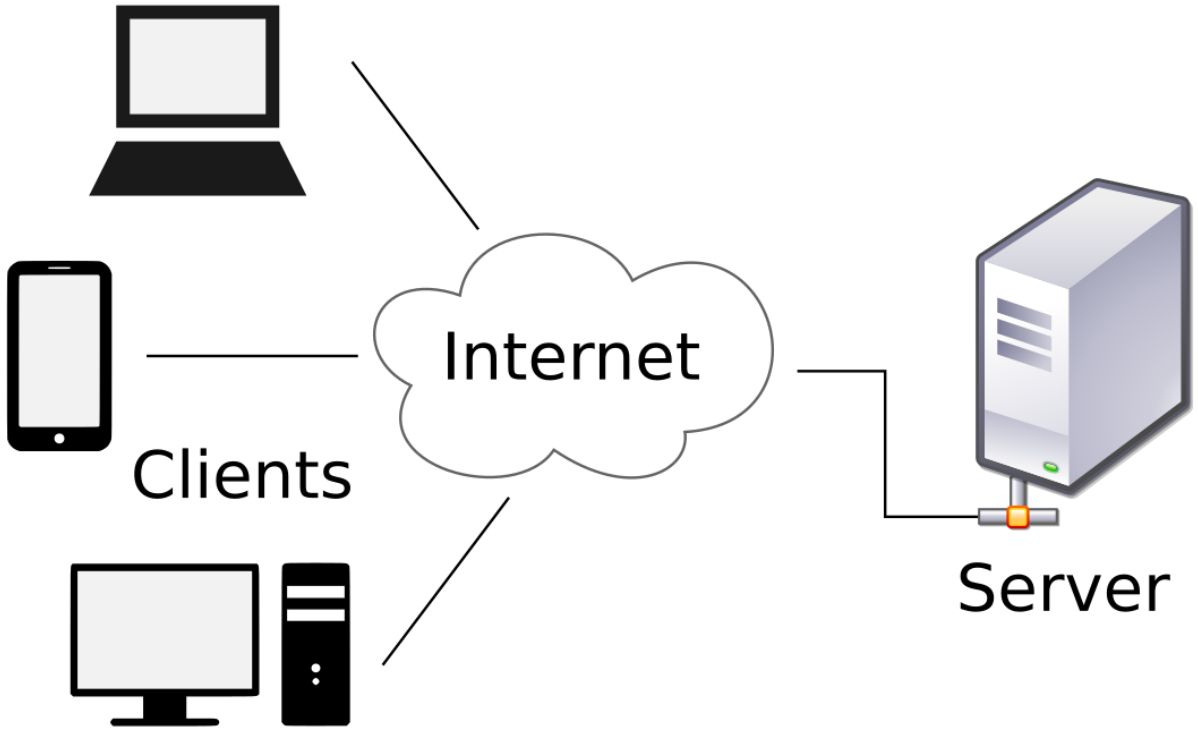
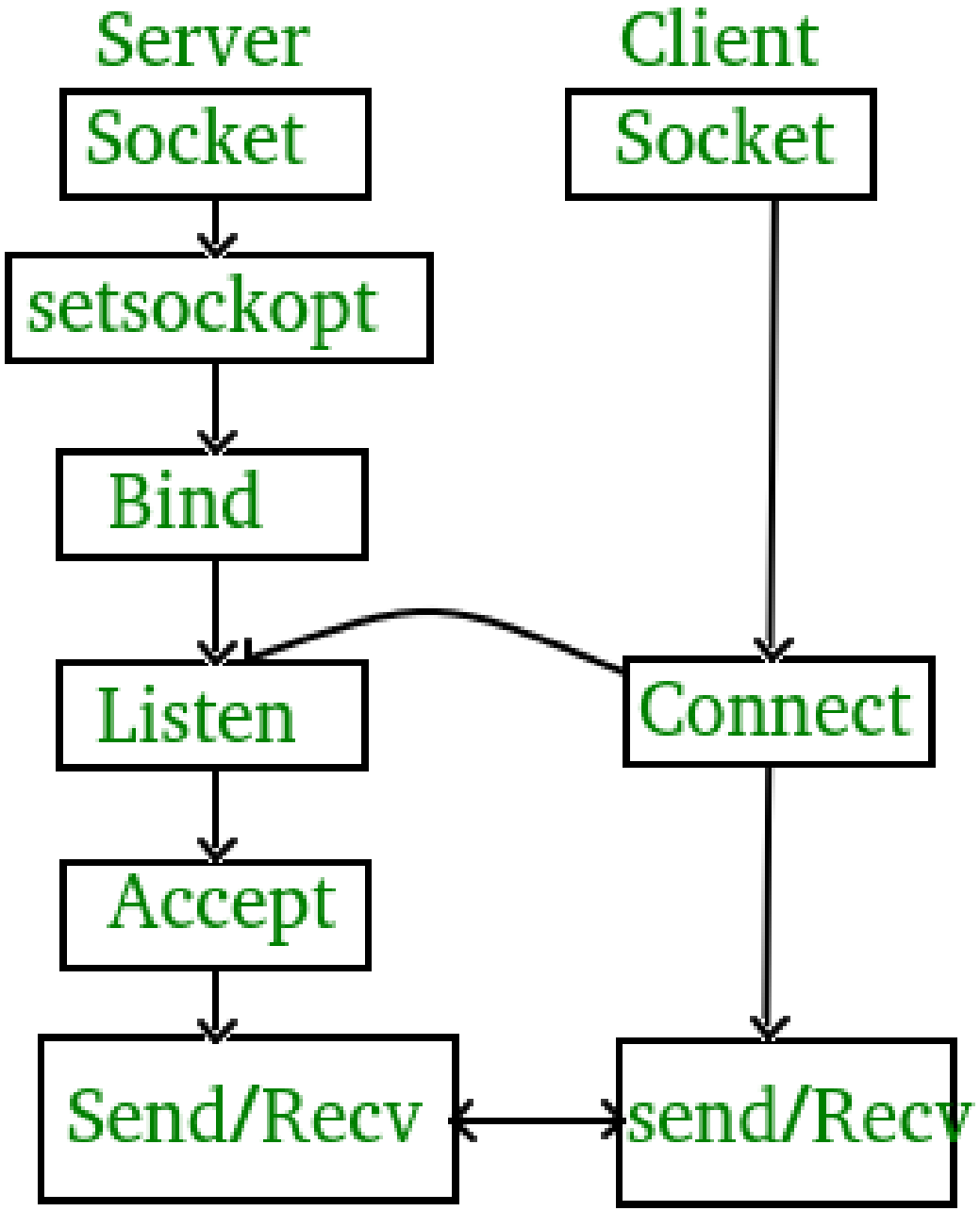
Lab 3: Create a TCP-Based Client-Server Chat Application


