(AC) 8/1/2020

Page No. / 1
Date / / /

Modl : Introduction.

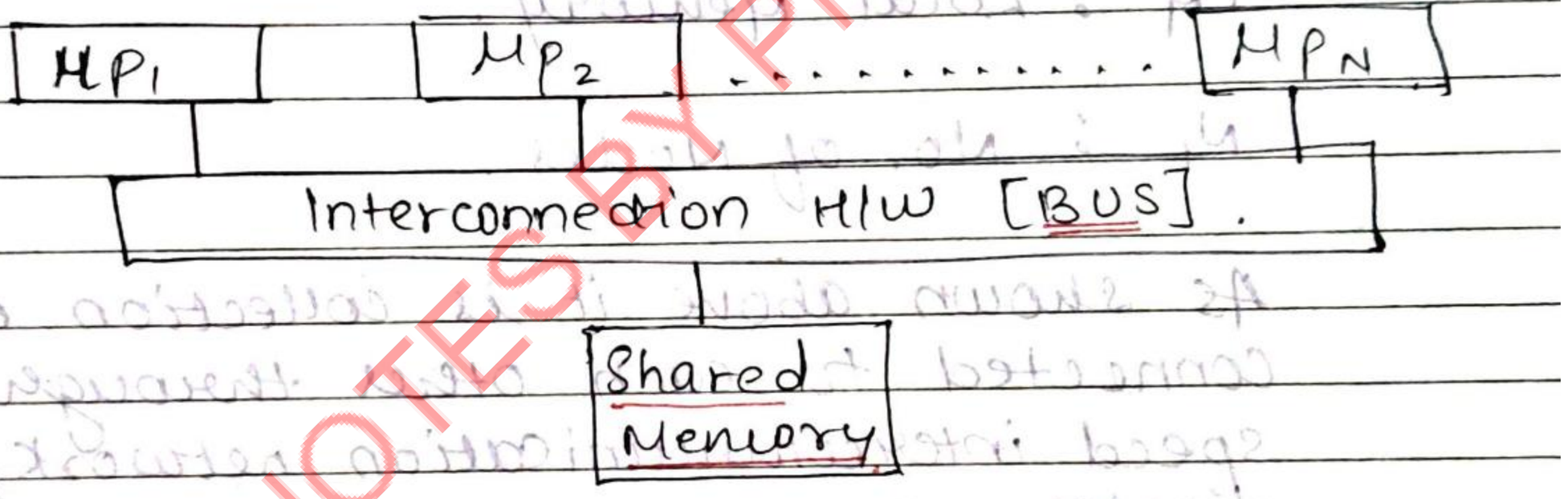
(*) Defination of Distributed Computing: ? coherent ?.

Collection of nodes connected to each other through a communication n/w and interact with each other through a message passing, this is known as Distributed System

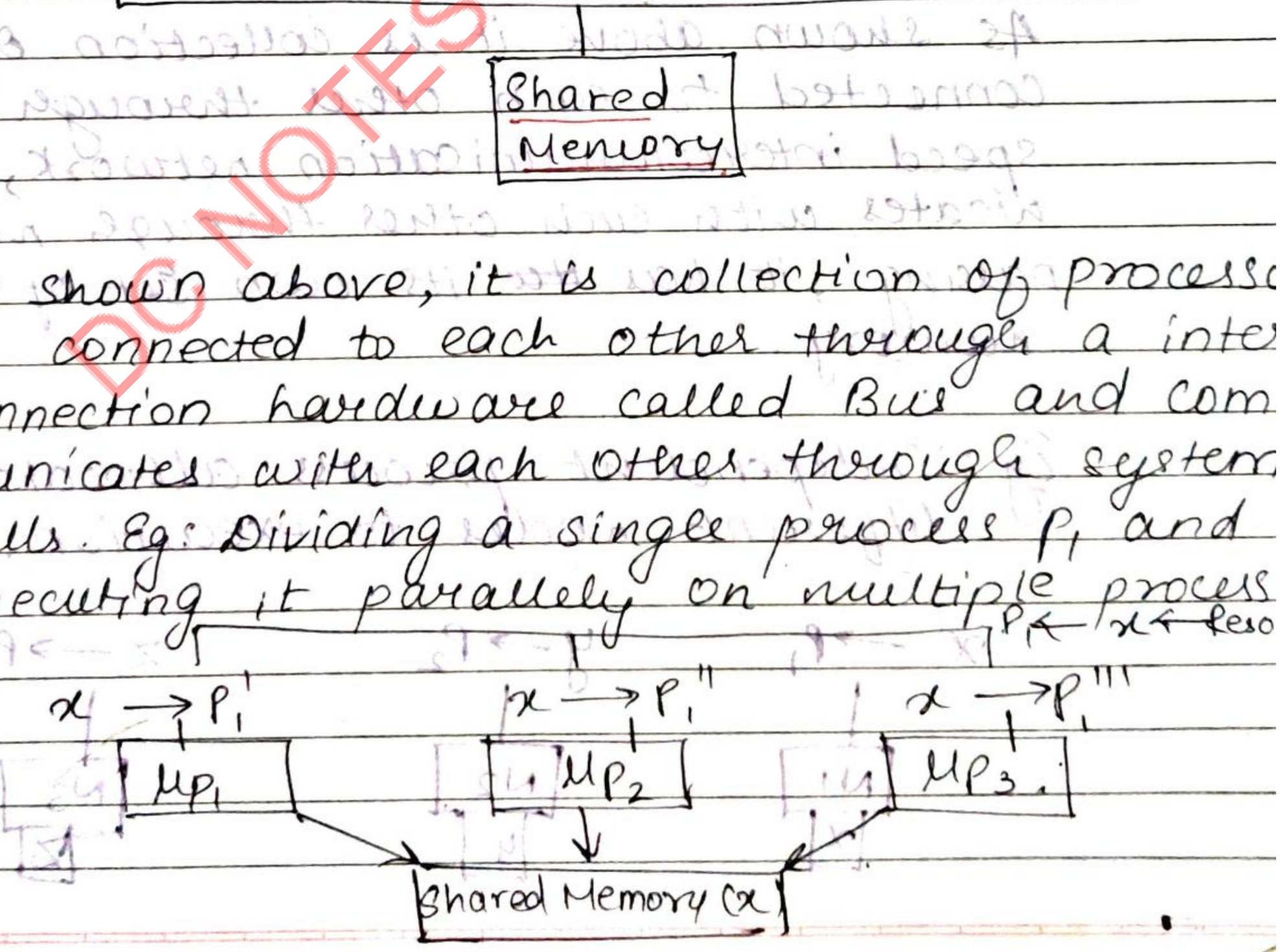
(*) Hardware concepts:

