Kalinga Institute of Industrial Technology- Deemed to be University

(Bhubneswar)

Distributed Operating System.
(CS 30009)

School of Computer Engineering

Introduction to distributed systems.

- Distributed Systems:- Putting together Large number of computing systems connected togather using High Speed Network.

- *“A distrubuted system is a collection of independent computers that appear to the users of the system as a single computer”.*

- The user may be assigned a workstation but the hardwares of other systems may be available as required dynamically.

- Overcomes the limitation of Single Compute Systems (Centralized Systems).
- Multiple CPU’s interconnected together. The whole systems looks like a single system for Application Programms.
- Distributed Systems radically needs different software for the operations and management.
- Example :- Banking System, Automatic Assembly Line.

Goals of Distributed Systems.

- Advantages of Distributed System over Centralized System.

  - Economics is the driving force towards Decentralization.

  - Grosch’s Law :- The computing power of a CPU is proportional to the square of the price. Applicable during mainframe era led to purchase of centralized big machines.

  - Not Applicable now.

  - It is effective to harness large number of cheap CPU’s, thus having better price to performance ratio in comparrision to mainframe systems (centralized)

- Higher Reliability over Centralized System. In case 5% of machines are down it will lead to 5% loss in performance only.

- Scalability :- More processors can be added as required rather than purchasing an additional mainframe or replecing a mainframe.

- Advantages of Distributed System over Independent Systems.

Scenario:- *”All the personals are given dedicated PC’s for work in office.”*

- If interconnected than the Sharing of localized data and hardware peripherals is efficient. (Seat Booking Applications by Clerks.)

- Enhanced peron to person communications.

Conclusion:- More Flexibity is achevied through distributed system than isolated system. Independent Systems can be connected using LAN’s.

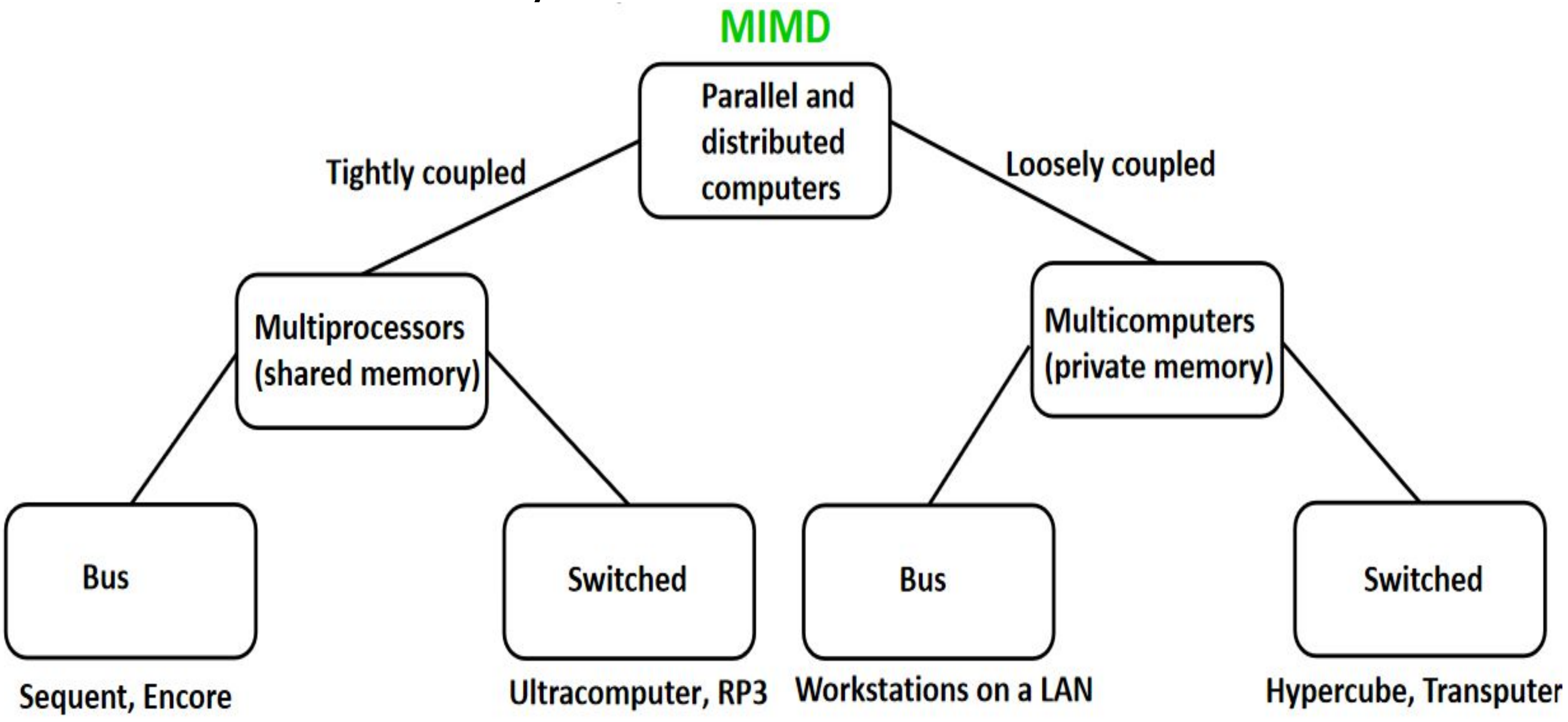
Challenges for Distributed Systems.

