Introduction to Distributed Operating Systems

Dr. Satyananda Champati Rai
Associate Professor
School of Computer Engineering
KIIT Deemed to be University
Bhubaneswar

UNIT-1: Fundamentals of Distributed OS

- Introduction to DOS
- Goals of DOS
- Hardware Concepts
- Software Concepts
- Networked OS
- True Distributed Systems
- Time Sharing Multiprocessor OS
- Design Issues
- System Architectures

#1: Introduction to DOS

- Background:
  - 1945 - 1985 : Beginning of modern computer era
  - Computers were expensive
  - Most organizations had only a handful of computers
  - Lack of communication among the computers
  - Most computers were operated independently
- Development of two advanced technologies
  - Powerful microprocessors 8-bit, 16-bit, 32-bit and 64-bit
  - High-speed computer networks (LAN, MAN, WAN)
- Subsequently, computing systems composed of large number of CPUs connected by a high-speed network were formed. It is known as **"Distributed Systems"**.

#1: Introduction to DOS Cont.

- Distributed System:
  - A distributed system is a collection of independent computers that appear to the users of the system as a single computer.
  - i.e. A Distributed system has two aspects – Hardware & Software
    1. **Hardware**: Machines are autonomous
    2. **Software**: The users think of the system as a single computer
- **Eg #1**: Network of workstations in an organization
- Eg: Pool of processors in the machine room that are not assigned to specific users but are allocated as needed.
- It might have a single file system, with all files accessible from all machines uniformly and using the same path name.
- A single command may be executed on user's own workstation/idle workstation / unassigned processors in the machine room.
- If the system as a whole looked and acted like a classical single-processor timesharing system, it would qualify as a **"Distributed System"**.

#1: Introduction to DOS Cont.

- **Eg #2**: A factory full of robots, each containing a powerful computer for handling vision, planning, communication, and other tasks.
- All the robots act like peripheral devices attached to the same central computer and the system can be programmed that way, it too count as a distributed system.

#1: Introduction to DOS Cont.

- **Eg #3**: Consider large bank with hundreds of branch offices all over the world. Each office has a master computer to store local accounts and handle local transactions. Each computer has the ability to communicate to all other branch computers and with a central computer at headquarters. If (transactions can be done **without regard to where a customer** or **account** is AND the **user do not notice any difference between this system and the old centralized mainframe** that it replaced) then it is a **"Distributed System"**.

#2: Motivation and Goals of Distributed Systems

- **Q**. Why a Distributed System is essential?
  - **Reasons**: Let us analyze the reasons of using Distributed System.
    1. **Economics:** The real driving force behind the trend toward decentralization is economics. Use of large number of chip CPUs together in a system potentially have a much better price/performance ratio than a single large centralized system would have.
    2. **Speed**: A collection of microprocessors cannot only give a better price/performance ration than a single mainframe, but may yield an absolute performance than no mainframe can achieve at any price. Multiple interconnected CPUs work together.
    3. **Inherent Distributed Application**: Eg- A supermarket chain. The completer system look like a single computer to the application programs, but implement decentrally, with one computer per store. This is a commercial distributed system. Eg #2: Computer Supported Cooperative Game. Eg #3: Cloud-based environment
    4. **Reliability**: if k% of the machines are down at any moment, the system should be able to continue to work with a k% loss in performance. For critical applications, such as control of nuclear reactor or aircraft using a DOS to achieve high reliability may be the dominant consideration.
    5. **Incremental Growth**: With a distributed system, we may add more processors to the system, thus allowing it to expand gradually as the need arises.
  - **Note: Grosch's law**: Computing power of CPU ∝ square of its price

