

Tugas Besar 1
IF3130 - Jaringan Komputer
"Batu TCP"

Implementasi protokol *TCP-like Go-Back-N*

Revisi 3 - 28/11/2023 19:10

Dipersiapkan oleh:
Asisten Lab Sistem Terdistribusi

Didukung oleh:



Waktu Mulai:
Selasa, 7 November 2023, 12.00 WIB

Waktu Akhir:
Selasa, 28 November 2023, 23.59 WIB

Daftar Revisi

1 - 23/11/2023 14:10 - Penambahan informasi terkait bonus permainan online dan referensi tugas besar tahun lalu

2 - 23/11/2023 16:55 - Penekanan penggunaan *library* dan batasan *socket*

3 - 28/11/2023 19:10 - Penjelasan tambahan untuk penilaian bonus

I. Latar Belakang

Kamu telah selesai mengatur konfigurasi router secara manual. Namun saat kamu membuka HP-mu untuk segera menonton stream pekola, kamu terkejut karena tiba-tiba HP-mu tidak bisa menyala.

"Lah kok gitu. Kalo gini, gimana caranya aku nonton stream PekolaaaAAAAAAA"

Biboo terlihat tertawa terbahak-bahak melihat kesialanmu. Kamu pun menghampirinya dengan putus asa.



Biboo yang mentertawakanmu

"Anu.. Toko HP terdekat ada di mana ya?"

Biboo makin terbahak-bahak mendengar perkataanmu. "Di dunia primitif ini kamu nanyain toko HP? Nggak salah nih?"

"Terus.. gimana cara keluar dari dunia ini?" ucapmu dengan lemas.

"Enak aja! Selesain dulu tugasmu sebelum keluar".

"TIDAKKKKK!!!!" Kamu pun berlari sekencang-kencangnya mencoba mencari portal ke luar. Namun, yang kamu lihat hanyalah hutan primitif tanpa ada tanda-tanda peradaban. Perlahan-lahan kamu pun merasa lelah. Semakin lelah dan... kamu tergeletak di tanah.

"Aduh.. kepalaku pusing" ucapmu dengan lemas. "Bondowoso.. Roro.. 1000 startup.. OH IYA! AKU BELUM MEMBUAT FITUR BARU UNTUK RORO YANG PAKE BAHASA JAVA!!"

"Kamu sudah bangun?" kamu mendengar suara seseorang di depanmu.

"Hah? Sangku? Aku di mana??" Kamu terkejut karena orang tersebut adalah Sangku, seorang jenius dari anime yang pernah kamu tonton.



Ishigami Sangku

"Kamu terlihat tergeletak di tanah di tanah berjarak 724 meter ke arah utara dari sini. Lalu aku suruh Kokaku membawamu ke sini"

"Huft.. Sepertinya tadi aku mimpi buruk. Terima kasih, Sangku! Kokaku! Ngomong-ngomong, apakah kamu bisa bikin HP? aku ingin menonton stream Pekola, tapi HP-ku rusak"

"Kebetulan sekarang kami mau bikin HP. Jadi, ayo bantu kami" Sangku mengeluarkan kertas berisi diagram roadmap untuk membuat HP. Sebagai anak Informatika, tentunya kamu merasa percaya diri dengan diagram karena kamu sudah mempelajari berbagai jenis diagram mulai dari **data flow diagram**, **sequence diagram**, sampai **diagram BPMN**.

