CS536: Networking Lab 1

Name: Mansi Shinde

PUID: **0034784153**

Part A. Sol. Building Client Server over Socket Programming

- Running Client and Server Application on localhost.

  Code files to refer - client.c, server.c

  Steps to execute the code:

  1. Creating executable file using gcc command

  2. Running the executable file

  Client Side:

  1. gcc client.c -o ./client
  2. ./client 127.0.0.1 3500 “Testing Message”

  Server Side:

  1. gcc server.c -o ./server
  2. ./server 3500

Client Response:

```
(base) mansishinde@Mansis-MacBook-Air PartA %
(base) mansishinde@Mansis-MacBook-Air PartA % gcc client.c -o ./client
(base) mansishinde@Mansis-MacBook-Air PartA %
(base) mansishinde@Mansis-MacBook-Air PartA %
(base) mansishinde@Mansis-MacBook-Air PartA % ./client 127.0.0.1 3500 "Hello! this is Client talking from localhost"
Connection established with Server

Sending the message to the server.

Modified message received from server:
HELLO! THIS IS CLIENT TALKING FROM LOCALHOST
Closing the connection
(base) mansishinde@Mansis-MacBook-Air PartA %
```

Server response:

```
(base) mansishinde@Mansis-MacBook-Air PartA %
(base) mansishinde@Mansis-MacBook-Air PartA % gcc server.c -o ./server
(base) mansishinde@Mansis-MacBook-Air PartA %
(base) mansishinde@Mansis-MacBook-Air PartA %
(base) mansishinde@Mansis-MacBook-Air PartA % ./server 3500
The server is ready to receive

message-from-client : 127.0.0.1 , 64924

Message: Hello! this is Client talking from localhost

Modified Message: HELLO! THIS IS CLIENT TALKING FROM LOCALHOST

close-client : 127.0.0.1 , 64924
```

- Running Client and Server Application on different host machines

  Client is running on amber10.cs.purdue.edu whose **IP address is 128.10.112.140** and **Server** is running on amber20.cs.purdue.edu whose **IP address is 128.10.112.150**

  Code files to refer - client.c, server.c

  Steps to execute the code:

  1. Creating executable file using gcc command

  2. Running the executable file

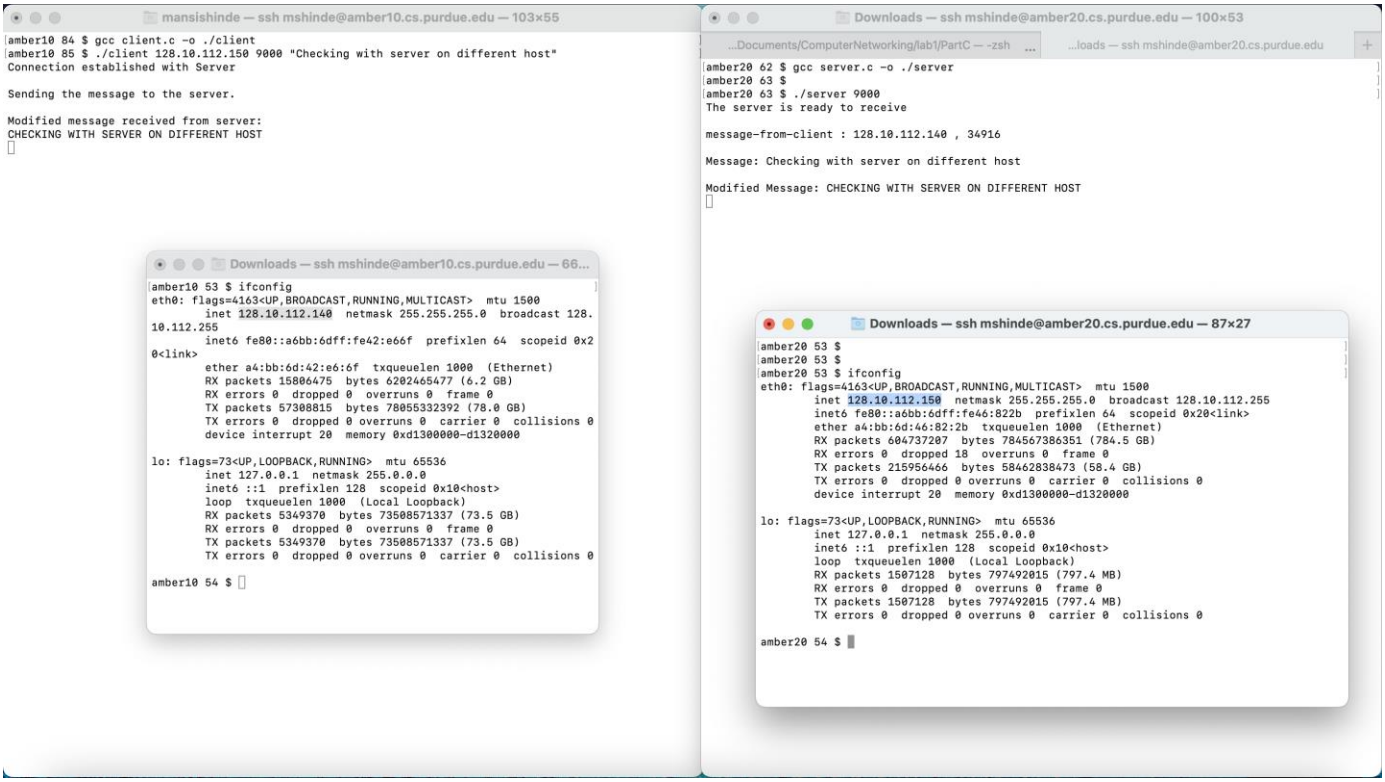
  Client Side:

  1. gcc client.c -o ./client
  2. ./client 128.10.112.150 9000 “Testing Message”

  Server Side:

  1. gcc server.c -o ./server
  2. ./server 9000

Output:



Handling Multiple Clients using pthread library

- Running Multiple clients on separate terminals on **localhost** and handling them using pthread.

  Code files to refer - client.c, serverMul.c

  Steps to execute the code:

  1. Creating executable file using gcc command

  2. Running the executable file. While running the server executable file, we have to pass the argument “lpthread” to notify the server that multiple clients will be connecting during the session.

