

Tugas1 Tugas 1 tugas 1 code

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

```
(base) joyyan@f3bfff207ab90:~/work/progjar/progjar$ python3 socket_info.py
timeout : None
[(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>, 6, '', ('103.94.189.4', 80))]
(base) joyyan@f3bfff207ab90:~/work/progjar/progjar$
```

Would you like to receive official Jupyter news? X
Please read the privacy policy.
[Open privacy policy](#) Yes No

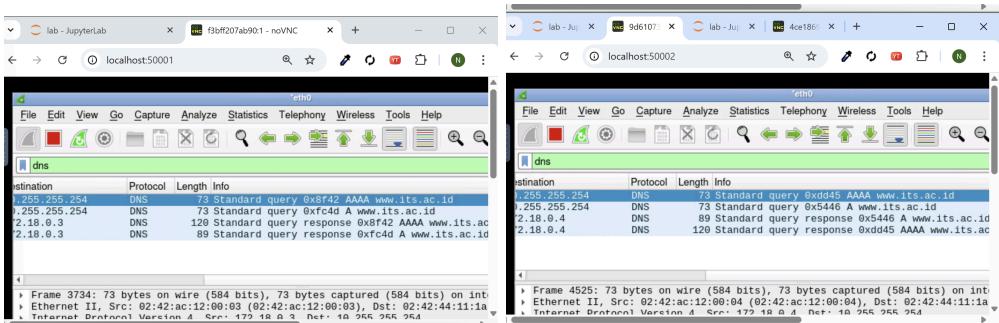
Simple 1 0 0 joyyan@f3bfff207ab90:~/work/progjar/progjar 1


```
(base) joyyan@9d6107341712:~/work/progjar/progjar$ python3 socket_info.py
timeout : None
[(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>, 6, '', ('103.94.189.4', 80))
(base) joyyan@9d6107341712:~/work/progjar/progjar$
```

Would you like to receive official Jupyter news? X
Please read the privacy policy.
[Open privacy policy](#) Yes No

Simple 1 0 0 joyyan@9d6107341712:~/work/progjar/progjar 1

Saat program dijalankan, kita bisa melihat socket yang baru dibuat awalnya tidak memiliki batas waktu (None), lalu diatur menjadi 10 detik untuk mencegah koneksi menggantung terlalu lama. Program kemudian sukses menemukan alamat website ITS (www.its.ac.id) pada port 80, menampilkan detail lengkap termasuk alamat IP 103.94.189.5 dan port tujuan. Ini membuktikan socket berhasil dibuat, timeout diatur, dan pencarian alamat website berjalan lancar.

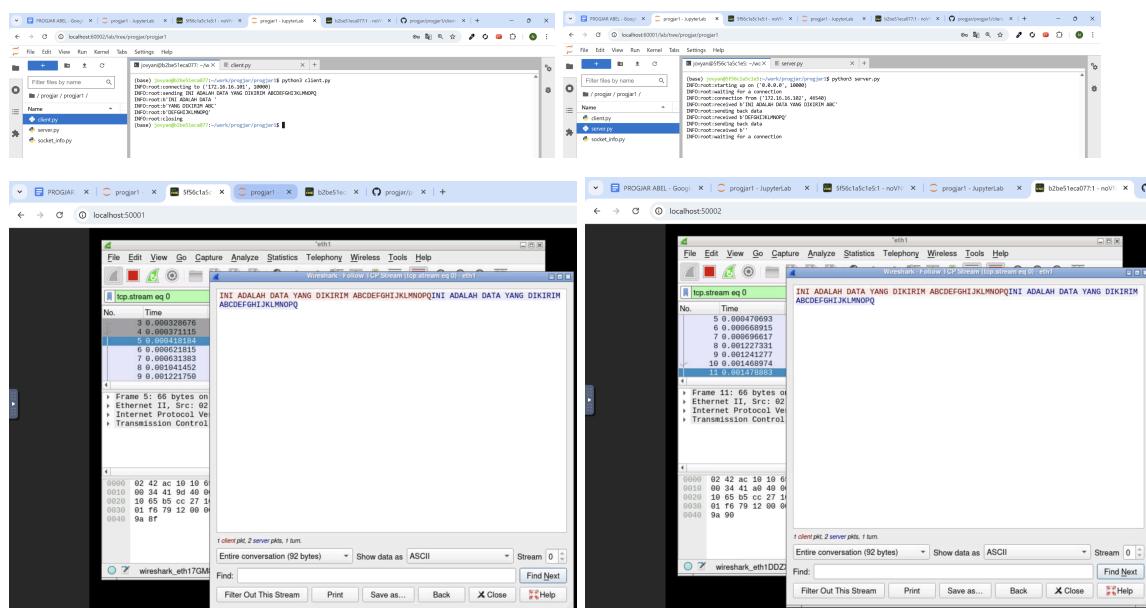


Saat menjalankan script socket_info.py di dua mesin berbeda, terlihat perbedaan IP karena Wireshark menggunakan antarmuka eth0. Yang menarik, ketika kita amati di kedua mesin, terjadi proses tanya-jawab DNS ke website ITS. Kedua mesin sama-sama mencari dua jenis record - A record untuk alamat IPv4 dan AAAA record untuk IPv6, menunjukkan bagaimana sistem mempersiapkan semua kemungkinan koneksi. Seperti detektif yang memeriksa semua petunjuk sebelum memutuskan rute terbaik untuk mencapai tujuannya.

