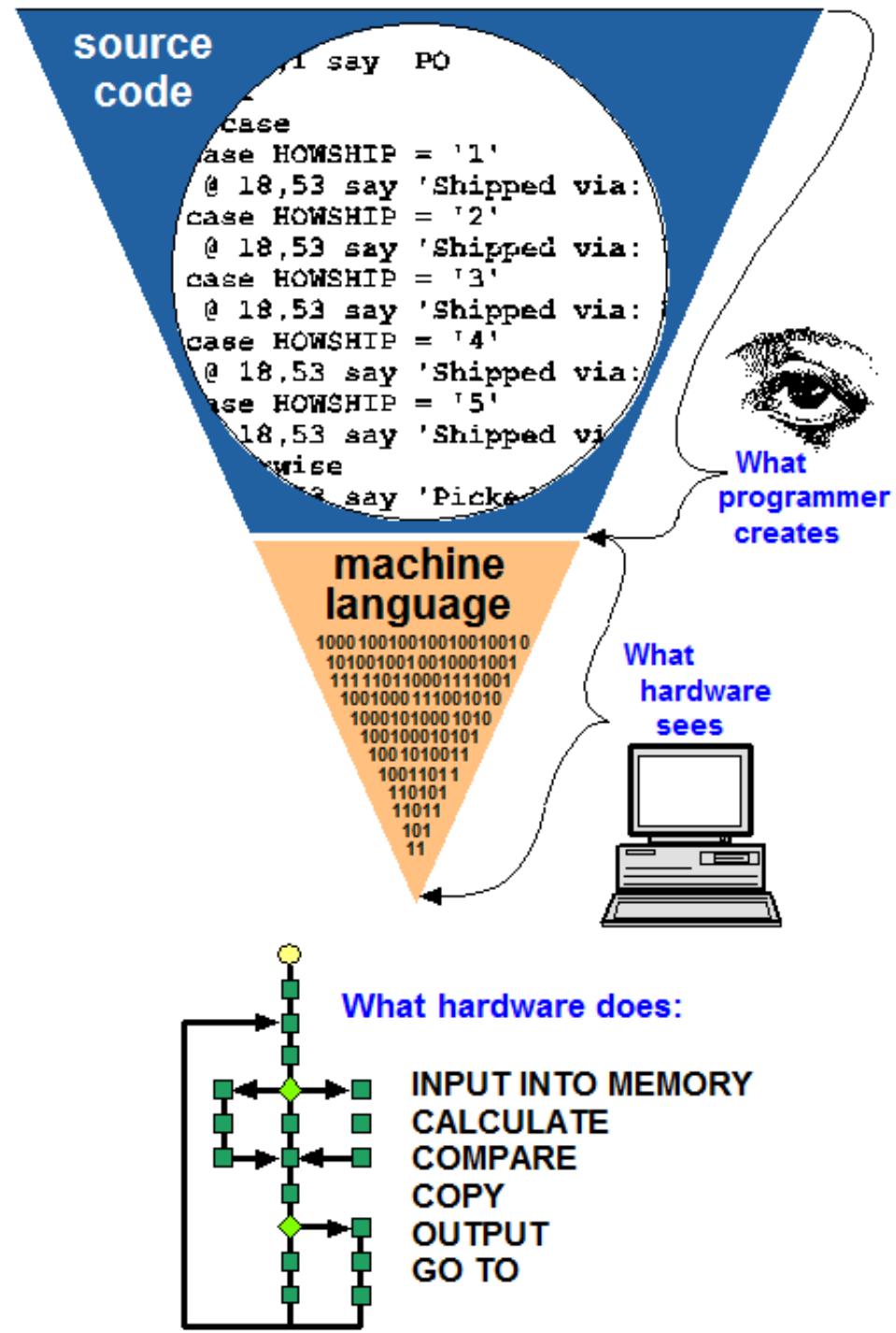


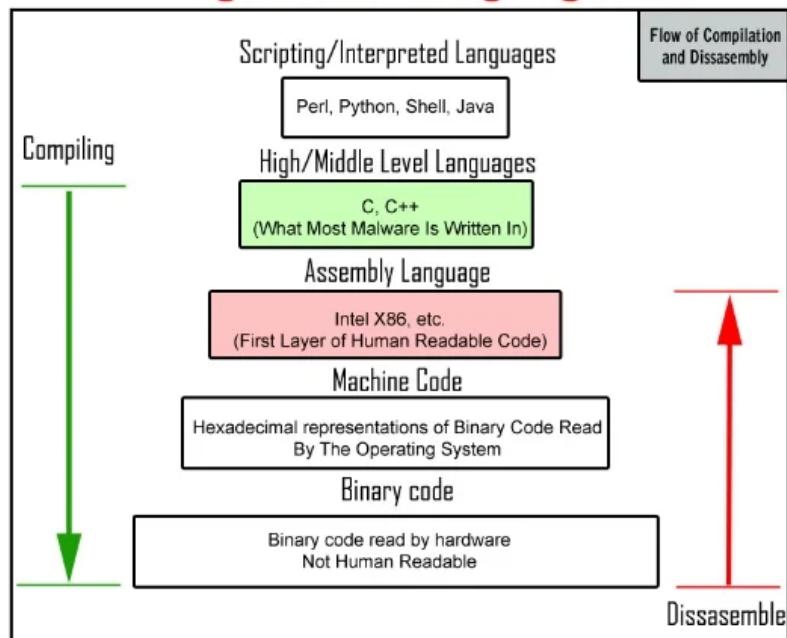
# Computer Science

Programming Language -> Communicate with Computer

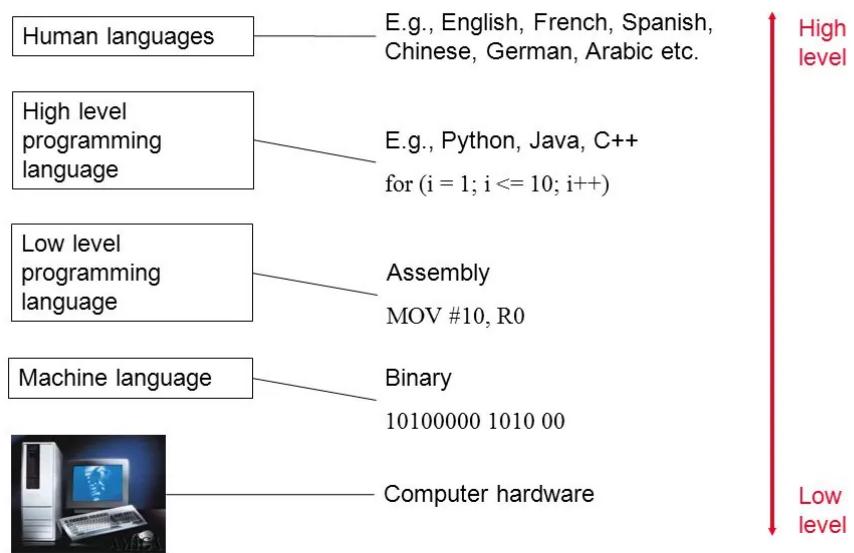
## SOURCE CODE TO MACHINE LANGUAGE



## High Level Languages

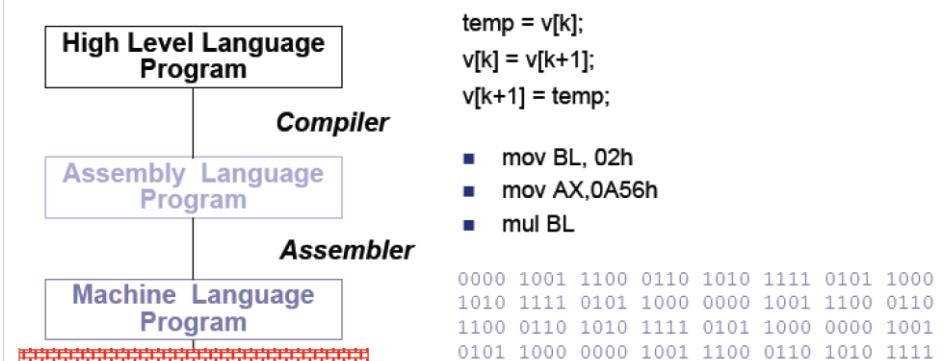


## High Vs. Low Level Languages



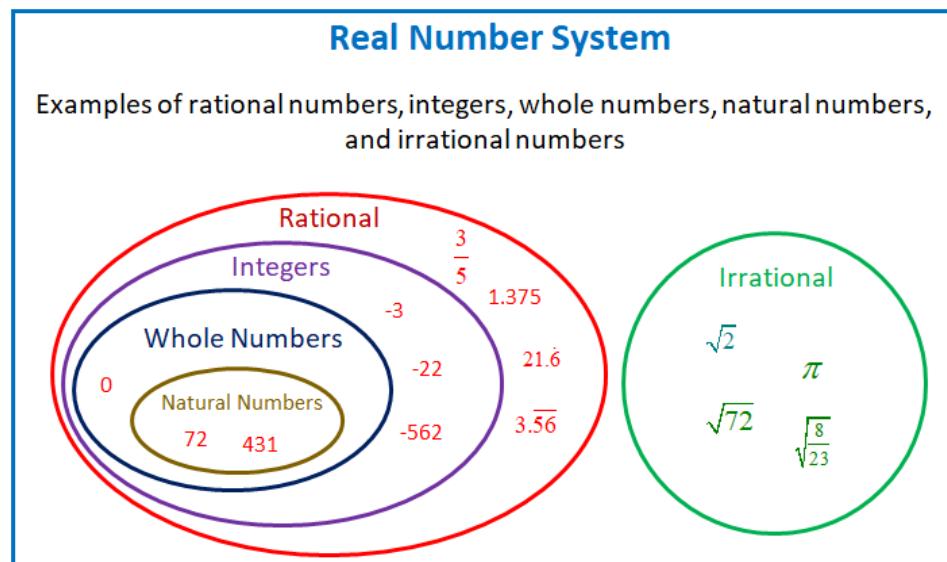
