

Overview of Computer Workshop

Instructor

Dr. B Krishna Priya Dr. Piyush Joshi

Outline

- Basic Computer Organization
- Processors and its organization
- CPU
- Memory
- Storage Devices
- Interfaces
- Number System (Binary)
- Types of Memories
- Channel and Bus Architectures
- Standard buses
- Devices and Controllers
- Ports and Connectors
- Bootstrap Loaders
- Inside of a typical desktop/laptop
- Motherboard and Switch settings and Jumpers
- Servers

Generation of Computer

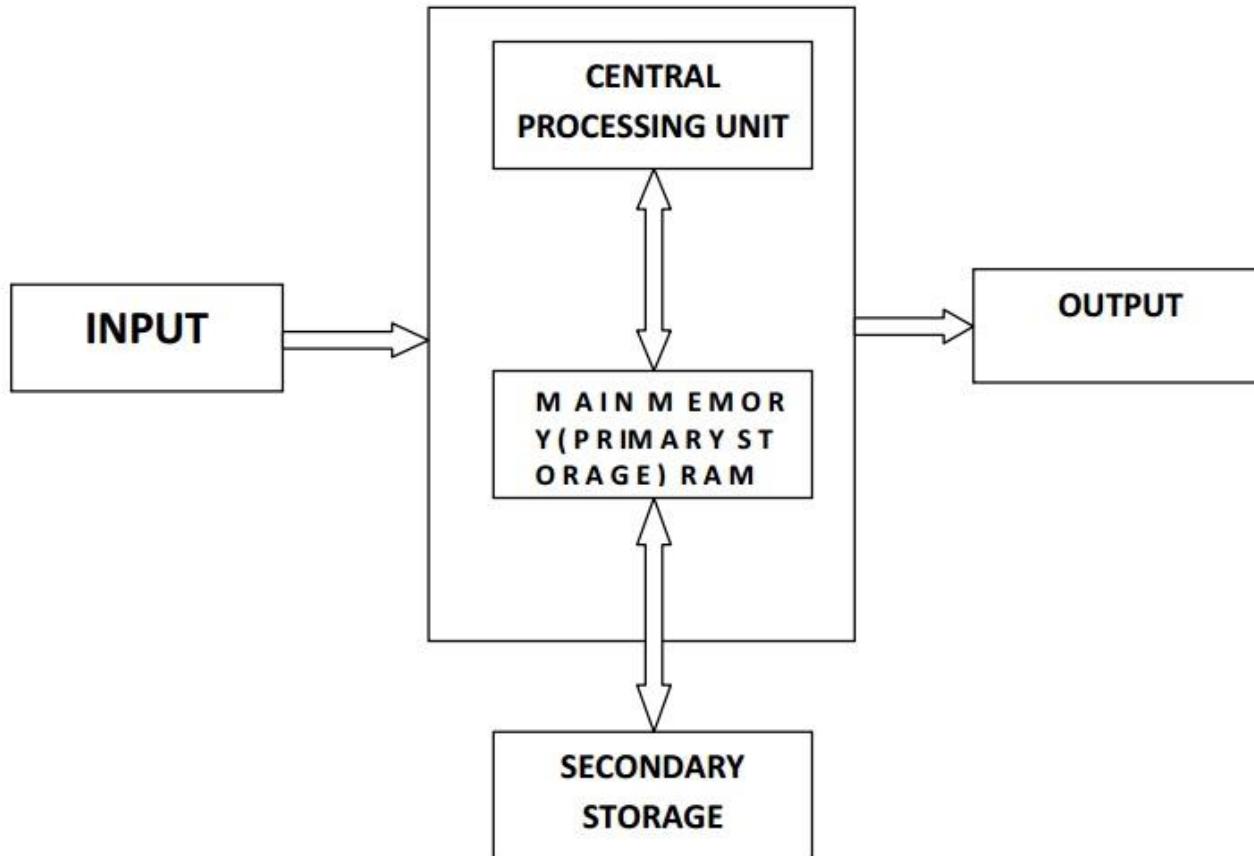
SN	Generation	Period	Main Component used	Merits/Demerits
1	First Generation	1942-1955	 Vacuum tubes	<ul style="list-style-type: none"> • Big in size • Consumed more power • Malfunction due to overheat • Machine Language was used
First Generation Computers - ENIAC , EDVAC , UNIVAC 1 ENIAC weighed about 27 tons, size 8 feet × 100 feet × 3 feet and consumed around 150 watts of power				
2	Second Generation	1955-1964	 Transistors	<ul style="list-style-type: none"> • Smaller compared to First Generation • Generated Less Heat • Consumed less power compared to first generation • Punched cards were used • First operating system was developed - Batch Processing and Multiprogramming Operating System • Machine language as well as Assembly language was used.
Second Generation Computers IBM 1401, IBM 1620, UNIVAC 1108				
3	Third Generation	1964-1975	 Integrated Circuits (IC)	<ul style="list-style-type: none"> • Computers were smaller, faster and more reliable • Consumed less power • High Level Languages were used
Third Generation Computers IBM 360 series, Honeywell 6000 series				
4	Fourth Generation	1975-1980	 Microprocessor Very Large Scale Integrated Circuits (VLSI)	<ul style="list-style-type: none"> • Smaller and Faster • Microcomputer series such as IBM and APPLE were developed • Portable Computers were introduced.

Basic of Computer organization

- A standard fully featured system configuration has basically four types of featured devices
 - Input devices
 - Output devices
 - Memory
 - Storage devices



Basic of Computer organization



CPU

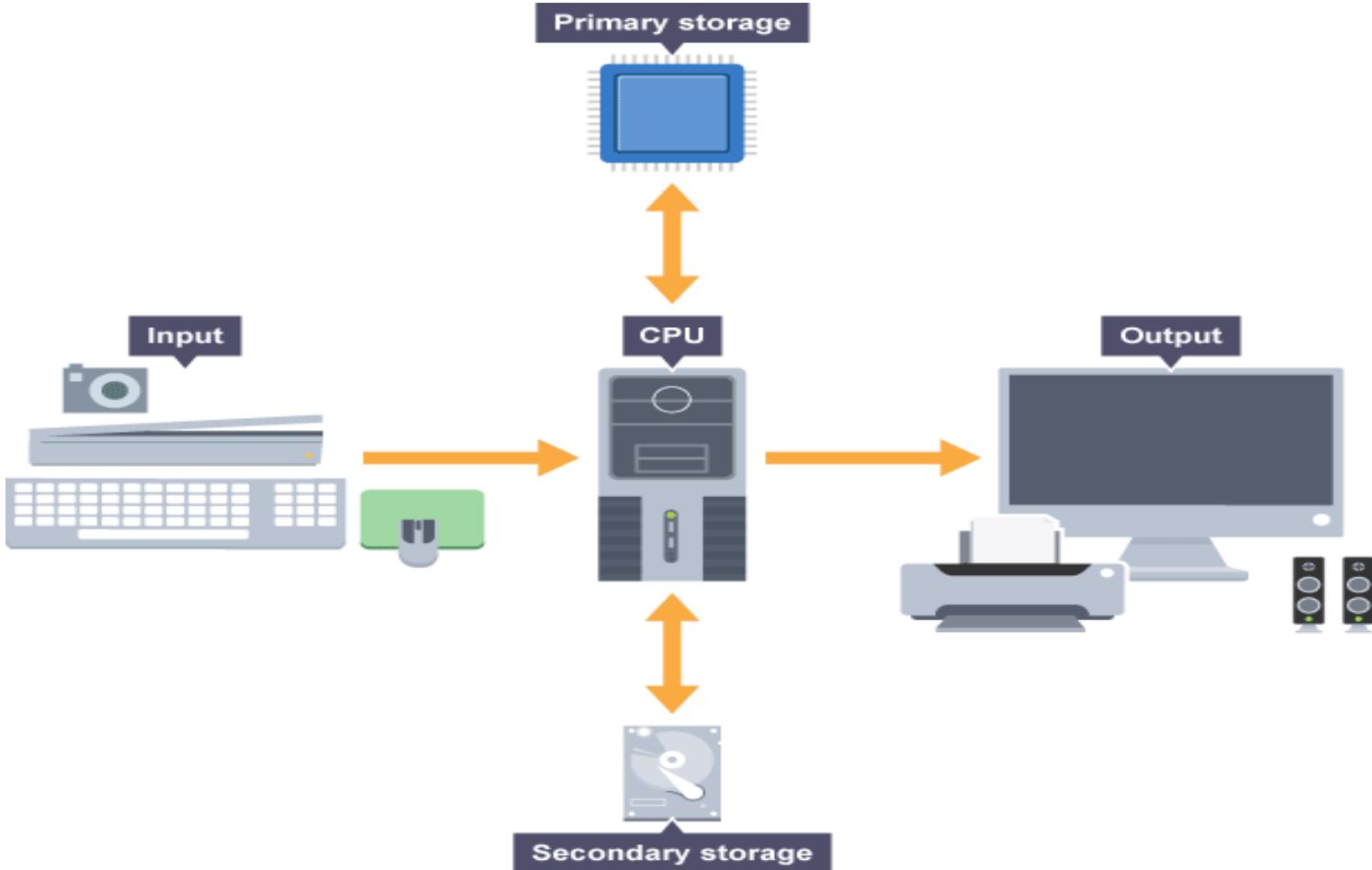
- The **central processing unit (CPU)** is the most important **hardware** component in a computer. It has two main functions:
 - to process **data** and **instructions**
 - to control the rest of the computer system
- All programs and data processing are run in the CPU and all hardware components are, to some extent, controlled by it.
- All **general purpose computers** follow the same basic model.



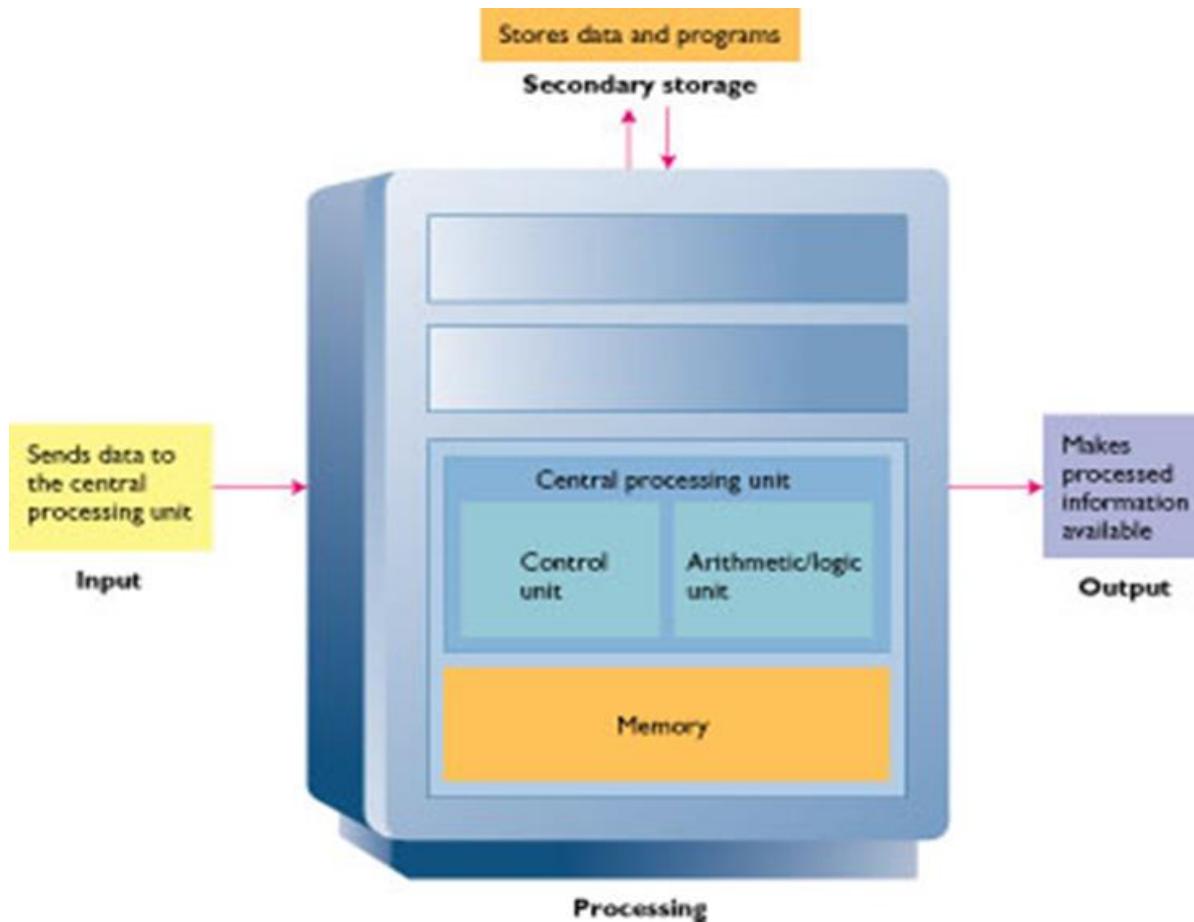
Common CPU components

- The main components of **central processing unit (CPU)** :
- **control unit (CU)**
- **arithmetic logic unit (ALU)**
- **registers**
- **cache**
- **buses**
- All the components work together to allow processing and system control.

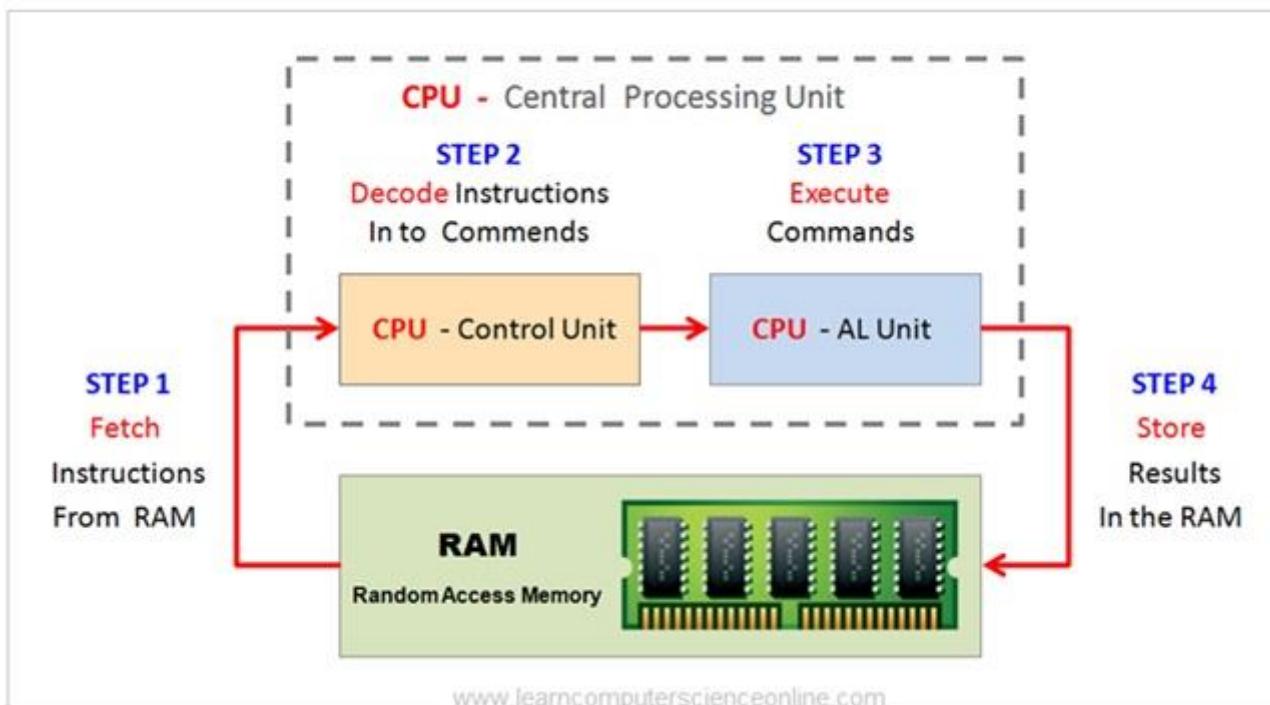
CPU



Common CPU components



CPU Process Instruction



CPU

- The CPU receives instructions and data from an **input** or **memory**. The instructions and data are processed by the CPU and the results are either sent to an **output** or transferred to **secondary storage**. Data is held in **primary storage** while it is being processed.
- Input is received from an **input device** such as a keyboard, mouse, camera or scanner. Output is sent to an **output device** such as a monitor, printer or speaker.

Common CPU components

Control unit (CU)

- Part of the hardware that is in-charge
- Directs the computer system to execute stored program instructions
- Communicates with other parts of the hardware
- it transfers **data** and instructions around the system

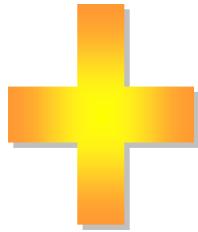
Common CPU components

Arithmetic logic unit (ALU)

- It performs arithmetic and logical operations (decisions).
- it acts as a gateway between **primary storage** and **secondary storage** - data transferred between them passes through the ALU.

Common CPU Components

Arithmetic Operations



Addition



Subtraction

Multiplication

Division



Common CPU Components

Logical Operations

- Evaluates conditions
- Makes comparisons
- Can compare
 - Numbers
 - Letters
 - Special characters

NOT
AND
OR



Common CPU components

Buses

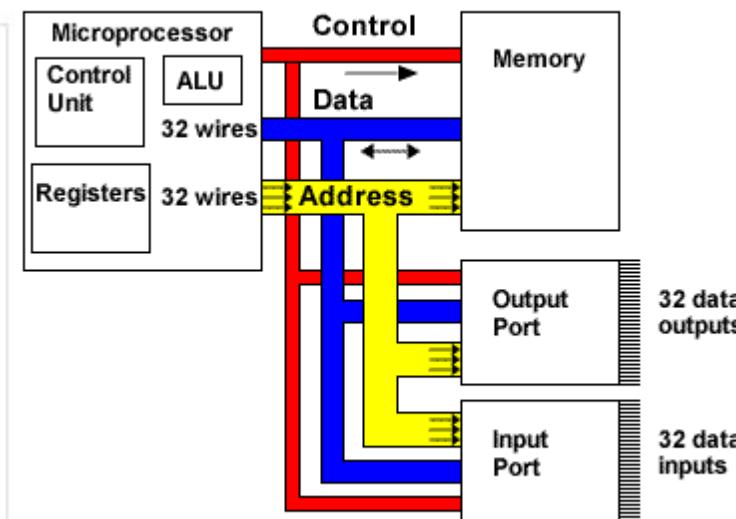
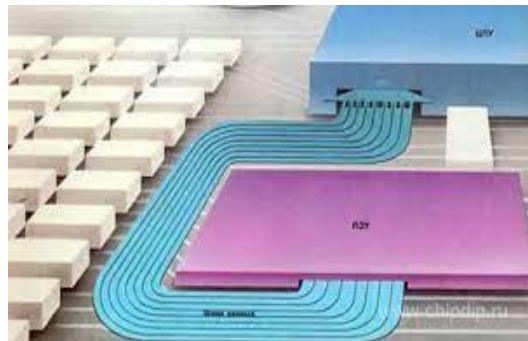
- A bus is a high-speed internal connection.
- Buses are used to send control signals and data between the processor and other components.

Three types of bus are used.

- Address bus - carries memory addresses from the processor to other components such as primary storage and input/output devices. The address bus is unidirectional.
- Data bus - carries the data between the processor and other components. The data bus is bidirectional.
- Control bus - carries control signals from the processor to other components. The control bus also carries the clock's pulses. The control bus is unidirectional.

Buses

Buses - Motherboard Buses



Common CPU components

Clock

- The CPU contains a clock which, along with the CU, is used to coordinate all of the computer's components.
- This is timing signal that synchronize all data transfers in a computer system.
- The CPU also executes instructions at a speed set by clock signal.
- Clock rate are measured in hertz (cycles per second)
- Hertz is measured interms unit of frequency.
- Frequency is the number of occurrences of a repeating element per unit time.
- Example: if a new born baby heart beat is 120 times a minute then it is i.e in 30 second the heart beats around 60 times and 1 min it is 120 so it is double of the time.
- In the 1980s, processors commonly ran at a rate of between 3 megahertz (MHz) and 5 MHz, which is 3 million to 5 million pulses or cycles per second.
- Today, processors commonly run at a rate of between 3 gigahertz (GHz) and 5 GHz, which is 3 billion to 5 billion pulses or cycles per second.

Input/output devices

Input devices

- An **input device** allows data such as text, images, video or sound to be entered into a computer system.
- Keyboard - The most common input device used for entering text. Includes keys for word processing and data inputting, and for performing specific functions.
- Mouse - The most popular pointing device used to control the cursor on the screen and activated when moved across a flat surface. There are usually two buttons that activate different features on the screen.
- Microphone - Used to convert analog sound waves into a digital form that can be saved and manipulated by the computer. A microphone can also be used to input voice commands but they do not work well in noisy environments.
- Scanner - Works in a similar way to a photocopier. Captures images from the source which are then converted into a digital form that can be stored.
- Assistive technology - Devices that are used by some people with disabilities to access computers. For example, a sip/puff tube can be used by someone with a disability such as paralysis to control a computer system.

Input/output devices

Output devices

- There are many outputs created by a computer system. These include printed documents, on-screen data and sound.
- An **output device** allows data to be transmitted by the computer in a human-friendly form, for example, sound being played through a speaker.
- Monitor
- Printer
- Speaker
- Projector

Overview of Computer Workshop

Instructor

Dr. B Krishna Priya Dr. Piyush Joshi

Outline

- Basic Computer Organization
- Processors and its organization
- CPU
- Memory
- Storage Devices
- Interfaces
- Number System (Binary)
- Types of Memories
- Channel and Bus Architectures
- Standard buses
- Devices and Controllers
- Ports and Connectors
- Bootstrap Loaders
- Inside of a typical desktop/laptop
- Motherboard and Switch settings and Jumpers
- Servers

MEMORY

Purpose of Storage :

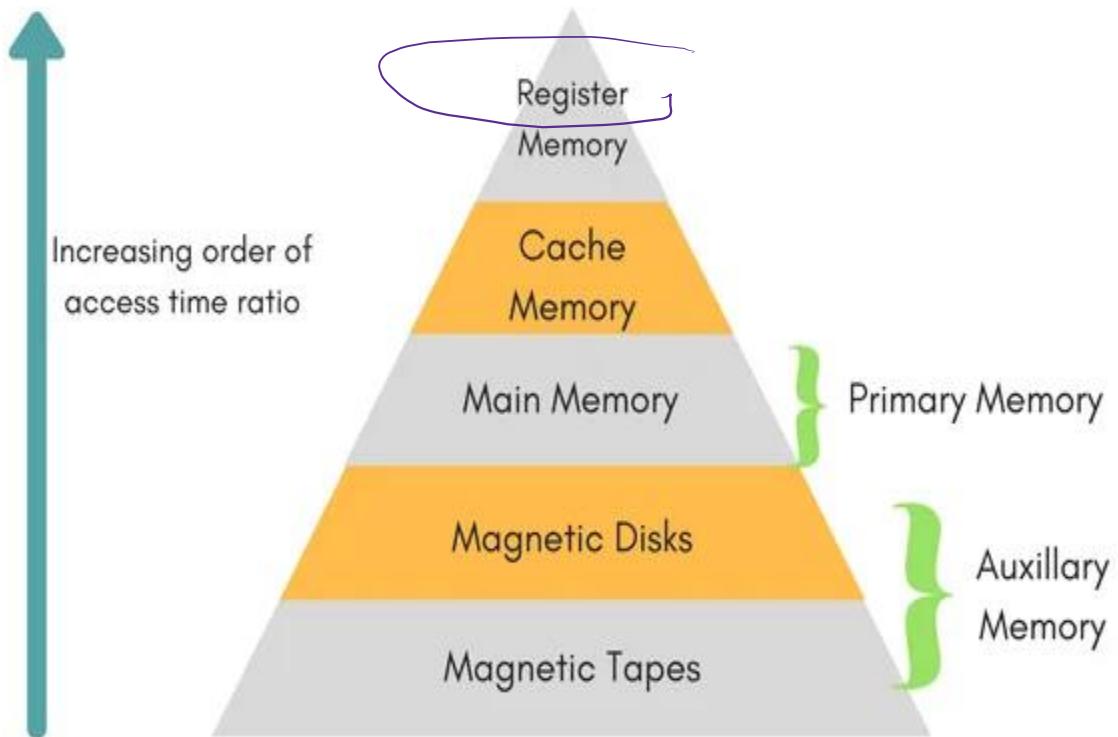
- The fundamental components of a general-purpose computer are arithmetic and logic unit, control circuitry, storage space, and input/output devices.
- If storage was removed, the device we had would be a simple calculator instead of a computer.
- The ability to store instructions that form a computer program, and the information that the instructions manipulate is what makes stored program architecture computers versatile.

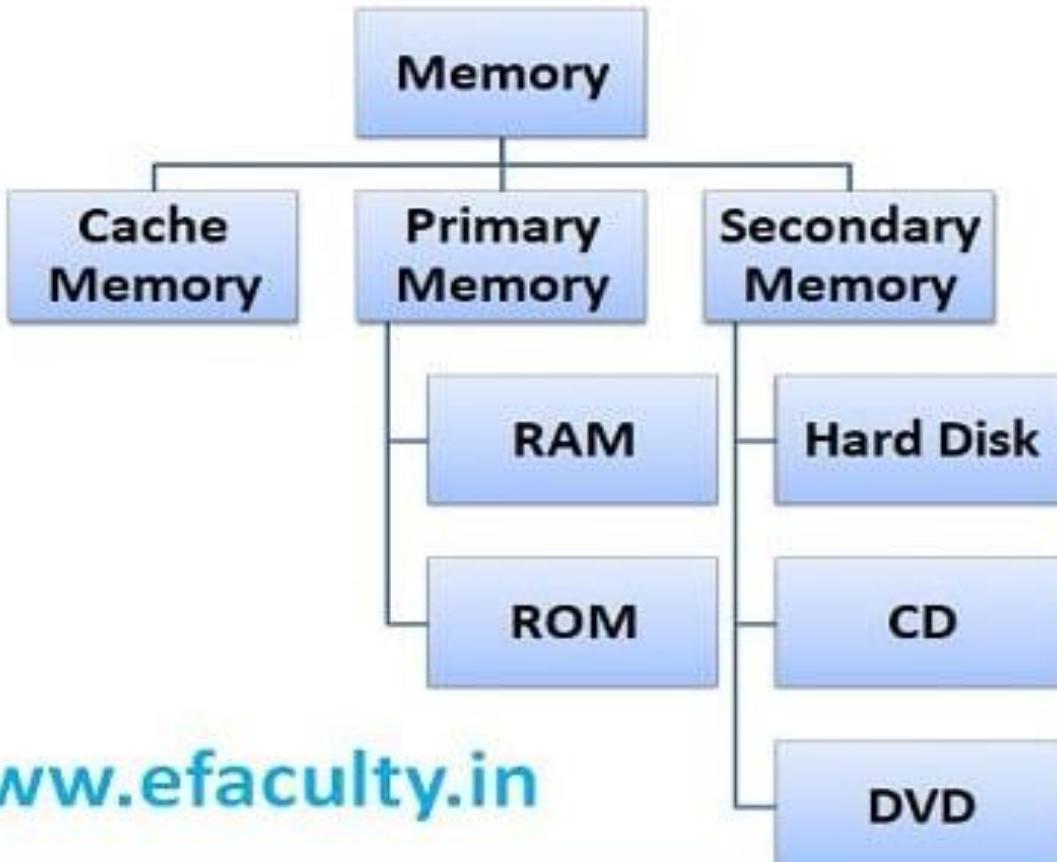
MEMORY

- A memory unit is the collection of storage units or devices together.
- The memory unit stores the information in the form of bits.
- Bit is either 0 or 1.
- Generally, memory/storage is classified into 2 categories:
- Volatile Memory: This loses its data, when power is switched off.
- Non-Volatile Memory: This is a permanent storage and does not lose any data when power is switched off.



Memory Hierarchy



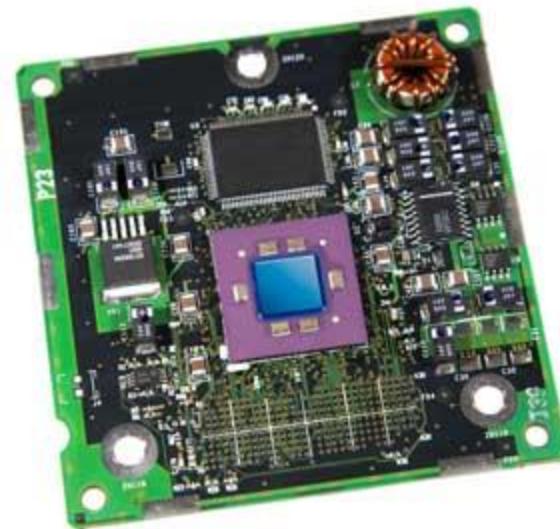


Registers

- Primary storage is directly connected to the central processing unit of the computer.
- It must be present for the CPU to function correctly.
- Processors Register:
 - It is the internal to the central processing unit.
 - Registers contain information that the arithmetic and logic unit needs to carry out the current instruction.
 - They are technically the fastest of all forms of computer storage.

Cache memory-SRAM

- It is a special type of internal memory used by many central processing units to increase their performance or "throughput".
- Some of the information in the main memory is duplicated in the cache memory, which is slightly slower but of much greater capacity than the processor registers, and faster but much smaller than main memory.



Cache memory-SRAM

Advantages

- Cache memory is faster than main memory.
- It consumes less access time as compared to main memory.
- It stores the program that can be executed within a short period of time.
- It stores data for temporary use.

Disadvantages

- The disadvantages of cache memory are as follows –
- Cache memory has limited capacity.
- It is very expensive.

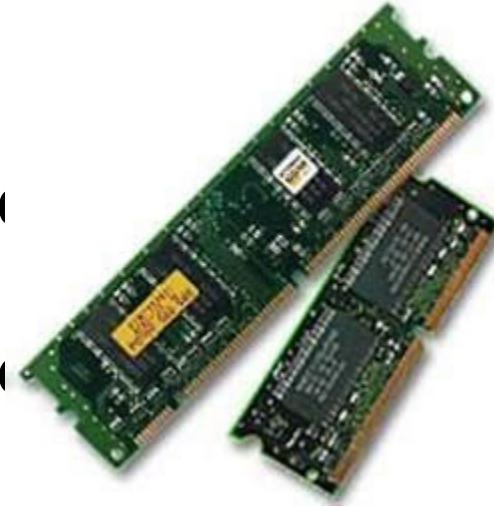
Main Memory

- It contains the programs that are currently being run in the CPU.
- The arithmetic and logic unit can very quickly transfer information between a processor register and locations in main storage, also known as a "memory addresses".
- In modern computers, electronic solid-state random access memory is used for main storage, and is directly connected to the CPU via a "memory bus" and a "data bus".
- Main or Primary memory uses two technologies
 - RAM(Volatile)
 - ROM(Non-Volatile)

Main Memory-DRAM

Characteristics of Main Memory

- These are semiconductor memories.
- Usually volatile memory.
- Data is lost in case power is switched off.
- It is the working memory of the computer.
- Faster than secondary memories.
- A computer cannot run without the primary memory.



Main Memory-**ROM**

- Stores crucial information essential to operate the system, like the program essential to boot the computer.
- It is not volatile.
- Always retains its data.
- Used in embedded systems or where the programming needs no change.
- Used in calculators and peripheral devices.
- ROM is further classified into 4 types- *MROM*, *PROM*, *EPROM*, and *EEPROM*.

- 1.PROM (Programmable read-only memory)** – It can be programmed by the user. Once programmed, the data and instructions in it cannot be changed.
- 2.EPROM (Erasable Programmable read only memory)** – It can be reprogrammed. To erase data from it, expose it to ultraviolet light. To reprogram it, erase all the previous data.
- 3.EEPROM (Electrically erasable programmable read only memory)** – The data can be erased by applying an electric field, with no need for ultraviolet light. We can erase only portions of the chip.
- 4.MROM(Marked ROM)** – The very first ROMs were hard-wired devices that contained a pre-programmed set of data or instructions. These kind of ROMs are known as masked ROMs, which are inexpensive.

Secondary or Auxiliary Memory

- This type of memory is also known as external memory or non-volatile.
- It is slower than the main memory. These are used for storing data/information permanently.
- CPU directly does not access these memories, instead they are accessed via input-output routines.
- The contents of secondary memories are first transferred to the main memory, and then the CPU can access it. For example, disk, CD-ROM, DVD, etc.



Secondary or Auxiliary Memory

Characteristics of Secondary Memory

- These are magnetic and optical memories.
- It is known as the backup memory.
- Data is permanently stored even if power is switched off.
- Computer may run without the secondary memory.
- Slower than primary memories.

Storage Devices

- The purpose of storage devices in a computer is to hold data or information and get that data to the CPU as quickly as possible when it is needed.
- Computers use disks for storage: hard disks that are located inside the computer, and floppy or compact disks that are used externally.
- Computers method of storing data & information for long term basis i.e. even after PC is switched off.

Storage Devices

- It is non - volatile
- Can be easily removed and moved & attached to some other device
- Memory capacity can be extended to a greater extent
- Cheaper than primary memory
- Storage Involves two processes
 - a) Writing data
 - b) Reading data

Floppy Disks

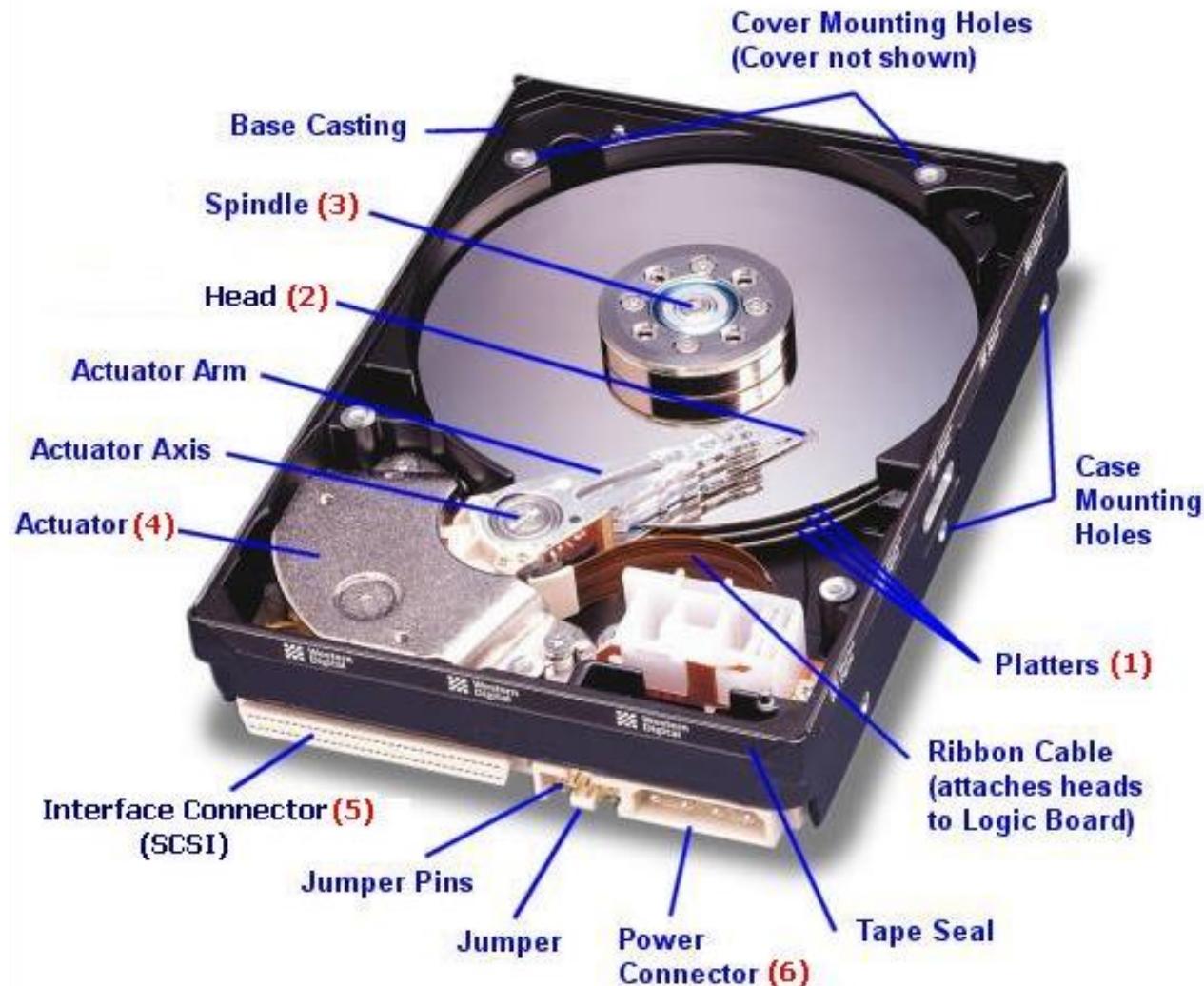
- Term “floppy disk” first used to refer to the 8” medium used with mini-computers and mainframes
- Then was used to refer to the PC floppy diskette which used a 5.25” platter – also called the minifloppy diskette.
- Now refers to the 3 1/2” floppy diskette – aka microfloppy diskette. This is the version that is used in computers today if any floppy technology is used at all.



Hard Disks

- Invented by Reynold Johnson in 1956
- Used in the IBM 305 RAMAC accounting computer
- Actuator saves data to the platters magnetically
- Most HDDs are connected to the motherboard using a 1 meter SATA-device cable
- Hard disk, also called hard disk drive or hard drive, magnetic storage medium for a computer.
- Hard disks are flat circular plates made of aluminum or glass and coated with a magnetic material.
- Hard disks for Personal Computers can store terabytes (trillions of bytes) of information.

Hard Disk Drive (HDD)



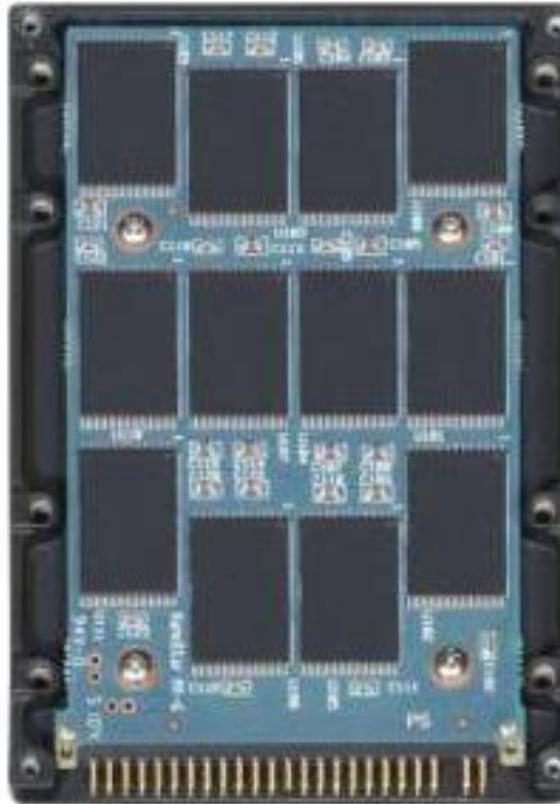
Optical storage

Optical Storage Devices Data is stored on a reflective surface so it can be read by a beam of laser light. Two Kinds of Optical Storage Devices

- Compact Disk Read-Only Memory (CD-ROM)
- CD-Recordable (CD-R)/CD-Rewritable (CD-RW)
- Digital Video Disk Read-Only Memory (DVD-ROM)
- DVD Recordable (DVD-R/DVD Rewritable (DVD-RW)
- Photo CD

Solid State Drive (SSD)

- SSD technology has been around since 1978
- Flash based SSD introduced by M-Systems in 1995
- Uses non-volatile flash memory to store data
- Volatile SSDs that use DRAM are also available



Optical Drives

- Enables a computer to read different types of media discs
- Some drive can only read discs, but recent drives are both reader and recorders

CD-Media

- CD-ROM
- CD-R was introduced in the mid-1990s
- Early CD-R drives require that the entire disc be burned in one session, which are called single-session drives
- Modern CD-R allow users to go back and burn additional data until it is full, which are called multisession drives
- Once the data is burned onto the CD-R disc, the data cannot be erased or altered
- have two speeds that matter: record speed & read speed
 - Ex: 8x24x
- Two types of CD-R disc:
 - 74-minute disc that holds approximately 650 MB
 - 80-minute disc that holds approximately 700 MB

CD-Media Continued

- CD-RW drives not only allows users to burn data onto a disc, but to burn over existing data
 - Three values:
 - first shows the CD-R write speed
 - second shows the CD-RW rewrite speed
 - third shows the read speed
 - example: 8x4x32x



DVD-Media

- Developed by a large consortium of electronics and entertainment firms during the early 1990s and released in 1995
- Lowest capacity- 4.37 GB of data or two hours of video
- Highest capacity-16 GB of data or more than eight hours of video
- Uses smaller pits than CD-Media and packs them more densely
- Comes in single-sided (SS) and double-sided (DS) format
- Comes in single-layer (SL) and dual-layer (DL) format

DVD Media Continued

- DVD-ROM
- Standard recordable DVD-Media
 - DVD-R
 - DVD-RW
 - DVD+R
 - DVD+RW



Hot-Swappable Devices

- A hot swappable device is one which can be attached or detached from a computer or other electronic device without having to reboot the computer.
- The most common hot swappable devices are universal serial bus (USB) devices.
- Many hot swappable USB devices have all the necessary software built in to the device, so simply plugging it in to the computer allows the computer to detect the USB device and start working with it.
- Firewire is another common interface used with hot swappable devices.

What's USB used for?

- Personal Data Transport
- Secure storage of data, application and software files
- System administration
- Application carriers
- Booting operating systems
- Brand and product promotion
- Backup
- Flash drives also store data densely compared to many removable media

Various USB Designs

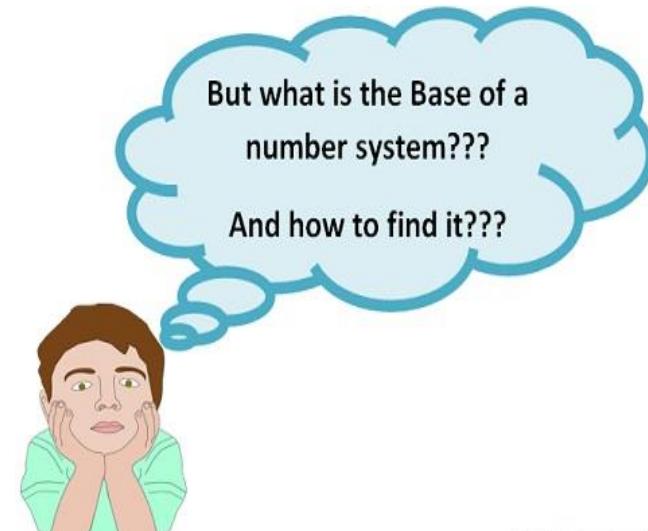


Number System

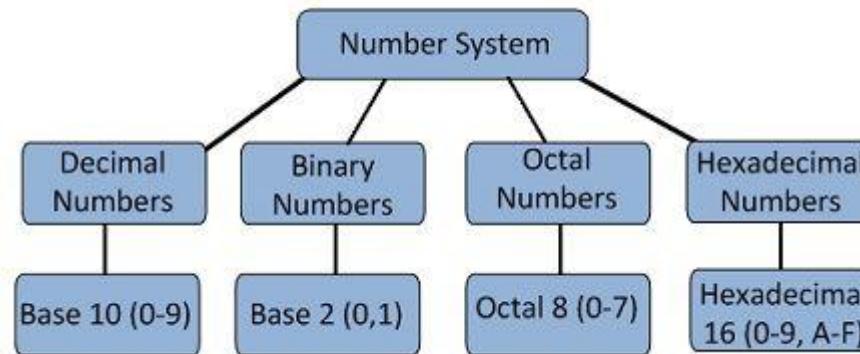
Dr. B Krishna Priya

Number System

- A digital system can understand positional number system only where there are a few symbols called digits and these symbols represent different values depending on the position they occupy in the number.
- A value of each digit in a number can be determined using
 - The digit
 - The position of the digit in the number
 - The base of the number system (where base is defined as the total number of digits available in the number system).



Number Systems



DECIMAL NUMBERS

- Decimal number systems each of the ten digits, 0 through 9, represents a certain quantity.
- The position of each digit in a decimal number indicates the magnitude of the quantity represented and can be assigned a weight.
- The weights for whole numbers are positive powers of ten that increases from right to left, beginning with $10^0 = 1$ that is $10^3 \ 10^2 \ 10^1 \ 10^0$

- For fractional numbers, the weights are negative powers of ten that decrease from left to right beginning with 10^{-1} that is $10^2 \ 10^1 \ 10^0 \cdot 10^{-1} \ 10^{-2} \ 10^{-3}$.
- The value of a decimal number is the sum of digits after each digit has been multiplied by its weights as in following examples

- Express the decimal number **87** as a sum of the values of each digit.
 - The digit 8 has a weight of 10 which is 10 as indicated by its position. The digit 7 has a weight of 1 which is 10^0 as indicated by its position.
 - $87 = (8 \times 10^1) + (7 \times 10^0)$
- Express the decimal number **725.45** as a sum of the values of each digit.
 - $725.45 = (7 \times 10^2) + (2 \times 10^1) + (5 \times 10^0) + (4 \times 10^{-1}) + (5 \times 10^{-2}) = 700 + 20 + 5 + 0.4 + 0.05$

BINARY NUMBERS

- The binary system is less complicated than the decimal system because it has only two digits, it is a base two system.
- The two binary digits (bits) are 1 and 0.
- The position of a 1 or 0 in a binary number indicates its weight, or value within the number, just as the position of a decimal digit determines the value of that digit.
- The weights in a binary number are based on power of two as:
- 2^4 2^3 2^2 2^1 2^0 . 2^{-1} 2^{-2}

- With 4 digits position we can count from zero to 15.
- In general, with n bits we can count up to a number equal to $K - 1$. Largest decimal number = $K - 1$.

For example: what is the largest decimal number that can be represented by 4 bits?

- $M=2^{\text{pow}N}-1=16-1=15$

With 4 bits, the maximum possible number is binary 1111 or decimal 15.

A binary number is a weighted number. The right-most bit is the least significant bit (LSB) in a binary whole number and has a weight of $2^0 = 1$.

- The weights increase from right to left by a power of two for each bit.
- The left-most bit is the most significant bit (MSB); its weight depends on the size of the binary number.

Decimal to Binary Conversion

$$1. (42)_{10} = (?)_2$$

Solution:

$$(42)_{10} = \underline{?}_2$$

2	42	
2	21	0
2	10	1
2	5	0
2	2	1
2	1	0
0	1	↑

$$\therefore (42)_{10} = (101010)_2$$

BINARY-TO-DECIMAL CONVERSION

- The decimal value of any binary number can be found by adding the weights of all bits that are 1 and discarding the weights of all bits that are 0.
- Example

Weight:	2^5	2^4	2^3	2^2	2^1	2^0
X						
Binary no:		1	0	1	1	0
Value		32	0	8	4	0
Sum						45

binary number (1110)₂ to a decimal number:

Hexadecimal Numbers

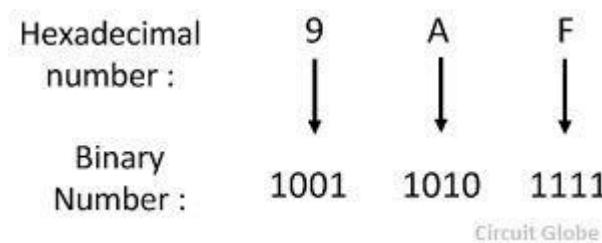
- The hexadecimal number system has sixteen digits and is used primarily as a compact way of displaying or writing binary numbers.
- Hexadecimal is widely used in computer and microprocessor applications.
- The hexadecimal system has a base of sixteen; it is composed of 16 digits and alphabetic characters.
- The maximum 3-digits hexadecimal number is FFF or decimal 4095 and maximum 4-digit hexadecimal number is FFFF or decimal 65.535

Hexadecimal Numbers

DECIMAL	HEX	BINARY
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hexadecimal to Binary Conversion

- To convert a hexadecimal number to a binary number, convert each hexadecimal digit to its four digit equivalent. For example, consider the hexadecimal number 9AF which is converted into a binary digit. The conversions are explained below.
- Therefore the equivalent binary number is 1001 1010 1111



BINARY-TO-HEXADECIMAL CONVERSION

- Simply break the binary number into 4-bit groups, starting at the right-most bit and replace each 4-bit group with the equivalent hexadecimal symbol as in the following example.
- **Convert the binary number to hexadecimal: 1100101001010111**
Solution:
- 1100 1010 0101 0111
- C A 5 7 = CA57

- **HEXADECIMAL-TO-DECIMAL CONVERSION**
- One way to find the decimal equivalent of a hexadecimal number is to first convert the hexadecimal number to binary and then convert from binary to decimal.
- Convert the hexadecimal number 1C to decimal:

$$\begin{array}{r} 1 \quad C \\ 0001 \quad 1100 = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = 28 \end{array}$$

DECIMAL-TO-HEXADECIMAL CONVERSION

- Repeated division of a decimal number by 16 will produce the equivalent hexadecimal number, formed by the remainders of the divisions. The first remainder produced is the least significant digit (LSD). Each successive division by 16 yields a remainder that becomes a digit in the equivalent hexadecimal number. When a quotient has a fractional part, the fractional part is multiplied by the divisor to get the remainder.

Contd..

- Convert the decimal number 650 to hexadecimal by repeated division by 16
-
- $650 / 16 = 40.625$ $0.625 \times 16 = 10 = A$ (LSD)
- $40 / 16 = 2.5$ $0.5 \times 16 = 8 = 8$
- $2 / 16 = 0.125$ $0.125 \times 16 = 2 = 2$ (MSD)
- The hexadecimal number is 28A

OCTAL NUMBERS

- the octal system provides a convenient way to express binary numbers and codes.
- it is used less frequently than hexadecimal in conjunction with computers and microprocessors to express binary quantities for input and output purposes.
- The octal system is composed of eight digits, which are: 0, 1, 2, 3, 4, 5, 6, 7
- To count above 7, begin another column and start over: 10, 11, 12, 13, 14, 15, 16, 17, 20, 21 and so on. Counting in octal is similar to counting in decimal, except that the digits 8 and 9 are not used.

OCTAL TO DECIMAL

Decimal	Octal	Binary
0	0	000
1	1	001
2	2	010
3	3	011
4	4	100
5	5	101
6	6	110
7	7	111

OCTAL-TO-DECIMAL CONVERSION

- Octal number system has a base of eight, each successive digit position is an increasing power of eight, beginning in the right-most column with 8^0 .
- The evaluation of an octal number in terms of its decimal equivalent is accomplished by multiplying each digit by its weight and summing the products.

OCTAL-TO-DECIMAL CONVERSION

- Let's convert octal number 2374 in decimal number.
- Weight: 8^3 8^2 8^1 8^0
- Octal number: 2 3 7 4
- **$2374 = (2 \times 8^3) + (3 \times 8^2) + (7 \times 8^1) + (4 \times 8^0) = 1276$**

DECIMAL-TO-OCTAL CONVERSION

- A method of converting a decimal number to an octal number is the repeated division-by-8 method, which is similar to the method used in the conversion of decimal numbers to binary or to hexadecimal.
- Let's convert the decimal number 359 to octal. Each successive division by 8 yields a remainder that becomes a digit in the equivalent octal number. The first remainder generated is the least significant digit (LSD).

DECIMAL-TO-OCTAL CONVERSION

- Convert to its octal equivalent:
- $427 / 8 = 53 \text{ R}3$ Divide by 8; R is LSD
- $53 / 8 = 6 \text{ R}5$ Divide Q by 8; R is next digit
- $6 / 8 = 0 \text{ R}6$ Repeat until Q = 0
- 653

OCTAL-TO-BINARY CONVERSION

- Because each octal digit can be represented by a 3-bit binary number, it is very easy to convert from octal to binary.
-
- **Octal/Binary Conversion**
- Octal Digit 0 1 2 3 4 5 6 7
- Binary 000 001 010 011 100 101 110 111
-
- Let's convert the octal numbers 25 and 140.
- Octal Digit 2 5 1 4 0
- Binary 010 101 001 100 000

BINARY-TO-OCTAL CONVERSION

- Conversion of a binary number to an octal number is the reverse of the octal-to-binary conversion.
- Let's convert the following binary numbers to octal:

- 1 1 0 1 0 1

- 1 0 1 1 1 0 0 1

- 6 5 = 65

- 5 7 1 = 571

Octal to Hexadecimal Conversion

1. Convert the octal number into **binary** and then convert the binary into hexadecimal.
2. Convert the octal number into **decimal** and then convert the decimal into hexadecimal.

Step 1 : Convert $(56)_8$ into Binary

In order to convert the octal number into binary, we need to express every octal value using 3 binary bits.

Binary equivalent of **5** is $(101)_2$.

Binary equivalent of **6** is $(110)_2$.

$$= (56)_8$$

$$= (101)(110)$$

$$= (101110)_2$$

Step 2 : Convert $(101110)_2$ into Hexadecimal

In order to convert the binary number into hexadecimal, we need to group every 4 binary bits and calculate the value[From left to right].

$(101110)_2$ in hexadecimal

$$= (101110)_2$$

$$= (10)(1110)$$

$$= (2)(14)$$

$$= (2e)_{16}$$

Octal Decimal Hexadecimal

Step 1 : Convert $(56)_8$ into Decimal

$$= 5*8^1 + 6*8^0$$

$$= 40 + 6$$

$$= (46)_{10}$$

Step 2 : Convert $(46)_{10}$ into hexadecimal

$$\begin{array}{r} 16 \mid 46 \\ \hline 2 \end{array} \quad \begin{array}{r} 16 \mid 2 \\ \hline 14 \end{array}$$

$$= (2e)_{16}$$

Hexadecimal to Octal Conversion

- **Hexadecimal-Binary-Octal**

Convert $(ff)_{16}$

Step 1: $(ff)_{16}$

$$= (1111)(1111)$$

$$= (11111111)_2$$

Step2: $(11111111)_2$ in Octal

$$= (11111111)_2$$

$$= (11)(111)(111)$$

$$= (377)_8$$

Hexadecimal-Decimal-Octal

Convert $(ff)_{16}$

$$\text{Step1: } 15*16^1 + 15*16^0$$

$$= 240 + 15$$

$$= (255)_{10}$$

Step2:

$$\begin{array}{r} 255 \\ \hline 8 | 31 \end{array} -$$

$$\begin{array}{r} 31 \\ \hline 8 | 3 \end{array} - 7$$

$$\begin{array}{r} 3 \\ \hline 8 | \end{array} - 7$$

$$= (377)_8$$

Overview of Computers

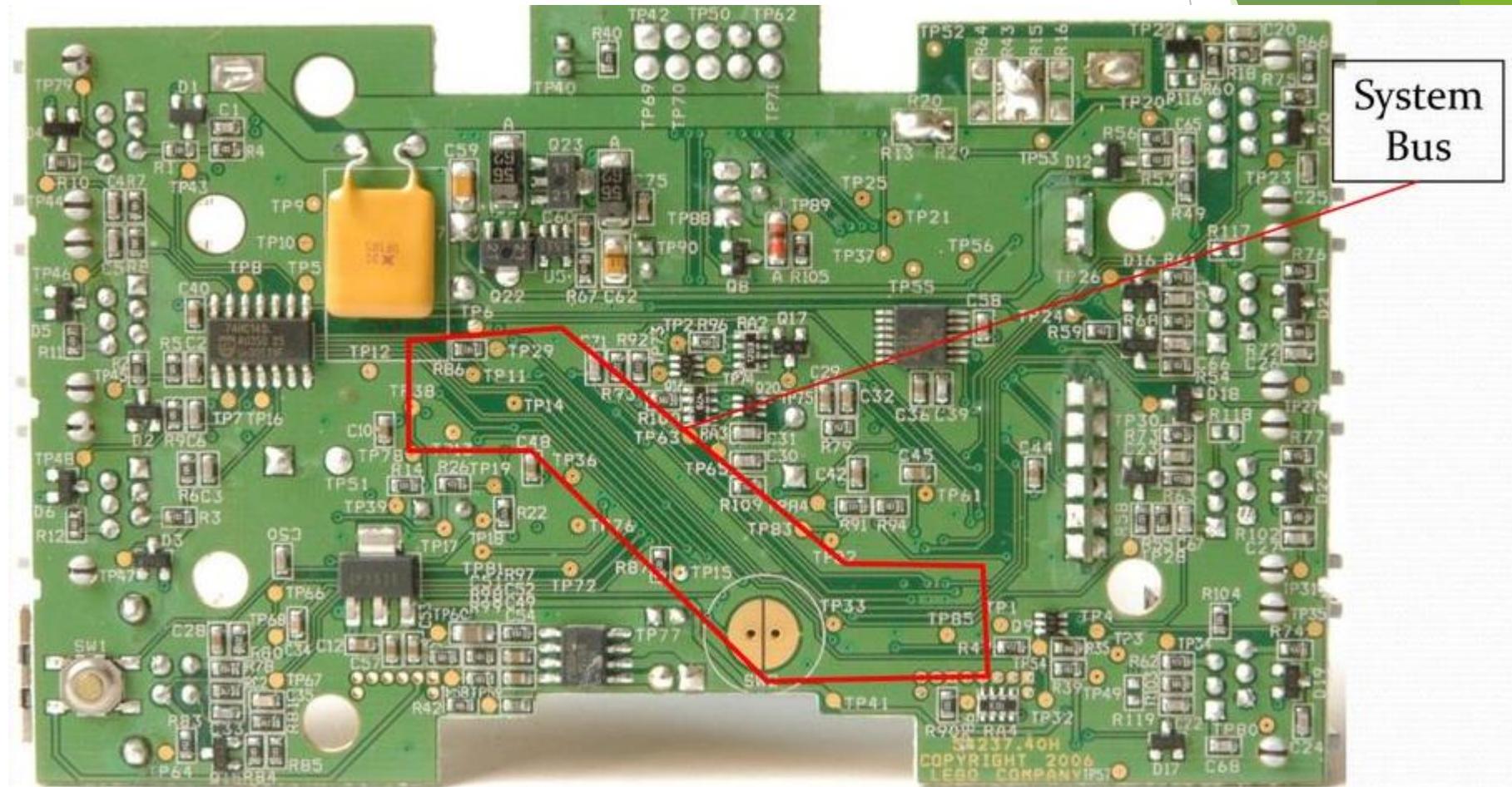
Dr. B Krishna Priya

System Bus in Computer Architecture

Bus

- ▶ The CPU sends various data values, instructions and information to all the devices and components inside the computer.
- ▶ If you look at the bottom of a motherboard you'll see a whole network of lines or electronic pathways that join the different components together.
- ▶ This network of wires or electronic pathways is called the 'Bus'

Bus



Bus

- ▶ A bus is a communication pathway connecting two or more devices.
- ▶ A key characteristic of a bus is that it is a shared transmission medium.
- ▶ Multiple devices connect to the bus, and a signal transmitted by any one device is available for reception by all other devices attached to the bus.
- ▶ If two devices transmit during the same time period, their signals will overlap and become garbled. Thus, only one device at a time can successfully transmit.

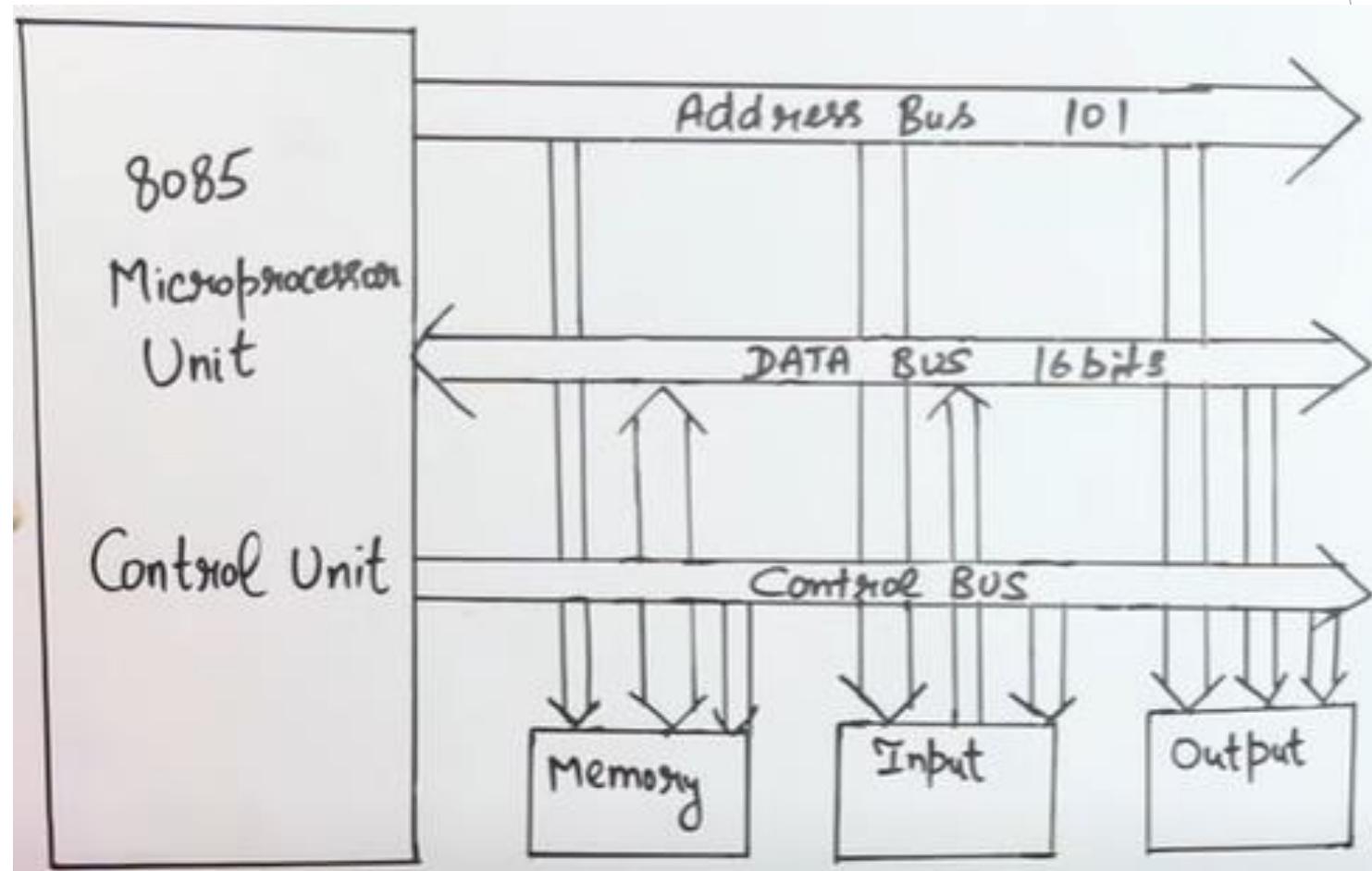
Bus

- ▶ Typically, a bus consists of multiple communication pathways, or lines.
- ▶ Each line is capable of transmitting signals representing binary 1 and binary 0.
- ▶ Several lines of a bus can be used to transmit binary digits simultaneously (in parallel).
- ▶ For example, an 8-bit unit of data can be transmitted over eight bus lines.
- ▶ Computer systems contain a number of different buses that provide pathways between components at various levels of the computer system hierarchy.

