Nama : Christoforus Indra Bagus Pratama

NRP : 5025231124

Kelas : Pemrograman Jaringan – D

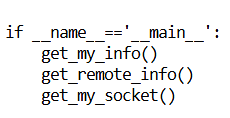
**TUGAS 1**

**Nomor 1**

Jalankan socket\_info.py di mesin-1 dan mesin-2, capturelah hasilnya, lakukan analisis menggunakan wireshark, capture hasilnya.

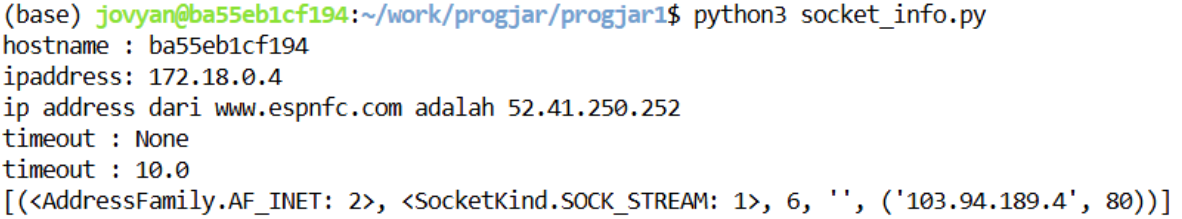
* Langkah-langkah pengerjaan

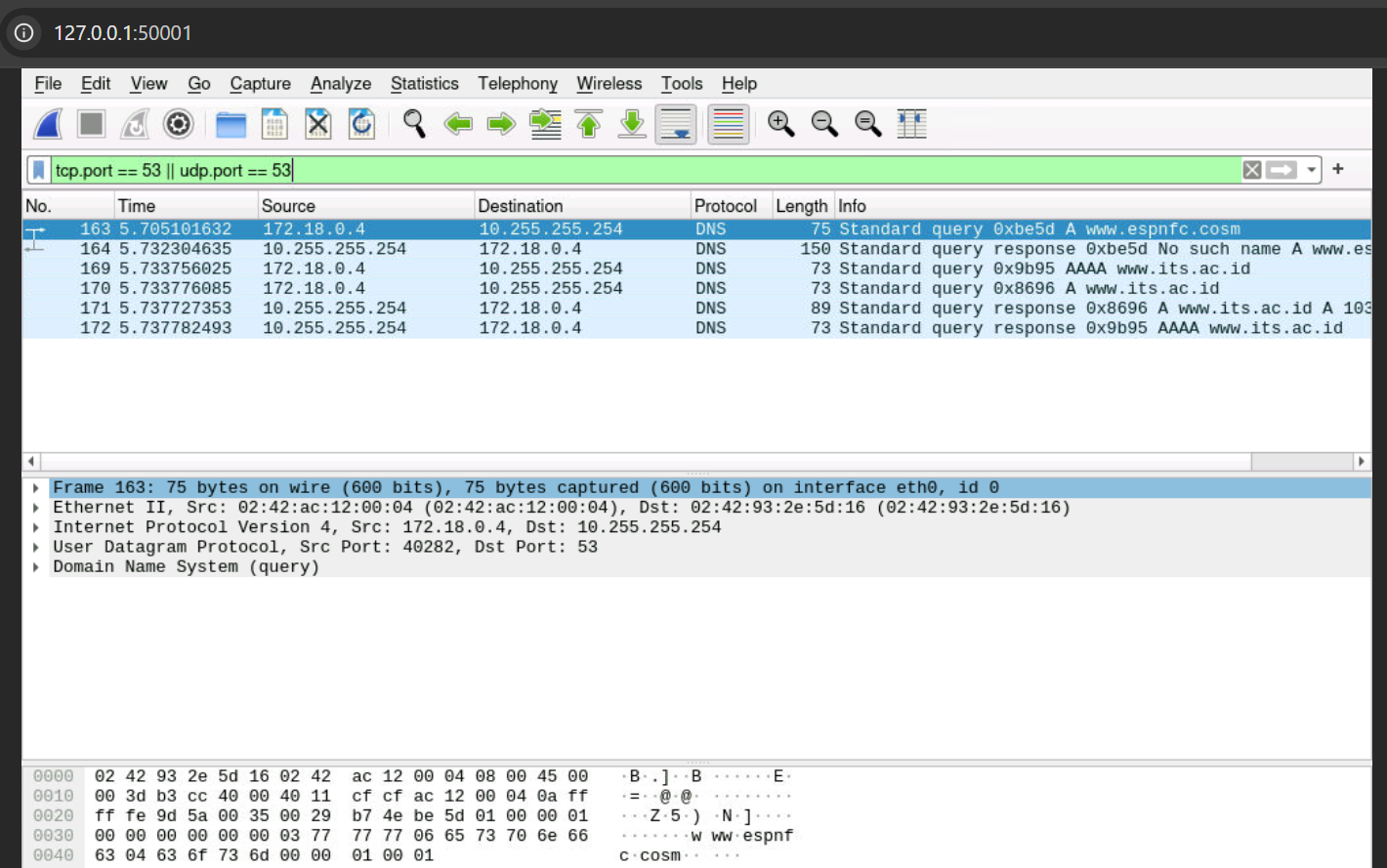
1. Jalankan mesin 1 dan mesin 2. Kemudian, pindah ke direktori /work/progjar/progjar1 dengan command : cd progjar/progjar1
2. Edit isi file dengan command : vim socket\_info.py. Lalu, hapus tanda comment pada perintah get\_remote\_info() dan get\_my\_socket() sehingga nanti perintah tersebut bisa dijalankan.



