Project Report

HTTP Client and Server

-Kameswar Chembrolu (801033360)
-Viswajith Govinda Rajan (801022159)

What is TCP?

TCP stands for Transmission Control Protocol. It offers reliable transportation between sending and receiving data, flow and congestion control to ensure that the server will not overload the client. Also, it is connection oriented, meaning it first initiates a connection between the client and server to ensure the reliability of data transfer.

What is HTTP?

HTTP stands for HyperText Transfer Protocol. It is an application layer protocol for the internet. HTTP utilises the Client/Server model, which means that the key components of the model are the server, which is the central hub that provides functionalities to the clients, which are the users.

A client is like a browser that requests the server to either send or save some data. The server sends the data requested by the client, both these communications utilize the HTTP protocol.

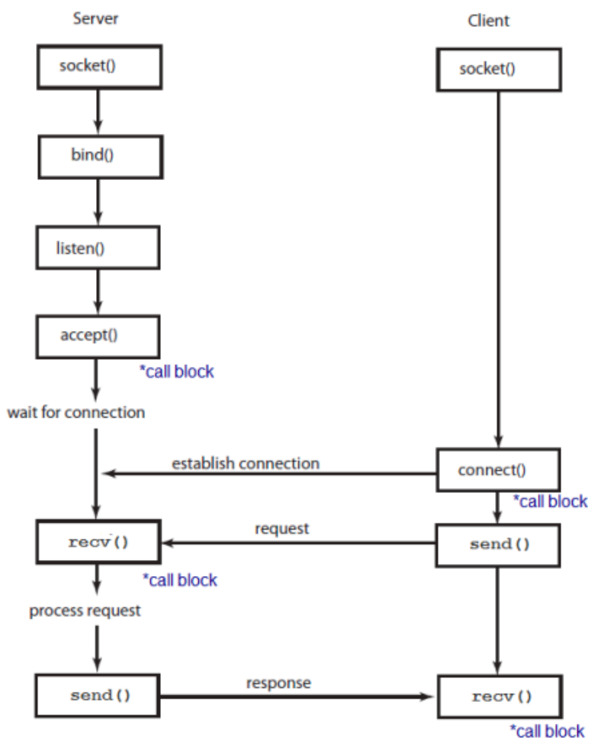


Fig: TCP/IP protocol Architecture using HTTP model

In the above figure, we can see that the server creates a socket first, which is then bound by the IP address and the port number so that the clients can locate it. The server then listens to the channel continuously to check whether the clients are requesting anything. When a client wants to initiate the communication with the server, it sends a request to establish a channel between the both of them. If the server accepts the connection, the client sends a request to the server to send some data. The server then processes the request and fetches the data which is then sent to the client as a response.

HTTP Client:

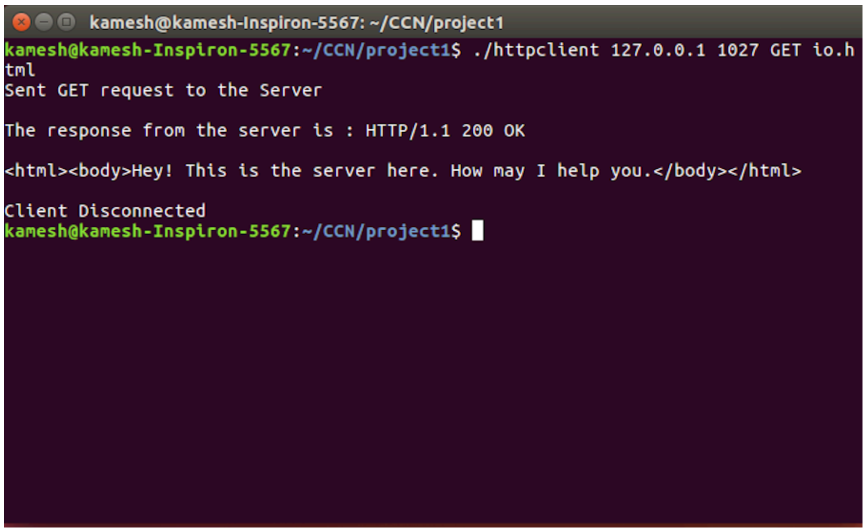
The client that we have programmed takes the server name, and port in this order from the command line to identify the server. We named our client httpclient and the server httpserver.

