



UNIVERSITY OF DHAKA

Department of Computer Science and Engineering

CSE-3111 : Computer Networking Lab

Lab Report 2 :Introduction to Client-Server
Communication using Socket Programming

Submitted By:

Name : Ahaj Mahhin Faiak

Roll No : 01

Name : Fardin Faiyaz

Roll No : 17

Submitted On :

January 31, 2024

Submitted To :

Dr. Md. Abdur Razzaque

Md Mahmudur Rahman

Md. Ashraful Islam

Md. Fahim Arefin

1 Introduction

The preliminary objective of this lab is to get familiar with the socket programming paradigm to learn how two processes establish connections in a client-server manner running on two different hosts in a local area network. This knowledge enhances our understanding in client-server architecture and helps grasp the intricacies of communication over TCP/IP protocol.

The second part of the lab-work also presents a glimpse of how modern networking applications are developed and highlights the concerns that are needed to be addressed.

1.1 Objectives

- Introduction to the Client-Server Architecture.
- Design and implement client-server communication systems using socket programming in a preferred programming language.
- Test the reliability and error handling of the client-server communication system, and evaluate its performance.

2 Theory

Client-Server architecture is a critical paradigm for communication among various devices in a network system. Socket programming facilitates the interaction, allowing processes to communicate over a network using the TCP/IP protocol suite. Understanding these concepts is essential for gaining insight into the workings of the internet.

Socket programming serves as a technique that facilitates the communication between different processes running on different hosts. In order to correctly identify the target host over the network, socket programming introduces the concept of a **socket**. A socket is an abstraction to represent the endpoint of a two-way communication between two devices over the network. It comprises of the host's IP address and the port number. The IP address aids in locating the target host over the network while the port number assists in determining the target process.

In Client-Server paradigm, one machine remains faithfully the service or resource provider throughout. Other devices attached to that network requests for services and resources from this machine. The machine who provides is known as the **server** and the machines requesting services are known

as the **clients**. Through Socket Programming, the clients are able to communicate with each-other indirectly through the server. The server handles the clients messages and facilitates the interactions between them.



Figure 1: Client Server Socket Programming

3 Methodology

3.1 Server

A server is a computer program or hardware device that provides services, resources, or functionality to other devices or programs, known as clients, within a network.

In the server side, we turn the server on and wait for a client to connect to it. After a client is connected to it, the server allows the client to send messages and request for functionalities according to the problem statements provided.

3.2 Client

A client refers to a software application, device, or process that initiates requests for services, resources, or data from the server over a network.

Here our client side is another application running on another machine on the same local area network. In the first problem, it acts as a simple client that orders for several functionalities like asking if a number is prime or not. In the second problem, the client acts as a customer who is presented

with several options such as checking the account balance, withdrawal of money and such. These are allowed after the verification process, of course.

3.3 Problem A : A Simple Chat App

After establishing a TCP connection with the server, the client then can ask for 3 functionalities to be provided:

- Checking if a integer number is prime or not
- Checking whether a string is palindrome or not
- Capital to smaller letter conversion for a line of text

The server does the operation and responds back by sending a message carrying the solution.

3.3.1 Server :

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    public static void main( String[] args ) {

        try {
            //create the server socket
            ServerSocket serverSocket = new
                ServerSocket(15000);
            System.out.println("Chat.Server
                connected");

            //connect to a client socket
            Socket socket = serverSocket.accept();
            System.out.println("Client connected");

            //for transmitting and receiving data
            ObjectInputStream ois = new
                ObjectInputStream(socket.getInputStream());
            ObjectOutputStream oos = new
                ObjectOutputStream(socket.getOutputStream());
```

```

while ( true ){

    //reading messages sent by
    client
    String in = (String)
        ois.readObject();
    System.out.println(in);

    String words[] = in.split(" ");

    String send = null;

    //implementing the code logic
    done at the server
    if(words[0].toLowerCase().equals("prime")){
        Integer n =
            Integer.parseInt(words[1]);

        send = isPrime(n);
    }
    else if (
        words[0].toLowerCase().equals("palindrome")){

        send = words[1]+ " " +
            isPalindrome(words[1]);
    }
    else{
        send = in.toUpperCase();
    }