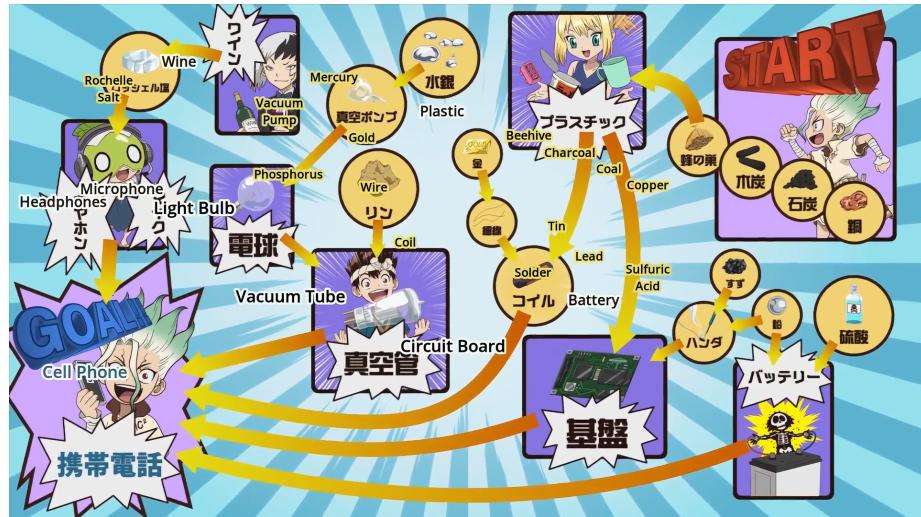


Diagram UML Roadmap untuk membuat HP Sangku

"Umm... Banyak juga ya.. Jadi kita harus cari semua bahan di sini?"

"Tepat sekali. Kalau kamu membantu kami, ada kemungkinan 1000000% HP ini akan jadi lebih cepat"

"Baiklah, aku akan membantu kalian. Demi Pekola!!!"

Singkat cerita, HP tersebut berhasil dibuat. Kamu terlihat sangat bahagia.

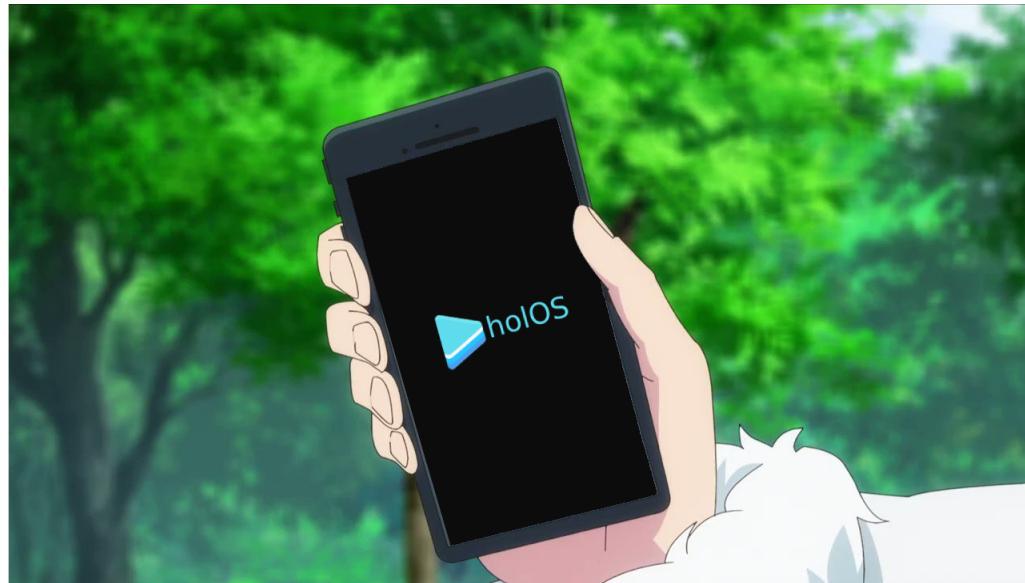
"Akhirnya aku bisa nonton stream Pekola"

"Jangan bahagia dulu. Ini kan baru hardware. Belum ada sistem operasi di dalam HP ini", ucap Sangku.

"Tenang saja. Di dimensi sebelumnya, aku sudah berhasil membuat sistem operasi bersama Docchi." Kamu pun menceritakan perjalananmu di dunia bit bersama Docchi.

"Wah keren juga kamu. Yaudah langsung pasang saja di sini"

Kamu memasang disk berisi HolOS yang sudah kamu modifikasi di sela-sela waktu luangmu agar dapat menampilkan GUI dan mendukung layar sentuh. Dengan ajaib, sistem operasi tersebut kompatibel dengan HP buatan Sangku.



holOS yang dipasang pada HP buatan Sangku

"Nah, sekarang tinggal connect aja ke router antar dimensi yang dijaga sama Biboo"

Kamu pun kembali ke tempat Biboo.

