Distributed Operating Systems

Overview

- Functions of Operating System
- Design Approaches
- Types of Advanced Operating Systems.

What is OS?

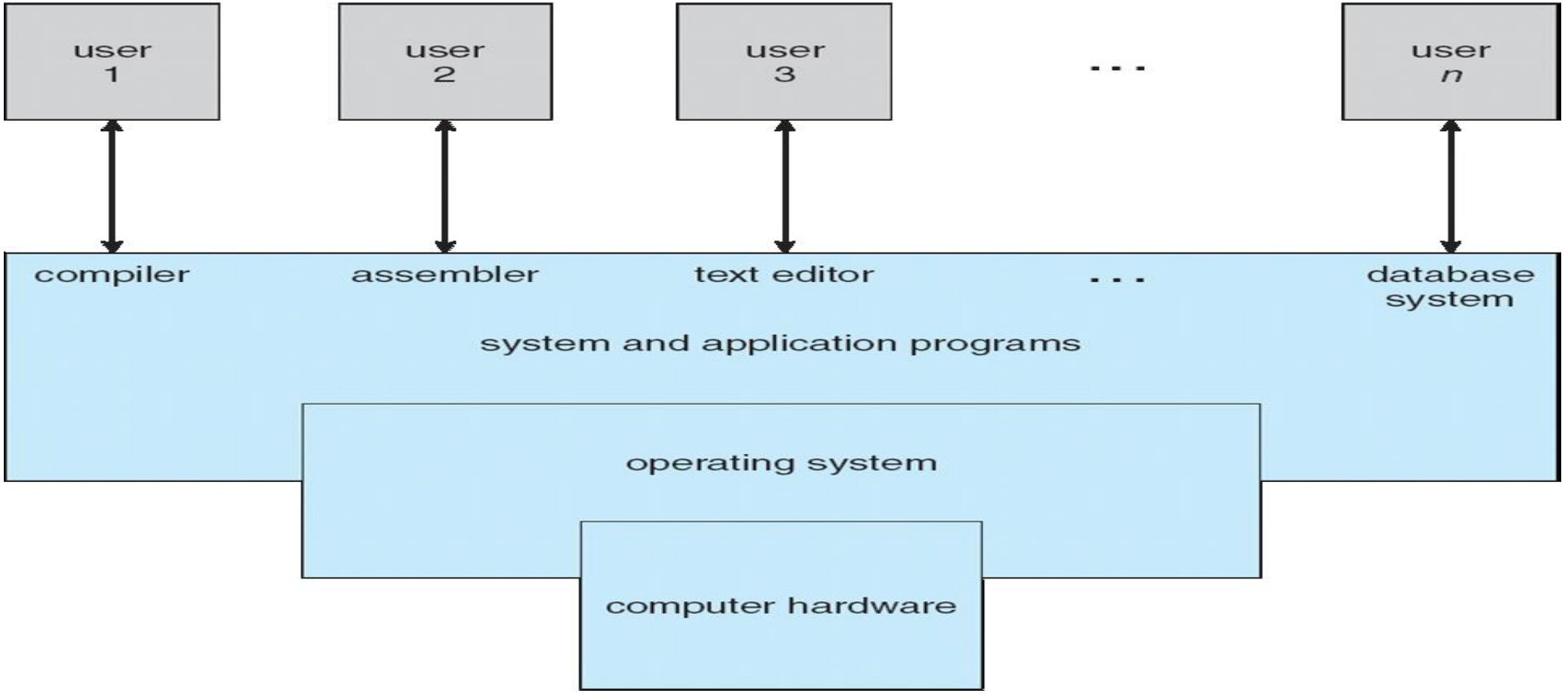
Operating System is a software, which makes a computer to actually work.

- It is the software that enables all the programs we use.
- The OS organizes and controls the hardware.

OS acts as an interface between the application programs and the machine hardware.

Examples: Windows, Linux, Unix and Mac OS, etc.,

Where OS will be in a System



What is Advanced OS?

- Traditional Operating System : which ran on stand- alone computers with single processors.
- Arise of Multiprocessor Systems and Distributed Systems.
- Due to the high demand and popularity of multiprocessor systems, advanced operating system have gained high importance.

Functions of an Operating System

Two basic functions of operating systems are:

1. **Resource management**
2. **User friendliness**

1. Resource Management

A User program accesses several hardware and software resources during its execution. (compiler, linker-loader, files, etc.).