[TP]

i) Tightly coupled System: (Multiprocessor)
It is also known as parallel System.



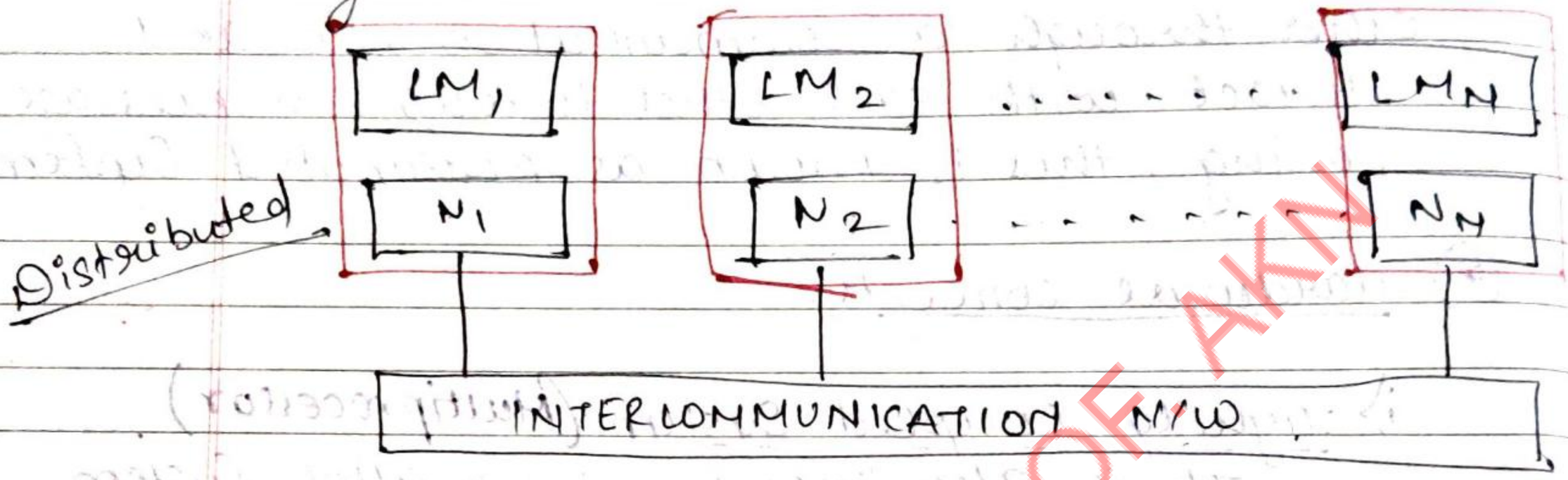
As shown above, it is collection of processors connected to each other through a interconnection hardware called Bus and communicates with each other through system calls. Eg: Dividing a single process P_1 and executing it parallely on multiple process



It has a Shared Memory.

ii.) Loosely Coupled System : (Multicomputer)

- It is also known as distributed system.

