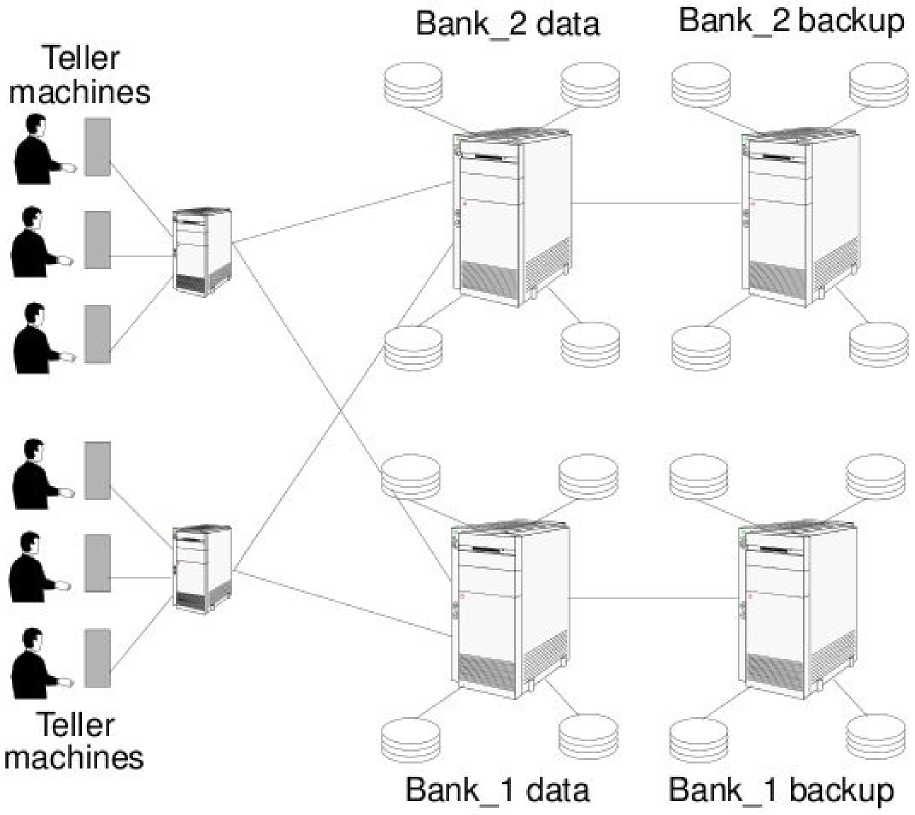
Chapter 8
Distributed OS

--By R.G.B

Introduction

- Multiple CPUs connected together to gain performance or redundancy
- Appears to end user as single system
- It has 2 different aspect hardware and software
- Eg: Parallel machine, Network servers, file server, webservers

Eg: Bank’s Server



Advantages and Disadvantages

- **Performance**
- **Distribution of Information evenly**
- **Reliability** (fault tolerance)
- **Incremental growth**
- **Sharing of data/resources**
- **Communication**

- **Difficulties of developing distributed software**
- **Networking problems**
- **Security problems**

Goals(Design Issues) Of Distributed OS

1. **Transparency**
2. **Communication**
3. **Performance & scalability**
4. **Heterogeneity**
5. **Openness**
6. **Reliability & fault tolerance**
7. **Security**

Goals(Design Issues) Of Distributed OS

- **Transparency** – System look like single computer

a) Access Transparency
b) Location Transparency
c) Mobility Transparency
d) Replication Transparency
e) Concurrency Transparency
f) Failure Transparency
g) Performance Transparency

Goals(Design Issues) Of Distributed OS

- **<u>Communication</u>** : Components of a distributed system have to communicate in order to interact

1. Networking infrastructure (interconnections &network software)
2. Appropriate communication primitives and models and their implementation
   1. Communication primitives
   a. Message Passing (send ,receive)
   b. **<u>R</u>**emote **<u>P</u>**rocedure **<u>C</u>**all
   b) Communication Models
   a) Client-Server
   b) Group Multicast

Goals(Design Issues) Of Distributed OS

- **Performance** :
- Several factors are influencing the performance of a distributed system

1. Performance of individual workstations
2. Speed of the communication infrastructure

- **Scalability**
- System should remain efficient even with a significant increase in the number of users and resources connected

1. Cost of adding resources should be reasonable
2. Performance loss with increased number of users and resources should be controlled
3. software resources should not run out because of increase in hardware

Goals(Design Issues) Of Distributed OS

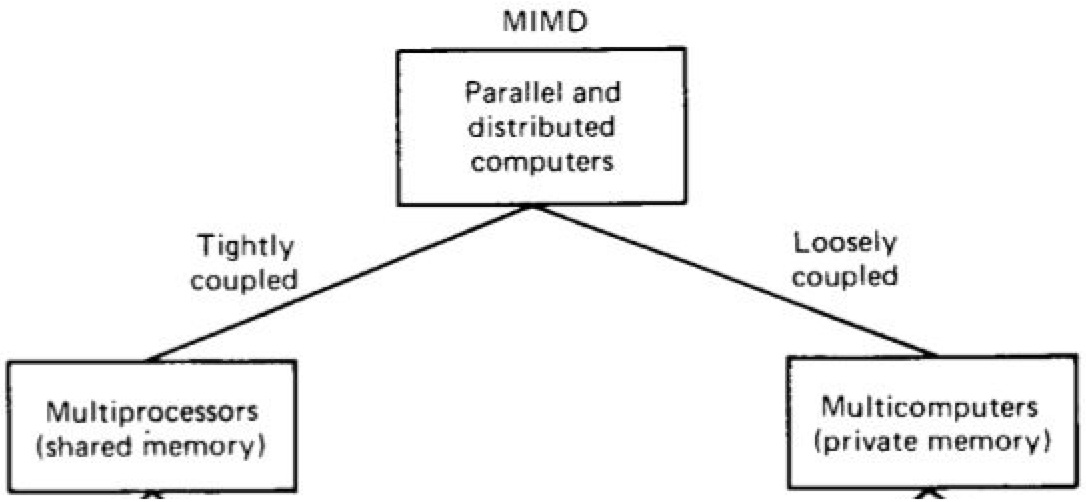
- **<u>Heterogeneity</u> : S**ystems that use more than one kind of processor
- **<u>Openness:</u>**

