Project Explanation

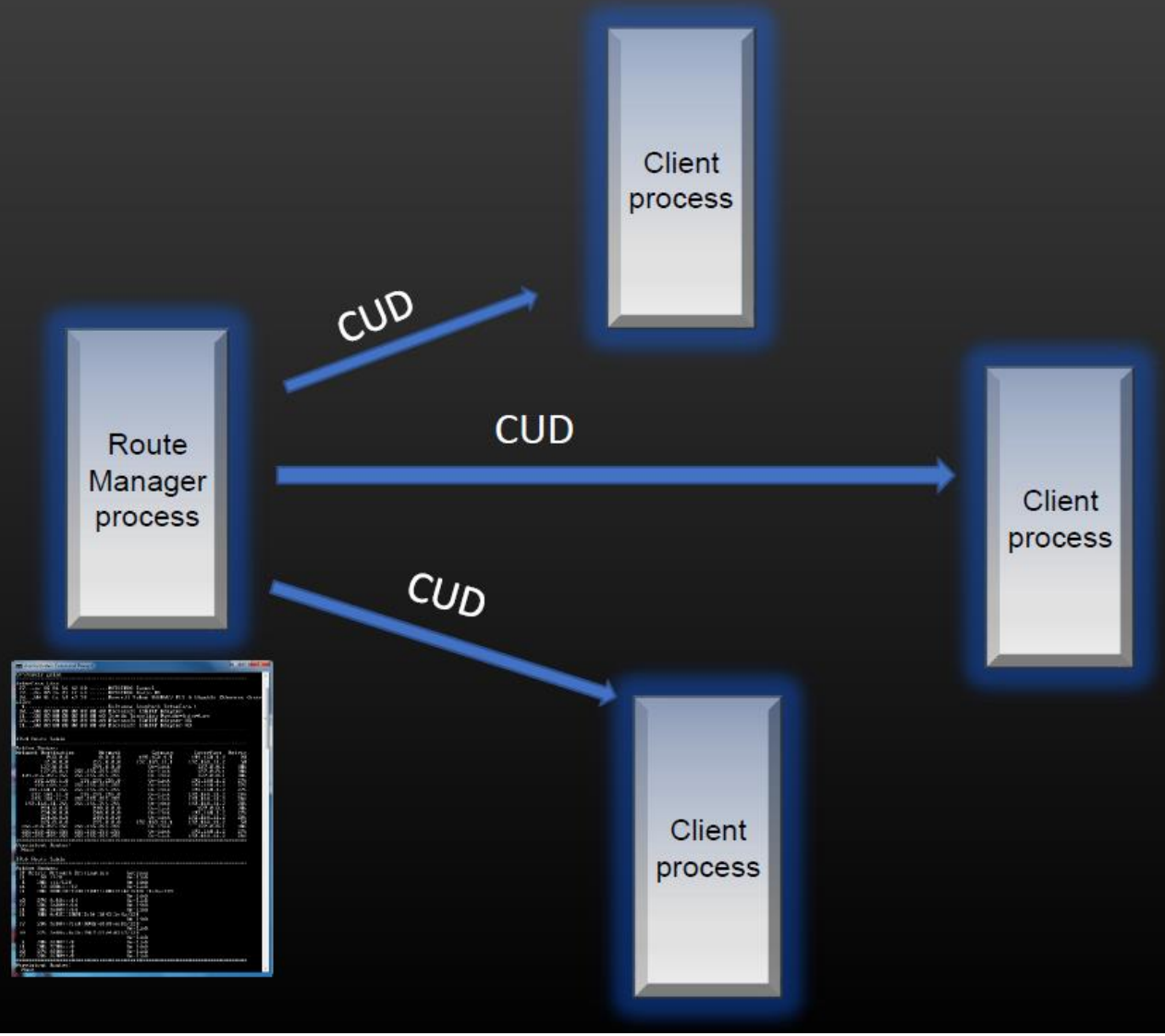
Unix domain socket Project -> Data Synchronization