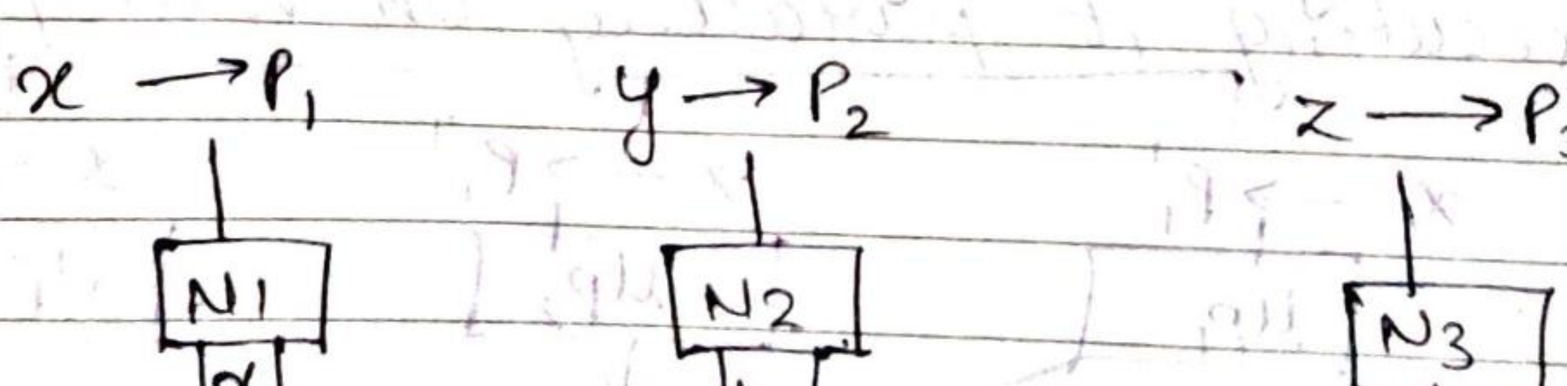


LM : Local Memory.

N_1 : No. of Nodes.

As shown above it is collection of nodes connected to each other through a high speed intercommunication network, communicates with each other through message passing, it has ~~the~~ its own set of local memory.

Eg : Independent process allocated to different nodes to execution.



(*) Software concepts.

To make distributed system work, an OS known as distributed based OS is required which are of following types.

i) DOS. (Distributed Operating System)
ii) NOS (Network operating System)

parameter	DOS	NOS.
1. System Image	- Gives the view of virtual independence.	- Gives the view of multiprocessor.
	- High transparency	- No transparency.
	- Good.	- Bad.
2. Autonomy.	- Same as an all nodes	- Different OS as per user choice.
	- Creates homogenous environment	- Creates Heterogenous environment
	- Bad.	- Good.

3. Tolerance. Fault Tolerance.	- Migrating job in case of failure, due to homogen- ous enviornm- ent.	- Migrating job difficult in case of failure due to hetero- genous enviorn- ment.
	- Good	- Bad.
4. Architecture.	Fig A	Fig B.

Fig A : DOS.

