Department of
Computer Science and Engineering

University of Dhaka

Title: Implementing Bit Stuffing and De-stuffing Using Socket Programming

CSE 2213: Data and Telecommunication Lab

Batch: 29/2nd Year 2nd Semester 2024

Course Instructors

Dr. Md. Mustafizur Rahman (MMR)
Mr. Palash Roy (PR)

1 Objective(s)

- To gather knowledge of simple one-way socket programming
- To implement client/server applications that communicate using socket programming.
- To implement bit stuffing on the client side and de-stuffing on the server side using Java socket programming..

2 Problem analysis

Socket is mainly the door between the application process and end-end-transport protocol. Java Socket programming can be connection-oriented or connectionless.

- Socket and ServerSocket classes are used for connection-oriented socket programming
- DatagramSocket and DatagramPacket classes are used for connection-less socket programming

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: **Socket** and **ServerSocket**. The **Socket** class is used to communicate client and server. Through this class, we can read and write message. The **ServerSocket** class is used at server-side. The **accept()** method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side. The simple flow-chart of socket programming can be summarized as follows:

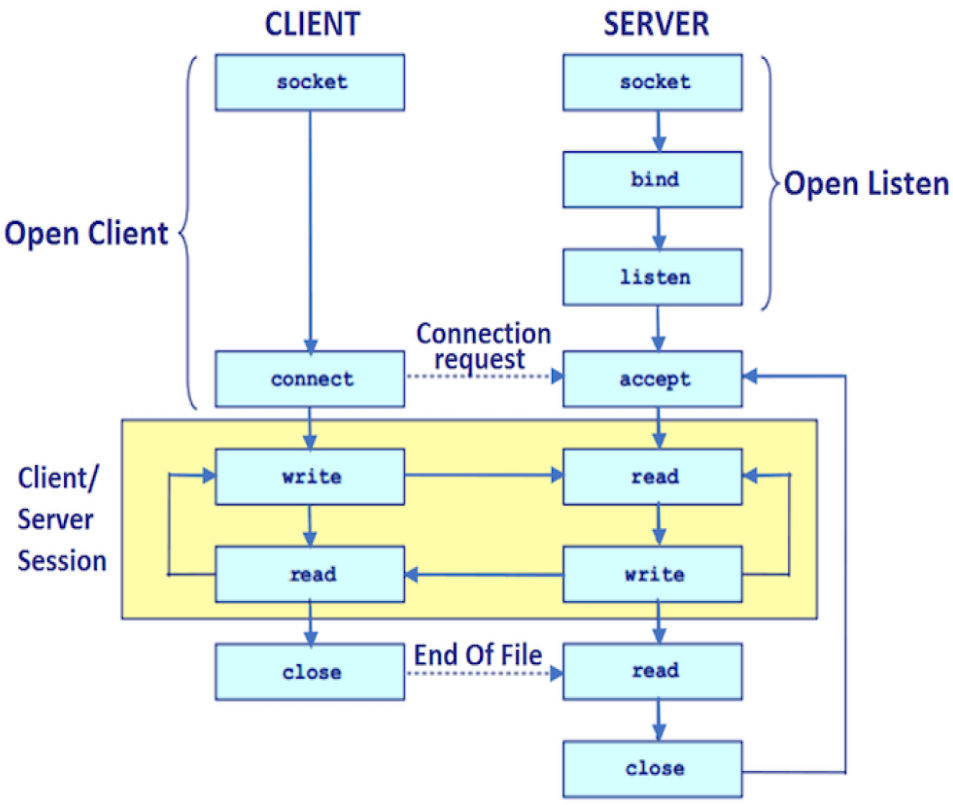

Figure 1: Flow chart

3 Procedure

3.1 Client Side Programming

In the client side, we need to divide the whole process into three subsection.

3.1.1 Establish a Socket Connection

To connect to other machine we need a socket connection. A socket connection means the two machines have information about each other's network location (**IP Address**) and **TCP port**. The **java.net.Socket** class represents a Socket. To open a socket, we have to write the following line:

Socket socket = new Socket("127.0.0.1", 5000)

Here, we have seen there have two arguments.

