Unit-5- Distributed System.

peer-to-peer Network.

→ peer-to-peer (P2P) Network is a Special N/w created with interconnected nodes("peers")

→ Each System can directly communicate with other Systems without the involvement of centralised Administrative System like Server.

→ This type of N/w is used to Share data, resources among each other.

→ All the tasks are equally divided b/w all nodes.

→ The nodes interact with each other to Share data & resources.

→ With this Architecture, there is the ability to provide large Combined storage, CPU power and other resources with low cost and high Scalability.

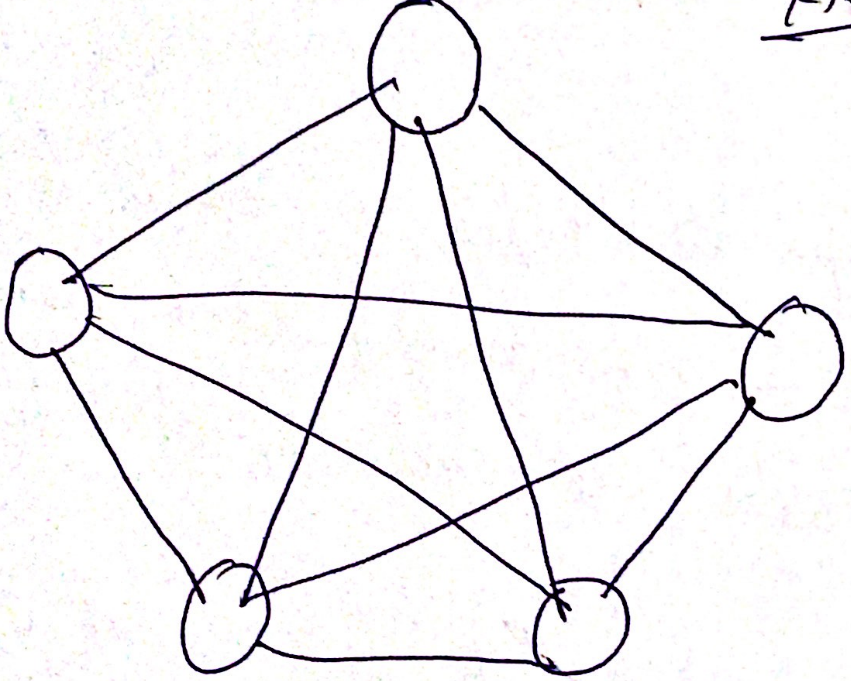
→ Each System in this N/w Contains its own

Security mechanisms.

Peer-to-peer (P2P) N/w

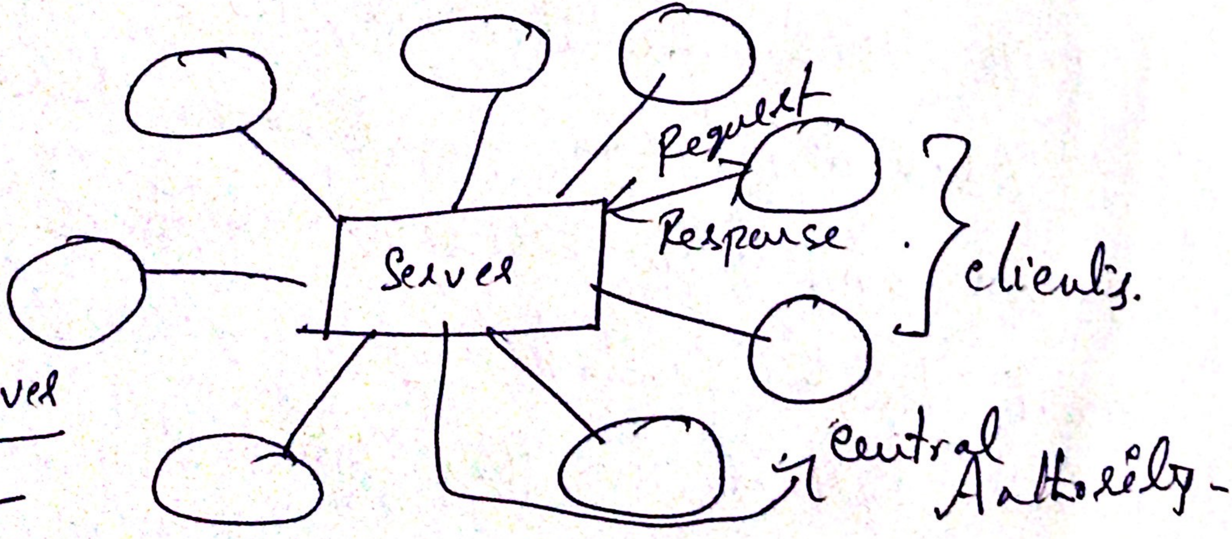
- The nodes in peer-to-peer N/w uses resources and provide resources to other nodes. Then resource sharing capacity of the peer-to-peer N/w increases.

-> This is different from client-server Network.



P2P Network without Central Authority.

-> ~~Each~~



client/Server Model

-) each Computer in the peer-to-peer N/w manages itself.

Disadvantages -

① It is difficult to backup the data, as because it is stored in different Computers and there is no central Server.

2) It is difficult to provide overall Security in P2P Network.

2 types of P2P Networks are existed.

They are (a) Structured P2P

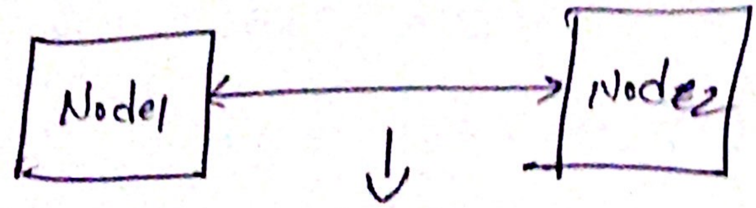
(b) Un Structured P2P.

Structured P2P:

-) The Construction of this structured N/w is based up on a Specific structure/ Topology.

-) The Resources are distributed remotely and indexed Using hash tables.

-) The message transfer b/w structured P2P N/w is Unicast transmission



Unicasting.

→ It is very easy to locate particular system in this P2P N/w.

→ Overheads are very less.

→ Moderate failure rates.

→ This type of N/w is Suitable for Large Scale applications.

Unstructured P2P Layout N/w.

→ The construction of this N/w has no specific structure.

→ The resources are indexed locally.

→ Overloads are very high

→ Supports high failure rates.

→ This is Suitable for Small Scale applications.

→ The messages can be broadcasting.

Napster. Napster is a Musicfile sharing Computer Service created by American College student Shawn Fanning in 1999.

-> Electronic copies of music can be shared with this.

-> With this thousands or more users can share their music with others.

-> up on downloading Napster and after registering in it, the users can share their music with others.

-> With this a largest music Library can be created

-> Napster Contains a central Server whic inturn Connected to many cluster of ~~Servicers~~ Servers.

-> The central Server Contains the following information of registered Users. It is