  Client Side:

  1. gcc client.c -o ./client
  2. ./client 127.0.0.1 6000 “Testing Message”

  Server Side:

  1. gcc serverMul.c -lpthread -o serverMul
  2. ./serverMul 6000

Output on serverMul side:

```
(base) mansishinde@Mansis-MacBook-Air PartA %
(base) mansishinde@Mansis-MacBook-Air PartA % ./serverMul 6000
The server is ready to receive


message-from-client : 127.0.0.1 , 64964
Message: Hello! this is Client talking from localhost 1
Modified Message: HELLO! THIS IS CLIENT TALKING FROM LOCALHOST 1

message-from-client : 127.0.0.1 , 64967
Message: Hello! this is Client talking from localhost 2
Modified Message: HELLO! THIS IS CLIENT TALKING FROM LOCALHOST 2

message-from-client : 127.0.0.1 , 64968
Message: Hello! this is Client talking from localhost 3
Modified Message: HELLO! THIS IS CLIENT TALKING FROM LOCALHOST 3

close-client : 127.0.0.1 , 64964

message-from-client : 127.0.0.1 , 64973
Message: Hello! this is Client talking from localhost 4
Modified Message: HELLO! THIS IS CLIENT TALKING FROM LOCALHOST 4

close-client : 127.0.0.1 , 64967

close-client : 127.0.0.1 , 64968
```

Output of client 1:

```
(base) mansishinde@Mansis-MacBook-Air PartA %
(base) mansishinde@Mansis-MacBook-Air PartA % ./client 127.0.0.1 6000 "Hello! this is Client talking from localhost 4"
Connection established with Server

Sending the message to the server.

Modified message received from server:
HELLO! THIS IS CLIENT TALKING FROM LOCALHOST 4
```

Output of client 2:

```
(base) mansishinde@Mansis-MacBook-Air PartA %
(base) mansishinde@Mansis-MacBook-Air PartA % ./client 127.0.0.1 6000 "Hello! this is Client talking from localhost 3"
Connection established with Server

Sending the message to the server.

Modified message received from server:
HELLO! THIS IS CLIENT TALKING FROM LOCALHOST 3
Closing the connection
(base) mansishinde@Mansis-MacBook-Air PartA %
```

Output of client 3:

```
(base) mansishinde@Mansis-MacBook-Air PartA % ./client 127.0.0.1 6000 "Hello! this is Client talking from localhost 2"
Connection established with Server

Sending the message to the server.

Modified message received from server:
HELLO! THIS IS CLIENT TALKING FROM LOCALHOST 2
Closing the connection
(base) mansishinde@Mansis-MacBook-Air PartA %
```

- Running Multiple clients on different host machines and handling them using pthread. Here, the server IP address taken is 128.10.112.132

  Code files to refer - client.c, serverMul.c

  Steps to execute the code:

  1. Creating executable file using gcc command

  2. Running the executable file. While running the server executable file, we have to pass the argument “lpthread” to notify the server that multiple clients will be connecting during the session.

  Client Side:

  1. gcc client.c -o ./client
  2. ./client 128.10.112.132 4700 “Testing Message”

  Server Side:

  1. gcc serverMul.c -lpthread -o serverMul
  2. ./serverMul 4700

Output of Client on amber06 machine:

```
amber06 55 $
amber06 55 $ ./client 128.10.112.132 4700 "This is the testing message from amber06 machine"
Connection established with Server

Sending the messageTo the server.

Modified sentence received from server:
THIS IS THE TESTING MESSAGE FROM AMBER06 MACHINE
```

Output of Client on amber01 machine:

```
amber01 52 $
amber01 52 $ ./client 128.10.112.132 4700 "This is the testing message from amber01 machine"
Connection established with Server

Sending the messageTo the server.

Modified sentence received from server:
THIS IS THE TESTING MESSAGE FROM AMBER01 MACHINE
```

Output of Client on amber04 machine:

```
amber04 52 $ ./client 128.10.112.132 4700 "This is the testing message from amber04 machine"
Connection established with Server

Sending the messageTo the server.

Modified sentence received from server:
THIS IS THE TESTING MESSAGE FROM AMBER04 MACHINE
Closing the connection
amber04 53 $
```

Output of Client on amber05 machine:

```
amber05 53 $ ./client 128.10.112.132 4700 "This is the testing message from amber05 machine"
Connection established with Server

Sending the messageTo the server.

Modified sentence received from server:
THIS IS THE TESTING MESSAGE FROM AMBER05 MACHINE

Closing the connection
amber05 54 $
```

Output on Server side:

```
amber02 57 $ gcc serverMul.c -lpthread -o serverMul
amber02 58 $
amber02 58 $ ./serverMul 4700
The server is ready to receive


message-from-client : 128.10.112.136 , 40442
Message: This is the testing message from amber06 machine
Modified Message: THIS IS THE TESTING MESSAGE FROM AMBER06 MACHINE

message-from-client : 128.10.112.135 , 58282
Message: This is the testing message from amber05 machine
Modified Message: THIS IS THE TESTING MESSAGE FROM AMBER05 MACHINE

message-from-client : 128.10.112.134 , 51804
Message: This is the testing message from amber04 machine
Modified Message: THIS IS THE TESTING MESSAGE FROM AMBER04 MACHINE

message-from-client : 128.10.112.131 , 50760
Message: This is the testing message from amber01 machine
Modified Message: THIS IS THE TESTING MESSAGE FROM AMBER01 MACHINE

close-client : 128.10.112.136 , 40442

message-from-client : 128.10.112.136 , 40444
Message: This is the testing message from amber06 machine
Modified Message: THIS IS THE TESTING MESSAGE FROM AMBER06 MACHINE

close-client : 128.10.112.135 , 58282
```

Part B. Sol. Develop Web Server over HTTP/1.1

- Running text.html, picture.html, bigpicture.html on various browsers (have tested on Chrome, Mozilla Firefox, Safari)

  Code files to refer - client1.c, server1.c

  Steps to execute the code:

  1. Creating server executable file using gcc command

  2. Running the server executable file. While running the server executable file, we must pass the argument "lpthread" to notify the server that multiple clients will be connecting during the session.

3. Run http commands on browser, for example: http://server-ip-address:server-port-no/filename. These commands will act as input for server1.c, where server will respond with 200, 400, 404 and 505 status codes.

