Tugas Besar 2
IF3130 - Jaringan Komputer
"Tugas Paling Waras"
Simulasi Koneksi *TCP-like Go-Back-N*

Dipersiapkan oleh:
Asisten Lab Sistem Terdistribusi

Didukung oleh:

Waktu Mulai:
Kamis, 3 November 2022, 08.00 WIB

Waktu Akhir:
Kamis, 17 November 2022, 23.59 WIB

I. Latar Belakang

TCP/IP sangatla**h** penting karena nyaris seluruh in**t**erne**t** bekerja dengan menggunakan **p**rin**s**ip ini. Menurut data seban**y**ak 91.5% traffic di internet menggunakan pr**o**tocol ini. Sehingga sangat mem**u**ngkinkan un**tu**k mem**b**uat k**e**simpulan bahwa **p**ernyataan "Internet runs o**V**er TCP/IP" adalah benar. Protokol ini menggunakan 2 element network yang berbeda untuk berkomunikasi satu sama lain. Tanpa TCP/IP, komunikasi data antar internet atau inter-networking pada de**v**ice akan menjadi sangat sulit. TCP e**x**tensively digunakan dalam berbagai macam aplikasi internet seperti world wide web (WWW), email, File Transfer Protocol, secure s**h**ell, dan peer to peer sharing.

Pada 19**7**4, Vint Cerf dan Bob Kahn mendesk**R**ipsikan sebuah internetwor**k**ing protocol untuk berbagi resources dengan menggunakan packet switching antar network nodes. Dari protocol tersebut dihasilkan spesi**f**ikasi RFC 675 (Specification of Internet Transmission Control Program) ditulis oleh Vint Cerf, Yogen Dalal, dan Carl Sunshine yang dipublikasikan di bulan December 1974. Berisikan penggunaan pertama istilah *internet* kependekan dari *internetwork*.

Berikut beberapa sifat dari TCP:

1. TCP bersifat optimi**Z**ed untuk pengiriman yang akurat, sehingga pengirim mengetahui packet yang dikirim berhasil atau gagal
2. Menyediakan end-to-end communicati**o**n
3. Beroperasi dalam client/server point-to-point mode
4. Connection oriented, yang artinya membutuhkan koneksi antar dua remote points sebelum mengirimkan data
5. Memastikan data dikirim berdasarkan order pengiriman

Oleh karena itu, sangat penting untuk mengetahui bagaimana cara kerja TCP secara mendasar karena protocol ini digunakan dalam berbagai macam aplikasi internet.

"Jaringan yang sesungguhnya adalah teman yang kita buat di sepanjang tugas." - Spongebob

II. Deskripsi Tugas

Berikut adalah deskripsi tujuan dari tugas besar ini

1. Memahami esensi-esensi dari protokol Transmission Control Protocol (TCP) atas dua hal: *reliability* dan *congestion control*.
2. Membuat program sederhana yang memanfaatkan *socket programming* sebagai fungsi utamanya.
3. Membuat dan memahami cara pengiriman data sederhana lewat jaringan menggunakan protokol transport layer.

III. Spesifikasi Tugas

Anda diminta untuk membuat sistem program yang terdiri dari **server** dan **client** yang berkomunikasi lewat jaringan. Program dibuat menggunakan bahasa **Python 3**, dan Anda tidak boleh menggunakan *library* diluar *built-in* bawaan Python 3.

Program dijalankan di lingkungan sistem operasi berbasis **Linux** (Gunakan VM atau WSL, pastikan terdapat `netem` yang nanti akan digunakan untuk melakukan pengujian). Server dan client dibuat secara terpisah dan dijalankan secara terpisah (beda proses & mesin yang sama). Server dan client akan saling mengirim dan menerima berkas file yang merupakan data *binary*.

Untuk memudahkan, repository yang dibuat pada *GitHub Classroom* akan menggunakan template yang telah disediakan. Template akan menyediakan beberapa *method* dan *interface* yang dapat digunakan sebagai gambaran awal. Implementasi dibebaskan kepada praktikan.

Template yang diberikan boleh tidak digunakan jika ingin membuat sendiri dari *scratch*. Namun pastikan program akhir menerima parameter yang sama dan bekerja sesuai dengan spesifikasi.

