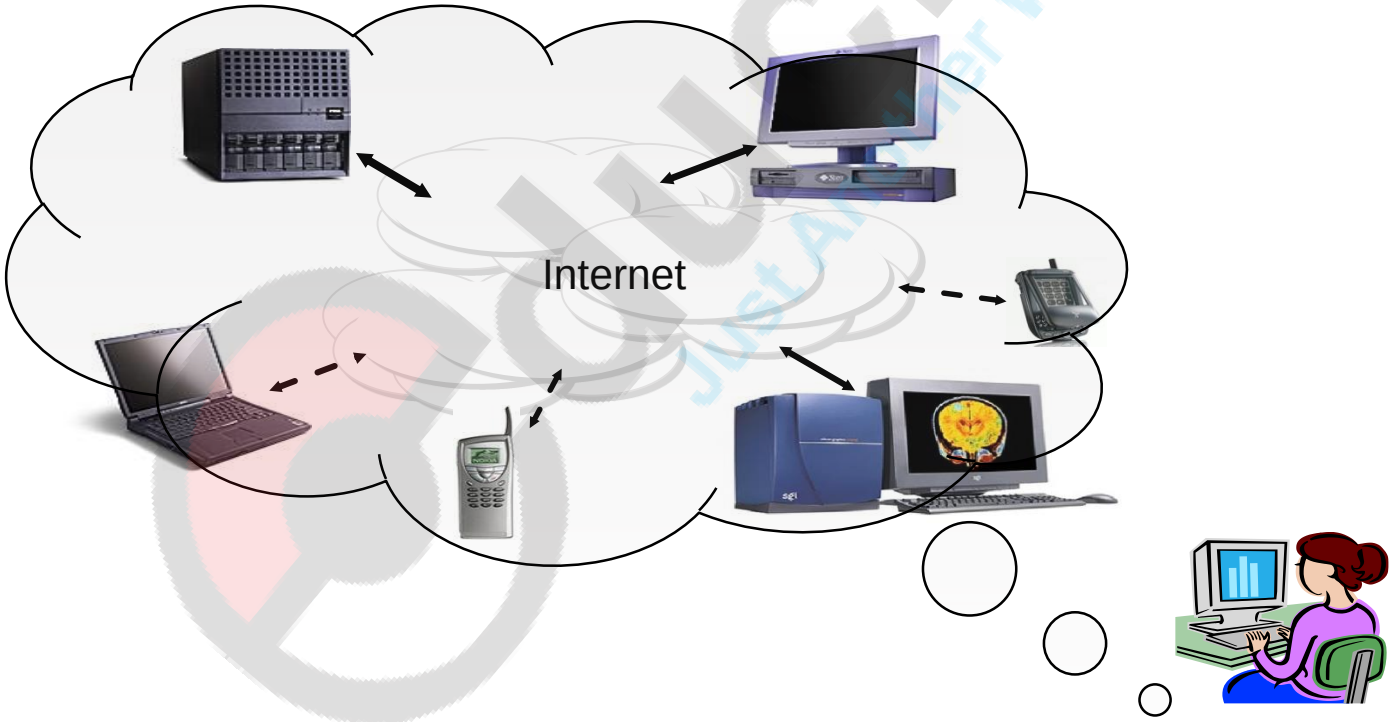
INTRODUCTION TO DISTRIBUTED SYSTEMS

References

- Distributed OS by Pradeep Sinha (PHI)
- Distributed OS by Sunita Mahajan

Distributed Computing Systems

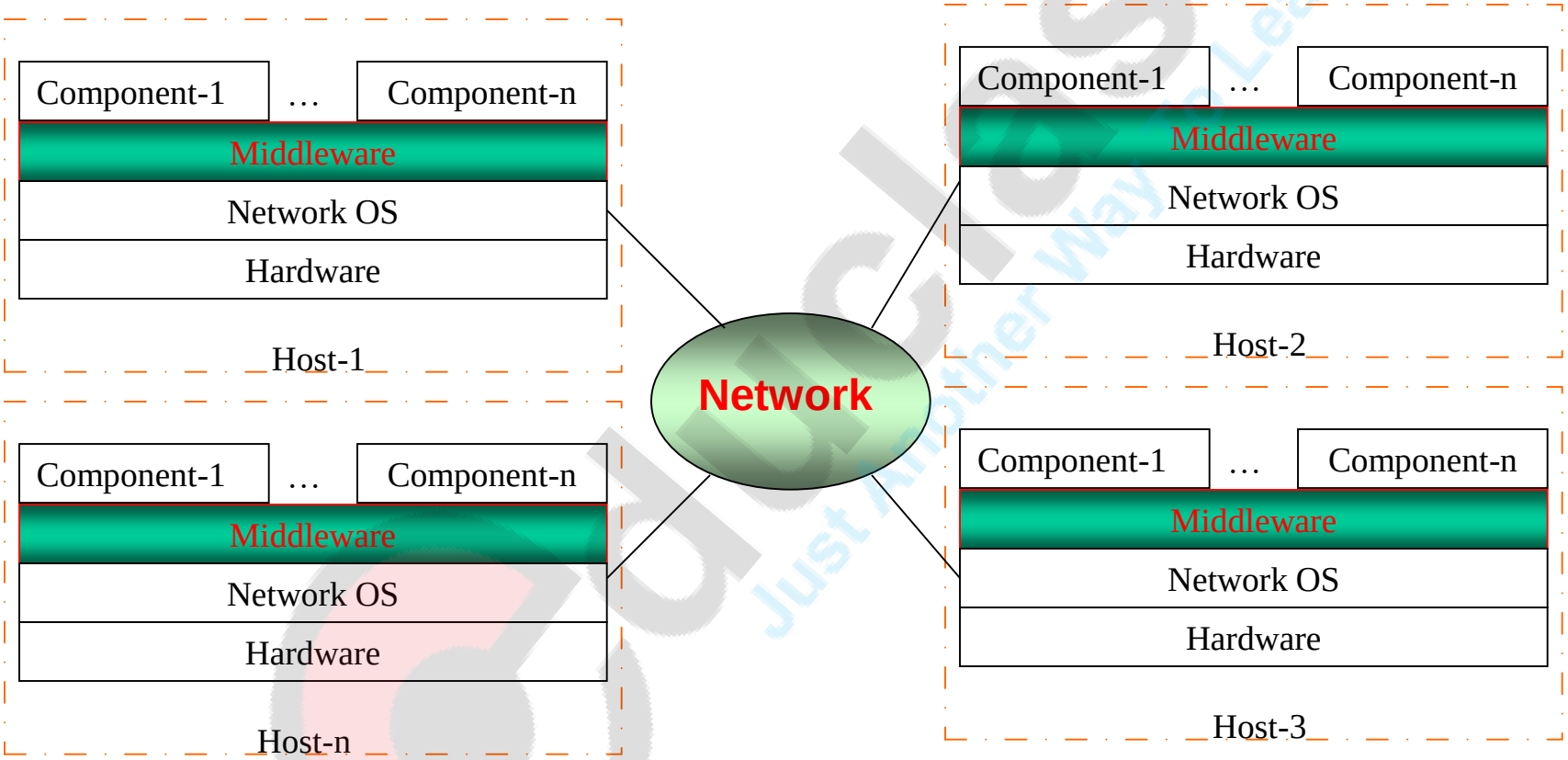
- A collection of independent computers that appears to its users as a single coherent system.



Definition of Distributed System

- This definition has two aspects
  - First one deals with Hardware: individual systems which are autonomous and heterogeneous i.e., different CPU designs and operating environment
  - Second is the software : that make the users think that they are dealing with a single and large powerful computing environment
  - Both are essential, as well as crucial for the concept of modern day distributed system

What Is a Distributed System?



Centralised Systems:

- System shared by users all the time
- All resources accessible
- Software runs in a single process
- Single physical location
- Single point of control
- Single point of failure

Distributed System Characteristics

- Multiple autonomous components
- Components are not shared by all users
- Resources may not be accessible
- Software runs in concurrent processes on different processors
- Multiple points of control
- Multiple points of failure

Evolution of Distributed Systems

- Early Computers were very bulky occupying large rooms, expensive and with low processing power
- Accessible only to few research centers and big industries
- Most of the processing was done in batch mode
- Initial inputs were using punched cards
- Output through big line printers and plotters
- Concept of networking was unknown
- Data stored in magnetic tapes
- None of them allowed multiple users to directly interact with a computer system and share resources simultaneously

Evolution of Distributed Systems

- 1970s saw the concept of time-sharing systems also known as main frames
  - Several dumb terminals connected to the main computer system
  - Multiple users could log into the system and share resources
  - The concept of time sharing gave the impression that he or she has his or her own computer
  - The computer switches rapidly from one user job to next user's job, executing only a small portion of each job at a time

Revolution

- Revolution from a machine that executed 1 instruction per sec costing several crores of rupees to systems that process few millions of instructions and cost only few lakhs of rupees
- 1980 saw two major changes:
  - Development of fast microprocessors
    - 8 bit → 16 bit → 32 bit → 64 bit became common
    - Each had more computing power than the earlier mainframes
  - Development of Local area Networks or LANs
    - 100s of machines in a building got interconnected
    - Allowing large amount data to be shared between system in a small interval of time
    - Wide Area Networks connecting millions of systems interconnected all over the world