Advantages of Distributed Systems over Independent PCs

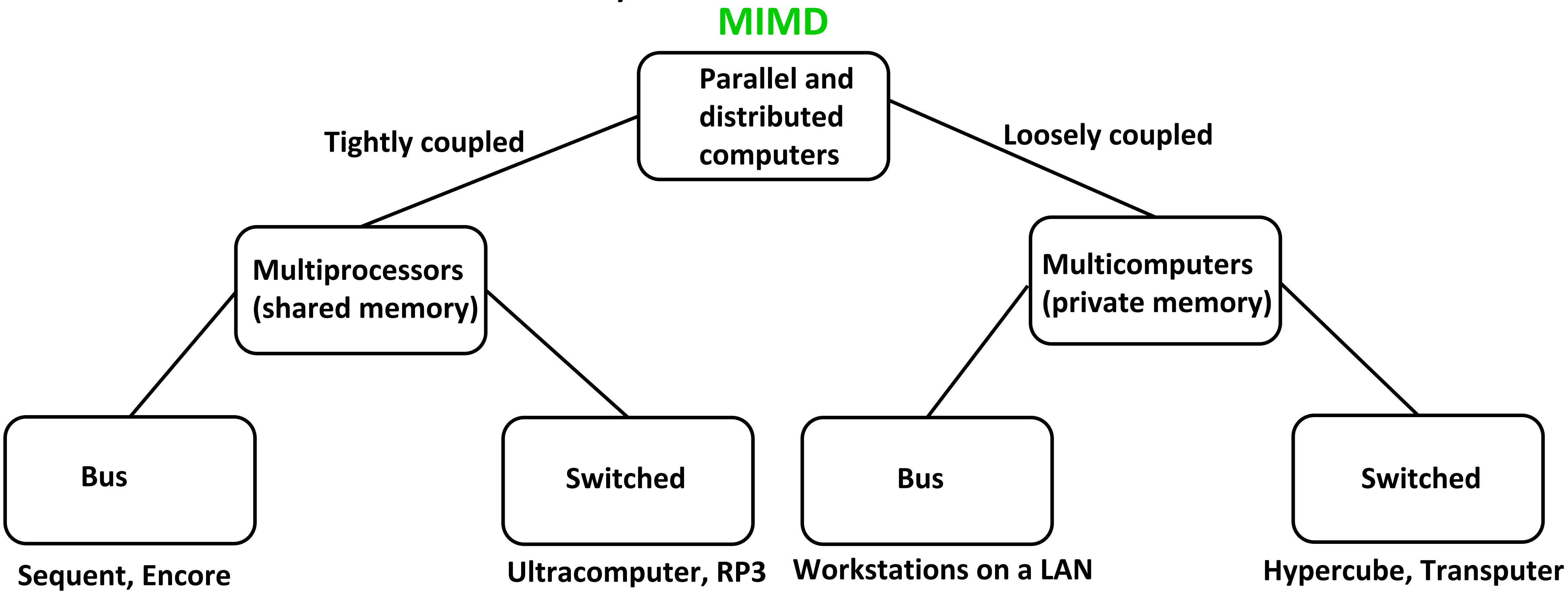
- **Data Sharing**: Allow many users access to a common data base
- **Device Sharing**: Allow many users to share expensive peripherals like color printer
- **Communication**: Make human-to-human communication easier (Eg: Email)
- **Flexibility**: Spread the work load over the available machines in the most cost effective way

Disadvantages of Distributed Systems

- **Software** : More software are required for different distributed applications for smooth functioning across the locations
- **Networking** : The network can saturate or cause other problems
- **Security**: Easy access also applies to secret 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**.

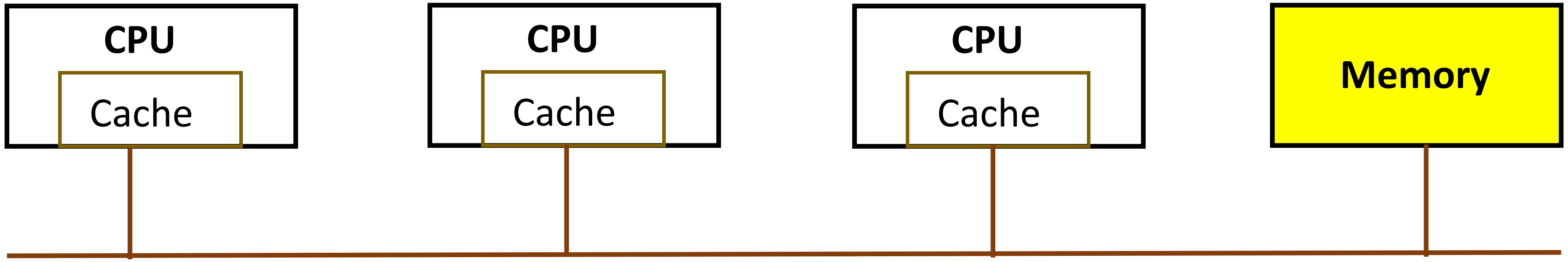


Fig: A bus-based multiprocessor

#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.

