PART-1: A Simple Web Client

I have developed a simple web client that takes four or six command line prompts from user. The details of both command prompts are as given below:

Command line has 4 arguments:

python Client.py <server_ip or name> <server_port> <path>

- <server_ip or name> : Ip address or domain name of server.
- <server_port> : Port number at which target server is hosted.
- <path>: The path at which requested file resides.

Command line has 4 arguments:

python Client.py <proxy_ip> <proxy_port> <server_ip> <server_port> <path>

- <proxy_ip>:Ip address of proxy server.
- <proxy_port>:Port number at which proxy server is hosted.
- <server_ip or name> : Ip address or domain name of server.
- <server_port> : Port number at which target server is hosted.
- <path>: The path at which requested file resides.

1. Based on the type of input received from the user it creates socket connection either to server or to the proxy server and calls function "**send_request()**" to send request.
2. Then **send_request() function** creates a GET request and send it to the serverand waits for response. Upon receiving the response it then returns it to the main function.
3. Then, the main function calls the function "**parse_html_and_fetch_resources()**" which parse the response html and find all the references and return as a list by joining it to the base html path to main function.
4. Then the main function calls again forms connection for each referenced object and fetch all the referenced objects.

Functions in the client code:

- get_ip_or_host(hostname_or_ip): Get Ip address if domain name is provided
- new_connect(proxy_port, proxy_ip, host_port, host_ip): Creates a new connection to the server.
- send_request(sock, path, host_ip, host_port): send and receive response from the server.

- parse_html_and_fetch_resources(html, base_url): Prase the HTML file to find path of resources
- close_socket(sock): close the opened sockets.

Modules used:

- from bs4 import BeautifulSoup : Helps to parse documenta nd find tags.
- from urllib.parse import urljoin : Helps to jpon paths and find the overall relative path.

Screenshots:

Command : python3 client.py gaia.cs.umass.edu 80 /networks/education/index.html

It will directly connect to server gaia.cs.umass.edu at port number 80 and send the file located at /networks/education/index.html

```
akshay@akshay:~/Desktop$ python3 client.py gaia.cs.umass.edu 80 /networks/education/index.html
HTTP/1.1 200 OK
Date: Sun, 05 Nov 2023 16:05:05 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Thu, 05 Apr 2001 20:54:33 GMT
ETag: "df5-3813860d73440"
Accept-Ranges: bytes
Content-Length: 3573
Content-Type: text/html; charset=UTF-8

<HTML>
<HEAD>
<TITLE>UMass Computer Networks Research Group - Education</TITLE>
</HEAD>


<BODY BACKGROUND="images/bg.jpg" MARGINHEIGHT=3 MARGINWIDTH=2 TOPMARGIN=0
LEFTMARGIN=0 RIGHTMARGIN=5
BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#5B69A6" VLINK="#5B69A6"
ALINK="#00FF00"
```

Here, we can see that it is fetching the base HTML file from the server after the HTTP/1.1 200 OK response.

```
<p>
<a href="http://occ.awlonline.com/bookbind/pubbooks/kurose-ross1/">Computer
Networking: A Top-Down Approach Featuring the Internet</a> by <a
href="http://www-net.cs.umass.edu/personnel/kurose.html">Jim Kurose</a> <p>

</tr>
</table>


</BODY>
</HTML>


['/networks/images/cnrg_logo1.jpg', '/networks/images/banner-education.jpg', '/networks/images/homebuttor
s/Collaborationsbutton.jpg', '/networks/images/Newsbutton.jpg', '/networks/images/peoplebutton.jpg', '/ne
onsbutton.jpg', '/networks/images/Researchbutton.jpg', '/networks/images/Resourcesbutton.jpg']
b'HTTP/1.1 200 OK\r\nDate: Sun, 05 Nov 2023 16:05:11 GMT\r\nServer: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-
l/2.0.11 Perl/v5.16.3\r\nLast-Modified: Thu, 05 Apr 2001 20:54:31 GMT\r\nETag: "e70-3813860b8afc0"\r\nAcc
ntent-Length: 3696\r\nContent-Type: image/jpeg\r\n\r\n\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x00H\x
C\x00\x05\x03\x04\x04\x04\x03\x05\x04\x04\x04\x05\x05\x05\x06\x07\x0c\x08\x07\x07\x07\x07\x0f\x0b\x0b\t\x
```

Here, we can see that once the complete base file is fetched then it a list of path is shown which is then fetched in next line. As the fetched objects are images hence the byte response are printed here.