Server Side:

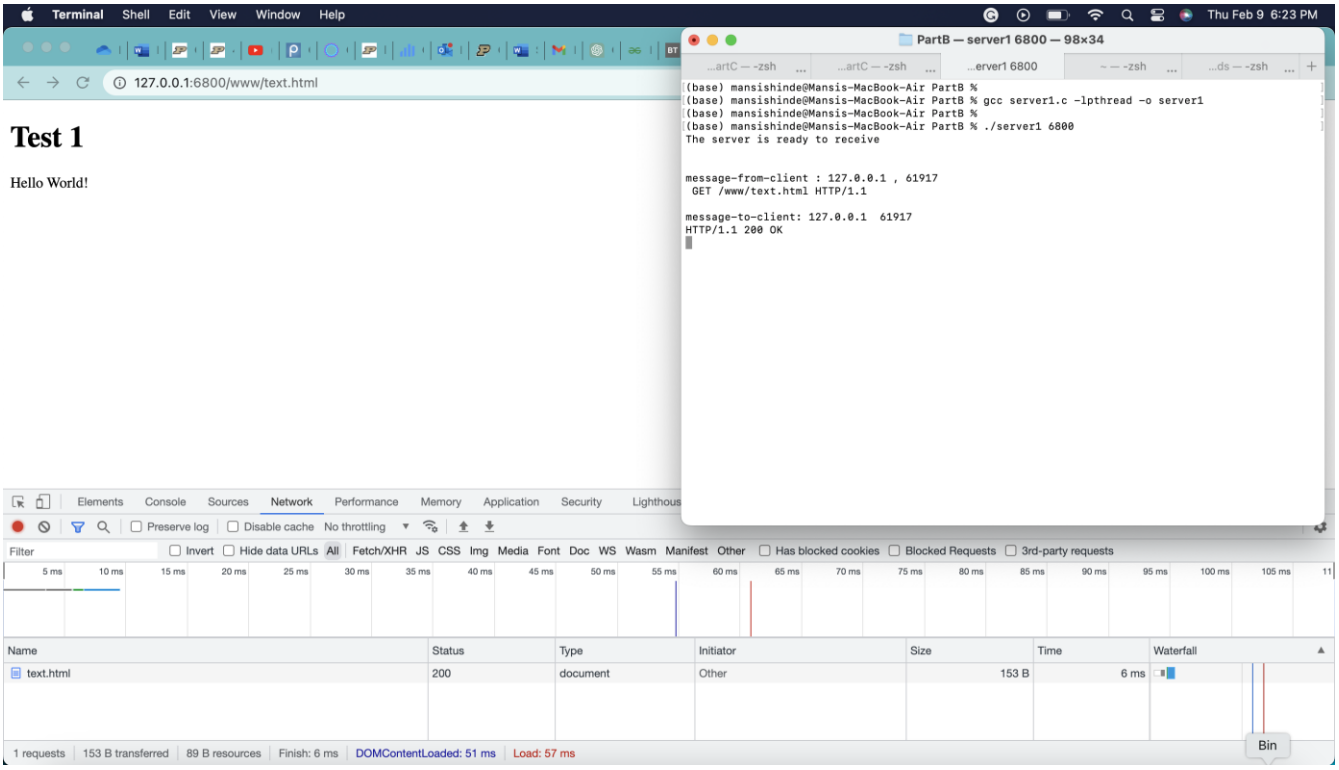
1. gcc server1.c -lpthread -o server1
2. ./server1 6800

Now, the client will start listening to incoming requests.

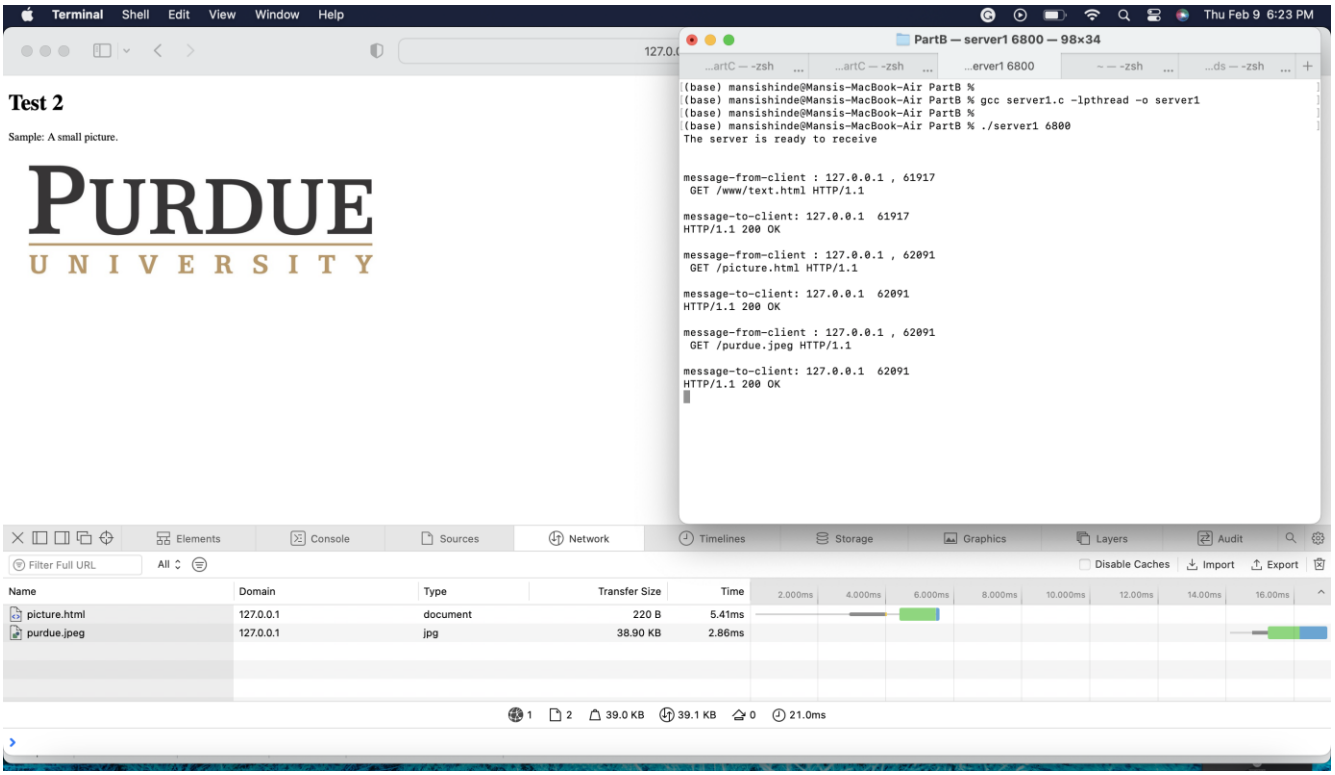
On Browser:

1. http://127.0.0.1:6800/text.html

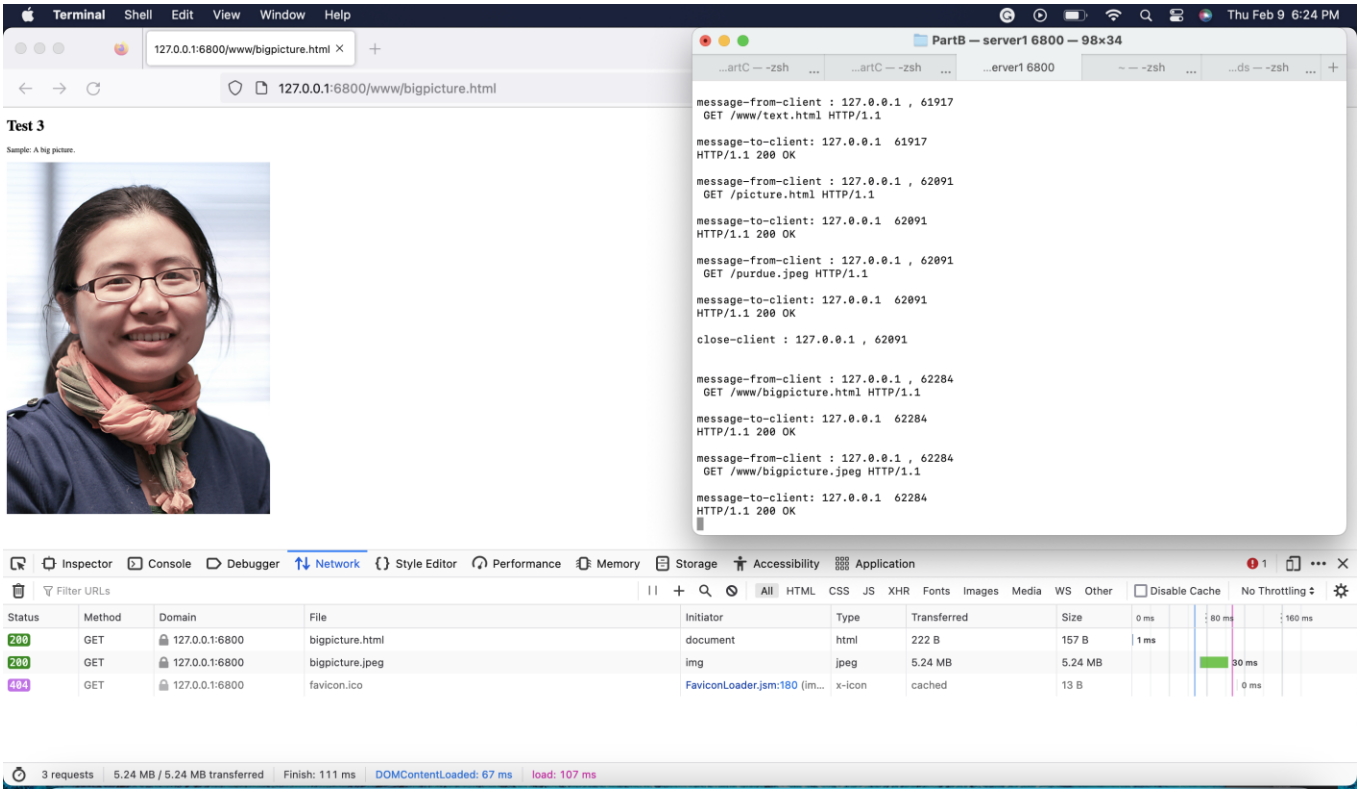
Browser executes http://127.0.0.1:6800/www/text.html , for which it gets HTTP/1.1 200 OK as a response. This http link has been run on Chrome Web Browser.



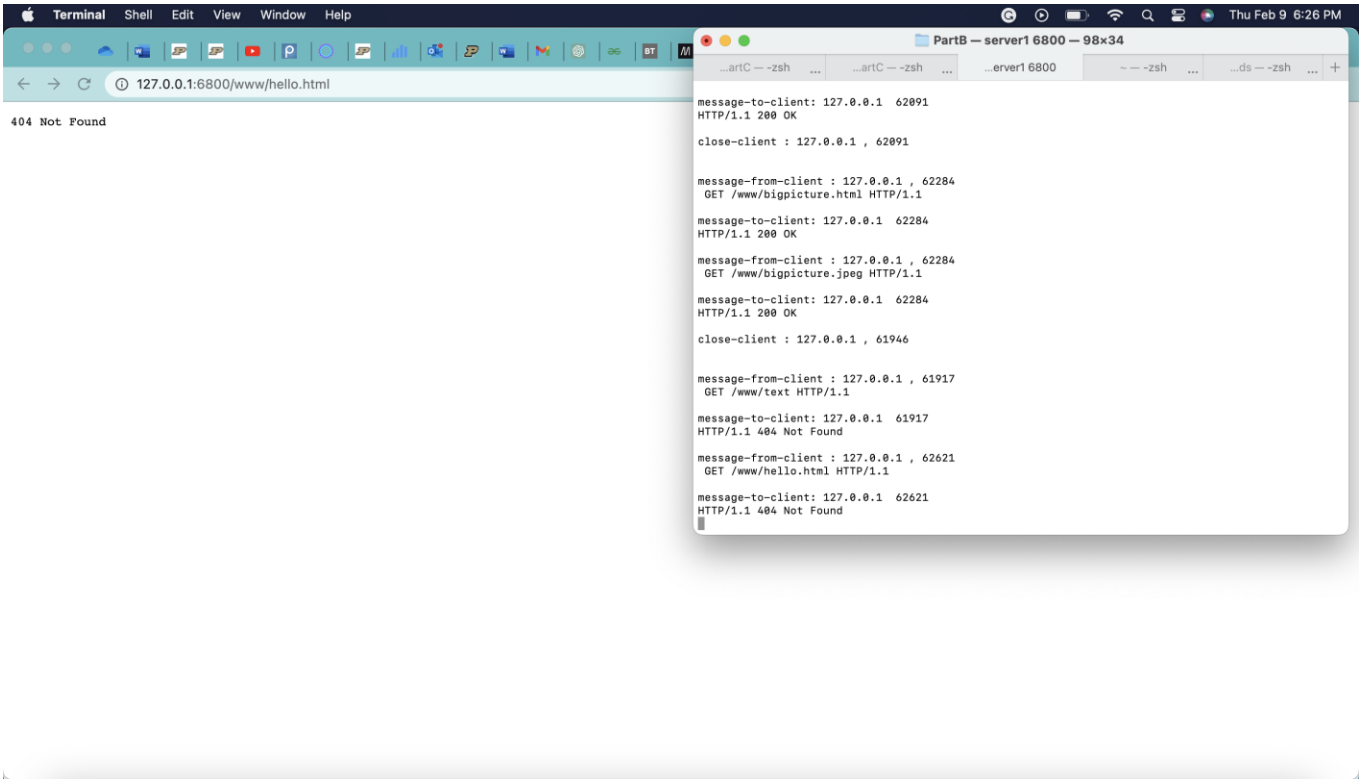
Browser executes http://127.0.0.1:6800/picture.html , for which it gets HTTP/1.1 200 OK as a response. This http link has been run on Safari Web Browser.