James Tam

## Levels of Representation



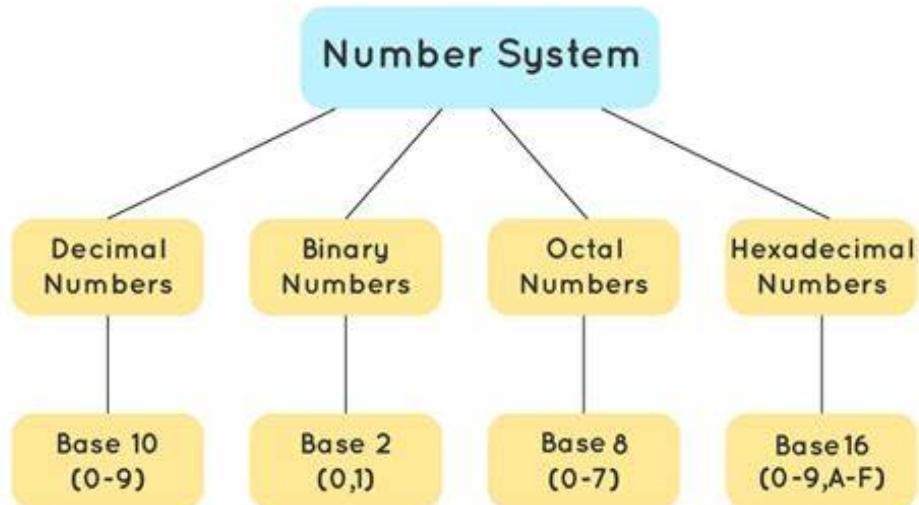
# Number System

## Real Number System



## Number System Conversion

### Types of Number System



# Number Systems Conversion Chart

Binary																
Place	$2^{11}$	$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$
Weight	2048	1024	512	256	128	64	32	16	8	4	2	1	0.5	0.25	0.125	0.0625