Note: The Verification of client through proxy is shown in next step after the development of the proxy.

PART-2: A Simple Web Proxy Server

I have developed a simple web proxy that first binds itsef to a IP address and Port number. Then, it keep listening for incoming connection from clients. Once it receives a connection from client then it start multithreading and call the function “handle_request()” to handle the request from the client.

The function a handle_client() receives the message from client and finds the IP address and the port number of the target server using function “find_host_port()” which it needs to connect from to fetch the data.

Then, it forms a connection to server and sends the modified request to server and wait for response. Once it receives the response from server the it parse and send it the client that requested it.

The function “find_host_port()” find the IP address and port number of the target server by splitting the GET message received from the server.

Functions in the client code:

- get_ip_or_host(hostname_or_ip): Get Ip address if domain name is provided
- handle_request(client_socket): Handle the client request, connect to server, and return the requested file to client.
- find_host_port(request):find the IP address and port number of the target server by splitting the GET message
- proxy_server():web proxy that first binds itsef to a IP address and Port number. Then, it keep listening for incoming connection from clients.

Modules used:

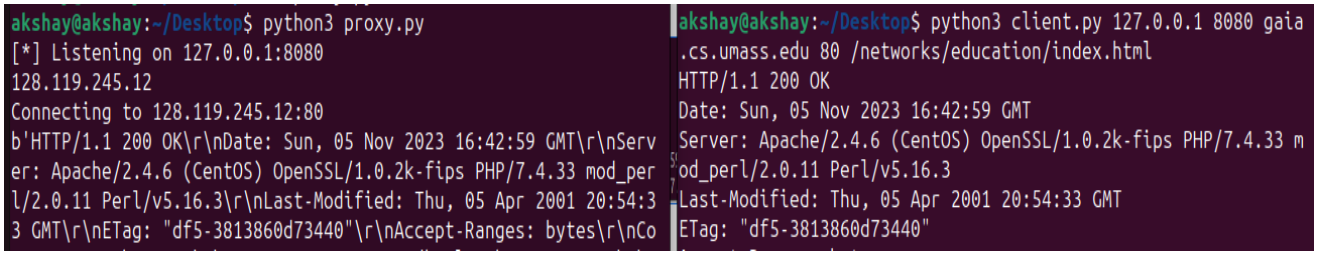
- Threading module to create multithreading.

Screenshots:

Run the proxy server

Command for my client code created in PART 1:

python3 client.py 127.0.0.1 8080 gaia.cs.umass.edu 80 /networks/education/index.html

