Assignment – A03

Webserver (Programming)

Penulis : **REN**

Versi : **2 (20240923-0800)**

Daftar Perubahan

Versi	Timestamp	Halaman	Perubahan
1	20240923-0800	All	First Release
2	20240923-1330	9, 11	Penjelasan mengenai header yang dikirim, serta penambahan header Content-Length. Perubahan pembagian poin nilai grader
3	20240923-2315	10	Fix URI untuk path /greeting -> /greet
4	20240930-1300	9, 10	Klarifikasi tipe konten dan testing yang dilakukan untuk kesesuaian NPM.

Daftar Isi

Assignment – A03

Webserver (Programming)

Informasi Umum

- Tipe Tugas : Individu
- Batas Waktu Pengumpulan : Jumat, 4 Oktober 2024 17.00 waktu SCeLE
- Format Penamaan Berkas : - Answer Sheet: A03_[NPM].pdf
  - Server Code: A03_[NPM]_Server.go
  - Client Code: A03_[NPM]_Client.go
- *Template* Lembar Jawaban : Click Here
- *Template* Code : Click Here

Ekspektasi Hasil Pembelajaran

Mahasiswa diharapkan dapat mengimplementasikan webserver (C3) menggunakan *socket programming*.

Prasyarat

Ekspektasi *Pre-Existing Knowledge*

Agar dapat mengerjakan tugas ini secara efektif, mahasiswa harus memiliki pemahaman dasar tentang materi berikut:

1. Application Layer protocols, terutama Hypertext Transfer Protocol (HTTP). Silakan rujuk materi baca untuk materi ini.
2. Go programming language. Silakan rujuk tutorial Go apa pun untuk materi ini.
3. Socket Programming in Go. Kami mengharapkan siswa telah menyelesaikan Hands-on Tutorial (H01) sebelum mengerjakan tugas ini.

Persiapan GCP Instance

Setiap mahasiswa diharapkan untuk menyiapkan 2 (dua) instance Google Compute Engine (GCE) yang akan menjalankan peran Client dan Server. Mesin yang dipilih harus memenuhi persyaratan awal yang ditetapkan di A01a dengan spesifikasi tambahan sebagai berikut:

1. Mesin sudah terinstal bahasa pemrograman Go.
2. Sistem harus dapat diakses melalui port TCP unik Anda seperti yang telah disebutkan di A01a.

Nomor Port Unik

Untuk kepentingan kuliah ini, kalian perlu memilih sebuah nomor dengan 4 sampai 5 digit angka yang akan digunakan sebagai nomor port unik yang nantinya dipakai untuk *networking services*. Silakan rujuk kembali dokumen A01a untuk nomor port ini.

Deskripsi

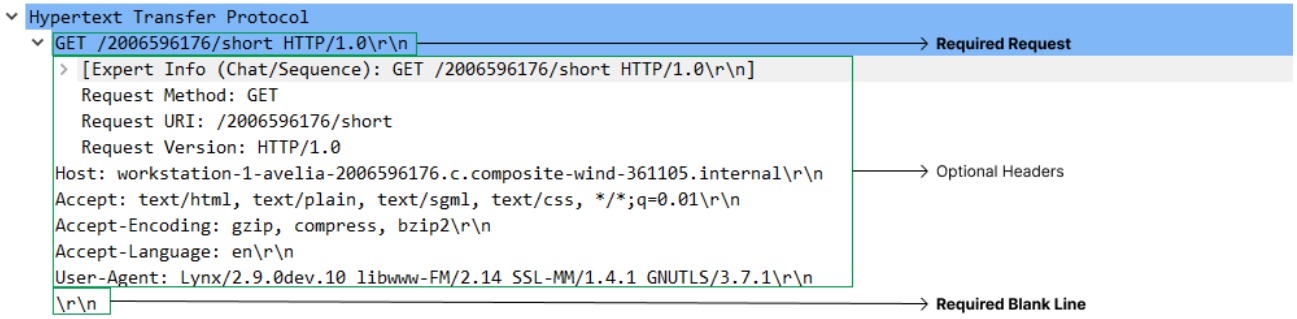
Hypertext Transfer Protocol (HTTP)

HTTP adalah protokol yang mendefinisikan komunikasi antar web browser (client) dan server. Client dan server bertukar data menggunakan pesan HTTP. Terdapat dua jenis pesan, yaitu *request* dan *response*. *Request* dapat dikirim oleh client untuk melakukan suatu tindakan di server. Server dapat menjawab *request* dengan *response*.

