



Operating Systems

Andrei Doncescu




Chapter 1: Introduction



Operating System Definition

- No universally accepted definition
- “The one program running at all times on the computer” is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program.



What is an Operating System?

- A program that acts as an intermediary between an user of a computer and the computer hardware:

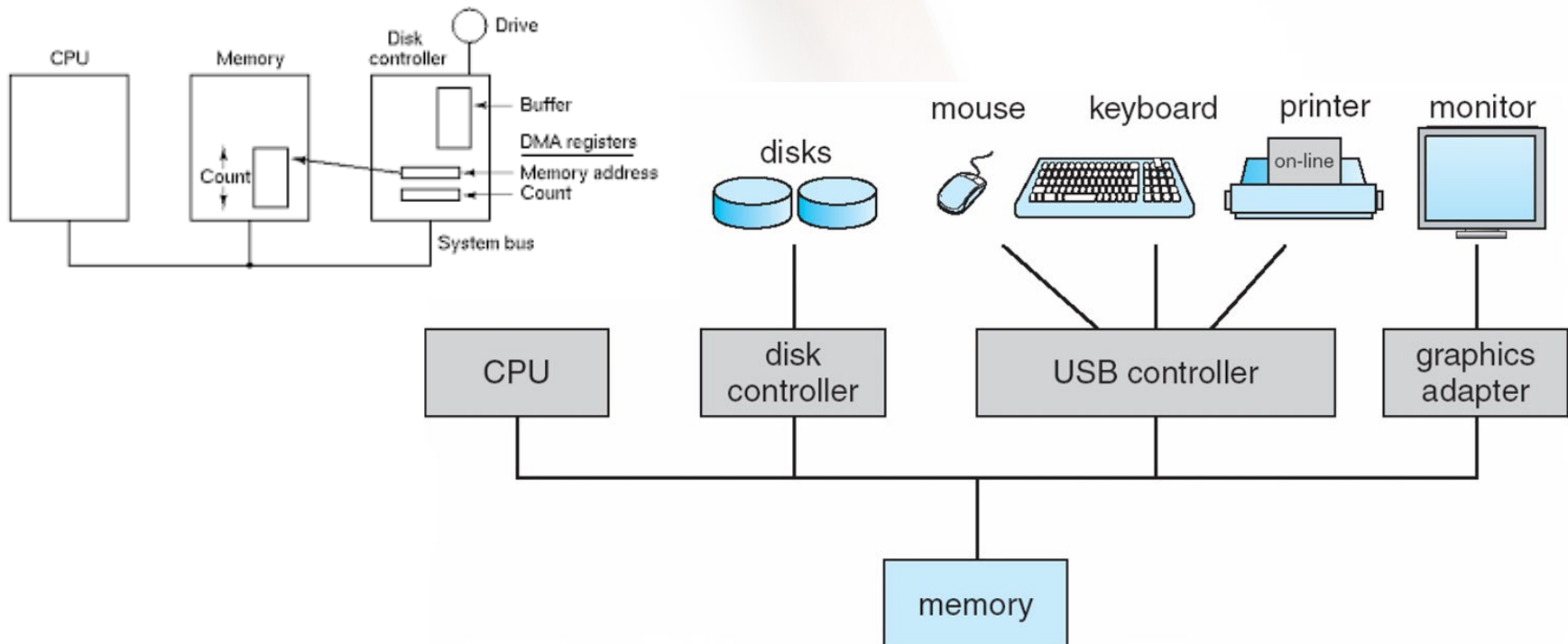
Manages the computer hardware

- Operating system goals:
 - Execute user programs and make solving user problems easier
 - Make the computer system convenient to use
 - Use the computer hardware in an efficient manner

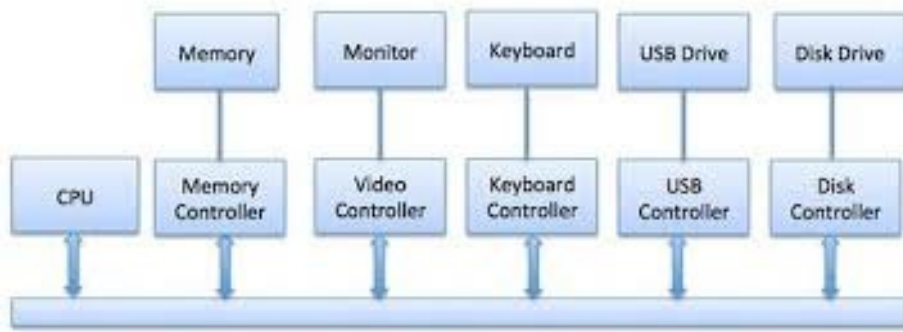
Computer System Organization

- Computer-system operation


- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles



Computer-System Operation



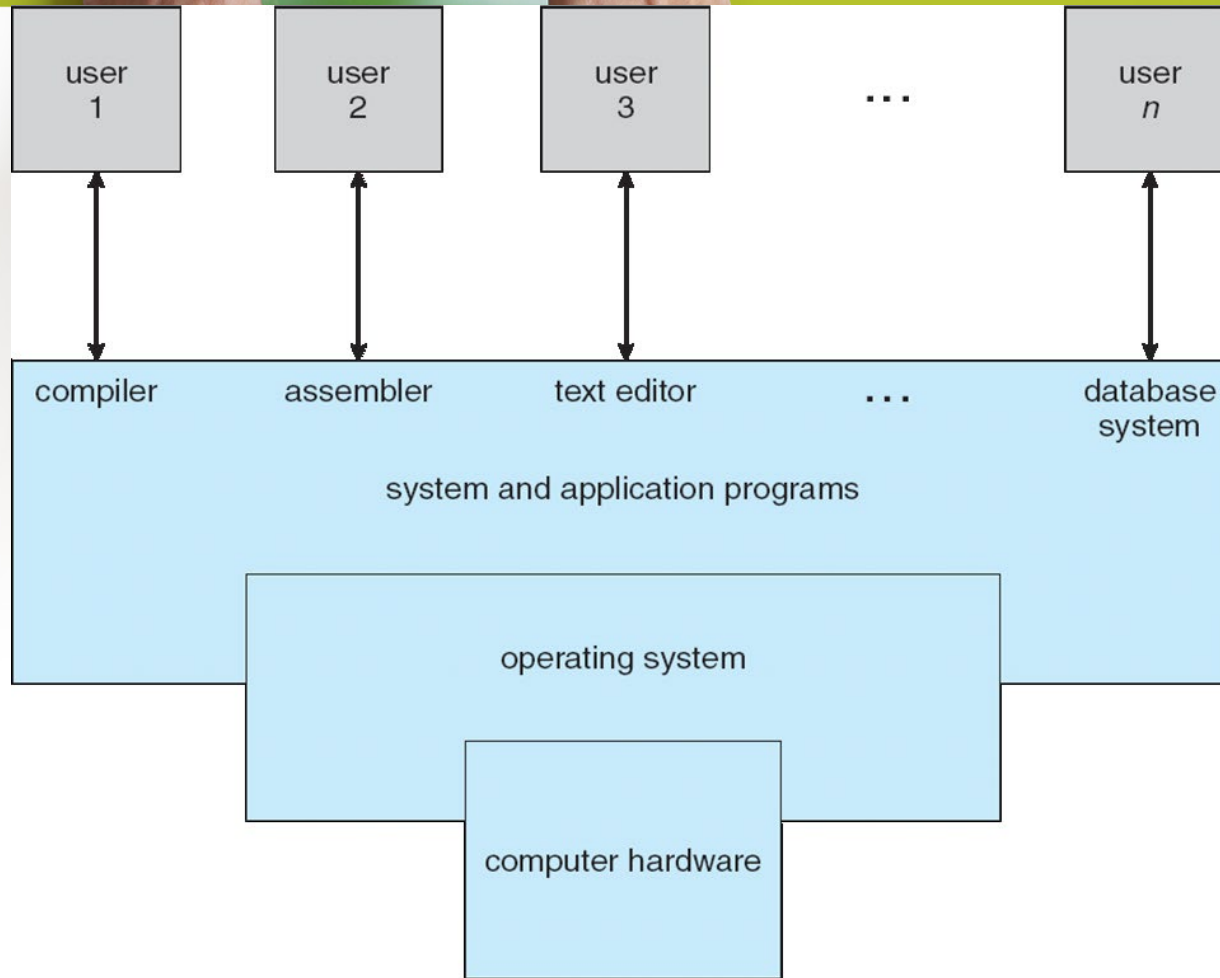
- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an *interrupt*



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

Four Components of a Computer System





Operating System Definition

- 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



Process Management Activities

The operating system is responsible for the following activities:

- 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



Chapter 2: History





Background History

- 1969 **Unix created at Bell Labs**
- 1977 Berkeley Development starts
- 1986 4.3BSD released
- 1990 Net/2 Release distributed
 - asserted to be freely distributable
- 1992 AT&T lawsuit
- 1994 4.4BSD-lite released
 - clean base - all suspect code deleted

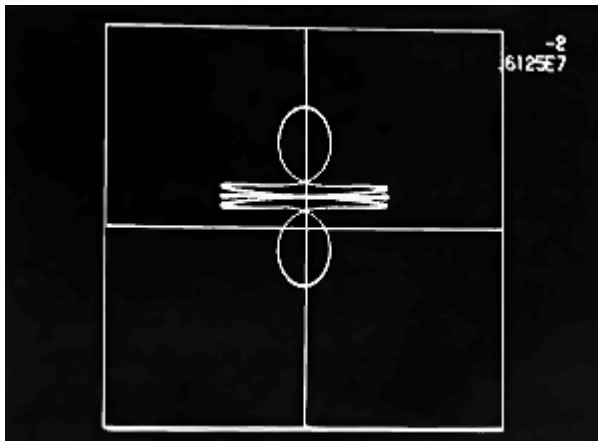


Overview

- What is Unix™ ?
- Brief History
- In the Present Day
- In Conclusion...

What is Unix™ ?

- Unix™ is an open source operating system.
- Unics : UNiplexed Information & Computing Service.
- Unix™ was first written for Space Travel, a computer game by Ken Thomson. Space Travel is the first Application created. It is also the first computer game ever created.



Space Travel: A game simulating travel in space. You navigate by zooming in/out, to reach earth or the other planets in your spaceship.



Brief History of Unix™

- 1969 : AT&T develops Multics (multiplex information & computing science)
 - Multics was an experimental OS
 - Ken Thompson used it to play Space Travel, a game he wrote on the Multics computer system.
 - Project was shelved and so did the multics system.
 - Ken decides to write re-work multics so he could play Space Travel on a smaller system left unused in the lab. Dennis made sure of that by recoding in C.

The PDP-7 system that **Unix™** was written on. It even ran Space Travel.



Brief History of Unix™

- 1970s: Redeveloped & recoded in C programming by Ken Thompson & Dennis Ritchie



- AT&T recognized their work and funded them with bigger systems. In return they created roff, a text processing system. Final versions is troff, which does text formatting.



- Meanwhile, Dennis teaches C programming & Unix at Berkeley University.

Brief History of Unix™

- 1977: Berkeley student Bill Joy releases BSD (Berkeley Software Distribution)
- 1980: MACH kernel for BSD from CMU
- 1988: NextStep is release with GUI
- 1991: Linus Torvalds releases
Unix-Like Linux

RIP: Dennis Ritchie, the creator of C programming passed away a week after Steve Jobs on Oct 12th 2011.

He has given to us very valuable technology and inspired developers to create.





History of Mach

- Mach's earliest roots go back to a system called RIG (Rochester Intelligent Gateway), which began at the University of Rochester in 1975.
 - Its main research goal was to demonstrate that operating systems could be structured in a modular way.
- When one of its designers, Richard Rashid, left the University of Rochester and moved to Carnegie-Mellon University in 1979, he wanted to continue developing message-passing operating systems but on more modern hardware.
- The machine selected was the PERQ. The new operating system for the PERQ was called Accent. It is an improvement of RIG.



Rochester's Intelligent Gateway

- Developed by University of Rochester, NY 1975
- Provided uniform access to a variety of computing facilities
- Disadvantages:
 1. Had limited address space
 2. No protection for ports
 3. No notification of failure of a process

- Developed by Richard Rashid, formerly of RIG, at Carnegie Mellon University
- The operating system improved upon RIG
 1. Added an innovation virtual memory
 2. Added transparent network messaging
 3. Flexible and powerful virtual memory



History of Mach

- By 1984 Accent was being used on 150 PERQs but it was clearly losing out to UNIX. This observation led Rashid to begin a third-generation operating systems project called Mach.
- Mach is compatible with UNIX, contains threads, multiprocessor support, and a virtual memory system.



History of Mach

- Developed at Carnegie Mellon University
- 1987 – Mach release 0 and release 1 included task and thread support
- 1988 – release 2
- 1989 – Mach 3.0 released with a smaller microkernel and a complex multiprocess system



Concept of the Mach Kernel

- The classic Unix kernel cannot support multiple processors.
- Mach introduced the idea of threads and tasks.
- Mach was one of the first systems to use a microkernel.
- A microkernel minimizes each OS service performed in the kernel.



Tasks and Threads

- A task is the basic unit of resource allocation.
 - It includes a virtual address space, access to resources, and may include one or more threads.
- A thread is a sequence of instruction executions.
 - All threads within a task share the resources of that task.



Tasks and Threads (cont.)

- All communication among threads is done by sending messages between ports.
- When a task is created, it is assigned several ports by the kernel
- The kernel manages all communication among ports by assigning send and receive permissions.

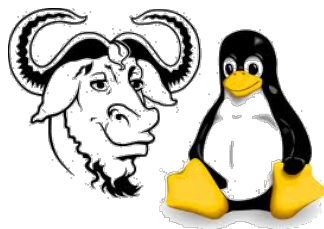


Problems with Mach

- Mach was not commercially successful because of the overhead in interprocess communication.
- System performance was slower than the traditional kernel.
- Mach had trouble handling memory when the physical memory ran low.

Present Day Unix™

- Workstations: Ubuntu, Gnu, Darwin, Windows, Mac OSx, MINIX, Xsystem, FreeBSD, Fedora, Open BSD ..
- Servers: HP-UX, AIX (IBM), Solaris (Oracle), Novell SLES, Red Hat Enterprise Linux
- Network: Cisco, Juniper



FreeBSD®



redhat.



Present Day Unix™

- **Embedded Devices:** Ezlink Card Readers, Aircon temperature control & more
- **Mobile Devices:** Compaq's iPAQ, Nokia, Motorola, Android tablets & mobile phones, Blackberry Playbook, iPhone, iPad, iPod, GPS system
- **Gaming:** Nintendo DS Lite, Sony Playstation 3, Xbox 360





In Conclusion...

- Unix™ has become the base for many kinds of computing systems
 - Low cost - no renewals or fees per workstation
 - Stability - no downtime due to its multi-processing capabilities & efficient kernel
 - Expandability - various hardware are supported
 - Development possibility
 - Customization - run company requirements
 - **Networking capabilities – it is a pioneer**
- Unix™ is the preferred choice by companies to run their File and Network Servers.



Operating Systems

Chapter 3: Concepts



Evolution of Operating Systems

- **Early Systems (1950)**
- Batch Operating System (1960)
- Interactive Operating System
- Multi-programmed Batch Systems (1970)
- Time-Sharing and Real-Time Systems (1970)
- Multiprocessor Operating Systems (1980)
- Distributed Operating System
- Networked/Distributed Systems (1980)
- Multiuser Operating System
- Multithreaded Operating System
- Web-based Systems (1990)

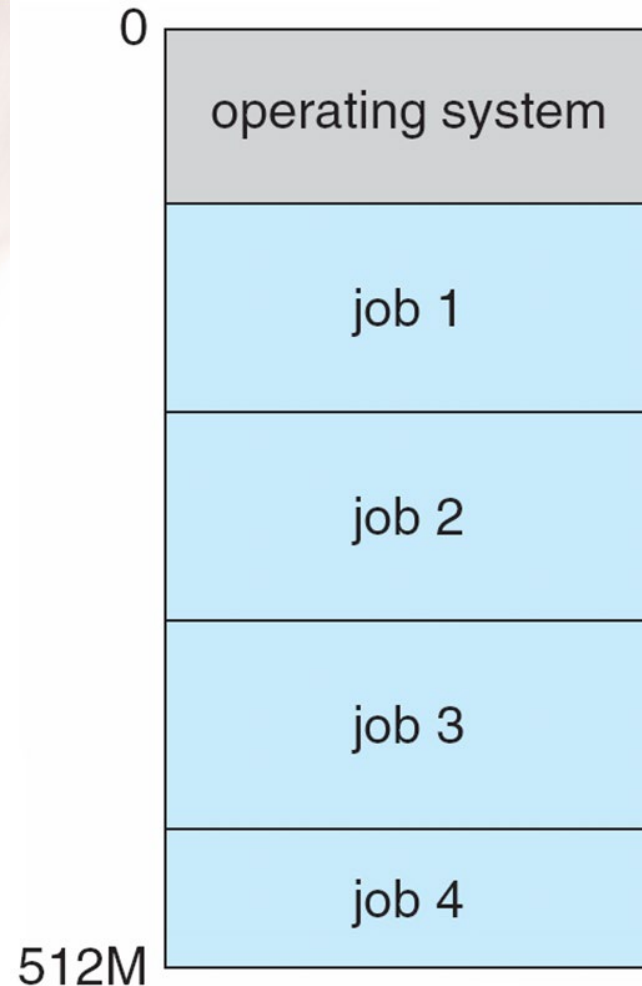
- The first operating system on microcomputer platform was Control Program for Microcomputers (CP/M)
- Intel 8080 in 1974 by Gary Kindall



Multiprogrammed Batch Systems

- First developed for batch systems in the 60s.
 - Go to the computer center and give them your program (stored on punch cards).
 - The computer operator “batched” several jobs together and loaded them into the computer.
 - Come back at 5:00 to get the results of your program.
- Batch systems are *non-interactive*.

Memory Layout for Multiprogrammed System



Automatic Job Sequencing allows control from one batch to another to be transferred automatically.

