V Semester B.Tech IT
Principles of Operating Systems
(Subject code: ICT 3122)
[4 0 0 4]

Syllabus

- **Introduction**
- Operating system structure, Operating system operations, Distributed systems, Special purpose systems, Computing environments, Open source operating systems. **[3 hours]**
- **CPU Scheduling**
- Process concepts: Process states, Process control block, Scheduling queues, Schedulers, Context switch, Multi-threaded programming: Overview, Multithreading models, Threading issues, Process scheduling: Basic concepts, Scheduling criteria, scheduling algorithms. [**7 hours]**
- **Process Synchronization**
- Synchronization: The Critical section problem, Synchronization hardware, Semaphores, Classic problems of synchronization, monitors. **[6 hours]**
- **Deadlocks**
- Deadlock characterization, Methods for handling deadlocks, Deadlock prevention, Deadlock avoidance, Deadlock detection, Recovery from deadlock. **[8 hours]**

Syllabus

- **Memory management**
- Memory management strategies, Swapping, Contiguous memory allocation, Paging, Structure of the page table, Segmentation. **[6 hours]**
- **Virtual Memory**
- Demand paging, copy on write, page replacement, allocation of frames, thrashing. **[7 hours]**
- **Storage Management**
- File concept, Access methods, directory structure, file system structure, directory implementation, allocation methods, free space management, disk structure, and disk-scheduling **[5 hours]**
- **Case study on UNIX based Operating system**
- Design principles, Kernel modules, Process management, Memory management. **[2 hours]**
- **Real time systems:** Characteristics of Real time operating systems, classification of real time systems, Micro kernels and RTOS, scheduling in RTOS, Rate monotonic scheduling, EDF, Priority inversion. **[4 hours]**

Course Objectives

- To get familiarized with the basic functionality and the evolution of different types of operating systems.
- To learn various algorithms related to CPU scheduling, deadlocks, memory management, and storage management.
- To understand the basic aspects of real time operating systems

Course Outcomes

- Illustrate the design principles and functionalities of different operating systems.
- Demonstrate the working of various algorithms for CPU scheduling, synchronization and deadlocks.
- Assess the memory management and storage management techniques and their suitability in different operating systems.
- Apply the concepts of Real Time Operating Systems in application development.

References

1. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, *Operating System Concepts,* 9th edition, Wiley, 2012.
2. William Stallings, *Operating Systems: Internals and Design Principles,* 9th edition, Pearson, 2017.
3. Phillip A Laplante, Seppo J Ovaska, *Real time systems design and analysis,* 4th edition, Wiley, 2013.
4. Rajib Mall, *Real time systems: Theory and Practice,* 2nd edition, Pearson, 2009.