"Loh, kok ngga bisa connect ke internet? Internetnya mati ya?"

"HAHAHAHAHAHA. Di HP-mu itu kan belum ada **protokol pengiriman pesan**. Membuat protokol itu adalah **tugasmu yang kedua!**"

"TIIDDAAAAACKKKKKKK!!!!"

~~Begitulah kehidupanmu yang kurang beruntung (lagi) dan harus mengerjakan tugas ini. Selamat mengerjakan.~~

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** maupun **Windows**. Server dan client dibuat secara terpisah dan dijalankan secara terpisah (dijalankan sebagai proses yang berbeda, namun dijalankan di mesin yang sama). Server dan client akan saling mengirim dan menerima berkas file yang merupakan data *binary*.

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)
...
```

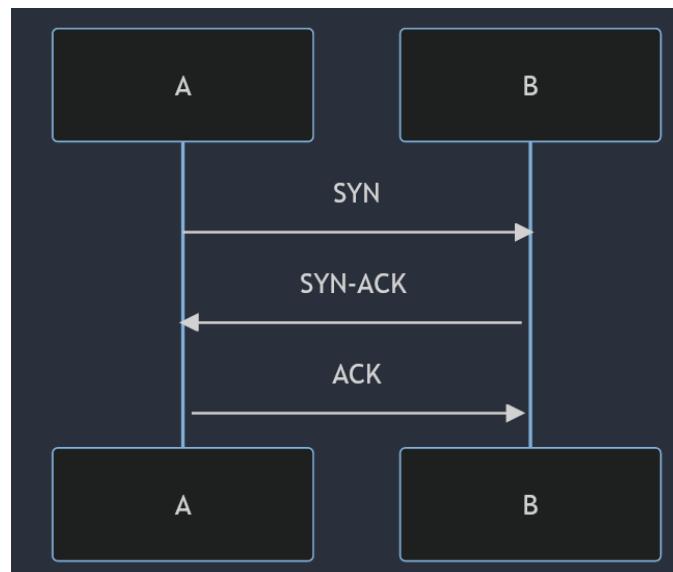
client.py 1234 1337 owo.zip

```
...
(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 dengan protokol UDP. Pastikan inisialisasi socket dengan `type=socket.SOCK_DGRAM`. **Pemakaian tipe socket `SOCK_STREAM (TCP)` akan mengakibatkan nilai tugas besar menjadi 0.**
2. Client melakukan pencarian server dengan mengirim *request* di *broadcast address*.
3. Data yang akan dikirim **tidak memiliki batasan tipe file / “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



Berikut adalah keterangan setiap proses:

- client akan mengirimkan *request segment* yang memiliki *flag SYN* kepada server. **SYN** merupakan *flag* yang menandakan bahwa segmen ini merupakan permulaan dari *three way handshake* yang dilakukan.
- server akan merespons *request client* dengan mengirimkan respons berupa segmen yang memiliki *flag SYN* dan **ACK**. **ACK** merupakan *flag* yang menandakan bahwa segmen ini merupakan balasan (*acknowledgement*) dari

suatu proses. **Flag ACK** ini digunakan untuk memberitahu client bahwa request segment yang client kirim berhasil diterima oleh server.

- client akan menerima respons dari server. Jika di segmen respon terdapat **flag SYN** dan **ACK**, client akan mengirimkan respon balik ke server yang berisi **flag ACK** untuk **flag SYN** yang dikirimkan oleh server.

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. Program yang melakukan inisiasi koneksi dibebaskan, boleh client terlebih dahulu atau server terlebih dahulu.

- 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

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

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

- Sequence Number** adalah angka urutan dari segmen yang dikirim. Indeks awal dibebaskan, tetapi pastikan konsisten pada server dan client.
- Acknowledgment Number** adalah angka yang segmen sebelumnya yang diterima (*Previous Sequence Number*) 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).
- Flags** merupakan penanda apakah segmen ini merupakan dari jenis segmen-segmen. Flags mungkin memiliki flag aktif lebih dari 1 (**SYN** dan **ACK** secara bersamaan). 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.
- 7. 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] [CLS] File transfer completed, initiating closing connection...
[!] [Client 1] [FIN] Sending FIN...