This process continues, until all of the programs presented in each batch are finished.

To implement automatic job sequencing, a program is used

This program is called a resident monitor



Resident Monitor

- A resident monitor is a program. It is also known as job sequencer.
- There is no idle time between exécutions.
- Disadvantages of a batch operating System
 - The CPU is often idle
 - It is difficult to provide the desired priority to the different programs

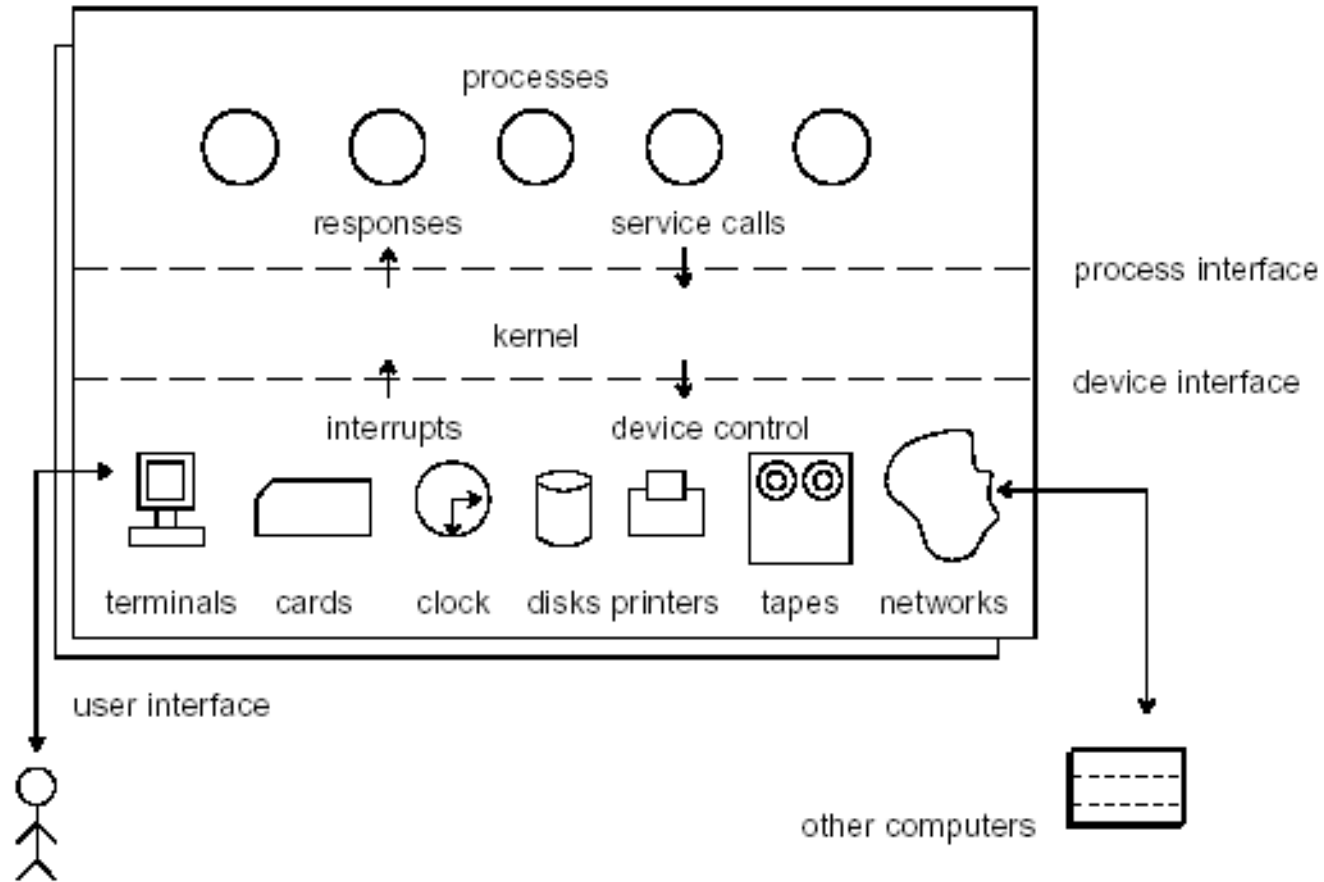


Time-Sharing

- Batch multiprogramming does not support interaction with users.
- Time-sharing extends Batch Multiprogramming to handle multiple interactive jobs – it's **Interactive Multiprogramming**.
- Multiple users simultaneously access the system through commands entered at terminals.
- Processor's time is shared among multiple users.



Time-sharing Architecture





Why Time-sharing?

- In Time-sharing (multitasking) the 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 (process) executing in memory.
 - CPU scheduling supports several jobs ready to run at the same time.
 - 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.



Why did Time-Sharing work?

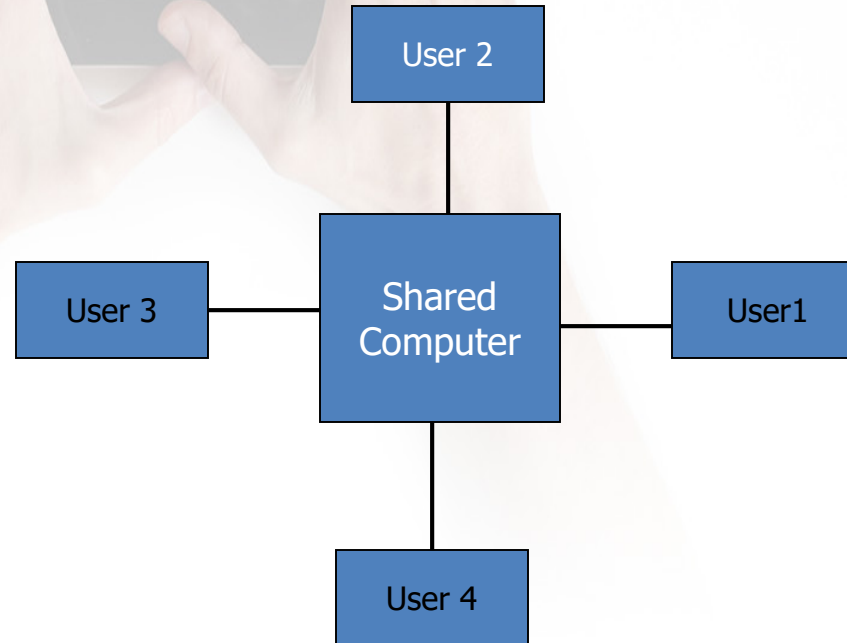
- Because of slow human reaction time, a typical user needs **2 seconds** of processing time per minute.
- Then many users should be able to share the same system without noticeable delay in the computer reaction time.
- The user should get a good response time.
- It fit the economic basis of the mainframe computer installation.



Time-Sharing Systems (Multitasking)

- Logical extension of multiprogramming termed *multitasking*.
- Quite often sitting at terminal using a “command line” interface to interact with computer.
 - Types in commands from keyboard.
 - A system program called a *shell* reads command from the command line and makes OS system calls to carry out commands.
- OS switches between user’s programs very quickly, generally in round-robin fashion.

Time-Sharing Systems (Multitasking)





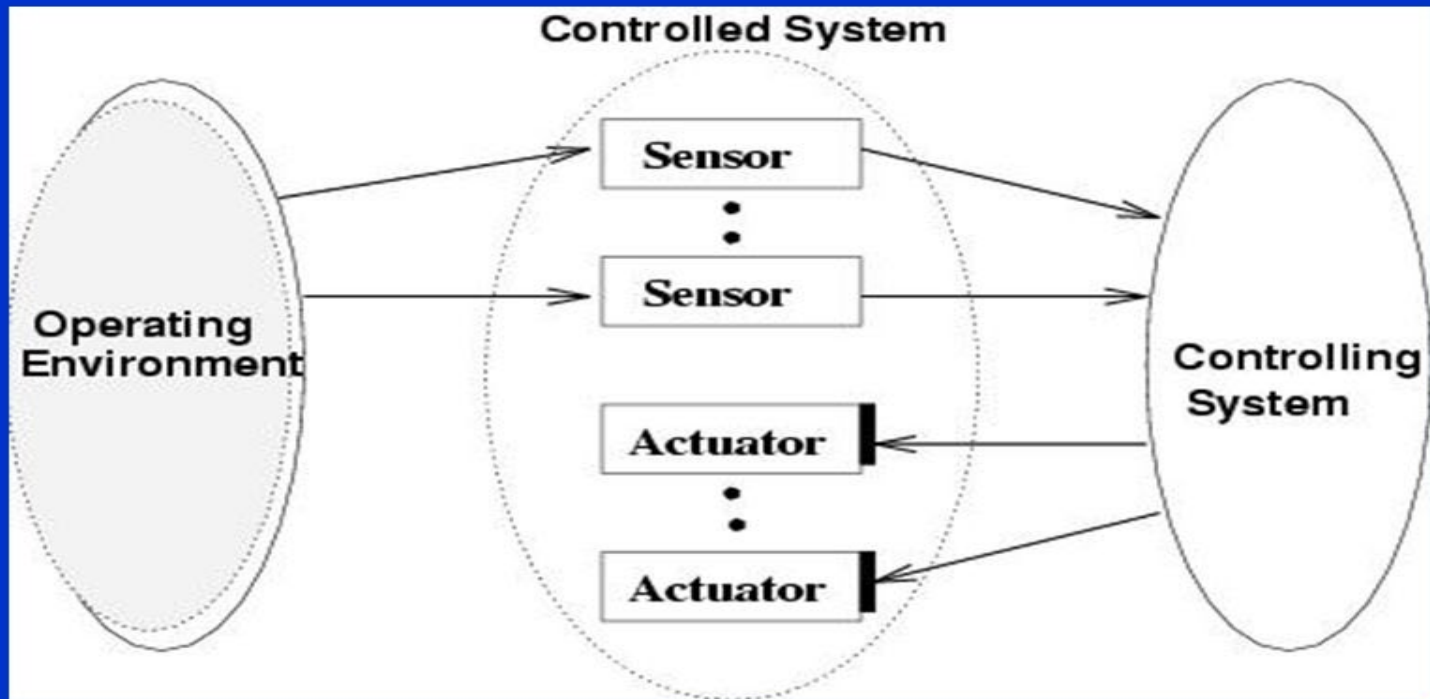
Special Purpose Systems RTOS

- **Real-time Embedded Systems**

- Found in car engines, microwave ovens etc. They have specific tasks and the systems they run on are usually primitive and so the O/S provides limited features. They have little or no GUI.
- Embedded systems almost always run RTOS. This is used when rigid time requirements have been placed on the operation of the CPU or flow of data.
- Sensors bring data to the computer which then analyzes this data. E.g.s of real-time system include medical imaging systems, weapon systems, automobile engine fuel injection systems.



A typical real-time embedded system





Real-Time Operating Systems RTOS

- Note that not all Operating Systems are general-purpose systems.
- Real-Time (RT) systems are dedicated systems that need to adhere to deadlines , i.e., **time constraints**.
- Correctness of the computation depends not only on the logical result but also on the time at which the results are produced.



Hard Real-Time Systems

- Hard real-time system must meet its deadline.
- Conflicts with time-sharing systems, not supported by general-purpose OSs.
- Often used as a control device in a dedicated application:
 - Industrial control
 - Robotics
- Secondary storage limited or absent, data/program is stored in short term memory, or Read-Only Memory (ROM).




Soft Real-Time Systems

- Soft real-time system more jitter than HR-T:
 - Deadlines desirable but not mandatory.
 - Limited utility in industrial control or robotics.
 - Useful in modern applications (multimedia, video conference, virtual reality) requiring advanced operating-system features.

A close-up photograph of a person's hands holding a smartphone. The left hand is positioned vertically on the left side of the phone, while the right hand's index finger is touching one of the social media icons on the screen. The screen displays a grid of icons for Facebook (f), Twitter (bird), Instagram (camera), Pinterest (p), Google+ (G+), and WhatsApp (phone handset). The background is a soft-focus green and yellow gradient.

Personal/Desktop Computers (1)

- *Personal computers* – computer system dedicated to a **single user**.
- I/O devices: keyboards, mice, display screens, small printers.
- User convenience and responsiveness.
- Can adopt technology developed for larger operating system; often individuals have sole use of computer and do not need advanced CPU utilization or protection features.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux)



(2) Personal/Desktop Systems

- Traditional personal computer **blurring over time.**
- Office environment:
 - PCs connected to a network, terminals attached to mainframe or minicomputers providing batch and timesharing.
 - Now portals allowing networked and remote systems access to same resources.
- Home networks:
 - Used to be single system, then modems
 - Now firewalled, networked.

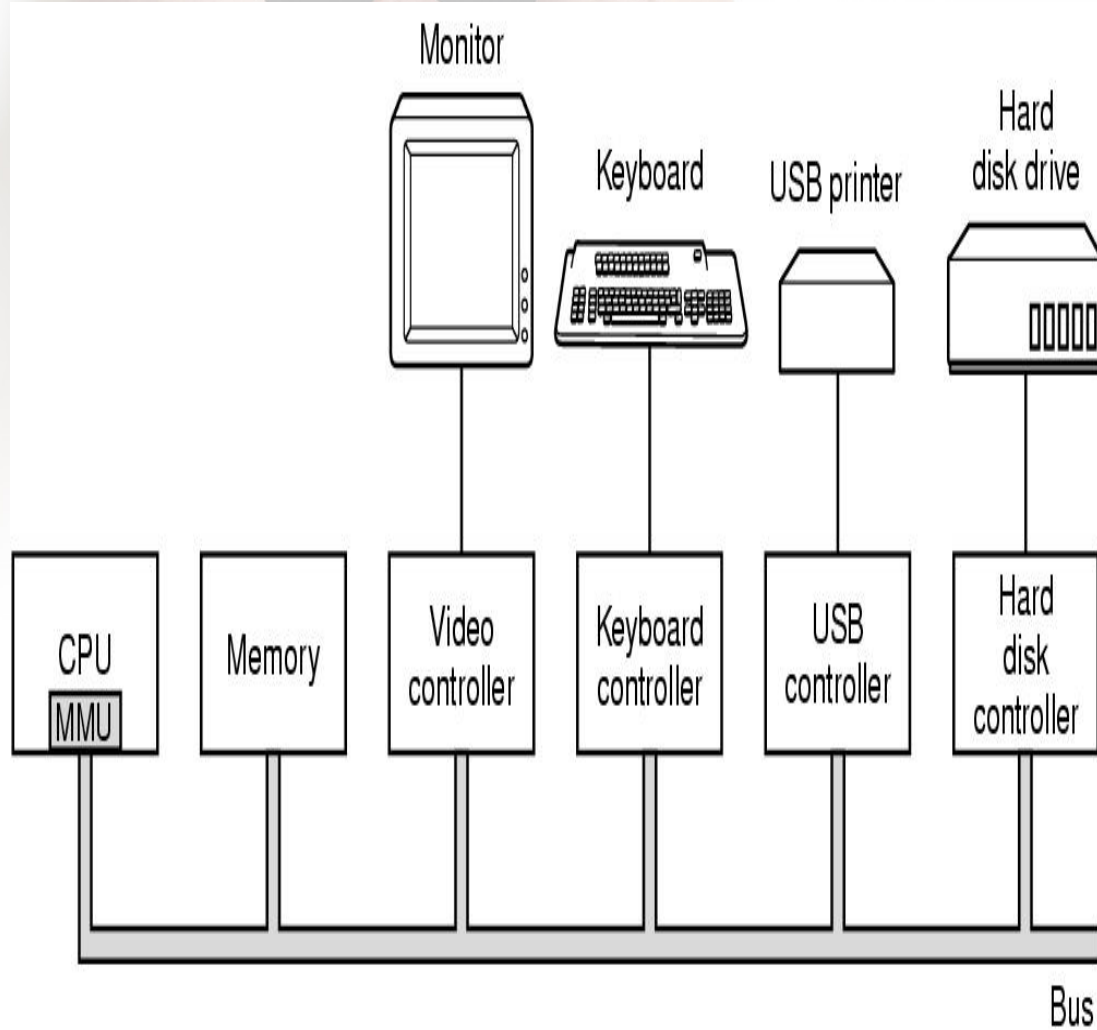


Two categories of Computer Systems

- **Single Instruction Single Data (SISD)** –
 - single processor executes a single instruction sequence to operate on data stored in a single memory.
 - This is a Uniprocessor.
- **Multiple Instruction Multiple Data (MIMD)** -
 - a set of processors simultaneously execute different instruction sequences on different data sets.
 - This is a Multiprocessor.