System Bus

- ▶ A bus that connects major computer components (processor, memory, I/O) is called a system bus.
- ▶ A system bus consists, typically, of from about fifty to hundreds of separate lines. Each line is assigned a particular meaning or function.
- ▶ System bus usually is separated into three functional groups .
 - ▶ 1. Data Bus
 - ▶ 2. Address Bus
 - ▶ 3. Control Bus
- ▶ In addition, there may be power distribution lines that supply power to the attached modules.

System Bus Architecture



Data Bus

- ▶ A collection of wires through which data is transmitted from one part of a computer to another.
- ▶ Data Bus can be thought of as a highway on which data travels within a computer.
- ▶ This bus connects all the computer components to the CPU and main memory.
- ▶ The data bus may consist of 32, 64, 128, or even more separate lines.
- ▶ The number of lines being referred to as the width of the data bus. Because each line can carry only 1 bit at a time, the number of lines determines how many bits can be transferred at a time.

Data Bus

- ▶ Bidirectional bus connects with memory.
- ▶ The size (width) of bus determines how much data can be transmitted at one time.
- ▶ A 16-bit bus can transmit 16 bits (2 bytes) of data at a time.
- ▶ 32-bit bus can transmit 32 bits(4 bytes) at a time.
- ▶ The size (width) of bus is a critical parameter in determining system performance.
- ▶ The wider the data bus, the better, but they are expensive.

Address Bus

- ▶ A collection of wires used to identify particular location in main memory is called Address Bus.
- ▶ Or in other words, the information used to describe the memory locations travels along the address bus.
- ▶ Clearly, the width of the address bus determines the maximum possible memory capacity of the system.
- ▶ N address lines directly address 2^N memory locations.

Address Bus

- ▶ It is an unidirectional bus.
- ▶ The CPU sends address to a particular memory locations and I/O ports.
- ▶ The address bus consists of 16 , 20 , 24 or more parallel signal lines.

Control Bus

- ▶ Because the data and address lines are shared by all components, there must be a means of controlling their use.
- ▶ **The control lines regulates the activity on the bus.**
- ▶ Control signals transmit both command and timing information among system modules.
- ▶ **The control bus carries signals that report the status of various devices.**

Control Bus

- ▶ Typical control bus signals are :
- ▶ Memory Read : causes data from the addressed location to be placed on the data bus.
- ▶ Memory Write : causes data on the bus to be written into the addressed location
- ▶ I/O write: causes data on the bus to be output to the addressed I/O port
- ▶ I/O read: causes data from the addressed I/O port to be placed on the bus

I/O Interface

- ▶ Direct Communication b/w processor and devices is not possible.
- ▶ It is due to:
 - ▶ Different manner of operation
 - ▶ Data transfer rate
 - ▶ Difference in word format
 - ▶ Difference in operating modes of peripherals

I/O Interface

I/O Bus & Interface Module

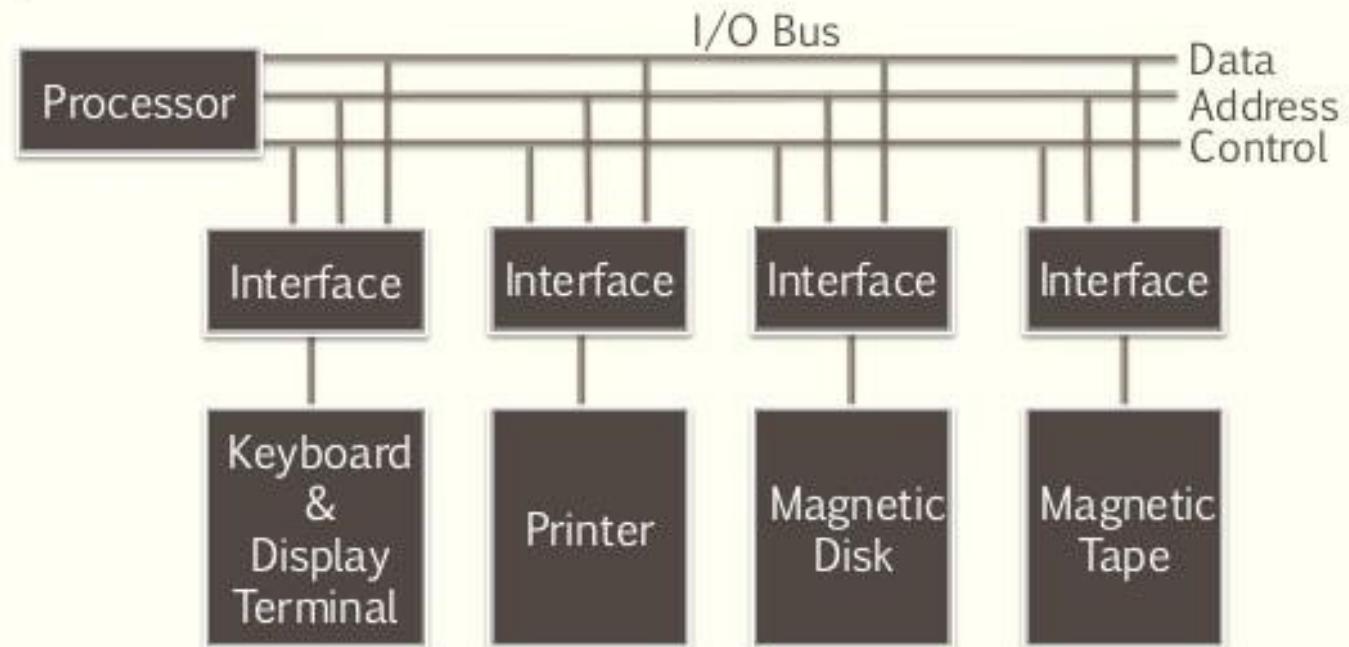
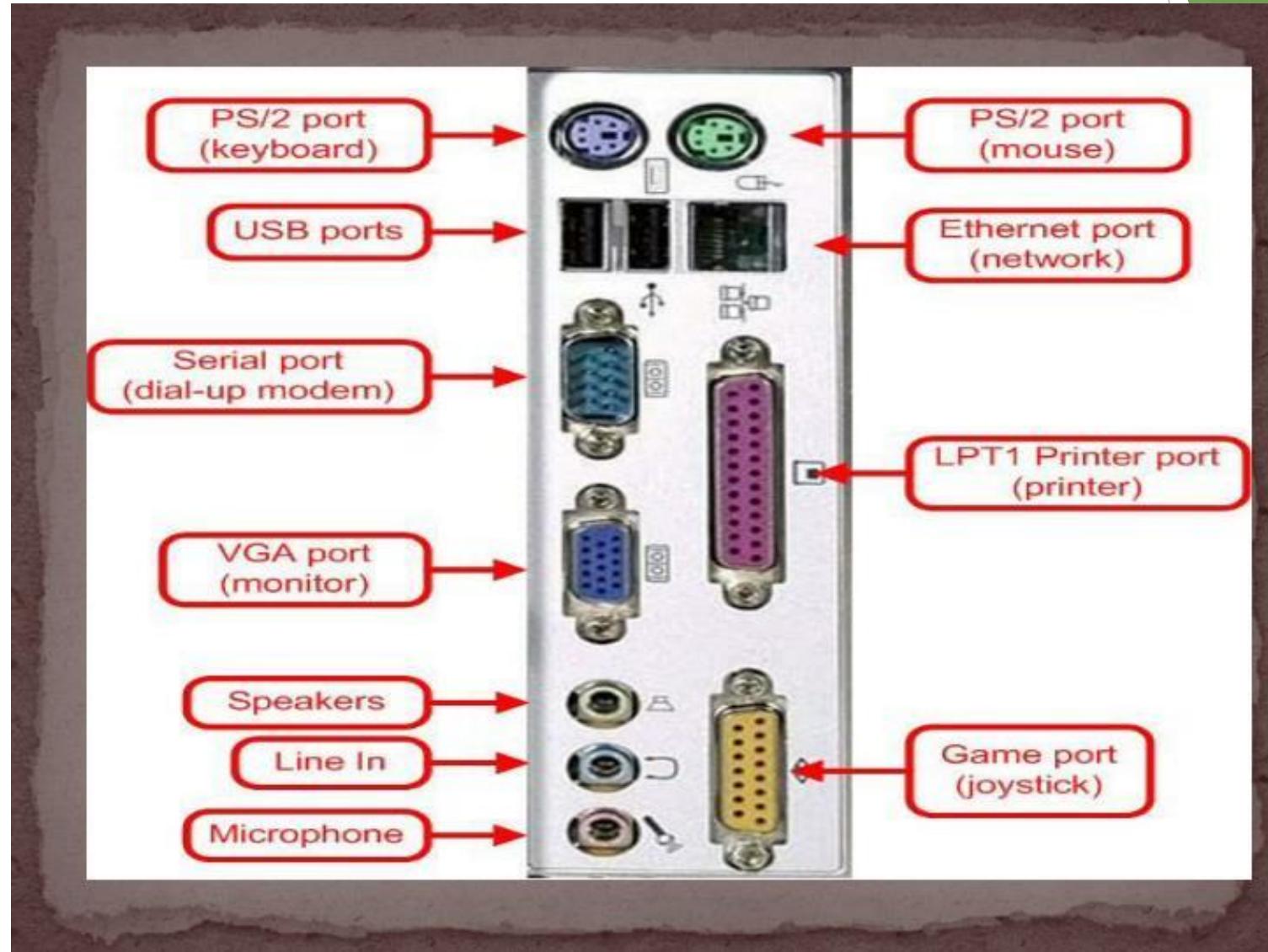


Fig. Connection of I/O Bus to input-output devices

Ports

- ▶ The point at which a peripheral attaches to.
- ▶ Communicates with a system unit so that the peripheral can send data to or receive information from the computer.
- ▶ The term JACK is sometimes used to identify audio and video ports.
- ▶ The front and back of system unit on desktop personal computer contain many ports.



Connectors

- ▶ Joins a cable and a port.
- ▶ A connector at one end of the cable attaches to a port on the **system unit**.
- ▶ A connector at the other end of the cable attaches to a port on the **peripheral**.



Motherboard

- ▶ A motherboard sometimes called the “Mobo” is the main circuit board found in computers and other advanced electronic devices.

Basic ATX Motherboard



Motherboard

- ▶ The motherboard gets its name because it is like a mother to all of the other circuit boards.
- ▶ The motherboard is the largest circuit board in the computer case and may have many smaller boards plugged into it.
- ▶ It's the glue that holds all of the most important parts of the computer together.

Motherboard

Form factor sizes of Motherboards



Standard-ATX



Micro-ATX



Mini-ITX



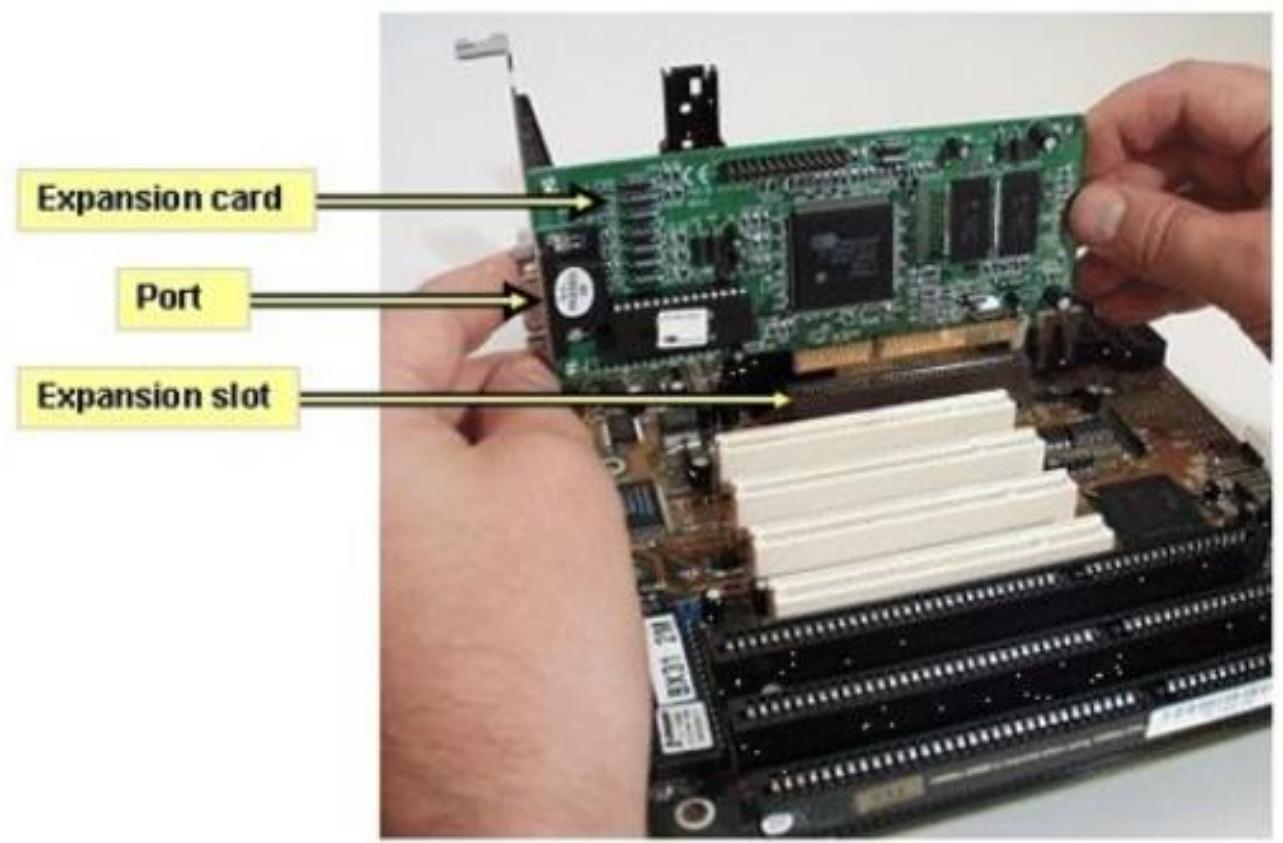
Nano-ITX



Pico-ITX

Motherboard cards

- ▶ On the motherboard, you will find several expansion cards. Each of these cards has a special purpose



Video card

- ▶ A video card is an type of card which generates a feed of images to a monitor or display.
- ▶ Cheaper computer systems may have integrated video capabilities. These built in video capabilities are less powerful than stand alone video cards.



High End Powerful
Video Card

Low End Basic
Video Card



Network card



Wireless WiFi
Network Card



Wired Network Card

Jumpers

- ▶ Jumpers allow the computer to close an electrical circuit, allowing the electricity to flow on a circuit board and perform a function.
- ▶ Jumpers consist of small pins that can be covered with a small plastic box (jumper block).



$$a+b = \text{LED}$$

$$a - b = 2$$

$$a \times c = 3$$

- ▶ For example, if a jumper on a hard drive is in "Position A", it may mean that the hard drive is to be the primary hard drive on the system.
- ▶ If the jumper is in "Position B" it may mean that the hard drive is to be the secondary hard drive in the computer.
- ▶ Jumpers are rare on most newer hardware today because of automatic configurations and software-controlled settings.

Overview of Computers

Dr. B Krishna Priya



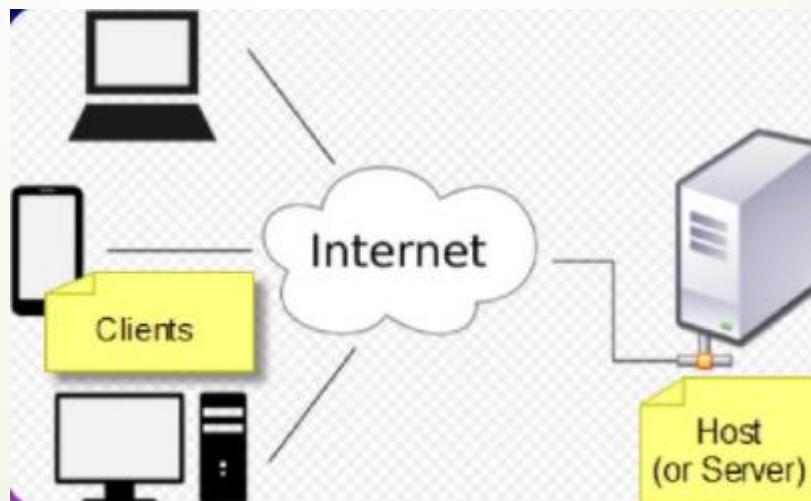


Server

- ▶ You may have heard the term server before, but what exactly is a Server?
- ▶ A Server in its most basic form:
 - ▶ Is a computer device that provides a service to another computer program and its user
- ▶ These server programs range in variety depending on what the server is designed for
- ▶ Using internet technologies, devices are able to establish a connection to a server to access it and make use of the resources or programs on the server

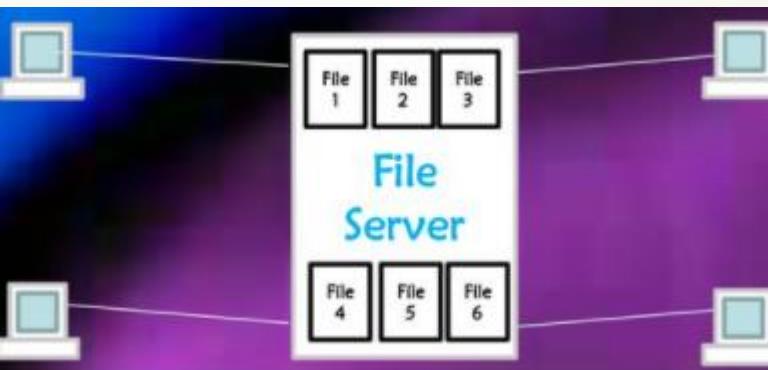
Server

- ▶ The essential point is that a server offers a “service” to devices connected to it
- ▶ On a network, a server is known as a host
- ▶ The devices that are connecting to the server are known as clients



Server Example

- ▶ The example shown below is known as a **File Server** because this server holds files that are shared around the business
- ▶ Instead of storing these files on their own computers, they can log into any computer that can access the server
- ▶ From here they can look for the file on the server, open it, edit it, save it, etc; and if another member is looking for the file they can connect to the server and access that same file





Server

- ▶ Servers can come in many shapes and forms.
- ▶ Technically any computer with a suitable server operating system / software installed, and a way for devices to connect to it, is capable of being a server
- ▶ However in most cases, servers are specialized computers which are designed and built for the specific task of being a server:
 - ▶ Servers are designed to be running 24-7
 - ▶ The processing power of servers can generate a lot of heat
 - ▶ They need to be capable of computing and handling the tasks they are built for
 - ▶ They need to be able to cope with the number of clients that are accessing it

Server

- ▶ For this reason you will find that servers often don't look anything like a typical desktop computer, laptop or mobile device.
- ▶ This is an image of a server that a company or organization may use.



- ▶ This server has server racks mounted inside a data center server cabinet





Types of Server

- ▶ There are many different types of Servers that are used for a range of purposes, below are a few common uses for servers:
- ▶ **File Server:** Stores files that can be accessed and shared with many clients
- ▶ **Web Server:** Holds and stores webpages for clients to access
- ▶ **Database Server:** Similar to a file server, but stores and holds a database system, again for many clients to access
- ▶ **Application Server:** Holds applications for clients to access
- ▶ **Game Server:** Holds a game/games for clients to access and share. Typically used for online gaming

Bootloader

- ▶ A bootloader or bootstrap loader, is a computer program that is responsible for booting a computer.
- ▶ When a computer is turned off, its software—including operating systems, application code, and data—remains stored on non-volatile memory.
- ▶ When the computer is powered on, it typically does not have an operating system or its loader in (RAM).
- ▶ The computer first executes a relatively small program stored in ROM to initialize RAM and load operating system into RAM.



Introduction to Database Management Systems

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use



Database Management System (DBMS)

- DBMS contains information about a particular enterprise
 - Collection of interrelated data
 - Set of programs to access the data
 - An environment that is both *convenient* and *efficient* to use
- Database Applications:
 - Banking: transactions
 - Airlines: reservations, schedules
 - Universities: registration, grades
 - Sales: customers, products, purchases
 - Online retailers: order tracking, customized recommendations
 - Manufacturing: production, inventory, orders, supply chain
 - Human resources: employee records, salaries, tax deductions
- Databases can be very large.
- Databases touch all aspects of our lives



University Database Example

- Application program examples
 - Add new students, instructors, and courses
 - Register students for courses, and generate class rosters
 - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems

STUDENT	Name	Roll_No	Class	Major	COURSE	CourseName	CourseNo	Dept
	Smith	17	1	CS		Data Structures	CS1310	CS
	Brown	8	2	CS		Discrete Mathematics	MATH2410	MATH
						Database	CS380	CS
GRADE_REPORT	Roll_No	CourseNo	Grade					
	17	MATH2410	B					
	17	CS1310	A					
	8	CS1310	A					



Drawbacks of using file systems to store data

- Data redundancy and inconsistency
 - ▶ Multiple file formats, duplication of information in different files
- Difficulty in accessing data
 - ▶ Need to write a new program to carry out each new task
- Data isolation — multiple files and formats
- Integrity problems
 - ▶ Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
 - ▶ Hard to add new constraints or change existing ones



Drawbacks of using file systems to store data (Cont.)

- Atomicity of updates
 - ▶ Failures may leave database in an inconsistent state with partial updates carried out
 - ▶ Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
 - ▶ Concurrent access needed for performance
 - ▶ Uncontrolled concurrent accesses can lead to inconsistencies
 - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
 - ▶ Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems



Data Models

- A collection of tools for describing
 - Data
 - Data relationships
 - Data semantics
 - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
 - Network model
 - Hierarchical model



Relational Model

- Relational model (Chapter 2)
- Example of tabular data in the relational model

The diagram shows a relational table with four columns: ID, name, dept_name, and salary. Two arrows point to the table from the right. One arrow points upwards from the bottom-right corner to the header row, labeled "Columns". Another arrow points downwards from the bottom-right corner to the first data row, labeled "Rows".

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table



A Sample Relational Database

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table



Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
 - DML also known as query language
- Two classes of languages
 - ▶ **Pure** – used for proving properties about computational power and for optimization
 - Relational Algebra (we focus in this course)
 - Tuple relational calculus
 - Domain relational calculus
 - ▶ **Commercial** – used in commercial systems
 - SQL is the most widely used commercial language



Data Definition Language (DDL)

- Specification notation for defining the database schema

Example:

```
create table instructor (
    ID      char(5),
    name    varchar(20),
    dept_name varchar(20),
    salary   numeric(8,2))
```

- DDL compiler generates a set of table templates stored in a **data dictionary**
- Data dictionary contains metadata (i.e., data about data)
 - Database schema
 - Integrity constraints
 - ▶ Primary key (ID uniquely identifies instructors)
 - ▶ Referential integrity (**references** constraint in SQL)
 - e.g. *dept_name* value in any *instructor* tuple must appear in *department* relation
 - Authorization



SQL

- **SQL**: widely used non-procedural language
 - Example: Find the name of the instructor with ID 22222

```
select name
from instructor
where instructor.ID = '22222'
```
 - Example: Find the ID and building of instructors in the Physics dept.

```
select instructor.ID, department.building
from instructor, department
where instructor.dept_name = department.dept_name and
      department.dept_name = 'Physics'
```
- Application programs generally access databases through one of
 - Language extensions to allow embedded SQL
 - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database
- Chapters 3, 4 and 5



Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
 - Business decision – What attributes should we record in the database?
 - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database



Database Design?

- Is there any problem with this design?

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000



Design Approaches

- Entity Relationship Model
 - Models an enterprise as a collection of *entities* and *relationships*
 - ▶ Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
 - Described by a set of *attributes*
 - ▶ Relationship: an association among several entities
 - Represented diagrammatically by an *entity-relationship diagram*:
- Normalization Theory
 - Formalize what designs are bad, and test for them



Object-Relational Data Models

- Relational model: flat, “atomic” values
- Object Relational Data Models
 - Extend the relational data model by including object orientation and constructs to deal with added data types.
 - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
 - Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
 - Provide upward compatibility with existing relational languages.



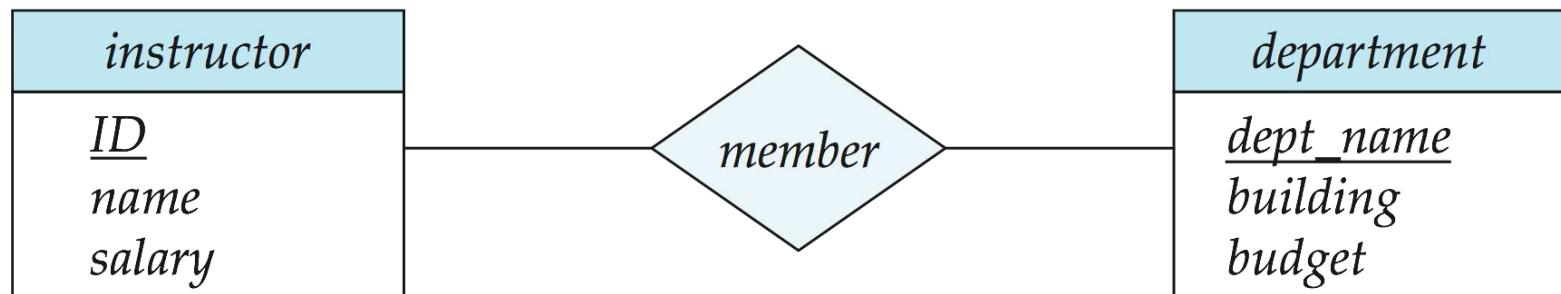
XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data



The Entity-Relationship Model

- Models an enterprise as a collection of *entities* and *relationships*
 - Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
 - ▶ Described by a set of *attributes*
 - Relationship: an association among several entities
- Represented diagrammatically by an *entity-relationship diagram*:



What happened to dept_name of **instructor** and **student**?

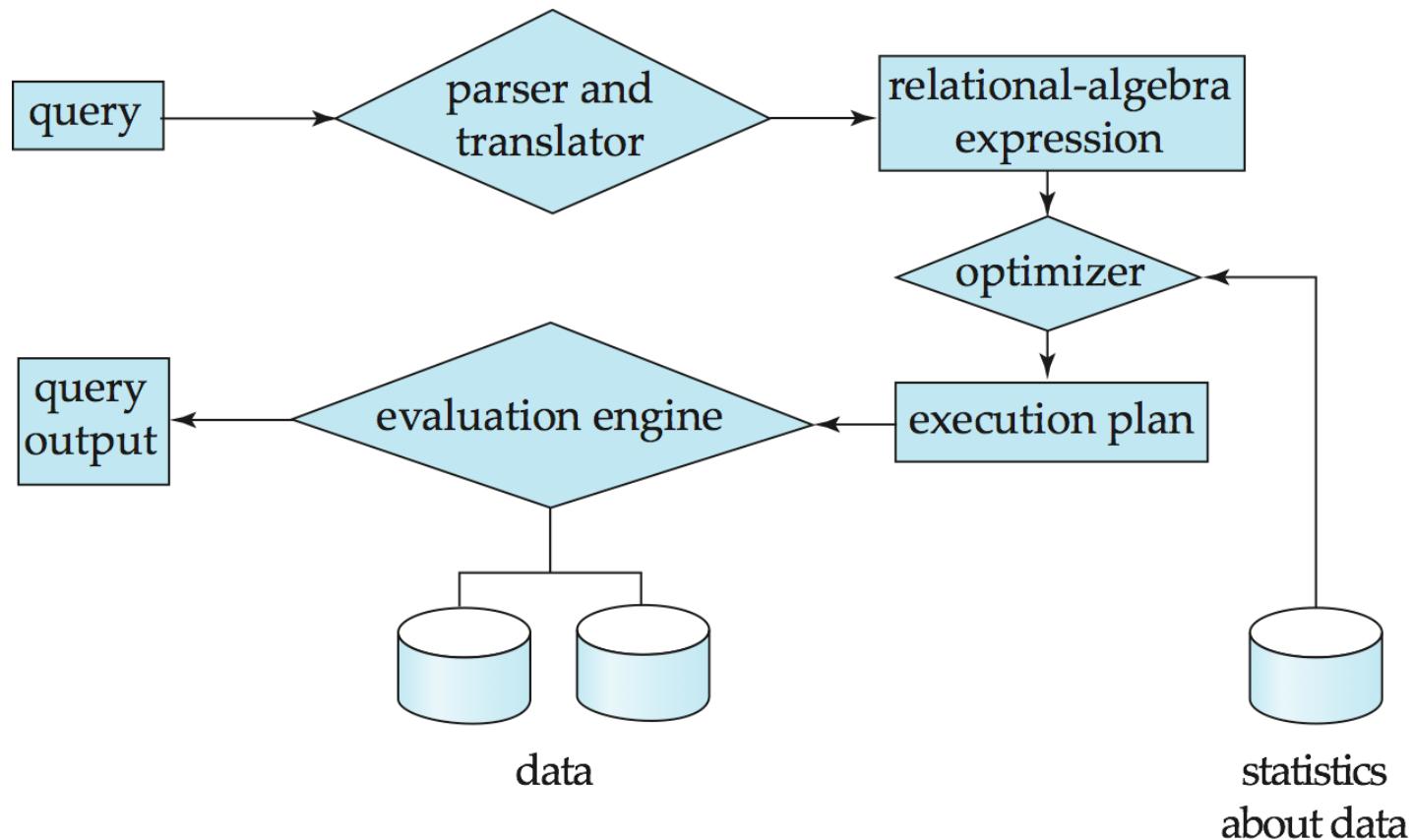


Storage Management

- **Storage manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
 - Interaction with the file manager
 - Efficient storing, retrieving and updating of data
- Issues:
 - Storage access
 - File organization
 - Indexing and hashing

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



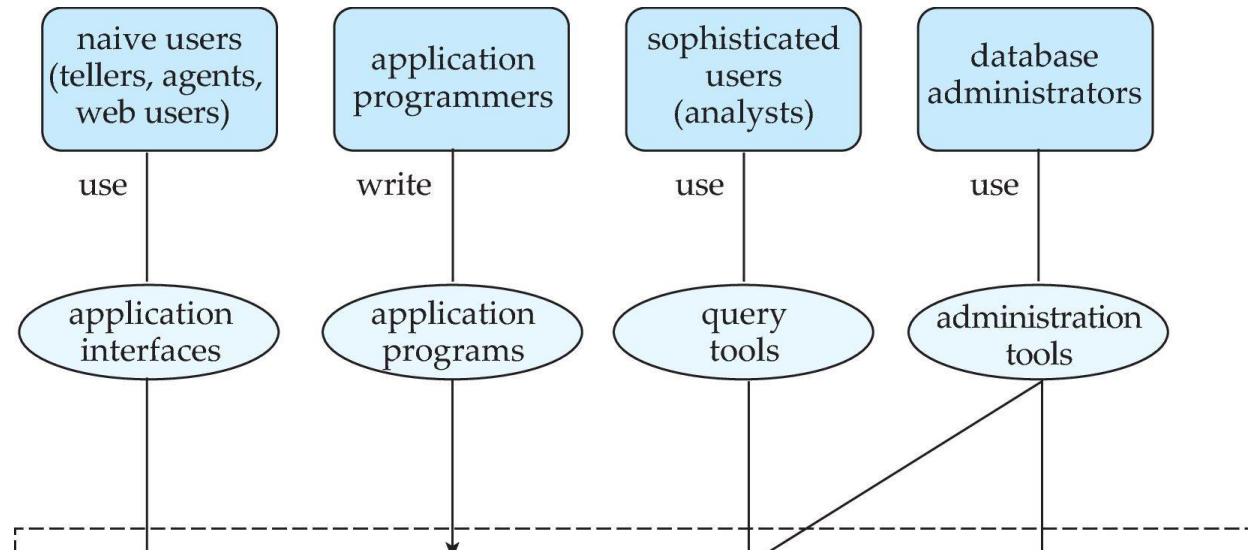


Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.



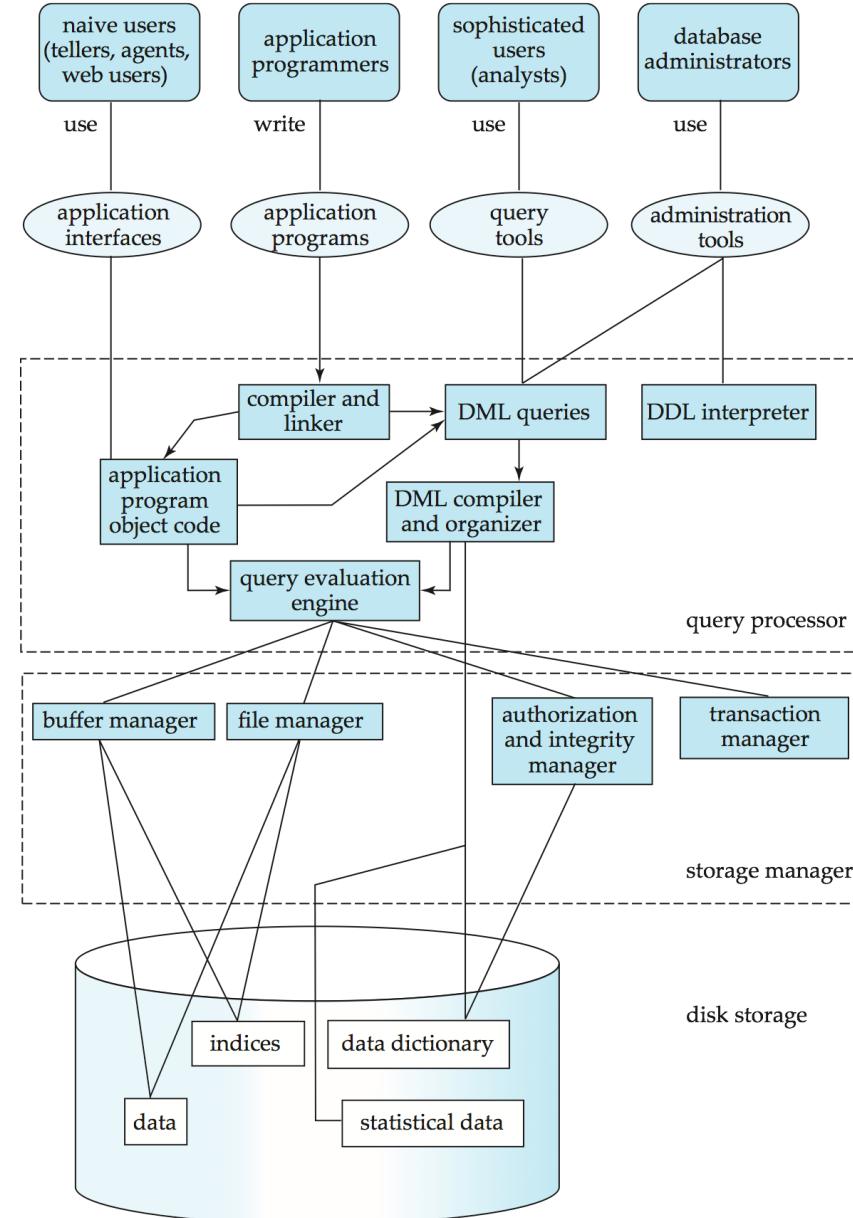
Database Users and Administrators



Database



Database System Internals





Database Architecture

The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed



History of Database Systems

- 1950s and early 1960s:
 - Data processing using magnetic tapes for storage
 - ▶ Tapes provided only sequential access
 - Punched cards for input
- Late 1960s and 1970s:
 - Hard disks allowed direct access to data
 - Network and hierarchical data models in widespread use
 - Ted Codd defines the relational data model
 - ▶ Would win the ACM Turing Award for this work
 - ▶ IBM Research begins System R prototype
 - ▶ UC Berkeley begins Ingres prototype
 - High-performance (for the era) transaction processing



History (cont.)

- 1980s:
 - Research relational prototypes evolve into commercial systems
 - ▶ SQL becomes industrial standard
 - Parallel and distributed database systems
 - Object-oriented database systems
- 1990s:
 - Large decision support and data-mining applications
 - Large multi-terabyte data warehouses
 - Emergence of Web commerce
- Early 2000s:
 - XML and XQuery standards
 - Automated database administration
- Later 2000s:
 - Giant data storage systems
 - ▶ Google BigTable, Yahoo PNuts, Amazon, ..

OVERVIEW OF COMPUTER WORKSHOP

Dr.S.Manipriya & Dr. P.V. Arun

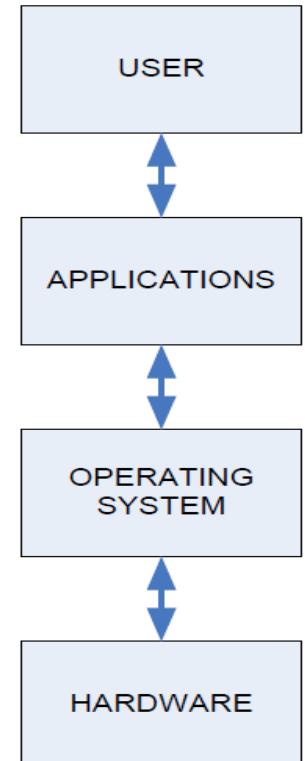
Operating Systems

- Program for overall operation of computer
- User can store and retrieve files
- Request the execution of programs, and provides the environment necessary to execute the programs requested
 - An Operating System (OS) is a program that manages the computer hardware.
 - It also provides a basis for Application Programs and acts as an intermediary between computer User and compter Hardware.

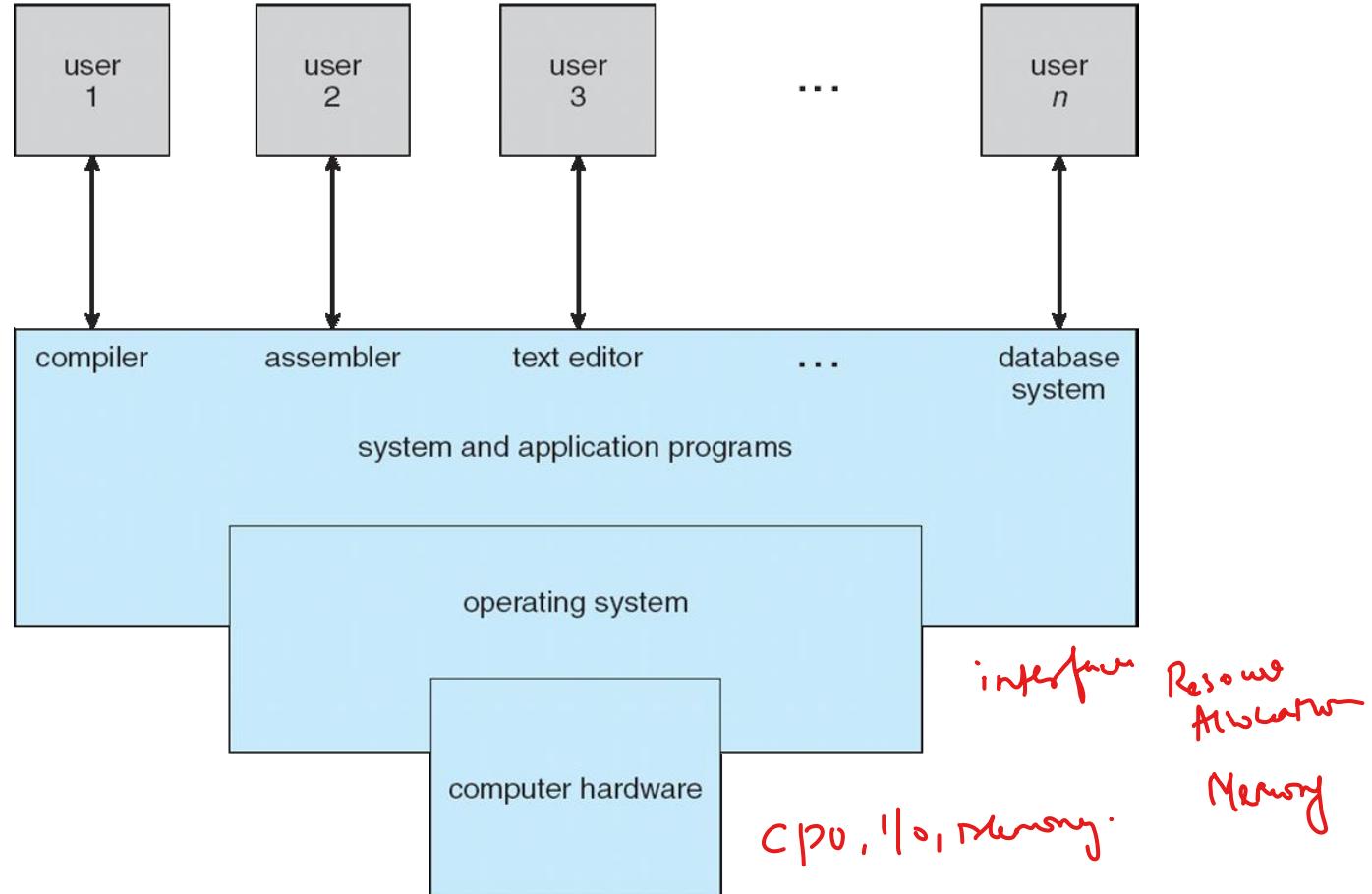
Computer System Structure

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



Operating System



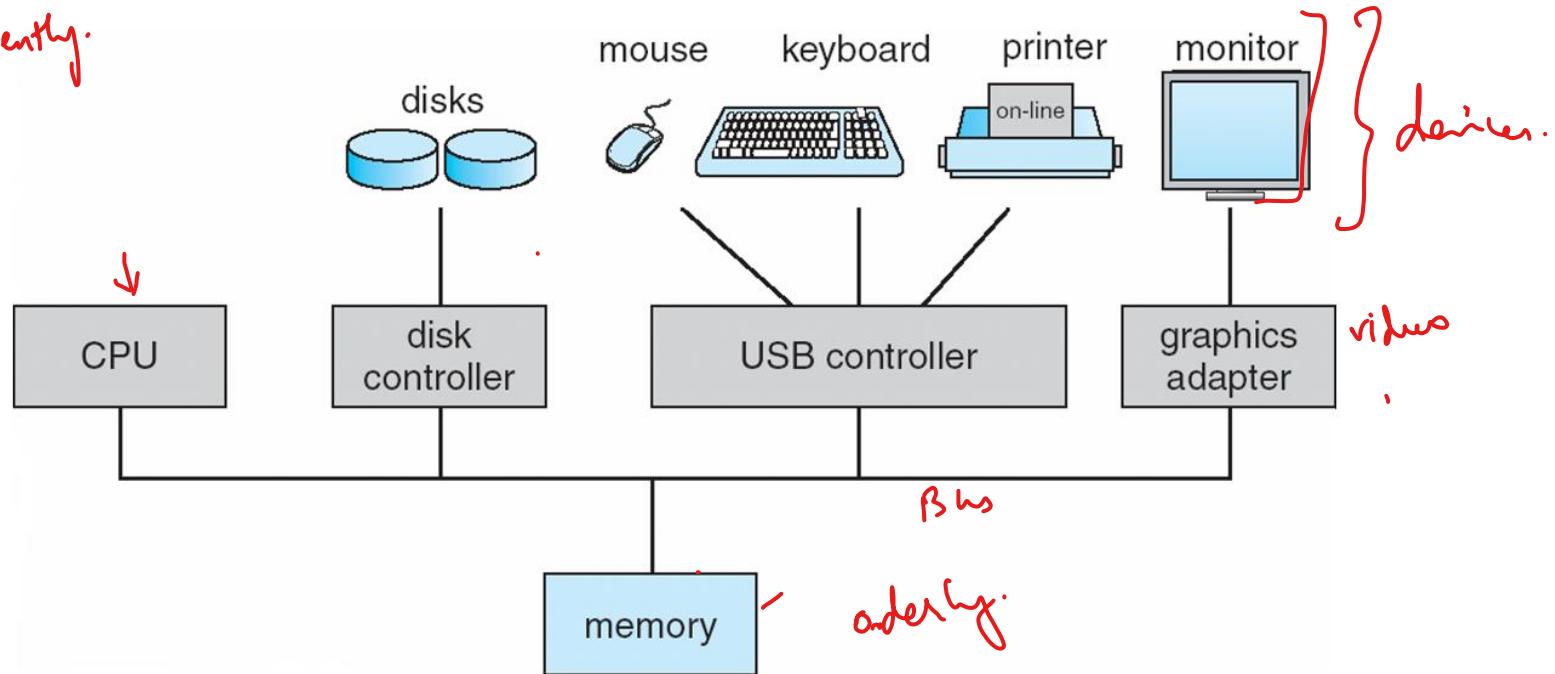
Goals of An Operating System

□ 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

Operating System- Computer System Operation

Concurrently.



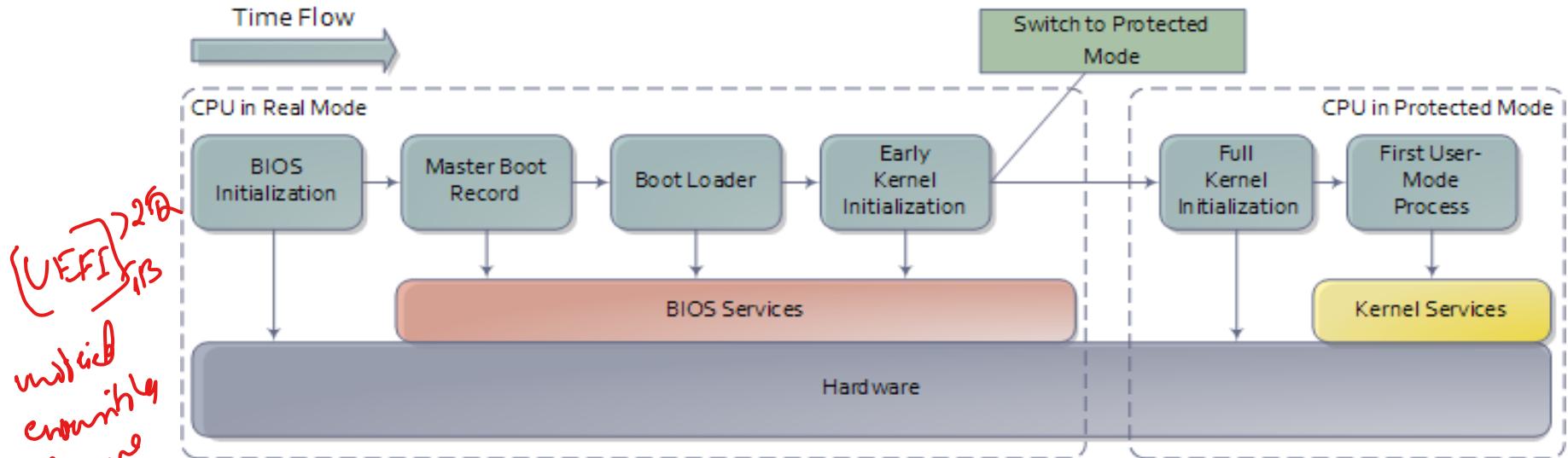
Central Processing Unit

- This is the brain of your computer.
- It performs all of the calculations.
- In order to do its job, the CPU needs commands to perform, and data to work with.
- The instructions and data travel to and from the CPU on the system bus.
- The operating system provides rules for how that information gets back and forth, and how it will be used by the CPU.

RAM – Random Access Memory

- This is like a desk, or a workspace, where your computer temporarily stores all of the information (data) and instructions (software or program code) that it is currently using.
- Each RAM chip contains millions of address spaces.
- Each address space is the same size, and has its own unique identifying number (address).
- The operating system provides the rules for using these memory spaces, and controls storage and retrieval of information from RAM.
- Device drivers for RAM chips are included with the operating system.

Starting an Operating System(Booting)



- Firewall
- ✓ Power On Switch sends electricity to the motherboard on a wire called the *Voltage Good* line.
 - ✓ If the power supply is good, then the **BIOS (Basic Input/Output System)** chip takes over.
 - ✓ In Real Mode, CPU is only capable of using approximately 1 MB of memory built into the motherboard.
 - ✓ The BIOS will do a **Power-On Self Test (POST)** to make sure that all hardware are working.

- ✓ BIOS will then look for a small sector at the very beginning of your primary hard disk called **MBR**. *Master Boot Record*.
- ✓ The MBR contains a list, or map, of all of the **partitions** on your computer's hard disk (or disks).
- ✓ After the MBR is found the **Bootstrap Loader** follows basic instructions for starting up the rest of the computer, including the operating system.
- ✓ In **Early Kernel Initialization** stage, a smaller core of the Kernel is activated.
- ✓ This core includes the **device drivers** needed to use computer's **RAM chips**.

BIOS

- BIOS firmware was stored in a ROM/EPROM (Erasable Programmable Read-Only Memory) chip known as **firmware** on the PC motherboard.
- BIOS can be accessed during the initial phases of the boot procedure by pressing del, F2 or F10.
- Finally, the firmware code cycles through all storage devices and looks for a **boot-loader**. (usually located in first sector of a disk which is 512 bytes)
- If the boot-loader is found, then the firmware hands over control of the computer to it.

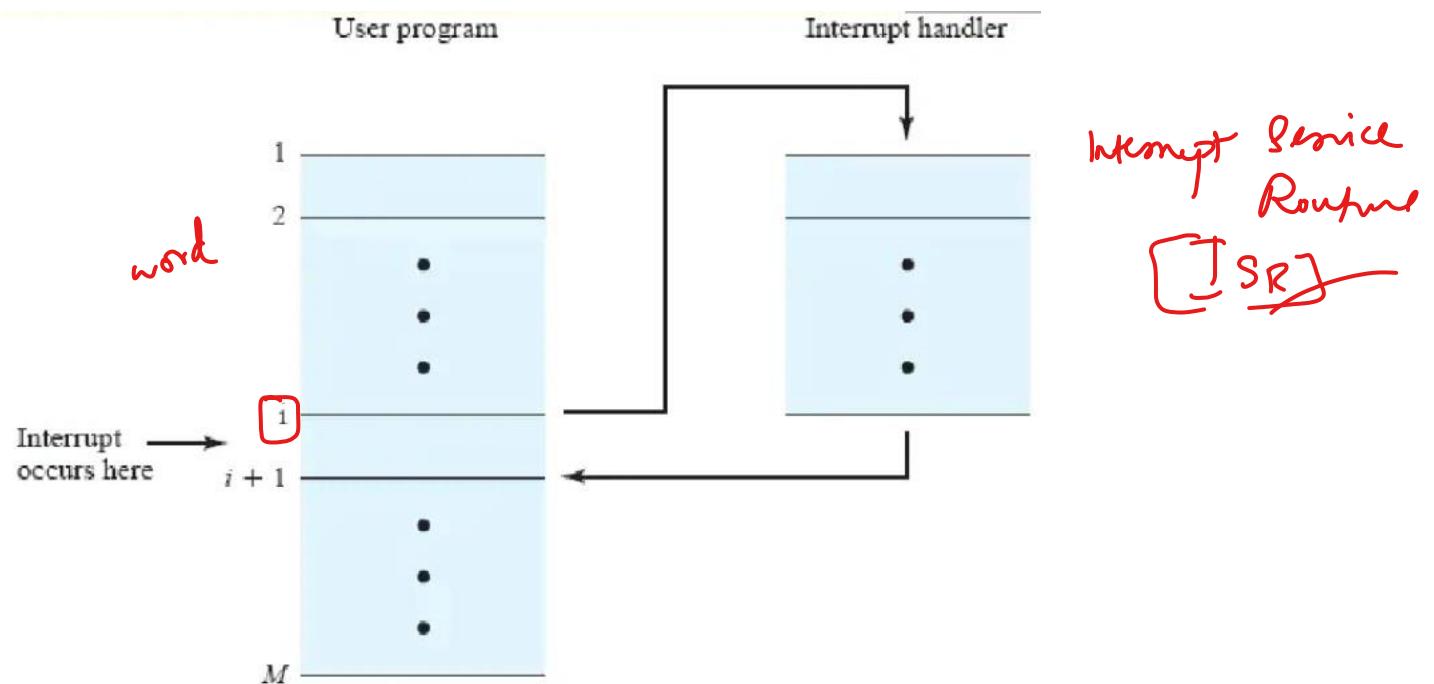
Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- An operating system is interrupt driven

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt Handling

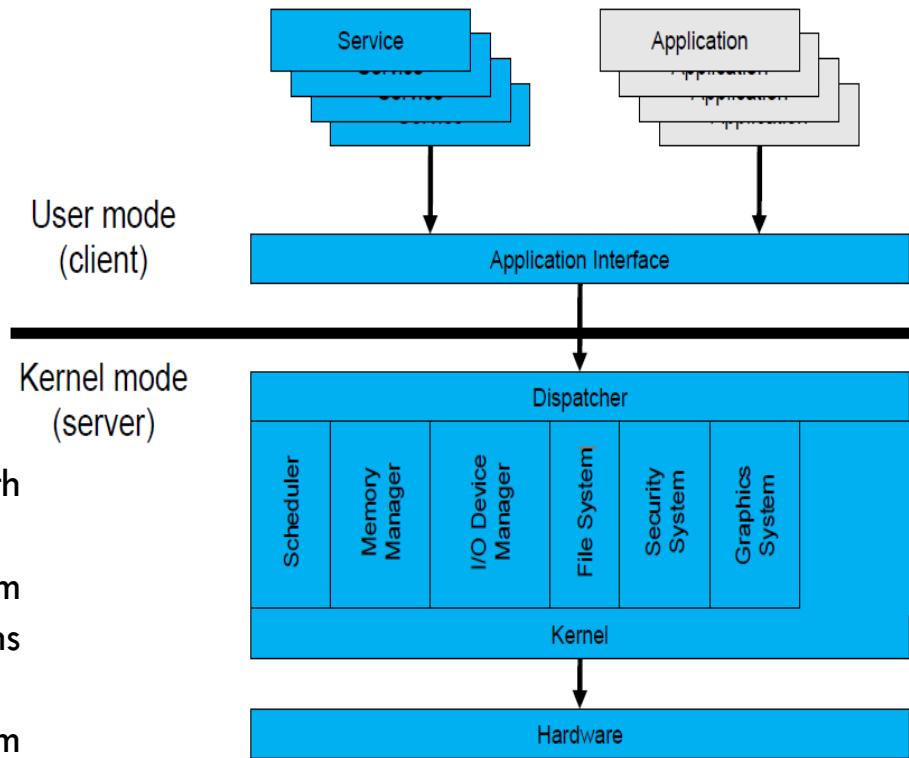


System Call

- **System call** is the programmatic way in which a computer program/user application requests a service from the kernel of the operating system on which it is executed.
- Application program is just a user-process. Due to security reasons, user applications are not given access to privileged resources(the ones controlled by OS).
- When they need to **do any I/O** or have **some more memory** or **spawn a process** or wait for **signal/interrupt**, it requests operating system to facilitate all these. This **request is made through System Call**.
- System calls are also called **software-interrupts**.

Operating System Mode

- ❖ The *User Mode* is concerned with the actual interface between the user and the system.
- ❖ It controls things like running applications and accessing files.



- ❖ The *Kernel Mode* is concerned with everything running in the background.
- ❖ It controls things like accessing system resources, controlling hardware functions and processing program instructions.
- ❖ *System calls* are used to change mode from User to Kernel.

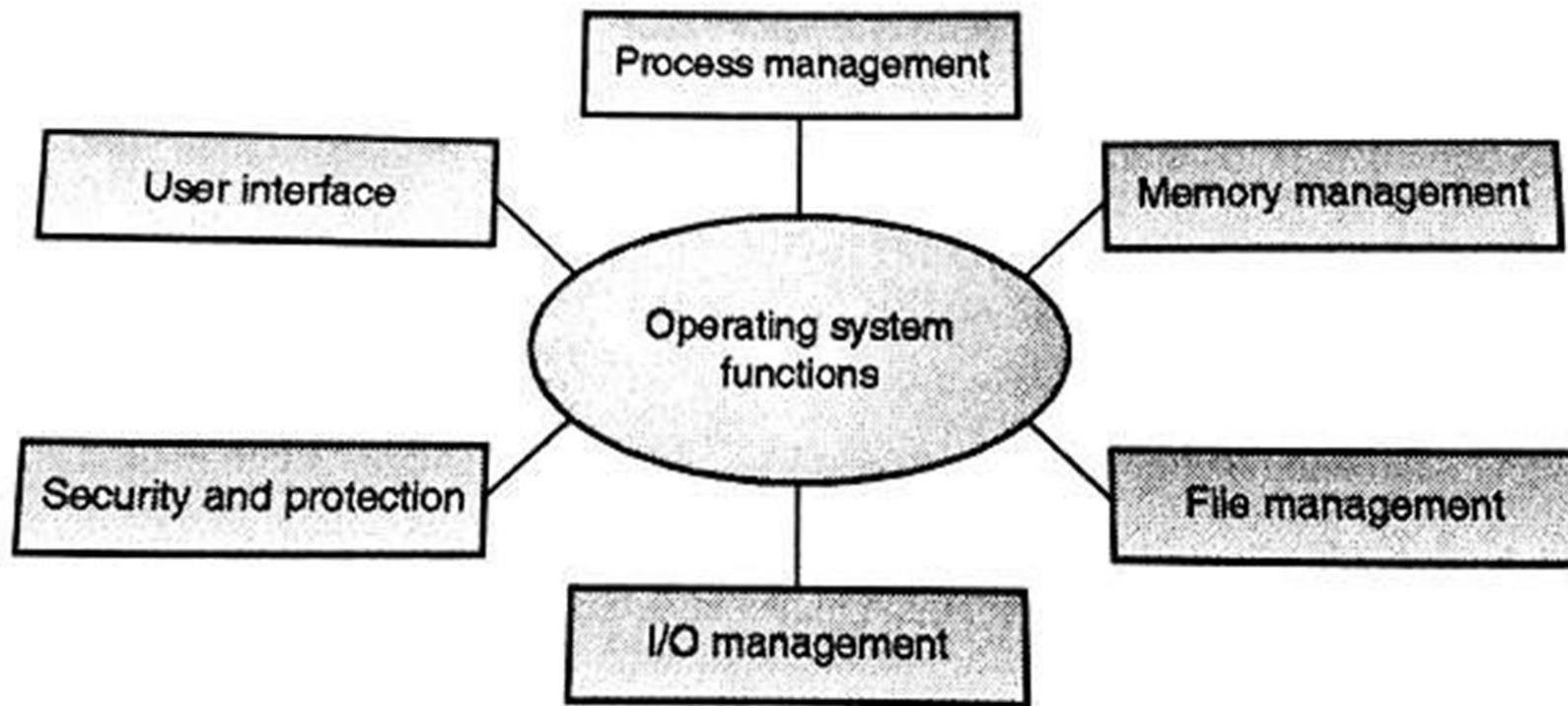
Kernel

- Kernel is a software code that reside in central core of OS. It has complete control over system.
- When operation system boots, kernel is first part of OS to load in main memory.
- Kernel remains in main memory for entire duration of computer session. The kernel code is usually loaded in to protected area of memory.
- Kernel performs it's task like executing processes and handling interrupts in kernel space.
- User performs it's task in user area of memory.
- This memory separation is made in order to prevent user data and kernel data from interfering with each other.
- Kernel does not interact directly with user, but it interacts using SHELL and other programs and hardware.

Kernel cont...

- Kernel includes:-
 1. **Scheduler**: It allocates the Kernel's processing time to various processes.
 2. **Supervisor**: It grants permission to use computer system resources to each process.
 3. **Interrupt handler** : It handles all requests from the various hardware devices which compete for kernel services.
 4. **Memory manager** : allocates space in memory for all users of kernel service.
- kernel provides services for process management, file management, I/O management, memory management.
- System calls are used to provide these type of services.

Function of OS



1. Process Management

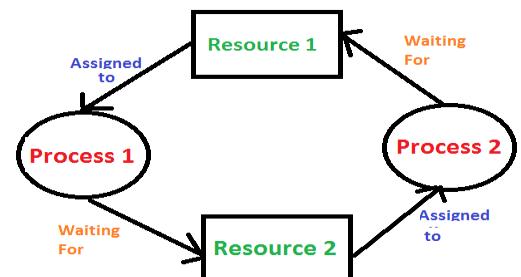
- A process is a program in execution.
- A process needs certain resources, including CPU time, memory, files, and I/O devices to accomplish its task.
- Simultaneous execution leads to multiple processes. Hence creation, execution and termination of a process are the most basic functionality of an OS
- If processes are **dependent**, than they may try to share same resources. thus task of **process synchronization** comes to the picture.
- If processes are **independent**, than a due care needs to be taken to avoid their overlapping in memory area.
- Based on priority, it is important to allow more important processes to execute first than others.

2. Memory management

- Memory is a large array of words or bytes, each with its own address.
- It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a **volatile** storage device. When the computer made turn off everything stored in RAM will be erased automatically.
- In addition to the physical RAM installed in your computer, most modern operating systems allow your computer to use a *virtual memory system*. *Virtual memory allows your computer to use part of a permanent storage device (such as a hard disk) as extra memory.*
- The operating system is responsible for the following activities in connections with memory management:
 - Keep track of which parts of memory are currently being used and by whom.
 - Decide which processes to load when memory space becomes available.
 - Allocate and de-allocate memory space as needed.

4. Device Management or I/O Management

- **Device controllers** are components on the motherboard (or on expansion cards) that act as an interface between the CPU and the actual device.
- **Device drivers**, which are the operating system software components that interact with the devices controllers.
- A special device (inside CPU) called the **Interrupt Controller** handles the task of receiving interrupt requests and prioritizes them to be forwarded to the processor.
- **Deadlocks** can occur when two (or more) processes have control of different I/O resources that are needed by the other processes, and they are unwilling to give up control of the device.
- It performs the following activities for device management.
 - Keeps tracks of all devices connected to system.
 - Designates a program responsible for every device known as Input/output controller.
 - Decides which process gets access to a certain device and for how long.
 - Allocates devices in an effective and efficient way.
 - Deallocates devices when they are no longer required.