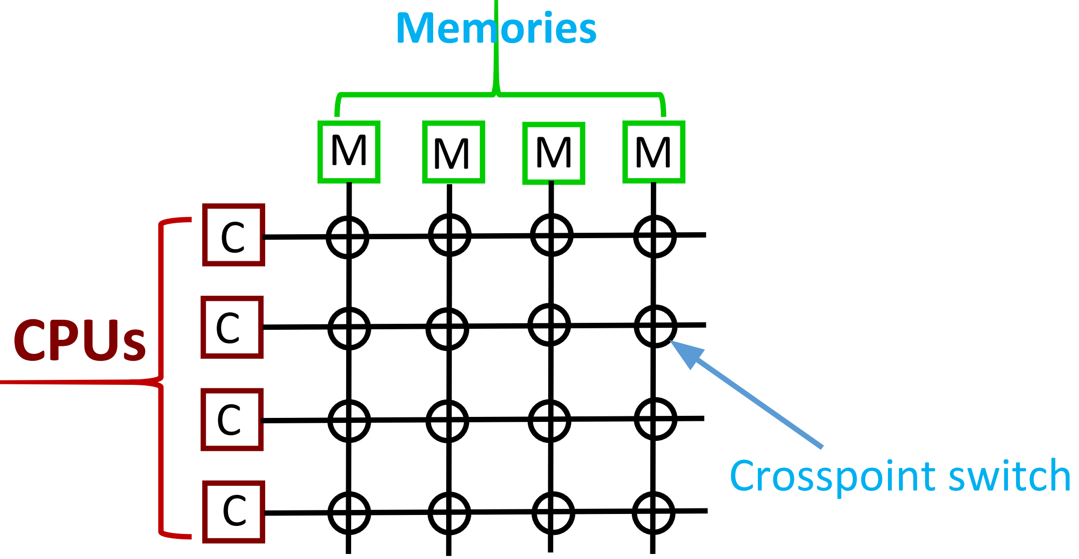


Fig. (a) A Crossbar switch

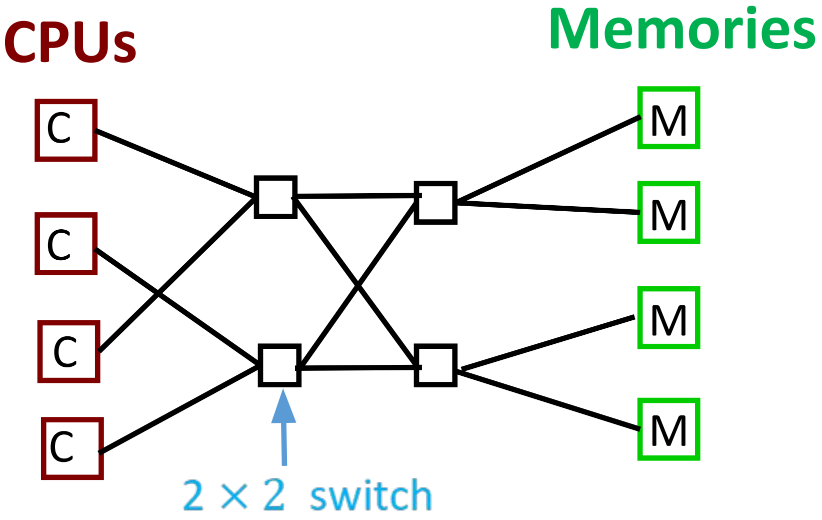


Fig. (b) An **Omega** switching network

#3.2: Switched Multiprocessors cont.