...
(Closing connection)
...

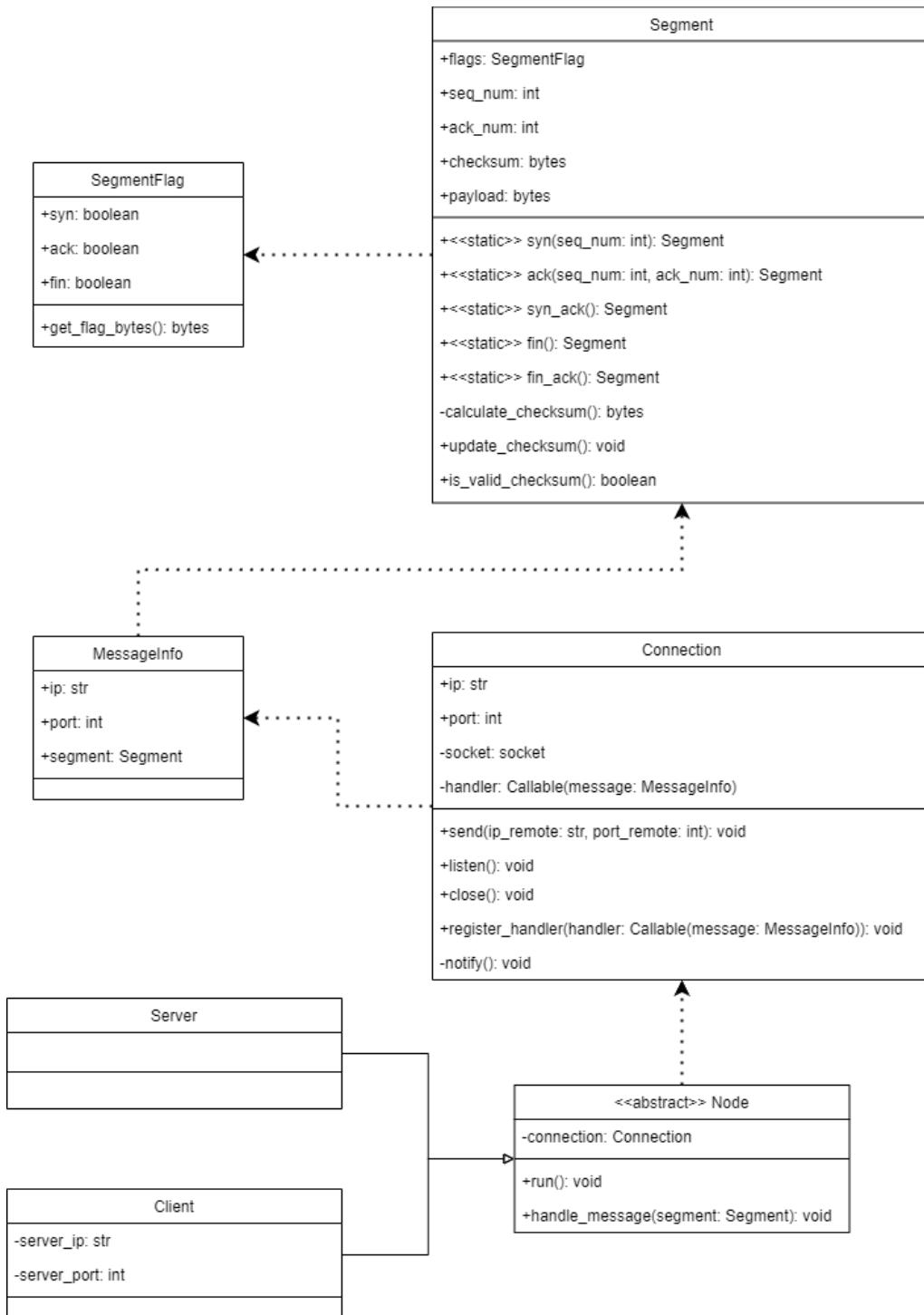
[!] [Handshake] Handshake to client 2...
...

```

8. Mekanisme pengiriman dilakukan dengan **automatic repeat request (ARQ) Go-Back-N**.
9. Setelah server selesai mengirimkan data, server dan client melakukan ***connection closing***. Untuk implementasi *connection closing* dapat mengikuti dokumentasi [berikut](#).
10. Untuk kriteria penilaian program terdapat pada bagian [penilaian](#).

3.2. Langkah Pengerjaan

Berikut contoh *class diagram* dari implementasi TCP dengan satu *client* dan satu *server*. Diagram (dan segmen ini) hanya berfungsi sebagai contoh, sehingga **tidak wajib diikuti**. Silahkan implementasikan sesuai kreativitas kalian (bisa lihat template kode tahun lalu juga jika diinginkan, dapat dilihat di bagian referensi).



3.2.1. Segment

Pengerjaan tugas besar dapat dimulai dari melengkapi segmen terlebih dahulu. Isi [Class Segment](#) sesuai dengan spesifikasi yang diberikan.

Segment

Bagian	Byte Offset	0	1	2	3		
Header (12 bytes)	0	Sequence Number					
	4	Acknowledgment Number					
	8	Flags	[Empty Padding]	Checksum			
Data	12	Data Payload (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.

Tips : Untuk *timeout*, dapat digunakan method [`socket.settimeout\(\)`](#) untuk memudahkan implementasi

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**. Tujuan utama dari *three way hand shake* adalah kedua buah *party* yang terlibat mengetahui adanya koneksi baru.

3.2.4. File Transfer

Algoritma ARQ (*Automatic Repeat Request*) Go-Back-N adalah sebagai berikut:

- Mekanisme **ARQ Go-Back-N** terdiri dari 2 mekanisme, yakni mekanisme mengirimkan data (**sender**), dan mekanisme menerima data (**receiver**)
- Untuk mekanisme menerima data (**receiver**), berikut langkah-langkahnya:
 1. Inisialisasi nilai nomor segmen (**Rn**) dengan 0. Nomor segmen menyatakan nomor urutan segmen yang akan dikirimkan.
 2. Lakukan pengulangan sampai pengiriman selesai. Di setiap iterasi:
 - a. Jika nomor segmen yang diterima sama dengan **Rn** dan segmen bebas error (*checksum* segmen yang diterima sudah sesuai), maka segmen tersebut akan diterima dan nilai **Rn** akan bertambah 1. Jika salah satu tidak terpenuhi , tolak segmen.
 - b. Kirimkan **ACK** untuk segmen yang baru saja diterima.
- Untuk mekanisme mengirimkan data (**sender**), berikut langkah-langkahnya:
 1. Inisialisasi nilai *sequence base* (**Sb**) dengan nilai 0. *Sequence base* menyatakan nomor segmen terendah yang bisa dikirimkan oleh program.