Example: CPU, Main memory, I/O devices, and various types of software

Manages the resources and allocates them to users in an efficient and fair manner.

It encompasses the following functions.

- Time management (CPU and disk scheduling).
- Space management (main and secondary storages).
- Process Synchroziation and deadlock handling.
- Accouting and status information.

2. User Friendliness

- It hides the unpleasant, low-level details and singularity of bare H/W machine . Friendlier interface to the machine.

- It encompasses the following functions.
- Execution environment

(Process management – creation, control, and termination, file manipulation, interrupt handling, support for I/O operations. Language support).

- Error detection and handling
- Protection and security
- Fault tolerance and failure recovery.

Design Approaches

- A typical operating system that supports a multiprogramming environment can easily be tens of megabytes in length and its design, implementation, and testing amounts to the undertaking of a huge software project.

- Deal with complexities of modern systems.

- How and what should be done in the context of operating systems

Separation of Policies and Mechanisms

Policies - What should be done?

Mechanisms - How it should be done?

Ex: In CPU scheduling, mechanisms provide the means to implement various scheduling disciplines and policy decides which CPU scheduling discipline will be used.

Design Approaches

- A good operating system design must separate policies from mechanisms which provide flexibility.
- A change in policy decisions will not require changes in mechanisms.

Three common approaches:

- **Layered Approach**
- **Kernel Approach**
- **Virtual Machine Approach**

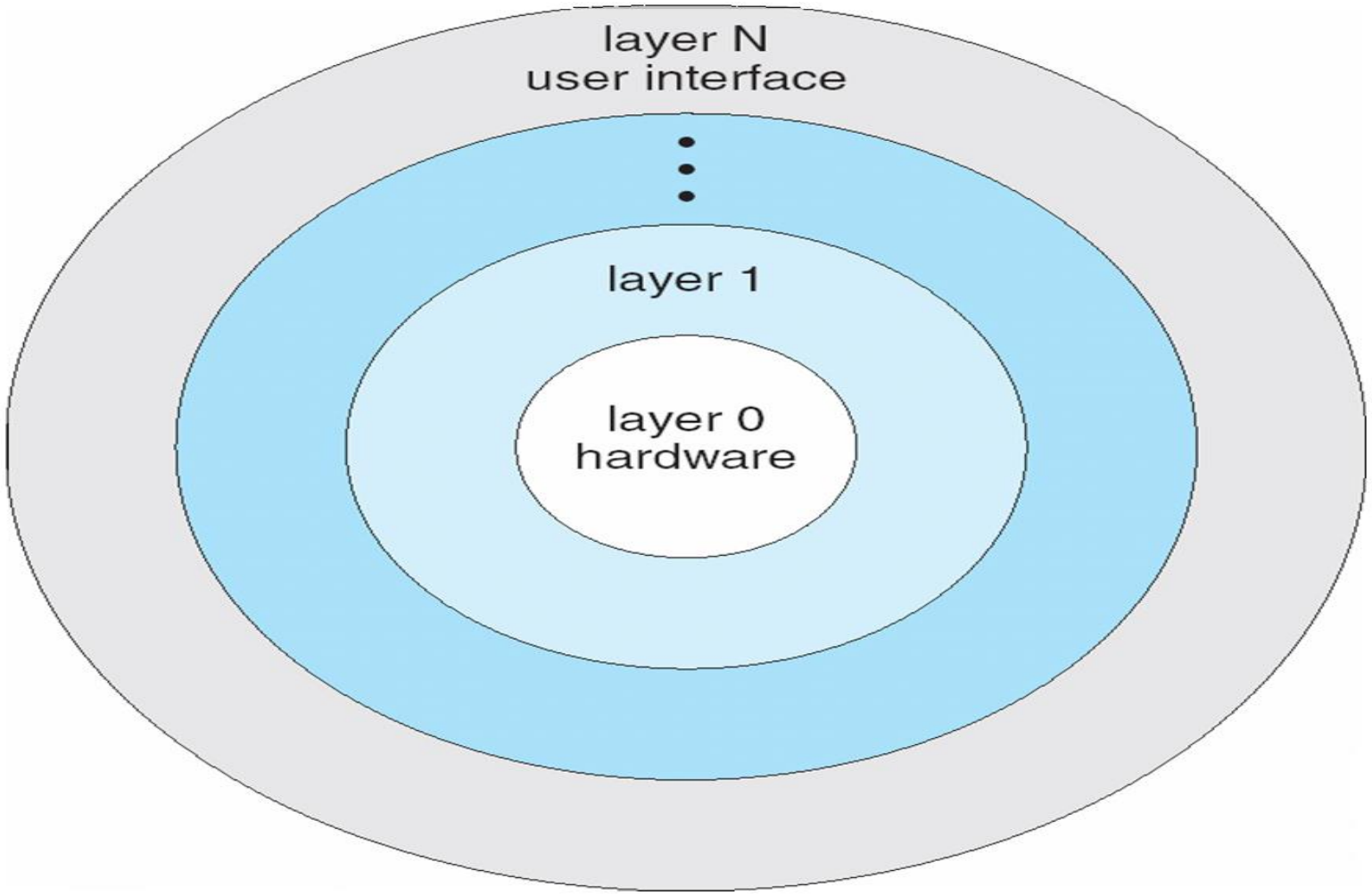
Layered Approach

- This approach divides the operating system into **several layers**.
- Each layer has **well-defined functionality** and **input-output interfaces** with the two adjacent layers.
- The bottom layer interfaces with **machine hardware** and top layer interfaces with **users or operators**.
- This approach has all the advantages of modular design.
- Each layer can be designed, coded and tested **independently**.
- This approach simplifies the design, specification, and implementation of an operating system.

- **Drawback :** functions must be carefully assigned to various layers because a layer can make use only of the functionality provided by the layers beneath it.

- Example : MULTICS(Multiplexed Information and Computing Service) operating system