Be able to interact with services from other open systems, irrespective of the underlying environment

1. Systems should have to well-defined interfaces
2. Systems should support portability of applications
3. Systems should easily interoperate

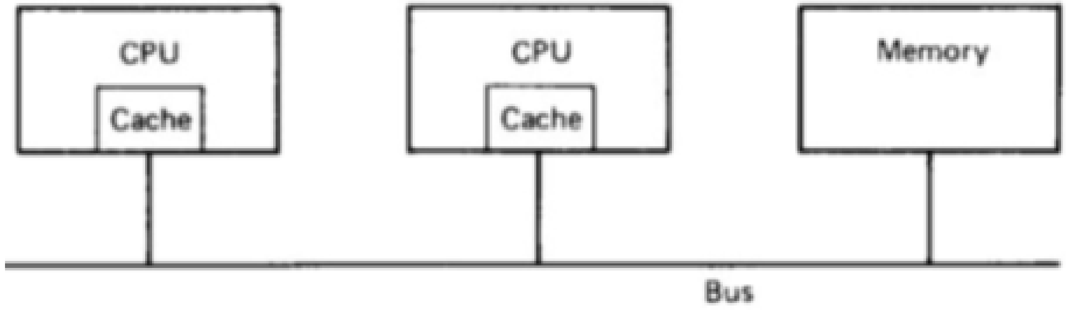
Network Architecture

- Divided into two

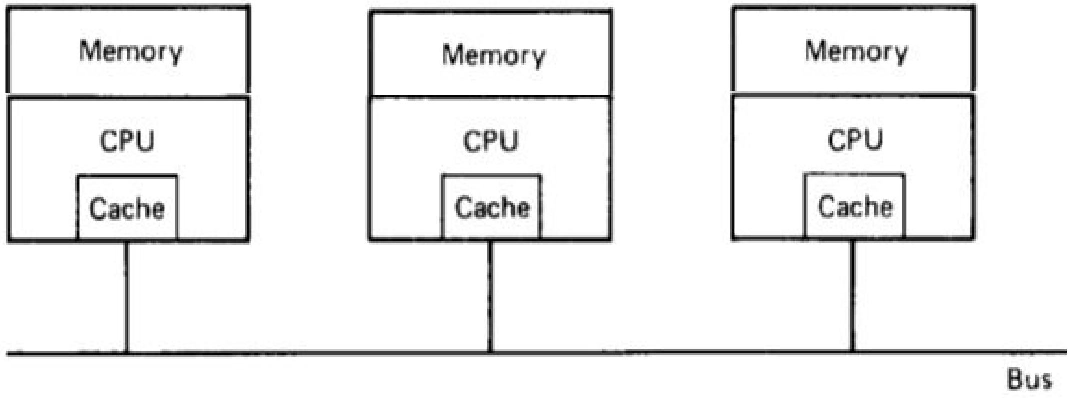


Network Architecture

Multi Processors(Shared Primary memory)