- Desingning , implemetation of software systems for managing and operations.
- Communication Systems for Interconnections and recovery of data lost in sharing.

- Security and access of *out of business* data.

#3: Hardware Concepts

- The multiple CPUs in distributed System - How the CPUs are interconnected and how they communicate ?



#3:Hardware Concepts Cont.

- Flynn’s Archtecture (1972) – **Two** essential characteristics :
  - The number of instruction streams
  - The number of data streams
  - SISD : Eg. All traditional uniprocessor computers (i.e., those having **only one CPU**)
  - SIMD : Eg. Array processors (**Super Computers**) – one instruction unit that fetches an instruction, and then commands many data units to carry out in parallel, each with its own data.
  - MISD : No known computers fit this model
  - MIMD : Eg: A group of independent computers, each with its own program counter (PC), program, and data. **All distributed systems** are MIMD.

Difference between Multiprocessors and Multicomputers

Sl. (Parameters)	Multiprocessors	Multicomputers
1. Memory	Single virtual address space that is shared by all CPUs	Every machine has its own private memory
2a. Bus Architecture	There is a single network, backplane, bus, cable, or other medium that connects all the machines	
2b. Switched Architecture	There are individual wires from machine to machine, with many different wiring patterns in use. Eg. Worldwide public telephone system	
3. Coupling	Tightly coupled	Loosely coupled
4. Communication time	Short	Long
5. Data rate	High	Low

#3.1: Bus-Based Multiprocessors

- It consist of k≥2 number of CPUs all connected to a common bus, along with a memory module. A high-speed backplane or motherboard into which CPU and memory cards can be inserted. A typical **bus has 32 or 64 address lines**, **32 or 64 data lines** and **32 or 64 control lines**, all of which operate **in parallel**.

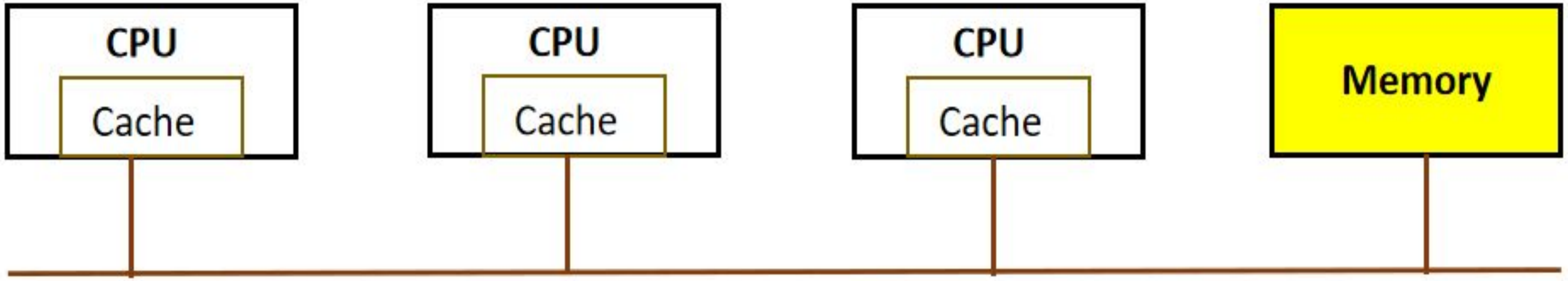


Fig: A bus-based multiprocessor

#3.1: Bus-Based Multiprocessors Cont.

- Characteristics:
  - Coherent memory
  - Bus is overloaded even with 4 or 5 CPUs, and performance will drop drastically
    1. Solu^n: Add a high-speed cache memory between the CPU and the bus.
    2. Cache sizes of 64K to 1M are used in general provides hit rate of 90%
    3. Write-through-cache
    4. Snooping-cache
    5. Snoopy-write-through-cache (combination of #3 and #4)

#3.2: Switched Multiprocessors

- When number of processors > 64, then bus-based multiprocessor is not suitable, in that case “Crossbar switch” or “omega switching” network is used.

