

Nama : Ivan Fairuz Adinata

NRP : 5025221167

Kelas : Progar D

LAPORAN TUGAS 3 PROGJAR D

1. Link GitHub: <https://github.com/mlikiwe/progar-d-2025-tugas3.git>
2. Isi Dokumen PROTOKOL.txt

FILE SERVER

TUJUAN: melayani client dalam request file server

ATURAN PROTOKOL:

- client harus mengirimkan request dalam bentuk string
- string harus dalam format
REQUEST spasi PARAMETER
- PARAMETER dapat berkembang menjadi PARAMETER1 spasi PARAMETER2 dan seterusnya

REQUEST YANG DILAYANI:

- informasi umum:
 - * Jika request tidak dikenali akan menghasilkan pesan
 - status: ERROR
 - data: request tidak dikenali
 - * Semua result akan diberikan dalam bentuk JSON dan diakhiri dengan character ascii code #13#10#13#10 atau "\r\n\r\n"

LIST

- * TUJUAN: untuk mendapatkan daftar seluruh file yang dilayani oleh file server
- * PARAMETER: tidak ada
- * RESULT:
 - BERHASIL:
 - status: OK
 - data: list file
 - GAGAL:
 - status: ERROR
 - data: pesan kesalahan

GET

- * TUJUAN: untuk mendapatkan isi file dengan menyebutkan nama file dalam parameter

- * PARAMETER:
 - PARAMETER1 : nama file
- * RESULT:
 - BERHASIL:
 - status: OK
 - data_namafile : nama file yang diminta
 - data_file : isi file yang diminta (dalam bentuk base64)
 - GAGAL:
 - status: ERROR
 - data: pesan kesalahan

UPLOAD

- * TUJUAN: untuk mengunggah file dari client ke server dengan menyebutkan nama file dalam parameter
- * PARAMETER:
 - PARAMETER1 : nama file
 - PARAMETER2 : isi file dengan encode base64
- * RESULT:
 - BERHASIL:
 - status: OK
 - data_namafile : nama file yang berhasil diunggah
 - GAGAL:
 - status: ERROR
 - data: pesan kesalahan

Perintah "UPLOAD" ini digunakan oleh client untuk mengirimkan sebuah file ke server agar dapat disimpan di dalam folder files. Request atau perintah ini membutuhkan dua parameter, yaitu nama file dan isi dari file tersebut yang telah di-encode dengan base64 sesuai dengan ketentuan. Untuk isi dari file akan otomatis di generate oleh fungsi yang ada di script. Setelah server menerima dan berhasil menyimpan file, maka akan dikirimkan respon dengan status OK dan nama file yang berhasil diunggah. Namun, apabila terjadi sebuah kesalahan, seperti kurangnya parameter atau proses decoding yang gagal, maka server akan merespons dengan status "ERROR" dan pesan kesalahan yang sesuai.

DELETE

- * TUJUAN: untuk menghapus file dari file server dengan menyebutkan nama file dalam parameter
- * PARAMETER:
 - PARAMETER1 : nama file

- * RESULT:
- BERHASIL:
 - status: OK
 - data: pesan konfirmasi bahwa file berhasil dihapus
- GAGAL:
 - status: ERROR
 - data: pesan kesalahan

Perintah "DELETE" memungkinkan client untuk menghapus file yang ada di folder files di server. Parameter yang diterima oleh request ini adalah nama file yang ingin dihapus. Server lalu akan melakukan pengecekan apakah file tersebut tersedia, apabila ditemukan, maka akan dilakukan penghapusan dari sistem penyimpanan. Setelah proses penghapusan berhasil, server akan mengirimkan respon dengan status OK beserta pesan konfirmasi. Namun, apabila terjadi sebuah kesalahan, seperti file tidak ditemukan atau kesalahan lain, server akan memberikan respons berupa status ERROR dan pesan kesalahan.