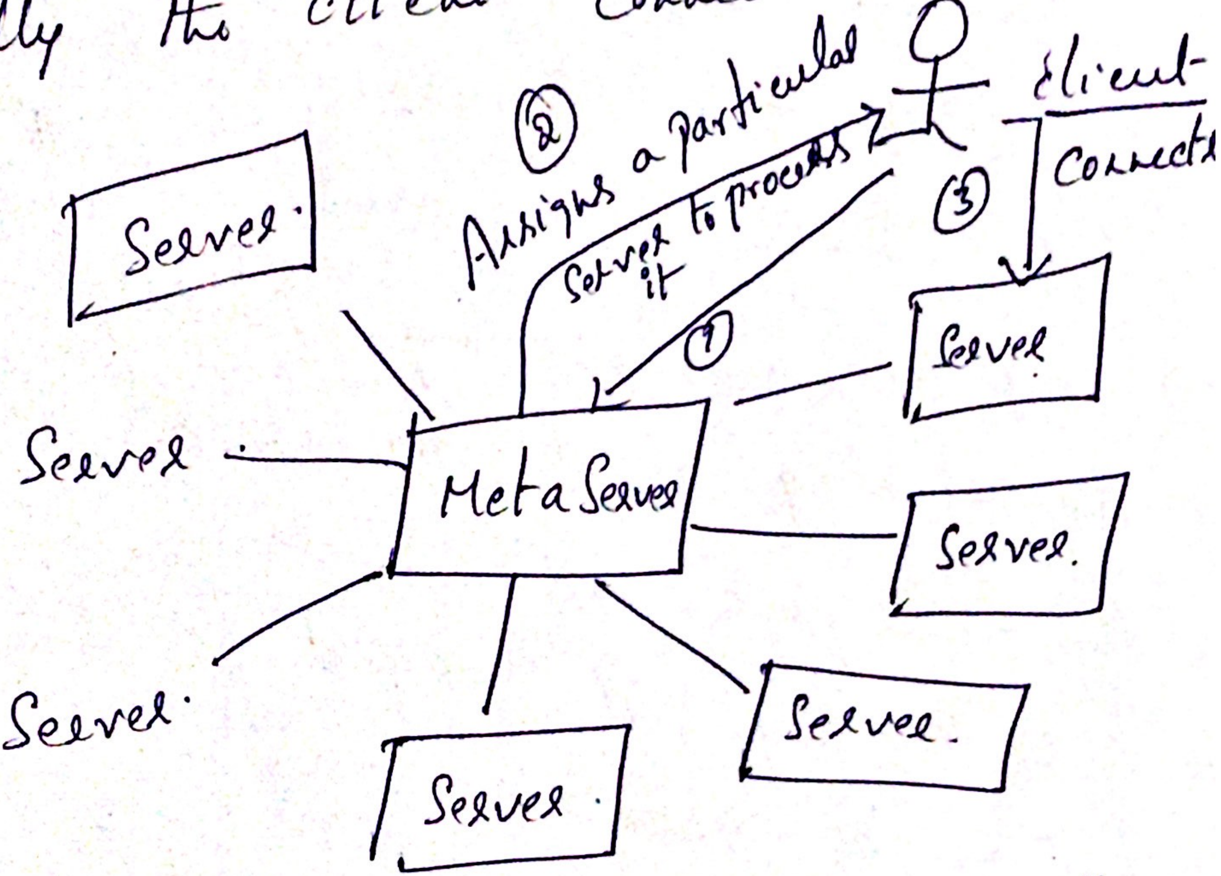
-> User's (clients') IP Address, port,

band widlt

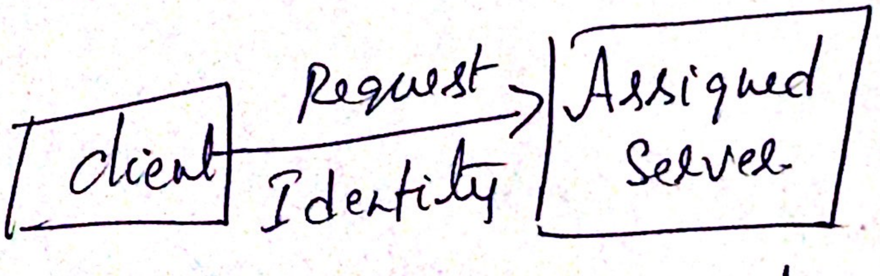
b) Information about the files that the users' want to share.

The ₵ required Content, or Songs Can be down loaded with the following.

a) Initially the client Connects to the Meta Server



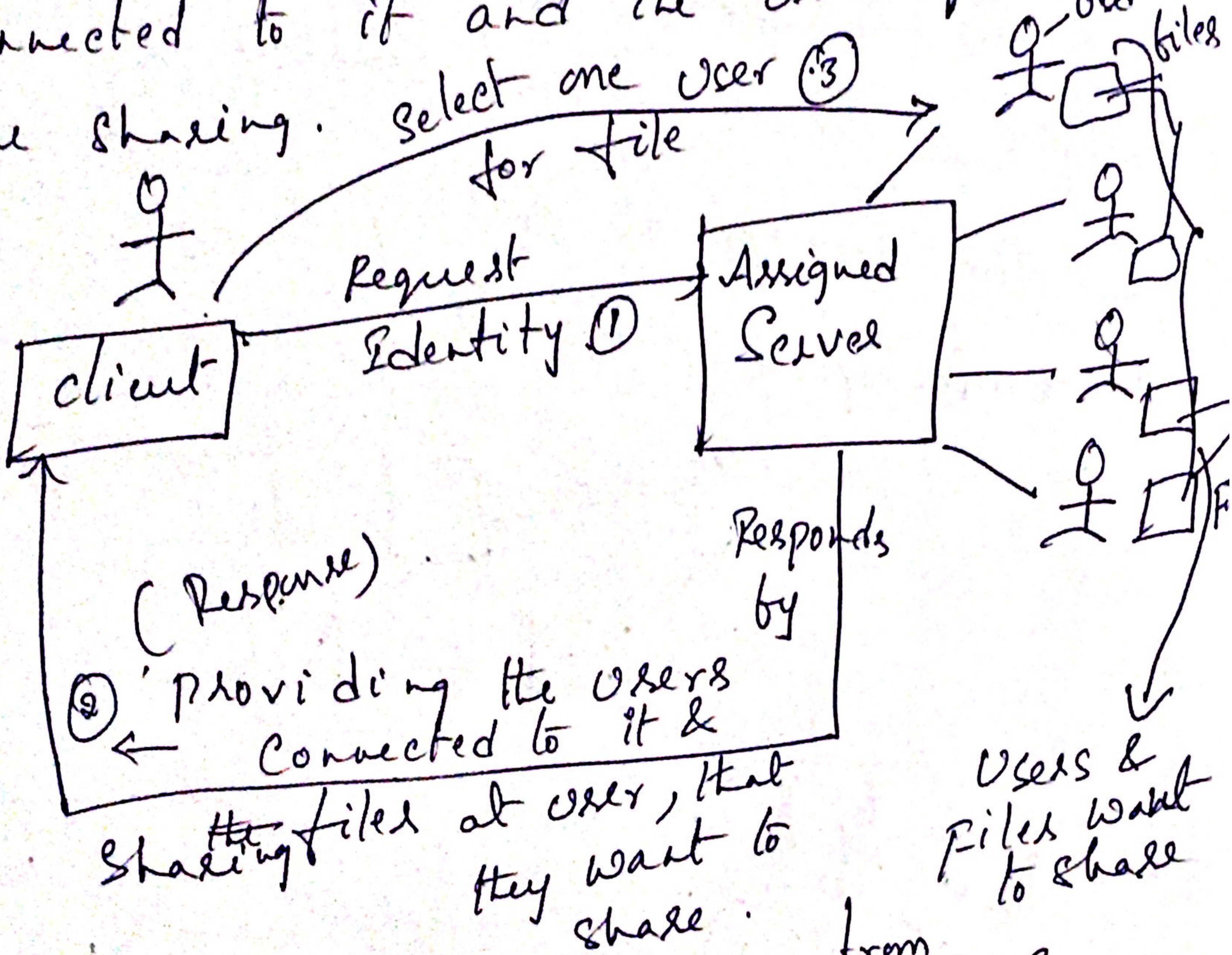
b) Then the meta Server assigns another Server to process the request.



c) The client then Connects to the assigned Server and Sends its all avail

request along with its identity.

-) Server then responds to the client and provides the information about the users Connected to it and the ~~users~~ files they are sharing.



-) up on receiving the response from the Server, the client chooses one of the Users and down load the required file.

