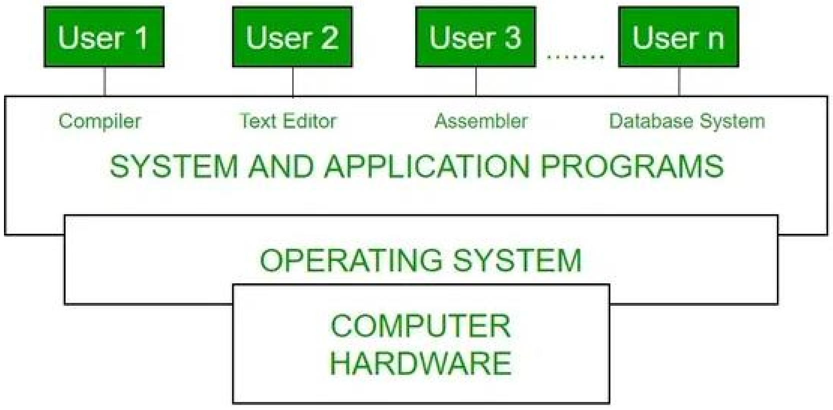
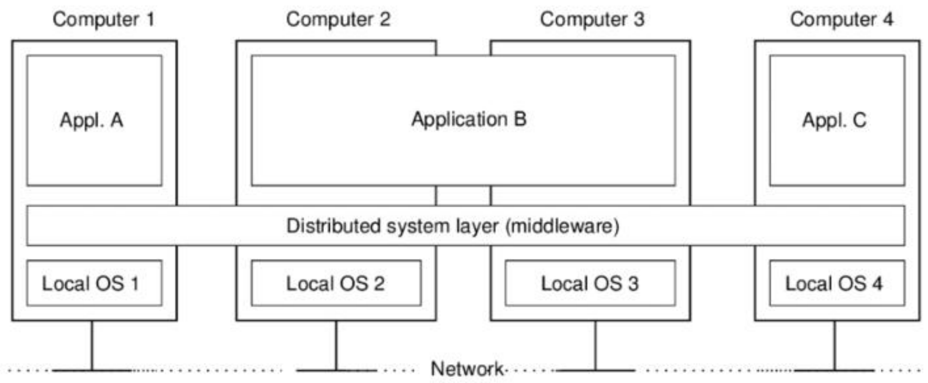
Chapter 3
Operating System Support

The Operating System Layer

- An Operating System (OS) is an interface between a computer user and computer hardware i.e OS acts as interface between application program and machine hardware.
- An Operating System (O.S.) is a system software that manages the hardware and software resources and provides common set of services to the application software.
- An operating system is a software that performs all the basic tasks like: file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.



- The OS layer is present below the middleware layer. OS provides problem-oriented abstractions of the underlying physical resources.
- The middleware - OS combination of a distributed system should have good performance.

