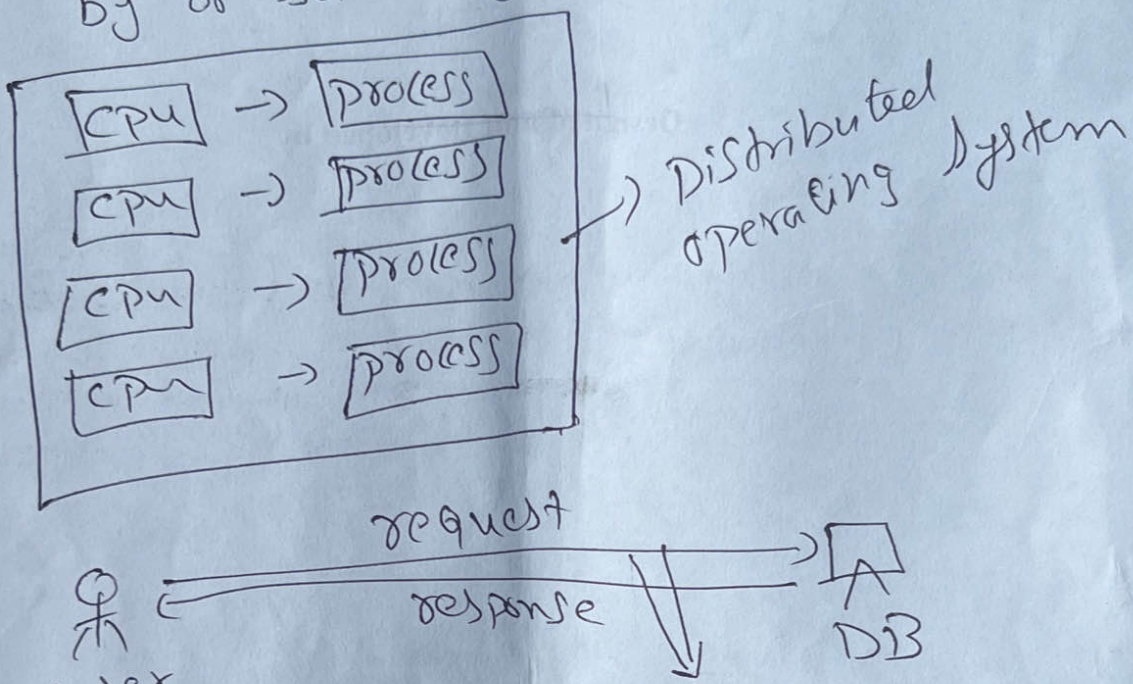
⊛ Introduction to Distributed operating System:-

CPU → process ⇒ This one CPU cannot manage all the things.

∴ The above drawback can be managed by or solved by below fig.

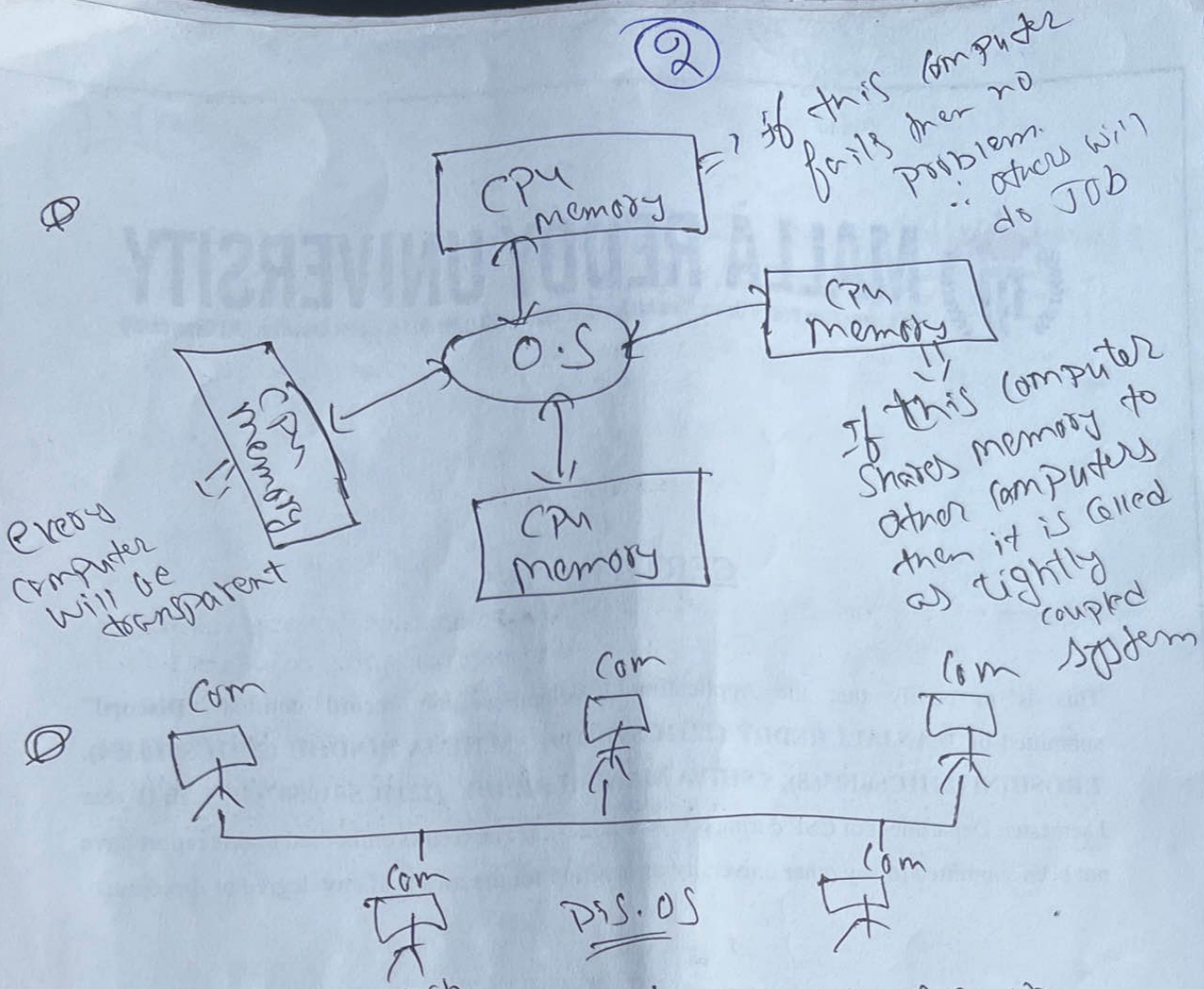


req & response will travelled through n/w & several distributed operating System.

Defination of Distributed operating System:-

→ A distributed OS is the S/w over a collection of independent, networked, communicating & physically seperate computational nodes.

(2)



1) Above each computers having own memory
2) connected in local area n/w & wide area n/w & metropolitian area n/w.
3) All computers are independent connective.
4) Physically separate
5) All computers will communicate each other

DIS O.S done
a) Collaborating in work
b) Sharing the work

Advantages of Dis. OS:-

1) Response time
2) output speed
3) memory utilization
4) N/w utilization.