## Binary, Hex, and Octal Conversions

Binary	Octal	Hexadecimal	Decimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	A	10
1011	13	B	11
1100	14	C	12
1101	15	D	13
1110	16	E	14
1111	17	F	15

## Methodology

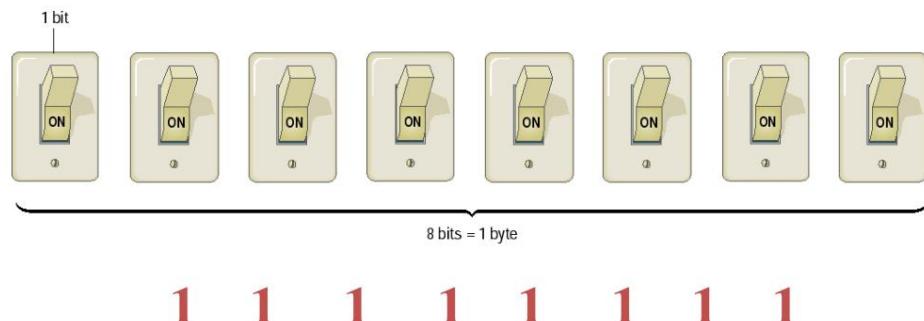
- Convert from decimal to binary  
Divide the decimal by the largest binary weight it is divisible by and place a "1" in that column. Then select the next largest weight, if it is divisible put a "1" in that column otherwise place a "0" in the column. Continue until all the columns have either a "1" or "0" resulting in a binary expression.
- Convert from decimal to hex.  
Convert to binary first, then group the binary in groups of 4 beginning on the right working to the left. For each group determine the hex value based on the table to the left.
- Convert to octal  
Convert to binary first, then group the binary in groups of 3 beginning on the right working to the left. For each group determine the octal value based on the table to the left.

# Binary Number System

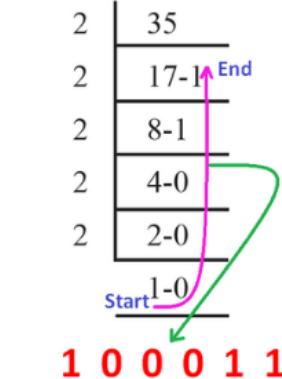
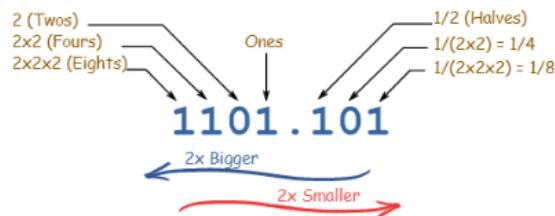
## Number System

### Bits and Bytes

- A single unit of data is called a bit, having a value of 1 or 0.
- Computers work with collections of bits, grouping them to represent larger pieces of data, such as letters of the alphabet.
- Eight bits make up one byte. A byte is the amount of memory needed to store one alphanumeric character.
- With one byte, the computer can represent one of 256 different symbols or characters.



# What is the Binary Number System?



**Electrical 4 U**

## Complement

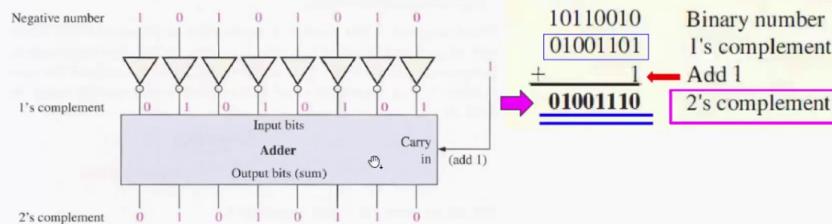
### 2.5) 1'S and 2'S Complements of Binary Numbers

#### Finding the 2's Complement

The 2's complement of a binary number is found by adding 1 to the LSB of the 1's complement.

