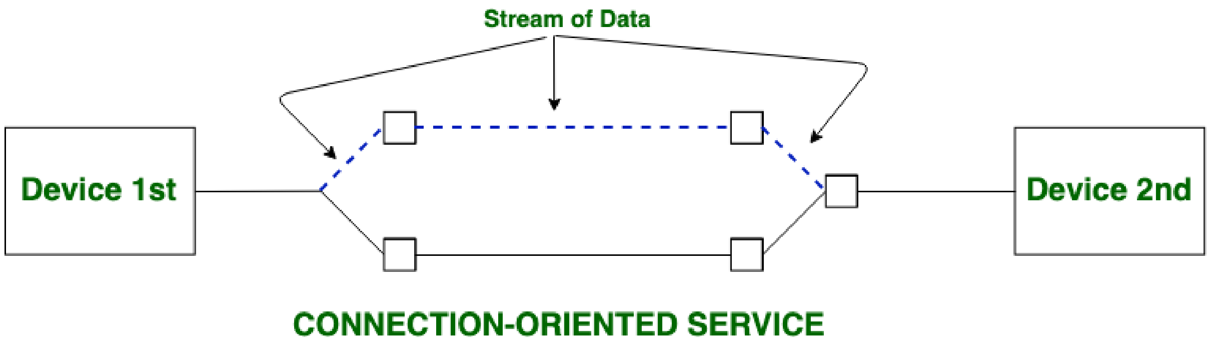
# DA LAB 3

## SAHIL BONDRE: U18CO021

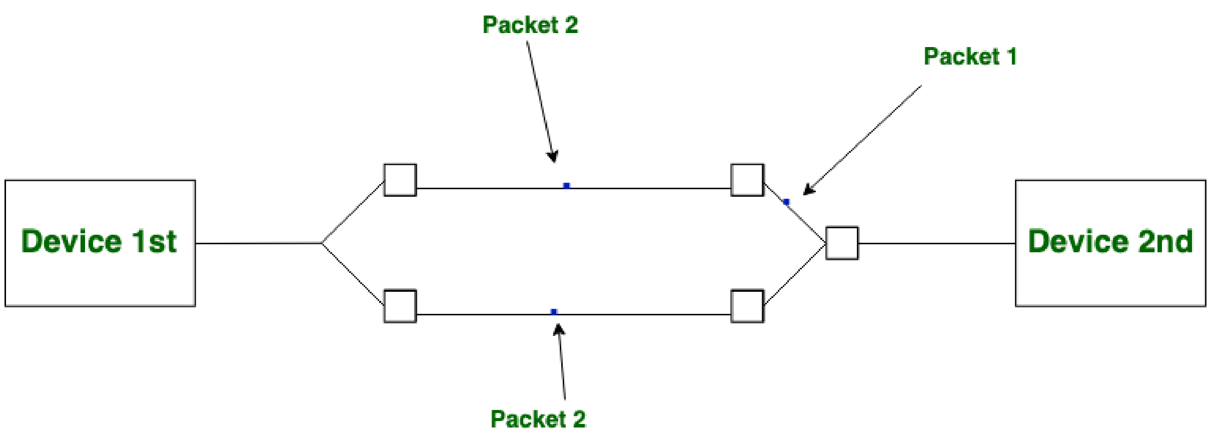
**Q1. To Study the basics of connection-oriented protocol and connectionless protocol. Also, explain the difference between them.**

Both Connection-oriented service and Connection-less service are used for the connection establishment between two or more than two devices. These types of services are offered by the network layer.

**Connection-oriented service** is related to the telephone system. It includes connection establishment and connection termination. In a connection-oriented service, the Handshake method is used to establish the connection between sender and receiver.



**Connection-less** service is related to the postal system. It does not include any connection establishment and connection termination. Connection-less Service does not give the guaranteed reliability. In this, Packets do not follow the same path to reach their destination.