Components of a simple personal computer





Multiprocessor Systems

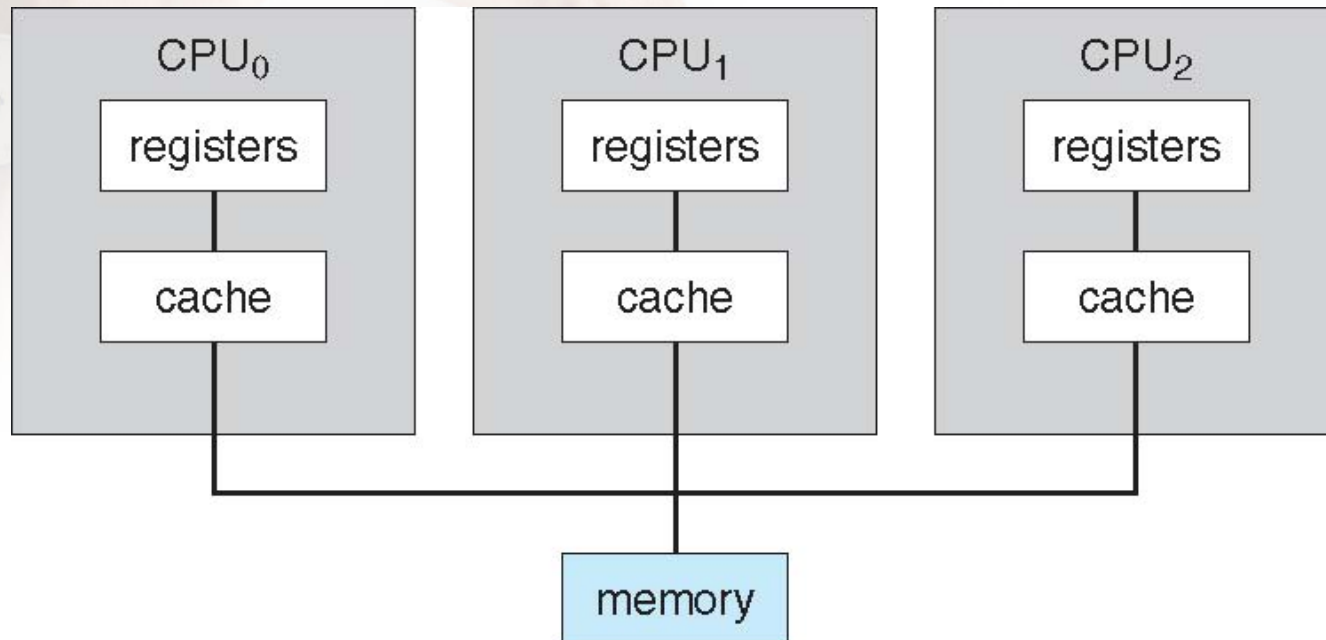
- System with several CPUs in close communication:
 - processors share memory and a clock.
 - communication usually takes place through the shared memory.
- Also known as parallel systems, tightly-coupled systems.
- Multiprocessors systems growing in use and importance
 - advantages include:
 - Increased throughput
 - Economy of scale
 - Increased reliability – graceful degradation or fault tolerance.



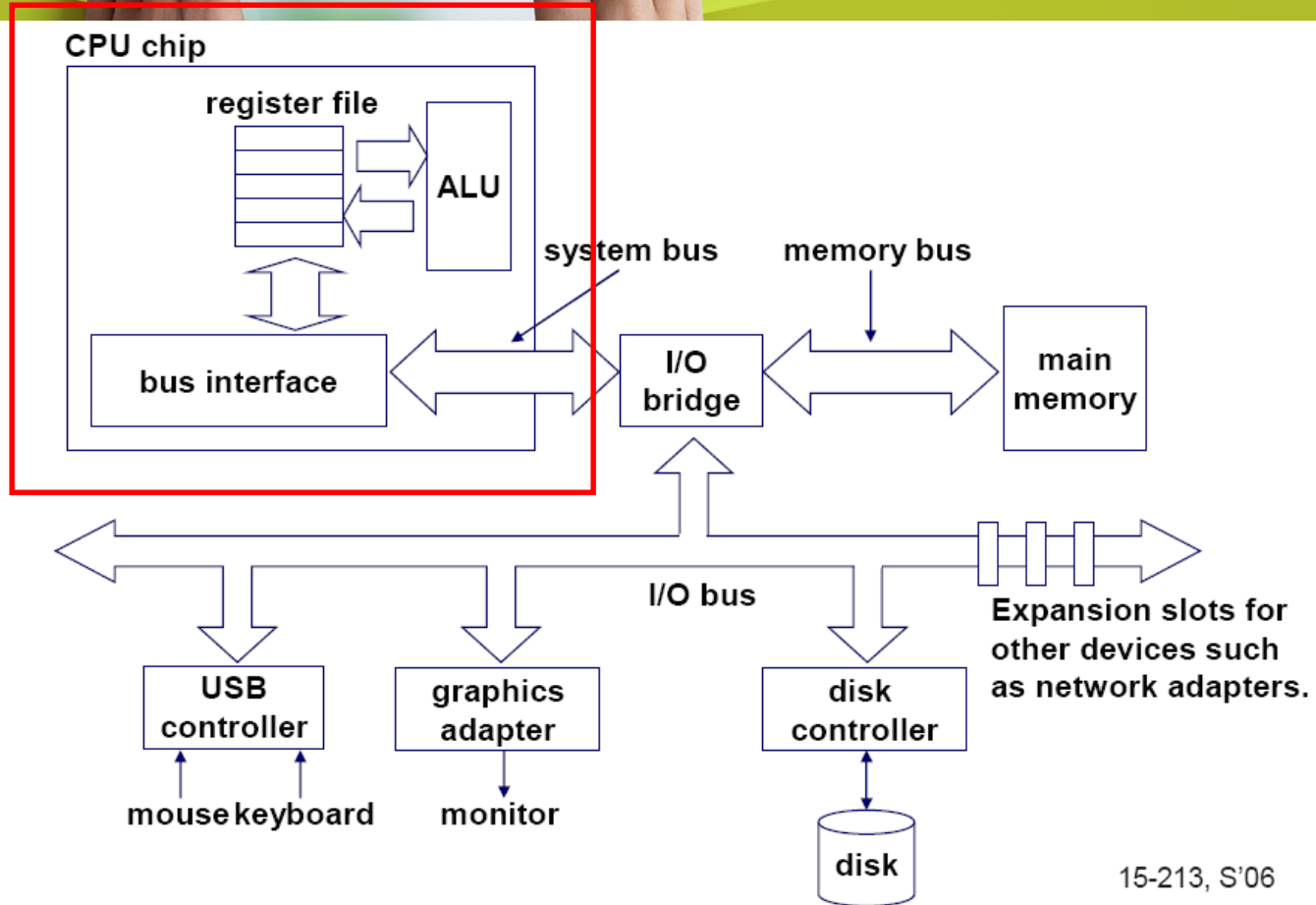
Types of Multiprocessor Systems

- *Asymmetric Multiprocessing*
 - master processor schedules and allocates specific work to slave processors.
- *Symmetric Multiprocessing (SMP)*
 - Each processor runs an identical copy of the OS.
 - Typically each processor does self-scheduling from the pool of available processes.
 - Most modern operating systems support SMP.

Symmetric Multiprocessing Architecture



Single-core computer

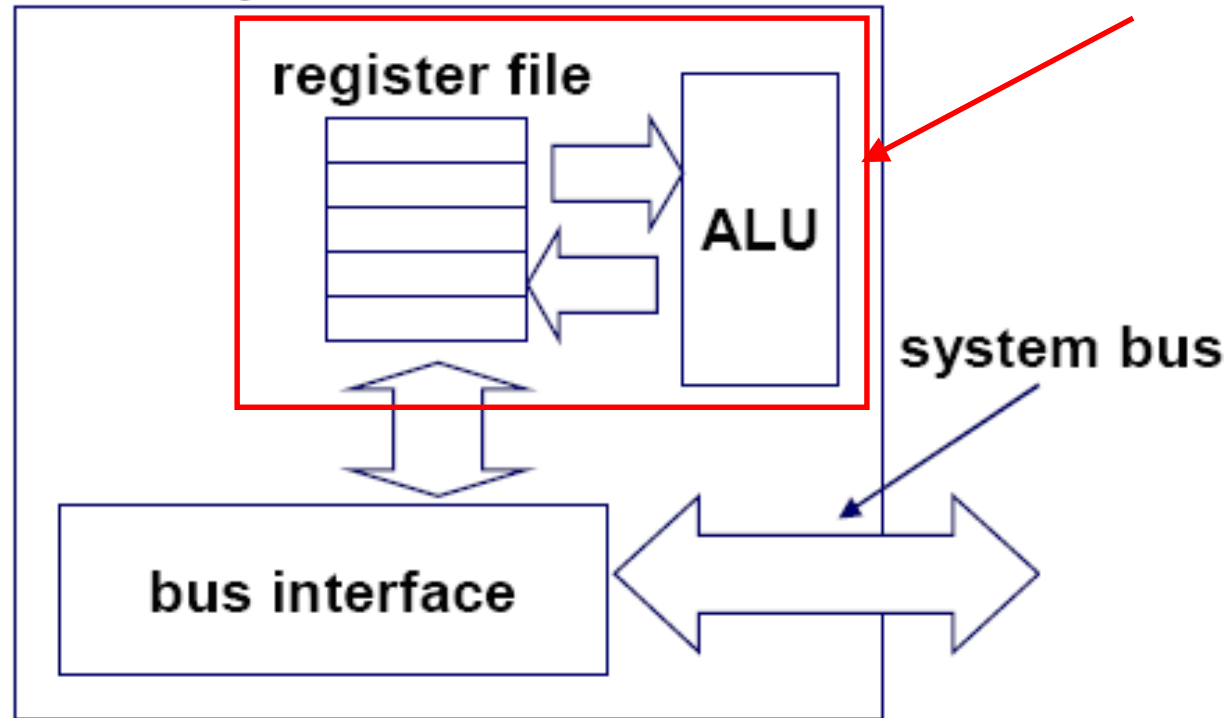




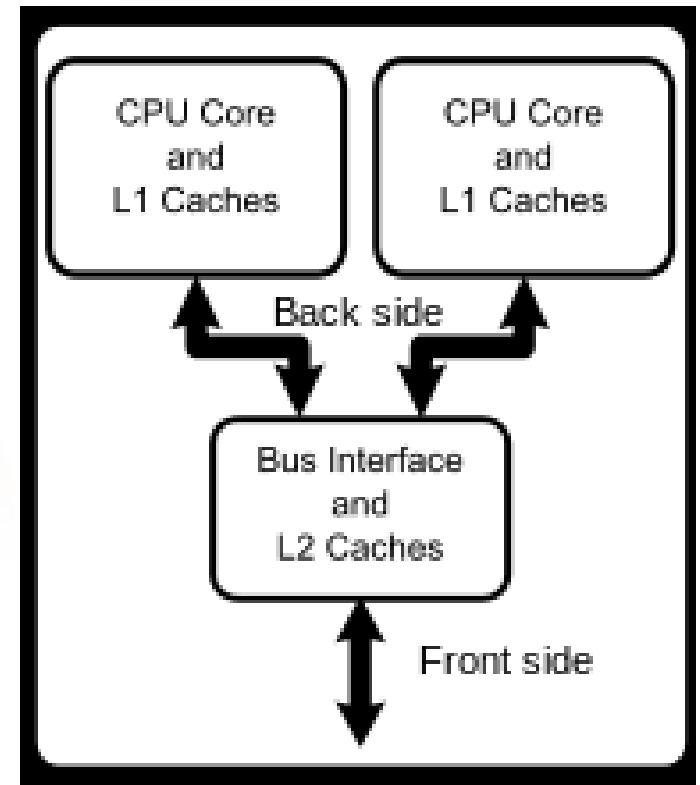
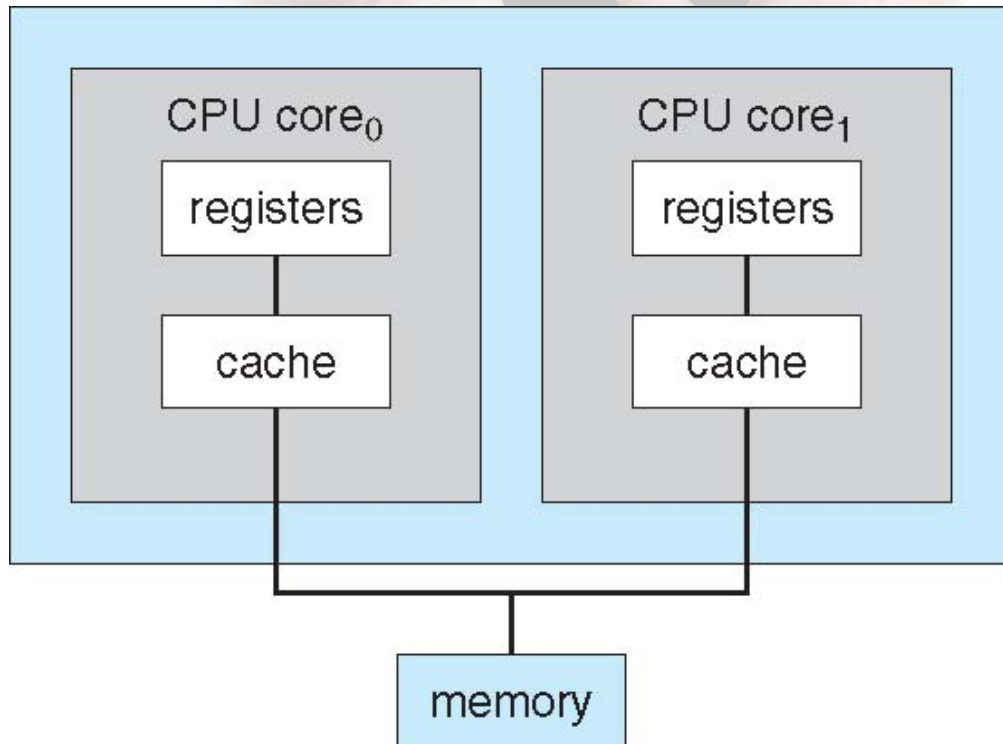
Single-core CPU chip

CPU chip

the single core

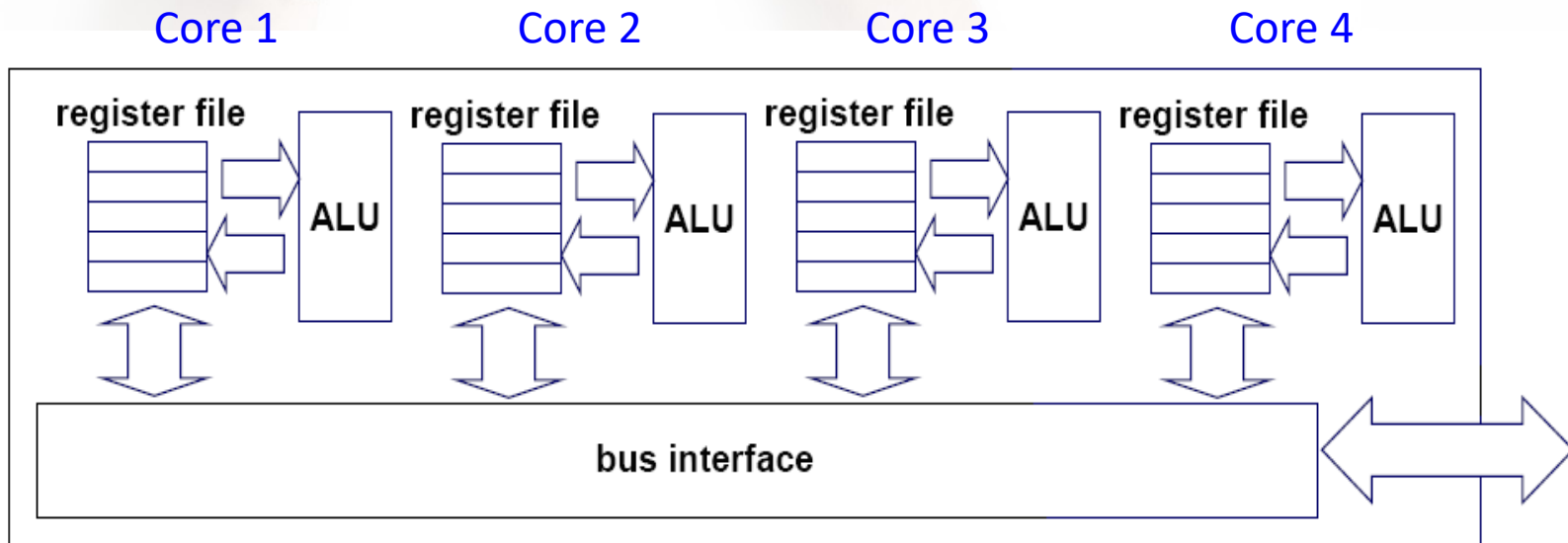


A Dual-Core Design



Multi-core architectures

Replicate multiple processor cores on a single die (integrated circuits).



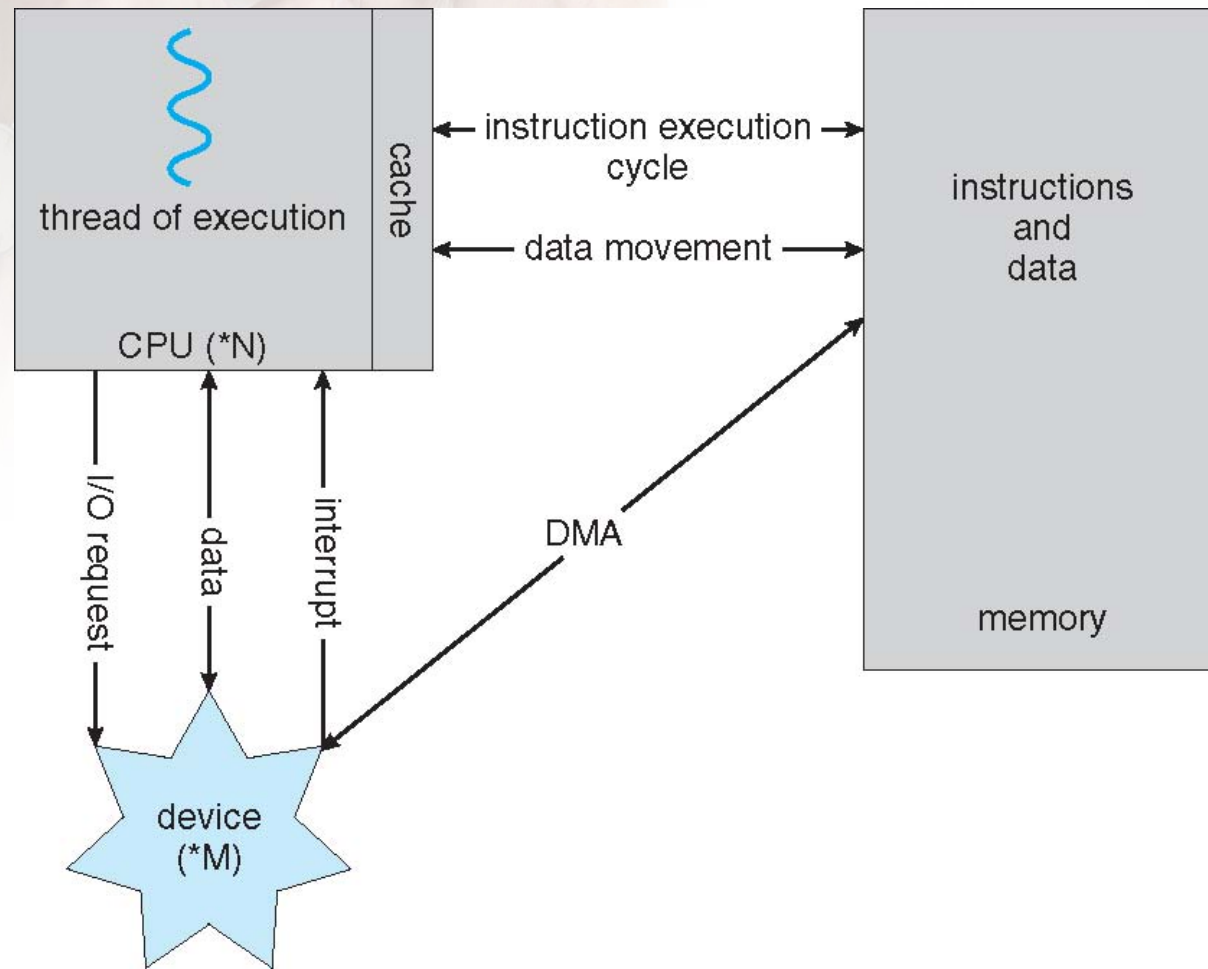
Multi-core CPU chip

A close-up photograph of a person's hands holding a smartphone. The screen displays a grid of social media sharing icons: Facebook (f), Twitter (bird), Email (envelope), Pinterest (p), Google+ (G+), and WhatsApp (phone handset). The background is a solid light green color.

