CSCI 53700-FALL 2016

Report for Assignment 2

-a simple implementation of FTP protocol

Feng Li

10/29/16

1. Problem Description

In this lab, we need to implement an simple version of FTP protocol, which can reinforce the principles of inter-process communication and the end-to-end argument.

A target system consists of two kinds of entities: server and clients. Java socket will be used for inter-process communication and data transmission. Simple credentials(user name and password) will be applied to the system to ensure security. Also, by using java security API, all file transmissions will be encrypted.

2. System Design

System Overview

The system can be demonstrated in figure 1. Server and clients communicate with each other using java sockets. For each client connection, one pair of control sockets is used, for each new file transmission a new connection will be constructed using a different port number.

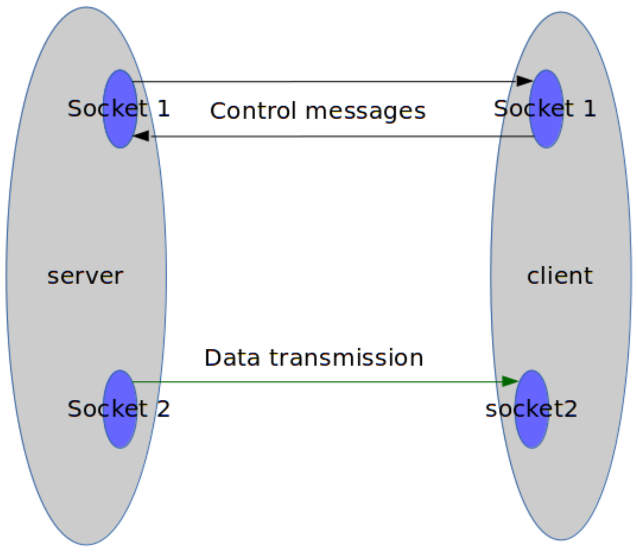

Figure 1 System Overview

User Authorization

To ensure the safety of normal system users, user authorization is used. The server maintains a table of the user name and password of each users. When one client connects to the system, it will be prompt with the choice to login or register. There will be check for the availability of user name when registering and also the correctness of password when login. The state machine in Fig 2 shows how the authorization works. In this figure, the red arrows happen when:

1. User tries to register with a already existing name
2. User tries to register with a invalid password
3. User tries to login with a not-existing username
4. User tries to login with a false password

In the current settings, server save all username and password information in a dictionary in memory(not in persistent storage). A xml file or database can be used for further reliability of the system.

The implementation of this part can be shown in the 'src/FtpProtocol' code.

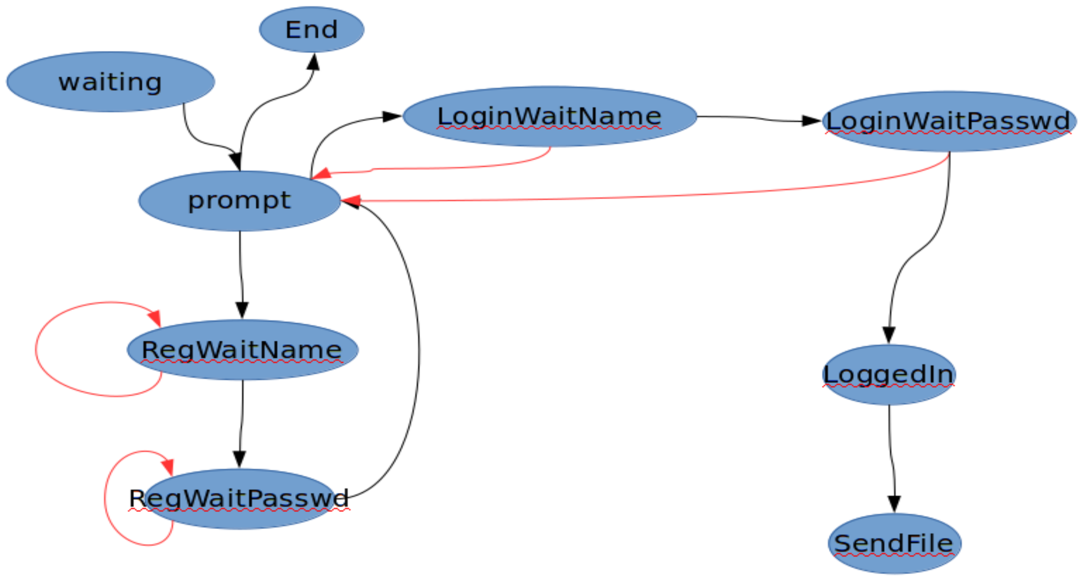

Figure 2 Simple State Machine of Authorization

Data Encryption

Since files which are transmitted may contain sensitive information and can be exposed to attach if the network is open, data need to get encrypted in the sending end and decrypted in the receiving end.