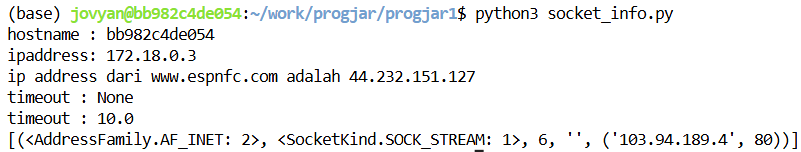
1. Jalankan program socket\_info.py di mesin 1 dan mesin 2 dengan command : python3 socket\_info.py
2. Lakukan analisis pada wireshark

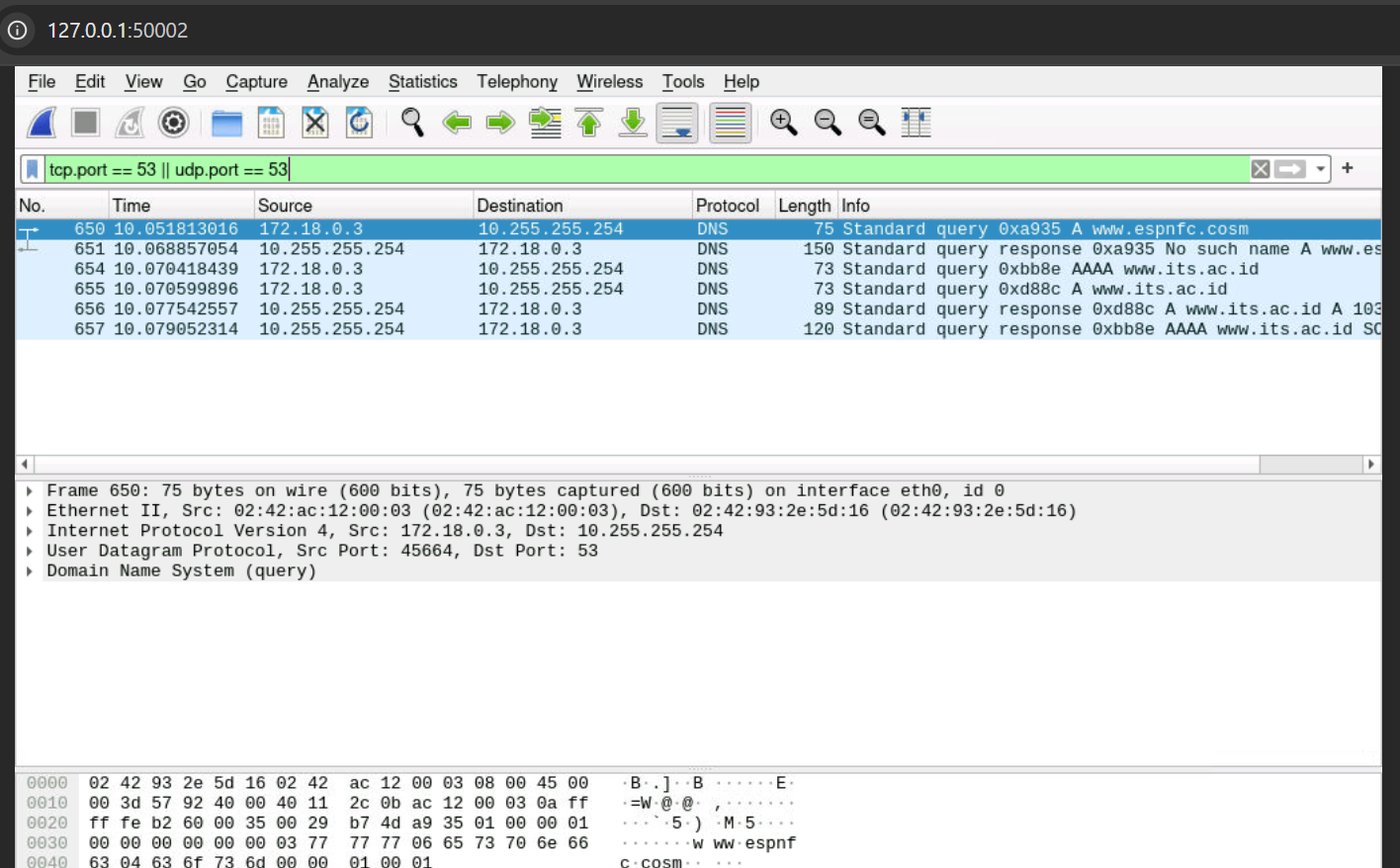
Mesin1 :





Mesin 2 :





Analisis :

Dikarenakan dalam program terdapat pemanggilan protocol TCP bahwa protocol itu akan menjalankan program dalam sekali run. Pertama, program melakukan pemanggilan [www.espnfc.com](http://www.espnfc.com) untuk mengajukan komunikasi. Kemudian, [www.espnfc.com](http://www.espnfc.com) memberikan respon balik kepada mesin 1. Kedua, program melakukan pemanggilan ke DNS [its.ac.id](http://its.ac.id) dengan mengakses port 53 untuk mengakses DNS, yaitu IPv4 adalah A record dan IPv6 adalah AAAA record. Terdapat IP asal yaitu 172.18.0.4 ke IP tujuan 10.255.255.254. Kemudian, [its.ac.id](http://iws.ac.id) memberikan response dan menyetujui untuk melakukan komunikasi dengan mesin 1.