Interaction with the Operating System

- OS perceives each core as a separate processor
- OS scheduler maps threads/processes to different cores
- Most major OS support multi-core today: Windows, Linux, Mac OS X, ...

How a Modern Computer Works





Definitions

- A program is a set of instructions which is prepared to perform a specific assignment if executed by a computer.
- A program need not be online; it could be stored on a flash memory and placed in one's pocket.
- A program is not an active entity. It is completely passive.



Definitions...

- The operating system creates a **process** from a program.
- To do so, it has to perform many activities, like assigning a name, allocating space, (partially) loading the corresponding program, etc.
- Roughly speaking, **A process is an active program.**
- A process is created to run a program by using computer facilities; like a human who is born to live his life.



Single-Programming

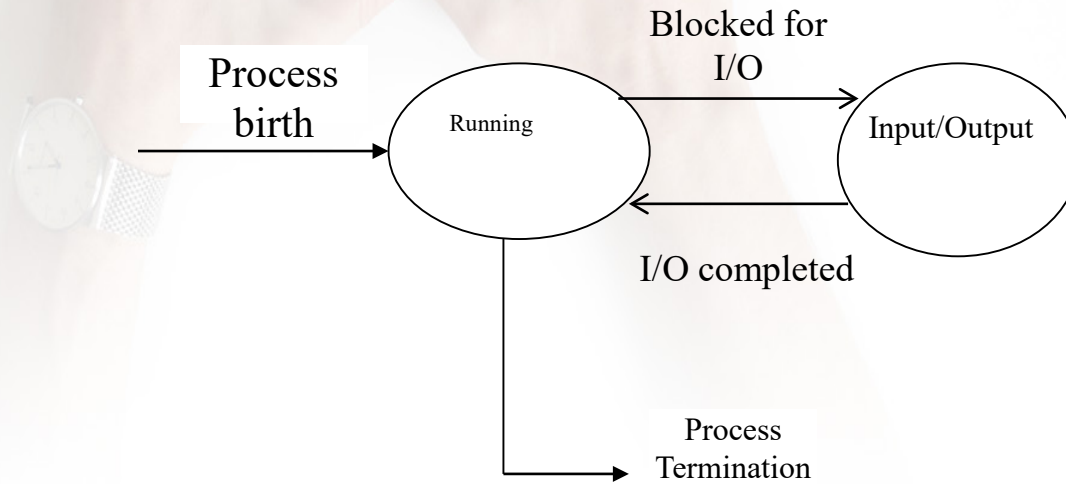
- In a single-programming environment there exist at the most one process at any given time; thus there is usually one ongoing activity at a time.
- From many devices within the computer often one device is active at any given time.
- This means, if a process has asked for a data to be entered by the user, the system has to wait until this data is entered before being able to proceed.
- If this data entry takes one second the CPU could have done millions of instructions if it did not have to wait.
- With single-programming the computer facilities are not used in an efficient manner.



Process States in Single-Programming

- With single-programming, right after a process is born the system starts executing its corresponding program's instruction.
- The instruction execution continues until the process needs to read some data from an input device or wants to write some results on an output device.
- There are special purpose processors called Input/Output (I/O) processors for transferring data from input devices to main memory and from main memory to output devices.
- It is understandable that such a processor will perform the specific task better than a general-purpose processor, i.e., CPU.
- While an I/O operation is in progress, the CPU has to wait and do nothing. After the I/O operation is completed, the CPU will resume the execution of the instructions.
- This cycle, of going through **process states** of running and input/output, may be repeated over and over, until the job is completed or, for some reason, the process is aborted.

Process's life cycle



The life cycle of processes in single-programming environments



Processor wait ratio

- If the average execution time of a program with single-programming is e and the average I/O time is b , then the following ratio is the CPU wait fraction (w). It is actually the fraction of the time the CPU is idle.

$$w = \frac{b}{e + b}$$

- For example, if execution time of programs is 10, of which 9 seconds is spent on I/O, then $w = 9/10 = 0.9$. This means, on the average, 90% of the CPU time is wasted.



Multiprogramming

- Basic idea of multiprogramming:
 - Keep multiple jobs in memory.
 - When one job blocks on I/O (or other events), the operating system:
 - Starts the I/O operation.
 - Switches to another job that is ready to execute.
 - Now the CPU and I/O device are executing in parallel.
 - When the I/O device has completed request, it generates an interrupt to inform the CPU.
 - Virtually all general purpose computers support multiprogramming.



The Multiprogramming Concept

- Multiprogramming is a technique that allows more than one program to be ready for execution (process) and provides the ability to switch from one process to another, even if the former is not completed.
- Of course, sometimes in the future we will have to switch back to the first process and resume (not restart) its computation.
- This technique works for both single-processor (like our personal computers) and multiprocessor (such as large main frame) computers.
- Multiprogramming is mainly accomplished by the operating system. The hardware provides some specific circuitry that may be used by the operating system in the course of facilitating multiprogramming.



Multiprogramming and PCs

- Do we need multiprogramming for PCs? Yes. All PC users like to run **many applications simultaneously**. Nobody runs for example an Internet explorer looking for an information while staring at the monitor for the results for a long time.
- In this era of computer usage, every general-purpose operating system must have the following capabilities:
 - It must provide an environment to run processes in a multiprogramming fashion.
 - It must act as a service provider for all common services that are usually needed by computer users, such as copying files, making new folders, compressing information, sending and receiving messages from other computers in the network, etc.
 - Its user interface must be easy to use and pleasant to work with.



Multiprogramming productivity

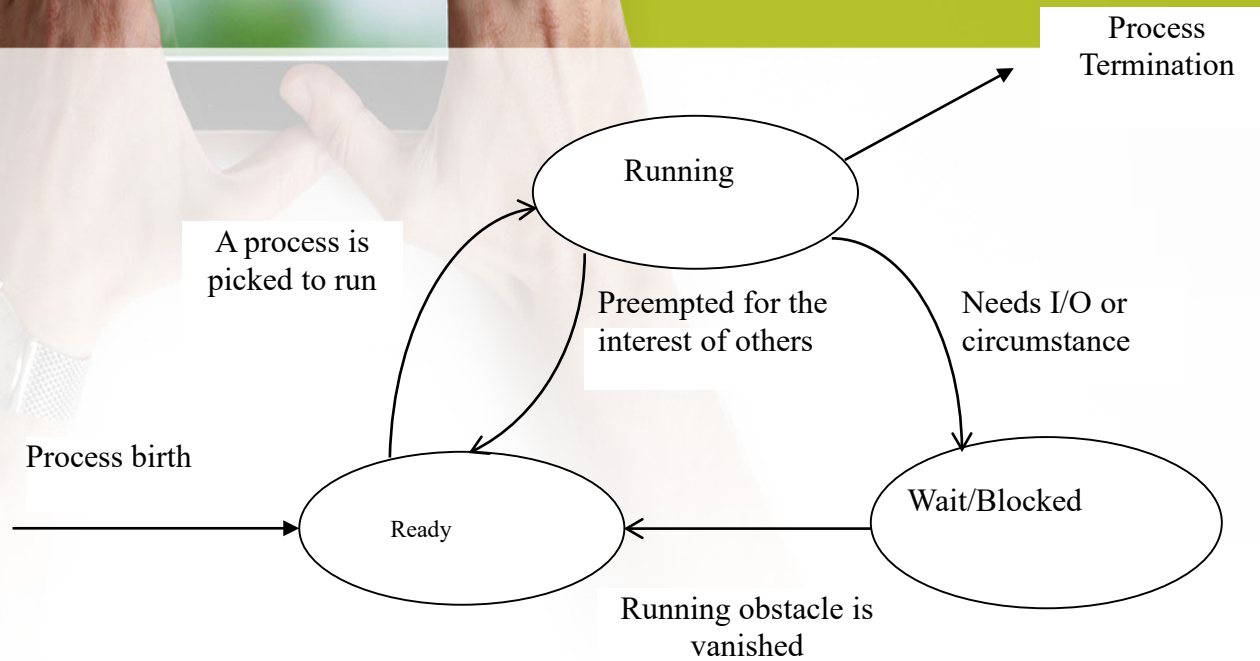
- Multiprogramming increases system productivity. If CPU wait time is represented by w in single-programming environment, the CPU wait time decreases to approximately w^n for a system running n processes simultaneously.
 - Example: If $w = .9$ then $w^5=0.59$; meaning that if we have five processes running simultaneously, the CPU utilization is increased by $(0.41-.10)*100 = 310\%$.
- By increasing the CPU utilization other device's utilization is also increased.



Process State Transition Diagram

- The life cycle of a process in multiprogramming is not the same as singleprogramming.
- A process may be ready to use the CPU to run its program while the CPU is running another program.
- The basic states are thus Ready, Running, and Wait/Blocked. Wait refers to a state in which the process is waiting for a device or an event and Blocked is for the case the process is waiting for its I/O to be completed by an I/O processor.

Process's life cycle



Basic process state transition diagram in multiprogramming



Requirements of Multiprogramming

- **Process Switching possibility:** the system must be able to safely switch from one process to another. This is called Context switching.
- **Direct Memory Access:** I/O processors must be able to directly access main memory without interference and conflicts.
- **The Interrupt System:** I/O processors and monitoring devices must be able to safely communicate with the CPU.



Multiprocessing

- If you think clearly, you will notice that we should have used multiprocessing instead of multiprogramming. This is true. Unfortunately, the term “multiprogramming” is recognized for this technique of the operating system and we will stick to it.
- On the other hand, “multiprocessing” is used for systems with more than one processor. Processors, in such a system, can collectively run many tasks simultaneously.



Multitasking

- Simultaneously executing programs are called **tasks**. Therefore, a system with the capability of **multitasking** allows users to activate more than one task, or application program, at a time. An Internet browser that searches for some information and A word-processing software that is activated to perform the word-processing task are applications.
- The operating system will switch between tasks based on the tasks current states and their requirements and priorities.
- Multitasking is only possible when **multiprogramming** is the fundamental capability of simultaneously executing pieces of software.
- Most modern operating systems, like UNIX, Linux, and Windows, support multitasking.



Process Deficiencies

- A process is created to run a program to perform a duty.
- What if we need to perform two or more similar duties?
- One approach is to create more than one exact same processes; each assigned to handle one of the two duties.
- This is a correct solution, but it spawns two major problems:
 - As the numbers of duties increase, the number of processes increases too, and very soon we will either run out of main memory or, in the case of virtual memory, we may reach an inefficient state of main memory.
 - By increasing the number of processes, the number of objects that compete for computer resources increases, too. It will led to an undesirable state in which many processes cannot complete their duty because they do not get the chance to use the resources needed.
- Thread is introduced to solve these problems

- Thread refers to a path through a program's instructions during its execution.
- Multithreading methodology allows more than one thread of execution for every process.
- Now, if we need to perform two or more similar duties we can create one process and from which create more than one thread; each assigned to handle one of the duties.
- All threads of a single process share the same:
 - address space
 - global data
 - files for storing and/or reading information
 - resources that are assigned to their corresponding process.



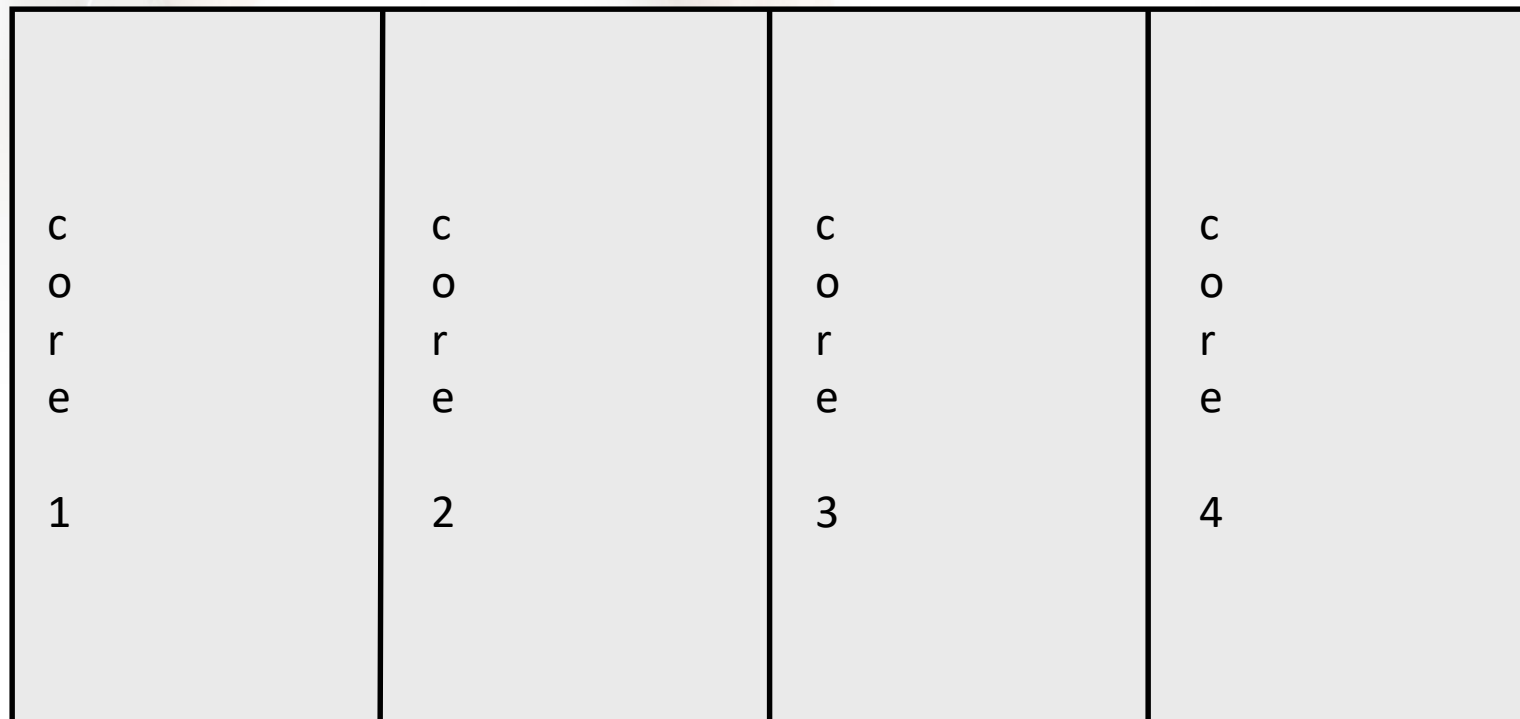
Multithreading

- A multithreading operating system is one that is capable of handling processes and threads at the same time and in which from each process the system is able to generate more than one thread.
- In such an operating system, there must be facilities for thread creation, deletion, switching, etc.
- Such an operating system allows users to generate more than one request to a process at a time. For example, a browser can be made to search simultaneously for more than one topic, even though there is only one copy of the “browser program” in main memory.
- The multiprogramming methodology and technique are essential in the implementation of multithreading. In this new environment, a thread becomes the smallest functional object to which CPU (or a PU) is assigned.