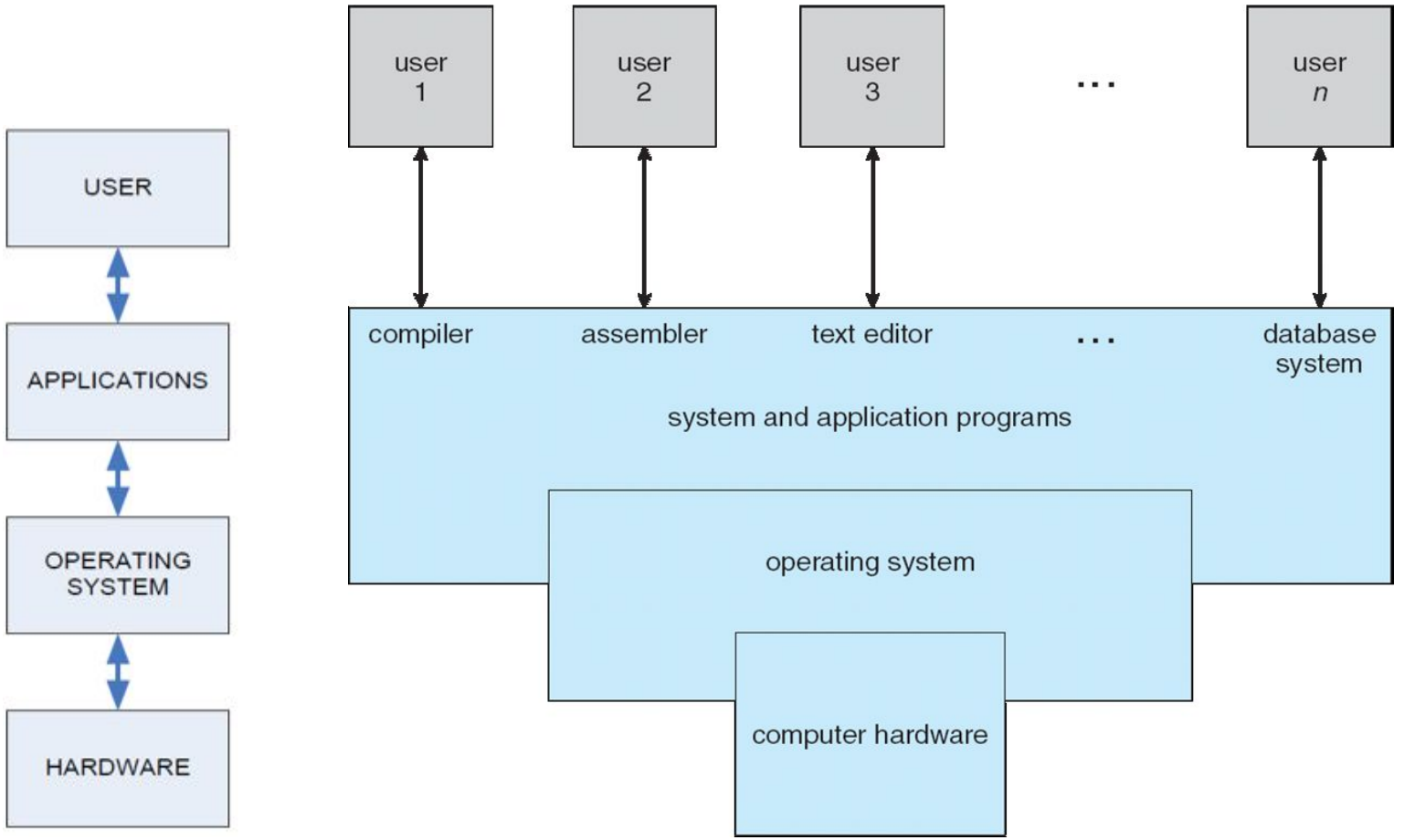
Chapter 1: Introduction

Computer System Structure

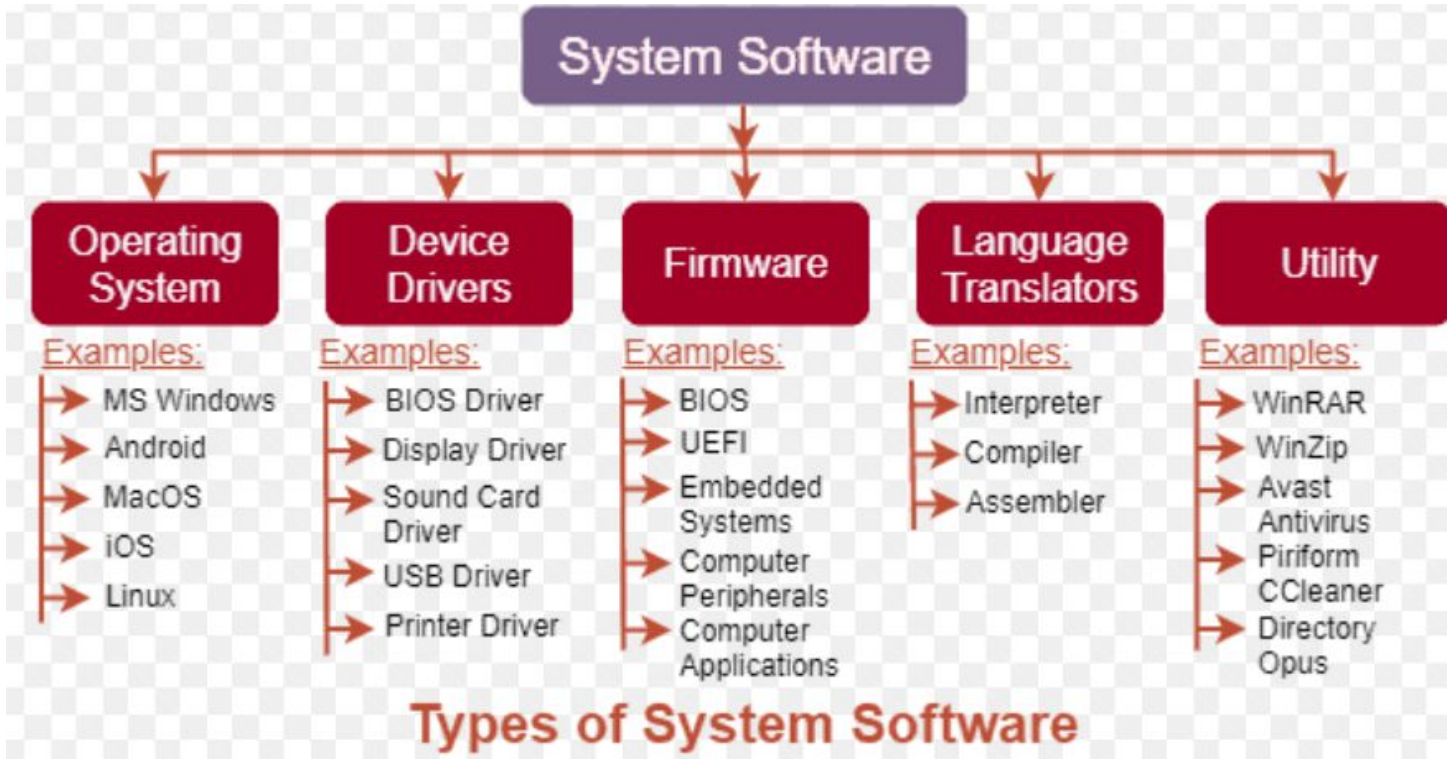
- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

Computer System Structure

Four Components of a Computer System



operating system is large and complex, it must be created piece by piece.


System Software
Operating System
Device Drivers
Firmware
Language Translators
Utility
Examples:
MS Windows
Android
MacOS
iOS
Linux
Examples:
BIOS Driver
Display Driver
Sound Card Driver
USB Driver
Printer Driver
Examples:
BIOS
UEFI
Embedded Systems
Computer Peripherals
Computer Applications
Examples:
Interpreter
Compiler
Assembler
Examples:
WinRAR
WinZip
Avast Antivirus
Piriform CCleaner
Directory Opus
Types of System Software

Operating System

- It is a system software
- A program that acts as an interface between user the computer hardware

- Operating system goals:
  - Execute user programs and make solving user problems easier-you don't need to write code to manage memory or CPU access manually.
  - Make the computer system convenient to use-You can drag and drop files, install apps, and print documents without needing to know hardware details.
  - Use the computer hardware in an efficient manner-prioritize processes, manage RAM, even if hardware resources are limited.