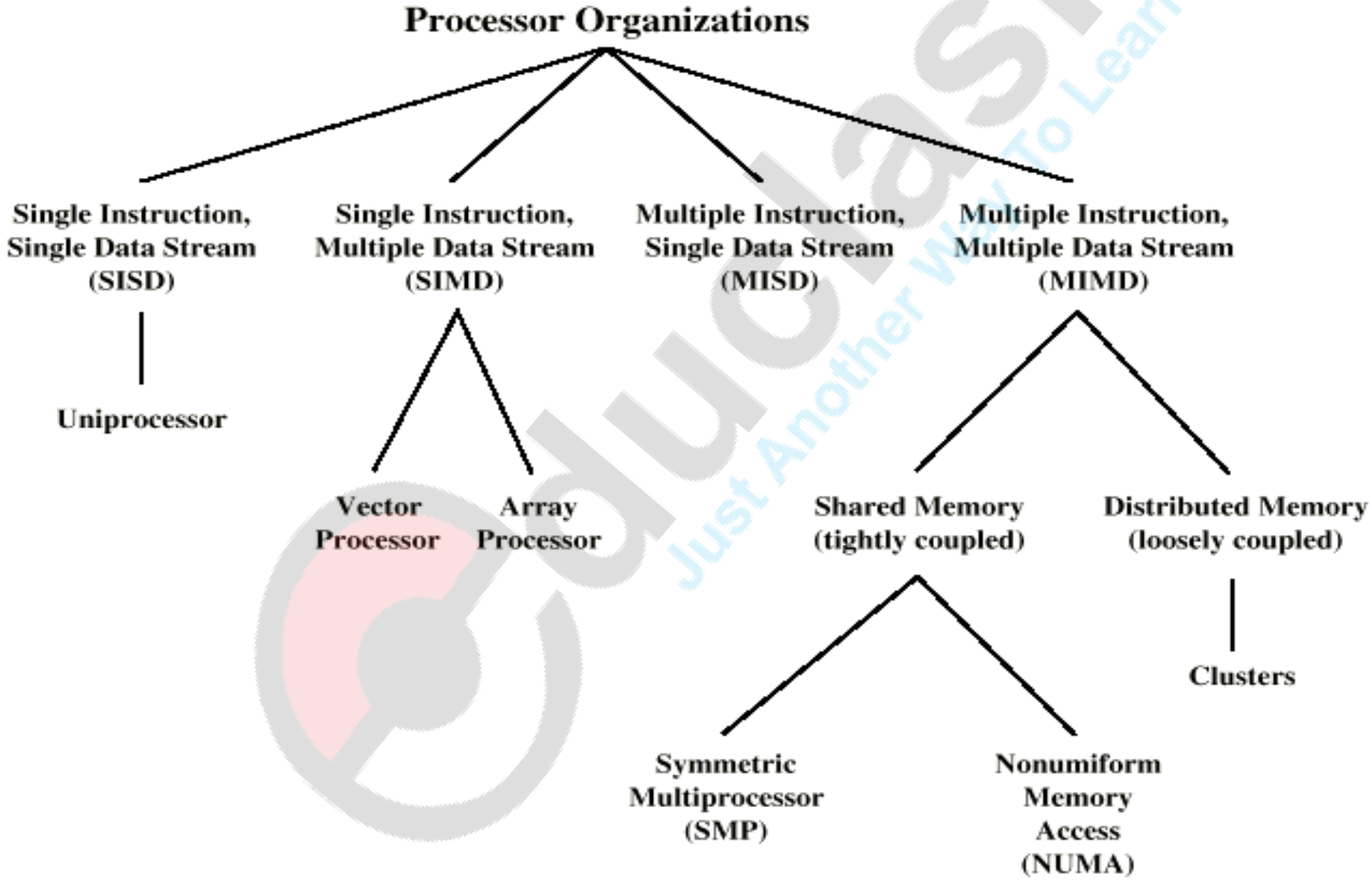
Evolution of DCS

- Batch Processing System
  - Batching together jobs with similar needs
  - Automatic sequencing of jobs with control cards
  - Off-line processing (By using buffering, spooling)
  - User does not directly interact with the computer system.
- Disadvantage
  - Less user interaction
  - No sharing of resources
  - Job set up time still significant for new batch of jobs
  - CPU remain idle during transition.
  - Speed mismatch (CPU & I/O dev)
- Time-Sharing Systems
  - Several dumb terminals are attached to main computer
  - Multiple user could now simultaneously execute interactive jobs and share the resources

Evolution of DCS (Cont'd)

- Time-Sharing Systems
  - The CPU is multiplexed among several jobs that are kept in memory
- Advantages
  - Reduces CPU idle time.
  - Avoid duplication of software
- Disadvantages
  - Increased overhead. (Time to swap page in and out)
  - Terminals could not be placed far from main computer
- Due to advancement in networking technologies LAN and WAN came into existence – lead to evolution of **Distributed Computing.**

Taxonomy of Parallel Processor Architectures (Flynn's Classification)

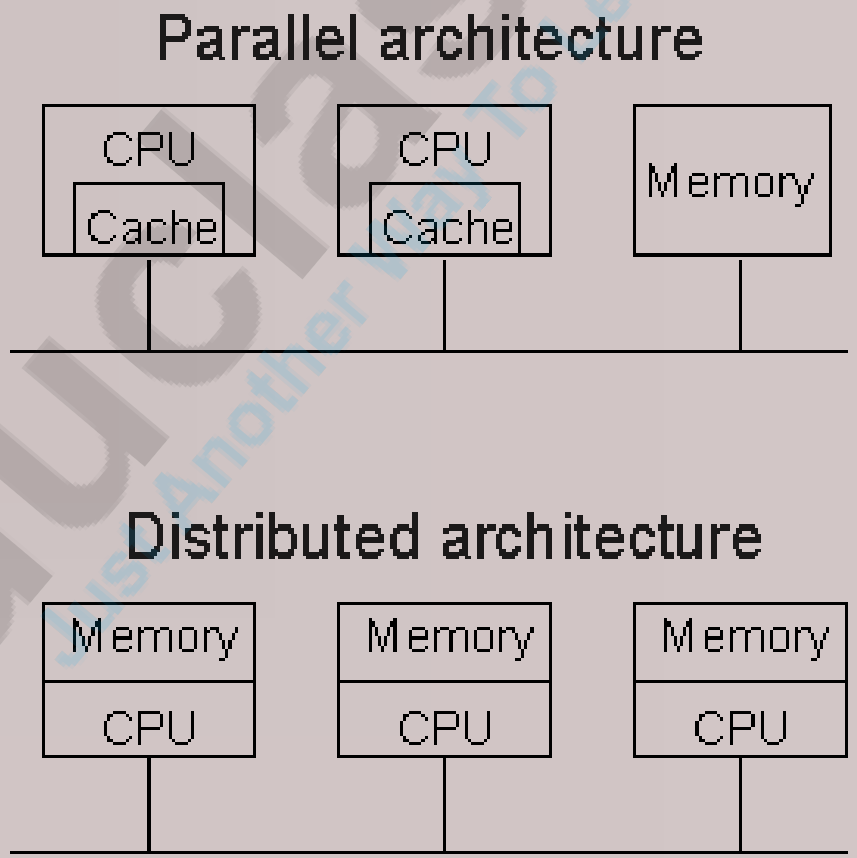


Hardware Considerations

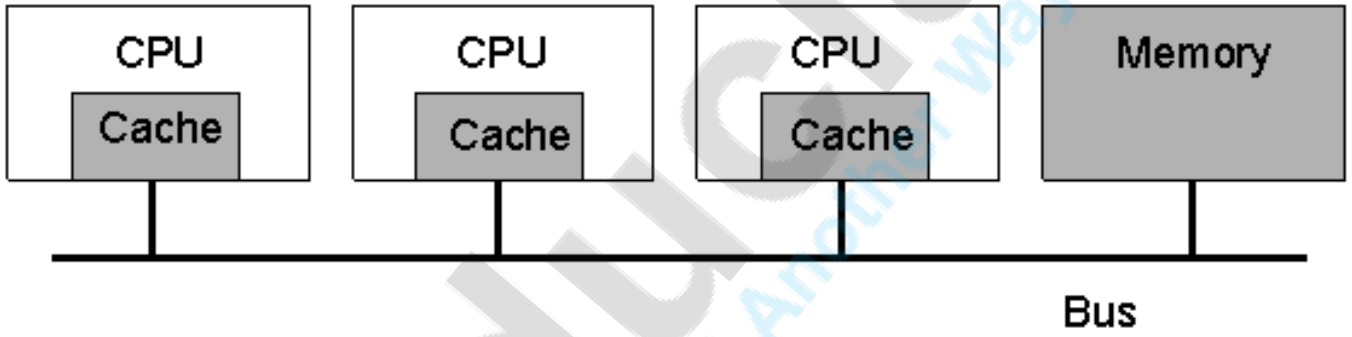
- Architecture of interconnected multiple processors are of two types:
- Tightly Coupled System
  - Single system wide primary memory
  - Communication takes place through shared memory
  - Systems are limited by bandwidth of share memory
  - It is also known as Parallel Processing Systems
- Loosely Coupled System
  - Each processor has its own local memory
  - It can have unlimited number of processors
  - Communication is done by passing message across the network
  - It is known as Distributed Computing System

Parallel vs. Distributed Architecture

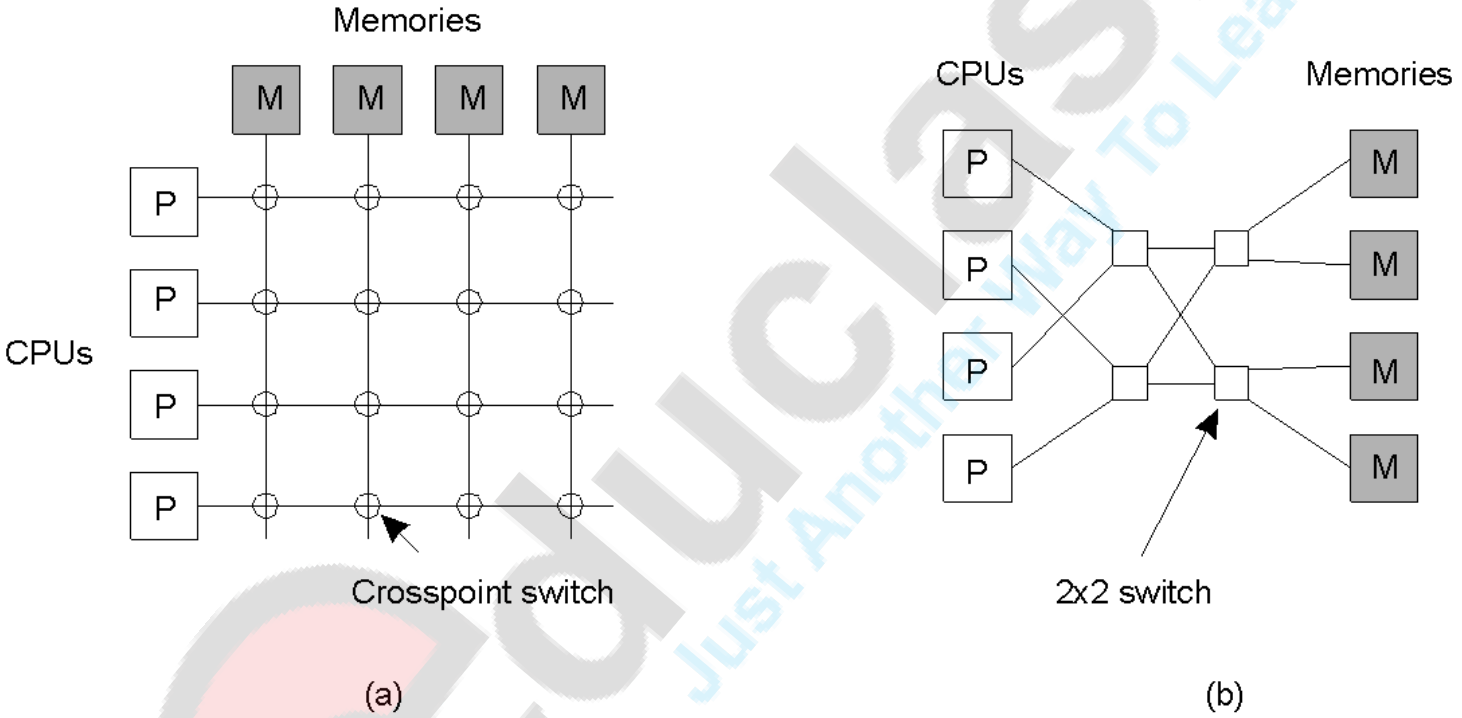
- Multiprocessors
  - Tightly coupled.
  - Shared memory.
- Multicomputers.
  - Loosely coupled.
  - Private memory.
  - Autonomous.