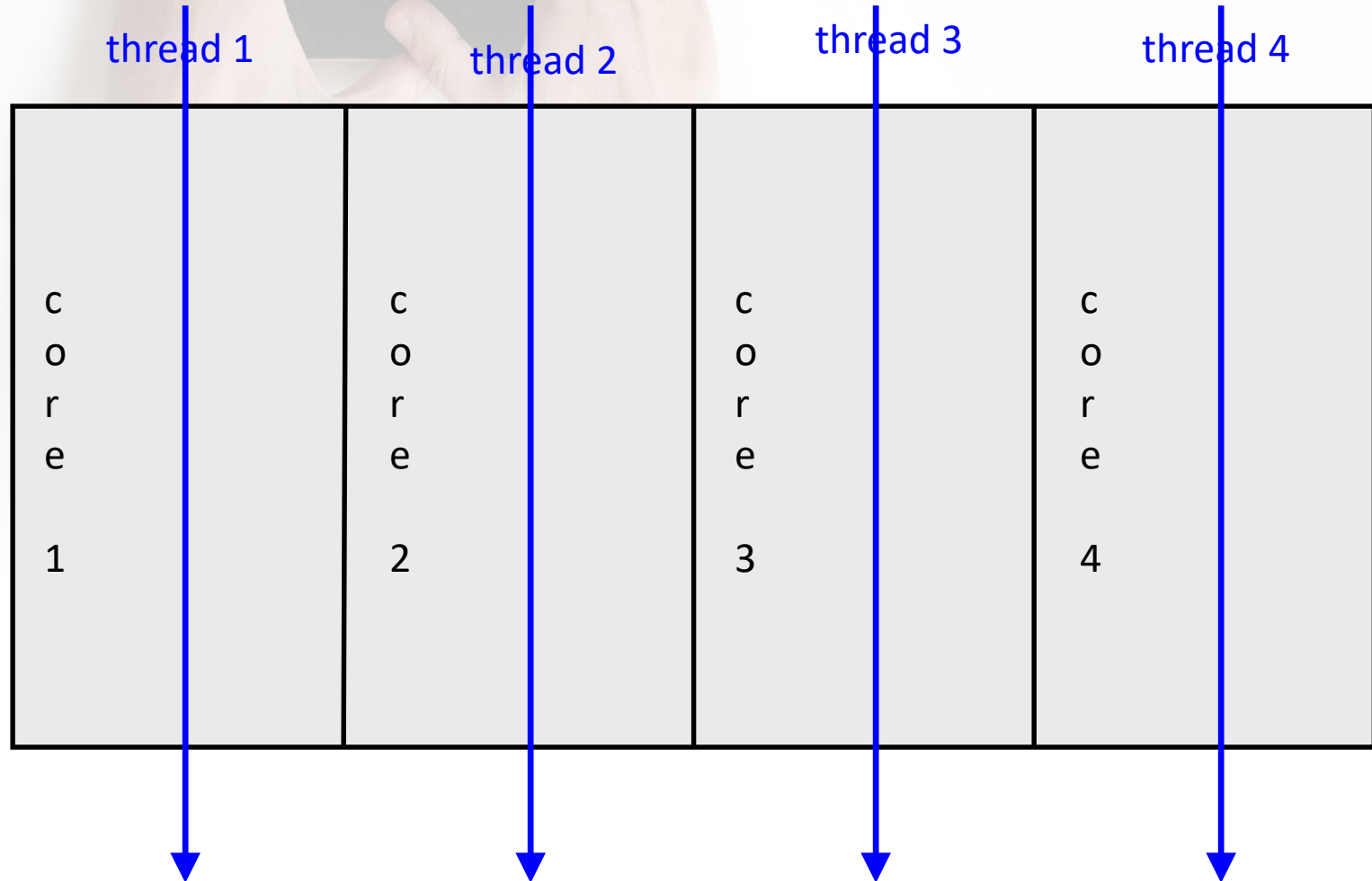


Multi-core CPU chip

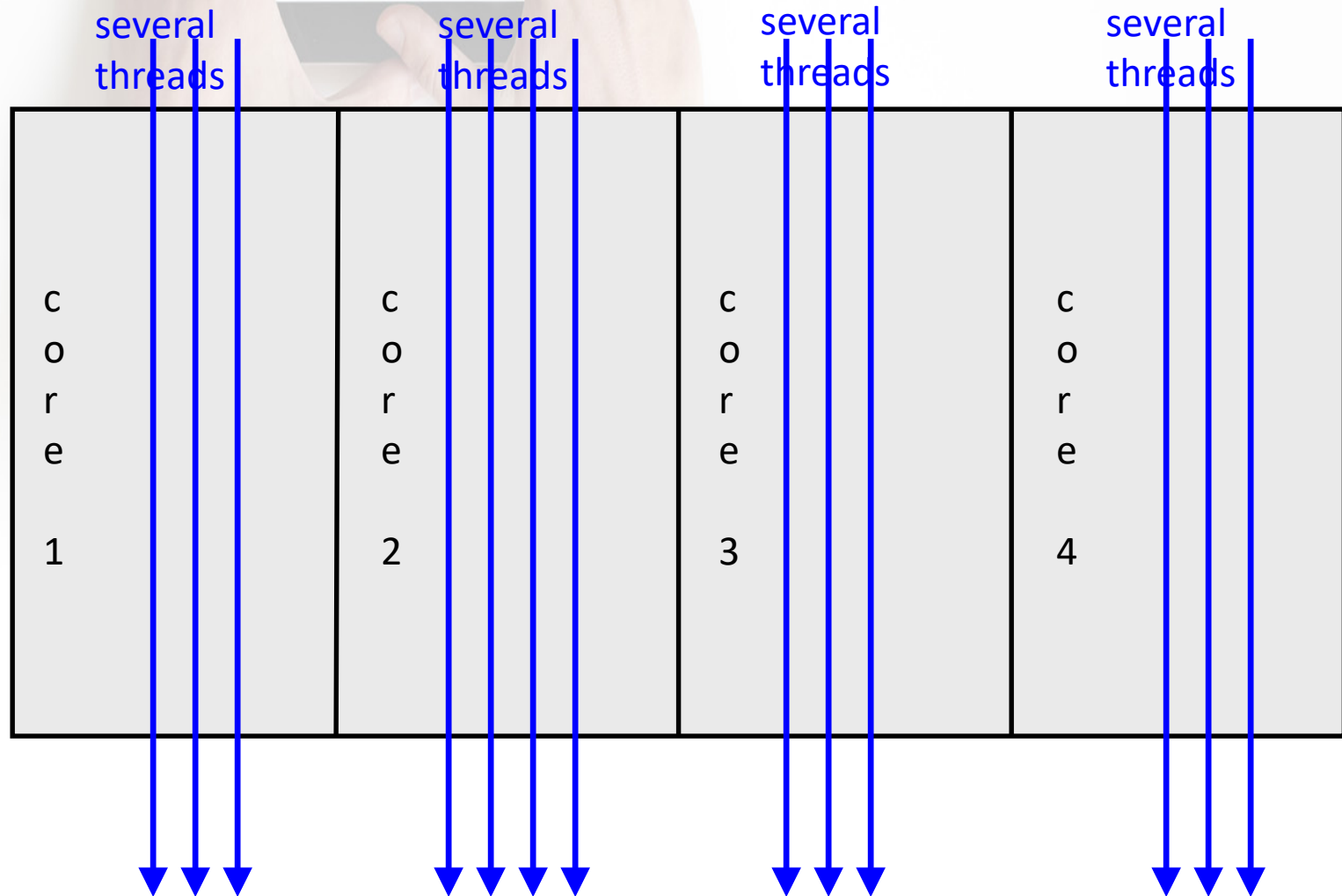
- The cores fit on a single processor socket
- Also called CMP (Chip Multi-Processor)



The cores run in parallel



Within each core, threads are time-sliced (just like on a uniprocessor)



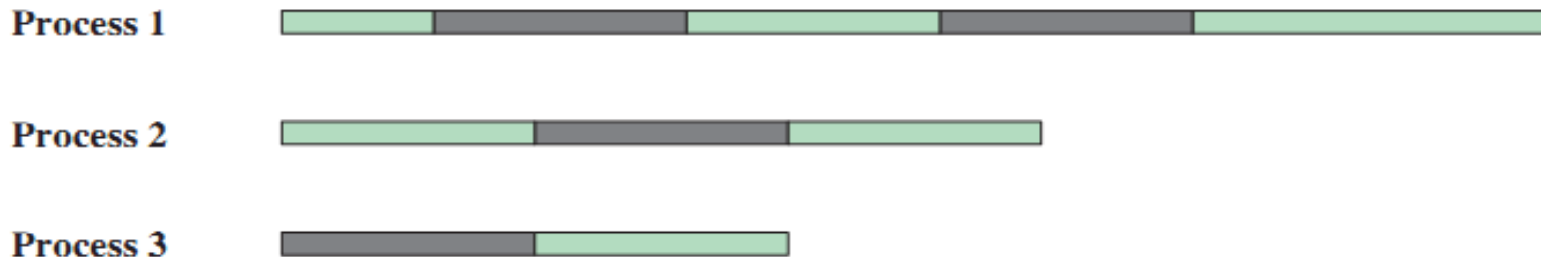


Multiprogramming vs. Multiprocessing

Time →

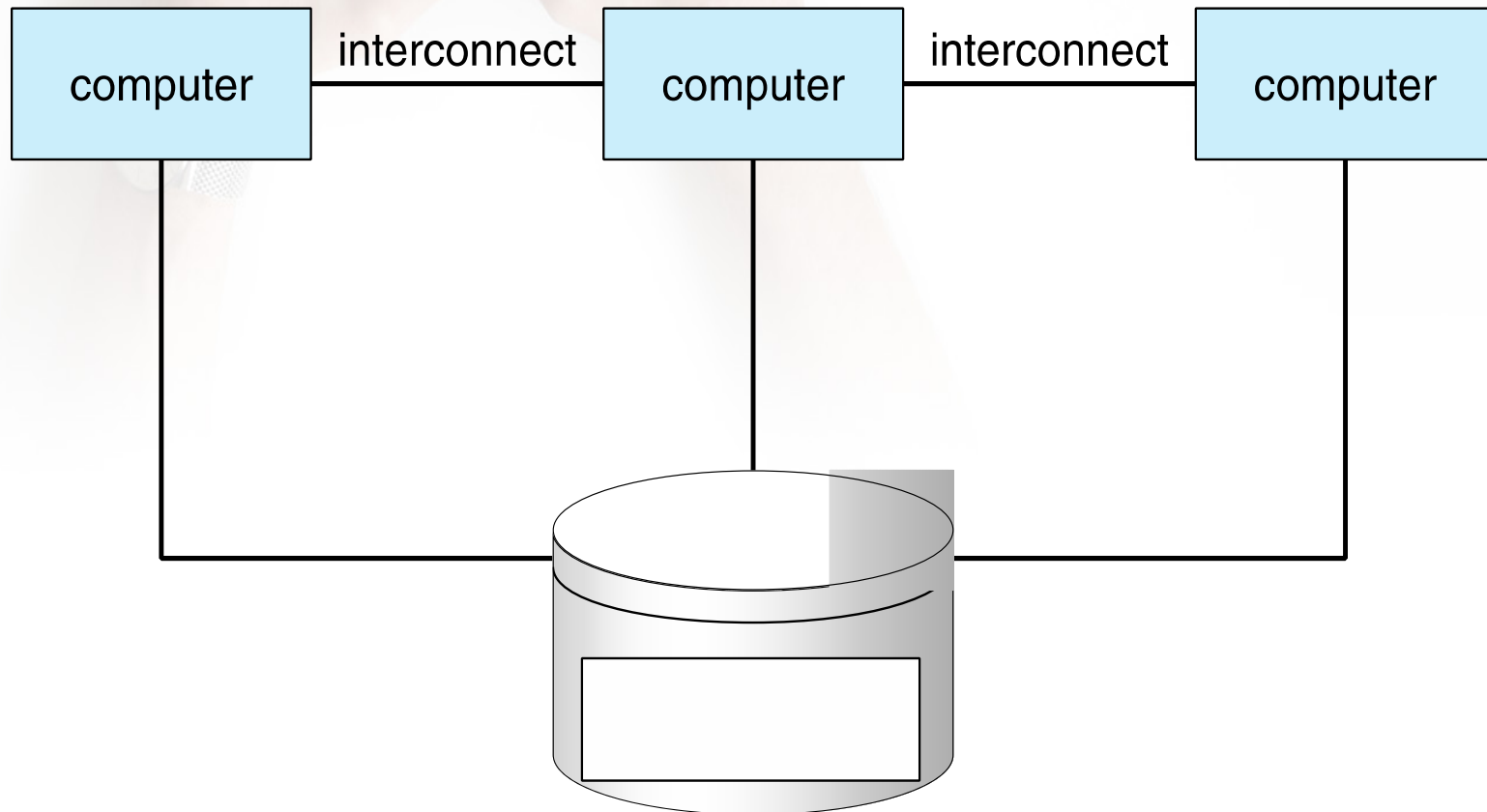


(a) Interleaving (multiprogramming, one processor)

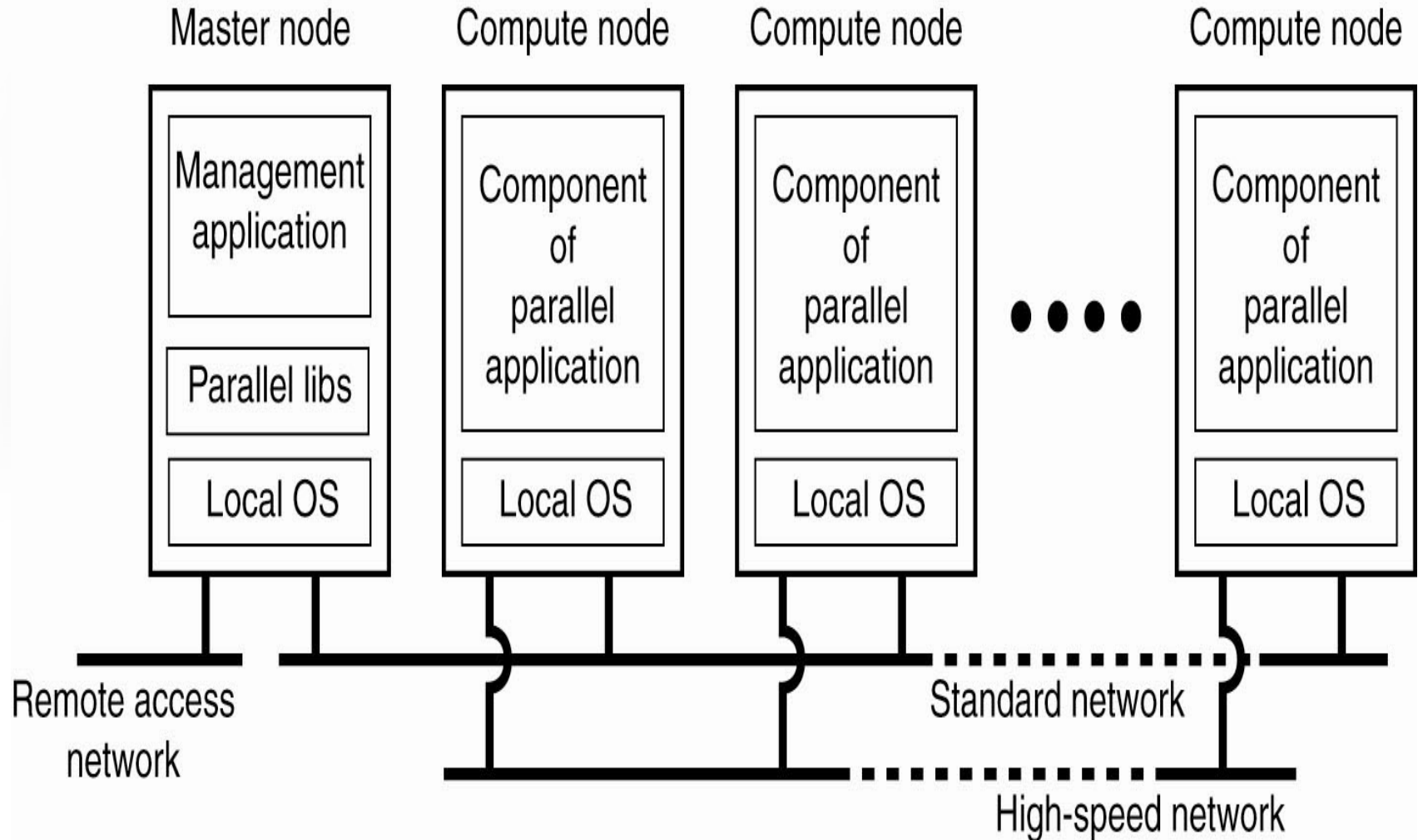


(b) Interleaving and overlapping (multiprocessing; two processors)

Clustered Systems Architecture



Architecture for Cluster Computing System





Clustered Systems (1)

- Like multiprocessor systems, but multiple systems working together.
- Also known as closely-coupled system.
- Clustering allows two or more systems to share external storage and balance CPU load:
 - processors also have their own external memory.
 - communication takes place through high-speed channels.
 - Provides high-availability



Clustered Systems (2)

- Usually sharing storage via a Storage-Area Network (SAN).
- Provides a high-availability service which survives failures:
 - Asymmetric clustering has one machine in hot-standby mode
 - Symmetric clustering has multiple nodes running applications, monitoring each other.
- Some clusters are used for high-performance computing (HPC) where applications must be written to use parallelization.