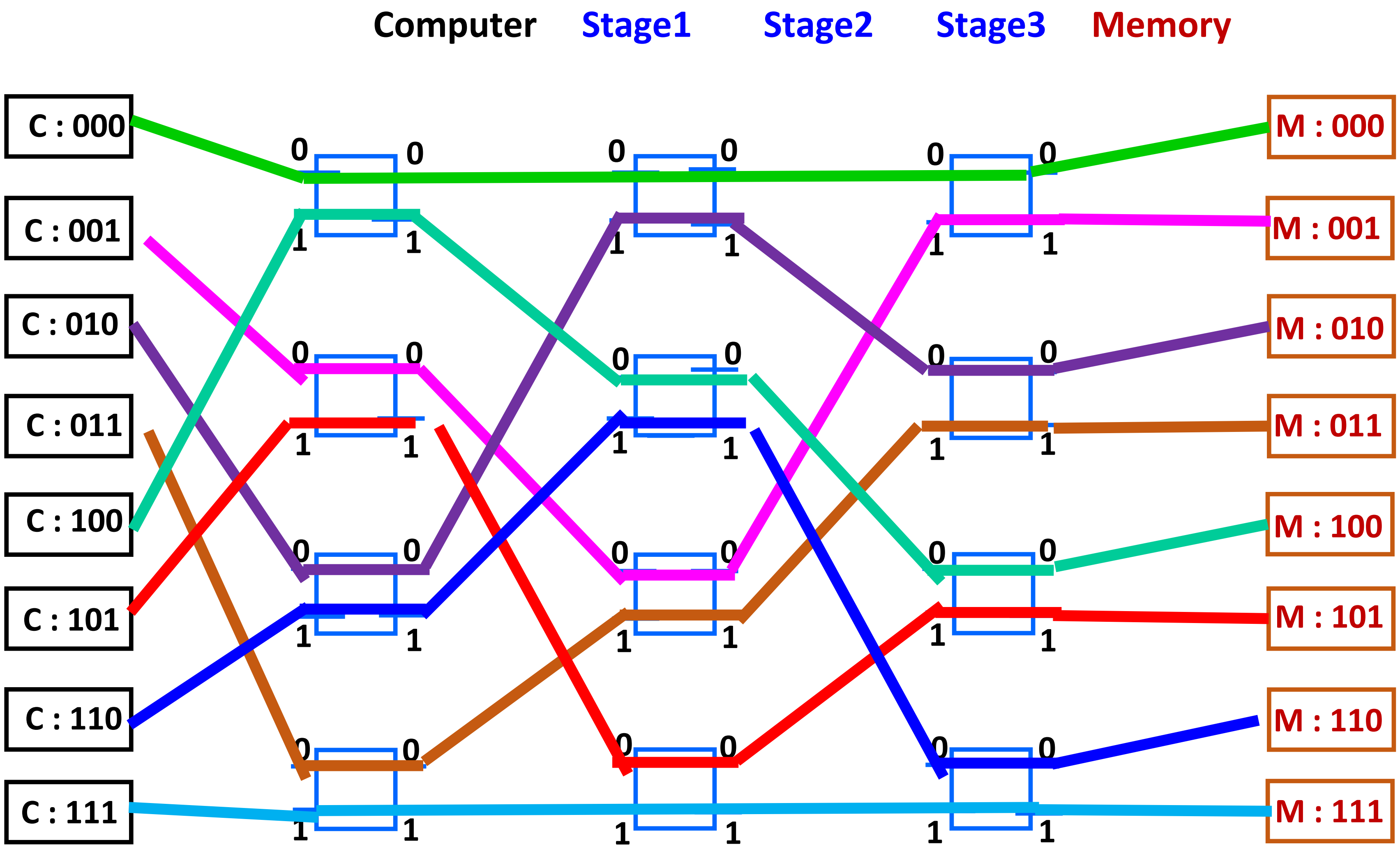
- Characteristics: Memory is divided into modules and connect them to the CPUs with a crossbar switch. (a tiny electronic crosspoint switch).
- A CPU can access a memory after **closing** the crosspoint switch
- If two CPUs try to access the same memory simultaneously, one of them will have to wait.
- 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.
- **Question**: Construct omega network for 8 computers with 8 memory. Explain the communication among any pair of computer to memory. How the conflict is resolved?

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 Rotate left shift operation 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)



Computer Stage1 Stage2 Stage3 Memory

C : 000

C : 001

C : 010

C : 011

C : 100

C : 101

C : 110

C : 111

M : 000

M : 001

M : 010

M : 011

M : 100

M : 101

M : 110

M : 111

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

#3.2: Switched Multiprocessors cont.

- Limitations of Omega Switching Network
  - Slow : Because of multi-stage to-and-from communication
  - Costly : Due to high performance switches
- Example: n=1024, No. of switching stages=log$_2$(1024)=10, To-and-fro stages=10 + 10 = 20, CPU is of modern RISC with chip running at 100 MIPS, So instruction execution time is = 10 nsec. The switching time must be 500 picosec (0.5 nsec). The complete multiprocessor required = log2(1024) * (1024/2) = 5120 number of 2 × 2 switches with 500-picosec switching capability. Hence costly.
- Conclusion: It is possible to build a large, tightly-coupled, shared memory multiprocessor, but it would be difficult and expensive.