Operating System

- Functionalities
  - Resource Management
  - Process Management
  - Storage Management
  - Memory Management
  - Security and Privacy

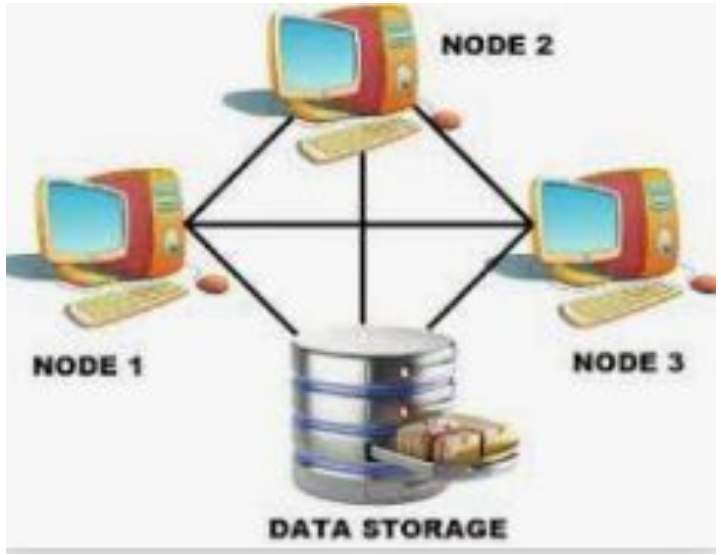
Operating Systems

- Types
  - **Multiprogramming:** they do not provide user interaction with the computer system
  - **Multitasking:** the switches occur so frequently that the users can interact with each program while it is running.
  - **Real Time:** operating system that is designed to meet hard deadlines and provide predictable and consistent performance. RTOSes are used in applications where real-time response is critical, such as avionics, robotics, industrial control, and medical equipment.

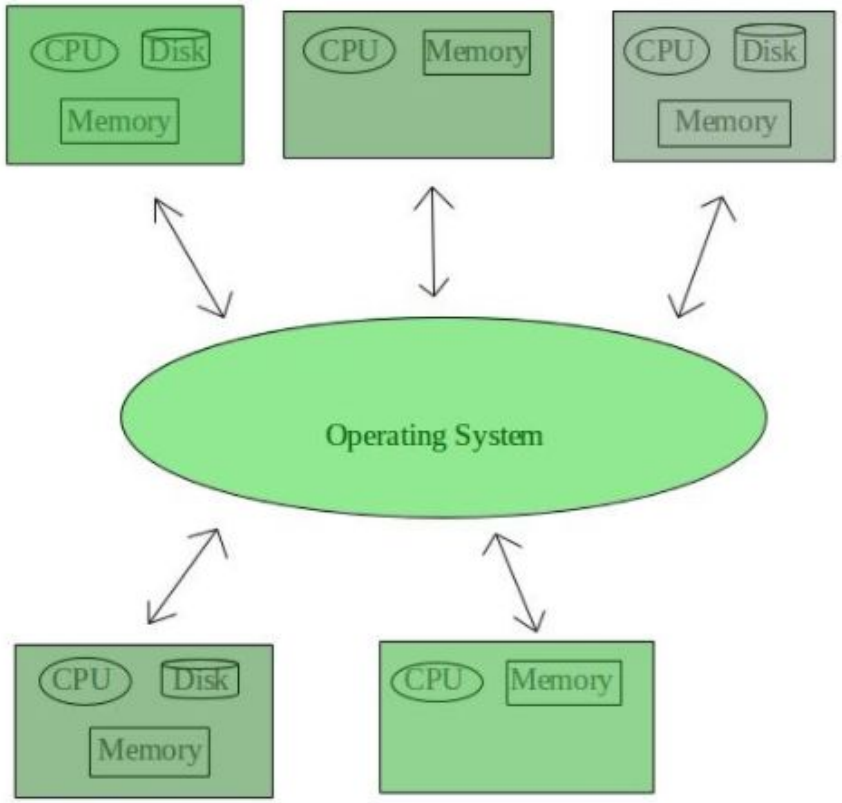
- **Distributed :**
- The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e., remote access is enabled within the devices connected in that network.
- **Clustered:**
- The purpose of a clustered operating system is to provide high availability and scalability by distributing the workload across multiple nodes, so that if one node fails, the workload can be automatically redirected to another node, ensuring that the system remains operational.

- **Embedded:** Embedded operating systems are designed for small, low-power devices, such as IoT devices, smartphones, medical devices, and industrial control systems.