Process will be divided into Sub (P_1)

↓

Sub P_2

↓

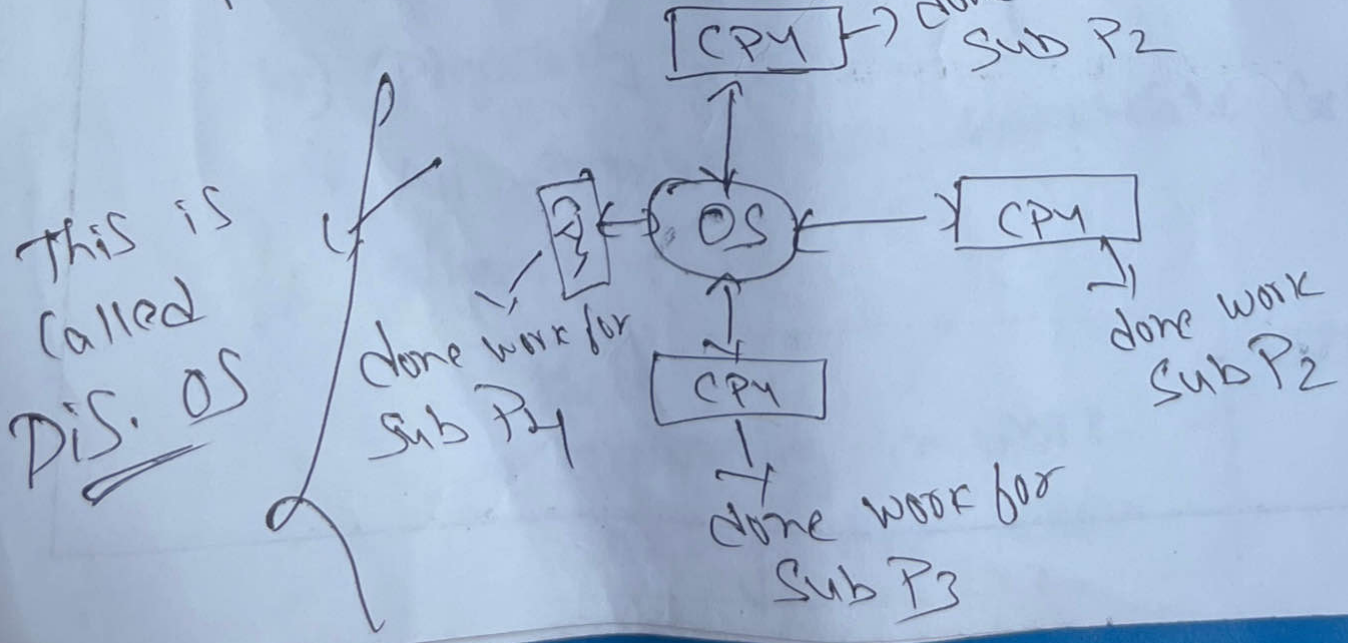
Sub P_3

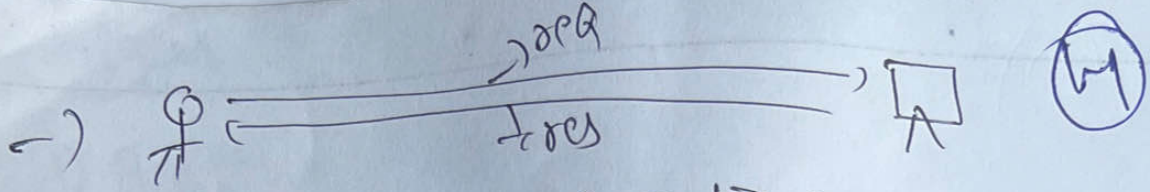
↓

Sub P_4

This is Dis. OS

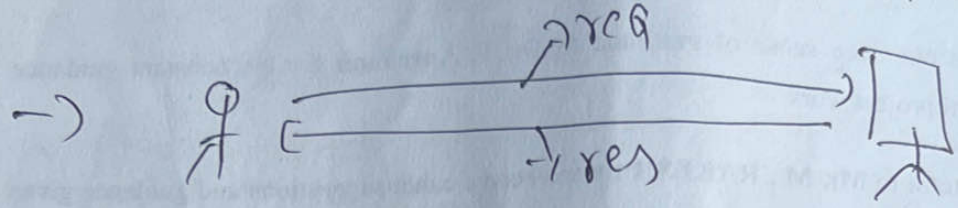
Sub P_2 work done by computer 1
Sub P_3 work done by computer 2
Sub P_4 work done by computer 3





1) Single Computer
2) Single CPU
3) Small Network

→ Server can't take req & response.
∴ No result



1) more computers
2) more CPU
3) Big N/W

→ Server can handle req & response
∴ Good result in short period of time.

→ multiple CPUs
↓
multiple process
↓
work completed

→ Dis. OS

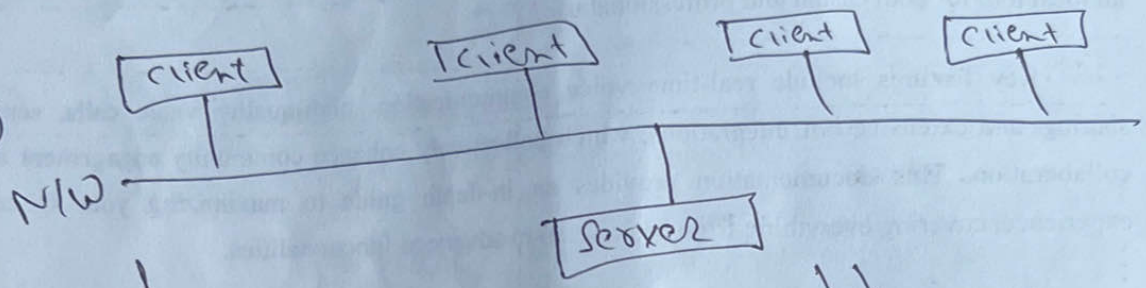
→ one computer depends upon
↓
another computer

→ tightly coupled.

Examples of dis. OS :-

a) Telephone
b) Internet
c) ATM machine
d) Mobiles usage

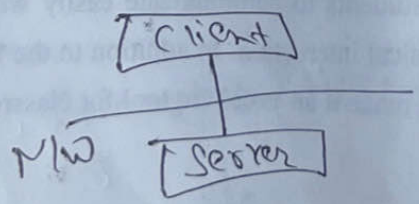
a)



This is also called as client-server system.

1)
-> Need Dis. OS
-> if NO Distributed OS is used here then client have to wait for long period of time so this results late o/p.

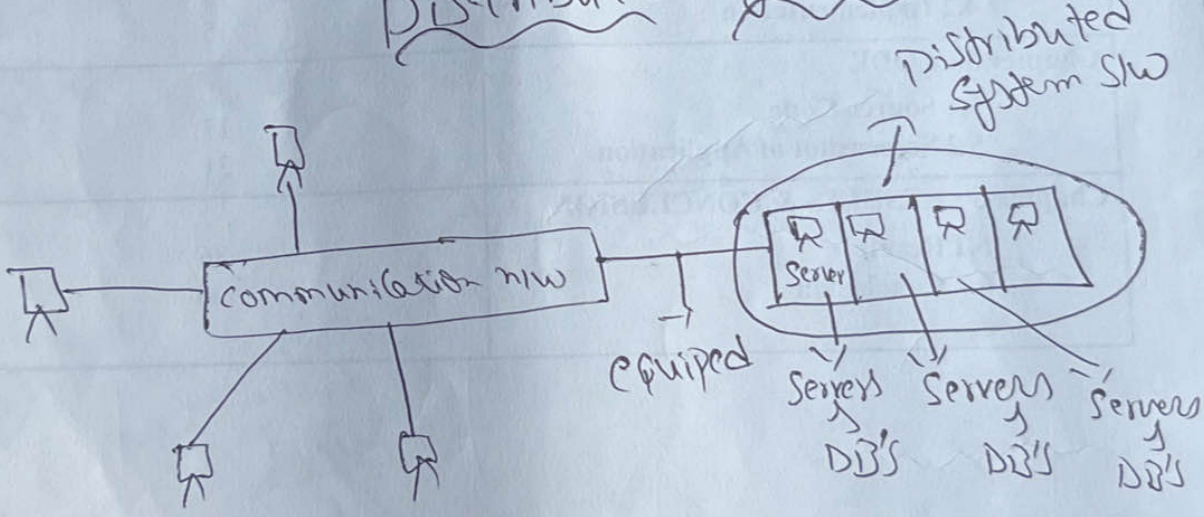
b)



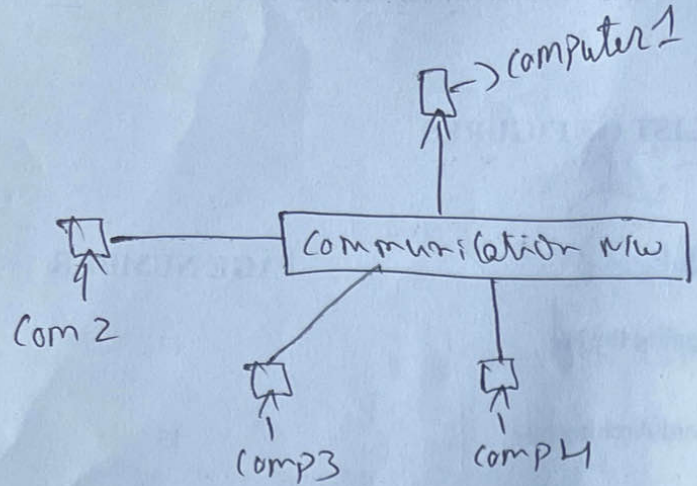
=> Here NO dis. OS is required
=> One Server can easily responds to single client