Pesan HTTP terdiri dari informasi berbentuk teks ASCII, dan terdiri dari beberapa baris. Berikut ini adalah contoh bagaimana pesan HTTP di-*encode* dalam ASCII (sisi kiri). Anda dapat memeriksa tabel ASCII Anda dan Anda akan menemukan bahwa teks di sebelah kanan adalah representasi dari kode ASCII yang ada di sisi kiri.

```
73 11 47 45 54 20 2f 32  30 30 36 35 39 36 31 37   s·GET /2 00659617
36 2f 73 68 6f 72 74 20  48 54 54 50 2f 31 2e 30   6/short  HTTP/1.0
0d 0a 48 6f 73 74 3a 20  77 6f 72 6b 73 74 61 74   ··Host:  workstat
69 6f 6e 2d 31 2d 61 76  65 6c 69 61 2d 32 30 30   ion-1-av elia-200
36 35 39 36 31 37 36 2e  63 2e 63 6f 6d 70 6f 73   6596176. c.compos
69 74 65 2d 77 69 6e 64  2d 33 36 31 31 30 35 2e   ite-wind -361105.
69 6e 74 65 72 6e 61 6c  0d 0a 41 63 63 65 70 74   internal ··Accept
```

Pada HTTP/1.1, pesan dikirim secara terbuka. HTTP/2 membagi pesan ke dalam beberapa frame HTTP. Perubahan ini memberikan optimisasi dan peningkatan *performance*.



Request dan *response* HTTP dimulai dengan beberapa baris header, masing-masing diakhiri CRLF (*carriage return-line feed*, atau "\r\n" dalam C-string). Baris pertama dari *request* (bagian Required Request) harus merupakan baris *request* atau *response* yang valid. Setelah baris pertama *request*, muncul *header-header* yang bersifat opsional. Informasi yang disediakan di bagian *request* ini dapat memberikan informasi yang berguna bagi client dan server, seperti *browser* apa yang sedang dipakai. Pada akhir baris-baris *header*, ada satu baris kosong (hanya terdiri dari CRLF, ditulis "\r\n" pada contoh). Bagian baris kosong ini diperlukan sesuai dengan spesifikasi HTTP. Untuk *response* HTTP, di bawah bagian tersebut terdapat *message body*. Bagian ini bisa berisi html dan dapat juga kosong.

Spesifikasi

Dalam tugas ini, Anda ditugaskan untuk menerapkan teknik socket programming untuk mengimplementasikan HTTP pada program client dan server yang dibangun dari awal. Tujuannya adalah untuk menghasilkan sepasang client dan server yang fungsional dan dapat berkomunikasi sesuai dengan HTTP/1.1. Pasangan ini dikatakan fungsional jika client dapat membuat request ke server dan server dapat menangani request tersebut menggunakan HTTP/1.1 dengan header yang ditentukan oleh tugas ini.

Sebelum kita mulai, berikut ini adalah beberapa aturan umum dan/atau ekspektasi untuk program yang akan dibuat:

1. Program client dan server harus diimplementasikan sebagai program independen, sehingga akan ada dua program yang berbeda.
2. Untuk memberikan pengalaman yang konsisten kepada siswa dan tim asisten, program harus ditulis dalam bahasa pemrograman Go.
3. Karena tugas ini mengharuskan Anda untuk mengimplementasikan protokol komunikasi sendiri dari awal, **Anda tidak diizinkan untuk menggunakan library diluar bawaan dan/atau library yang terkait dengan HTTP**. Library bawaan seperti fmt, net, os, dll. dan library-library untuk marshal tipe data seperti encoding/xml atau encoding/json masih diperbolehkan. Jika Anda ragu, silakan berkonsultasi dengan tim asisten mengenai library yang diizinkan.
4. Karena Anda akan mengimplementasikan HTTP/1.1, Anda diharapkan untuk menggunakan TCP sebagai tipe *socket*.
5. Pekerjaan Anda akan dinilai dengan bantuan "grader" menggunakan teknik *unit testing*. Oleh karena itu, Anda diharuskan untuk menggunakan template kode yang diberikan. Setidaknya Anda mengikuti instruksi yang diberikan dalam *method skeleton* yang disediakan.
6. Kami telah menyediakan beberapa struct sebagai kesepakatan tentang bagaimana data harus disimpan dan digunakan dalam program, seperti HTTPRequest, HTTPResponse, Student, dan GreetResponse. Pastikan bahwa Anda mengikuti spesifikasi dari struct tersebut.
7. Ingat bahwa HTTP menentukan penggunaan CRLF ("\r\n") untuk memisahkan setiap baris. Harap perhatikan hal ini, terutama jika Anda mengembangkan program Anda di sistem *nix (macOS atau Linux) yang menggunakan LF ("\n") sebagai karakter default baris baru.

Ekspektasi Struktur Pesan HTTP

Karena tidak ada struktur lengkap pesan HTTP yang "wajib" secara konsisten, kami berharap Anda setidaknya mengikuti struktur dasar message HTTP, yaitu header dan data (body). Untuk header, informasi berikut ini harus ada seperti yang ditunjukkan pada struct HTTPRequest dan HTTPResponse:

1. HTTP *Request* (HTTPRequest struct)
    a. *HTTP Request Line*, yang berisi method request, target (URI), dan versi HTTP.
    b. *Content type* yang dapat diterima.
    c. Host dari *request* tersebut.
2. HTTP Response (HTTPResponse struct)
    a. *HTTP Response Line*, which consists of the response version, status code, and status text.

b. *Response content type*.

Harap rujuk spesifikasi HTTP mengenai informasi atribut tersebut. Dokumentasi dari MDN Web Docs mungkin dapat membantu.

Spesifikasi Program Client

Program *client* Anda diwajibkan memiliki spesifikasi sebagai berikut:

1. Program diwajibkan meminta input ke user untuk hal berikut:
   a. URL dari *request* yang akan dibuat.
   b. *MIME type* dari *request*.
2. Program diwajibkan dapat merubah input menjadi pesan HTTP request menggunakan method RequestEncoder. Buat method *encoding request* untuk *client* Anda yang dapat merubah struct HTTPRequest menjadi pesan HTTP request, yang nanti akan dikirim ke server.

```
func RequestEncoder(req HttpRequest) []byte {
    // do request message encoding
    return []byte(result)
}
```

3. Program wajib berkomunikasi dengan server yang dinyatakan dalam URL menggunakan HTTP dan dapat menerima *response*.
4. Program wajib merubah *response* dari format pesan HTTP menjadi format yang dapat dibaca menggunakan method ResponseDecoder. Dalam *template* program yang disediakan, ada kerangka method ResponseDecoder yang menerima satu argumen: *raw response message* dalam bentuk *array byte* yang me-*return* struct HTTPResponse.

```
func ResponseDecoder(bytestream []byte) HttpResponse {
    // do response message decoding
    return res
}
```

5. Program wajib mencetak hasil *response* dengan format berikut
   1. Program *client* akan bertindak seperti browser. Pertama, program ini akan meminta URL. URL harus menyertakan protokol, host, port, dan URI, sebagai contoh, (http://123.123.123.123:7281/{URI})

      ```
      input the url: http://127.0.0.1:6200/
      ```

   2. Lalu, program akan meminta *MIME type*.

      ```
      input the data type: text/html
      ```

   3. Setelah client berhasil mengirim *request* ke server, program akan mencetak *response* dari server. Apabila *response* bukan berupa HTML, maka cetak hasil *parse*-nya.

```
PS                                          go run .\client\client.go
input the url: http://127.0.0.1:3000/greet/2106639945
input the data type: application/json
Status Code: 200
Body: {"Student":{"Nama":"Ren","Npm":"2106639945"},"Greeter":"Ren"}
Parsed: {{Ren 2106639945} Ren}
PS                                          go run .\client\client.go
input the url: http://127.0.0.1:3000/
input the data type: text/html
Status Code: 200
Body: <html><body><h1>Halo, dunia! Aku Ren</h1></body></html>
PS                                          >
```

