'..%...n.j
0030  80 10 01 f6 58 50 00 00 01 01 08 0a 73 2e cb cb  ....XP.....s...
0040  70 73 4c 42                                     psLB
```

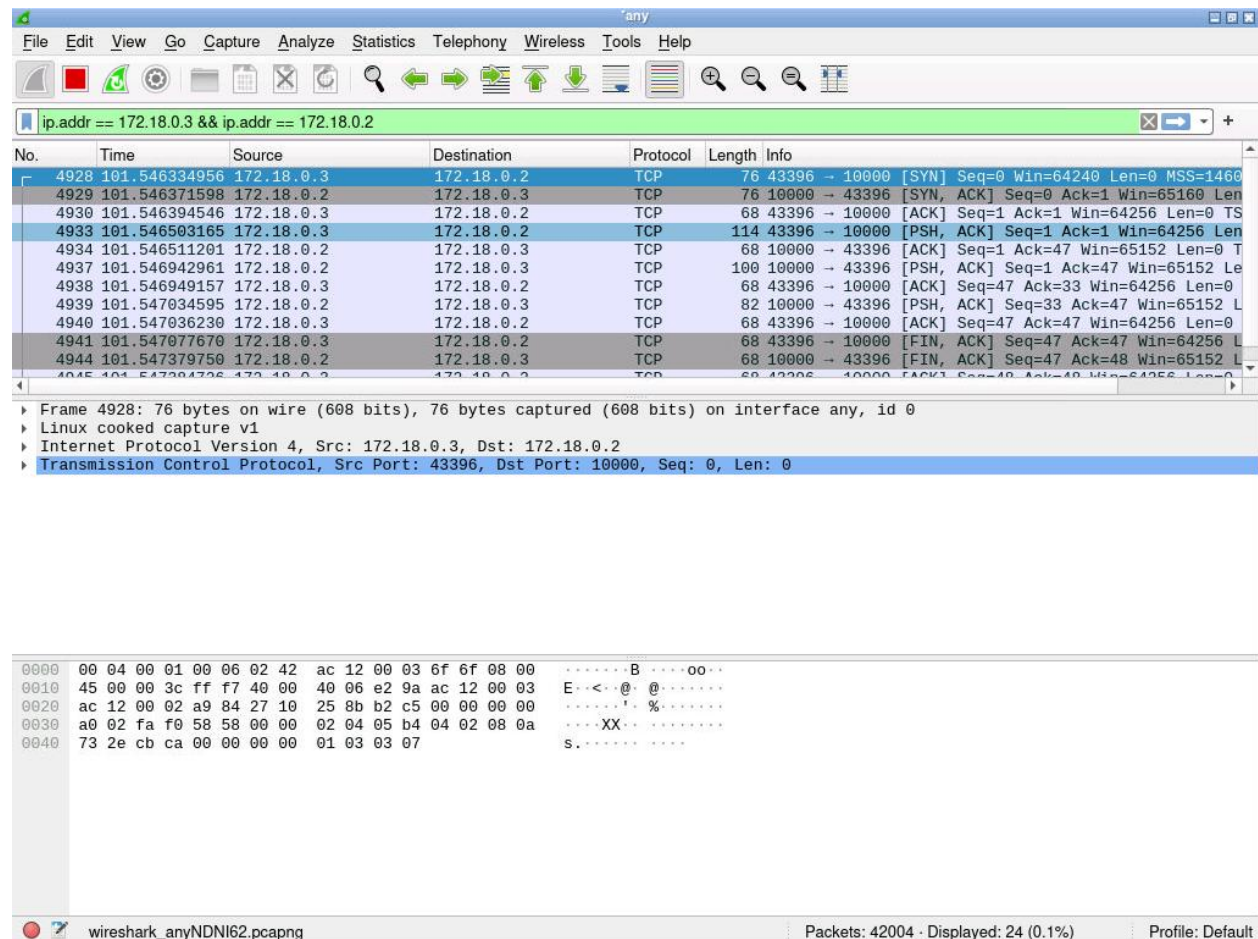
wireshark\_any7KFO62.pcapng Packets: 43307 · Displayed: 24 (0.1%) Profile: Default

