

371 - Mini Project Specification

Cameron Way & Melvin Zhang

CMPT 371 - Ouldooz

Dec 2022

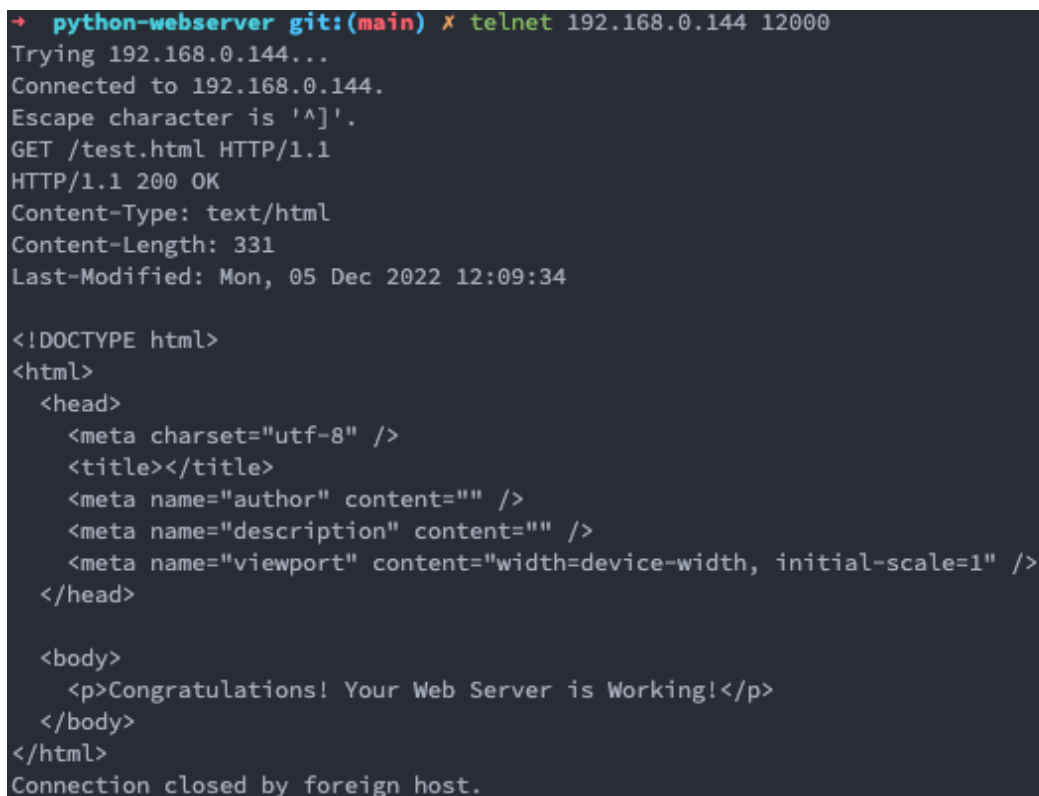
Table of Contents

Step 2: Web Server (Single-Threaded)	2
2.1 Testing Method	2
2.2 Additional Method	5
 Step 3: Web Proxy	 6
3.1 Design	6
3.2 Testing	6
 Step 4: Web Server (Multi-Threaded)	 8
4.1 Testing	8

Step 2: Web Server (Single-Threaded)

2.1 Testing Method

200 OK: To test this, we simply requested the test.html file that we knew was in the server's directory. We used telnet to connect to our server and sent a simple "GET /test.html HTTP/1.1" request. We were able to get the file. We also tested within a browser and were able to display the web page.



```
+ python-webserver git:(main) x telnet 192.168.0.144 12000
Trying 192.168.0.144...
Connected to 192.168.0.144.
Escape character is '^]'.
GET /test.html HTTP/1.1
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 331
Last-Modified: Mon, 05 Dec 2022 12:09:34

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title></title>
    <meta name="author" content="" />
    <meta name="description" content="" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
  </head>

  <body>
    <p>Congratulations! Your Web Server is Working!</p>
  </body>
</html>
Connection closed by foreign host.
```