    //sending messages to client
    oos.writeObject(send);
}

} catch (IOException e) {
    System.out.println(e.getCause());
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    throw new RuntimeException(e);
}
}

static String isPrime(Integer n){

    if(n==1){
        return n.toString() + " is a prime";
    }
}

```

```

        for (int i=2; i<n; i++){
            if (n%i==0){
                return n.toString() + " is not
                    a prime";
            }
        }
        return n.toString() + " is a prime";
    }

    static String isPalindrome(String string){

        int i = 0;
        int j = string.length()-1;

        while(i<=j){
            if(string.charAt(i) !=
                string.charAt(j)){
                return "not a palindrome";
            }

            i++;
            j--;
        }

        return "is a palindrome !!";
    }
}

```

3.3.2 Client :

```

import java.io.*;
import java.net.*;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) {
        try {
            // a socket to connect to server
            Socket socket = new
                Socket("192.168.1.102", 15000);
            System.out.println("Connected to server
                1");

            //objects created for sending and
                receiving data

```

```

        ObjectOutputStream oos = new
            ObjectOutputStream(socket.getOutputStream());
        ObjectInputStream ois = new
            ObjectInputStream(socket.getInputStream());

        //for taking clients inputs
        Scanner scanner = new
            Scanner(System.in);

        while (true) {
            System.out.println("What do u
                want to know? : ");
            String in = scanner.nextLine();

            //sending messages to server
            oos.writeObject(in);

            //reading messages sent from
            servers
            String serverMsg = (String)
                ois.readObject();
            System.out.println(serverMsg);

        }

    } catch (Exception e) {
        System.out.println(e);
    }
}

```

4 Problem B : A Banking System

A simplified bank ATM system. Here the user is the client and the bank server acts as the Server and provides service to the user. The User after connection is to Provide:

- ID
- Password

After the User is verified They can

- Withdraw Cash from ATM.

- See Current Balance
- Exit

The process is similar to that of Problem A

4.1 BankServer:

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Random;

public class server {
    private static double balance = 1000.0;

    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(8080);
        System.out.println("Bank server is running...");

        while (true) {
            Socket clientSocket = serverSocket.accept();
            new Thread(() ->
                handleClient(clientSocket)).start();
        }

        private static void handleClient(Socket clientSocket) {
            try (
                ObjectOutputStream out = new
                    ObjectOutputStream(clientSocket.getOutputStream());
                ObjectInputStream in = new
                    ObjectInputStream(clientSocket.getInputStream()))
            {
                while (true) {

                    String action = (String) in.readObject();

                    if (shouldSimulateError()) {
                        out.writeObject("Error");
                        continue;
                    } else if (action.equals("withdraw")) {
                        double amount = (Double) in.readObject();
                        if (amount > 0 && balance >= amount) {
                            balance -= amount;
                            out.writeObject("Withdrawal successful.
                                Remaining balance: $" +
                                    String.valueOf(balance));
                        }
                    }
                }
            }
        }
    }
}
```



```

        } else {
            out.writeObject("Invalid withdrawal
                             amount or insufficient funds.");
        }
    } else if (action.equals("getBalance")) {
        out.writeObject("your current balance is" +
            String.valueOf(balance));
    } else if (action.equals("exit")) {
        break;
    }
}
} catch (IOException | ClassNotFoundException e) {
    e.printStackTrace();
}
}

private static boolean shouldSimulateError() {
    Random random = new Random();
    int randomNumber = random.nextInt(100);
    return randomNumber < 90;
}
}

```

4.2 Bank Client :

```

import java.io.*;
import java.net.Socket;
import java.util.Random;
import java.util.Scanner;

public class client {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try (Socket socket = new Socket("localhost", 8080);
            ObjectOutputStream out = new
                ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream in = new
                ObjectInputStream(socket.getInputStream())) {

            System.out.println("Connected to the bank server.");

            while (true) {

                Thread.sleep(300);

                //          System.out.println("\nChoose an action:");
                //          System.out.println("1. Withdraw");
            }
        }
    }
}