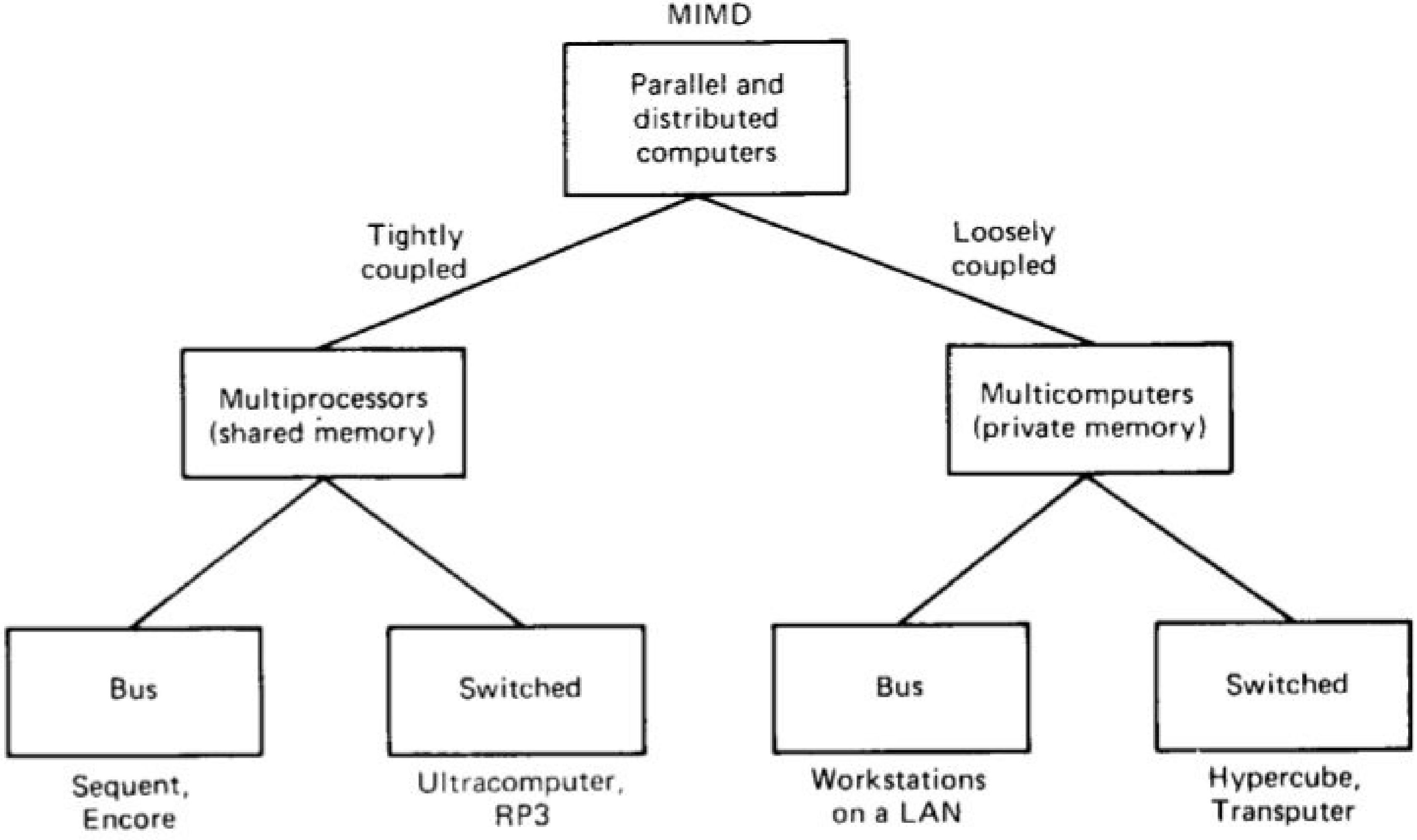
Multi Computers(Separate Primary memory)



Hardware and Software Concept

- Even though all distributed system consists of multiple CPUs
- Several ways the hardware can be organized in terms of how they are interconnected and how they communicate
- **Flynn** proposed the following categories
  1. **Single Instruction Single Data (SISD) stream**
  2. **Single Instruction Multiple Data (SIMD) stream**
  3. **Multiple Instruction Single Data (MISD) stream**
  4. **Multiple Instruction Multiple Data (MIMD) stream**

Topologies/Taxonomy *(All Distributing system are MIMD)*



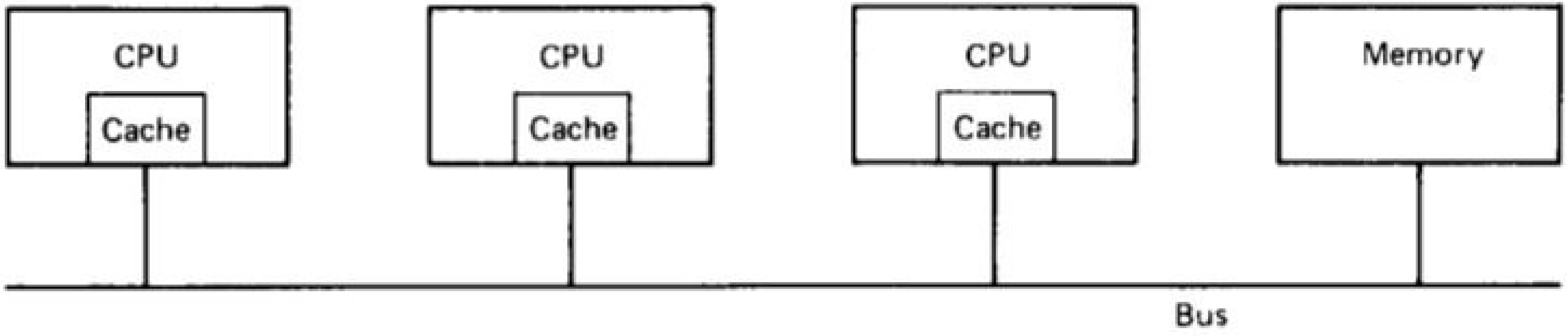
Topologies/Taxonomy

- Distributed Computer

1. Multiprocessor(Tightly Coupled → *Shared memory*)
   a) Bus
   b) Switched → Crossbar Switch and Omega Switch
2. Multicomputer(Loosely Coupled → *Independent memory*)
   a) Bus
   b) Switched