- First argument – **IP address of Server**. (127.0.0.1 is the IP address of localhost, where code will run on single stand-alone machine).
- Second argument – **TCP Port**. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

3.1.2 Communication

To communicate over a socket connection, streams are used to both input and output the data. In the one-way socket programming, client only sends data to the server. Therefore, **getOutputStream()** method is used to send the output through the socket.

3.1.3 Closing the connection

The socket connection is closed explicitly once the message to server is sent.

3.2 Server Side Programming

Similar to Client Side Programming, there are three phases of Server Side Programming.

3.2.1 Establish a Socket Connection

To write a server application two sockets are needed.

- A **ServerSocket** which waits for the client requests (when a client makes a new Socket()). This socket is also known as handshaking socket.
- A plain old **Socket** socket to use for communication with the client. This socket is known as communication port.

3.2.2 Communication

In the one-way socket programming, server is used to receive data from the client side. Therefore, **getInputputStream()** method is used to receive the output through the socket from the client side.

3.2.3 Close the Connection

After finishing, it is important to close the connection by closing the socket as well as input/output streams. The detailed algorithms of the server side and client side connections are given in Algorithm 1 and 2.

Algorithm 1: Algorithm of Server Side Socket Connection

1: Step - 1: Create a ServerSocket object namely handshaking socket which takes port number as input
2: Step - 2: Create a plain Socket object that accepts client request
3: Step - 3: Create an object of DataInputStream class which is used for read data
4: Step - 4: Read the data from the client side using readUTF() function and print that data until client sends "Stop"
5: Step -5: Close the connection

Algorithm 2: Algorithm of Client Side Socket Connection

1: Step - 1: Create a Socket object which takes IP address and port number as input.
2: Step - 2: Create an object of DataOutputStream class which is used for send data to the server side.
3: Step - 3: Create another object of BufferedReader class which is used for storing data in the buffer
4: Step - 4: Send the data to the server side using writeUTF() function and until client sends "Stop"
5: Step -5: Close the connection

4 Implementation in Java

```
/* Server Side Code */
import java.io.DataInputStream;
import java.io.IOException;
import java.net.*;

public class ServerOneWay {
    public static void main(String[] args) throws IOException{
        ServerSocket ss = new ServerSocket (5000);
        System.out.println("Server is connected at port no: " + ss.getLocalPort
            ());
        System.out.println("Server is connecting\n");
        System.out.println("Waiting for the client\n");
        Socket s = ss.accept();
        System.out.println("Client request is accepted at port no: " + s.getPort
            ());
        System.out.println("Server's Communication Port: "+s.getLocalPort());
        DataInputStream input = new DataInputStream(s.getInputStream());
        String str = "";

        while(!str.equals("stop")){
        str = input.readUTF();
        System.out.println("Client Says: "+str );
        }
        s.close();
        input.close();
    }

}
```

```
/* Client Side Code */
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.*;

public class ClientOneWay {
    public static void main(String[] args) throws IOException{
        Socket s = new Socket ("localhost", 5000);
        System.out.println("Client Connected at server Handshaking port " + s.
            getPort());
        System.out.println("Client's communcation port " + s.getLocalPort());
        System.out.println("Client is Connected");
        System.out.println("Enter the messages that you want to send and send \"
            stop\" to close the connection:");
```

```
15          DataOutputStream output = new DataOutputStream(s.getOutputStream());
16          BufferedReader read = new BufferedReader(new InputStreamReader(System.in
                ));
17          String str = "";
18          while(!str.equals("stop")){
19          str = read.readLine();
20          output.writeUTF(str);
21          }
22
23          output.close();
24          read.close();
25          s.close();
26      }
27  }
```

5 Input/Output

To execute the above code, at first you need to run server side code, then run client side code. Output of the client side programming is given below.

```
Client Connected at server Handshaking port 5000
Client's Communication Port: 56190
Client is Connected
Enter the messages that you want to send and send "stop" to close the connection:
Hello
Hi
Bye
stop
```

Output of the server side programming is given below.

```
Server is connected at port no: 5000
Server is connecting
Waiting for the client
Client request is accepted at port no: 56190
Server's Communication Port: 5000
Client Says: Hello
Client Says: Hi
Client Says: Bye
Client Says: stop
```

6 Discussion & Conclusion

Based on the focused objective(s) to understand socket programming, this task will help us to learn about the basic structure of one-way socket programming. The additional lab exercise of two-way socket programming will help us to be confident in the fulfillment of the objective (s).

