# HTTP Server

CS431 - Computer Networks

Ahmed Bahgat Hussein Elsherif
18010078

## Server

A minimal file server was implemented which parses HTTP requests and sends the corresponding file over the TCP socket (if exists).

### Features

- Multithreaded concurrent connections implemented in a thread pool, to save the time overhead of creating and destroying threads. Multiple concurrency controls constructs were used to ensure thread safety such as mutex locks and condition variables.
- An HTTP request parser to parse HTTP verb, URI, version, and headers. (I was only interested in Content-Length)
- Object oriented design: Server, Request, and Response were implemented in classes
- Dynamic timeout: timeout is calculated based on the number of queued connections. The number is divided by the number of physical threads in the CPU to calculate what I denote by the number of groups to be served. A maximum time out is exponentially reduced based on the number of groups.

$$\#\ groups = \frac{\#\ queued\ connections}{\#\ physical\ threads}$$

$$timeout = max\ timeout\ /\ 2^{\#\ groups}$$

## Design

```
class Server
{
   // server main loop
   void Listen();
   // constructor
   Server(int port);
   // initialize thread pool
   void init_server_pool();
   // close server
   void closeServer();
   // accept connection from client
   void acceptClient();
   // enqueue the client to the connection queue
   void enqueue(int socket);
   // dequeue client from the connection queue
   int dequeue();
   // serve get response
   int serve_get(int client_socket, Request &request);
   // serve post response
   int serve_post(int client_socket, Request &request);
};
```

## Client

### Features

- Parse commands from file and send GET/POST request accordingly
- GET files from server and save them on desk
- POST files to server
- Minimal error handling

### Design

```
// Parse commands from file

size_t parse_input();

// initialize connection with server

int init_socket(const char *hostname, int port);

// send get request to the socket with the specified URI

int serve_get(int server_socket, const char *uri);

// send post request to the socket with the specified URI

int serve_post(int server_socket, const char *uri);
```

## Helper classes

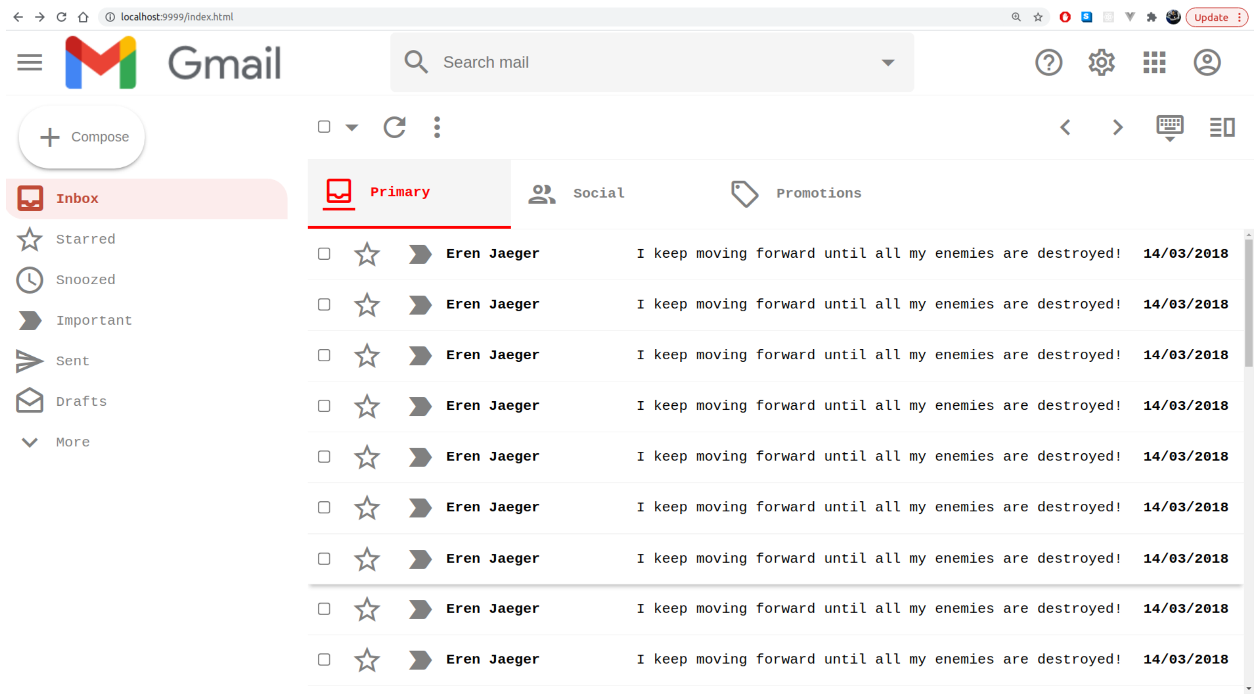
```
class Request{};

class Response{};
```

