Distributed System Labwork 1

TCP File Transfer

Group 8

1 Introduction

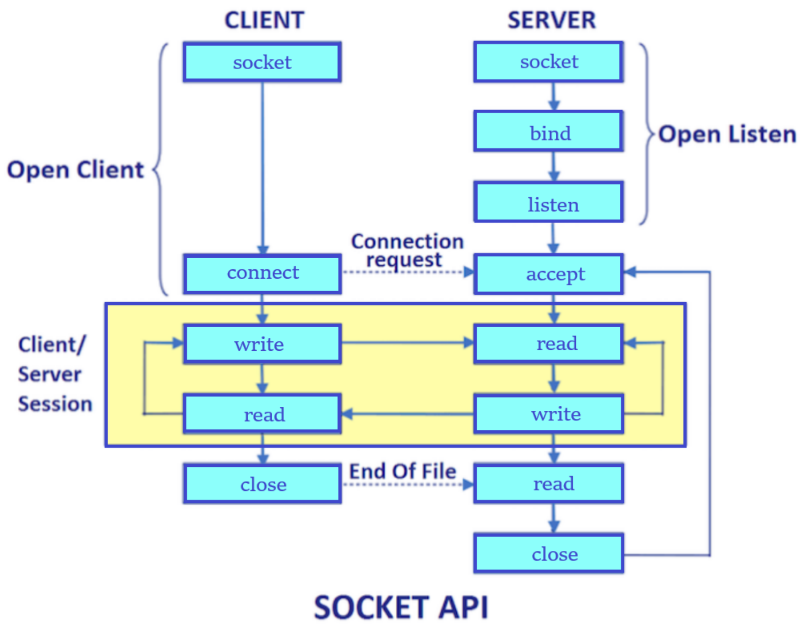
There are four main stages when implementing for transferring file data between client and server using socket:
* The Server creates a Server Socket and bind it to a specific port number and IP address.
* The Server listens for a connection from the Client on the created socket and accept it. Then the Server will accept incoming connection from the Client.
* Data written to the output stream of the Client will be received on the input stream of the Server Socket. And the data written into the output stream on the Server Socket will be received on the input stream of the Client.
* The sessions keeps going on and sending data until the Server or the Client closes the connection.

In order to organize the system, FTP Server and FTP User are needed. In the FTP Server, there are Server PI and Server DTP while in FTP User are User DTP, User PI and User Interface. All factor inside of FTP Server and FTP User are connected to each others. The Server DTP is connected with User DTP via Data Connection while Server PI is connected with User PI via FTP Commands/ FTP Replies. Given the limited budget, the server and the client is one the same machine. Therefore, it is crucial that the server and the client are different folder to avoid overwriting of existing files.

TCP data is organized as a stream of bytes, much like a file. The datagram nature of the network is concealed.

* Client and server socket programs * Call sequence in socket programs * Blocking, nonblocking, and asynchronous socket calls * Testing a program using a miscellaneous server * Testing a local machine using a loopback address * Accessing required data sets

2 Socket API Illustration



3 Client Socket

Coding Implementation

```
import java.io.*;
import java.net.*;

class ClientSocket{

 public static void main(String argv[]) throws Exception{
  String messageSent;
  String messageReceipt;

  BufferedReader userInput = new BufferedReader( new InputStreamReader(System.in));
  //Data entry


  Socket clientSocket = new Socket("127.0.0.1", 8888);
  //Create a socket client which localhost receives at port 8888

  DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());
  //Create sending stream

  BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
  //Create receiving stream


  messages = userInput.readLine();

  outToServer.writeBytes(messages + '\n');
  //Sending


  messageReceipt = inFromServer.readLine();
  //Receive from sever (text transformed into uppercase format to make it easier to notice

  System.out.println("FROM SERVER: " + messageReceipt);


  clientSocket.close();
 }

}
```

4 Server Socket

Coding Implementation

```
import java.io.*;
import java.net.*;

class ServerSocket{

  public static void main(String argv[]) throws Exception{
    String clientMessage;
    String sentMessage;

    ServerSocket welcomeSocket = new ServerSocket(8888);
    //Create a socket which open port 8888 to transfer out


    while(true){
      Socket connectionSocket = welcomeSocket.accept();
      //Accepting incomming connection from client


      BufferedReader inFromClient =
      new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));
      //Create a receiving stream

      DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());
      //Create a sending stream


      clientSentence = inFromClient.readLine();


      System.out.println("Received: " + clientMessage);
      //Original information received from Client

      sentMessage = clientMessage.toUpperCase() + '\n';
      //Changing into uppercase

      outToClient.writeBytes(capitalizedSentence);
      //Sending back to c1
    }

  }

}
```

5 Group members

Bi9-102 Vu Duc Hieu
Bi9-180 Dang Yen Nhi
Bi9-090 Tran Truong Giang
Bi9-237 Dao Minh Vu
Bi9-079 Tran Dai Duong