

Tugas Besar 2

IF3130 - Jaringan Komputer

"Gotta Catch 'Em All"

Simulasi Koneksi TCP di atas UDP dengan Go-Back-N

Dipersiapkan oleh:
Asisten Lab Sistem Terdistribusi

Didukung oleh:



Waktu Mulai:

11 November 2021, 09.00 WIB

Waktu Akhir:

25 November 2021, 23.59 WIB

Latar Belakang



Setelah Anda menyelamatkan Pokemaniac dari transformasinya yang katastrofik, Anda kembali jalan untuk melanjutkan perjalanan ke Vermilion City, untuk Gym Leader yang ada di kota itu. Sebelum Anda melanjutkan perjalanan, HP Anda berdering. Anda berpikir, siapa orang yang menghubungi Anda di tengah perjalanan ini. Anda membuka HP Anda, dan Anda terkejut, ternyata professor Anda yang tinggal di dekat rumah Anda menghubungi Anda lewat video call.

"Apa kabar? Bagaimana progress Pokedex Anda?" tanya professor tersebut. Anda panik, karena Anda jarang menangkap Pokemon lain karena tidak Anda butuhkan. Itulah akibat kesombongan Anda karena selalu memakai Charizard Anda di setiap pertarungan, jadinya Anda sulit menangkap Pokemon yang lebih lemah. Anda menjawab sudah lumayan banyak ke professor Anda. "Oh begitu? Coba saya lihat. Kebetulan Pokedex Anda terhubung lewat Internet dengan PC saya. Saya bisa SSH ke Pokedex-nya, kebetulan di atas TCP, sehingga informasinya pasti terjamin benar. Kamu tahu lah, hal yang biasa" professor tersebut mengoceh. Anda tidak mengerti maksud beliau apa. "Wah, baru 10% dari keseluruhan hipotesis ya. Hmm, bagaimana kalau Anda berkeliling di sekitar Vermillion? Sekalian, ini saya berikan *e-ticket* ke sebuah kapal feri, siapa tahu Anda bisa melihat-lihat dan bertanya dengan orang-orang di kapal tersebut mengenai Pokemon langka yang ada. Ayo tangkap semuanya! Nanti kabari saya lagi ya" professor tersebut langsung menutup videocallnya.

Anda akhirnya menyanggupi tugas Anda dengan sedikit malas, dan Anda mulai mencari untuk menangkap semua Pokemon yang ada.

Tujuan

Tugas ini memiliki tujuan untuk

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

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 kakas diluar kakas bawaan Python 3. Program dijalankan di lingkungan sistem operasi berbasis **Linux**. Server dan client dibuat secara terpisah dan dijalankan secara terpisah.

Server dan client dijalankan dengan argumen *port* pada antarmuka command line:

```
$ python server.py [port] [path berkas]
```

```
$ python client.py [port] [path penyimpanan berkas]
```

Setiap program **dibuat secara verbose**, yaitu setiap langkah dalam komunikasi yang dilakukan di-output ke stdout, dalam hal ini adalah terminal. Berikut contohnya:

server.py terminal

```
...
(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 terminal

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

Sistem program mengikuti syarat sebagai berikut:

1. Server dan client berada di satu *broadcast domain* yang sama
2. Server dan client berkomunikasi menggunakan **socket UDP**.
3. Client melakukan pencarian server dengan mengirim *request* di *broadcast address*.
4. Server mendengarkan *request* dari client dari *broadcast address*. Apabila server mendapatkan *request*, server akan menyimpan list client. Setiap server mendapatkan client, server akan prompt ke pengguna untuk melanjutkan *listening* atau tidak. Apabila tidak, maka server akan mulai mengirimkan berkas secara sekuensial ke list client.