```
jovyan@2285cc6d77e6: ~/work$ python3 socket_info.py
timeout : None
timeout : 10.0
[(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>, 6, '', ('103.94.189.4', 80))]
(base) jovyan@2285cc6d77e6: ~/work$ python3 server.py
INFO:root:starting up on ('0.0.0.0', 10000)
INFO:root:waiting for a connection
INFO:root:connection from ('172.18.0.3', 41482)
INFO:root:received b'INI ADALAH DATA YANG DIKIRIM ABC'
INFO:root:sending back data
INFO:root:received b'DEFGHIJKLMNOPQ'
INFO:root:sending back data
INFO:root:received b''
INFO:root:waiting for a connection
```

## b. Screenshot Mesin 2



```
jovyan@ee91a07ecadb: ~/work$ python3 socket_info.py
timeout : None
timeout : 10.0
[(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>, 6, '', ('103.94.189.4', 80))]
(jovyan@ee91a07ecadb:~/work$ python3 client.py
INFO:root:connecting to ('172.18.0.2', 10000)
INFO:root:sending INI ADALAH DATA YANG DIKIRIM ABCDEFGHIJKLMNOPQ
INFO:root:b'INI ADALAH DATA '
INFO:root:b'YANG DIKIRIM ABC'
INFO:root:b'DEFGHIJKLMNOPQ'
INFO:root:closing
(base) jovyan@ee91a07ecadb:~/work$
```



Frame 4928: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0

Linux cooked capture v1

Internet Protocol Version 4, Src: 172.18.0.3, Dst: 172.18.0.2

Transmission Control Protocol, Src Port: 43396, Dst Port: 10000, Seq: 0, Len: 0

0000 00 04 00 01 00 06 02 42 ac 12 00 03 6f 6f 08 00 .....B.....oo..

0010 45 00 00 3c ff f7 40 00 40 06 e2 9a ac 12 00 03 E...<...@...@...

0020 ac 12 00 02 a9 84 27 10 25 8b b2 c5 00 00 00 00 .....'.%.....

0030 a0 02 fa f0 58 58 00 00 02 04 05 b4 04 02 08 0a ...XX.....

0040 73 2e cb ca 00 00 00 00 01 03 03 07 s.....

wireshark\_anyNDNI62.pcapng Packets: 42004 · Displayed: 24 (0.1%) Profile: Default

### c. Penjelasan

Dikarenakan saya menghapus container dan mengulang mesin nya, maka ip saya berubah:

Dari:

Ip address Mesin 1 = 172.18.0.3

Ip address Mesin 2 = 172.18.0.4

Menjadi:

Ip address Mesin 1 = 172.18.0.2

Ip address Mesin 2 = 172.18.0.3

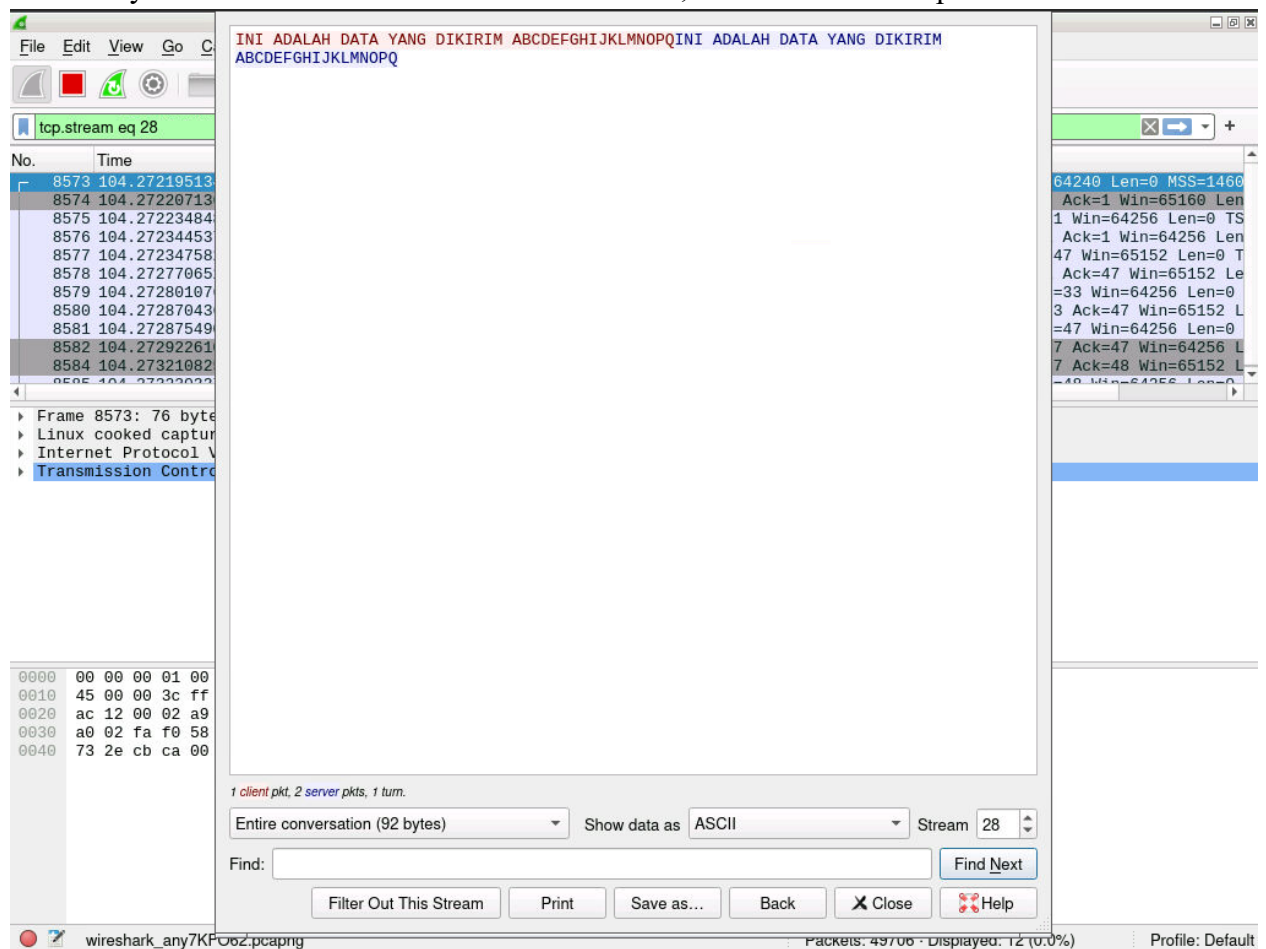
Pada percobaan kali ini, kita akan membuat mesin 1 menjadi sebuah server, dan mesin 2 sebagai client nya. Dimana client akan mencoba menghubungkan ke server dengan three way handshake, kemudian akan mengirimkan sebuah pesan, dan ketika selesai mengirim semua pesan, maka client akan langsung terputus dari koneksi tersebut, namun dapat dilihat bahwa, pada terminal server.py, server tidak bisa menerima semua pesan sekaligus, namun server memecah pesan dengan bantuan tcp layer, sehingga pesan akan dibaca secara bertahap. Dapat dilihat bahwa pesan yang diterima oleh server, tidak sesuai dengan pesan yang dikirim dari client ( dilihat dari terminal )