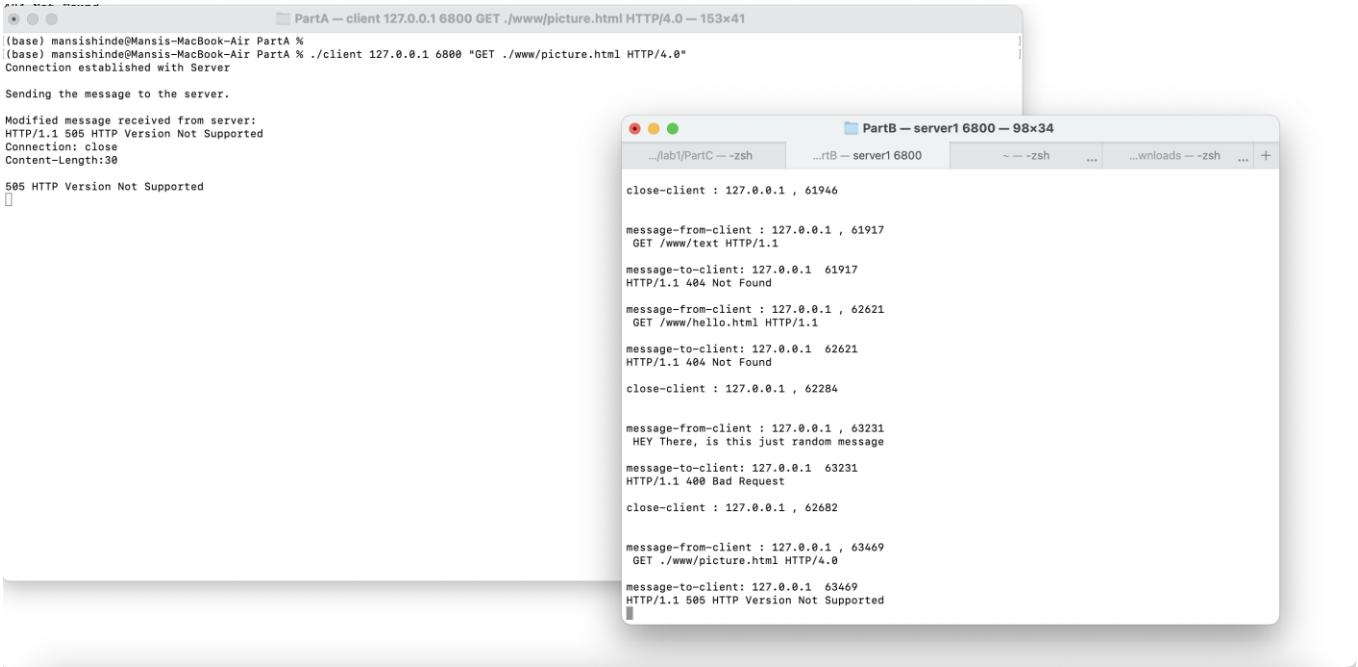
Browser executes http://127.0.0.1:6800/www/bigpicture.html , for which it gets HTTP/1.1 200 OK as a response. This http link has been run on Mozilla Firefox Web Browser.



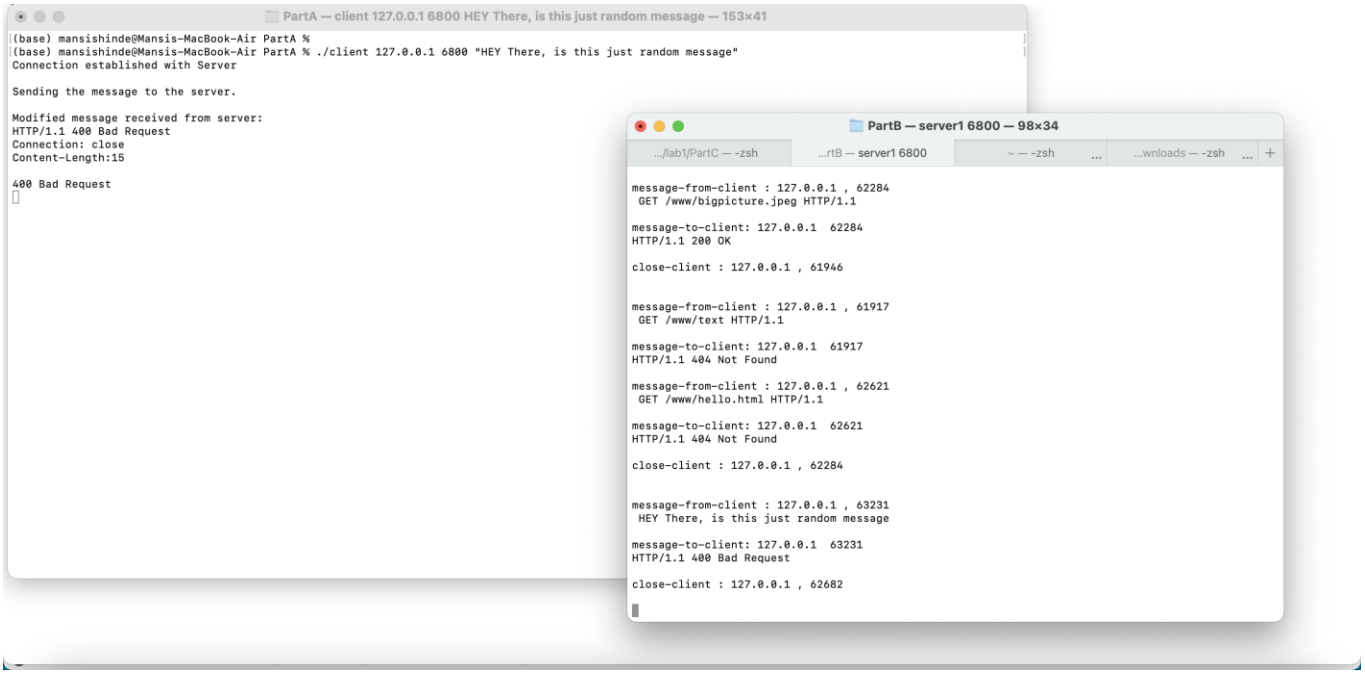
Example of http request and response with status code “**404 Not Found**” as it is unable to find the file hello.html