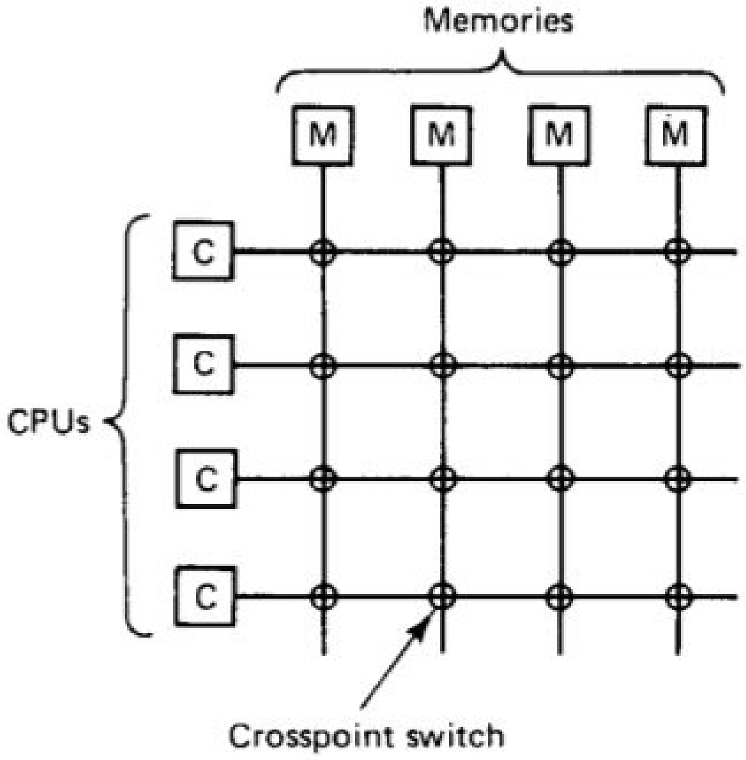
Topologies- *Bus based multiprocessor*

- Bus based multiprocessor consists of some number of CPUs all connected to a common bus, along with a memory module



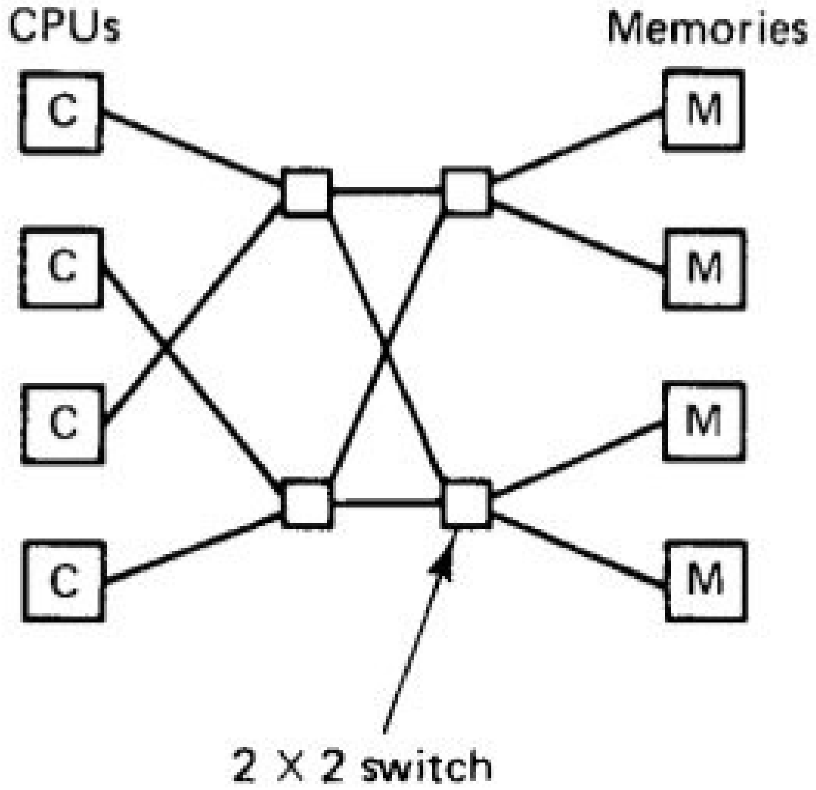
Topologies- *Cross bar switching multiprocessor*

- Memory is divided into the modules and are connected to the CPUs with the crossbar switch
- Each CPU and each memory has a connection coming out of it
- At every intersection is a tiny electronic crosspoint switch that can be opened and closed in hardware
- When a CPU wants to access a particular memory, the crosspoint switch connecting them is closed, to allow the access to take place
- If two CPUs try to access the same memory simultaneously, one of them will have to wait.
- The downside of the crossbar switch is that with n CPUs and n memories, n^2 crosspoint switches are needed



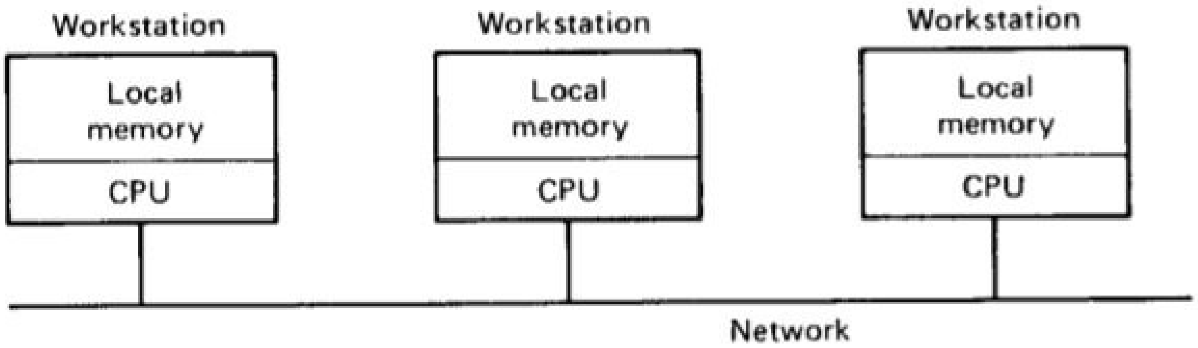
Topologies- *Omega switching multiprocessor*

- It contains 2X2 switches each having two inputs and two outputs
- every CPU can access every memory.
- In general case, with n CPUs and n memories, the omega network requires $\log_2 n$ switching stages, each containing n/2 switches, for a total of $\frac{2\log_2 n}{2}$ switches