* DS is loosely coupled
  → No shared memory
* Advantages of DiS. OS:-
  1) Resource sharing
  2) Computation speed up
  3) Reliability
  4) Communication
  5) Quickly responses
  6) Client no need to wait
  7) extensibility
  8) cloud concepts completely depends upon Dis. OS

Distributed System



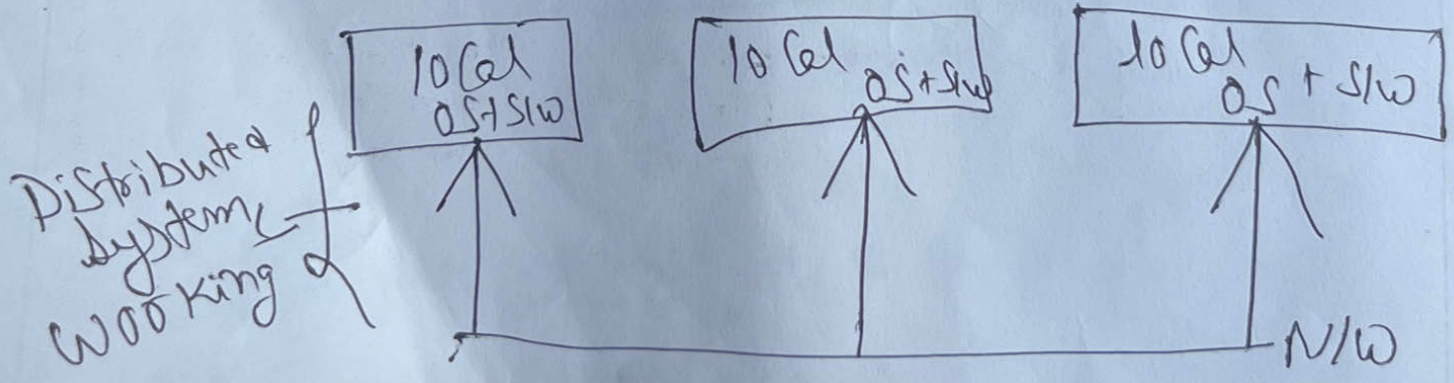
-) Distributed System:- It is a collection of computers linked by a communication n/w & equipped with distributed system SW.



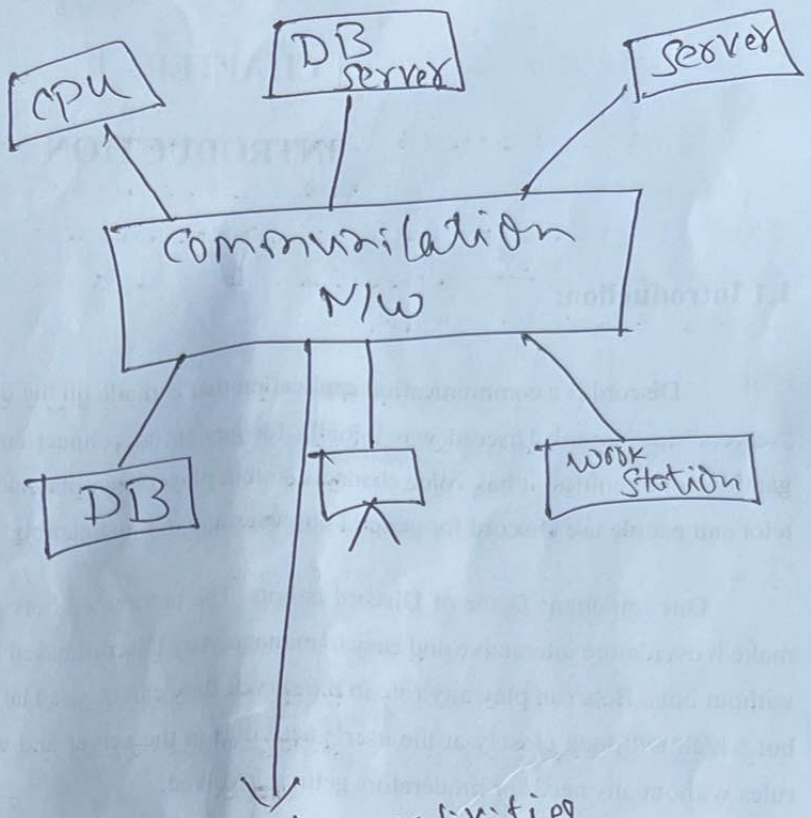
→ coordinate all 4 computer activities
→ coordinate all 4 computer work
→ Sharing resources among 4 computers.
→ Sharing S/w's among 4 computers
→ Sharing data among 4 computers

→ for successfull completion of all above things we need Distributed operating system

→ The concept is called as Distributed system.



Distributed operating system diagram (8)



Coordinate activities
&
Sharing resources

↓

Stores Data in DB.

→ Multiple entities (more users can send a request)

This can be done by many nodes

∴ The above is called Dis. OS

✱ Hardware concepts in Dis. OS:-

① Hardware concepts
   ⓐ multiprocessor bus based
   ⓑ multiprocessor switch based

② S/w concepts
   ⓐ Distributed operating system
   ⓑ n/w OS
   ⓒ Middleware OS

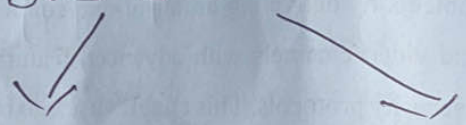
Hardware concepts in Dis. System:-

Shared memory bus based
Private memory bus based
Shared memory Switch based
~~Shared~~ private memory Switch based

SI

MISD
multiple inst
single Data

Bus Based

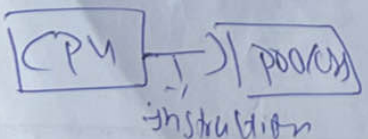


Shared memory private memory

Switch Based

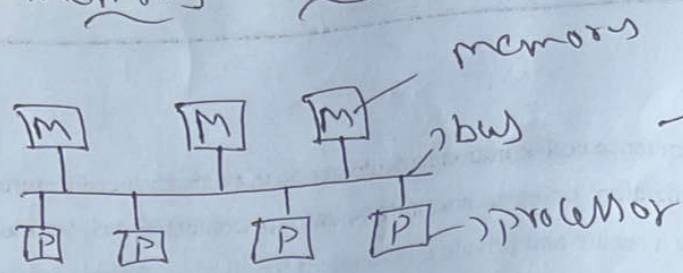
Shared memory private memory.

SISD → Single instruction Single Data



SIMD → Single instruction multiple Data.
MIMD → Multiple " " " .

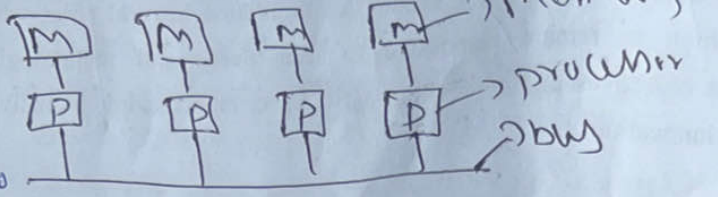
Shared memory Bus-based



computers will share memory

→ in Dhullapally bus stop all can go & travel in Telangana buses (etc)

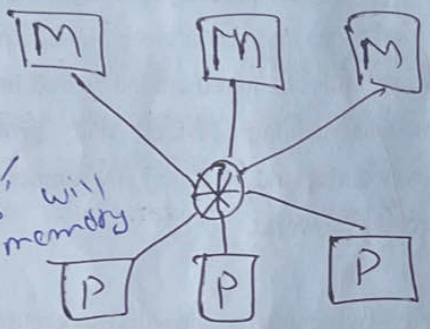
private memory Bus-based



computers will not share memory

→ in Dhullapally bus stop, only mrun students will travel in mrun buses.