Layered Approach



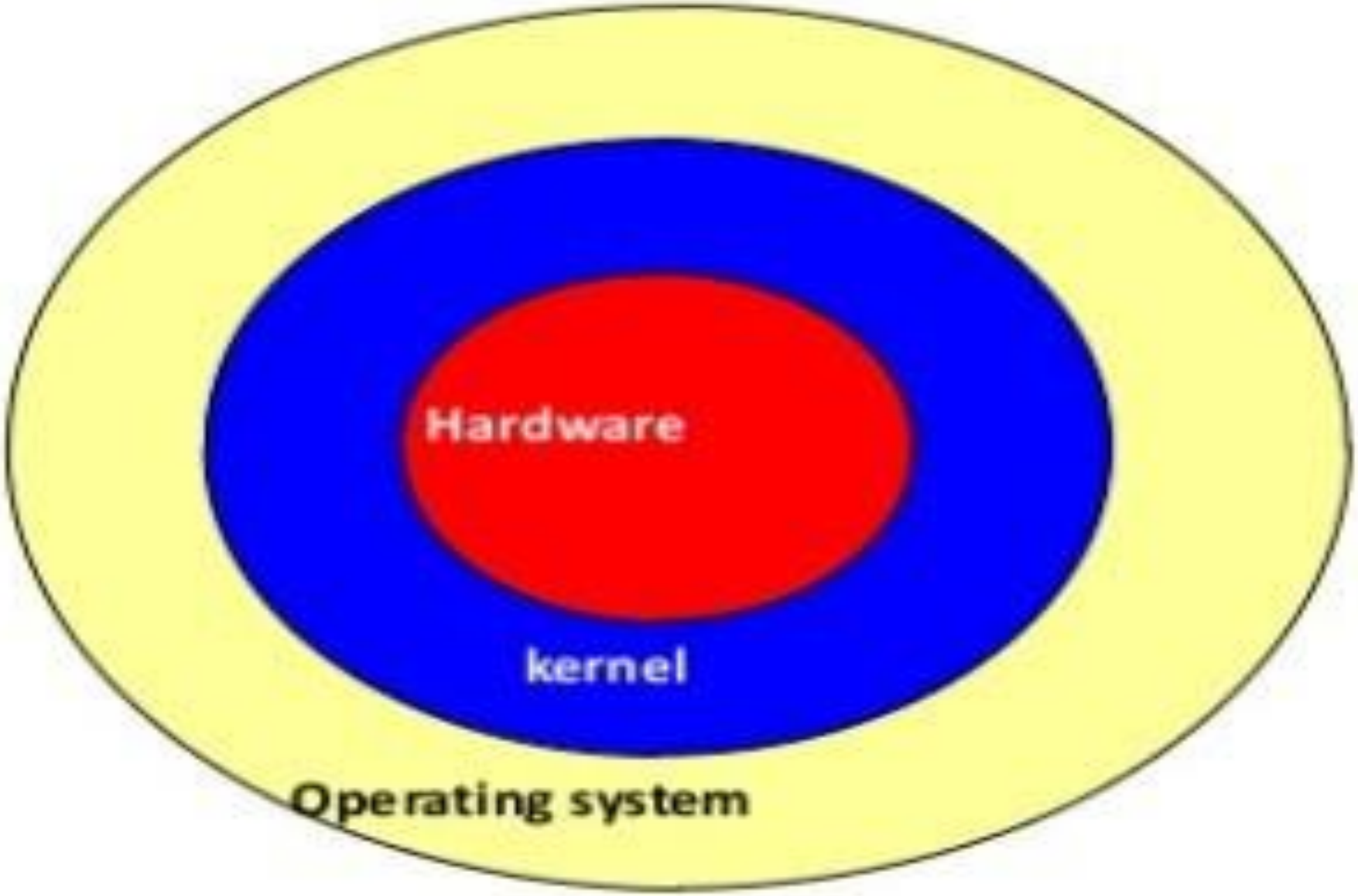
Kernel Approach

- Kernel contains a collection of primitives which are used to build the OS.
- The kernel (nucleus) is a collection of primitive facilities over which the rest of the operating system is built, using the functions provided by the kernel.
- Kernel provides an environment to build operating systems
- Policy and optimization decisions are not made at the kernel level.
- Kernel should support only mechanisms and policy decisions are left to the outer layer.
- OS implements policy .
- Kernel implements mechanisms.