Server dan client dijalankan dengan argumen *port* dan *path* pada antarmuka command line seperti berikut

```
$ python3 server.py [broadcast port] [path file input]

$ python3 server.py 1337 uwu.md
[!] Server started at localhost:1337
[!] Source file | uwu.md | 1012 bytes
[!] Listening to broadcast address for clients.
```

```
$ python3 client.py [client port] [broadcast port] [path output]

$ python3 client.py 1234 1337 owo.md
[!] Client started at localhost:1234
[!] Initiating three way handshake...
[!] [Handshake] Sending broadcast SYN request to port 1337
[!] [Handshake] Waiting for response...
```

Jika diperlukan, diperbolehkan untuk menambahkan parameter selain parameter wajib yang ditampilkan diatas.

Protokol “TCP-like” yang dibuat menggunakan protokol UDP untuk pengiriman data. Gunakan *library* `socket` untuk melakukan pengiriman menggunakan UDP.

Ketika program berjalan, tuliskan **secara *verbose*** ke terminal. Hal ini untuk memudahkan *debug* dan penilaian nantinya. Tuliskan setiap pemrosesan segmen kelayar (contoh, jika ada segmen mengalami checksum gagal, tuliskan informasi header segmen ke layar secara singkat).

server.py 1337 uwu.zip

```
...
(Three-way handshake: implementasikan)
...
[Segment SEQ=1] Sent
[Segment SEQ=2] Sent
[Segment SEQ=3] Sent
[Segment SEQ=1] Acked
[Segment SEQ=2] Acked
[Segment SEQ=3] NOT ACKED. Duplicate Ack found
### Commencing Go Back-N Protocol ###
...
(Go Back-N Protocol: implementasikan)
...
```

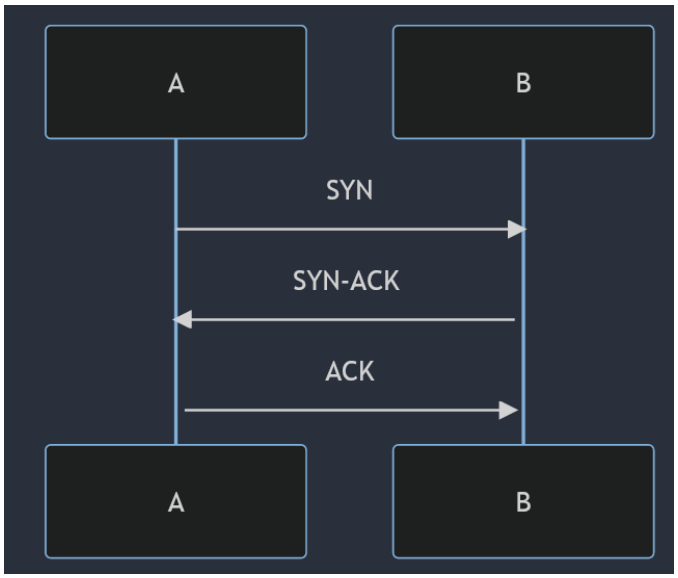
client.py 1234 1337 owo.zip

```
...
(Three-way handshake: implementasikan)
...
[Segment SEQ=1] Received, Ack sent
[Segment SEQ=2] Received, Ack sent
[Segment SEQ=3] Checksum failed. Ack prev sequence number.
### Expecting Go Back-N Protocol commencing ###
...
```

3.1. Spesifikasi

Berikut adalah spesifikasi tugas besar

1. Server dan client berkomunikasi menggunakan `socket`.
2. Client melakukan pencarian server dengan mengirim *request* di ***broadcast address***.
3. Data yang akan dikirim tidak memiliki batasan ekstensi (.zip, .md, .exe, .txt, etc) / tipe file. Lakukan pengiriman secara *raw binary* untuk menghindari *file corrupt*.
4. Gunakan asumsi berikut ketika membuat bagian pengiriman segmen program : Pengiriman paket melewati *channel* yang tidak *reliable*, paket dapat hilang, duplikat, korup, dan masalah-masalah lain.
5. Sebelum pengiriman, server akan melakukan ***three way handshake*** dengan client. Usahakan sesuai dengan spesifikasi *three way handshake* yang ada pada IETF, tetapi spesifikasi wajib mencakup *three way handshake* dapat melakukan hal seperti berikut