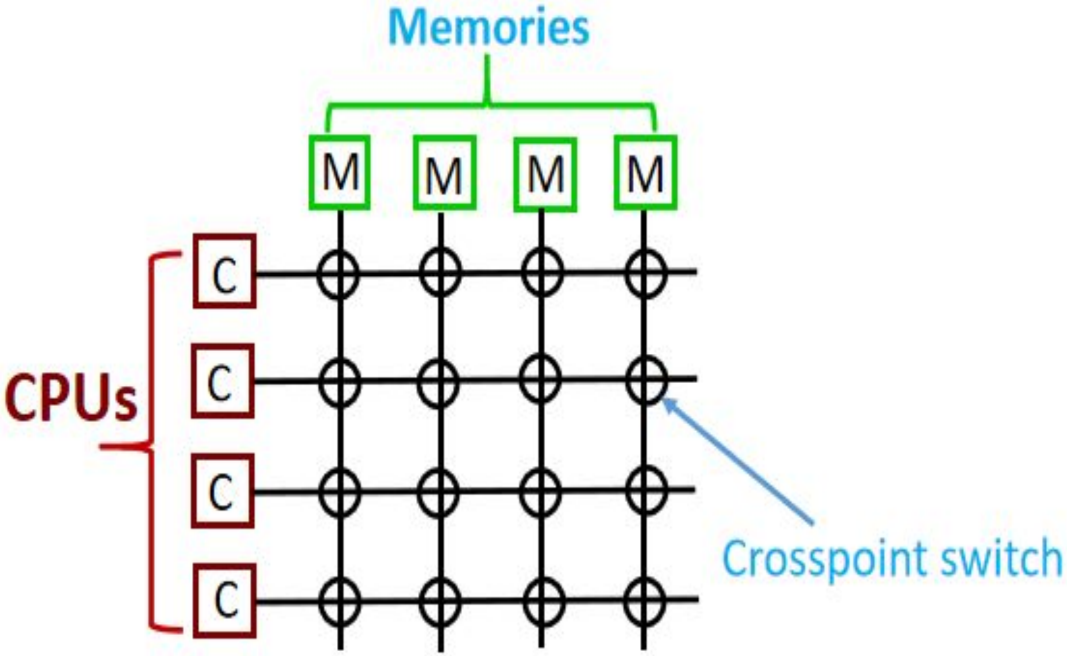


Fig. (a) A Crossbar switch

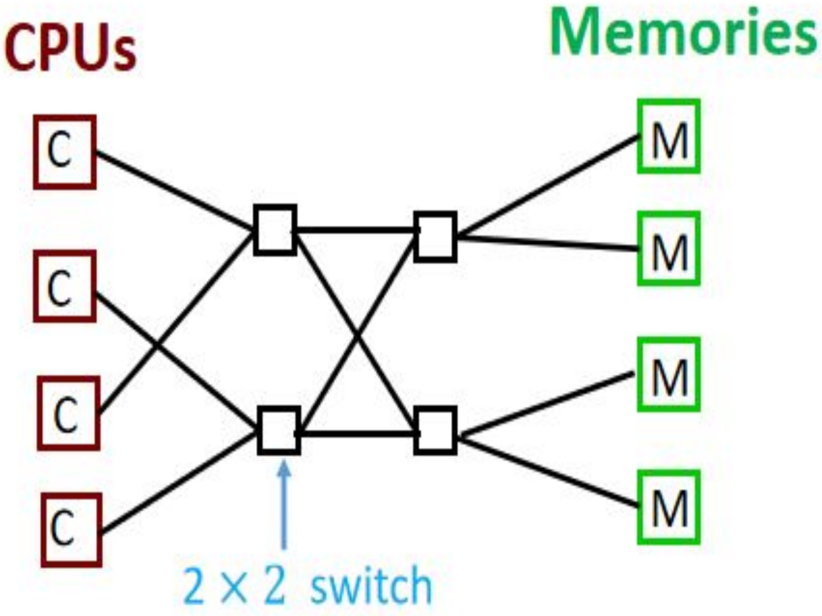


Fig. (b) An **Omega** switching network

#3.2: Switched Multiprocessors cont.

- Characteristics: Memory is divided into modules and connect them to the CPUs with a crossbar switch. (a tiny electronic crosspoint switch).
- A CPU can access a memory after **closing** the crosspoint switch
- If two CPUs try to access the same memory simultaneously, one of them will have to wait.
- Limitation (crossbar switch):
  - **n^2** crosspoint switches are required with n CPUs and n memories. **(Maximum 64 CPUs)**
- Remedy:
  - **Omega Network** (it contains four 2×2 switches, each having two inputs and two outputs)
  - With n CPUs and n memories, the omega network requires **$\log_2 n$** switching stages, each containing **$n/2$** switches, for a total of **$(n \log_2 n)/2$** switches.

