

Indiana University Purdue University Indianapolis
Department of Computer and Information Science
CSCI 53700
Fall 2018

Assignment – 2

Devanapally Sai Krishna

Contents

Introduction:	3
Design Choice:	3
Primary choice:	3
Detailed Design:	3
Byzantine Failure:	4
Checking Data Integrity:	4
Results:	4

Introduction:

The aim of this assignment is to send a file from one system to another using sockets in java. To establish a connection between server and client an authentication will be used, while sending a file the data in the file will be encrypted. In a case where transferred file has some error server will try to resend the file 5 times. And the server may exhibit Byzantines behavior. The main concept of this assignment is to simulate end-to-end argument principle using server and socket programming.

Design Choice:

Primary choice:

1. To establish a connection between server and client using server socket programming where a user must provide an authentication for the connection to be established. This is done by using an array where index 0 is used to store a predefined username and password in index 1.
2. When the client side is run, there will be 2 options given where option 1 will be login here after entering the username and password, system will validate it with the array where the previous username and password was predefined, if they match connection will be established and ask for the file name which is to be transferred. Option 2 will be exit and here the socket will be closed.
3. After the connection is established perfectly, the server asked the file which is to be sent, on the client side a message will be showed asked for the file name to be sent, after the file name is entered the file is transferred to the client from server with proper encryption.
4. If the file requested is not present on the server, it shows a message saying file not available and the connection closes.
5. To check if the file has been sent correctly, we check the size of the file on the client side and server side if both match the file has been sent correctly without any errors or else, we send the file once again.
6. The Encryption used here is AES, which is a predefined in java.Security and java.Crypto packages.

Detailed Design:

1.FileServer.java:

Here a server is Started by opening a socket in a defined port using java.net package. If the port is already in use, then the server returns a return an error code saying the port is already in use. If the connection is established, then it creates a server thread which acts as a server to clients. By this technique we can give multiple requests to a server.

Here we can simulate byzantine failure by changing the string parameter from “N” to “Y” in the while condition. We will investigate it further.

2.FileServerThread.java:

Here check for the authorization between the client and the server and take a file name as input. First when the client is run it asks for a login, in our case username and password are hard coded as saikrishna. Once the client enters the login credentials and the credentials are validated, server asks for the file name which it wants to send and if the file is available it starts the transfer or else it will display a message saying file not available.

One the server finds the specified file it will open the file and convert it into byte stream followed by encryption and send it to the client. Here if the file is not found on the server, it will close the connection.

3. FileClient.java:

Here once the client establishes the connection with the server and enters the file name which he wants, client opens a file where the data from the client will be written. The data which is being received from the server is encrypted and it is decrypted and on the client side and the data is written on the new file. Once the entire file is written client closes the connection.

Byzantine Failure:

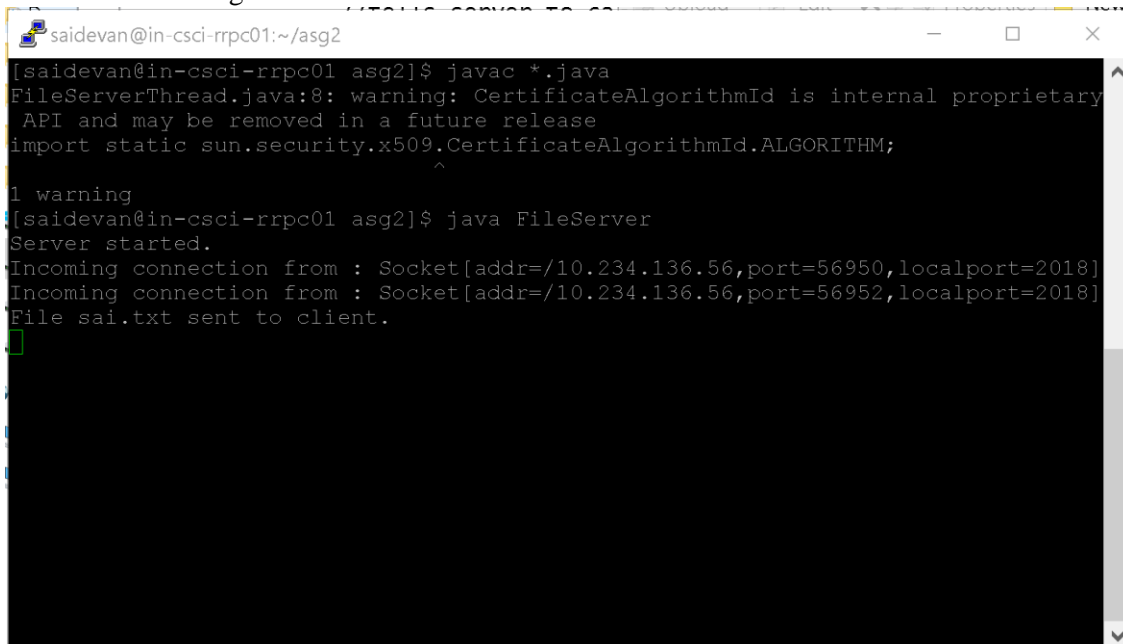
Here we simulate byzantine failure by changing the parameter from N to Y in FileServer.java, where wrong data is written on the server and the server tries 5 times before closing.

