Nikhil Bhirud

Part 1 : Socket Programming

HTTP Client and Server:

The client accepts the command line arguments as per the below syntax:

httpclient hostname port command filename

In response to a GET/PUT command from the client, the below sequence of actions are performed on Client and Server:

Client initiates a connection to the server using TCP on any port mentioned by the user and passes the necessary arguments to GET/PUT a required file from/on the server as mentioned in the above syntax.

The server then establishes connection to the client on the incoming port and socket and identifies the authenticity of the type of request. Based on the command (GET/PUT) executed in the client, performs the necessary steps and then closes the connection.

In case of GET operation, any file that is being fetched by the client from the server is read and displayed on the console on the client end. A screenshot (fig.a) has been attached depicting the same process on the client and server.

In case of PUT operation, any file that is being put on the server by the client is read by the server and locally copied in the local path of server (Serverfiles in this case). A screenshot (fig.b) has been attached depicting the same process on the client and server.

Steps How to Execute:

1. Compile the Httpclient and Httpserver using commands: javac httpclient.java; javac httpserver.java.

2. Start the server using the command: java httpserver 5678 (any port number as required).

3. Execute the command to get/put from the client end as below:

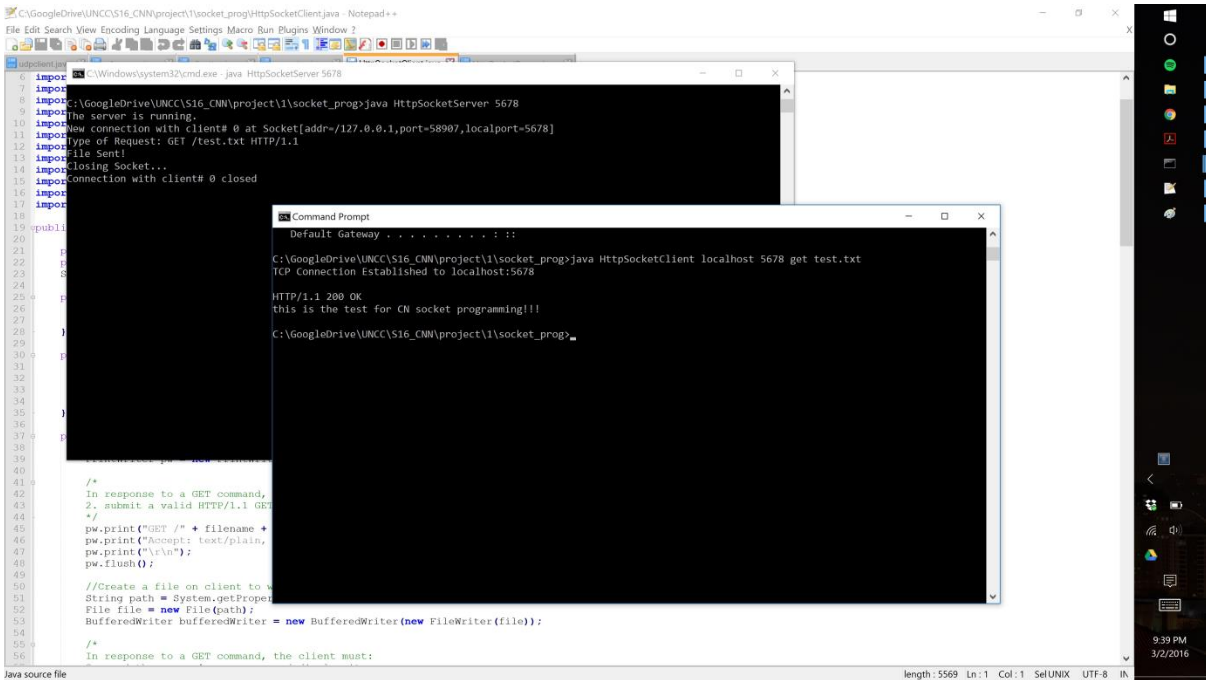
java httpclient <hostserver> 5678 GET test.txt

Here the executable is HttpSocketClient and HttpSocketServer.

4. The server and client display the necessary responses based on the success or failure to execute the command (200 OK or 404 Not Found messages).

5. Once the command is successful, check for necessary files in the local path (clientfiles/serverfiles) to verify if the files are created successfully.