Pada mesin 2, hasil analisis juga sama tetapi hanya berbeda IP asal, yaitu 172.18.0.3.

Alasan menggunakan filter port 53 pada wireshark :

karena socket.getaddrinfo('www.its.ac.id',80,proto-socket.IPPROTO\_TCP) memicu DNS lookup-query yang dikirim ke port 53 untuk menerjemahkan nama domain menjadi alamat IP."

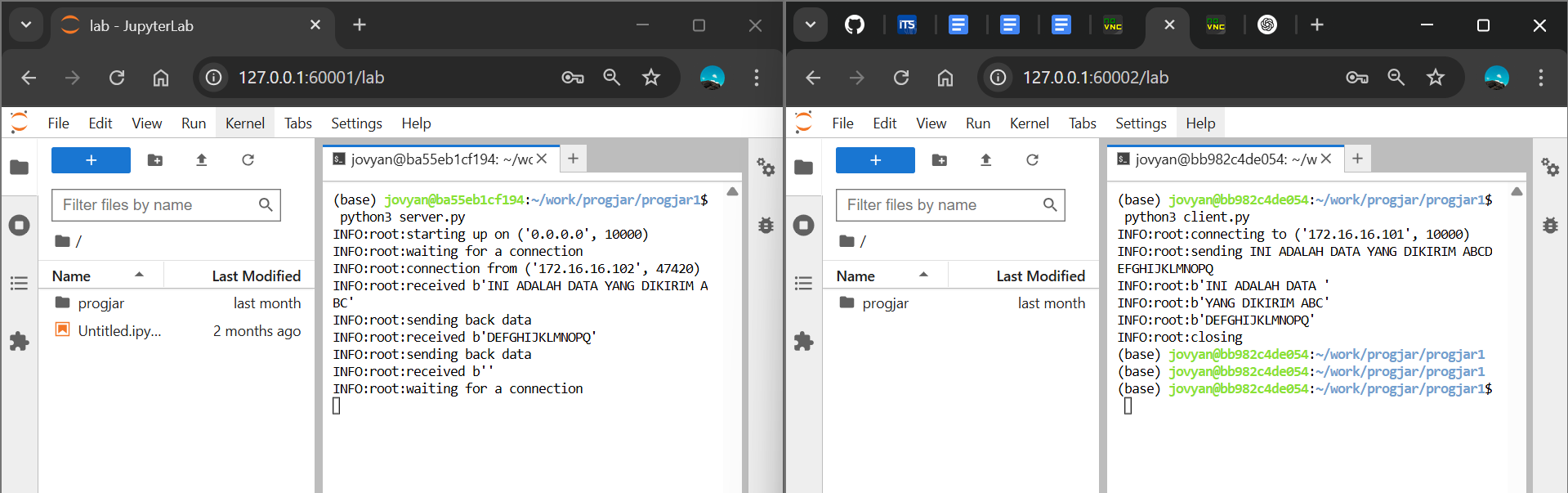
**Nomor 2**

Jalankan server.py di mesin-1 dan client.py di mesin-2, sesuaikan isi program, pastikan komunikasi dapat dilakukan, capturelah hasilnya, lakukan analisis menggunakan wireshark, capture hasilnya