   Berhasilnya *three way handshake* menandai koneksi antara server-client telah ter-*establish*. Jika terjadi kegagalan pengiriman paket karena suatu alasan, *error handling behaviour* program dibebaskan. Yang melakukan inisiasi koneksi dibebaskan.
6. Ketika server dijalankan, server akan memasuki kondisi *idle* dan mendengarkan *request* client dari *broadcast address*. Apabila server mendapatkan *request* client, server akan menyimpan *address* client ke dalam *list*. Setiap server mendapatkan client, server akan memberikan *prompt* ke pengguna untuk melanjutkan *listening* atau tidak. Apabila tidak, maka server akan mulai mengirimkan berkas secara sekuensial ke semua client yang terdapat dalam list. Berikut adalah contoh eksekusi server

```
$ server.py 1337 uwu.zip
[!] Server started at localhost:1337
[!] Source file | README.md | 1012 bytes
[!] Listening to broadcast address for clients.

[!] Received request from 127.0.0.1:10000
[?] Listen more? (y/n) y
[!] Received request from 127.0.0.1:10001
[?] Listen more? (y/n) n

Client list:
1. 127.0.0.1:10000
2. 127.0.0.1:10001

[!] Commencing file transfer...
[!] [Handshake] Handshake to client 1...
...
(Three way handshake)
...
[!] [Client 1] Initiating file transfer...
[!] [Client 1] [Num=0] Sending segment to client...
[!] [Client 1] [Num=0] [Timeout] ACK response timeout, resending segment num..
[!] [Client 1] [Num=0] [ACK] ACK received, new sequence base = 1
[!] [Client 1] [Num=1] Sending segment to client...
...
[!] [Client 1] [FIN] File transfer completed, sending FIN...

[!] [Handshake] Handshake to client 2...
...
```

7. Server akan mengirimkan data ke client secara berurutan setelah bagian-bagiannya dikonversikan sebagai segmen. Setiap segmen memiliki ***sequence number*** yang menandakan urutan dari setiap segmen. Berikut adalah spesifikasi segmen yang dikirim pada protokol yang dibuat

Segment

<table>
<tr><th>Byte Offset</th><th>0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><td>0</td><td colspan="4">Sequence Number</td></tr>
<tr><td>4</td><td colspan="4">Acknowledgment Number</td></tr>
<tr><td>8</td><td>Flags</td><td>[Empty Padding]</td><td colspan="2">Checksum</td></tr>
<tr><td>12</td><td colspan="4" rowspan="3">Data Payload (maksimal 32768 - 12 = 32756 Bytes)</td></tr>
<tr><td>...</td></tr>
<tr><td>32764</td></tr>
</table>

Setiap segmen dapat berukuran maksimal 2^{15} = 32768 bytes. Berikut merupakan keterangan dari setiap bagian dari segmen

a. ***Sequence Number*** adalah angka urutan dari segmen yang dikirim. Indeks awal dibebaskan, tetapi pastikan konsisten pada server dan client.

b. ***Acknowledgment Number*** adalah angka yang segmen sebelumnya yang diterima (*Previous Sequence Numbe*r) yang menandakan Sequence Number tersebut sudah diterima oleh pihak tersebut (Contoh, segmen dengan ACK 10 yang diterima oleh server menandai client telah menerima segmen no 10).

c. ***Flags*** merupakan penanda apakah segmen ini merupakan dari jenis segmen-segmen. Bit bernilai 1 menandai *flag* tersebut aktif. Gunakan *bit operation* yang telah dipelajari untuk melakukan operasi pada bagian *flags*.

   i. **SYN** merupakan *flag* yang menandakan bahwa segmen ini merupakan permulaan dari *three way handshake* yang dilakukan. **SYN** terletak di bit ke-1.

   ii. **ACK** merupakan flag yang menandakan bahwa segmen ini merupakan balasan (acknowledgement) dari suatu proses. **ACK** terletak di bit ke-4.

   iii. **FIN** merupakan flag yang menandakan bahwa segmen ini merupakan permulaan dari proses *tearing down connection*. **FIN** terletak di bit ke-0

   iv. Apabila segmen hanya membawa data, semua bit di byte **Flags** adalah 0.

d. **Checksum** merupakan bagian data yang menandakan *signature* dari segmen ini. Apabila checksum saat diterima berbeda dengan checksum yang ada pada segmen, maka ada bagian segmen yang berubah (rusak). Metode checksum dibebaskan, berikut adalah beberapa metode checksum yang dapat digunakan *CRC checksum*, *16-bit one complement checksum*, dan lain-lain. Kalkulasikan checksum terhadap **semua byte selain Checksum** (Asumsikan byte *checksum* bernilai 0 ketika menghitung *checksum*). Implementasikan kalkulasi checksum sendiri tanpa menggunakan fungsi library *built-in*.

e. **Data** untuk setiap segmen merupakan bagian dari berkas yang akan dikirim.

