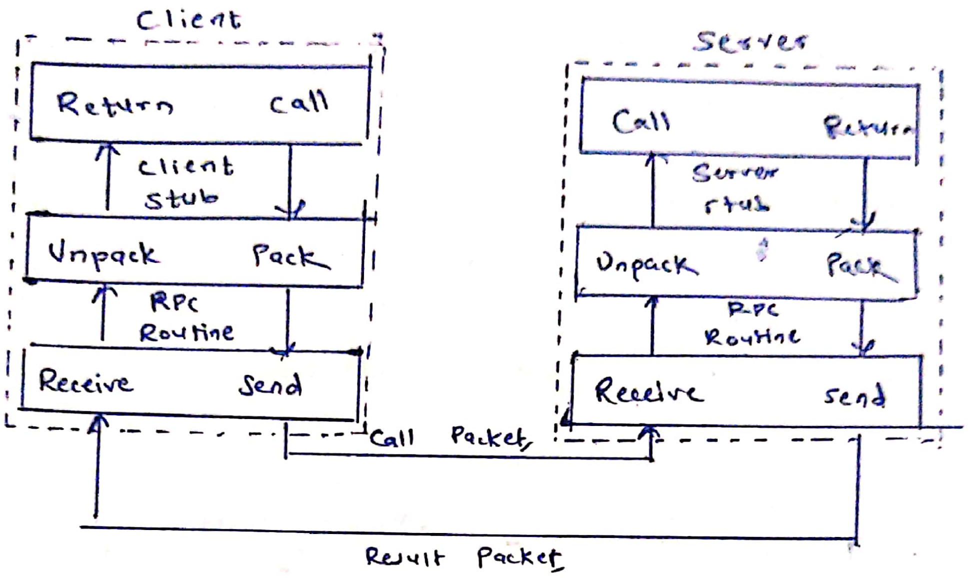
Remote Procedure Call
RPC

- It is a protocol that one program can use to request a service from a program located in another computer on a network without having to understand the network detail.



RPC Steps:

1. Client procedure calls client stub in normal way
2. Client stub builds message, calls local OS.
3. Client's OS sends message to remote OS.
4. Remote OS gives message to server stub.
5. Server stub unpacks parameters, calls server.
6. Server does work, returns result to the stub.
7. Server stub packs it in message, calls local os.
8. Server's OS sends message to client's O.S.
9. Client's OS gives message to client stub.
10. stub unpacks result, returns to client.

---

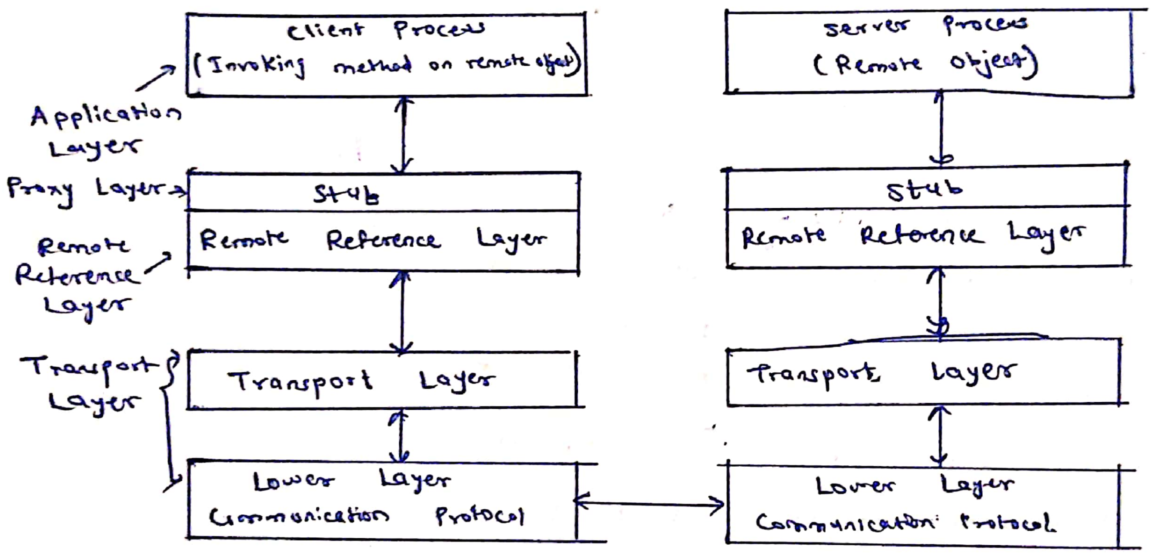
1. The Client
   - → Initiates RPC
   - → Invokes client stub.
2. The Client stub
3. The RPC Routine
   - → Handles transmission of message between client and server.
4. The Server stub.
   - → makes a perfectly normal call to invoke the appropriate procedure in server.
5. The server
   - → Executes the appropriate procedure