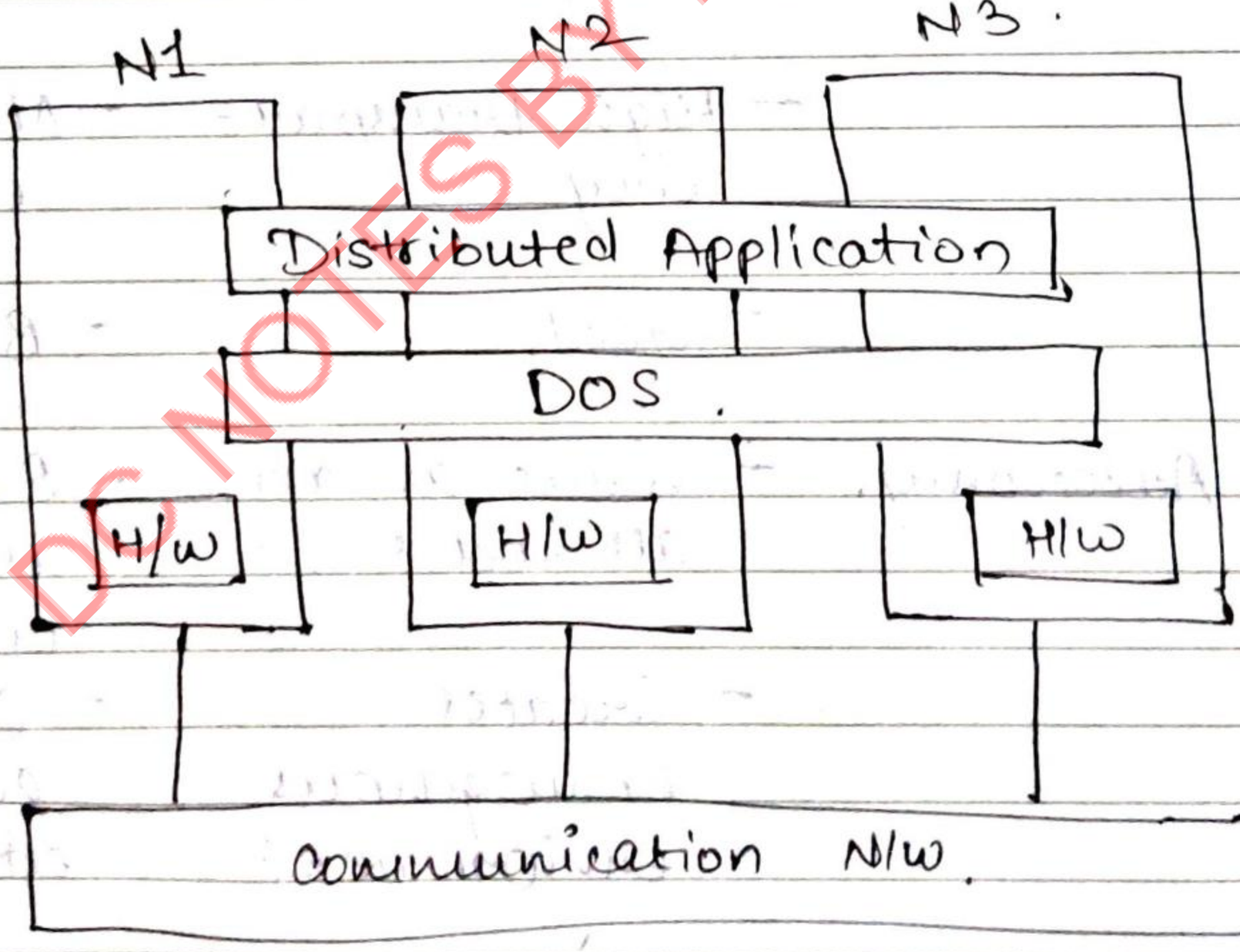
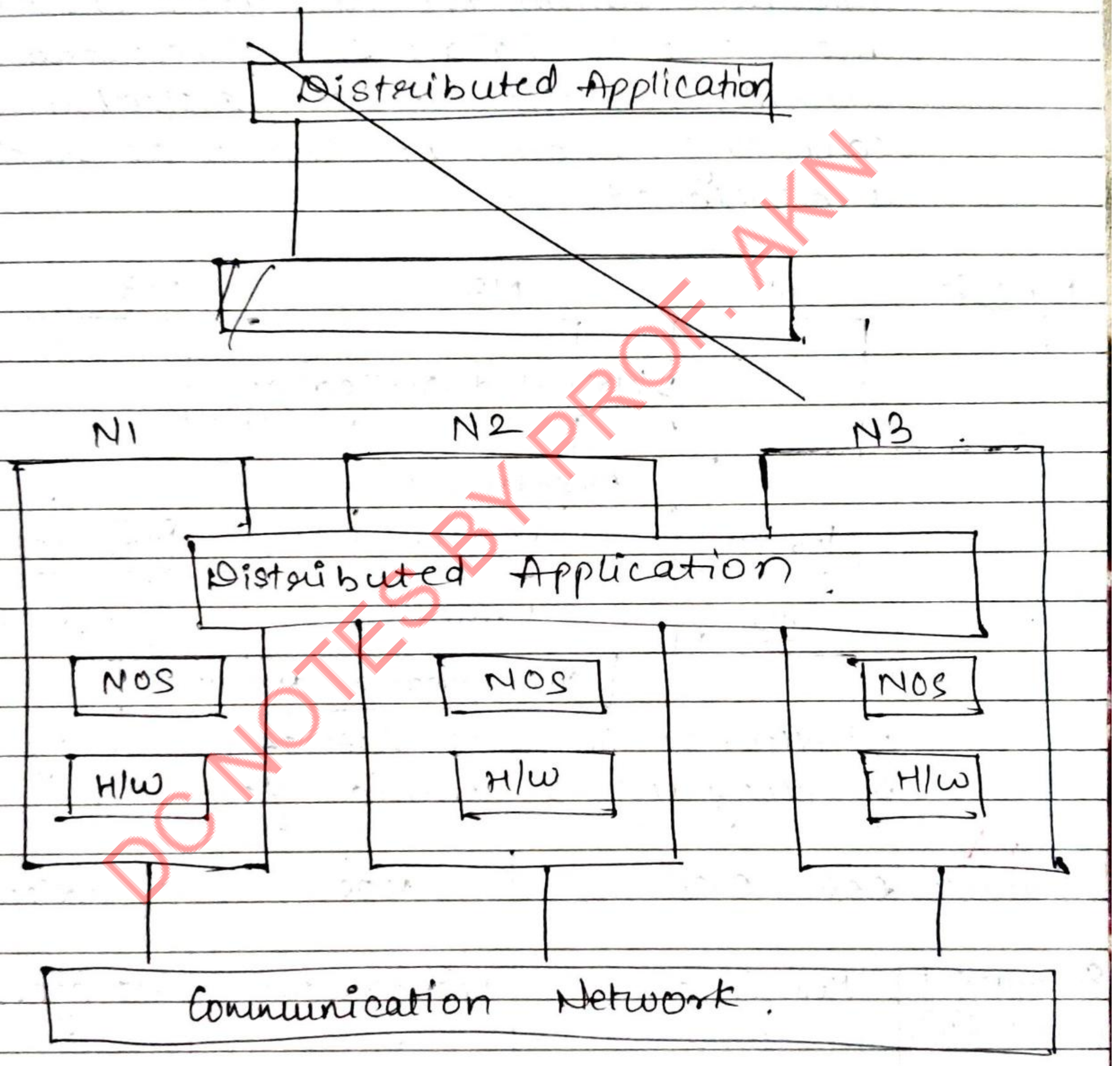


Fig B : NOS.