Clustered Operating System



Distributed Operating System



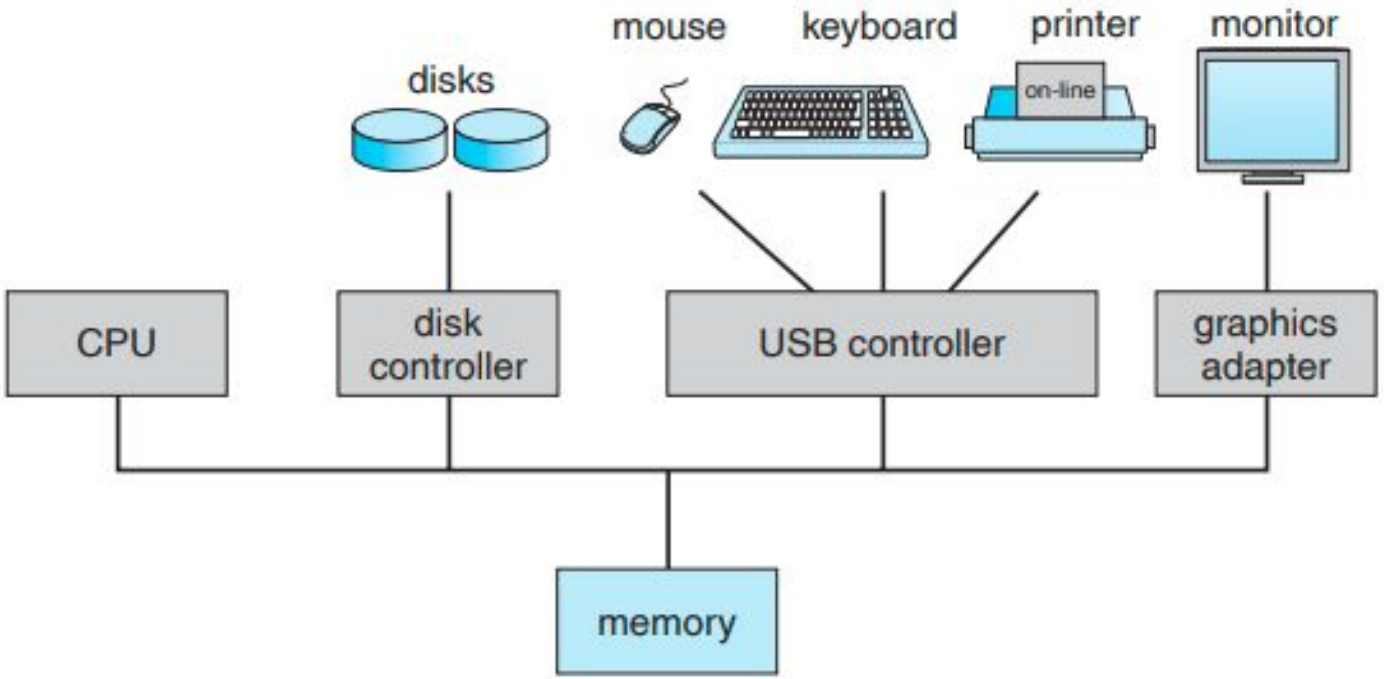
Operating System Definition

- System View
  - OS is a **resource allocator**
    - Manages all resources
    - Decides between conflicting requests for efficient and fair resource use
  - OS is a **control program**
    - Controls execution of programs to prevent errors and improper use of the computer.

Operating System Definition (Cont.)

- No universally accepted definition
- “The one program running at all times on the computer” is the **kernel**.
- Everything else is either,
  - a system program, or an application program.
- The **OS** includes the **kernel + system programs**.

Computer-System Organization

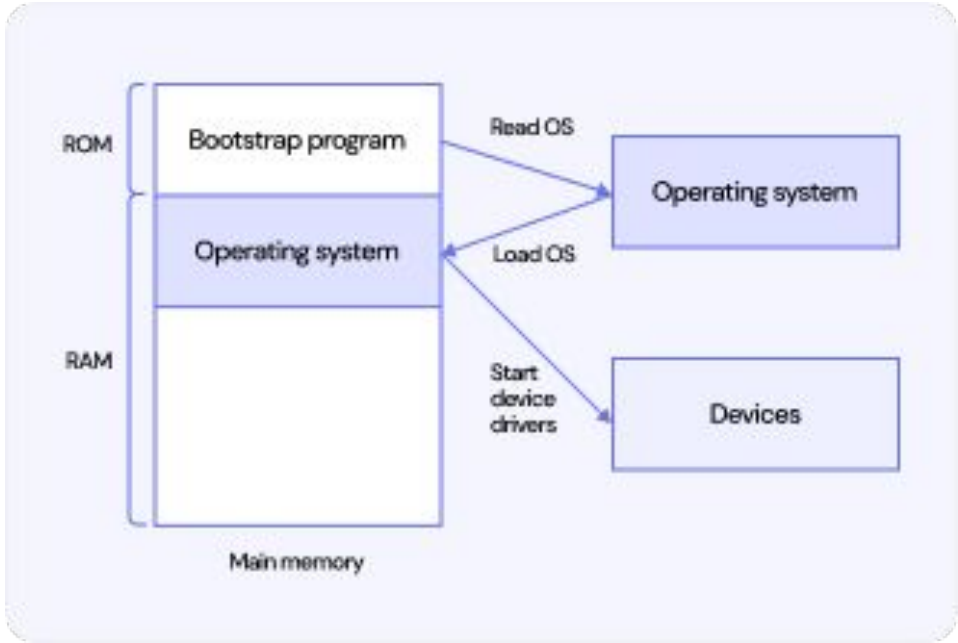


A modern computer system.

- One or more CPUs
- Number of device controllers
- Shared memory
- Memory controller

Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Power-On or Reboot: When the computer is powered on or rebooted, the CPU starts executing the bootstrap program stored in the ROM/EPROM.
  - Initialization: The bootstrap program initializes hardware components and performs necessary checks.
  - Locate OS Kernel: Searches for a **bootable device,** it locates the OS kernel on the storage media.
  - Load OS Kernel: It loads the OS kernel into RAM.
  - Transfer Control: It transfers control to the OS kernel, which then continues with the system's startup process.



Operating-System Operations

- **Modern OS are Interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices: A signal created and sent to the CPU that is caused by some action taken by a **hardware** device. keystroke depressions and mouse movements cause hardware interrupts.
  - Software interrupt (**exception** or **trap):**
    - Software error (e.g., division by zero)
    - invalid memory access
    - Request from user program to access a specific operating system service
  - For each type of interrupt, separate segments of code in the operating system determine what action should be taken.
  - An interrupt service routine is provided that is responsible for dealing with the interrupt.
  - Other process problems include infinite loop, processes modifying each other or the operating system

“Bootstrap is a term used to describe the process of loading an operating system into memory and starting it. BIOS, on the other hand, is a firmware that provides basic low-level services to the operating system, such as initializing hardware, testing memory, and providing a boot loader to load the operating system from a storage device.

An example of a Bootstrap program is the Master Boot Record (MBR) on a computer's hard drive. The MBR is a small piece of code stored in the first sector of the hard drive that is executed when the computer is powered on. It contains the boot loader, which loads the operating system into memory and starts it. This process is called bootstrapping, hence the name Bootstrap program.”