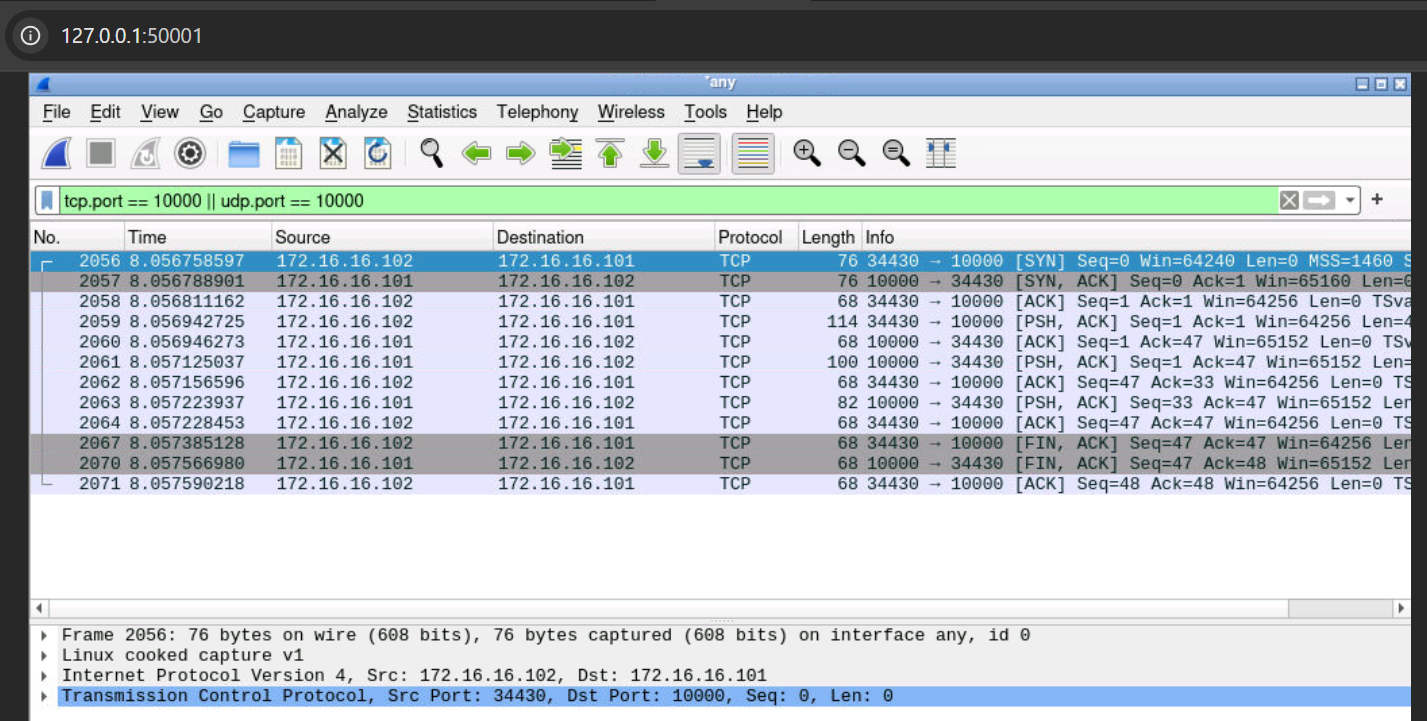
* Langkah-langkah pengerjaan

1. Ubahlah server\_address pada file [client.py](http://client.py) pada mesin2 dari ‘localhost” menjadi ‘172.16.16.101’.
2. Jalankan program dengan command `python3 [server.py](http://server.py)` pada mesin1 dan `python3 [client.py](http://client.py)` pada mesin2.
3. Lakukan analisis pada wireshark

Hasil :



Mesin 1:



Analisis :

Mesin 1 menjalankan server yang mendengarkan (listen) pada semua alamat IP (0.0.0.0) di port 10000. Di sisi lain, Mesin 2 bertindak sebagai client dan mengirimkan kalimat dalam bentuk byte. Dalam skrip client.py, perintah : sock.sendall(message.encode()), mengirimkan pesan b'INI ADALAH DATA YANG DIKIRIM ABCDEFGHIJKLMNOPQ' ke server yang beralamat di 172.16.16.101:10000. Server menerima data ini, memprosesnya, lalu membalas.

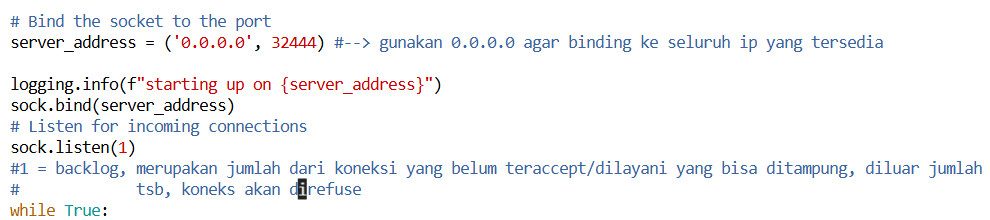
Karena TCP adalah protokol berbasis aliran (stream), client kadang menerima balasan server dalam potongan-potongan (“chunks”), misalnya b'INI ADALAH DATA '. Ini terjadi karena data dipisah sesuai kapasitas buffer internal. Setelah semua pertukaran selesai, client menutup koneksi—ditandai server menerima b'' (empty bytes) sebagai sinyal end-of-stream. Dengan begitu, komunikasi byte-by-byte antara server dan client melalui socket Python berjalan lancar dan terstruktur.

**Nomor 3**

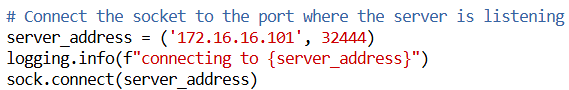
Jalankan kembali soal nomor 2, namun kali ini rubahlah komunikasi agar berjalan di port 32444, kirimkan isi sebuah file, dan capturelah hasilnya, lakukan analisis menggunakan wireshark, capture hasilnya

* Langkah - langkah pengerjaan

1. Ubah port di file [server.py](http://server.py) menjadi 32444 di mesin1 dengan command : vim [server.py](http://server.py)