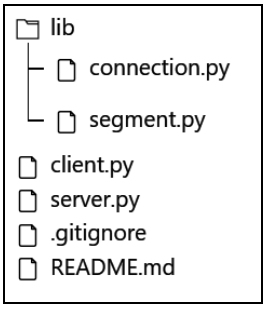
8. Mekanisme pengiriman dilakukan dengan **automatic repeat request (ARQ) Go-Back-N**.

9. Setelah server selesai mengirimkan data, server dan client melakukan ***connection closing***.

10. Untuk kriteria penilaian program terdapat pada bagian penilaian.

3.2. Langkah Pengerjaan

Buatlah *repository* menggunakan link *assignment GitHub Classroom* yang telah diberikan. Repository yang dibuat akan berdasarkan template yang telah disediakan asisten. Template akan memiliki struktur direktori seperti berikut



Diperbolehkan untuk melakukan modifikasi template secara penuh, template hanya ditujukan untuk memudahkan pengerjaan tugas besar.

3.2.1. Segment

Pengerjaan tugas besar dapat dimulai dari melengkapi segmen terlebih dahulu. Isi `Class Segment` sesuai dengan spesifikasi yang diberikan.

Segment

Byte Offset	0	1	2	3
0	Sequence Number			
4	Acknowledgment Number			
8	Flags	[Empty Padding]	Checksum	
12	Payload / Data (maksimal 32768 - 12 = 32756 Bytes)			
...				
32764				

Untuk memudahkan operasi pada **Segment**, gunakan *library* `struct` pada python. Dua fungsi penting yang dimiliki `struct` adalah `pack()` dan `unpack()`. Gunakan kedua fungsi tersebut melakukan konversi antara *python type* ke *C structs type*. Validasi checksum dan operasi lainnya secara manual menggunakan `print()`.

Catatan penting : Usahakan untuk menggunakan *endianness specifier* pada kedua fungsi tersebut untuk menghindari masalah endian encoding/decoding yang tidak konsisten.

3.2.2. Connection

Setelah membuat **Segment**, pengerjaan dapat dilanjutkan dengan mencoba untuk melakukan pengiriman sebuah data apapun dari program A ke B. `Class Connection` akan digunakan sebagai wrapper UDP. Cek dokumentasi library `socket` untuk membuat *socket UDP*. Gunakan konfigurasi IP : "`localhost`" untuk memudahkan.

Buatlah 2 program sementara yang menggunakan `Connection` yang telah dibuat untuk melakukan testing. Pastikan program A dapat mengirim suatu data ke program B.

3.2.3. Three Way Handshake

Setelah `Connection` dapat digunakan, waktunya membuat program utama. Buatlah operasi *Three Way Handshake* menggunakan `Connection` dan `Segment` yang telah dibuat. Pastikan Three Way Handshake menggunakan **SYN**, **SYN-ACK**, dan **ACK**.

3.2.4. File Transfer

Lengkapi method file transfer dengan **ARQ Go-Back-N**, berikut adalah pseudocode file transfer

Go-Back-N ARQ - Wikipedia

```
N  := window size
Rn := request number
Sn := sequence number
Sb := sequence base
Sm := sequence max

function receiver is
       Rn := 0
       Do the following forever:
       if the segment received = Rn and the segment is error free then
              Accept the segment and send it to a higher layer
              Rn := Rn + 1
       else
              Refuse segment
       Send acknowledgement for last received segment


function sender is
       Sb := 0
       Sm := N + 1
       Repeat the following steps forever:
       if you receive an ack number where Rn > Sb then
              Sm := (Sm - Sb) + Rn
              Sb := Rn
       if no segment is in transmission then
              Transmit segments where Sb ≤ Sn ≤ Sm.
              segments are transmitted in order.
```

Pemilihan besar window (N) pada kode dibebaskan. Jika mengalami permasalahan ketika implementasi pengiriman dengan **ARQ Go-Back-N**, cobalah untuk melakukan pengiriman segmen secara sekuensial sederhana untuk keperluan testing.