 2. Inisialisasi nilai *sequence max* (**Sm**) dengan *window size* (**N**) ditambah 1. *Sequence max* menyatakan nomor segmen tertinggi yang bisa dikirimkan oleh program. *Window size* adalah jumlah paket yang bisa dikirimkan tanpa menunggu **ACK** terlebih dahulu.
 3. Lakukan pengulangan sampai pengiriman selesai. Di setiap iterasi:
 - a. Jika **sender** menerima **ACK** dengan nomor segmen (**Rn**) lebih besar dari **Sb**, maka nilai **Sm** dan **Sb** akan diperbarui sehingga nilai **Sb** (batas bawah) akan sama dengan **Rn** (nomor segmen), dan nilai **Sm** akan menyesuaikan sehingga selisih **Sm** dan **Sb** akan tetap sama.
 - b. Kirimkan segmen jika dan hanya jika nomor segmen berada diantara **Sb** dan **Sm** secara inklusif ($Sb \leq Sn \leq Sm$) dan program sedang tidak mengirimkan segmen.

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

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

- Koneksi
 - socket
- *Binary data manipulation*
 - struct
 - binascii
- **Types or class helper**
 - dataclass
 - typing
 - abc
- *Miscellaneous*
 - argparse
 - threading
 - math
 - time

Jika membutuhkan *library* yang lain, silakan tanyakan pada QnA.

3.3. Bonus

Anda dapat mengerjakan bonus untuk mendapatkan **nilai tambahan** (emphasis pada *tambahan*). Mohon untuk mengerjakan bagian utama terlebih dahulu, pastikan fungsionalitas utama berjalan dengan baik sebelum mengerjakan bonus. Kerjakan bonus setelah bagian utama sudah sesuai dengan testing pada demo (jangan lupa tubes dokumen dan sabun gan :)). Nilai tambahan bonus akan **relatif lebih sedikit** dibandingkan program utama.

Bonus yang berbeda tidak harus dijalankan secara bersamaan, misalnya jika mengimplementasikan bonus ECC (no. 4), paralel (no. 3) dan game (no. 7), program tidak perlu di paralelisasi dan tidak perlu mengimplementasikan ECC untuk game, atau program juga tidak perlu mengimplementasikan ECC di paralel jika tidak memungkinkan, atau yang lainnya. **Jika tidak ada ketentuan khusus**, bonus boleh memiliki kondisi tertentu untuk menjalankannya, sebagai contoh implementasi ECC hanya akan berjalan untuk payload saja, tidak keseluruhan segmen. Jika ragu, Anda boleh konfirmasi ke asisten untuk penilaian bonus dan memastikan bonus yang Anda kerjakan sudah sesuai. Bonus akan dinilai *case-by-case basis*, jadi *feel free* untuk menyampaikan kondisi khusus dalam menjalankan bonus Anda beserta rasionalisasi atau alasan keterbatasan tersebut kepada asisten saat demo.

Berikut merupakan bonus yang dapat Anda kerjakan :

1. [1] Optimalkan manajemen memori pada pengiriman berkas, misalnya dengan *streaming processing* yang hanya menggunakan buffer berukuran kecil dan memanfaatkan *seek()*. Hal ini didasari oleh penggunaan RAM pada kinerja program yang Anda buat, biasanya memuat volume data yang besar, sehingga mendegradasikan kinerja mesin secara menyeluruh.