#3.3: Bus-Based Multicomputers

- In bus-based multicomputer system it is required to communicate CPU-to-CPU. This requires high-speed backplane bus.
- It is a collection of workstations on a LAN, where each system has its own local memory. No shared memory.
- Speed is 10-100 Mbps

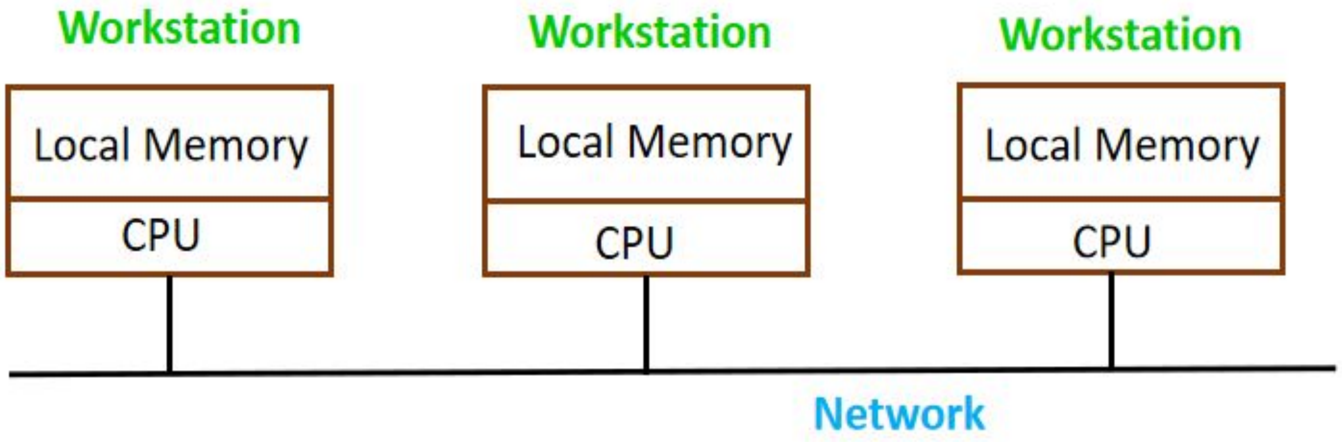


Fig. A multicomputer consisting of workstations on a LAN

#3.4: Switched Multicomputers

- Each CPU has direct and exclusive access to its own, private memory.
- Two popular topologies are:
  - Grid based switched multicomputer
  - Hypercube based switched multicomputer