Multiprocessors (1)



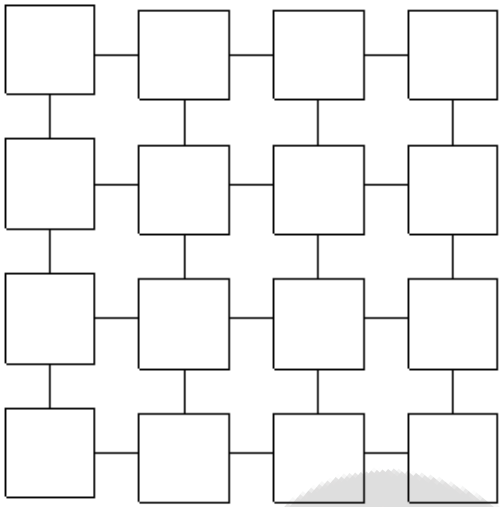
Multiprocessors (2)



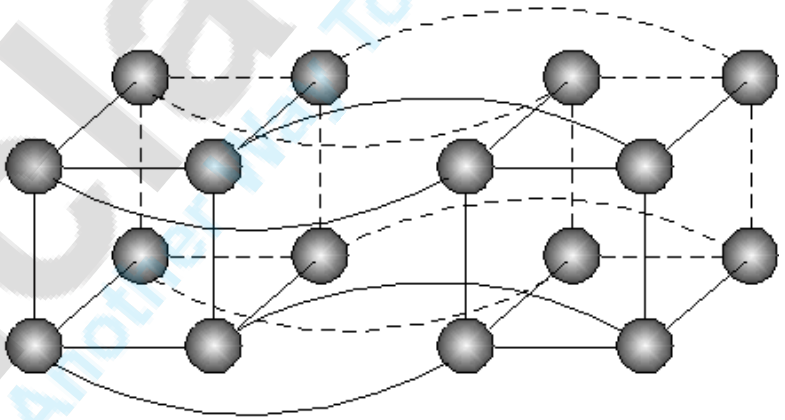
a) A crossbar switch

b) An omega switching network

Homogeneous Multicomputer Systems



(a)



(b)

a) Grid
b) Hypercube

Parallel vs. Distributed Systems

	Parallel Systems	Distributed Systems
Memory	**Tightly coupled shared** memory	**Distributed** memory Message passing, RPC, and/or use of distributed shared memory
Control	**Global clock** control	**No global clock** control. Synchronization algorithms needed
Processor interconnection	Bus, mesh, tree, mesh of tree, and hypercube network	Ethernet(bus), token ring and SCI (ring)
Main focus	**Performance** - Scientific computing	**Performance** - cost and scalability, Reliability, Information/resource sharing

Distributed Computing Systems

- A collection of mostly heterogeneous nodes connected by one or more interconnection networks which provides access to system-wide shared resources and services.
- It is basically a collection of interconnected processors covering wide geographical area in which each processor has its own local memory and other peripherals.
- The communication between any two processor takes place by message passing over communication network.

Characteristics of a DS

- Another important aspect of Distributed computing is its characteristics
  - The differences between the various computers hardware and software
  - Way these systems communicate with each other
  - The entire architecture of a DS is hidden from the user
  - Users and applications can interact with a DS in a consistent and uniform way, irrespective of where and when the interaction takes place

Distributed Application Examples

- Automated banking systems
- Tracking roaming cellular phones
- Global positioning systems
- Passenger reservation system: railways and airlines
- Retail point-of-sale terminals in large malls
- Air-traffic control
- Avionics (fly-by-wire)
- Research Institutions
- The World Wide Web

Motivation for Distributed Systems

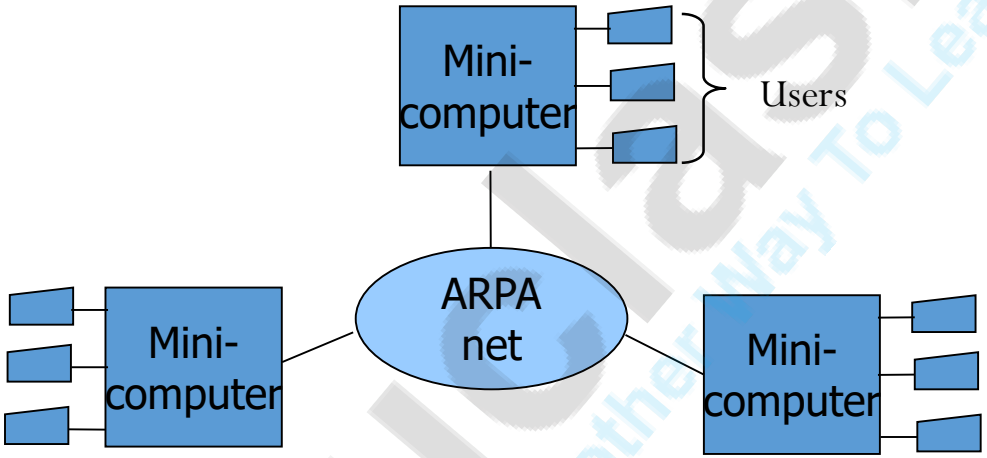
- Share resources
- Personalise environments
- Transparency
- Location independence
- People & information are distributed
- Performance & cost
- Modularity & expandability
- Availability & reliability

Disadvantages of Distributed Systems

- Software: difficult to develop software for distributed systems
- Network: saturation, lossy transmissions
- Security: easy access also applies to secret data

Distributed Computing System Models

Minicomputer Model

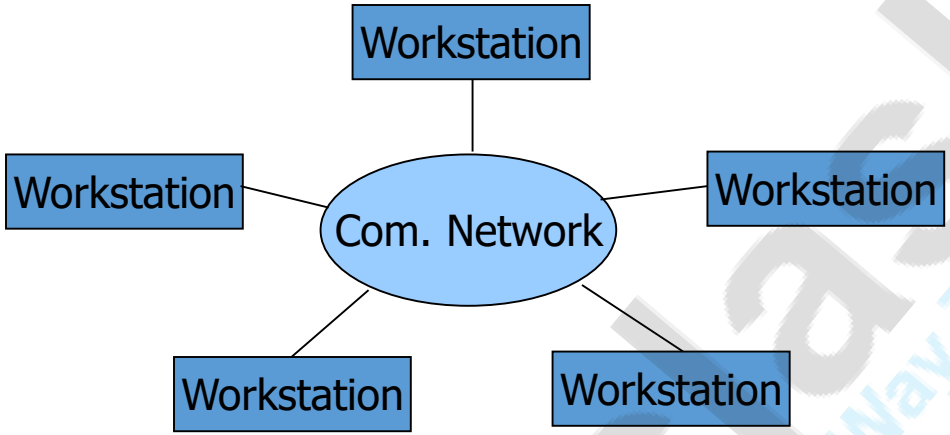


- Extension of Centralized Time sharing system
  - User must log on to his home minicomputer.
  - Thereafter, he can log on to a remote machine by telnet.
  - Does not reflect uniprocessor image.
- Used basically for Resource sharing (Database, High-performance devices)

Minicomputer Model (Cont'd)

- A DCS based on minicomputer may be a collection of large no. of minicomputers interconnected by fast communication network
- Each mini-computer may have multiple users logged on to it through interactive terminals
- The network allows the user to access remote resources on machines other than the one to which he is logged in
- The minicomputer model may be used when resource sharing (information databases of different types, with each type of database located on different machine) with remote users is desired
- Most of early research institutes computer network is based on minicomputer models e.g. ARPAnet(Advanced Research Project Agency Network).