3. File Management

- A file is a collection of related information defined by its creator.
- *File systems provide the conventions for the encoding, storage and management of data on a storage device such as a hard disk.*
 - FAT12 (floppy disks)
 - FAT16 (DOS and older versions of Windows)
 - FAT32 (older versions of Windows)
 - NTFS (newer versions of Windows)
 - EXT3 (Unix/Linux)
 - HFS+ (Mac OS X)
- The operating system is responsible for the following activities in connections with file management:
 - ◆ File creation and deletion.
 - ◆ Directory creation and deletion.
 - ◆ Support of primitives for manipulating files and directories.
 - ◆ Mapping files onto secondary storage.
 - ◆ File backup on stable (nonvolatile) storage media.

5. Security & Protection

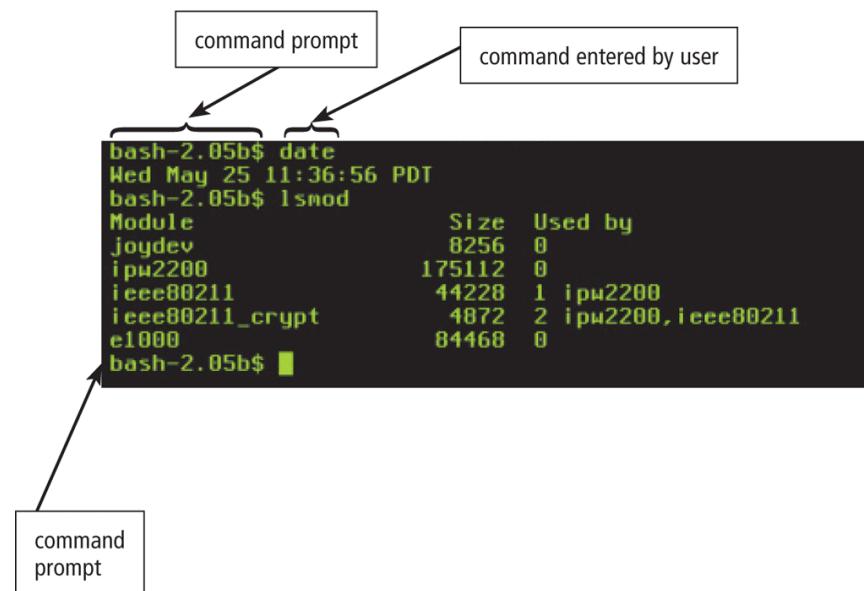
- The operating system uses password protection to protect user data and similar other techniques.
- It also prevents unauthorized access to programs and user data by assigning access right permission to files and directories.
- The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other.

6. User Interface Mechanism

- A **user interface (UI)** controls how you enter data and instructions and how information is displayed on the screen
- There are two types of user interfaces
 1. Command Line Interface
 2. Graphical user Interface

1. Command-line interface

- In a command-line interface, a user types commands represented by short keywords or abbreviations or presses special keys on the keyboard to enter data and instructions



2. Graphical User Interface

- With a graphical user interface (GUI), you interact with menus and visual images



History of Operating System

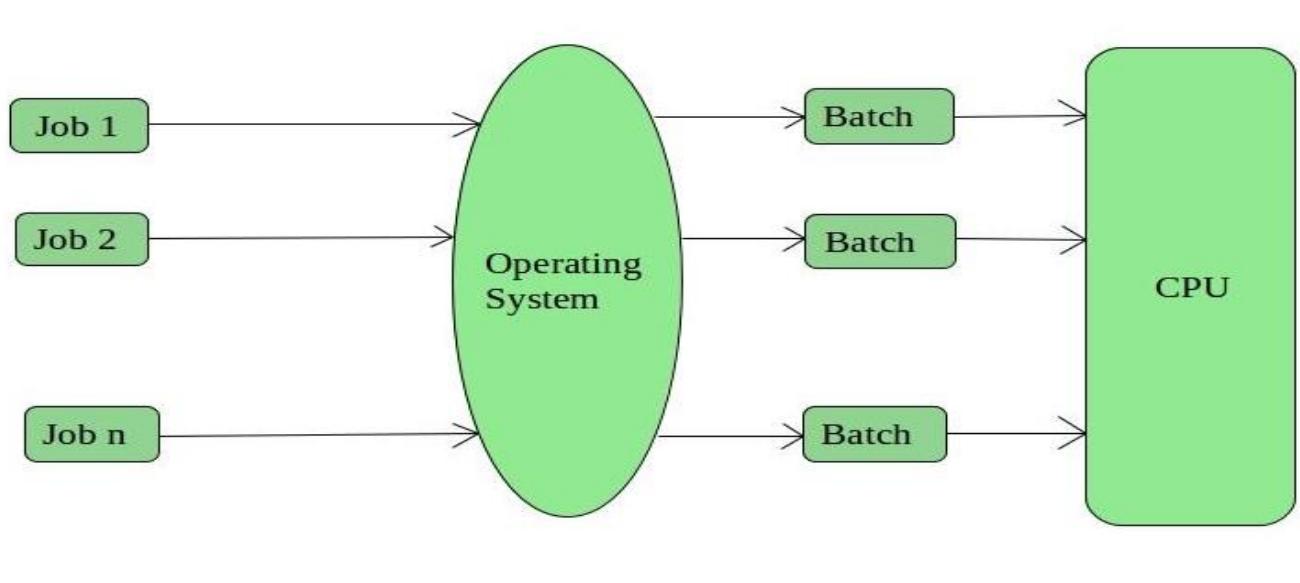
- ❖ **The First Generation (1940's to early 1950's)**
 - No Operating System
 - All programming was done in absolute machine language, often by wiring up plug-boards to control the machine's basic functions.
- ❖ **The Second Generation (1955-1965)**
 - First operating system was introduced in the early 1950's. It was called GMOS
 - Created by General Motors for IBM's machine the 701.
 - Single-stream batch processing systems
- ❖ **The Third Generation (1965-1980)**
 - Introduction of multiprogramming
 - Development of Minicomputer
- ❖ **The Fourth Generation (1980-Present Day)**
 - Development of PCs
 - Birth of Windows/MaC OS

Types of Operating Systems

1. Batch Operating System
2. Multiprogramming Operating System
3. Time-Sharing OS
4. Multiprocessing OS
5. Distributed OS
6. Network OS
7. Real Time OS
8. Embedded OS

1. Batch Operating System

- The users of this type of operating system does not interact with the computer directly.
- Each user prepares his job on an off-line device like punch cards and submits it to the computer operator
- There is an operator which takes similar jobs having the same requirement and group them into batches.



1. Batch Operating System

cont..

Advantages of Batch Operating System:

- Processors of the batch systems know how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time for the batch system is very less
- It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System:

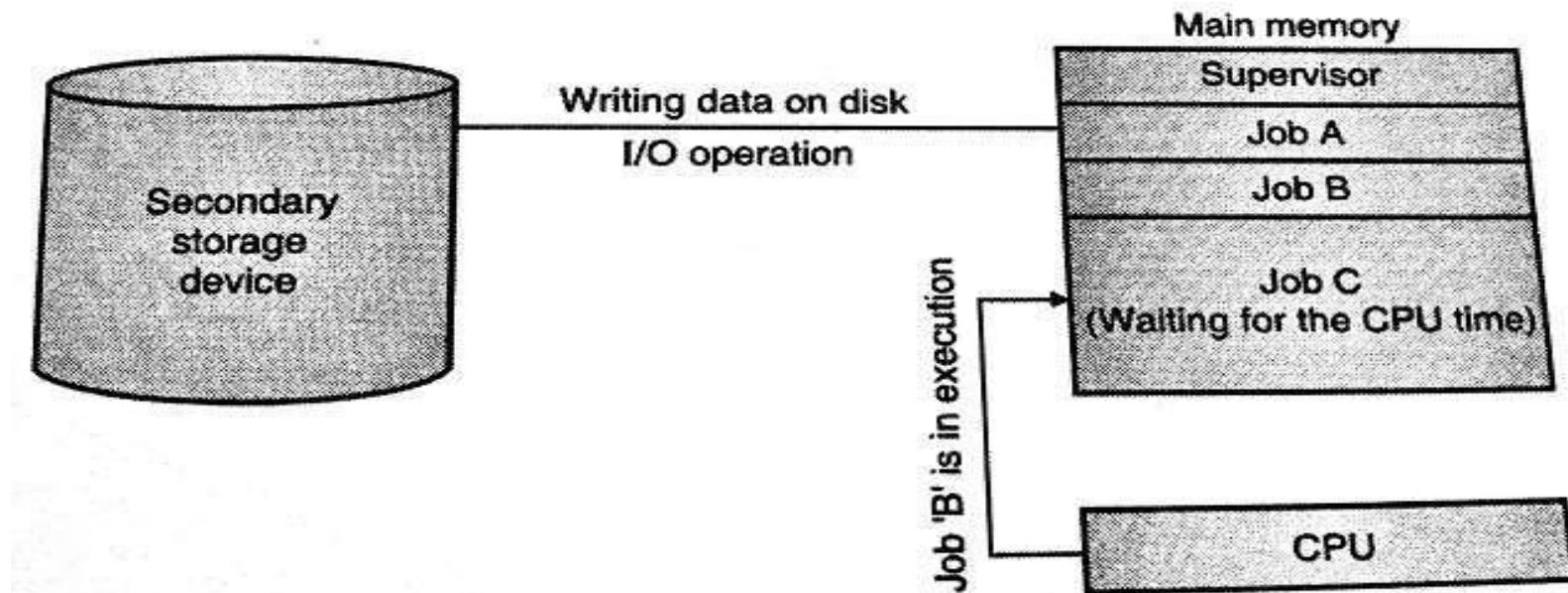
- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometimes costly
- The other jobs will have to wait for an unknown time if any job fails

Examples of Batch based Operating System:

IBM's MVS

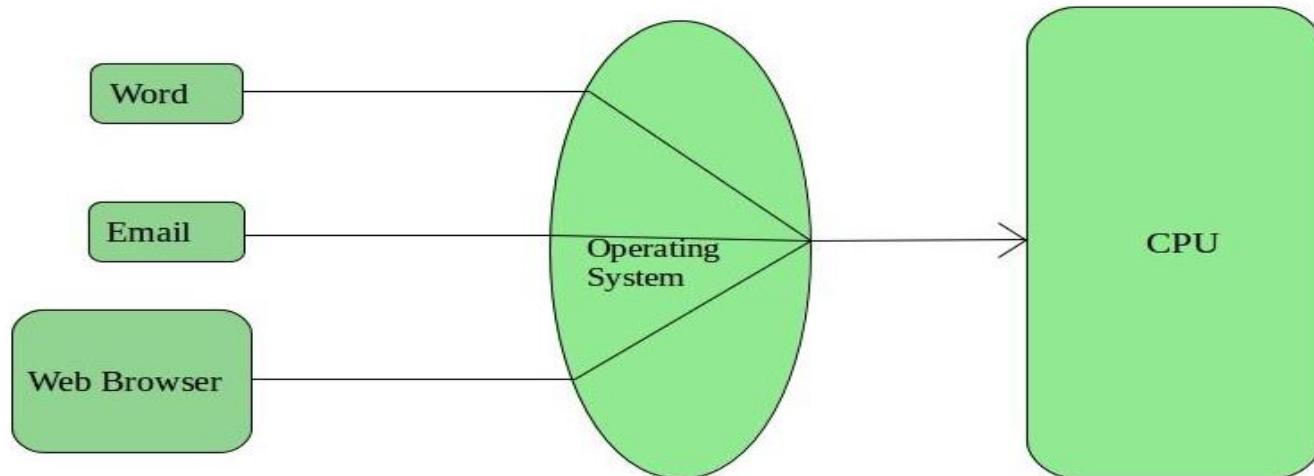
2. Multiprogramming Operating System:

- This type of OS is used to execute more than one jobs simultaneously by a single processor.
- It increases CPU utilization by organizing jobs so that the CPU always has one job to execute.
- Multiprogramming operating systems use the mechanism of job scheduling and CPU scheduling.



3. Time-Sharing Operating Systems

- Each task is given some time to execute so that all the tasks work smoothly.
- These systems are also known as **Multi-tasking Systems**.
- The task can be from a single user or different users also.
- The time that each task gets to execute is called quantum.
- After this time interval is over OS switches over to the next task.



3. Time-Sharing Operating Systems cont..

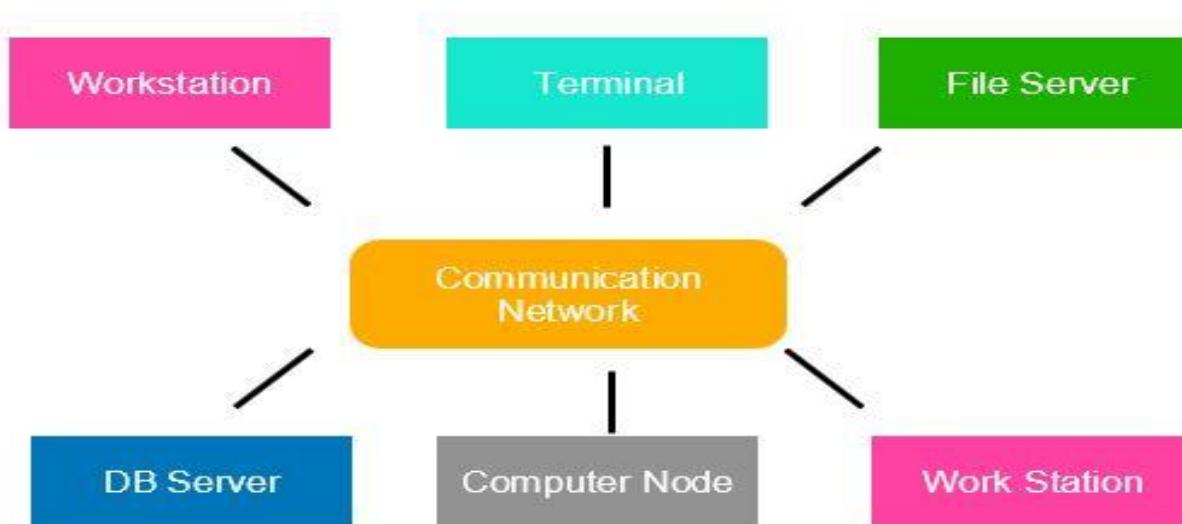
- **Advantages of Time-Sharing OS:**
 - Each task gets an equal opportunity
 - Fewer chances of duplication of software
 - CPU idle time can be reduced
- **Disadvantages of Time-Sharing OS:**
 - Reliability problem
 - One must have to take care of the security and integrity of user programs and data
 - Data communication problem
- **Examples of Time-Sharing Oss**
 - Multics, Unix, etc.

4. Multiprocessor operating systems

- Multiprocessor operating systems are also known as **parallel OS** or **tightly coupled OS**.
- Such operating systems have more than one processor in close communication that sharing the computer bus, the clock and sometimes memory and peripheral devices.
- It executes multiple jobs at the same time and makes the processing faster.
- It supports large physical address space and larger virtual address space.
- If one processor fails then other processor should retrieve the interrupted process state so execution of process can continue.
- Inter-processes communication mechanism is provided and implemented in hardware.

5. Distributed Operating System

- Various autonomous interconnected computers communicate with each other using a shared communication network.
- Independent systems possess their own memory unit and CPU.
- These are referred to as **loosely coupled systems**.
- Examples:- Locus, DYSEAC



6. Network Operating System

- These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions.
- These types of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network.
- The “other” computers are called client computers, and each computer that connects to a network server must be running client software designed to request a specific service.
- popularly known as **tightly coupled systems**.

6. Network Operating System

Advantages of Network Operating System:

- Highly stable centralized servers
- Security concerns are handled through servers
- New technologies and hardware up-gradation are easily integrated into the system
- Server access is possible remotely from different locations and types of systems

Disadvantages of Network Operating System:

- Servers are costly
- User has to depend on a central location for most operations
- Maintenance and updates are required regularly

Examples of Network Operating System are:

Microsoft Windows Server 2003/2008/2012, UNIX, Linux, Mac OS X, Novell NetWare, and BSD, etc.

7. Real-Time Operating System

- These types of OSs serve real-time systems.
- The time interval required to process and respond to inputs is very small.
- This time interval is called **response time**.
- **Real-time systems** are used when there are time requirements that are very strict like
 - missile systems,
 - air traffic control systems,
 - robots, etc.

8. Embedded Operating System

- An embedded operating system is one that is built into the circuitry of an electronic device.
- Embedded operating systems are now found in automobiles, bar-code scanners, cell phones, medical equipment, and personal digital assistants.
- The most popular embedded operating systems for consumer products, such as PDAs, include the following:
 - Windows XP Embedded
 - Windows CE .NET:- it supports wireless communications, multimedia and Web browsing. It also allows for the use of smaller versions of Microsoft Word, Excel, and Outlook.
 - Palm OS:- It is the standard operating system for Palm-brand PDAs as well as other proprietary handheld devices.
 - Symbian:- OS found in “smart” cell phones from Nokia and Sony Ericsson

Popular Operating Systems



Windows



Linux



Ubuntu



Mac OS X
iOS



Android

File Systems

File Systems

- Many important applications need to store more information than have in virtual address space of a process
- The information must survive the termination of the process using it.
- Multiple processes must be able to access the information concurrently.

File Systems

- Disks are used to store files
- Information is stored in blocks on the disks
- Can read and write blocks

File Systems

- Use file system as an abstraction to deal with accessing the information kept in blocks on a disk
- Files are created by a process
- Thousands of them on a disk
- Managed by the OS

File Systems

- OS structures them, names them, protects them
- Two ways of looking at file system
 - User - how do we name a file, protect it, organize the files
 - Implementation - how are they organized on a disk
- Start with user, then go to implementer

File Systems

The user point of view

- Naming
- Structure
- Directories

Naming

- ❖ FAT (16 and 32) were used in first Windows systems
- ❖ NTFS was used in the UNIX systems
- ❖ All OS's use suffix as part of name
- ❖ UNIX does not always enforce a meaning for the suffixes
- ❖ DOS does enforce a meaning

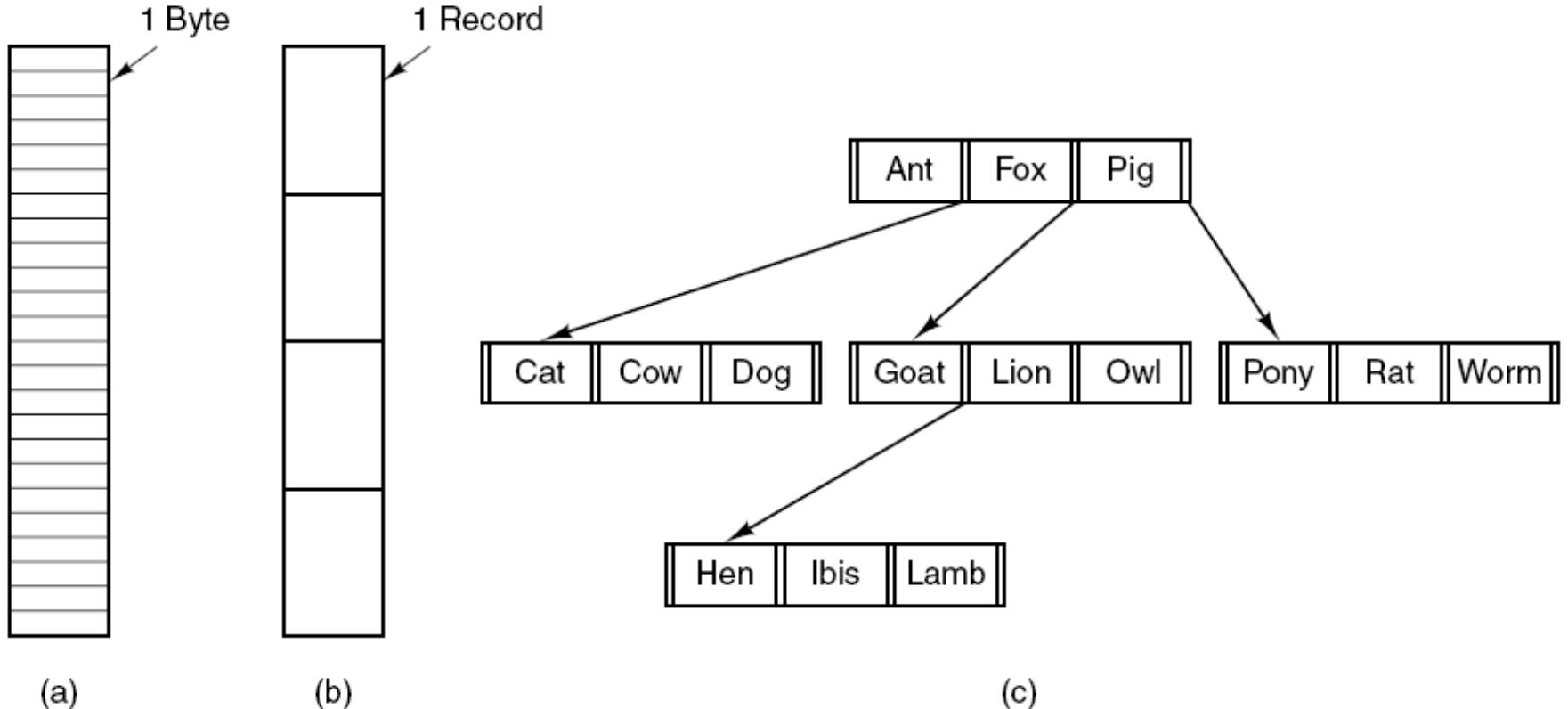
Suffix Examples

Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

File Structure

- ❖ Byte sequences
 - ❖ Maximum flexibility - can put anything in
 - ❖ Unix and Windows use this approach
- ❖ Fixed length records (card images in the old days)
- ❖ Tree of records- uses key field to find records in the tree

File Structure



Three kinds of files. (a) Byte sequence.
(b) Record sequence. (c) Tree.

File Types

- Regular- contains user information
 ASCII (Data files) or Binary (Executable)
- Directories
- Character special files - model serial (e.g. printers) I/O devices
- Block special files - model disks

File Access

- Sequential access- read from the beginning, can't skip around
 - Corresponds to magnetic tape
- Random access- start where you want to start
 - Came into play with disks
 - Necessary for many applications, e.g. airline reservation system

File Attributes (hypothetical OS)

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

System Calls for files

- Create -with no data, sets some attributes
- Delete-to free disk space
- Open- after create, gets attributes and disk addresses into main memory
- Close- frees table space used by attributes and addresses
- Read-usually from current pointer position. Need to specify buffer into which data is placed
- Write-usually to current position

System Calls for files

- Append- at the end of the file
- Seek-puts file pointer at specific place in file. Read or write from that position on
- Get Attributes-e.g. make needs most recent modification times to arrange for group compilation
- Set Attributes-e.g. protection attributes
- Rename

How can system calls be used?

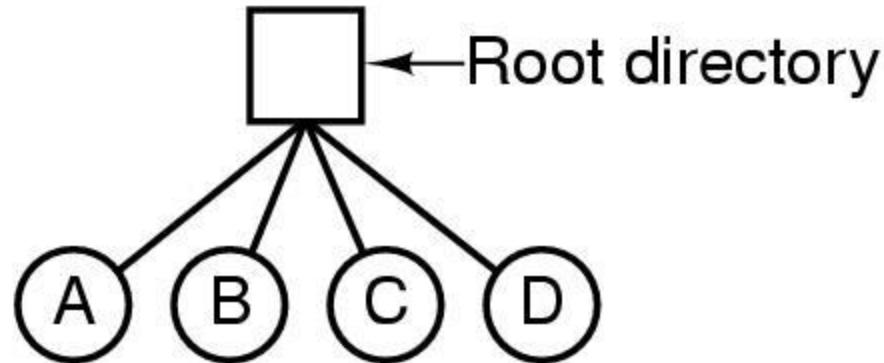
An example-copyfile abc xyz

- Copies file abc to xyz
- Uses system calls (read, write)
- Reads and writes in 4K chunks
- Read (system call) into a buffer
- Write (system call) from buffer to output file

Directories

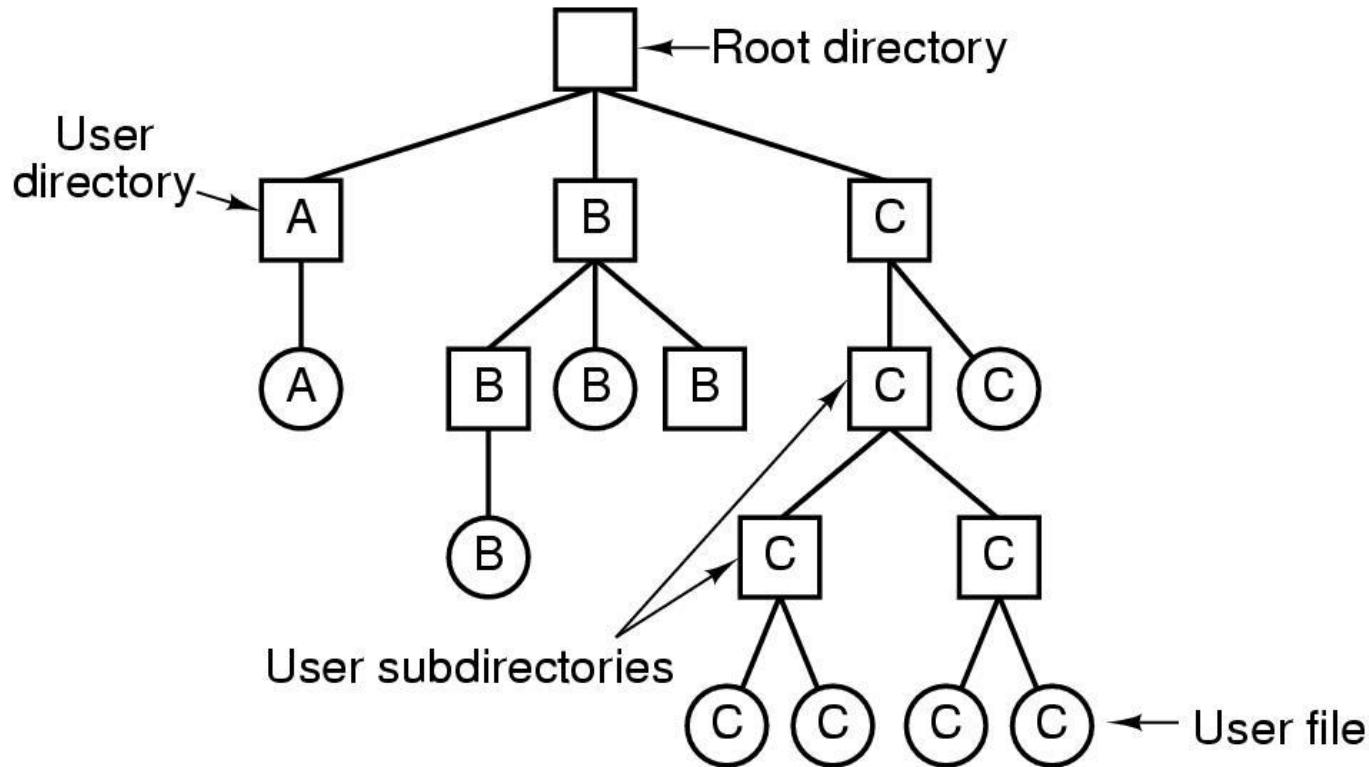
- Files which are used to organize a collection of files
- Also called folders in Windows

Single Level Directory Systems



A single-level directory system containing four files.

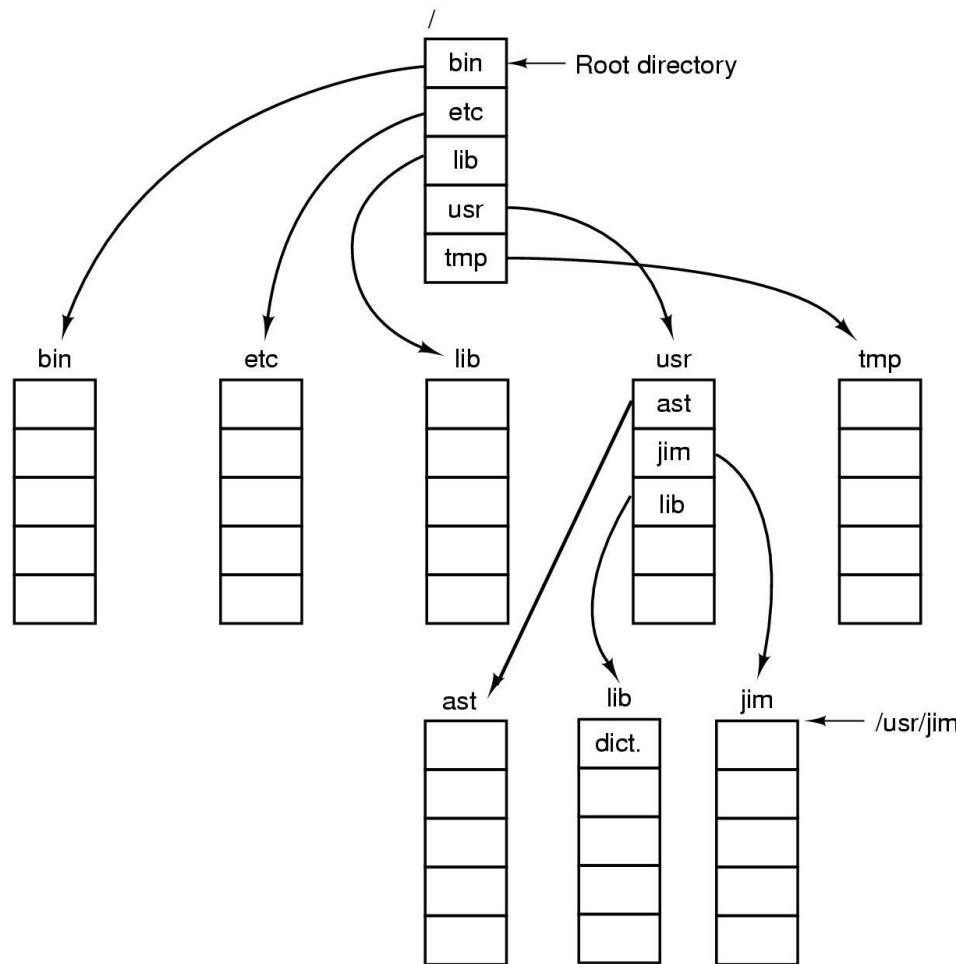
Hierarchical Directory Systems



Path names

- Absolute /usr/carl/cs310/midterm/answers
- Relative cs310/midterm/answers
 - . Refers to current (working) directory
 - .. Refers to parent of current directory

Path Names



A UNIX directory tree.

Directory Operations

- Create
- Delete
- Opendir
- Closedir
- Rename, ...

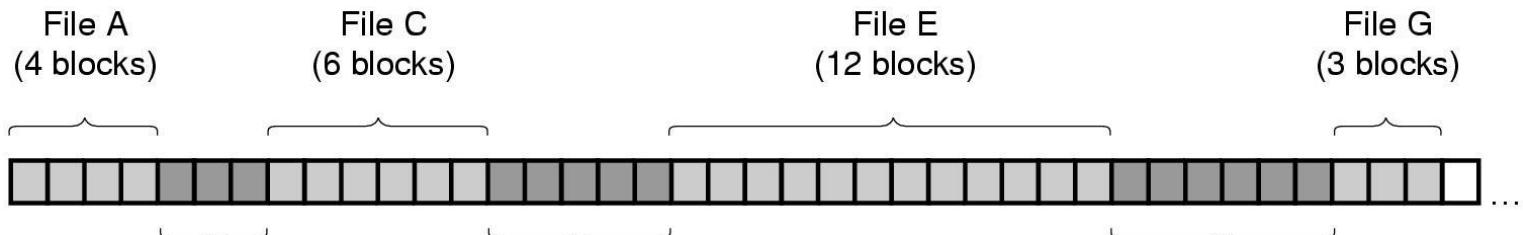
File Implementation

- Files stored on disks.
- File blocks are placed on disk blocks

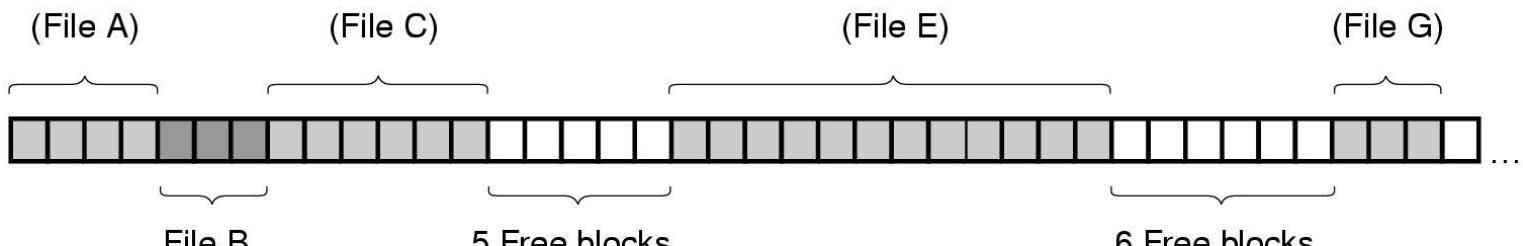
Allocating Blocks to files

- Most important implementation issue
- Methods
 - Contiguous allocation
 - Linked list allocation
 - Linked list using table
 - I-nodes

Contiguous Allocation



(a)



(b)

(a) Contiguous allocation of disk space for 7 files.

(b) The state of the disk after files D and F have been removed.

Contiguous Allocation

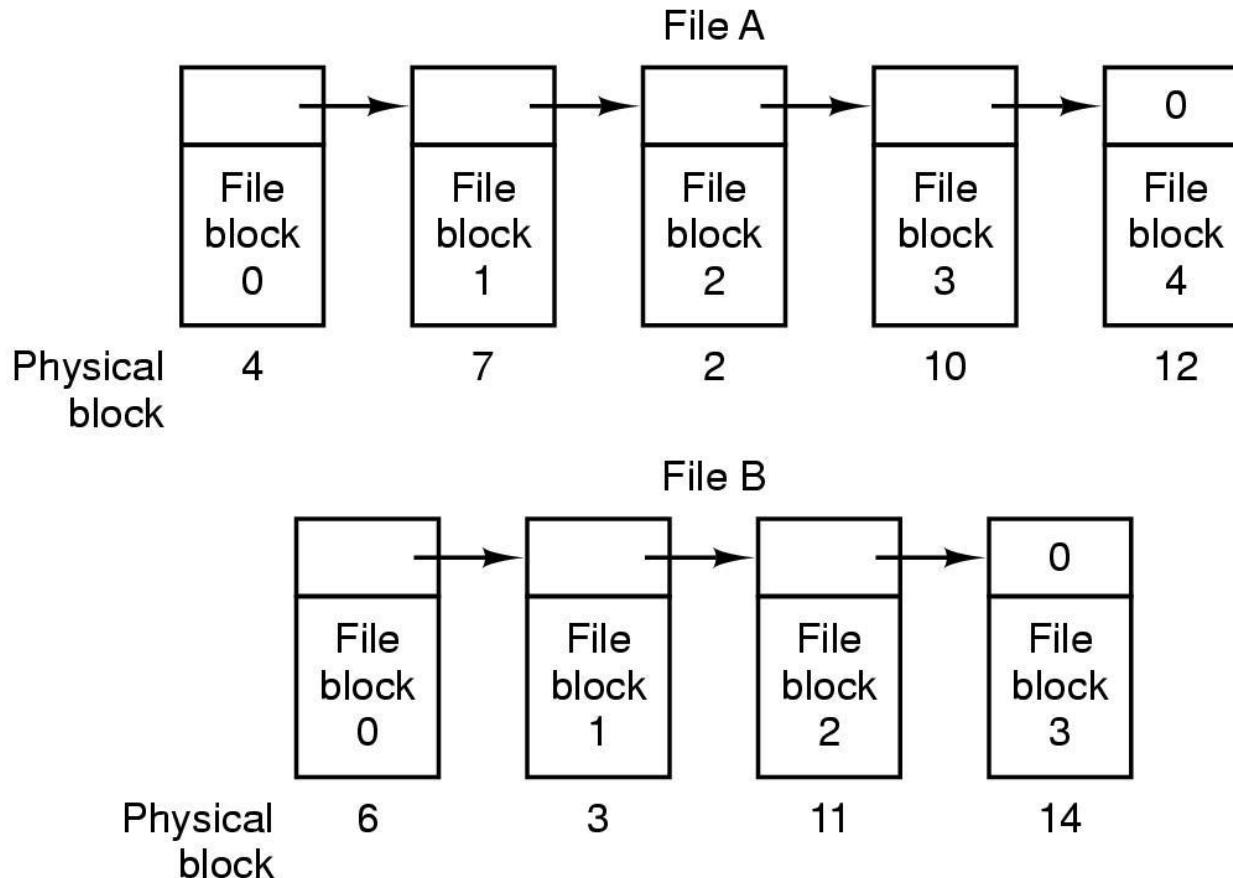
The good

- Easy to implement
- Read performance is great. Only need one seek to locate the first block in the file. The rest is easy.

The bad-disk becomes fragmented over time

- CD-ROM's use contiguous allocation because the file size is known in advance
- DVD's are stored in a few consecutive 1 GB files because standard for DVD only allows a 1 GB file max

Linked List Allocation



Storing a file as a linked list of disk blocks.

Linked Lists

The good

- Gets rid of fragmentation

The bad

- Random access is slow. Need to chase pointers to get to a block

Linked lists using a table in memory

- Put pointers in table in memory
- File Allocation Table (FAT)
- Windows

The Solution-Linked List Allocation Using a Table in Memory

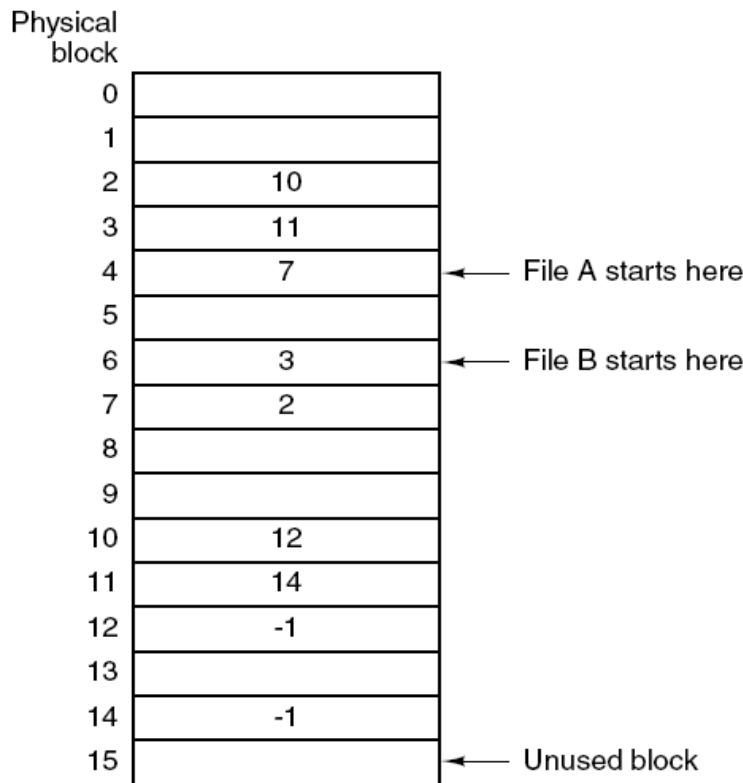


Figure 4-12. Linked list allocation using a file allocation table in main memory.

Linked lists using a table in memory

- The bad-table becomes really big
- E.g 200 GB disk with 1 KB blocks needs a 600 MB table
- Growth of the table size is linear with the growth of the disk size

Implementing Directories

- Directory specifies block addresses by providing
 - Address of first block (contiguous)
 - Number of first block (linked)

File System Management and Optimization- the dirty work

- Disk space management
- File System Backups
- File System Consistency
- File System Performance

Overview of Computer Workshop

Unit-3, Lecture – 1

By

Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

What is an Operation System?

- An *operating system* is the low-level software that schedules tasks, allocates storage, and handles the interfaces to peripheral hardware, such as printers, disk drives, the screen, keyboard, and mouse.
- An operating system has two main parts: the *kernel* and the *system programs*.
- The *kernel* allocates machine resources—including *memory* and *disk space*, and *CPU cycles* to all other programs that run on the computer.
- The system programs include device drivers, libraries, utility programs, shells (command interpreters), configuration scripts and files etc.
- Common contemporary OSs include *Microsoft Windows*, *Mac OS X*, and *Linux*.

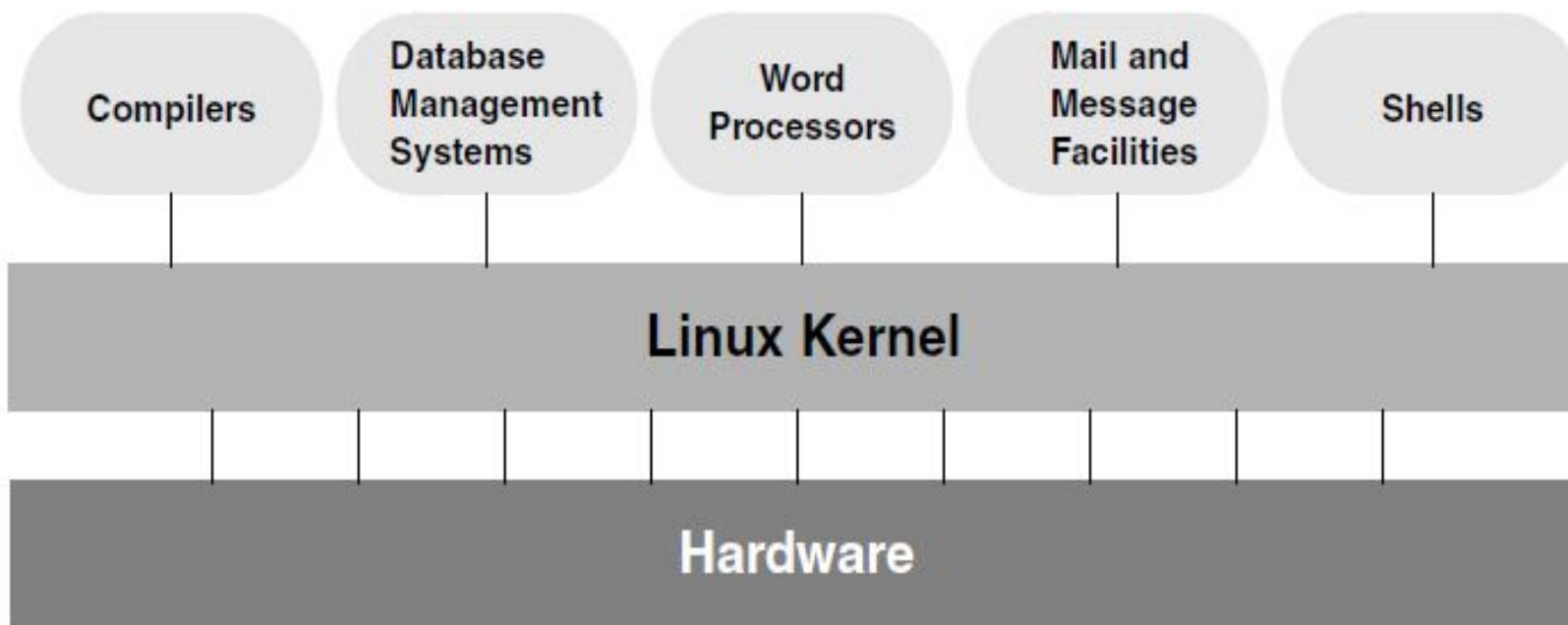
Linux

- Linux is a member of the large family of Unix-like OS.
- The Linux kernel was developed by Finnish undergraduate student Linus Torvalds, who used the Internet to make the source code immediately available to others for free.
- Torvalds released Linux version 0.01 in September 1991.
- Programmers around the world were quick to extend the kernel and develop other tools, adding functionality to match that already found in both BSD UNIX and System V UNIX (SVR4) as well as new functionality.
- The name Linux is a combination of Linus and UNIX.

Features of Linux

- Linux is an **open-source** operating system: its source code is open and available to anyone to study.
- It is a **control program** for computers like any other OS.
- It has family of utility programs and a set of tools that allow users to connect and use these utilities to build systems and applications.
- Linux has a kernel programming interface.
- Linux can support many users simultaneously i.e. it is **multi-user OS**.
- Linux is a fully protected **multitasking** operating system, allowing each user to run more than one job at a time.
- Linux is portable.
- Linux is predominantly known for its use in servers.
- There are several Linux Distributions, such as: Ubuntu Linux, Red Hat Enterprise Linux, Linux Mint, Debian, Fedora etc.

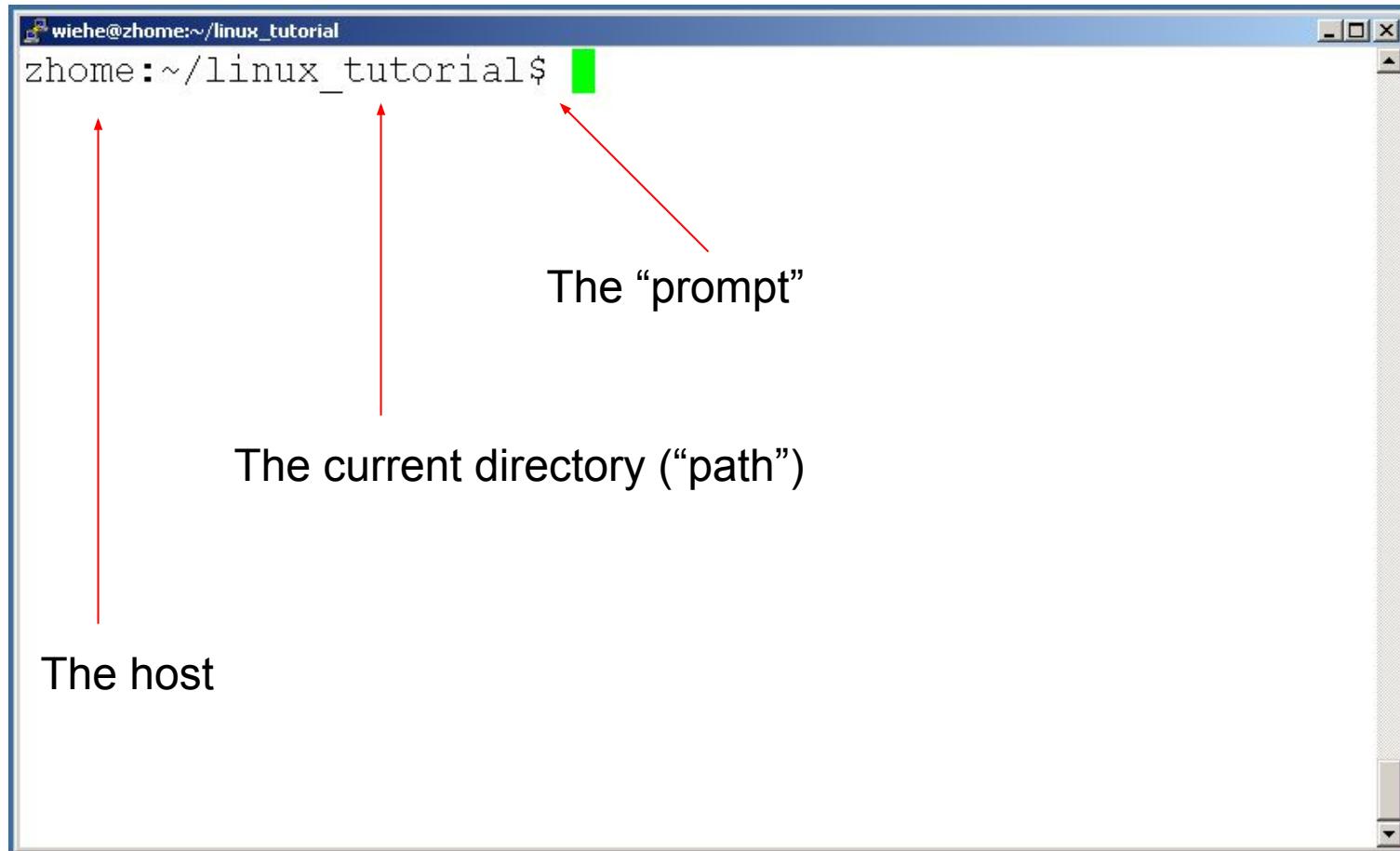
A layered view of Linux OS



User interface in Linux

- There are two different ways in which the user can interact with the Linux OS
 - Through a **graphical user interface (GUI)** in which the user uses a mouse to manipulate windows
 - Through the **command line interface (CLI)**, in which the user types commands at a prompt.
- **CLI**: It is a non-graphical, text-based interface to the computer system, where the user types in a command and the computer then successfully executes it.
- The Terminal is the platform or the IDE that provides the command line interface (CLI) environment to the user.
- The CLI terminal accepts the commands that the user types and passes to a shell or command interpreter.
- The shell interprets the command and calls the program you want.
- If the output is produced by the specific command, then this text is displayed in the terminal.

A typical Linux terminal



Basic commands in Linux

- Commands are case sensitive. Lower case alphabets are used in the commands without using any unnecessary characters.
- **date** : It is used to display date, time, time zone, and more.

Syntax: **date**

- **cal** : It is used to display the current month's calendar with the current date highlighted.

Syntax: **cal**

- **who**: It gives the information about the users logged on to the system. The first column displays the user name, second column displays the system's name used for connection and the rest displays the details when the users logged in.

Syntax: **who**

- **pwd** : It is used to display the location of the current working directory.

Syntax: `pwd`

- **man** :`man` stands for manual which is a reference book of a Linux operating system. It is similar to HELP file found in popular software.

Syntax: `man command_name`

- **clear**: This command clears all the clutter on the terminal and gives you a clean window to work on, just like when you launch the terminal.

Syntax: `clear`

- **echo**: It is used to display a line of string/text that is passed as the argument.

Syntax: `echo [options] [string]`

Using '`>>`' with 'echo' command appends a line to a file.

Syntax: `echo [string] >>filename` `echo Hi>>file1`

Directory Related commands

- **mkdir** : It is used to create a directory in your present working directory.

Syntax: **mkdir dir_name** **mkdir dir1**

- **rmdir** : It is used to remove a directory (Ensure that the directory is empty).

Syntax: **rmdir dir_name**

- **cd**: It is used to change the current directory.

Syntax: **cd dir_name**

Options:

- **cd** change directory to the home directory.
- **cd ..** move to the parent directory of current directory one level up from the current directory.

File related commands

- **touch** : It is used to create empty files.

Syntax: `touch file_name` or `touch file_name1 file_name2`

- **cat** : It is used to create a new file, displaying the content of the file or adding new content to the file.

Syntax: `cat > file_name` (for creating new file named `file_name`)

Options:

- `cat filename` (display content of the file)
- `cat filename1 filename2 filename3` (display content of multiple files)
- `cat filename1 > filename2` (Create a new file `filename2` and redirect the content of `filename1` to `filename2`. If `filename2` is an existing file then its content is overwritten by content of `filename1`)
- `cat filename1 >> filename2` (Append the content of `filename1` to the end of content of `filename2`)

- **tac**: It is the reverse of the cat command. It displays the contents of the file in reverse order.

Syntax: **tac filename**

- **rm** : It is used to delete a file.

Syntax: **rm filename**

- **mv** :It is used to rename a file or to move a file from one directory to other directory.

Syntax: **mv oldfilename newfilename**

- **cp**: It is used to copy the content of one file to another file.

Syntax: **cp source_filename destination_filename**

- **ls**: It makes a list of file names or directory present in the current working directory.

Syntax: **ls**

Option:

ls -l (Long listing of files)

ls -a (lists all files including hidden files starting with '.')

ls -i (lists files along with the inode number)

ls -s (lists files along with their size)

ls -S (lists by sorting with file size)

Overview of Computer Workshop

Unit-3, Lecture – 2

By

Dr. Priyambada Subudhi

Assistant Professor

IIIT Sri City

Some more commands related to Files

- **wc**: It is used to display the total number of lines and characters in one or more files.
- By default it displays four-columnar output. First column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which are given as argument.

Syntax: **wc filename**

Options:

wc -w filename	number of words present in a file
wc -l filename	number of lines present in a file
wc -c filename	count of bytes present in a file

- **sort**: It sorts its input into alphabetical order line by line.

Syntax: `sort filename`

Options:

`sort -r` Reverse normal order

`sort -n` Sort in numeric order

`sort -nr` Sort in reverse numeric order

- **head** : It prints the first 10 lines of a file.

Syntax: `head filename`

Options:

`head -n filename` (print first n lines of the file)

- **tail**: It prints the last 10 lines of a file.

Syntax: `tail filename`

Options:

`tail -n filename` (print last n lines of the file)

`tail +n filename` (starts printing from the (n+1)th line)

- **cmp**: It prints the first place where two files differ. cmp is used when one wants to be sure that two files really have the same contents. It is fast and it works on any kind of file.

Syntax: `cmp filename1 filename2`

- **diff** : It reports on all lines that are changed, added and deleted. It is used when the files are expected to be somewhat different and one wants to know exactly which lines differ. diff works only on files of text.

Syntax: `diff filename1 filename2`

- **grep**: It prints the line from the file matching with specified pattern.

Syntax: `grep pattern filename`

Option:

`grep -v pattern filename` (prints lines from the file not matching with the pattern)

Pipe

- A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux and other Unix-like operating systems to send the output of one command/program/process to another command/program/process for further processing.
- It is denoted by a vertical bar (|).
- E.g. The next command line displays the number of files in a directory. The wc (word count) command with the -w (words) option displays the number of words in its standard input or in a file you specify on the command line:

```
$ ls | wc -w
```

Assignment -1 (Additional Questions)

14. Check the output of the following command:

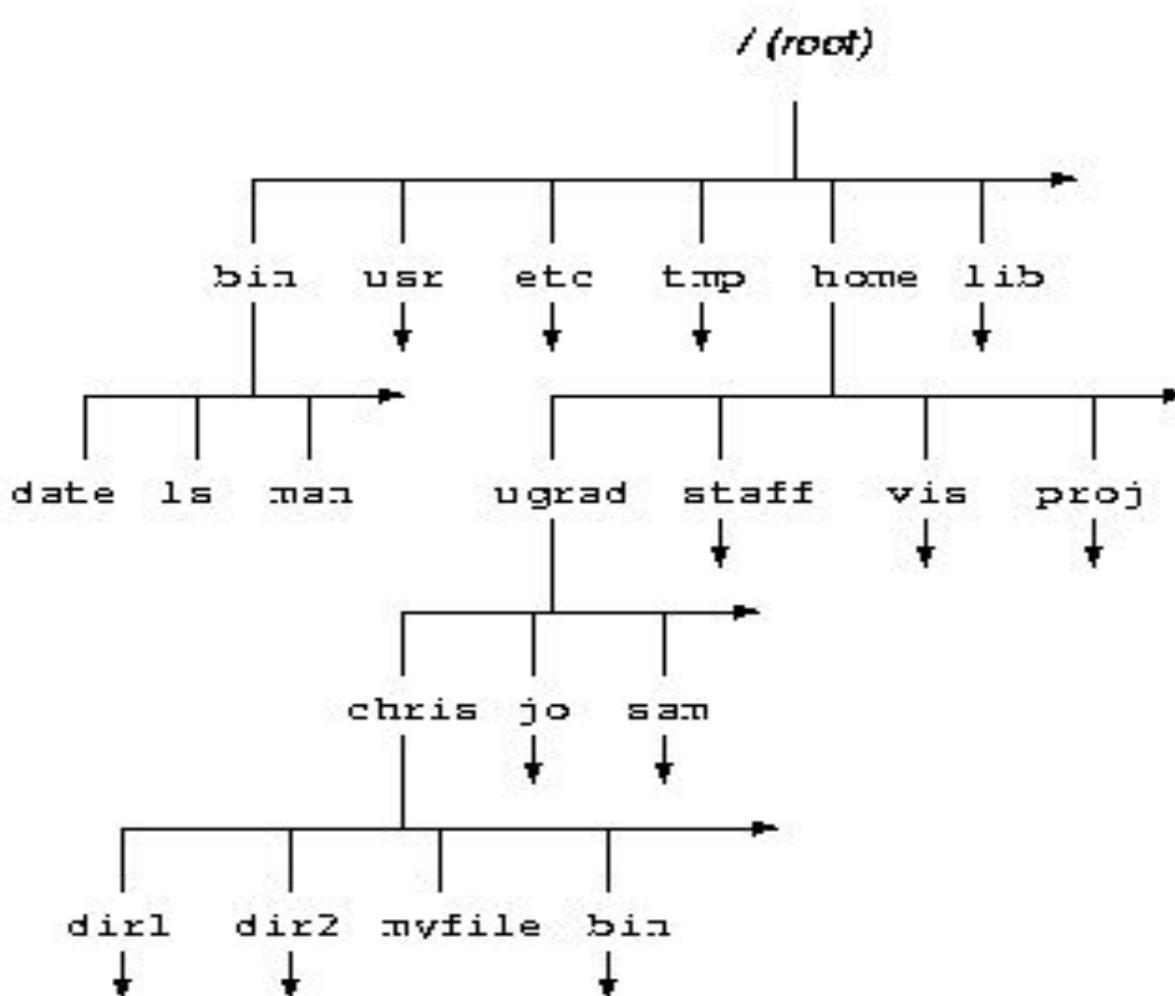
```
cmp reginfo personalinfo  
diff reginfo personalinfo
```

15. Write a command to count the number of files in the current working directory and display that number.

Linux Filesystem

- **Filesystem:** A filesystem provides the way in which files and directories are placed logically for storage and retrieval on a storage disk.
- The general-purpose computer system needs to store data systematically so that we can easily access the files in less time.
- **Linux Filesystem:** Linux has hierarchical filesystem much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.
- Each directory has a name and can hold other files and directories. Directories, in turn, are arranged under other directories and so forth in a tree like organization.
- All other directories in Linux can be accessed from the root directory.

Linux Filesystem structure



Linux File System Features

- **Specifying paths:** Linux does not use the backslash (\) to separate the components; it uses forward slash (/) as an alternative. For example, as in Windows, the data may be stored in C:\ My Documents\ Work, whereas, in Linux, it would be stored in /home/ My Document/ Work.
- **Partition, Directories, and Drives:** Linux does not use drive letters to organize the drive as Windows does. In Linux, we cannot tell whether we are addressing a partition, a network device, or an "ordinary" directory and a Drive.
- **Case Sensitivity:** Linux file system is case sensitive. It distinguishes between lowercase and uppercase file names.
- **File Extensions:** In Linux, a file may have the extension '.txt,' but it is not necessary that a file should have a file extension.
- **Hidden files:** Linux distinguishes between standard files and hidden files, mostly the configuration files are hidden in Linux OS. Usually, we don't need to access or read the hidden files. The hidden files in Linux are represented by a dot (.) before the file name (e.g., .ignore).

Pathnames

- A path is a unique location to a file or a folder in a file system of an OS. A path to a file is a combination of / and alpha-numeric characters.
- **Absolute Path-name:** An *absolute path* is defined as specifying the location of a file or directory from the root directory(/). In other words, we can say that an absolute path is a complete path from start of actual file system from / directory.
- **Relative Path-name:** *Relative path* is defined as the path related to the present working directly(pwd). It starts at your current directory.

Overview of Computer Workshop

Unit-3, Lecture – 3

By

Dr. Priyambada Subudhi

Assistant Professor

IIIT Sri City

File Descriptor and Inode

- Linux makes a clear distinction between a file and the information about a file.
- Each file consists of a sequence of bytes. The file does not include any control information, such as its length or an end-of-file (EOF) delimiter.
- All information needed by the file system to handle a file is included in a data structure called an [inode](#).
- Each file has its own inode, which the file system uses to identify the file.

Information in Inode

- File type
- Number of hard links associated with the file
- File length in bytes
- Device ID (i.e., an identifier of the device containing the file)
- Inode number that identifies the file within the filesystem
- UID of the file owner
- User group ID of the file
- Several timestamps that specify the inode status change time, the last access time, and the last modify time
- Access rights and file mode

Links

- In your Linux file system, a link is a connection between a file name and the actual data on the disk. More than one filename can link to the same data.
- There are two types of links in Linux OS:
- Hard Link:

- They are the low-level links. It links more than one filename with the same Inode and it represents the physical location of a file.
- When hard link is created for a file, it directly points to the Inode of the original file in the disk space, which means no new Inode is created.
- Directories are not created using hard links and they can not cross filesystem boundaries.
- When the source file is removed or moved, then hard links are not affected.
- Command:

In original_filename link_name

■ Soft link or Symbolic Link:

- Soft links are very common. It represents a virtual or abstract location of the file.
- It is just like the shortcuts created in Windows.
- A soft link doesn't contain any information or content of the linked file, instead it has a pointer to the location of the linked file.
- In other words, a new file is created with new Inode, having a pointer to the Inode location of the original file.
- It is used to create link between directories and can cross filesystem boundaries.
- When the source file is removed or moved, then soft links are not updated.

■ Command:

ln -s original_filename link_name

File Access rights

- Linux is a multi-user operating system which can be accessed by many users simultaneously.
- Linux can also be used in mainframes and servers without any modifications.
- But this raises security concerns as an unsolicited or malign user can corrupt, change or remove crucial data.
- For effective security, Linux divides authorization into 2 levels.
 - Ownership
 - Permission

Ownership of files

- Every file and directory on your Linux system is assigned 3 types of owner, given below

- **User:** A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.
- **Group:** A user- group can contain multiple users. All users belonging to a group will have the same access permissions to the file.

Suppose you have a project where a number of people require access to a file, instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.

- **Other:** Any other users who has access to a file. This person has neither created the file nor he belongs to the user group.

Permission or access rights

- Every file and directory in your Linux system has following 3 permission defined for all the 3 owners discussed above.

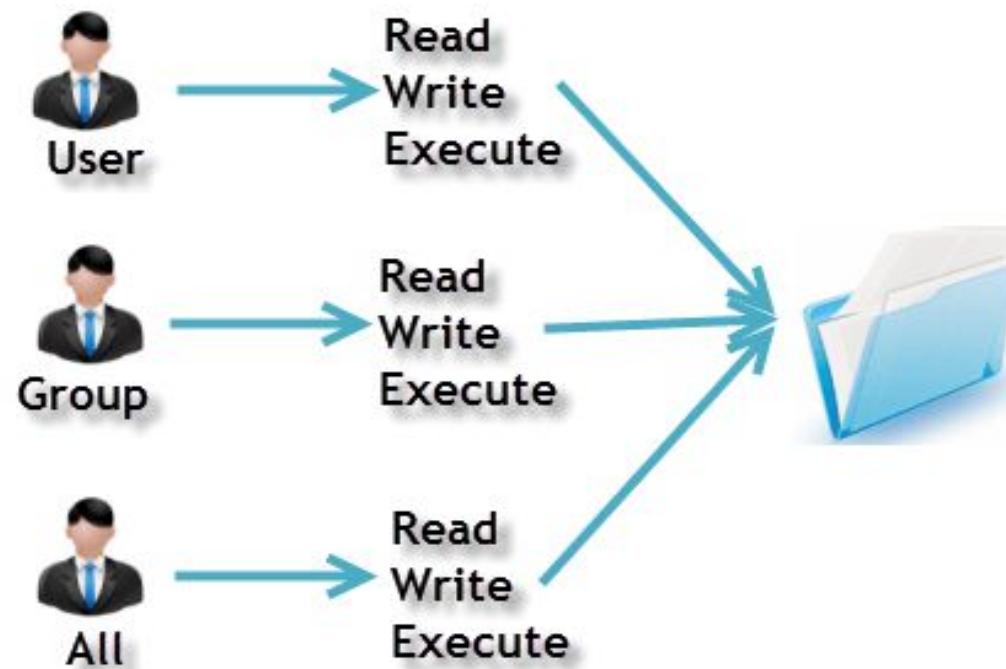
- **Read:** This permission gives you the authority to open and read a file. Read permission on a directory gives you the ability to lists it's content.
- **Write:** The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory.

Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.

- **Execute:** In Windows, an executable program usually has an extension ".exe" and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code (provided read & write permissions are set), but not run it.

File Permissions in Linux

Owners assigned Permission On Every File and Directory



- Let's see file permissions in Linux with examples:

`ls -l` on terminal gives

File type and Access Permissions.

```
home@VirtualBox: ~
home@VirtualBox:~$ ls -l
-rwx-rw-r-- 1 home home 0 2012-08-30 19:06 My File
```

- Here, the first '-' implies that we have selected a file

- rwx-rw-r--
indicates
file

- Else, if it were a directory, `d` would have been shown.

d represents directory

```
drwxr-xr-x 2 ubuntu ubuntu 80 Sep 6 07:27 Desktop
```

■ The first part of the code is 'rw-'. This suggests that the owner 'Home' can:

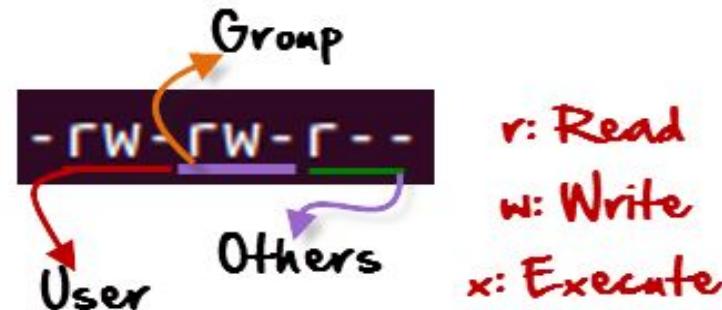
- Read the file
- Write or edit the file
- He cannot execute the file since the execute bit is set to '-'.

■ The second part is 'rw-'. It for the user group 'Home' and group-members can:

- Read the file
- Write or edit the file

■ The third part is for the world which means any user. It says 'r--'. This means the user can only:

- Read the file



Command "chmod"

- It is used to change the access mode of a file. chmod stands for 'change mode'.
- Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the other users.

Syntax: `chmod permissions filename`

- There are 2 ways to use the command -
 - Absolute mode
 - Symbolic mode

Absolute Mode

- In this mode, file permissions are not represented as characters but as a three-digit octal number. The numeric code for different type of permissions is represented as in the table:

Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read + Write	rw-
7	Read + Write +Execute	rwx

■ Example:

Checking Current File Permissions

```
ubuntu@ubuntu:~$ ls -l sample  
-rw-rw-r-- 1 ubuntu ubuntu 15 Sep 6 08:00 sample
```

chmod 764 and checking permissions again

```
ubuntu@ubuntu:~$ chmod 764 sample  
ubuntu@ubuntu:~$ ls -l sample  
-rwxrw-r-- 1 ubuntu ubuntu 15 Sep 6 08:00 sample
```

■ '764' absolute code says the following:

- Owner can read, write and execute
 - Usergroup can read and write
 - World can only read This is shown as '-rwxrw-r-,,
- This is shown as '-rwxrw-r-

Symbolic Mode

- In the symbolic mode, one can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.

Operator	Description
+	Adds a permission to a file or directory
-	Removes the permission
=	Sets the permission and overrides the permissions set earlier.

User Denotations	
U	user/owner
G	Group
O	Other
A	All

Example:

Current File Permissions

```
home@VirtualBox:~$ ls -l sample  
-rw-rw-r-- 1 home home 55 2012-09-10 10:59 sample
```

Setting permissions to the 'other' users

```
home@VirtualBox:~$ chmod o=rwx sample  
home@VirtualBox:~$ ls -l sample  
-rw-rw-rwx 1 home home 55 2012-09-10 10:59 sample
```

Adding 'execute' permission to the usergroup

```
home@VirtualBox:~$ chmod g+x sample  
home@VirtualBox:~$ ls -l sample  
-rwx-rwxrwx 1 home home 55 2012-09-10 10:59 sample
```

Removing 'read' permission for 'user'

```
home@VirtualBox:~$ chmod u-r sample  
home@VirtualBox:~$ ls -l sample  
--w-rwxrwx 1 home home 55 2012-09-10 10:59 sample
```

Assignment -1 (Additional Questions)

16. Write a command to give write permission to all the users of file reginfo.
17. Write a command to discard write permission from group users group users of file reginfo.
18. Write the command to set rwx permissions for all the users of file reginfo.



Shell Programming

Lec-04

Shell

- A shell or command interpreter in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output.
- It is the interface through which a user works on the programs, commands, and scripts.
- A shell is accessed by a terminal which runs it.
- When you run the terminal, the Shell issues a **command prompt (usually \$)**, where you can type your input, which is then executed when you hit the Enter key. The output or the result is thereafter displayed on the terminal.

Types of Shell

- There are two main shells in Linux:
- 1. **The Bourne Shell**: The prompt for this shell is \$ and its derivatives are listed below:
 - Bourne shell also is known as sh
 - Korn Shell also known as ksh
 - **Bourne Again SHell** also known as bash (most popular)
- 2. **The C shell**: The prompt for this shell is %, and its subcategories are:
 - C shell also is known as csh
 - Tops C shell also is known as tcsh
- You can check your current shell using command
 - echo \$SHELL

Shell Scripts (1)

- Basically, a shell script is a text file with Unix commands in it.
- Shell scripts usually begin with a `#!` and a shell name (shebang)
 - For example: `#!/bin/sh`
 - If they do not, the user's current shell will be used
- Any Linux command can go in a shell script
 - Commands are executed in order or in the flow determined by control statements.
- Different shells have different control structures
 - The `#!` line is very important
 - We will write shell scripts with the Bourne shell (`sh`)

Shell Scripts (2)

- Why write shell scripts?
 - To avoid repetition:
 - If you do a sequence of steps with standard Unix commands over and over, why not do it all with just one command?
 - To automate difficult tasks:
 - Many commands have subtle and difficult options that you don't want to figure out or remember every time.

Assigning Command Output to a Variable

- Using backquotes, we can assign the output of a command to a variable:

```
#!/bin/sh
```

```
files=`ls`
```

```
echo $files
```

- Very useful in numerical computation:

```
#!/bin/sh
```

```
value=`expr 12345 + 54321`
```

```
echo $value
```

Please keep space in mind ☺

Using expr for Calculations

- Variables as arguments:

```
count=5
```

```
count=`expr $count + 1`
```

```
echo $count
```

```
6
```

- Variables are replaced with their values by the shell!

- expr supports the following operators:

- arithmetic operators: +,-,*/,% *

- comparison operators: <, <=, ==, !=, >=, >

- precedence is the same as C, Java

Control Statements

- Without control statements, execution within a shell script flows from one statement to the next in succession.
- Control statements control the flow of execution in a programming language
- The two most common types of control statements:
 - conditionals: if/then/else, case, ...
 - loop statements: while, for, until, do, ...

Conditionals

- Conditionals are used to “test” something.
 - In Java or C, they test whether a Boolean variable is true or false.
 - In a Bourne shell script, the only thing you can test is whether or not a command is “successful”
- Every well behaved command returns back a **return code**.
 - 0 if it was successful
 - Non-zero if it was unsuccessful (actually 1..255)

The if Statement

- Simple form:

if(cond)

if decision_command_1 { ggg }

then

command_set_1

fi

grep returns 0 if it finds something
returns non-zero otherwise

- Example:

if grep unix myfile >/dev/null

then

echo "It's there"

fi



redirect to /dev/null so that
"intermediate" results do not get
printed

if and else

```
if grep "UNIX" myfile >/dev/null  
then  
    echo UNIX occurs in myfile  
else  
    echo No!  
    echo UNIX does not occur in myfile  
fi
```

if and elif

```
if grep "UNIX" myfile >/dev/null  
then  
    echo "UNIX occurs in file"  
elif grep "DOS" myfile >/dev/null  
then  
    echo "Unix does not occur, but DOS does"  
else  
    echo "Nobody is there"  
fi
```

case Statement

- You can use multiple **if...elif** statements to perform a multiway branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable.
- Shell supports **case...esac** statement which handles exactly this situation, and it does so more efficiently than repeated if...elif statements.

Syntax case...esac

```
case string in
  pattern1)
    command_set_11
  ;;
  pattern2)
    command_set_2
  ;;
...
*)
  default statement to be executed
esac
```

```
#!/bin/sh
```

```
FRUIT="kiwi"
```

```
case "$FRUIT" in
```

```
    "apple") echo "Apple pie is quite tasty."
```

```
;;
```

```
    "banana") echo "I like banana nut bread."
```

```
;;
```

```
    "kiwi") echo "New Zealand is famous for kiwi."
```

```
;;
```

```
esac
```

for Loops

- for loops allow the repetition of a command for a specific set of values
- Syntax:

```
for var in value1 value2 ... for i in 0 1 2 3 4  
do  
    command_set  
done
```

- command_set is executed with each value of var (value1, value2, ...) in sequence

for Loop Example (1)

```
#!/bin/sh
# timetable – print out a multiplication table
for i in 1 2 3
do
    for j in 1 2 3
    do
        value=`expr $i \* $j`
        //In shell, * represents all files in the current directory so use '\*'
        echo -n "$value "
    done
done
```

for Loop Example (2)

```
#!/bin/sh
files=`ls`      echo *
for i in $files
do
    echo -n "$i "
done
```

- Find filenames in files in current directory

for Loop Example (3)

```
#!/bin/sh
# file-poke – tell us stuff about files
for i in *
do
    echo -n "$i "
done
```

- Same as previous slide, only a little more condensed.



Shell Programming

Lec-05

Use of Semicolons

- Instead of being on separate lines, statements can be separated by a semicolon (;)
 - For example:

```
if grep "UNIX" myfile>/dev/null; then echo  
"Got it"; fi
```

Use of Colon

- Sometimes it is useful to have a command which does “nothing”.
- The : (colon) command in Unix does nothing

```
#!/bin/sh
```

```
if grep unix myfile>/dev/null
```

```
then
```

```
:
```

```
else
```

```
    echo "Sorry, unix was not found"
```

```
fi
```

The test statement – File Tests

- `test -f file` does `file` exist and is not a directory?
- `test -d file` does `file` exist and is a directory?
- `test -x file` does `file` exist and is executable?
- `test -s file` does `file` exist and is longer than 0 bytes?

```
#!/bin/sh
count=0
for i in *; do
    if test -x $i; then
        count=`expr $count + 1`
    fi
done
echo Total of $count files are executable.
```

The test Command – Integer Tests

- Integers can also be compared: >, < , >= – Use -eq, -ne, -lt, -le, -gt, -ge
- For example:

```
#!/bin/sh
smallest=10000
for i in 5 8 19 8 7 3; do
    if test $i -lt $smallest; then
        smallest=$i
    fi
done
echo $smallest
```

Use of []

- The **test** statement has an alias as []
 - Each bracket must be surrounded by spaces!
 - This is supposed to be a bit easier to read.
- For example:

```
#!/bin/sh
smallest=10000
for i in 5 8 19 8 7 3; do
    if [ $i -lt $smallest ] ; then
        smallest=$i
    fi
done
echo $smallest
```



Shell Programming

Lec-06

The while Loop

- ❖ While loops repeat statements as long as the next Unix command is successful.
- ❖ For example:

```
#!/bin/sh
i=1
sum=0
while [ $i -le 100 ]; do
    sum=`expr $sum + $i`
    i=`expr $i + 1`
done
echo The sum is $sum.
```

Nesting while Loops

- It is possible to use a while loop as part of the body of another while loop.
- Syntax

```
while command1 ; # this is loop1, the outer loop  
do
```

Statement(s) to be executed if command1 is true

```
while command2 ; # this is loop2, the inner loop  
do
```

Statement(s) to be executed if command2 is true

```
done
```

```
done
```

Command Line Arguments (1)

- Shell scripts would not be very useful if we could not pass arguments to them on the command line `./q1.sh first second`
- Shell script positional arguments are “numbered” from left to right
 - **\$1** - first argument after command
 - **\$2** - second argument after command
 - ... up to \$9
 - They are called “positional parameters”.

Command Line Arguments (2)

- Example: get a particular line of a file
 - Write a command with the format:
getlineno linenumber filename
#!/bin/sh ./getlineno 5 file1
head -\$1 \$2 | tail -1
- Other variables related to arguments:
 - **\$0** name of the command running
 - **\$*** All the arguments
 - **\$#** the number of arguments

Reading Variables From Standard Input (1)

- The `read` command reads one line of input from the terminal and assigns it to variables given as arguments
- Syntax: `read var1 var2 var3 ...`
 - Action: reads a line of input from standard input
 - Assign first word to `var1`, second word to `var2`, ...
 - The last variable gets any excess words on the line.

Reading Variables from Standard Input (2)

- Example:

```
$ read X Y Z
```

Here are some words as input

```
% echo $X
```

Here

```
% echo $Y
```

are

```
% echo $Z
```

some words as input

Assignment-2 Additional Questions

1. Write a shell script to find the largest among the 3 given numbers (take input from user).
2. Write a shell script to ask your name, program name and enrolment number and print it on the screen.

Redirection in Bourne Shell Scripts (2)

- **Input redirection:**

- `wc -l users.txt`
 - Output: 2 users.txt
- `wc -l < users.txt`
 - Output: 2
- ‘<<’ operator as an instruction to read input until it finds a line containing the specified delimiter. All the input lines up to the line containing the delimiter are then fed into the standard input of the command.
- E.g. `wc -l << EOF`
 - This is a simple lookup program
 - for good (and bad) restaurants
 - in Cape Town.
 - EOF
- Output: 3

Redirection in Bourne Shell Scripts (3)

Another example:

```
#!/bin/sh
```

```
cat << EOF
```

```
This is a simple lookup program  
for good (and bad) restaurants  
in Cape Town.
```

```
EOF
```

- Output:

```
This is a simple lookup program  
for good (and bad) restaurants  
in Cape Town.
```

Redirection in Bourne Shell Scripts (4)

Discard the output:

command > /dev/null

e.g. grep “UNIX” file.txt > /dev/null

Assignment-2 Additional Questions

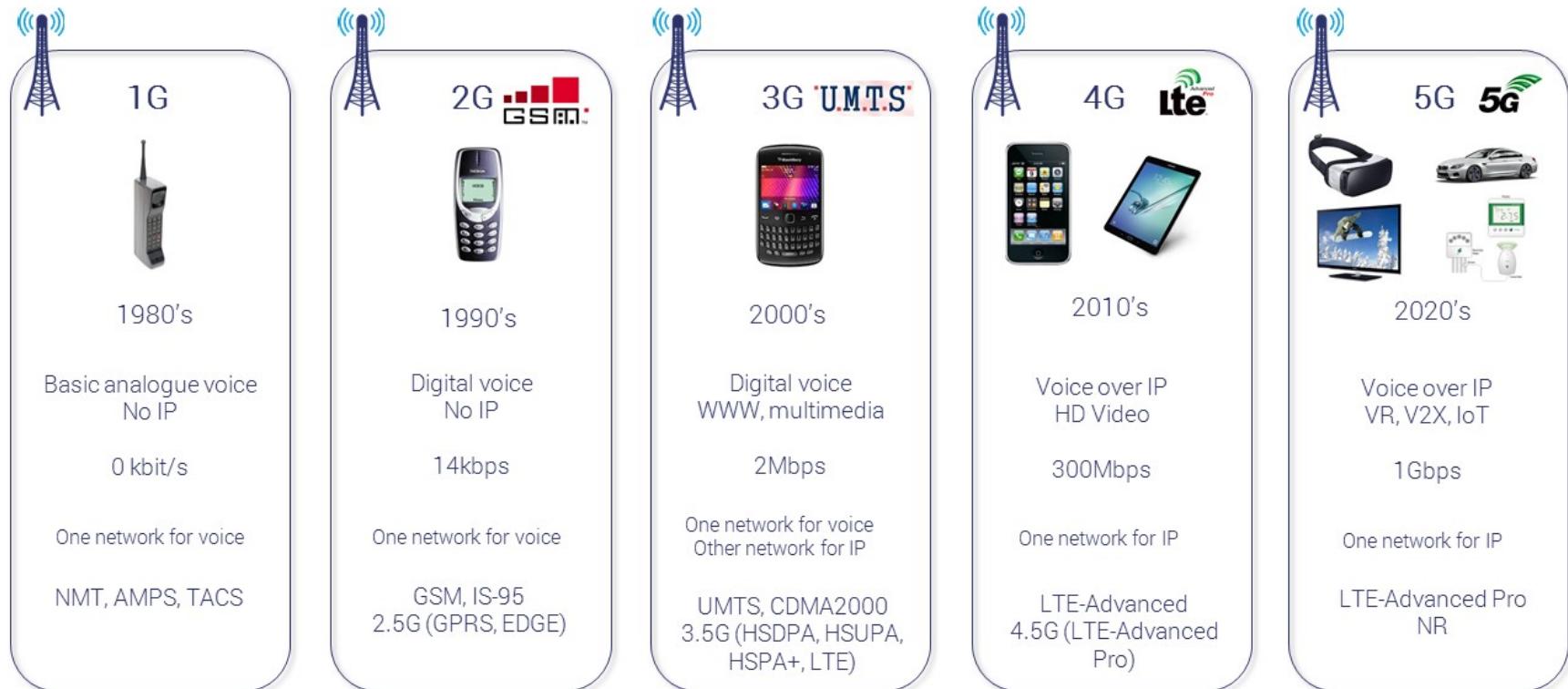
3. Write a shell program to concatenate two strings and find the length of the resultant string (Hint: $s3 = \$s1\$s2$. $s3$ will become the concatenation of $s1$ and $s2$).
4. Write a shell program to display the alternate digits in a given 7-digit number starting from the first digit.
5. Write a shell script to find the smallest of three numbers given as command line argument.

Overview of Computers Workshop

Wireless and Networks
S2022 (Sem-1)

Mobile Networks Evolution

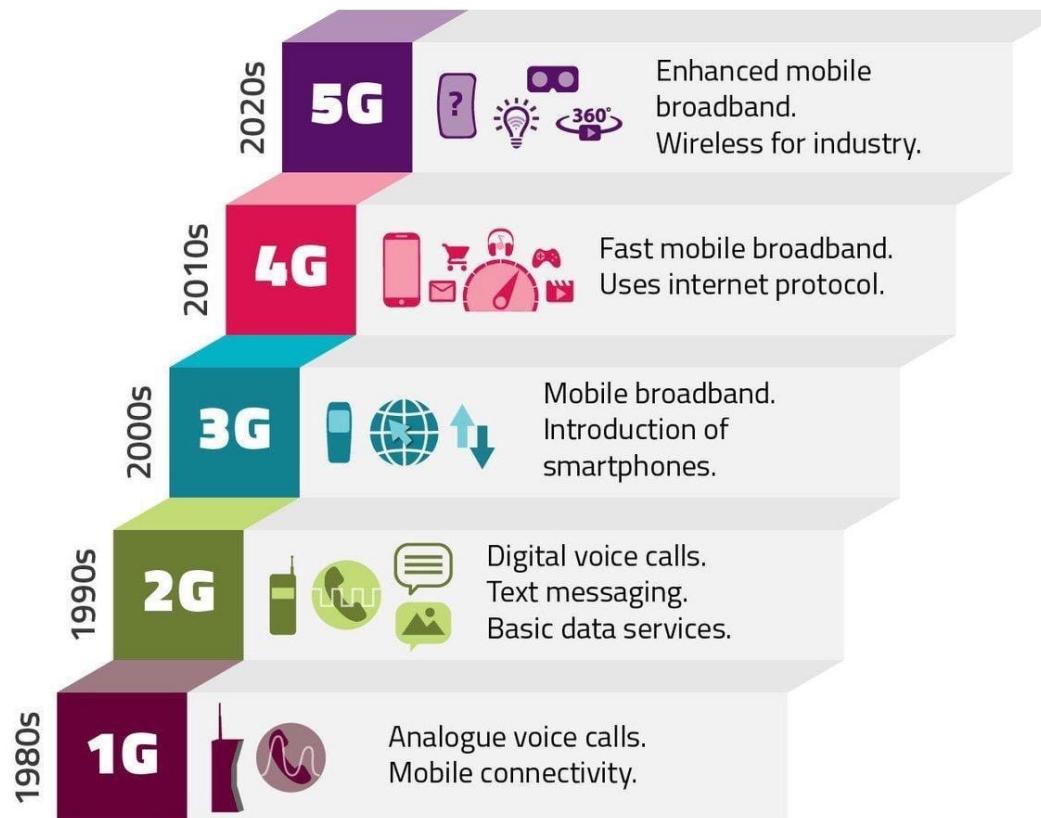
Mobile Wireless Networks - Evolution



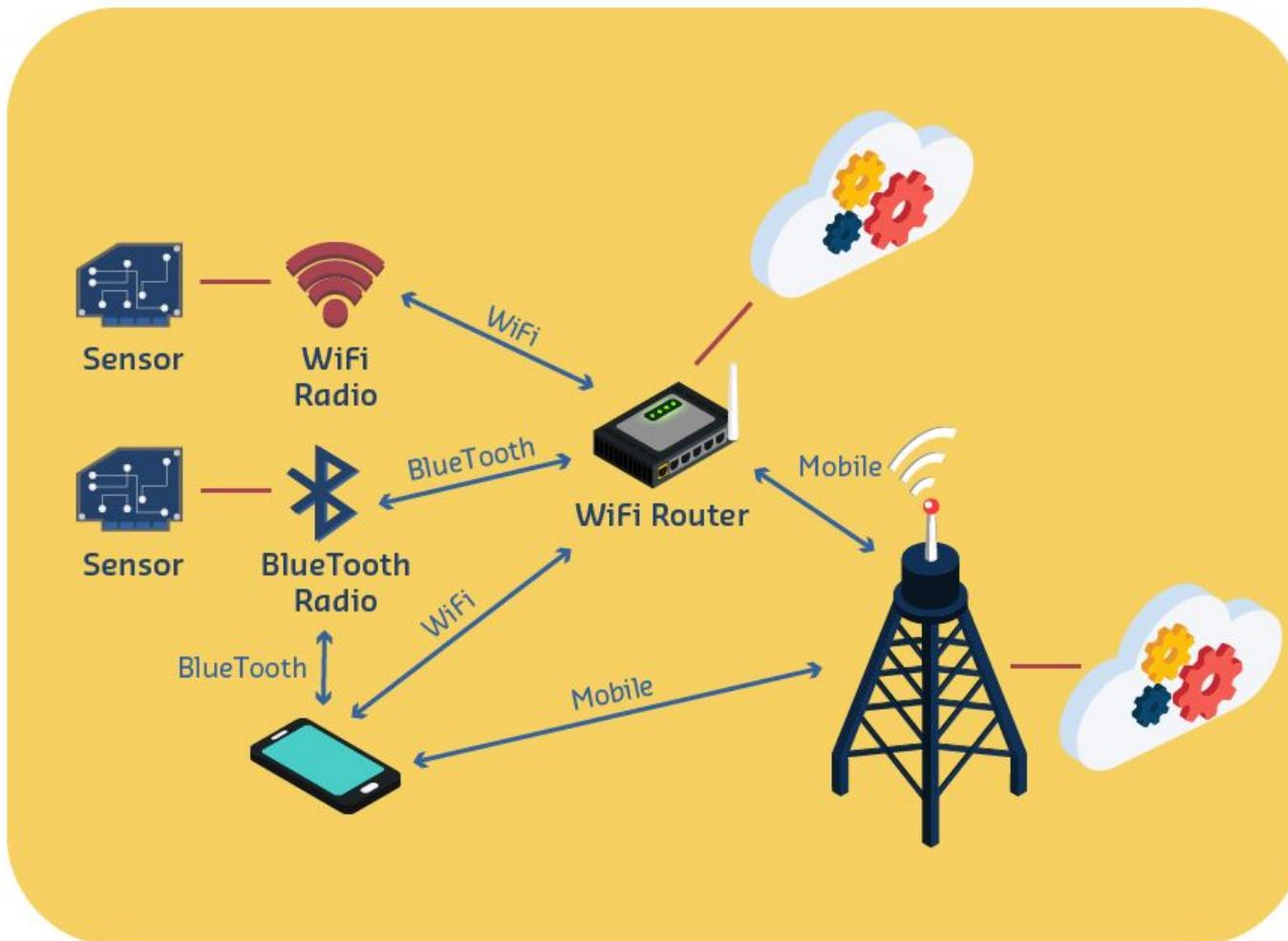
Network Speed

1G	2G	3G	4G	5G
  1G 	  2G 	  3G 	  4G 	 5G 
speed in kilobit per second 2.4 Kbps 	speed in kilobit per second 64 Kbps 	speed in kilobit per second 2,000 Kbps 	speed in kilobit per second 100,000 Kbps 	speed in gigabit per second 1Gbps 
Analog Voice 	Digital Voice + Simple Data 	Mobile Broadband 	Faster and Better Faster Content Delivery More Connections 	Real World Applications 

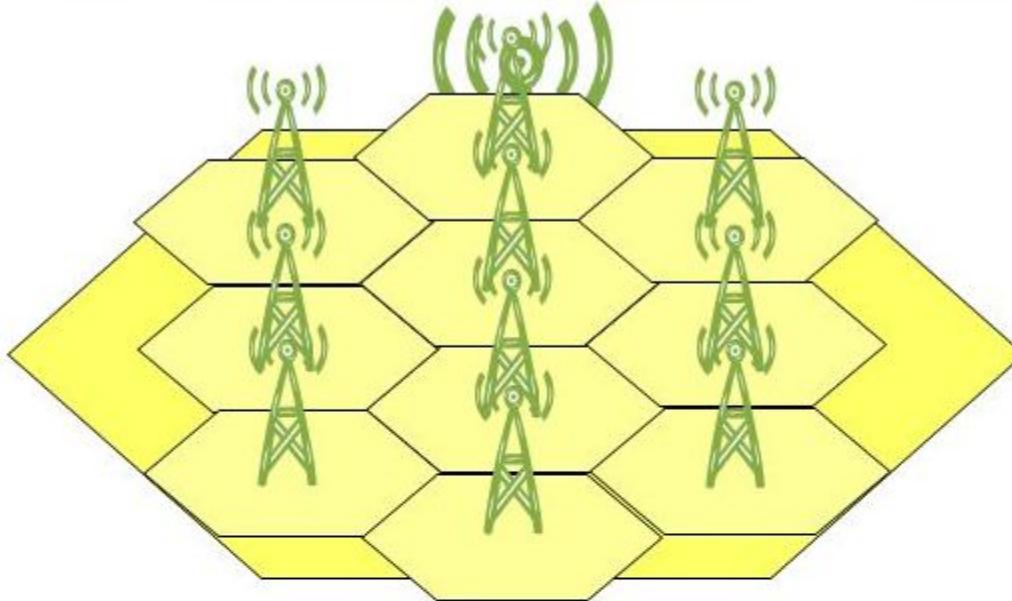
What in each step?



Types of Wireless Network

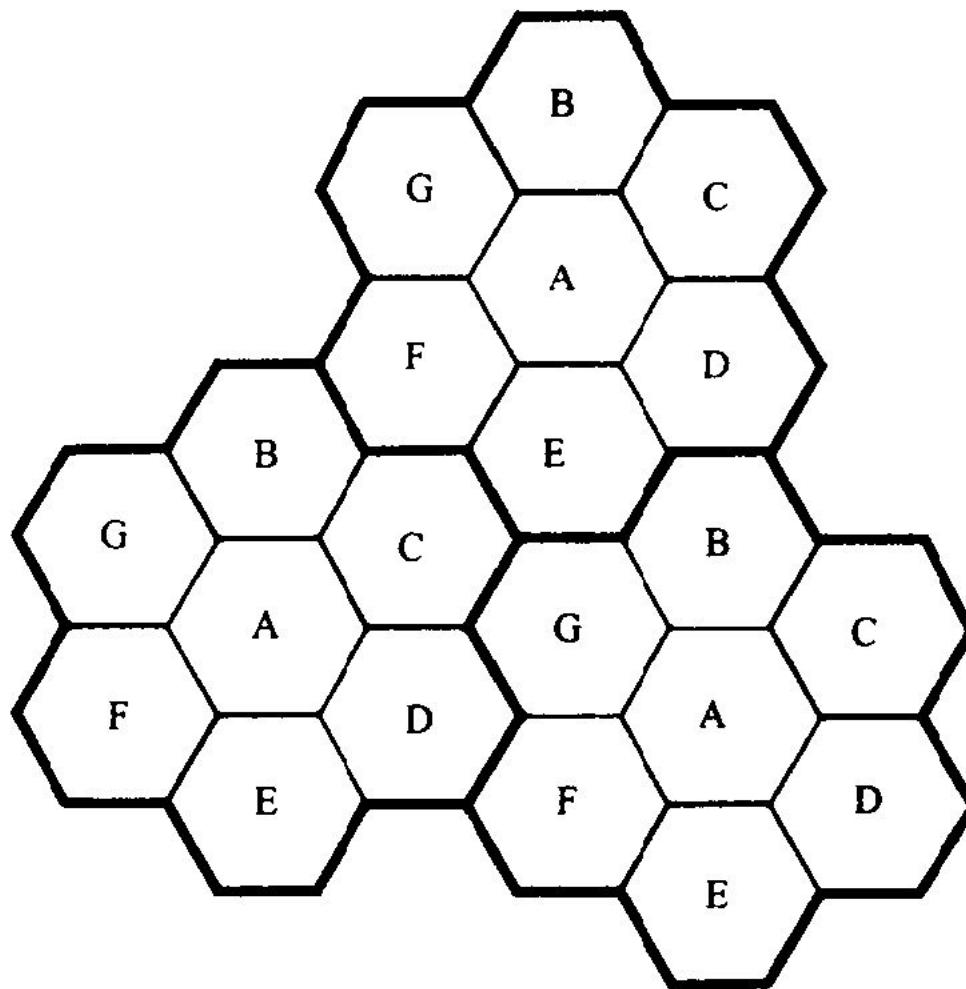


Cellular Concept



- Early Mobile Communications
 - Single, high powered transmitter with an antenna mounted on a tall tower
- The Cellular Concept
 - Replace a single high power transmitter (large cell) with many low power transmitters (small cells) each providing coverage to only a small portion of the service area

Cellular Concept

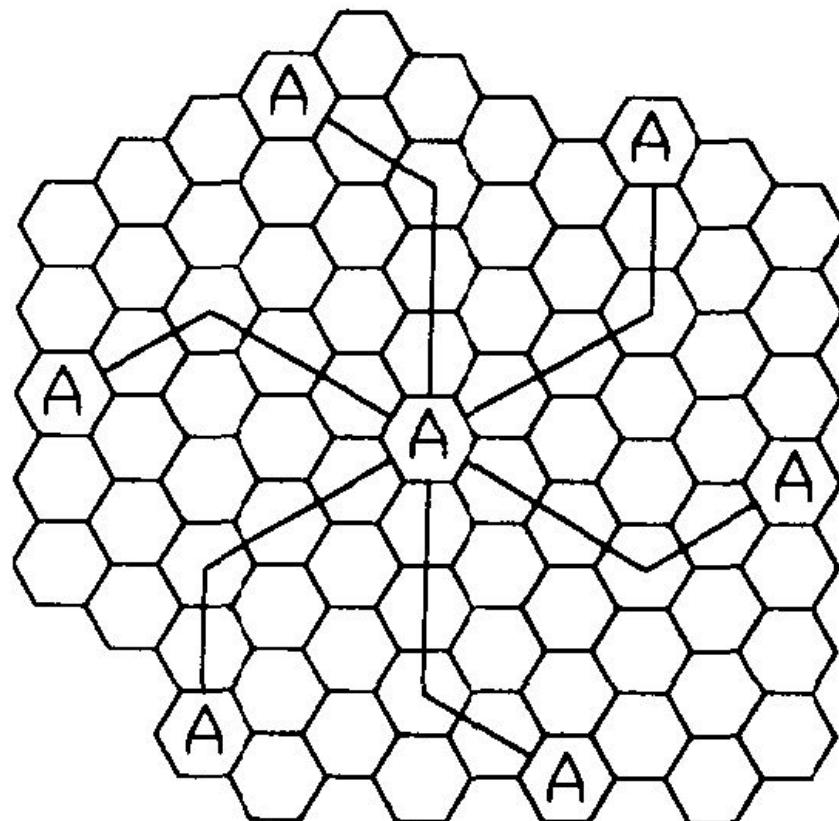


Frequency Reuse

$$S = kN$$

$$C = MkN = MS$$

$$N = i^2 + ij + j^2$$

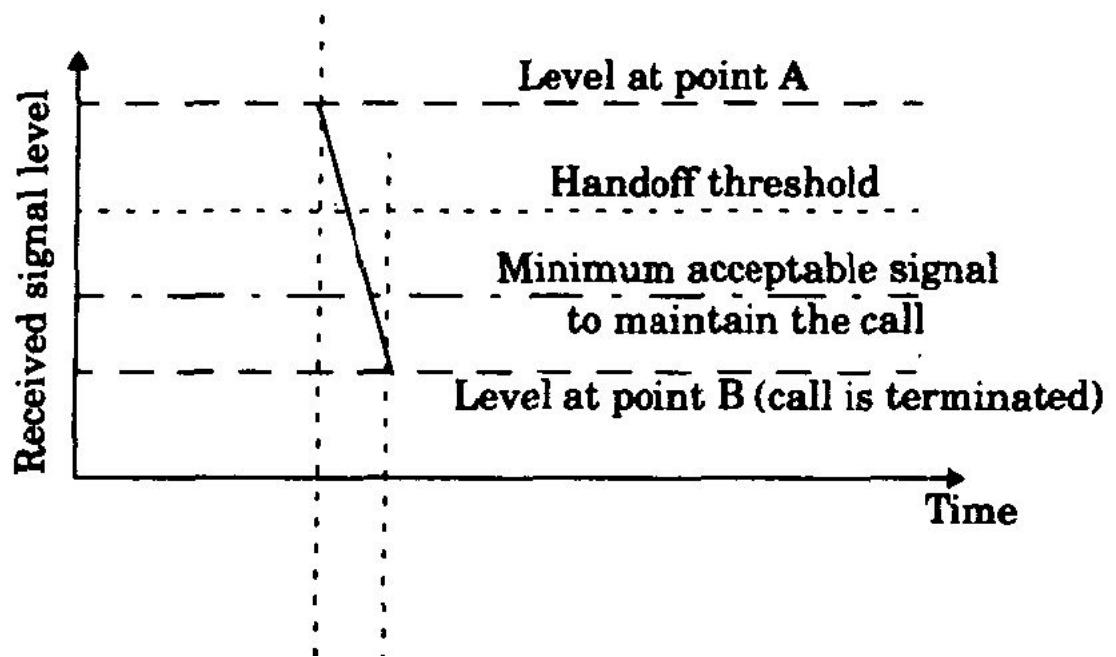


$$Q = \frac{D}{R} = \sqrt{3N}$$

	Cluster Size (N)	Co-channel Reuse Ratio(Q)
$i = 1, j = 1$	3	3
$i = 1, j = 2$	7	4.58
$i = 2, j = 2$	12	6
$i = 1, j = 3$	13	6.24

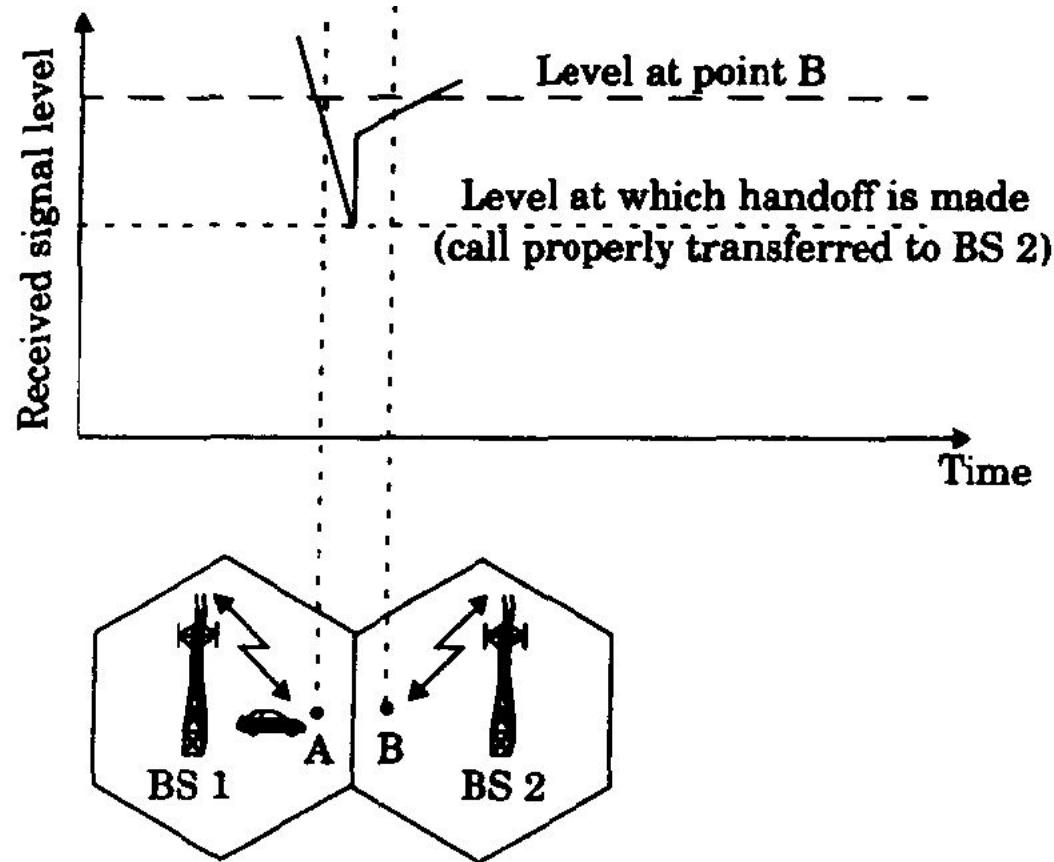
Handover - 1

(a) Improper handoff situation

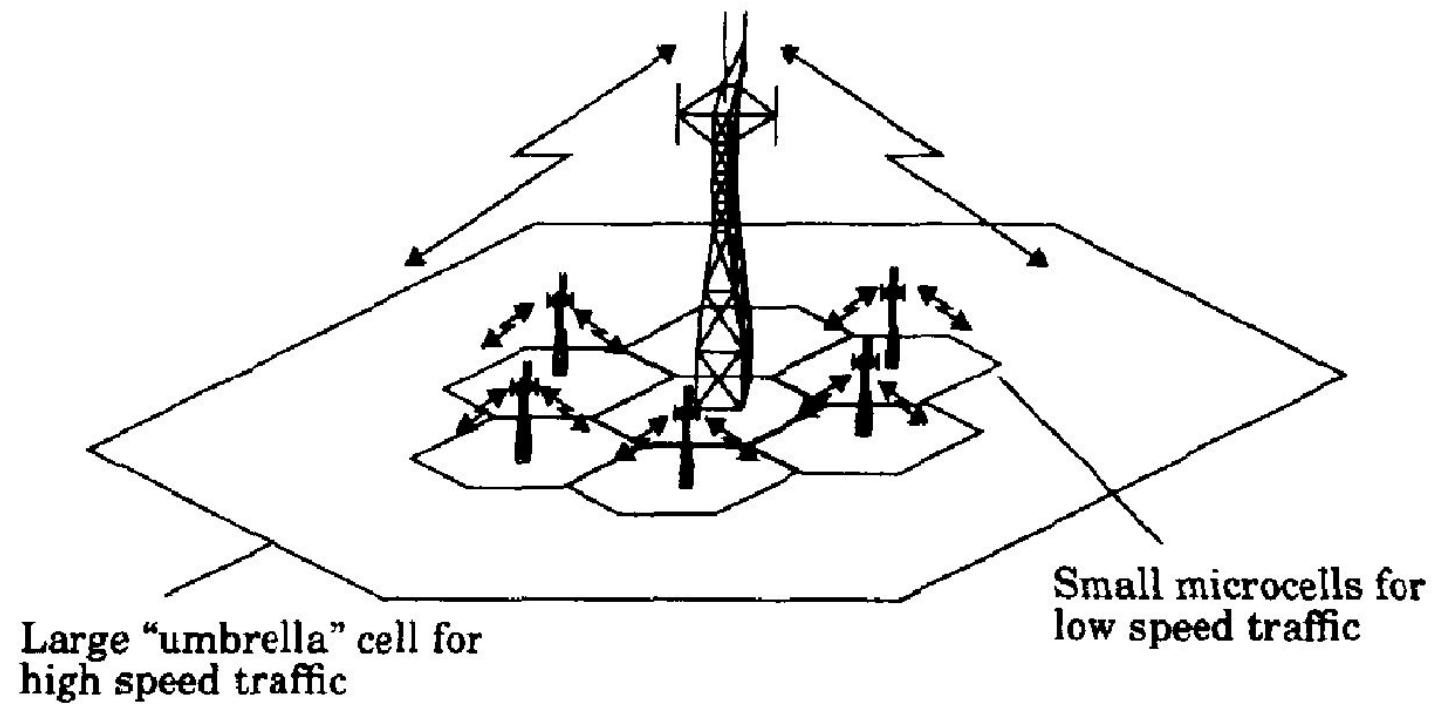


Handover - 2

(b) Proper handoff situation



Umbrella cell Approach



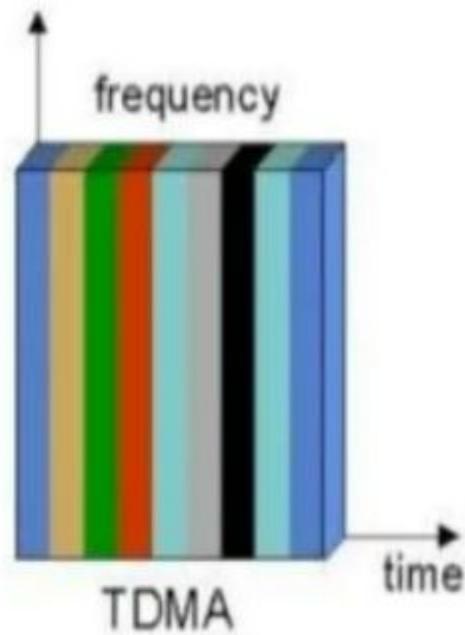
$$\frac{S}{I} = \frac{S}{\sum\limits_{i=1}^{i_0} I_i}$$

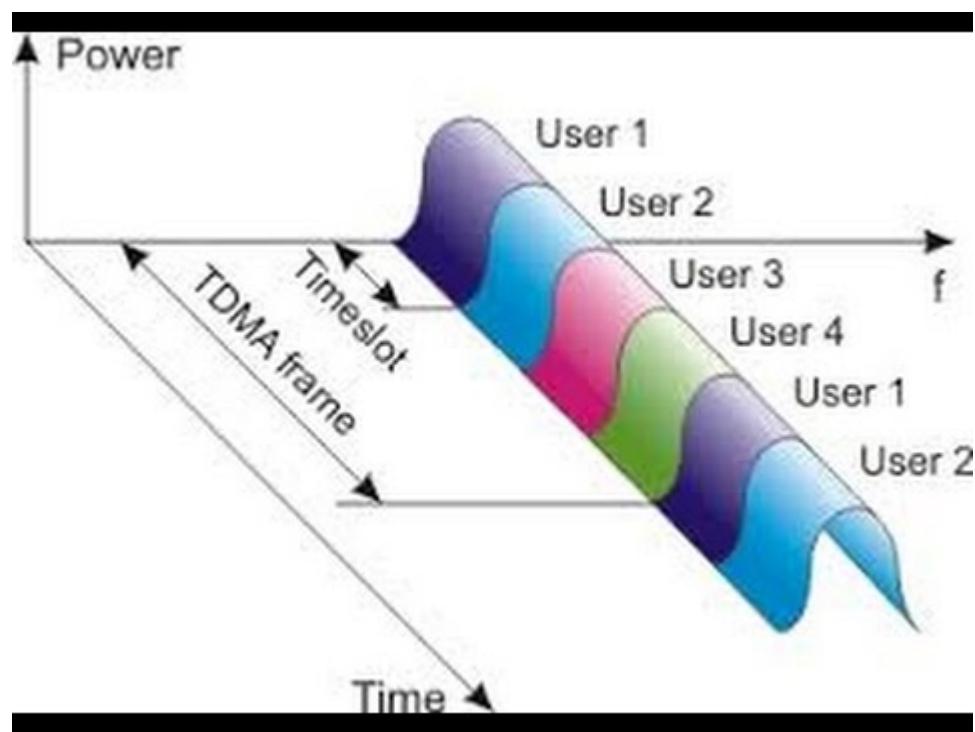
Multiple Access Schemes

OCW – Wireless
S2022 (Feb'22)

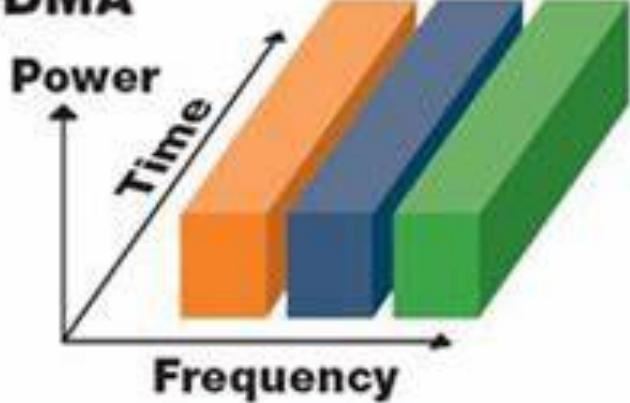
Time Division Multiple Access (TDMA)

- Each user is allowed to transmit only within specified time intervals (Time Slots). Different users transmit in different Time Slots.
- When users transmit, they occupy the whole frequency bandwidth(separation among users is performed in the time domain).

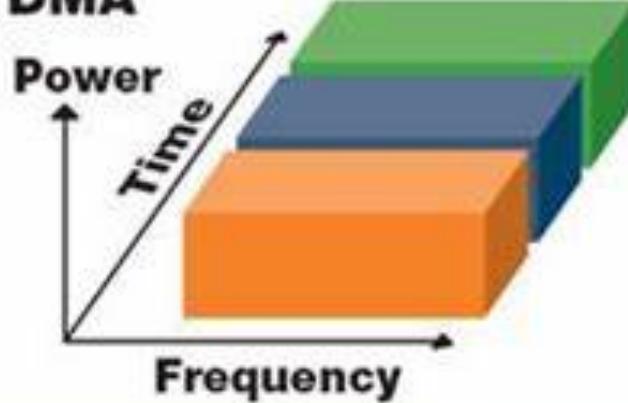




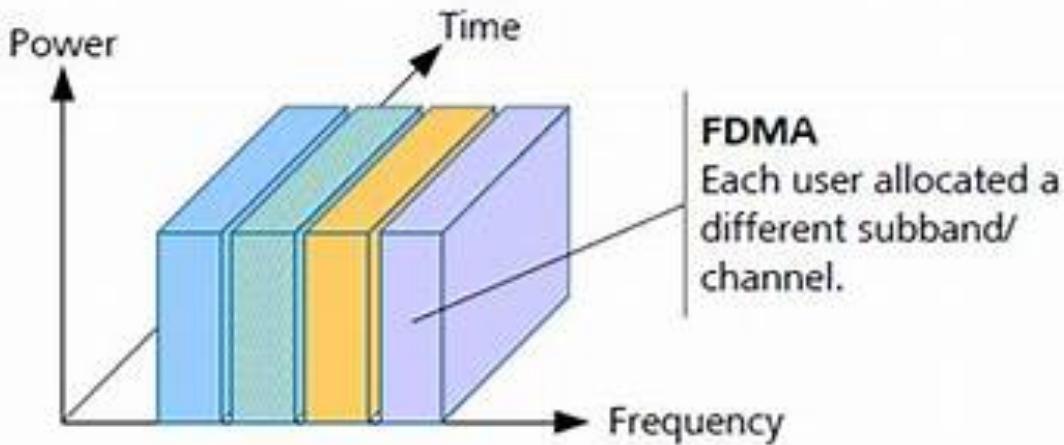
FDMA

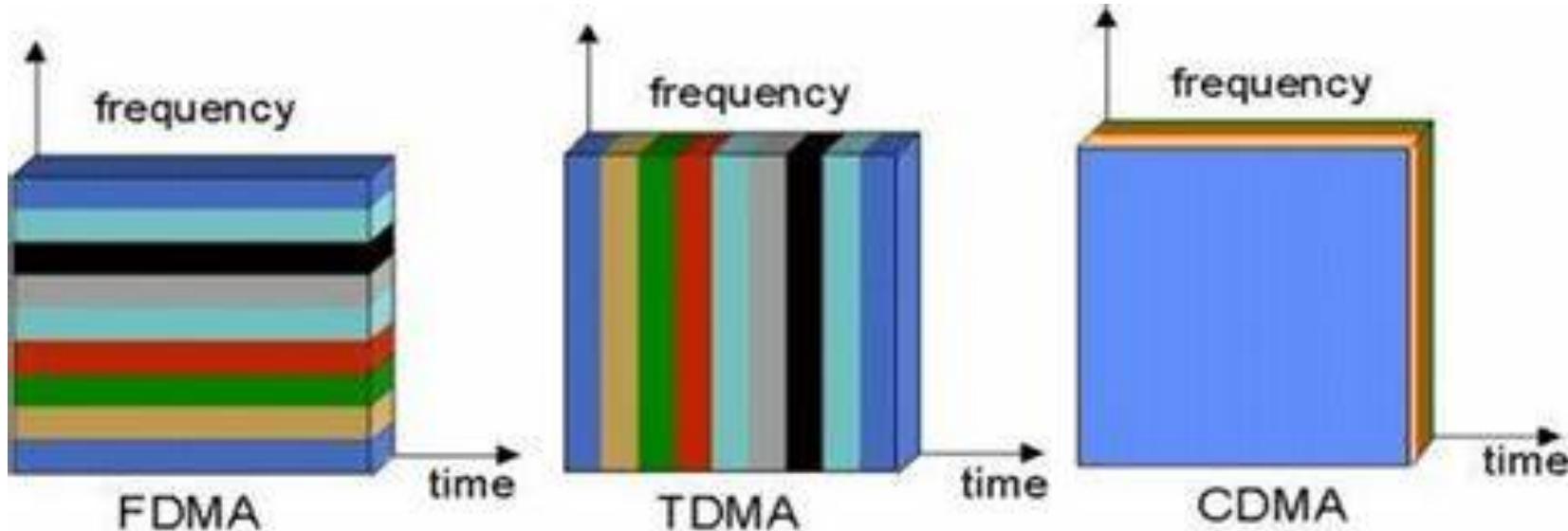


TDMA



Frequency Division Multiple Access







Multiple Access Technologies

- ❖ **FDMA** (example: AMPS)

- Frequency Division Multiple Access**

- ♦ each user has a private frequency

- ❖ **TDMA** (examples: IS-54/136, GSM)

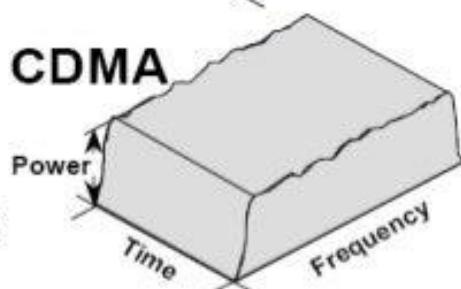
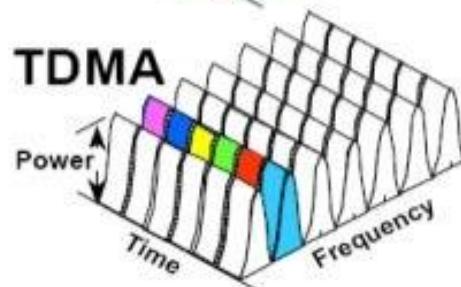
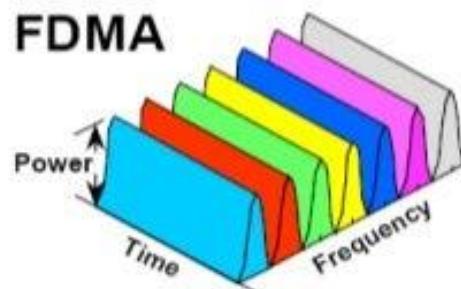
- Time Division Multiple Access**

- ♦ each user has a private time on a private frequency

- ❖ **CDMA** (IS-95, J-Std. 008)

- Code Division Multiple Access**

- ♦ users co-mingle in time and frequency but each user has a private code



Orthogonal Codes...for CDMA

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

A 4×4 Hadamard matrix is created by multiplying each element of the 2×2 matrix by another 2×2 matrix.

$$H_4 = \begin{bmatrix} 1 * \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & 1 * \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ 1 * \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & -1 * \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \end{bmatrix}$$

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Mobile Devices and Computing

Introduction

Modern Mobile Communication System

Main components & systems



Handhelds



Cell Tower



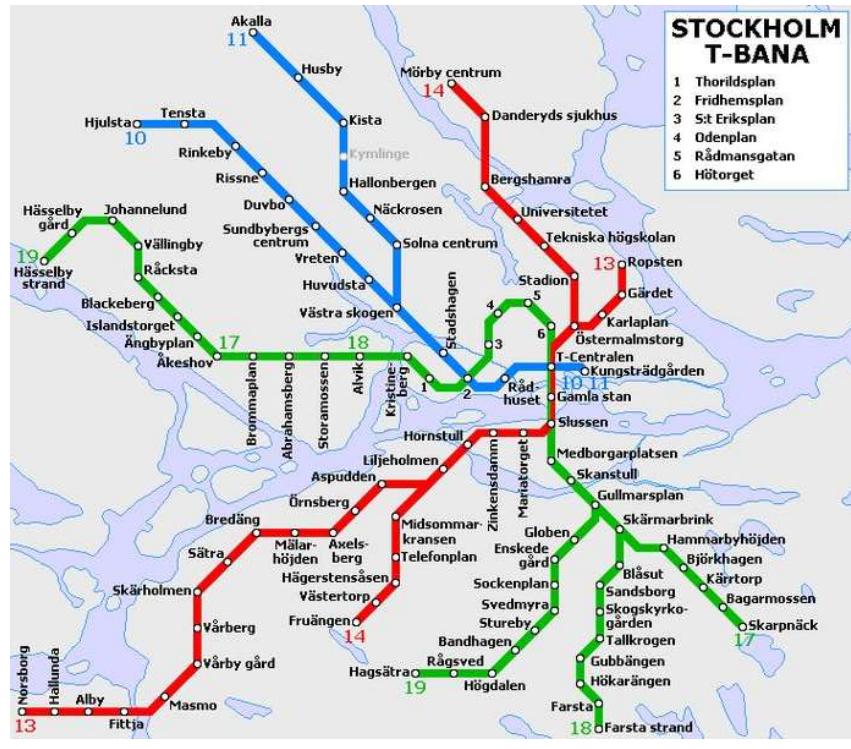
Internet

4G Wireless Communication Network



<https://www.gapwireless.com/5g/attachment/infographic-1-4g/>

Transport Network (Metro Map)

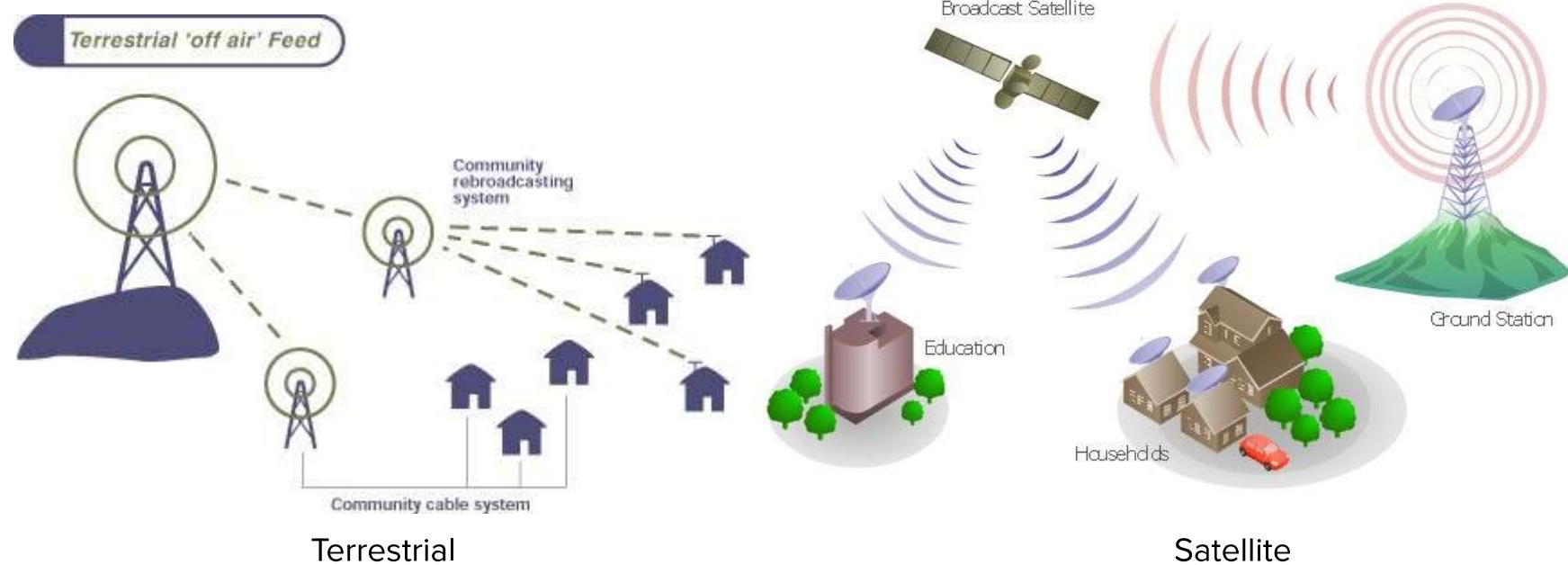


Stations

Central Junctions - Main
Transmitter/Repeater
Sub junction - Base

Station Terminals - Terminals

Radio Systems - Backbone of Comm.

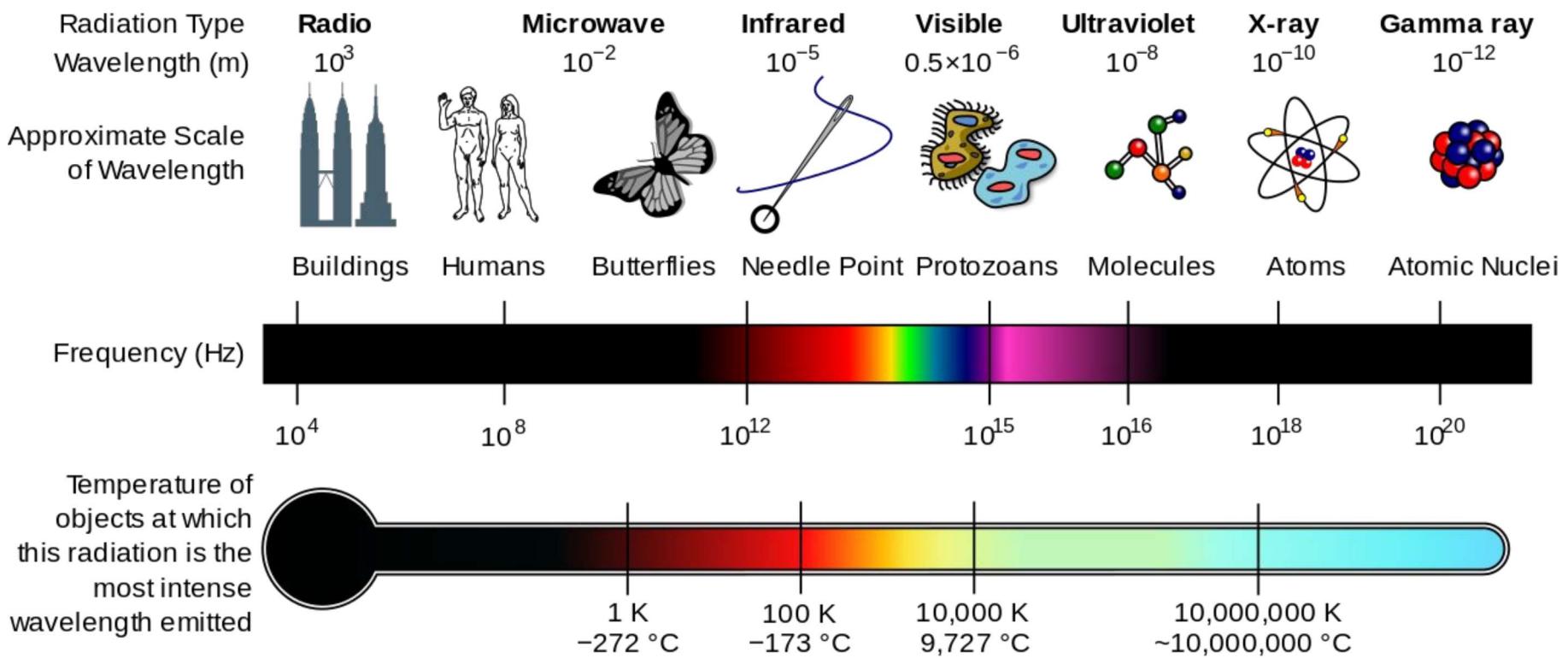


Refs: <https://www.hitechmv.com/wp-content/uploads/2014/05/terrestrial-11.gif>;
<https://www.conceptdraw.com/examples/telecommunication-rf-diagram>

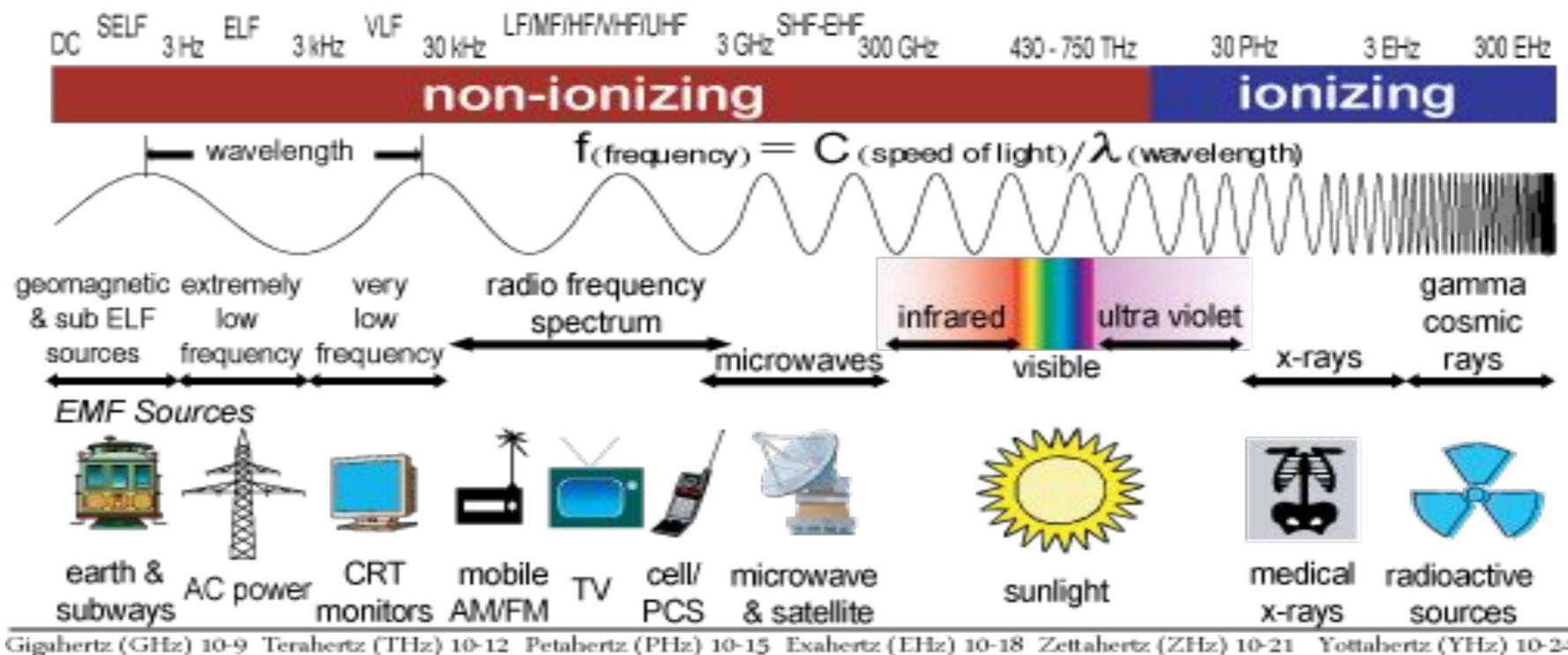
Penetrates Earth's Atmosphere?