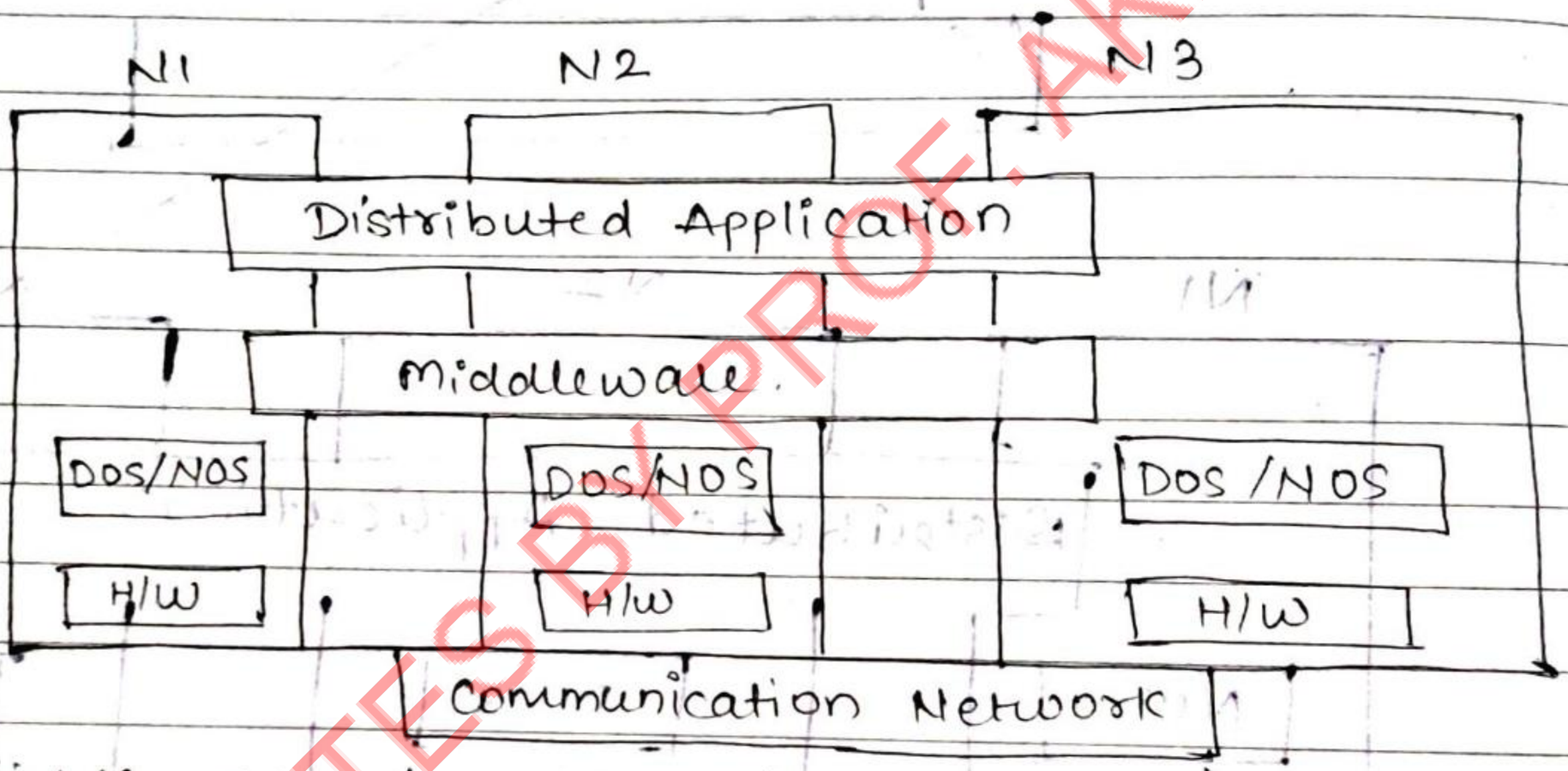


13/01/2020

Middleware.

Neither NOS nor DOS qualifies as a good operating system for distributed enviornment. and hence to handle such heterogenous enviornment, a concept of DCE ; Distributed computing Enviornment ; known as middleware is developed.

Fig c :-



Middleware handles heterogeneousinity by designing a common format.

parameter	~~DOS~~ DOS	NOS	Middleware.
Enviornment (w.r.t H/W).	Homogenous	Heterogenous	Heterogenous.
No. of OS	Single	multiple	Multiple.
Degree of transparancy.	High	Low / No.	High

Parameter	DOS	NOS	Middleware.
Fault Tolerance.	High	Low	High.
Openess.	close	Open.	Open.
Architecture.	Fig A	Fig B	Fig C.

⊛ Design Issues of Distributed System. (Detail ans : Notes).

{state for IA} only name.

1. Heterogenity:
Different architecture w.r.t hlw and slw. Soln: use of middleware.

2. Security:
Since all resources are in network, it should be highly protected
Soln: use of cryptography, use of authentication, use of authoricity.

3. Relicability:
focusing on failure free operation. (fault tolerance system)
Eg: If system A fails then its resources can be migrated to the other devices in the nlw.

Date / / /

4. Flexibility :

Ease of upgradation and modification due to independent modules.

5. Scalability :

Adding more devices should not affect working of distributed system.