CONNECTIONlESS SERVICE

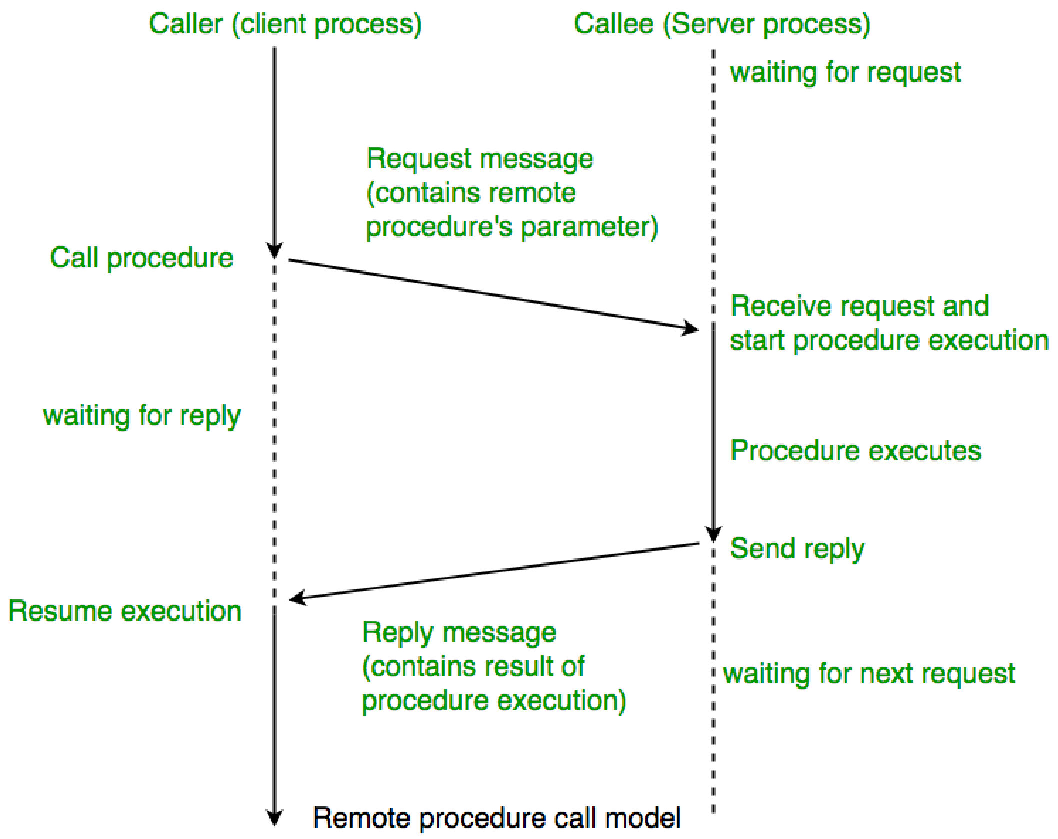
| **Connection-Oriented Service** | **Connection-Less Service** |
|---|---|
| Connection-oriented service is related to the telephone system. | Connection-less service is related to the postal system. |
| Connection-oriented service is preferred by long and steady communication. | Connection-less Service is preferred by bursty communication. |
| A connection-oriented service is feasible. | Connection-less Service is not feasible. |
| In connection-oriented Service, Congestion is not possible. | In connection-less Service, Congestion is possible. |
| In connection-oriented Service, Packets follow the same route. | In connection-less Service, Packets do not follow the same route. |