Common Functions of Interrupts

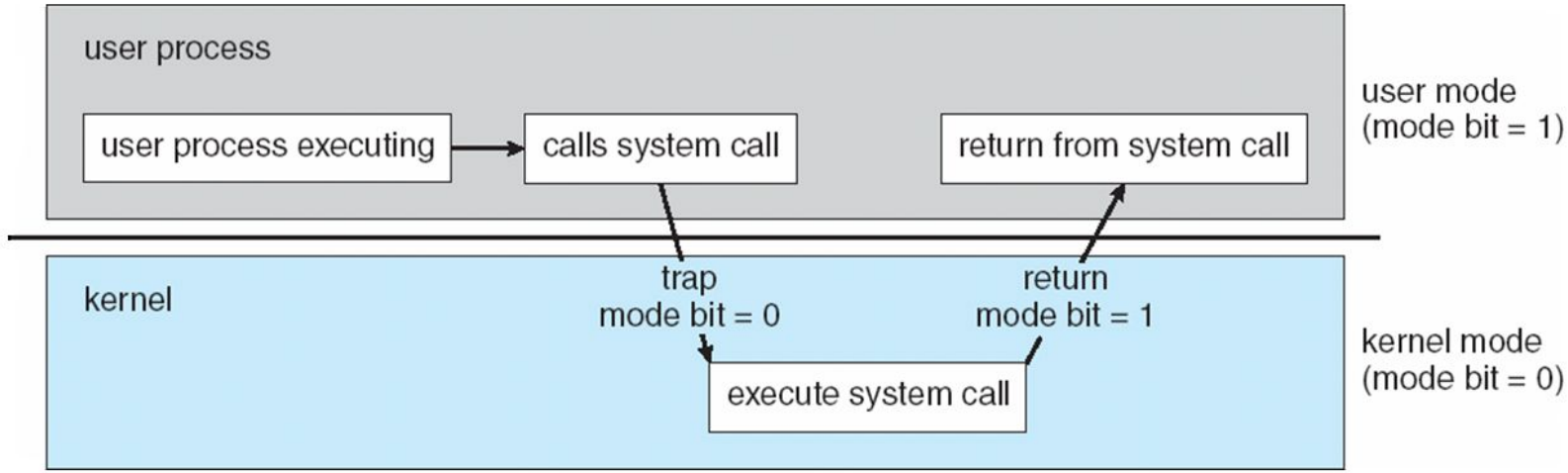
- Interrupt transfers control to the interrupt service routine (ISR) generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction before ISR executes.
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

- A key is pressed on the keyboard, generating a hardware interrupt.
- The keyboard controller sends an interrupt signal to the CPU.
- The CPU uses the interrupt vector to find the address of the ISR associated with the keyboard interrupt.
- The CPU saves the address of the interrupted instruction and the current state of the CPU registers.
- The CPU transfers control to the keyboard ISR, which processes the keypress (e.g., adding the character to a buffer).
- After the ISR completes, the CPU restores the saved state and resumes execution of the interrupted program.

Operating-System Operations (cont.)

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged instructions(i/o instructions, turn of interrupts, set timer, clear memory etc)**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode



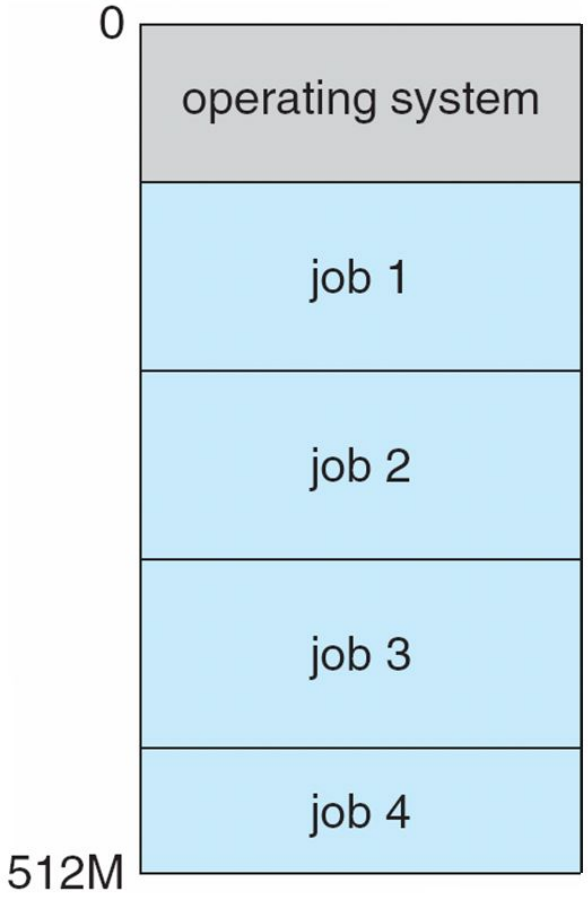
- When the computer system is executing on behalf of a user application, the system is in user mode.
- However, when a user application requests a service from the operating system (via a system call), it must transition from user to kernel mode to fulfill the request.

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Timer is set to interrupt the computer after some period time
  - Keep a counter that is decremented by the physical clock.
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

- A program with a 7-minute time limit, for example, would have its counter initialized to 420. Every second, the timer interrupts, and the counter is decremented by 1. As long as the counter is positive, control is returned to the user program. When the counter becomes negative, the operating system terminates the program for exceeding the assigned time limit.

Memory Layout for Multiprogrammed System



job pool
job scheduling
CPU scheduling
Swapping
Virtual memory

Operating System Structure

- **Fundamental execution models** that shape how the OS is organized:

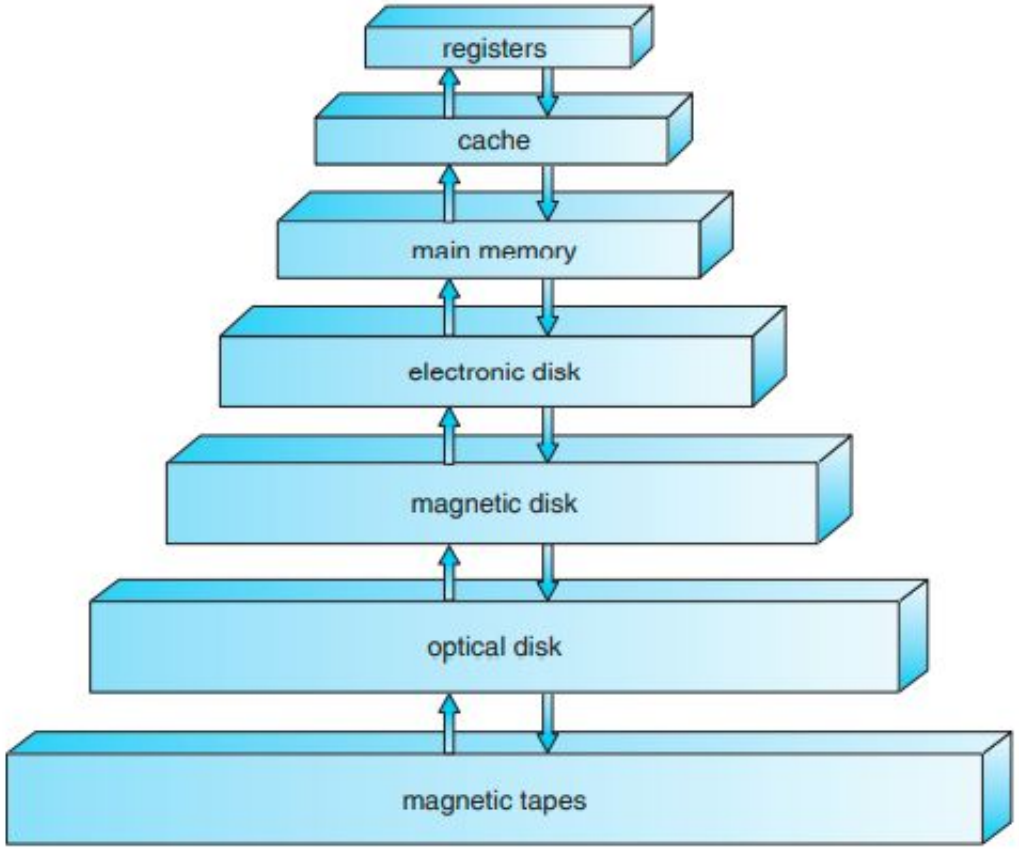
1. **Multiprogramming** (**Batch system**) needed for efficiency
   - Single user cannot keep CPU and I/O devices busy at all times
   - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
   - A subset of total jobs in system is kept in memory
   - One job selected and run via **job scheduling(from job pool)**
   - When it has to wait (for I/O for example), OS switches to another job

2. **Timesharing** (**multitasking**) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
   - **Response time** should be < 1 second
   - Each user has at least one program executing in memory □ **process**
   - If several jobs ready to run at the same time □ **CPU scheduling(picking jobs from memory)**
   - If processes don't fit in memory, **swapping** moves them in and out to run
   - **Virtual memory** allows execution of processes not completely in memory

Storage Structure

- Main Memory
  - RAM
  - Main memory is usually too small to store all needed programs and data permanently.
  - Main memory is a volatile storage device that loses its contents when power is turned off or otherwise lost
- Read-only memory (ROM)
  - A typical instruction–execution cycle
- Secondary storage
  - hold large quantities of data permanently
  - magnetic disk

Storage-device hierarchy



Overview of OS functionalities

Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a ***passive entity***, process is an ***active entity***.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
  - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

Computing Environments - Traditional

- Stand-alone general purpose machines
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming default– even home systems use **firewalls** to protect home computers from Internet attacks

Computing Environments – Distributed

- Distributed computing
  - Collection of separate, possibly heterogeneous, systems networked together
    - **Network** is a communications path, **TCP/IP** most common
      - **Local Area Network** (**LAN**)
      - **Wide Area Network** (**WAN**)
      - **Metropolitan Area Network** (**MAN**)
      - **Personal Area Network** (**PAN**)
  - **Network Operating System** provides features between systems across network
    - Communication scheme allows systems to exchange messages
    - Illusion of a single system

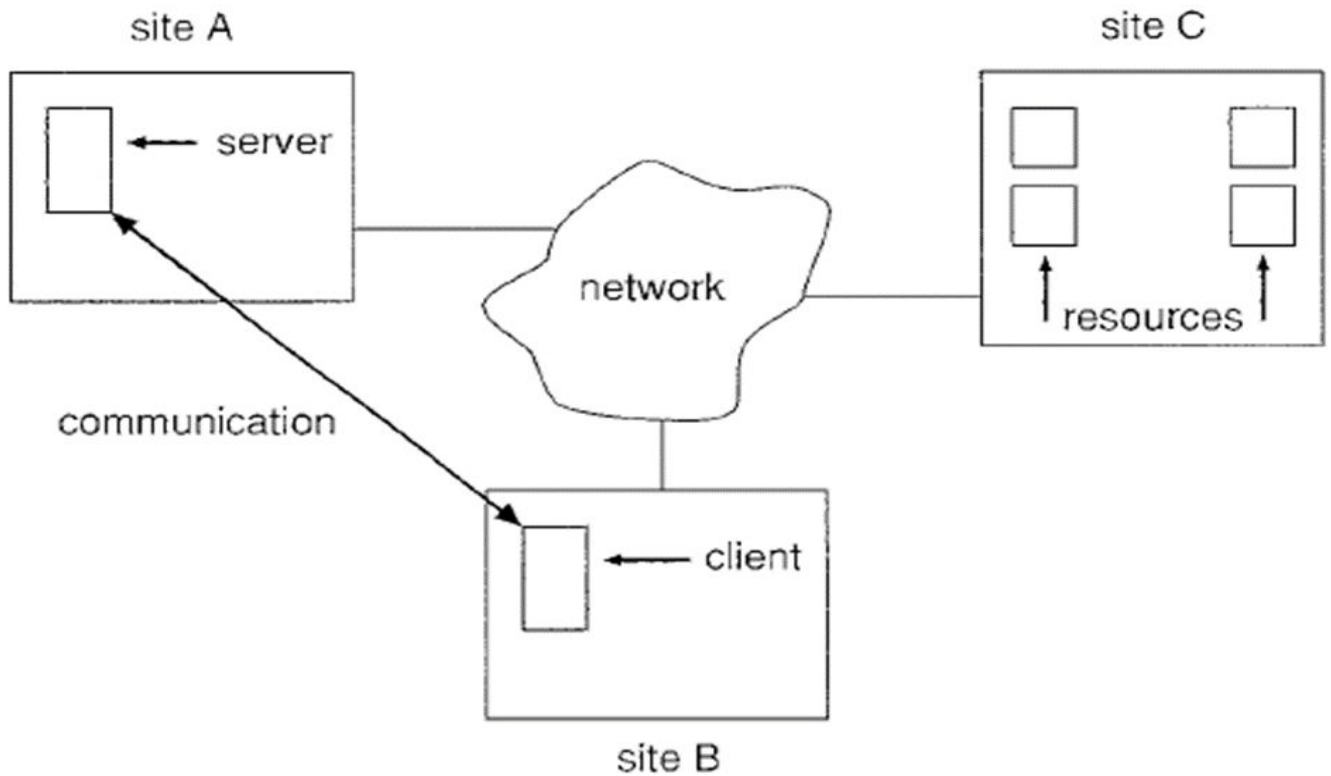
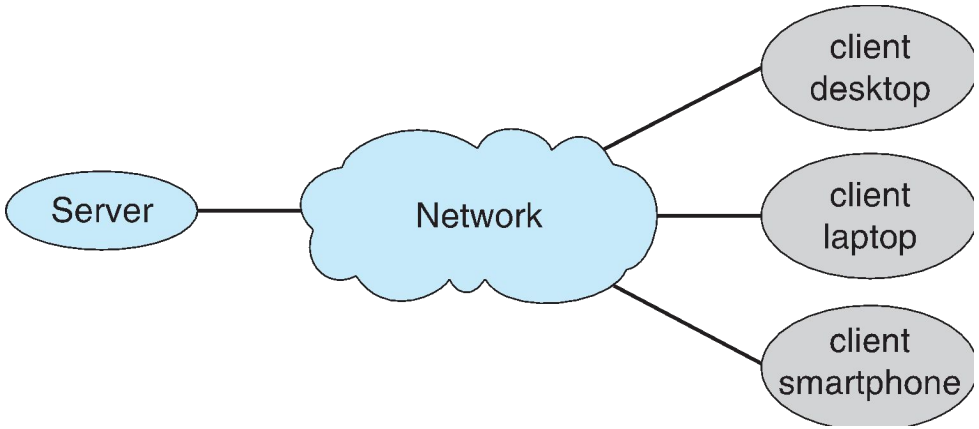