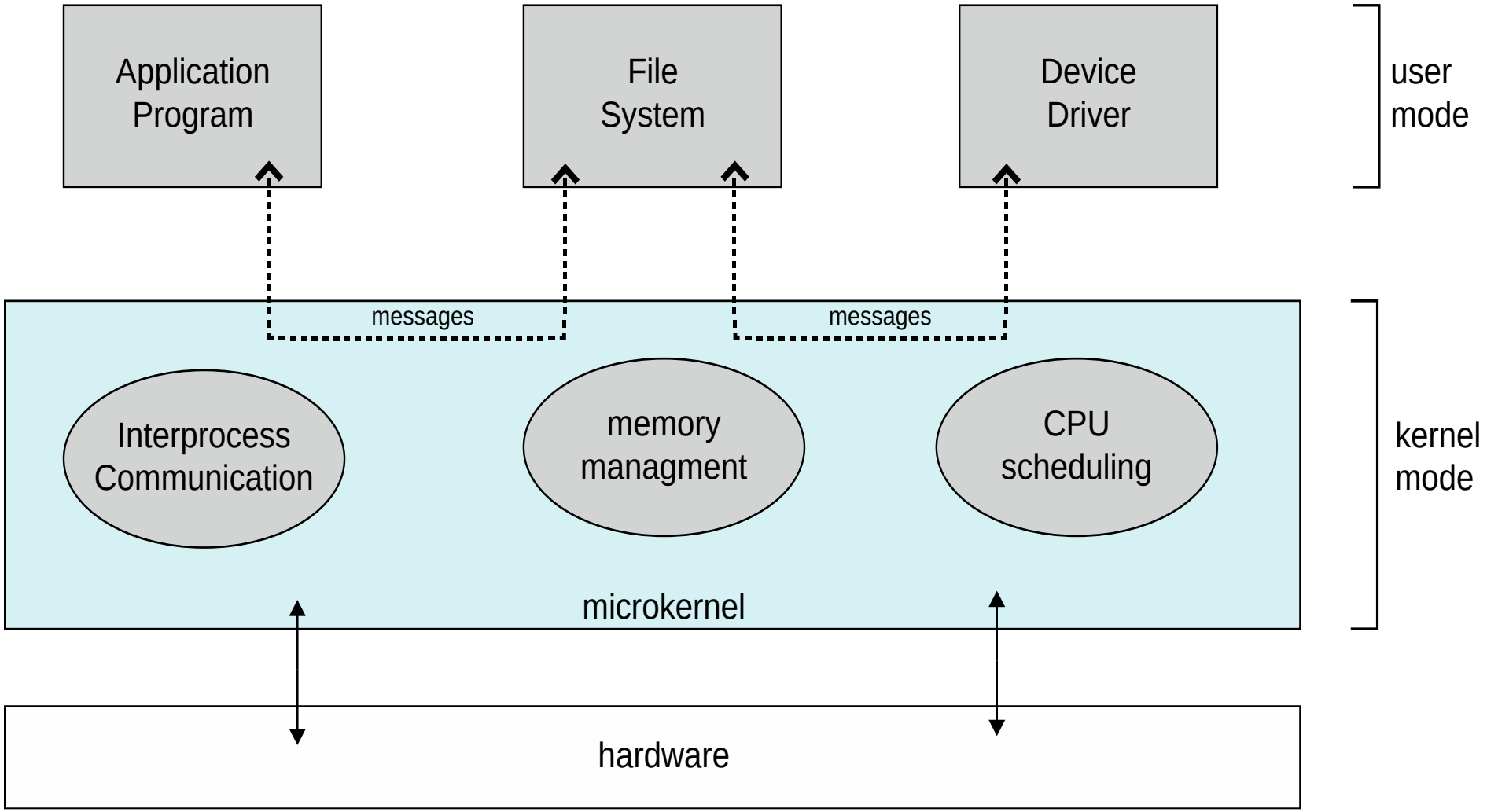
Kernel Approach

- An operating system is an orderly growth of software over the kernel where all decisions regarding process scheduling, resource allocation, execution environment, file system, and resource protection, etc. are made.
- A kernel should contain a minimal set of functionality that is adequate to build an operating system with a given set of objectives.
- Including too much functionality in a kernel results in low flexibility at a higher level, whereas including too little functionality in a kernel results in low functional support at a higher level.
- Example : C.mmp (Kernal is Hydra)

Kernel Approach



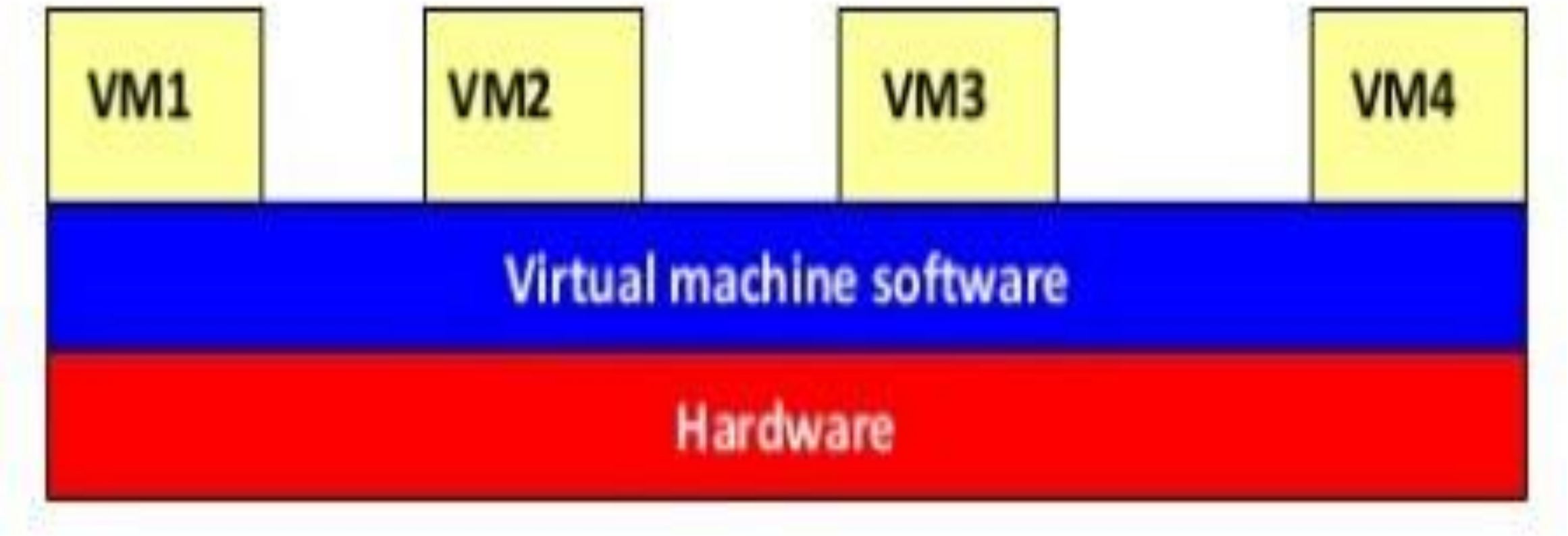
Kernel Approach



Virtual Machine Approach

- A virtual machine software layer on the hardware of the machine gives the illusion that all machine hardware (i.e. the processor, main memory, secondary storage..) is the sole disposal of each user.
- The virtual machine software creates this illusion by appropriately **time-multiplexing the system resources** among all the users of the machine.
- Virtual machine is a **copy** of the bare hardware of the system.
- A user can also run a single-user operating system on this virtual machine.
- The virtual machine concept provides flexibility in that it allows different operating systems to run on different virtual machines.
- Virtual machine software is **huge and complex.**
- Example : IBM 370 system with the virtual machine software VM/370.