Figure 1: Example of 200 OK

304 Not Modified: As seen in figure 2, we tested this method by sending a "If-Modified-Since" header with date/times before, after and equal to the modified date of the file. When the If-Modified-Since is older than the modified time of the file, the server returns a 200 OK with the file in its body. If the date in the header is equivalent, or newer than the modified time of the file, the server returns a 304 with an empty body.

```

[➔ ~ telnet 192.168.0.144 12000
Trying 192.168.0.144...
Connected to 192.168.0.144.
Escape character is '^]'.
GET /test.html HTTP/1.1
If-Modified-Since: Mon, 05 Dec 2022 12:08:34 PST

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 331
Last-Modified: Mon, 05 Dec 2022 12:09:34

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title></title>
    <meta name="author" content="" />
    <meta name="description" content="" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
  </head>

  <body>
    <p>Congratulations! Your Web Server is Working!</p>
  </body>
</html>
Connection closed by foreign host.
[➔ ~ telnet 192.168.0.144 12000
Trying 192.168.0.144...
Connected to 192.168.0.144.
Escape character is '^]'.
GET /test.html HTTP/1.1
If-Modified-Since: Mon, 05 Dec 2022 12:09:34 PST

HTTP/1.1 304 Not Modified
Content-Type: text/html
Content-Length: 0
Last-Modified: Mon, 05 Dec 2022 12:09:34

Connection closed by foreign host.
[➔ ~ telnet 192.168.0.144 12000
Trying 192.168.0.144...
Connected to 192.168.0.144.
Escape character is '^]'.
GET /test.html HTTP/1.1
If-Modified-Since: Mon, 05 Dec 2022 12:10:34 PST

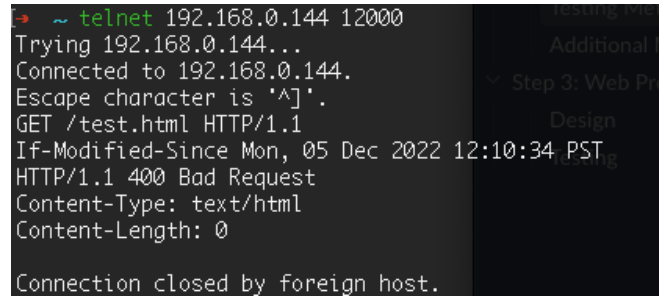
HTTP/1.1 304 Not Modified
Content-Type: text/html
Content-Length: 0
Last-Modified: Mon, 05 Dec 2022 12:09:34

Connection closed by foreign host.

```

Figure 2: Example of 304 Not Modified

400 Bad Request: If the request is malformed, we return a 400 bad request. We detect this if an error occurs while parsing the request headers. In this example, the header is missing the colon, therefore we return a 400 bad request.



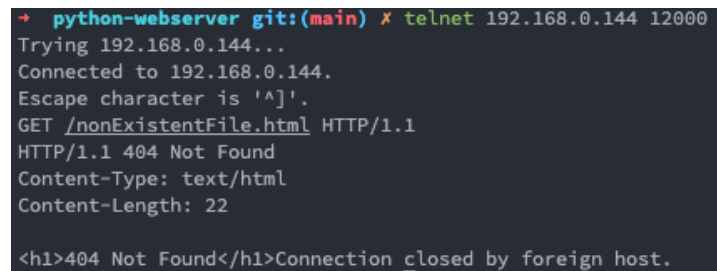
```

➔ ~ telnet 192.168.0.144 12000
Trying 192.168.0.144...
Connected to 192.168.0.144.
Escape character is '^]'.
GET /test.html HTTP/1.1
If-Modified-Since Mon, 05 Dec 2022 12:10:34 PST
HTTP/1.1 400 Bad Request
Content-Type: text/html
Content-Length: 0
Connection closed by foreign host.

```

Figure 3: Example of 400 Bad Request

404 Not Found: We tested this by simply requesting a file that does not exist in the directory. When a 404 error occurs we decided to return a HTML object to display to the client as well.



```

➔ python-webserver git:(main) ✕ telnet 192.168.0.144 12000
Trying 192.168.0.144...
Connected to 192.168.0.144.
Escape character is '^]'.
GET /nonExistentFile.html HTTP/1.1
HTTP/1.1 404 Not Found
Content-Type: text/html
Content-Length: 22
<h1>404 Not Found</h1>Connection closed by foreign host.

```

Figure 4: Example of 404 Not Found

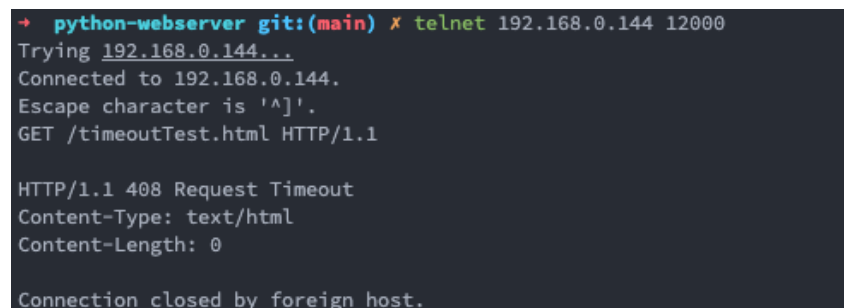
408 Request Timed Out: To test the request that times out, we have added the following code to our server:

```

if path == "timeoutTest.html":
    time.sleep(20) # simulate a request that times out
    path = "test.html"

```

When the client requests timeoutTest.html, we are waiting for longer than the timeout to simulate a request that is timing out. Our server then hits its timeout and responds with a 408. As seen in figure 5, that is exactly what happens.



```

➔ python-webserver git:(main) ✕ telnet 192.168.0.144 12000
Trying 192.168.0.144...
Connected to 192.168.0.144.
Escape character is '^]'.
GET /timeoutTest.html HTTP/1.1

HTTP/1.1 408 Request Timeout
Content-Type: text/html
Content-Length: 0
Connection closed by foreign host.