Shared memory Switch based:-

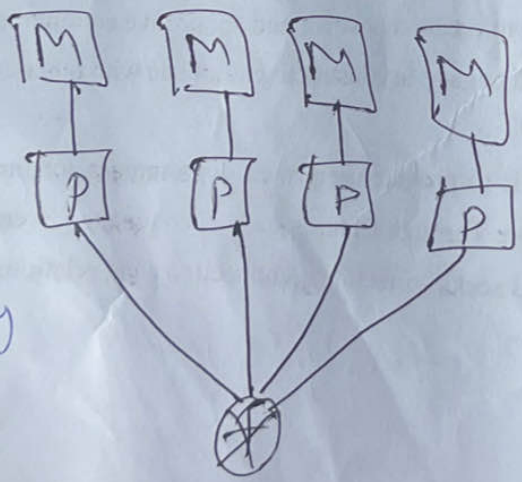


computers will share memory

→ All computers in Dist. of

→ in fixed point all can travel in telangana buses

Private memory Switch based:-



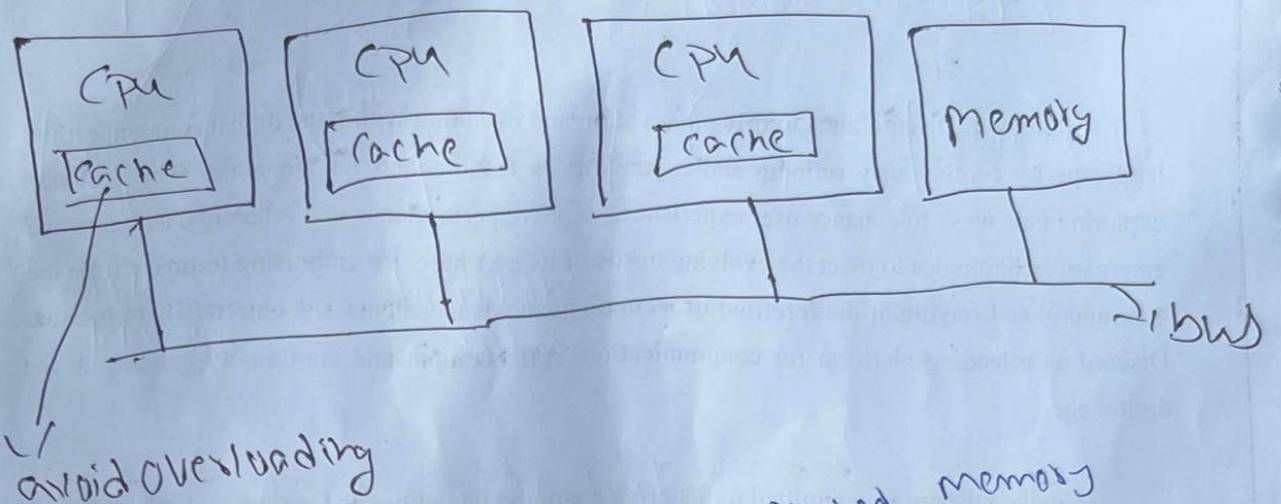
computers will not share memory

→ in fixed point, only mrun students will travel in mrun buses.

* Multiprocessors Bus Based → Shared memory (11)

→ limited scalability (~~less~~ memories are filled then no chance to get another memory in bus)

→ avoid overloading



avoid overloading

Multiprocessors Switch Based → Shared memory

Different CPU's → can access → simultaneously → Different memories

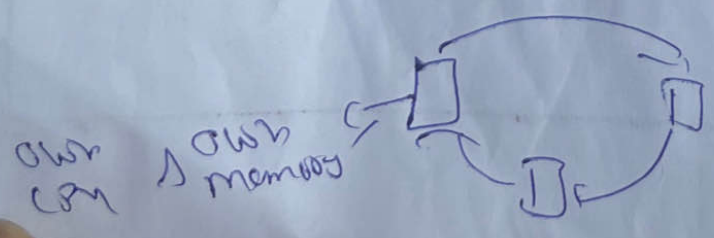
Multiprocessors means → shared memory

* S/w concepts in DiS OS:-

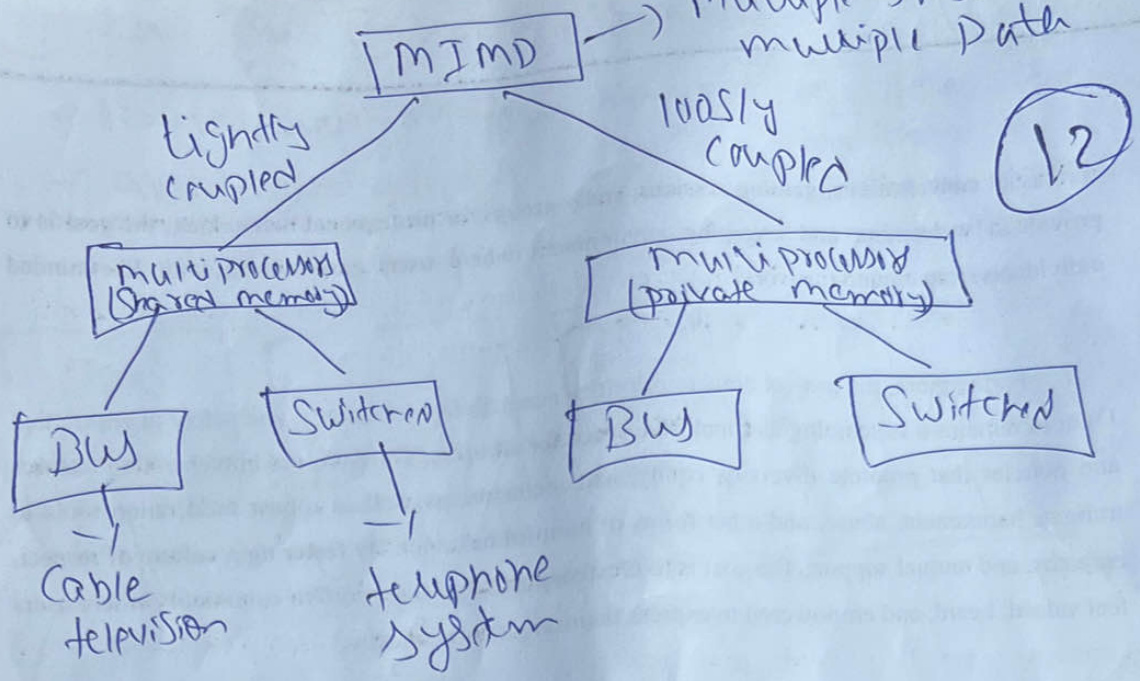
(a) internet connection in labs → N/w DiS OS

(b) Usuage of internet + phone calls → DiS OS

(c) Middle ware OS



loosely coupled Archi



12

(A) Issues in Distributed OS:-

1. global knowledge
2. Naming
3. Scalability
4. Compatibility
5. process Synchronisation
6. Resource management
7. Security
8. Structure of OS
9. Client Server Computing model

* Challenges of DS:-

→ more CPUs
↓
more servers
↓
more req
↓
more response
↓↑
So cant decrease complex

→ going
Manage for all this CPUs ↓ servers ↓ req ↓ response is very tough.

→ Cant send sms for failure tasks. if one task is failure then we can't able to update the news of failure task to others

→ migration

→ load balancing

→ openness (like auto to drop out from auto to place)

→ Scalability

→ Security

UNIT-2

14

Communication in Distributed System:-

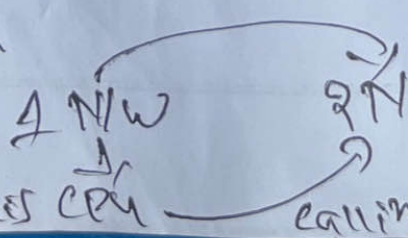
- issues in communication.
  1. Message oriented communication
  2. Remote procedure calls (RPC)
  3. Remote method invocation (RMI)
     - RMI is also RPC but specific to remote obj
  4. Stream oriented communication
     - group of peoples in interview room. one man will come and calls for interview.