Types of Distributed Operating Systems

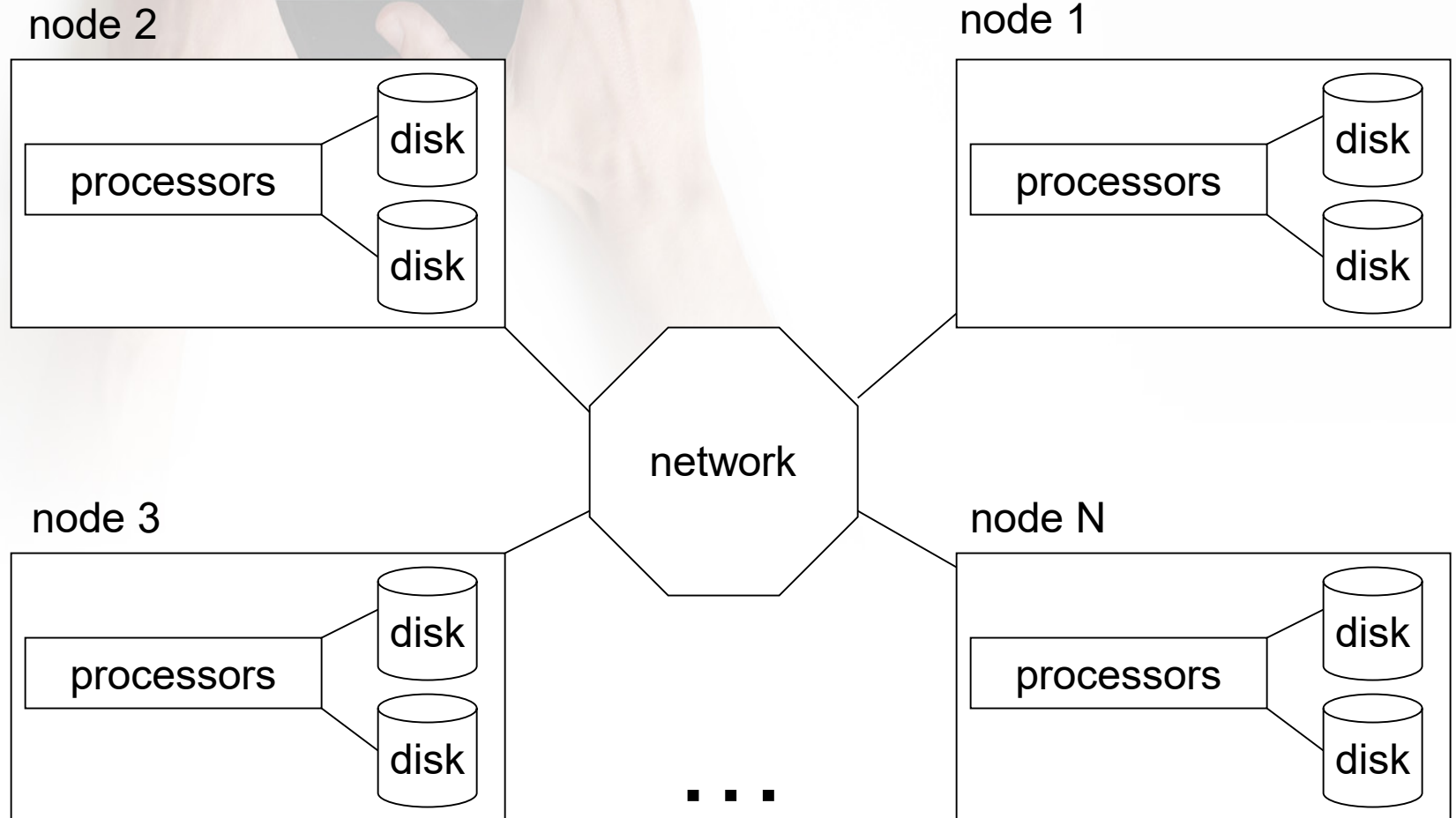
- Network Operating Systems
- Distributed Operating Systems



Networked Systems

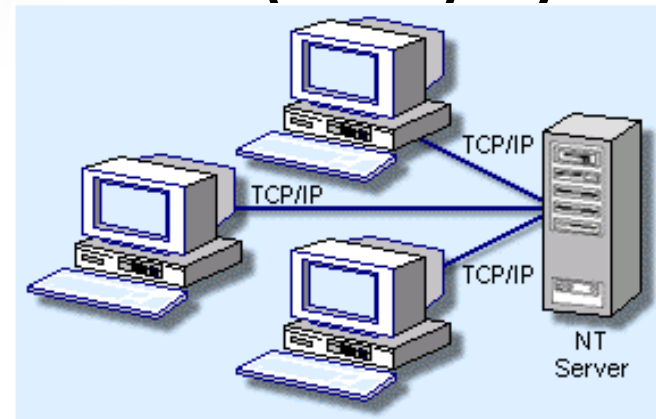
- Distribute resources and the computation among several physical processors.
- *Loosely coupled system:*
 - each processor has its own local memory.
 - processors communicate with one another through various communications lines.
- Advantages:
 - Resources Sharing
 - Computation speed up – load sharing
 - Reliability

Networked System Structure



Networked Systems

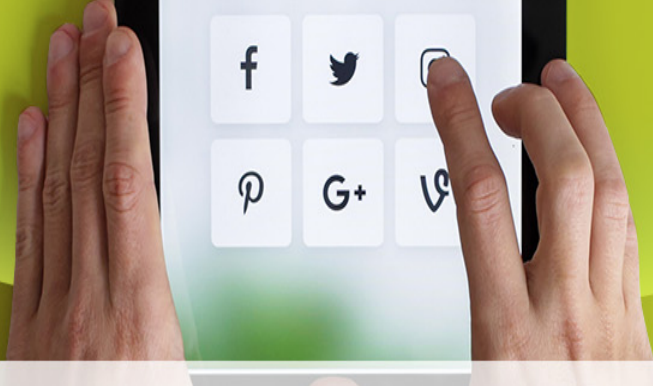
- Requires networking infrastructure.
- Most are Local area networks (LAN) or Wide area networks (WAN).
- May be either Centralized Sever or Client/Server or Peer-to-Peer (P2P) systems.



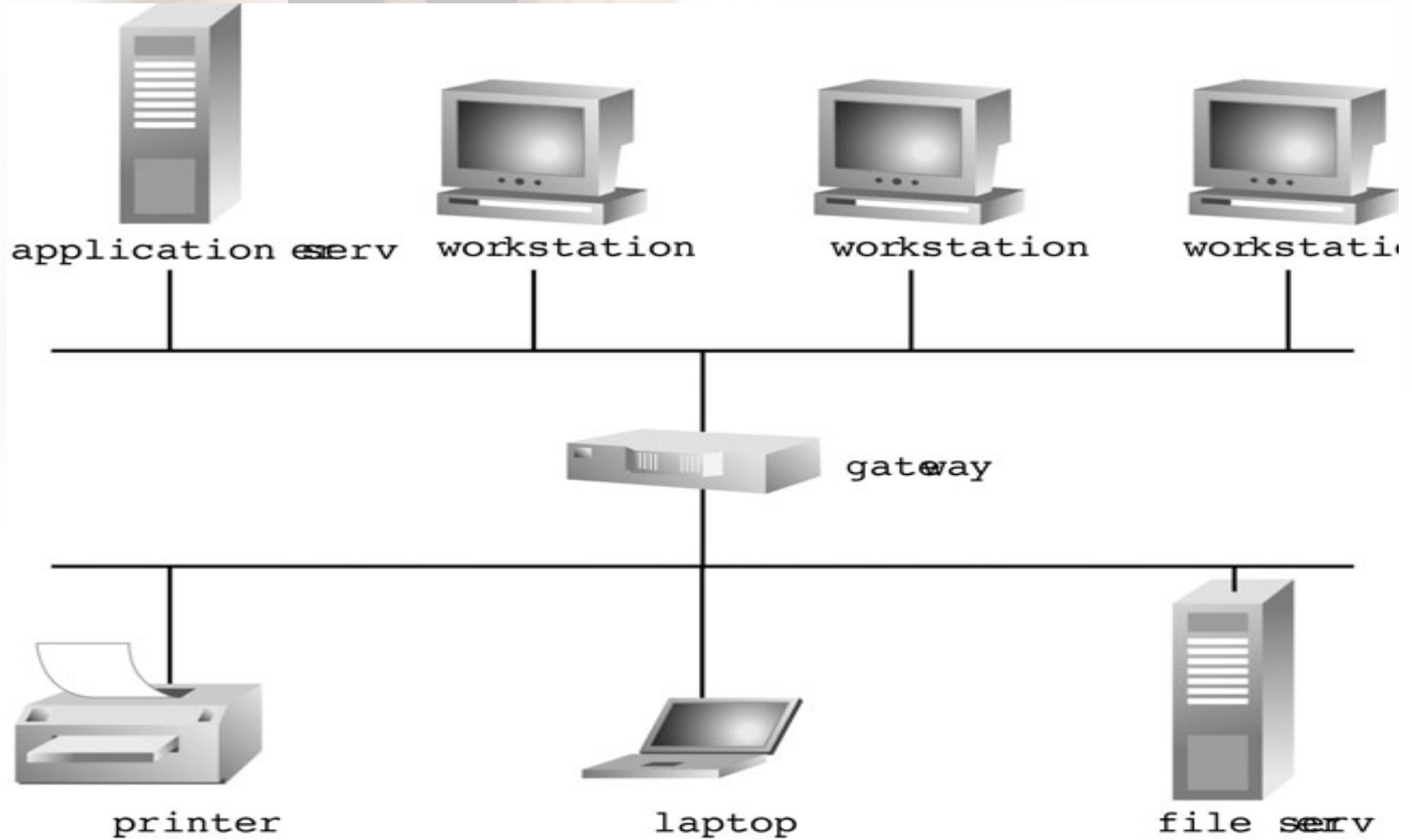


Local Area Network (LAN)

- LAN designed to cover small geographical area:
 - Multiaccess bus, ring, or star network.
 - Speed \approx 10–100 Megabits/second.
 - Broadcast is fast and cheap.
 - Nodes:
 - usually workstations and/or personal computers.
 - a few (usually one or two) mainframes.



Example of Local Area Network (LAN)



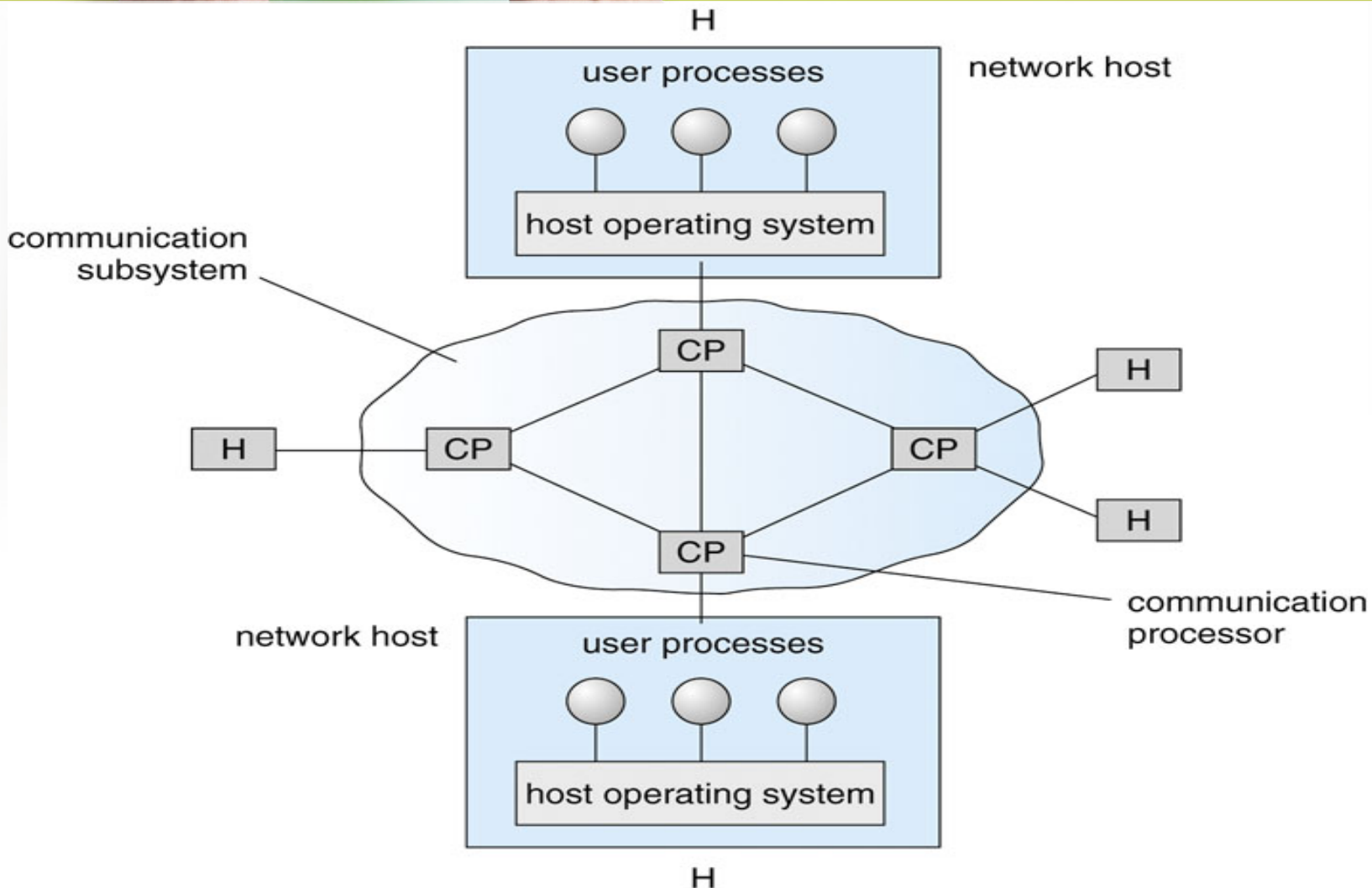


Wide-Area Network (WAN)

- Links geographically separated sites:
 - Point-to-point connections over long-haul lines (often leased from a phone company).
 - Speed \approx 1.544–45 Megabits/second.
 - Broadcast usually requires multiple messages.
 - Nodes:
 - usually a high percentage of mainframes.

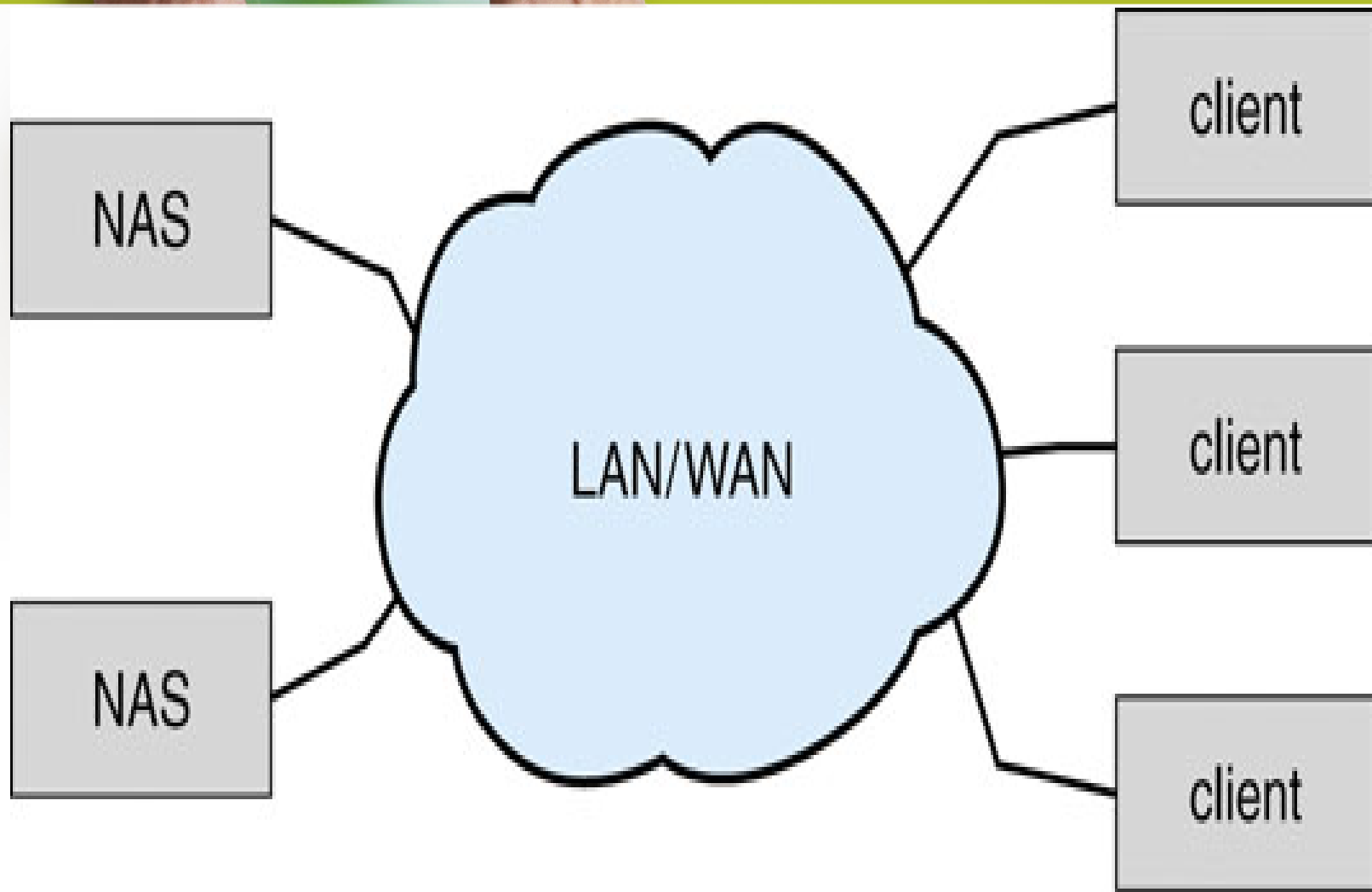


Example of Wide Area Network (WAN)