2. [1] Menambahkan mekanisme protokol untuk mendukung pengiriman *metadata* dari berkas yang dikirimkan, minimal nama berkas dan ekstensi (boleh menambahkan tipe *segment* sendiri).
3. [2] Implementasi file transfer secara dua arah (*peer to peer*), sehingga program terbuka untuk menerima maupun mengirim file. Metode dibebaskan, asalkan koneksi tetap *reliable* dengan protokol TCP yang benar.
4. [3] Optimasi paralelisasi pada program server. Lakukan modifikasi di program server Anda untuk dapat:
 - Melakukan paralelisasi mendengarkan klien di *broadcast address* dan mengirimkan berkas ke klien, sehingga tidak terjadi *blocking* saat server mendengarkan klien baru di *broadcast address*.
 - Melakukan pengiriman secara **paralel** ke multi-klien untuk meningkatkan performa sistem program Anda dari segi waktu.

Berikan opsi untuk mengaktifkan fitur ini atau tidak.

5. [4] Implementasi *error correcting codes*. Jika terdeteksi error yang memungkinkan untuk dikoreksi, lakukan koreksi pada segmen tersebut se bisa mungkin sebelum didrop. Metode ECC yang digunakan dibebaskan (Hamming, Reed-Solomon, etc). Pastikan checksum setelah melakukan koreksi ECC valid.
6. [4] Pengirim dan penerima melakukan komunikasi di dua *end device* (baik *physical* atau *virtual*) 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. **Pastikan untuk menampilkan cara konfigurasi bagian ini pada akhir demo jika mengerjakan.**
7. [10] Buatlah permainan online 2-player sederhana yang memanfaatkan protokol TCP ini seperti Card Game, Chess, atau Tic-Tac-Toe. Permainan dapat berjalan dengan lancar **bahkan jika terjadi masalah pada koneksi** (lakukan simulasi jaringan buruk seperti pada tata cara demo). Permainan harus berjalan **melalui protokol TCP yang telah dibuat**, bukan dengan cara lain (dilarang menggunakan file, unix socket, signal, message queue, shared memory, pipe, dan berbagai *approach* untuk Inter-Process Communication tanpa melalui network interface). **Implementasi permainan tidak boleh menggunakan library di luar dari built-in Python, Anda harus mengimplementasikan permainan sendiri dari awal.** Antarmuka dan interaksi permainan dibebaskan, boleh berbentuk CLI ataupun GUI (diperbolehkan menggunakan kakas seperti pyqt atau tkinter untuk membuat antarmuka GUI). Tidak ada perbedaan penilaian antara antarmuka/interaksi permainan yang sederhana maupun kompleks, selama permainan berhasil berjalan melalui protokol yang telah dibuat, Anda akan mendapatkan nilai maksimal bonus ini. Silahkan berkreasi sebebas mungkin untuk implementasi permainan dengan protokol yang telah dibuat.