GET:

When a client requests the server to send it data, we invoke this command in the command line:

./httpclient 127.0.0.1 1027 GET io.html

When we invoke this command, the client connects to the server via a TCP connection, and submits an HTTP/1.1 GET request to the server. In return, the server responds with the status code and the file in question.



```
kamesh@kamesh-Inspiron-5567: ~/CCN/project1
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 GET io.h
tml
Sent GET request to the Server

The response from the server is : HTTP/1.1 200 OK

<html><body>Hey! This is the server here. How may I help you.</body></html>

Client Disconnected
kamesh@kamesh-Inspiron-5567:~/CCN/project1$
```

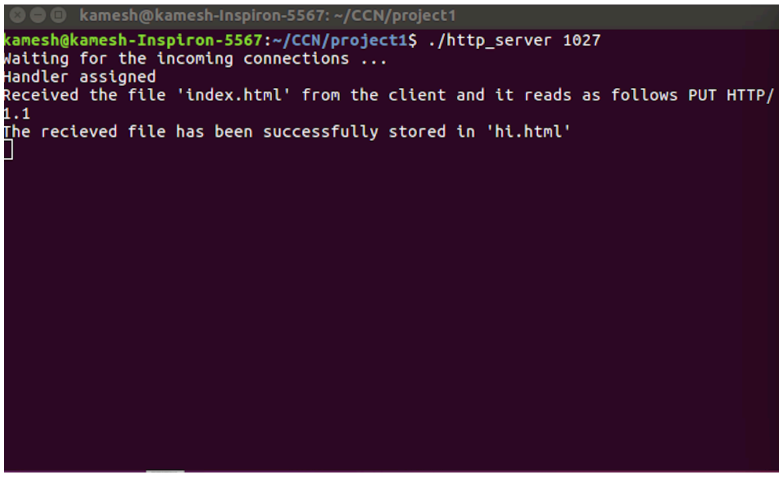
Fig: Invoking the GET request at the client

PUT:

When the client requests the server to store some data, we invoke this command:

./httpclient 127.0.0.1 1027 PUT index.html

When we invoke this command, the client connects to the server via a TCP connection and submits an HTTP/1.1 PUT request to the server and then sends the file to the server. Once the file is sent, the client waits for the response from the server.



```
kamesh@kamesh-Inspiron-5567: ~/CCN/project1
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./http_server 1027
Waiting for the incoming connections ...
Handler assigned
Received the file 'index.html' from the client and it reads as follows PUT HTTP/
1.1
The recieved file has been successfully stored in 'hi.html'
```

Fig: Invoking the PUT request at the client

HTTP Server:

In our program, our server is named http_server and the port number that we have specified for the server to listen to is 1027.

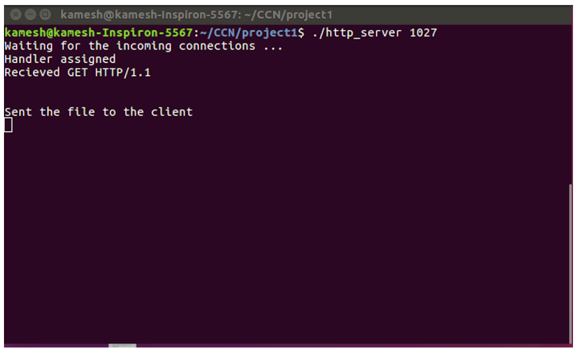
When the client requests the server to store some data, we invoke this command:

./http_server 1027

We used the socket() call to create the socket and use the bind() to bind it to the port number that it is supposed to listen to. We use the listen() function to keep listening to the incoming connections from the clients. Once the client connection is accepted by the server, we use the read() function to assess the request that we have received from the client.
Depending on the request of the server, we have programmed the server to respond in these ways:

For a GET request:

The server sends a "200 Ok" message followed by the file that was requested, which appears at the client end.