Electromagnetic Spectrum

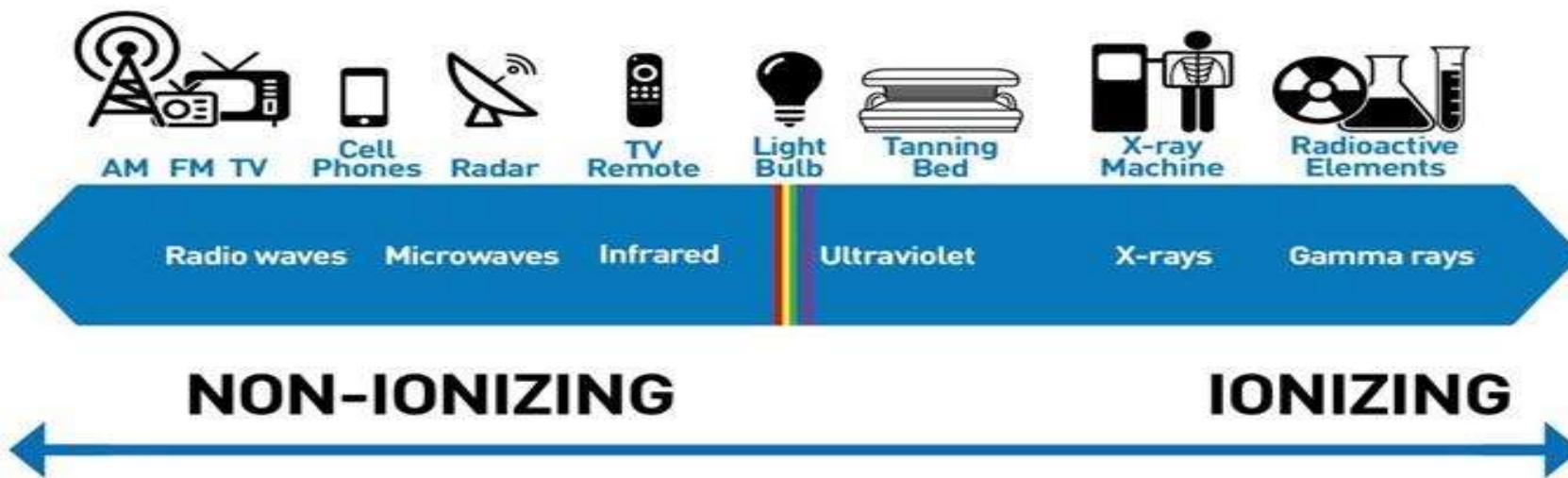


THE ELECTROMAGNETIC SPECTRUM



IONIZATION

Electromagnetic Spectrum

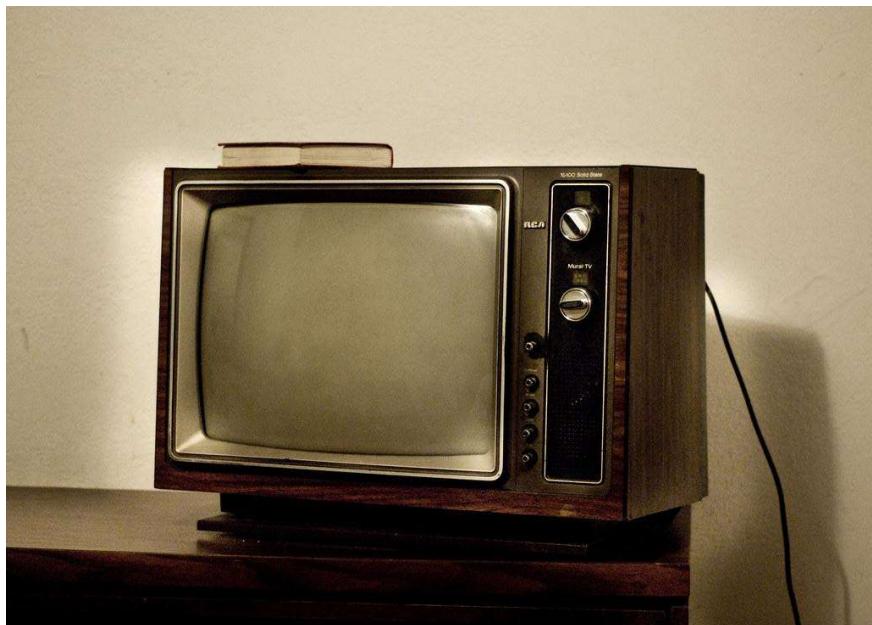


Conventional Radio - 1920s - Audio - (<=) 5kHz



Evolution of radio to podcast <https://visual.ly/community/Infographics/history/history-radio>

Television = Radio + Visuals 1930s - 50MHz

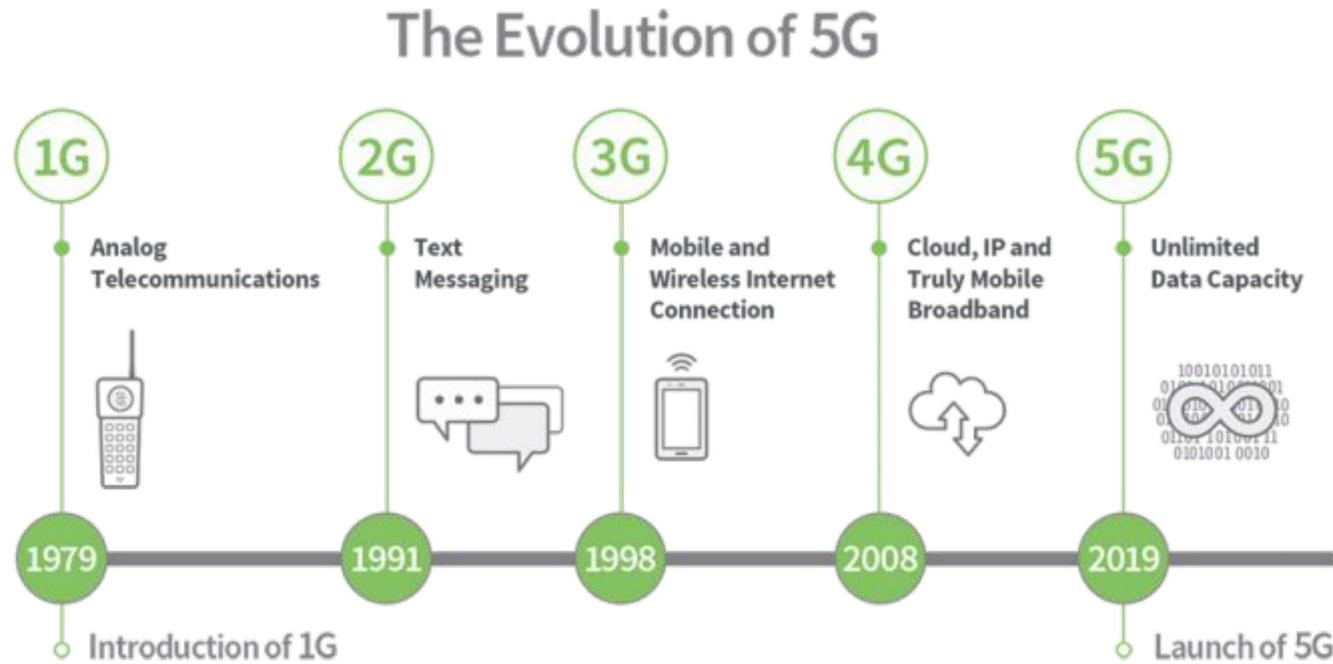


Evolution: <https://visual.ly/community/Infographics/technology/televisions-six-eras>

Digital Packet Radio - Low GHz

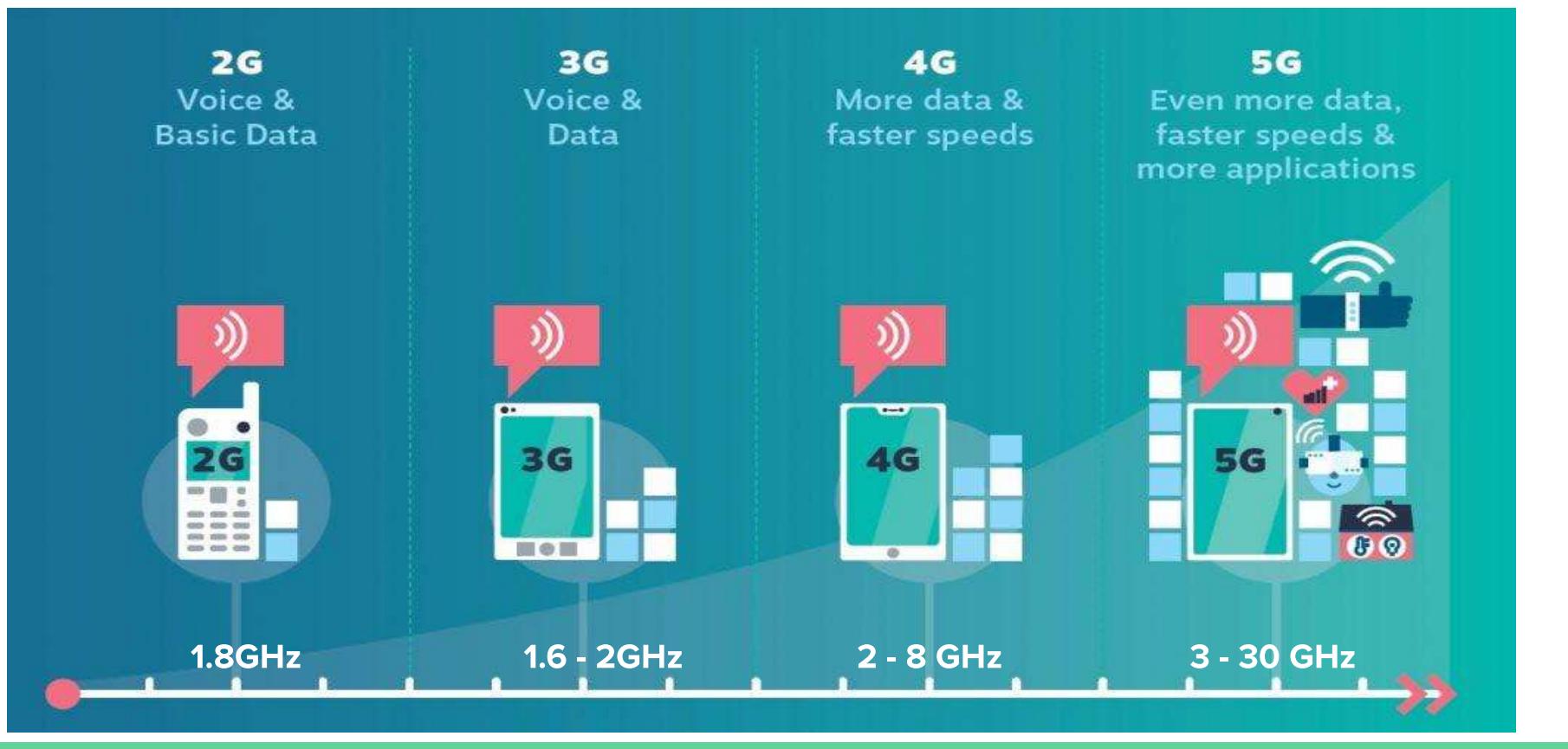


Evolution of wireless communication technology

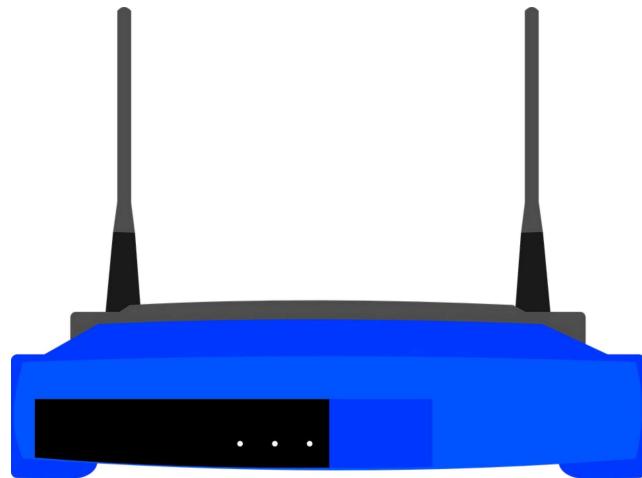


Ref: <https://www.digi.com/blog/post/what-is-5g-part-1-evolution-and-the-next-generation>

Cell Phone - <= 30GHz



Wireless LAN & Wifi Internet - 802.11 std. - 2 to 60 GHz



Access Point



Wifi



WLAN Modems

Overview: https://en.wikipedia.org/wiki/IEEE_802.11

Cell Phone Technology

- A Modern History of Human Communication
- Cell Phone Usage Statistics
- Cell Phone Traffic: Data vs. Voice
- Cell Phone Device Technology

<https://joaogeraldes.wordpress.com/2010/09/07/38-infographic-explores-mobile-phone-evolution-facts-figures-history-statistics/>

Evolution of Mobile Phones

Early phones were bulky, inefficient and limited in functionality



1973 - 2006



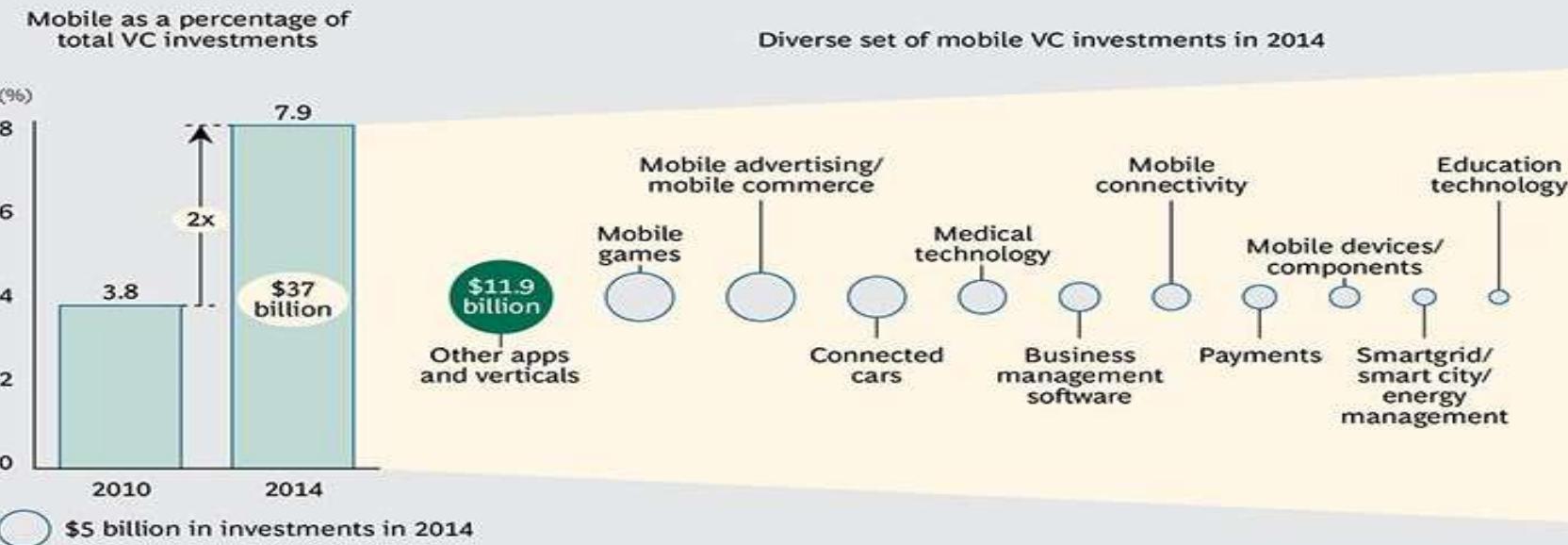
2007 - 2016

Ref: <https://studentwork.prattsi.org/infovis/visualization/evolution-of-mobile-phone/>

Ref: <https://www.iphonelife.com/content/evolution-iphone-every-model-2007-2016>

Market Importance

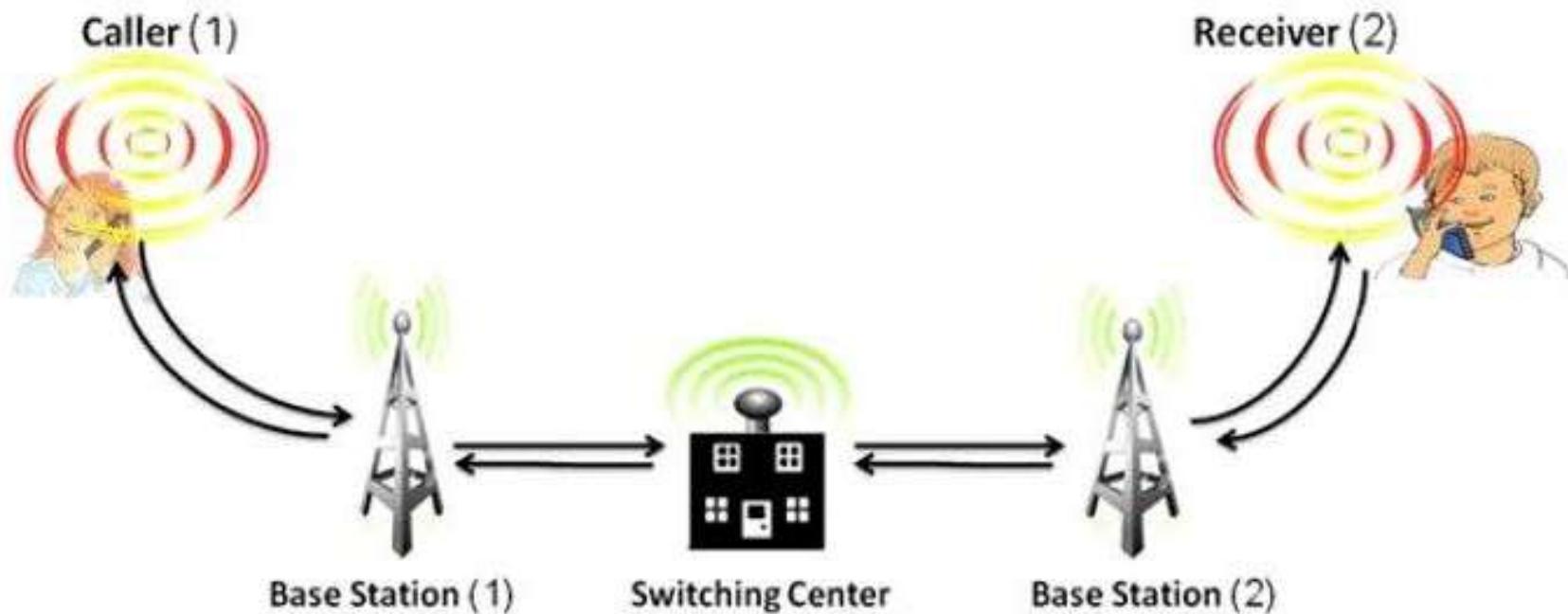
EXHIBIT 5 | Venture Capital Investments in Mobile Are Growing Rapidly



Sources: Quid private-investment database; Capital IQ; BCG analysis.

Ref: <https://www.bcg.com/publications/2015/telecommunications-technology-industries-the-mobile-revolution>

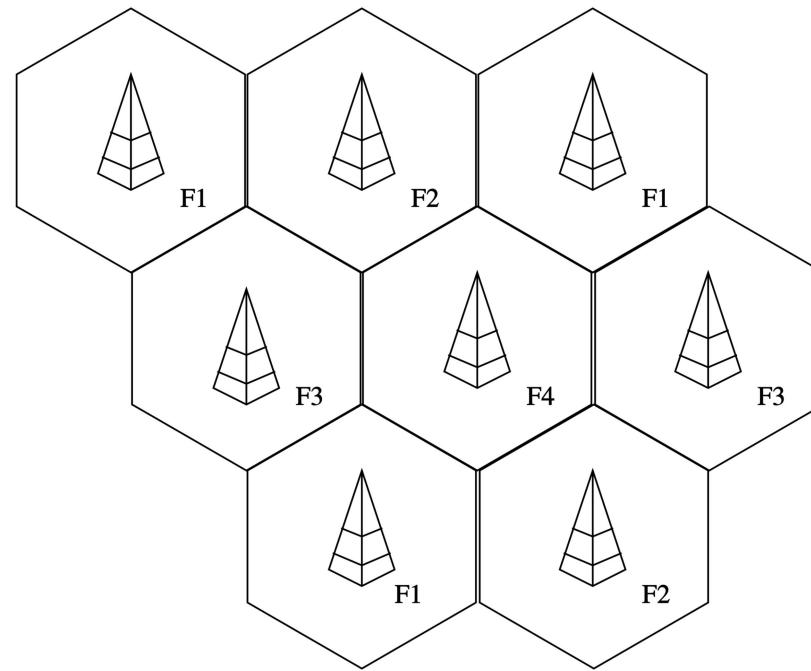
Working of a Cell Phone Call



<https://rmowat.wordpress.com/2011/12/02/how-a-cellular-network-works/>

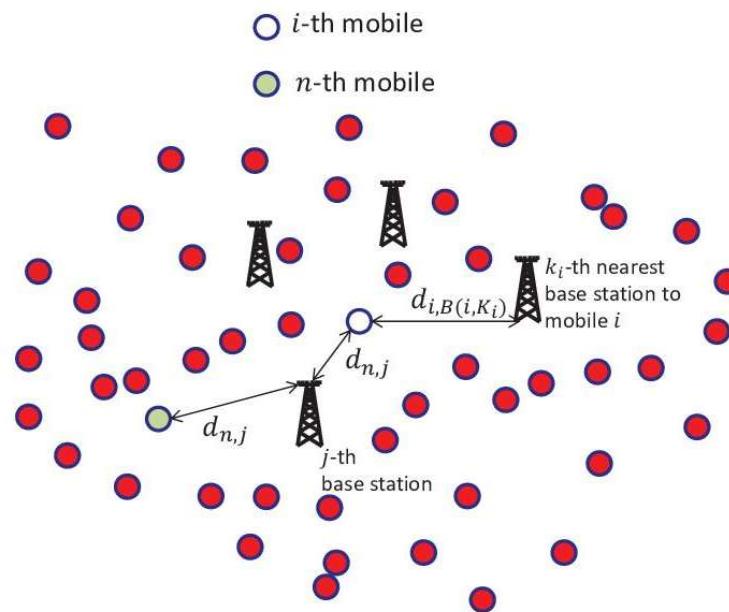
First step

The system divides the service areas into a set of cells or geographical zones



Second Step

Mobile phone sends out radio signals Based on the time delay between the phone and the base station - the nearest station is identified

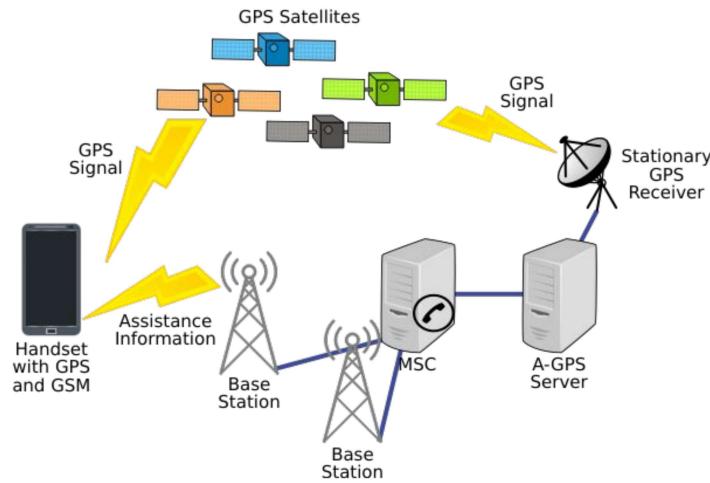


Third Step

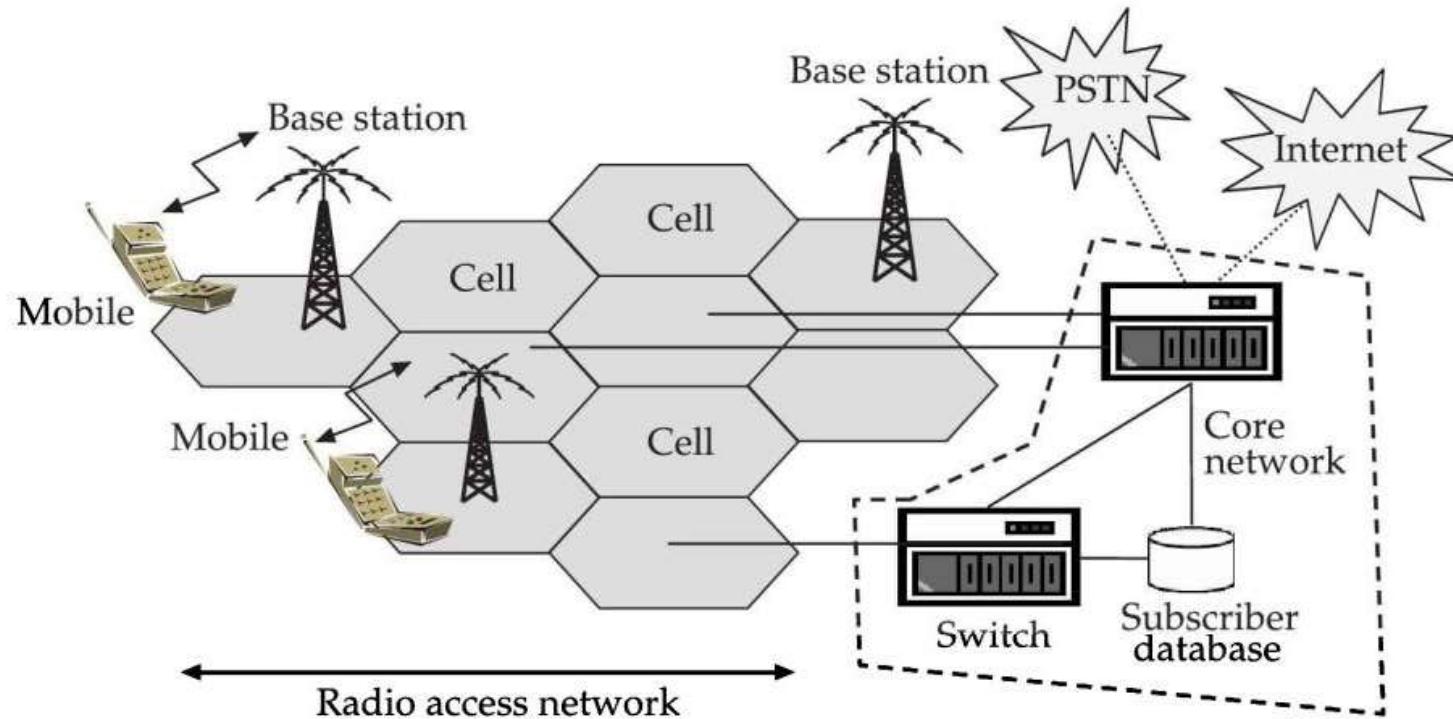
Base station contacts the mobile switching center

The MSC - middleman - to mediate a call between the caller and the receiver

When a user is moving, the MSC hands over the call to nearest base station



Architecture of mobile telecommunications system



Radio Access Network

- Communication link between the mobile phone and the base station - occurs through radio waves
 - Radio waves - EM waves of large wavelengths
 - The shape of each cell is a hexagon
 - Improves the distance between base station and the mobile phone
 - Comm. Channel - A radio link between the phone antenna and the base station antenna or between two base stations and so on
 - Control Channel - control messages - for e.g. making a call - FSK
 - Voice Channel - voice messages - for actual conversations - FM
-

Radio Access Network Contd.

- Base station - two antennas - one for transmitting signals (high power) - other for receiving signals (very lower power)
 - Signals - waves or some form of EM energy - carrying information and function of space and time
 - Antennas - require greater power for transmission due to inverse square law - receive little power due to same law
 - Handheld device - one antenna - for both transmitting and receiving signals
 - One antenna - saves power and space
-

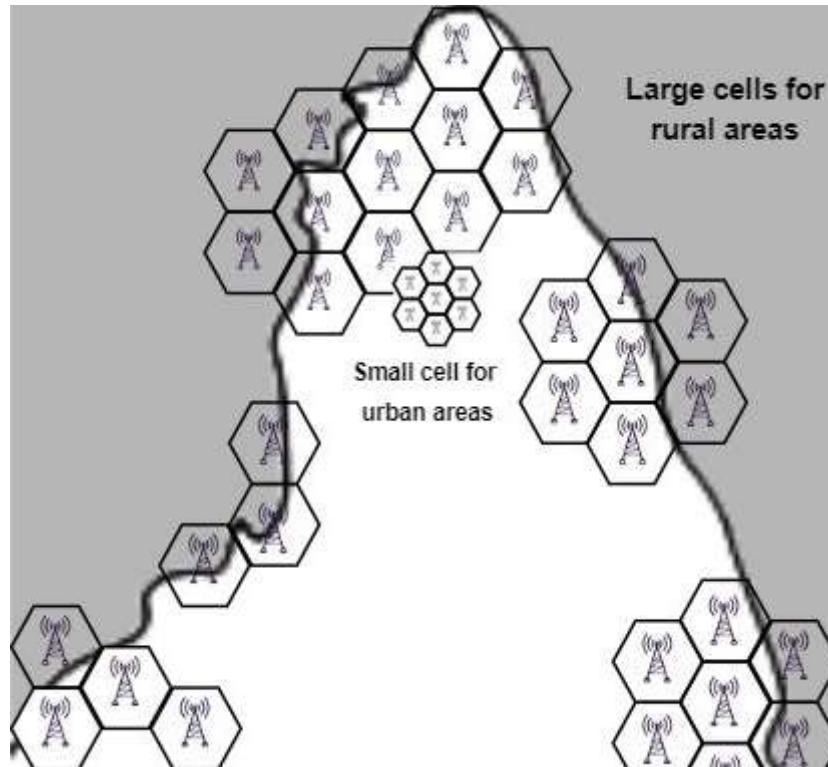
Considerations in Urban Area vs Rural Area

Advantages

- Because cells are smaller, system capacity increases. Also, less power is used by mobiles and Base Stations.

Disadvantages

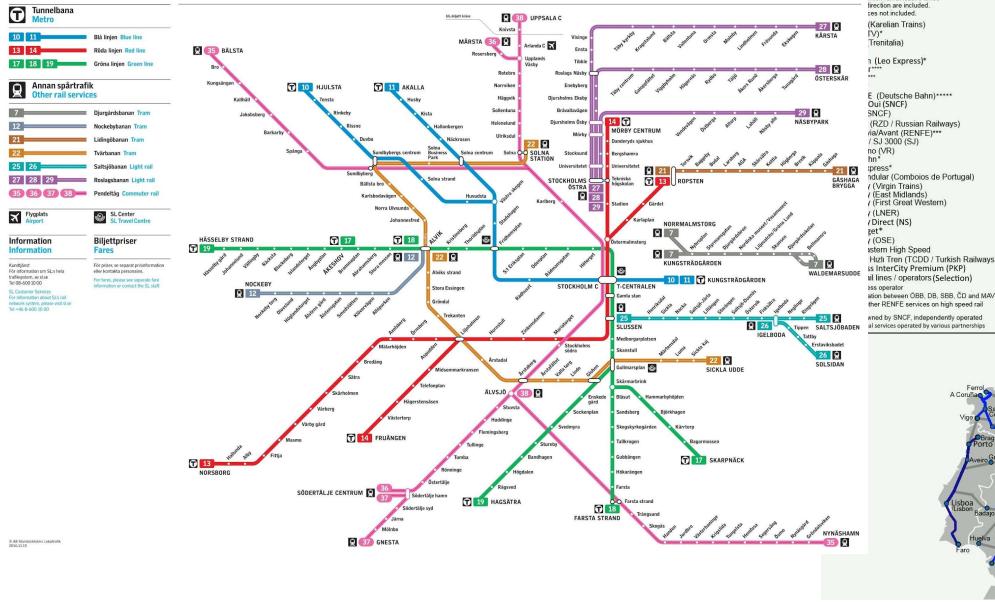
- Handoffs become more common. To prevent handoffs and dropped calls, umbrella cells are needed for high speed traffic.
- Many new base stations are needed, increasing system complexity and load of MSC.
- Since all channels are not split simultaneously, special care have to be taken for proper allocation of the channels.



<https://www.ques10.com/p/48262/coverage-and-capacity-improvement-techniques-1/>

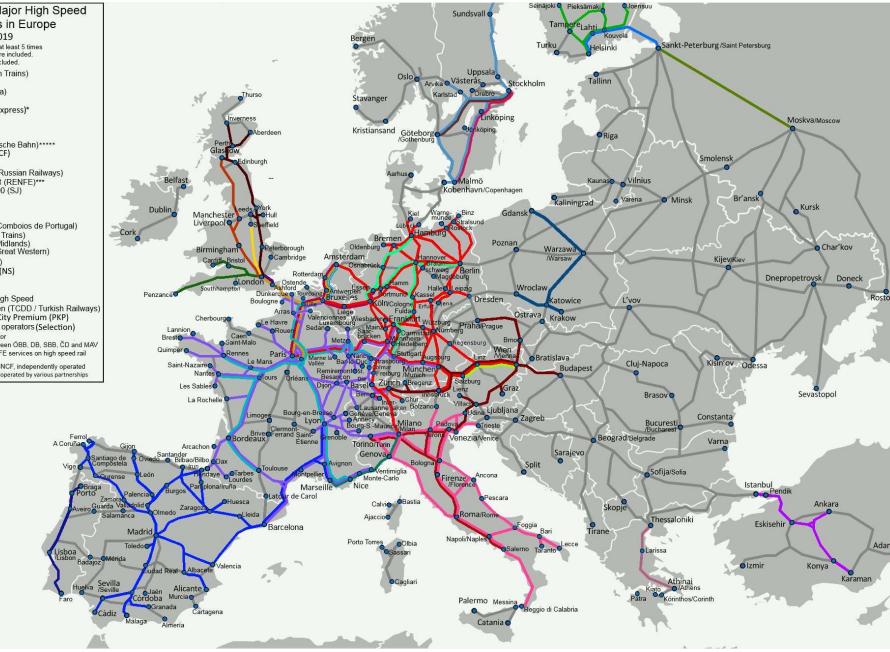
Comparison with Railway network

Stockholm · Spårtrafikkarta Rail Network Map



Map of Major High Speed Rail Operators in Europe
July 2019
Map run at least 5 times
recently. Services not included:
(Eurostar) Trains (TGV)
Trenitalia

1 (Leo Express)*
2 (Trenitalia Combinio de Portugal) (Vrgo Trains) (Oui (SNCF)) (SNCF)
3 (RZD) Russian Railways (Alstom) (RENFE)** (SJ 3000 (SJ)) no VR (The press)*
4 (DB) Deutsche Bahn (EAD) (East Midlands) (Trenitalia) (West) (LNER) (Direct NS) (DB) (OSE) (Orient-Express) (High Speed Train) (TCDD) (Turkish Railways) (InterCity Premium (PKP)) (Other operators) (Selection) (SNCB) independently operated (SNCB) (operated by various partners)



Core Network

- Local base station contacts the Mobile Switching Center
 - When a user is moving the MSC handles the allocation of base station - user
 - Roaming - when a user travels outside the home network his service is handled by a partner network (**PSTN - Public Switched Telephone Network**)
 - Backbone - Roaming is possible because of underground wired network - using fiber optics
 - When the terrain is challenging - microwave links are used - Satellites can also be used for long distance communication
 - MSC is also connected to local telephone network - enable communication with traditional phone networks and users
 - Routing of voice and message calls is also handled
 - Database consists of subscriber information and billing information
-

Mobile Devices and Computing

Mobile Computing

Overview

Mobility in computing



Micro computers

Portable Computers

Tablet computers

Mobile Phones

Enabling Technologies

Mobile Computing - Apps

Software

Sensors

Wifi

Mini Mother-boards

Electronics/Optics

VLSI

Computing Hardware - VLSI - PCBs



Intel CPU



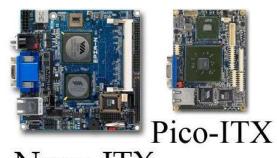
Standard-ATX



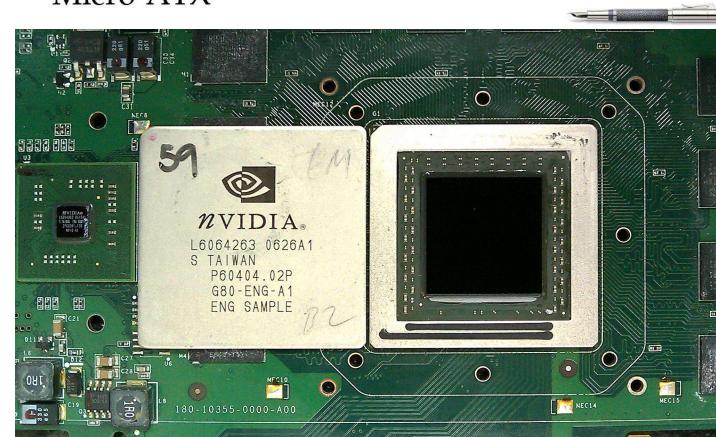
Micro-ATX



Mini-ITX

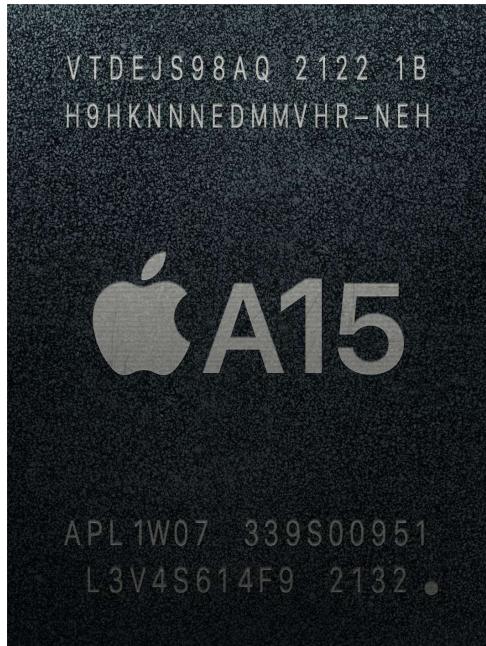


Pico-ITX
Nano-ITX



Nvidia GPU

Mobile Chip - CPUs, GPUs and Neural Engines



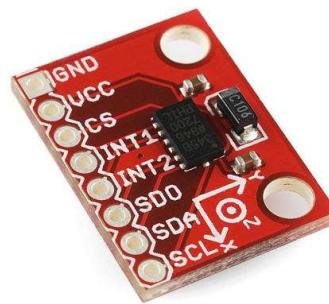
Sensing Hardware



Pressure
Sensor



Mobile
Camera

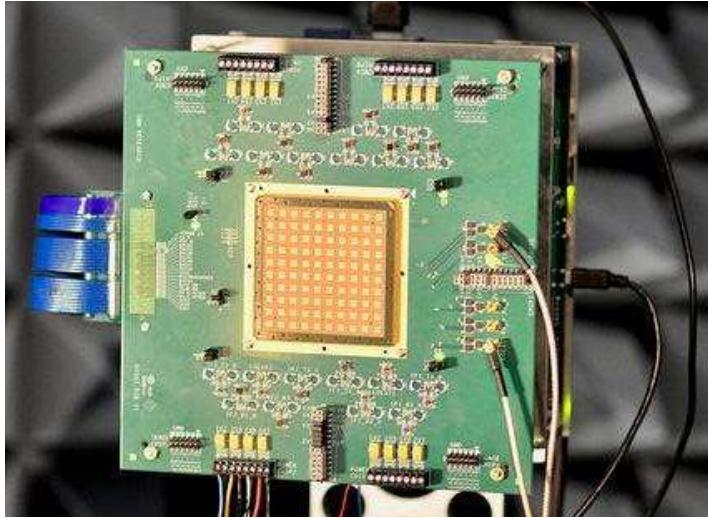


Positioning
Sensor



Temperature
Sensor

Antenna on a mobile phone



Mobile Phone Battery

- Li-ion battery
- At least 24 hours before a recharge
- Wireless charging
- Adapter based charging



Mobile LCD display

- Large screen
- High resolution
2532-by-1170-pixel
resolution
- High refresh rate
- High contrast rate
- Millions of colors



Mobile Software - Apps



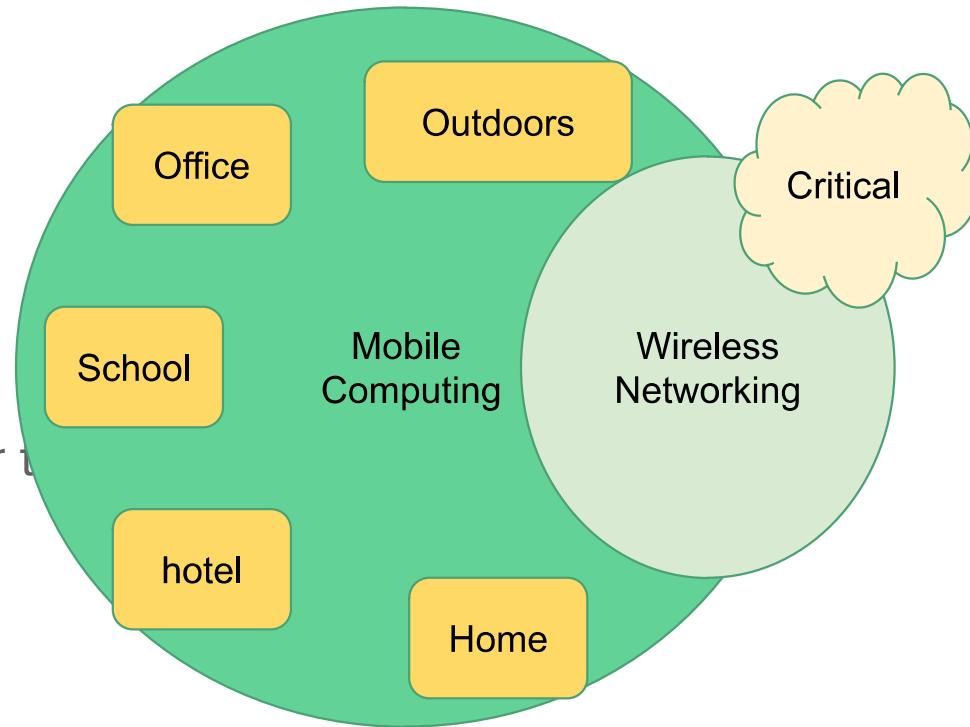
Mobile Computing

- Synonyms
 - Ubiquitous computing, Nomadic computing
 - Definition
 - “Ability to compute anywhere and anytime”
 - Mobile Communication and Computing
 - User can be moving
-

Mobile Computing vs. Wireless Networking

Mobile Computing vs. Wireless Networking

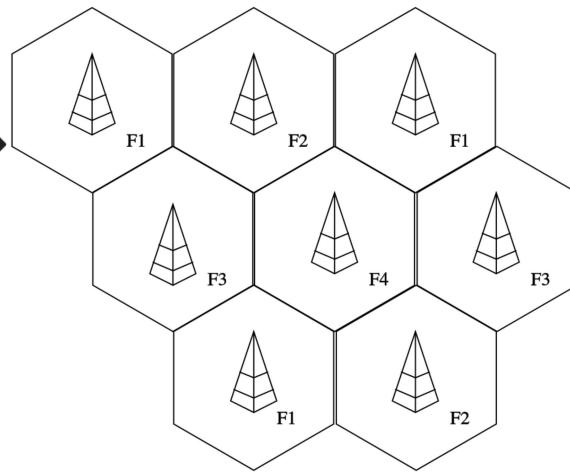
- Mobile Computing
 - Access information and remote computational resources
- Wireless Networking
 - Provide infrastructure for the above services



Kinds of Wireless Networks



WLAN



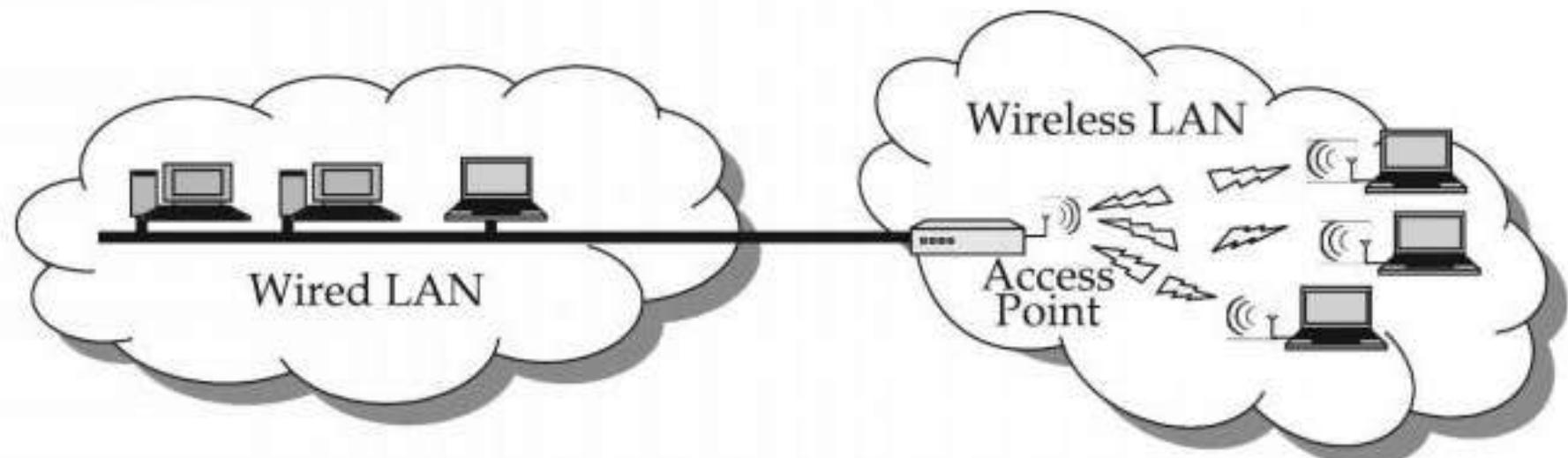
CWN



PAN

Type 1 - WLans

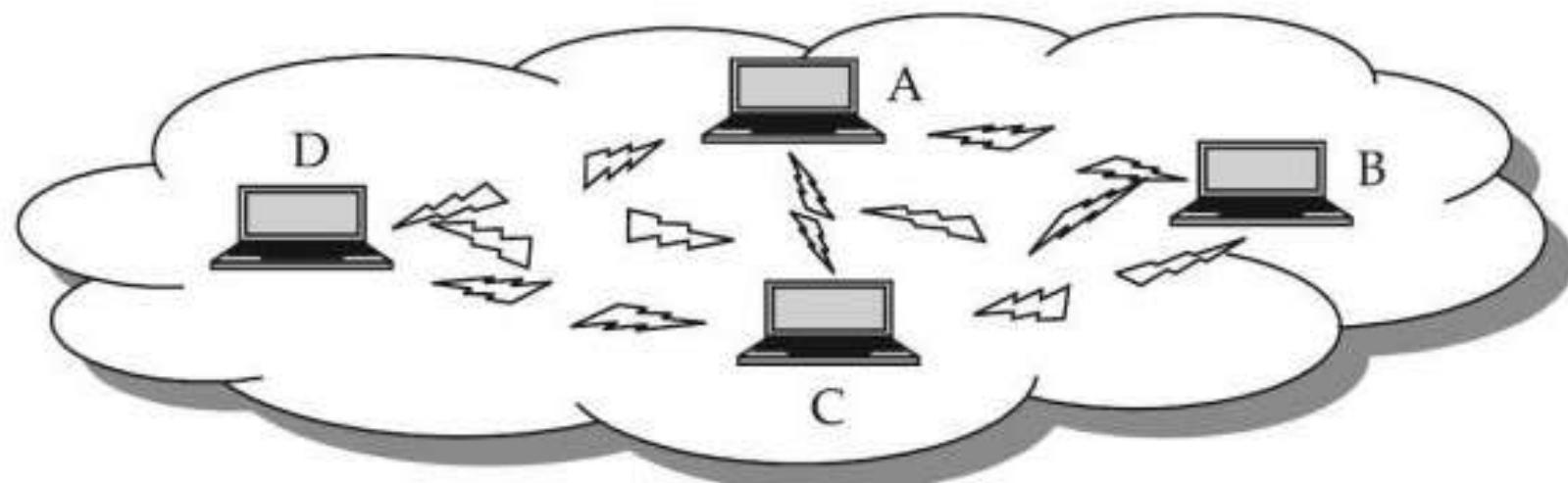
Fixed Infrastructure



Characteristics of Type 1 - WLans

- Single hop access to network via access point (AP)
 - Important: Only the last hop is wireless via the AP
 - AP - Bridge between the wired network and the wireless network
 - Requires authentication - eg. password
 - It depends on the IEEE 802.11 Wireless LAN Protocol
-

Adhoc Networks



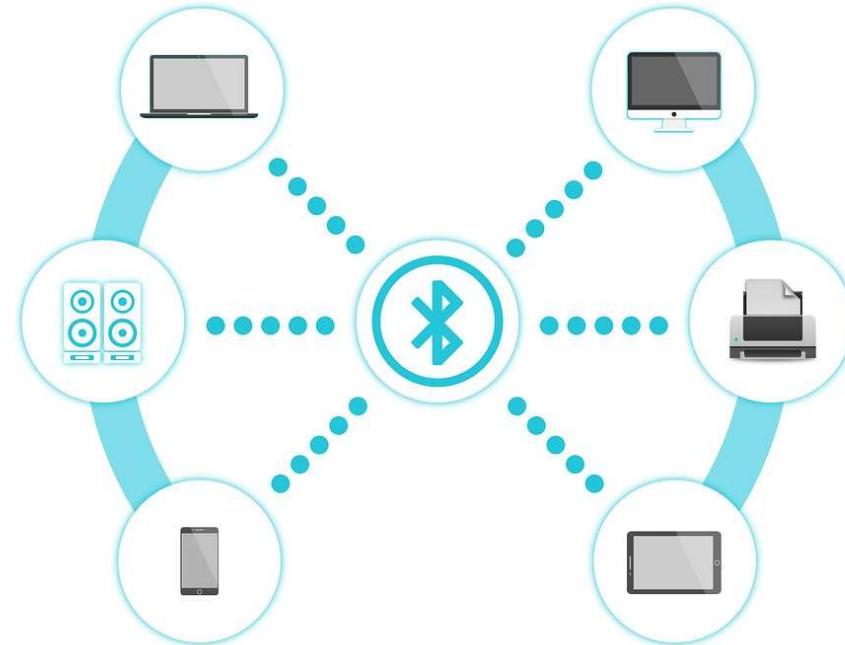
Characteristics of Adhoc Networks

- Multi-hop wireless communication network
 - Does not required fixed infrastructure
 - Intermediate nodes - Wireless connections - hops
 - Eg. Node A can get signal from C via B or D or directly
 - Recent advances: The Bluetooth Technology
-



Bluetooth Technology

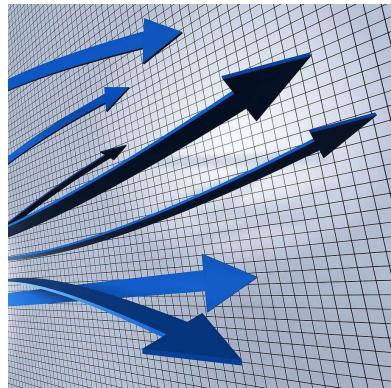
- Connect with printers, cameras, scanners and computers
- Bluetooth is a popular choice for ad-hoc wireless networks
- Easy to enable networks
- Automatic connections
- PANs - piconets
- Ad-hoc networks - scatternets



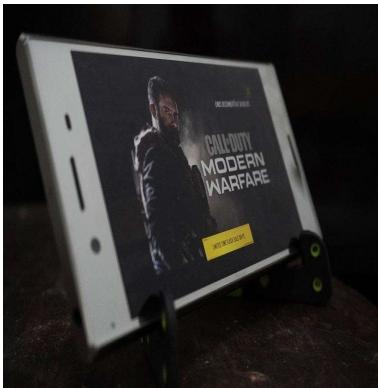
Mobile Computing Applications



Productivity



Investments



Games

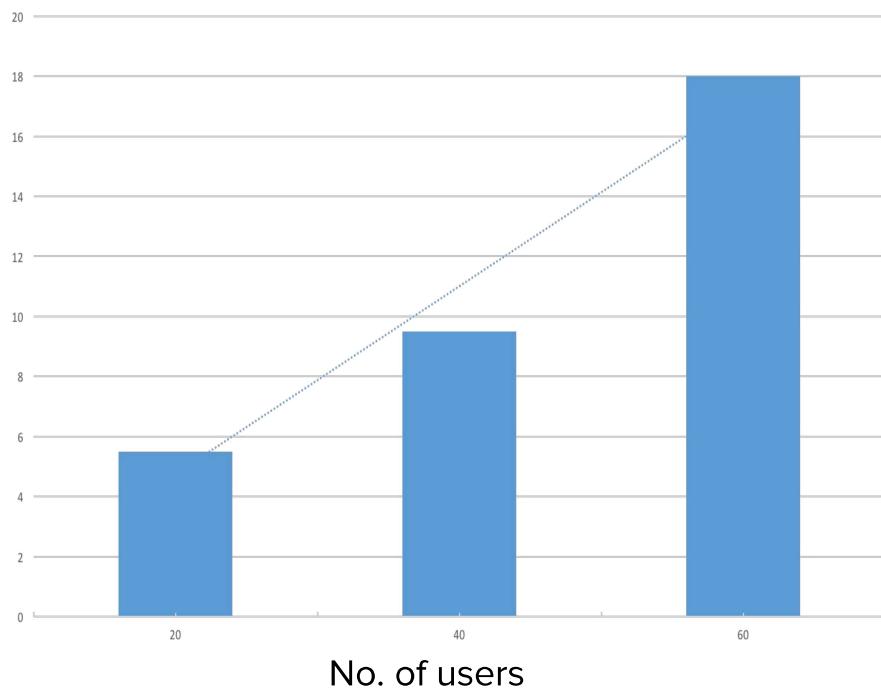


Messaging

Advantages

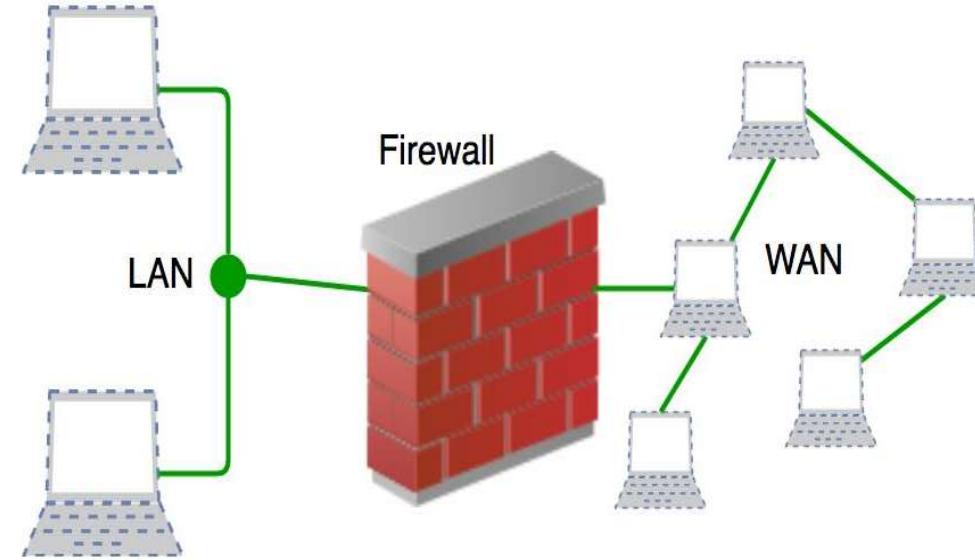


Ease of deployment



Scalability

Major Challenge - Data Security



OCW: Unit 4

UNIT 4

Topics to be covered:

TCP/IP,

IP Address and significance,

Internet and its Protocols (http, https, ftp) - Security related topics

TCP/IP protocol family

- ❖ IP : Internet Protocol
 - UDP : User Datagram Protocol
 - RTP, traceroute
 - TCP : Transmission Control Protocol
 - HTTP, FTP, ssh

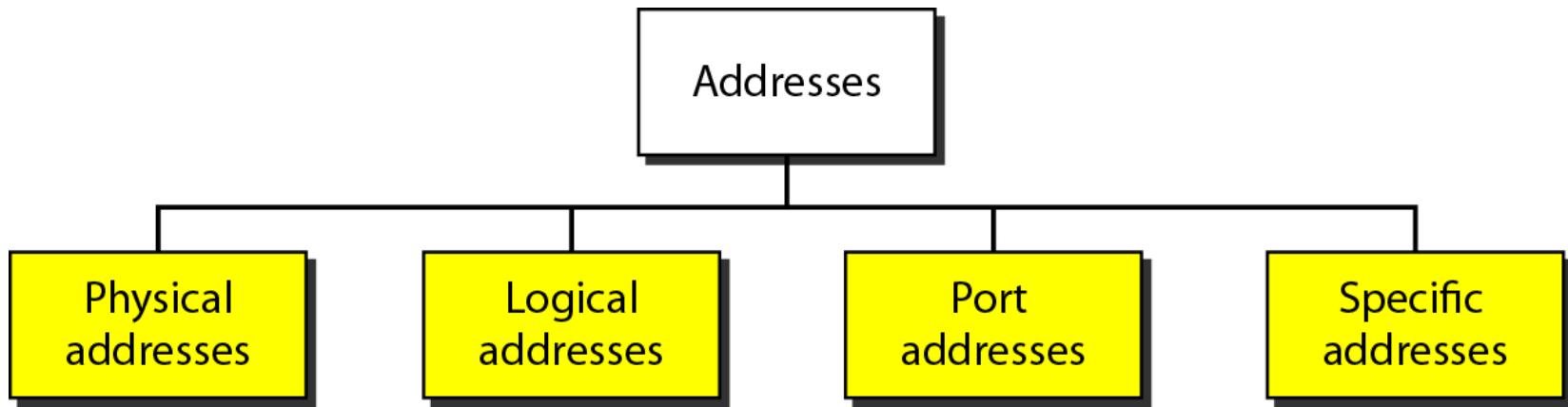
What is an internet?