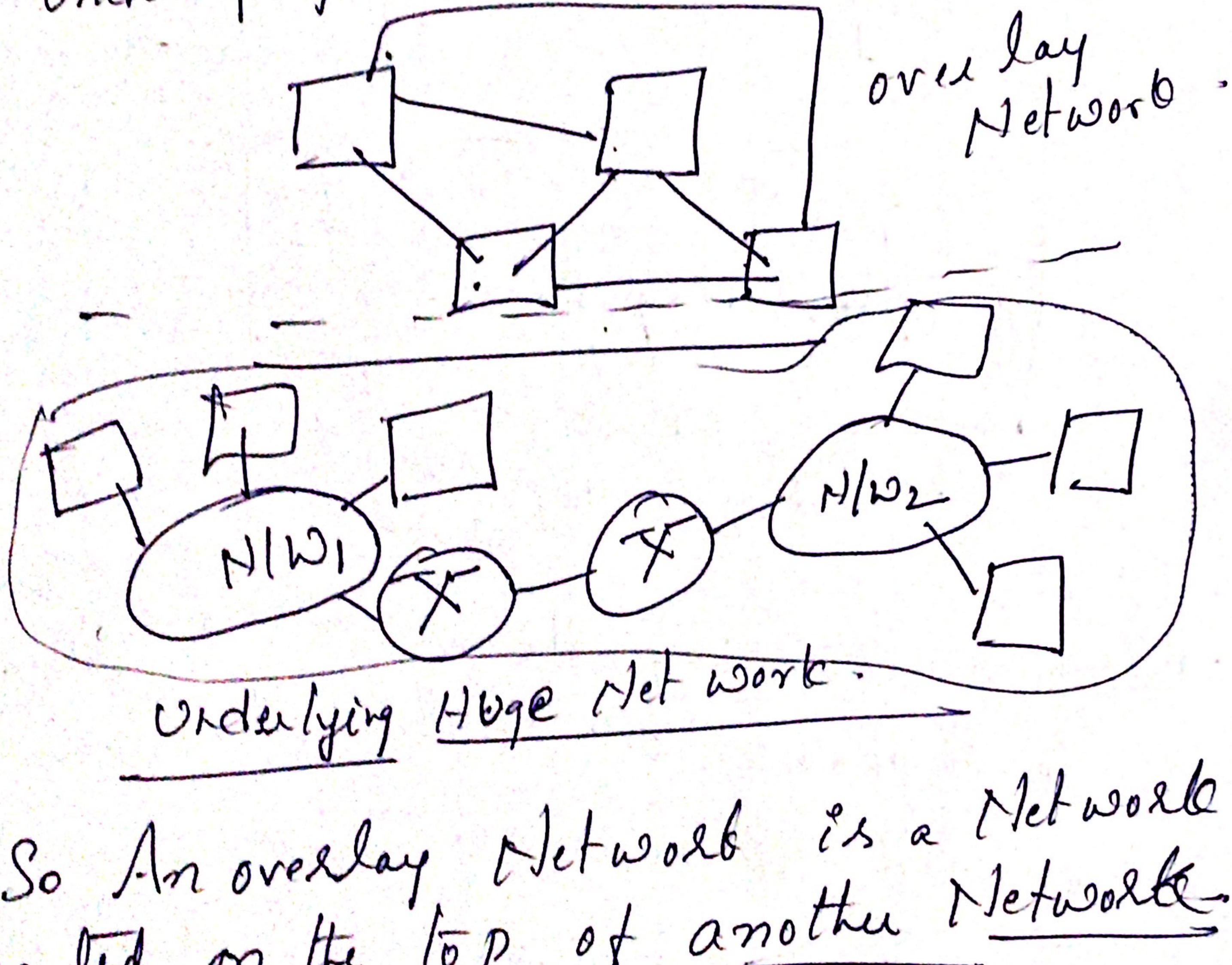
-) Like this every user can share files as well, they can also down load required file.

Overlay Network.

overlay Network is Constructed over another Network.

for Example Connecting Internet over telephone lines is an overlay Network.

-> The Topology of overlay Network independent of underlying Network.



So An overlay Network is a Network created on the top of another Network.

-> In P2P overlay N/w Application-level overlays are used. This N/w is used to share resources which are available across Network.

-> While Searching for particular ~~res~~ resource from other nodes, the concept of Graphs are used.

2 types of ~~lay~~ overlay Networks are there. They are

a) Structured overlays &

b) Unstructured overlays.

structured overlays.

->In this Architecture the Network is constructed with a specific structure.

-> In this Architectures, nodes cooperatively maintain Routing information.

-> Node detection & location by mapping it is very fast

-> Insertions & deletions in this type is a little over head.

-> This type of Network typically uses hash table interface for mapping.

Unstructured overlays-

- → This Architecture donot have a particular structure
- → Indexing used in this type is Local to Indexing.
- → Insertion of & Deletion of nodes is very easy.
- → Searching for a particular file is a overhead task and requires huge time.

Data Indexing & Overlays.

The data in a P2P Network is identified by using Indexing.

Indexing mechanism are classified as

a) Centralized Index.

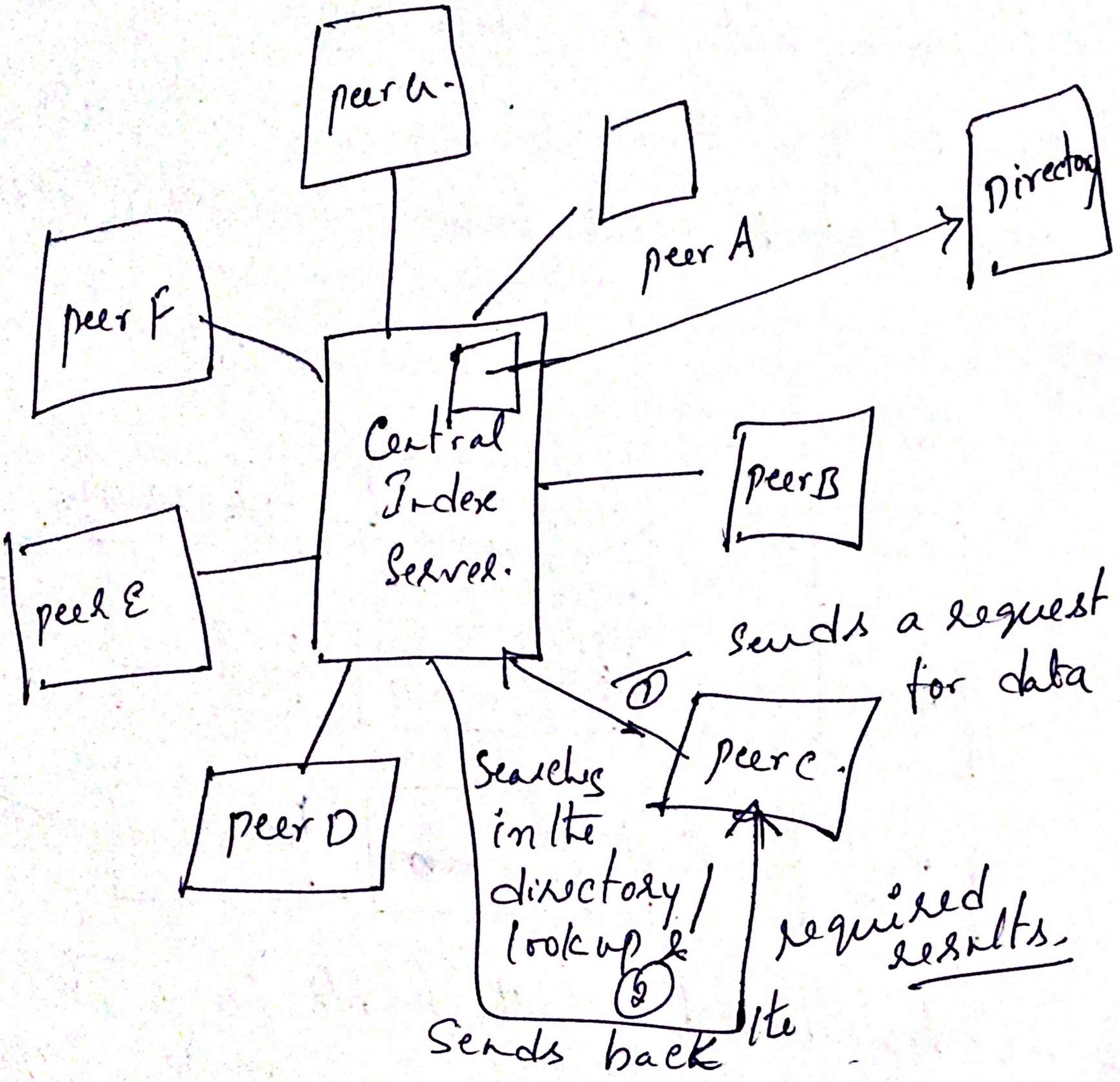
b) Local Index.

c) Distributed Index.

Centralized Index. In this Index, one or more central Servers are used to store the references to the data-

- → whenever there is a need for data, the particular peer sends a message to the

Central Server.



→ Then the central Server for Searches in the directory / lookup for the required resources and Sends back the required results.