IV. Penilaian

Berikut adalah proporsi penilaian dari tugas besar :

1. Server-client dapat mengirim data menggunakan protokol yang dibuat (bagian ini akan dinilai sebagai nilai kelompok oleh asisten) (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 dengan baik (checksum sama) (20)
 - d. Server/client dapat mengirim/menerima data dengan kondisi *network* buruk (30)
 - i. File yang dikirim kurang dari 1 MB (5)
 - ii. File yang dikirim 1 MB-10 MB (10)
 - iii. File yang dikirim lebih dari 10 MB (15)
2. Keberjalan demo (Penjelasan, keaktifan, etc) (Bagian ini akan dinilai secara individu oleh asisten) (30)

Program akan ditest dan dinilai minimal menggunakan Python 3.10. Pastikan penggunaan fungsi library sesuai dengan versi minimal tersebut. Usahakan hasil penggeraan akan berjalan dengan baik di platform tersebut. Jika ada requirement khusus, tuliskan cara eksekusi program dan requirement tersebut pada *README.md*.

Seluruh repository penggeraan tugas besar akan dicek satu-per-satu untuk pengecekan plagiarisme dan kondisi kelompok. Informasi yang diberikan pada *form peer review* juga akan digunakan sebagai sumber informasi kondisi kelompok, anggota kelompok yang tidak berkontribusi sama sekali akan ditindaklanjuti. Sistem penilaian akhir individu tugas besar akan mirip seperti tugas besar laboratorium sistem paralel dan terdistribusi sebelumnya, yaitu *weighting* dari nilai kelompok berdasarkan poin *peer review* dan poin dari asisten (*rule of thumb* dari *weighting* yang digunakan: **Nilai individu \leq kelompok**).

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**”.
2. Gunakan [link assignment](#) untuk membuat repository (Gunakan tombol *Can't find your name? Skip to the next step*).
3. Pengumpulan dilakukan dengan membuat **release** dengan tag **v1.0** pada repository yang telah kelompok Anda buat sebelum deadline. Jika terdapat revisi, tambahkan angka minor pada versi tag (**v1.1**, **v1.2**, ..., **v1.x**) Pastikan tag sesuai format.
Repository team yang tidak memiliki tag ini akan dianggap tidak mengumpulkan Tugas Besar ini.
4. Kreativitas dalam pengerjaan sangat dianjurkan untuk memperdalam pemahaman. Penilaian sepenuhnya didasarkan dari [kriteria penilaian](#), bukan detail implementasi.
5. 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 siswa dapat membentuk kelompok yang beranggotakan 4 orang.
6. Setiap kelompok diwajibkan untuk membuat tim dalam Github Classroom dengan **nama yang sama pada spreadsheet kelompok**.
