**ITCS 6166/8166 Computer Communication Networks**

**Project – 1 Report**

**HTTP Client and Server**

**Team Members:**

1. Arunkumar Bagavathi (800888454)
2. Chetan Borse (800936059)

**Programming Language: Python Version 3**

**Objective:**

The objective of this project is to implement HTTP server and HTTP client that run-on simple HTTP/1.1. In this project, our goal is to implement GET and PUT requests from client to the server and the server handles the client requests.

**HTTP GET and PUT requests:**

GET and PUT requests are most commonly used methods between server and client in HTTP/1.1. In GET request, client requests the server to return a file (more like downloading a file from the server). In PUT request, client asks the server to store a file in server’s location (more like uploading a file to the server).

We used socket programming and TCP connection to handle GET and PUT requests and returning server response.

**HTTP Server:**

In HTTP, the server runs at a particular port in a particular IP address (localhost or 127.0.0.1 in case of single machine running both server and client). We implemented a multithreaded server, which can serve requests from multiple clients at a same time. Thus, the server is ready to serve the client request all the time. Server can do the following:

* Setup connection with the client
* Receive GET request – Returns a message: “200 OK” followed by the requested file if the file exists or returns a message: “404 Not Found” if the requested file does not exist
* Receive PUT request - Accept and save the file in the server directory and responds to the client with “200 OK File Created” message
* After finishing the client request, closes the TCP connection and starts listening for other client requests

**HTTP Client:**

In HTTP, client can request a file from the server or request to save client’s file in the server after successfully connecting with the server. Only after client sending some request to the server, any GET or PUT task can be achieved. Client can do the following:

* Connecting with the server which is currently running at a given IP address and port number
* Giving a GET request to the server – requesting a file from the server
* Giving a PUT request to the server – requesting the server to save a file in server directory after successfully reading the given file
* Closing the connecting with server once the server responds to client with a success message

**Instructions to run the python code:**

1. Unzip the zip file that is attached with the submission
2. Go to the unzipped folder using terminal or load the folder in Anaconda software
3. Start the server using the command:

**python ServerApp.py**

1. Optionally, you can specify server IP address, server port number and server directory for storing files using -t, -p and -w parameters respectively like

**python ServerApp.py -t <IP\_ADDRESS> -p <PORT\_NUMBER> -w <SERVER\_FILE\_DIRECTORY>**

These parameters were made optional. So, the server program will take default values assigned to it even if you do not give these values. Default server IP\_ADDRESS of is set to **127.0.0.1** and default server PORT\_NUMBER is set to **8080**We created a default server and client directory for saving files in server and client respectively. The directories are inside the *data* folder. Each client and server directory contains one file. “index.html” in the server folder and “client\_index.html”

1. Start the client with the command:

**Python ClientApp.py -s <CLIENT\_IP> -c <CLIENT\_PORT> -t <SERVER\_IP> -p <SERVER\_PORT> -m <REQUEST\_METHOD> -f <FILE\_NAME> -d <CLIENT\_FILE\_DIRECTORY>**

All parameters for running the client contains default values. Default values are:

* CLIENT\_IP – 127.0.0.1
* CLIENT\_PORT – 8081
* SERVER\_IP – 127.0.0.1
* SERVER\_PORT – 8080
* REQUEST\_METHOD – GET
* FILE\_NAME – index.html
* CLIENT\_FILE\_DIRECTORY – points to project\_directory/data/client

1. Thus, the command **python ClientApp.py** sends the server, running in IP address 127.0.0.1 at port number 8080, a GET request to return a file “index.html”
2. To make a proper GET request, use the command:

**python ClientApp.py -t <SERVER\_IP> -p <SERVER\_PORT> -m GET -f <FILE\_NAME>**

1. To make a proper PUT request, use the command:

**python ClientApp.py -t <SERVER\_IP> -p <SERVER\_PORT> -m PUT -f <FILE\_NAME> -d <FILE\_DIRECTORY>**

**Results:**

1. **Running GET command and the file exists in the server**

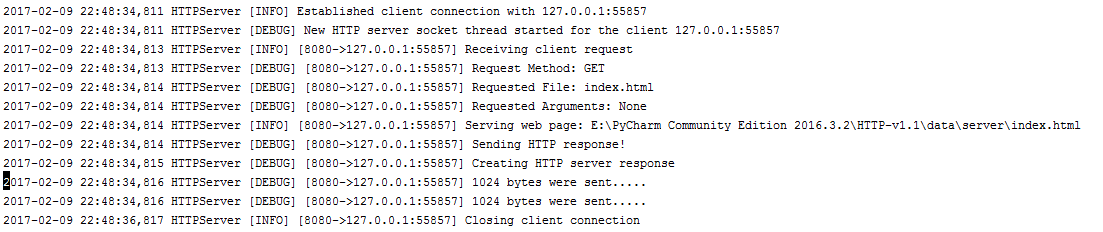


Figure : Server responding to client's GET request

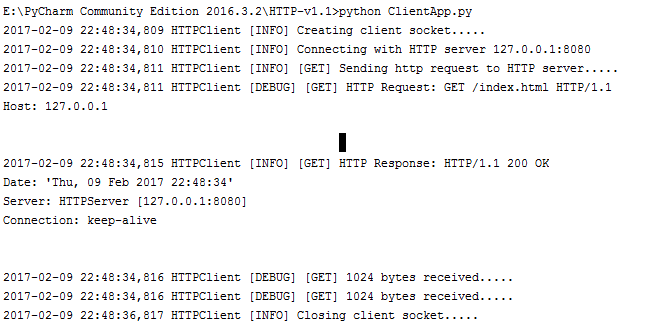


Figure : Client giving GET request and receiving response from the server

Client receives “200 OK” message and the requested file “index.html” is now present in the default client directory

1. **Running a GET command and the requested file does not exist in the server**



Figure : Server responding to the client request that requested file does not exist

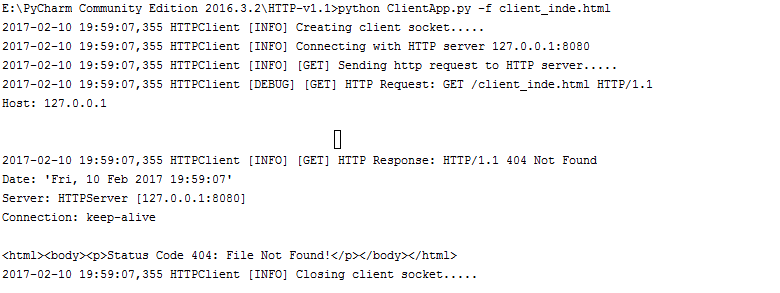


Figure : Client requesting the unavailable file and getting "404" error message

1. **Running a PUT command**

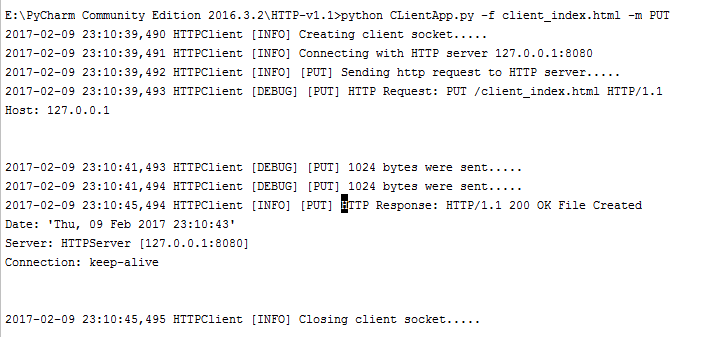


Figure 5: Client sending a PUT request

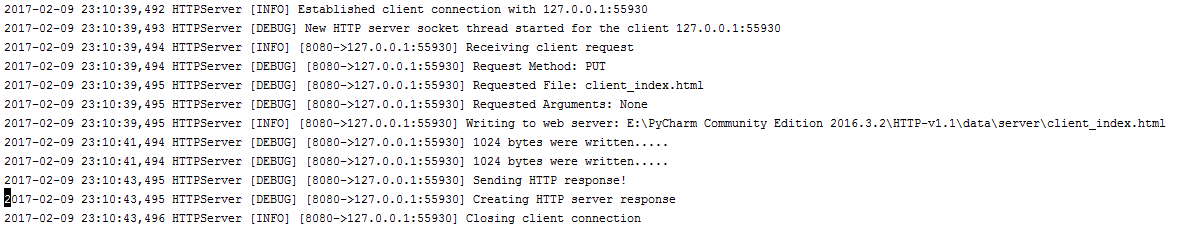


Figure 6: Server processing the PUT request

Now, the client\_index.html is present in server directory

**Exception in the code:**

We can able to terminate the server using **Ctrl + C** only in the Unix or Mac machine. But the same command is not working in the Windows command prompt