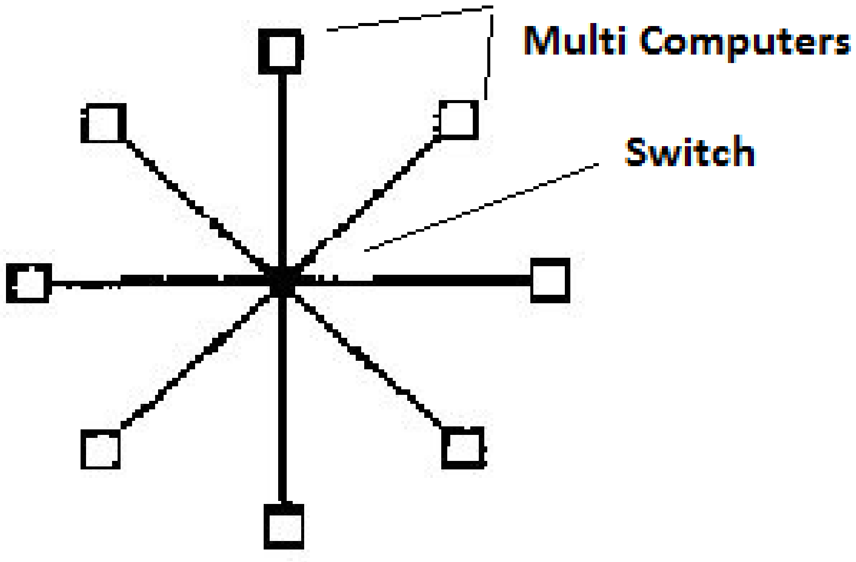
Topologies- *Bus based multicomputer*

- Each CPU has a direct connection to its own local memory.
- CPUs communicate with each other with help of LAN/ Network Cable



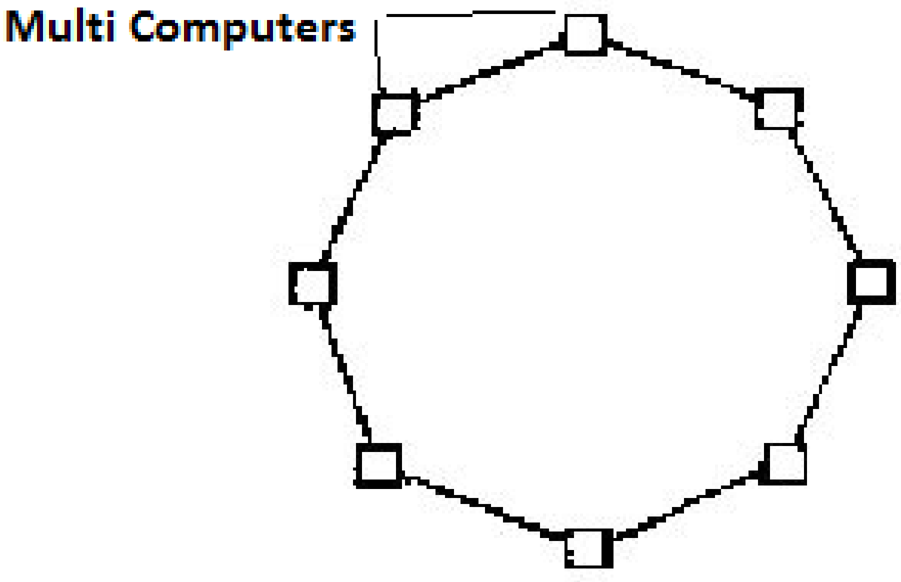
Topologies- *Multicomputer Star*

- Central Controlling device (Switch) connect all multi computer together
- All data traffic goes through switch



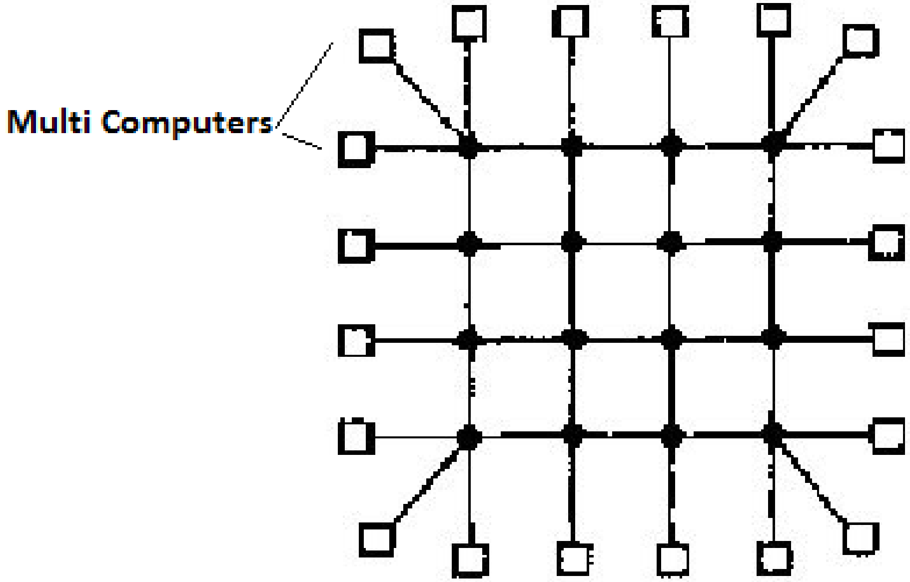
Topologies- *Multicomputer Ring*

- All multi computer connected in ring structure
- Each computer have an in and out interface
- Data will be passed through 2,3 if you send data from 1 to 4