→ transfering set of data from sender to receiver

(2) Remote procedure call:-

→ Calling a cpu or server which is present on another N/w by remote way is called remote procedure call

→ Ex: gmail



Client —RPC req→ Server

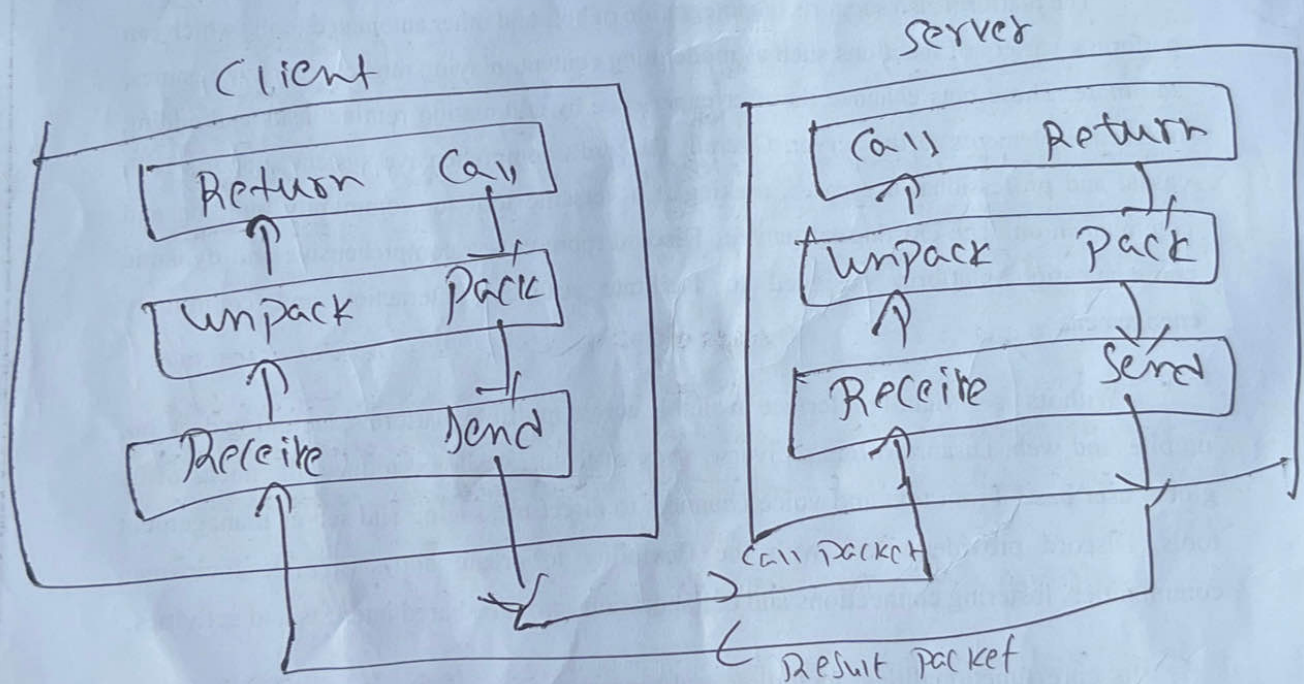
15

Client → (Client Stub) → RPC → Server

Packing (req)
unpacking (res)

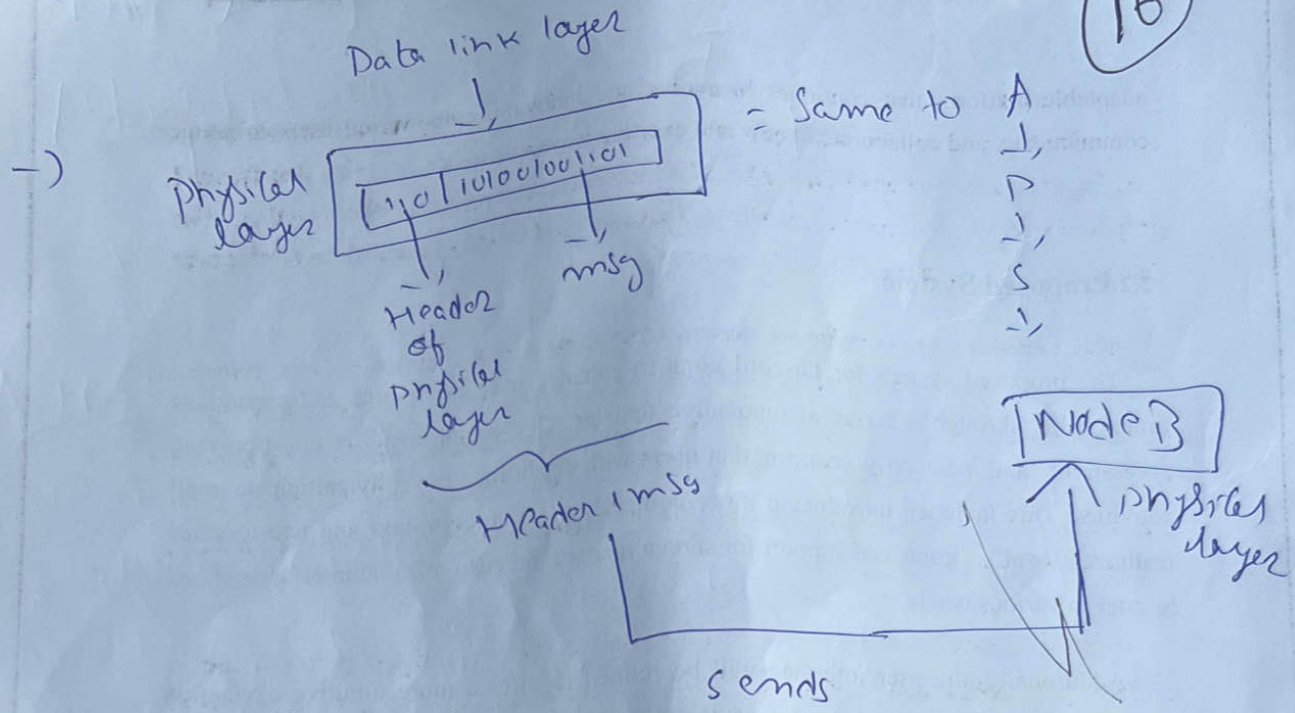
Client → client stub → RPC → Server
(req pack)

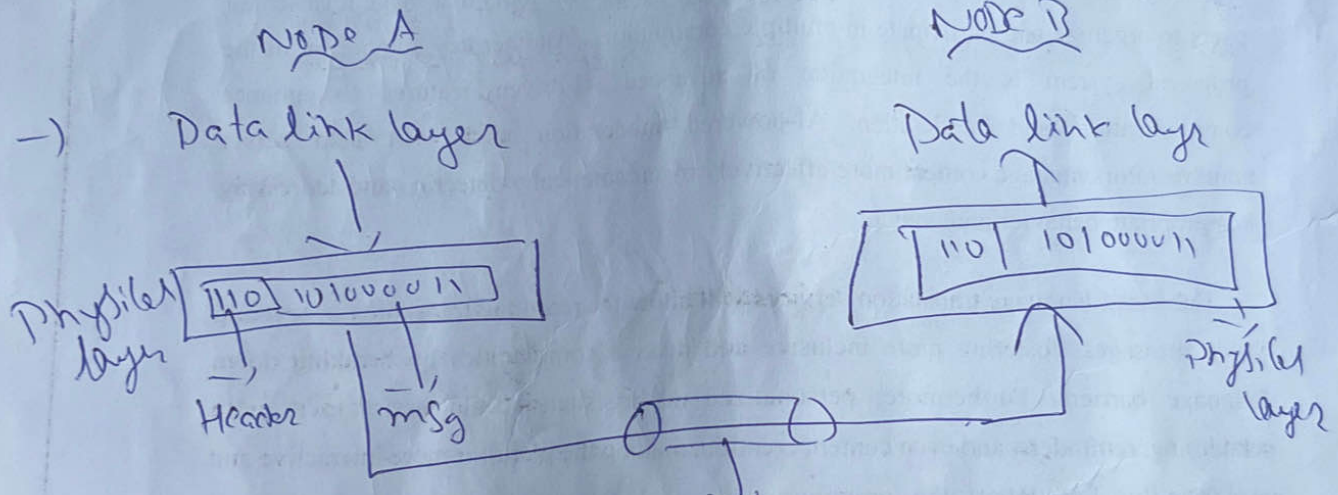
Client ← client stub ← RPC ← Server
(res pack)



- in RPC client itself acts as a server. (taking food owmly or self service)
- client calling server concept is there in RPC. (customer gives signal or calls server for to serve diffrn)
- customer calls server by signal
  similarly
  client calls server by RPC