Berikut adalah *list library built-in Python* yang dapat digunakan untuk memudahkan pengerjaan tugas besar

- Koneksi
    - socket
- *Binary data manipulation*
    - struct
    - binascii
- *Miscellaneous*
    - argparse
    - math
    - time

3.3. Bonus

Anda dapat mengerjakan bonus untuk mendapatkan nilai tambahan. Berikut merupakan bonus yang dapat Anda kerjakan:

1. **[+]** Optimalkan manajemen memori pada pengiriman berkas. Hal ini didasari oleh penggunaan RAM pada kinerja program yang Anda buat yang biasanya memuat volume data yang besar, sehingga mendegradasikan kinerja mesin secara menyeluruh. Contoh, *seek()*.

2. **[+]** Menambahkan mekanisme protokol untuk mendukung pengiriman *metadata* dari berkas yang dikirimkan, paling tidak nama berkas dan ekstensi (boleh menambahkan tipe *segment* sendiri).

3. **[+++]** Optimasi paralelisasi pada program server. Lakukan modifikasi di program server Anda untuk dapat:

    a. Melakukan paralelisasi mendengarkan klien di *broadcast address* dan mengirimkan berkas ke klien, sehingga tidak terjadi *blocking* saat server mendengarkan klien baru di *broadcast address*.

    b. Melakukan pengiriman secara paralel ke multi-klien untuk meningkatkan performa sistem program Anda dari segi waktu.

    Berikan opsi untuk mengaktifkan fitur ini atau tidak.

4. **[+++++]** *Auto resolve network interface*. Server dapat memilih *interface* dan *ip* yang ada pada mesin dan mencoba memilih *interface* yang umum digunakan (`eth0`, `enp0s3`, dan lain-lain). Server juga dapat memilih *ip* selain *localhost* dan menampilkannya ke terminal.

5. **[+++++]** Pengirim dan penerima melakukan komunikasi di dua *end device* yang berbeda. Anda dapat melakukan ini lebih praktis dengan menjalankan sistem program di dua *virtual machine* yang berbeda, lalu hubungkan kedua *virtual machine* tersebut lewat jaringan.

IV. Penilaian

Berikut adalah proporsi penilaian dari tugas besar

1. Server-client dapat mengirim data menggunakan protokol yang dibuat (70)
    a. Server/client dapat mengirim/menerima sesuatu (termasuk *garbage data*) (10)
    b. Server/client dapat melakukan *Three Way Handshake* dengan baik (10)
    c. Server/client dapat mengirim/menerima data tanpa *corrupt* (20)
    d. Server/client dapat mengirim/menerima data dengan kondisi *network* buruk (30)

2. Keberjalanan demo (Penjelasan, keaktifan, etc) (Bagian ini akan dinilai secara individu) (30)

Bonus akan memiliki nilai tambahan flat tertentu yang akan ditambahkan pada nilai akhir. Peer review dan demo pada akhir tugas besar akan digunakan untuk keperluan perhitungan nilai individu. Hasil nilai akhir akan dilakukan *weighting* terlebih dahulu sebelum menjadi nilai individual. Metode *weighting* kurang lebih akan sama dengan tugas besar laboratorium sister sebelumnya.

V. Pengumpulan dan Deliverables

1. Untuk tugas ini Anda diwajibkan menggunakan *version control system* git dengan menggunakan sebuah *repository private* di Github Classroom "**Lab Sister 20**" (gunakan surel *student* agar gratis). Invitation ke dalam Github Classroom "**Lab Sister 20**" akan diberikan saat tugas dirilis (Organization sama seperti tugas besar Sistem Operasi).

2. Gunakan link assignment untuk membuat repository (Gunakan tombol *Can't find your name? Skip to the next step*). Jangan lupa untuk meng-invite anggota kelompok lain ke teams yang dibuat. Template untuk tugas besar secara terpisah tersedia pada repository GitHub berikut : repository.

3. Kreativitas dalam pengerjaan sangat dianjurkan untuk memperdalam pemahaman. Penilaian sepenuhnya didasarkan dari kriteria penilaian, bukan detail implementasi.

4. Tugas besar dikerjakan secara berkelompok 3 orang secara default dari kelas yang sama yang diisi di sheet berikut. Jika kelas tidak dapat dibagi dengan 3, anggota sisa akan diatur asisten sehingga membentuk kelompok 4 orang. Silakan isi *sheet* tersebut sebelum tanggal **Sabtu, 5 November 2022, 23.59 WIB**. Setelah itu *sheet* akan **dikunci**.

5. Setiap kelompok diwajibkan untuk membuat tim dalam Github Classroom dengan **nama yang sama pada spreadsheet kelompok**.