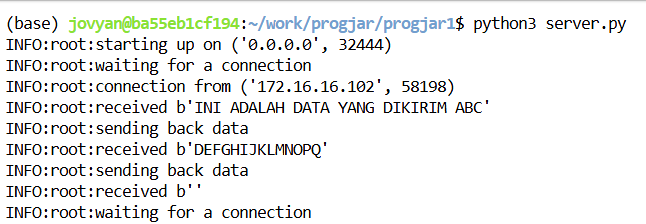


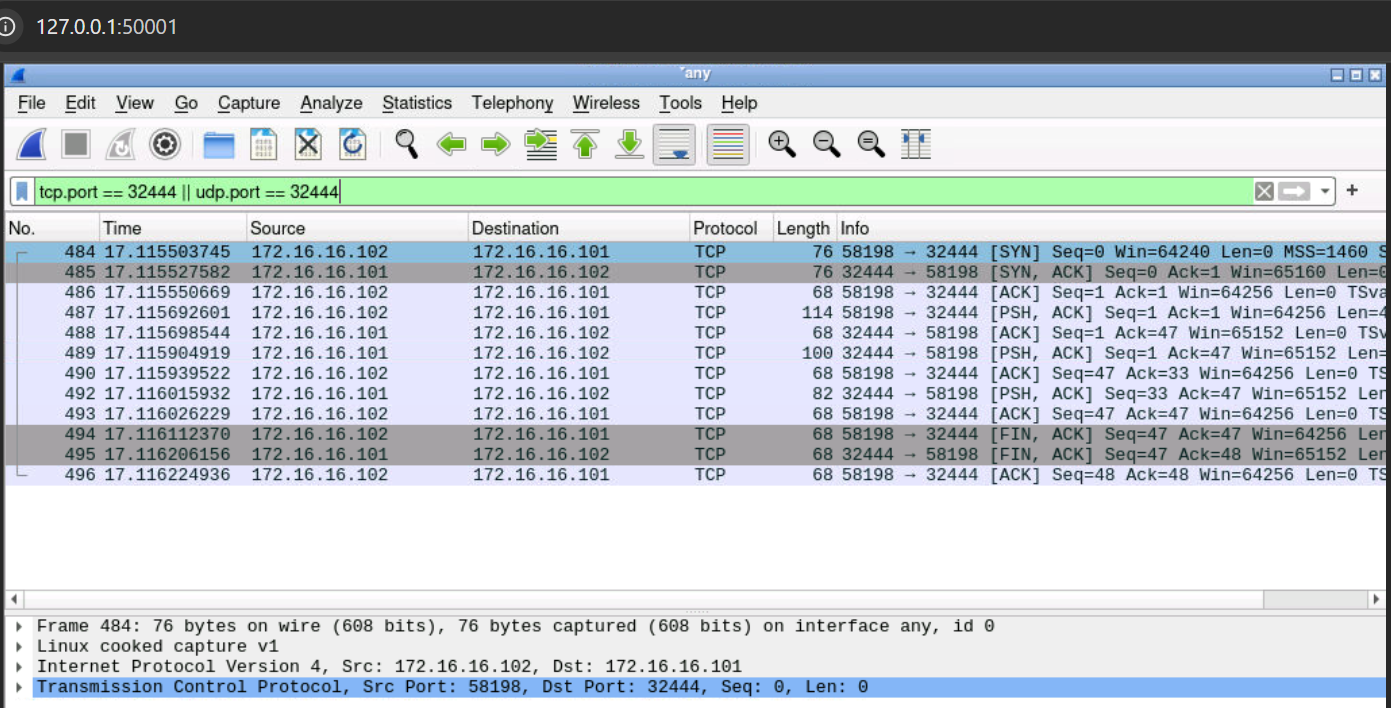
1. Ubah port di file client[.py](http://server.py) menjadi 32444 di mesin2 dengan command : vim client[.py](http://server.py)



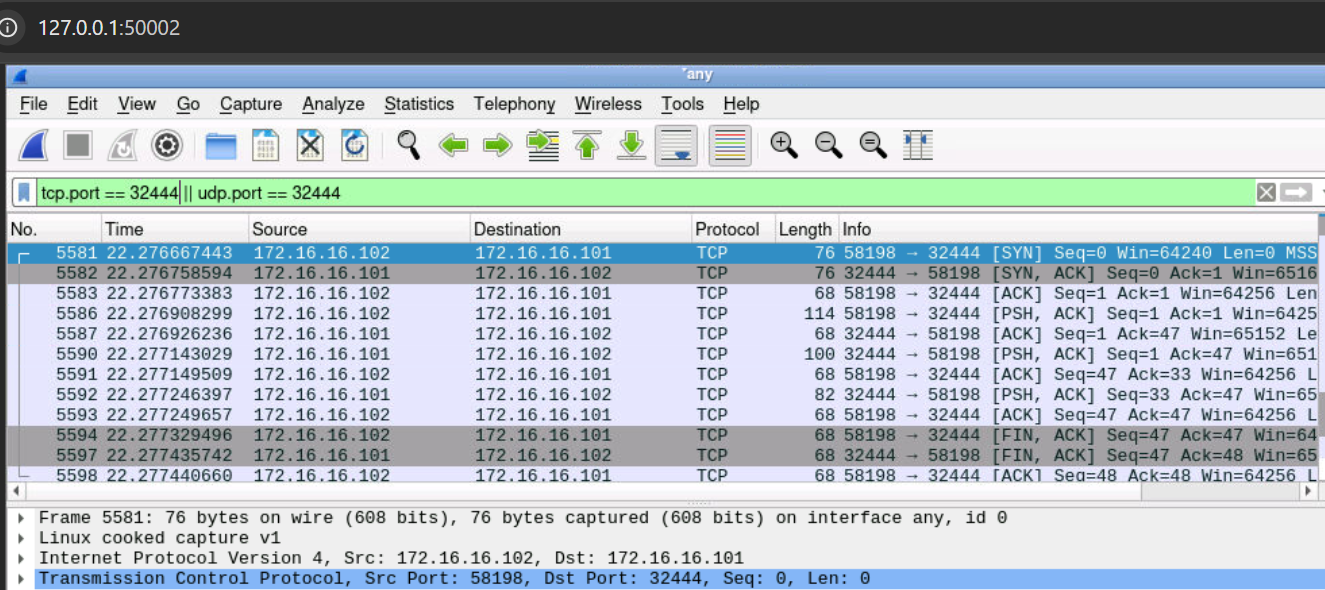
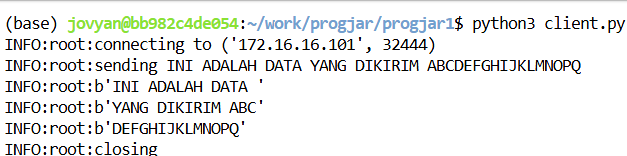
1. Jalankan program [server.py](http://server.py) di mesin 1 dan [client.py](http://clinet.py) di mesin 2
2. Lakukan analisis di wireshark