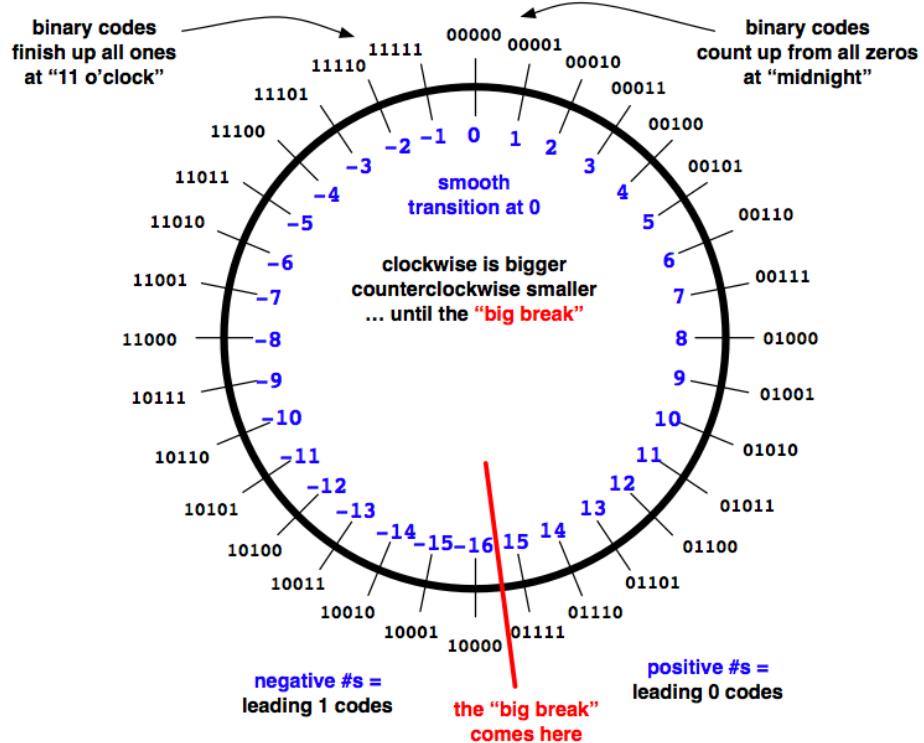
$$\text{2's complement} = (\text{1's complement}) + 1$$

**Example 17:** Find the 2's Complement of 10110010



Source: Digital Fundamentals by Thomas L. Floyd

27



# Complements of numbers

## (r-1 )'s Complement

- Given a number  $N$  in base  $r$  having  $n$  digits,
- the  $(r- 1)$ 's complement of  $N$  is defined as

$$(r^n - 1) - N$$

- For decimal numbers the base or  $r = 10$  and  $r- 1 = 9$ ,

- so the 9's complement of  $N$  is  $(10^n - 1) - N$

- $99999..... - N$

9	9	9	9	9
Digit n	Digit n-1	Next digit	Next digit	First digit

## Complements

- There are two types of complements for each base- $r$  system: the radix complement and diminished radix complement.

→ the  $r$ 's complement and the second as the  $(r - 1)$ 's complement.

### Diminished Radix Complement

Given a number  $N$  in case  $r$  having  $n$  digits, the  $(r - 1)$ 's complement of  $N$  is defined as  $(r^n - 1) - N$ . For decimal numbers,  $r = 10$  and  $r - 1 = 9$ , so the 9's complement of  $N$  is  $(10^n - 1) - N$ .

#### Example:

The 9's complement of 546700 is  $999999 - 546700 = 453299$ .

The 9's complement of 012398 is  $999999 - 012398 = 987601$ .

- For binary numbers,  $r = 2$  and  $r - 1 = 1$ , so the 1's complement of  $N$  is  $(2^n - 1) - N$ .

#### Example:

The 1's complement of 1011000 is 0100111

The 1's complement of 0101101 is 1010010

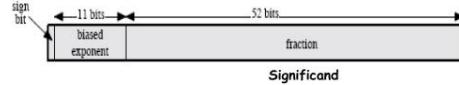
## Precision floating-point

## Double Precision Floating Point Numbers IEEE Standard

64 bit Double Precision Floating Point Numbers are stored as:

S EEEEEEEEEE FFFFFFFF...FFFF...FFFF...FFFF...FFFF...FFFF...FFFF...FFFF

S: Sign - 1 bit  
E: Exponent - 11 bits  
F: Fraction - 52 bits



The value V:

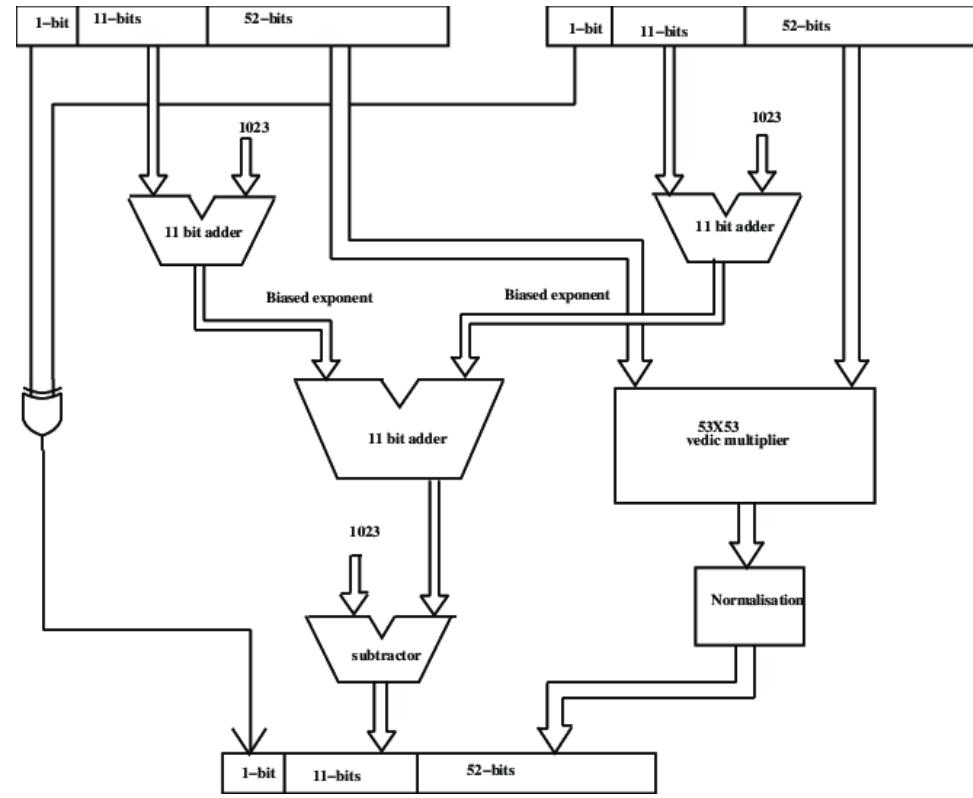
- If E=2047 and F is nonzero, then V= NaN ("Not a Number")
- If E=2047 and F is zero and S is 1, then V= - Infinity
- If E=2047 and F is zero and S is 0, then V= Infinity
- If 0<E<2047 then V= (-1)\*\*S \* 2 \*\* (E-1023) \* (1.F) (exponent range = -1023 to +1024)
- If E=0 and F is nonzero, then V= (-1)\*\*S \* 2 \*\* (-1022) \* (0.F) ("unnormalized" values)
- If E=0 and F is zero and S is 1, then V= - 0
- If E=0 and F is zero and S is 0, then V= 0