#3.3: Bus-Based Multicomputers

- In bus-based multicomputer system it is required to communicate CPU-to-CPU. This need not be a 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

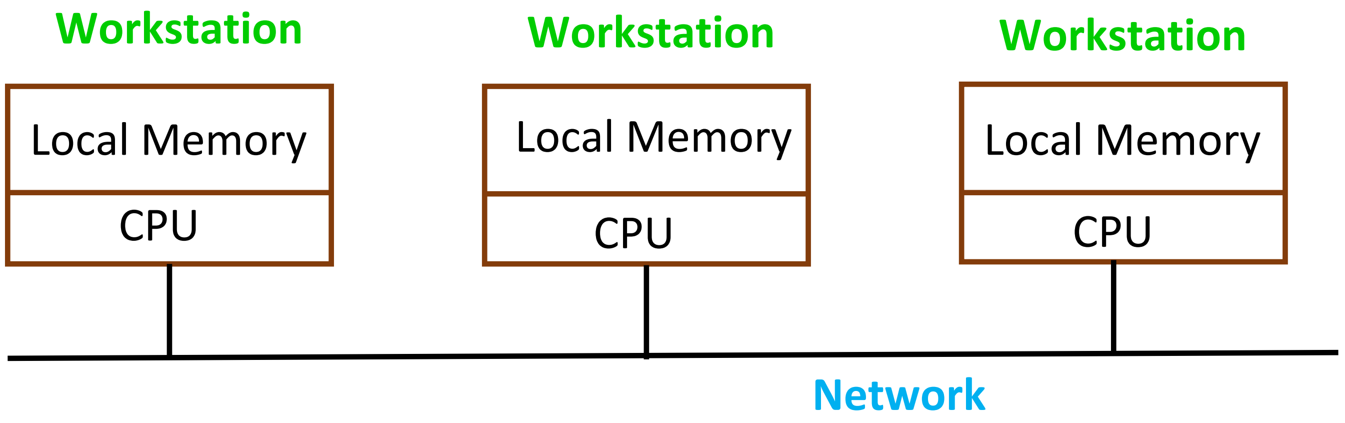


Fig. A multicomputer consisting of workstations on a LAN

#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

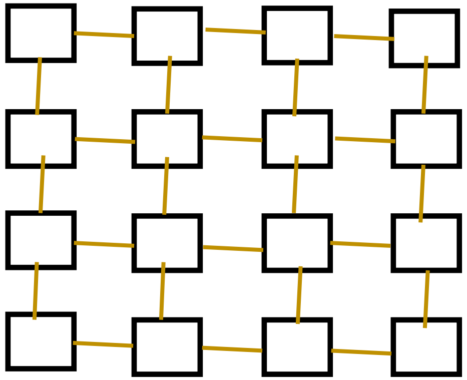


Fig: A Grid-based switched multicomputers

4D-Hypercube

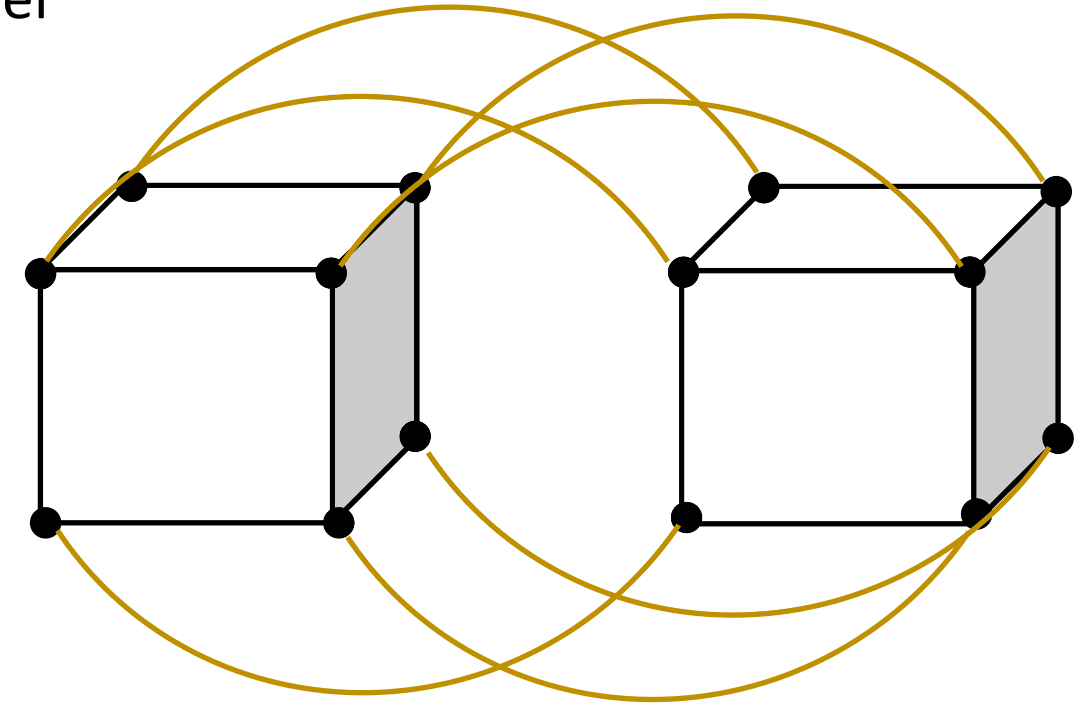