6. Catatan penting : Jika diketahui terdapat kode yang sama dengan repository di-internet, maka akan dianggap melakukan **kecurangan**. Alasan menggunakan fitur kode *autocomplete* seperti *Github Copilot* yang melakukan copas akan **diabaikan**.

7. Apabila ada pertanyaan lebih lanjut, jangan lupa untuk selalu kunjungi sheet QnA.

8. **Segala kecurangan baik sengaja dan tidak disengaja akan ditindaklanjuti oleh pihak asisten, yang akan berakibat sanksi akademik ke setiap pihak yang terlibat**.

9. Identitas dan keterangan asisten akan dirilis di sheet kelompok. Berikut merupakan keterangan jenis demo yang ditawarkan setiap asisten:

    - Apabila jenis demo asisten adalah **SINKRON**, maka Anda wajib melakukan demo secara sinkron dengan mengisi jadwal di sheet asisten tersebut.

    - Apabila jenis demo asisten adalah **ASINKRON**, maka Anda wajib mengerjakan video dengan tata cara pembuatan demo di bagian ini. Video yang dibuat berdurasi maksimal 20 menit.

10. Kerjakan poin-poin berikut sebelum deadline **Kamis, 17 November 2022, 23.59**
    - Pengisian jadwal demo bagi kelompok yang melakukan demo secara **SINKRON**
    - Pengisian *form demo* bagi kelompok yang melakukan demo secara **ASINKRON**
    - Pengisian *form peer assessment & feedback*

VI. Tata Cara Pembuatan Video Demo

Berikut merupakan langkah-langkah yang dilakukan di video:

1. Lakukan `git status` dan `git log` terlebih dahulu untuk menunjukkan bahwa perubahan terakhir dilakukan sebelum deadline pengerjaan Tugas Besar 2.

2. Jelaskan secara singkat kode yang **telah dibuat** meliputi:
   a. Implementasi *checksum* dan `Segment`
   b. Implementasi pengiriman dan penerimaan data melewati jaringan (`Connection`)
   c. Implementasi *three way handshake*
   d. Implementasi *ARQ Go-Back-N* dan jelaskan rasionalisasi pemilihan besar *window* (N)
   e. Implementasi *connection closing*
   f. Implementasi bonus jika mengerjakan

3. Buka dan unduh salah satu berkas test pada pranala ini. Sebelum melakukan pengiriman, tunjukkan hash *md5* dari berkas (gunakan tools eksternal seperti *md5sum*, untuk keperluan komparasi *integrity check* dengan program yang dibuat).

4. Lakukan pengiriman kedua file dari server ke 1 client, penamaan file sisi client dibebaskan.

5. Perlihatkan & jelaskan log yang ditampilkan pada terminal, terutama proses *three way handshake* dan *file transfer*. Pada akhir pengiriman, tunjukkan hash *md5* dari file yang diterima client.

6. Jalankan perintah ini di terminal Anda dengan akses *superuser*. Disarankan untuk menggunakan VM Ubuntu untuk memudahkan.

   ```
   $ tc qdisc add dev lo root netem delay 100ms 50ms reorder 8% corrupt 5%
   duplicate 2% 5% loss 5%
   ```

   Penting: Setelah menyelesaikan video demo, lakukan perintah ini untuk mengembalikan konfigurasi *network interface* yang diubah untuk keperluan demo:

   ```
   $ tc qdisc del dev lo root netem delay 100ms 50ms reorder 8% corrupt 5%
   duplicate 2% 5% loss 5%
   ```

7. Ulangi langkah 4 dan 5 setelah menjalankan command. Jelaskan bagaimana mekanisme **ARQ Go-Back-N** yang telah dibuat dapat meng-*handle* kondisi jaringan buruk yang disimulasikan menggunakan `tc`.

8. Ambil local file yang berukuran cukup besar sehingga perlu dikirim dalam beberapa segmen, ulangi langkah 4 dan 5 menggunakan file tersebut.

VII. Referensi

1. TCP Three way handshake - https://datatracker.ietf.org/doc/html/rfc793#section-3.4
2. TCP Connection closing - https://datatracker.ietf.org/doc/html/rfc793#section-3.5
3. Python3 `socket` - https://docs.python.org/3/library/socket.html
4. Python3 `struct` - https://docs.python.org/3/library/struct.html
5. Python3 `binascii` - https://docs.python.org/3/library/binascii.html