Mesin 1 :





Mesin 2 :



Analisis :

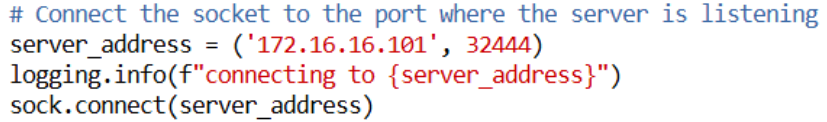
Mesin 1 menjalankan server Python yang “listen” pada semua alamat IP (0.0.0.0) di port 32444. Mesin 2 berperan sebagai client; ia membuka koneksi TCP ke 172.16.16.101:32444 lalu mengirimkan sebuah string yang diubah ke format byte : sock.sendall(message.encode()). Contohnya, pesan b'INI ADALAH DATA YANG DIKIRIM ABCDEFGHIJKLMNOPQ' dikirimkan dari client ke server. Setelah server menerima dan memproses data ini, server mengirim balasan—tetapi karena TCP adalah protokol aliran, client kadang menerima respons dalam potongan-potongan kecil (chunks), misalnya b'INI ADALAH DATA ' sesuai kapasitas buffer. Ketika pertukaran data selesai, client menutup koneksi dengan mengirim segmen FIN, yang kemudian diakui oleh server dengan segmen ACK. Server akhirnya juga mengirim FIN, diikuti oleh ACK dari client. Sinyal b'' (empty bytes) pada sisi server menandai bahwa aliran data telah berakhir. Dengan demikian, terjadi satu sesi komunikasi penuh—mulai dari tiga-way handshake, pertukaran data byte-by-byte, hingga four-way FIN/ACK teardown—semuanya berjalan lancar dan terstruktur.

