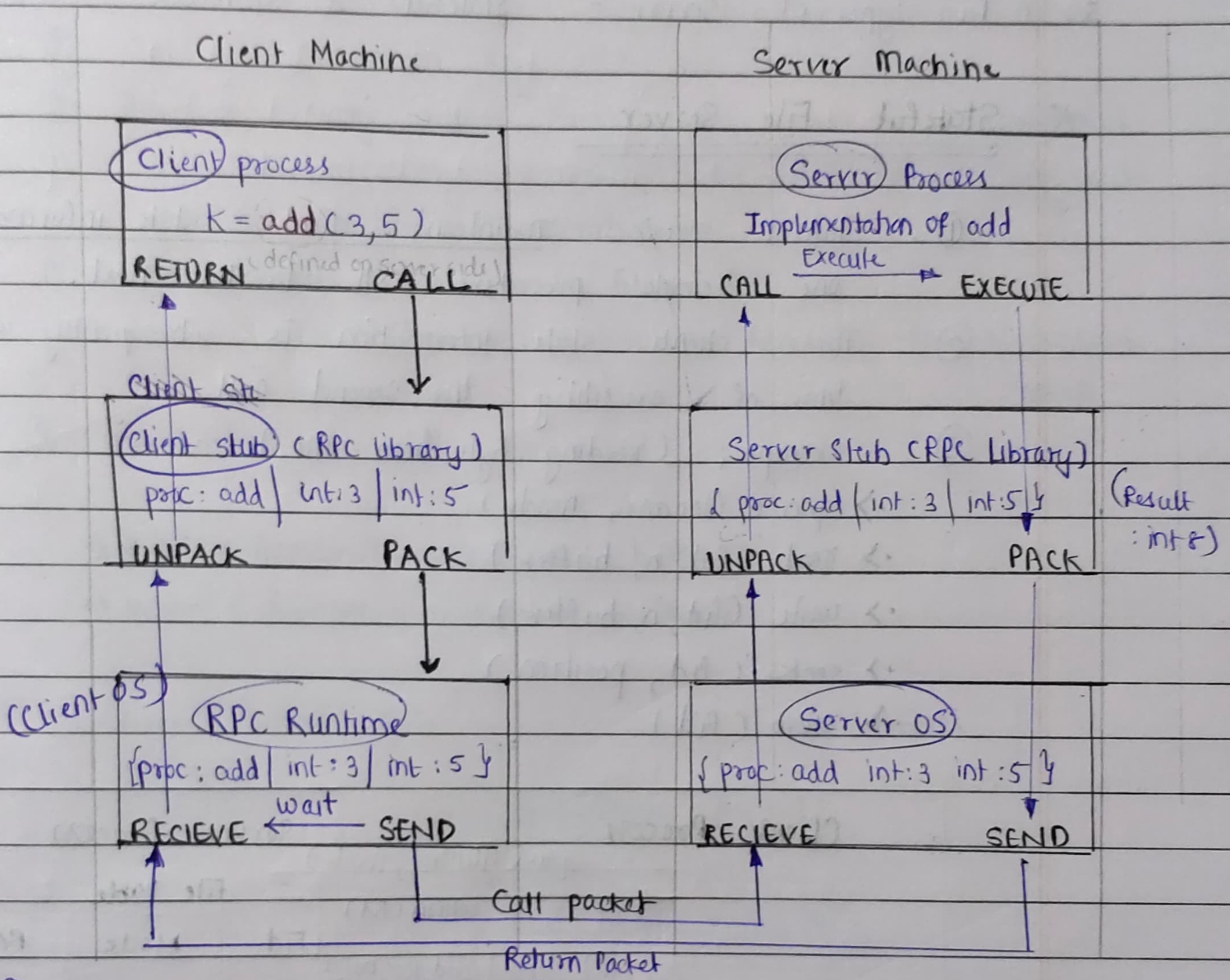
Tutorial - 4

U19CS012

1.> Explain the difference between remote procedure calls and local calls.