⑧ Layered protocol in DCS:-

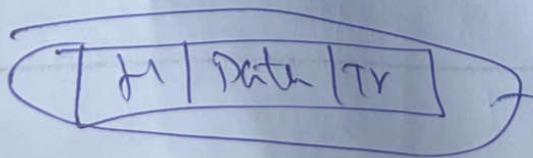


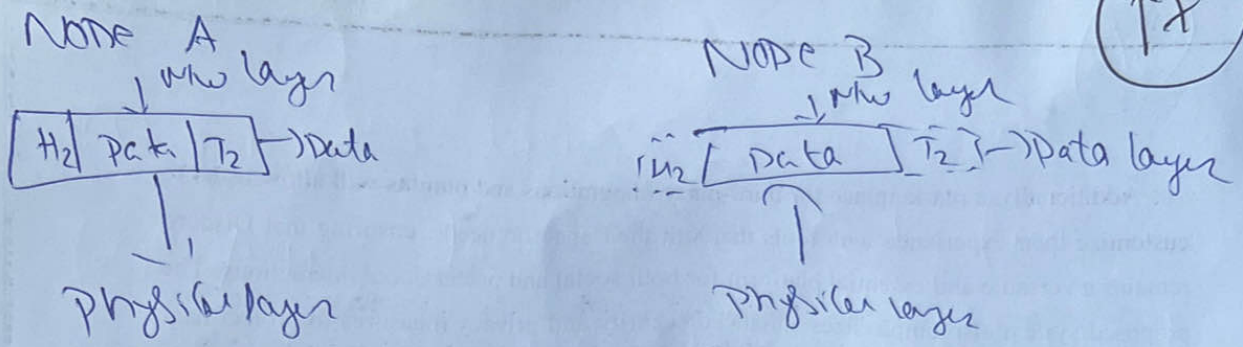


- Transmission media
- Analog signal

- APP, PRE, SESS, PHY → follows Header + msg

- Data link layer :- → frames moving
  → error detection & error correction
- Only Datalink layer has trailer





AIM :-

- using data on internet
- Television (netflix, cable, Disney hot star)
- radio (single layer can transmit data) ex fm : 93.5

using single n/w we can send data

Before AIM :-

router = traffic man

- Some draw backs.
- what are the draw backs
  - fixed size (for transmitting data)
  - voice traffic

fixed size :- more packets have to send then problem. So ∴ fixed size is problem.

- For to solve the fixed size problem the solution is AIM.
- Asynchronous Transfer Mode :-
  - not occuring at the same time.

3 ATM layers:-

① physical layer → so important

② ATM layer → cells into frames → imp for sem

③ ATM adaption layer. → 2nd N/w takes the responsibility of 1st N/w process.

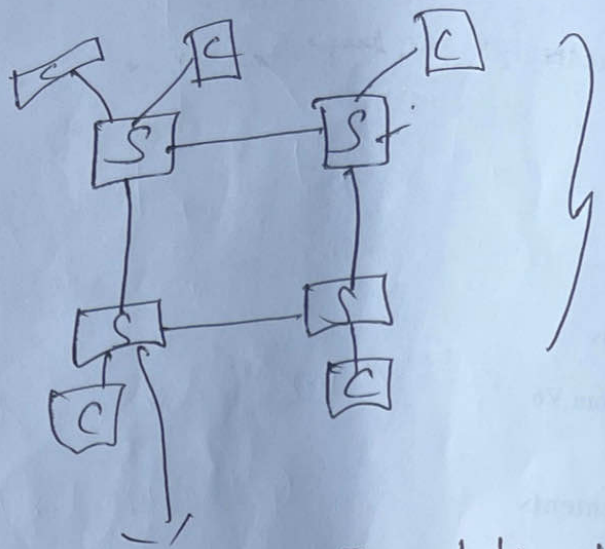
* Identify the path for transmission of packets
+
flow control
+
priority of frames to send
+
error detecting
+
small frames failures is concentrated by a big n/w.

→ ATM cell header
),
comes under ATM layer

ATM adaption layer:-
→ Handles breaking of packets into reassembling.

ATm switching

S → Switch → Both I/p & o/p



→ more advantages like metro map in Telangana

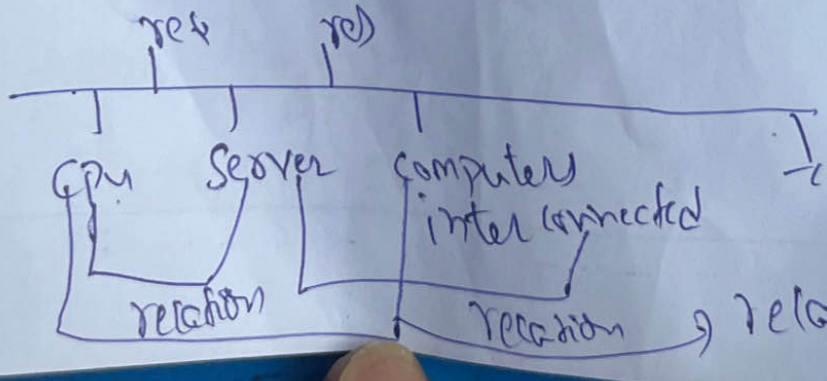
information stored

S → Switch. think it is a metro station.

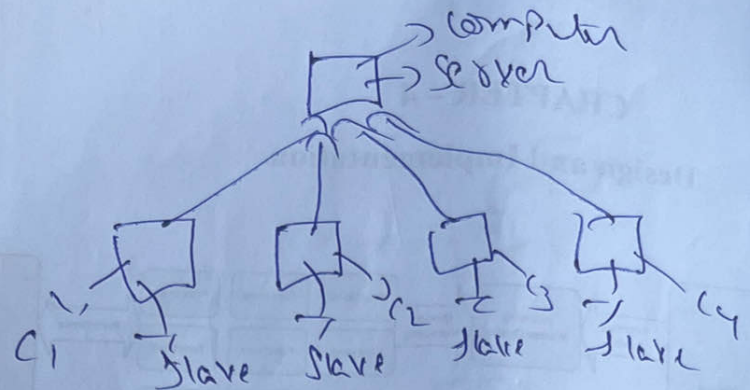
→ if two train comes same time at same direction. → drawback.

→ if two trains comes in opp. direction then no problem. → because o/p's direction are different.

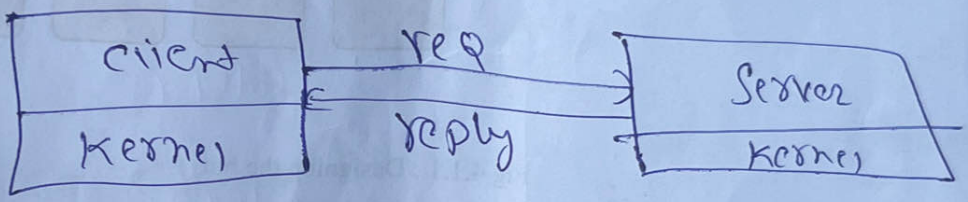
client-Server model in DOS



Ex:- labs → one computer is server & remaining all are slaves.

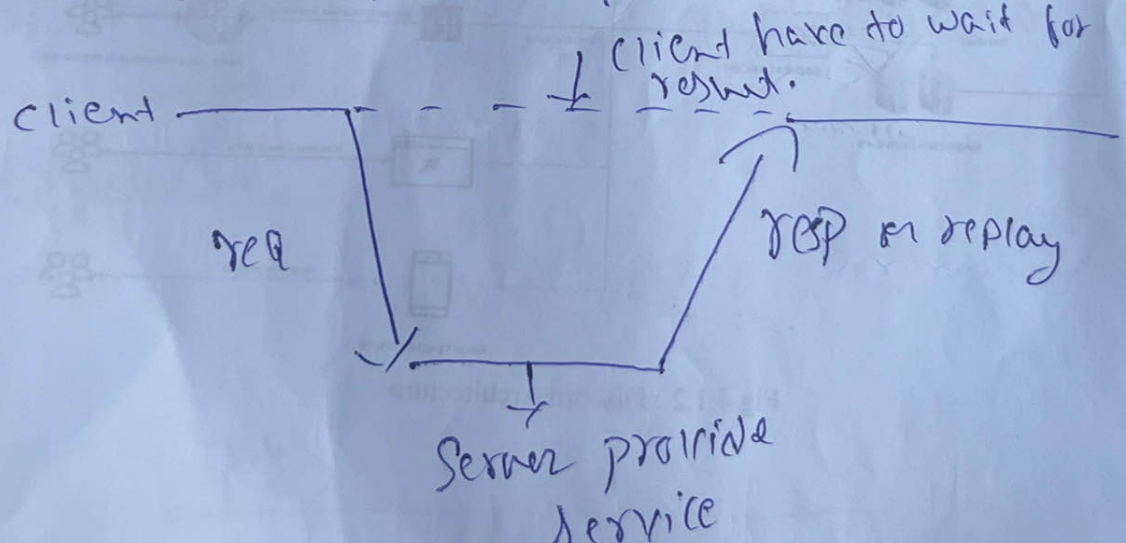


→ Client & Server roles are assigned & changeable (frequently)