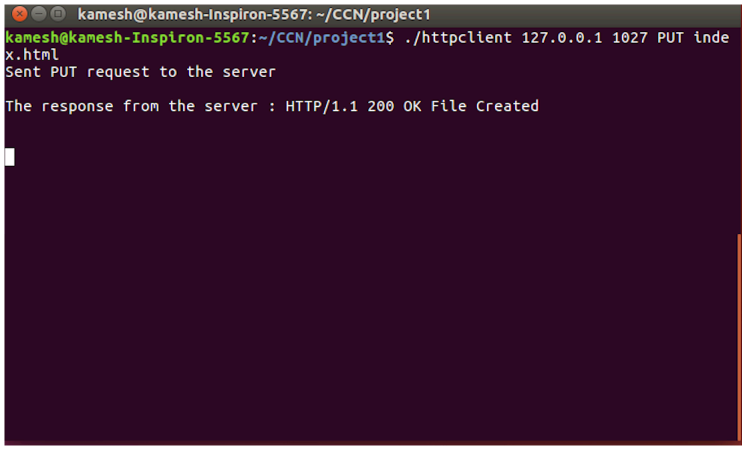


```
kamesh@kamesh-Inspiron-5567: ~/CCN/project1
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./http_server 1027
Waiting for the incoming connections ...
Handler assigned
Recieved GET HTTP/1.1


Sent the file to the client
```

Fig: Receiving the GET request at the server

For a PUT request:

The server saves the file as index.html.

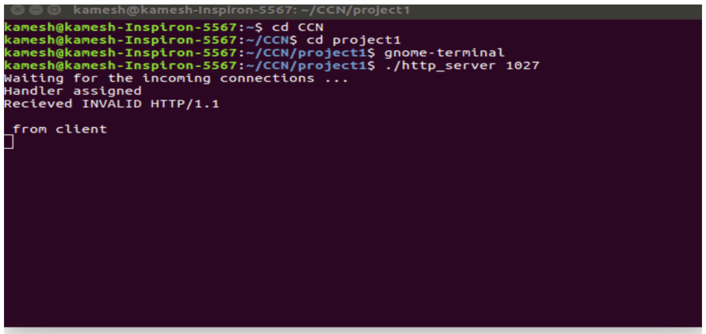


```
kamesh@kamesh-Inspiron-5567: ~/CCN/project1
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 PUT inde
x.html
Sent PUT request to the server

The response from the server : HTTP/1.1 200 OK File Created
```

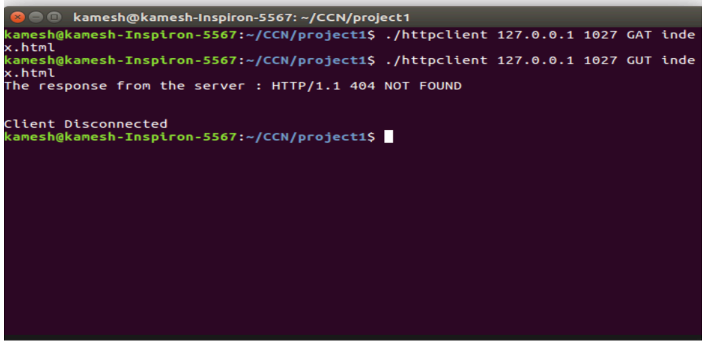
Fig: Receiving the PUT request at the server

Error condition:

In this condition, when the server is unable to service the request the client, it sends the 404 Not Found error.