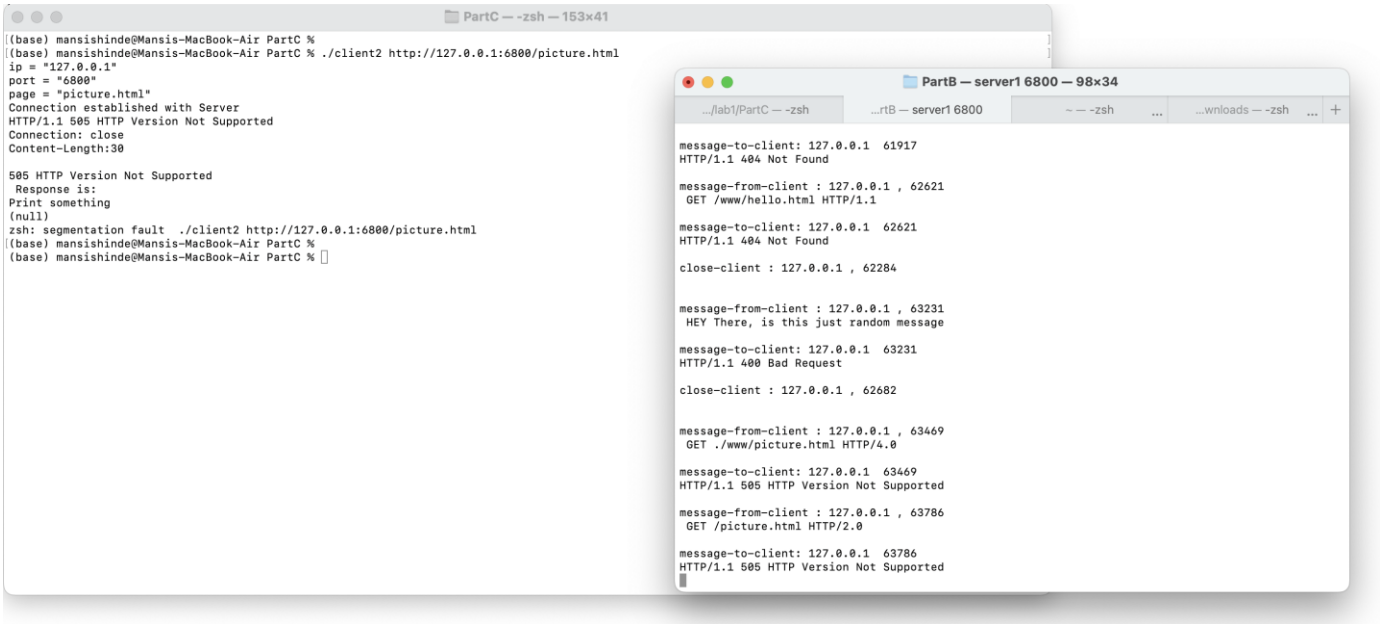
Example of http request and response with status code “**505 HTTP Version Not Supported**” as the server is running on HTTP/1.1 version whereas it is getting a response from a version which is not supported by the HTTP.



Example of http request and response with status code “**400 Bad Request**” as it is unable to get the request Method (like GET, PUT etc.), version (like HTTP/1.1, HTTP/2.0 etc.) and protocol which is required for getting HTTP response.



Example of http request and response with status code “505 HTTP Version Not Supported” as the server is running on HTTP/1.1 but it is getting the request from the client which is running on HTTP/2.0.



Part C. Sol. Develop Web Server over HTTP/2.0

- Executing Server and Client on Localhost over HTTP/1.1

  Code files to refer - client2.c, server2.c

  Steps to execute the code:

  1. Creating client and server executable files using gcc command

  2. Running executable files

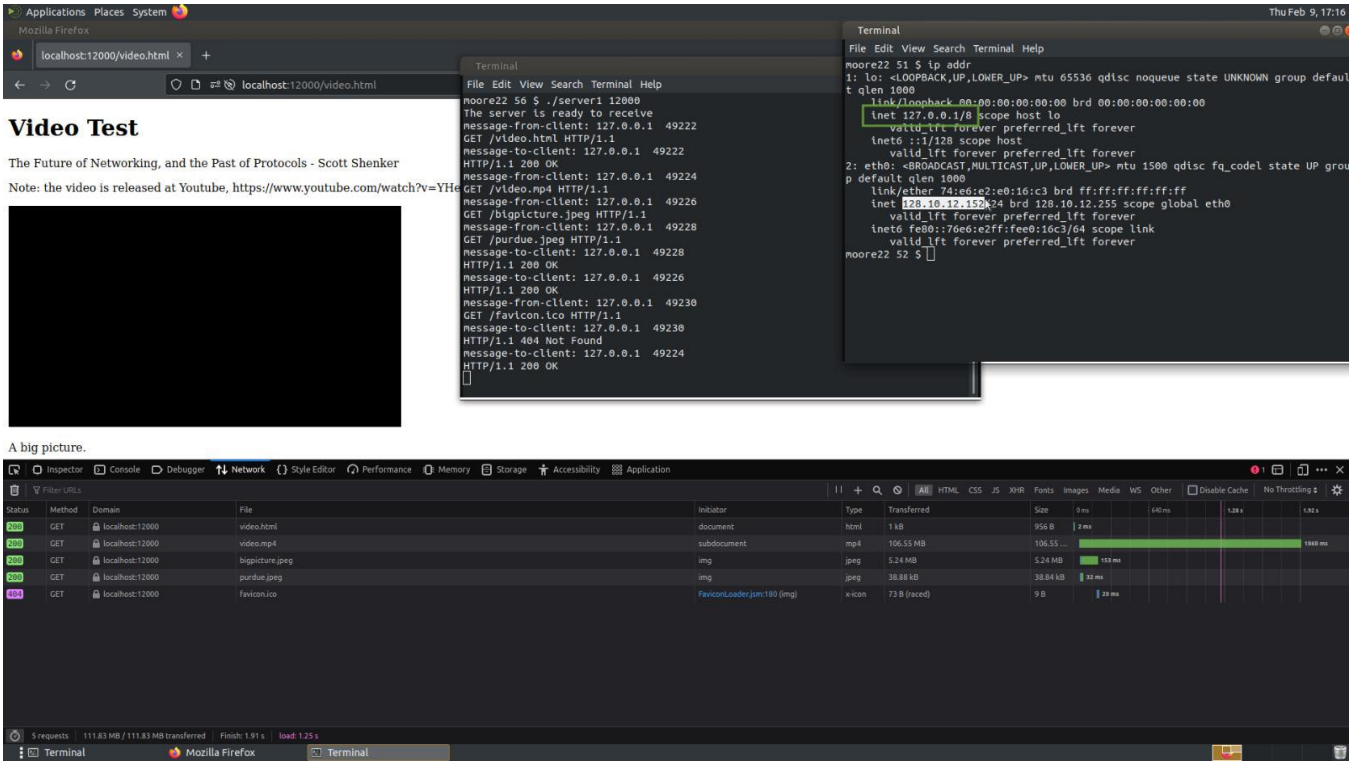
  Server Side:

  1. gcc server2.c -lpthread -o server2
  2. ./server2 12000

Now, the client will start listening to incoming requests.

On Browser:

1. http://127.0.0.1:12000/video.html



Executing web server over HTTP/1.1 took about **2.05 (2 ms + 153 ms+ 32 ms+ 1868 ms) seconds** to render all 4 objects and display on the browser.