| No.  | Penjelasan  |
|------|---|
| 8573 | SYN dikirim dari 172.18.0.3 ke 172.18.0.2 untuk memulai koneksi. Ini adalah langkah pertama dari three-way handshake. |
| 8574 | SYN, ACK diterima dari 172.18.0.2 sebagai balasan. Langkah kedua handshake.   |
| 8575 | ACK dari 172.18.0.3 untuk menyelesaikan handshake. Sekarang koneksi telah dibuka.                                     |
| 8576 | Paket PSH, ACK dikirim dari 172.18.0.3 berisi data. PSH menunjukkan data segera dikirim ke aplikasi penerima.         |
| 8577 | ACK dari 172.18.0.2 sebagai tanda penerimaan data sebelumnya.   |
| 8578 | Kiriman data lanjutan (PSH, ACK) dari 172.18.0.3.   |
| 8579 | ACK dari 172.18.0.2 sebagai tanggapan dari data sebelumnya.   |
| 8580 | FIN, ACK dari 172.18.0.3. Ini menandai permintaan penutupan koneksi dari sisi pengirim.                               |
| 8581 | ACK dari 172.18.0.2. Konfirmasi bahwa permintaan FIN diterima.  |
| 8582 | FIN, ACK dari 172.18.0.2 untuk mengakhiri koneksi dari sisi penerima.   |
| 8583 | ACK terakhir dari 172.18.0.3 sebagai respon terhadap FIN dari 172.18.0.2. Koneksi TCP resmi ditutup.                  |

| No.  | Penjelasan   |
|------|--|
| 4928 | SYN dikirim dari 172.18.0.3 ke 172.18.0.2 (ditampilkan dari perspektif Wireshark Mesin 2). |
| 4929 | SYN, ACK diterima dari 172.18.0.2.   |
| 4930 | ACK dikirim kembali dari 172.18.0.3. Handshake selesai.                                    |
| 4933 | Paket PSH, ACK berisi data dari 172.18.0.3.  |
| 4934 | ACK sebagai respon terhadap data dari 172.18.0.2.  |
| 4937 | Data tambahan dikirim lagi dari 172.18.0.3.  |
| 4938 | ACK diterima dari 172.18.0.2 sebagai respons.  |
| 4939 | FIN, ACK dari 172.18.0.3 menandakan permintaan mengakhiri koneksi.                         |
| 4940 | ACK dari 172.18.0.2 sebagai balasan untuk FIN.   |
| 4941 | FIN, ACK dari 172.18.0.2, permintaan balasan untuk menutup sesi.                           |
| 4944 | ACK terakhir dari 172.18.0.3. Proses 4-way termination TCP selesai.                        |



Ketika saya follow stream dari kedua mesin tersebut, maka akan menampilkan:



3. Jalankan server di port 3244, lalu kirimkan file dari client menuju server.

a. Screenshot Mesin 1