Workstation Model

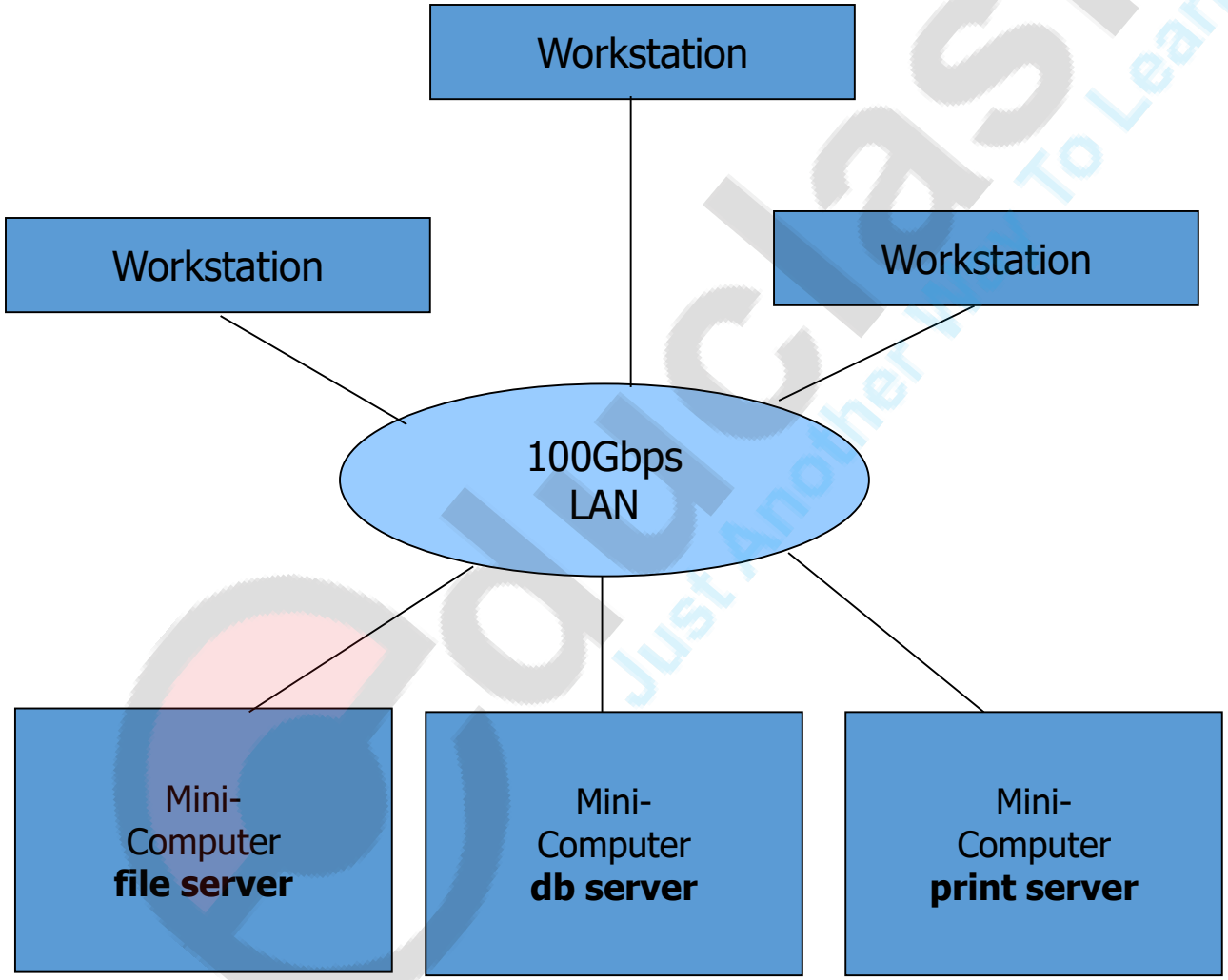


- Workstation
  - A powerful, single-user computer, like a personal computer, but has a more powerful microprocessor. Each has its own local disk and a local file system – diskfull workstation.
- A DCS based on workstations with a communication network is shown above.
- The concept is very useful in an office of university where there are many workstations which are idling most of the time
- To utilize these idle system came the Workstation model

Workstation Model (cont'd)

- Process migration
  - Users first log on to his/her personal workstation.
  - If there are idle remote workstations, one or more processes are migrated to one of them.
  - Result of execution migrated back to user's workstation
- Issues to be resolved: (will be covered later in detail)
  - How to find an idle workstation
  - How to migrate a job
  - What if a local user logs on to the machine executing process of another machine – run two processes simultaneously, kill remote process, migrate process back to its home workstation ?
- Examples – TIFR, Sprite System, Xerox PARC

Workstation-Server Model



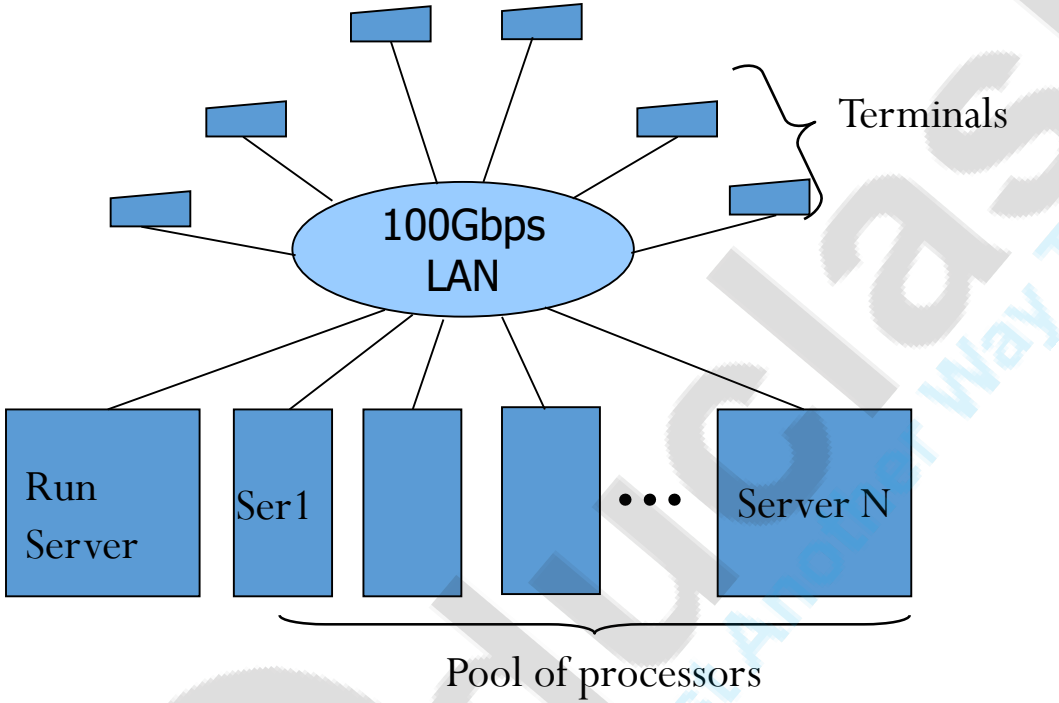
Workstation-Server Model (Cont'd)

- Client workstations
  - Largely Diskless
  - The file systems used by these systems must be implemented on diskful workstation or minicomputer equipped with a disk for file storage
  - Local disk of diskful workstation used for storage of temporary files etc.
- Server minicomputers
  - Each minicomputer is dedicated to one or more different types of services, for managing & providing access to shared resources.
  - Multiple servers used for a service for better scalability and higher reliability.
- User logs on to his machine. Normal computation activities carried at home workstation but file services are provided by special servers.
- No process migration involved. Request Response Protocol implemented

Workstation-Server Model (Cont'd)

- Advantages
  - Cheaper – few minicomputers vs. large no. of diskfull workstations
  - Backup and hardware maintenance easier
  - Flexibility to access files from any file server
  - No process migration
  - Guaranteed response time
  - Client-Server model of communication
- Disadvantage
  - Does not exploit idle workstations
- Most widely used model for building distributed systems
- Example: TELCO Design Dept, ADA & ADE

Processor-Pool Model



- Processors (microcomputers and minicomputers) are pooled together to be shared by the users as needed.
- Each processor has its own memory to load and run a system program or an application program of the DCS.

Processor-Pool Model (Cont'd)

- Clients:
  - They log in one of terminals (diskless workstations or graphic terminals)
  - All services are dispatched to servers.
- Servers:
  - Necessary number of processors are allocated to each user from the pool by run server
- No concept of home machine. User logs on to system as whole.
- Better utilization of processing power
- Greater flexibility – processors can act as extra servers
- Unsuitable for high performance interactive application as communication is slow between processor & terminal - less interactivity
- Example – Amoeba, Cambridge Distributed System

Hybrid Model

- The Workstation-Server model is the most widely used model for building distributed computing systems
  - Because large no of users at any given time are performing simple tasks like editing files or performing some database query
- Where users require heavy computational facility, Processor-Pool model is more advantageous
- Hence the Hybrid model, that combines advantage of both the workstation – server and processor - pool model
- Based on workstation – server model with additional pool of processors for high computational jobs
- The processor in the pool can perform large computations
- WS-server model can perform user interactive jobs.
- Hybrid model is more expensive to implement

Features of Distributed Computing Systems

- From the discussion of the models for DCS, it is clear that DCS are much more complex and difficult to build than traditional centralized systems
- The system software of a DCS should also be capable of handling communication and security issues, which are very simple in centralized systems
- Dependence on communication network is very high
- Widely distributed resource sharing brings its own set of constraints and privacy issues
- Still DCS is becoming popular Why?

1. Inherently Distributed Applications
   - Several applications are distributive by nature and hence require a DCS

Features of Distributed Computing Systems

- For example Nationwide Employee Data base, maintained at each branch office (Aadhar)
- Hence DCS provides the necessary features for collecting, preprocessing and accessing data from different distributed systems

2. Information Sharing among Distributed Users

- Need for efficient person to person communication, sharing of information over great distances
- Information generated by one user can easily be shared across the system irrespective of the geographic distances
- Such environment is also referred to as computer-supported co-operative working

Features of Distributed Computing Systems

3. Resource Sharing

- In addition to data sharing DCS also supports extensive resource sharing as software libraries, databases as well as hardware devices printers, hard disk, plotters and other very exclusive peripherals in an effective manner

4. Better Price Performance Ratio

- As the resources are very effectively and efficiently used and shared, it automatically reduces the cost of ownership of the entire network
- By effective sharing of computing power and other resources, it provides a better price to performance ratio

5. Shorter Response time and Higher Throughput

- Because of the multiplicity of the processors, it will have on the whole better performance than a single processor centralized systems

Features of Distributed Computing Systems

- It is possible to design multithreaded application, so that a single application can run on multiple CPUs at the same time to give better throughput
- The performance of the DCS is normally only restricted by the communication network on which it is designed

6. Higher Reliability

- Reliability refers to degree of tolerance against errors and component failures in a system
- The multiplicity of storage devices and processors supports multiple copies of critical information within the network and execution of important computations redundantly to protect them against catastrophic failures
- Another aspect of higher reliability is high availability of resources
- However one should remember that reliability comes at the cost of performance

Features of Distributed Computing Systems

7. Extensibility and Incremental Growth

- The DCS is very easily capable of incremental growth. i.e., gradually increase power and functionality of a DCS
- Additional CPUs, peripherals can easily be integrated to address future demands