Note: 2047 decimal = 1111111111 in binary (11 bits)

## Biased Exponent - Cont'd

- ❖ For double precision, exponent field is 11 bits
  - ◊ E can be in the range 0 to 2047
  - ◊ E = 0 and E = 2047 are reserved for special use
  - ◊ E = 1 to 2046 are used for normalized floating point numbers
  - ◊ Bias = 1023 (half of 2046), val(E) = E - 1023
  - ◊ val(E=1) = -1022, val(E=1023) = 0, val(E=2046) = 1023
- ❖ Value of a Normalized Floating Point Number is

$$\begin{aligned}
 & (-1)^S \times (1.F)_2 \times 2^{E - \text{Bias}} \\
 & (-1)^S \times (1.f_1 f_2 f_3 f_4 \dots)_2 \times 2^{E - \text{Bias}} \\
 & (-1)^S \times (1 + f_1 \times 2^{-1} + f_2 \times 2^{-2} + f_3 \times 2^{-3} + f_4 \times 2^{-4} \dots)_2 \times 2^{E - \text{Bias}}
 \end{aligned}$$



## ASCII

ASCII control characters			ASCII printable characters												Extended ASCII characters											
00	NULL	(Null character)	32	space	64	@	96	'	128	ç	160	á	192	l	224	ó										
01	SOH	(Start of Header)	33	!	65	A	97	a	129	ü	161	í	193	ł	225	ó										
02	STX	(Start of Text)	34	"	66	B	98	b	130	é	162	ó	194	ł	226	ó										
03	ETX	(End of Text)	35	#	67	C	99	c	131	ä	163	ú	195	ł	227	ó										
04	EOT	(End of Trans.)	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	ó										
05	ENQ	(Enquiry)	37	%	69	E	101	e	133	à	165	N	197	+	229	ó										
06	ACK	(Acknowledgement)	38	&	70	F	102	f	134	à	166	»	198	á	230	ú										
07	BEL	(Bell)	39	'	71	G	103	g	135	ç	167	°	199	À	231	ó										
08	BS	(Backspace)	40	(	72	H	104	h	136	é	168	ż	200	ł	232	ó										
09	HT	(Horizontal Tab)	41	)	73	I	105	i	137	è	169	®	201	ú	233	ó										
10	LF	(Line feed)	42	*	74	J	106	j	138	è	170	¬	202	ł	234	ó										
11	VT	(Vertical Tab)	43	+	75	K	107	k	139	í	171	½	203	ł	235	ó										
12	FF	(Form feed)	44	,	76	L	108	l	140	í	172	¼	204	ł	236	ó										
13	CR	(Carriage return)	45	-	77	M	109	m	141	í	173	ı	205	=	237	ó										
14	SO	(Carriage return)	46	.	78	N	110	n	142	À	174	«	206	‡	238	—										
15	SI	(Shift In)	47	/	79	O	111	o	143	Á	175	»	207	■	239	·										
16	DLE	(Data link escape)	48	0	80	P	112	p	144	É	176	²	208	ó	240	≡										
17	DC1	(Device control 1)	49	1	81	Q	113	q	145	æ	177	...	209	đ	241	±										
18	DC2	(Device control 2)	50	2	82	R	114	r	146	Æ	178	...	210	é	242	¾										
19	DC3	(Device control 3)	51	3	83	S	115	s	147	ö	179	...	211	ë	243	¾										
20	DC4	(Device control 4)	52	4	84	T	116	t	148	ö	180	—	212	é	244	¶										
21	NAK	(Negative acknowl.)	53	5	85	U	117	u	149	ö	181	À	213	ı	245	§										
22	SYN	(Synchronous idle)	54	6	86	V	118	v	150	ö	182	Á	214	ı	246	÷										
23	ETB	(End of trans. block)	55	7	87	W	119	w	151	ö	183	À	215	ı	247	·										
24	CAN	(Cancel)	56	8	88	X	120	x	152	ý	184	®	216	ý	248	·										
25	EM	(End of medium)	57	9	89	Y	121	y	153	ó	185	—	217	ı	249	..										
26	SUB	(Substitute)	58	:	90	Z	122	z	154	ö	186	—	218	—	250	.										
27	ESC	(Escape)	59	;	91	[	123	{	155	ø	187	—	219	—	251	·										
28	FS	(File separator)	60	<	92	\	124		156	£	188	—	220	—	252	·										
29	GS	(Group separator)	61	=	93	]	125	}	157	ø	189	¢	221	—	253	·										
30	RS	(Record separator)	62	>	94	^	126	~	158	×	190	¥	222	—	254	■										
31	US	(Unit separator)	63	?	95	—			159	f	191	—	223	—	255	nbsp										
127	DEL	(Delete)							159	f	191	—	223	—	255	nbsp										