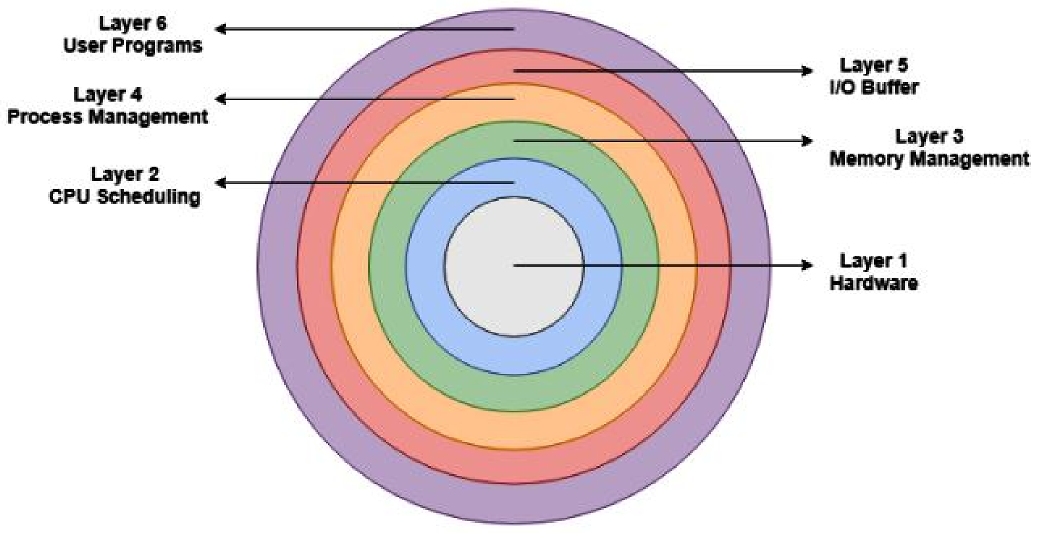


- Middleware is responsible to provide proper utilization of local resources to implement mechanisms for remote invocations between objects or processes at the nodes. Kernel and server processes are responsible to manage resources and present clients with an interface to the resources.
- Examples: Windows, Linux, UNIX, MAC OS, etc.
- <u>Clients access resources by making invocations to the server object or system calls to the kernel.</u>

- Kernel, as a resource manager, must provide encapsulation, protection and concurrent processing.

OS Functionality (Assignment)

I. Processs Management
II. Thread Management
III. Communication Management
IV. Memory Management
V. Supervisor Functionality
VI. File Management, etc.



Layers of operating system

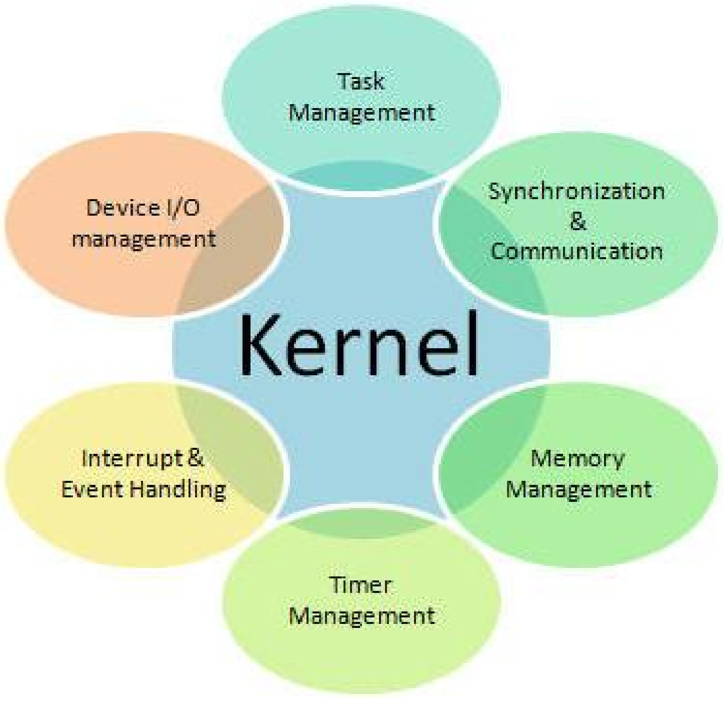
Protection

- The underlying resources in a distributed system should be protected from illegitimate access.
- To ensure protection of resources, operating system must provides a means to provide clients to perform operations on the resources if and only if they have rights to do so.
- For example: Consider a file with only read or write operations. The illegitimate access would be access of file for write by the client who have read access only. Resources can be protected by the use of type-safe programming language like JAVA in which no module can access a target module without having a reference to it.
- Hardware support can be employed to protect modules from illegitimate invocations, for which kernel should be implemented.