```
(base) jovyan@2285cc6d77e6:~/work$ python3 server_2.py
INFO:root:starting up on ('0.0.0.0', 32444)
INFO:root:waiting for a connection
```

```
(base) jovyan@2285cc6d77e6:~/work$ python3 server_2.py
INFO:root:starting up on ('0.0.0.0', 32444)
INFO:root:waiting for a connection
INFO:root:connection from ('172.18.0.3', 44204)
INFO:root:received b'INI ADALAH DATA YANG DIKIRIM ABC'
INFO:root:sending back data
INFO:root:received b'DEFGHIJKLMNOPQ'
INFO:root:sending back data
INFO:root:received b'Halo server mesin 1, ini Client'
INFO:root:sending back data
INFO:root:ERROR: [Errno 104] Connection reset by peer
INFO:root:closing
(base) jovyan@2285cc6d77e6:~/work$
```

**b. Screenshot Mesin 2**

```
(base) jovyan@ee91a07ecadb:~/work$ python3 client_2.py
INFO:root:connecting to ('172.18.0.2', 32444)
INFO:root:sending INI ADALAH DATA YANG DIKIRIM ABCDEFGHIJKLMNOPQ
INFO:root:b'INI ADALAH DATA '
INFO:root:b'YANG DIKIRIM ABC'
INFO:root:b'DEFGHIJKLMNOPQ'
INFO:root:File 'file_dikirim.txt' sent successfully.
INFO:root:closing
(base) jovyan@ee91a07ecadb:~/work$ █
```

**c. Penjelasan**

Pertama tama, kita perlu mengubah program server.py dan juga client.py.

server.py

sebelum:

```
server_address = ('0.0.0.0', 10000)
```

sesudah:

```
server_address = ('0.0.0.0', 32444)
```

client.py

tambahkan kode:

```
file_path = 'file_dikirim.txt'
if os.path.exists(file_path):
    with open(file_path, 'rb') as f:
        file_data = f.read()
        sock.sendall(file_data)
        logging.info(f'File '{file_path}' sent successfully.")
else:
    logging.warning(f'File '{file_path}' not found. Skipping file send.")
```

Penjelasan outputnya masih sama dengan soal no 2, namun ada tambahan sedikit, dimana setelah server menerima pesan secara langsung dari client, server tidak langsung menutup koneksinya, namun server membaca pesan yang berada di dalam file file\_dikirim.txt, lalu mengforward menjadi pesan yang akan ditampilkan setelah server membaca satu per satu message yang sudah

di pisahkan oleh tcp. Berbeda dengan pesan yang dikirim secara langsung, pesan yang dikirim melalui file, akan dibaca secara utuh karena dibaca dan dikirim dalam chunk/block besar. Sedangkan string langsung, terkadang akan dikirim menjadi bagian bagian oleh tcp.

#### Wireshark Mesin 1.

| No.  | Penjelasan   |
|------|--|
| 1174 | Paket ini menunjukkan bahwa <b>client (172.18.0.3)</b> mengirimkan data ke <b>server (172.18.0.2)</b> melalui koneksi TCP. |
| 1175 | Paket ini memberi <b>balasan dari server</b> ke client setelah menerima data, berisi pengiriman ulang pesan.               |

#### Wireshark Mesin 2.

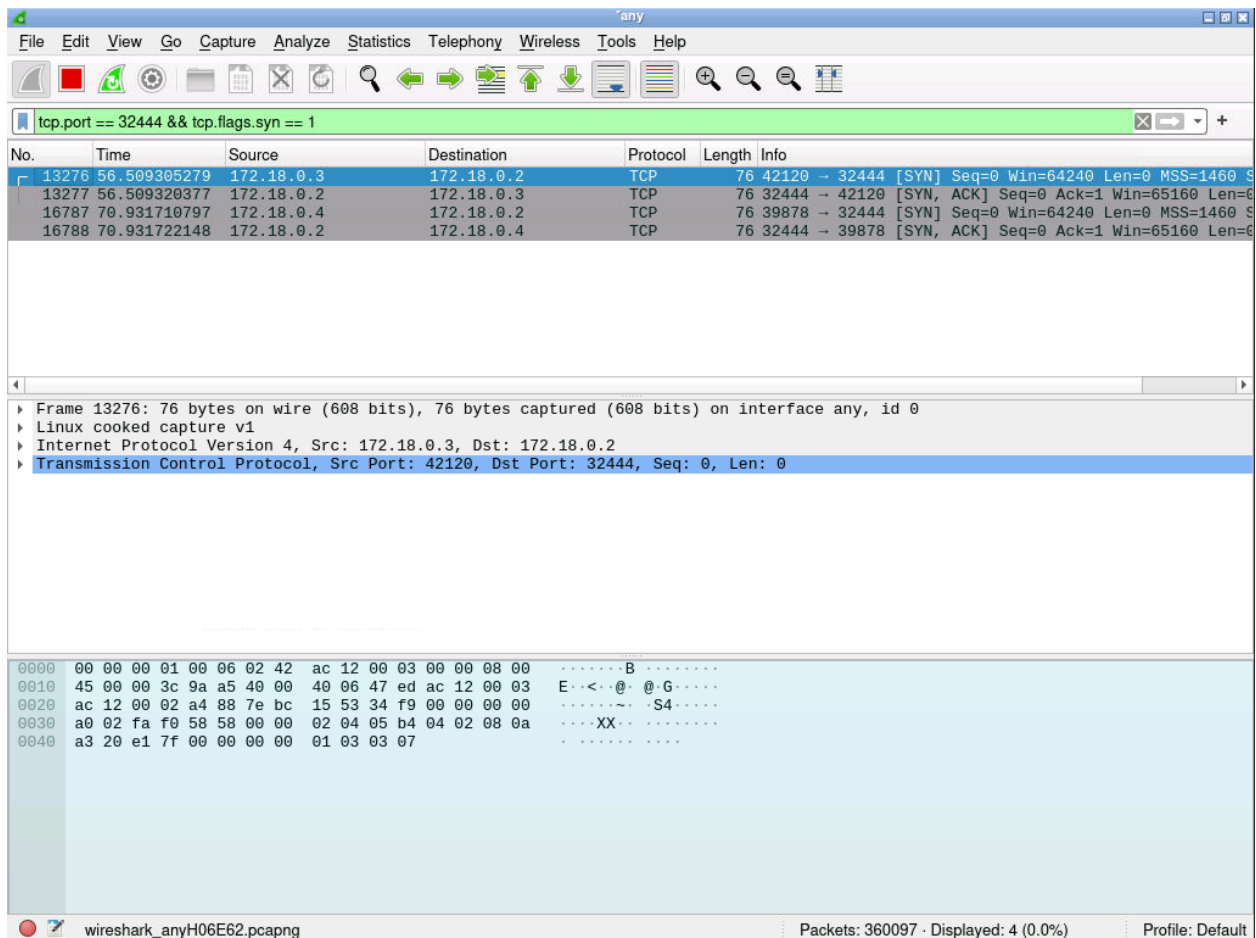
| No. | Penjelasan   |
|-----|--|
| 847 | Sama seperti 1174, ini adalah <b>client mengirim data</b> ke server. |
| 848 | Sama seperti 1175, ini adalah server mengirim balik data ke client.  |

4. Jalankan client di mesin-2 dan mesin-3 dengan server berada di mesin-1, jalankan client secara bersamaan, apakah yang terjadi? capturelah hasilnya, lakukan analysis menggunakan wireshark, capture hasilnya.
  - a. Screenshot Mesin 1