3. Penjelasan Modifikasi di client dan server

- Modifikasi file_client_cli.py
 - Menambahkan method remote_upload() dan method remote_delete()
 - Method remote_upload() akan bertanggung jawab untuk mengunggah file ke server. Dalam method ini akan ditambahkan baris kode untuk encode menjadi base64 sesuai dengan ketentuan soal. Selanjutnya, string perintah/request "UPLOAD" akan dikonstruksi dan dikirim ke server menggunakan method send_command(). Sementara itu, method remote_delete() digunakan untuk menghapus file yang ada pada server. Fungsi ini membentuk sebuah string perintah DELETE yang diikuti dengan nama file, kemudian mengirimkan perintah tersebut ke server dengan menggunakan send_command().
 - Menambahkan CLI untuk client user
 - Modifikasi ini dilakukan pada method main di mana nantinya user bisa mengetikkan langsung message yang akan dikirim, seperti LIST, GET, UPLOAD, dan DELETE. Selanjutnya, akan dilakukan pengecekan dan untuk setiap perintah message, akan memanggil method remote yang akan diteruskan ke file protocol dan server.

- Modifikasi file_interface.py
 - Menambahkan method upload() dan delete()

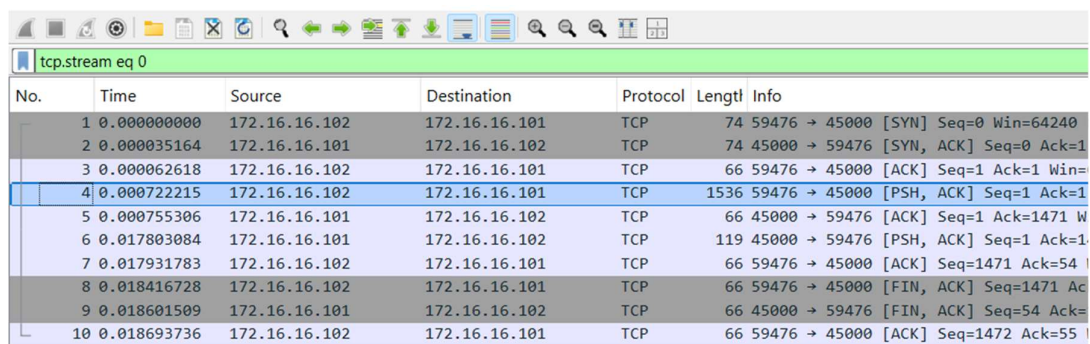
Di dalam file interface, method upload() dan delete() akan berfungsi sebagai handler untuk client yang akan mengirimkan request ke server. Method upload() akan menerima parameter berupa list params dari client, yaitu nama file dan isi file dalam base64. Nantinya, method ini akan melakukan decode dan dilanjutkan untuk write isi file nya. Sementara itu, method delete() akan menangani request untuk delete. Method ini juga menerima params yaitu nama file yang akan dihapus. Di method ini akan memanfaatkan os.remove untuk menghapus file di server.

- Modifikasi lain
 - Menyesuaikan ip untuk client ke server
 - Menyesuaikan port di mana server akan berjalan

4. Capture Request UPLOAD dan DELETE

Untuk melakukan demonstrasi request UPLOAD dan DELETE, akan dilakukan pada file PROTOKOL.txt yang terdapat di client. File tersebut akan dikirim ke server dengan perintah UPLOAD dan selanjutnya akan dihapus dengan perintah DELETE. Client akan berjalan di mesin-2 dan server di mesin-1. Berikut merupakan tangkapan traffic dari wireshark selama upload file dan delete file

- UPLOAD File



| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|---------------|---------------|----------|--------|--------------------------------------|
| 1 | 0.000000000 | 172.16.16.102 | 172.16.16.101 | TCP | 74 | 59476 → 45000 [SYN] Seq=0 Win=64240 |
| 2 | 0.000035164 | 172.16.16.101 | 172.16.16.102 | TCP | 74 | 45000 → 59476 [SYN, ACK] Seq=0 Ack=1 |
| 3 | 0.000062618 | 172.16.16.102 | 172.16.16.101 | TCP | 66 | 59476 → 45000 [ACK] Seq=1 Ack=1 Win= |
| 4 | 0.000722215 | 172.16.16.102 | 172.16.16.101 | TCP | 1536 | 59476 → 45000 [PSH, ACK] Seq=1 Ack=1 |
| 5 | 0.000755306 | 172.16.16.101 | 172.16.16.102 | TCP | 66 | 45000 → 59476 [ACK] Seq=1 Ack=1471 W |
| 6 | 0.017803084 | 172.16.16.101 | 172.16.16.102 | TCP | 119 | 45000 → 59476 [PSH, ACK] Seq=1 Ack=1 |
| 7 | 0.017931783 | 172.16.16.102 | 172.16.16.101 | TCP | 66 | 59476 → 45000 [ACK] Seq=1471 Ack=54 |
| 8 | 0.018416728 | 172.16.16.102 | 172.16.16.101 | TCP | 66 | 59476 → 45000 [FIN, ACK] Seq=1471 Ac |
| 9 | 0.018601509 | 172.16.16.101 | 172.16.16.102 | TCP | 66 | 45000 → 59476 [FIN, ACK] Seq=54 Ack= |
| 10 | 0.018693736 | 172.16.16.102 | 172.16.16.101 | TCP | 66 | 59476 → 45000 [ACK] Seq=1472 Ack=55 |