Figure 16.1 A distributed system.

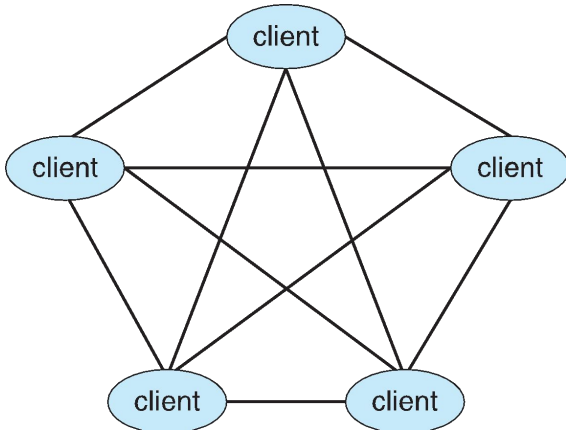
Computing Environments – Client-Server

- Client-Server Computing
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server system** provides an interface to client to request services
    - **File-server system** provides interface for clients to store and retrieve files



Computing Environments - Peer-to-Peer

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network

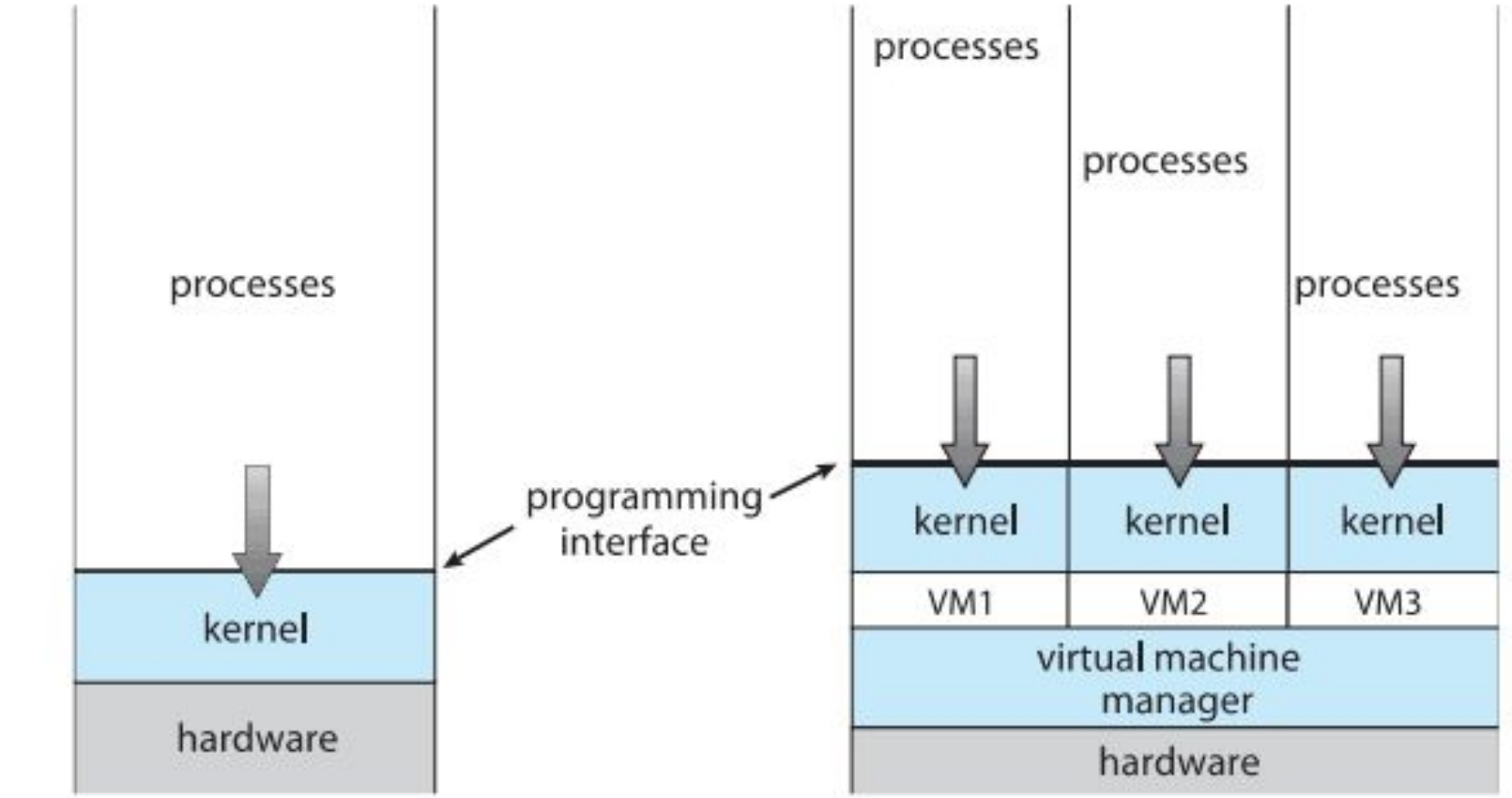


    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via ***discovery protocol***

Computing Environments - Virtualization

- virtualization refers to the act of creating a virtual version of something, including virtual computer hardware platforms, storage devices, and computer network resources.