Kernel

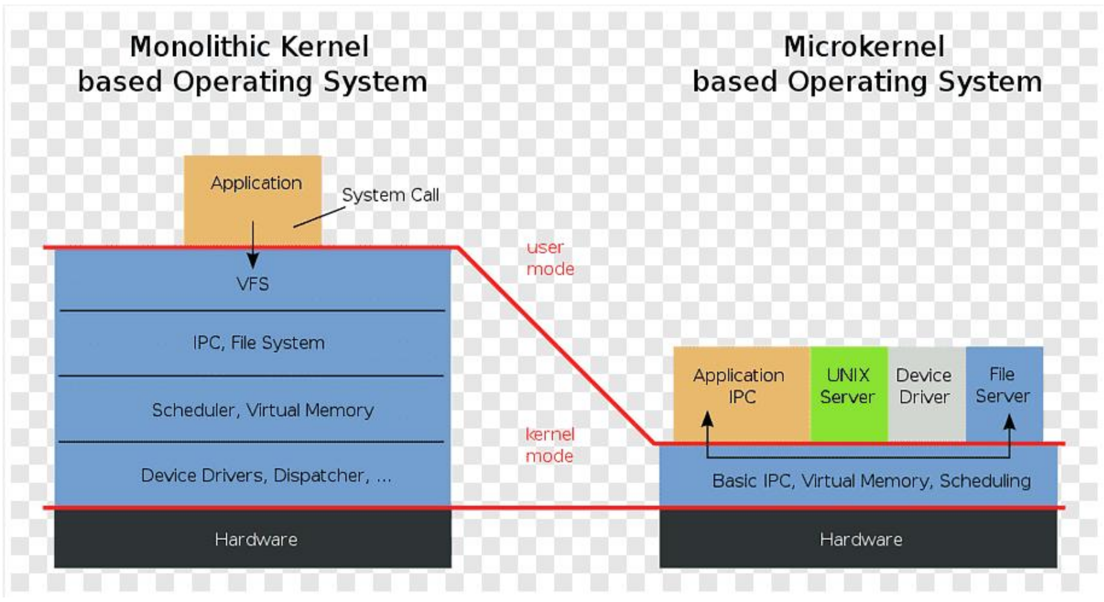
- A kernel is a program that is executed with complete access privileges for the physical resources on its host computer.
- It controls memory management.
- It ensures access of physical resources by the acceptable codes only.