7 Lab Task (Please implement yourself and show the output to the instructor)

Establish a two-way TCP connection in between a server process, running on host A, and a client process, running on host B. Implement a bit Stuffing & De-Stuffing algorithm and verify its functionality with a sample input. The client will take a sample input data sequence and stuff it using your algorithm. The server will receive the stuffed string and de-stuff it to retrieve the original data sequence.

7.1 Background Theory

Bit stuffing is a technique used in computer networks to ensure data is transmitted correctly. The data link layer is responsible for framing, which is the division of a stream of bits from the network layer into manageable units (called frames). Frames could be of fixed size or variable size. In variable-size framing, we need a way to define the end of the frame and the beginning of the next frame. A Bit stuffing is the insertion of non-information bits into data. Note that stuffed bits should not be confused with overhead bits. Overhead bits are non-data bits that are necessary for transmission (usually as part of headers, checksums, etc.).

In Bit stuffing, a common rule is that after five consecutive 1s, a 0 bit is stuffed in the input bit stream. On the other hand, Bit de-stuffing is the process of removing the stuffed bits in the output stream. A pictorial view is depicted in Fig 2.

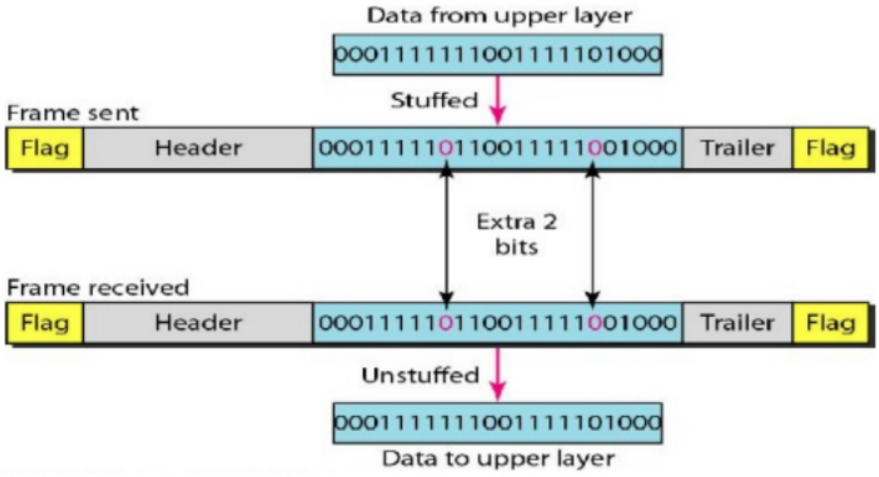

Figure 2: Bit stuffing and de-stuffing

7.2 Problem Analysis

A detailed step-by-step algorithms of client and server side are as follows

7.2.1 Bit Stuffing Algorithm (Client)

1. Initialize a counter count = 0 and an empty string stuffed.
2. For each bit in the input binary string:
   - If bit is 1, increment count and append to stuffed.
   - If count == 5, append '0' (stuffed bit) and reset count.
   - If bit is 0, append it to stuffed and reset count.
3. Send the stuffed string to the server over a TCP socket.

7.2.2 Bit De-Stuffing Algorithm (Server)

1. Read the received stuffed string.
2. Initialize count = 0 and empty string destuffed.
3. Traverse each bit:
   - If bit is 1, increment count and append.
   - If count == 5, skip the next bit (the stuffed '0') and reset count.
   - If bit is 0, append it and reset the count.
4. Display the original binary string.

8 Policy

Copying from the Internet, classmates, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.