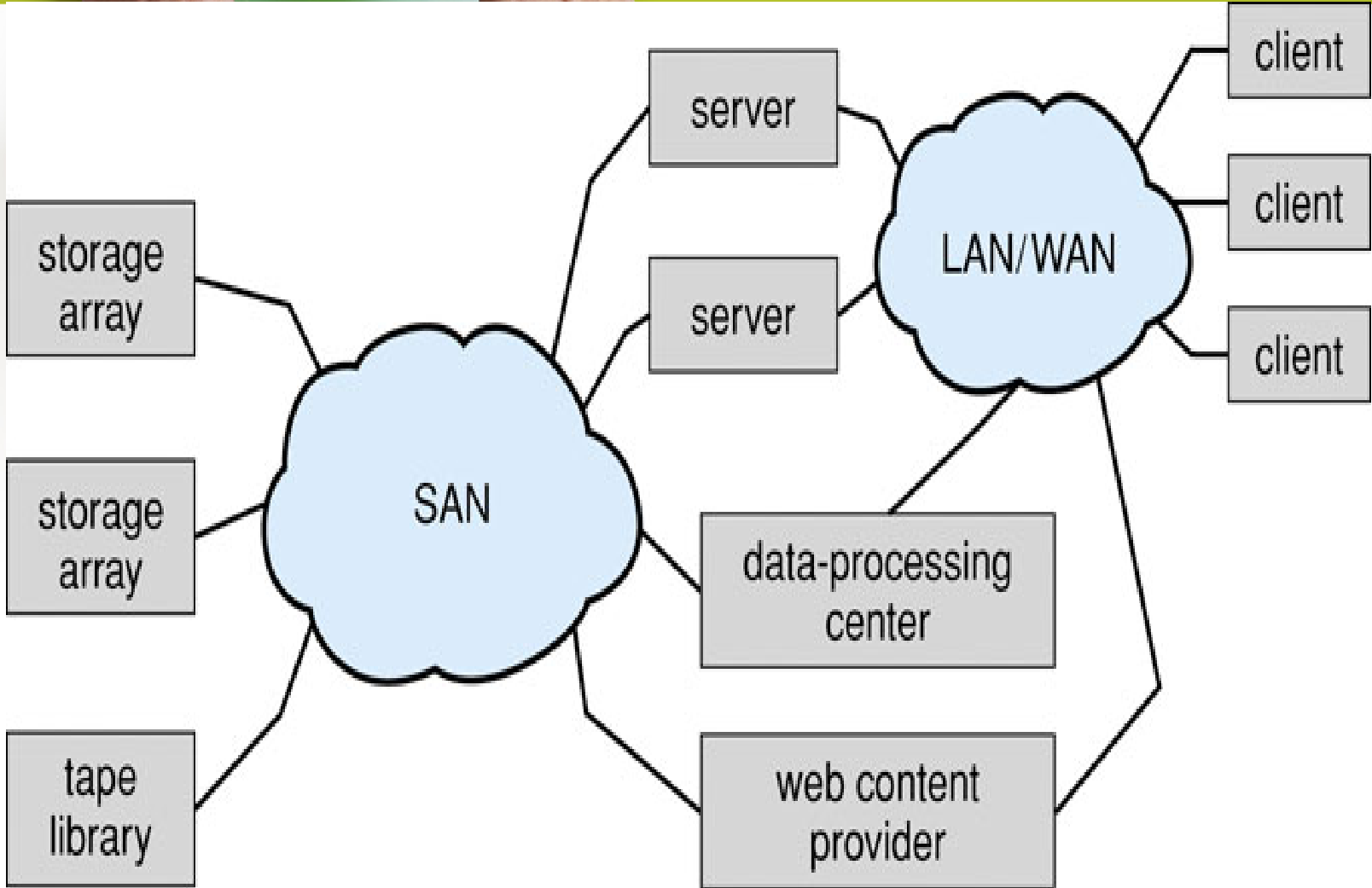




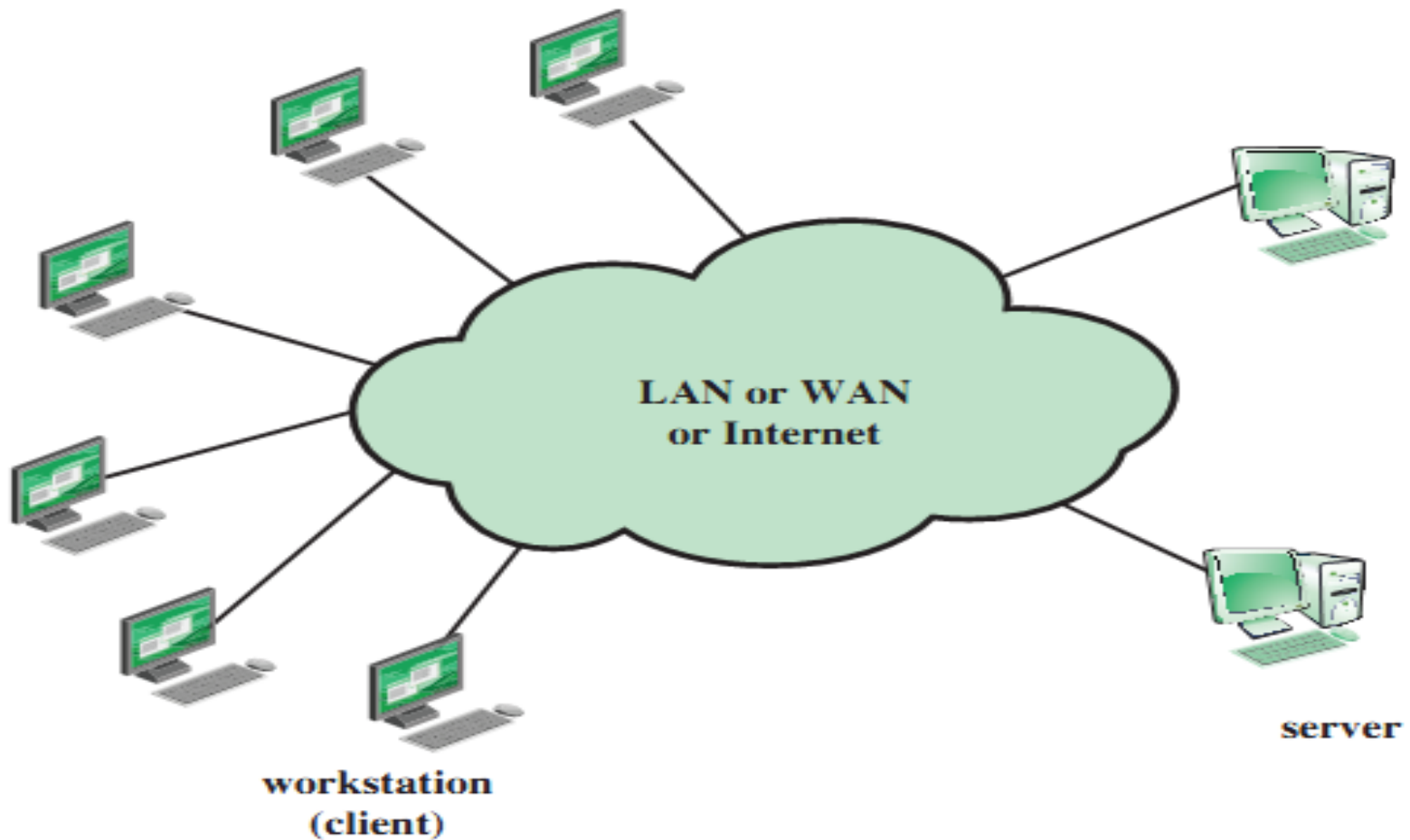
Network-attached Storage (NAS)



Storage-area Network (SAN)

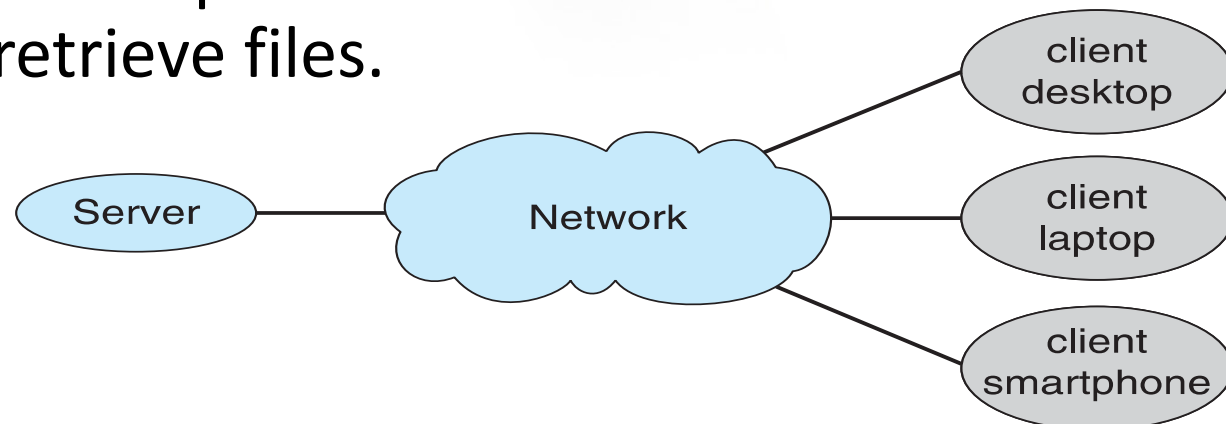


Generic Client/Server Environment



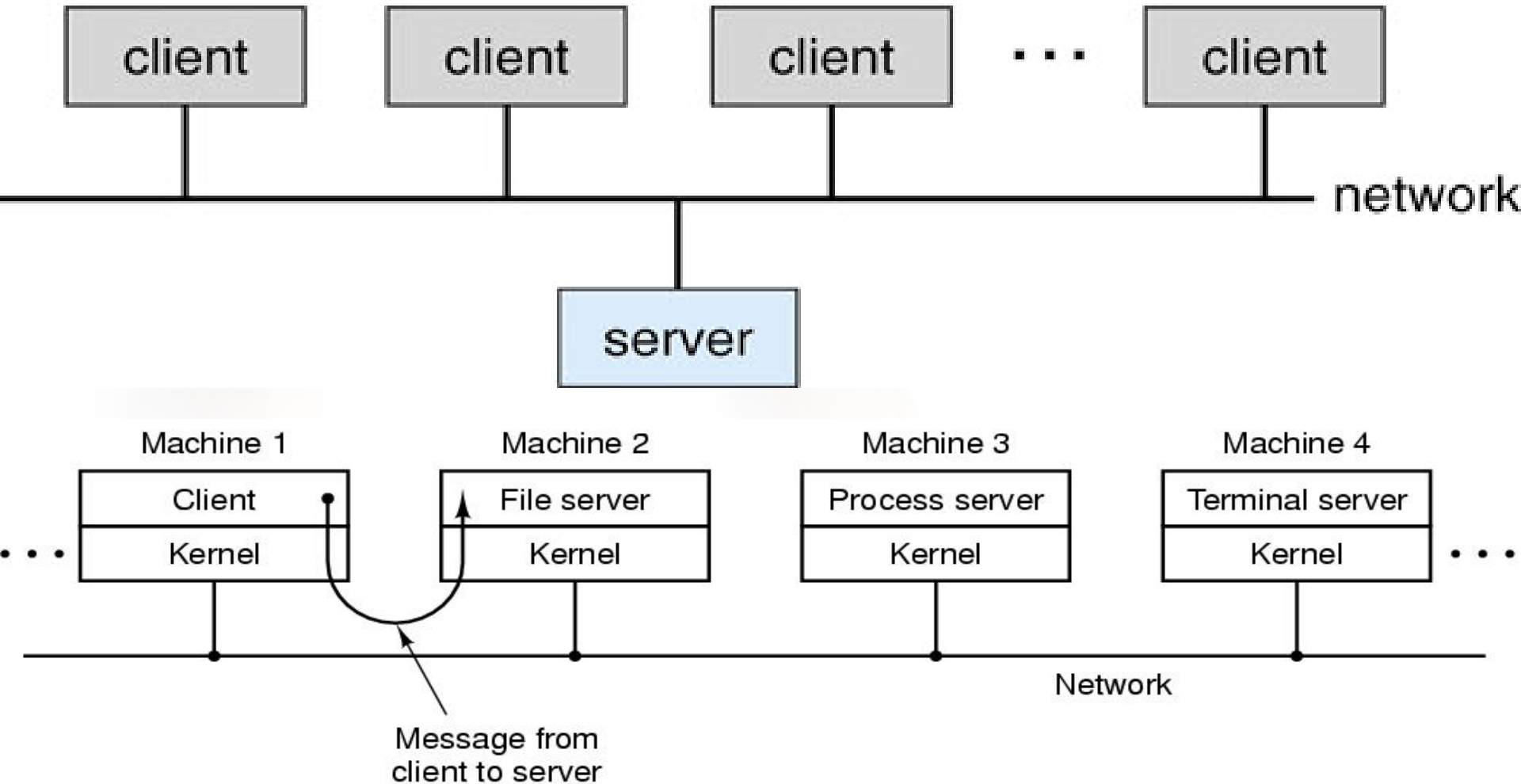
Client/Server Computing

- Dumb terminals supplanted by smart PCs.
- Many systems are now servers, responding to requests generated by clients:
 - Compute-server provides an interface to client to request services (i.e., database).
 - File-server provides interface for clients to store and retrieve files.





General Structure of a Client/Server System

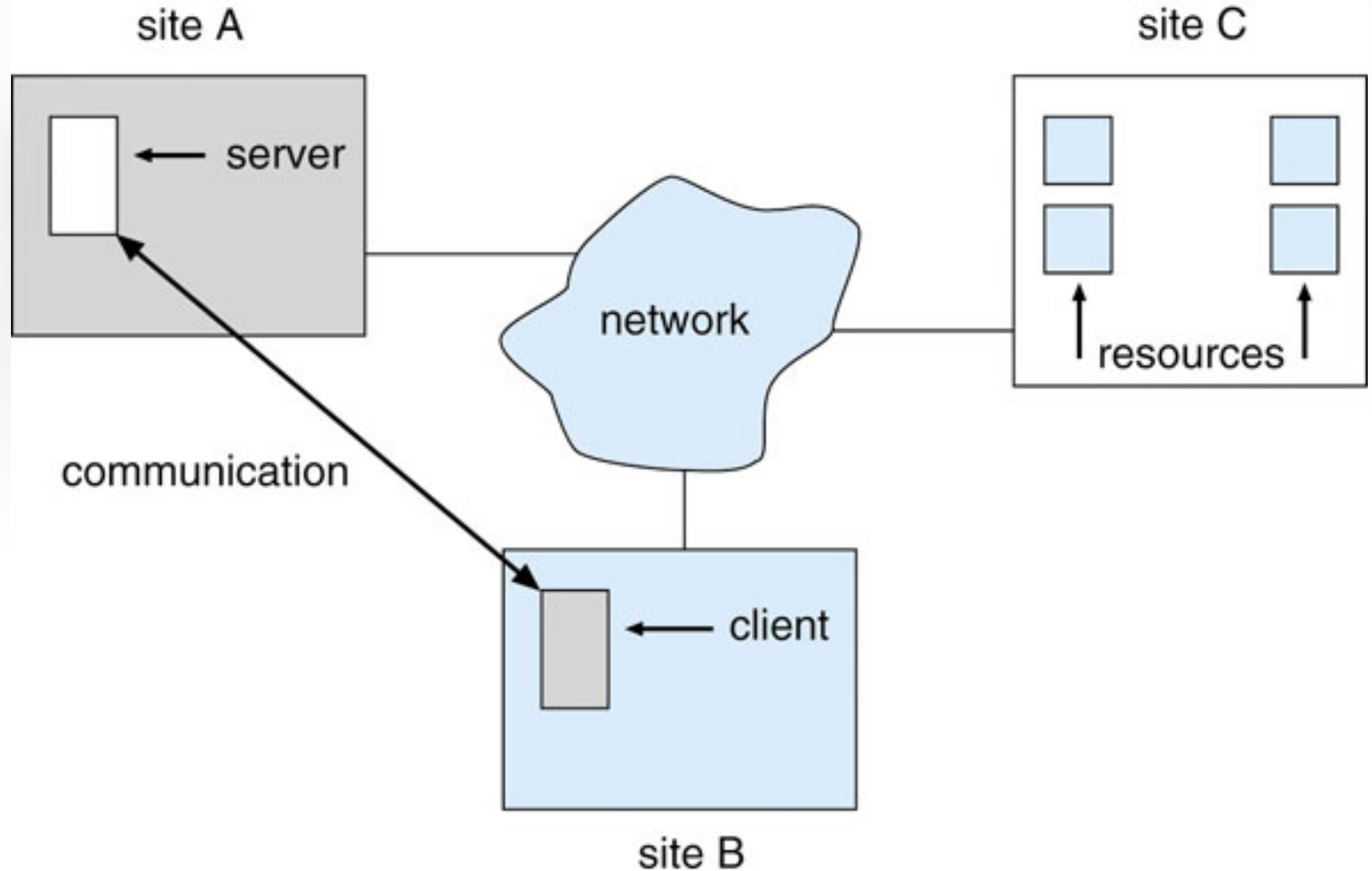




Distributed Systems

- Distributed system is collection of loosely coupled processors interconnected by a communications network.
- Processors variously called *nodes*, *computers*, *machines*, *hosts*.
- Reasons for distributed systems:
 - Resource sharing:
 - sharing and printing files at remote sites.
 - processing information in a distributed database.
 - using remote specialized hardware devices.
 - Computation speedup – load sharing.
 - Reliability – detect and recover from site failure, function transfer, reintegrate failed site.
 - Communication – message passing.

Distributed Systems

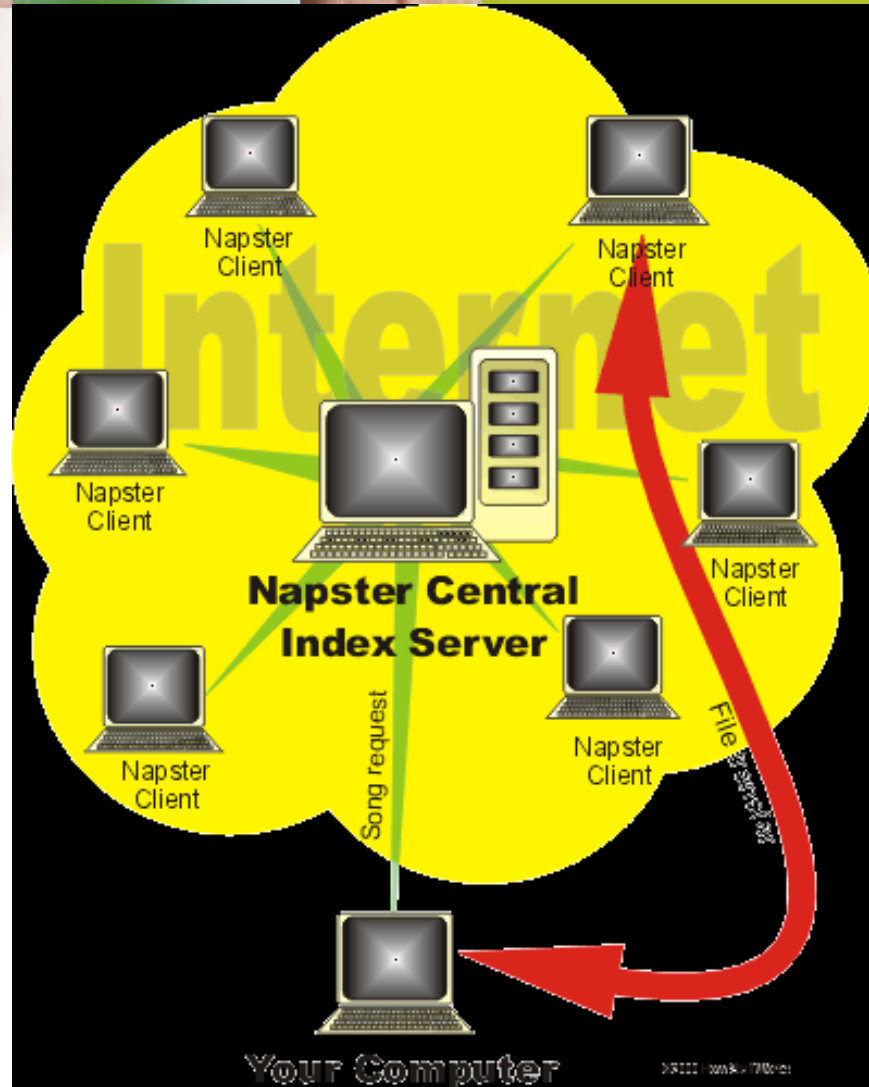





Peer-To-Peer (P2P)

- 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
 - Broadcasts request for service and responds to requests for service via discovery protocol.
- Examples include Napster and Gnutella, Voice over IP (VoIP) such as Skype.