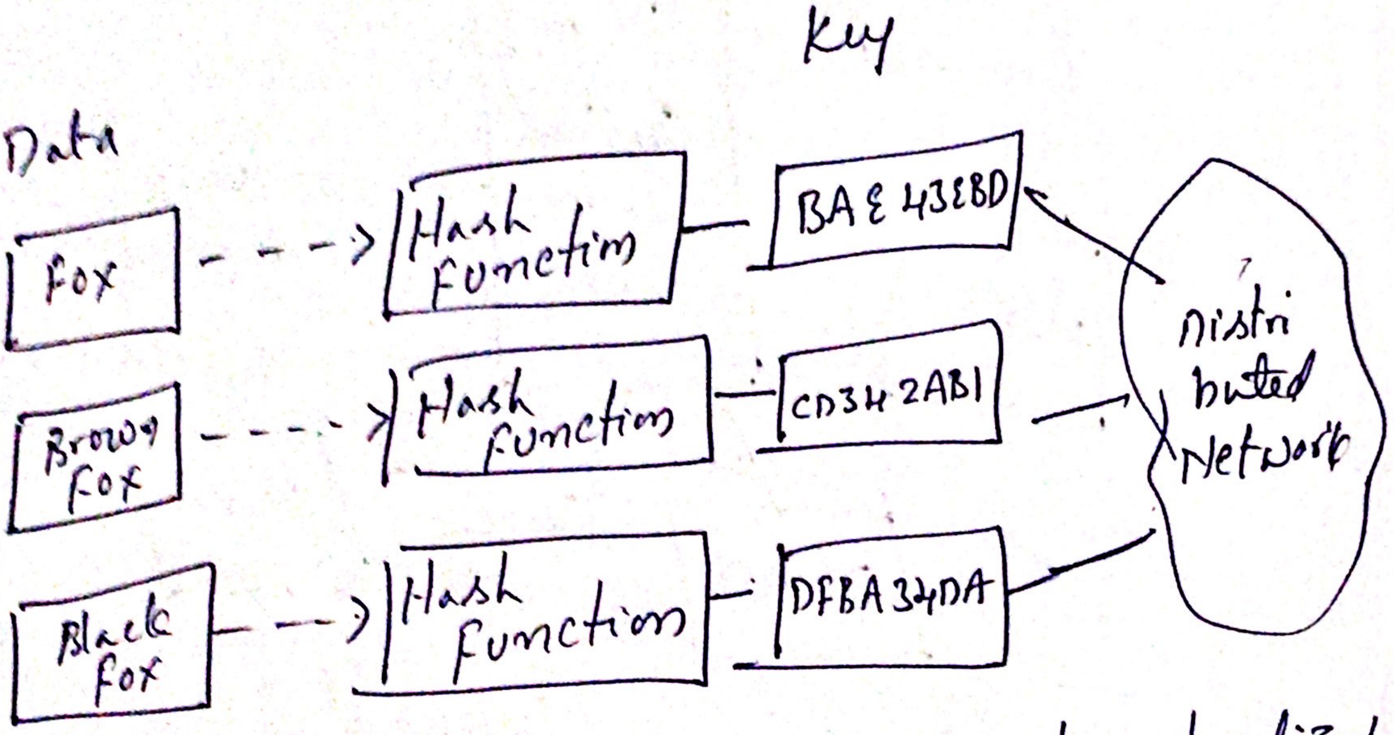
→ once the requested peer obtained the list it establishes a direct connection to the peers

Distributed Index :—

There is no central Authority in this type of Index.

→ The Indices at various objects are scattered, through out the Network.

→ In this type a Special Structure called Distributed Hash table is used (DHT)



→ A distributed Hash table is a decentralized storage system which contains the keys generated through Hash function

→ with these key values the nodes are connected to the required nodes.

→ These are generally fault tolerant because data is replicated across multiple nodes.

→ The node address is mapped to a logical identifier in the key space, the data is retrieved through this key.

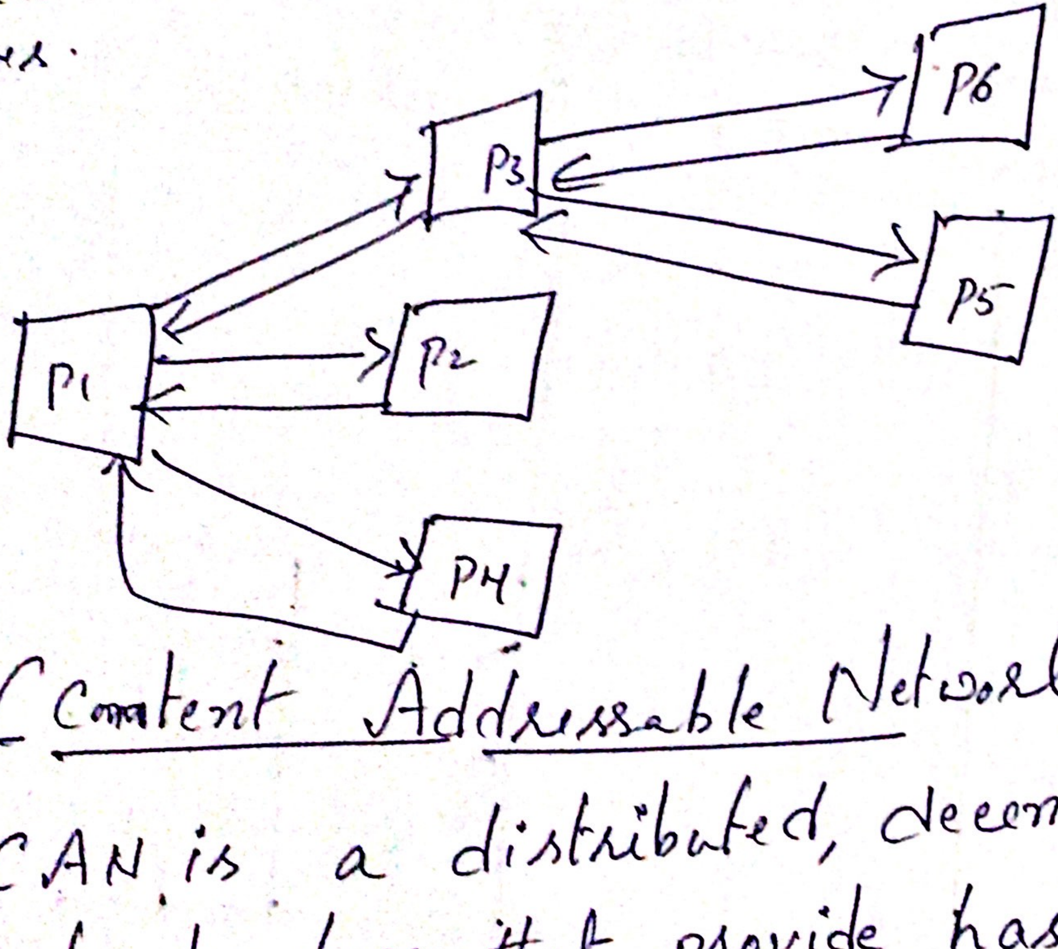
Local Indexing:

This type of Indexing is typically used in Unstructured overlays.

Here objects are searched with the local Index.

-> Each peer maintains a local Index for data Searching.

-> Nodes are Connected through neighbouring nodes.