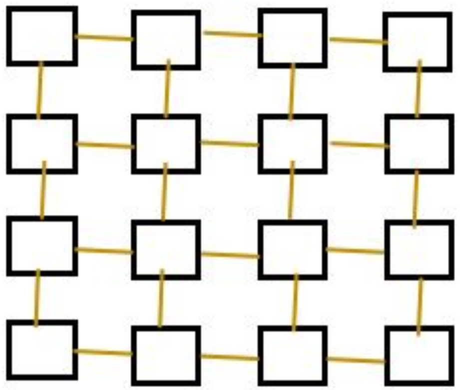


Fig: A Grid-based switched multicomputers

4D-Hypercube

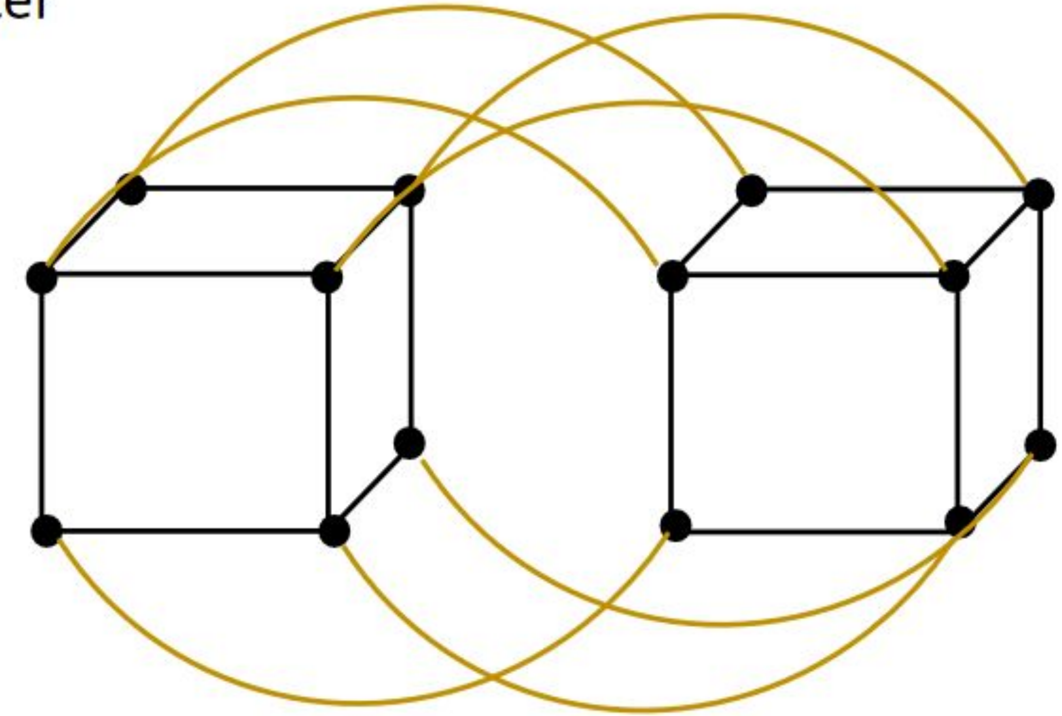


Fig: A Hypercube-based switched multicomputers

#4: Software Concepts

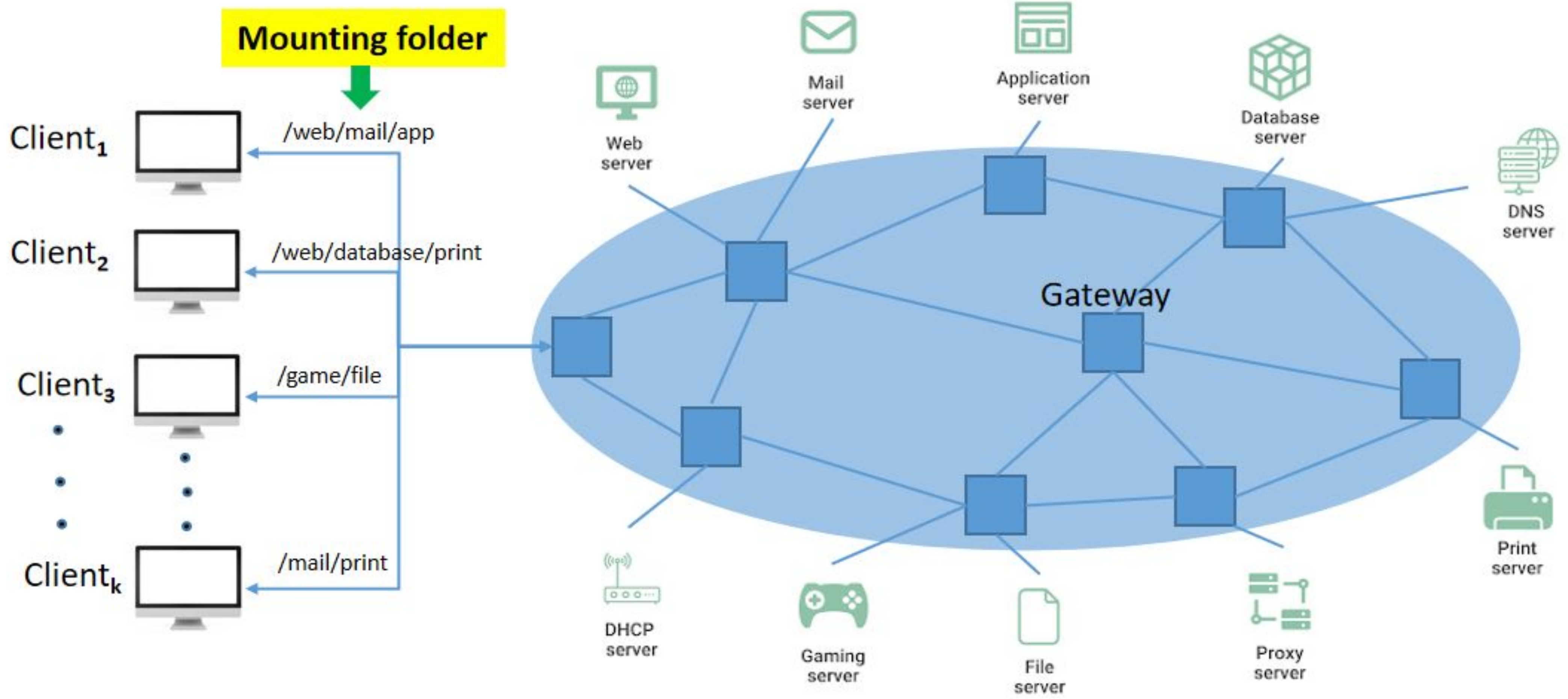
- Two types of OSs for multiple CPU systems
  - Loosely-coupled:
    1. A group of PCs, each has its own CPU, its own memory, Its own HDD, its own OS, but shares some resources, such as printers, databases, over a LAN/MAN/WAN.
    2. i.e. All the PCs are independent of one another.
    3. All the PCs are connected via some **network**
  - Tightly-coupled:
    1. All CPUs are at one place
    2. The microprocessors execute in **parallel**