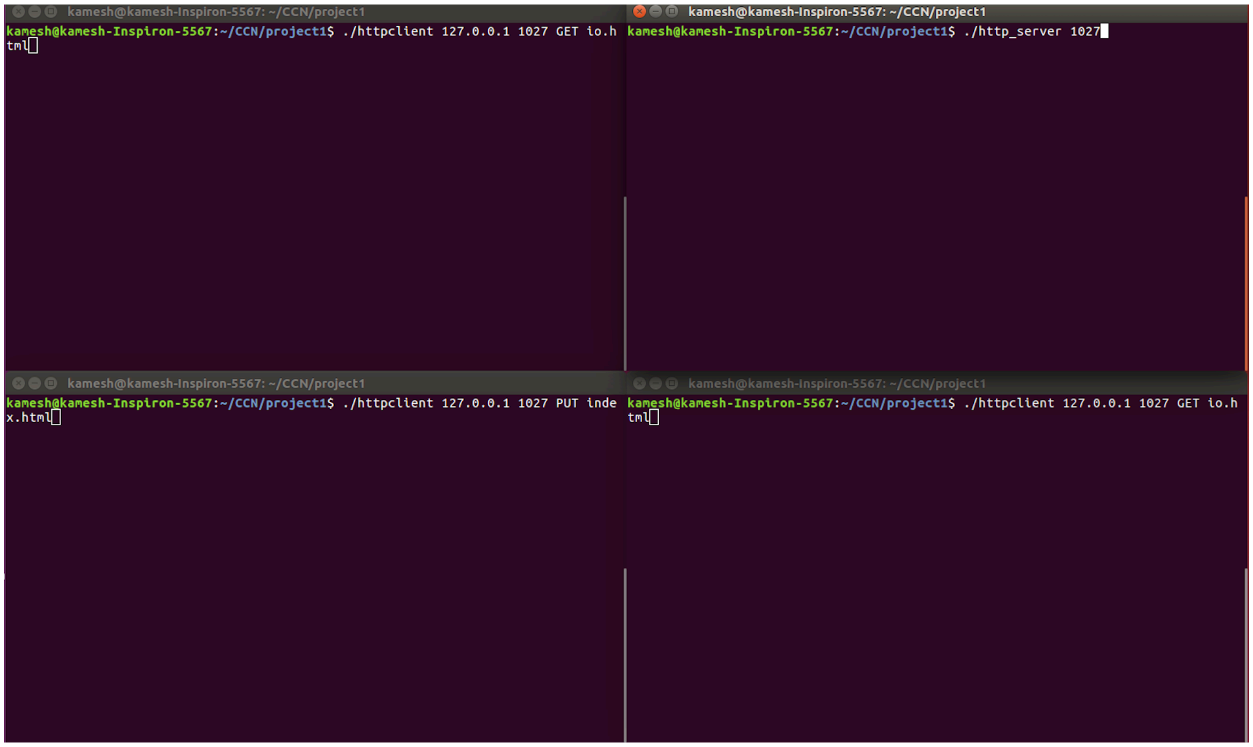
```
kamesh@kamesh-Inspiron-5567: ~/CCN/project1
kamesh@kamesh-Inspiron-5567:~$ cd CCN
kamesh@kamesh-Inspiron-5567:~/CCN$ cd project1
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ gnome-terminal
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./http_server 1027
Waiting for the incoming connections ...
Handler assigned
Recieved INVALID HTTP/1.1

 from client
```



```
kamesh@kamesh-Inspiron-5567: ~/CCN/project1
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 GAT inde
x.html
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 GUT inde
x.html
The response from the server : HTTP/1.1 404 NOT FOUND


Client Disconnected
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ 
```

Multithreading:

Here, we try to implement the multithreading concept where we have multiple clients that try to get serviced by the server simultaneously. In this case, we can see that the server is able to handle all the service requests in parallel without any errors.