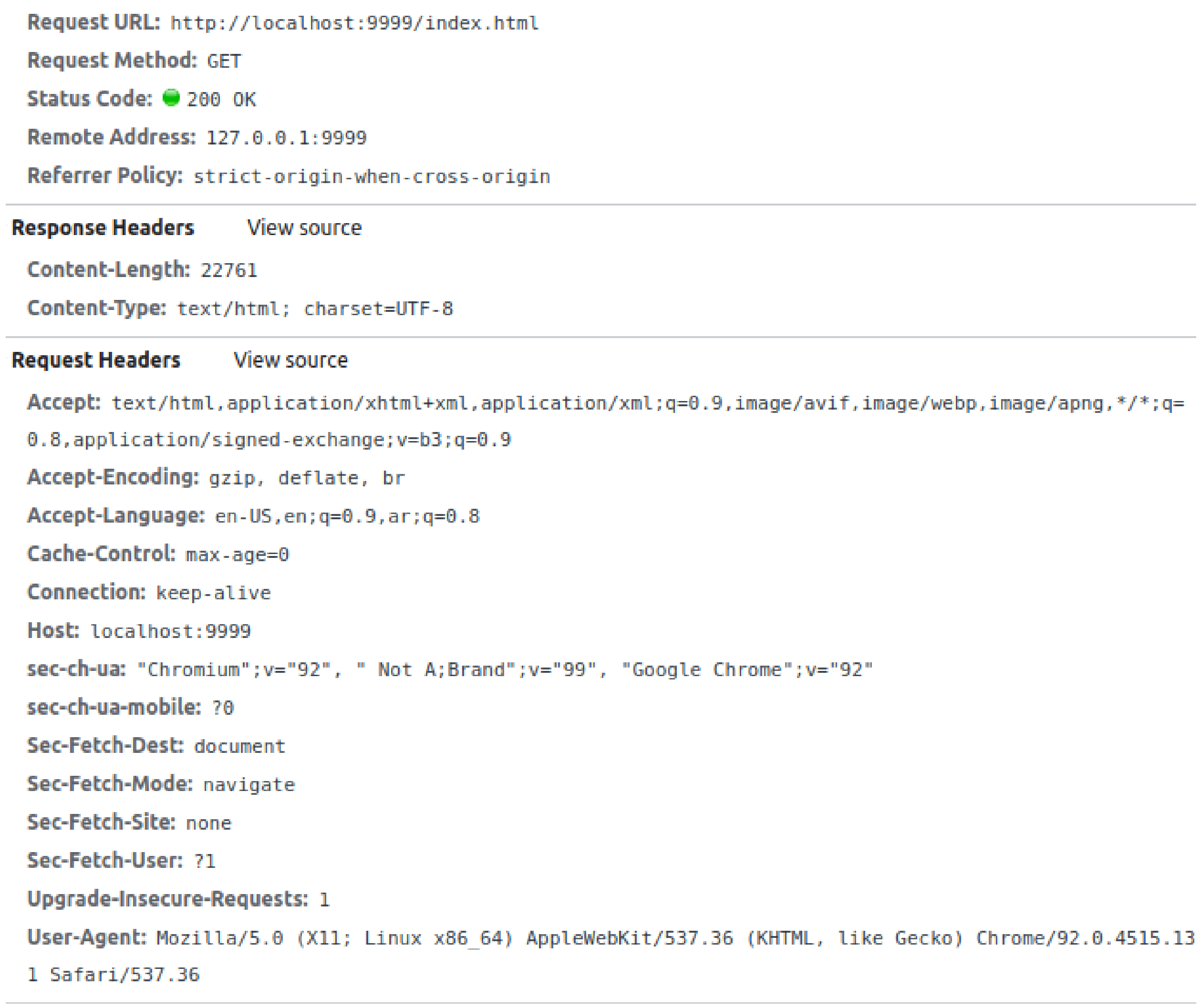
Both classes parse the request/response string