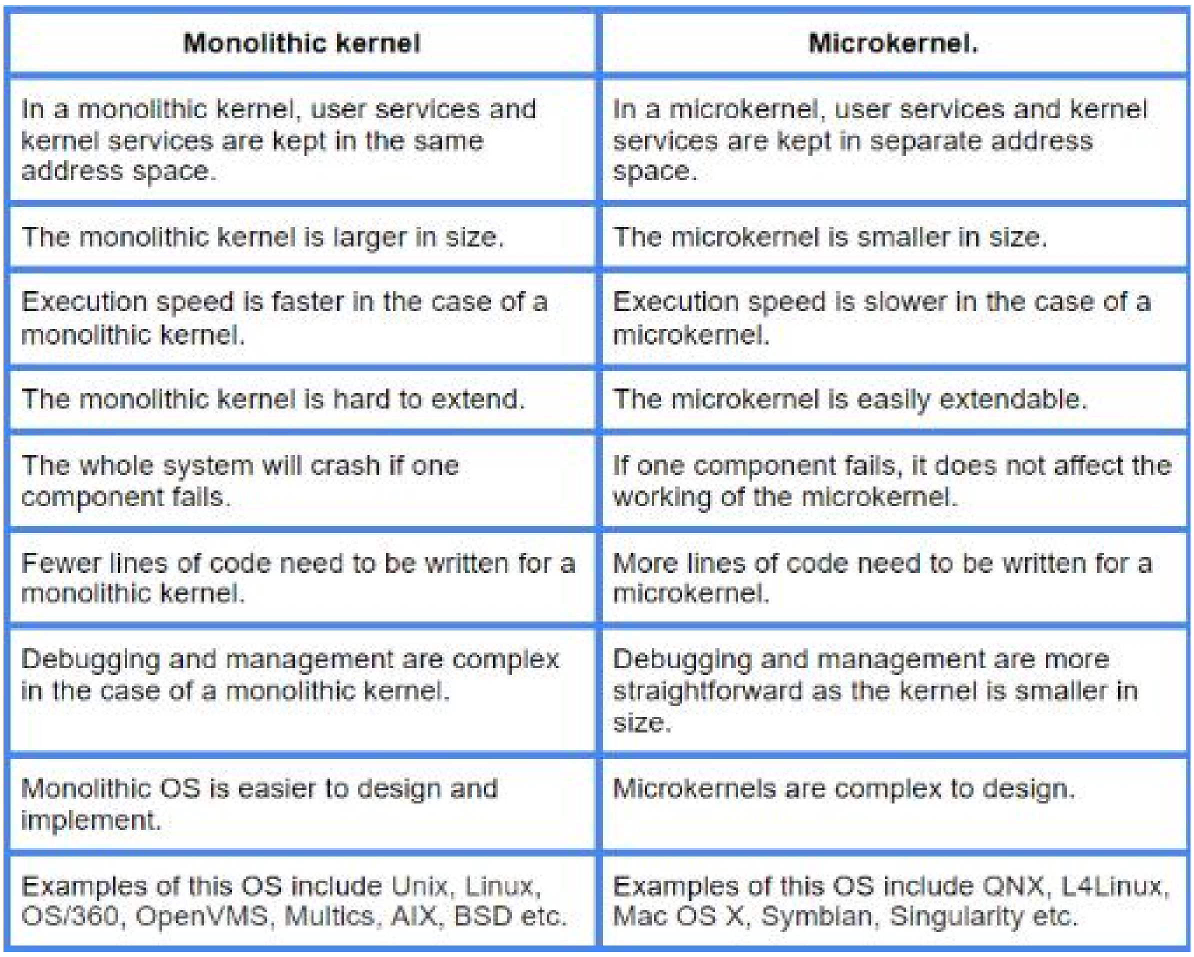
- A kernel process executes in supervisor mode and restricts other processes to execute in user mode.
- It sets up address space for the process. A process is unable to access memory outside its address space. A process can switch the address space via interrupt or system call trap.



Types of Kernel

1) Monolithic Kernel
2) Micro Kernel


Monolithic Kernel based Operating System
Application
System Call
VFS
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
Hardware
user mode
kernel mode
Microkernel based Operating System
Application IPC
UNIX Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling
Hardware



Monolithic kernel	Microkernel.
In a monolithic kernel, user services and kernel services are kept in the same address space.	In a microkernel, user services and kernel services are kept in separate address space.
The monolithic kernel is larger in size.	The microkernel is smaller in size.
Execution speed is faster in the case of a monolithic kernel.	Execution speed is slower in the case of a microkernel.
The monolithic kernel is hard to extend.	The microkernel is easily extendable.
The whole system will crash if one component fails.	If one component fails, it does not affect the working of the microkernel.
Fewer lines of code need to be written for a monolithic kernel.	More lines of code need to be written for a microkernel.
Debugging and management are complex in the case of a monolithic kernel.	Debugging and management are more straightforward as the kernel is smaller in size.
Monolithic OS is easier to design and implement.	Microkernels are complex to design.
Examples of this OS include Unix, Linux, OS/360, OpenVMS, Multics, AIX, BSD etc.	Examples of this OS include QNX, L4Linux, Mac OS X, Symbian, Singularity etc.

- ➢ Micro kernel is suitable for distributed operating system.
- ➢ The services of distributed system are complex and segregation of micro kernel makes it easy.
- ➢ Micro kernel provides faster communication among the processes with low overhead.

Deadlock in Distributed System

A deadlock is a condition in a system where a set of processes (or threads) have requests for resources that can never be satisfied. Essentially, a process cannot proceed because it needs to obtain a resource held by another process but it itself is holding a resource that the other process needs.