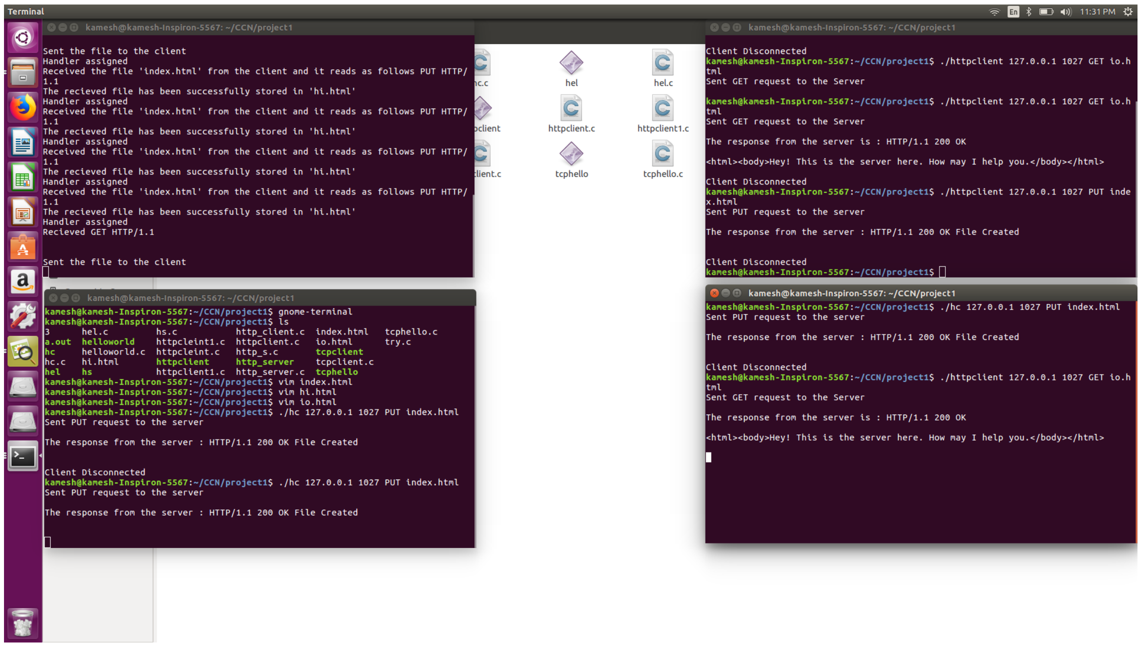
```
kamesh@kamesh-Inspiron-5567: ~/CCN/project1
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 GET io.h
tml
```

```
kamesh@kamesh-Inspiron-5567: ~/CCN/project1
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./http_server 1027
```

```
kamesh@kamesh-Inspiron-5567: ~/CCN/project1
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 PUT inde
x.html
```

```
kamesh@kamesh-Inspiron-5567: ~/CCN/project1
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 GET io.h
tml
```

Fig: Here, we see the server initialized and the clients sending their requests to the server



```
Sent the file to the client
Handler assigned
Received the file 'index.html' from the client and it reads as follows PUT HTTP/
1.1
The recieved file has been successfully stored in 'hi.html'
Handler assigned
Received the file 'index.html' from the client and it reads as follows PUT HTTP/
1.1
The recieved file has been successfully stored in 'hi.html'
Handler assigned
Received the file 'index.html' from the client and it reads as follows PUT HTTP/
1.1
The recieved file has been successfully stored in 'hi.html'
Handler assigned
Received the file 'index.html' from the client and it reads as follows PUT HTTP/
1.1
The recieved file has been successfully stored in 'hi.html'
Handler assigned
Recieved GET HTTP/1.1


Sent the file to the client
```

```
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ gnome-terminal
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ls
3       hel.c         hs.c           http_client.c  index.html   tcphello.c
a.out   helloworld    httpcleint1.c  httpclient.c   io.html      try.c
hc      helloworld.c  httpcleint.c   http_s.c       tcpclient
hc.c    hi.html       httpclient     http_server    tcpclient.c
hel     hs            httpclient1.c  http_server.c  tcphello
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ vim index.html
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ vim hi.html
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ vim io.html
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./hc 127.0.0.1 1027 PUT index.html
Sent PUT request to the server

The response from the server : HTTP/1.1 200 OK File Created


Client Disconnected
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./hc 127.0.0.1 1027 PUT index.html
Sent PUT request to the server

The response from the server : HTTP/1.1 200 OK File Created
```