Let us Create a routing table Manager (RTM) process

- A RTM is in-charge of a Layer 3 routing table
- Its responsibility is to maintain the L3 routing table, and sends notification of any change in the routing table contents to connected clients
- State of routing table needs to be synchronized across all clients at any point of time

Unix domain socket Project -> Data Synchronization

A big picture



Route Manager Server process sends CUD (Create, Update And Delete) notification to all connected Client process)

Let us discuss the project step by step

Unix domain socket Project -> Data Synchronization

- Route Manager process maintains a routing table
- This table is managed by Admin
- Sample of table is shown below.
- You are free to choose the Data structure to represent and manage this table

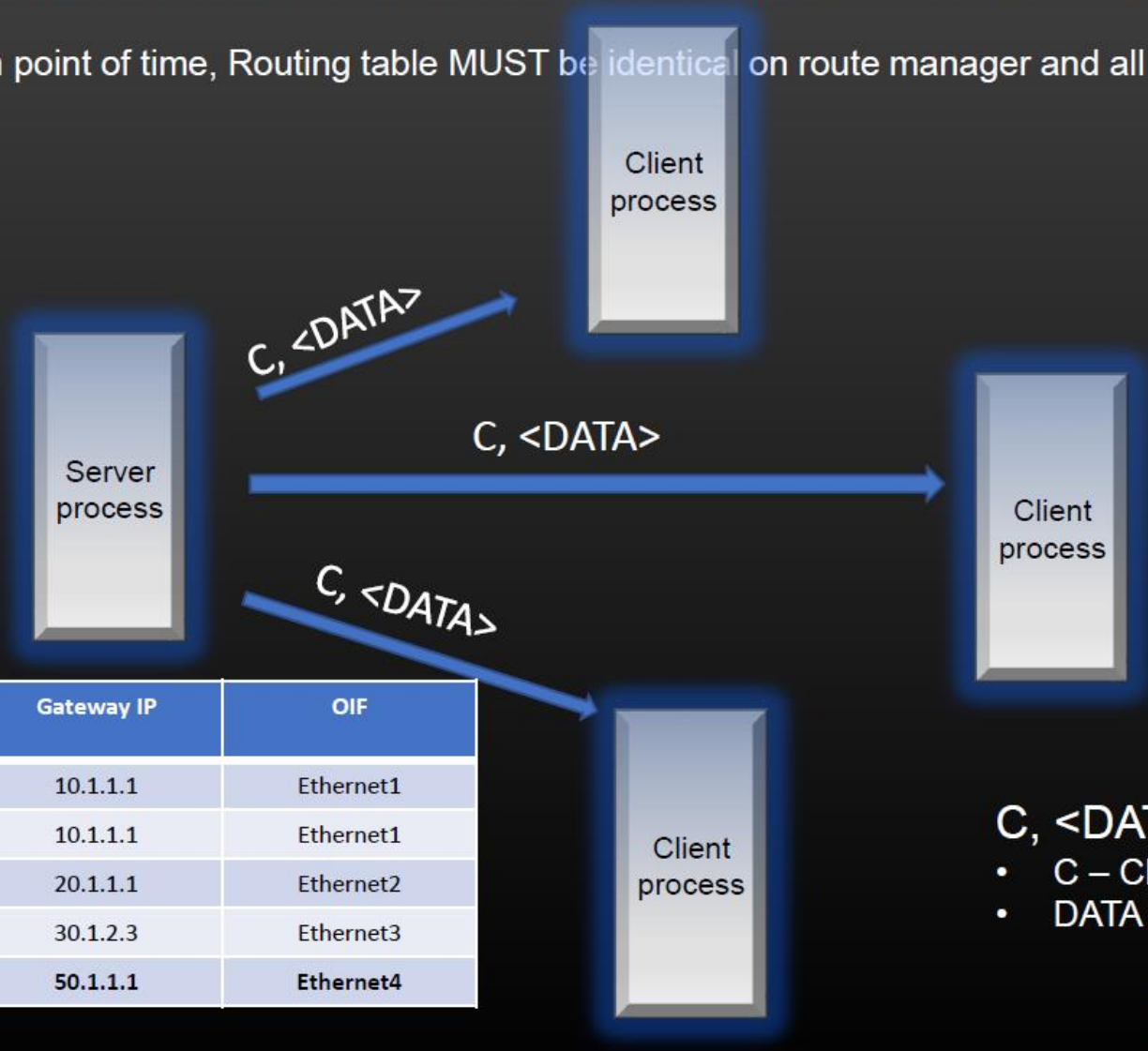
Note :
Mask – [0,32]