	Remote Procedure Calls		Local Procedure Calls
①	Address space is disjoint from the calling program	①	Address space is same
②	called (remote) procedure cannot have access to any variables or data values in calling program's environment.	③	Have access
③	Remote calls can fail often and it occurs without the knowledge of user. crash (vulnerable to failure) : • processor • communication problem of network	③	Local calls generally does not fail and are easily handled.
④	Absense of shared memory	④	It has shared memory
⑤	meaningless making call by reference, using address in arguments and pointers.	⑤	Doesnot make meaningless call's
⑥	consumes more time (100- 1000 times more) than local procedure calls. (Why? due to involvement of communication network).	©	Faster than RPC

2) What is the sequence of events during remote procedure call?



1. Client procedure calls clients stub.
2. client stub builds message, and call 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 {unmarshalling}.
6. Server does work, return results to stub.
7. Server stub pack it in message, calls local OS.
8. Server's OS's sends message to the client's OS.
9. Client's OS gives message to client stub.
10. Stub unpacks result, returns to client.

3.> Explain server management.

3.> ① Two types of server - Stateful & Stateless

* Stateful File Server

① Stateful server maintain's client's state information from one remote procedure call to the next.

② These clients state information is subsequently used at the time of executing the second call.

③ example; (reading byte from file)

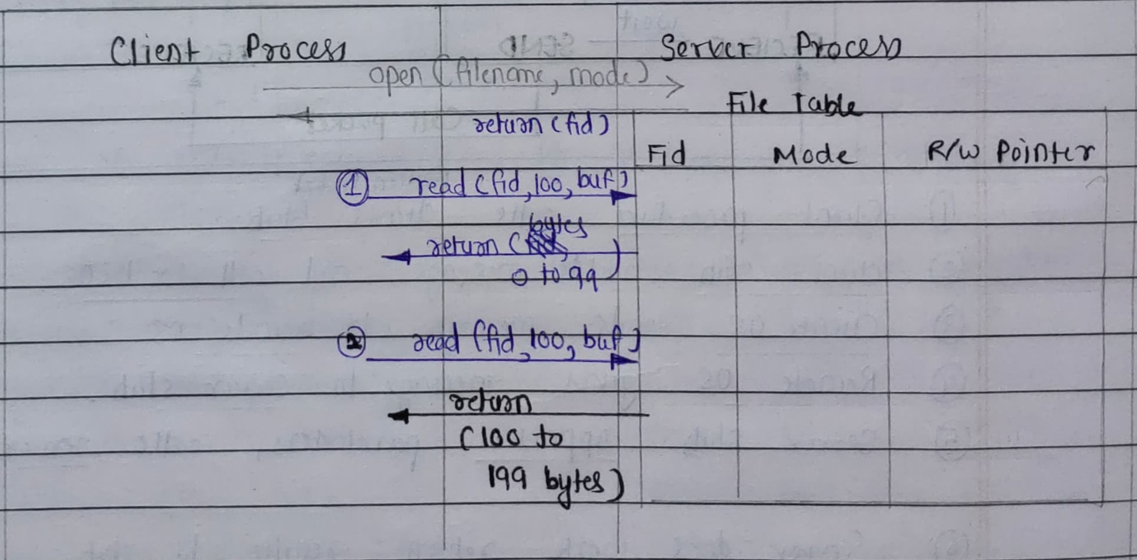
·> open (filename, mode)

·> read (fid, n, buffer)

·> write (fid, n, buffer)

·> seek (fid, position)

·> close (fid)



After opening a file, if a client makes two subsequent Read (fid, 100, buf) requests, for the first request the first 100 bytes (bytes 0 to 99) will be read and for second request the next 100 bytes (byte 100 to 199) will be read.