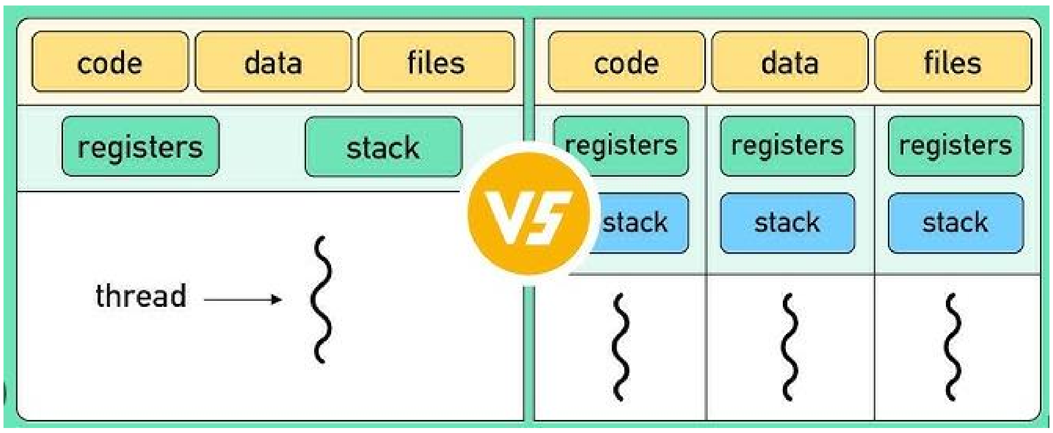
4 Conditions of deadlock:

- ➢ Mutual exclusion- A resource can be held by at most one process.
- ➢ Hold and wait- Processes that already hold resources can wait for another resource.
- ➢ Non-preemption- A resource, once granted, cannot be taken away.
- ➢ Circular wait- Two or more processes are waiting for resources held by one of the other processes

Process and threads

- Process is an instance of a computer program that is being executed.
- The program in execution is known as process.
- It consists of code and its activity.
- Process Control Block (PCB) controls the operation of any process. PCB contains information about processes. For example: process priority, process id, process state, CPU, register, etc.
- The process can have the following states like new, ready, running, waiting, terminated, suspended.

- In traditional OS, each process has an address space and a single thread of control.
- The distributes systems require internal concurrency for which process is multi-threaded.