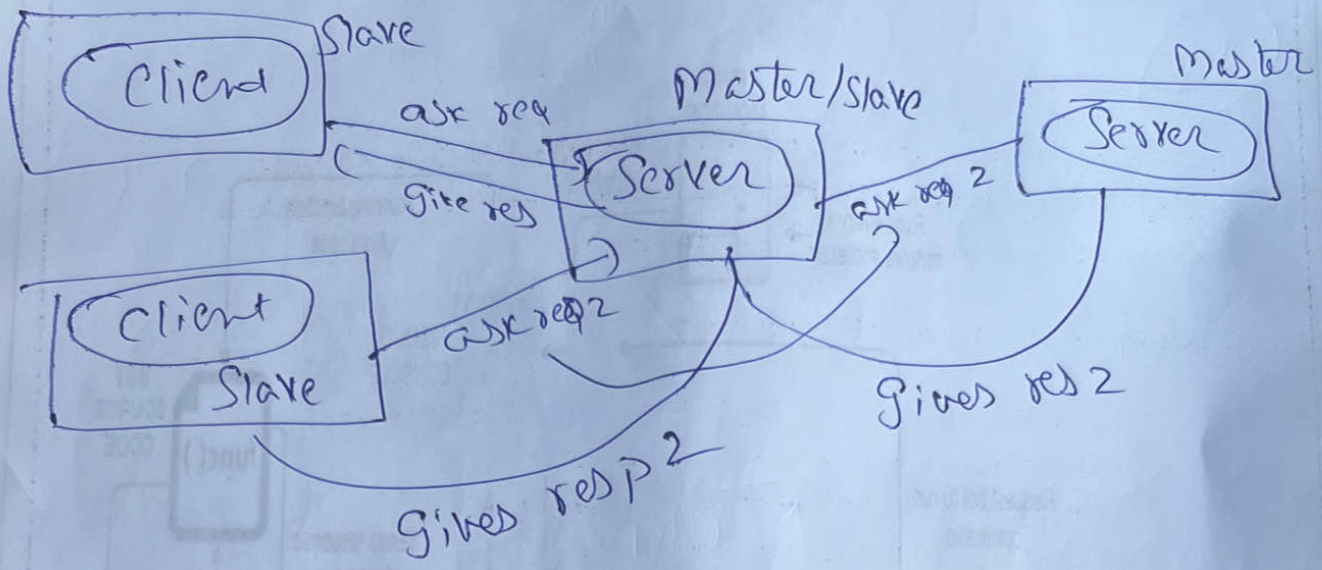


→ Some times client will act as Server and Some times Server will act as Client.

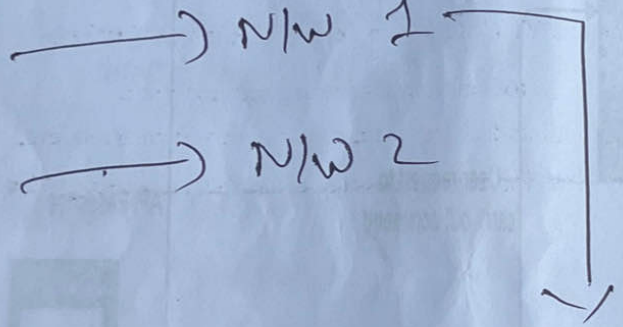
Client - Server timing diagram:-



Client invoke individual servers:-



Remote procedure call:-



N/w 1 is calling packets in N/w 2 is called RPC

Why N/w 1 calls packets or program or service from N/w 2

→ For Security

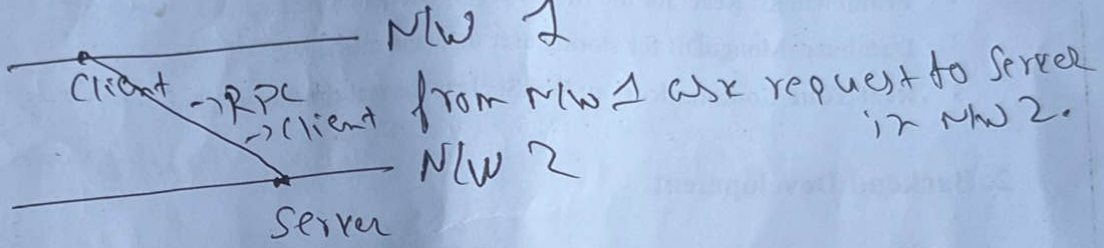
→ Entering user name & password details in another computer because of speed & security

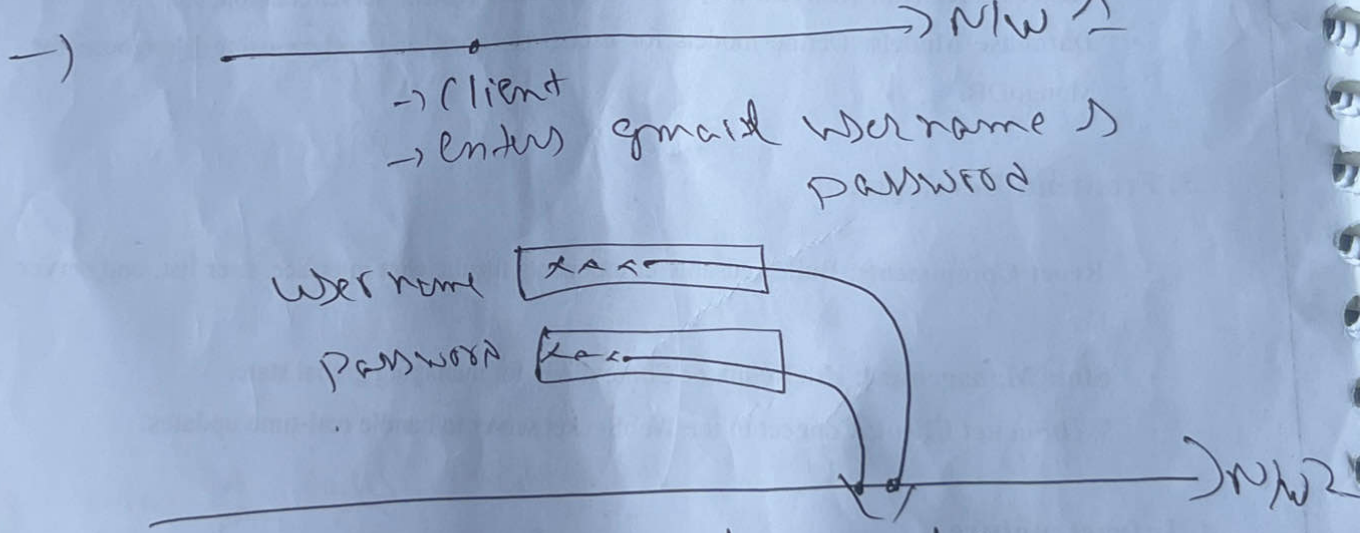
→ Security + Speed.

→ RPC have some protocols for to use service from another n/w. (have to check whether n/w is free or not)

→ RPC follows client server model.

RPC follows { principals.
client server model.