Spesifikasi Program Server

Program server Anda diwajibkan memiliki spesifikasi sebagai berikut:

1. Server harus dapat membuka koneksi di port unik Anda.
2. Server wajib merubah *request* dari format pesan HTTP menjadi format struct HTTPRequest menggunakan method RequestDecoder. Dalam *template* program yang disediakan, ada kerangka method RequestDecoder yang menerima satu argumen: *raw request message* dalam bentuk *array byte* yang me-*return* struct HTTPRequest.

   ```
   func RequestDecoder(bytestream []byte) HttpRequest {
       // do request message decoding
       return req
   }
   ```

   Anda diwajibkan untuk menggunakan method ini untuk men-*decode* pesan *request*. Method ini akan diekspos ke *grader* untuk menilai hasil decoding Anda. Anda dapat menambahkan method untuk membantu Anda, tetapi Anda tidak diizinkan untuk mengubah spesifikasi method ini. **Format hasil decoding harus mengikuti struktur yang telah disediakan**.

3. Server harus dapat menangani *request* HTTP yang masuk sesuai dengan URI-nya dengan mempertimbangkan header-header dengan struct HTTPResponse sebagai outputnya.

   Terdapat tiga hal yang perlu Anda lakukan:

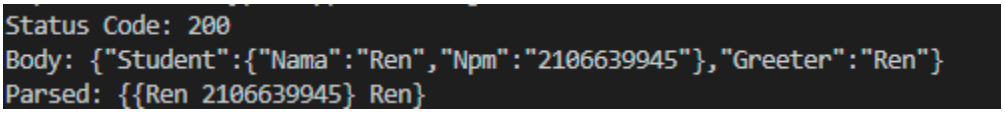
   1. Menerima *request* ke '/'

      a. Konten *response* berbentuk text/html.
      b. *Status code* yang dikirim adalah 200.
      c. Pesan yang dikirim adalah "Halo, dunia! Aku <NAMA_ANDA>".
      d. Screenshot berikut adalah contoh *response*

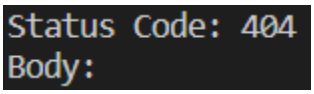
      ```
      Status Code: 200
      Body: <html><body><h1>Halo, dunia! Aku Ren</h1></body></html>
      ```

2. Menerima *request* ke '/greet/<NPM>' dan '/greet/<NPM>?name=<str>'

    a. Konten yang dikirim berbentuk application/json atau application/xml, bergantung dengan tipe apa saja yang diterima *client*.
    b. Konten yang dikirimkan adalah suatu struct yang berisi dua properti: Student dan Greeter.
        a. Student adalah suatu struct yang memiliki dia properti: Nama dan Npm, nilai dari properti tersebut didasarkan dari nama dan NPM Anda.
        b. Greeter adalah nilai dari parameter "name" di URI, atau nama Anda jika nilai parameter kosong.
    c. Apabila *client* memberikan tipe lain selain application/json atau application/xml, cukup kirimkan dalam bentuk application/json.
        a. Apabila *client* dapat menerima *multiple*, maka cukup kirim application/json.
        b. Apabila *client* memberikan *q value*, maka cukup kirim application/json.
    d. Apabila nilai NPM tidak sesuai dengan NPM Anda, maka server mengirimkan *status code* 404 tanpa isi (body) apapun.
    e. Screenshot berikut adalah contoh *response*



3. Jika *client* membuat *request* ke URI yang tidak ditentukan di sini, kirimkan *response* HTTP dengan *status code* 404. Anda dapat membiarkan *field* lainnya kosong. Screenshot berikut adalah contoh *response*.