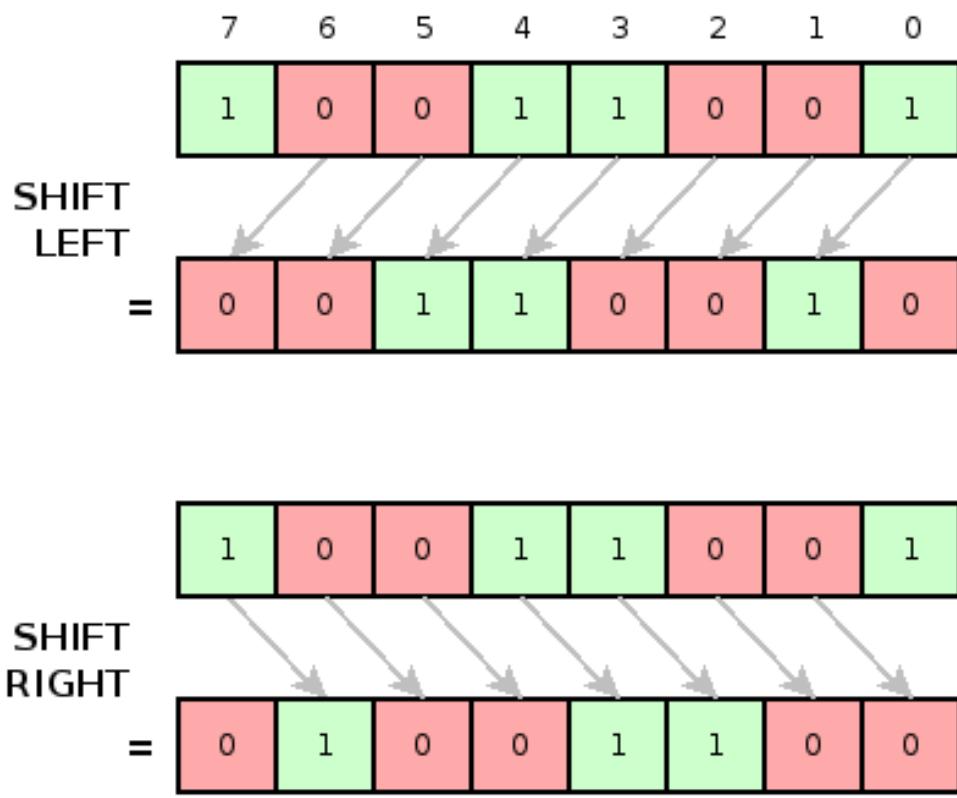
## Bitwise Operation

# Bitwise Operators

int a = 10, b = 2 for all examples below

Operator	Meaning	Example	Result
<code>~</code>	Bitwise unary NOT	<code>~a</code>	-11
<code>&amp;</code>	Bitwise AND	<code>a&amp;b</code>	2
<code> </code>	Bitwise OR	<code>a b</code>	10
<code>^</code>	Bitwise Ex-OR	<code>a^b</code>	8
<code>&gt;&gt;</code>	Shift right	<code>a&gt;&gt;1</code>	5
<code>&gt;&gt;&gt;</code>	Shift right zero fill	<code>a&gt;&gt;&gt;1</code>	5
<code>&lt;&lt;</code>	Shift left	<code>a&lt;&lt;1</code>	20
<code>&amp;=</code>	Bitwise AND assignment	<code>a &amp;= b</code>	2
<code> =</code>	Bitwise OR assignment	<code>a  = b</code>	10
<code>^=</code>	Bitwise Ex-OR assignment	<code>a ^= b</code>	8
<code>&gt;&gt;=</code>	Shift right assignment	<code>a &gt;&gt;= 1</code>	5
<code>&gt;&gt;&gt;=</code>	Shift right zero fill assignment	<code>a &gt;&gt;&gt;= 1</code>	5
<code>&lt;&lt;=</code>	Shift left assignment	<code>a &lt;&lt;= 1</code>	20

Startertutorials.com



## Machine Language

## Machine Language

- Instructions, like registers and words of data, are also 32 bits long
  - Example: add \$t0, \$s1, \$s2
  - registers have numbers, \$t0=8, \$s1=17, \$s2=18
- Instruction Format:

000000	10001	10010	01000	00000	100000
op	rs	rt	rd	shamt	funct

### • Can you guess what the field names stand for?

op = Basic operation of the instruction: opcode

rs = The first register source operand

rt = The second register source operand

rd = The register destination operand

shamt = shift amount

funct = function code

25

## Assembly Language

### Assembly Language



- Tied to the specifics of the underlying machine
- Commands and names to make the code readable and writeable by humans
- Hand-coded assembly code may be more efficient
- E.g., IA32 from Intel

```

    movl $0, %ecx
.loop:  cmpl $1, %edx
        jle .endloop
        addl $1, %ecx
        movl %edx, %eax
        andl $1, %eax
        je .else
        movl %edx, %eax
        addl %eax, %edx
        addl %eax, %edx
        addl $1, %edx
        jmp .endif
.else:  sarl $1, %edx
.endif: jmp .loop
.endloop:

```



## Reading IA32 Assembly Language

- Assembler directives: starting with a period (“.”)
  - E.g., “.section .text” to start the text section of memory
  - E.g., “.loop” for the address of an instruction
- Referring to a register: percent size (“%”)
  - E.g., “%ecx” or “%eip”
- Referring to a constant: dollar sign (“\$”)
  - E.g., “\$1” for the number 1
- Storing result: typically in the second argument
  - E.g. “addl \$1, %ecx” increments register ECX
  - E.g., “movl %edx, %eax” moves EDX to EAX
- Comment: pound sign (“#”)
  - E.g., “# Purpose: Convert lower to upper case”