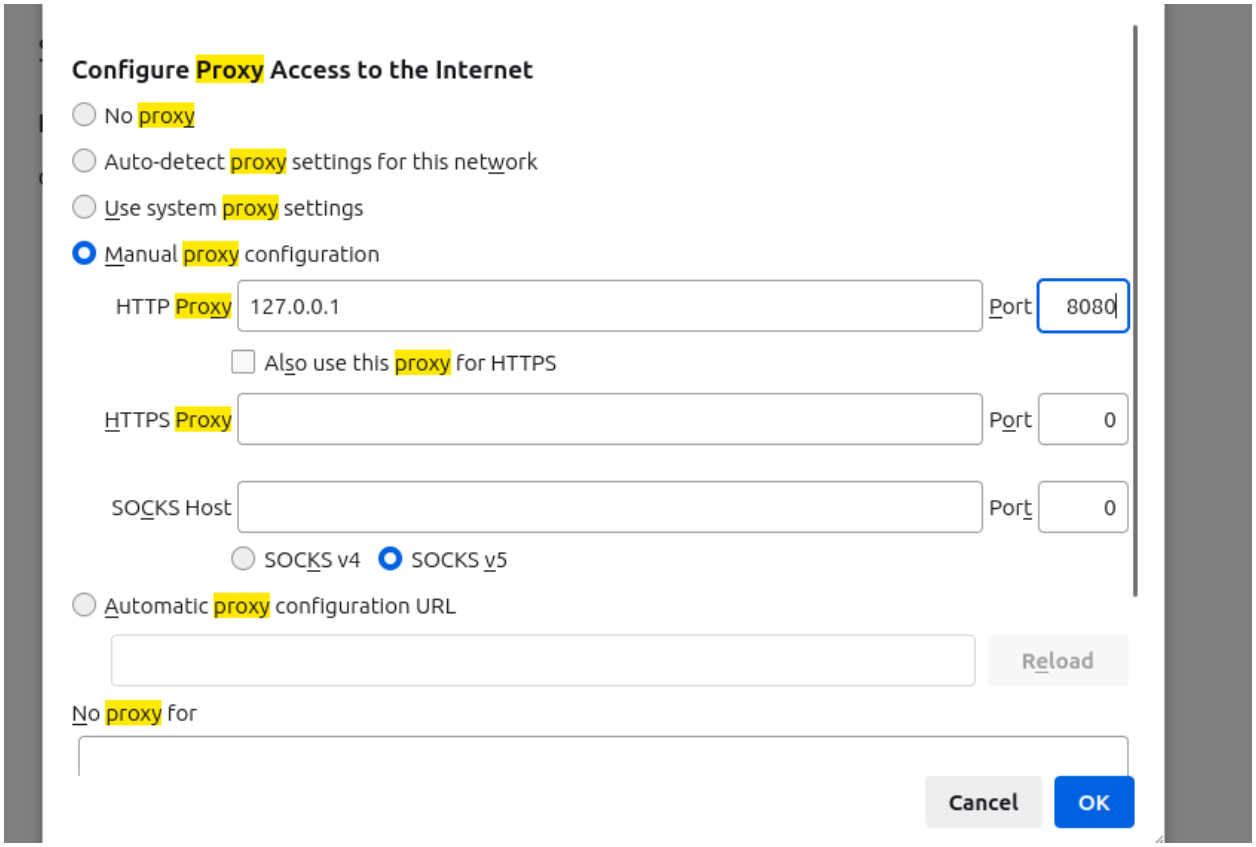


Receiving the response at both proxy and server:

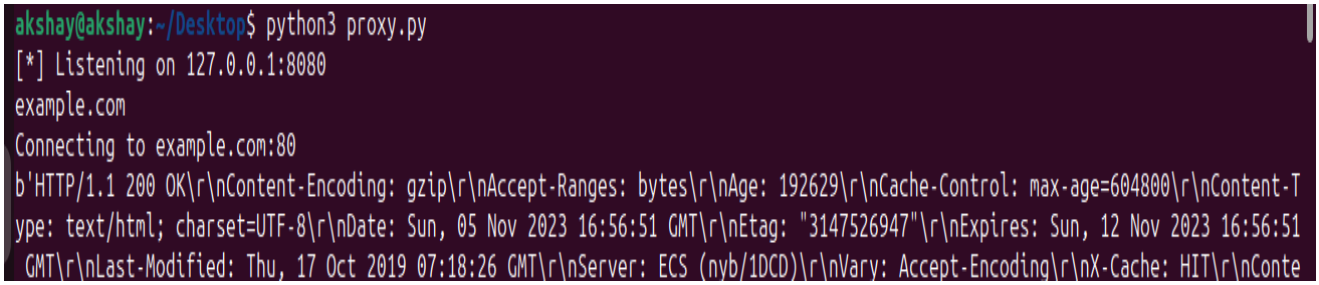
```
akshay@akshay:~/Desktop$ vi proxy.py
akshay@akshay:~/Desktop$ python3 proxy.py
[*] Listening on 127.0.0.1:8080
128.119.245.12
Connecting to 128.119.245.12:80
HTTP/1.1 200 OK
Date: Sun, 05 Nov 2023 16:49:49 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod
_perl/2.0.11 Perl/v5.16.3
Last-Modified: Thu, 05 Apr 2001 20:54:33 GMT
ETag: "df5-3813860d73440"
Accept-Ranges: bytes
Content-Length: 3573
Content-Type: text/html; charset=UTF-8

<HTML>
<HEAD>
<TITLE>UMass Computer Networks Research Group - Education</TITLE
>
</HEAD>


<BODY BACKGROUND="images/bg.jpg" MARGINHEIGHT=3 MARGINWIDTH=2 TO
PMARGIN=0
LEFTMARGIN=0 RIGHTMARGIN=5
BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#5B69A6" VLINK="#5B69A6"
ALINK="#00FF00"
```