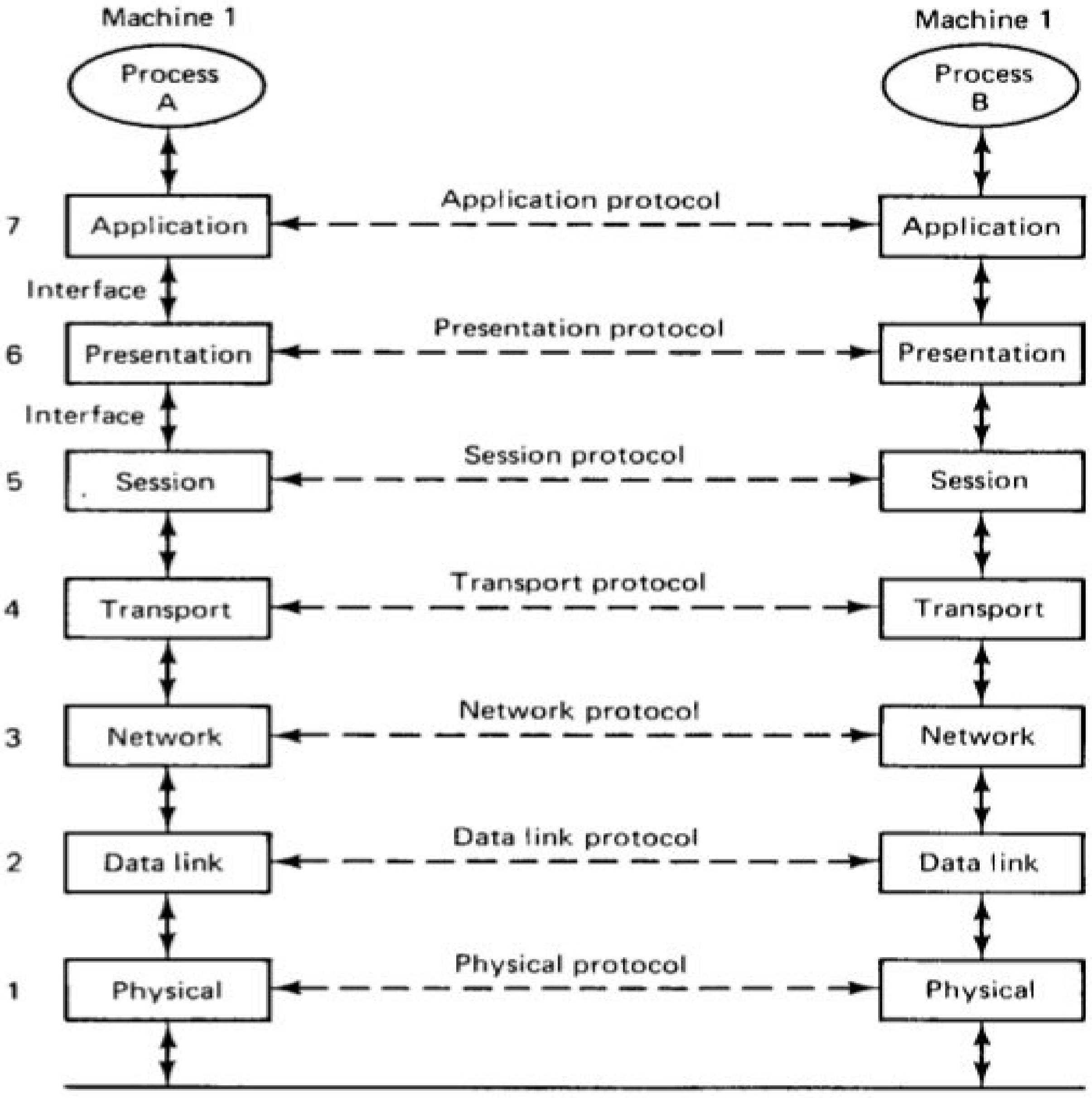


Topologies- *Multicomputer Grid*

- All computer have connection to all the other computer
- Arranged in Grid structure
- Even if one connection is all other work properly

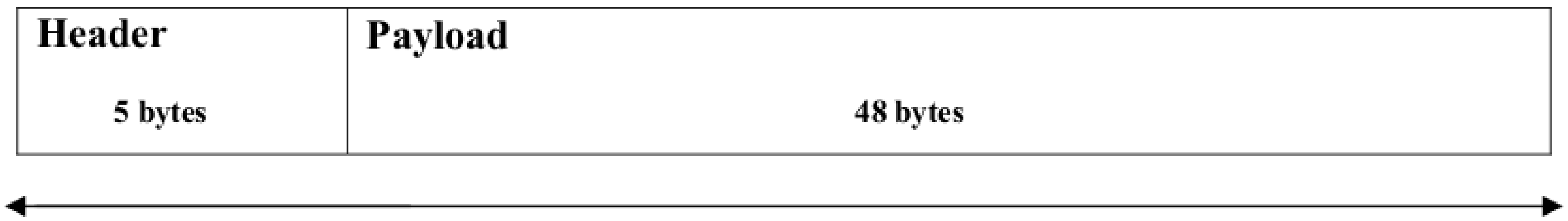


Communication in Distributed OS(ISO/OSI Model)



