TCP/IP Implementation using Python Socket Programming
LAB # 11

Spring 2025

Submitted by: **Mohsin Sajjad**
Registration No: **22pwsce2149**

Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: _____

Submitted to:
Dr. Yasir Saleem Afridi
Month Day, Year (30 05, 2025)

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

CSE 303L: Data Communication and Computer Networks

Credit Hours: 1

Demonstration of Concepts	**Poor (Does not meet expectation (1))** The student failed to demonstrate a clear understanding of the assignment concepts	**Fair (Meet Expectation (2-3))** The student demonstrated a clear understanding of some of the assignment concepts	**Good (Exceeds Expectation (4-5)** The student demonstrated a clear understanding of the assignment concepts	**Score** **30%**
Accuracy	The student mis-configured enough network settings that the lab computer couldn't function properly on the network	The student configured enough network settings that the lab computer partially functioned on the network	The student configured the network settings that the lab computer fully functioned on the network	**30%**
Following Directions	The student clearly failed to follow the verbal and written instructions to successfully complete the lab	The student failed to follow the some of the verbal and written instructions to successfully complete all requirements of the lab	The student followed the verbal and written instructions to successfully complete requirements of the lab	**20%**
Time Utilization	The student failed to complete even part of the lab in the allotted amount of time	The student failed to complete the entire lab in the allotted amount of time	The student completed the lab in its entirety in the al	**20%**

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.

CODE:

Server:

```python
import socket

# Create socket object
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind to localhost on port 12345
server_socket.bind(('localhost', 12345))

# Listen for connections (maximum 1 client in queue)
server_socket.listen(1)
print("Server is waiting for client connection...")

# Accept a client connection
client_socket, address = server_socket.accept()
print(f"Connected to {address}")

# Receive message from client
message = client_socket.recv(1024).decode()
print(f"Received from client: {message}")

# Send response to client
client_socket.send("Hello Client, Message received!".encode())

# Close sockets
client_socket.close()
server_socket.close()
```

CLIENT:

```python
import socket

# Create socket object
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to server at localhost on port 12345
client_socket.connect(('localhost', 12345))

# Send message to server
client_socket.send("Hello Server!".encode())

# Receive response from server
response = client_socket.recv(1024).decode()
print(f"Received from server: {response}")

# Close socket
client_socket.close()
```

Output:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

D:\Semester_Data\6th_semester\CCN Lab\My lab reports\lab11>python client.py
Received from server: Hello Client, Message received!

D:\Semester_Data\6th_semester\CCN Lab\My lab reports\lab11>
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

D:\Semester_Data\6th_semester\CCN Lab\My lab reports\lab11>python server.py
Server is waiting for client connection...
Connected to ('127.0.0.1', 55832)
Received from client: Hello Server!

D:\Semester_Data\6th_semester\CCN Lab\My lab reports\lab11>
```

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.