#5: Networked Operating Systems

- A network of workstations, must have their own OS, may have own HDD, connected by a LAN and execute all commands locally.
- Provision of one or more file sever(s) across the network
- Sometimes a user may log into another workstation remotely by using remote login command :
  - $ rlogin IP_Address_of_Machine
- It also allows copy of any file from one machine to another machine:
  - $ **rcp machine_source/file1 machine_destination/file2**
- The **operating system** that is used in this kind of environment must manage the individual workstations and file servers and take care of the communication between them.

#5: Networked Operating Systems Cont.

- Accessing different servers by different clients through mounting



#6: True Distributed Systems: Fundamental Properties

- A distributed system is one that runs on a collection of networked machines but acts like **a virtual uniprocessor**.
- There must be a single, **global interprocess communication** mechanism so that any process can talk to any other process.
- Local and remote communication **must be same**.
- There must be a **global protection** scheme.
- **Process management** must also be same everywhere
- There must be **a single set of system calls** available in all machines and must make sense in distributed environment
- The **file system** must look same everywhere too. Every file should be visible at every location, subject to protection and security constraints.
- **Identical kernels** must run on all the CPUs in the system.

#7: Timesharing Multiprocessor system

- Tightly-coupled software on tightly-coupled hardware
- **Key characteristics**: Existence of a single run queue – A list of all the processes in the system that are logically unblocked and ready to run. A run queue is a **data structure** kept in the shared memory.
- The methods used on the multiprocessor to achieve the appearance of a virtual uniprocessor are not applicable to machines that do not have shared memory.
- **Eg**: Dedicated database machines

#7: Timesharing Multiprocessor system Cont.

- 3 CPUs and 5 processes that are ready to run. All the 5 processes are in the shared memory and 3 of them are currently running- A in CPU_1, B in CPU_2 and C in CPU_3 Speed is 10-100 Mbps
- A, B, C are executing, D & E are ready to run
- Mutual exclusion is achieved by Semaphore, Monitor
- If(currently running process has I/O < θ)
  - Then OS makes other process busy waiting