Internet
Clients
Server


Server
Client
Socket
Socket
setsockopt
Bind
Listen
Connect
Accept
Send/Recv
send/Recv

Creating the Server

The server will wait for a client to connect. Once connected, it will receive a message from the client, respond with a message, and continue this loop until the connection is closed.

Server Program

```
import socket
def start_server():
    # Create a socket object using IPv4 (AF_INET) and TCP (SOCK_STREAM)
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Bind the socket to an IP address and a port
    server_socket.bind(('0.0.0.0', 12345))  # '0.0.0.0' allows any network interface
    # Start listening for incoming connections (max 5 queued connections)
    server_socket.listen(5)
    print("Server is listening on port 12345...")
    client_socket, client_address = server_socket.accept() # Accept a client connection
    print(f"Connection established with {client_address}")
    while True:
        client_message = client_socket.recv(1024).decode('utf-8') # Receive data from the client
        if not client_message:
            break  # Break the loop if the client sends an empty message
        print(f"Client: {client_message}") # Send a response back to the client
                server_message = input("Server: ")
        client_socket.send(server_message.encode('utf-8'))
    client_socket.close() # Close the client connection
if __name__ == "__main__":
    start_server()
```

TCP Client Code:

The client will connect to the server, send messages, and wait for the server's response. It will continue this process until the connection is closed.

Client Program

```
import socket
def start_client():
    # Create a socket object using IPv4 (AF_INET) and TCP (SOCK_STREAM)
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Connect to the server (replace 'server_ip' with the actual server IP
address)
    client_socket.connect(('127.0.0.1', 12345))  # Use 'localhost' for testing on
the same machine
    while True:
        # Send a message to the server
        client_message = input("Client: ")
        client_socket.send(client_message.encode('utf-8'))
        # Receive a response from the server
        server_message = client_socket.recv(1024).decode('utf-8')
        if not server_message:
            break  # Break the loop if the server sends an empty message
        print(f"Server: {server_message}")
    # Close the connection
    client_socket.close()
if __name__ == "__main__":
    start_client()
```