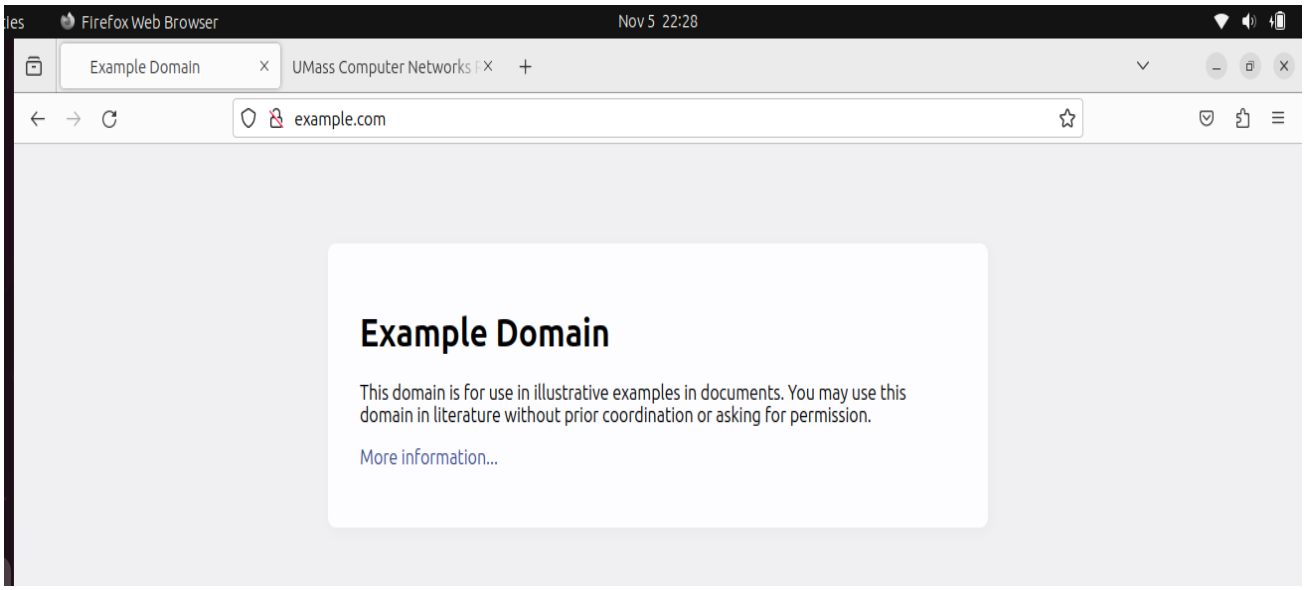
```
akshay@akshay:~/Desktop$ python3 client.py 127.0.0.1 8080 gai
a.cs.umass.edu 80 /networks/education/index.html
HTTP/1.1 200 OK
Date: Sun, 05 Nov 2023 16:49:49 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33
mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Thu, 05 Apr 2001 20:54:33 GMT
ETag: "df5-3813860d73440"
Accept-Ranges: bytes
Content-Length: 3573
Content-Type: text/html; charset=UTF-8

<HTML>
<HEAD>
<TITLE>UMass Computer Networks Research Group - Education</TI
TLE>
</HEAD>


<BODY BACKGROUND="images/bg.jpg" MARGINHEIGHT=3 MARGINWIDTH=2
 TOPMARGIN=0
LEFTMARGIN=0 RIGHTMARGIN=5
BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#5B69A6" VLINK="#5B69A
6"
ALINK="#00FF00"
>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH="700">
```