Gambar tersebut menunjukkan urutan komunikasi TCP antara client (172.16.16.102) dan server (172.16.16.101) saat proses upload file, yaitu ketika client mengirimkan message “UPLOAD PROTOKOL.txt” kepada server. Pertama, client dan server akan melakukan handshake tiga arah. Ketika koneksi sudah terhubung, maka client akan mulai mengirimkan file ke server, ditandai dengan adanya pengiriman flag [PSH, ACK] yang menandai server akan segera memproses data tersebut. Server lalu

mengirimkan flag [ACK] untuk menandai bahwa server sudah menerima paket tersebut. Setelah data selesai dikirim, client mengirimkan [FIN, ACK] di paket 8 untuk mengakhiri sesi, lalu server merespons dengan [ACK] di paket 9 dan akhirnya menutup koneksi dengan [FIN, ACK] di paket 10. Berikut merupakan inspect pada TCP stream yang ada di wireshark.



- DELETE File

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|---------------|---------------|----------|--------|---|
| 1 | 0.000000000 | 172.16.16.102 | 172.16.16.101 | TCP | 74 | 57564 → 45000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 |
| 2 | 0.000027637 | 172.16.16.101 | 172.16.16.102 | TCP | 74 | 45000 → 57564 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 |
| 3 | 0.000064004 | 172.16.16.102 | 172.16.16.101 | TCP | 66 | 57564 → 45000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 |
| 4 | 0.000596081 | 172.16.16.102 | 172.16.16.101 | TCP | 87 | 57564 → 45000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=87 |
| 5 | 0.000615477 | 172.16.16.101 | 172.16.16.102 | TCP | 66 | 45000 → 57564 [ACK] Seq=1 Ack=22 Win=65152 Len=0 |
| 6 | 0.016528074 | 172.16.16.101 | 172.16.16.102 | TCP | 136 | 45000 → 57564 [PSH, ACK] Seq=1 Ack=22 Win=65152 Len=136 |
| 7 | 0.016683909 | 172.16.16.102 | 172.16.16.101 | TCP | 66 | 57564 → 45000 [ACK] Seq=22 Ack=71 Win=64256 Len=0 |
| 8 | 0.017659119 | 172.16.16.102 | 172.16.16.101 | TCP | 66 | 57564 → 45000 [FIN, ACK] Seq=22 Ack=71 Win=64256 Len=0 |
| 9 | 0.017798460 | 172.16.16.101 | 172.16.16.102 | TCP | 66 | 45000 → 57564 [FIN, ACK] Seq=71 Ack=23 Win=65152 Len=0 |
| 10 | 0.017842798 | 172.16.16.102 | 172.16.16.101 | TCP | 66 | 57564 → 45000 [ACK] Seq=23 Ack=72 Win=64256 Len=0 |

Gambar tersebut menunjukkan urutan komunikasi TCP antara client (172.16.16.102) dan server (172.16.16.101) saat proses delete file, yaitu ketika client mengirimkan message “DELETE PROTOKOL.txt” kepada server. Sama seperti proses TCP sebelumnya, akan dilakukan handshake antara client dan server. Setelah berhasil connect, maka client akan mengirim request ke server dengan [PSH, ACK] dan akan direspon oleh server dengan [ACK] yang berarti request sudah diterima. Apabila request penghapusan sudah dilakukan, server gantian mengirimkan pesan status respons sukses atau gagal dari penghapusan file dengan [PSH, ACK] dan client pun

membalas dengan [ACK]. Setelah semua proses selesai, client mengirimkan [FIN, ACK] di paket 8 untuk mengakhiri sesi, lalu server merespons dengan [ACK] di paket 9 dan akhirnya menutup koneksi dengan [FIN, ACK] di paket 10. Berikut merupakan inspect pada TCP stream yang ada di wireshark.

