

**Vidyavardhini’s College of Engineering & Technology**

Department of Computer Engineering Academic Year : 2024-25

|  |  |  |  |
| --- | --- | --- | --- |
| **Class:** | **BE** | **Semester:** | **VIII** |
| **Course Code:** | **CSL801** | **Course Name:** | **Distributed Computing Lab** |

|  |  |
| --- | --- |
| **Name of Student:** | **Pratima Dinkar Bombe** |
| **Roll No. :** | **07** |
| **Division:** | **-** |
| **Experiment No.:** | **01** |
| **Title of Experiment:** | **Inter-Process Communication** |
| **Date of Submission:** | **06/01/2025** |
| **Date of Correction:** | **20/01/2025** |

Evaluation

|  |  |  |
| --- | --- | --- |
| **Performance Indicator** | **Max. Marks** | **Marks Obtained** |
| Performance | 5 |  |
| Understanding | 5 |  |
| Journal work and timely submission | 10 |  |
| Total | 20 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Performance Indicator** | **Exceed Expectations (EE)** | **Meet Expectations (ME)** | **Below Expectations (BE)** |
| Performance | 4-5 | 2-3 | 1 |
| Understanding | 4-5 | 2-3 | 1 |
| Journal work and timely submission | 8-10 | 5-8 | 1-4 |

**Checked by**

**Name of Faculty : Ms. Swati Varma**

**Signature :**

**Date :**

EXPERIMENT 1

**Aim:** To implement inter-process communication

**Objective:** Develop a program to implement inter-process communication using socket programming

**Theory**:-

URLs and URLConnections provide a relatively high-level mechanism for accessing resources on the Internet. Sometimes your programs require lower-level network communication, for example, when you want to write a client-server application.

In client-server applications, the server provides some service, such as processing database queries or sending out current stock prices. The client uses the service provided by the server, either displaying database query results to the user or making stock purchase recommendations to an investor. The communication that occurs between the client and the server must be reliable. That is, no data can be dropped and it must arrive on the client side in the same order in which the server sent it.

TCP provides a reliable, point-to-point communication channel that client-server applications on the Internet use to communicate with each other. To communicate over TCP, a client program and a server program establish a connection to one another. Each program binds a socket to its end of the connection. To communicate, the client and the server each reads from and writes to the socket bound to the connection.

**What Is a Socket?**

A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program. The java.net package provides two classes--Socket and ServerSocket--that implement the client side of the connection and the server side of the connection, respectively.

**Socket class**

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Important methods

Method Description

1) public InputStream getInputStream() returns the InputStream attached with this socket.

2) public OutputStream getOutputStream() returns the OutputStream attached with this socket.

3) public synchronized void close() closes this socket

**ServerSocket class**

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

Important methods

Method Description

1) public Socket accept() returns the socket and establish a connection between server and client.

2) public synchronized void close() closes the server socket.

**Code:**

**MyServer.java**

import java.io.\*;

import java.net.\*;

public class MyServer {

public static void main(String[] args){

try{

ServerSocket ss=new ServerSocket(6666);

Socket s=ss.accept();//establishes connection

DataInputStream dis=new DataInputStream(s.getInputStream());

String str=(String)dis.readUTF();

System.out.println("message= "+str);

ss.close();

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

}

}

**MyClient.java**

import java.io.\*;

import java.net.\*;

public class MyClient {

public static void main(String[] args) {

try{

Socket s=new Socket("localhost",6666);

DataOutputStream dout=new DataOutputStream(s.getOutputStream());

dout.writeUTF("Welcome to DC Practicals");

dout.flush();

dout.close();

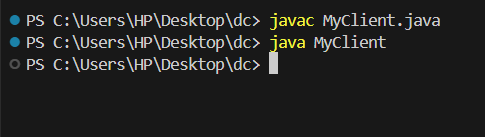
s.close();

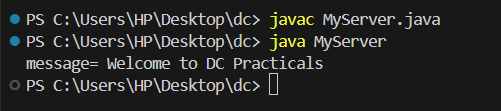
}catch(Exception e){System.out.println(e);}

}

}

**Output:**

****

****

**Conclusion**: Socket programming enables inter-process communication (IPC) between processes on the same system or across a network. Unlike shared memory, pipes, or message queues, sockets support remote communication, making them ideal for client-server applications. In this experiment, the client sends "Welcome to DC Practicals" to the server, demonstrating reliable TCP-based data exchange. While sockets offer flexibility and scalability, they may introduce network overhead compared to other IPC methods.