Java CipherInputStream and CipherOutputStream is used both in the sending end and receiving end. An Cypher object will be constructed in both ends using the same security key in both ends. AES encryption method is used using a String password as input. Both server and client will have a copy of that string in advance so they can identify each other. Details can be seen in 'src/dataTransform.java'.

File transimission

Using the CypherStream described above, the programming model can become much easier:

After finding the requested path in the server, in the server side, data will be read from the corresponding FileInputStream and redirected into CypherOutputStream; in the client side, data will be read from CypherInputStream and redirected into the destination FileOutputStream, as described in figure 3.

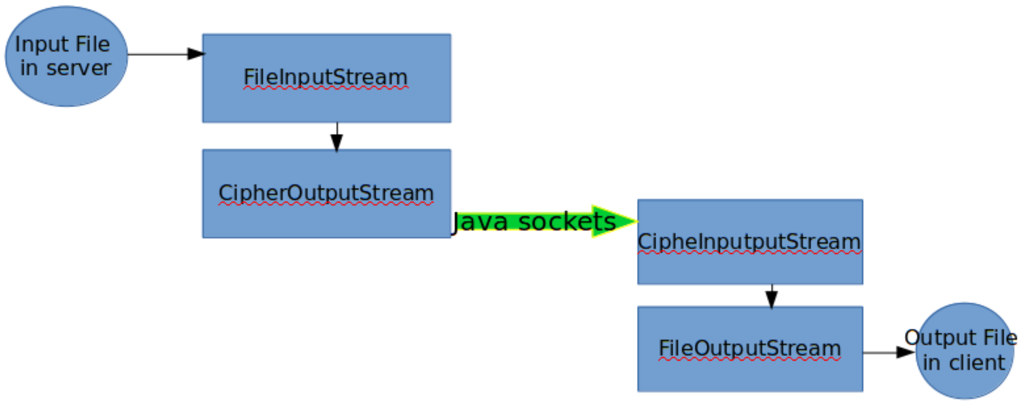

Figure 3 file transmission

Failure Model

To satisfy the integrity property of reliable communication, checksum will be used each time the client receives a file. For single-file transmission, if it is not successful, the transferring task should be retried. After several retries, if client still cannot get the desired file, the client will get disconnected. In this way, even some arbitrary failure can be effectively converted into omission failures. And those omission failures can be addressed by re-transmits messages that do not arrive or get corrupted in the destination.

3. Implementation

Java sockets are used for inter-process communication. Standard FTP protocol use port 21 for control and port 20 for data transfer. I also design my communication system in this way since it can separate these two parts and make sure that the system(communication channel) is not corrupted even when data transmission is not successful.

To be more detailed, once a client get connected, the ‘control sockets’ are constructed, and for each file transfer, another pair of ‘data sockets’ will be used to transmit the file from server to client. The ‘data sockets’ will close each time the file transmission finishes or turns out not successful.

4. Results

As shown in the sample outputs(Results.txt) from the source code package, the implemented system can identify the principal of clients using credentials and the correction of file transfer is guaranteed by using checksum, reties and time-outs(at most three retries).

5. Conclusion

Ftp is a typical server-client distributed system. Using API like java sockets, inter-process programming can become much easier.

For the sake of security, authorization(credentials) is used to let server identify the principal behind any particular invocation; encryption is used in communication channel to ensure the privacy and integrity of the data transmitted across it.

In the aspects of reliability, checksum is used to ensure the correctness of transmission. Retries and timeout is used to make the system reliable.

End-to-End arguments can be demonstrated in this system in both data transmission(retries when there is failure), encryption parts. In this way, even with a not-so-reliable underlying network, file transmissions can be secure and reliable with the knowledge and help of the two ends.