Peer-to-Peer Computing



Your Computer

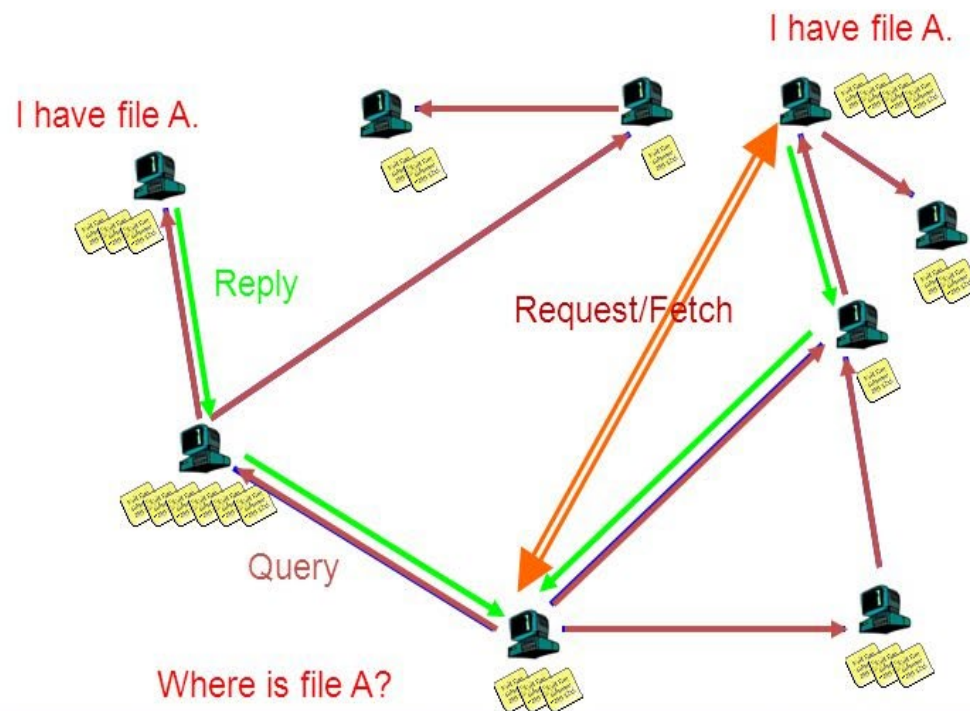


Peer-to-Peer Computing

- Another approach is to have a client first discover the node that provides the desired service by broadcasting a request for service to all other nodes in the network.
- The node (or nodes) providing that service responds to the peer making the request.
- To support this approach a *discover protocol* must be provided that allows peers to discover services offered by peers. This is totally distributed approach.
- E.g. Gnutella

Peer-to-Peer Computing

Gnutella: Search



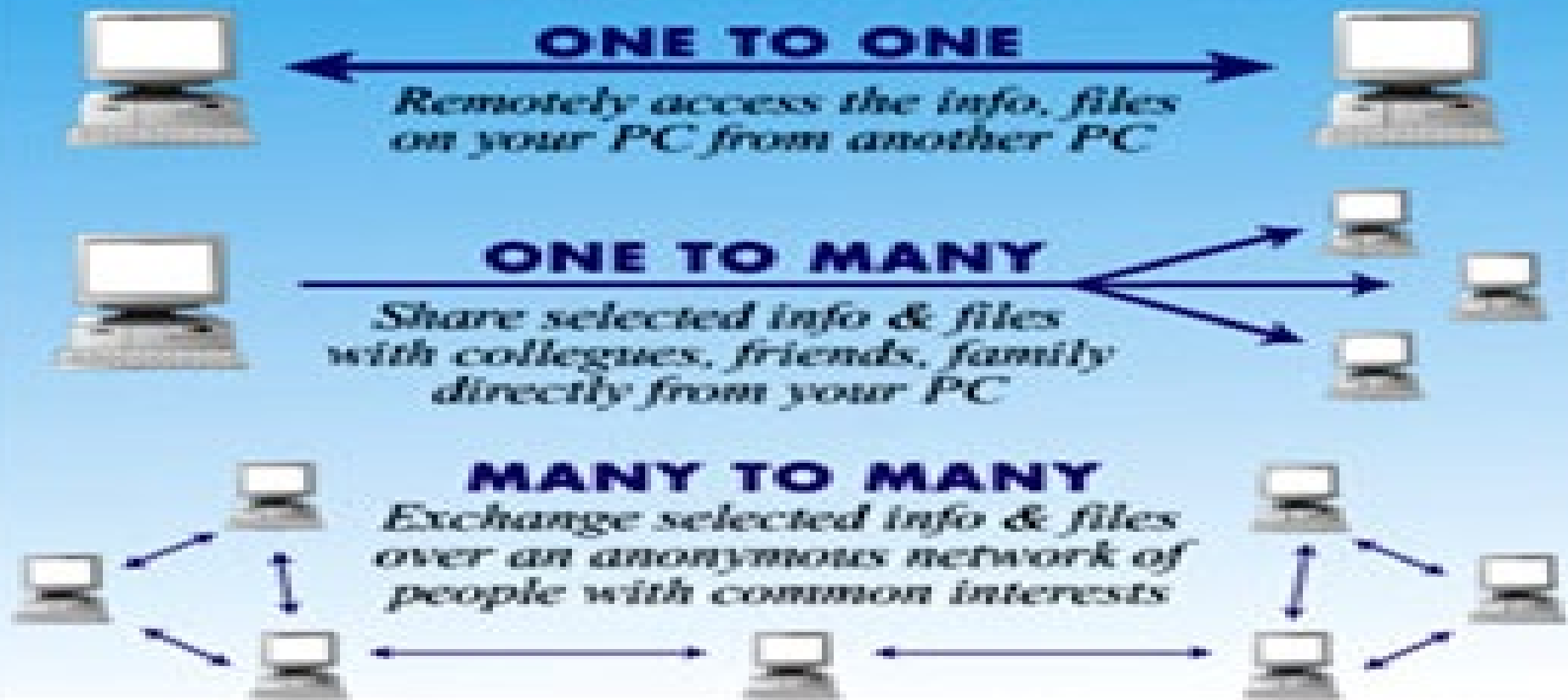
Q: Compare with Napster (publishing, searching, anything else)



Peer-To-Peer (P2P) Systems

DISTRIBUTED SERVICE

P2P sharing takes the central server out of the loop





(NOS) Networked Operating Systems

- Each computer runs independently from other computers on the network.
- Provides mainly file sharing.
- Users are aware of multiplicity of machines.
- Access to resources of various machines is done explicitly by:
 - Remote logging into the appropriate remote machine (telnet, ssh).
 - Remote Desktop (Microsoft Windows).
 - Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism.



Distributed Operating Systems (DOS not MS-DOS)

- Gives the impression there is a single operating system controlling the network.
- Users not aware of multiplicity of machines
 - Access to remote resources similar to access to local ones.
- Network is mostly transparent – it's a powerful virtual machine.
- Data Migration – transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task.
- Computation Migration – transfer the computation, rather than the data, across the system.
- Process Migration – execute an entire process, or parts of it, at different sites.



Web-based Systems

- Web has become ubiquitous.
- PCs most prevalent devices.
- More devices becoming networked to allow web access.
- New category of devices to manage Web traffic among similar servers:
Load Balancers.
- Basis for Grids/Cloud Computing.

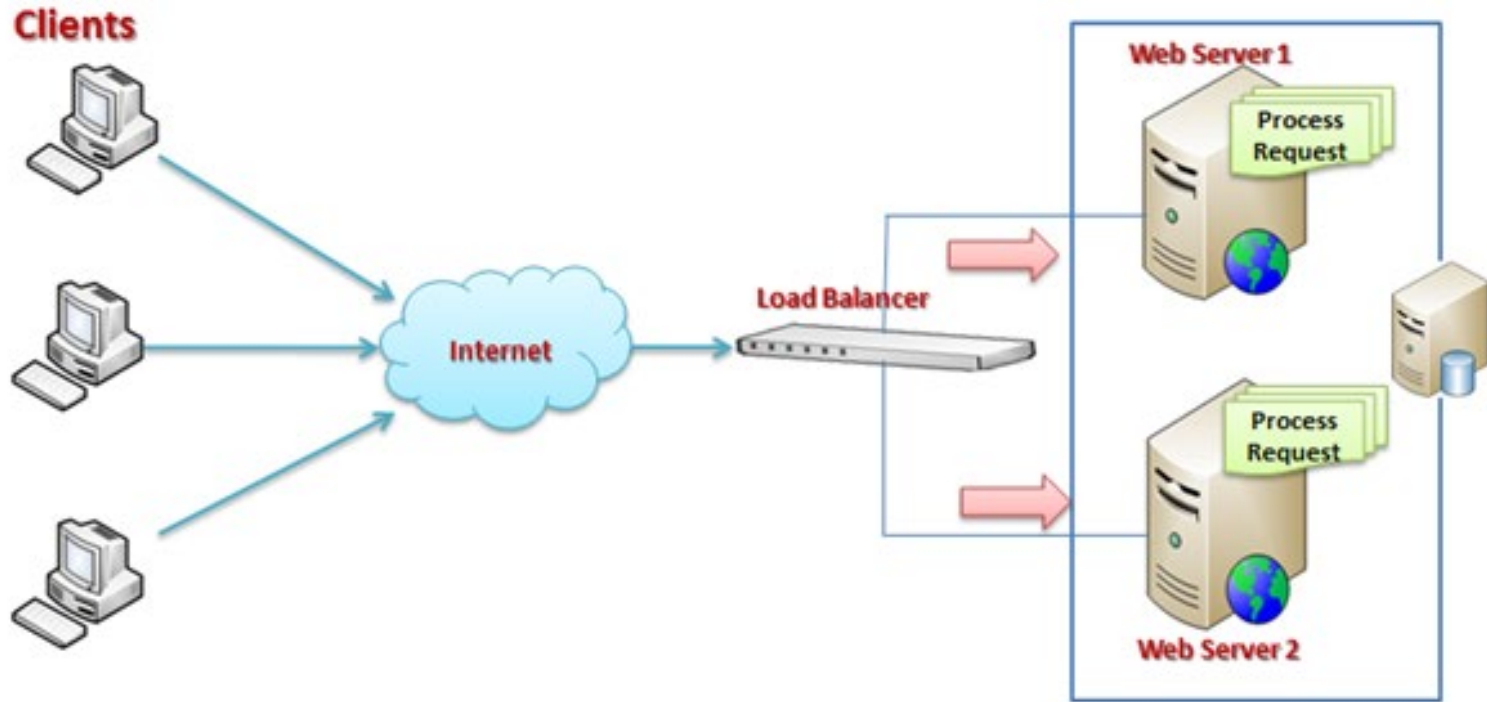
A close-up photograph of a person's hands holding a tablet. The screen shows a grid of social media icons: Facebook (f), Twitter (bird), and a circular icon with a person silhouette. Below these are icons for Pinterest (p), Google+ (G+), and a speech bubble icon. The background is a solid light green color.

Web-Based Computing

- PCs used to be more prevalent devices but now mobile devices (e.g. smart phones and tablets) are more prevalent modes of access
- Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows XP, which can be clients and servers



Web-Based Computing





Grid Computing Systems

- Collection of computer resources, usually owned by multiple parties and in multiple locations, connected together such that users can share access to their combined power:
 - Can easily span a wide-area network
 - Heterogeneous environment
 - Crosses administrative/geographic boundaries
 - Supports Virtual Organizations (VOs)
- Examples: EGEE - Enabling Grids for E-Science (Europe), Open Science Grid (USA).



Cloud Computing Systems (1)

- Collection of computer resources, usually owned by a single entity, connected together such that users can lease access to a share of their combined power:
 - Location independence: the user can access the desired service from anywhere in the world, using any device with any (supported) system.
 - Cost-effectiveness: the whole infrastructure is owned by the provider and requires no capital outlay by the user.
 - Reliability: enhanced by way of multiple redundant sites, though outages can occur, leaving users unable to remedy the situation.



Cloud Computing Systems (2)

- Scalability: user needs can be tailored to available resources as demand dictates – cost benefit is obvious.
- Security: low risk of data loss thanks to centralization, though problems with control over sensitive data need to be solved.
- Readily consumable: the user usually does not need to do much deployment or customization, as the provided services are easy to adopt and ready-to-use.
- Examples: Amazon EC2 (Elastic Compute Cloud), Google App Engine, IBM Enterprise Data Center, MS Windows Azure, SUN Cloud Computing.

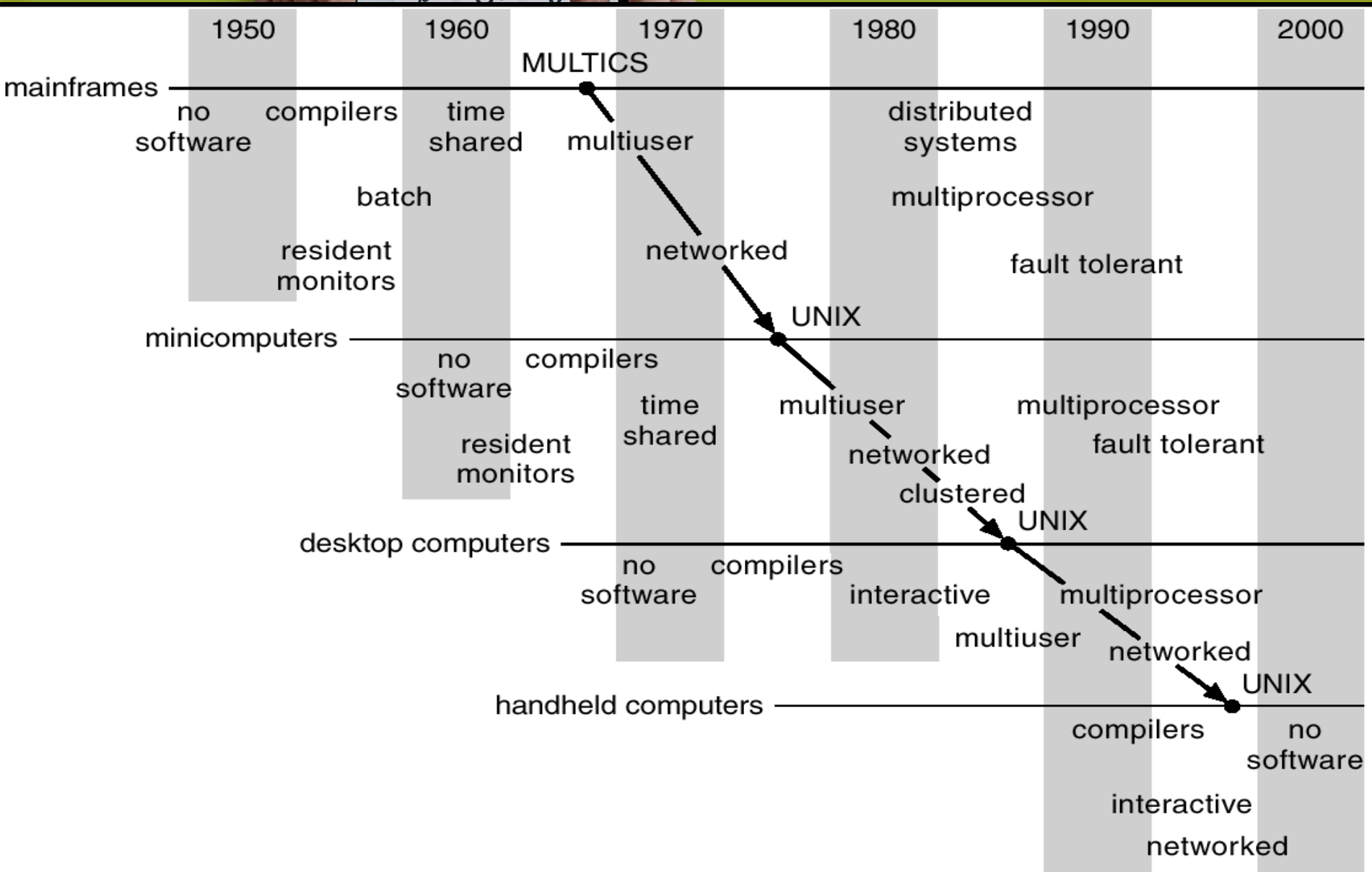



Handheld Systems

- Handheld systems are also dedicated:
 - Personal Digital Assistants (PDAs).
 - Cellular telephones.
- Issues:
 - Limited memory
 - Slow processors
 - Small display screens
 - Support for multimedia (images, video).

- Handheld smartphones, tablets, etc.
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope).
- Use IEEE 802.11 wireless, or cellular data networks for connectivity.
- Allows new types of apps like ***augmented reality***.
- Leaders are Apple iOS and Google Android.

Migration of OS Concepts and Features





Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has “copyleft” **GNU Public License (GPL)**
- Examples include **GNU/Linux, BSD UNIX** (including core of **Mac OS X**), and **Sun Solaris**