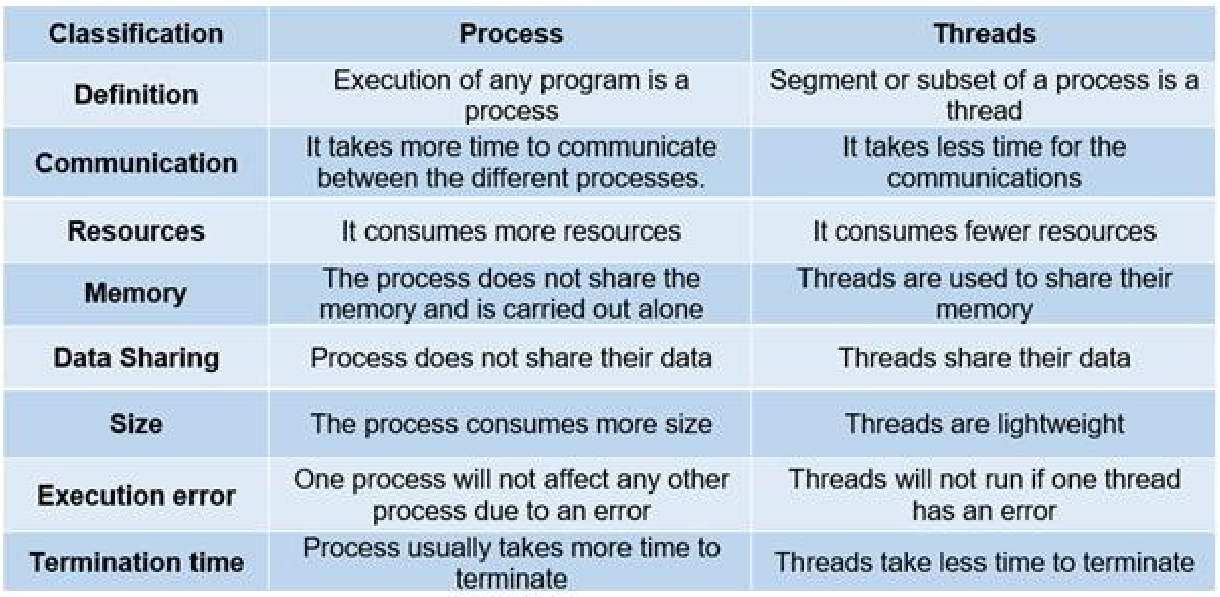


- Thread is the smallest unit of processing that can be performed in an operating system.
- Thread is a light weighted process that shares the same address space of the corresponding process but runs in quasi-parallel.
- Thread is the operating system abstraction of an activity.
- A process contains execution environment which is local kernel managed resources to which its threads have access.

- Thread is the segment of a process means a process can have multiple threads and these multiple threads are contained within a process.
- A thread has 3 states: running, ready, and blocked.

Importance of Thread in Distributed System

- Distributed system needs to support multiple users. Such concurrency is impossible without multi-threaded support.
- Thread is able to block system calls without blocking the entire process. This makes distributed system possible to communicate by maintaining multiple logical connections at the same time.
- Thread can run in a single address space parallel on different CPU.
- Threads can share a common buffer which make it possible to implement producer-consumer problem easily.



Classification	Process	Threads
Definition	Execution of any program is a process	Segment or subset of a process is a thread
Communication	It takes more time to communicate between the different processes.	It takes less time for the communications
Resources	It consumes more resources	It consumes fewer resources
Memory	The process does not share the memory and is carried out alone	Threads are used to share their memory
Data Sharing	Process does not share their data	Threads share their data
Size	The process consumes more size	Threads are lightweight
Execution error	One process will not affect any other process due to an error	Threads will not run if one thread has an error
Termination time	Process usually takes more time to terminate	Threads take less time to terminate

<u>Communication and Invocation (Assignment)</u>

Operating system architecture

Network OS

- Network OS has networking capability. They can be used to access remote resources.
- Each node has its own system image and a user is capable to log in to another computer and run processes there.
- It allows for sharing of files and printer access among multiple computers in a network.

- Network Operating System is a computer operating system that facilitates to connection and communication of various autonomous computers over a network. An Autonomous computer is an independent computer that has its own local memory, hardware, and O.S.
- It is self capable to perform operations and processing for a single user.
- They can either run the same or different O.S.

- Ex: Windows Server 2003, Linux, Mac OS X, etc.

Distributed OS

- Distributed OS is an operating system that produces a single system image for all the resources in the distributed system.
- Users are never concerned with where their programs run.
- The OS has control over all the nodes in the system.
- It is an OS that runs on multiple machines.

- Distributed Operating System is a model where distributed applications are running on multiple computers linked by communications.
- A distributed operating system is an extension of the network operating system that supports higher levels of communication and integration of the machines on the network.
- This system looks to its users like an ordinary centralized operating system but runs on multiple, independent central processing units (CPUs).

- Ex: Solaris for SUN multiprocessor workstations, OSF/1 which is Unix compatible, MICROS, etc.