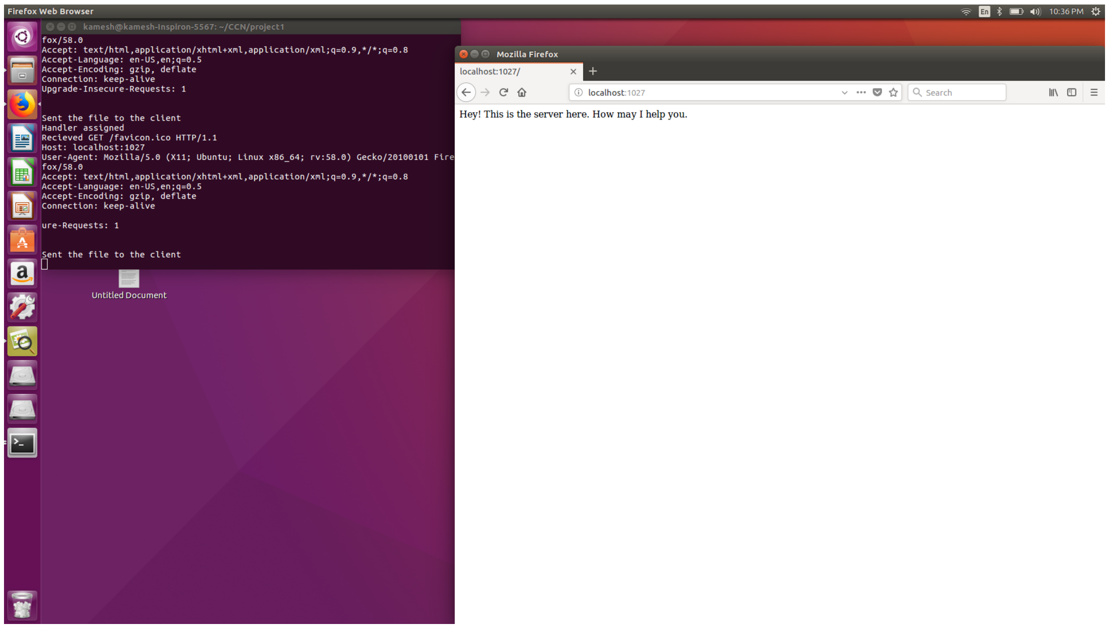
```
Client Disconnected
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 GET io.h
tml
Sent GET request to the Server

kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 GET io.h
tml
Sent GET request to the Server

The response from the server is : HTTP/1.1 200 OK

<html><body>Hey! This is the server here. How may I help you.</body></html>

Client Disconnected
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 PUT inde
x.html
Sent PUT request to the server

The response from the server : HTTP/1.1 200 OK File Created


Client Disconnected
kamesh@kamesh-Inspiron-5567:~/CCN/project1$
```

```
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./hc 127.0.0.1 1027 PUT index.html
Sent PUT request to the server

The response from the server : HTTP/1.1 200 OK File Created


Client Disconnected
kamesh@kamesh-Inspiron-5567:~/CCN/project1$ ./httpclient 127.0.0.1 1027 GET io.h
tml
Sent GET request to the Server

The response from the server is : HTTP/1.1 200 OK

<html><body>Hey! This is the server here. How may I help you.</body></html>
```

Fig: We see that the server has been able to service each and every client in parallel.

Checking the Server with a Browser:

Here, we check if our server is running by accessing it via a live browser.



```
fox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1


Sent the file to the client
Handler assigned
Recieved GET /favicon.ico HTTP/1.1
Host: localhost:1027
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:58.0) Gecko/20100101 Fire
fox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

ure-Requests: 1


Sent the file to the client
```

Hey! This is the server here. How may I help you.

Specifications regarding the project:

1. Programming Language- C
2. OS- Linux

References:

1. Computer Networking: A Top Down Approach (6th Edition)- Jim Kurose, Ross Addison-Wesly.
2. Github.com
3. Youtube.com