```

```

//          System.out.println("2. Check Balance");
//          System.out.println("3. Exit");
//          System.out.print("Enter your choice: ");

//          int choice = scanner.nextInt();
//          scanner.nextLine(); // consume the newline
character

int choice = 2;

long startTime = System.nanoTime(); // Capture
start time

boolean validResponse = false;

while(!validResponse) {

    if(sendError()){
        choice = 4;
    }else{
        choice = 2;
    }

    switch (choice) {
        case 1:
            System.out.print("Enter the
withdrawal amount: $");
            double withdrawalAmount =
scanner.nextDouble();
            out.writeObject("withdraw");
            out.writeObject(withdrawalAmount);

//          String withdrawalResponse = (String)
in.readObject();
//          System.out.println(withdrawalResponse);
break;

        case 2:
            out.writeObject("getBalance");
            double balance = (double)
//          in.readObject();
//          System.out.println("Your current
balance: $" + balance);
break;

        case 3:
            out.writeObject("exit");
            System.out.println("Exiting the

```

```

        bank application. Goodbye!");
        return;

    case 4:
        out.writeObject("Error");
        break;

    default:
        System.out.println("Invalid choice.
            Please enter a valid option.");
        continue;
    }

    String response = (String) in.readObject();

    if(response.equals("Error")){

    }else{
        break;
    }
}

long endTime = System.nanoTime(); // Capture
    end time
long elapsedTime = endTime - startTime; //
    Calculate elapsed time
System.out.println("Time between request and
    reply: " + elapsedTime + " nano seconds");
}
} catch (IOException | ClassNotFoundException |
    InterruptedException e) {
    e.printStackTrace();
}
}

private static boolean sendError(){
    Random random = new Random();
    int randomNumber = random.nextInt(100);
    return randomNumber < 90;
}
}

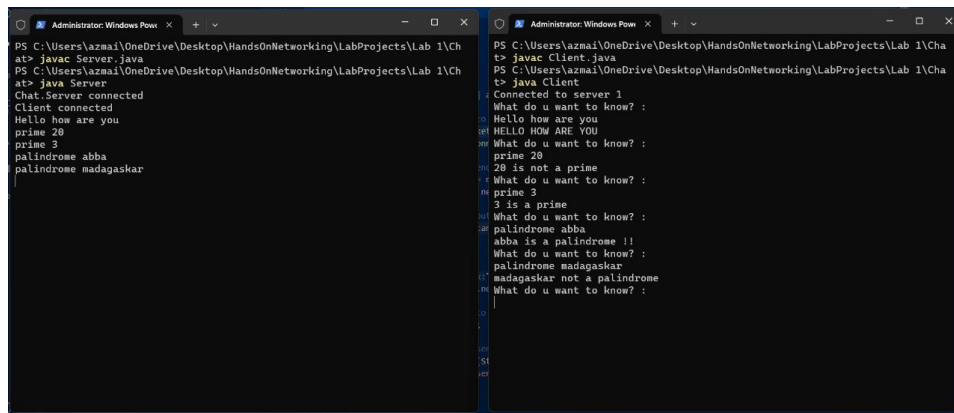
```

5 Experimental result

5.1 Problem A : A Simple Chat App

The client send a message "prime x"(x is an integer number) specify it wants to know if x is prime or not;it send "palindrome str" (str is a string) to specify that it wants to know whether "str" is a palindrome or not, or just sends a normal message to server.