```

Figure 5: Example of 408 Request Timed Out

2.2 Additional Method

405 Method Not Allowed: We included this to just ensure that the server is receiving GET requests, as that is what it is limited to handling.

500 Internal Server Error: Included because if an error occurs, we do not want the server to go down. We will catch the error without causing the program to crash. In this case we just log the error, but in production one might notify an admin.

Step 3: Web Proxy

3.1 Design

A web proxy is typically used within a local network, and therefore much closer to the client than the original server. However, a web proxy does not have access to the files on the server, so when a request for a file comes in, it must send a conditional get to the server to ensure it serves the most up to date file. If the conditional get returns a 200 OK, the web proxy knows the file must have been updated, so it will store a copy locally, and serve the file to the client. If the conditional get returns a 304 Not Modified however, it knows that it has the most up to date version, and therefore serves the copy out of its cache.

In our implementation of our web proxy, the files are stored in memory (in a Dict). This limits the number of files that can be stored, but is faster than writing them on disk. When the proxy receives a request, it first checks if there is a local copy. If there is, it sends a conditional get. It does so by including the "If-Modified-Since" header, which tells the server only to send the file if it has been modified.

3.2 Testing

```
+ python-webserver git:(main) ✗ python3 WebProxyServer.py
Getting local IP address
Local IP address is 192.168.0.144
Listening on 192.168.0.144, port 12001
LOG - Thread: 1542797 - Handling GET request on path /test.html
LOG - Thread: 1542797 - File is not memory
LOG - Thread: 1542797 - Received file from server, storing file in memory
LOG - Thread: 1542797 - Sending response to client

LOG - Thread: 1542797 - Handling GET request on path /test.html
LOG - Thread: 1542797 - File is in memory
LOG - Thread: 1542797 - Sending conditional GET request
LOG - Thread: 1542797 - File not modified, sending file from memory
LOG - Thread: 1542797 - Sending response to client

LOG - Thread: 1542797 - Handling GET request on path /test.html
LOG - Thread: 1542797 - File is in memory
LOG - Thread: 1542797 - Sending conditional GET request
LOG - Thread: 1542797 - Received file from server, storing file in memory
LOG - Thread: 1542797 - Sending response to client

LOG - Thread: 1542797 - Handling GET request on path /test.html
LOG - Thread: 1542797 - File is in memory
LOG - Thread: 1542797 - Sending conditional GET request
LOG - Thread: 1542797 - File not modified, sending file from memory
LOG - Thread: 1542797 - Sending response to client
```

Figure 6: Proxy Log Example

To test the web proxy, the client would need to set up the web proxy in their network settings. For our testing purposes, we will just hit the web proxy's IP and port, as if it were the original server. We have added logs to the web proxy so we can see the conditional get in action. As you can see in figure 6, when we received the first request, the file was not in memory so we requested it from the server. We then stored it in memory and sent the file to the client. On the next request, the file was in memory so we sent a conditional get. The conditional get returned a 304, therefore we sent the file to the client from memory.

We then needed to ensure that if the file was modified the server would correctly send a 200 OK, and the web proxy would update its local copy and send it to the client. In the third request, you can see that is exactly what happens. The server sends the file, therefore it must have been updated, the proxy updates its copy, and sends it to the client. In the fourth request, you can see that it once again has the most up to date version and serves the file from the proxy's memory.

Step 4: Web Server (Multi-Threaded)

4.1 Testing

As you can see in Figure 6, the server handles three connections concurrently. The main thread listens on port 80, and the responses are sent from separate ports to improve performance. To test the multithreaded connectivity, we wrote a script (TestClient.py) to open three TCP connections and request documents.

```
→ python-webserver git:(main) ✗ pls python3 MultiThreadedServer.py
Getting local IP address
Local IP address is 142.58.63.109
Listening on 142.58.63.109, port 80
Test file: http://142.58.63.109:80/test.html
LOG - Thread: 1861383 - Handling new connection on port 55379
LOG - Thread: 1861403 - Handling new connection on port 55380
LOG - Thread: 1861423 - Handling new connection on port 55381
LOG - Thread: 1861383 - Handling GET request on path /largeTest.html
LOG - Thread: 1861383 - Responding with status code 200 from port 55379
LOG - Thread: 1861383 - Closing thread 1861383
LOG - Thread: 1861423 - Handling GET request on path /test.html
LOG - Thread: 1861423 - Responding with status code 200 from port 55381
LOG - Thread: 1861423 - Closing thread 1861423
LOG - Thread: 1861403 - Handling GET request on path /largeTest.html
LOG - Thread: 1861403 - Responding with status code 200 from port 55380
LOG - Thread: 1861403 - Closing thread 1861403
```

Figure 6: Example of multithreaded connections