CAN (Content Addressable Network)

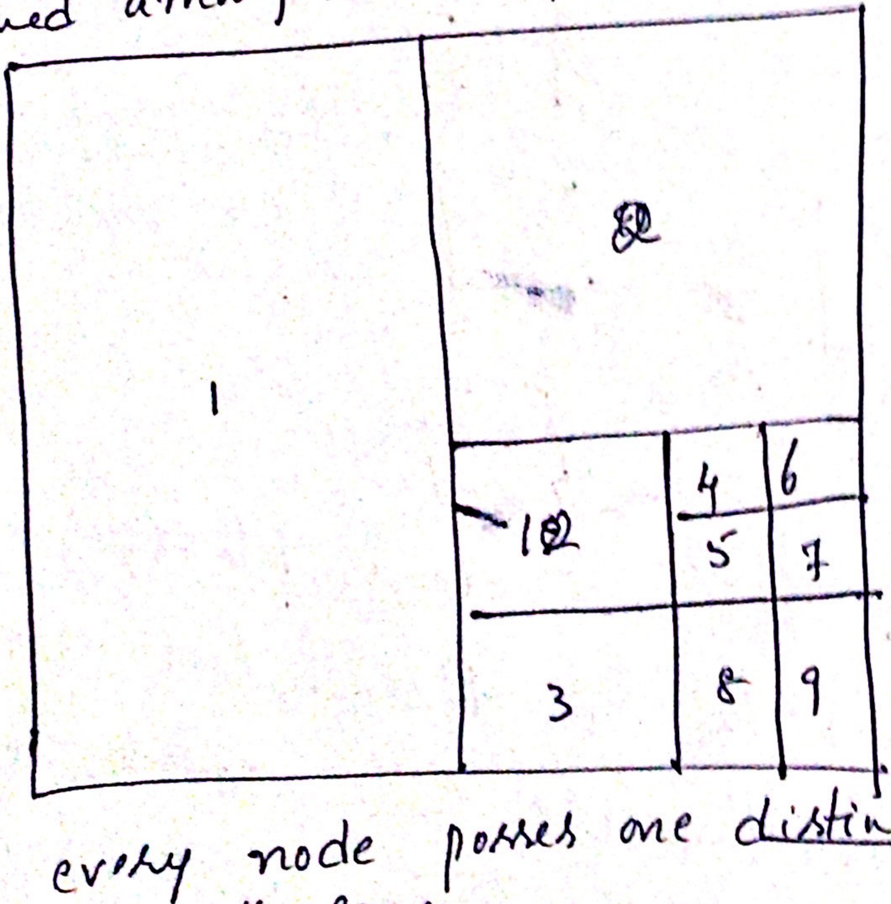
CAN is a distributed, decentralized P2P Infrastructure, that provide hash table functionality. So it is treated as an Indexing mechanism used to map the objects to their locations in the Network.

~~Not only index~~

It is also useful as Large-Scale-storage management System and wide Area Name resolution Services

-> This CAN Consists of n-dimensional Coordinate Space with with virtual logical address

-> points within the space are identified with coordinates. The entire space is dynamically partitioned among all the nodes in the system.



So that every node posses one distinct zone within the overall space

-> It is a Self organising, fault tolerant, scalable Architectural design implemented at the application layer.

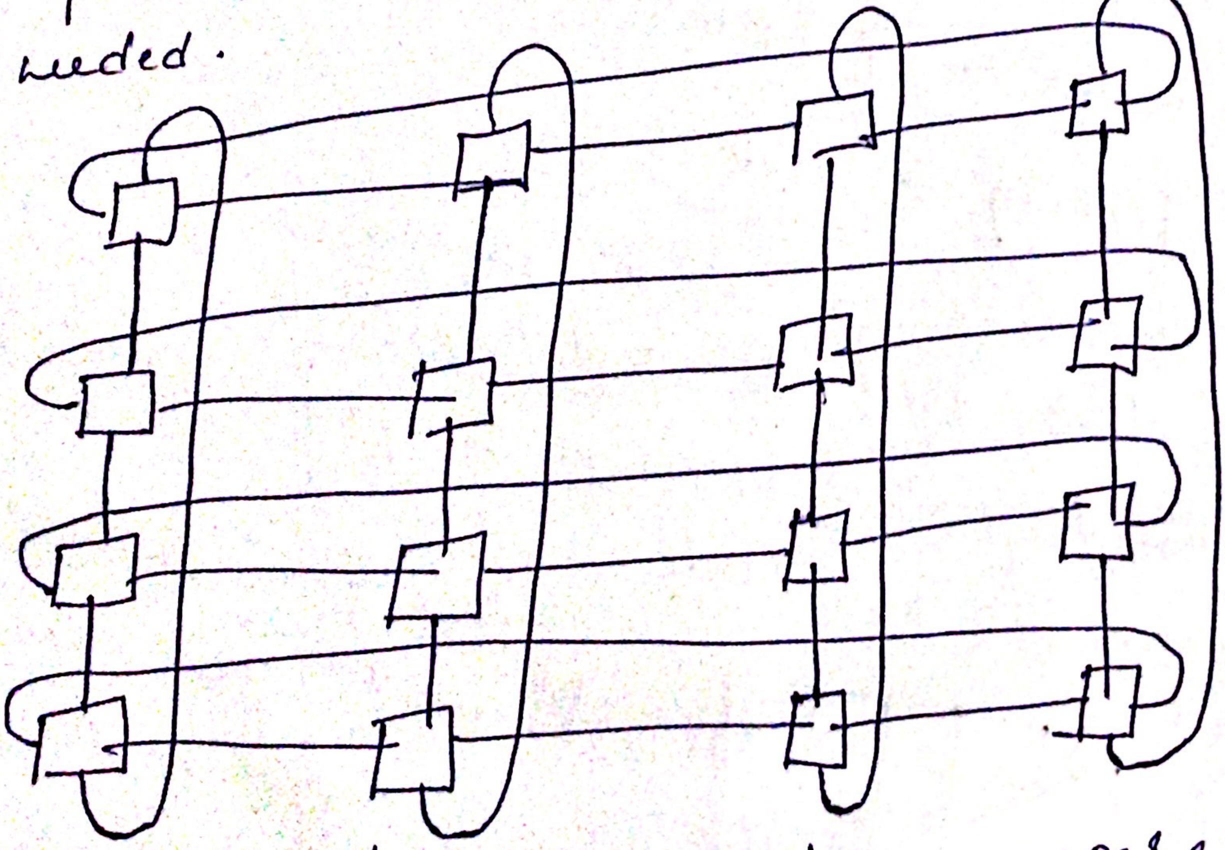
Hence, the 3 core components of a CAN design are

a) The CAN virtual coordinate Space :- It is a partition, Space among the nodes.

b) Routing: To locate a particular node, routing is required.

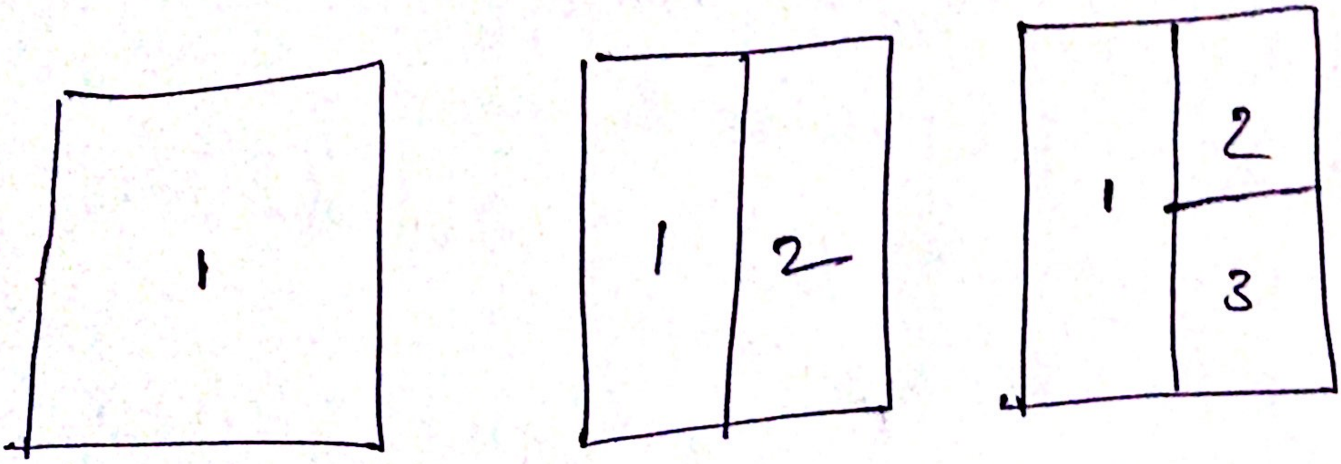
The node in the CAN contains a Routing table with Ip Address of the node & coordinate zone of its neighbours.

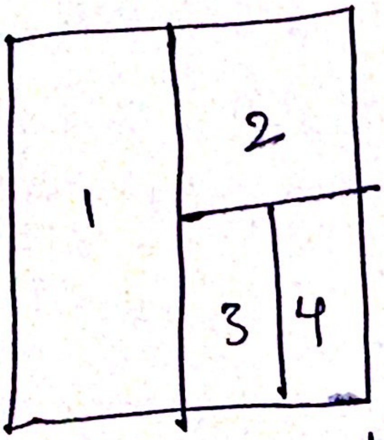
c) A Special environment to maintain the CAN. whenever a node is Joining as well as departing from the zone a mechanism is needed.



d) CAN performs 3 basic operations.