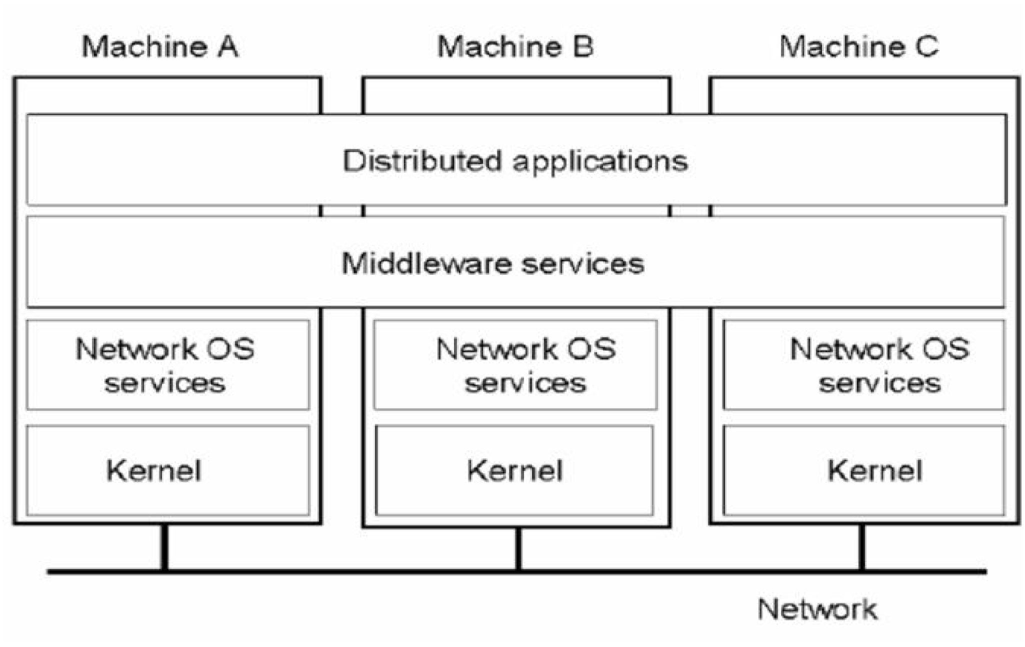
Preference of NOS over DOS

- Users invest in applications to meet their current problem-solving needs. So, they do not tend to adapt to a new operating system which is unable to run their applications even if they are more efficient.
- Users always prefer to have a degree of autonomy for their machines, even in a closely-knit organization. This is because they do not want to spoil their process performance due to the process run by other users.

PARAMETER	DISTRIBUTED OPERATING SYSTEM (DOS)	NETWORK OPERATING SYSTEM (NOS)
Objective	Better Management of hardware resources	Remotely servicing multiple clients.
Communication	Communication is mostly message-based or shared memory based	Communication is file-based or shared folder based.
Scalability	In terms of Scalability, DOS is lesser than NOS	When it comes to Scalability NOS is more scalable
Fault Tolerance	It has high Fault Tolerance	The fault tolerance in Network Operating System is low
Implementation	The ease of implementation is low.	It has ease of implementation.
Openness	Closed	Closed
Rate of autonomy	Low	High
Resource management	Resources are managed through Global central or distributed management	Resources are handled at each node.
Use	Tightly coupled and used in multiprocessor and homogenous computers	Loosely coupled and used in homogenous computers.
Type of Architecture	N-tier Client-server Architecture	2-tier Client-server Architectu

<u>DOS as a Middleware</u>

- ✧ A distributed software support layer that abstracts over the complexity and heterogeneity of the underlying distributed environment with its multitude of network technologies, operating systems, and implementation languages.
- ✧ The primary role of middleware is to ease the task of developing, deploying, and managing distributed applications by providing a simple, consistent, and integrated distributed programming environment.



- ✧ Middleware is the infrastructure that facilitates the creation of business applications, and provides core services like concurrency, transactions, threading, messaging, and the SCA framework for service-oriented architecture applications. It also provides security and enables high availability functionality.

- ✧ Middleware in the context of distributed applications is software that provides services beyond those provided by the operating system to enable the various components of a distributed system to communicate and manage data.
- ✧ Middleware supports and simplifies complex distributed applications.

- ✧ In distributed systems, it hides the distributed nature of the application. It keeps a collection of interconnected parts that are operational and running in distributed locations, out of view making things easier and simpler to manage.

- ✧ <u>Functions:</u>
  - ✓ Hides the intricacies of distributed applications
  - ✓ Hides the heterogeneity of hardware, operating systems, and protocols
  - ✓ Provides uniform and high-level interfaces used to make interoperable, reusable, and portable applications
  - ✓ Provides a set of common services that minimize duplication of efforts and enhances collaboration between applications.