6. Availability :

Systems should be always available due to reliability.

7. Emulation of existing software :

Backward compatible.

Imp ★ 8. Transparancy :

User should be able to see only what is desired for him.

i) Location Transparancy.

User shouldn't be aware of the fact that where exactly the server is located.

ii) Access transparancy

User should'nt be aware of the fact that how the data is accessed and presented.

iii) Persistent transparancy.

User shouldnt be aware of the fact that from which part of the memory data is accessed.

Imp (viva) iv) Relocation transparancy.

User shouldn't be aware of the fact that data is moved to another node before execution.

Imp (viva) v) Migration transparancy.

User shouldn't be aware of the fact that data is moved to another node during execution.

vi) Concurrency transparancy.

User shouldn't be aware of the fact that the data is processed concurrently.

vii) Failure transparancy.

User shouldn't be aware of the fact that system has failed and still replying.

viii) Replication transparancy.

User shouldn't be aware of the fact that the data is copied on multiple nodes.

(*) Goals of Distributed System /

Why distributed System is gaining popularity?

{Refer notes (pdf) }.

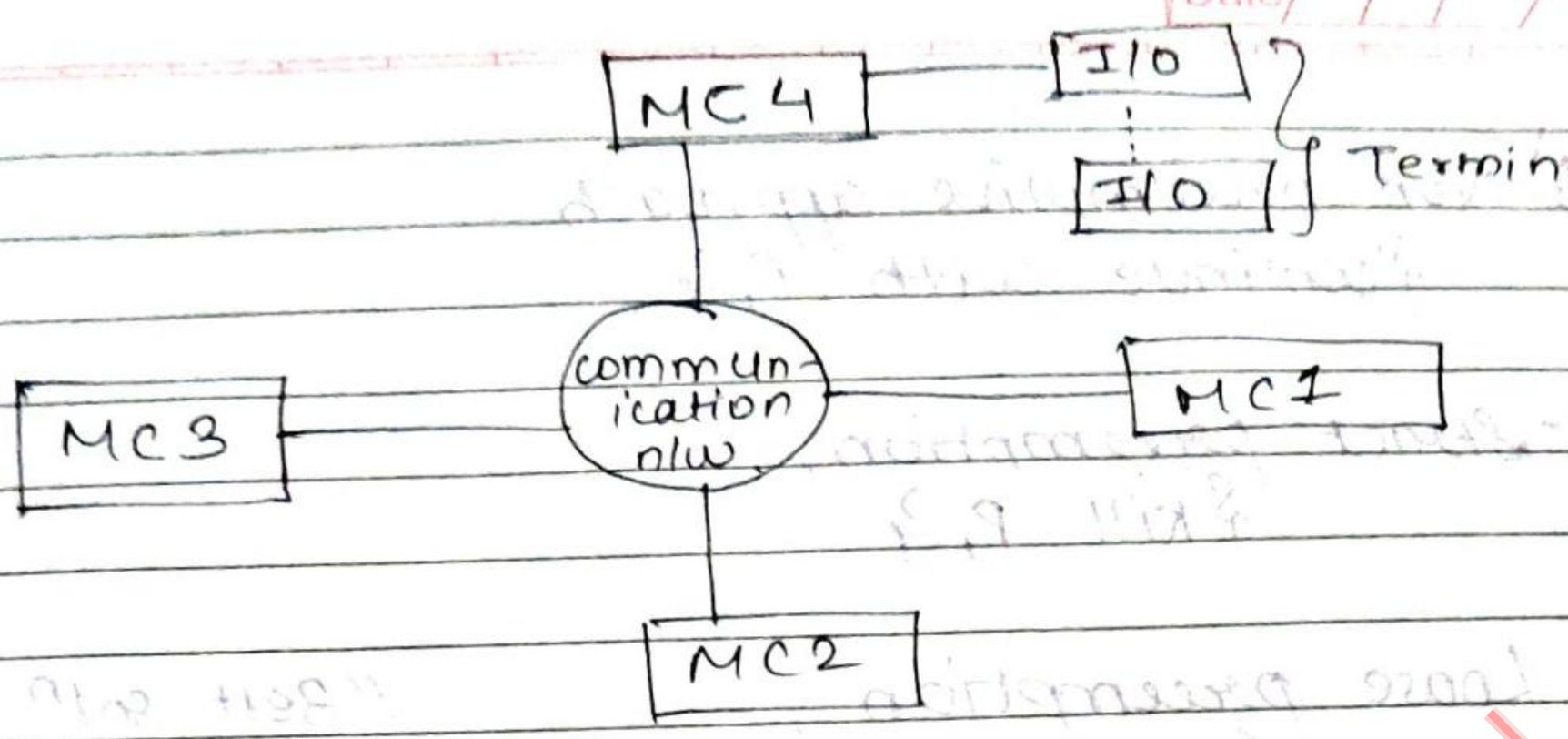
14/01/2020.

(*) Distributed computing Models.:

System {diag. imp}

i) Mini computer.

