Assignment 10

CS 181AG: Network Algorithmics

Due: Dec 9, 2022 10pm PT

This assignment will guide you through setting up a server that your browser will connect to and display your html.

For this assignment, I encourage you to explore resources online for debugging tips. As far as the honor code, you may look at any resources that are helpful to you to complete the assignment, but you must write the code yourself. Additionally, please read through the entire set of instructions before starting.

Overview
In class, we set up a client and server that could communicate with each other. We then learned about http and the format of http messages. In this assignment, you will create only a server, and your browser will be the client that sends a request and expects a response in the correct format.

Your server will handle one request at at time. It will accept and parse an http request message, get the requested file from the server's file system (which you will create), create an http response consisting of the the initial line, header, and message, and send the response to the client (your browser). You will know that the server successfully did its job when your server contents appear in your browser. If the requested file is not found in your file system, your server will send a 404 Not Found message.

Step 1: Setting up your server

1. Using code similar to class, create a server that binds to an IP and port (use 127.0.0.1 because you will be running this locally). Also similar to class, your server should indefinitely be able to accept new connections, though only one at a time. You can expect the messages to be no more than 4096 bytes.
2. Put a simple html file, hello.html (it could just say "Hello World"), in the same folder as your server file.

Step 2: Running the client

1. While your server code is running, open a browser and provide the corresponding URL, for example, 127.0.0.1:65432/hello.html (note that the port must match the port at which your server is listening).
2. When you run this, think about what you expect the request to look like on the server side, i.e., the message format. Ensure that your server receives this request. It's okay if 1) it appears to send more than once 2) Your browser shows an error (it hasn't received anything back yet!)

Step 3: Generating a Server Response

1. First, observe the message received by the client and ensure that the request format is what you expected. In particular, does the initial line match the expected request format?
2. Next, you need to generate a server response. Before doing anything with the requested file, I recommend just sending a string as the response to ensure that the browser can recognize the format of your message. Recall that a message has the following format:

```
Initial line
Header
Header (cont. for any additional header lines)

Message
```

For a response, the initial line consists of three parts: http version (use 1.1), the status code, and extra information. For the header lines, the two important lines are as follows:

```
Content-Type:text/html (in this case, because we want it to be parsed as html)
Connection:close
```

The message can for now just be your sample string, which will show up in the browser if received correctly. **Note that the header begins on a new line, as does each subsequent header line. Also note that there is a blank line between the header and message.** All of the above can be written as one string, which must be encoded before you send. Ensure that your message is displayed at the browser.

3. Once you have sanity checked that your simple string displays correctly at the browser, let's instead send the contents of the requested file, if it exists. First, you need to get the path of the requested file. Hint: you know the format of the request's initial line.

   To check whether the file exists, you can encapsulate reading the contents with a try-except. Reading the contents can be done using something like the following:

   ```
   file = open(file_requested,'r')
   contents = file.read()
   file.close()
   ```

   This will only work if the name of requested file perfectly matches what is in your file system (you might need to strip extra characters). When sending the message, make sure you enter the correct status code based on whether the file was found or not (you only need to consider two cases - it was found and succesfully returned, or it was not found). If it was not found, you can simply send a message for the browser to display saying that the requested file was not found (use the same header lines).

What to Turn In

Please name your server file **assignment10_server.py** and turn it in on Gradescope. Please name the html file **hello.html**, though it can contain whatever contents you want (just don't leave it blank). Please use port **65432** when turning it in. I will check your work by running your server and requesting hello.html on my browser.