Fig: A Hypercube-based switched multicomputers

#4: Software Concepts

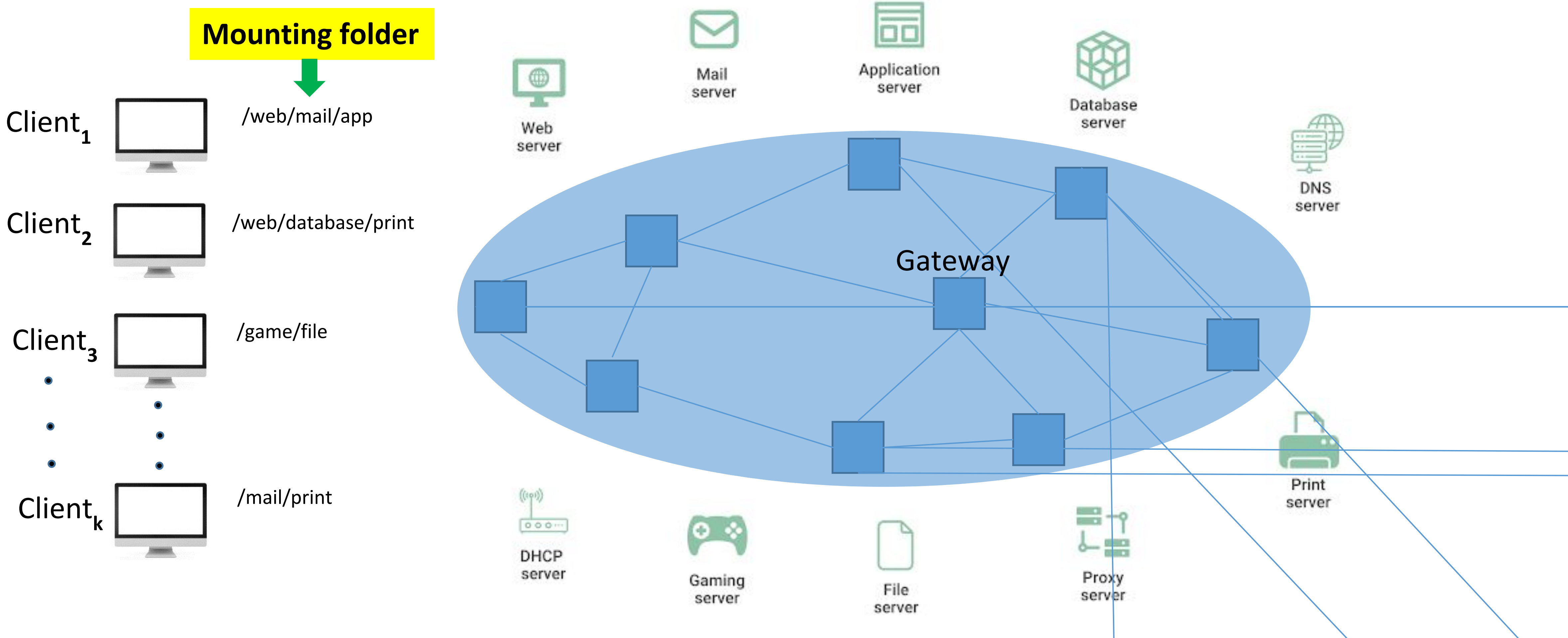
- Two types of OSs for multiple CPU systems
  - Loosely-coupled:
    1. A group of PCs, each has its own CPU, its own memory, Its own HDD, its own OS, but shares some resources, such as printers, databases, over a LAN/MAN/WAN.
    2. i.e. All the PCs are independent of one another.
    3. All the PCs 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