CAN Examp.





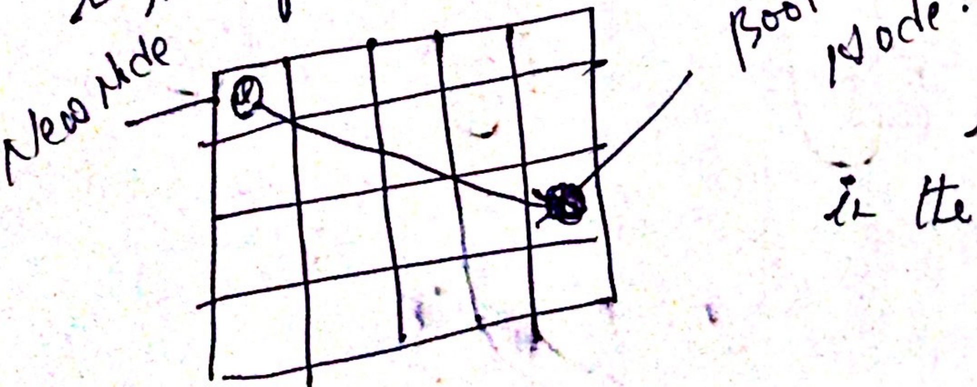
⊕ - Initially we have only one node. The entire space is occupied by that node.

-> If we have 2 nodes it is divided into 2 partitions/ ~~to~~ Like wise 3 nodes & 4 nodes.

-> Joining a node.

when ever a node wants to join the network it check with the Bootstrap Node.

Bootstrap Node identifies the space and splits the region for new Node.



A Boot strap node contains the list of participants in the CAN N/w.

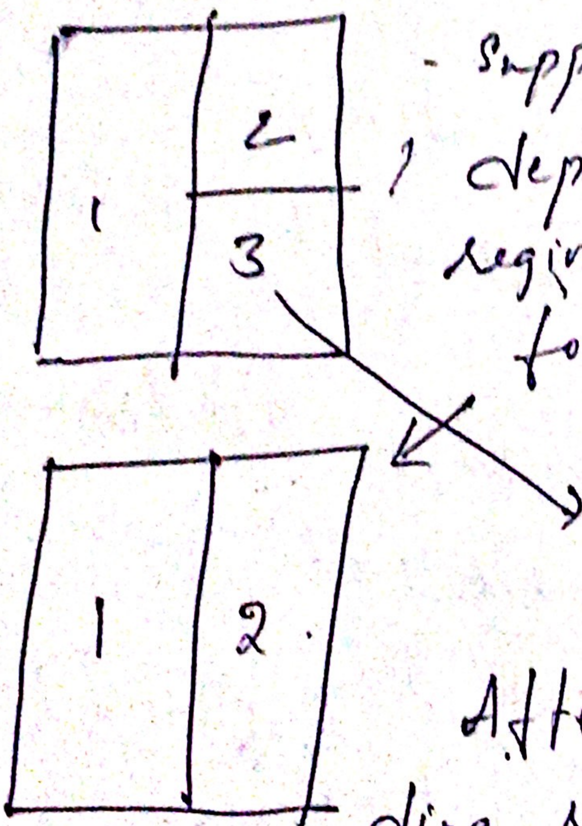
Routing in CAN.

Searching a particular node can be done with Routing technique using Hash table.

- Departure from the CAN.

When ever a node wants to leave CAN the ~~zone is~~ corresponding zone of the node is merged with its neighbouring node.

- Suppose if node 3 wants to depart from CAN, the region region is merged like as follows.



The node want to Depart from CAN.

After merging, the corresponding routing tables need to update across the Network.

Tapestry.

Tapestry is a P2P overlay which provides a distributed Hash table for searching a particular element in the N/w in Distributed applications.

→ It is also used to visualise resources with decentralized approach.

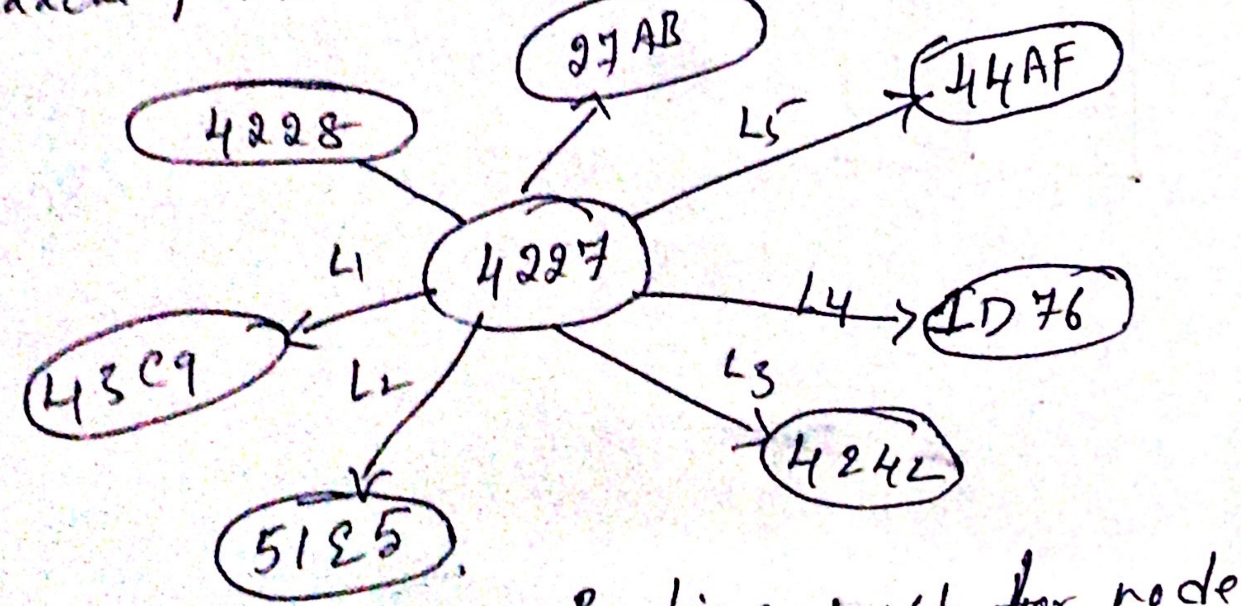
→ The communication is through Broadcasting Multicasting approach.

→ It reduces the message Latency

→ Each object here is searched by its name, not by its location

→ Each object has an Unique ID/Name.

