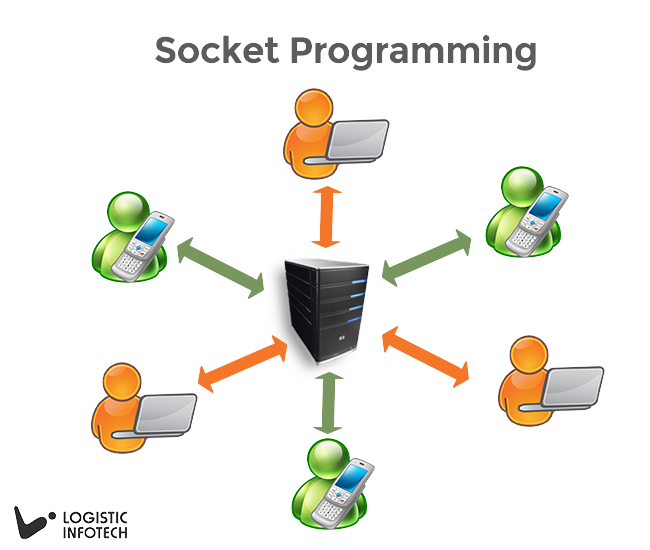
**Lab 12: Socket Programming**

**Background**

Socket programming is a way of connecting two nodes on a network to communicate with each other. Basically, it is a one-way Client and Server setup where a Client connects, sends messages to the server and the server shows them using socket connection. One socket (node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while the client reaches out to the server. Before going deeper into Server and Client code, it is strongly recommended to go through [TCP/IP Model](https://www.geeksforgeeks.org/computer-network-tcpip-model/).



**Client-Side Programming**

Before creating client’s socket, a user must decide what ‘*IP Address*’ that he want to connect to, in this case, it is the *localhost*. At the same time, we also need the ‘*Family*’ method that will belong to the socket itself. Then, through the ‘connect’ method, we will connect the socket to the server. Before sending any message, it must be converted into a byte array. Then and only then, it can be sent to the server through the ‘*send*’ method. Later, thanks to the ‘receive’ method we are going to get a byte array as answer by the server. It is notable that just like in the C language, the ‘send’ and ‘receive’ methods still return the number of bytes sent or received.

**Server-Side Programming**

In the same way, we need an ‘IP address’ that identifies the server in order to let the clients to connect. After creating the socket, we call the ‘bind’ method which binds the IP to the socket. Then, call the ‘listen’ method. This operation is responsible for creating the waiting queue which will be related to every opened ‘socket’. The ‘listen’ method takes as input the maximum number of clients that can stay in the waiting queue. As stated above, there is a communication with the client through ‘send’ and ‘receive’ methods.

Note: Don’t forget the conversion into a byte array.

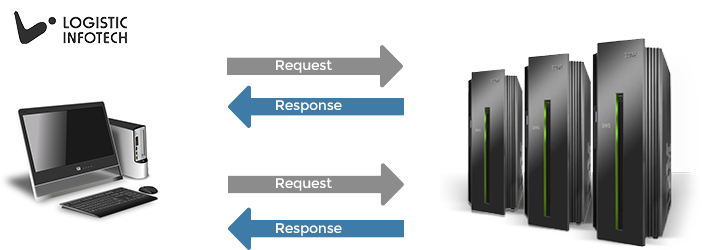
# Socket Programming For Real Time Applications

Socket Programming provides the ability to implement real time analytic, instant messaging, binary streaming, and document collaboration. Socket.IO controls the connection transparently.  Socket.IO is a custom real time transport protocol. Socket.IO needs the use of socket.IO libraries on both client and server side.  It is also used to make real time apps feasible in every browser and mobile devices and its open source too.

### What’s Socket?

A socket is an endpoint of a two-way communication link between two different programs running on the network. We can simply say that socket.IO is the Node.js module for real-time exchange data between the client and server. So let’s take an example of traditional polling, the below image shows how data can be exchanged using socket.IO programming:

Here, in the example client periodically sent the request to check with server even though there is no change in data. So, it might be possible that server has to handle a bulk of unnecessary request.



Thus, we can say a socket is one of the most fundamental technologies of computer networking. Sockets allow applications to communicate using built in mechanism of hardware and operating system. Many of the apps from today’s most popular software packages including web browsers, peer to peer file sharing systems, and even more instant messaging applications rely on the concept of Sockets.

As we all know the famous messaging app that is the WhatsApp Messenger is also using the socket programming concept for free messaging or chatting. There is not only messaging application which uses socket programming but there are also many apps which are using it to send and receive the data from client and server like Grofers, Karigar, OLA and Uber. Uber’s concept is about the Driver and user communication like sending request for taxi, at other end driver will accept or decline request). In these apps users send requests for any service or product and server provides them the proper services and products in particular time period.

**Server Code:**

using System;

using System.IO;

using System.Net.Sockets;

namespace Server01

{

class Program

{

static void Main(string[] args)

{

TcpListener server = new TcpListener(8888);

server.Start();

Console.WriteLine("Server started waiting for clients");

Socket socketForClients = server.AcceptSocket();

if (socketForClients.Connected)

{

NetworkStream ns = new NetworkStream(socketForClients);

StreamWriter sw = new StreamWriter(ns);

Console.WriteLine("Server>> welcome client.");

sw.WriteLine("Welcome Client");

sw.Flush();

StreamReader sr = new StreamReader(ns);

Console.WriteLine(sr.ReadLine());

sw.Close();

ns.Close();

}

socketForClients.Close();

}

}

}

**Client Code:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Net.Sockets;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Client01

{

public partial class Form1 : Form

{

TcpClient client = new TcpClient();

public Form1()

{

InitializeComponent();

client = new TcpClient("127.0.0.1", 8888);

NetworkStream ns = client.GetStream();

StreamReader sr = new StreamReader(ns);

richTextBox1.Text = "Server "+ sr.ReadLine();

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text!= "")

{

NetworkStream ns = client.GetStream();

StreamWriter sw = new StreamWriter(ns);

sw.WriteLine(textBox1.Text);

sw.Flush();

sw.Close();

ns.Close();

}

}

}

}



Exercises

You have to consider a chatbot for coffee shop to order the coffee from different devices.