Virtual Machine Approach



Types of Advanced Operating Systems

Aadvanced operating systems has from 2 dimensions.

1.It has come from advanced operating systems of multicomputer systems and is driven by a wide variety of high-speed architectures

2. Aadvanced operating systems driven by applications

Types

1. Distributed operating systems
2. Multiprocessor operating systems
3. Database operating systems
4. Real-time operating systems

Types of Advanced Operating Systems

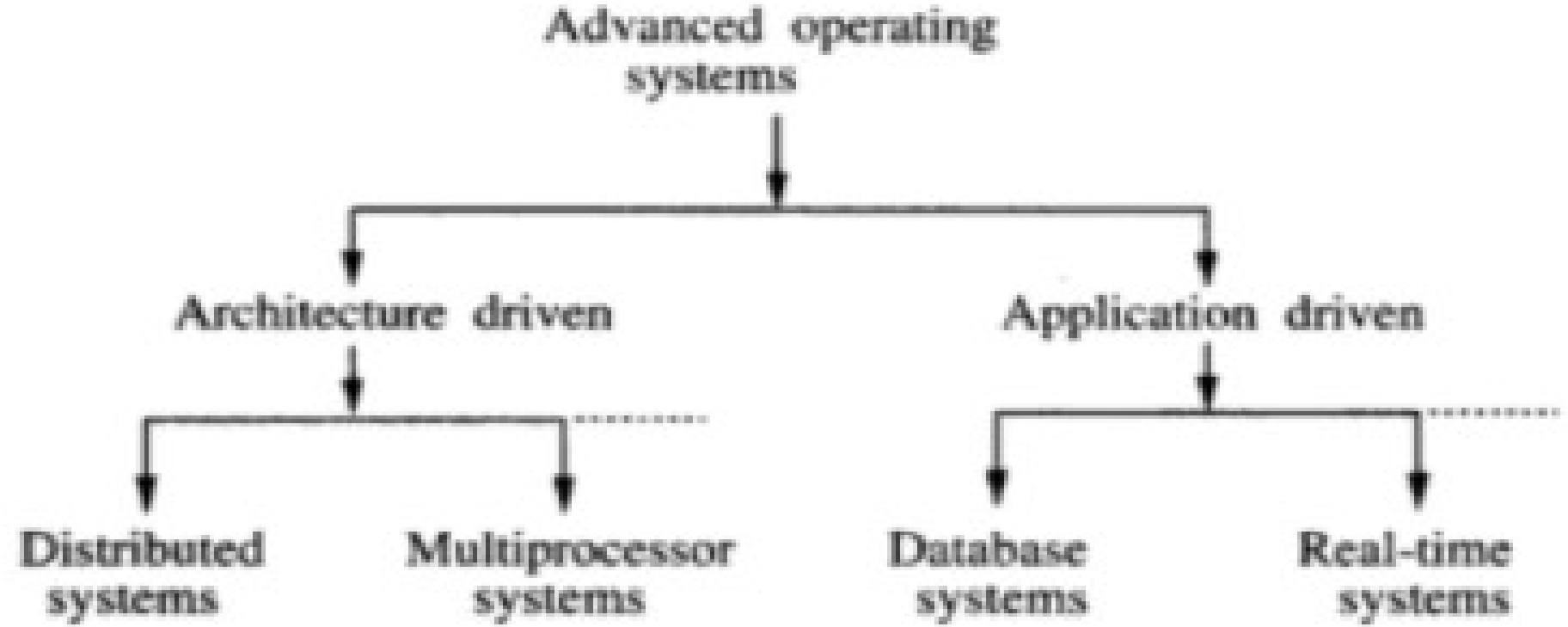


FIGURE 1.3
A classification of advanced operating systems.

