COMP416:Project 1 StratoNet

Kemal Batuhan Başkaya 64240

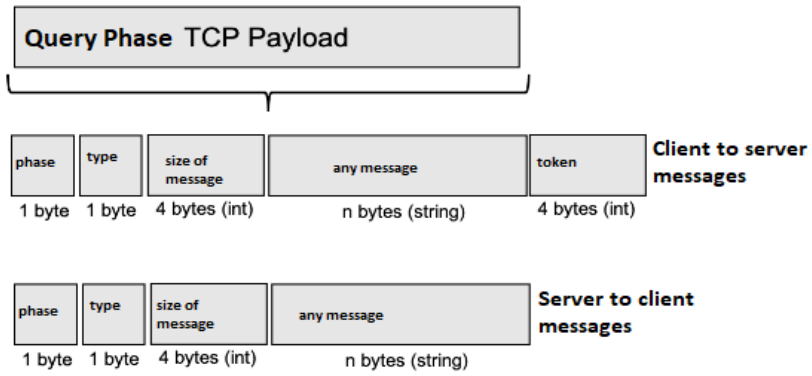
Description

In this first project of COMP416 course, we were assigned to develop a basic server-client pair. Both the client and server runs on the same server, though as different programs. We were suppose to use sockets to connect these applications and create a protocol for their communication. A basic protocol template was given to us for the authentication part. In the authentication part client connects and authenticates itself by logging in. After that, it can request either NASA Astronomy Picture of Day or Mars weather info. Server requests these information from NASA servers through its API, and send it to the client through a new data socket. I used the PS1 and PS2 codes as a template for my project because they were simple and clear. I built the requirements of this project on this template.

Protocol

When a client connects to the already known welcoming socket, the server moves the client to new a command socket. The client is only allowed to use login, quit and help command at this point. The client initiates the login phase by sending its username. The server asks for a password. If the given username-password combination doesn't match the database (a simple file), the server asks for password, up to 3 times. It should be made clear that even if the given username is not in the database, the server still asks for password 3 times. After successful authentication, the client is given an authentication token, which will and shall be provided when a request is made to the server.

After the authentication, the client is provided with a new welcoming socket, which creates a new file transfer thread and a file transfer socket for the client. Upon connecting to file socket, the client sends its token for initializing and matching this socket with the command/authentication socket. This thread updates a hashmap which will be used for communicaitons between command thread/file transfer thread of the server. Now the client can request mars weather info or APOD file through the command socket. This is done by a protocol which is similar to the authentication phase. All command socket protocol for query phase is like following:



PHASE	TYPE	SIZE	MESSAGE	TOKEN	
1	0	1	(byte)0	(int)token	client->server requesting APOD
1	1	1	(byte)0	(int)token	client->server requesting Mars Weather Pressure
1	2	4	(int)hash of file	not sent	server->client responding the hash file of transferred file
1	3	size of string	(string) failure message	not sent	server->client explaining failure

The authentication phase protocol was already given in the project pdf. Only **addition to the given auth. protocol** is, the server sends another message after Auth_Success (type code 3), where it gives the port to the file transfer socket with type code 4. After authentication client can start requesting files. When making a request, if the client sends a wrong token, server informs the client about the problem and closes connection. If the token was correct, this command thread sends a job to the file thread where this thread accesses corresponding NASA API, extracts the relevant object, hashes it, sends the hash to the command thread, and sends the object through its file socket. The command thread sends the hash through its command socket. The client downloads the file and verifies the file using the hash. If it doesn't verify, the user is notified about the issue. It is up to user if the client requests again or not.

Explanation

StratoServer

Main.java

Just like PS1 and PS2, my project has a main file where StratoServer.java is initialized.

StratoServer.java

In this file, the welcoming socket (ServerSocket) is created. This sockets address is localhost:4444 by default, but the port could be changed if it is given as an argument to main.java. Then the user database file is checked. Then the welcoming socket for file transfer is created(with hardcoded default port 4455). Then each of these welcoming sockets's timeout is set to 50 miliseconds. This is done because the program should listen to both sockets and serversocket.accept method might listen to only 1 socket forever if we don't set a timeout. This would stop the 2nd socket from being listened. When a connection is accepted, a new socket is created and ran on a new thread. The port is automatically given by the ServerSocket.

There is also an Hashmap<Integer,StratoFileThread> hashmap in this file. This hashmap lets the command/authentication thread to find and communicate with its corresponding file transfer thread. The token of the client is the key, and the file thread is the value in the map.

StratoServerThread.java (Command/Authentication Thread)

Upon creation, this file initializes a Queue<Integer>, which will be a mechanism used when the file thread downloads the file and send its hash to this queue. When this thread starts running, the input/output streams and datainput/dataoutput streams of the socket are created. Then the thread starts running a loop, listening to the client. It reads the first 2 bytes of the data input stream. The first bit from the client indicates the phase, 0 for authentication, 1 for query. The second bit is the type bit, indicating the stage. Its explanation is made under the Protocol title. Shortly:

00: auth_request(uname) or auth_request(pssword), -1-1: client disconnect, 10: APOD request, 11: mars weather request.

Depending on the first to bytes, the code continues to corresponding IF { } segments of the code.