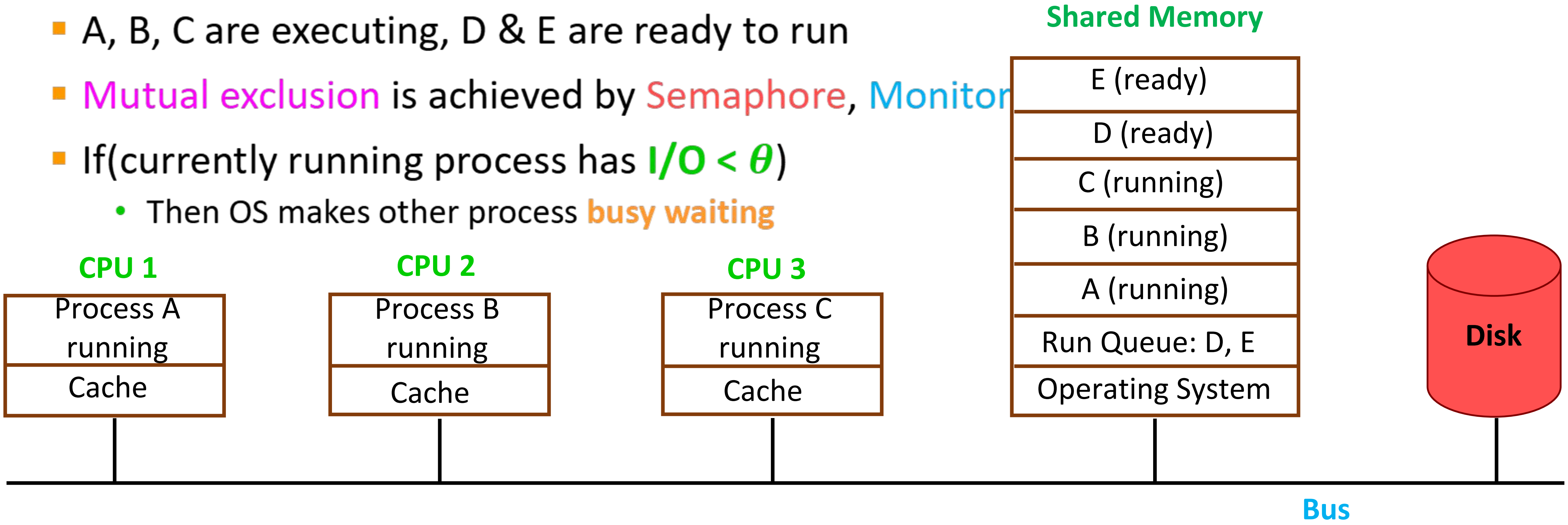


Fig. A multiprocessor with a single run queue

#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?

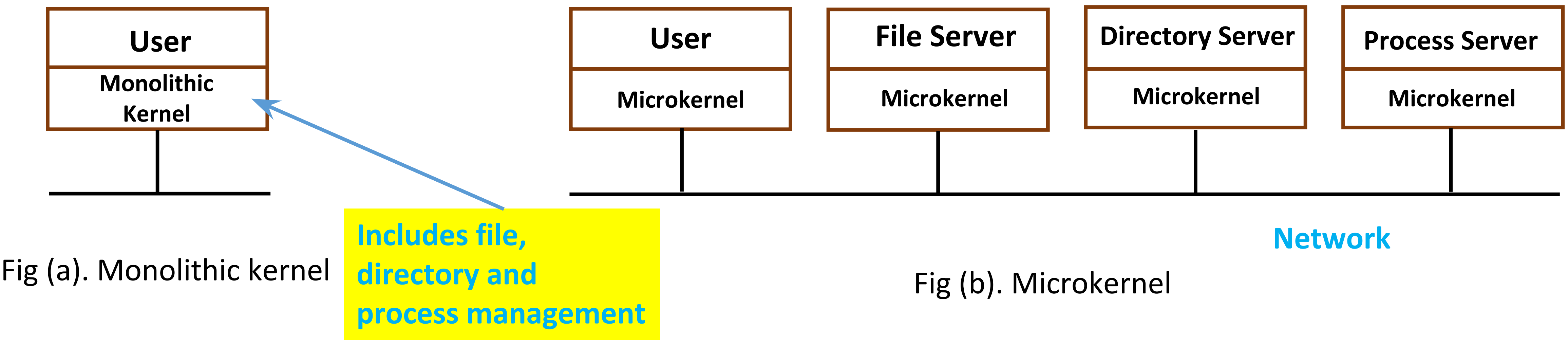
- How to achieve the single-system image?
- How do the system designers fool everyone into thinking that the collection systems is simply an old-fashioned timesharing system? i.e. the working details about the systems must be hidden to the users.
- How to achieve transparency?
  - Hide the distribution 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