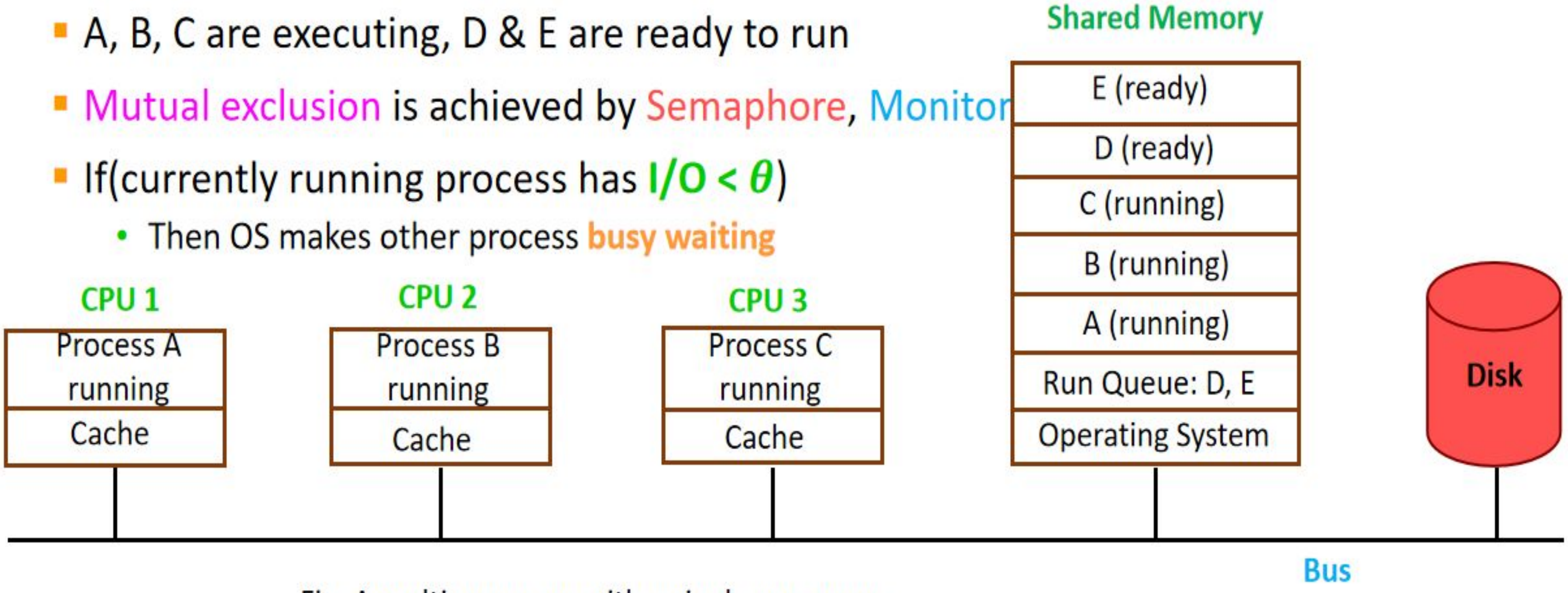


Fig. A multiprocessor with a single run queue

#7: Timesharing Multiprocessor system Cont.

- Comparison among three kinds of systems

Parameter	Network OS	Distributed OS	Multiprocessor OS
Does it look like a virtual uniprocessor	No	Yes	Yes
Do all have to run the same OS?	No	Yes	Yes
How many copies of the OS are there?	N	N	1
How is communication achieved?	Shared files	Messages	Shared memory
Are agreed upon network protocols required ?	Yes	Yes	No
Is there a single run queue?	No	No	Yes
Does file sharing have well-defined semantics?	Usually no	Yes	Yes

Fig: Comparison of three different ways of organizing **n** CPUs

#8: Design Issues: How to Build a DOS?

- How to maintain Transparency?
- How to make it Flexible?
- How to ensure Reliability?
- How to achieve desired Performance?
- How to adopt millions & billions of systems altogether to scale up?

#8.1: How to maintain Transparency in DOS?

- How to achieve the single-system image?
- How do the system designers fool everyone into thinking that the collection systems is simply an old-fashioned timesharing system? i.e. the working details about the systems must be hidden to the users.
- How to achieve transparency?
  - Hide the distribution from the users (It is a higher level work, easy to do)
    1. **$ make file_name // To recompile a large number of files in a directory**
  - Design the system call interface so that the existence of multiple processors is not visible. (It is a lower level work, difficult to achieve)
- Transparency means : Hidden

#8.1.1: Different Types of Transparency

1) **Location Transparency :** The users cannot tell where the resources are located ***(Eg: Location of Servers, PCs, Printers, Files, data bases, S/Ws not known to the users)***
2) **Migration Transparency:** Resources can move at will without changing their names. ***(Eg. File, Directory, S/Ws can be transferred from one server to other server without change of name)***
3) **Replication Transparency:** The users cannot tell how many copies exist ***(Eg. Server is free to replicate the heavily used files and provide the service without the knowledge of the users)***
4) **Concurrency Transparency:** Multiple users can share resources automatically. ***(Eg. If two or more users are accessing same resource, then one should not see others access)***
5) **Parallelism Transparency:** Activities can occur in parallel without the knowledge of the users. **(Eg. Programmers must not know the number of CPUs available in the whole system)**

#8.2: How to make DOS Flexible?

- **Monolithic kernel** (Eg. Centralized OS augmented with networking facilities and the integration of remote services)or **Microkernel** (Eg. Small OS with specific services)?
- Flexibility + Transparency = DOS

