# Lab Report-01:TCP/IP Connection


## Gandaki College of Engineering and Science

*Distributed System*

***Lab Experiment: TCP/IP protocol***

Submitted By:

Name: Dipendra Raut Kurmi

Roll.No: 18

Batch: 2021SE

Submitted To:

Er. Amrit Poudel

Lecturer at Gandaki College Of Enginering and Science

## Objective:

To understand and implement a simple TCP/IP connection using Java, demonstrating how a client and server communicate over a network.

## Theory:

TCP/IP stands for Transmission Control Protocol / Internet Protocol. It is the foundational communication protocol suite used on the Internet and most modern networks. TCP/IP ensures reliable communication between devices and consists of multiple layers that work together to deliver data accurately and efficiently.

Key Components:
- IP (Internet Protocol): Responsible for addressing and routing data packets.
- TCP (Transmission Control Protocol): Provides reliable, ordered, and error-checked delivery of data.

TCP/IP Model Layers:
1. Application Layer: Provides services to the user (HTTP, FTP, SMTP, DNS).
2. Transport Layer: Manages data transmission (TCP, UDP).
3. Internet Layer: Handles addressing and routing (IP, ICMP).
4. Network Access Layer: Manages physical transmission (Ethernet, Wi-Fi).

In this lab, we implement a basic TCP client-server program where the client sends a message to the server, and the server responds after processing it.

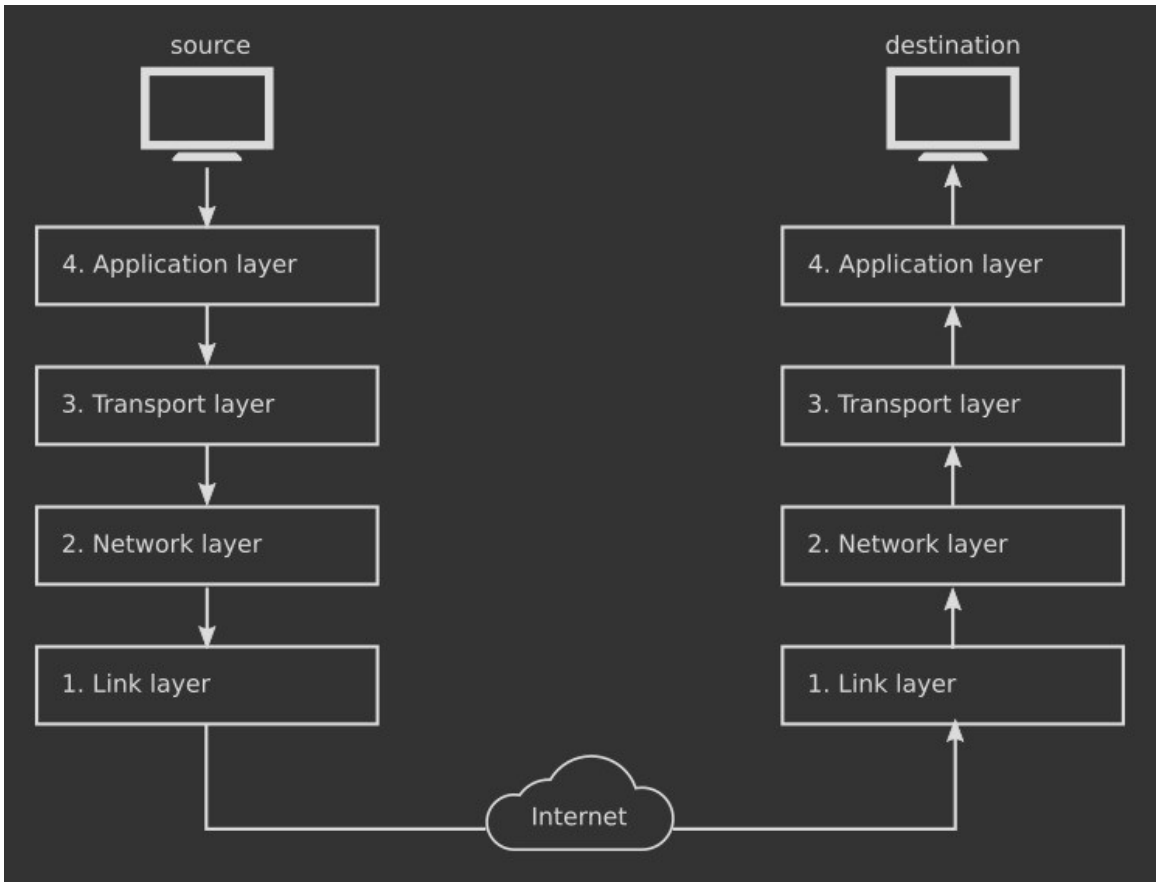


Figure: TCP/IP connection

## Code:

**// TCPServer.java**

```
import java.io.*;

import java.net.*;

public class TCPServer {
public static void main(String[] args) {
int port = 6789; // Port the server listens on

try (ServerSocket welcomeSocket = new ServerSocket(port)) {
System.out.println("Server started. Listening on port " + port);

while (true) {
// Wait for client connection
Socket connectionSocket = welcomeSocket.accept();
System.out.println("Client connected: " + connectionSocket.getInetAddress().getHostAddress());
```

```
// Set up input and output streams
BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());

// Read message from client
String clientSentence = inFromClient.readLine();
System.out.println("Received: " + clientSentence);

// Process and send response
String capitalizedSentence = clientSentence.toUpperCase() + '\n';
outToClient.writeBytes(capitalizedSentence);

// Close connection
connectionSocket.close();
System.out.println("Connection closed.");
}

} catch (IOException e) {
System.err.println("Server exception: " + e.getMessage());
e.printStackTrace();
}
}
}
```

**//TCPClient.java**

```
import java.io.*;

import java.net.*;

public class TCPClient {
public static void main(String[] args) {
String serverAddress = "127.0.0.1"; // localhost (change to server IP if remote)
int port = 6789;

try (Socket clientSocket = new Socket(serverAddress, port)) {
// Set up output and input streams
DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());
BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

// User input from console
BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
System.out.print("Enter message: ");
```

```
String sentence = userInput.readLine();

// Send message to server
outToServer.writeBytes(sentence + '\n');

// Read server response
String modifiedSentence = inFromServer.readLine();
System.out.println("FROM SERVER: " + modifiedSentence);

} catch (IOException e) {
System.err.println("Client exception: " + e.getMessage());
e.printStackTrace();
}
}
}
```

## Result:

When running the TCPServer.java and TCPClient.java programs:
1. The server listens on port 6789.
2. The client connects to the server and sends a message.
3. The server receives the message, processes it (converts it to uppercase), and sends it back to the client.
4. The client receives and displays the server's response.

**Sample output:**
Server:

```
dipendra@dipendra-Vostro-15-3510:~/Documents/BE/7th Semester/DS_lab/Lab-01$ java TCPServer
Server started. Listening on port 6789
Client connected: 127.0.0.1
Received: hello i am dipendra
Connection closed.
```

Client:

```
dipendra@dipendra-Vostro-15-3510:~/Documents/BE/7th Semester/DS_lab/Lab-01$ java TCPClient
Enter message: hello i am dipendra
FROM SERVER: HELLO I AM DIPENDRA
dipendra@dipendra-Vostro-15-3510:~/Documents/BE/7th Semester/DS_lab/Lab-01$
```

## Conclusion:

Hence, we successfully connected client and server through TCP/IP protocol.