U19CS012

* Stateless File server

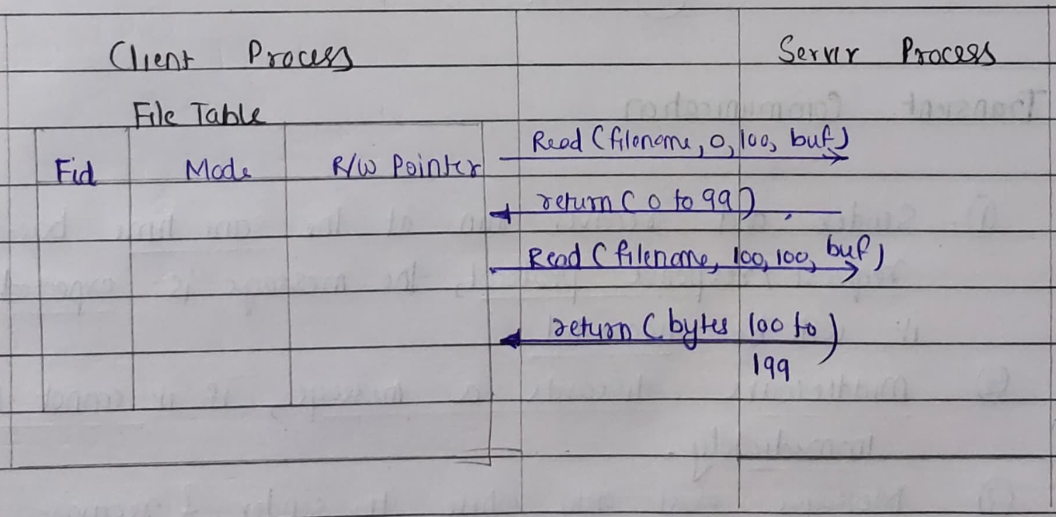
① Does not maintain any client-side information

② Therefore, every request from client must be accompanied with all necessary parameters to successfully carry out the ~~desired~~ desired operation.

③ Each request identifies the file & the position in the file for the read/write access.

④ Operation's in Stateless file server

- Read (filename, position, n, buffer)
- Write (filename, position, n, buffer)



Client Process

Server Process

File Table

Fid	Mode	R/W Pointer

Read (filename, 0, 100, buf)

return (0 to 99)

Read (filename, 100, 100, buf)

return (bytes 100 to 199)

⑤ ~~After~~ File server → does not keep track of any file state info. resulting from previous operation

for similar effect → [client needs to (remember) ↳ previous location]

read (filename, 0, 100, buffer)
read (filename, 100, 100, buffer)

* (No recory if crash occurs)

4.> Write a brief note on communication types in Distributed System.

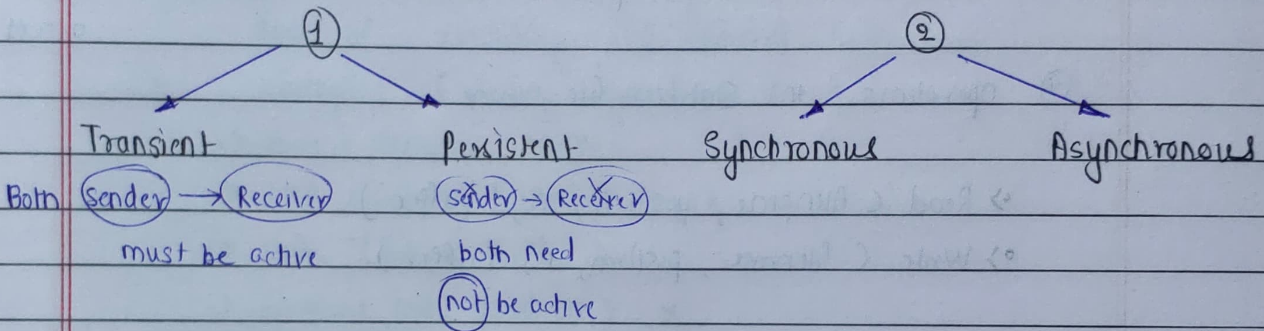
4.> ① In Distributed system, processes run on different machines.

② Processes can only exchange info. through message passing

- Harder to program than shared memory communication

③ Communication in DS → Low-Level message passing

Types of Communication



(1A) Transient Communication

① Sender and receiver run at the same time based on request/response protocol, the message is expected otherwise it will be discarded.

② Middleware discards a message, if it cannot be delivered immediately.

③ Messages exist only while the sender & receiver are running.

④ Communication errors / inactive receiver
↳ cause the message to be discarded.

⑤ Transport-Level communication is Transient.

(1B) Persistent Communication (opposite of transient)

① Messages are stored by Middleware until receiver can accept it

② Receiving application need not be executing when the message is submitted.

[Example : Email]

(2A) Synchronous Communication

① Sender blocks until its request is known to be accepted.
② Sender and Receiver must be active at same time.
③ Sender execution is continued only if the previous message is received and processed.

(2B) Asynchronous Communication

① Sender continues execution immediately after sending a message.
② Message stored by middleware upon submission.

×