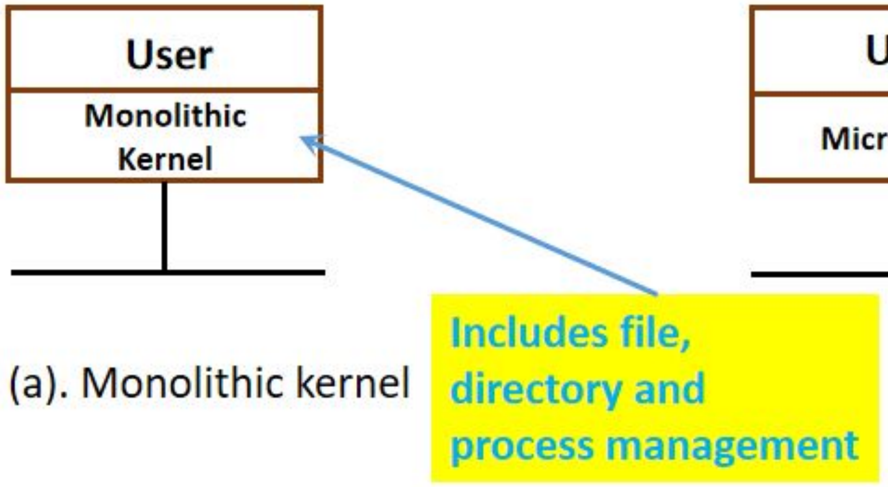


Fig (a). Monolithic kernel

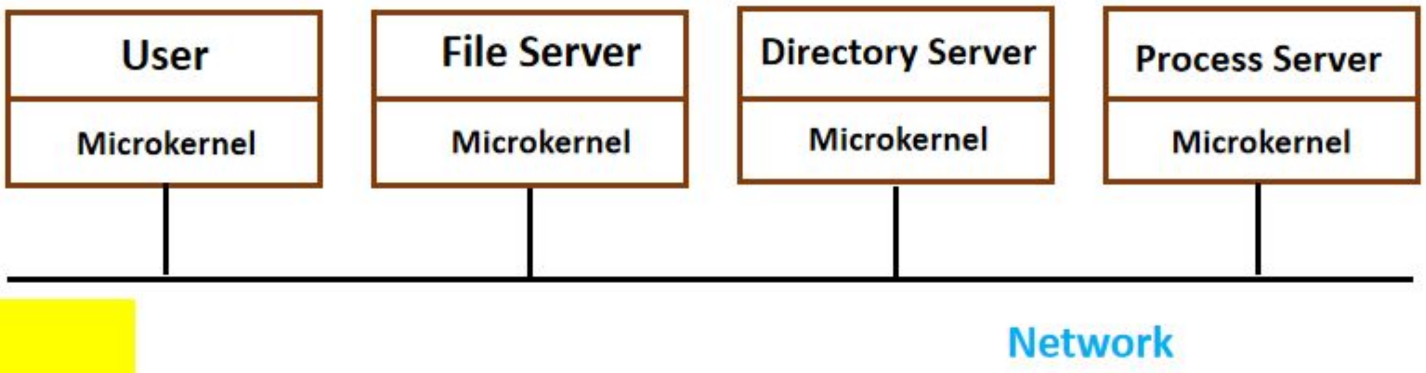


Fig (b). Microkernel

#8.2 Monolithic kernel Vs Microkernel

Monolithic Kernel

- Traditional kernel that provides most services itself.
- It is less flexible
- Centralized OS, augmented with networking facilities and integration of remote services
- The system calls trap the kernel
- Own disks and local files
- DOS that are extension of UNIX OS use this approach
- Eg. Sprite OS

Microkernel

- Kernel should provide as little as possible; bulk services available from user level servers.
- It is more flexible & highly modular
- It provides four services – IPC, Memory mgmt., Process mgmt. and scheduling, Low-level I/O
- It does not provide the file system, directory system, process mgmt.,
- Other services are user level service
- It is highly modular
- Eg. **Amoeba OS**

#8.3: Reliability

- **Availability** of a system is defined as the fraction of time that the system is usable.
- Tool to improve availability is **redundancy** – i.e. keep h/w and s/w replicas, so that if one of them fails the other will take over its place
- A highly reliable system must be highly available, but that is not enough. **No data loss**, **not grabled**, **consistent across**, **security**, **protection** from unauthorized access
- System should be **fault tolerance** (Capability to mask failures)
- All the separate services should be arranged in such a manner that it should **not** add substantial **overhead** to the system.

#8.4: Performance of DOS

- Response Time
- Throughput
- System utilization
- Network capacity consumed
- Communication overhead **dwarfs** the extra CPU cycles gained
- **Fine-grained parallelism** (Large number of small computations)
- **Coarse-grained parallelism** (Large computations, low interaction rates, and little data)
- Cost time

#8.5: Scalability

- Centralized database (without mirror) are almost as bad as centralized components – vulnerable to failure
- Any algorithm that operates by collecting information from all sites, sends it to a single machine for processing, and then distributes the results **must be avoided**.
- Only decentralized algorithms should be used.
- The characteristics of these algorithms are
  - No machine has complete information about the system state
  - Machines make decisions based only on local information
  - Failure of one machine does not spoil the algorithm
  - There is no implicit assumption that a global clock exist

Problem: Omega Switched Network (8×8)

- **Question:** Construct 8×8 omega network for communication between 8 computers and 8 memory locations
- **Ans:** There will be $\log_2(8) = 3$ stages of switches
- Each stage has 8/2= 4 number of switches
- Total number of switches = 12
- In the switch the upper part is for 0 bit and lower part is for 1 bit
- For connection bit-wise left shift is used. It is as follows
- 000 -> 000 -> 000 -> 000 -> 000
- 001 -> 010 -> 100 -> 001 -> 001
- 010 -> 100 -> 001 -> 010 -> 010
- 011 -> 110 -> 101 -> 011-> 011
- 100 -> 001 -> 010 -> 100 -> 100
- 101 -> 011 -> 110 -> 101 -> 101
- 110 -> 101 -> 011 -> 110 -> 110
- 111 ->111 -> 111 -> 111 -> 111

Problem: Omega Switched Network (8 × 8) cont.

- Explanation: 100 -> 001 -> 010 -> 100 -> 100 (bitwise-left-shift for switches)