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

SERVER Program ini membuat socket TCP/IP dan mengaktifkan opsi reuse address (SO_REUSEADDR) agar bisa memakai kembali alamat yang sama. Server diikat ke IP 0.0.0.0 pada port 10000, sehingga bisa menerima koneksi dari semua IP. Server kemudian mendengarkan satu koneksi dalam satu waktu. Dalam loop tanpa akhir, server menunggu client masuk, mencatat alamat client, menerima data, lalu mengirimkan kembali data tersebut. Jika tidak ada data diterima, koneksi ditutup.

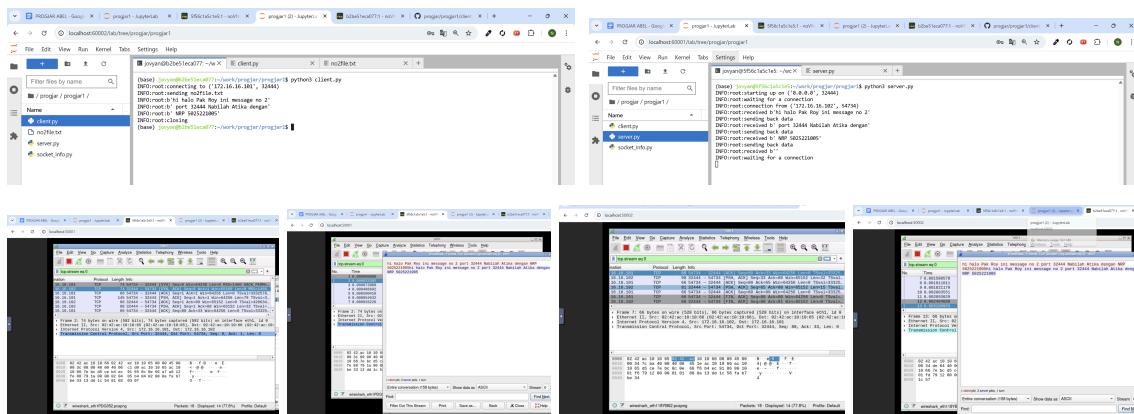
CLIENT Kode ini membuat socket TCP/IP dan menghubungkan client ke server di IP 172.16.16.101 melalui port 10000. Setelah berhasil tersambung, client mengirimkan pesan 'INI ADALAH DATA YANG DIKIRIM ABCDEFGHIJKLMNOPQ' dan mencatatnya. Selanjutnya, client menerima balasan dari server secara bertahap dalam ukuran 16 byte hingga seluruh data terkumpul, lalu mencatat setiap bagian yang diterima.



Mesin 1 berperan sebagai server yang terus menunggu pesan dari client pada port 10000, tanpa peduli alamat IP pengirim. Mesin2 bertindak sebagai client yang mengirimkan sebuah string ke server. Server lalu membalas dengan mengirimkan kembali string tersebut. Karena komunikasi dibatasi hanya 32 byte per kali kirim, proses ini diulang sampai seluruh data selesai dikirim. Setelah itu, client menutup koneksi, sedangkan server tetap aktif menunggu koneksi baru.