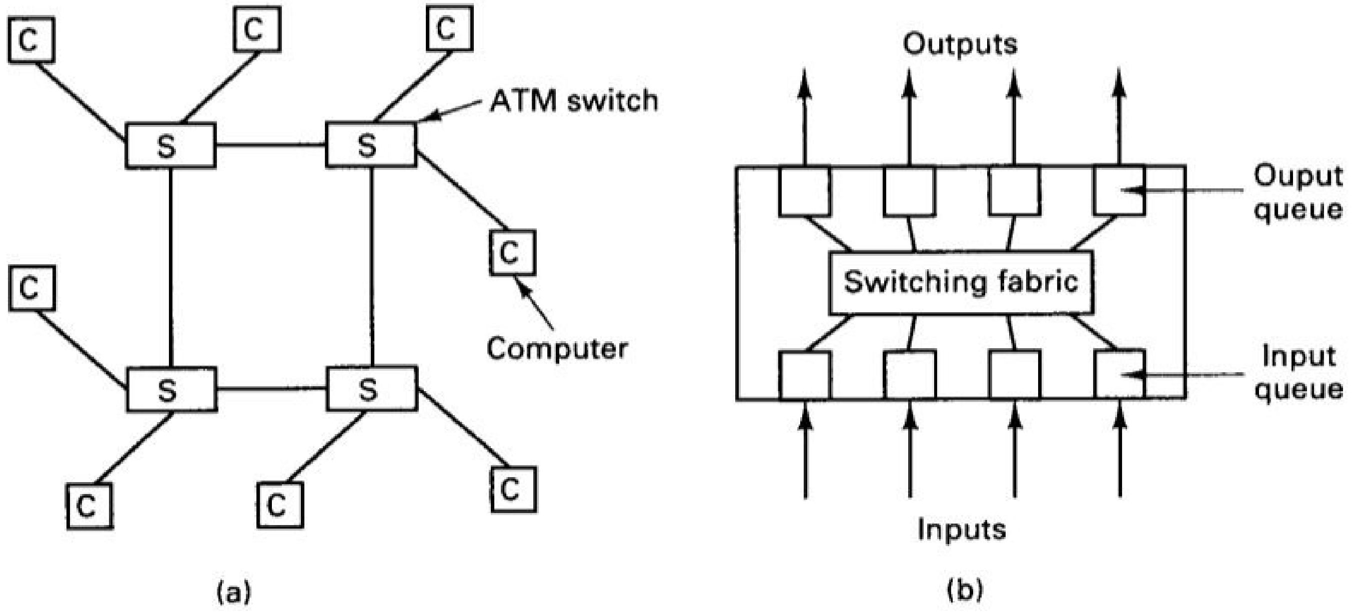
Asynchronous Transfer Mode(ATM)

- ATM devices do not send and receive information at fixed speeds or using a timer
- Negotiate transmission speeds based on hardware and information flow reliability
- Fixed-size cell structure used for packaging information.
- ATM transfers information in fixed-size units called cells
- Each cell consists of 53 octets, or bytes



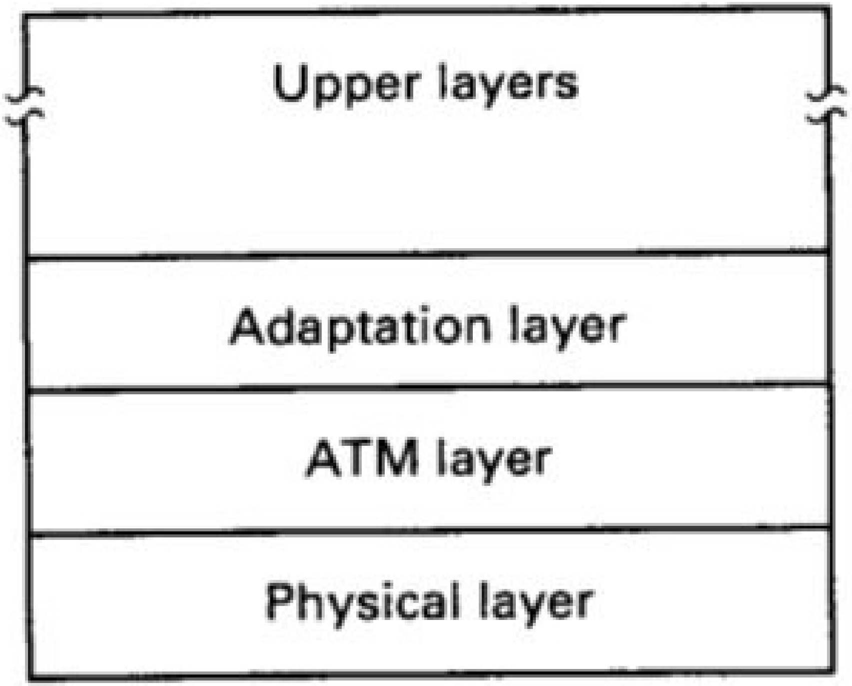
Asynchronous Transfer Mode(ATM)

- ATM network with 4 switches
- Each of these switches has 4 ports, each used for both input and output lines



Asynchronous Transfer Mode(ATM)