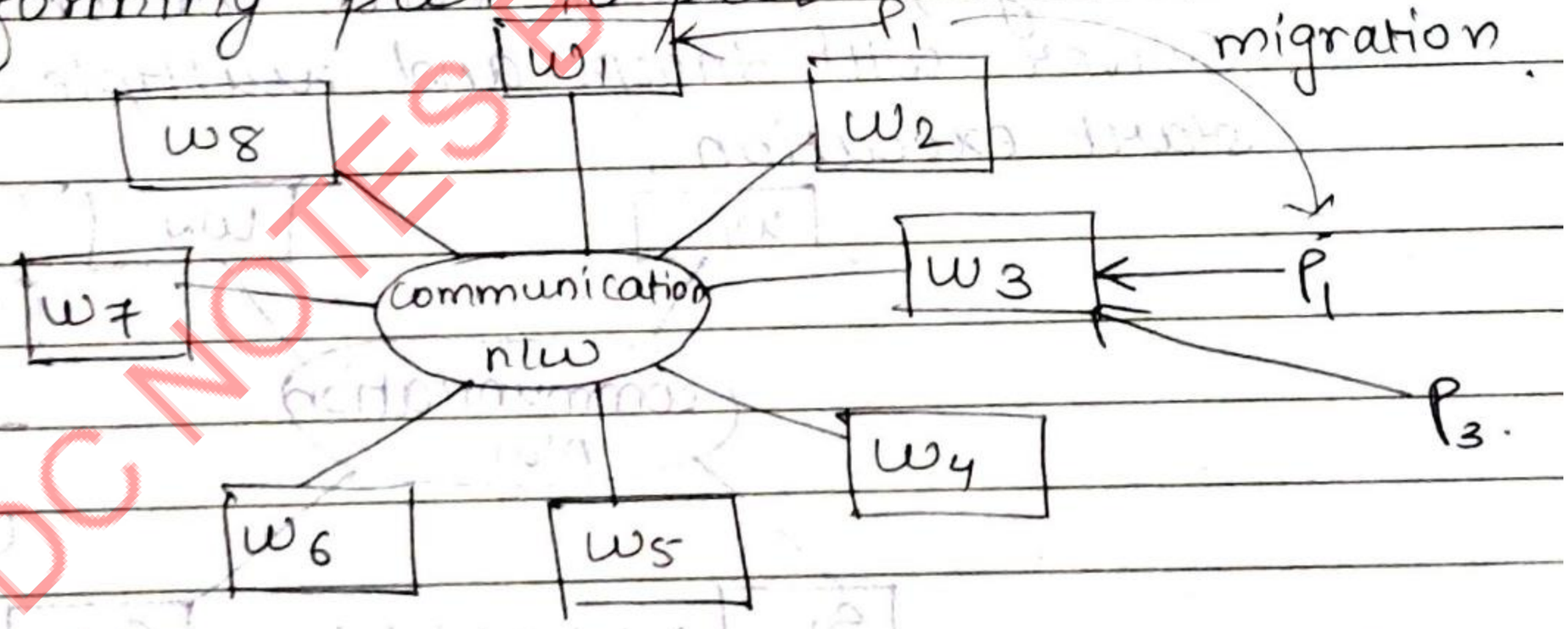
Collection of mini computers connected to each other through communication n/w and allowing user interact with I/o terminals.



Only local logins allowed.
Information sharing but no resource sharing

2) Workstation model.

Collection of workstation connected ~~through~~ to each other through communication n/w forming peer to peer architecture.



Allows information and resource sharing.

Challenge : W_1 migrates P_1 to W_3, after 80% execution
W_3 while executing P_1. (alone)
local process P_3 is submitted for execution,
Now, the question is W_3 will execute P_1 or P_3?

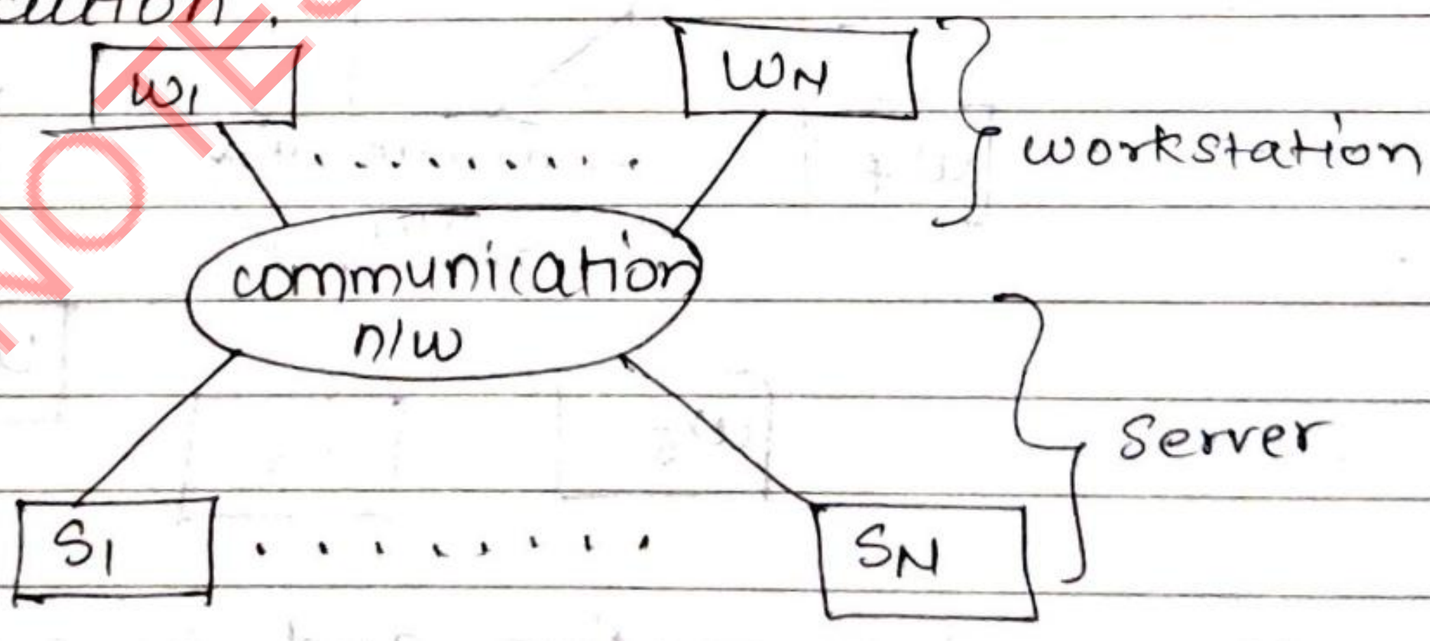
Soln :- i) Non-preemptive approach.
{continue with P1 }.