## Bonus

### Use server with browser



```
Eren Jaeger    I keep moving forward until all my enemies are destroyed!    14/03/2018
Eren Jaeger    I keep moving forward until all my enemies are destroyed!    14/03/2018
Eren Jaeger    I keep moving forward until all my enemies are destroyed!    14/03/2018
Eren Jaeger    I keep moving forward until all my enemies are destroyed!    14/03/2018
Eren Jaeger    I keep moving forward until all my enemies are destroyed!    14/03/2018
Eren Jaeger    I keep moving forward until all my enemies are destroyed!    14/03/2018
Eren Jaeger    I keep moving forward until all my enemies are destroyed!    14/03/2018
Eren Jaeger    I keep moving forward until all my enemies are destroyed!    14/03/2018
Eren Jaeger    I keep moving forward until all my enemies are destroyed!    14/03/2018
```



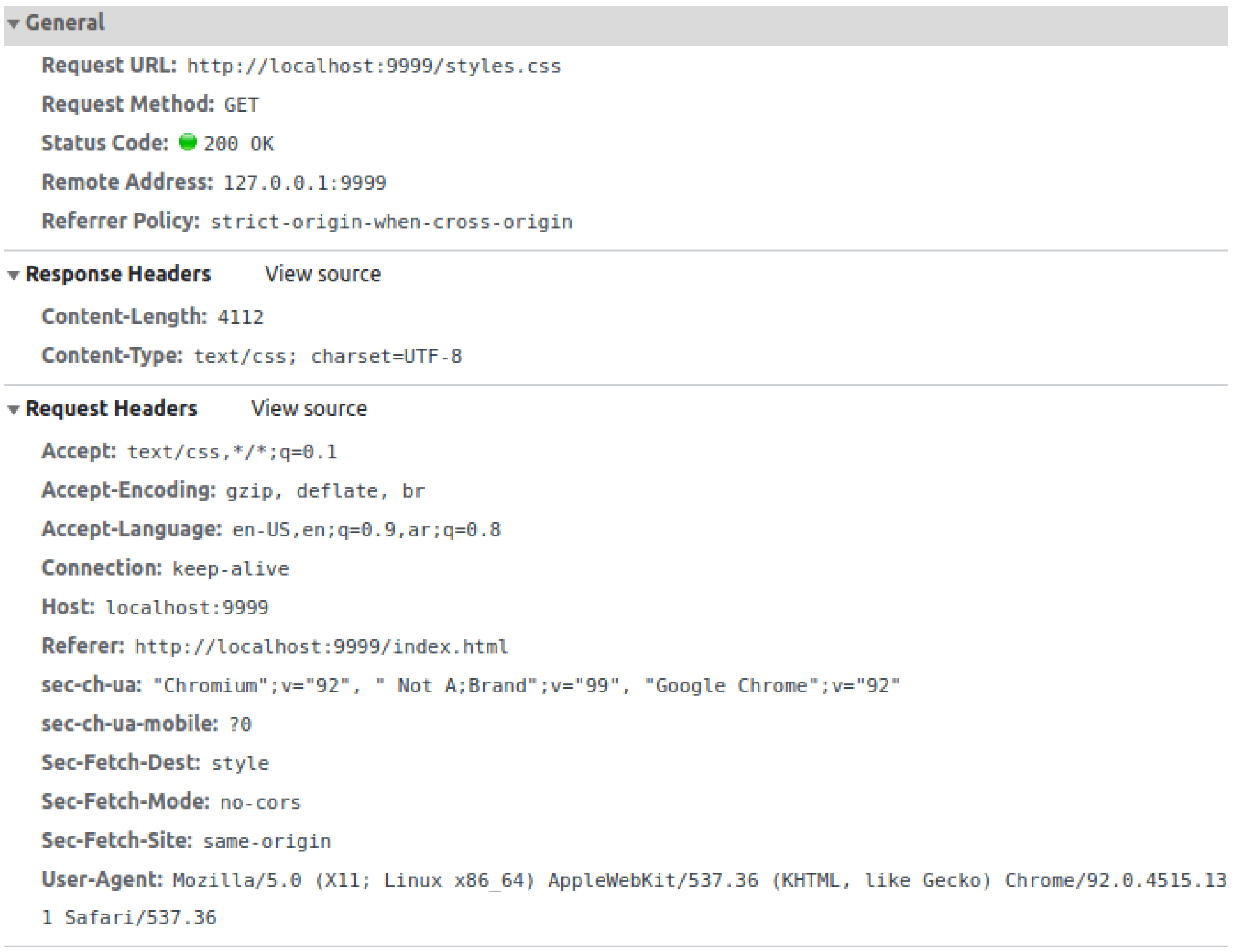
**Request URL:** `http://localhost:9999/index.html`
**Request Method:** GET
**Status Code:** ● `200 OK`
**Remote Address:** `127.0.0.1:9999`
**Referrer Policy:** `strict-origin-when-cross-origin`