<u>Case 00</u>

This is an authentication phase message. As this is the first 00 coded message in this while loop run, it is followed by the length of the uname, and then the uname. The uname is read byte by byte. Then the server is set with a 15 second timeout. If the client fails to enter his password in 15 seconds, the server disconnects from the client. After setting the timeout, server sends the client a password request. The server keeps how many times this request is sent. If the client fails to enter

correctly and re-asked for password 3 times, the server disconnects from the client. If the client correctly enters password, timeout limit is removed, authentication flag is set and the server sends the client the token it just generated (from port + uname) and the port to file transfer welcoming socket.

```
114     AUTHSTATUS=true;
115     TOKEN=(threadsocket.getLocalPort()+ipayload.hashCode());
116     dos.writeByte(0);dos.writeByte(3);dos.writeInt(4);dos.writeInt(TOKEN);
117     dos.flush();
118     dos.writeByte(0);dos.writeByte(4);dos.writeInt(4);dos.writeInt(4455);
119     dos.flush();
120     break;
```

Case -1-1

This message just informs the server about the client's disconnection. The server tries to disconnect from the socket too.

Case 1x

If the incoming message starts with byte 1, the server understands that it is a query request. Then it reads the remaining message bytes. First, it checks if the provided token is correct. If it isn't, the client is informed and the connection is closed on both the file and command sockets.

```
134 System.out.println("IN QUERY PHASE");
135 blength=dis.readInt();
136 dis.readByte();
137 int ttoken=dis.readInt();
138 if(ttoken!=TOKEN) {
139     String s="Token error";
140     System.out.println("TOKEN ERROR at : "+threadsocket.getRemoteSocketAddress());
141     dos.writeByte(1);dos.writeByte(3);dos.writeInt(s.length());dos.writeBytes(s);
142     dos.flush();
143     disconnect();
144     stratmain.getsft(TOKEN).disconnect();
145     break;
```

If the token was correct, it checks what kind of a request was this by looking at the 2nd byte. 0 is an APOD request and 1 is a Mars Weather request. Depending on the request, the command thread(StratoServerThread) sends a Job to the file transfer thread through the hashmap in StratoServer.java.

```
180 stratmain.getsft(TOKEN).addjobs(1, "https://api.nasa.gov/insight_weather/?api_key=FCRklbRrllQC1pevtk1vAQp24VTOCBzYM41OG7vF&feedtype=json&ver=1.0", this);
181 System.out.println("ADDING MARS JOB");
```

When the file transfer thread finishes it's job and sends the hash of file back to the command thread, it sends the hash to the client.

StratoFileThread.java

This class saves the corresping clients token and creates a Queue<Job>upon creation. When it starts to run, it creates the input/output streams and datainput/dataoutput streams of the file transfer socket. Then it adds itself to the HashMap at StratoServer.java. Finally, it enters a while loop where it waits for a job. The Job is a class inside the StratoFileThread.java. It contains the job code(0 for APOD, 1 for Mars Weather), the corresponding NASA API link, and the StratoServerThread which issued the job.

```
187     private class JobNode{
188         int j;//query type APOD0 or MARS1
189         String link;//link to the nasa api
190         StratoServerThread sst;//job issuing client
191         public JobNode (int j, String l,StratoServerThread s ) {
192             this.j=j;
193             link=l;
194             sst=s;
195         }
```

When it gets a job, it connects to the corresponding NASA API. For APOD, it extracts the hd_url of the image if it exists, else it extracts the normal url. Then it downloads the image, saves it to the server, then sends its size and the image to the client through the file transfer socket. Then it sends the command thread the hash of the file.

If the job was for Mars Weather, it extracts the available sols, picks one randomly, then sends its atmosperic pressure info the the client through the file transfer socket. The same hash operations are executed.

StratoClient

The client is composed of a single java file, like the echoserver PS project. I decided this would be better for handling the scanner and using it when I need.

StratoClient.java

Upon starting the StratoClient, it automatically connects to the StratoServer welcoming socket. Then it waits for user input. The commands are “HELP!”,”LOGIN! uname”,”MARS!”,”APOD!” and “QUIT!”. The StratoClient manages the input by using a switch.

QUIT!

This command informs the the server about disconnection, disconnects from the server and stops execution.

HELP!

This command shows the possible commands for StratoClient.

LOGIN!

If the client was already logged in, this command doesn’t work. This command should be followed by a username. When this comand is entered correctly, the client messages the server as told in the project pdf. Then it waits for a respond. If the server re-requests for password, the user is asked for password again. If server messages login fail, the client disconnects and stops execution. If login was successful, the token given by server is saved, and the client connects to the port given by server with a second socket. This socket is the file transfer socket. The client informs the file transfer thread by sending its token.

APOD! Or MARS!

When this command is entered, client first checks if it is logged in. If it is, it sends the server an APOD request message(starting with bytes 1 and 0, and ends with the token). Then it waits for the hash of the file which will come from the file transfer socket. If an failure message comes instead(bytes 1 3), the client disconnects. When the hash message from the command socket comes (bytes 1 2) , the client reads the file transfer socket for the size of the file and then the file itself. After downloading the file, it hashes it and compares it to the hash sent from the server. If it is a match, a success message is displayed. If it doesn’t match, the user is informed about this. The only difference of APOD! and MARS! Is the client expects a .jpg file for APOD! request and a string for the MARS! request.

Testing

When testing the code, only 1 Main.java should be running of the StratoClient. All the clients will be connecting to this single server. Multiple StratoClient.java may be running at the same time.

Multiple clients can login as the same user at the same time. In order to test the case of hash mismatch, a random hash can be written manually inside the code.