CS425: Computer Networks - Assignment 0

Objective

In this assignment, you will create a simple HTTP server from scratch in Python or C++. This will help you understand the basics of network communication and the HTTP protocol, serving as the foundation for more advanced concepts in Computer Networks.

Task Description

You are required to:

1. Build an HTTP server that listens on a specific port.
2. Handle HTTP GET requests and serve static files from a specific directory.
3. Implement basic error handling (e.g., return 404 for missing files).
4. Add a logging feature to log requests and responses.

Requirements

Core Features

1. **Start the Server**:
   - The server should start by specifying the port number and a directory to serve files from.
   - Example command:

```
python http_server.py 8080 ./static
```

   OR

```
./http_server 8080 ./static
```

2. **Handle HTTP GET Requests**:
   - Serve static files (e.g., `html`, `css`, `js`, `png`, `jpg`) from the specified directory.
   - If the requested file exists in the directory, return it with the correct HTTP status code `200 OK` and appropriate `Content-Type` header.
   - If the requested file does not exist, return an HTTP status code `404 Not Found` with an error page.
3. **Logging**:
   - Log each incoming request in the following format:

```
[timestamp] [Client_IP:Client_Port] "<HTTP_Method> <Path> <HTTP_Version>" <
    Status_Code>
```

   - Example log entry:

```
[2024-12-16 12:30:45] [127.0.0.1:56789] "GET /index.html HTTP/1.1" 200
```

4. **Error Handling**:
   - Return a meaningful `404.html` error page if the file is not found.
   - Ensure your server does not crash for invalid requests.

Optional Features (Bonus)

- **Directory Listing**: If a directory is requested, list its contents in an HTML page.
- **MIME Type Detection**: Dynamically detect the MIME type of files served.
- **Graceful Shutdown**: Allow the server to handle `SIGINT` or `Ctrl+C` to shutdown cleanly.
- **Concurrency**: Handle multiple requests simultaneously (e.g., using threading in Python or multithreading in C++).

Instructions

1. **Setup**:
   - Install any required dependencies for your language (if any).
   - Create a `static` folder in the same directory as your code and add test files like `index.html`, `style.css`, and `image.png`.
   - Example file structure:

```
static/
  index.html
  style.css
  image.png
```

2. **Server Code**:
   - Write code to implement the HTTP server.
   - Use `TCP` sockets for communication.
3. **Testing**:
   - Test your server with a web browser and tools like `curl` or `Postman`.

```
curl -v http://127.0.0.1:8080/index.html
```

4. **Submission**:
   - Submit the source code and a brief README explaining how to run your server.