**Response Headers** View source

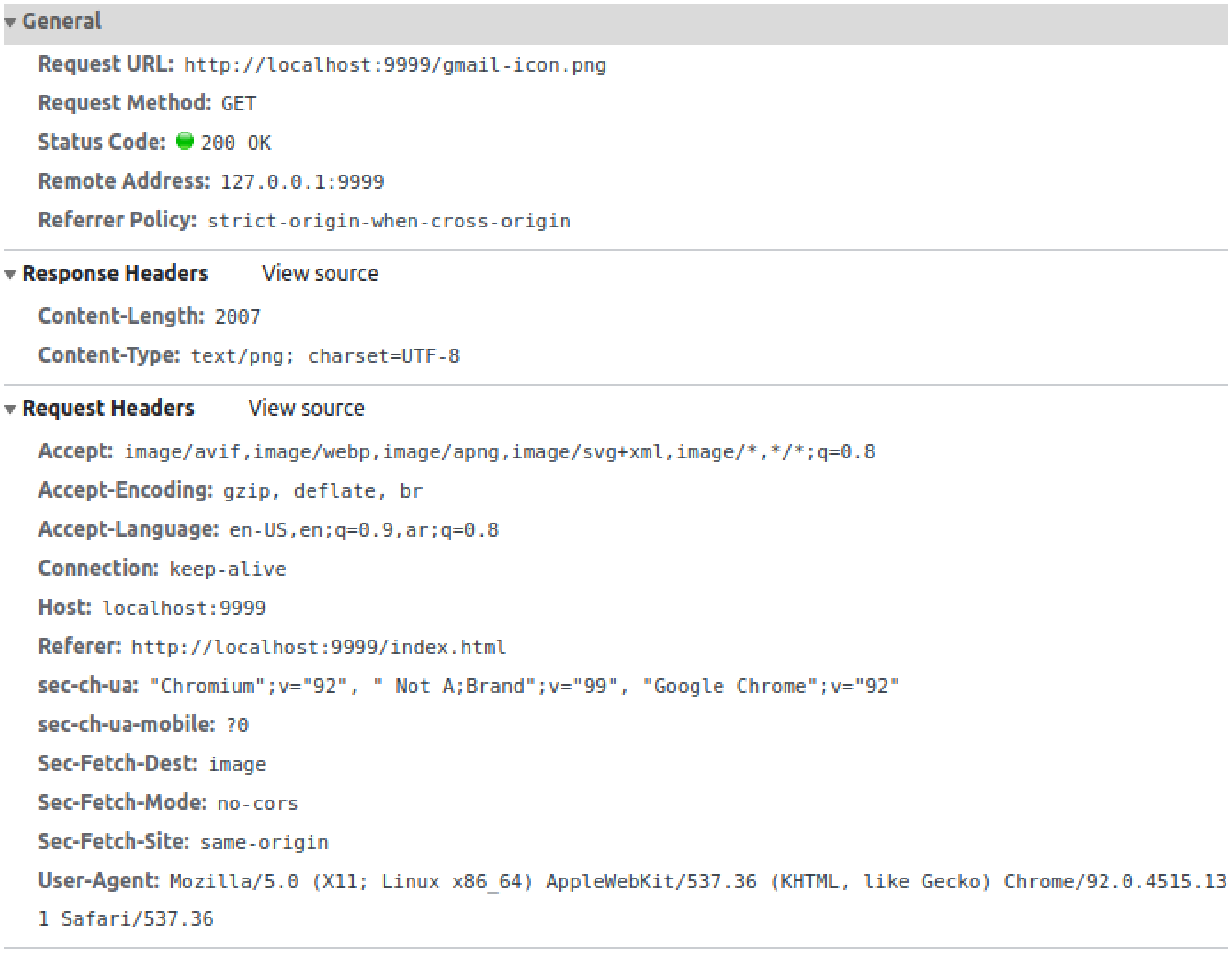
**Content-Length:** `22761`
**Content-Type:** `text/html; charset=UTF-8`