ii) Strict preemption.
{kill P1 }

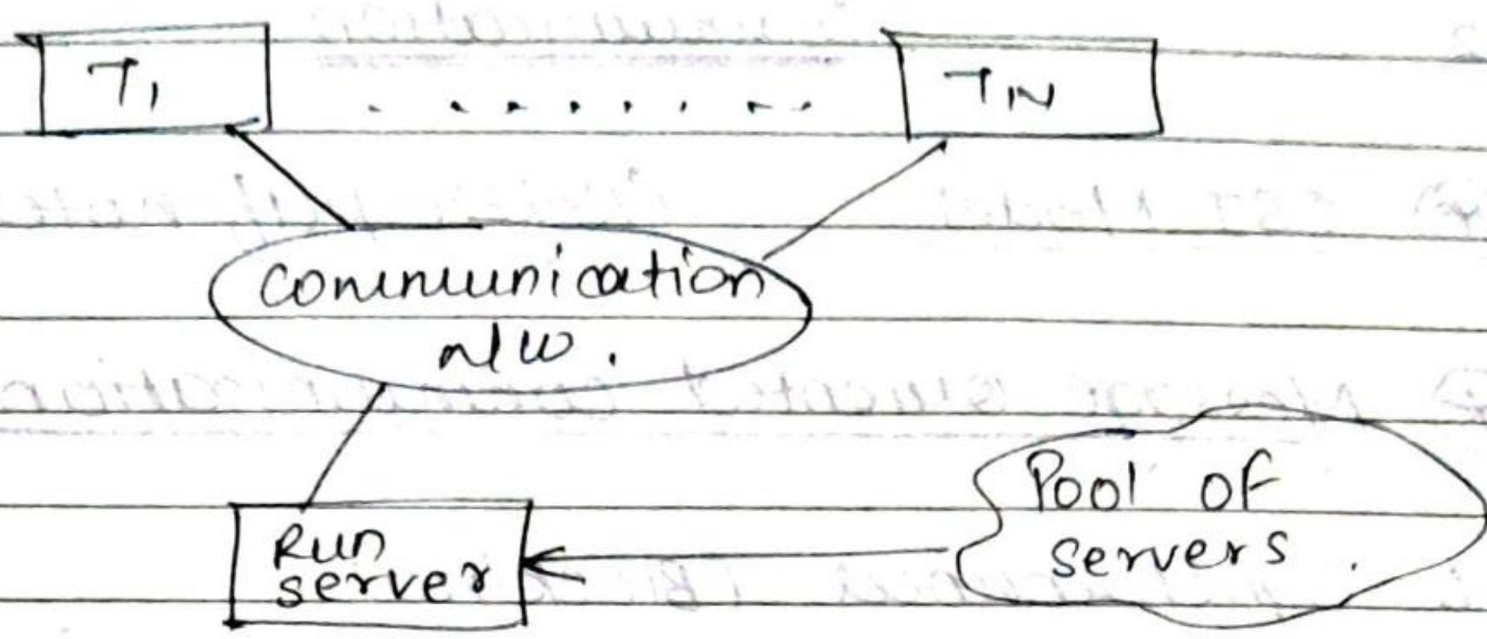
iii) Loose preemption // Best soln.
{suspend P1 }.

3) Workstation server

This is the traditional client server architecture which is the architecture of modern era. It supports information, resource sharing, migration, etc. The only issue is server will snoop and multiple server may start execution.



4) Processor pool
It contains set of terminals and pool of servers managed by run server. Any request coming from terminal will go to run server and run server from pool will allocate the required server.



Issue : i) Single point for failure w.r.t run server.
ii) Even small task would be executed at server's end, due to use of terminals

5) Hybrid.

It is a combination of processor pool and workstation server.