4. Server harus memberikan header yang sesuai dengan response yang dibuat dari handler. (Content-Type, Content-Length)
5. Server harus dapat merubah hasil *response* dari struct menjadi pesan *response* HTTP dengan menggunakan ResponseEncoder yang kemudian dikirim melalui *socket.*

```
func ResponseEncoder(res HttpResponse) []byte {
    // do response message encoding
    return []byte(concenatedRes)
}
```

6. Server harus dapat melayani lebih dari satu *request*, tidak selesai setelah satu *request.*

[40 poin] Self-Testing

Catatan: Kami menyarankan Anda untuk melakukan seluruh prosedur *self-test* sekaligus untuk memastikan konsistensi. Paling tidak, pastikan bahwa semua instance tidak dimatikan antar eksekusi uji coba pertama dan terakhir.

Untuk mendapatkan poin untuk tugas ini, Anda harus dapat mendemonstrasikan bahwa pasangan program client-server Anda berfungsi dengan membuat beberapa *request* yang mencakup persyaratan tugas ini. Akan ada 3 *test suite* yang harus dijalankan yang menunjukkan kemampuan server Anda untuk menangani URI sesuai dengan persyaratan:

1. **(Test Code: 01) GET '/'**
   Untuk tes ini, Anda diharuskan untuk melakukan *request* GET ke / URI server.
2. GET /greet/<NPM>
   Untuk tes ini, Anda diharuskan untuk melakukan *request* GET ke URI /greet/<NPM> pada server dengan mengharapkan jenis konten tertentu. Rangkaian pengujian ini berisi tiga subtes:
   a. **(Test Code: 02a)** JSON sebagai *content type* (MIME Type: application/json)
   b. **(Test Code: 02b)** XML sebagai content type (MIME Type: application/xml)
   c. **(Test Code: 02c)** Kesesuaian nilai NPM pada URI, tes menggunakan nilai NPM yang tidak sesuai dengan NPM Anda
3. GET /greet/<NPM>?name=<str>
   Untuk tes ini, Anda diharuskan untuk melakukan *request* GET ke URI /greet/<NPM> pada server dengan mengharapkan jenis konten tertentu, serta dengan nama *greeter* yang berbeda. Rangkaian pengujian ini berisi tiga subtes:
   a. **(Test Code: 03a)** JSON sebagai *content type* (MIME Type: application/json)
   b. **(Test Code: 03b)** XML sebagai content type (MIME Type: application/xml)
   c. **(Test Code: 03c)** Kesesuaian nilai NPM pada URI, tes menggunakan nilai NPM yang tidak sesuai dengan NPM Anda
4. **(Test Code: 04)** Tes dengan URI lain (404 Handler)
   Untuk tes ini, Anda diminta untuk melakukan *request* GET ke URI yang tidak tercakup dalam penugasan ini yang seharusnya menghasilkan respons 404 Not Found. Gunakan URI /lainnya untuk tes ini.

Prosedur untuk menjalankan rangkaian self-test adalah sebagai berikut:

1. Pilih satu instance yang bertindak sebagai *client* (selanjutnya disebut Client) dan *instance* lain yang bertindak sebagai server (selanjutnya disebut Server). Laporkan nama instance dan alamat IP publiknya di lembar jawaban Anda.
   Untuk bukti alamat IP publik, Anda harus memberikan screenshot eksekusi perintah berikut:
   ```
   curl -H "Metadata-Flavor: Google"
   http://metadata/computeMetadata/v1/instance/network-
   interfaces/0/access-configs/0/external-ip -w "\n"
   ```
2. Jalankan program server di **Server**.

3. Jalankan program *client* di Client dan berikan input sesuai dengan rangkaian tes pertama. Ambil screenshot dari terminal Client Anda yang menunjukkan input dan output, lalu laporkan di lembar jawaban Anda.
4. Ulangi Langkah 3 untuk setiap subtes. Subtes dalam suatu tes dihitung sebagai satu tes. Oleh karena itu, akan ada 8 eksekusi tes.

Catatan: Semua tes harus dijalankan dalam pasangan client-server yang sama. Jika *force majeure* menghalangi Anda untuk melakukan hal ini, silakan buka tiket dan beri tahu tim asisten tentang kondisi Anda.