**Request Headers** View source

**Accept:** `text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9`
**Accept-Encoding:** `gzip, deflate, br`
**Accept-Language:** `en-US,en;q=0.9,ar;q=0.8`
**Cache-Control:** `max-age=0`
**Connection:** `keep-alive`
**Host:** `localhost:9999`
**sec-ch-ua:** `"Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"`
**sec-ch-ua-mobile:** `?0`
**Sec-Fetch-Dest:** `document`
**Sec-Fetch-Mode:** `navigate`
**Sec-Fetch-Site:** `none`
**Sec-Fetch-User:** `?1`
**Upgrade-Insecure-Requests:** `1`
**User-Agent:** `Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36`



```
▾ General
    Request URL: http://localhost:9999/styles.css
    Request Method: GET
    Status Code: ● 200 OK
    Remote Address: 127.0.0.1:9999
    Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers     View source
    Content-Length: 4112
    Content-Type: text/css; charset=UTF-8
▾ Request Headers     View source
    Accept: text/css,*/*;q=0.1
    Accept-Encoding: gzip, deflate, br
    Accept-Language: en-US,en;q=0.9,ar;q=0.8
    Connection: keep-alive
    Host: localhost:9999
    Referer: http://localhost:9999/index.html
    sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"
    sec-ch-ua-mobile: ?0
    Sec-Fetch-Dest: style
    Sec-Fetch-Mode: no-cors
    Sec-Fetch-Site: same-origin
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.13
    1 Safari/537.36
```



```
▾ General
    Request URL: http://localhost:9999/gmail-icon.png
    Request Method: GET
    Status Code: ● 200 OK
    Remote Address: 127.0.0.1:9999
    Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers     View source
    Content-Length: 2007
    Content-Type: text/png; charset=UTF-8
▾ Request Headers     View source
    Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
    Accept-Encoding: gzip, deflate, br
    Accept-Language: en-US,en;q=0.9,ar;q=0.8
    Connection: keep-alive
    Host: localhost:9999
    Referer: http://localhost:9999/index.html
    sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"
    sec-ch-ua-mobile: ?0
    Sec-Fetch-Dest: image
    Sec-Fetch-Mode: no-cors
    Sec-Fetch-Site: same-origin
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.13
    1 Safari/537.36
```

## Performance Evaluation

I wrote a BASH script that ran $X numbers of clients concurrently. But, I really could not think of a way to evaluate performance. However, the server served all the requests properly.

## How to run

- First compile both server and clients using Makefile supplied by using

```
make
```

- To run server

```
./server [port]
```

- To run client

```
./client [requests-file-path [client-folder [hostname [port]]]]
```

**Client-folder is without slashes**

- The requests file follows:

```
client_get path
client_post path
```

**Note that path in post is any path on the client machine**

- You can use the supplied bash script to run multiple clients that parse the same request.txt file

```
./run_clients.sh port number
```