2. Networking with C++

Server Requirements:

1. **Server Initialization**: The server must be able to initialize and start listening on a specified port.
2. **Port Configuration**: The server should allow configuration of the port number on which it will listen for incoming connections.
3. **Connection Handling**: The server should be ready to accept incoming client connections as soon as it starts listening on the specified port.
4. **Connection State**: The server must maintain an active listening state until it is explicitly stopped or encounters an error.

Client Requirements:

1. **Client Initialization**: The client must be able to start and initiate a connection request to a server.
2. **Server Identification**: The client must be able to connect to a server identified by a specific IP address and port number.
3. **Connection Request**: The client should be able to establish a connection with the server if the server is listening on the specified IP and port.
4. **Connection State**: The client must maintain the connection with the server as long as it is active, and it should handle connection closure gracefully.

Test Cases for Server:

1. **Server Initialization Test**:
   - **Objective**: Verify that the server starts correctly and begins listening on the specified port.
   - **Steps**:
     1. Configure the server with a specific port number.
     2. Start the server.
     3. Check if the server is listening on the specified port using `netstat` or a `ss`.

- **Expected Result**: The server should be listed as listening on the configured port.

2. **Port Configuration Test**:
   - **Objective**: Ensure that the server correctly listens on different specified ports.
   - **Steps**:
     1. Start the server with port A.
     2. Verify the server is listening on port A.
     3. Stop the server and restart it with port B.
     4. Verify the server is listening on port B.
   - **Expected Result**: The server should successfully listen on both port A and port B as specified.
3. **Connection Acceptance Test**:
   - **Objective**: Confirm that the server accepts incoming connections.
   - **Steps**:
     1. Start the server.
     2. Use a client to attempt a connection to the server.
     3. Check the server logs or use a monitoring tool to verify the connection was accepted.
   - **Expected Result**: The server should accept the incoming connection from the client.

---

2. Client Requirements:

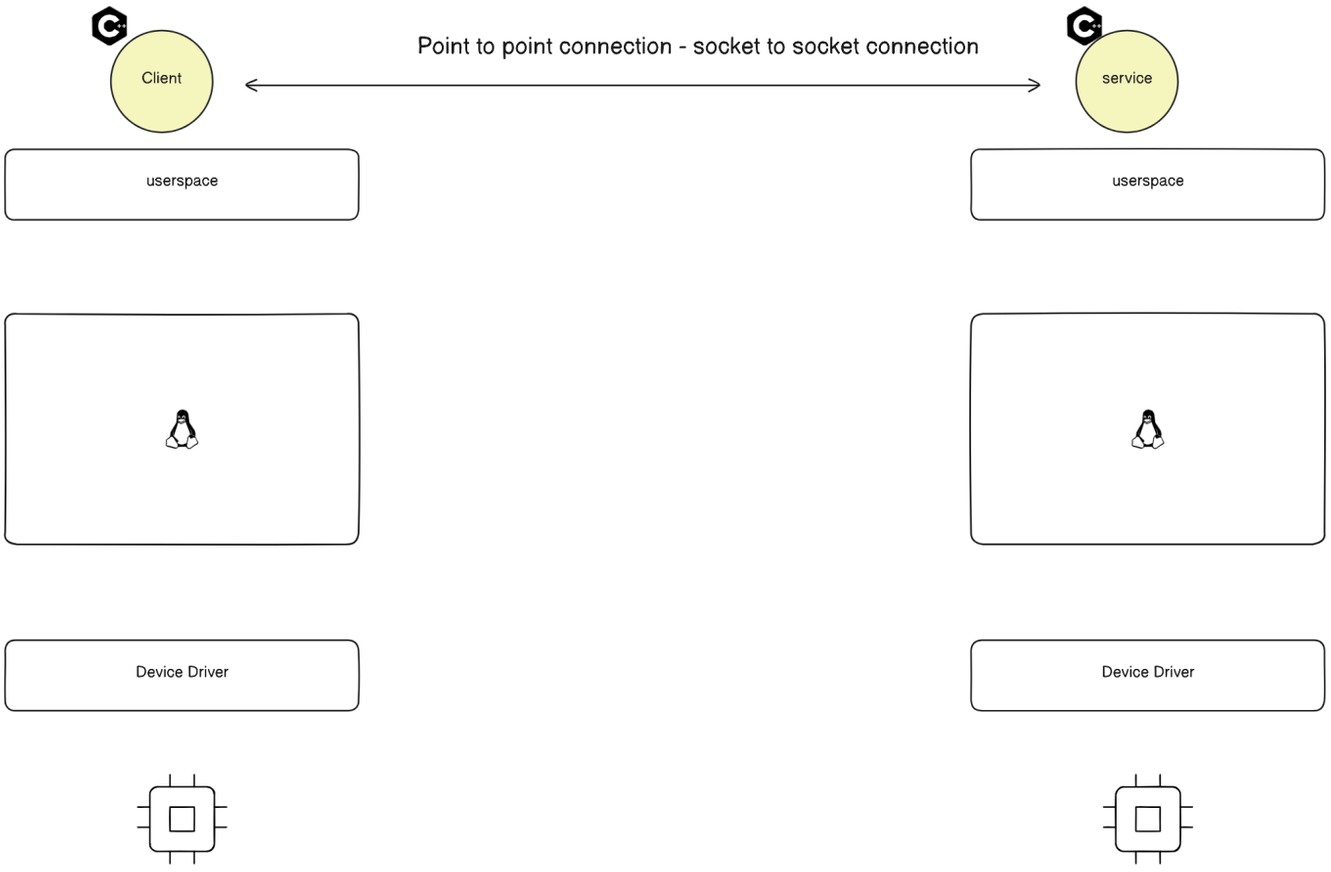
1. **Client Initialization**: The client must be able to start and initiate a connection request to a server.
2. **Server Identification**: The client must be able to connect to a server identified by a specific IP address and port number.
3. **Connection Request**: The client should be able to establish a connection with the server if the server is listening on the specified IP and port.
4. **Connection State**: The client must maintain the connection with the server as long as it is active, and it should handle connection closure gracefully.

Test Cases for Client:

1. **Client Initialization Test**:
   - **Objective**: Ensure the client starts correctly and attempts to connect to the server.
   - **Steps**:
     1. Start the client with a valid server IP and port number.
     2. Check if the client sends a connection request to the server.

- **Expected Result**: The client should successfully initiate a connection request to the server.

2. **Server Identification Test**:
   - **Objective**: Verify that the client can connect to different servers based on IP and port number.
   - **Steps**:
     1. Start the client with server A's IP and port number.
     2. Verify connection to server A.
     3. Reconfigure the client to connect to server B with a different IP and port.
     4. Verify connection to server B.
   - **Expected Result**: The client should successfully connect to both servers as configured.
3. **Connection Failure Handling Test**:
   - **Objective**: Ensure the client correctly handles cases where the server is unavailable or refuses the connection.
   - **Steps**:
     1. Start the client with a server IP and port where no server is running.
     2. Observe how the client reacts to the failed connection attempt.
   - **Expected Result**: The client should gracefully handle the failed connection attempt, possibly by retrying or showing an error message.


Point to point connection - socket to socket connection
Client
service
userspace
userspace
Device Driver
Device Driver