**Q2. To Study RPC and the protocols.**

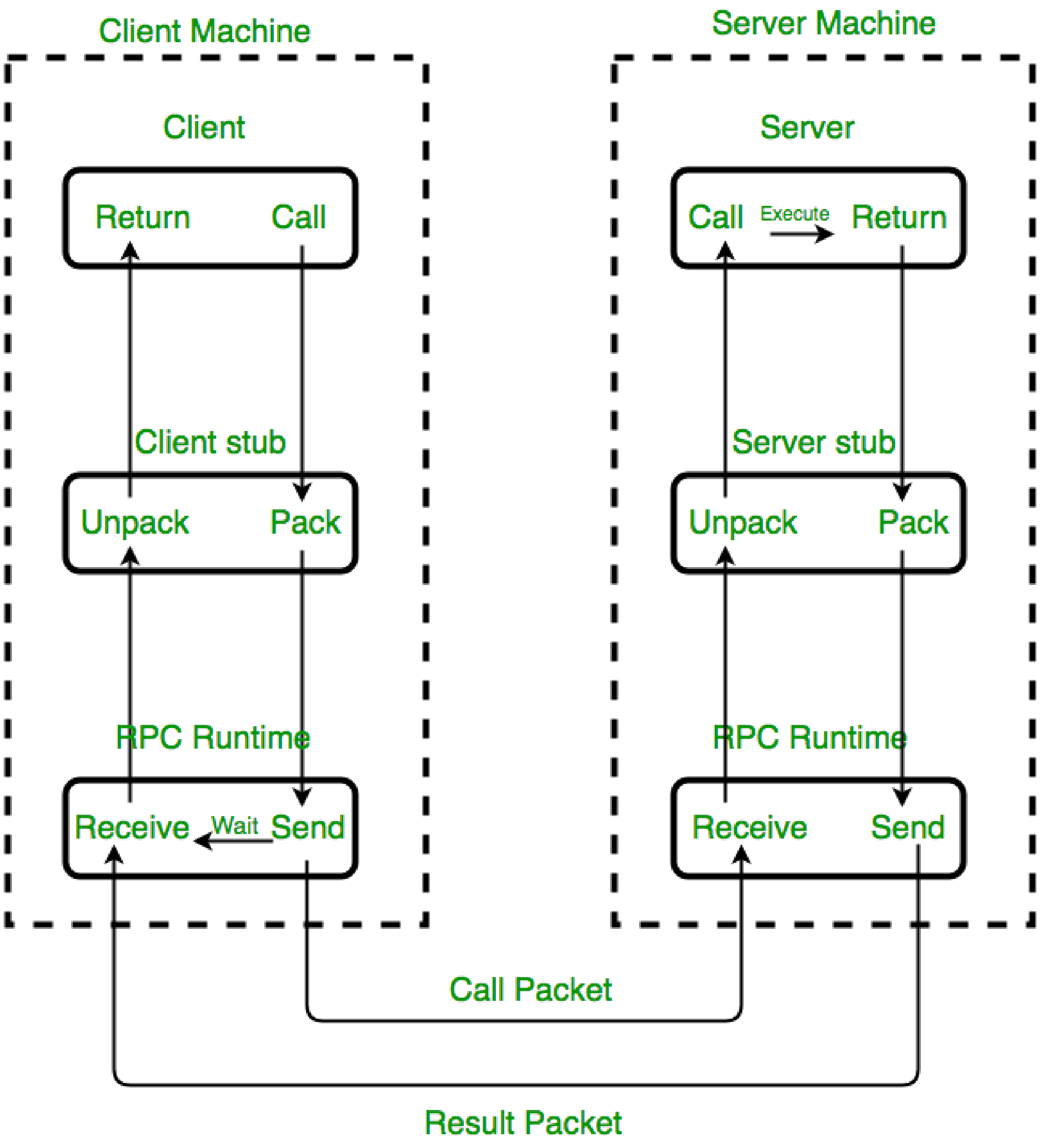
RPC is a request-response protocol. An RPC is initiated by the client, which sends a request message to a known remote server to execute a specified procedure with supplied parameters. The remote server sends a response to the client, and the application continues its process. While the server is processing the call, the client is blocked (it waits until the server has finished processing before resuming execution), unless the client sends an asynchronous request to the server.

**The sequence of events**

1. The client calls the client stub. The call is a local procedure call, with parameters pushed onto the stack in the normal way.
2. The client stub packs the parameters into a message and makes a system call to send the message. Packing the parameters is called marshalling.
3. The client's local operating system sends the message from the client machine to the server machine.
4. The local operating system on the server machine passes the incoming packets to the server stub.
5. The server stub unpacks the parameters from the message. Unpacking the parameters is called unmarshalling.
6. Finally, the server stub calls the server procedure. The reply traces the same steps in the reverse direction.



Remote procedure call model



Implementation of RPC mechanism

**Advantages of RPC:**

1. RPC provides ABSTRACTION i.e message-passing nature of network communication is hidden from the user.
2. RPC often omits many of the protocol layers to improve performance. Even a small performance improvement is important because a program may invoke RPCs often.

3. RPC enables the usage of the applications in the distributed environment, not only in the local environment.
4. With RPC code re-writing / re-developing effort is minimized.
5. Process-oriented and thread oriented models supported by RPC.

**RPC Implementation in different languages:**

1. Java's Java Remote Method Invocation (Java RMI) API provides similar functionality to standard Unix RPC methods.
2. Go provides package “rpc” for implementing RPC, with support for asynchronous calls.
3. RPyC implements RPC mechanisms in Python, with support for asynchronous calls.
4. Distributed Ruby (DRb) allows Ruby programs to communicate with each other on the same machine or over a network. DRb uses remote method invocation (RMI) to pass commands and data between processes.