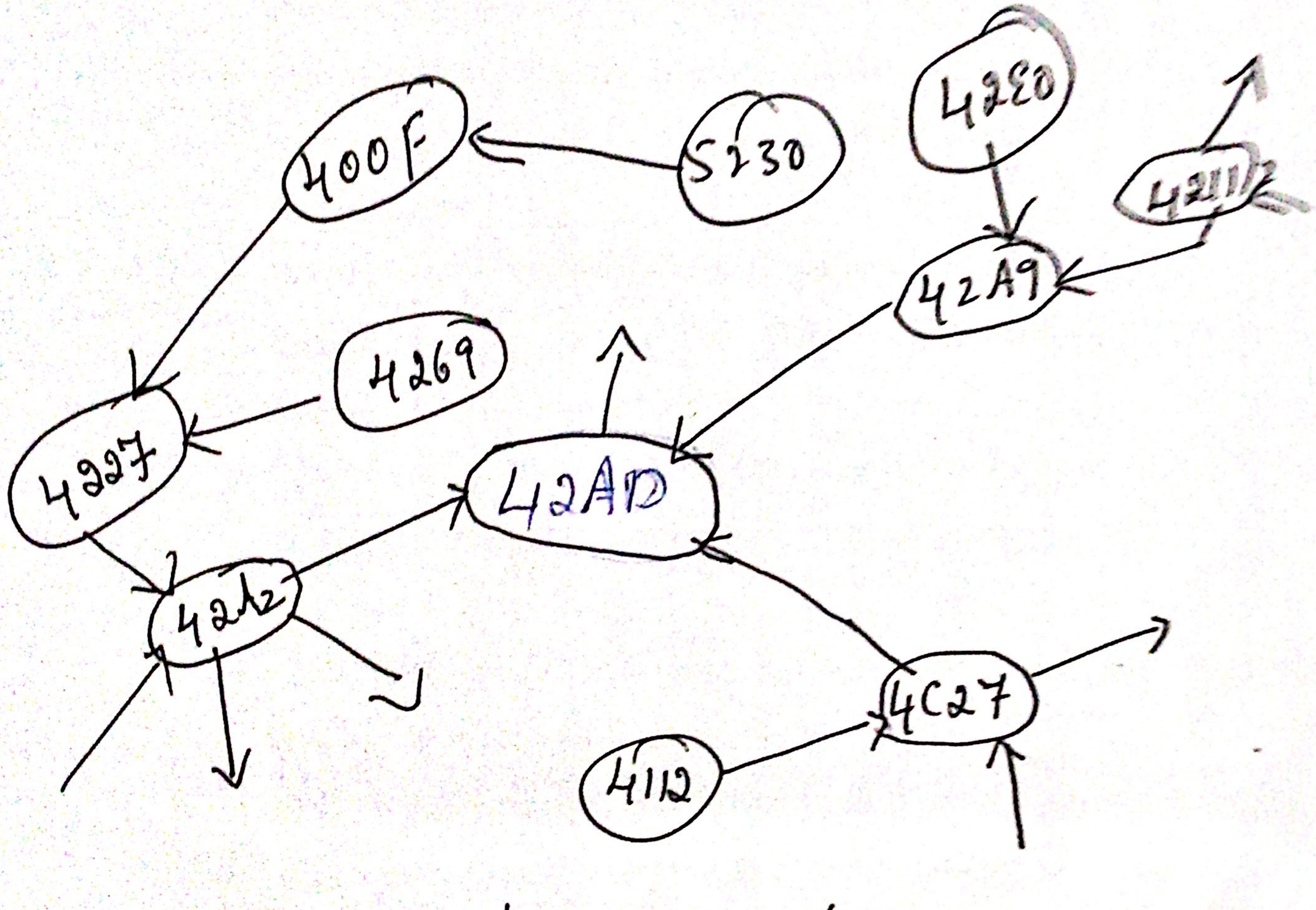
→ Searching is through root Node.



Routing mesh for node 4227.

→ Routing in Tapestry can be done with Route root Node.

→ Each node is mapped with rootnode, then that is redirected appropriately towards destination.



Routing Sxeample.

Distributed shared memory

12.1 Abstraction and advantages

- It appears as if a single memory is there for distributed applications.
- DSM is a mechanism that manages memory across multiple nodes and makes inter-process communications transparent to end-users.
- The distributed shared memory (DSM) implements the shared memory model in distributed systems but it doesn't have physical shared memory.
- All the nodes share the virtual address space provided by the shared memory model. The Data moves between the main memories of different nodes.
- Programmers can access the data across the network using only *read* and *write* primitives from that memory Like in a uniprocessor system.
- Programmers do not have to deal with *send* and *receive* communication primitives.
- DSM is a mechanism of allowing user processes to access shared data without using inter-process communications.
- The DSM abstraction is illustrated in the following Figure. A part of each computer's memory is allotted for shared space, and the remainder is private memory. To provide programmers with the illusion of a single shared address space, a memory mapping

management layer is required to manage the *shared virtual memory* space.

DSM has the following advantages:

1. Communication across the network is achieved by the read/write abstraction that simplifies the task of programmers.

2. A single address space is provided, thereby providing the possibility of avoiding data movement across multiple address spaces, and simplifying *passing-by-reference* and passing complex data structures containing pointers.

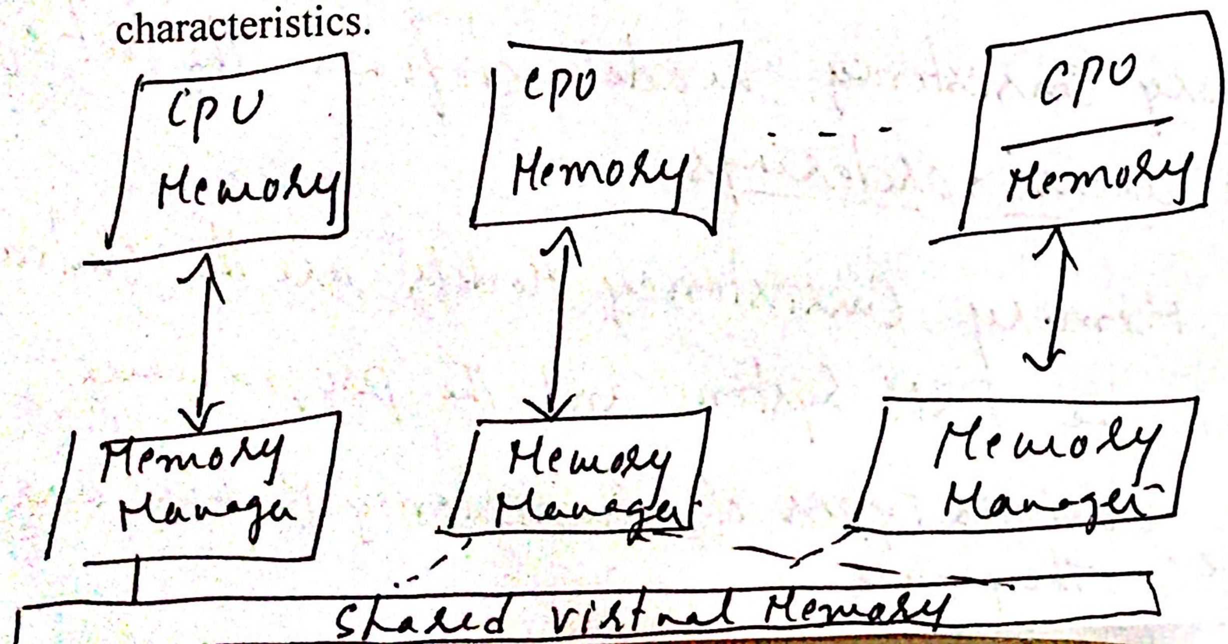
3. If a block of data needs to be moved, the system can exploit locality of reference to reduce the communication overhead.

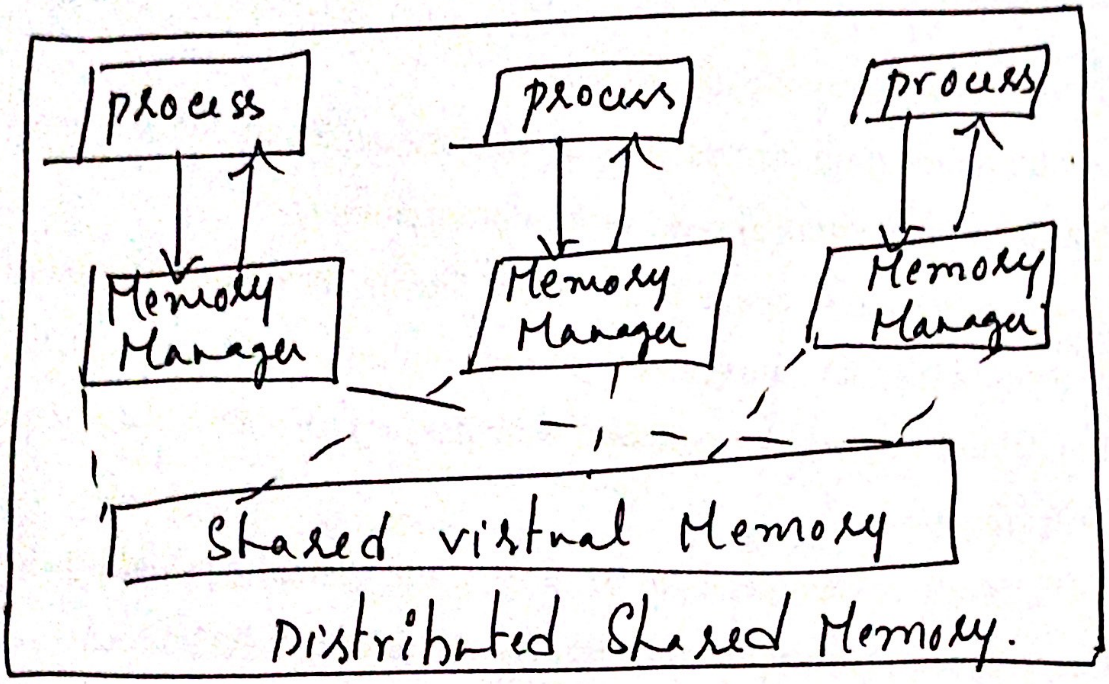
4. DSM is often cheaper than using dedicated multiprocessor systems, because it uses simpler software interfaces and off-the-shelf hardware.