7. **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**.
8. Apabila ada pertanyaan lebih lanjut, jangan lupa untuk selalu kunjungi [sheet QnA](#).
9. Akan terdapat asistensi opsional bagi kelompok yang membutuhkan bantuan lebih lanjut. Setiap kelompok yang membutuhkan asistensi dapat mengisi form Asistensi IF3130 Jaringan Komputer pada [form berikut](#). **Asisten akan mengontak maksimal paling lambat H+1 setelah request**.
10. **Segala kecurangan baik sengaja dan tidak disengaja akan ditindaklanjuti oleh pihak asisten, yang akan berakibat sanksi akademik ke setiap pihak yang terlibat.**
11. Deadline dari Tugas Besar ini adalah **Selasa, 28 November 2023, 23.59**
12. Terdapat demo yang akan dilaksanakan setelah tugas besar ini. Demo dilakukan secara **SINKRON**. Sinkron *by default* menggunakan meeting secara *online*. Jika ingin *offline*, diskusikan dengan asisten terkait. Isi kelompok kalian pada *sheet demo* yang akan diberitahukan kemudian sebelum tanggal **Kamis, 30 November 2023, 23.59**.

13. Sebelum demo, harap baca dan pahami [Tata Cara Demo](#). Siapkan *tools* yang akan dipakai.
14. Identitas dan keterangan asisten akan diumumkan setelah waktu *deadline* tugas besar.

VI. Tata Cara Demo

Berikut merupakan langkah-langkah yang dilakukan saat demo:

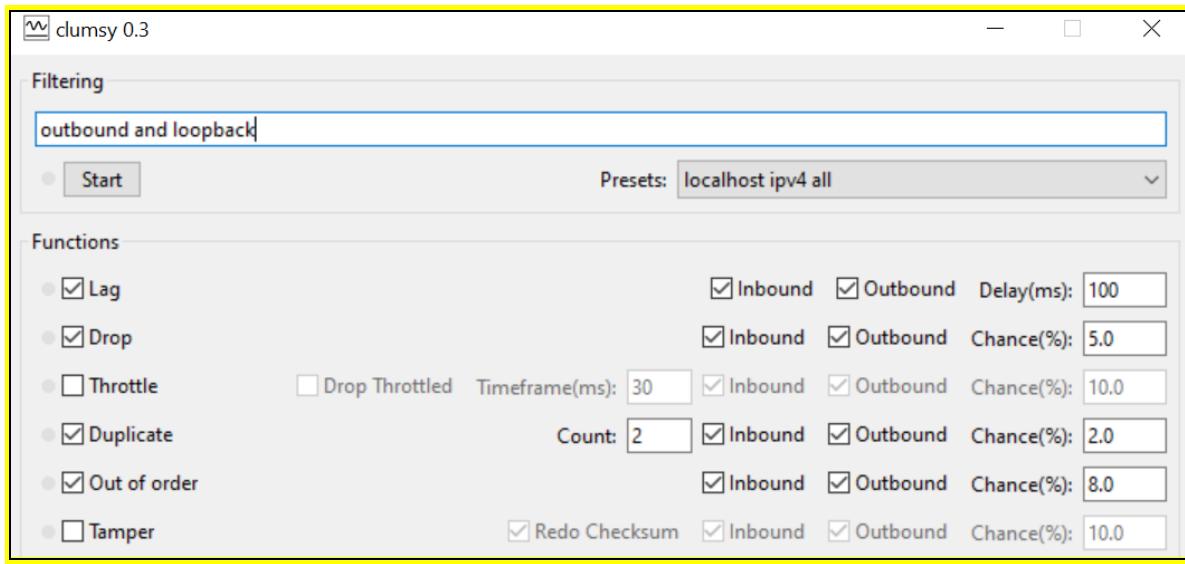
1. Lakukan git status dan git log terlebih dahulu untuk menunjukkan bahwa perubahan terakhir dilakukan sebelum deadline pengerjaan Tugas Besar 1.
2. Jelaskan secara singkat kode yang **telah dibuat** meliputi:
 - a. Implementasi *checksum* dan Segment
 - b. Implementasi pengiriman dan penerimaan data 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](#). Tunjukkan hash md5 dari berkas (gunakan tools eksternal seperti *md5sum*, untuk keperluan komparasi *integrity check* dengan program yang dibuat).
4. Lakukan pengiriman 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. Untuk linux, jalankan perintah ini di terminal Anda dengan akses *superuser*. Disarankan untuk menggunakan VM Ubuntu untuk memudahkan. Bagian ini akan digunakan untuk mensimulasikan jaringan buruk

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

Penting: Setelah menyelesaikan 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%
```

Untuk Windows, substitusi command tc dengan github.com/clumsy. Gunakan konfigurasi berikut



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>
6. tc-netem man page - <https://man7.org/linux/man-pages/man8/tc-netem.8.html>
7. Go-Back-N ARQ - Wikipedia - https://en.wikipedia.org/wiki/Go-Back-N_ARQ
8. Referensi spesifikasi Tugas Besar tahun 2022 - [IF3130 - Tugas Besar 2](#)