Includes file, directory and process management

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

- IPV4 ---- 2^{32} : Number of unique IP addresses for unique systems
- IPV6 ---- $2^{128} \approx 3.4 \times 10^{38}$: Number of unique systems to accommodate IoT
- 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

# 8.5: Scalability cont.

- **Netflix's** own custom global CDN (Content delivery network)
- Distributed GIS technology enables modern online mapping systems (such as **Google Maps** and Bing Maps), Location-based services (LBS), web-based GIS (such as ArcGIS Online) and numerous map-enabled applications.
- Examples of distributed OS are **Solaris**, **AIX**, **OSF**, etc.
- Messaging systems like **WhatsApp** are known as distributed systems. They are designed to operate on a distributed infrastructure, allowing messages to be sent and received across multiple servers and devices.
- With **AWS** High-Performance Computing (HPC), you can accelerate innovation with fast networking and virtually unlimited **distributed computing** infrastructure.
- **Facebook** uses thousands of distributed systems and microservices to power their ecosystem. In order to communicate with each other, these microservices rely on a message queue. Facebook Ordered Queueing Service (**FOQS**) is an internal Facebook tool that fills that role.

#8.5 Scalability Cont.

- **Dropbox's key management** infrastructure is designed with operational, technical, and procedural security controls with very limited direct access to keys. Encryption key generation, exchange, and storage is distributed for decentralized processing.
- **Spotify** uses a distributed network of servers to store and deliver music and podcasts to its users. The company has multiple server locations worldwide, which work together to ensure the content is provided quickly and reliably.
- Social media applications, such as Facebook, **X**, and **Instagram**, are designed to be accessed by users from different devices and locations, and their data and processing are distributed across multiple servers and data centers.

# 8.5: Scalability Cont.

- **Azure** Service Fabric is an example of a distributed systems platform that is designed specifically for building microservices-based applications
- **MongoDB** is a general-purpose, document-based, distributed database management system built for modern application developers
- The **Hadoop** distributed file system acts as the master server and can manage the files, control a client's access to files, and overseas file operating processes such as renaming, opening, and closing files.
- **Bitcoin** is a secure, distributed system that manages consensus about the state of accounts and the authorized transactions among them

# 8.5: Scalability Cont.

- **Git** is a Distributed Version Control System (DVCS) used to save different versions of a file (or set of files) so that any version is retrievable at will
- **Zoom** builds its own distributed cloud-native infrastructure. Cloud-Native means the system is architected to use cloud technology from the ground up. The microservices allow developers to seamlessly grow capacity.

Q&A

Homework Questions

1) A multicomputer with 256 CPUs is organized as a 16 × 16 grid. What is the worst-case delay (in hops) that a message might have to take?
2) Consider a 256- CPU hypercube. What is the worst-case delay in terms of hops?
3) A multiprocessor ha 4096 50-MIPS CPUs connected to memory by an omega network. How fast do the switches have to be allow a request to go to memory and back in one instruction time?
4) An experimental file server is up 3/4 of the time and down 1/4 of the time, due to bugs. How many times does this file server have to be replicated to give an availability of at least 99 percent?
5) Show the communication path for sending a message from computer 3 to memory 6 in a 8×8 omega network.
6) Explain the principle to avoid any conflict in a 8×8 omega network
7) Construct a 16×16 omega network and explain its connection mechanisms.

Thank You!