8. Better Flexibility

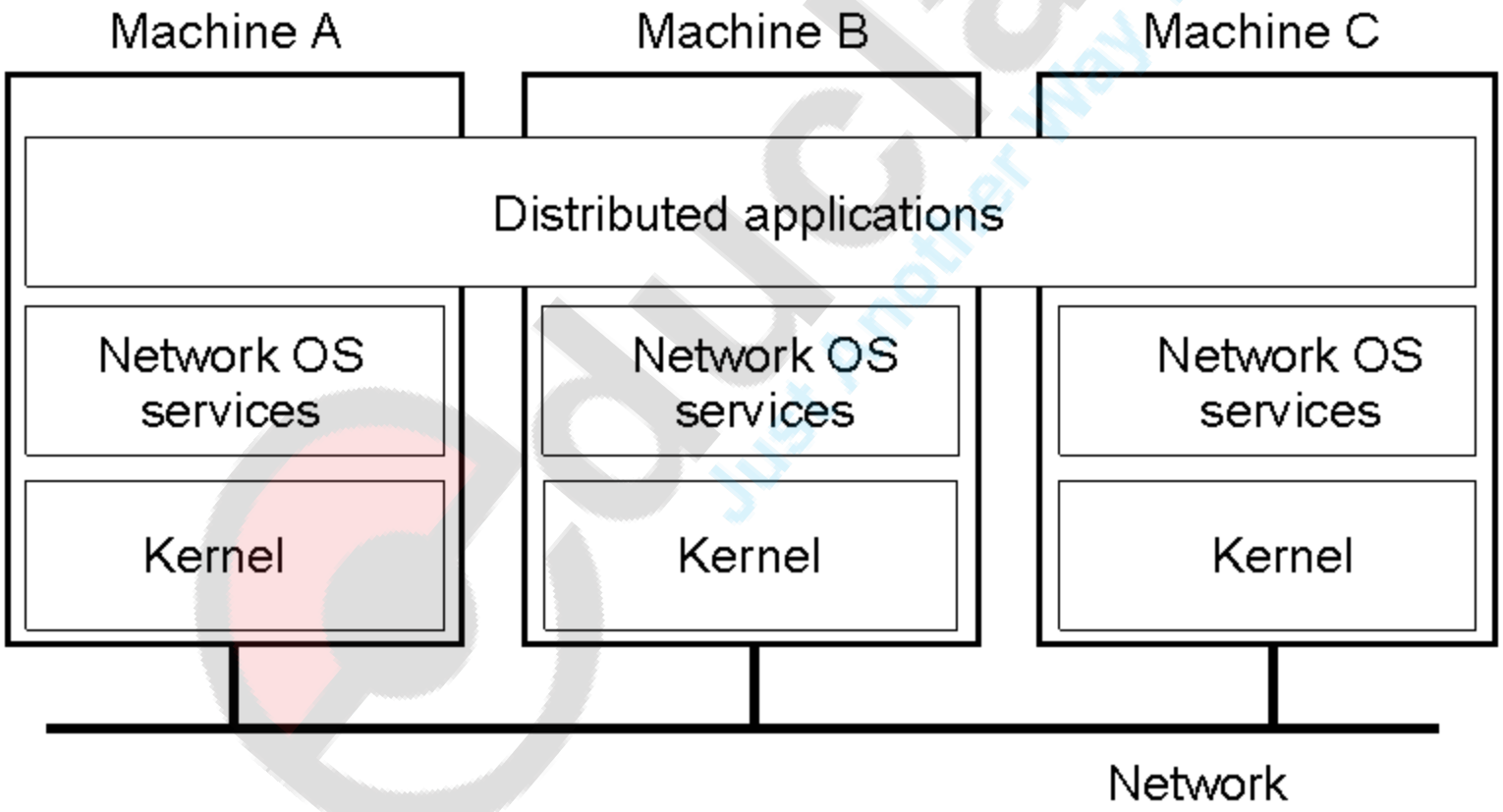
- Different types of computers can be integrated to address different computational needs. For example, Systems for data storage have completely different specifications when compared to a computationally intensive system
- As the DCS is based on hybrid model, the different data processing requirements can be very effectively addressed in a cost effective manner

Complexities

- More Software Components
  - The more software components that comprise a system, greater is the chance of errors occurring
- Security
  - Providing easy distributed access increases the risk of a security breach occurring
- Networking
  - The underlying network can saturate or cause other problems.
- Complexity
  - Complete information about system environment never available

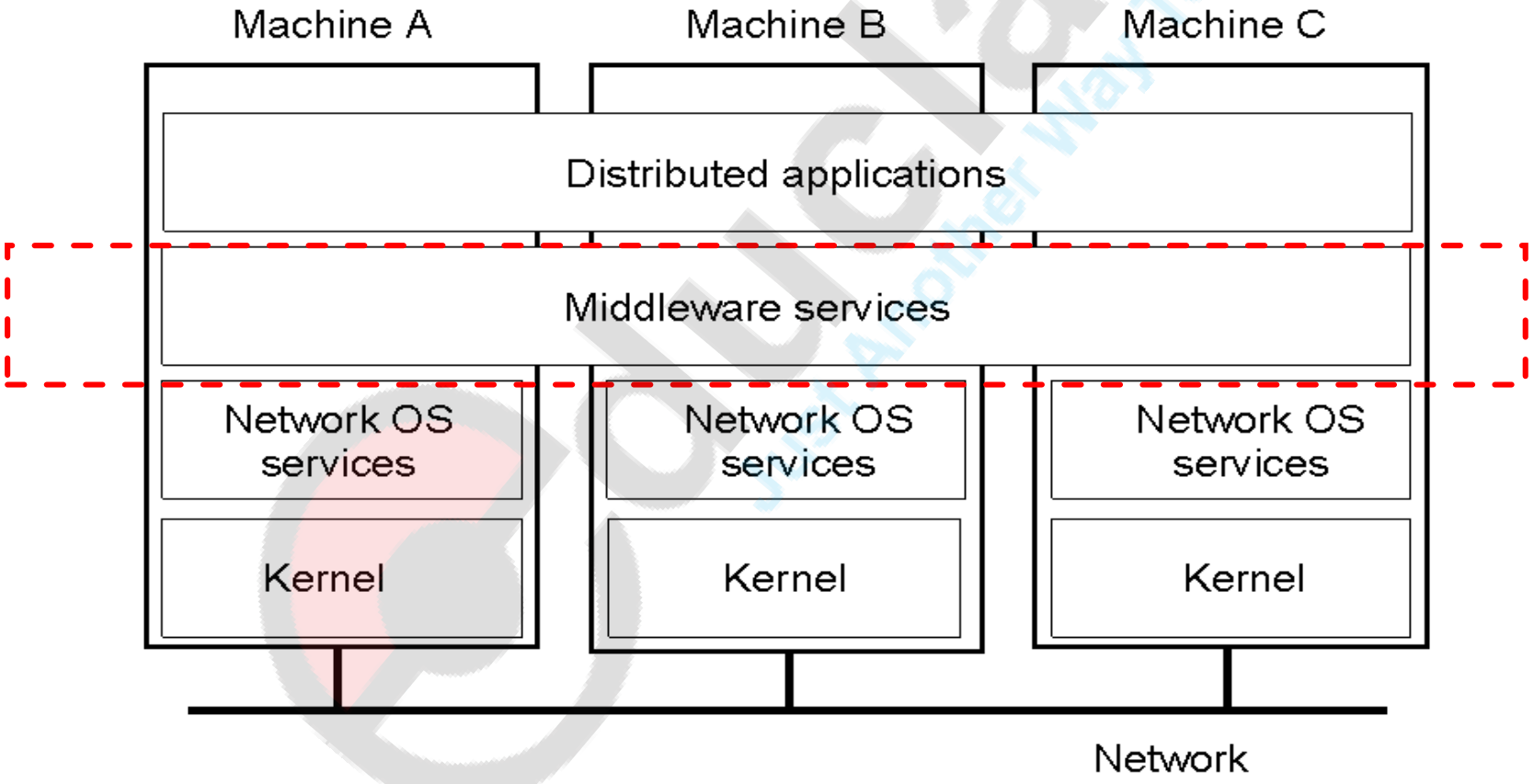
Network Operating System

General structure of a network operating system



Positioning Middleware

General structure of a distributed system as middleware



NOS vs DOS

- Three most commonly used features to differentiate NOS and DOS are System Image, autonomy, and fault tolerance capability
  - System Image
    - In case of NOS, the users are aware of the fact that multiple computers are in use connected by high speed network
    - DOS on the other hand hides the existence of multiple computers and provides a single system image to its users
    - In NOS user need to know location of a resource to use it
    - IN DOS user uses same set of system calls to access the local as well as remote resources
    - Hence key differentiator is the transparency

NOS vs DOS (Cont'd)

- Autonomy
  - NOS is built on a set of existing centralized OS and handles the interfacing and coordination of remote operations and communication between the operating system
  - In NOS each computer has its own unique operating system and the only coordination is for the communication between two processes on different systems
  - Each computer functions independently and take independent decisions about the creation and termination of their own processes and management of local resources
  - DOS on the other hand, have a single system wide operating system and each computer of the DCS runs a part of this global operating system
  - DOS tightly interweaves all the computers of a DCS so that they work in close cooperation with each other for efficient and effective utilization of the various resources of the system
  - Hence autonomy is higher in NOS than in DOS

NOS vs DOS (Cont'd)

- Fault tolerance Capability
  - NOS provides very little or no fault tolerance capability
  - DOS ensures that users are unaffected by the failure of a machine in a DCS
  - Hence DOS offers high level of fault tolerance when compared to NOS
- Hence a DCS that uses NOS is normally referred to as Network System, while one that used DOS is normally referred to as True Distributed System or just Distributed System

Comparison of Different OS

Criteria	Network OS	Distributed OS	Multiprocessor time-sharing OS
Projects a virtual uniprocessor image	No	Yes	Yes
Runs same operating system	No	Yes	Yes
Copies of operating system	*N*	*N*	1
Access to files	Sharing	Messages in memory	Sharing
Network protocols required	Yes	Yes	No
Single run queue	No	No	Yes
Well-defined file sharing	Usually no	Yes	Yes
Coupling of Hardware & Software	**LHLS**	**LHTS**	**THTS**

ISSUES IN DESIGNING A DISTRIBUTED OPERATING SYSTEM

Transparency

- An important goal of a DCS is to hide the fact that its processes and resources are distributed across multiple computers
- A Distributed System is able to present itself to users and applications as if it were only a single computer system.
- The concept of transparency can be applied to several aspects of a distributed system
  - Access, Location, Migration, Relocation, Replication, Concurrency, Failure and Persistence