Computing Environments - Virtualization



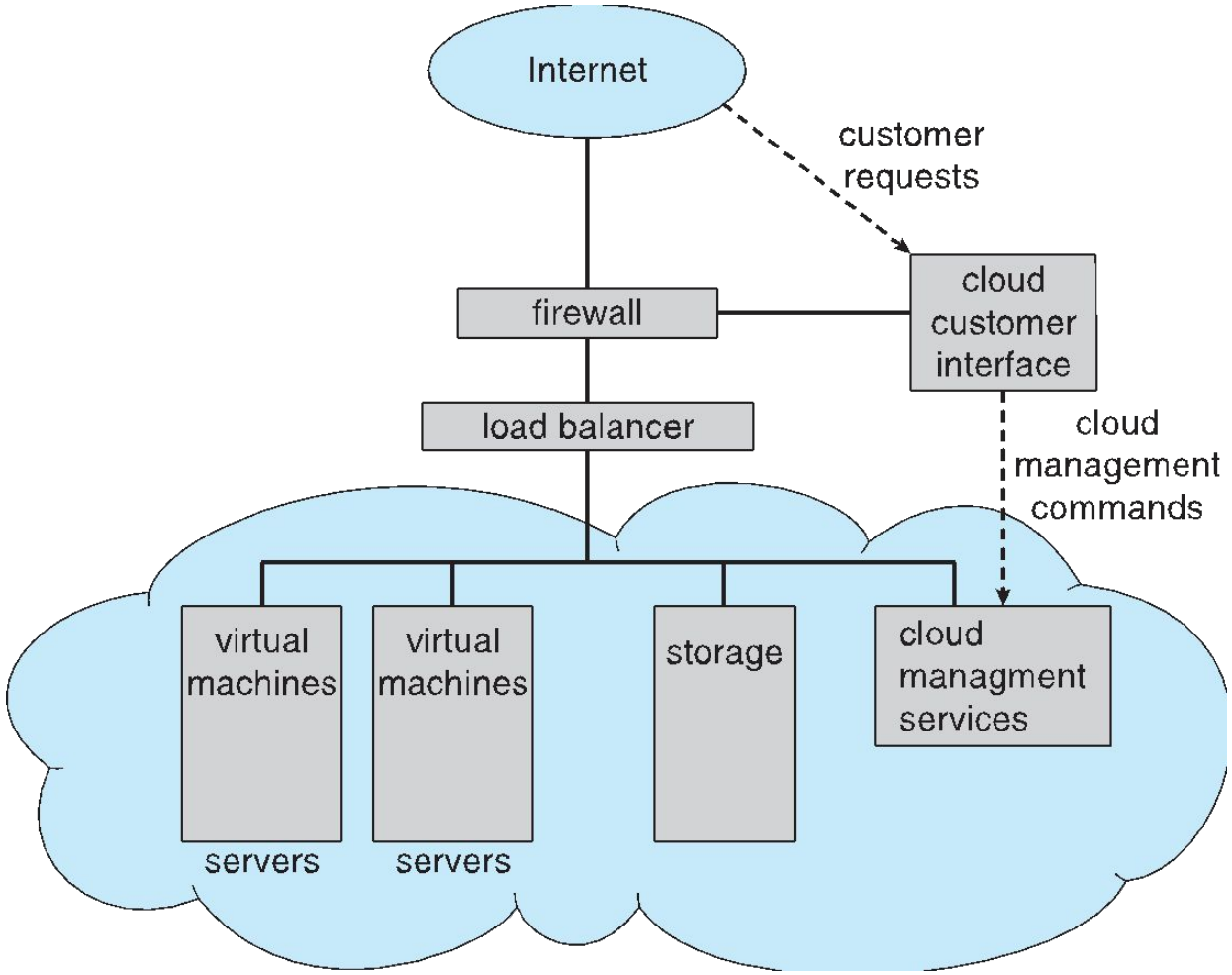
Non-virtual machine

Virtual machine

Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor)
  - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e.,Google App Engine)
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use)

Computing Environments – Cloud Computing



Computing Environments – Real-Time Embedded Systems

- Real-time embedded systems most widespread form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source(Windows)**
- Examples Open source include **GNU/Linux** and **BSD UNIX**
- There are many benefits to open-source operating systems/ including a community of interested (and usually unpaid) programmers who contribute to the code by helping to debug it, analyze it, provide support, and suggest changes.