Proxy setting in browser:

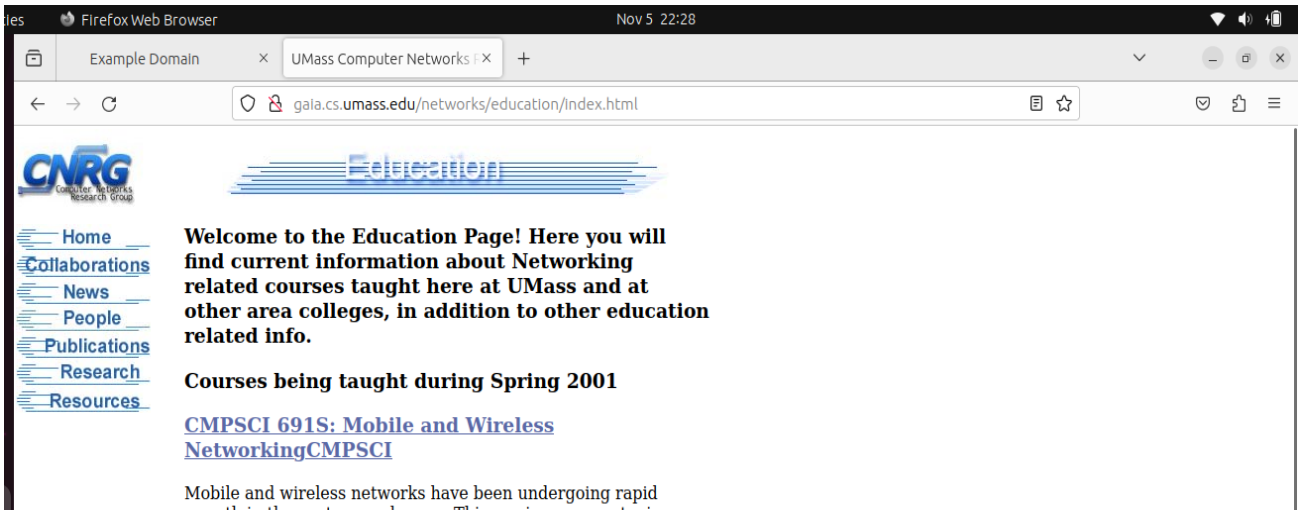


Screenshot of example.com when hitted from the browser:



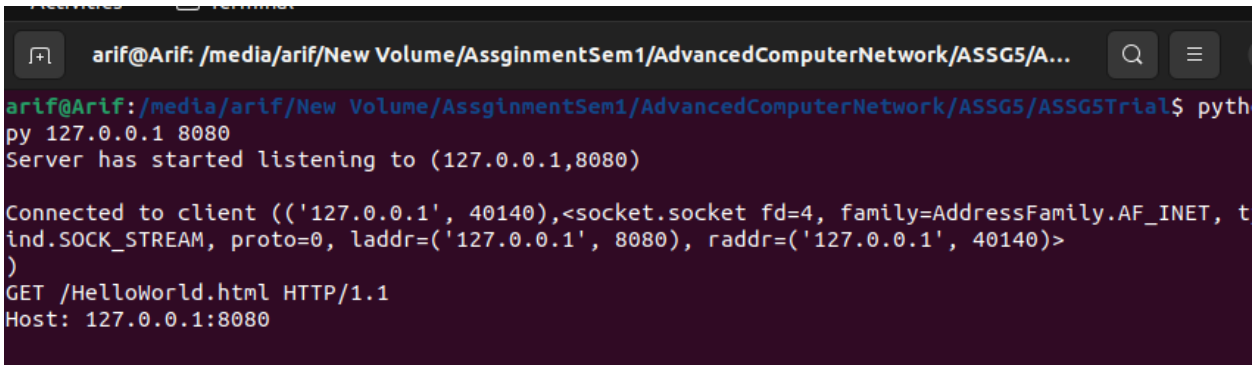


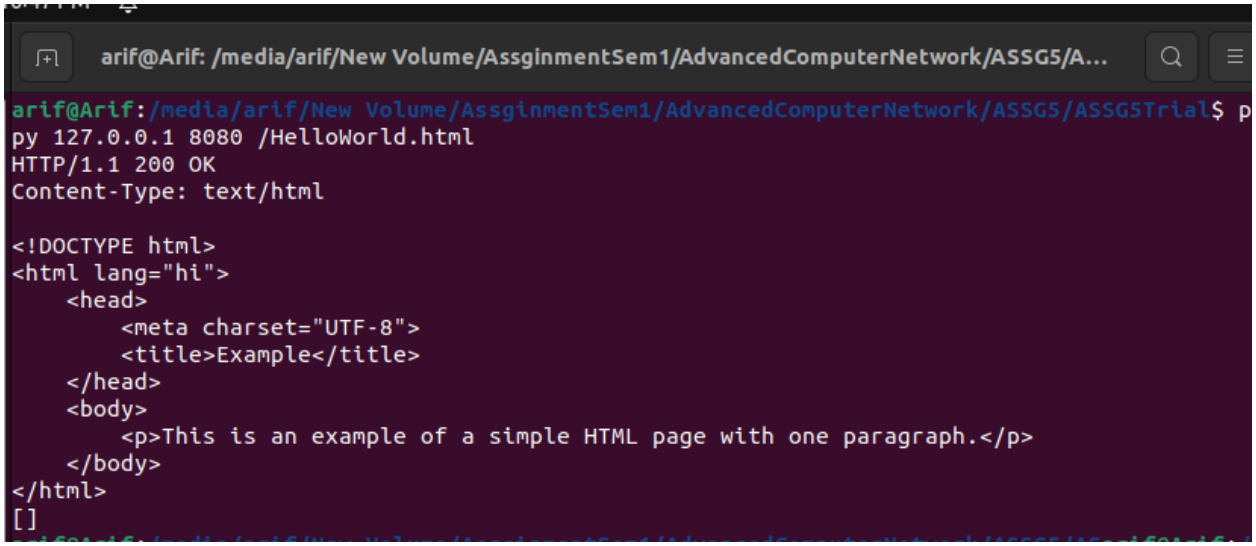
Screenshot of example.com when hitted from the browser:



PART-3: A Simple Web Server

In this part I have a multithreaded server which first bind itself to a IP address and port number which it receives from the command line argument. Then it start listening for request from client and once it receives response from the client then creates a thread to handle it. It first decode the get message to find the requested path and the serves the requested file .





PART-4: URL and Content Filtering at web proxy and web usage stats at web proxy:

In this part I am extending the proxy server created in PART 2. I have created few more functions to convert the response received from the server into Hindi and store the details of the user in a file for various use case as per the requirement.