Remote Method Invocation

RMI

- It is a set of protocols being developed by Sun's javasoft division that enables java objects to communicate remotely with other java objects.
- Characteristics :
  1. RMI is a simple protocol as compared to CORBA & DCOM.
  2. It works only with java objects
  3. Based ~~on~~ RPC and developed in 1980s
- RMI Goals:
  1. Seamless object remote invocation.
  2. Call backs from server to client.
  3. Distributed Garbage Collection.

  Note: In RMI, all objects must be written in Java.



1. Application Layer
   - Responsible for running client and server application.
   - Client application invokes method defined by server application.
2. Proxy layer
   - Responsible for creating client stub at client side. by packing request sent ~~by~~ message sent by the client process.
   - Also responsible for creating skeleton by ~~-~~ packing response message sent by server.
3. Remote Reference Layer
   - Checks order/semantics and remote reference used by client process using remote reference protocol.
   - RRL transmits message and data to RMI Transport Layer.
4. Transport Layer
   - Responsible for establishing and maintaining stream oriented communication between client and server.
   - Responsible for managing send request & Request Reply messages between client and server.

Group Communication

Modes of Communication

① Unicast

→ 1 ↔ 1

→ Point to point.

② Anycast

→ 1 ~~to~~ Nearest p.. of several identical nodes.

③ Netcast

→ 1 ~~to~~ many, 1 at a time.

④ multicast

→ 1 to many

⑤ Broadcast

→ 1 to all.

Types of group communication possible.

① One to many (single sender and multiple receiver)

→ Receiver processes messages from groups are of two types

ⓐ closed group v/s Open Group

ⓑ Peer group v/s Hierarchical group

② Many to one (multiple senders and one receiver)

→ many senders sends the message to selective receiver.

→ Receiver can be selective or non-selective.

ⓐ selective
- specifies a unique sender, the message exchange takes place only if the sender sends the message.

ⓑ Non selective
- specifies a set of senders and if ~~any~~ onesender in the set sends the message to this receiver, message exchange will take place

③ Many to many (multiple senders and multiple receivers)
- Many senders sends message to many receivers.
- Semantics ordered message delivery are

ⓐ Absolute ordering

ⓑ Consistent ordering

ⓒ Casual ordering

Stream-oriented Communication

- It is a form of communication in which timing plays a crucial role.
- Transmission Modes:

① synchronous
- Specifies maximum end to end delay
- Variance between two packets is ok.

② Asynchronous
- End to End delay can be maximum

③ Isochronous
- Specifies maximum end to end delay and variance too.

- Stream-oriented communication uses token bucket algorithm as it overcomes drawbacks of leaky bucket algorithm.

message-oriented communication	stream-oriented communication
① It is used by UDP (User Datagram Protocol)	① It is used by TCP (Transmission Control Protocol).
② Data is sent by application in discrete packages called message	② Data is sent by with no particular structure.
③ Communication is connectionless, data is sent without any setup.	③ Communication is oriented, connection established before communication.
④ It is unreliable as no data acknowledgement	④ It is reliable as data acknowledged
⑤ Low overhead	⑤ High overhead
⑥ Retransmission is not performed	⑥ Lost data is reframe automatically.
⑦ Transmission speed is very high	⑦ Transmission speed is low
⑧ Suitable for applications like audio, video where speed is critical than loss of message.	⑧ Suitable for applications like e-mail where data must be persistent through delivered late.