Checking Data Integrity:

To check the integrity of the file transferred we check the size of file before and after receiving the file. If they don't match we resend the file again.

Results:

1. Normal Functioning:



```
saidevan@in-csci-rrpc01:~/asg2
[saidevan@in-csci-rrpc01 asg2]$ javac *.java
FileServerThread.java:8: warning: CertificateAlgorithmId is internal proprietary
API and may be removed in a future release
import static sun.security.x509.CertificateAlgorithmId.ALGORITHM;
                                ^
1 warning
[saidevan@in-csci-rrpc01 asg2]$ java FileServer
Server started.
Incoming connection from : Socket[addr=/10.234.136.56,port=56950,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56952,localport=2018]
File sai.txt sent to client.
```

```
saidevan@in-csci-rrpc02:~/asg2
[saidevan@in-csci-rrpc02 asg2]$ java FileClient
1. Login
2. Exit

Make selection: 1
username: saikrishna
password: saikrishna
Login successful

Enter file name:
sai.txt
File sai.txt received from Server.
File successfully downloaded
[saidevan@in-csci-rrpc02 asg2]$
```

2. Byzantine Failure:

```
saidevan@in-csci-rrpc01:~/asg2
import static sun.security.x509.CertificateAlgorithmId.ALGORITHM;
^
1 warning
[saidevan@in-csci-rrpc01 asg2]$ java FileServer
Server started.
Incoming connection from : Socket[addr=/10.234.136.56,port=56950,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56952,localport=2018]
File sai.txt sent to client.
^C[saidevan@in-csci-rrpc01 asg2]$ javac *.java
FileServerThread.java:8: warning: CertificateAlgorithmId is internal proprietary
API and may be removed in a future release
import static sun.security.x509.CertificateAlgorithmId.ALGORITHM;
^
1 warning
[saidevan@in-csci-rrpc01 asg2]$ java FileServer
Server started.
Incoming connection from : Socket[addr=/10.234.136.56,port=56960,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56964,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56966,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56968,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56970,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56972,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56974,localport=2018]

```

```
saidevan@in-csci-rrpc02:~/asg2
[saidevan@in-csci-rrpc02 asg2]$ java FileClient
1. Login
2. Exit

Make selection: 1
username: saikrishna
password: saikrishna
Login successful

Enter file name:
sai.txt
File sai.txt received from Server.
Error downloading the file, file size does not match up. retrying...
File sai.txt received from Server.
Error downloading the file, file size does not match up. retrying...
File sai.txt received from Server.
Error downloading the file, file size does not match up. retrying...
File sai.txt received from Server.
Error downloading the file, file size does not match up. retrying...
File sai.txt received from Server.
Error downloading the file, file size does not match up. retrying...
File sai.txt received from Server.
5 retries finished. closing connection to server. Please contact server admin
[saidevan@in-csci-rrpc02 asg2]$
```

3. File not found:

```
saidevan@in-csci-rrpc01:~/asg2
1 warning
[saidevan@in-csci-rrpc01 asg2]$ java FileServer
Server started.
Incoming connection from : Socket[addr=/10.234.136.56,port=56960,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56964,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56966,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56968,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56970,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56972,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56974,localport=2018]
^C[saidevan@in-csci-rrpc01 asg2]$ java FileServer
Server started.
Incoming connection from : Socket[addr=/10.234.136.56,port=56980,localport=2018]
Incoming connection from : Socket[addr=/10.234.136.56,port=56982,localport=2018]
java.io.FileNotFoundException: asd.txt (No such file or directory)
    at java.io.FileInputStream.open0(Native Method)
    at java.io.FileInputStream.open(FileInputStream.java:195)
    at java.io.FileInputStream.<init>(FileInputStream.java:138)
    at FileServerThread.sendFile(FileServerThread.java:94)
    at FileServerThread.run(FileServerThread.java:39)
    at java.lang.Thread.run(Thread.java:745)
File does not exist!
```

```
saidevan@in-csci-rrpc02:~/asg2
Error downloading the file, file size does not match up. retrying...
File sai.txt received from Server.
Error downloading the file, file size does not match up. retrying...
File sai.txt received from Server.
Error downloading the file, file size does not match up. retrying...
File sai.txt received from Server.
Error downloading the file, file size does not match up. retrying...
File sai.txt received from Server.
Error downloading the file, file size does not match up. retrying...
File sai.txt received from Server.
5 retries finished. closing connection to server. Please contact server admin
[saidevan@in-csci-rrpc02 asg2]$ java FileClient
1. Login
2. Exit

Make selection: 1
username: saikrishna
password: saikrishna
Login successful

Enter file name:
asd.txt
File not found on server
[saidevan@in-csci-rrpc02 asg2]$
```