- Executing Server and Client on Localhost over HTTP/2.0

  Steps to Execute:

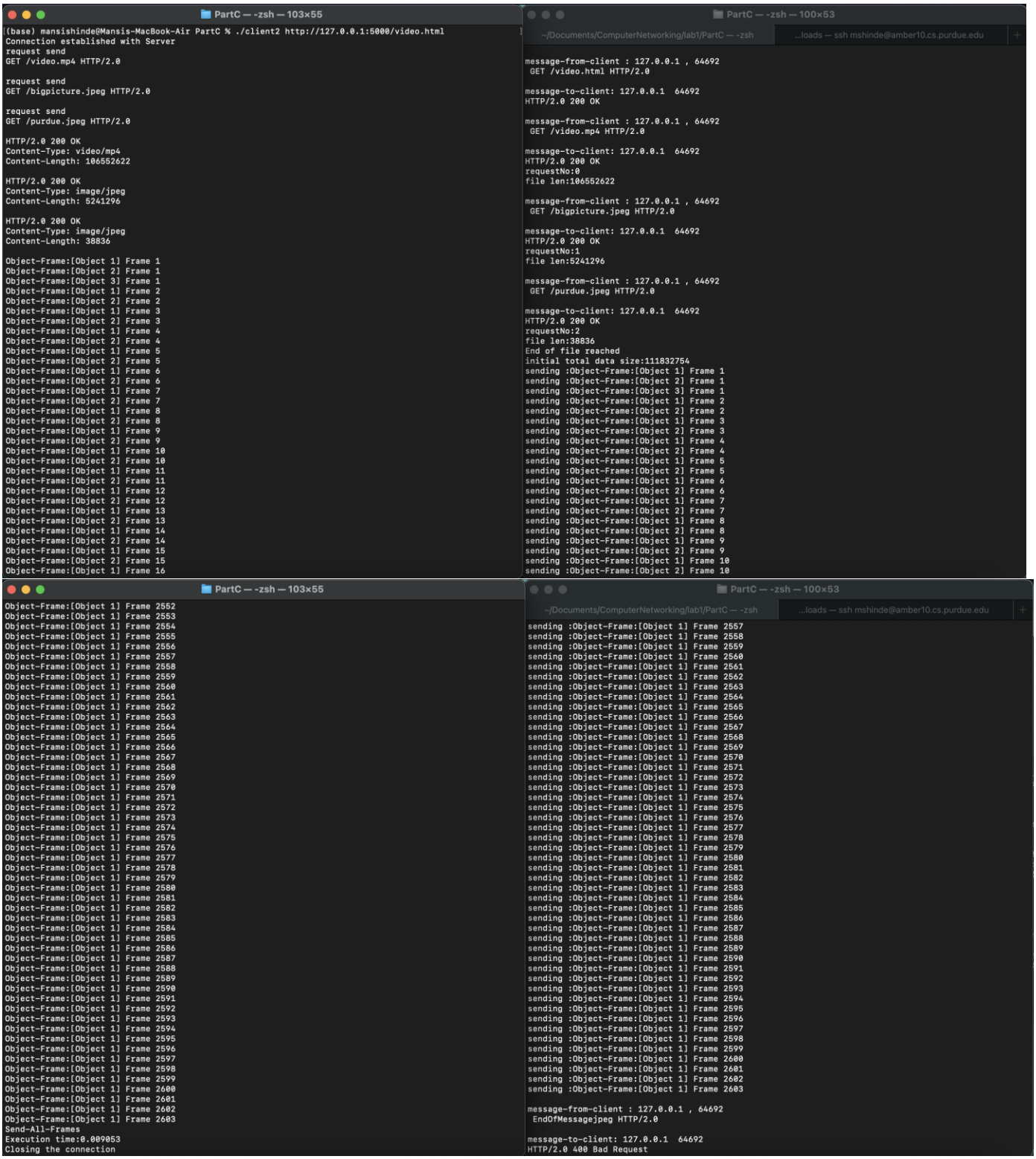
  Server Side:

  1. gcc server2.c -lpthread -o server2
  2. ./server2 5000

Now, the client will start listening to incoming requests.

  Client Side:

  1. gcc client2.c -o ./client2
  2. ./client2 http://127.0.0.1:5000/video.html



Whereas, executing web server over HTTP/2.0 took about **0.009 seconds** to render all 4 objects.

We see a huge difference between the performance of HTTP 1.1 and 2.0 due to the following reason:

HTTP 2.0 supports multiple requests at the same time over a single connection, in our case, multiple GET requests for objects video.mp4, picture.jpeg, bigpicture.jpeg are send to the server at the same time over single connection, whereas in HTTP 1.1 only one request is handled at a time, which leads to better performance.

Therefore, HTTP 2.0 provides several performance improvements over HTTP 1.1, including faster data transfer, reduced latency etc.

- Executing Server and Client on separate hosts over HTTP/1.1

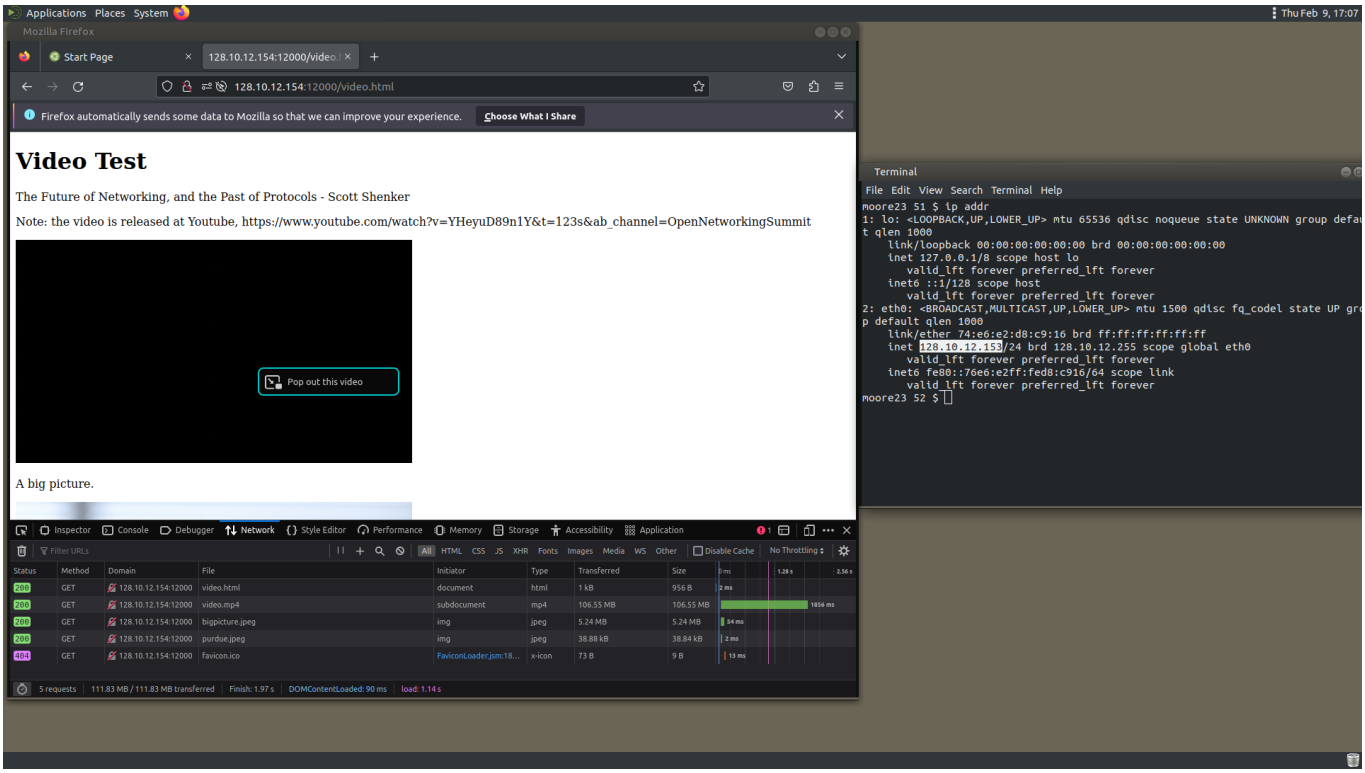
  Server Side:

  3. gcc server2.c -lpthread -o server2
  4. ./server2 12000

Now, the client will start listening to incoming requests.

On Browser:

1. http://128.10.12.154:12000/video.html



Executing web server over HTTP/1.1 over different hosts (here server is running on 128.10.12.154 and browser on 128.10.12.153), took about 1.9 **(1856 ms + 2 ms + 2 ms + 54 ms) seconds** to render all 4 objects and display on the browser.

- Executing Server and Client on separate hosts over HTTP/2.0

  Steps to Execute:

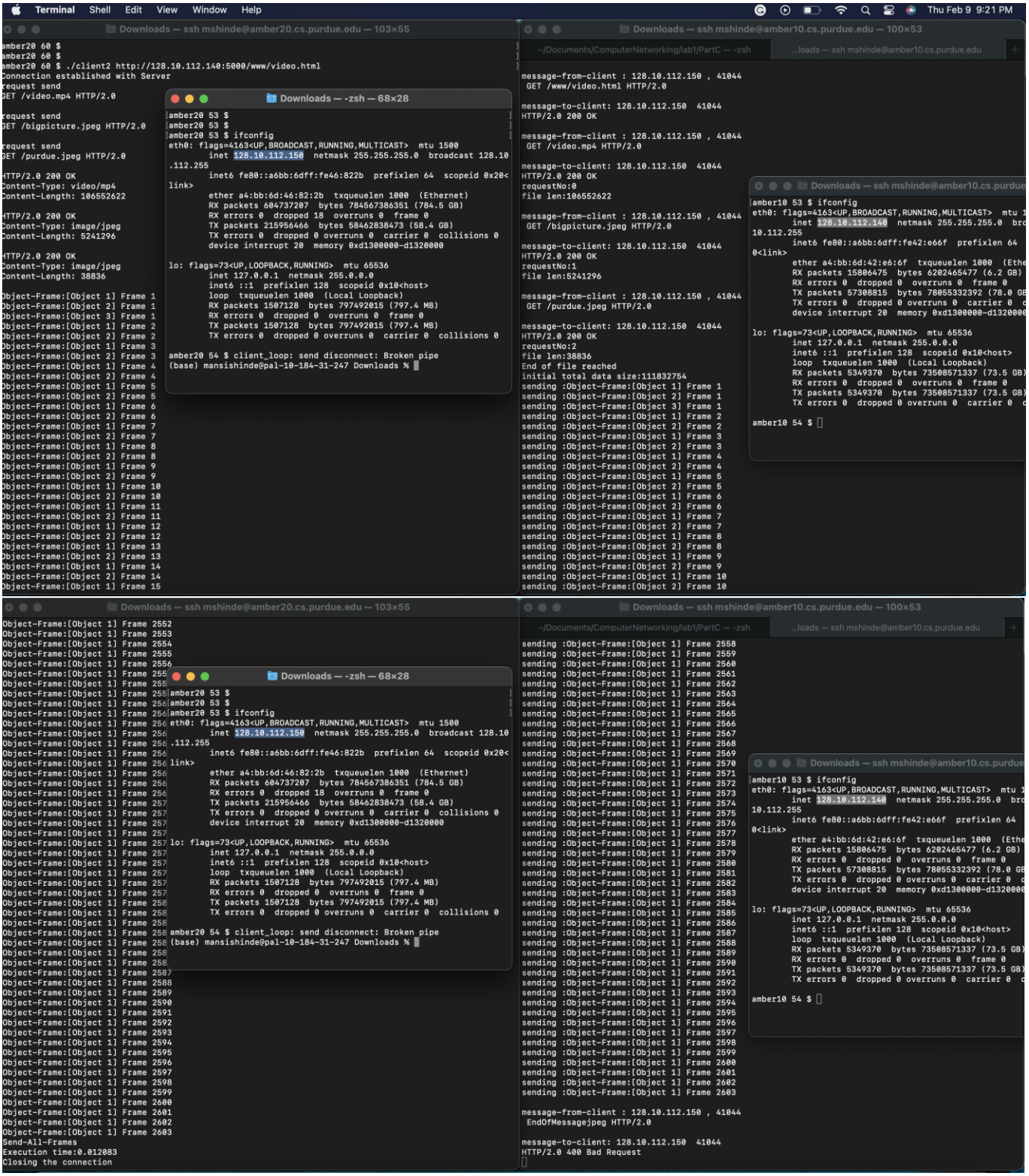
  Server Side:

  3. gcc server2.c -lpthread -o server2
  4. ./server2 5000

Now, the client will start listening to incoming requests.

  Client Side:

  3. gcc client2.c -o ./client2
  4. ./client2 http://128.10.112.150:5000/video.html



Whereas, executing web server over HTTP/2.0 took about 0.012 seconds to render all 4 objects.

We see that, executing client server on different host machines takes a little more time as compared to running it on localhosts because:

When the client and server are running on the same machine, network latency and overhead are reduced as the data does not have to travel over the network. These results improve performance for both HTTP/1.1 and HTTP/2.0.

When client and server are running on different hosts, network latency and overhead increase as data must travel over the network. In this scenario, performance benefits of HTTP 2.0 as seen in above case can help reduce the impact of latency and improve overall performance compared to HTTP 1.1