**Nomor 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 analisis menggunakan wireshark, capture hasilnya

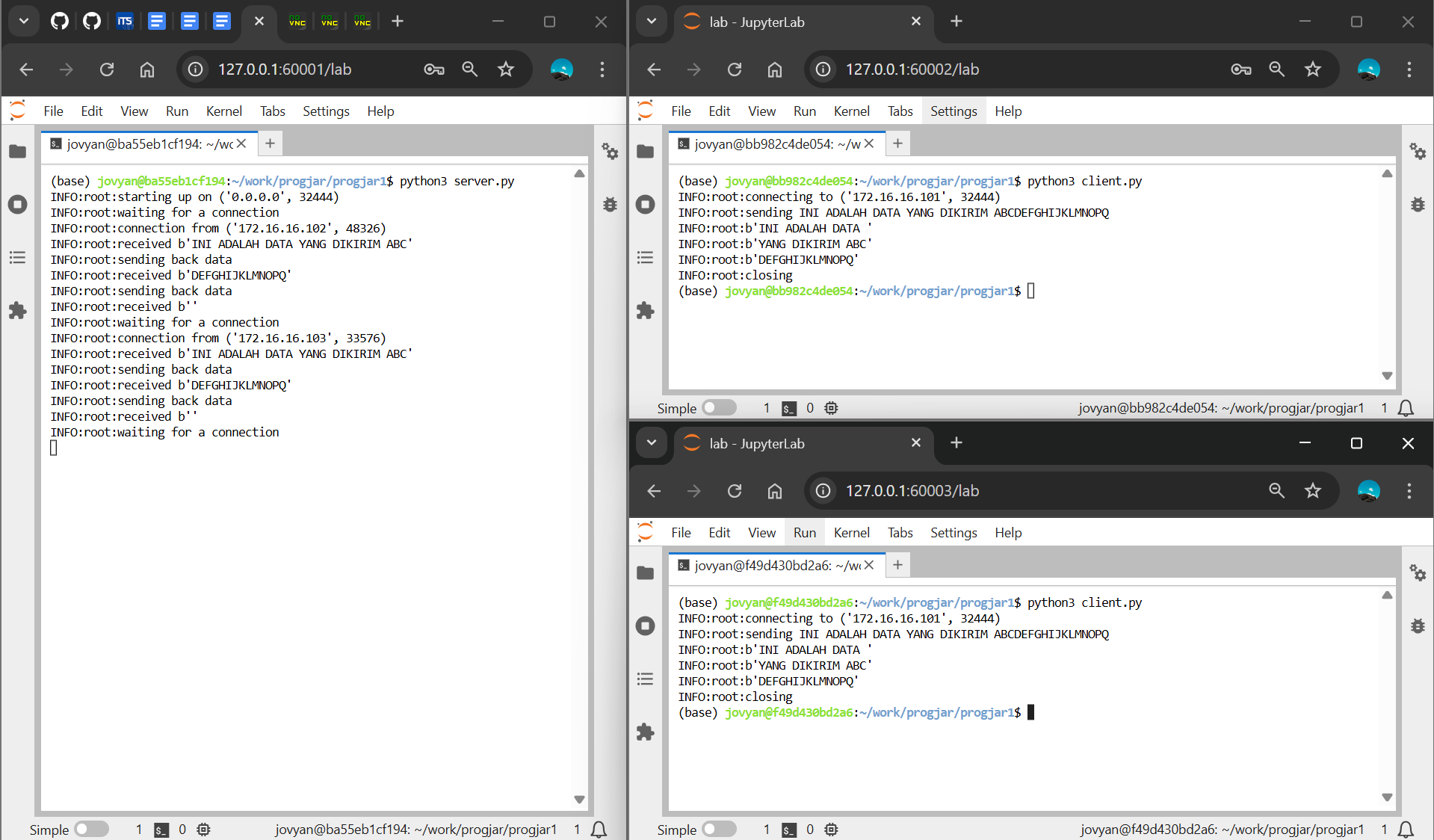
* Langkah-langkah pengerjaan

1. Ubah port pada [client.py](http://client.py) di mesin3 menjadi 32444 dan ubah server address dari ‘localhost’ menjadi ‘172.16.16.101’ dengan command : vim [client.py](http://client.py)

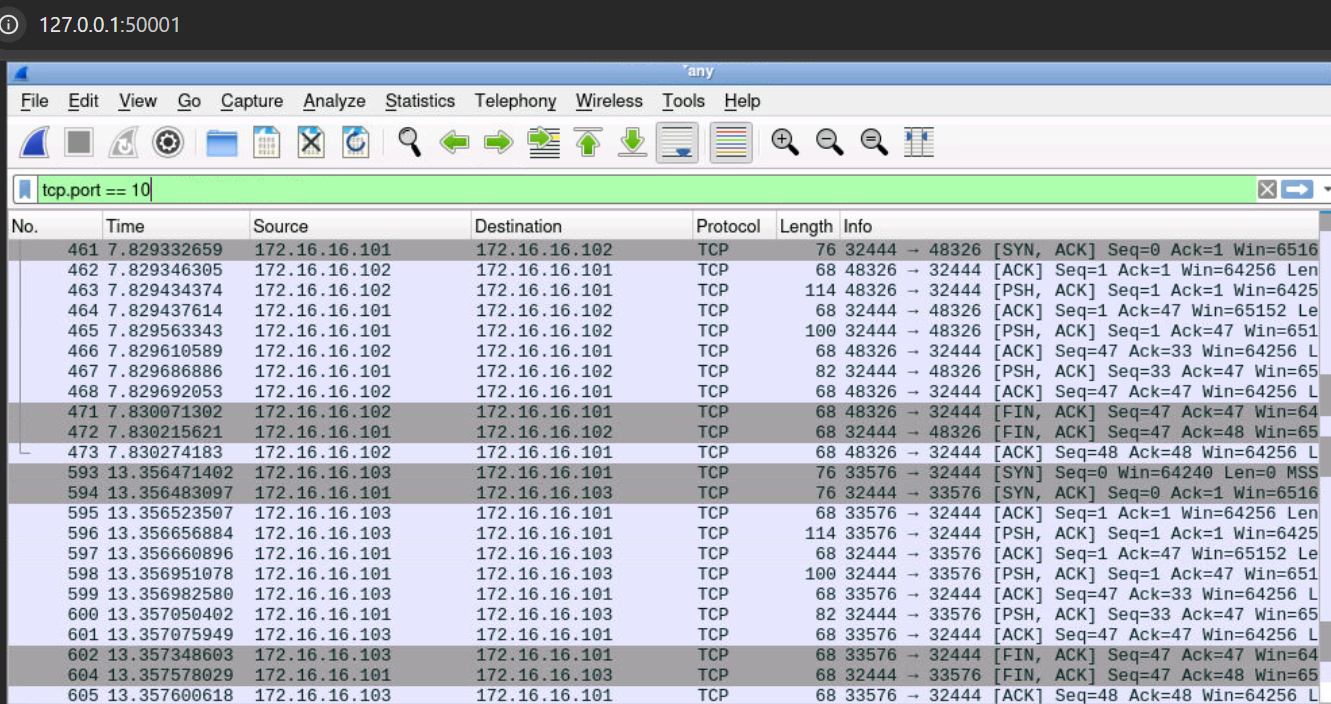


1. Jalankan program [server.py](http://server.py) di mesin 1 dan [client.py](http://client.py) di mesin 2 dan mesin 3
2. Lakukan analisis pada wireshark