28

## High level language

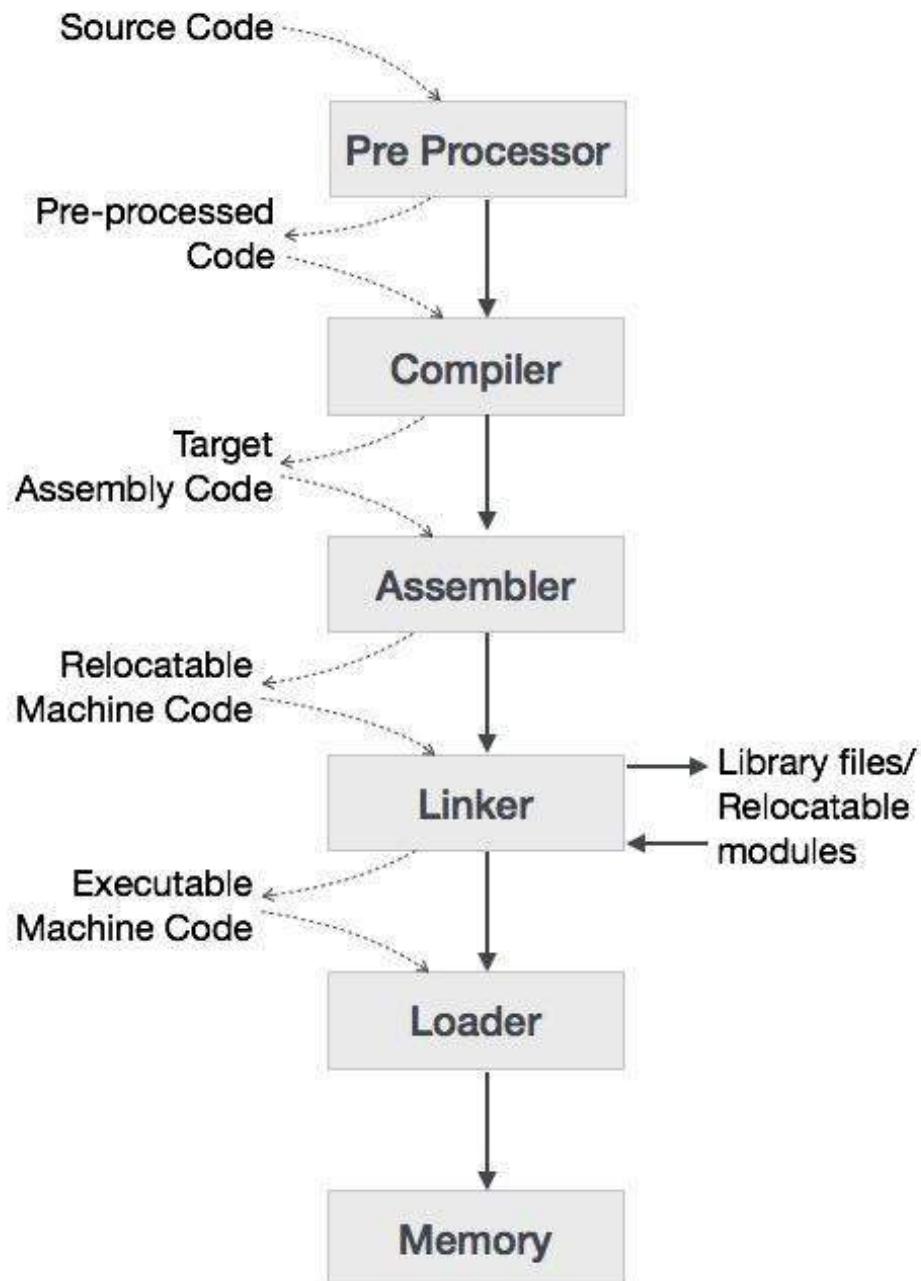


- As stated earlier, a program written in any programming language is a set of logically related instructions. These instructions have two parts, as shown in the following figure:
- The two parts of a programming language instruction are:
  - Operation code (opcode): This part instructs a computer about the operation to be performed.
  - Operand: This part instructs the computer about the location of the data on which the operation specified by the opcode is to be performed.

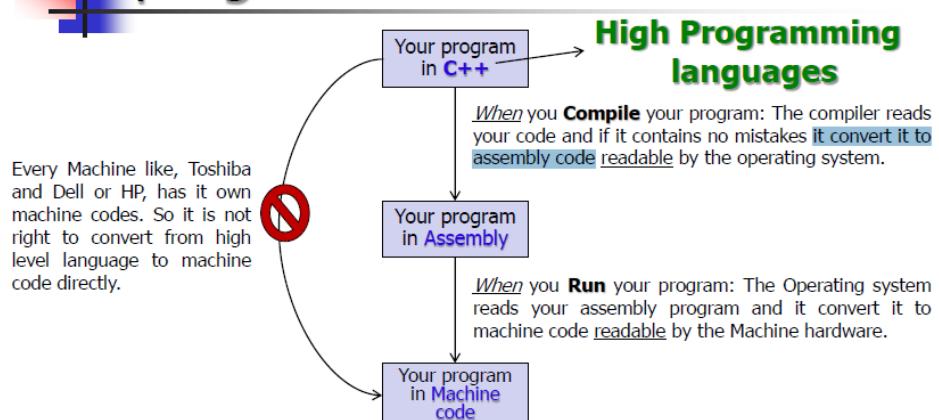


- For example, in the instruction Add A and B, Add is the opcode and A and B are operands