```
(base) jovyan@2285cc6d77e6:~/work$ python3 server_2.py
INFO:root:starting up on ('0.0.0.0', 32444)
INFO:root:waiting for a connection
```

```
INFO:root:connection from ('172.18.0.3', 42120)
INFO:root:received b'INI ADALAH DATA YANG DIKIRIM ABC'
INFO:root:sending back data
INFO:root:received b'DEFGHIJKLMNOPQ'
INFO:root:sending back data
INFO:root:received b'Halo Server, ini Pesan dari Mesi'
INFO:root:sending back data
INFO:root:received b'n 2'
INFO:root:sending back data
INFO:root:received b''
INFO:root:waiting for a connection
```

```
INFO:root:connection from ('172.18.0.4', 39878)
INFO:root:received b'INI ADALAH DATA YANG DIKIRIM ABC'
INFO:root:sending back data
INFO:root:received b'DFGHIJKLMNOPQ'
INFO:root:sending back data
INFO:root:received b'Halo Server, ini Pesan dari Mesi'
INFO:root:sending back data
INFO:root:received b'n 3'
INFO:root:sending back data
INFO:root:received b''
INFO:root:waiting for a connection
```



## b. Screenshot Mesin 2

```
INFO:root:connecting to ('172.18.0.2', 32444)
INFO:root:sending INI ADALAH DATA YANG DIKIRIM ABCDEFGHIJKLMNOPQ
INFO:root:b'INI ADALAH DATA '
INFO:root:b'YANG DIKIRIM ABC'
INFO:root:b'DFGHIJKLMNOPQ'
INFO:root:File 'file_mesin2.txt' sent successfully.
INFO:root:b'Halo Server, ini Pesan dari Mesi'
INFO:root:b'n 2'
INFO:root:closing
(base) jovyan@15bfad691b00:~/work$
```

c. Screenshot Mesin 3

```
INFO:root:connecting to ('172.18.0.2', 32444)
INFO:root:sending INI ADALAH DATA YANG DIKIRIM ABCDEFGHIJKLMNOPQ
INFO:root:b'INI ADALAH DATA '
INFO:root:b'YANG DIKIRIM ABC'
INFO:root:b'DEFGHIJKLMNOPQ'
INFO:root:File 'file_mesin3.txt' sent successfully.
INFO:root:b'Halo Server, ini Pesan dari Mesi'
INFO:root:b'n 3'
INFO:root:closing
(base) jovyan@8a58e811c2e2:~/work$
```

d. Penjelasan

Disini saya menambahkan mesin 3 sebagai client baru, yang menjalankan program client.py yang sama dengan mesin 2, namun perbedaan nya adalah pesan yang dikirim, dimana akan saya memodifikasi untuk file yang dikirim, dan message yang dikirim.

Masih menggunakan konfigurasi yang sama, dimana port server nya menggunakan 32444

Mesin 2:

```
file_path = 'file_mesin2.txt'
```

Dengan isi pesan

```
Halo Server, ini Pesan dari Mesin 2
```

Mesin 3:

```
file_path = 'file_mesin3.txt'
```

Dengan isi pesan

```
Halo Server, ini Pesan dari Mesin 3
```

Dapat dilihat dari screenshot Wireshark di mesin 1 (server), bahwa mesin 2 (client 1) pertama kali mencoba melakukan koneksi ke server. Setelah koneksi berhasil dilakukan melalui proses three-way handshake, mesin 2 mulai mengirimkan pesan dan file.

Tidak lama kemudian, terlihat adanya koneksi baru dari mesin 3 (client 2) ke server. Server pun menerima koneksi tersebut dan melakukan handshake juga.

Hal ini menunjukkan bahwa server mampu menangani koneksi dari beberapa client, meskipun secara berurutan (satu per satu). Artinya, server tidak langsung menutup koneksi setelah satu client selesai, tapi tetap dalam kondisi aktif untuk menerima koneksi baru dari client lain.

Muhammad Nabil Fadhil  
5025221200

Dengan demikian, server dapat dianggap memiliki kemampuan menangani koneksi secara paralel apabila didukung dengan implementasi multi-threading atau multi-processing, atau setidaknya mampu menangani beberapa koneksi masuk secara bergantian tanpa langsung menutup aplikasi.

Muhammad Nabil Fadhil  
5025221200