- ❖ A set of *interconnected networks*
- ❖ The Internet is the most famous example
- ❖ Networks can be completely different
 - Ethernet, ATM, modem, ...
 - (TCP/)IP is what links them

What is an internet? (cont)

- ❖ *Routers* are devices on multiple networks that pass traffic between them
- ❖ Individual networks pass traffic from one router or endpoint to another
- ❖ TCP/IP hides the details as much as possible

Addresses in TCP/IP



Interface Between the Process: API

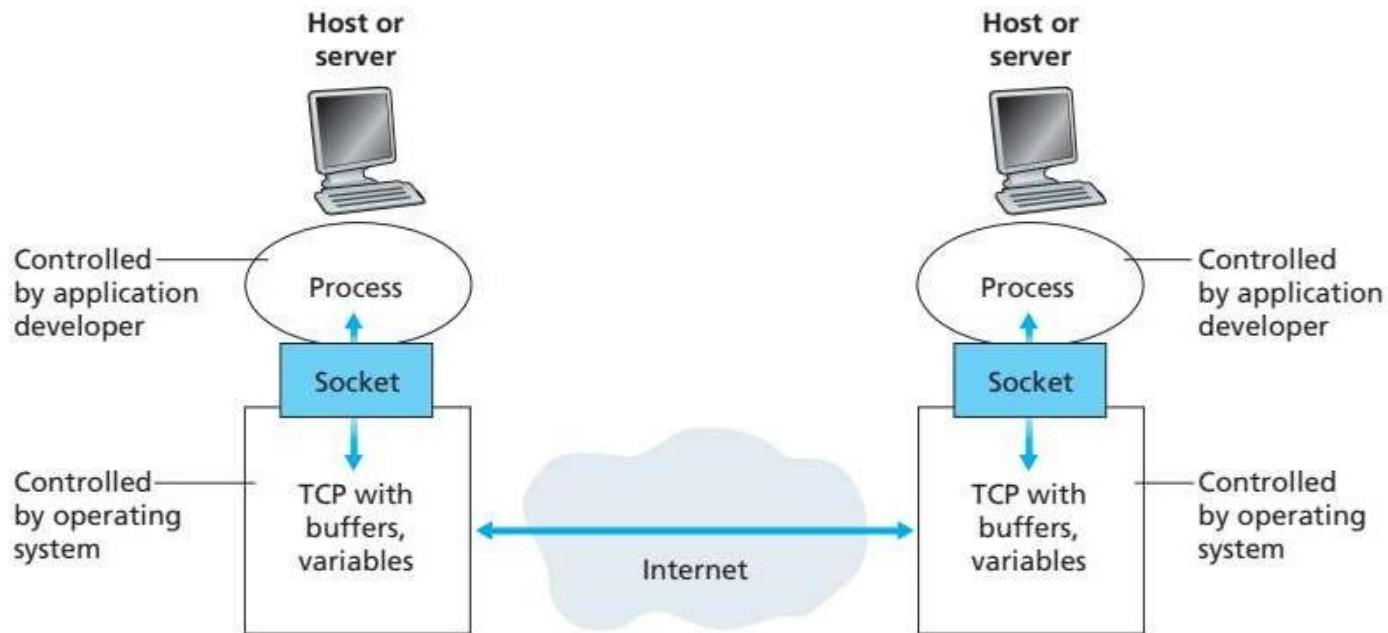


Figure 2.3 ♦ Application processes, sockets, and underlying transport protocol

Services of Transport Layer

- **Reliable data transfer:** Guaranteed data delivery service.
- **Throughput**
- **Timing:** for example, it is guaranteed that a packet will be delivered no more than 100 msec later.
- **security:** end-point authentication, encryption and decryption.

Requirements of Applications

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1 Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Instant messaging	No loss	Elastic	Yes and no

Figure 2.4 ♦ Requirements of selected network applications

Transport protocols

- Transmission Control Protocol (TCP)
 - Connection oriented service: handshaking, full-duplex connection
 - Reliable data transfer service: packets get delivered without error and in proper order.
 - Congestion control
 - User Datagram Protocol (UDP)
 - Connectionless
 - Unreliable data transfer service.
 - No congestion control
-
- can often provide satisfactory service to time-sensitive applications,
 - cannot provide any timing or throughput guarantees

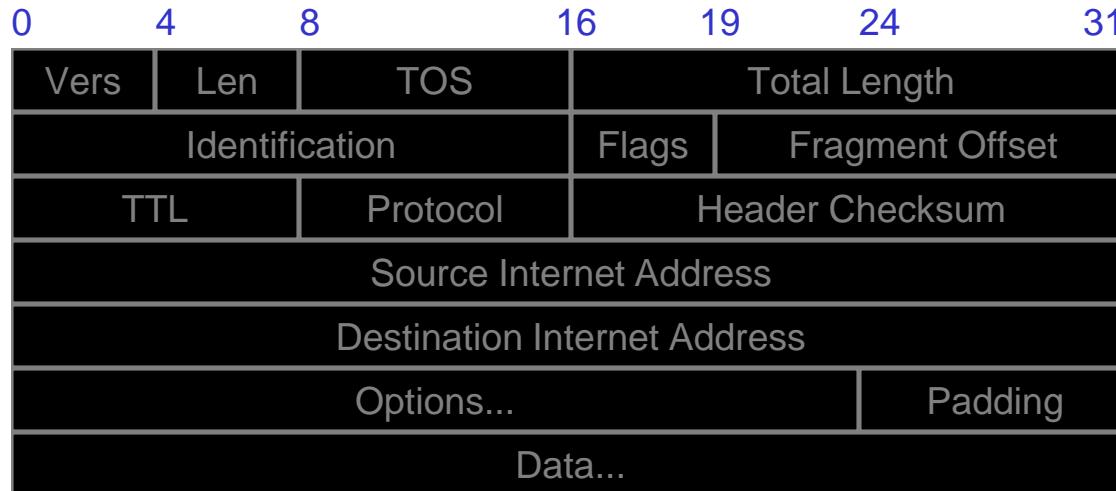
Applications

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

IP

- ❖ Responsible for end to end transmission
- ❖ Sends data in individual packets
- ❖ Maximum size of packet is determined by the networks
 - Fragmented if too large
- ❖ Unreliable
 - Packets might be lost, corrupted, duplicated, delivered out of order

IP Datagram



Field	Purpose
Vers	IP version number
Len	Length of IP header (4 octet units)
TOS	Type of Service
T. Length	Length of entire datagram (octets)
Ident.	IP datagram ID (for frag/reassembly)
Flags	Don't/More fragments
Frag Off	Fragment Offset

Field	Purpose
TTL	Time To Live - Max # of hops
Protocol	Higher level protocol (1=ICMP, 6=TCP, 17=UDP)
Checksum	Checksum for the IP header
Source IA	Originator's Internet Address
Dest. IA	Final Destination Internet Address
Options	Source route, time stamp, etc.
Data...	Higher level protocol data

You just need to know the IP addresses, TTL and protocol #

Routing

- ❖ How does a device know where to send a packet?
 - All devices need to know what IP addresses are on directly attached networks
 - If the destination is on a local network, send it directly there

Routing (cont)

- ❖ If the destination address isn't local
 - Most non-router devices just send everything to a single local router
 - Routers need to know which network corresponds to each possible IP address

Allocation of addresses

- ❖ Controlled centrally by ICANN
 - Fairly strict rules on further delegation to avoid wastage
 - Have to demonstrate actual need for them
- ❖ Organizations that got in early have bigger allocations than they really need

UNIT 4

Topics to be covered:

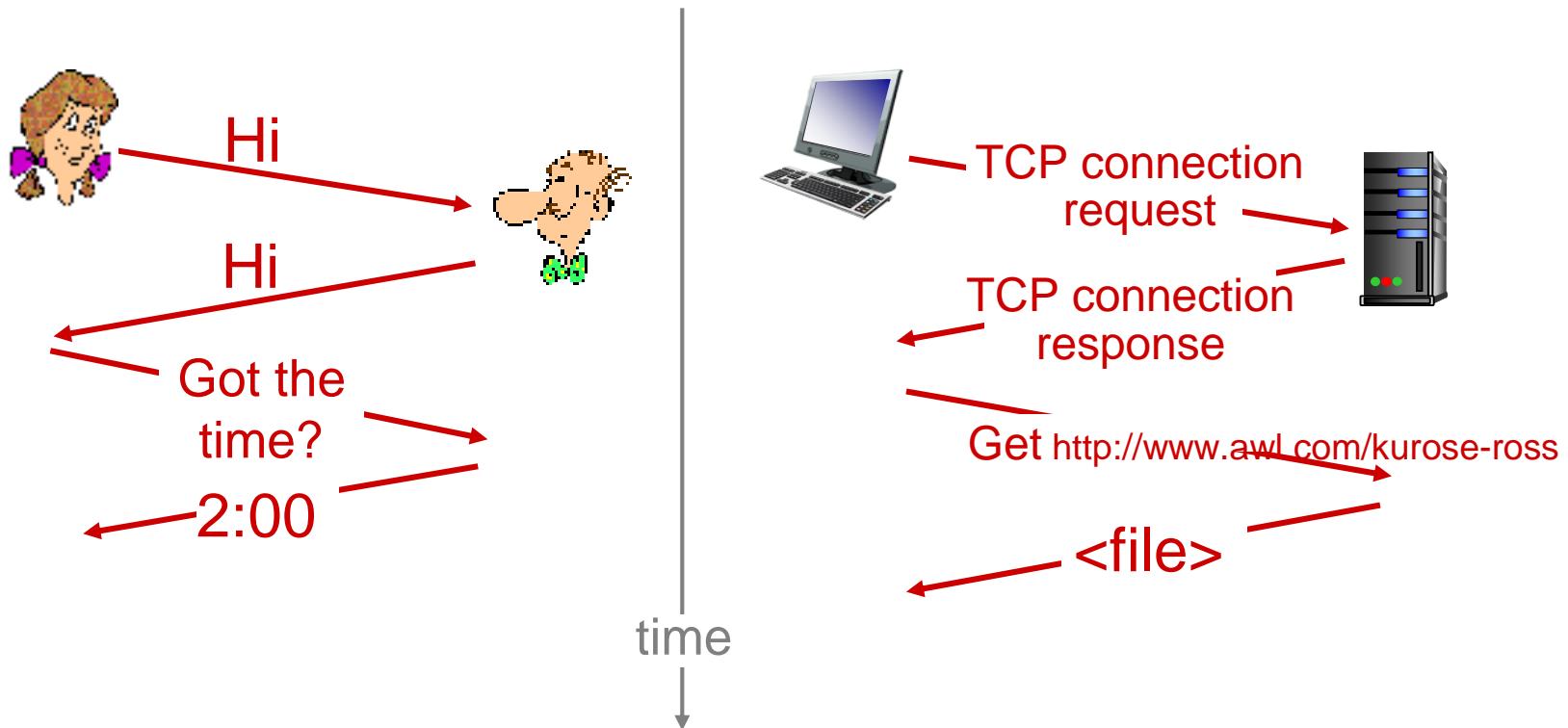
TCP/IP,

IP Address and significance,

Internet and its Protocols (http, https, ftp) - Security related topics

What's a protocol?

a human protocol and a computer network protocol:



Q: other human protocols?

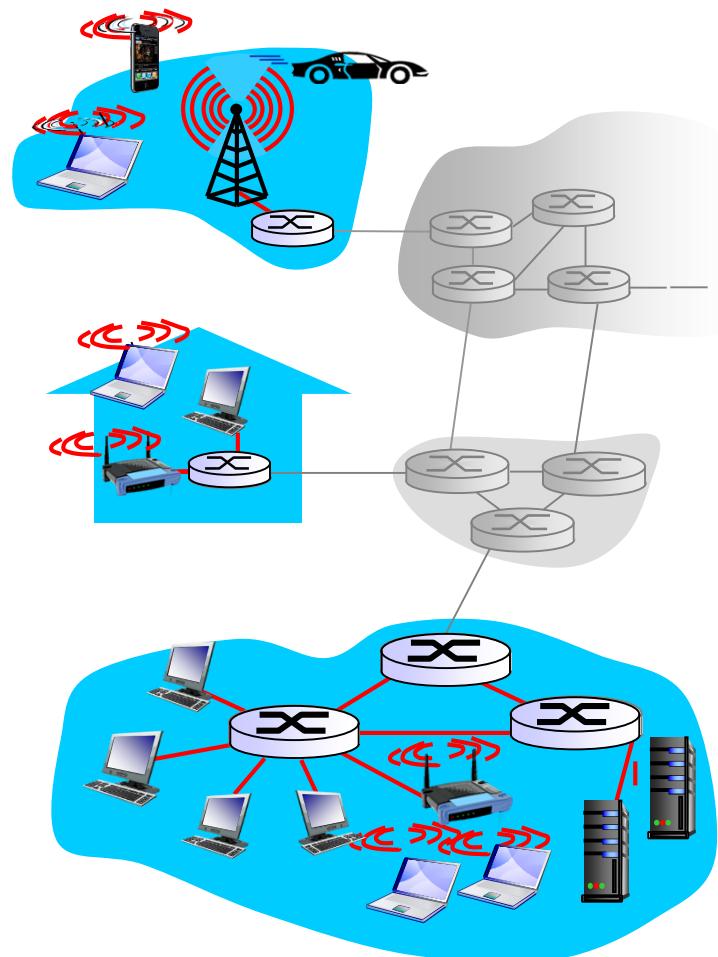
Access networks and physical media

Q: How to connect end systems to edge router?

- ❖ residential access nets
- ❖ institutional access networks (school, company)
- ❖ mobile access networks

keep in mind:

- ❖ bandwidth (bits per second) of access network?
- ❖ shared or dedicated?



Network security

- ❖ field of network security:
 - how bad guys can attack computer networks
 - how we can defend networks against attacks
 - how to design architectures that are immune to attacks
- ❖ Internet not originally designed with (much) security in mind
 - *original vision:* “a group of mutually trusting users attached to a transparent network” ☺
 - Internet protocol designers playing “catch-up”
 - security considerations in all layers!

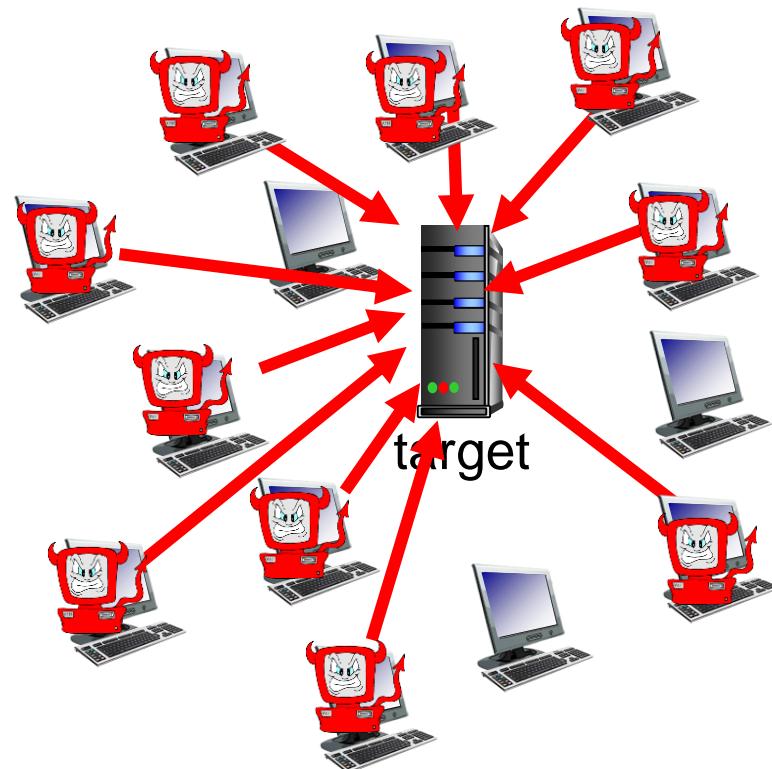
Bad guys: put malware into hosts via Internet

- ❖ malware can get in host from:
 - *virus*: self-replicating infection by receiving/executing object (e.g., e-mail attachment)
 - *worm*: self-replicating infection by passively receiving object that gets itself executed
- ❖ **spyware malware** can record keystrokes, web sites visited, upload info to collection site
- ❖ infected host can be enrolled in **botnet**, used for spam, DDoS attacks

Bad guys: attack server, network infrastructure

Denial of Service (DoS): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

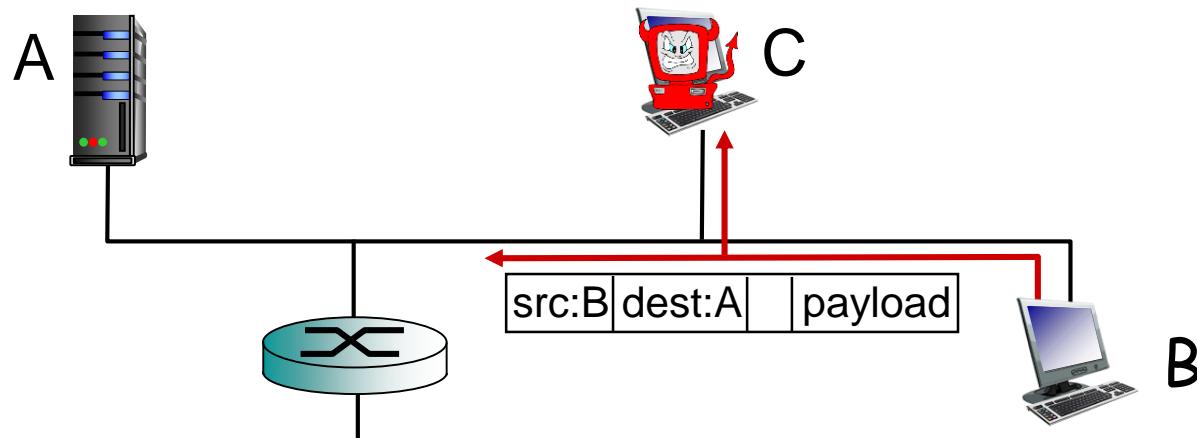
1. select target
2. break into hosts around the network (see botnet)
3. send packets to target from compromised hosts



Bad guys can sniff packets

packet “sniffing”:

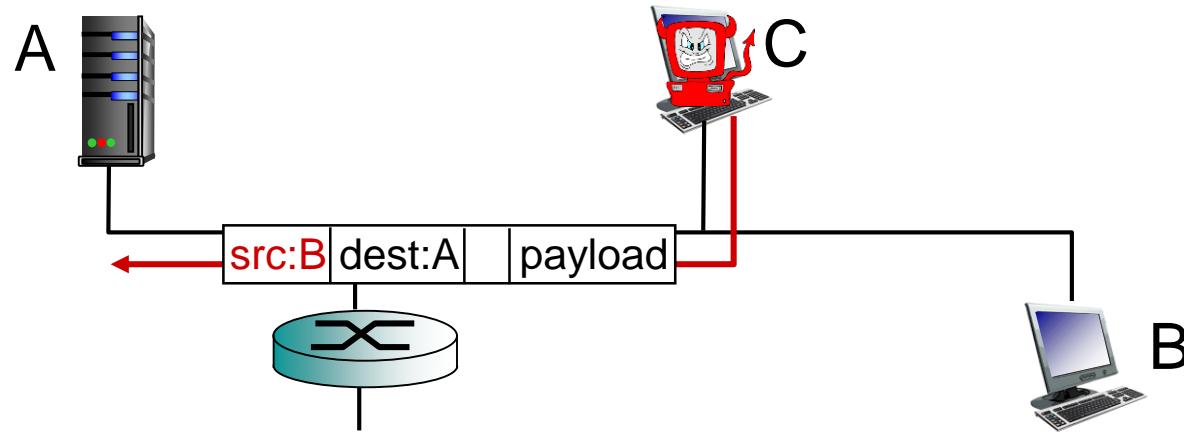
- broadcast media (shared ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g., including passwords!) passing by



- ❖ wireshark software used for end-of-chapter labs is a (free) packet-sniffer

Bad guys can use fake addresses

IP spoofing: send packet with false source address





Overview of Computers

Module-IV

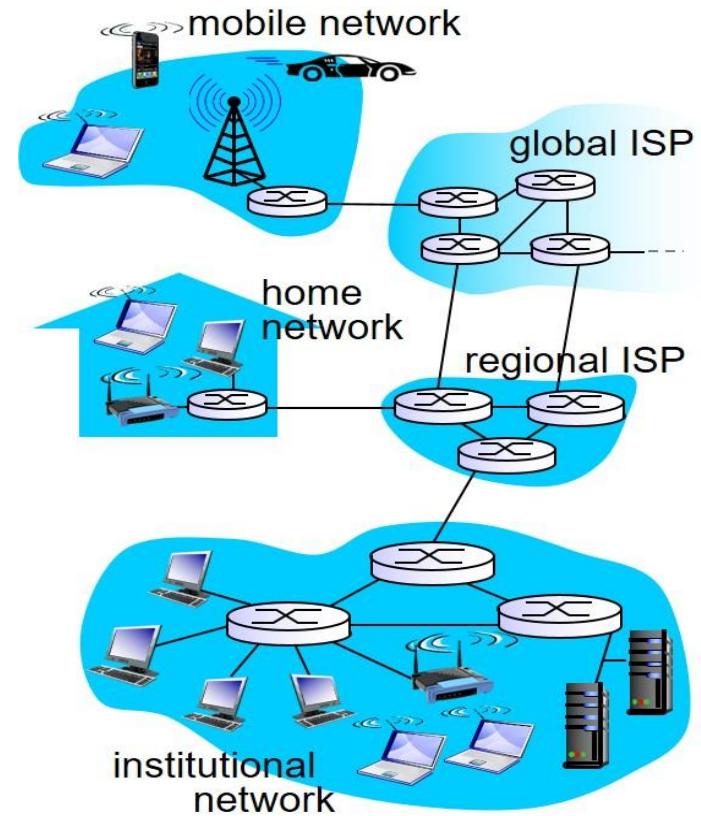
Computer Networks

Contents

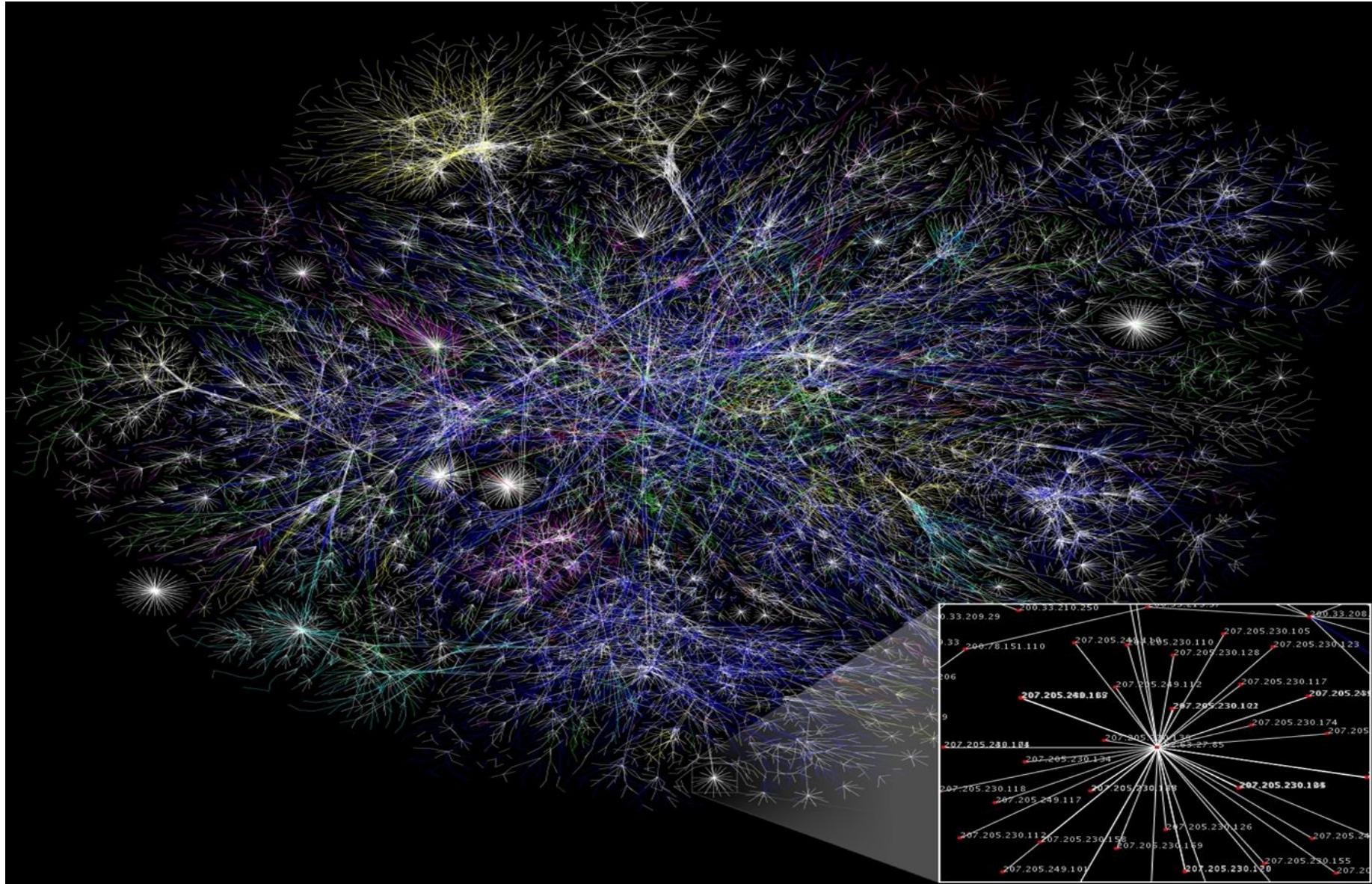
- Introduction to Computer Networks
- Networks and Types of Networks
 - Protocol Layers
 - Ethernet

What is a Network?

- A network is an interconnection of devices.
- The computers/laptops connected to the network are known as end systems or hosts.
- The digital data is fragmented into packets.

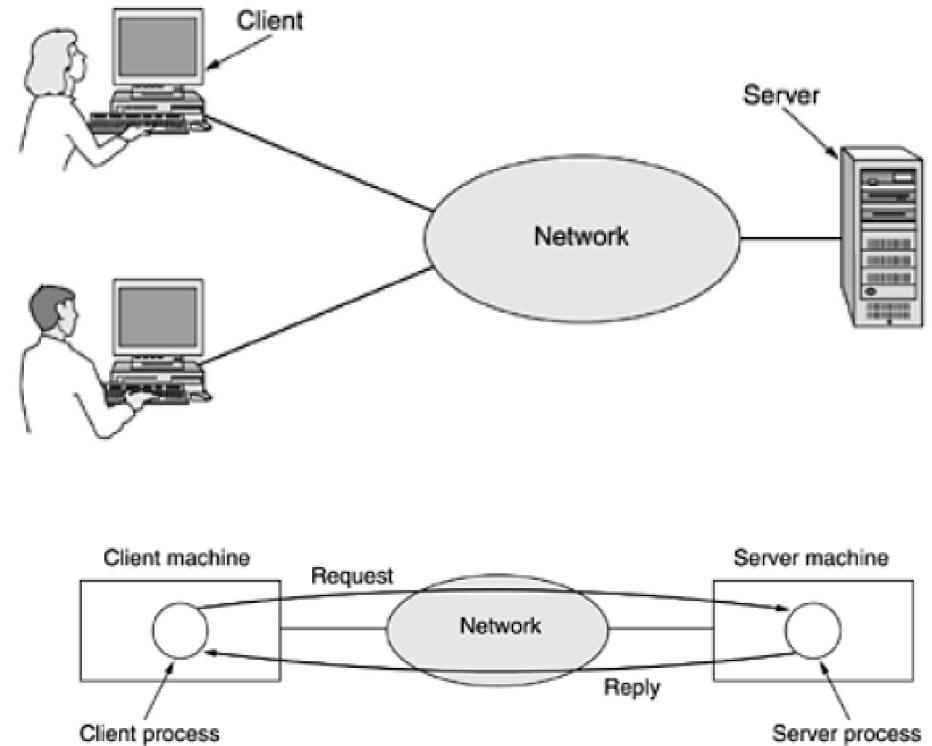


Internet - Snapshot of an Example Network



Uses of Computer network

- Business applications
 - Resource sharing
 - powerful medium of communication (email and online document preparation)
 - Video conferencing
 - Doing business electronically with other companies (ex: Isuzu).
 - Doing business with consumer (online market).



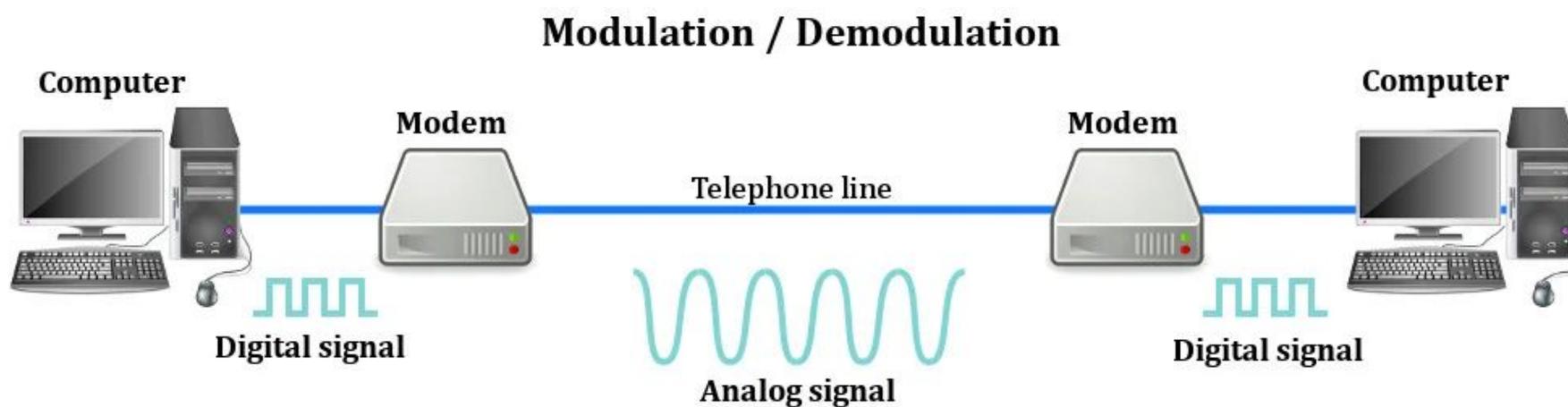
Uses of Computer network

- Home applications
 - Why do people buy computer for home use?
 - Earlier days it is for word processing and gaming , now for “Internet access”
 - Internet provides access to **remote information**, **person- to-person communication**, **entertainment**, **e-commerce**.

Tag	Full name	Example
B2C	Business-to-consumer	Ordering books on-line
B2B	Business-to-business	Car manufacturer ordering tires from supplier
G2C	Government-to-consumer	Government distributing tax forms electronically
C2C	Consumer-to-consumer	Auctioning second-hand products on line
P2P	Peer-to-peer	File sharing

Network Essentials

- **Modem**
- “**Modulator and demodulator**”, is a hardware device that converts data into a format suitable for a transmission medium so that it can be transmitted from one computer to another.
- **Ethernet**
- System for connecting the number of computers to form a LAN.



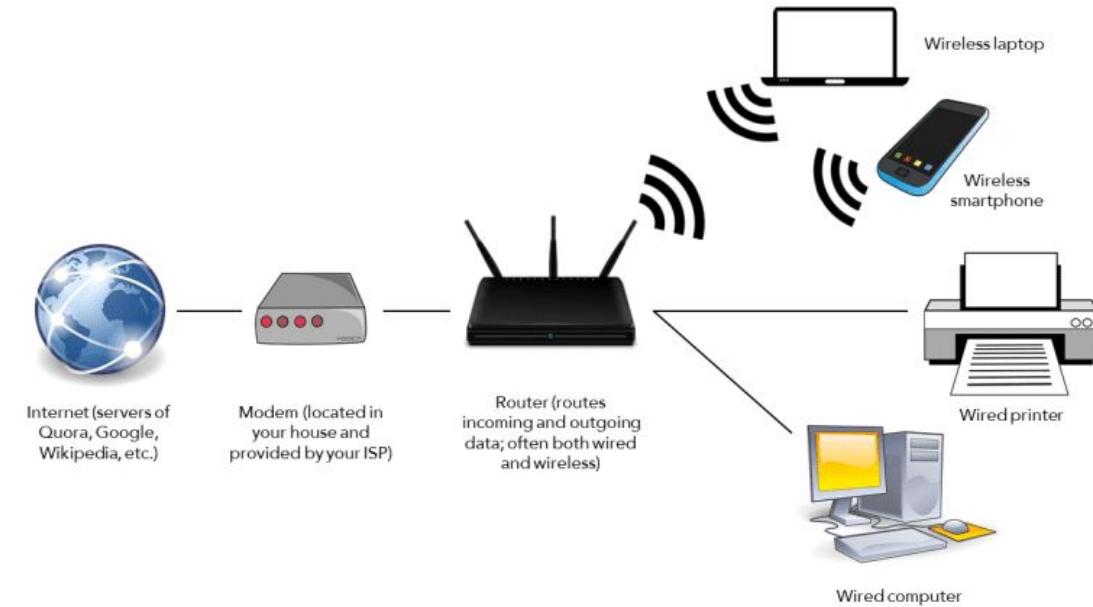
Network Essentials

Router

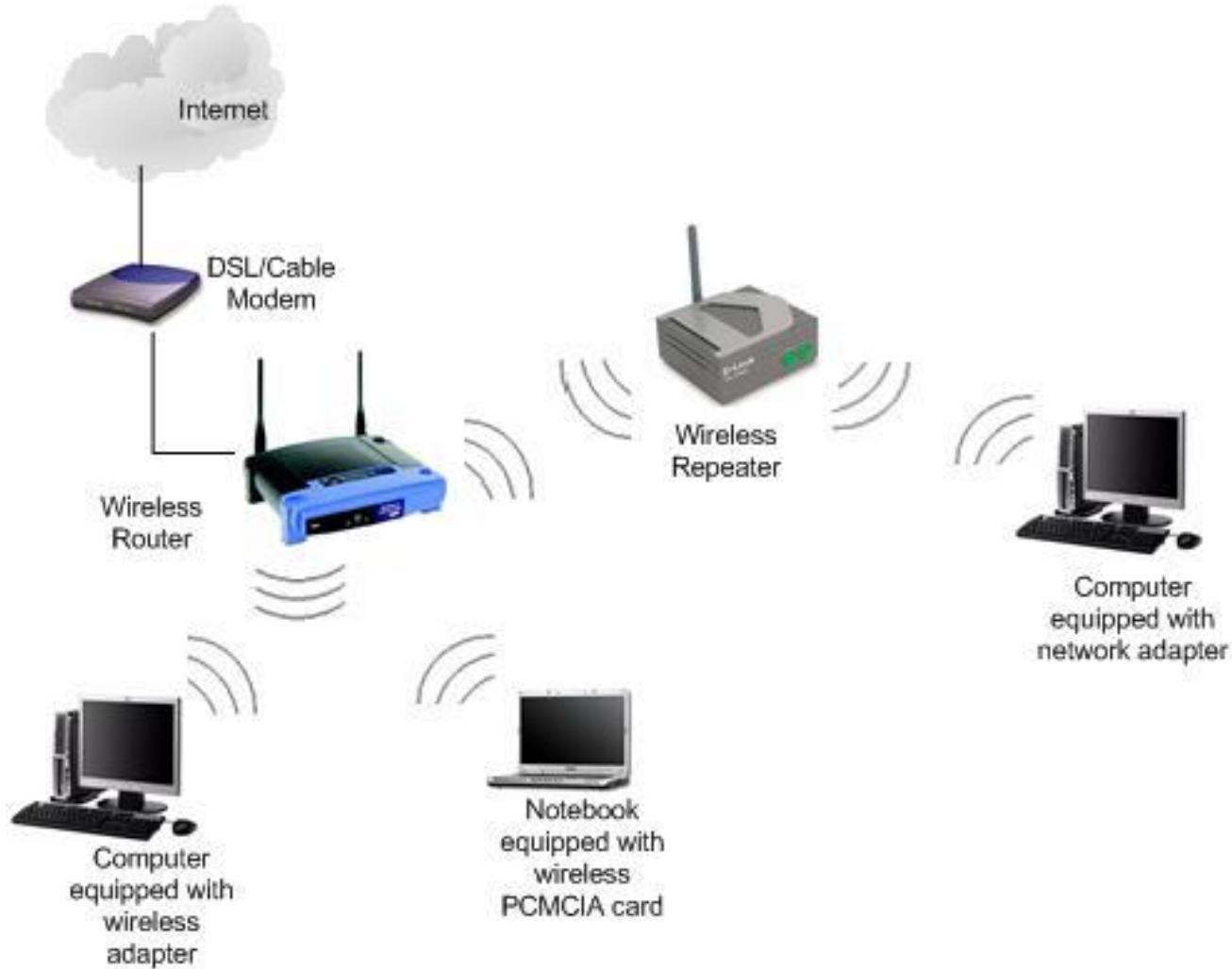
- A router is a device that forwards data packets along networks.
- A router is connected to at least two networks, commonly two LANs or WANs or a LAN and its ISP's network.

Repeater

- A **network** device used to regenerate or replicate a signal.
- **Repeaters** are used in transmission systems to regenerate analog or digital signals distorted by transmission loss.
- Analog **repeaters** frequently can only amplify the signal while digital **repeaters** can reconstruct a signal to near its original quality.

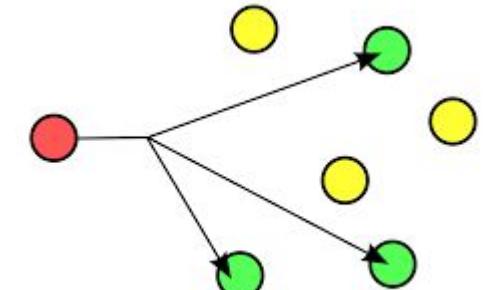
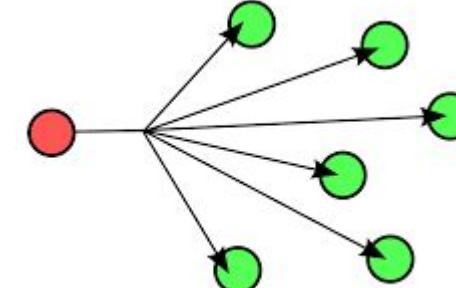
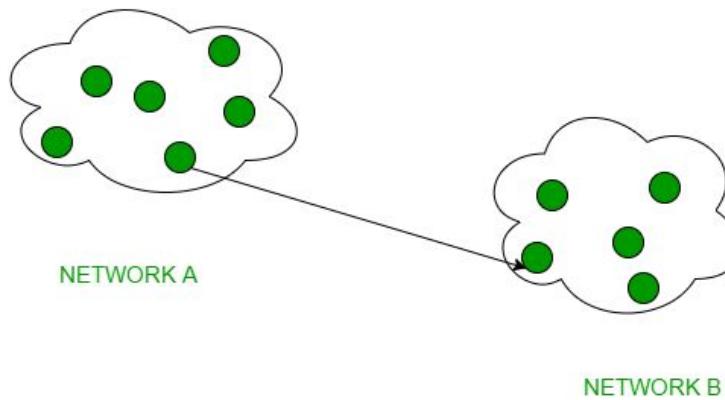


Wireless Network



Classification of Networks

- Transmission technology:
 - Unicasting : transmission with exactly one sender and exactly one receiver
 - Broadcasting : information is intended to all hosts
 - Multicasting : information is intended for a subset of hosts in the network



Network Hardware: Classification

Interprocessor Distance	Processors located in same	
1 m	Square meter	Personal area network
10 m	Room	
100 m	Building	Local area network
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	
1000 km	Continent	Wide area network
10,000 km	Planet	The Internet

Classification of Networks:

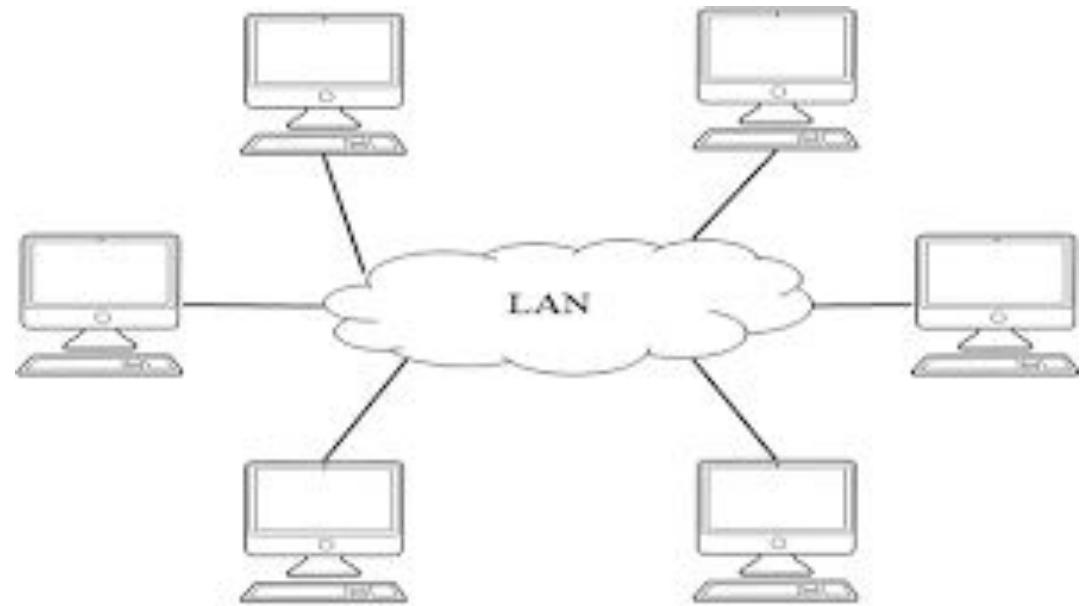
- Personal area networks (PANs)
 - Organized around an individual person, within a small office or residence.
 - Within the range of few meters
 - Notable example is Bluetooth
 - Watching movies on online streaming service to TV
 - With multiple uses within a same residence then, referred as Home Area Network (HAN).



Connecting peripherals to computer via Bluetooth

Classification of Networks:

- Local area networks (LANs)
 - Typically an individual office building: suitable for sharing resources (data storage and printers).
 - Range: It can reach few hundred meters, can be increased further using wireless repeaters.
 - Wireless LAN: WLAN



Privately owned network: wireless/wired connections.

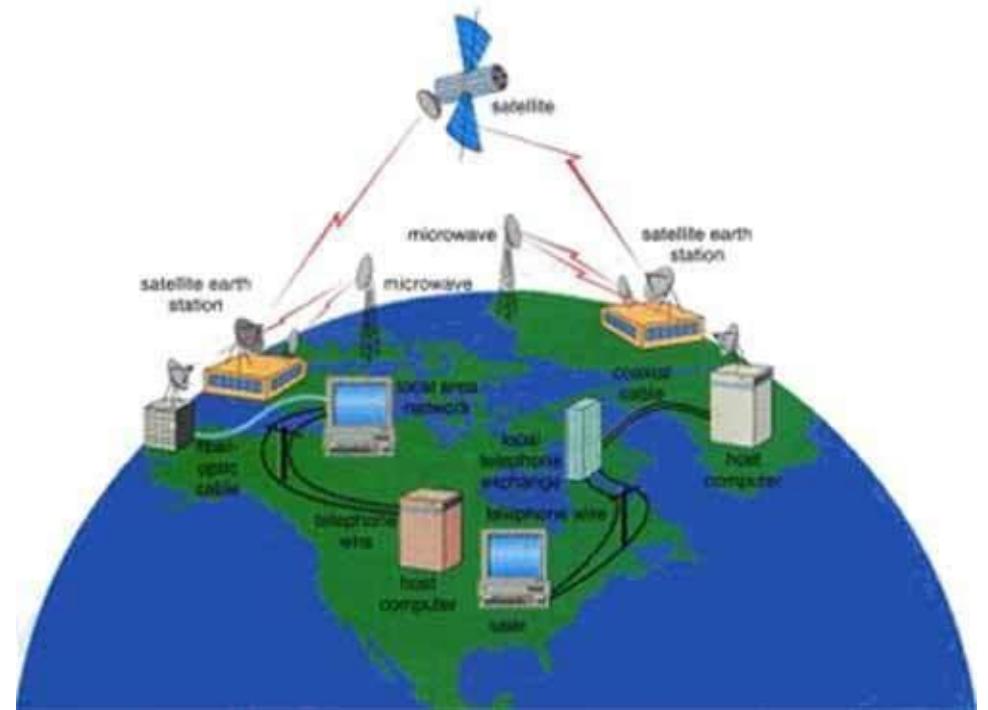
Classification of Networks:

- Metropolitan area networks (MANs)
 - Computer network across entire city, college campus or small region.
 - Referred as Campus Area Network (CAN).
 - Range: from several miles to tens of miles.
 - Connect several LANs together to form a bigger network.



Classification of Networks:

- Wide area networks (WANs)
 - Occupies a very large area, such as an entire country or the entire world
 - can contain multiple smaller networks, such as LANs or MANs
 - The most well-known WAN is the “Internet”



Physical (Transmission) Media

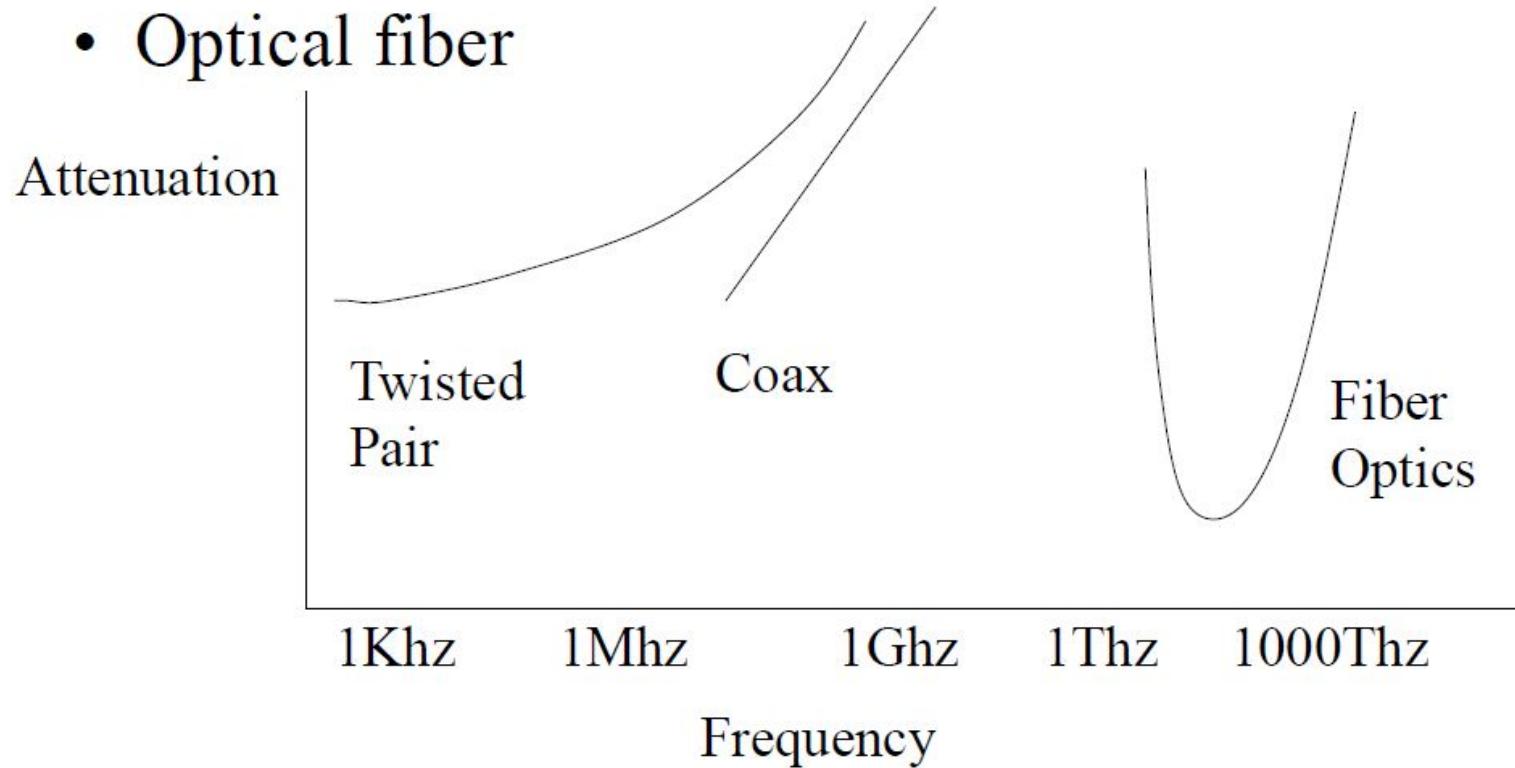
- Physical media fall into two categories: **guided media** and **unguided media**.
- With guided media, the waves are guided along a solid medium, such as a fiber-optic cable, a twisted-pair copper wire, or a coaxial cable.
- With unguided media, the waves propagate in the atmosphere and in outer space, such as in a wireless LAN or a digital satellite channel.

Examples:

- HFC uses combination of fibre cable and coaxial cable.
- DSL and Ethernet use copper wires.
- Mobile access network uses radio spectrum
- Cost involved?

Guided Transmission Media

- Twisted Pair
- Coaxial cable
- Optical fiber



Twisted Copper Wire

Least expensive and most commonly used in home and work environments.

Unshielded Twisted pairs (UTP): used for computer networks within a building (LANs), ranges from 10 MBPS to 10GBPS. Suffers from external EM interference.

Shielded Twisted Pairs (STP): More expensive and harder to handle

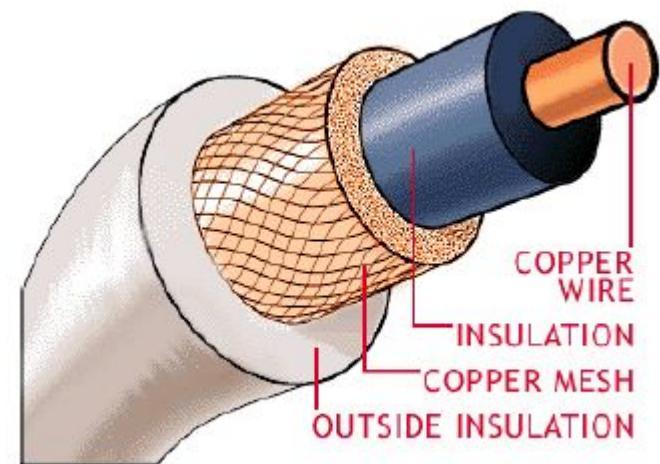
Coaxial Cable

Shielded and less susceptible to noise and attenuation than the twisted pair.

Applications:

Television distribution

Long distance telephone transmission



Fiber Optics

Have greater capacity

Smaller size and weight

Lower attenuation, not vulnerable to interference

Guided Media

Media	Network Type		Transmission Distance		Error Rates		Speed
	Type	Cost	Distance	Security	Rates		
Twisted Pair	LAN	Low	Short	Good	Low	Low-high	
Coaxial Cable	LAN	Mod.	Short-Mod	Good	Low	Low-high	
Fiber Optics	any	High	Mod.-long	V. Good	V.Low	High-V.High	

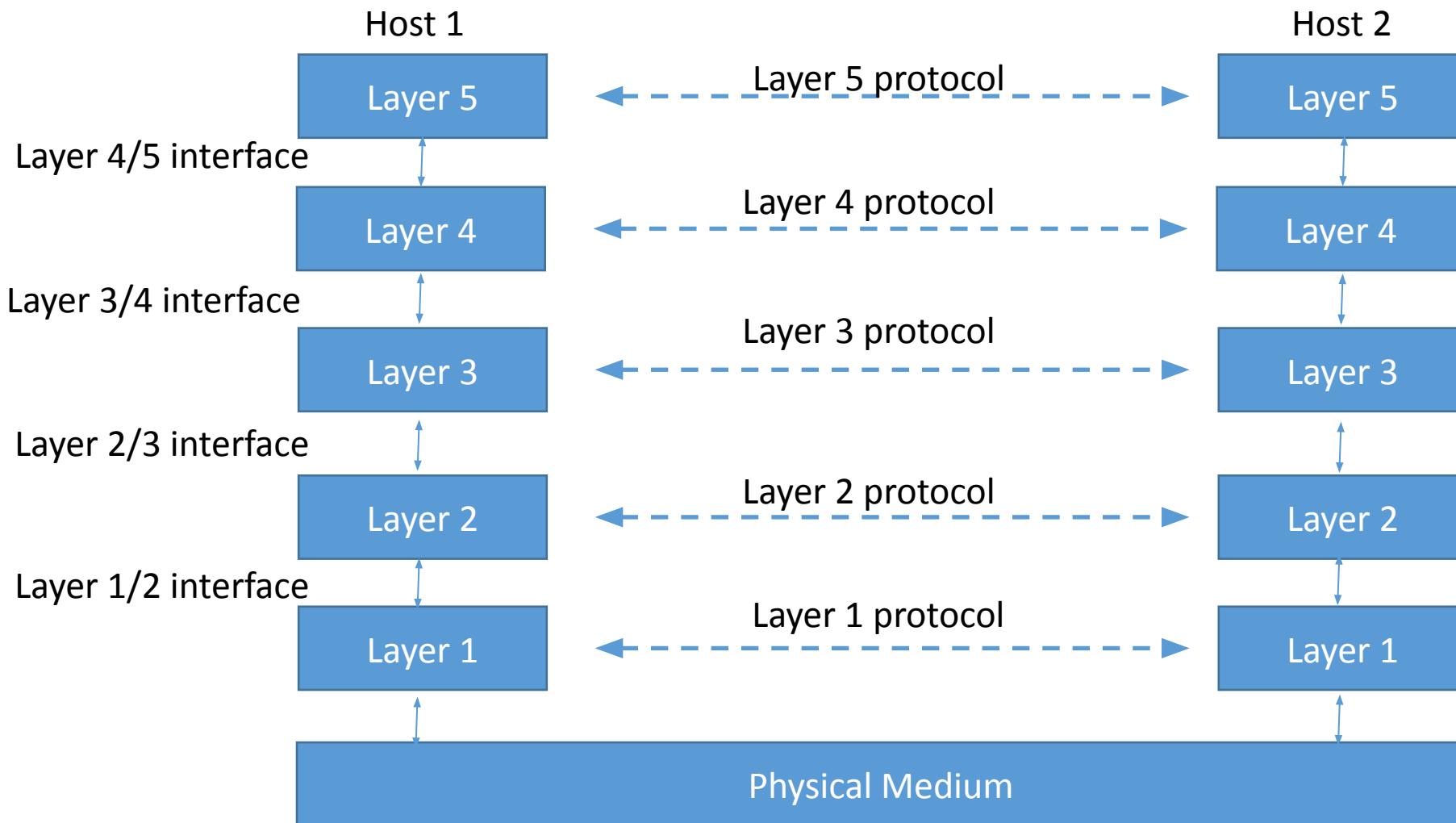
Unguided Media: Wireless transmission

- Transmission and reception via antenna!
- The characteristics of a radio channel depend significantly on the propagation environment and the distance over which a signal is to be carried
- Environmental considerations determine path loss and shadow fading.
- broadly classified into three groups:
- Over very short distance: spanning from ten to a few hundred meters
- Operate in the wide area, spanning tens of kilometers, cellular access technologies use wide-area radio channels
- wireless LAN technologies use local-area radio channels

Network Software

- Protocol
 - Is an agreement between the communicating parties on how communication is to proceed.
 - Violation of protocol will make communication more difficult, if not completely impossible.

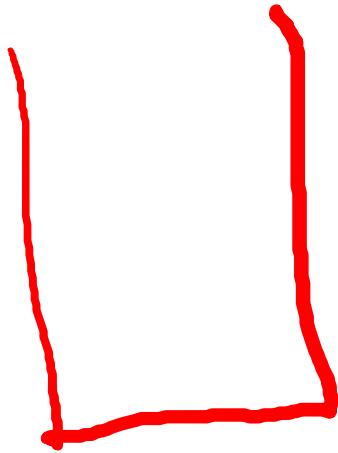
Layers, protocols, and interfaces



Layered Network Architecture

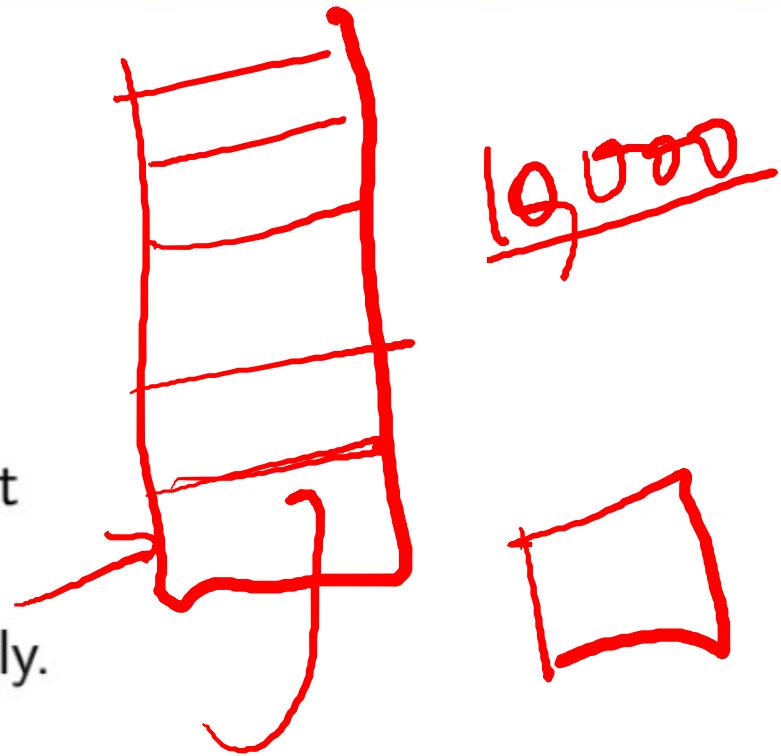
Why Layered Architecture?

- Organizing a network is a **big and complicated task**.
- Divide and conquer
- Example: Organization of an institute
 - academic section
 - finance section
 - administration section
 - procurement section



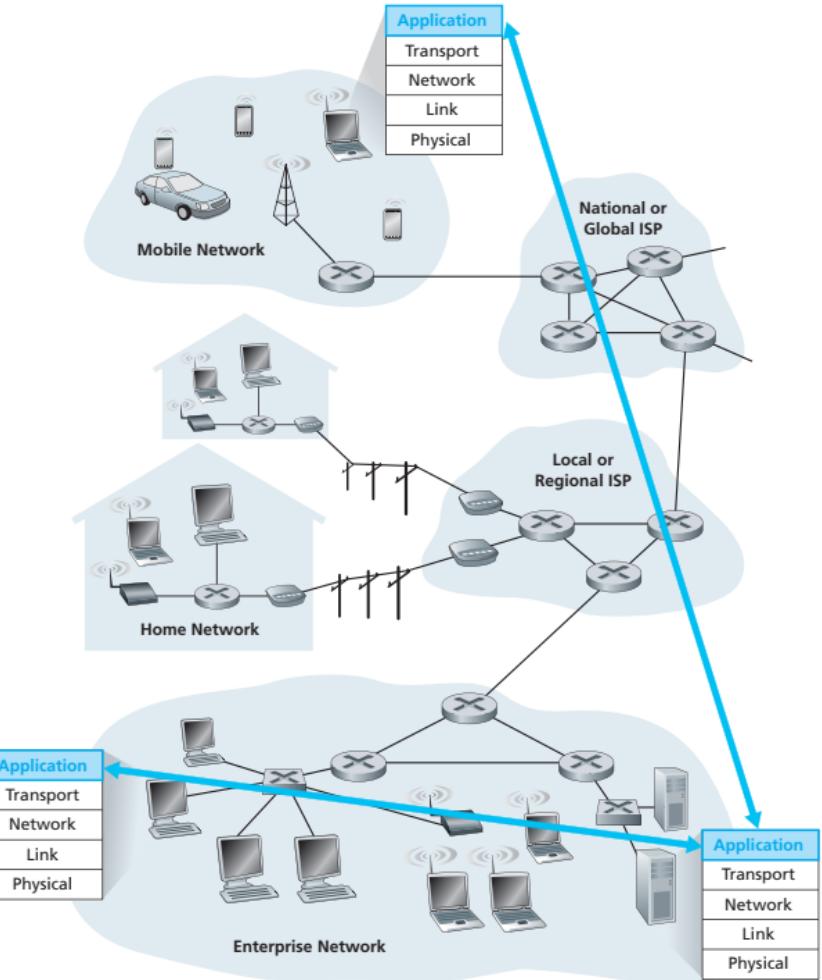
Advantages of Layered Architecture

- Divide the design issues into **small pieces**.
- A layer provides a **service** (set of actions) to the immediate higher layer.
- New technologies can be adopted in a layer without affecting other layers.
- Each layer can be analysed and tested independently.

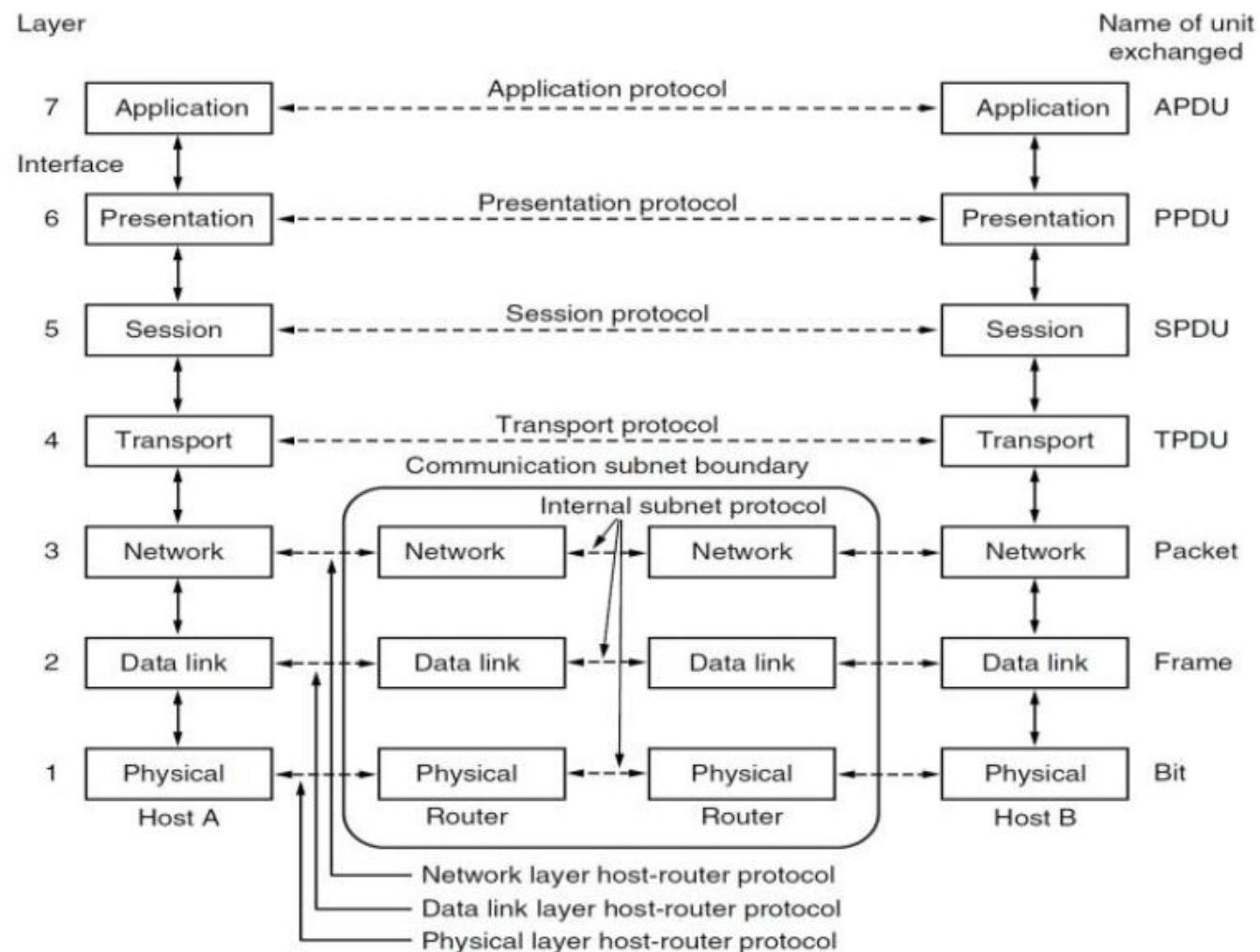


Open System Interconnection (OSI) Reference Model

- Developed by International Organization for Standardization (ISO)
- 7-layer model:
 - Application layer
 - Presentation layer
 - Session layer
 - Transport layer
 - **Network layer**
 - **Data-link layer**
 - **Physical layer**



Layers



Application Layer

- Consists of user programs, network applications that does work at hand
- Examples:
 - File transfer, Remote login, Mail, Web access
- Protocols: FTP, Telnet, Simple Mail Transfer Protocol(SMTP), HTTP.

