**INDIAN INSTITUTE OF TECHNOLOGY**

**KHARAGPUR**

COMPUTER COMMUNICATION LABORATORY
A REPORT ON

# "Implementation of Inter-Process Communication using Sockets"

October 29, 2020

**Y SAI SANJEET**
(16EC35025)

**DEPT OF ELECTRONICS AND ELECTRICAL COMMUNICATION ENGINEERING**

## Contents

## OBJECTIVE

The aim is to realize a server-client style of communication between processes using socket programming.

## DISCUSSION

- Since socket server and client are necessary for the upcoming experiments as well, a simple Socket Library is implemented as a part of this experiment.
- Various functionalities associated with socket programming, such as the creation of sockets, binding a socket to an address, connecting to a socket, reading and writing data, etc. are abstracted into the respective functions in the server and client classes (more details in CODE EXPLANATION).
- This alleviates the need of writing this excessive code for future experiments, thereby making the future codes more readable.
- The library has only been tested for inter-process communication on the same machine for the scope of this experiment but should be easily extendable for communication over the internet.
- The server has to be run first followed by the client, else the client would terminate as the server is not reachable. This can be avoided by trying to connect to the server in a loop with a timeout but hasn't been implemented in the scope of this experiment.

## CODE EXPLANATION

This section gives a brief overview of the socket library (namely the classes defined in `SocketServer.h` and `SocketClient.h`, present in `src/Socket/`). The user side of the code (present in `server.cpp` and `client.cpp` in the directory `Assignment2/`) becomes very straightforward if using the library.

**SocketServer:**

- The `SocketServer` class contains the abstraction for a socket-based server.
- A static method creates a `SocketServer` object and the constructor creates a socket server using the properties given by the user.

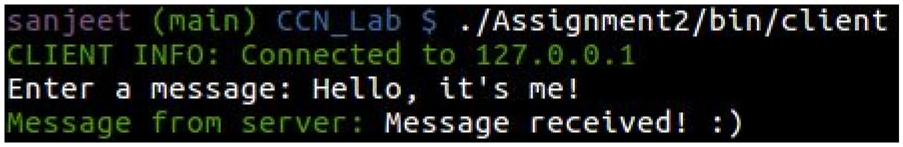
- There is a method `Bind` for binding the server to an IP address and start listening to the incoming connections. It binds to localhost by default but can be extended to bind to a specific IP address.
- The method `Accept` captures an incoming connection and returns the client's descriptor ID, which can be later used for reading data.
- The methods `Read` and `Send` are used for reading and sending data to connected clients. The data is passed and received as a C++ string for ease of usage in other parts of the user code.

**SocketClient:**

- The `SocketClient` class is very similar to the `SocketServer` class.
- The constructor creates a new socket client, and the only methods are `Connect`, `Read`, and `Send`.
- The `Connect` method is used to connect to a socket server and the `Read` and `Send` methods are exactly the same as that of `SocketServer`.

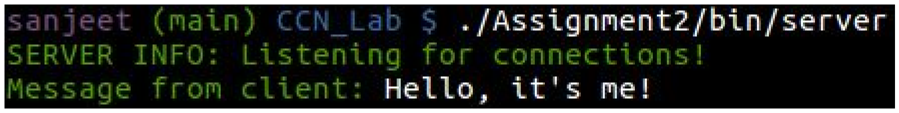
## RESULTS

The client-side:



```
sanjeet (main) CCN_Lab $ ./Assignment2/bin/client
CLIENT INFO: Connected to 127.0.0.1
Enter a message: Hello, it's me!
Message from server: Message received! :)
```

The server-side:



```
sanjeet (main) CCN_Lab $ ./Assignment2/bin/server
SERVER INFO: Listening for connections!
Message from client: Hello, it's me!
```

## CONCLUSION

- A server-client communication between processes is established.
- A simple Socket library is developed for ease of completing future experiments.

## REFERENCES

Code based on https://www.geeksforgeeks.org/socket-programming-cc