```
$ python server.py 1337
```

```
Server started at port 1337...
Listening to broadcast address for clients.
```

```
[!] Client (127.0.0.1:10000) found
[?] Listen more? (y/n) y
[!] Client (127.0.0.1:10001) found
[?] Listen more? (y/n) n
```

```
Two clients found:
1. 127.0.0.1:10000
2. 127.0.0.1:10001
```

```
Commencing file transfer...
```

5. Sebelum pengiriman, server akan melakukan [*three way handshake*](#) dengan client.

6. 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. Pengiriman harus bersifat **terurut** dan **tidak mengalami kerusakan**. Berikut merupakan anatomi dari segmen yang dikirim:

Octet Offset	0	1	2	3
0	Sequence Number			
4	Acknowledgment Number			
8	Flags	[empty]	Checksum	
12	Data (maksimal 32768 Bytes)			
...				
32777				

Berikut merupakan keterangan dari setiap bagian dari segmen:

- Sequence Number adalah angka urutan dari segmen yang dikirim.
- Acknowledgment Number adalah angka yang dikirimkan beserta segmen ini yang merupakan Sequence Number segmen sebelumnya yang diterima (Previous Sequence Number) yang menandakan Sequence Number tersebut sudah diterima oleh pihak tersebut.
- Flags merupakan penanda apakah segmen ini merupakan dari jenis segmen-segmen pada komunikasi TCP. Apabila bit ke-n adalah 1, maka segmen merupakan salah satu dari jenis berikut:
 - SYN merupakan flag yang menandakan bahwa segmen ini merupakan permulaan dari *three way handshake* yang dilakukan. SYN terletak di bit ke-1 dari byte Flags
 - ACK merupakan flag yang menandakan bahwa segmen ini merupakan balasan (acknowledgement) dari suatu proses. ACK terletak di bit ke-4 dari byte Flags.
 - FIN merupakan flag yang menandakan bahwa segmen ini merupakan permulaan dari proses *tearing down connection*. FIN terletak di bit ke-0 dari byte Flags.
 - 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). Checksum yang diimplementasikan berupa checksum 16-bit *one's complement*, yang dilakukan dari semua *chunk* 16-bit pada setiap segmen TCP.
 - e. Data untuk setiap segmen merupakan bagian dari berkas yang akan dikirim.
7. Mekanisme pengiriman dilakukan dengan automatic repeat request (ARQ) **Go-Back-N**. Berikut merupakan *pseudocode* dari ARQ Go-Back-N:

```
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 packet received = Rn and the packet is error free then
      Accept the packet and send it to a higher layer
      Rn := Rn + 1
    else
      Refuse packet
      Send acknowledgement for last received packet

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 packet is in transmission then
      Transmit packets where Sb ≤ Sn ≤ Sm.
      Packets are transmitted in order.
```

Pemilihan besar window (N) dibebaskan kepada Anda. Jangan lupa sertakan rasionalisasinya di video penilaian.

8. Setelah server selesai mengirimkan data, server dan client melakukan [*connection closing*](#).

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

Pengumpulan dan Deliverables

- Tugas besar dikerjakan secara berkelompok yang diisi di [sheet berikut](#). Silakan isi *sheet* tersebut sebelum tanggal 13 November 2021, 23.59 WIB. Setelah itu *sheet* akan **dikunci**.
- Setiap kelompok mengerjakan tugas di <https://gitlab.informatika.org> (apabila ada anggota kelompok yang tidak terdaftar di gitlab informatika, silakan menggunakan [github](#)) dengan *repository* yang **private** selama pengerjaan. *Repository* **di-public** setelah pengerjaan. Pastikan **perubahan terakhir dilakukan sebelum deadline** yang telah diberikan.
- Apabila ada pertanyaan lebih lanjut, jangan lupa untuk selalu kunjungi s.id/qna-prak.
- **Segala kecurangan baik sengaja dan tidak disengaja akan ditindaklanjuti oleh pihak asisten, yang akan berakibat sanksi akademik ke setiap pihak yang terlibat.**

•