Transparency in a Distributed System

- Access Transparency
  - Hide differences how a resource(local or remote) is accessed. Use global set of system calls & global resource naming facility (ex URL)
  - Accessing methods for a local and a remote resource should be same
- Location Transparency
  - Hide where a resource is located
  - Name transparency: Name of resource should not reveal its physical location.
  - Resource names must be unique system wide, so that they can be moved easily across systems
  - User Mobility: User should be able to freely log on to any machine in the system and access a resource with same name

Transparency in a Distributed System

- Replication Transparency
  - Naming of replicas – map user supplied name of resource to appropriate replica (additional copies) of files and other resources on different nodes of a DCS
  - Replication control – issues are how, where, and when
- Failure Transparency
  - Deals with masking the user from partial failures in the system such as, communication link failure, a machine failure, or a storage device crash.
  - Partial failure transparency
  - Complete failure transparency
- Migration Transparency
  - Movement of object is handled automatically by system and following issues are taken care of :

Transparency in a Distributed System

- Migration decision made automatically by system
- Name of resource remains same on migration from one node to another
- IPC ensures proper receipt of message by process, even if it further migrates

- Concurrency Transparency
  - Hide that a resource may be shared by several competitive users. It is achieved by
    - Event ordering property
    - Mutual exclusion property
    - No starvation property
    - No deadlock property

- Performance Transparency
  - System is automatically reconfigured as per load varying in the system.
- Scaling Transparency
  - System can expand in scale without disrupting activities of users
  - Open system architecture and scalable algorithms

Reliability

- In general DCS is expected to be more reliable than a Centralized system because of multiple instances of resources
- Their presence alone is not enough the DOS has to be designed to fully utilize these resources effectively
- Faults
  - A fault: a mechanical or algorithmic defect that may generate an error
  - A fault: in a system causes system failure
  - The system failure: are of two types
    - Fail stop : system stops functioning after changing its state in which its failure can be detected
    - Byzantine failure : System continues to function but produces wrong results
      - Undetected software bugs often cause byzantine failure of a system and
      - More difficult to detect and deal with than fail stop failure
  - The DOS must be designed properly to avoid faults, tolerate faults, and to detect and recover from faults

Reliability (Cont'd)

- Fault Avoidance
  - Occurrence of faults is minimized by utilizing reliable components in the design of the system
  - Conservative design practices increases the system reliability
  - Though DOS has no control over the hardware utilized, it can thoroughly test the systems for better reliability
- Fault tolerance
  - The ability of the system: to continue functioning in the event of partial system failure, may be with degraded performance, e.g, a memory card failure
  - Redundancy techniques
    - Avoid single point failure by replicating critical hardware and software components
    - For example RAID
    -

Reliability (Cont'd)

-
  - Hence larger the replication higher is the reliability
  - Increases the system overhead and cost of ownership
- Distributed control
  - Avoiding single point of failure by distributed control mechanism
  - For example a highly available Distributed File System should have multiple file servers.
- Fault detection and recovery
  - Atomic transaction: A collection of operations which can not be separated. i.e., either all or none of the actions are effective
    - This enables crash recovery much easier and
    - Maintains data integrity
    - Provides the facility for roll back of transactions

Reliability (Cont'd)

- State full & Stateless servers
  - The client server model is frequently used in DCS to service user requests
  - Two service paradigms namely stateful or stateless
  - They are distinguished by whether or not the history of the serviced requests between a client and a server affects the execution of the next request
  - The stateful approach does depend on the history of the serviced requests while stateless servers does not depend on it
  - Stateless service paradigm makes crash recovery simple as no client state information is maintained by the server
  - The stateful service require complex crash recovery process as both client as well as the server detect crashes efficiently and reliably

- Acknowledgements & timeout based retransmissions of messages
  - Failure of communication link or a node can result in loss of messages
  - Hence a reliable interprocess communication mechanism need to be implemented
  - Receiver must send ACK for a message, if not received within fixed time limit, assume loss of message and retransmit
  - A problem associated with this method is duplicate messages and the receiver should have a mechanism to detect and handle them (auto generation of sequence nos.)

Flexibility

- Flexibility is an important issue in the design of open distributed system they should be flexible for the following reasons
  - Ease of modification
    - To address design bug discovered later
    - Changed system environment
    - New user requirements
    - Incorporation of changes with minimum interruption to the users
  - Ease of enhancement or upgradation
    - Addition of new functionalities to make a DCS more powerful and easy to use
  - An important design factor is the kernel model. This part of the OS operates in a separate address space and can not be accessed by the user processes

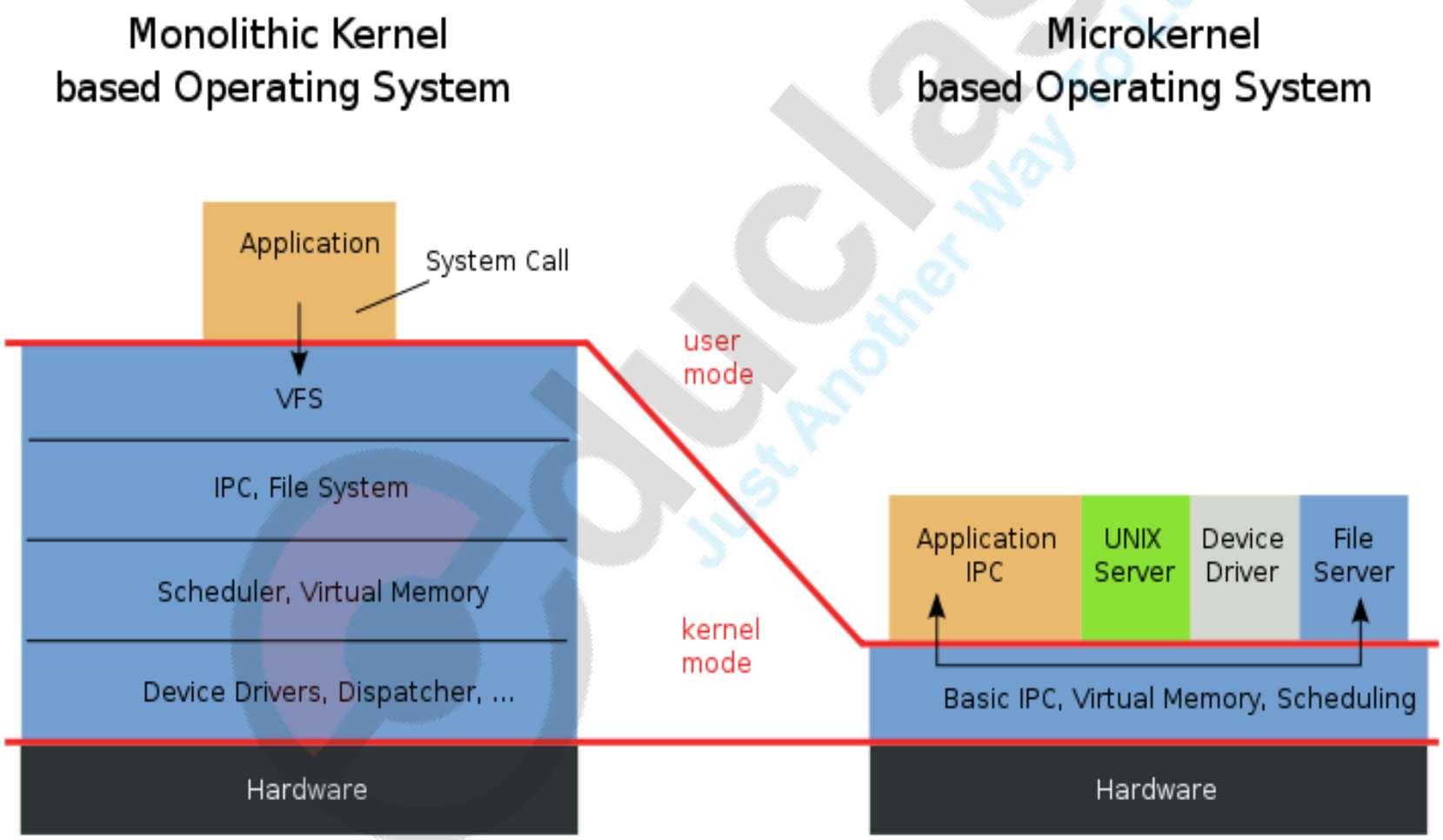
Flexibility (cont'd)

- Choosing appropriate kernel: two commonly used models
  - Monolithic kernel : Kernel where the entire operating system is working in the kernel space. The function include process management, memory management, device management, file management, name management, inter process communication etc
    - Many operating systems based on UNIX uses monolithic structure
  - Micro kernel : Kernel is reduced to contain minimal facilities necessary
    - Only services provided by the kernel is IPC, low level device management and minimum level of process management and some memory management

Flexibility (cont'd)

- Other system services such as file management, name management, additional process and memory management and most of system call handling are implemented in user space in form of server processes.
- Compared to monolithic, microkernel model has several advantages
  - Easy reconfiguration of the operating system.
  - Highly modular in nature and easily modifiable.
  - Easy to design, implement and modify.
  - It also provides flexibility to user for implementing his own services.

Monolithic kernel vs. Micro kernel



- Addition of new services does not require rebooting of kernel as in the case of monolithic kernel
- Penalty for all these advantages is the performance
- As each service is an independent process with its own address space, need to have special message based IPC between processes while performing a job
- Still microkernel is the preferred model.