[60 poin] Penilaian Grader

Bagian tugas ini akan dinilai menggunakan *grader*. Jangan khawatir, asisten akan memasukkan kode Anda ke grader saat menilai tugas Anda. Anda dapat menambahkan method untuk membantu Anda, tetapi Anda tidak diizinkan untuk mengubah spesifikasi method.

[5 poin] Request Encoder

Anda diwajibkan menggunakan metode yang disediakan dalam *template* untuk meng-*encode* HTTP *request*.

[10 poin] Request Decoder

Anda diwajibkan menggunakan metode yang disediakan dalam *template* untuk meng-*decode* HTTP *request*.

[5 poin] Response Encoder

Anda diwajibkan menggunakan metode yang disediakan dalam template untuk meng-*encode* HTTP *response*.

[10 poin] Response Decoder

Anda diwajibkan menggunakan metode yang disediakan dalam *template* untuk meng-*decode* HTTP *response*.

[25 poin] /greet/<NPM> Handler (termasuk /?name=)

Anda diwajibkan menggunakan metode HandlerRequest yang disediakan di dalam *template* untuk menangani *routing* setiap *request*. Handler harus dapat mengembalikan *status code* dan *response body* yang valid sesuai dengan *request* HTTP yang diminta.

[5 poin] Handler URI Lain

Anda diwajibkan menggunakan metode HandlerRequest yang disediakan di dalam *template* untuk menangani *routing* setiap *request*. Handler harus dapat mengembalikan *status code* dan *response body* yang valid sesuai dengan *request* HTTP yang diminta.

Pengumpulan Berkas

Untuk tugas ini, Anda diharuskan untuk mengirimkan file berikut ke slot submisi SCeLE:

1. File lembar jawaban: **A03_[NPM].pdf**
2. *Source code* untuk program *client:* **A03_[NPM]_Client.go**
3. *Source code* untuk program server*:* **A03_[NPM]_Server.go**

Peraturan

Keterlambatan

Anda diharapkan dapat mengumpulkan hasil pekerjaan yang dilakukan sebelum batas waktu pengumpulan. Jika terdapat kondisi di mana Anda terpaksa terlambat mengumpulkan hasil pekerjaan, terdapat jangka waktu tambahan di mana Anda masih diperbolehkan mengumpulkan hasil pekerjaan dengan konsekuensi tertentu. Jika X adalah durasi setelah batas waktu pengumpulan yang ditetapkan sampai waktu Anda mengumpulkan hasil pekerjaan, Anda akan menerima penalti nilai pekerjaan sebagaimana diatur pada peraturan berikut ini:

- $X < 10$ menit : Tidak ada penalti
- 10 menit $\leq X < 2$ jam : 25% penalti
- 2 jam $\leq X < 4$ jam : 50% penalti
- 4 jam $\leq X < 6$ jam : 75% penalti
- $X \geq 6$ jam : Cut-off (Pekerjaan anda tidak akan diterima)

Plagiarisme

Anda diperbolehkan berdiskusi tentang pekerjaan Anda dengan peserta kuliah lain atau pihak lainnya, namun Anda harus memastikan bahwa semua pekerjaan yang dikumpulkan adalah murni hasil pekerjaan Anda sendiri. Anda dilarang keras melakukan tindak plagiarisme atau kecurangan akademik lainnya. Menurut kamus daring Merriam-Webster, plagiarisme berarti:

- Mencuri dan mengklaim (ide atau kata orang lain) sebagai milik sendiri
- Menggunakan hasil (karya/pekerjaan orang lain) sebagai milik sendiri
- Melakukan pencurian literatur/sastra
- Merepresentasikan ulang sebuah ide/produk yang sudah ada sebagai sesuatu yang bersifat baru dan orisinil.

Tim pengajar memiliki hak untuk meminta klarifikasi terkait dugaan ketidakjujuran akademik, terutama plagiarisme, dan memberikan konsekuensi berupa pengurangan nilai hasil pekerjaan atau pencabutan nilai (nilai diubah menjadi nol) untuk hasil pekerjaan yang terkonfirmasi dikerjakan secara tidak jujur.