Presentation Layer

- Concerned with syntax and semantics of information transmitted
 - Translation
 - Encoding data: Data compression/conversion, encryption and decryption
-

Session Layer

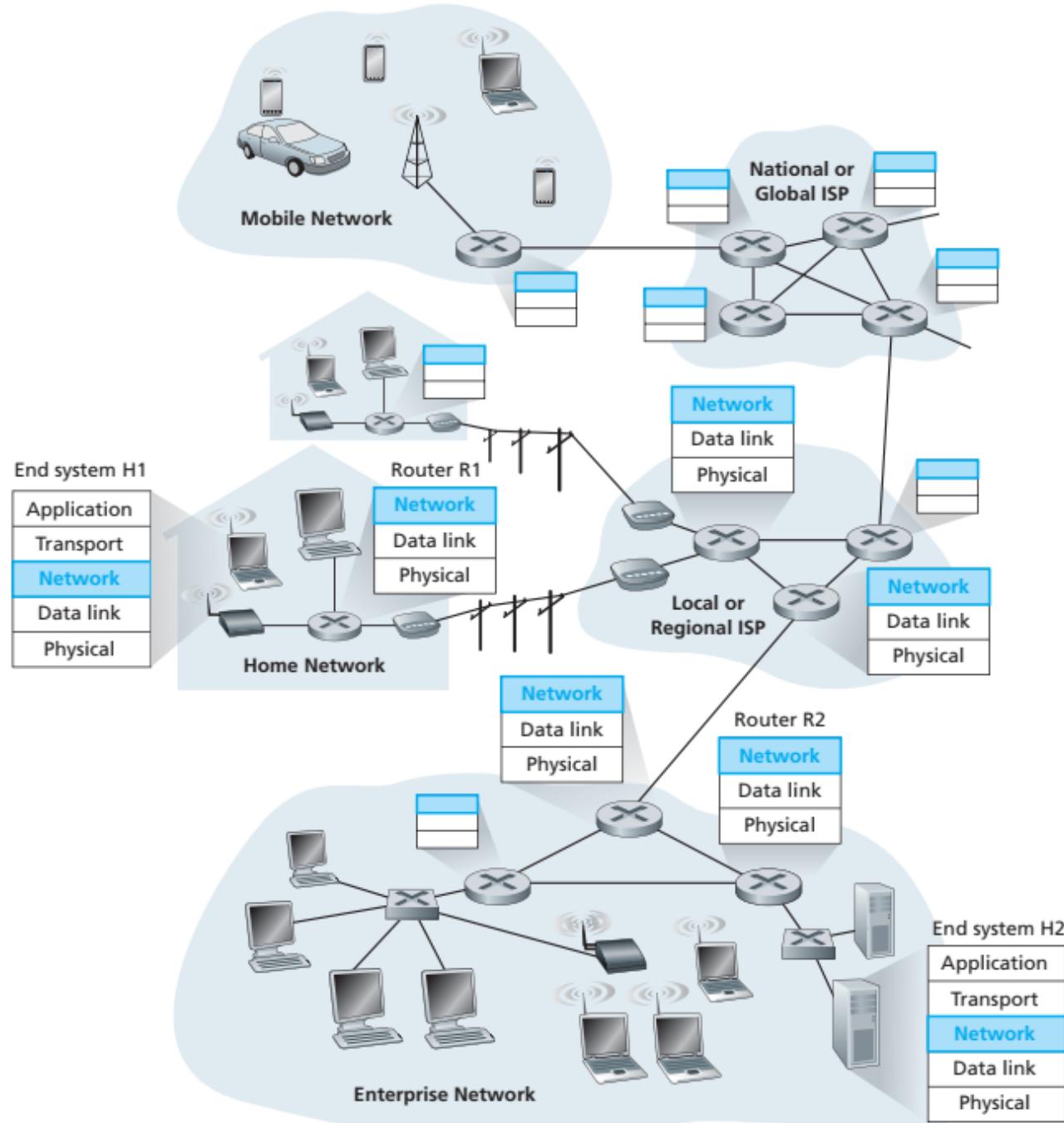
- Allows to establish a session between peers
- Dialogue control: Session can allow bidirectional traffic or only unidirectional traffic.
- Token management: In some protocols, it is required that both sides do not attempt same operation at same time.
Session layer provides tokens to perform such actions
- Synchronization: Pausing and resuming a download.

Transport Layer

- Connection-oriented services to applications
 - flow control
 - guaranteed delivery of messages to destination
- Ensures data delivery is
 - error-free
 - in sequence
 - no loss, duplication and corruption of packets

Network Layer

- Interface between host and network
- Routing
- Congestion and deadlock
- Internetworking



Data-Link Layer and Physical Layer

- **Data-link layer**
 - Takes packet from network layer and moves it to the next router
 - error-free delivery: computes error detection information
- **Physical layer**
 - Controls transmission into the network cable.
 - Defines electrical signals.

Internet Protocol Stack

- Application layer
- Transport layer
- Network layer
- Data-link layer
- Physical layer

Encapsulation

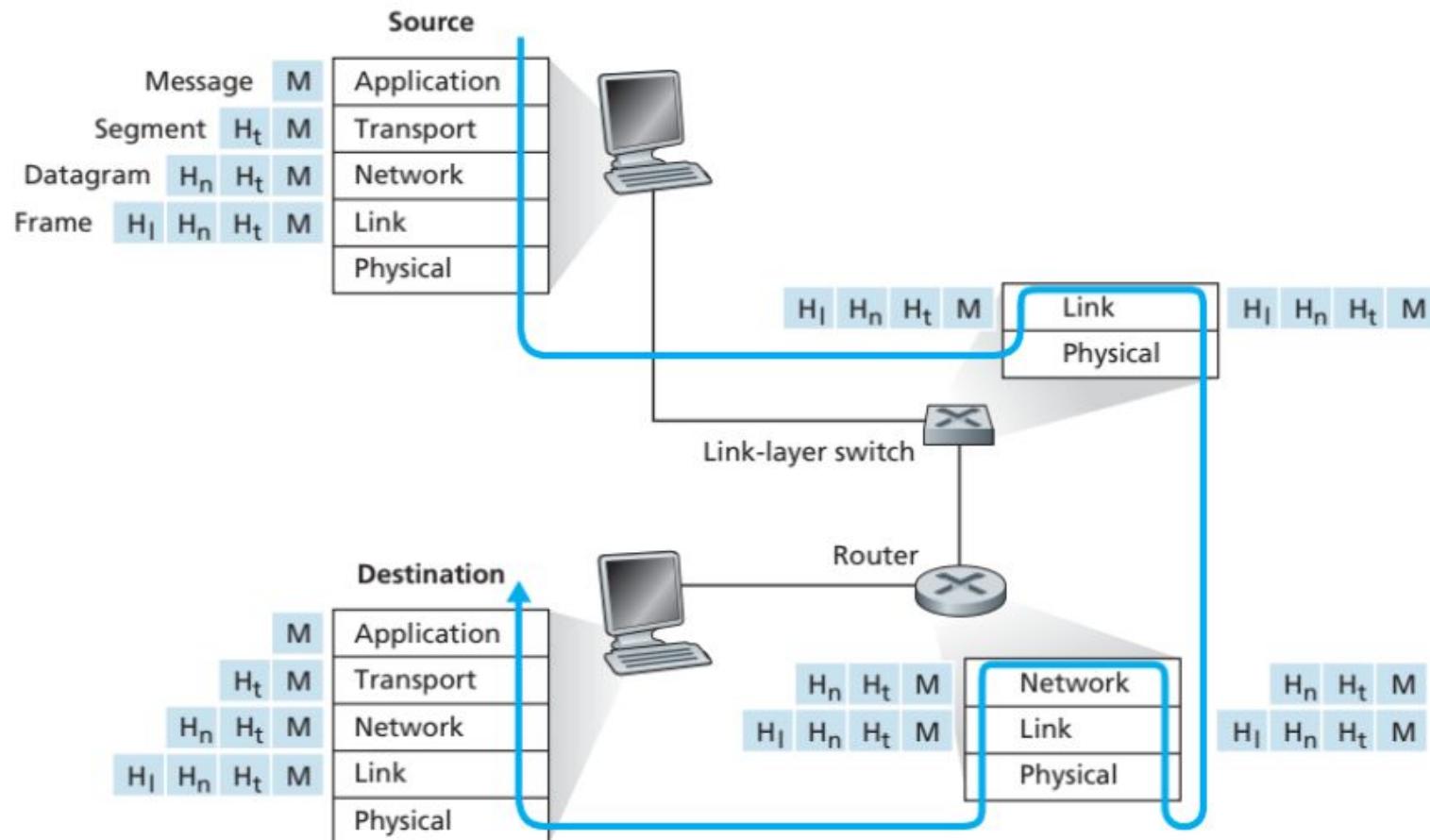


Figure 1.24 ♦ Hosts, routers, and link-layer switches; each contains a different set of layers, reflecting their differences in



Overview of Computers

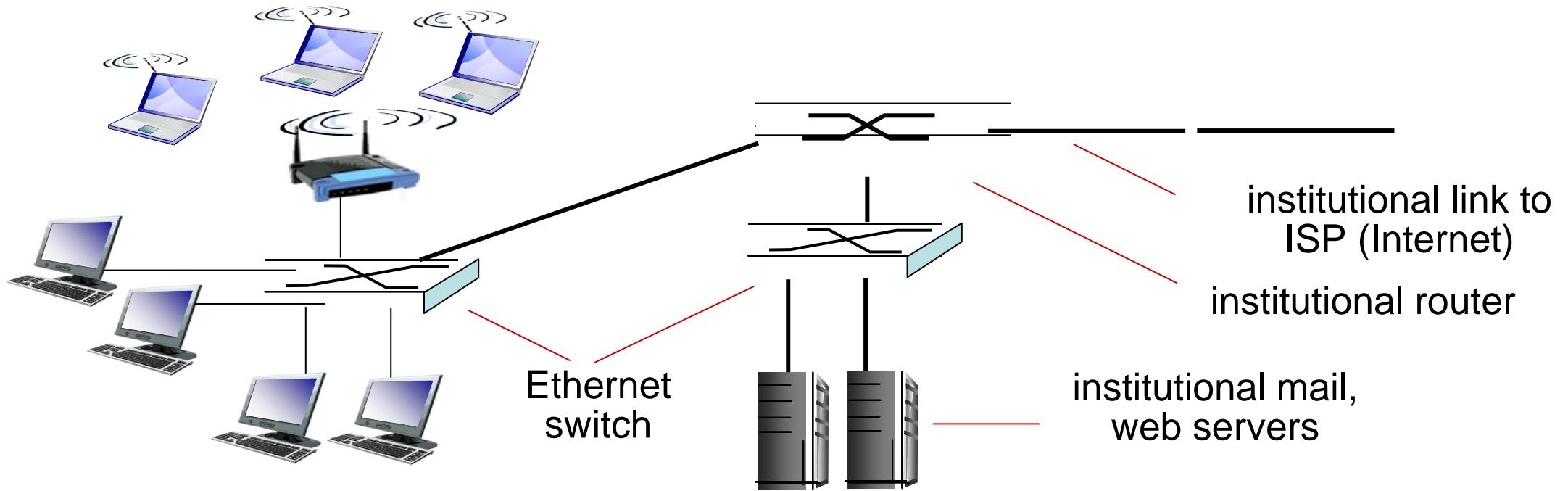
Module-IV

Computer Networks

Contents

- Introduction to Computer Networks
- Networks and Types of Networks
- Protocol Layers
- Ethernet

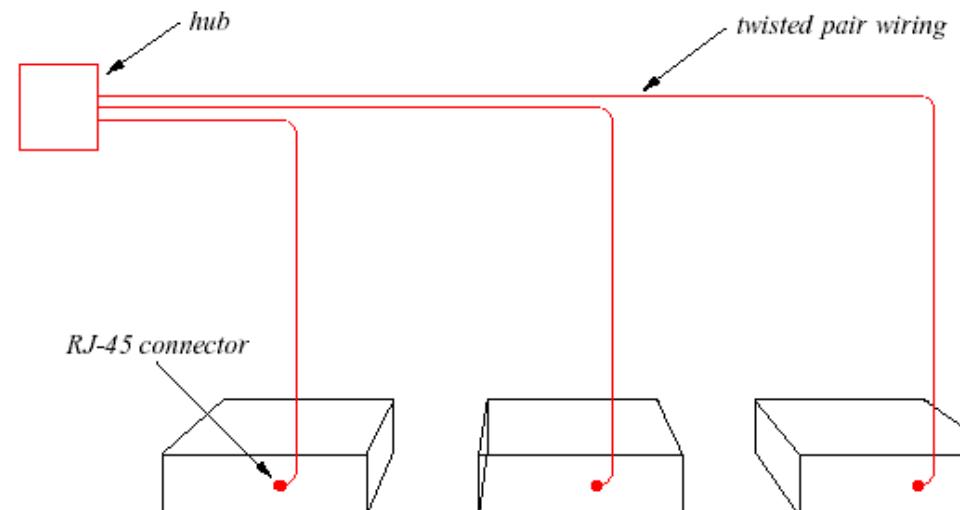
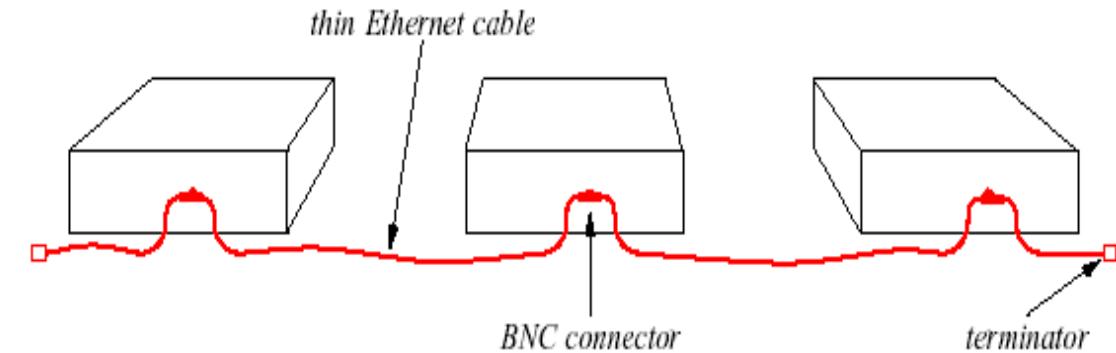
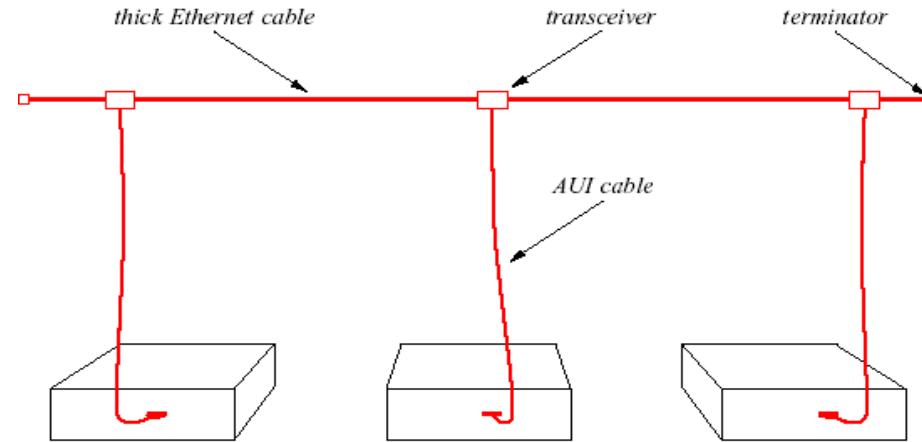
Enterprise access networks (Ethernet)



- ❖ typically used in companies, universities, etc
- ❖ 10 Mbps, 100Mbps, 1Gbps, 10Gbps transmission rates
- ❖ today, end systems typically connect into Ethernet switch

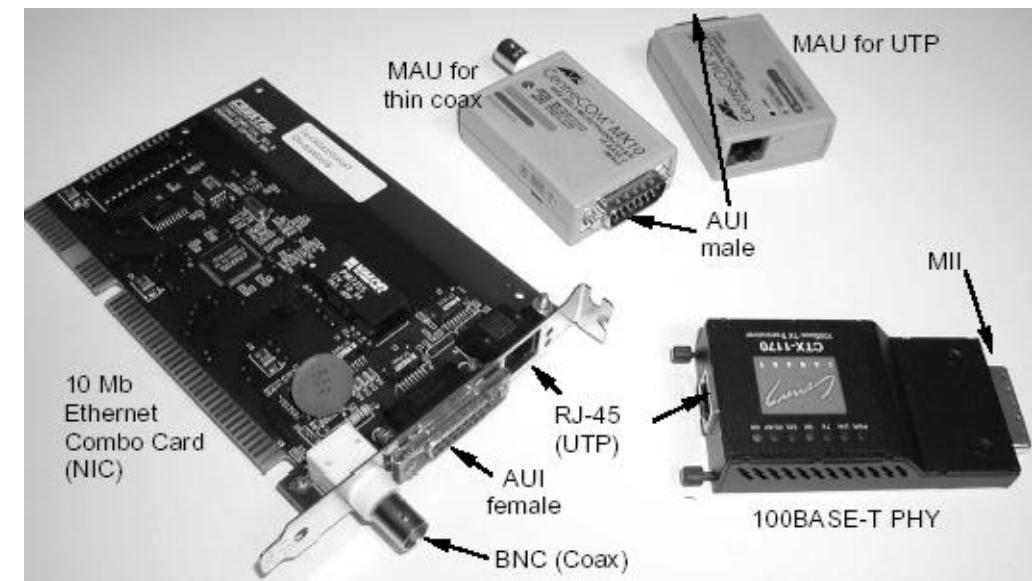
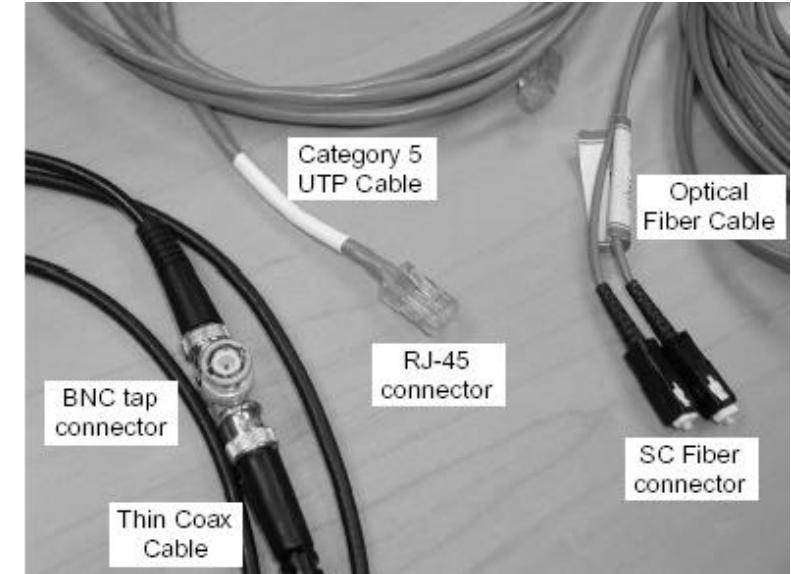
Ethernet Generations

- Original Ethernet:
 - Coaxial cable (10Base5)
 - Thicknet.
- Next Generation:
 - Thin coax cable (10Base2)
 - Thinnet.
- Modern Ethernet:
 - Twisted pair ethernet (10BaseT)
 - Uses hub: physical star but logical bus.



Ethernet Components

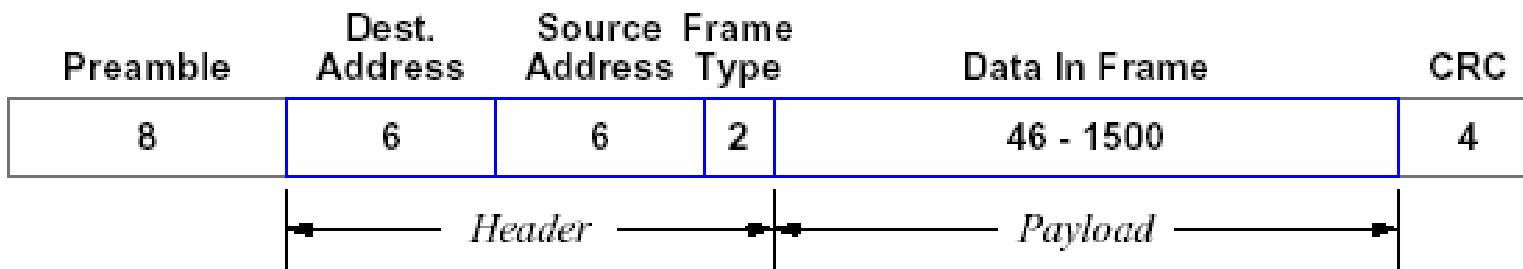
- **NIC – Network Interface Card**
 - Integrated Tx/Rx – direct interface to medium.
- **MAU – Media Attachment Unit**
 - Attaches network interface to the medium (integrated into NIC).
- **AUI – Attachment Unit Interface**
 - Decouple physical layer -reuse MAC design with different media.
- **MII – Media Independent Interface**
 - Like AUI for gigabit / faster ethernets.



Ethernet Addressing

- 48-bit address
- Address assigned when NIC card is manufactured.
- Packets can be sent to
 - Single address – Unicast
 - All stations on network – Broadcast (address = all 1s.)
 - Subset of stations – Multicast
- Broadcast (address = all 1s.)
 - All receivers accept unicast / broadcasts.
- Half addresses reserved for multicast (2^{47})
 - NIC can accept zero or more multicasts.

Ethernet Frame



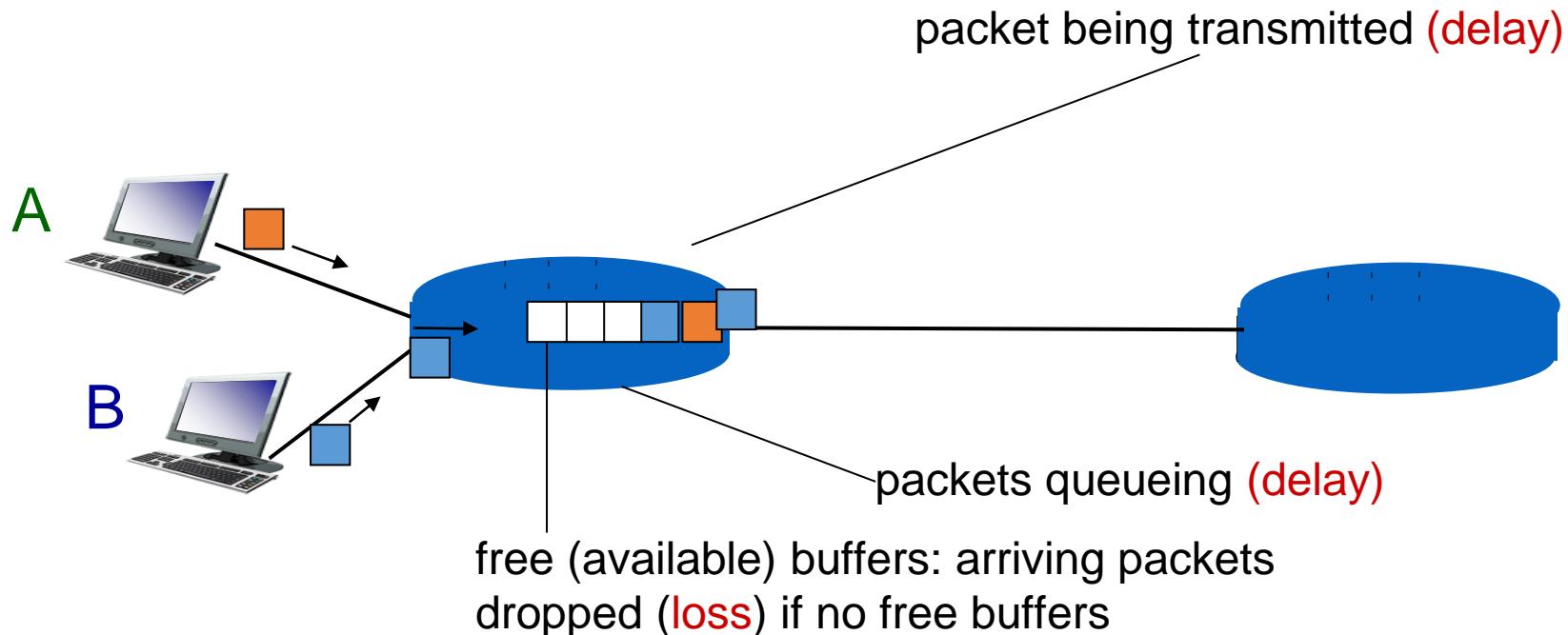
Recent Developments

- **100Base-FX**
 - LED light source / MMF / 2 km max distance.
 - Modal dispersion – limited bandwidth
-
- **100Base-SX (IEEE 802.3z)**
 - Short wavelength laser (850 nm)
 - Max distance = 5 km.
-
- **100Base-LX**
 - Long wavelength laser (1310 nm)
 - Max distance = 5 km.

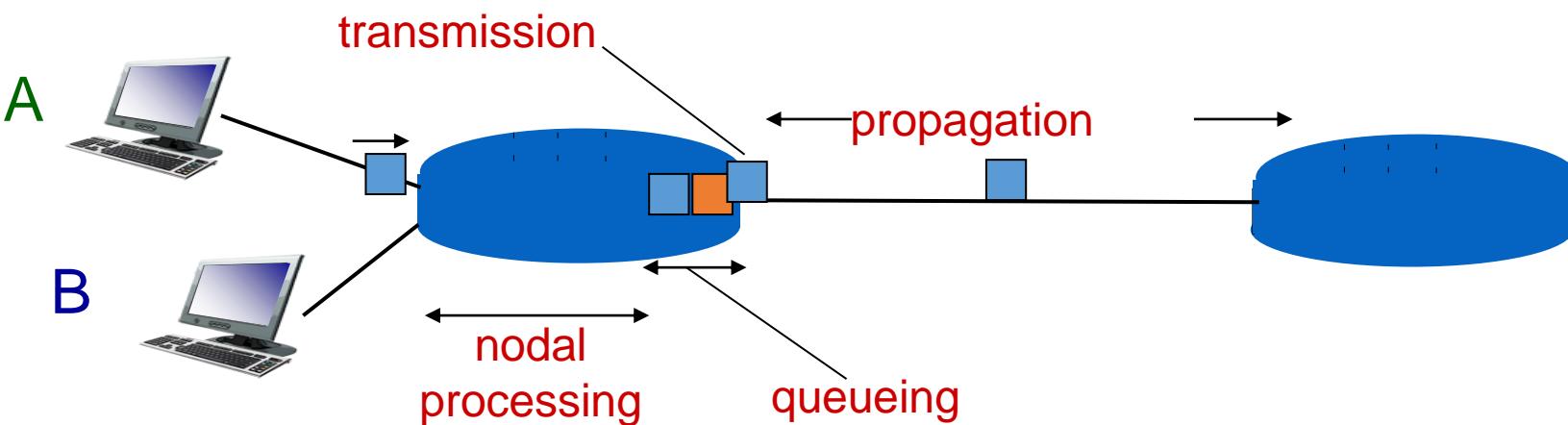
How do loss and delay occur?

packets *queue* in router buffers

- ❖ packet arrival rate to link (temporarily) exceeds output link capacity
- ❖ packets queue, wait for turn



Four sources of packet delay



$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

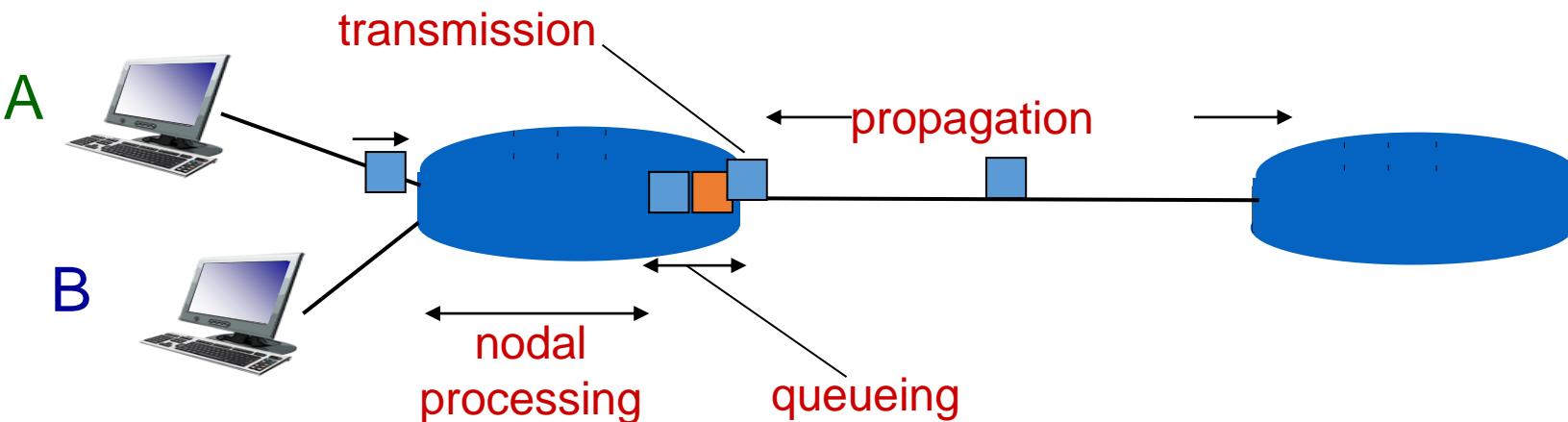
d_{proc} : nodal processing

- check bit errors
- determine output link
- typically < msec

d_{queue} : queueing delay

- time waiting at output link for transmission
- depends on congestion level of router

Four sources of packet delay



$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

d_{trans} : transmission delay:

- L : packet length (bits)
- R : link *bandwidth (bps)*
- $d_{\text{trans}} = L/R$

d_{prop} : propagation delay:

- d : length of physical link
- s : propagation speed in medium ($\sim 2 \times 10^8$ m/sec)
- $d_{\text{prop}} = d/s$

d_{trans} and d_{prop}
very different

* Check out the Java applet for an interactive animation on trans vs. prop delay

Processing Delay

- Time required to **examine** the packets header
 - Determines where to direct the packet
 - Check for errors
- Order of microseconds

Queuing Delay

- If a router is **busy** in processing and transmitting a packet, a freshly arrived packet has to wait in **queue** (buffer) for its turn.
- No queuing delay if the router is idle.
- Queuing delay varies with time and location. In general, it is a random variable.
- Order of microseconds to milliseconds.

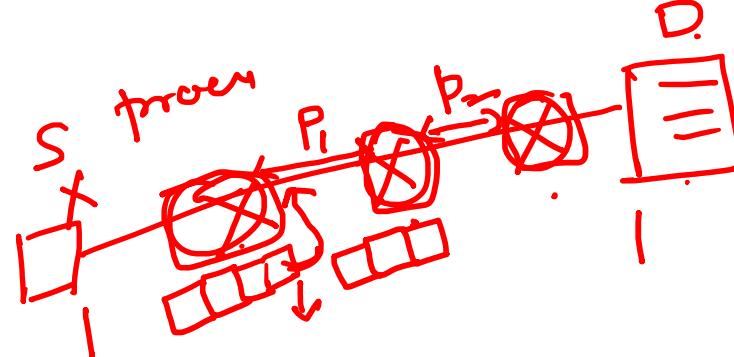
Transmission Delay

- Time required to **push** the packet into the link
- If the length of the packet is L bits and transmission rate of the link is R bps, then

$$\text{Transmission delay} = \frac{L}{R}$$

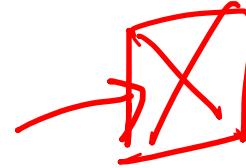
- Order of microseconds to milliseconds

Propagation Delay



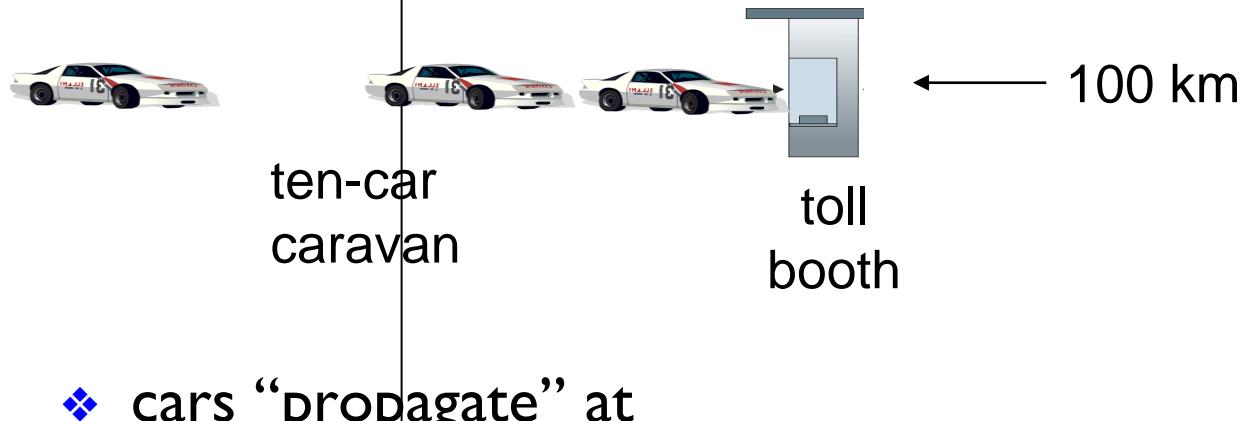
- Time required to **propagate** from one end of the link to the other end
- The propagation speed depends on the physical link between the routers
- In general, propagation speed s , is in the order of $2 \times 10^8 - 3 \times 10^8 \text{ m/s}$.
- Propagation speed depends on the distance bewteen the routers, d
- Propagation delay = $\frac{d}{s}$

Traffic Intensity

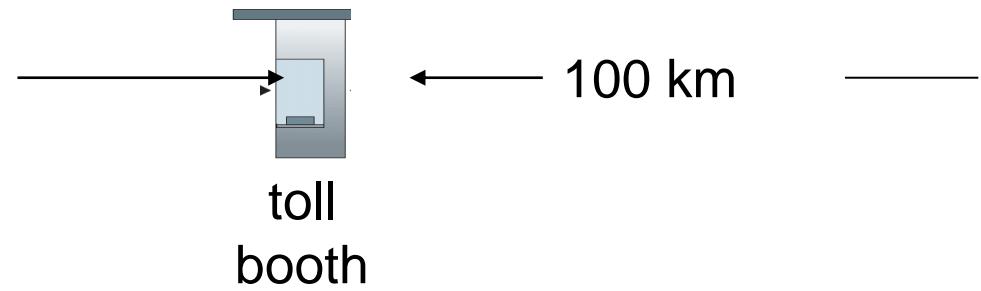


- Queuing delays are **random** in nature
- Arrivals to a queue are also **random** in nature
- Traffic intensity is an indication of queuing delay
- Let a be the average number of packets arriving at a queue
- Each packet is of length L bits adn transmission rate is R bps
- **Traffic intensity** = $\frac{La}{R}$

Caravan analogy

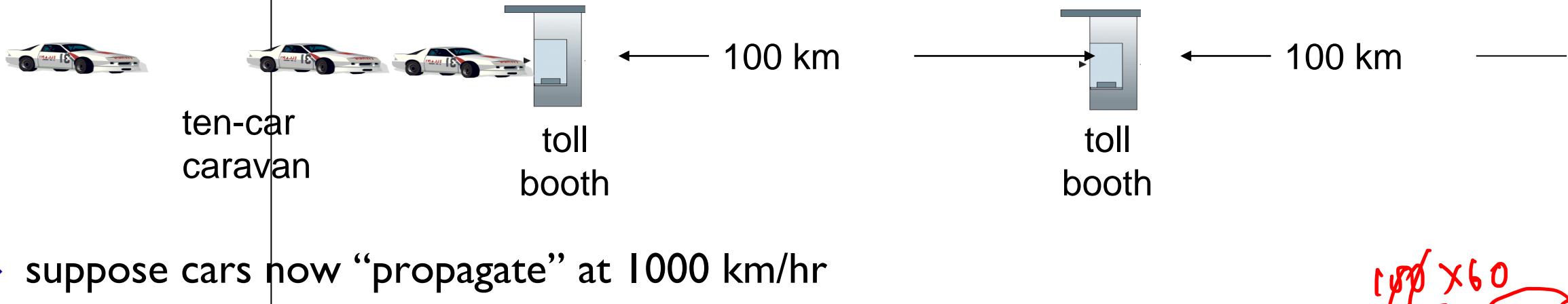


- ❖ cars “propagate” at 100 km/hr
- ❖ toll booth takes 12 sec to service car (bit transmission time)
- ❖ car~bit; caravan ~ packet
- ❖ **Q: How long until caravan is lined up before 2nd toll booth?**



- time to “push” entire caravan through toll booth onto highway = $12 \times 10 = 120$ sec
- time for last car to propagate from 1st to 2nd toll both: $100\text{km}/(100\text{km/hr}) = 1$ hr
- **A: 62 minutes**

Caravan analogy (more)



- ❖ suppose cars now “propagate” at 1000 km/hr
- ❖ and suppose toll booth now takes one min to service a car
- ❖ Q: Will cars arrive to 2nd booth before all cars serviced at first booth?

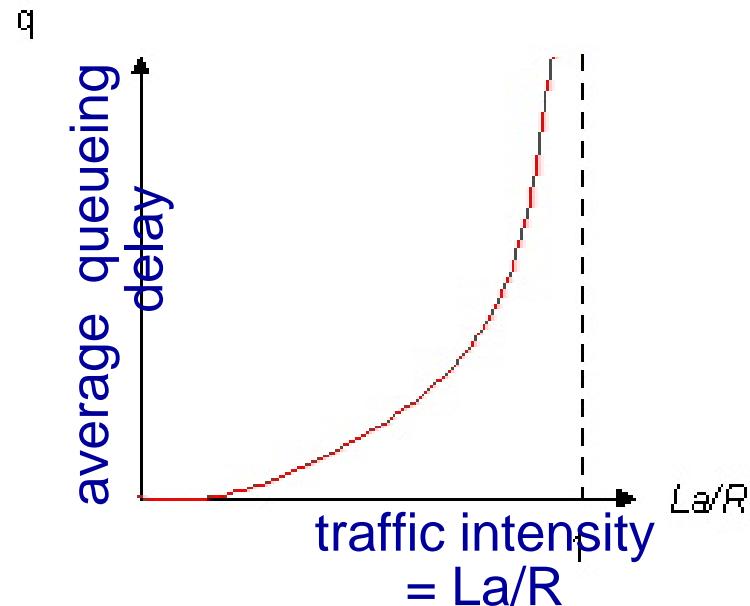
$$\frac{100 \times 60}{1000} = 6 \text{ min}$$

1 min

- A: Yes! after 7 min, 1st car arrives at second booth; three cars still at 1st booth.

Queueing delay (revisited)

- ❖ R : link bandwidth (bps)
- ❖ L : packet length (bits)
- ❖ a : average packet arrival rate



- ❖ $La/R \sim 0$: avg. queueing delay small
- ❖ $La/R \rightarrow 1$: avg. queueing delay large
- ❖ $La/R > 1$: more “work” arriving than can be serviced, average delay infinite!



$La/R \sim 0$

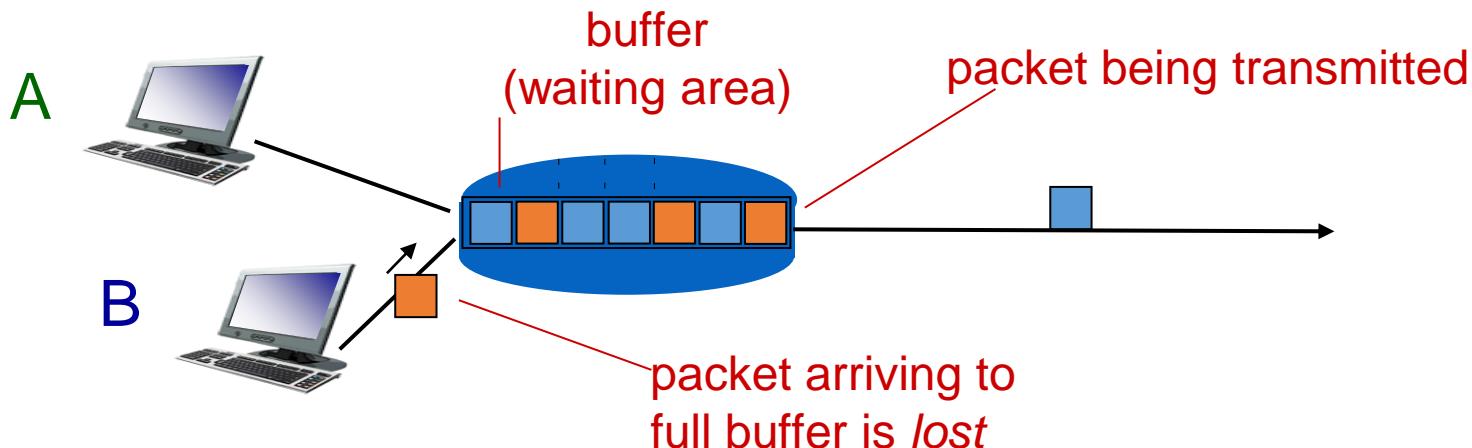


$La/R \rightarrow 1$

* Check out the Java applet for an interactive animation on queuing and loss

Packet loss

- ❖ queue (aka buffer) preceding link in buffer has finite capacity
- ❖ packet arriving to full queue dropped (aka lost)
- ❖ lost packet may be retransmitted by previous node, by source end system, or not at all



* Check out the Java applet for an interactive animation on queuing and loss

Overview of Computers

Workshop (OCW) Lab

Unit-4

Contents

- TCP/IP
- Internet and its Protocols (HTTP, HTTPS, FTP)
- IP Address and significance
- Static & Dynamic IP Addresses
- LAN network
- NS2 (Network Simulator) Demo

Transmission Control Protocol/ Internet Protocol

- **Protocol:** Set of rules that allow electronic devices to communicate with each other.

Types of protocols:

Communication protocols: TCP/IP, UDP and HTTP

Network Management protocols : SNMP and ICMP

Security protocols: SFTP and HTTPS

Transmission Control Protocol/ Internet Protocol

- **TCP/IP model** - A communication protocol used to interconnect network devices on the internet.
- It is a concise version of the OSI Model and comprises four layers in its structure.

Basics of TCP/IP Model	
Full-Form	Transmission Control Protocol/ Internet Protocol
Developed By	Department of Defence (DoD), United States
Developed in	During the 1970s
Function of TCP	Collecting and Reassembling Data Packets
Function of IP	Sending the Data Packets to the correct destination
Number of Layers in TCP/IP Model	4 layers

Layers of the TCP/IP Model

The TCP/IP model : structured with four different layers.

1. Network Access Layer
2. Internet Layer
3. Host to Host or Transport Layer
4. Application Layer

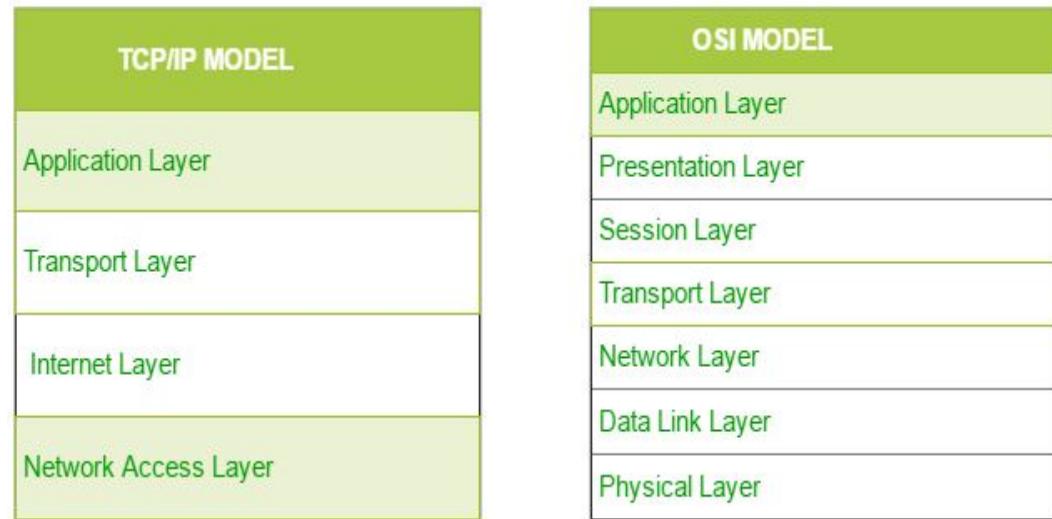


Fig: Comparison of the TCP/IP and OSI model

Network Access Layer

- Bottom-most layer
- A combination of the Data Link and Physical Layer of the OSI model
- Function: To transmit the data between two devices, connected in a network
- IP Datagram → Frames.

OSI Model	TCP/IP Stack
Data Link	
Physical	Network Access/Link

Internet Layer

Function: Selects the best path through the networks for packets to travel.

Three different protocols used in this layer.

- **IP:** Detects the IP address of a device. Decides the path with which the data transmitted.
- **ARP:** Stands for Address Resolution Protocol. The physical address from the IP address determined.
- **ICMP:** Stands for Internet Control Message Protocol. Inform the user about the errors and cannot rectify the problem.

Host to Host Layer

Function: Error-free delivery of data.

Two main protocols present in this layer:

- **TCP:** Manages the flow of data, i.e. the sequence and segmentation of the data
- **UDP:** connection-free protocol: cost-effective but less reliable.

TCP	UDP
Reliable	Unreliable
Connection-oriented	Connectionless
Retransmission	No retransmission
Segment sequencing	No sequencing
Acknowledge segments	No acknowledgment

Application Layer

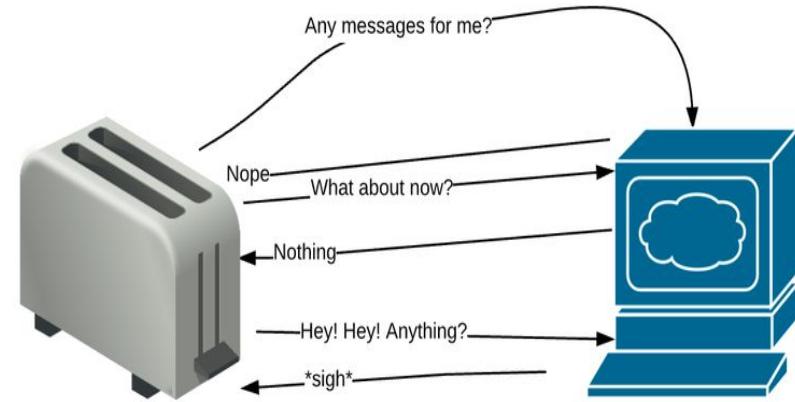
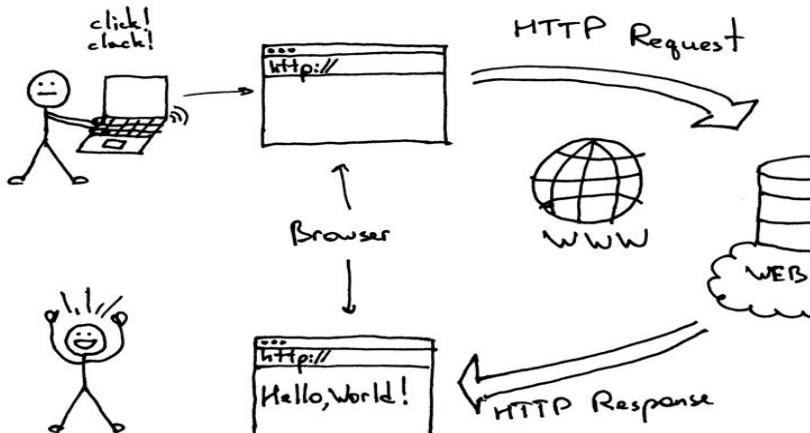
Function: To allow access to network resources.

Multiple protocols are present in this layer:

1. HTTP
2. NTP
3. TELNET
4. FTP
5. SMTP

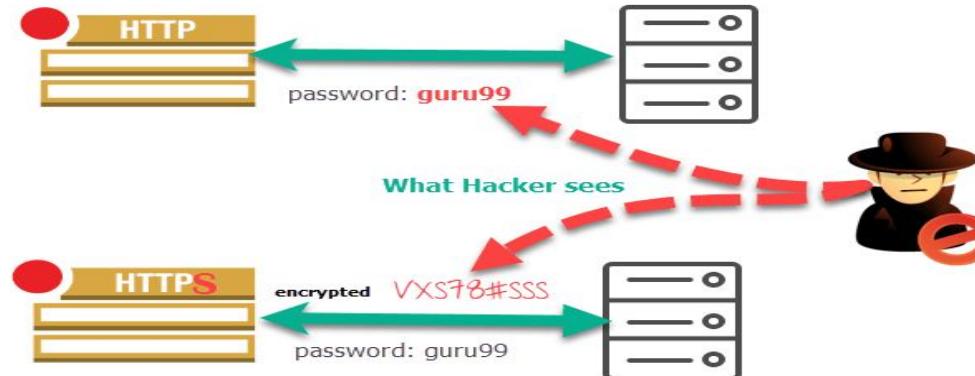
HTTP (HyperText Transfer Protocol)

- Application layer protocol designed to transfer information between networked devices.
- Typical flow involves a client (eg: browsers) machine making a request to a server (eg: computer in cloud), which then sends a response message.
- Simple request-response protocol, sent using TCP/IP sockets.
- **HTTP Request / Response:**
 - Communication between clients and servers is done by **requests** and **responses**:



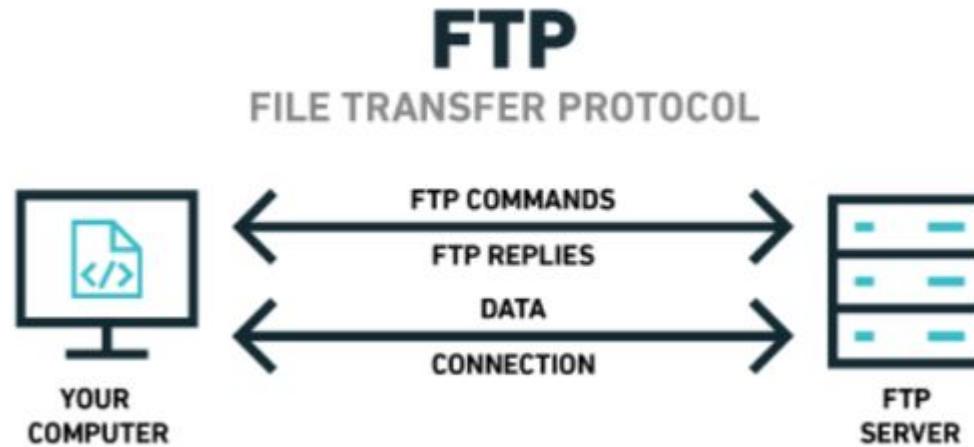
HTTPS (HyperText Transfer Protocol Secure)

- Secure version of HTTP.
- It is encrypted to increase security of data transfer.
- Important for transmitting sensitive data, eg: logging into bank account, email service, or health insurance provider.
- HTTPS uses an encryption protocol called Transport Layer Security (TLS).
- Eg:
 - Before encryption: **This is a string of text that is completely readable**
 - After encryption: **ITM0IRyiEhVpa6VnKyExMiEgNveroyWBPIgGyfkfIYjDaaFf/Kn3bo3OfghBPDWo6AfSHINtL8N7ITEwlXc1gU5X73xMsJormzzXlwOyrCs+9XCPk63Y+z0=**



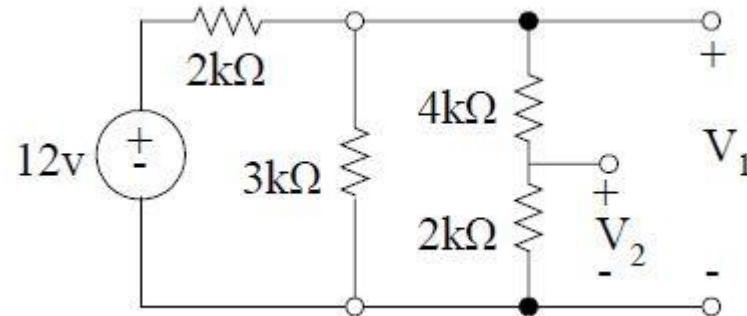
FTP (File transfer protocol)

- Application layer protocol that moves files between local and remote file systems over a network.
- It runs on the top of TCP / IP.
- File Transfer Protocol works in a client-server model.



IP address

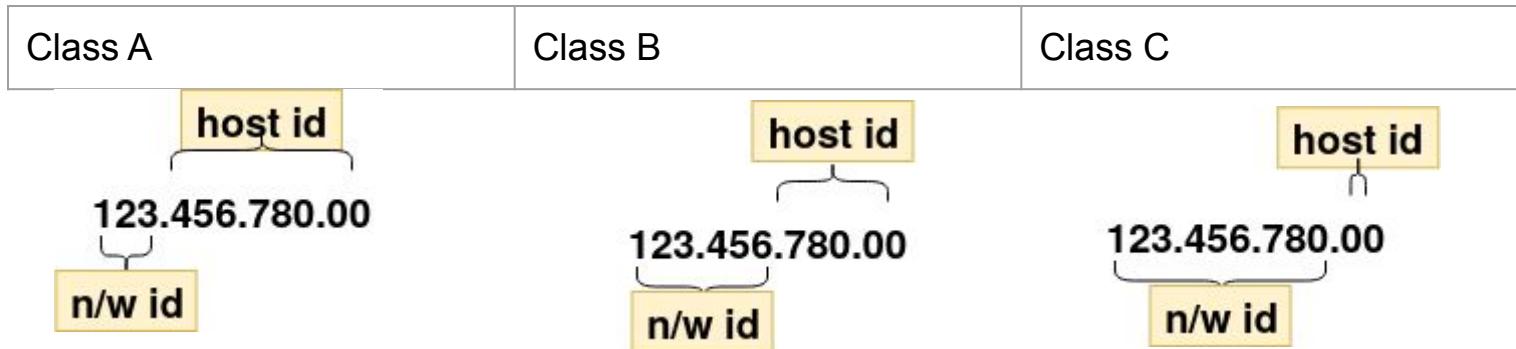
- Network
- Host
- Subnetwork
- Subnet Mask



IP addressing

- IP address:

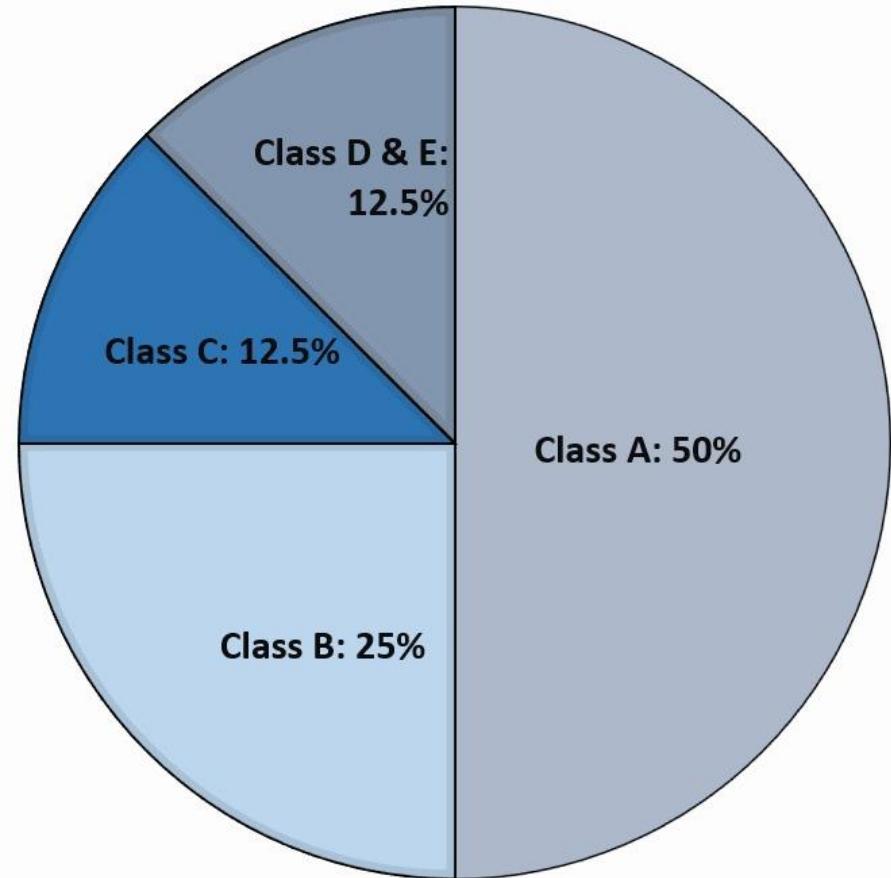
- Three classes:



- Class A: 0.0.0.0 - 127.255.255.255
- Class B: 128.0.0.0 - 191.255.255.255
- Class C: 192.0.0.0 - 223.255.255.255

CLASSFULL IPV4 ADDRESS ALLOCATION

Class A Class B Class C Class D & Class E



Static and dynamic IP addressing

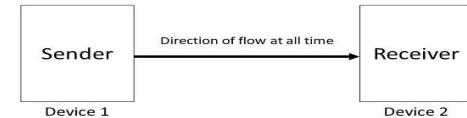
- **Static:**
 - Fixed addressing.
 - Used by **servers** or important equipments.
 - Adds to system cost.
- **Dynamic:**
 - Subject to change, at a moment.
 - **Assigned as needed**, by Dynamic Host Configuration Protocol (DHCP) Servers.
 - Used as **IPv4 does not provide enough static address**.
 - On Internet→assigned by ISP DHCP server.
 - Home or business network→assigned by network router.



LAN network

- **Simplex**

- Communication channel which sends information in only one direction.
- Eg: Radio station usually sends signals to the audience but never receives signals from them.
- The good part is that its entire bandwidth can be used during the transmission.



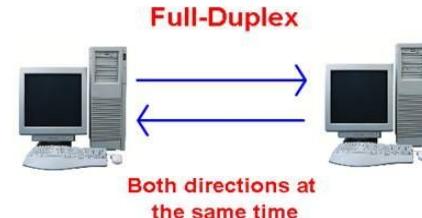
- **Half duplex**

- Data can be transmitted in both directions but not at the same time.
- Eg: Walkie talkie.
- An advantage of half-duplex is that the single track is cheaper than the double tracks.



- **Full duplex**

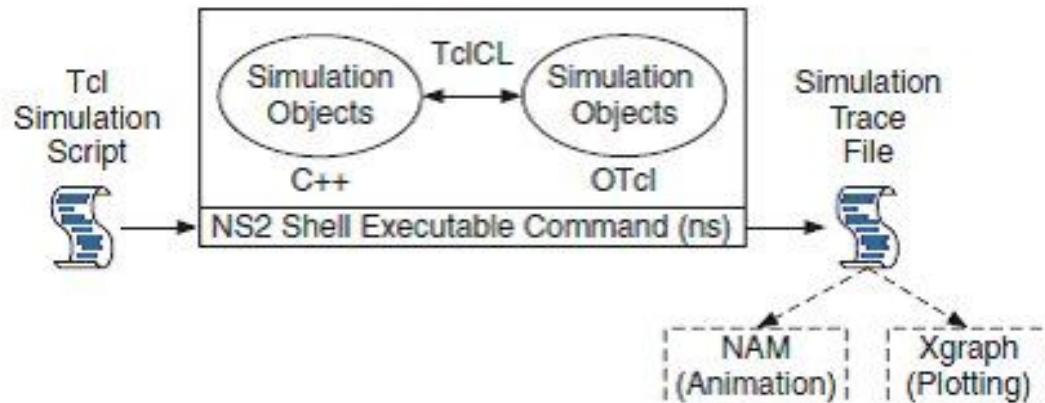
- Able to transmit data in both directions on a signal carrier at the same time.
- Eg: Telephone is an example.
- Using the full duplex mode can greatly increase the efficiency of communication.



Network Simulator 2 (NS2) demonstration

- Discrete event simulator (models the operation of a system as a discrete sequence of events in time).
- Substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.
- **Applications of NS2:**

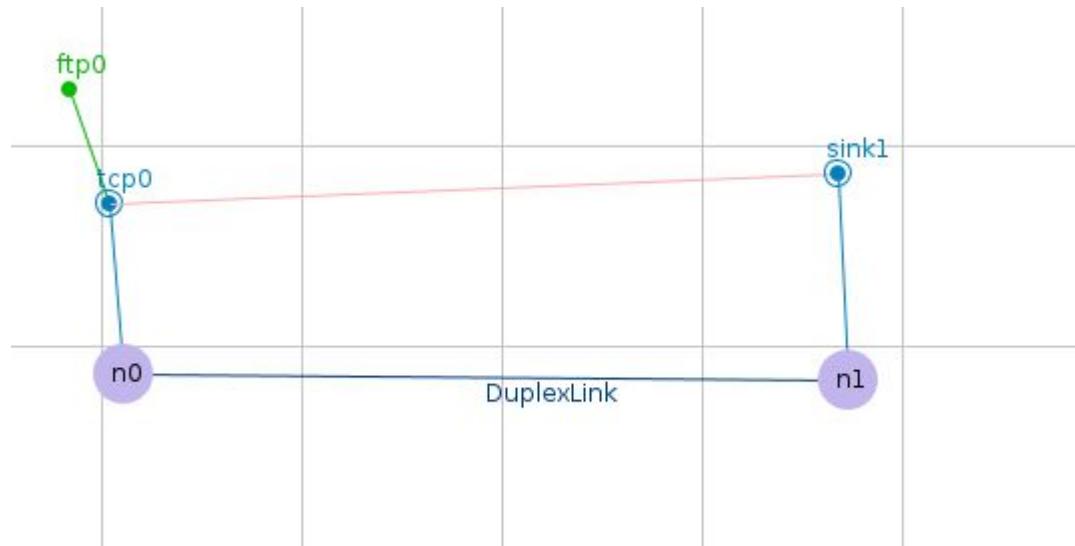
- Networking education
- Topology Generation.
- Scenario generation.