Performance

- Performance should be as good as centralized system.
- Various performance metrics:
  - response time
  - throughput
  - system utilization
  - network capacity utilization
- Design issues to increase performance
  - Batch if possible:
    - transfer data across the network in large chunks is more efficient,
    - piggybacking of acknowledgements with the next message etc.
  - Cache whenever possible:
    - at the clients' sites, improves performance as it provides data immediately where it is needed
    - It also reduces contention on centralized resources

Performance (Cont'd)

- Minimize copying of data:
  - Data copying overhead
  - moving data in and out buffers
  - Involves multiple CPU operations and hence a substantial overhead
- Minimize network traffic (Internode communication overhead)
  - Migrating processes closer to resources required
  - Clustering of processes that communicate heavily onto single system
  - Unnecessary collection of Global information as routine procedure
- Fine grain parallelism (involve large no. of small computations but more interaction)
  - Multiprocessing and multi threading
- vs. coarse grained parallelism (involve large computations)

Scalability

- Capability of a system to adapt to increased service load
  - It is inevitable that DCS will grow with time to address increased work load or new applications or organizational changes in a company
  - Hence a good DCS should be able to handle growth of nodes, users and applications
  - Avoid centralized entities: like central file server, single database for entire system makes the system non-scalable
    - Failure of the centralized entity often brings the entire system down
    - Performance of centralized entity becomes bottleneck of performance
    - Increases network overhead in a wide area network
    - Replication of resources and distributed control algorithms are frequently used to overcome centralized entities
    - In fact for better scalability, as far as practicable, a functionally symmetric configuration should be used where all the nodes have nearly equal role to play in the operation of the system

Scalability (Cont'd)

- Avoid centralized algorithms
  - Operates by collecting information from all nodes, processing this information on a single node and then distribute the results to other nodes.
  - For example clock synchronization is some thing never implemented on a DCS
- Perform most operations on client workstations
  - If possible an operation should be performed on the clients own W/S rather than on a server machine
  - Server being shared by several clients, its cycles are more important than the client system cycles
  - This principle enhances the scalability of the DCS
  - Caching is a frequently used technique to achieve this principle

Heterogeneity

- Caused by interconnected sets of dissimilar hardware or software systems (Ex, different topologies, protocols, word lengths etc)
  - Designing such DCS is far more difficult than designing homogeneous DCS using systems based on same or closely related hardware and software
  - Incompatibilities can be of different types, word length, byte ordering, communication protocols, topologies of networks, OS intricacies, etc.
  - Data and instruction formats depend on each machine architecture
  - One of the most common process is the use of intermediate standard data format.

Security

- In order to trust a DCS and rely on it, the various resources should be protected against destruction, and unauthorized access
- More difficult in a DCS when compared to a centralized system
- Difficult due to lack of a single point of control & use of insecure networks for data communication
- In a DCS, an intruder with malicious intensions can hide identification information leading to several Security concerns:
  - It should be possible for the sender of a message to know that the message was received by the intended user
  - The receiver of a message should be able to know that message was sent by the genuine sender
  - Messages are not stolen, plagiarized or changed by an intruder
  - Cryptography is the only known practical method for dealing with these security aspects of DCS

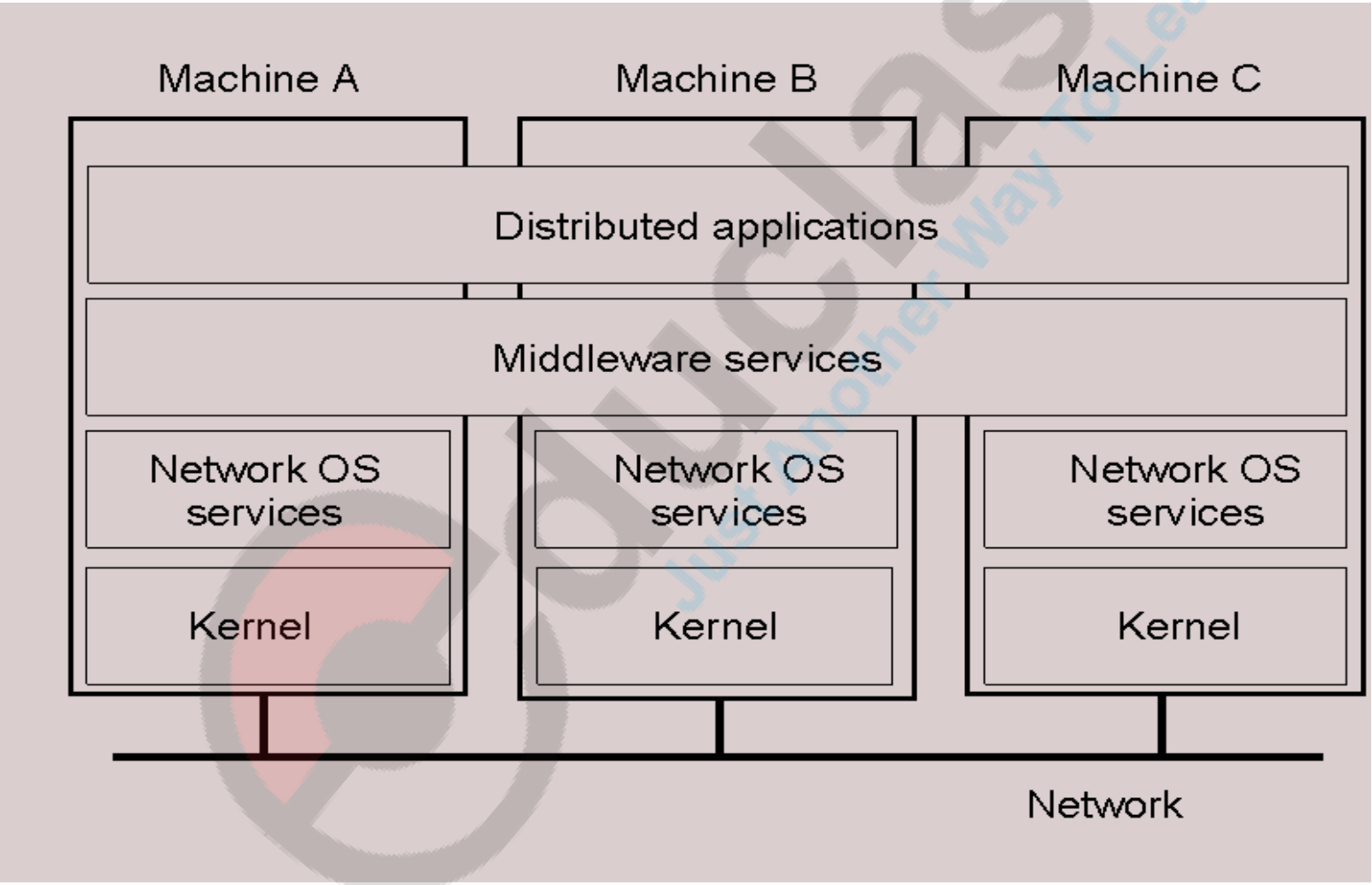
Emulation of Existing Operating Systems

- For commercial success, it is important that a newly designed DCS be able to emulate existing popular operating systems such as UNIX
- This is very necessary to be able to run a vast amount existing old software, with out rewriting them
- At the same time new applications can be developed taking full advantage of the new DCS
- This will ensure that both software can run side by side, in a seamless manner

Middleware

- Middleware is an additional layer of software that is used in NOS to more or less hide the heterogeneity of the collection of underlying platforms but also to improve distribution transparency
- It offers a higher level of abstraction.
- It is placed in the middle between applications & NOS.

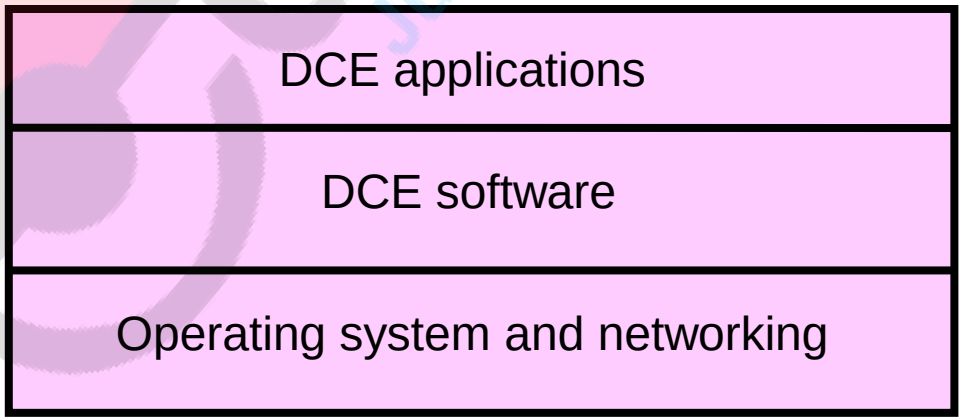
Distributed System as Middleware



Distributed Computing Environment (DCE)

Distributed Computing Environment (DCE)

- It is vender independent environment defined by the Open Software Foundation (OSF), a consortium of computer manufacturers including IBM, HP and DEC
- It is neither an OS nor an application
- Rather, it is an integrated set of services and tools that can be installed as a coherent environment on top of existing OS and serve as a platform for building and running distributed application



Distributed Computing Environment (DCE)

- A primary goal of DCE is vendor independence
- It runs on many different kinds of computers, OS, and network hardware manufactured by different vendors
- Some of the operating systems to which DCE can be easily ported are OSF/1, AIX, ULTRIX, HP-UX, IRIX, SINIX, SUNOS, SOLARIS, UNIX System V, VMS, Windows, etc.
- It can also be used with any network hardware and transport software including TCP/IP, X.25 and other similar products
- It hides differences between machines by automatically performing data-type conversions, thus making heterogeneous nature of system transparent to application programmers

How Was DCE Created

- OSF did not create DCE from scratch
- It created DCE by taking advantage of the work already done at universities and industries in the area of distributed computing
- OSF issued a Request for Technology (RFT) asking for tools and services needed to build a coherent DCE
- From this a group of experts selected those tools and services which the committee believed provided the best solutions
- Based on this OSF further developed and wrote the code almost in C to produce a single integrated package that was available to the world as DCE
- Version 1.0 of DCE was released in 1992

DCE Components

- DCE is a blend of various technologies, nicely integrated by OSF
- Each of these technologies forms a component of DCE
- Some of the main components are:

1. Threads Package: It provides a simple programming model for building concurrent applications
   - It includes operations to create and control multiple threads of execution in a single process and to synchronize access to global data within an application
2. RPC facility: It provides the programmer with number of powerful tools necessary to build client-server applications
   - In fact DCE-RPC facility is the basis for all communication in DCE as the programming model underlying all of DCE is the client-server model

DCE Components(cont'd)

- It is easy to use and is network and protocol independent, providing secure communication between client and a server

3. Distributed Time Service: It closely synchronizes the clock of all the computers in the system

   - It also supports external time sources to synchronize the clocks of the computers

4. Name Services: The name services of DCE include the Cell Directory Services (CDS), The Global Directory Service (GDS) and the Global Directory Agent (GDA)

   - These services allow resources such as servers, files, devices and so on, to be uniquely named and accessed in a location transparent manner

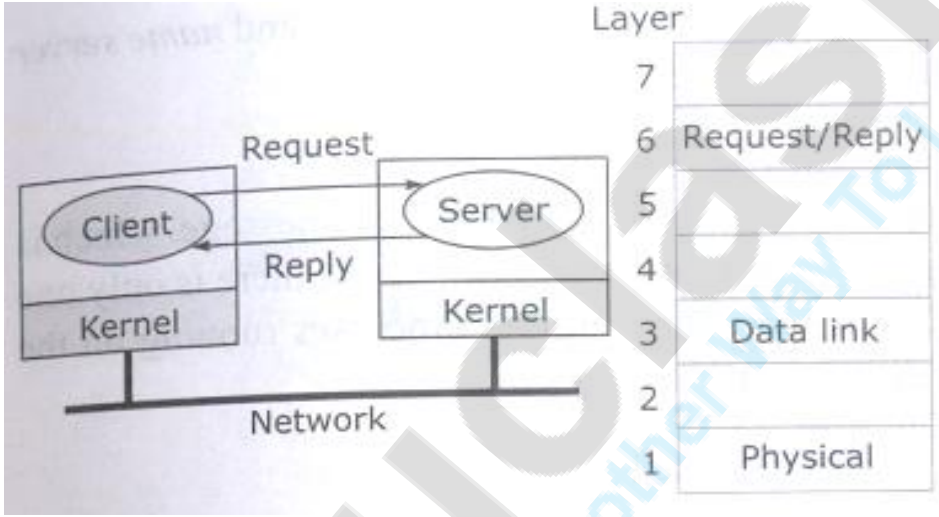
DCE Components(cont'd)

5. Security Service: It provides tools needed for authentication and authorization to protect system resources against illegitimate access
6. Distributed File Service (DFS): It provides a system wide file system that has such characteristics as location transparency, high performance, and high availability

- All DCE services run in coordination with each other.