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

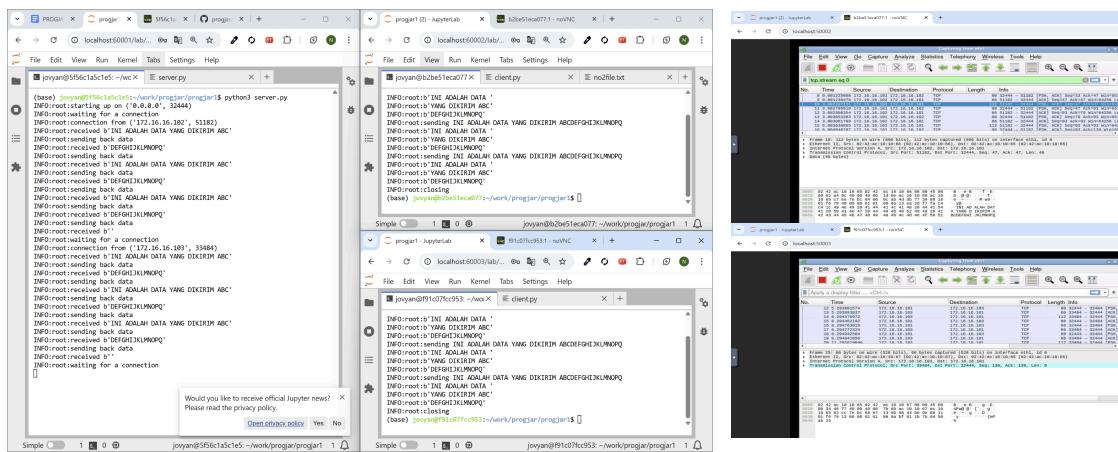
SERVER Program ini membuat socket TCP/IP dan mengaktifkan opsi reuse address (SO_REUSEADDR) agar bisa memakai kembali alamat yang sama. Server diikat ke IP 0.0.0.0 pada port 32444, sehingga bisa menerima koneksi dari semua IP. Server kemudian mendengarkan satu koneksi dalam satu waktu. Dalam loop tanpa akhir, server menunggu client masuk, mencatat alamat client, menerima data, lalu mengirimkan kembali data tersebut. Jika tidak ada data diterima, koneksi ditutup. CLIENT Program ini membuat socket TCP/IP dan menghubungkan klien ke server di IP 172.16.16.101 melalui port 32444. Setelah koneksi berhasil, klien membaca isi file no2file.txt, lalu mengirimkan seluruh isi file tersebut ke server. Selama proses ini, klien mencatat status pengiriman. Setelah data dikirim, klien menunggu balasan dari server dan terus menerima data dalam potongan sampai seluruh respon diterima. Setiap bagian yang diterima dicatat, dan setelah proses selesai, koneksi ditutup.



Meniru pola pada soal 2, Mesin 1 bertindak sebagai server yang menunggu koneksi client melalui socket TCP pada port 32444, dengan binding ke seluruh IP menggunakan alamat 0.0.0.0 dan menggunakan loop while True untuk terus menerima koneksi. Client dari Mesin 2 dan Mesin 3 menggunakan script client.py untuk membaca isi file no2file.txt dan mengirimkannya ke server, yang kemudian meng-echo kembali isi file tersebut. Karena server tidak menggunakan thread, hanya satu koneksi yang bisa dilayani pada satu waktu, sehingga saat Mesin 2 sedang dilayani, koneksi dari Mesin 3 harus menunggu hingga koneksi sebelumnya ditutup. Selama proses ini, komunikasi berhasil ditangkap menggunakan Wireshark, yang menunjukkan proses handshake, pengiriman file, dan echo data dari server secara berurutan.

4. Jalankan client di mesin-2 dan mesin-3 dengan server berada di mesin-I, jalankan client secara bersamaan, apakah yang terjadi ? capturelah hasilnya, lakukan analisis menggunakan wireshark, capture hasilnya

Dalam percobaan ini, digunakan dua buah client yang dijalankan secara terpisah di Mesin 2 dan Mesin 3. Untuk keperluan uji coba, kode client.py telah dimodifikasi agar mengirimkan pesan sebanyak tiga kali, masing-masing disertai jeda waktu selama tiga detik antar pengiriman. Hal ini dilakukan guna mengamati bagaimana server menangani beberapa pengiriman data secara bertahap dari setiap client, serta untuk memperjelas urutan komunikasi yang terekam dalam proses perekaman paket menggunakan Wireshark.



Hasil pengamatan menunjukkan bahwa komunikasi antara Mesin 1 dan Mesin 2 terekam pada detik ke-0, 3, dan 6, dengan jeda waktu konsisten sekitar 3 detik di setiap pengiriman. Sementara itu, interaksi antara Mesin 1 dan Mesin 3 tercatat pada detik ke-5, 8, dan 11, juga dengan pola jeda yang serupa. Pola waktu ini sejalan dengan penyesuaian pada kode client.py yang telah dimodifikasi untuk mengirim pesan dalam interval waktu tertentu. Keterbatasan server yang tidak menggunakan mekanisme multithreading menyebabkan hanya satu client yang bisa dilayani dalam satu waktu. Akibatnya, ketika Mesin 2 sedang terhubung dan berkomunikasi dengan server, Mesin 3 harus menunggu hingga proses tersebut selesai dan koneksi ditutup terlebih dahulu. Barulah setelah itu server dapat menerima dan memproses permintaan dari client berikutnya. Meskipun demikian, server tetap berada dalam kondisi siaga untuk melayani koneksi selanjutnya secara berurutan. Dari hasil tersebut, terlihat bahwa server mampu melayani lebih dari satu client secara bergantian tanpa hambatan, di mana setiap client tetap dapat melakukan komunikasi sesuai jeda waktu yang telah ditentukan dalam program.