22.1

Client (sending msg or req)

↓

Client Stub (Packs)

↓

RPC run (takes order to run by client)

↓

RPC starts running

↓

Unpack (from pack to unpack)

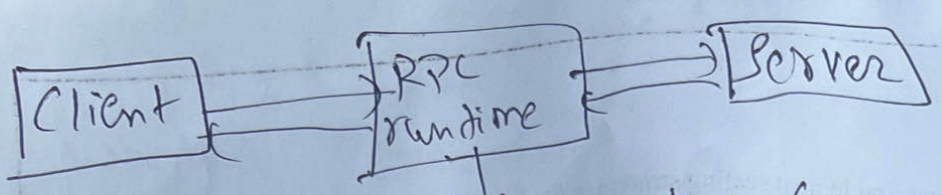
↓

Submits to user

happens in n/w

happens in another n/w

The above concept is called RPC.



↓
manages the communication b/w client & server.

Client Stub:- → used for to protect packets where packets are packed is known as Client Stub

Server Stub:- → used for to protect packets where packets are unpacked is known as Server Stub.

Client
↓
call or req
↓
pack(Client Stub)
↓
Send from client → ↕ N/W
↓
receive to Server → another N/W
↓
unpack (Server Stub)
↓
~~asked~~ faked response or result
↓
Submit

RPC

Server
↓
response (to user) → local OS
↓
packed
↓
sending to client machine or client N/w
↓
reached to client machine or client side N/w
↓
unpack (Server Stub) → remote OS
↓
reached to user

) Happened in Server machine } → RPC
) Happened in client machine

RPC Comes when:-

a) after pack
b) after unpack

Om namaha Srivayer

Group Communication:-

→ Data must be arranged in a structure from to maintain good group communication.

DS → group communication ✓

If DS is not there → no group communication

DS = Data Structure

External data representation:-

RPL → Remote procedure call
RMI → Remote method Invocation
g-c → group communication

3 will good only when data is structured or data must be format. (like tree)

Message Passing

→ Synchronous msg passing
→ Asynchronous msg passing → client will not wait for server Ack. it sends group of packets continuously

Synchronous msg Passing:-

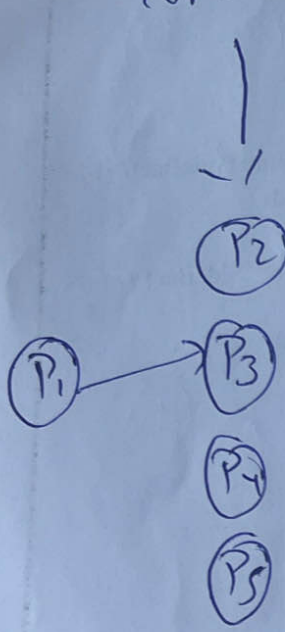
Client or Sender — req → Server or receiver

have to wait for server sender Ack !

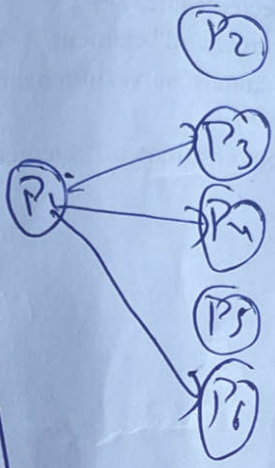
V

Group communication

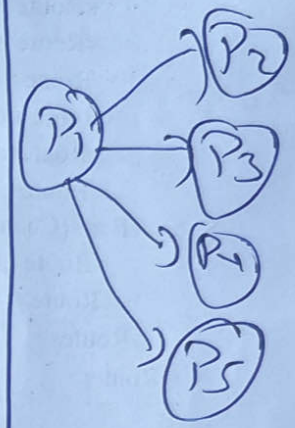
- Unicast communication
- Multicast communication
- Broadcast communication



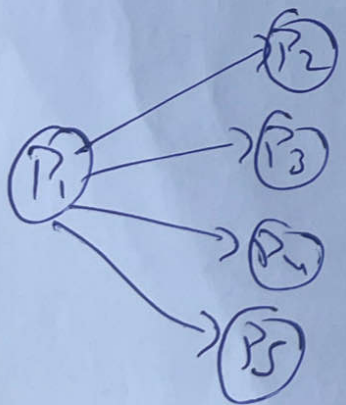
P_1 process communicating with only P_3 process (unicast)



P_1 process communicating with only a group of the process (multicast)



P_1 process will communicating with all process



Broadcast communication

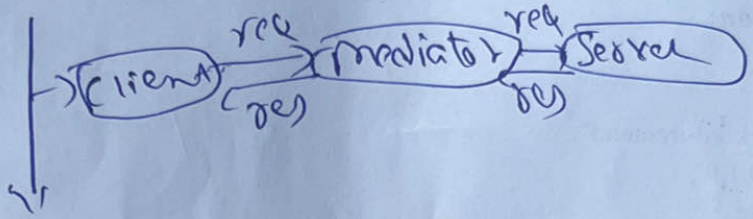
P_1 process communicating will all process at a same time is called group communication, or broadcast communication

Communication mechanism:-

(a) message passing
- Synchronous msg Passing.
- Asynchronous " "

(b) RPC → remote procedure Call

(c) Publish - Subscribe messaging



(d) Distributed shared memory

N_1 sends req, or res to N_2 indirectly.

(e) Peer to Peer (RP) Communication

N_1 sends req or res to N_2 directly.

Essential features of group Communication:-

→ Atomicity (all or nothing):-
When a msg is sent to a group, it will either arrive correctly or none of them.

Om Namaha Shivaya

UNIT-3

Synchronization:

→ Co-ordination of actions is called Synchronization

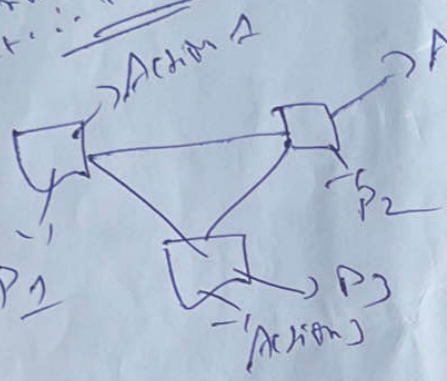
→ No co-ordination of actions is called asynchronization

→ Asynchronization → initial stage

actions will repeat ∴ time waste

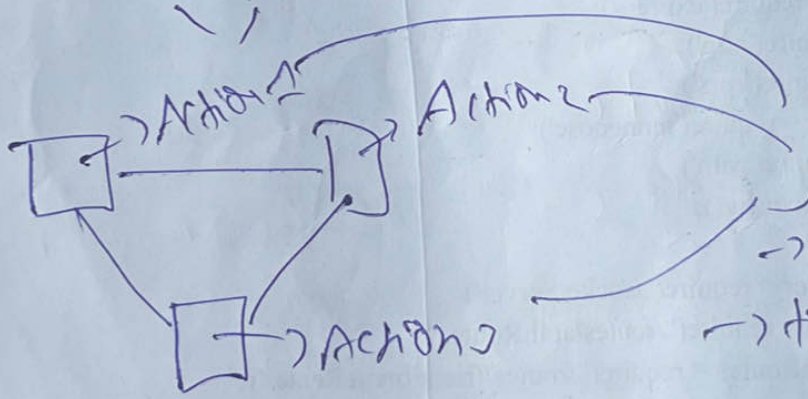
→ Synchronization → (Some times need to cooperate / Synchronize)

actions are not repeat ∴ time no waste



→ Actions are not co-ordination.
∴ it is called as asynchronization.

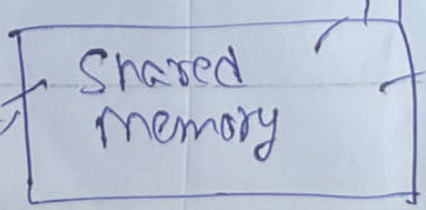
|| makes



→ co-ordinating
→ must need
→ time save

⊛ Synchronization in Centralized

Because of this no action is repeated



process will check the action status.
Each process can see the updation on this shared memory

P_1 → takes Q_1 action at t_1 time

↓ after some time (if P_1 completed Q_1 action)

P_1 → leaves Q_1 action ⇒ update in shared memory (∵ so no other process touches Q_1)

↓

P_1 → takes another action Q_2 at t_2 time

↓

P_1 → P_1 update in shared memory about Q_2 action is in process.
∵ Don't touch Q_2

→ Synchronization in centralized ones

↓

→ This is easy
→ event are in order.

Synchronization in centralized

→ Easy
→ Shared memory is there
→ Synchronization is happened so easy
→

Synchronization in distributed system

→ harder
→ No shared memory
→ Synchronization is happened but difficult