- **ATM Layers**

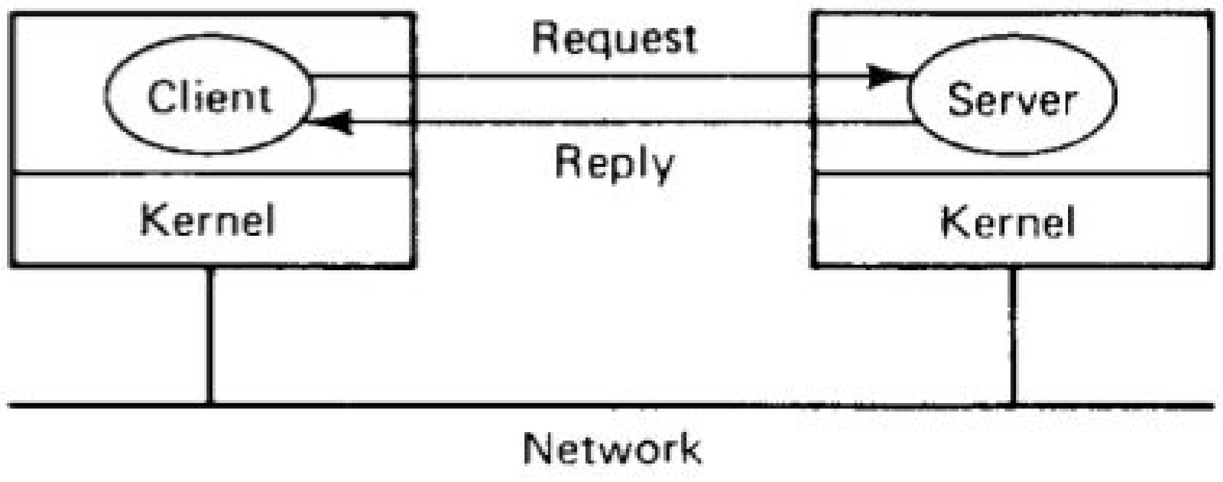


Client Server Model

- Operating-system as a group of cooperating process, called servers, that offer services to the users, called clients
- The client and server normally all run the same microkernel, with both clients and severs running as user processes
- A machine may run a single process or it may run multiple clients, multiple servers or mixture of both
- To avoid the considerable overhead of the connection oriented protocol such as TCP and OSI Model are used

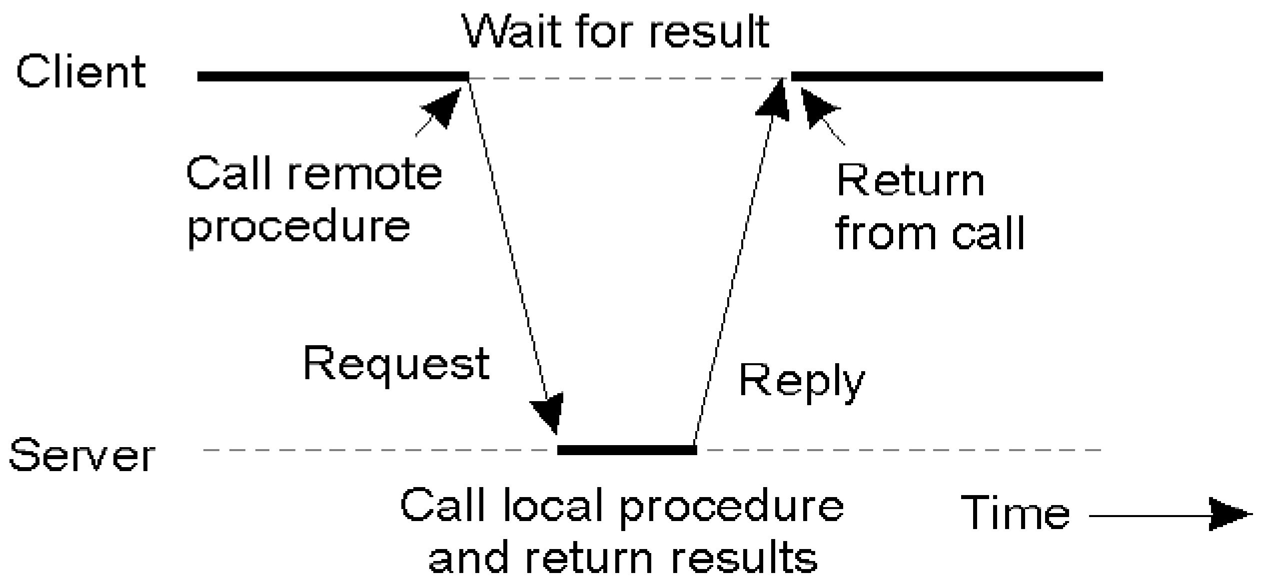
Client Server Model

- The client sends a request message to the server asking for some services (eg. read a block of a file)
- The server does the work and returns the data requested or an error-code indicating why the work could not be performed



Remote Procedure Call(RPC)

- Inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network)



Remote Procedure Call(RPC)

- **<u>PRINCIPLE OF RPC:</u>**
- Client makes procedure call to the client stub
- Server is written as a standard procedure
- Stubs take care of packaging arguments and sending messages
- Packaging parameters is called marshalling
- Stub compiler generates stub automatically from specs in an Interface Definition Language(IDL)
- Simplifies programmer task

Remote Procedure Call(RPC)

Remote procedure call occurs in the following steps:

1. The client procedure calls the client stub in the normal way.
2. The client stub builds a message and calls the local operating system.
3. The client’s OS sends the message to the remote OS.
4. The remote OS gives the message to the server stub.
5. The server stub unpacks the parameters and calls the server.
6. The server does the work and returns the result to the stub.
7. The server stub packs it in a message and calls its local OS.
8. The server’s OS sends the message to the client’s OS.
9. The client’s OS gives the message to the client stub.
10. The stub unpacks the result and returns to the client.