5. There is no bottleneck presented by a single memory access bus.

6. DSM effectively provides a large (virtual) main memory.

7. DSM provides portability of programs written using DSM. This portability arises due to a common DSM programming interface, which is independent of the operating system and other low-level system characteristics.





Distributed Shared Memory.

Memory Consistency Models:—

Memory Consistency models provide the Memory Coherance. Memory Coherance is the ability of the System to execute memory operations Correctly.

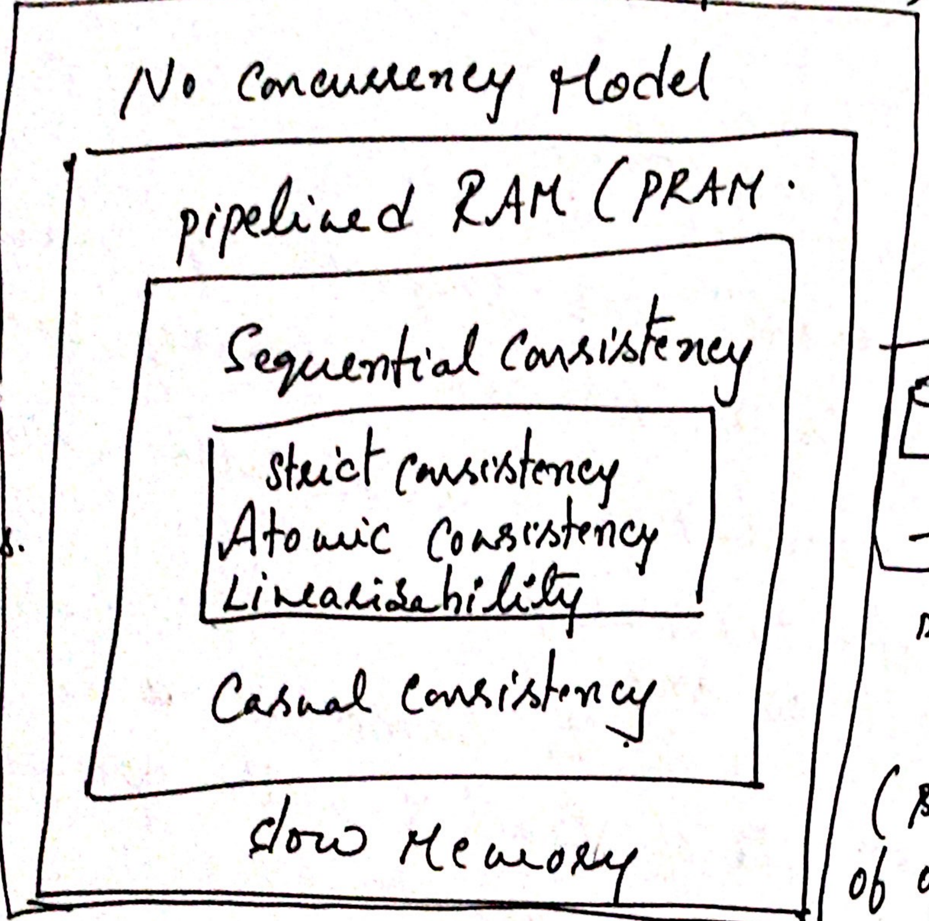
Memory Consistency model defines the set of Memory access orderings.

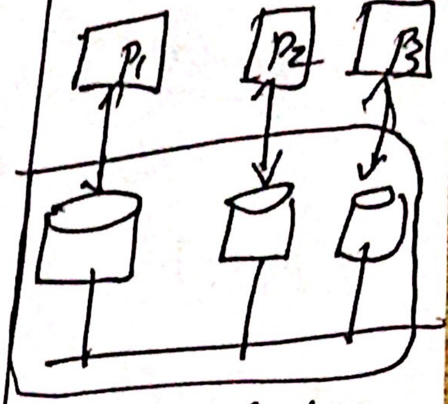
So Memory Consistency Models are the Contract between the DSM System and the programmer.

There various types of Consistency Models are There. They are

-> Consistency models mainly specify the ordering of operation on to the shared data store (which is distributed across multiple processors)

Basically consistency models helps how the data can be used by processors.





Distributed Data store.

(basically 2 types of operation are performed on Distributed data store)

① strict Consistency:

This model is also known as Atomic Consistency Model.

This is the strongest consistency model. Under this model a 'write' to a variable by any processor needs to be seen instantaneously by all processors.

This is the most rigid model. This is the deterministic Model, i.e the programmer's expected will be received every time

-> The execution order of programs between processors must be the order in which, they were issued.

Sequential Consistency.

This model was proposed by Lamport (1979). It is a weaker memory model.

-> A 'write' operation to a variable can't be seen instantaneously

-> The Result of any execution is the same as in the same order ~~that~~ the of Read & write operations performed by all operations.

-> This is Similar to Serializability.

Ex:- Assume 3 processors P1, P2, P3.

process p1	process p2	process p3.
x=1	y=1	z=1
print(y,z)	print(x,z)	print(x,y)

Initially all variables are set to 0.

-> write operation is happening initially i.e x=1, y=1 & z=1.

-> Later Read operation is happening print (y,z), print(x,z) & print(x,y)

Three processors are interlinked - They are executing Concurrently

-> The 6 statements taken can be ordered in 6! = 720. But there are not valid combinations.

-> only 90 possible valid executions can be done as per the Rules of Sequential - consistency.

Casual consistency:-

This Consistency defines that the write operation, which are potentially casually related must be seen by all the processors in the same order.

Example:
Interaction through a distributed shared database.

-> process p1 writes data item 'X'

-> then p2 reads 'X' and writes 'Y'

-> Reading of 'X' and writing of 'Y' are potentially casually related, becaurse the computation of 'y' may have dependent on the value of 'x' as read by 'P2' (i.e The value written by P1) So they are Casually related.

Ex2

P1 — W(x)a . W(x)c
P2 — R(x) a W(x)bg
P3 — R(x) a R(x)c R(x)b
P4 — R(x)a R(x)b R(x)c.

-> The writes of $W_2(x)b$ and $W_1(x)c$ are concurrent. So it is not required that all processes see them in the Same order.

FIFO Consistency.

Writes done by a Single process are seen by seen by all other processes in the order in which they were issued.

-> writes from different processes may be seen in a different order by different processes.

This is also called 'PRAM Consistency' (or) pipelined RAM.

-> Easy to implement. There are no guarantees about the order in which different processes sees writes - Except that two or more writes from a Single process must be seen in order.

Example.

P1 - W(x)a

P2 - R(x)a W(x)b W(x)c

P3 - R(x)b R(x)a R(x)c

P4 - R(x)a R(x)b R(x)c.

A valid sequence of FIFO Consistency events.

Sometimes, Not all applications need to See all writes, let alone Seeing them in the Same order leads to weak consistency.

Slow consistency:-

In slow Consistency, if a process reads a value, which was previously written to a memory location, it can't Subsequently read any earlier value from that location.

→ writes performed by a process are immediately visible to that process. It is a weak model than PRAM & Cache Consistency.

Example:-

First process writes 1 to the memory location X and then it writes 1 to the memory location 'y'

The Second process reads '1' from 'y' and it then reads '0' from x even though 'x' was written before 'Y'

slow Consistency

P1 W(x)1 W(Y)1

P2 R(Y) → 1 R(x) → 0.

Casual Consistency Example

Sequence	P1	P2
1	$W_1(x)3$	
2	.	$W_2(x)5$
3		$R_1(x)3$

W_1 is not Casually related to W_2. R_1 is Sequentially inconsistent but is Casually consistent

(5) Sequence	P1	P2	P3	P4.
1	W(x)1	R(x)1	R(x)1	R(x)1
2		W(x)2		
3	W(x)3		R(x)3	R(x)2
4			R(x)2	R(x)3.