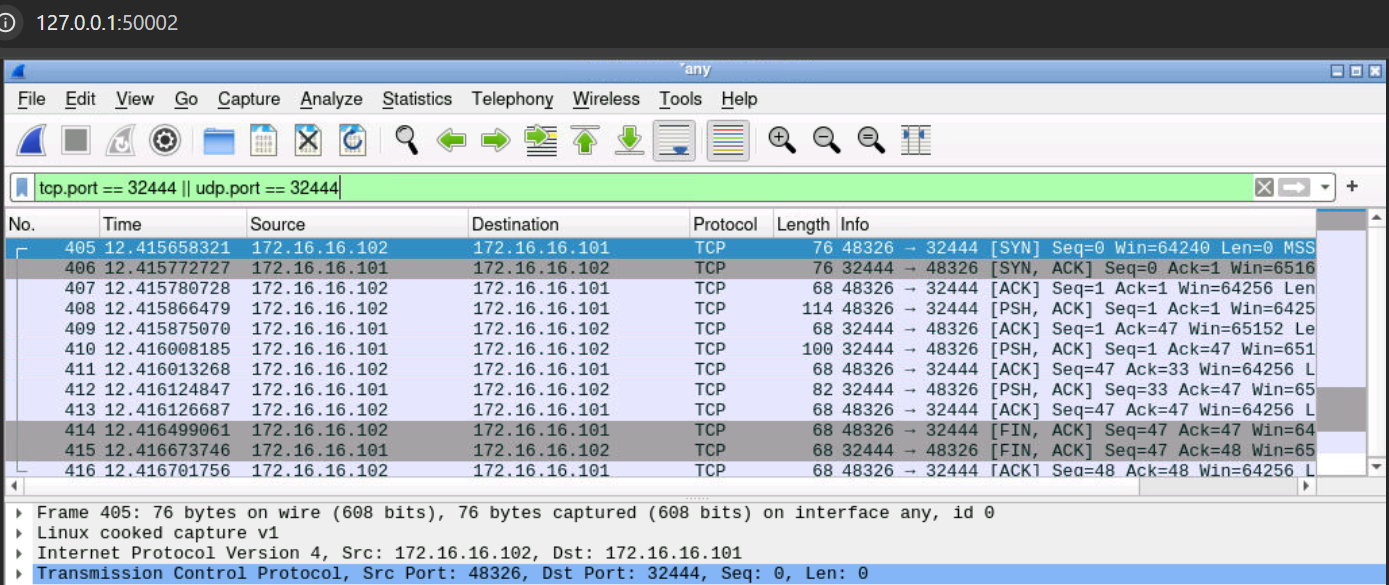
Hasil :



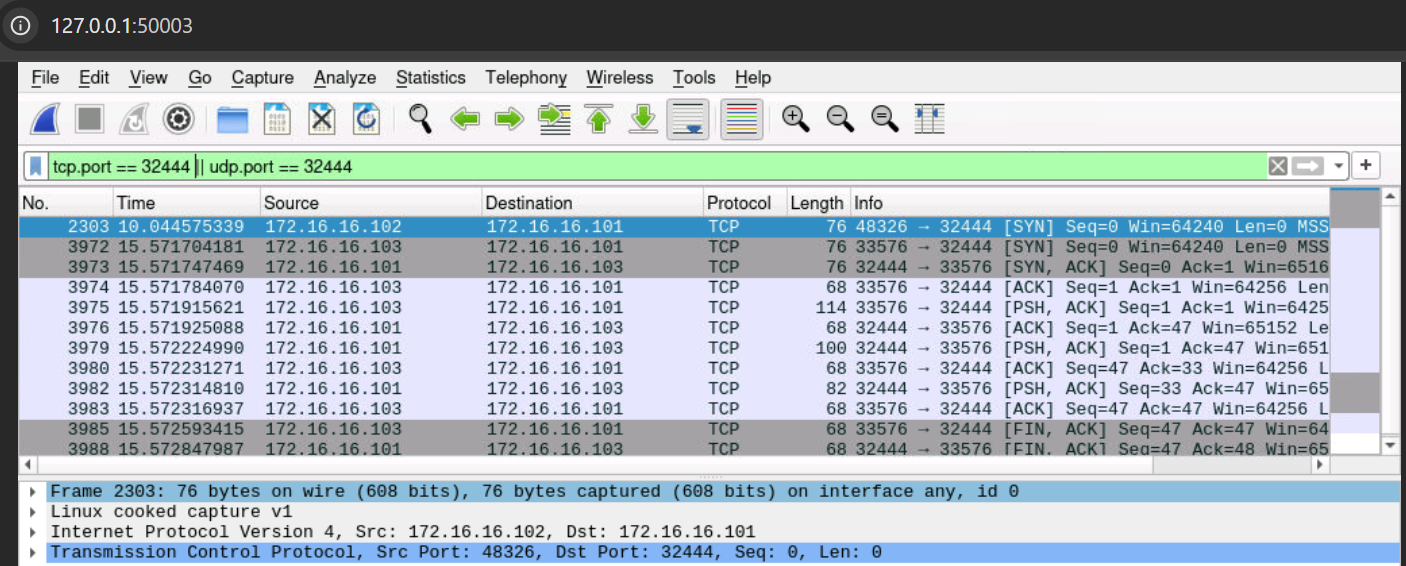
Mesin 1 :



Mesin 2 :



Mesin 3 :



Analisis :

Ketika kedua client (mesin-2 dan mesin-3) dijalankan serentak menuju server di mesin-1, Wireshark memperlihatkan bahwa server sebenarnya hanya memproses satu koneksi pada satu waktu. Pertama, mesin-2 menginisiasi tiga‑way handshake dengan mengirim paket SYN ke 172.16.16.101:32444, direspons oleh server dengan SYN‑ACK, lalu mesin-2 membalas ACK untuk menyelesaikan handshake. Setelah pertukaran data selesai, mesin-2 mengirim FIN dan server menjawab dengan ACK, menandai berakhirnya sesi.

Begitu sesi mesin-2 benar‑benar terteardown, server baru menerima SYN dari mesin-3. Proses handshake, pertukaran data, dan teardown (FIN/ACK) berjalan persis sama seperti pada mesin-2. Karena server hanya satu socket utama yang menangani accept, koneksi mesin-3 menunggu hingga mesin-2 selesai, sehingga pola di Wireshark tampak berurutan: dari SYN–SYN‑ACK–ACK–FIN/ACK milik mesin-2, dilanjutkan oleh rangkaian SYN–SYN‑ACK–ACK–FIN/ACK milik mesin-3. Hasilnya, meski client dijalankan bersamaan, komunikasi tetap terlayani secara bergantian.