Client –Server Model

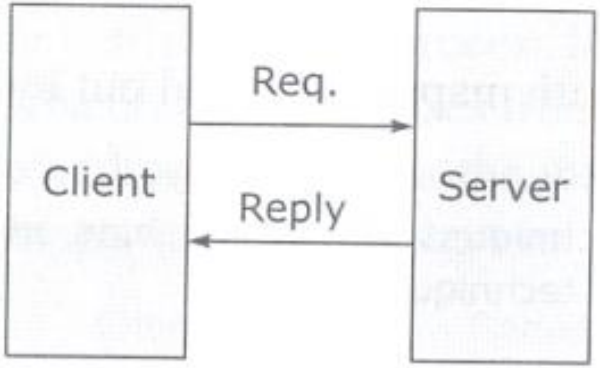
- Group of cooperating services called servers offers services to the users called clients.



- As shown above, request-response protocol is used.
- As only three layers are basically involved in any communication, less functionalities are required.
- Data link and physical layer are involved in transferring packets between the client and server.
- Request – Reply Protocol (Presentation Layer) defines set of requests and replies to the corresponding requests.
- Other layers are not required.

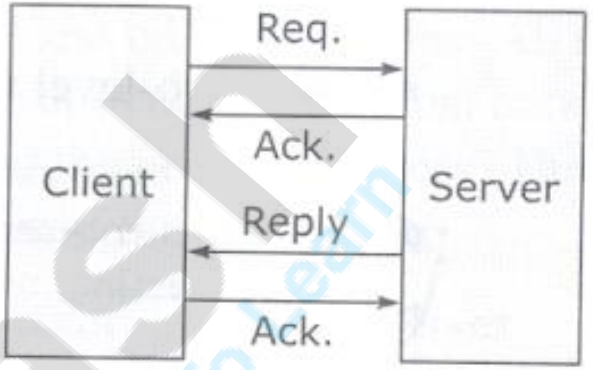
Client – Server Implementation

- All networks have maximum packet size.
- Packet broken into smaller packets, if large.
- Reliability of message passing is ensured by receiving acknowledgements.
  - Acknowledgement can be sent for all packets of a message.
  - Or it could be sent just once with the last packet.
  - This reduces network traffic.
- Different types of packets are: REQ, Reply, ACK, AYA, IAA
- Levels in client-server architecture:
  - User Interface Level
  - Processing Level
  - Data Level
- In a DS, all nodes can act as Server or Client.



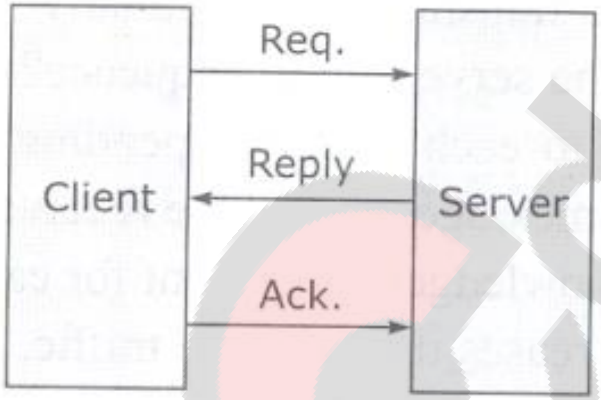
(a)

The client–server system uses a Request–Reply protocol with no acknowledgement.



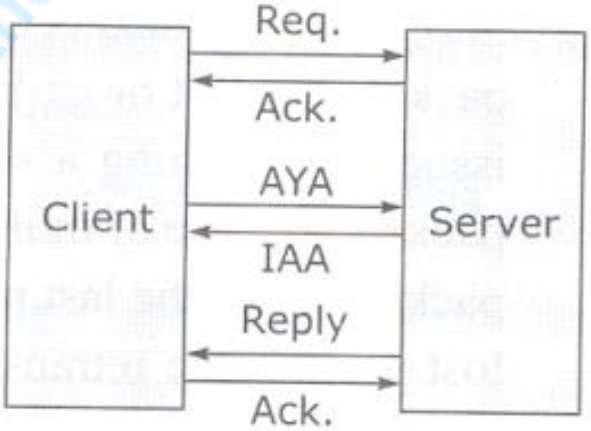
(b)

Each packet is acknowledged separately.



(c)

A reply is sent as acknowledgement, so the number of packets is reduced.



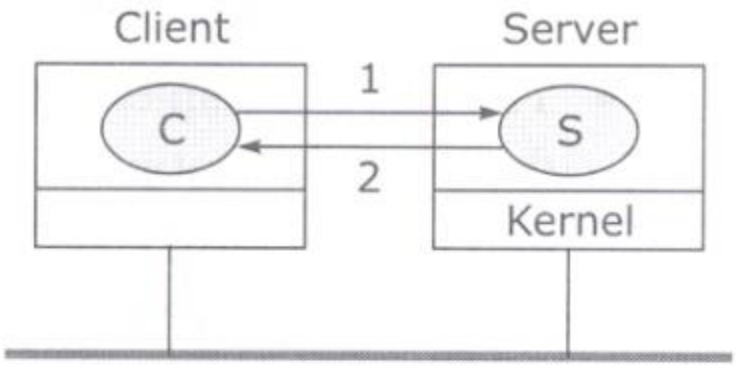
(d)

The client checks if the server is still alive and on the network.

Client – Server Addressing

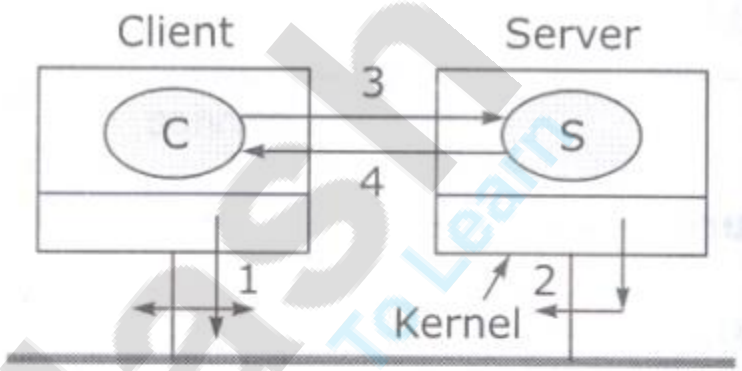
- Address required for message communication.
- Three main addressing techniques are:
  - Machine Addressing –
    - In this scheme, machine address is sent by client as part of message.
    - This method works well when only one process is running on server, otherwise, process id also need to be included.
  - Process Addressing –
    - Client sends message to process instead of machine.
    - A two-part name comprises of machine ID and process ID.
    - Process ID , if not unique, wont be transparent.
    - To implement transparency, unique addresses allocated with the help of counters.
    - Message is usually broadcasted, machine identifies its process name and receives message.

- Name Server Technique
  - Uses name server, as, broadcasting is overhead.
  - The process addressing techniques are as follows:
    - Name server address is hardwired into clients.
    - For communication, process name is sent to name server.
    - Name server replies with the address of machine where process is located.



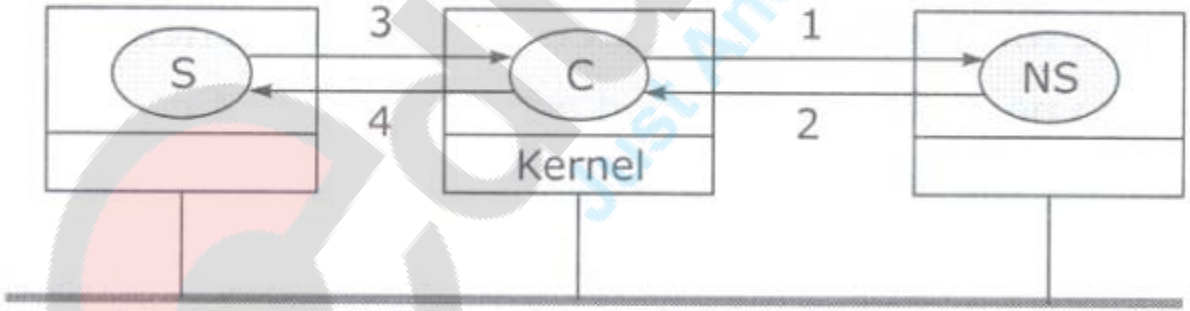
1: Request to client
2: Reply to server

(a) Machine addressing



1: Broadcast 2: Give own location
3: Request 4: Reply

(b) Process addressing



1: Lookup in name server 2: Reply from NS
3: Request 4: Reply

(c) Name server technique

University Questions from last 5 years

- Distributed System Models (Short Notes)
- Suppose a component of a distributed model suddenly crashes. How will this inconvenience the users when one of the following happens:-
  - The system uses processor-pool model and crashed component is a processor in the pool.
  - In processor-pool model , a user terminal crashes.
  - The system uses a workstation-server model and server crashes.
  - In the workstation-server model , one of the client crashes.
- Distributed Operating System and Issues in designing a distributed operating system.
- Short note on DCE.