1. Distributed Operating Systems

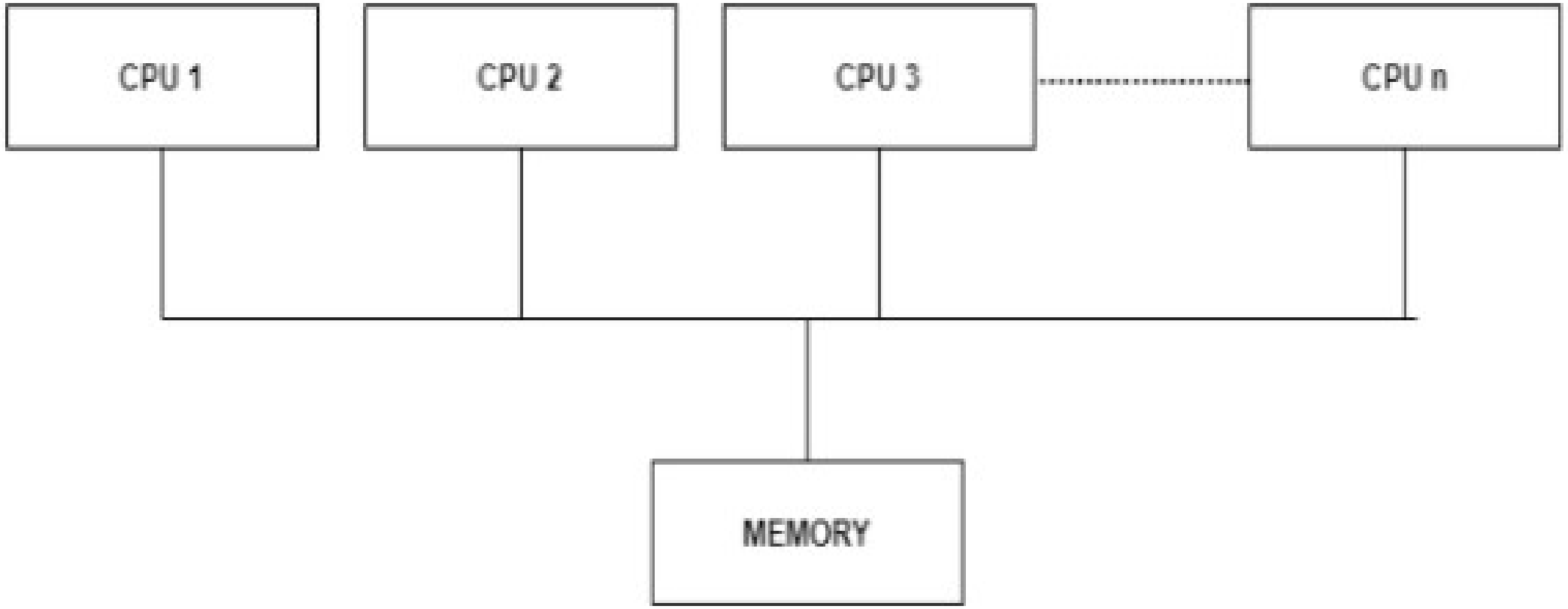
- Are operating systems for a network of autonomous computers connected by a communication network.
- Controls and manages the hardware and software resources of a distributed system such that its users view the entire system as a powerful monolithic computer system.
- When a program is executed in a distributed system, the user is not aware of where the program is executed or of the location of the resources accessed.
- Practical issues:
  - lack of shared memory
  - lack of global clock
  - unpredictable communication delays.

2. Multiprocessor Operating Systems

- Consists of a set of processors that shares a set of physical memory blocks over interconnection network.
- Is a tightly coupled system where the processors share an address space.
- Controls and manages the hardware and software resources such that users view the system as a powerful uniprocessor system
- User is not aware of the presence of multiple processors and the interconnection network
- Practical issues:

  - increased complexity of synchronization, scheduling, memory management, protection and security.

2. Multiprocessor Operating Systems



Multiprocessing Architecture

3. Database Operating Systems

- Must support

    1. The concept of transaction, operations to store, retrieve, and manipulate a large volume of data efficiently

    2. Primitives for concurrency control and system failure recovery

- Must have buffer management system to store temporary data and data retrieved from secondary storage

4. Real-Time Operating Systems

- Jobs have completion deadlines
- Major issue in the design is the scheduling of jobs in such a way that a maximum number of jobs satisfy their deadlines.
- Other issue include designing languages and primitives to effectively prepare and execute a job schedule