LAB #11: TCP/IP Implementation using Python Socket Programming

# Objectives of Lab

- To understand the implementation of TCP/IP using Python socket programming.  
- To establish a basic client-server communication system.  
- To explore the functionality of Python’s socket library.

# Introduction to Python Programming

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is widely used in web development, data analysis, artificial intelligence, scientific computing, and network programming. Due to its vast standard library and community support, Python is ideal for rapid application development.

# Introduction to Python Socket Library and Its Various Functions

Python’s socket module provides a standard way of networking in Python and is used for implementing clients and servers. It supports both TCP and UDP protocols.  
  
Key functions used in this lab:  
- socket.socket() – Creates a new socket object.  
- bind() – Associates the socket with a specific IP and port.  
- listen() – Enables a server to accept connections.  
- accept() – Accepts a connection request from a client.  
- connect() – Connects a client to a server.  
- send() / sendall() – Sends data from the client to the server.  
- recv() – Receives data from the connection.  
- close() – Closes the socket.

# Client-Server Communication Using Socket Library

## Server Code (TCP Server)

import socket  
  
# Create socket object  
server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
server\_socket.bind(('localhost', 12345))  
server\_socket.listen(1)  
  
print("Server is waiting for client connection...")  
  
# Accept connection  
client\_socket, address = server\_socket.accept()  
print(f"Connected to {address}")  
  
# Receive message  
message = client\_socket.recv(1024).decode()  
print(f"Received from client: {message}")  
  
# Send response  
client\_socket.send("Hello Client, Message received!".encode())  
  
# Close sockets  
client\_socket.close()  
server\_socket.close()

## Client Code (TCP Client)

import socket  
  
# Create socket object  
client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
client\_socket.connect(('localhost', 12345))  
  
# Send message  
client\_socket.send("Hello Server!".encode())  
  
# Receive response  
response = client\_socket.recv(1024).decode()  
print(f"Received from server: {response}")  
  
# Close socket  
client\_socket.close()

## Sample Output

Server Output:  
Server is waiting for client connection...  
Connected to ('127.0.0.1', <port>)  
Received from client: Hello Server!

Client Output:  
Received from server: Hello Client, Message received!

# Flowchart of Client-Server Communication Using Python Socket Library

CLIENT SERVER  
 | |  
 | socket() | <- Create socket  
 | connect() | <- bind(), listen()  
 |-----------------------------> | <- accept()  
 | send() |  
 |-----------------------------> |  
 | | <- recv()  
 | | <- send()  
 | recv() |  
 |<----------------------------- |  
 | close() | <- close()  
 | |

# Conclusion

This lab successfully demonstrated the fundamentals of TCP/IP communication using Python's socket programming. We implemented a basic client-server model where a client sends a message to a server and receives a confirmation in response. This lays the foundation for understanding more advanced network communication systems.