Basic architecture of NS.

NS2 Example:

TCP example



NS2 Example

```
# Simulation parameters setup
set val(stop) 10.0; # time of simulation end

# Initialization
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile

#Nodes Definition
#Create 2 nodes
set n0 [$ns node]
set n1 [$ns node]

# Links Definition
#Createlinks between nodes
$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n1 50

#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right
```

NS2 Example

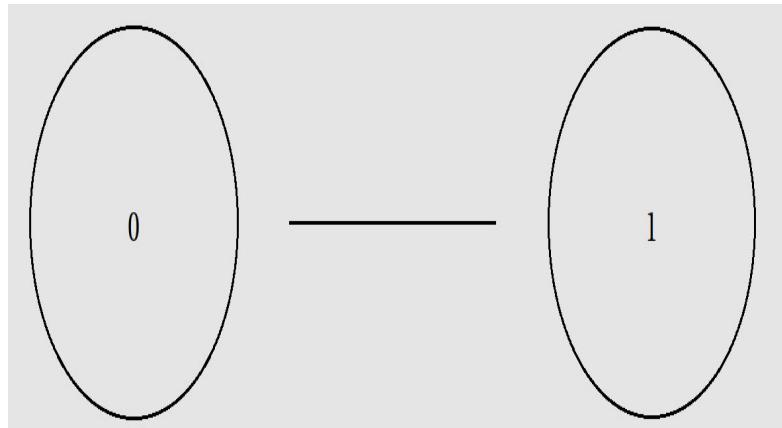
```
# Agents Definition  
#Setup a TCP connection  
set tcp0 [new Agent/TCP]  
$ns attach-agent $n0 $tcp0  
set sink1 [new Agent/TCPSink]  
$ns attach-agent $n1 $sink1  
$ns connect $tcp0 $sink1  
$tcp0 set packetSize_ 1500
```

```
#Applications Definition  
#Setup a FTP Application over TCP connection  
set ftp0 [new Application/FTP]  
$ftp0 attach-agent $tcp0  
$ns at 1.0 "$ftp0 start"  
$ns at 2.0 "$ftp0 stop"
```

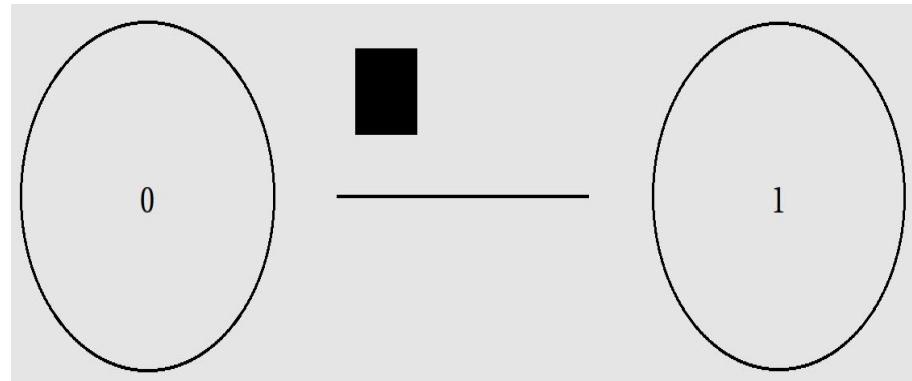
```
# Termination  
#Define a 'finish' procedure  
proc finish {} {  
    global ns tracefile namfile  
    $ns flush-trace  
    close $tracefile  
    close $namfile  
    exec nam out.nam &  
    exit 0  
}  
  
$ns at $val(stop) "$ns nam-end-wireless  
$val(stop)"  
$ns at $val(stop) "finish"  
$ns at $val(stop) "puts \"done\" ; $ns halt"  
$ns run
```

NS2 Example

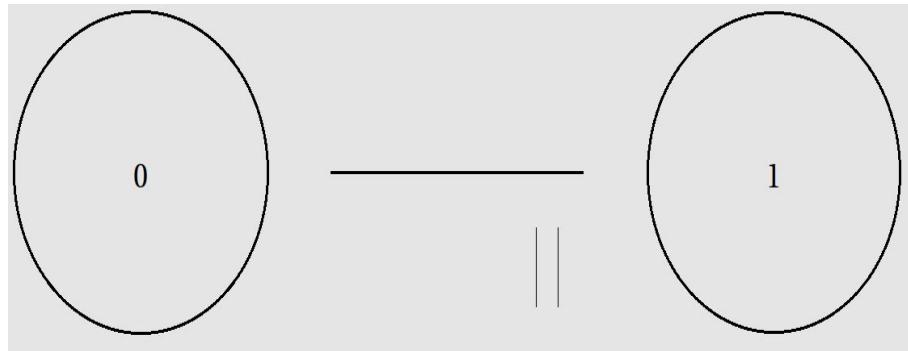
Results:



Initialisation in NAM



Node 0 sending data to node 1 using tcp



Node 1 sending ACK to node 0

NS2 Example

Trace file:

```
1 + 1 0 1 tcp 40 ----- 0 0.0 1.0 0 0  
2 - 1 0 1 tcp 40 ----- 0 0.0 1.0 0 0  
3 r 1.010003 0 1 tcp 40 ----- 0 0.0 1.0 0 0  
4 + 1.010003 1 0 ack 40 ----- 0 1.0 0.0 0 1  
5 - 1.010003 1 0 ack 40 ----- 0 1.0 0.0 0 1  
6 r 1.020006 1 0 ack 40 ----- 0 1.0 0.0 0 1  
7 + 1.020006 0 1 tcp 1540 ----- 0 0.0 1.0 1 2  
8 - 1.020006 0 1 tcp 1540 ----- 0 0.0 1.0 1 2  
9 + 1.020006 0 1 tcp 1540 ----- 0 0.0 1.0 2 3  
10 - 1.02013 0 1 tcp 1540 ----- 0 0.0 1.0 2 3  
11 r 1.03013 0 1 tcp 1540 ----- 0 0.0 1.0 1 2  
12 + 1.03013 1 0 ack 40 ----- 0 1.0 0.0 1 4  
13 - 1.03013 1 0 ack 40 ----- 0 1.0 0.0 1 4  
14 r 1.030253 0 1 tcp 1540 ----- 0 0.0 1.0 2 3  
15 + 1.030253 1 0 ack 40 ----- 0 1.0 0.0 2 5  
16 - 1.030253 1 0 ack 40 ----- 0 1.0 0.0 2 5  
17 r 1.040133 1 0 ack 40 ----- 0 1.0 0.0 1 4  
18 + 1.040133 0 1 tcp 1540 ----- 0 0.0 1.0 3 6  
19 - 1.040133 0 1 tcp 1540 ----- 0 0.0 1.0 3 6  
20 + 1.040133 0 1 tcp 1540 ----- 0 0.0 1.0 4 7  
21 r 1.040256 1 0 ack 40 ----- 0 1.0 0.0 2 5  
22 + 1.040256 0 1 tcp 1540 ----- 0 0.0 1.0 5 8  
23 + 1.040256 0 1 tcp 1540 ----- 0 0.0 1.0 6 9  
24 - 1.040256 0 1 tcp 1540 ----- 0 0.0 1.0 4 7  
25 - 1.040379 0 1 tcp 1540 ----- 0 0.0 1.0 5 8  
26 - 1.040502 0 1 tcp 1540 ----- 0 0.0 1.0 6 9  
27 r 1.050256 0 1 tcp 1540 ----- 0 0.0 1.0 3 6  
28 + 1.050256 1 0 ack 40 ----- 0 1.0 0.0 3 10  
29 - 1.050256 1 0 ack 40 ----- 0 1.0 0.0 3 10  
30 r 1.050379 0 1 tcp 1540 ----- 0 0.0 1.0 4 7  
31 + 1.050379 1 0 ack 40 ----- 0 1.0 0.0 4 11  
32 - 1.050379 1 0 ack 40 ----- 0 1.0 0.0 4 11  
33 r 1.050502 0 1 tcp 1540 ----- 0 0.0 1.0 5 8  
34 + 1.050502 1 0 ack 40 ----- 0 1.0 0.0 5 12  
35 - 1.050502 1 0 ack 40 ----- 0 1.0 0.0 5 12  
36 r 1.050626 0 1 tcp 1540 ----- 0 0.0 1.0 6 9  
37 + 1.050626 1 0 ack 40 ----- 0 1.0 0.0 6 13  
38 - 1.050626 1 0 ack 40 ----- 0 1.0 0.0 6 13  
39 r 1.060259 1 0 ack 40 ----- 0 1.0 0.0 3 10  
40 + 1.060259 0 1 tcp 1540 ----- 0 0.0 1.0 7 14  
41 - 1.060259 0 1 tcp 1540 ----- 0 0.0 1.0 7 14  
42 + 1.060259 0 1 tcp 1540 ----- 0 0.0 1.0 8 15  
43 r 1.060382 1 0 ack 40 ----- 0 1.0 0.0 4 11  
44 + 1.060382 0 1 tcp 1540 ----- 0 0.0 1.0 9 16  
45 + 1.060382 0 1 tcp 1540 ----- 0 0.0 1.0 10 17
```

Network animator file:

```
1 V -t * -v 1.0a5 -a 0  
2 A -t * -n 1 -p 0 -o 0x7fffffff -c 30 -a 1  
3 A -t * -h 1 -m 1073741823 -s 0  
4 n -t * -a 0 -s 0 -S UP -v circle -c black -i black  
5 n -t * -a 1 -s 1 -S UP -v circle -c black -i black  
6 l -t * -s 0 -d 1 -S UP -r 100000000 -D 0.01 -c black -o right  
7 + -t 1 -s 0 -d 1 -p tcp -e 40 -c 0 -i 0 -a 0 -x {0.0 1.0 0 ----- null}  
8 - -t 1 -s 0 -d 1 -p tcp -e 40 -c 0 -i 0 -a 0 -x {0.0 1.0 0 ----- null}  
9 h -t 1 -s 0 -d 1 -p tcp -e 40 -c 0 -i 0 -a 0 -x {0.0 1.0 -1 ----- null}  
10 r -t 1.0100032 -s 0 -d 1 -p tcp -e 40 -c 0 -i 0 -a 0 -x {0.0 1.0 0 ----- null}  
11 + -t 1.0100032 -s 1 -d 0 -p ack -e 40 -c 0 -i 1 -a 0 -x {1.0 0.0 0 ----- null}  
12 - -t 1.0100032 -s 1 -d 0 -p ack -e 40 -c 0 -i 1 -a 0 -x {1.0 0.0 0 ----- null}  
13 h -t 1.0100032 -s 1 -d 0 -p ack -e 40 -c 0 -i 1 -a 0 -x {1.0 0.0 -1 ----- null}  
14 r -t 1.0200064 -s 1 -d 0 -p ack -e 40 -c 0 -i 1 -a 0 -x {1.0 0.0 0 ----- null}  
15 + -t 1.0200064 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 2 -a 0 -x {0.0 1.0 1 ----- null}  
16 - -t 1.0200064 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 2 -a 0 -x {0.0 1.0 1 ----- null}  
17 h -t 1.0200064 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 2 -a 0 -x {0.0 1.0 -1 ----- null}  
18 + -t 1.0200064 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 3 -a 0 -x {0.0 1.0 2 ----- null}  
19 - -t 1.0201296 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 3 -a 0 -x {0.0 1.0 2 ----- null}  
20 h -t 1.0201296 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 3 -a 0 -x {0.0 1.0 -1 ----- null}  
21 r -t 1.0301296 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 2 -a 0 -x {0.0 1.0 1 ----- null}  
22 + -t 1.0301296 -s 1 -d 0 -p ack -e 40 -c 0 -i 4 -a 0 -x {1.0 0.0 1 ----- null}  
23 - -t 1.0301296 -s 1 -d 0 -p ack -e 40 -c 0 -i 4 -a 0 -x {1.0 0.0 1 ----- null}  
24 h -t 1.0301296 -s 1 -d 0 -p ack -e 40 -c 0 -i 4 -a 0 -x {1.0 0.0 -1 ----- null}  
25 r -t 1.0302528 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 3 -a 0 -x {0.0 1.0 2 ----- null}  
26 + -t 1.0302528 -s 1 -d 0 -p ack -e 40 -c 0 -i 5 -a 0 -x {1.0 0.0 2 ----- null}  
27 - -t 1.0302528 -s 1 -d 0 -p ack -e 40 -c 0 -i 5 -a 0 -x {1.0 0.0 2 ----- null}  
28 h -t 1.0302528 -s 1 -d 0 -p ack -e 40 -c 0 -i 5 -a 0 -x {1.0 0.0 -1 ----- null}  
29 r -t 1.0401328 -s 1 -d 0 -p ack -e 40 -c 0 -i 4 -a 0 -x {1.0 0.0 1 ----- null}  
30 + -t 1.0401328 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 6 -a 0 -x {0.0 1.0 3 ----- null}  
31 - -t 1.0401328 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 6 -a 0 -x {0.0 1.0 3 ----- null}  
32 h -t 1.0401328 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 6 -a 0 -x {0.0 1.0 -1 ----- null}  
33 + -t 1.0401328 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 7 -a 0 -x {0.0 1.0 4 ----- null}  
34 r -t 1.040256 -s 1 -d 0 -p ack -e 40 -c 0 -i 5 -a 0 -x {1.0 0.0 2 ----- null}  
35 + -t 1.040256 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 8 -a 0 -x {0.0 1.0 5 ----- null}  
36 + -t 1.040256 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 9 -a 0 -x {0.0 1.0 6 ----- null}  
37 - -t 1.040256 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 7 -a 0 -x {0.0 1.0 4 ----- null}  
38 h -t 1.040256 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 7 -a 0 -x {0.0 1.0 -1 ----- null}  
39 - -t 1.0403792 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 8 -a 0 -x {0.0 1.0 5 ----- null}  
40 h -t 1.0403792 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 8 -a 0 -x {0.0 1.0 -1 ----- null}  
41 - -t 1.0405024 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 9 -a 0 -x {0.0 1.0 6 ----- null}  
42 h -t 1.0405024 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 9 -a 0 -x {0.0 1.0 -1 ----- null}  
43 r -t 1.050256 -s 0 -d 1 -p tcp -e 1540 -c 0 -i 6 -a 0 -x {0.0 1.0 3 ----- null}
```

NS2 Example

UDP example



NS2 Example

```
set val(stop) 10.0;# time of simulation end  
  
#Initialization  
#Create a ns simulator  
set ns [new Simulator]  
  
#Open the NS trace file  
set tracefile [open out.tr w]  
$ns trace-all $tracefile  
  
#Open the NAM trace file  
set namfile [open out.nam w]  
$ns namtrace-all $namfile
```

```
#Nodes Definition  
#Create 2 nodes  
set n0 [$ns node]  
set n1 [$ns node]  
  
#Links Definition  
#Create links between nodes  
$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail  
$ns queue-limit $n0 $n1 50  
  
#Give node position (for NAM)  
$ns duplex-link-op $n0 $n1 orient right-down
```

NS2 Example

```
#Agents Definition
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null1 [new Agent/Null]
$ns attach-agent $n1 $null1
$ns connect $udp0 $null1
$udp0 set packetSize_ 1500

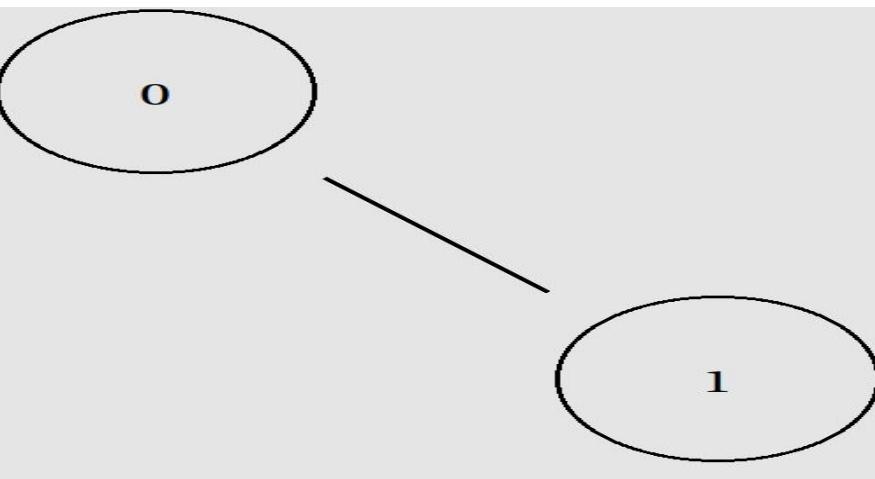
#Applications Definition
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr0 stop"

#Termination
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}

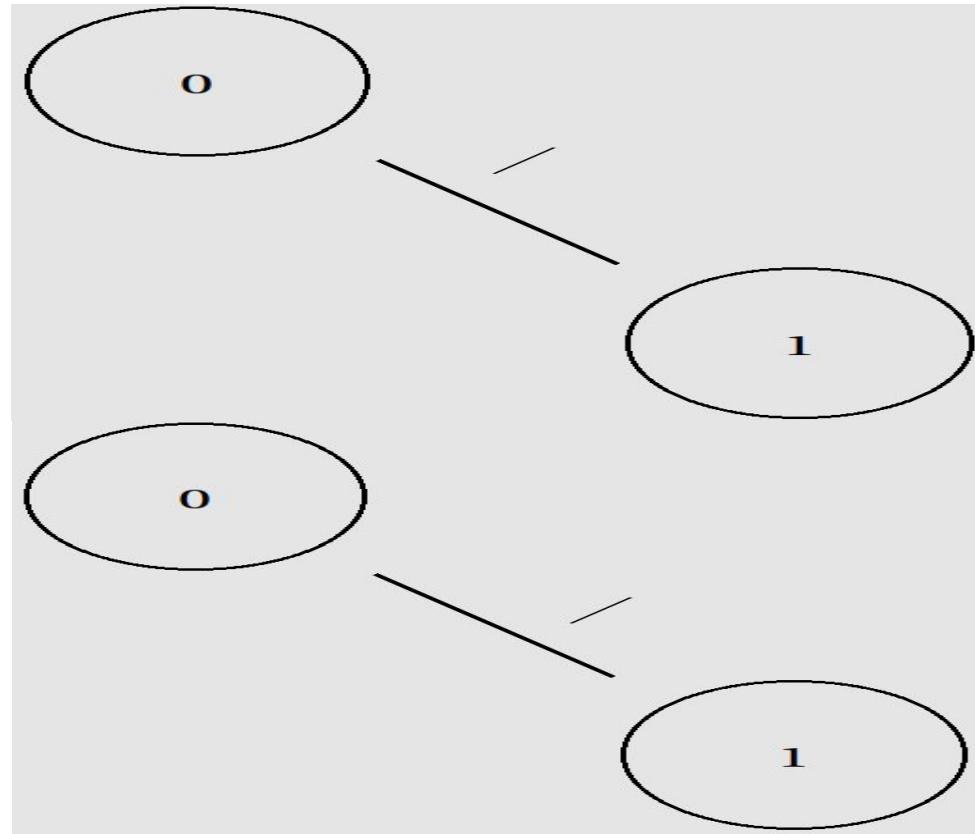
$ns at $val(stop) "$ns nam-end-wireless
$val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

NS2 Example

Results:



Initialisation in NAM



Node 0 sending data to node 1 using udp

NS2 Example

Trace file:

```

1 + 1 0 1 cbr 1000 ----- 0 0.0 1.0 0 0
2 - 1 0 1 cbr 1000 ----- 0 0.0 1.0 0 0
3 + 1.008 0 1 cbr 1000 ----- 0 0.0 1.0 1 1
4 - 1.008 0 1 cbr 1000 ----- 0 0.0 1.0 1 1
5 r 1.01008 0 1 cbr 1000 ----- 0 0.0 1.0 0 0
6 + 1.016 0 1 cbr 1000 ----- 0 0.0 1.0 2 2
7 - 1.016 0 1 cbr 1000 ----- 0 0.0 1.0 2 2
8 r 1.01808 0 1 cbr 1000 ----- 0 0.0 1.0 1 1
9 + 1.024 0 1 cbr 1000 ----- 0 0.0 1.0 3 3
10 - 1.024 0 1 cbr 1000 ----- 0 0.0 1.0 3 3
11 r 1.02608 0 1 cbr 1000 ----- 0 0.0 1.0 2 2
12 + 1.032 0 1 cbr 1000 ----- 0 0.0 1.0 4 4
13 - 1.032 0 1 cbr 1000 ----- 0 0.0 1.0 4 4
14 r 1.03408 0 1 cbr 1000 ----- 0 0.0 1.0 3 3
15 + 1.04 0 1 cbr 1000 ----- 0 0.0 1.0 5 5
16 - 1.04 0 1 cbr 1000 ----- 0 0.0 1.0 5 5
17 r 1.04208 0 1 cbr 1000 ----- 0 0.0 1.0 4 4
18 + 1.048 0 1 cbr 1000 ----- 0 0.0 1.0 6 6
19 - 1.048 0 1 cbr 1000 ----- 0 0.0 1.0 6 6
20 r 1.05008 0 1 cbr 1000 ----- 0 0.0 1.0 5 5
21 + 1.056 0 1 cbr 1000 ----- 0 0.0 1.0 7 7
22 - 1.056 0 1 cbr 1000 ----- 0 0.0 1.0 7 7
23 r 1.05808 0 1 cbr 1000 ----- 0 0.0 1.0 6 6
24 + 1.064 0 1 cbr 1000 ----- 0 0.0 1.0 8 8
25 - 1.064 0 1 cbr 1000 ----- 0 0.0 1.0 8 8
26 r 1.06608 0 1 cbr 1000 ----- 0 0.0 1.0 7 7
27 + 1.072 0 1 cbr 1000 ----- 0 0.0 1.0 9 9
28 - 1.072 0 1 cbr 1000 ----- 0 0.0 1.0 9 9
29 r 1.07408 0 1 cbr 1000 ----- 0 0.0 1.0 8 8
30 + 1.08 0 1 cbr 1000 ----- 0 0.0 1.0 10 10
31 - 1.08 0 1 cbr 1000 ----- 0 0.0 1.0 10 10
32 r 1.08208 0 1 cbr 1000 ----- 0 0.0 1.0 9 9
33 + 1.088 0 1 cbr 1000 ----- 0 0.0 1.0 11 11
34 - 1.088 0 1 cbr 1000 ----- 0 0.0 1.0 11 11
35 r 1.09008 0 1 cbr 1000 ----- 0 0.0 1.0 10 10
36 + 1.096 0 1 cbr 1000 ----- 0 0.0 1.0 12 12
37 - 1.096 0 1 cbr 1000 ----- 0 0.0 1.0 12 12
38 r 1.09808 0 1 cbr 1000 ----- 0 0.0 1.0 11 11
39 + 1.104 0 1 cbr 1000 ----- 0 0.0 1.0 13 13
40 - 1.104 0 1 cbr 1000 ----- 0 0.0 1.0 13 13
41 r 1.10608 0 1 cbr 1000 ----- 0 0.0 1.0 12 12
42 + 1.112 0 1 cbr 1000 ----- 0 0.0 1.0 14 14
43 - 1.112 0 1 cbr 1000 ----- 0 0.0 1.0 14 14
44 r 1.11408 0 1 cbr 1000 ----- 0 0.0 1.0 13 13
45 + 1.12 0 1 cbr 1000 ----- 0 0.0 1.0 15 15

```

Nam file:

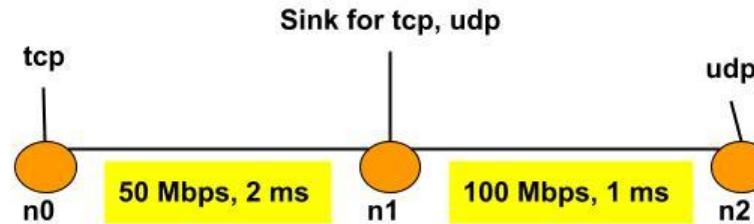
```

1 V -t * -v 1.0a5 -a 0
2 A -t * -n 1 -p 0 -o 0x7fffffff -c 30 -a 1
3 A -t * -h 1 -m 1073741823 -s 0
4 n -t * -a 0 -s 0 -S UP -v circle -c black -i black
5 n -t * -a 1 -s 1 -S UP -v circle -c black -i black
6 l -t * -s 0 -d 1 -S UP -r 100000000 -D 0.01 -c black -o right-down
7 + -t 1 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 0 -a 0 -x {0.0 1.0 0 ----- null}
8 - -t 1 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 0 -a 0 -x {0.0 1.0 0 ----- null}
9 h -t 1 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 0 -a 0 -x {0.0 1.0 -1 ----- null}
10 + -t 1.008 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 1 -a 0 -x {0.0 1.0 1 ----- null}
11 - -t 1.008 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 1 -a 0 -x {0.0 1.0 1 ----- null}
12 h -t 1.008 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 1 -a 0 -x {0.0 1.0 -1 ----- null}
13 r -t 1.01008 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 0 -a 0 -x {0.0 1.0 0 ----- null}
14 + -t 1.016 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 2 -a 0 -x {0.0 1.0 2 ----- null}
15 - -t 1.016 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 2 -a 0 -x {0.0 1.0 2 ----- null}
16 h -t 1.016 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 2 -a 0 -x {0.0 1.0 -1 ----- null}
17 r -t 1.01808 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 1 -a 0 -x {0.0 1.0 1 ----- null}
18 + -t 1.024 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 3 -a 0 -x {0.0 1.0 3 ----- null}
19 - -t 1.024 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 3 -a 0 -x {0.0 1.0 3 ----- null}
20 h -t 1.024 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 3 -a 0 -x {0.0 1.0 -1 ----- null}
21 r -t 1.02608 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 2 -a 0 -x {0.0 1.0 2 ----- null}
22 + -t 1.032 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 4 -a 0 -x {0.0 1.0 4 ----- null}
23 - -t 1.032 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 4 -a 0 -x {0.0 1.0 4 ----- null}
24 h -t 1.032 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 4 -a 0 -x {0.0 1.0 -1 ----- null}
25 r -t 1.03408 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 3 -a 0 -x {0.0 1.0 3 ----- null}
26 + -t 1.04 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 5 -a 0 -x {0.0 1.0 5 ----- null}
27 - -t 1.04 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 5 -a 0 -x {0.0 1.0 5 ----- null}
28 h -t 1.04 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 5 -a 0 -x {0.0 1.0 -1 ----- null}
29 r -t 1.04208 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 4 -a 0 -x {0.0 1.0 4 ----- null}
30 + -t 1.048 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 6 -a 0 -x {0.0 1.0 6 ----- null}
31 - -t 1.048 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 6 -a 0 -x {0.0 1.0 6 ----- null}
32 h -t 1.048 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 6 -a 0 -x {0.0 1.0 -1 ----- null}
33 r -t 1.05008 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 5 -a 0 -x {0.0 1.0 5 ----- null}
34 + -t 1.056 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 7 -a 0 -x {0.0 1.0 7 ----- null}
35 - -t 1.056 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 7 -a 0 -x {0.0 1.0 7 ----- null}
36 h -t 1.056 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 7 -a 0 -x {0.0 1.0 -1 ----- null}
37 r -t 1.05808 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 6 -a 0 -x {0.0 1.0 6 ----- null}
38 + -t 1.064 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 8 -a 0 -x {0.0 1.0 8 ----- null}
39 - -t 1.064 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 8 -a 0 -x {0.0 1.0 8 ----- null}
40 h -t 1.064 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 8 -a 0 -x {0.0 1.0 -1 ----- null}
41 r -t 1.06608 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 7 -a 0 -x {0.0 1.0 7 ----- null}
42 + -t 1.072 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 9 -a 0 -x {0.0 1.0 9 ----- null}
43 - -t 1.072 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 9 -a 0 -x {0.0 1.0 9 ----- null}
44 h -t 1.072 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 9 -a 0 -x {0.0 1.0 -1 ----- null}
45 r -t 1.07408 -s 0 -d 1 -p cbr -e 1000 -c 0 -i 8 -a 0 -x {0.0 1.0 8 ----- null}

```

Sample exercises can try

- Q1: Implement this diagram in NS2 and show the screenshots of:
 - .tr file
 - .nam file
 - Output flow using NAM



- Q2: Search for meaning of columns of trace file. What can we do with the trace files?

References:

- <https://www.isi.edu/nsnam/ns/>
- <https://www.tutorialsweb.com/ns2/NS2-1.htm>

THANK YOU

Q1: Let Host X has IP address 11.110.2.112 and Host Y has IP address 11.110.2.92 and both Host use the same subnet mask Z. Which of the subnet mask Z given below can be used if both hosts are in the same network?

options :

1. 225.225.255.224
2. 225.225.255.252
3. 225.225.255.192
4. 225.225.128.0

Q2. A subnet has been assigned a subnet mask of 255.255.255.224. What is the maximum number of hosts that can belong to this subnet?

Options:

1. 14
2. 30
3. 62
4. 126

Q3. A class B host address is to be split into a 5 bit subnet number. Choose the correct answer/s with number of subnets and maximum number of hosts in each subnet respectively?

Options:

1. 30 subnets and 2048 Hosts
2. 30 subnets and 2046 Hosts
3. 30 subnets and 1022 Hosts
4. 32 subnets and 2046 Hosts

Q4. Two computers C1 and C2 are configured as follows. C1 has IP address 142.16.128.0 and netmask 255.255.240.0. C2 has IP address 142.16.136.0 and netmask 255.255.248.0. Which one of the following statements is true?

Options:

1. C1 and C2 both assume they are on the same network
2. C2 assume C1 is on same network, but C1 assume C2 is on a different network
3. C1 assumes C2 is on same network, but C2 assume C1 is on a different network
4. C1 and C2 both assume they are on different networks

Q5. if a Class B network has a subnet mask of 255.255.254.0. What is the maximum number of hosts per subnet?

Options:

1. 2046
2. 1022
3. 510
4. 1024

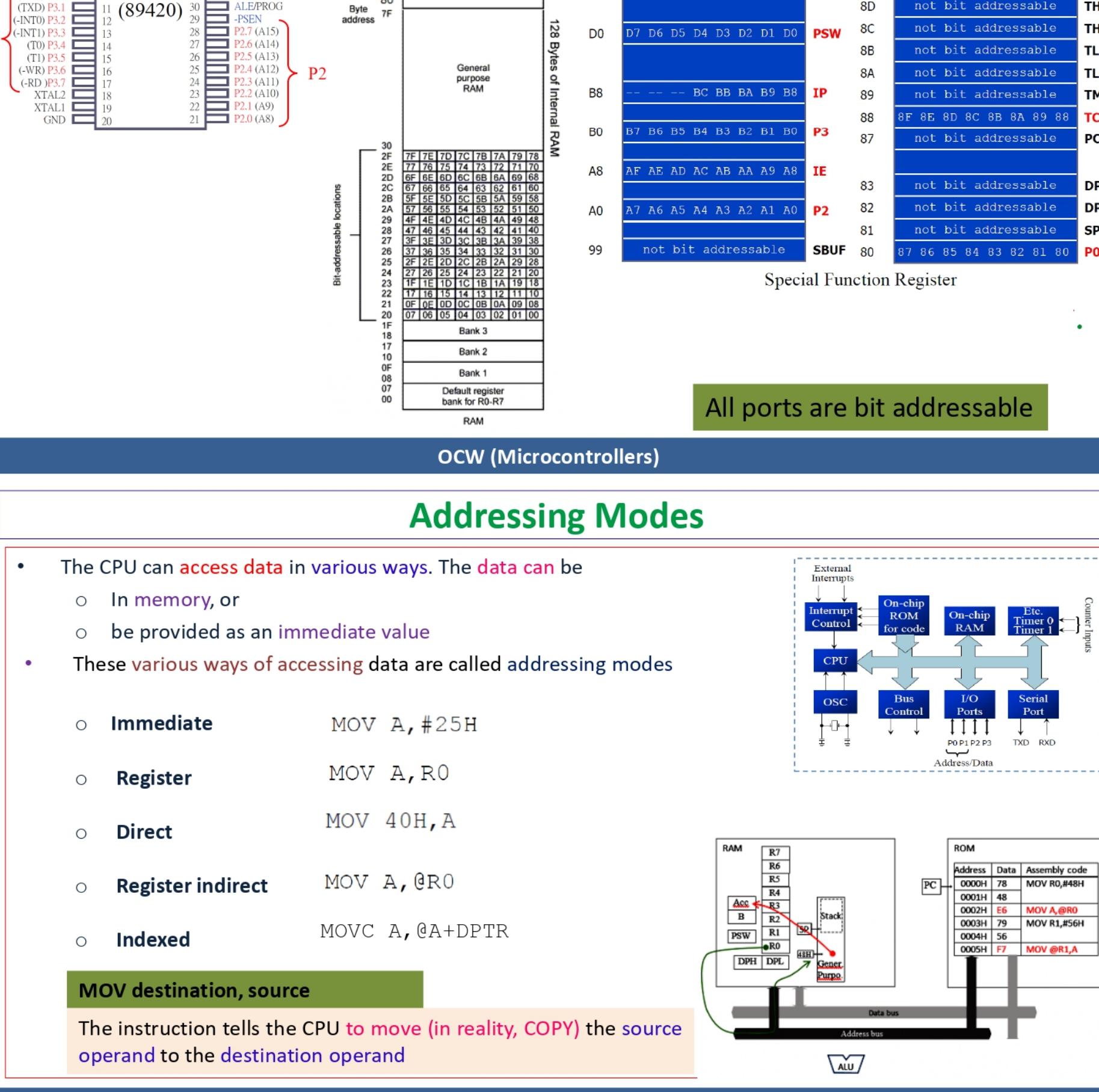
Warning notification!!!!

- The PPTs are prepared for the offline interactive teaching in the class using the materials from different books and web. The author may not have the legal permission for online sharing those materials via social media/web/email or use in business etc in public domain.
- Therefore, students are requested not to share the PPTs outside the class/institute, which can violation the copy-write related issues.

OCW (Microcontrollers)

Microcontroller

- Microcontroller**
 - Contains CPU with fixed amount of RAM, ROM, I/O ports, Timer embedded on a **single chip**
 - The **fixed amount of on-chip RAM, ROM, and number of I/O ports makes them ideal for many applications in which cost and space are critical**
 - In many applications, the **space it takes, the power it consumes, and the price per unit are much more critical considerations than the computing power**



OCW (Microcontrollers)

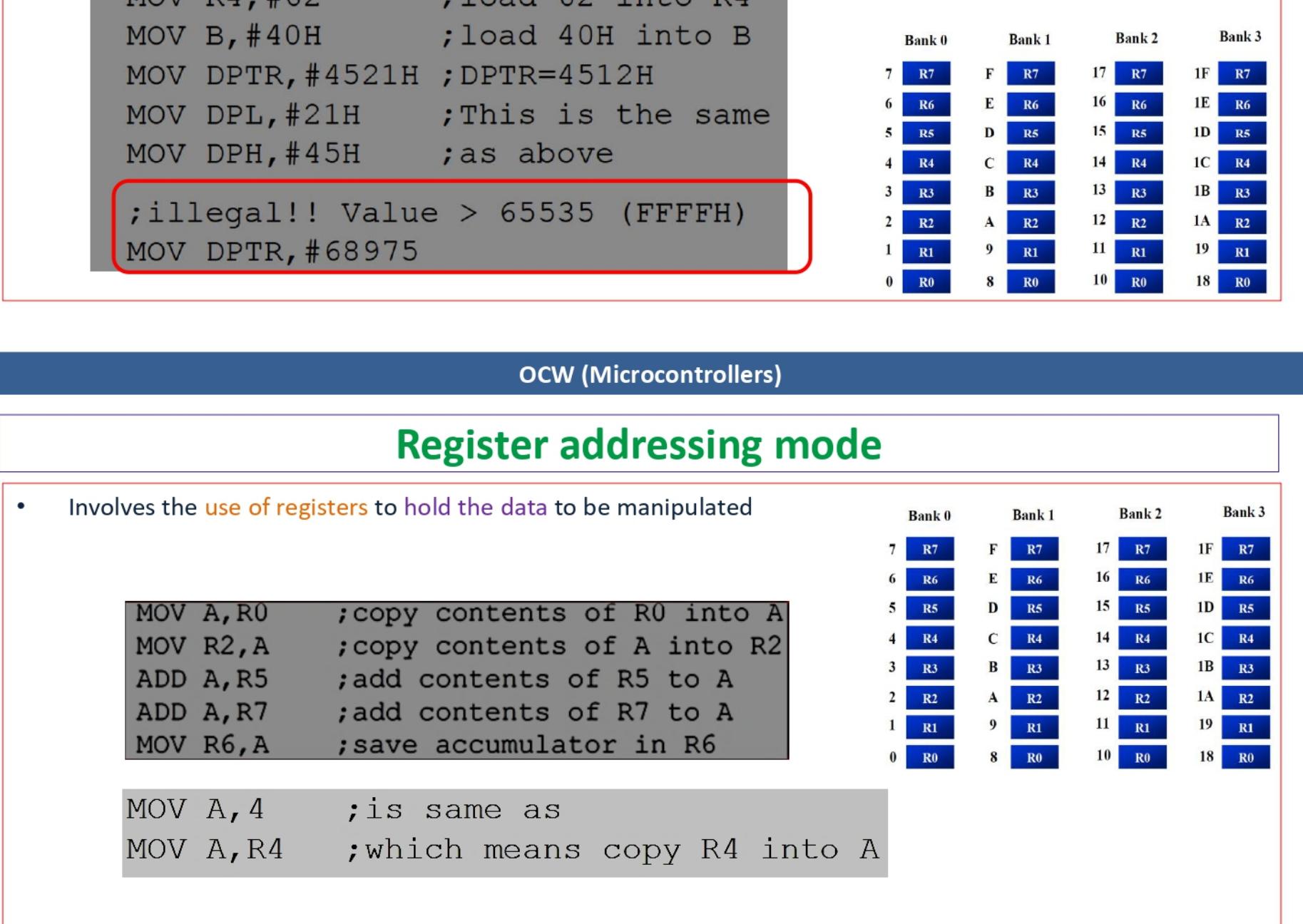
Microcontroller versus General-Purpose Microprocessor

- General-purpose microprocessors**
 - No I/O ports, ROM, I/O port
 - Must add** RAM, ROM, I/O ports, and timers **externally** to make them functional
 - Make the system bulkier and much more expensive
 - Have the advantage of versatility on the amount of RAM, ROM, and I/O ports

OCW (Microcontrollers)

CPU and other several supporting chips

CPU and others in a single chip



Application of microcontrollers

- An **embedded product** mostly uses **microcontroller** to do one task
- Home**
 - Appliances, intercom, telephones, security systems, garage door openers, answering machines, fax machines, home computers, TVs, cable TV tuner, VCRs, remote controls, video games, cellular phones, musical instruments, sewing machines, lighting control, paging, camera, pinball machines, toys, exercise equipment
- Office**
 - Telephones, computers, security systems, fax machines, microwave, copier, laser printer, color printer, paging
- Auto**
 - Engine control, air bag, instrumentation, security system, transmission control, entertainment, climate control, cellular phone, keyless entry

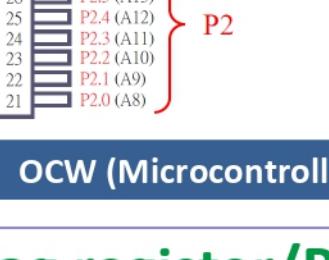
OCW (Microcontrollers)

8051 suppliers

- Intel (original)
- Atmel
- Philips/Sigetics
- AMD
- Infineon (formerly Siemens)
- Matra
- Dallas Semiconductor/Maxim

Keil 8051 simulation software (c51) at

<https://www.keil.com/download/product/>



Text book:

1. Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. McKinlay. **The 8051 Microcontroller and Embedded Systems: Using Assembly and C**, Pearson Education India; 2nd edition (1 January 2007)

OCW (Microcontrollers)

Microcontroller 8051

- Intel introduced 8051 in 1981
- The 8051 is an **8-bit processor**
 - The CPU can work on **only 8 bits** of data at a time
 - 128 bytes of RAM
 - 4K bytes of on-chip ROM
 - Two timers**
 - One serial port**
 - Four I/O ports, each 8 bits wide**
 - 16-bit address bus

OCW (Microcontrollers)

RAM memory space allocation

- CPU uses register to store information temporary
- A, B, R0, R1, R2, R3, R4, R5, R6, R7, DPTR (data pointer), and PC (program counter)
- All registered are **8-bits**, except DPTR and PC
- The job of the programmer to break down data larger than 8 bits

OCW (Microcontrollers)

RAM memory space allocation

- External Interrupts
- On-chip ROM for code
- On-chip RAM for data
- OSC
- Bus Control
- I/O Ports
- Serial Port
- Address Data

OCW (Microcontrollers)

MOV destination, source

The instruction tells the CPU to move (in reality, COPY) the source operand to the destination operand

OCW (Microcontrollers)

MOV destination, source

MOV A, #55H ;load value 55H into reg. A
MOV R0,A ;copy contents of A into R0
;now A=55H

signifies that it is a value

OCW (Microcontrollers)

Immediate addressing mode

- The source operand is a constant
- Operand comes **immediate after the opcode** (preceded by # sign)
- Can load information into any registers, including 16-bit DPTR register

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

MOV R0,A ;copy contents of A into R0
;now A=R0=55H

OCW (Microcontrollers)

MOV A, #55H ;load value 55H into reg. A

Warning notification!!!!

- The PPTs are prepared for the offline interactive teaching in the class using the materials from different books and web. The author may not have the legal permission for online sharing those materials via social media/web/email or use in business etc in public domain.
- Therefore, students are requested not to share the PPTs outside the class/institute, which can violation the copy-write related issues.

OCW (Microcontrollers)

Embedded C

- Embedded C is an extension of C Program Language
- Commonly used in development of Embedded Systems
- Popular compiler for embedded C are Keil, Green Hill software

Advantage of C over Assembly

- Easier and less time consuming to write program in C than Assembly
- C is easier to modify
- C provide support to use the available libraries
- C code is portable to other microcontrollers with little or no modification

C data types widely used for the 8051

- A data type declares the type of data i.e. variable can store such as integer, character
- C programmers is to understand the data types → create smaller Hex files

Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
(signed) char	8-bit	-128 to +127
unsigned int	16-bit	0 to 65535
(signd) int	16-bit	-32,768 to +32,767
sbit	1-bit	SFR bit-addressable only
bit	1-bit	RAM bit-addressable only
sfr	8-bit	RAM addresses 80 - FFH only

OCW (Microcontrollers)

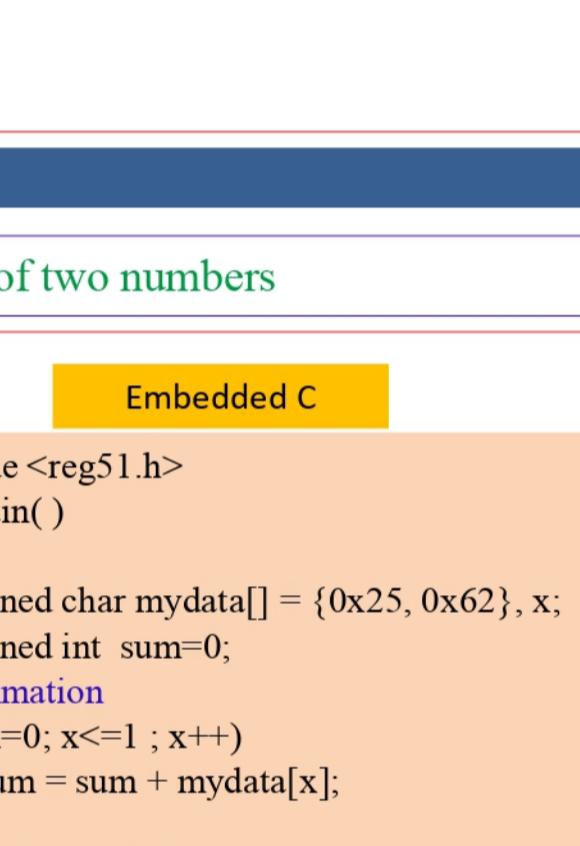
Write an embedded C program to load a number into Accumulator, and Port 1, 2

Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
(signed) char	8-bit	-128 to +127
unsigned int	16-bit	0 to 65535
(signd) int	16-bit	-32,768 to +32,767
sbit	1-bit	SFR bit-addressable only
bit	1-bit	RAM bit-addressable only
sfr	8-bit	RAM addresses 80 - FFH only

```
unsigned char i = 100; // 100 as a base 10 literal  
unsigned char j = 0x64; // 100 in hex, indicated by leading 0x
```

Assembly

```
ORG 0000H ; start at 0000H  
MOV A, #06H ; load 06H into ACC  
MOV P1, #55H ; send 55H to port 1  
MOV P2, #AAH ; send AAH to port 2  
END ; end of code
```



Embedded C

```
# include <reg51.h> // Header file for 8051  
void main()  
{  
    ACC = 0x06; // load 06H into ACC  
    P1 = 0x55; // send 55H to port 1  
    P2 = 0xAA; // send AAH value to port 2  
}
```

OCW (Microcontrollers)

Write an 8051 C program to send values 00 – FF continuously to port P1

Assembly

```
ORG 0000H ; start at 0000H  
MOV R0, #00H ; Initialize R0 as the counter  
  
Again: MOV P1, R0 ; Send the R0 contents to P1  
    INC R0 ; R0 = R0 + 1 (incremented by 1)  
    SJMP Again ; move execution to label "Again"  
  
END
```

Embedded C

```
# include <reg51.h>  
void main()  
{  
    unsigned char z; // z → 0 to FF (255)  
    while(1) // repeat forever  
    {  
        for (z = 0; z <= 255; z++)  
            P1=z; // send/assign z value to port  
    }  
}
```

OCW (Microcontrollers)

Write 8051 C program to toggle only bit P2.4 continuously without disturbing the rest of the bits of P2

Assembly

```
ORG 0000H ; start at 0000H  
Again: SETB P2.4 ; Set the pin/bit of P2.4 High  
    CLR P2.4 ; clear the pin/bit of P2.4  
    SJMP Again ; repeat the process "Again"  
  
END
```

Embedded C

```
# include <reg51.h>  
sbit mybit = P2^4; // declare single bit  
void main()  
{  
    while(1) // repeat forever  
    {  
        mybit=1; // set/turn-on P2.4  
        mybit=0; // clear/turn-off P2.4  
    }  
}
```

OCW (Microcontrollers)

Write 8051 C program to get byte from P1 continuously

Assembly

```
ORG 0000H ; start at 0000H  
MOV P1, #0FFH ; write all 1's in port P1 to make I/P  
Again: MOV A, P1 ; get byte from P1 into ACC  
    SJMP Again ; repeat the process "Again"  
  
END
```

Embedded C

```
# include <reg51.h>  
void main()  
{  
    unsigned char mybit; // define a variable mybit  
    P1 = 0xFF; // make P1 as input port  
    while(1) // repeat forever  
    {  
        mybit=P1; // get byte from P1  
    }  
}
```

OCW (Microcontrollers)

Write an 8051 C program to convert 11111101 (FD Hex) to decimal and display the digits on P0, P1 and P2

Quotient Remainder
FD/10 19H = (25)10 3 (low digit) LSD
19/10 2H 5 (middle digit)
 2 (high digit) (MSD)

Assembly

```
ORG 0000H ; start at 0000H  
MOV A, #0FDH ; load the hex value into A  
MOV B, #10H ; move 10 into B  
DIV AB ; A/B → A (Quotient), B (remainder)
```

$(FD)_{16} = (253)_{10} \rightarrow \text{Display } 2\ 5\ 3$

Embedded C

```
# include <reg51.h>
```

OCW (Microcontrollers)

Write an 8051 C program to convert 11111101 (FD Hex) to decimal and display the digits on P0, P1 and P2

Quotient Remainder
FD/10 19H = (25)10 3 (low digit) LSD
19/10 2H 5 (middle digit)
 2 (high digit) (MSD)

Assembly

```
ORG 0000H ; start at 0000H  
MOV A, #0FDH ; load the hex value into A  
MOV B, #10H ; move 10 into B  
DIV AB ; A/B → A (Quotient), B (remainder)
```

$(FD)_{16} = (253)_{10} \rightarrow \text{Display } 2\ 5\ 3$

Embedded C

```
# include <reg51.h>
```

OCW (Microcontrollers)

Write an 8051 C program to convert 11111101 (FD Hex) to decimal and display the digits on P0, P1 and P2

Quotient Remainder
FD/10 19H = (25)10 3 (low digit) LSD
19/10 2H 5 (middle digit)
 2 (high digit) (MSD)

Assembly

```
ORG 0000H ; start at 0000H  
MOV A, #0FDH ; load the hex value into A  
MOV B, #10H ; move 10 into B  
DIV AB ; A/B → A (Quotient), B (remainder)
```

$(FD)_{16} = (253)_{10} \rightarrow \text{Display } 2\ 5\ 3$

Embedded C

```
# include <reg51.h>
```

OCW (Microcontrollers)

Write an 8051 C program to convert 11111101 (FD Hex) to decimal and display the digits on P0, P1 and P2

Quotient Remainder
FD/10 19H = (25)10 3 (low digit) LSD
19/10 2H 5 (middle digit)
 2 (high digit) (MSD)

Assembly

```
ORG 0000H ; start at 0000H  
MOV A, #0FDH ; load the hex value into A  
MOV B, #10H ; move 10 into B  
DIV AB ; A/B → A (Quotient), B (remainder)
```

$(FD)_{16} = (253)_{10} \rightarrow \text{Display } 2\ 5\ 3$

Embedded C

```
# include <reg51.h>
```

OCW (Microcontrollers)

Write an 8051 C program to convert 11111101 (FD Hex) to decimal and display the digits on P0, P1 and P2

Quotient Remainder
FD/10 19H = (25)10 3 (low digit) LSD
19/10 2H 5 (middle digit)
 2 (high digit) (MSD)

Assembly

```
ORG 0000H ; start at 0000H  
MOV A, #0FDH ; load the hex value into A  
MOV B, #10H ; move 10 into B  
DIV AB ; A/B → A (Quotient), B (remainder)
```

$(FD)_{16} = (253)_{10} \rightarrow \text{Display } 2\ 5\ 3$

Embedded C

```
# include <reg51.h>
```

OCW (Microcontrollers)

Write an 8051 C program to convert 11111101 (FD Hex) to decimal and display the digits on P0, P1 and P2

Quotient Remainder
FD/10 19H = (25)10 3 (low digit) LSD
19/10 2H 5 (middle digit)
 2 (high digit) (MSD)

Assembly

```
ORG 0000H ; start at 0000H  
MOV A, #0FDH ; load the hex value into A  
MOV B, #10H ; move 10 into B  
DIV AB ; A/B → A (Quotient), B (remainder)
```

$(FD)_{16} = (253)_{10} \rightarrow \text{Display } 2\ 5\ 3$

Embedded C

```
# include <reg51.h>
```

OCW (Microcontrollers)

Write an 8051 C program to convert 11111101 (FD Hex) to decimal and display the digits on P0, P1 and P2

Quotient Remainder
FD/10 19H = (25)10 3 (low digit) LSD
19/10 2H 5 (middle digit)
 2 (high digit) (MSD)

Assembly

```
ORG 0000H ; start at 0000H  
MOV A, #0FDH ; load the hex value into A  
MOV B, #10H ; move 10 into B  
DIV AB ; A/B → A (Quotient), B (remainder)
```

$(FD)_{16} = (253)_{10} \rightarrow \text{Display } 2\ 5\ 3$

Embedded C

```
# include <reg51.h>
```

OCW (Microcontrollers)

Write an 8051 C program to convert 11111101 (FD Hex) to decimal and display the digits on P0, P1 and P2

Quotient Remainder
FD/10 19H = (25)10 3 (low digit) LSD
19/10 2H 5 (middle digit)
 2 (high digit) (MSD)

Assembly

```
ORG 0000H ; start at 0000H  
MOV A, #0FDH ; load the hex value into A  
MOV B, #10H ; move 10 into B  
DIV AB ; A/B → A (Quotient), B (remainder)
```

$(FD)_{16} = (253)_{10} \rightarrow \text{Display } 2\ 5\ 3$

Embedded C

```
# include <reg51.h>
```

OCW (Microcontrollers)

Write an 8051 C program to convert 11111101 (FD Hex) to decimal and display the digits on P0, P1 and P2

Quotient Remainder
FD/10 19H = (25)10 3 (low digit) LSD
19/10 2H 5 (middle digit)

Warning notification!!!!

- The PPTs are prepared for the offline interactive teaching in the class using the materials from different books and web. The author may not have the legal permission for online sharing those materials via social media/web/email or use in business etc in public domain.
- Therefore, students are requested not to share the PPTs outside the class/institute, which can violation the copy-write related issues.

OCW (Microcontrollers)

Programming structure of 8051

[label:] Mnemonic [operands] [:comment]		
1 0000 ORG 0H ;start (origin) at 0		
2 0000 7D25 MOV R5, #25H ;load 25H into R5		
3 0002 7F34 MOV R7, #34H ;load 34H into R7		
4 0004 7400 MOV A, #0 ;load 0 into A		
5 0006 2D ADD A, R5 ;add contents of R5 to A		
6 0007 2F ADD A, R7 ;add contents of R7 to A		;now A = A + R5
7 0008 2412 ADD A, #12H ;add to A value 12H		;now A = A + 12H
8 000A 80EF HERE: SJMP HERE ;stay in this loop		
9 000C END ;end of assembly source file		

ORG - directives do not generate any machine code and are used only by the assembler

HERE: The label field allows the program to refer to a line of code by name

All 8051 members start at memory address 0000 when they're powered Up

PC has the value of 0000

The first opcode is burned into ROM address 0000H, the 8051 looks for the first instruction when it is booted

This is achieved by the ORG statement in the source program

OCW (Microcontrollers)

Directives

- DB (define byte) : Used to define data: can be in decimal, binary, Hex or ascii

ORG 500H		
DATA1: DB 28 ; DECIMAL (1C in Hex);	MOV A, #28	
DATA2: DB 00110101B ; BINARY (35 in Hex);	MOV A, #00110101B	
DATA3: DB 39H ; HEX	MOV A, #39H	
DATA4: DB "2591" ; ASCII NUMBERS	MOV A, #"2591"	

- ORG (origin): The ORG directive is used to indicate the beginning of the address

OCW (Microcontrollers)

- END : This indicates to the assembler the end of the source (asm) file
 - The END directive is the last line of an 8051 program

- EQU (equate): This is used to define a constant without occupying a memory location
 - It does not set aside storage for a data item but associates a constant value with a data label
 - When the label appears in the program, its constant value will be substituted for the label

```
COUNT EQU 25
...
MOV R3, #COUNT
```

ROM memory map 8051:

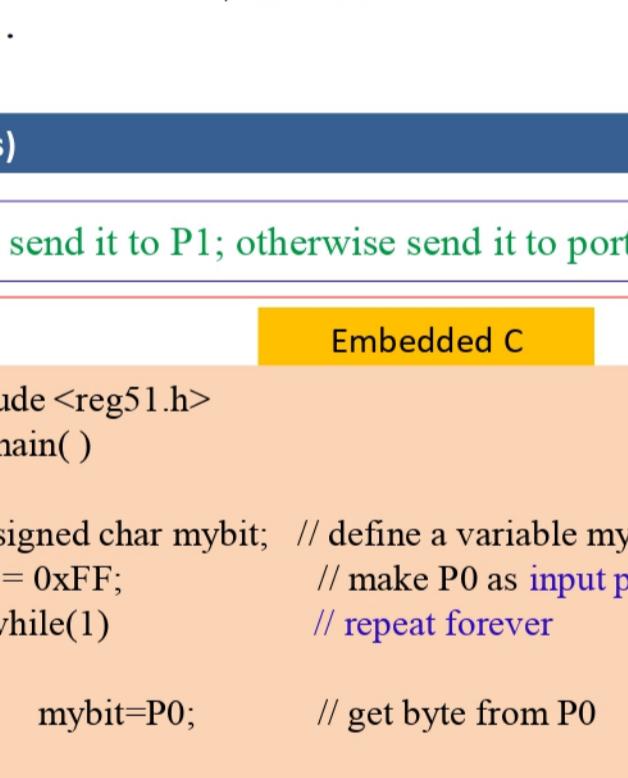
- PC is a 16-bit register and hence it can support 0000 to FFFFH (i.e 64 k bytes)
- But 8051 has 4k bytes ROM memory, so first address 0000 and last address OFFFH



OCW (Microcontrollers)

Execution of code

- After the program is burned into ROM
- the opcode and operand are placed in ROM memory location starting at 0000



ROM contents

Address	Code
0000	7D
0001	25
0002	7F
0003	34
0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

- When 8051 is powered up, the PC has 0000 and starts to fetch the first opcode from location 0000 of program ROM
 - Upon executing the opcode 7D, the CPU fetches the value 25 and places it in R5
 - Then the PC is incremented to point to 0002, containing opcode 7F

OCW (Microcontrollers)

ROM contents

Address	Code
0000	7D
0001	25
0002	7F
0003	34
0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

- Upon executing the opcode 7F, the value 34H is moved into R7
- The PC is incremented to 0004
- The instruction at location 0004 is executed and now PC = 0006

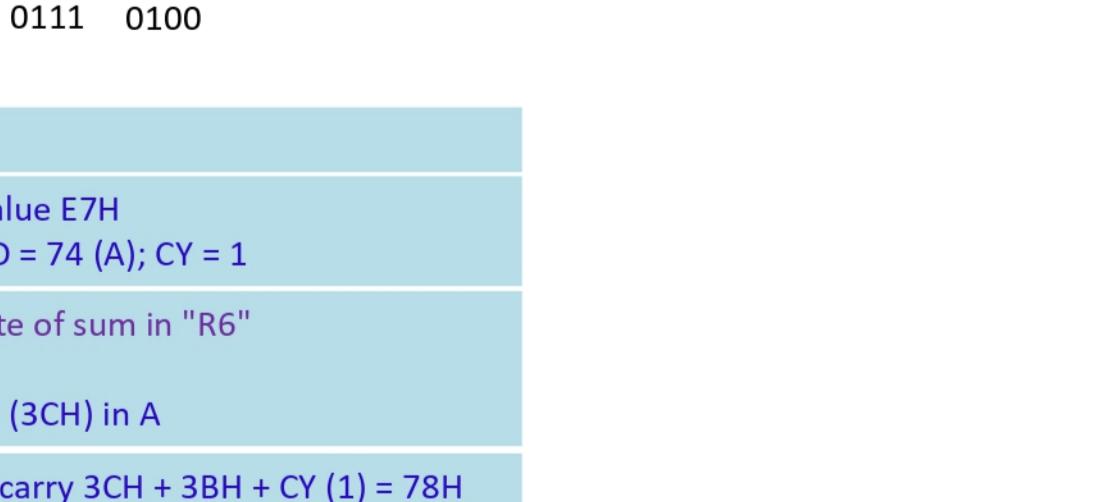
OCW (Microcontrollers)

- Write a program to read the temperature and test it for the value 75. According to the test results, place the temperature value into the registers indicated by the following.

If T = 75 then A = 75

If T < 75 then R1 = T

If T > 75 then R2 = T



OCW (Microcontrollers)

Compiler: High-level code → Machine code

- Compiler: is a computer program that reads the program written in a source (high) language, translates it into the machine code



```
int main()
{
    // Variable declaration
    int a, b, sum;
    // Take two numbers as input from the user
    scanf("%d %d", &a, &b);
    // Add the numbers and assign the value
    sum = a + b;
    // Use the calculated value
    printf("%d\n", sum);
    return 0;
}
```



```
PROGRAM HIGH LEVEL
    ↓
    CONVERTER
    ↓
    MACHINE CODE LOW LEVEL
    ↓
    Binary Machine Code
```

OCW (Microcontrollers)

- CJNE (compare and jump if not equal)

CJNE destination,source

- The destination operand can be in the accumulator or in one of the Rn registers
- The source operand can be in a register, in memory, or immediate
- It changes the CY flag to indicate if the destination operand is larger or smaller

Compare	Carry Flag
destination > source	CY = 0
destination < source	CY = 1

```
CJNE R5,#80,NOT_EQUAL ;check R5 for 80
    ...
NOT_EQUAL:
    JNC NEXT ;jump if R5 > 80
    ...
NEXT: ...
```

OCW (Microcontrollers)

- Difference between Assembler Vs Compiler

Assembler	Compiler
Assembler converts the assembly language to the machine language	Compiler converts the source code (high language) written by the programmer to a machine language
Input to the assembler is assembly language code	Input to the compiler is high level source code
It is difficult to debug	It is easy to debugging

OCW (Microcontrollers)

- Execution of code

- After the program is burned into ROM
- the opcode and operand are placed in ROM memory location starting at 0000

END : This indicates to the assembler the end of the source (asm) file

o The END directive is the last line of an 8051 program

- ORG (origin): The ORG directive is used to indicate the beginning of the address

- EQU (equate): This is used to define a constant without occupying a memory location

- It does not set aside storage for a data item but associates a constant value with a data label

- When the label appears in the program, its constant value will be substituted for the label

```
COUNT EQU 25
...
MOV R3, #COUNT
```

OCW (Microcontrollers)

- ROM memory map 8051:

ROM memory map 8051:

- PC is a 16-bit register and hence it can support 0000 to FFFFH (i.e 64 k bytes)

- But 8051 has 4k bytes ROM memory, so first address 0000 and last address OFFFH

OCW (Microcontrollers)

- Execution of code

- After the program is burned into ROM
- the opcode and operand are placed in ROM memory location starting at 0000

END : This indicates to the assembler the end of the source (asm) file

- ORG (origin): The ORG directive is used to indicate the beginning of the address

- EQU (equate): This is used to define a constant without occupying a memory location

- It does not set aside storage for a data item but associates a constant value with a data label

- When the label appears in the program, its constant value will be substituted for the label

```
COUNT EQU 25
...
MOV R3, #COUNT
```

OCW (Microcontrollers)

- ROM memory map 8051:

ROM memory map 8051:

- PC is a 16-bit register and hence it can support 0000 to FFFFH (i.e 64 k bytes)

- But 8051 has 4k bytes ROM memory, so first address 0000 and last address OFFFH

OCW (Microcontrollers)

- Execution of code

- After the program is burned into ROM
- the opcode and operand are placed in ROM memory location starting at 0000

END : This indicates to the assembler the end of the source (asm) file

- ORG (origin