The server responds back by doing appropriate calculation and sends a response back carrying the evaluation result.



```
PS C:\Users\azmai\OneDrive\Desktop\HandsOnNetworking\LabProjects\Lab 1\Cha
at> javac Server.java
PS C:\Users\azmai\OneDrive\Desktop\HandsOnNetworking\LabProjects\Lab 1\Cha
at> java Server
Chat.Server connected
Client connected
Hello how are you
prime 20
prime 3
palindrome abba
palindrome madagaskar

PS C:\Users\azmai\OneDrive\Desktop\HandsOnNetworking\LabProjects\Lab 1\Cha
t> javac Client.java
PS C:\Users\azmai\OneDrive\Desktop\HandsOnNetworking\LabProjects\Lab 1\Cha
t> java Client
Connected to server 1
What do u want to know? :
Hello how are you
HELLO HOW ARE YOU
What do u want to know? :
prime 20
20 is not a prime
What do u want to know? :
prime 3
3 is a prime
What do u want to know? :
palindrome abba
abba is a palindrome !!
What do u want to know? :
palindrome madagaskar
madagaskar not a palindrome
What do u want to know? :
```

Figure 2: A Simple Chat App

5.2 Problem B : A Banking System

User can Access the server by entering appropriate ID and Password The User can Withdraw Money. It is done by sending request to Server. Error Implementation showed following result:

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\
Connected to the bank server.

Choose an action:
1. Withdraw
2. Check Balance
3. Exit
Enter your choice: 1
Enter the withdrawal amount: $20
Withdrawal successful. Remaining balance: $980.0
Time between request and reply: 876123100 nano seconds

Choose an action:
1. Withdraw
2. Check Balance
3. Exit
Enter your choice: 2
your current balance is980.0
Time between request and reply: 1149700 nano seconds

Choose an action:
1. Withdraw
2. Check Balance
3. Exit
Enter your choice: 3
Exiting the bank application. Goodbye!

Process finished with exit code 0
```

Figure 3: A Bank ATM Code

5.2.1 Error Handling

Error implementation Was done in two steps. The first step is compromised of an erroneous message sent from the client. In this case, the client, sends an erroneous message to the server and the server notifies the client regarding this and the client sends the same message via the connection again. Hence lengthening the time required.

The second step was an error on the server side. In this case, the server sends an error message to the client and the client requests again to the server to send the proper reply. This has been tested for several percentages, ranging from nothing that is zero to 90

The First Graph portrays the Error Percentage and Average Time Graph when the Server returns error to client

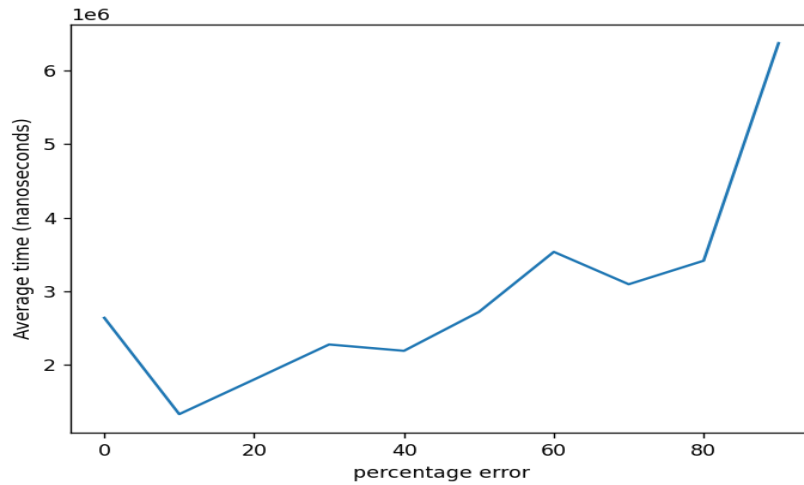


Figure 4: Server Error Time Graph

The Second Graph portrays the Error Percentage and Average Time Graph when the Client sends erroneous message to Server



Figure 5: Client Error Time Graph

6 Experience

1. We gained first-hand experience in constructing a networking application utilizing socket programming techniques.
2. We had the opportunity to delve into the intricacies of Client-Server architecture, understanding its mechanisms and functionalities.
3. We experienced a few of the experience handling basic level errors and exceptions in building networking applications through this lab.

References

- [1] Computer networking : a top-down approach 8th ed.
- [2] Client-Server paradigm with Socket Programming :
https://www.linkedin.com/pulse/introduction-socket-programming-aneshka-goyal?trk=articles_directory
- [3] Socket Programming in Java:
<https://www.geeksforgeeks.org/socket-programming-in-java/>