Destination Subnet (Key)	Gateway IP	OIF
122.1.1.1/32	10.1.1.1	Ethernet1
130.1.1.0/24	10.1.1.1	Ethernet1
126.30.34.0/24	20.1.1.1	Ethernet2
220.1.0.0/16	30.1.2.3	Ethernet3

- Operation supported on the table :
  - Insert - <Destination/mask> <Gateway IP> <OIF>
  - Update - <Destination/mask> <new Gateway IP> <new OIF>
  - Delete - <Destination/mask>

Unix domain socket Project -> Data Synchronization

- Whenever the User perform any CUD operation on the routing table, Route Manager Server process sync that particular operation to all connected clients
- When new client connects to the server, Server sends the entire table state to this newly connected client
- At any given point of time, Routing table MUST be identical on route manager and all connected clients



Destination Subnet (Key)	Gateway IP	OIF
122.1.1.1/32	10.1.1.1	Ethernet1
130.1.1.0/24	10.1.1.1	Ethernet1
126.30.34.0/24	20.1.1.1	Ethernet2
220.1.0.0/16	30.1.2.3	Ethernet3
100.100.100.0/24	**50.1.1.1**	**Ethernet4**

Destination Subnet (Key)	Gateway IP	OIF
122.1.1.1/32	10.1.1.1	Ethernet1
130.1.1.0/24	10.1.1.1	Ethernet1
126.30.34.0/24	20.1.1.1	Ethernet2
220.1.0.0/16	30.1.2.3	Ethernet3
100.100.100.0/24	**50.1.1.1**	**Ethernet4**

C, <DATA>

- C – CREATE (operation code)
- DATA - <Dest/mask> <Gw IP> <OIF>

Provide a menu-driven approach to show routing table Contents on Server and Client processes

Unix domain socket Project -> Data Synchronization

Data Structures Suggestions :

The operation code could be Enums :

```
typedef enum{

        CREATE,
        UPDATE,
        DELETE
} OPCPDE;
```

The structure which contains the data to be synchronized can be :

```
typedef struct _msg_body {

        char destination[16];
        char mask;
        char gateway_ip[16];
        char oif[32];
} msg_body_t;
```

The final msg that needs to be synced from routing table manager process to all clients
Should have Operation code and Body

```
typedef  struct _sync_msg {

        OPCODE op_code;
        msg_body_t msg_body;
} sync_msg_t;
```

Socket programming APIs

1. socket() – Used to create a connection/master socket on server side. Used to create data socket on Client side
2. select() – Used for monitoring multiple file descriptors. Can be used on both client and server sides. Blocking system call. Blocks until new connection request Or data arrived on FDs present in fd_set.
3. accept() – Used on server side. Used to accept the client new connection request and establish connection with client
4. bind() – Used on server side. Used by the Server application process to inform operating system the criteria of packets of interests
5. listen() – Used on server side. Used by the Server application process to inform operating system the length of Queue of incoming connection request/data request from clients
6. read() – Used on the server and client side. Used by the Server/client process to read the data arrived on communication file descriptors. This call is blocking call by default. Process block if data hasn't arrived yet.
7. write() – Used to send data to client/server. Used on Server and client sides.
8. close() – Used to close the connection. Used by both – clients and Server

Thank You