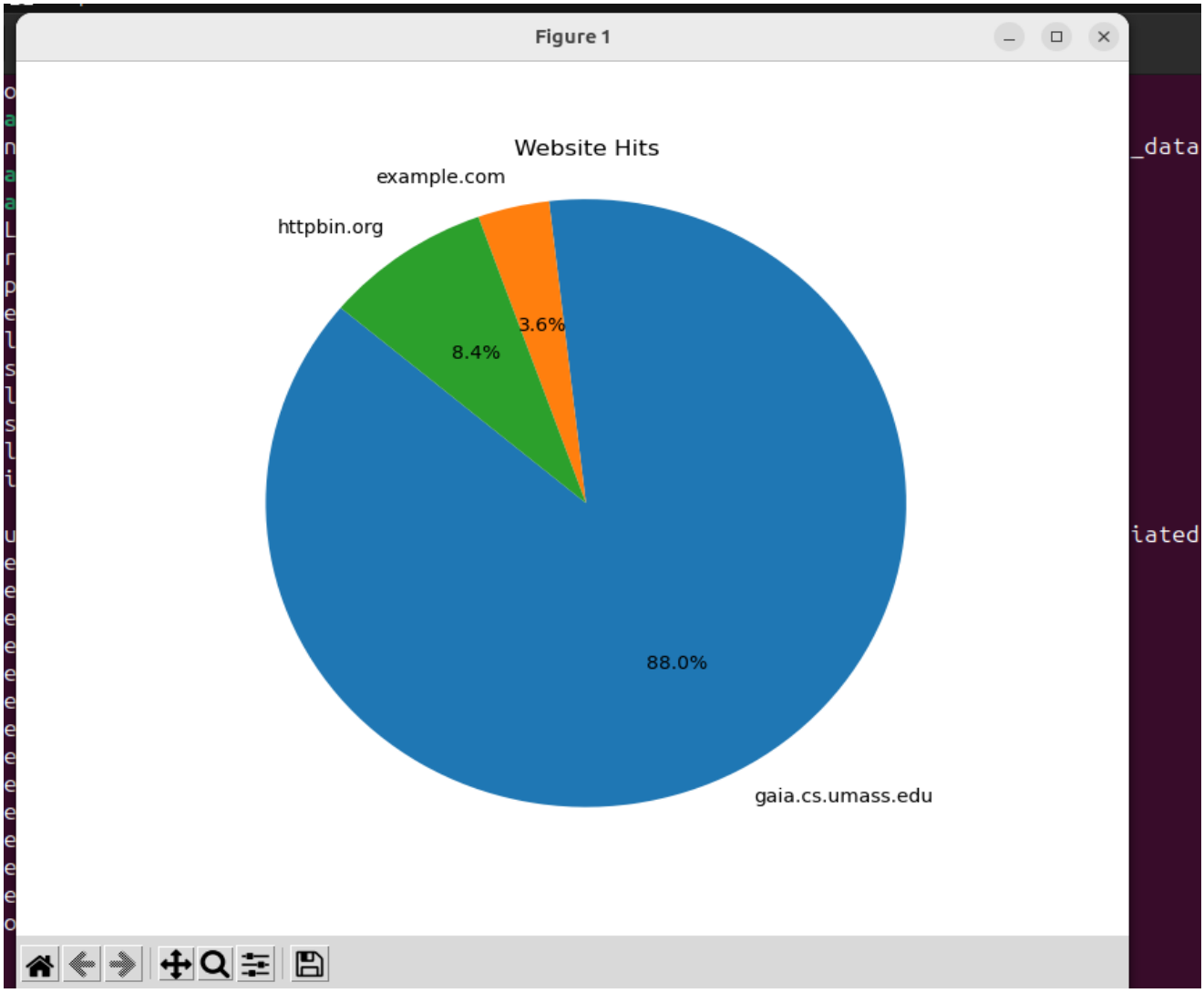
Given below the added functions and its functionalities:

- generate_pie_chart(): This function is used to create pie chart from the data that is stored in the file.
- save_user_data(user_data): This function is used to store the data in file before closing the connection.
- load_user_data(): This function is used to load the data of user from the file to show statistics and append new data to it.

Given below the screenshot of extended proxy and statistics:

```
akshay@akshay:~/Desktop$ vi ExtendedProxy.py
akshay@akshay:~/Desktop$ python3 ExtendedProxy.py
[*] Listening on 127.0.0.1:8080
```

```
Connecting to example.com:80
Connecting to gaia.cs.umass.edu:80
Connecting to gaia.cs.umass.edu:80
Connecting to gaia.cs.umass.edu:80
Connecting to gaia.cs.umass.edu:80
```



```
Content-Type: text/html

<!DOCTYPE html>
<html lang="hi">
    <head>
        <meta charset="UTF-8">
        <title>Example</title>
    </head>
    <body>
        <p>This is an example of a simple HTML page with one paragraph.</p>
    </body>
</html>
[]
```