(fig.a) GET request:

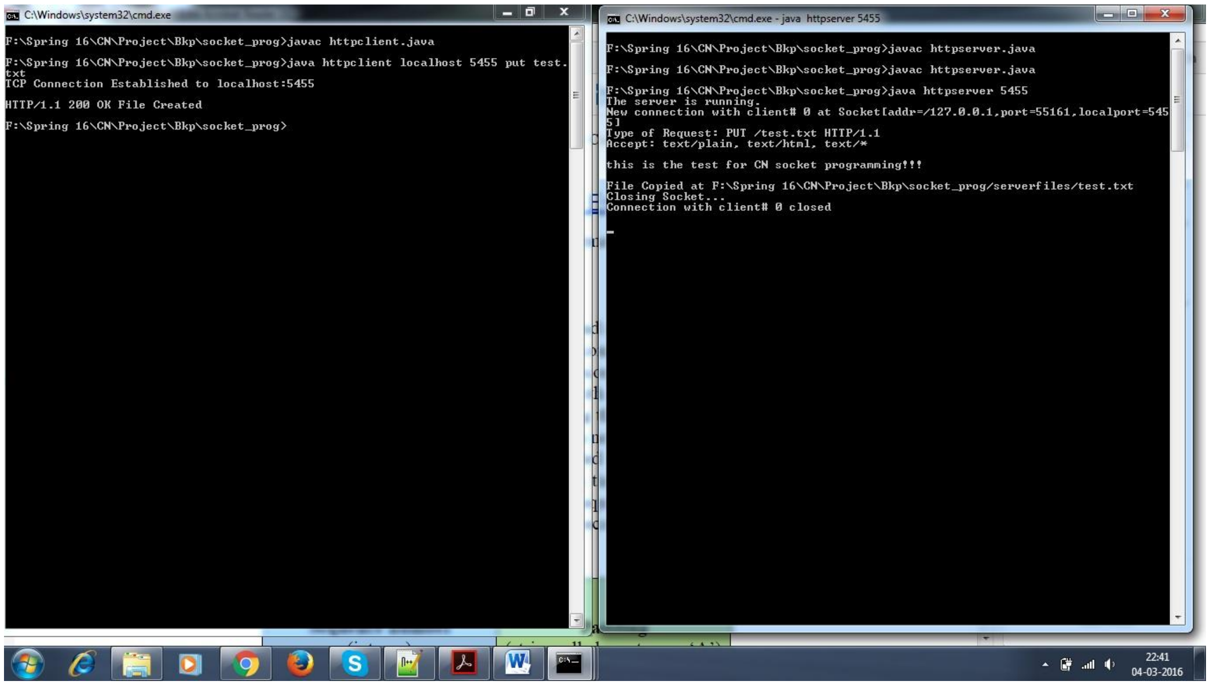


```
C:\GoogleDrive\UNCC\S16_CNN\project\1\socket_prog>java HttpSocketServer 5678
The server is running.
New connection with client# 0 at Socket[addr=/127.0.0.1,port=58907,localport=5678]
Type of Request: GET /test.txt HTTP/1.1
File Sent!
Closing Socket...
Connection with client# 0 closed
```

```
   Default Gateway . . . . . . . . . : ::

C:\GoogleDrive\UNCC\S16_CNN\project\1\socket_prog>java HttpSocketClient localhost 5678 get test.txt
TCP Connection Established to localhost:5678

HTTP/1.1 200 OK
this is the test for CN socket programming!!!

C:\GoogleDrive\UNCC\S16_CNN\project\1\socket_prog>
```

```
        /*
        In response to a GET command,
        2. submit a valid HTTP/1.1 GET
        */
        pw.print("GET /" + filename +
        pw.print("Accept: text/plain,
        pw.print("\r\n");
        pw.flush();

        //Create a file on client to w
        String path = System.getProper
        File file = new File(path);
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(file));

        /*
        In response to a GET command, the client must:
```

(fig.b) PUT request:



```
F:\Spring 16\CN\Project\Bkp\socket_prog>javac httpclient.java

F:\Spring 16\CN\Project\Bkp\socket_prog>java httpclient localhost 5455 put test.
txt
TCP Connection Established to localhost:5455

HTTP/1.1 200 OK File Created

F:\Spring 16\CN\Project\Bkp\socket_prog>
```

```
F:\Spring 16\CN\Project\Bkp\socket_prog>javac httpserver.java

F:\Spring 16\CN\Project\Bkp\socket_prog>javac httpserver.java

F:\Spring 16\CN\Project\Bkp\socket_prog>java httpserver 5455
The server is running.
New connection with client# 0 at Socket[addr=/127.0.0.1,port=55161,localport=545
5]
Type of Request: PUT /test.txt HTTP/1.1
Accept: text/plain, text/html, text/*

this is the test for CN socket programming!!!

File Copied at F:\Spring 16\CN\Project\Bkp\socket_prog/serverfiles/test.txt
Closing Socket...
Connection with client# 0 closed
```