National University of Sciences & Technology (NUST)
Balochistan Campus (NBC), Department of Computer Science

**Department of Computer Science**

# EE-353: Computer Networks

**Class: BSCS Fall 2021**

## Lab 12: Socket Programming (Remote Procedure Calls)

**Date: 19 December 2023**

**Time: 02:00 - 05:00 pm**

**Instructor: Dr Mohsin Raza**

**Lab Engineer: Engr Muhammad Abid Hussain**

# Lab 12: Socket Programming (Remote Procedure Calls)

**Introduction**

This lab is about Socket programming, specifically, Remote Procedure Calls (RPC), which enables communication between processes running on different machines over a network. It provides a mechanism for remote method invocation, allowing programs to execute functions on remote systems as if they were local. By utilizing sockets, RPC facilitates the exchange of data and coordination between distributed systems, enabling efficient and transparent communication across networks. It is a fundamental concept in distributed computing and enables the development of client-server applications.

**Objectives**

This lab will review the objective of Socket Programming with Remote Procedure Calls (RPC) to enable seamless communication and interaction between distributed systems. It aims to provide a transparent mechanism for invoking functions or procedures on remote machines as if they were local. By using RPC, the objective is to simplify the development of distributed applications, promote code reusability, and enhance the efficiency of inter-process communication across networked systems. RPC facilitates the integration and collaboration of distributed components, enabling the creation of robust and scalable

**Tools/Software Requirements**

Packet Tracer/PuTTY

**Lab Tasks**

You are required to upload the lab tasks on Google Classroom/MS Teams and the name of those tasks must be in this format YourFullName_reg#.docx

Remember to comment on your code properly. Inappropriate or no comment will result in the deduction of marks.

**Q.No.01: What is Socket programming and why use this programming?**

Answer:- Socket programming is a method that enables communication between computers over a network through the use of sockets. A socket serves as an endpoint for sending or receiving data, allowing processes on different devices to exchange information. This programming paradigm is commonly employed in building networked applications and distributed systems. Here are some key aspects of socket programming:

1. **Communication Models:**
    - **Client-Server:** Socket programming often follows a client-server model, where one device (the server) provides services, and other devices (clients) connect to the server to utilize those services. This model is prevalent in various networked applications.
2. **Protocols:**
    - **TCP (Transmission Control Protocol):** Offers reliable, connection-oriented communication. It ensures that data is delivered accurately and in the correct order, making it suitable for applications where data integrity is crucial.
    - **UDP (User Datagram Protocol):** Provides connectionless, unreliable communication. It is often used in situations where low latency is more critical than guaranteed delivery, such as real-time applications.
3. **Use Cases:**
    - **Web Applications:** Socket programming is fundamental for developing web servers and supporting communication between web clients and servers.
    - **Messaging Systems:** Chat applications and messaging systems utilize socket programming to facilitate real-time communication between users.
    - **Data Transfer:** It is employed in scenarios where large amounts of data need to be transferred between systems, such as file transfers.

## Q.No.02: Basic Functionality of socket programming?

Answer:- `Socket-Desc` is n unique identifier (like a file handle) for the socket created.

`Domain` is the address family, such as `AF_INET` (address family Internet), when ports are used for communication.

`Type` is the way in which data will be transmitted, such as in a continuous, consecutive, connection-oriented, reliable stream of bytes (`SOCK_STREAM`) in independent individual, connectionless, unreliable packets (`SOCK_DGRAM`) in the form of raw bytes (`SOCK_RAW`).

`Protocol` is the low-level protocol used for data transmission. This is TCP for stream sockets and UDP for datagram sockets.

### `bind(Socket-Desc, Packed-Structure-Address)`

This function associates the server socket descriptor with information specified in the second parameter, especially the port on which the server will be listening for requests from clients.

`Socket-Desc` is the unique socket descriptor obtained from the socket function.

`Packed-Structure-Address` is a collection of information, such as the address family, the server port, any client address indicator stored in a particular format that can be understood by the `bind` function. An example is: `<unsigned short><short in network order><null padded 4 char string><8 null bytes>`.

### `connect(Socket-Desc, Packed-Structure-Address)`

The `connect` function is used by a client to establish connection to a server.

`Socket-Desc` is the client socket descriptor.

`Packed-Structure-Address` is a collection of information about the server to connect to, such as the address family, server port, and server IP address, stored in a particular format that can be understood by the

`connect` function. Here's an example: `<unsigned short><short in network order><null padded 4 char string><8 null bytes>`.

### `select(Socket-Desc/s)`

The `select` function is used to check the status of a socket descriptor to determine, for example, whether it is ready for reading or writing, or whether it has an error condition pending.

### `listen(Socket-Desc, QueueSize)`

This function is used by a connection-oriented server to specify the number of client requests that are allowed at a time and are queued for service.

`Socket-Desc` is the server socket descriptor on which the server is listening for requests from clients.

`QueueSize` specifies the number of client requests that can be queued until they can be serviced by the server on that socket. If there are more client requests than the number specified, then the client making the request might receive "connection refused" errors.

### `accept (Client-Socket-Desc, Server-Socket-Desc)`

The `accept` function is used by a connection-oriented server to wait for and accept a client connection request.

`Client-Socket-Desc` is the new socket descriptor that is created by the function to represent the client whose request the server has accepted and will be processing.

`Server-Socket-Desc` is the server socket descriptor on which the server listens and accepts client requests for processing.

**Deliverables**

Compile a single word document by filling in the solution part and submit this Word file on Google Classroom/MS Teams. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on Google Classroom/MS Teams. In case of any problems with submissions on Google Classroom/MS Teams, submit your Lab assignments by emailing them to: lab.engrcs@nbc.nust.edu.pk