## Compiler and Assembler



## Compiling and running your program



## LINKER VS LOADER VS COMPILER

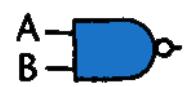
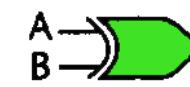
LINKER	LOADER	COMPILER
A computer utility program that takes one or more object files generated by a compiler and combines them into a single executable file	A part of an operating system that is responsible for loading programs to memory	A software that transforms computer code written in one programming language into another programming language
Combines multiple object code and links them with libraries	Prepares the executable file for running	Transforms the source code into object code

Visit [www.pediaa.com](http://www.pediaa.com)

## Logic Design

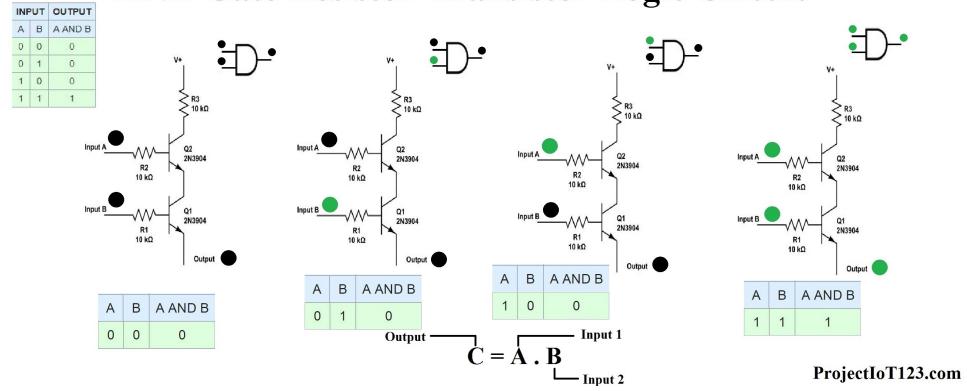
- Logic Gates

 <b>Buffer</b> $F = A$	 <b>AND</b> $F = AB$	 <b>OR</b> $F = A+B$	 <b>XOR</b> $F = A \oplus B$																																																			
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>F</th></tr> <tr> <td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td></tr> </table>	A	F	0	0	1	1	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0
A	F																																																					
0	0																																																					
1	1																																																					
A	B	F																																																				
0	0	0																																																				
0	1	0																																																				
1	0	0																																																				
1	1	1																																																				
A	B	F																																																				
0	0	0																																																				
0	1	1																																																				
1	0	1																																																				
1	1	0																																																				
A	B	F																																																				
0	0	0																																																				
0	1	1																																																				
1	0	1																																																				
1	1	0																																																				

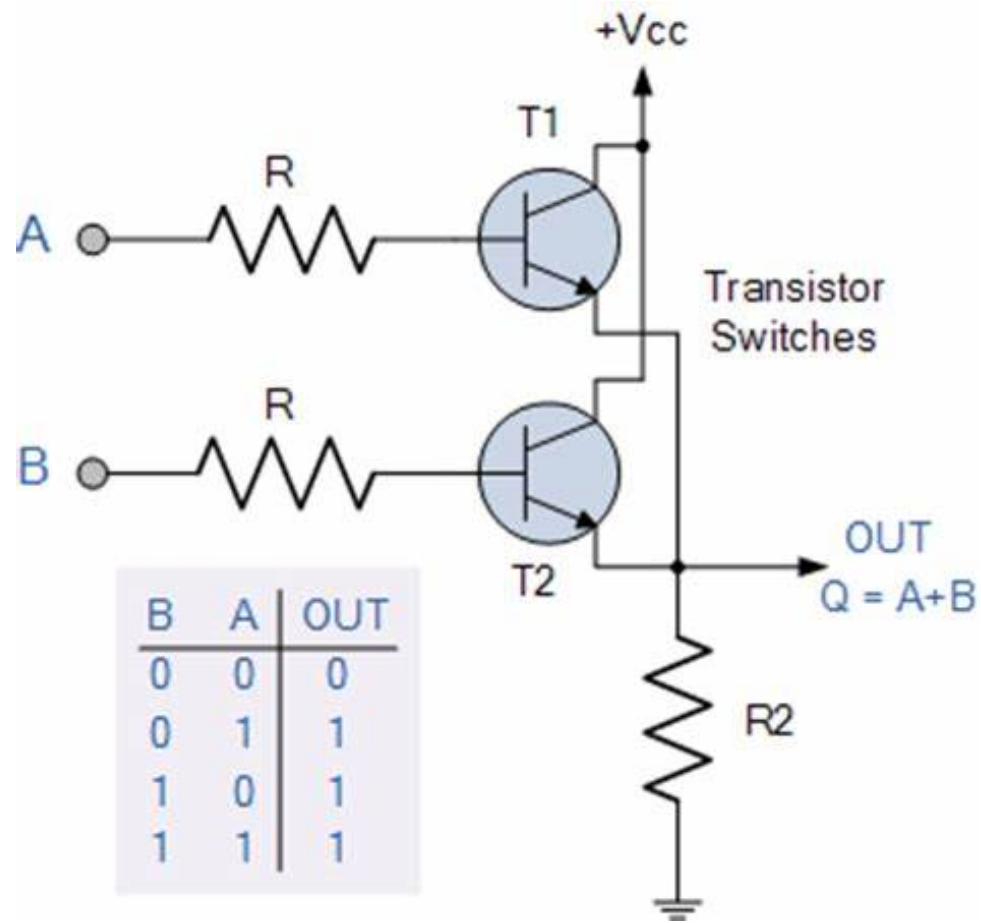
 <b>Inverter</b> $F = \bar{A}$	 <b>NAND</b> $F = \bar{AB}$	 <b>NOR</b> $F = \bar{A+B}$	 <b>XNOR</b> $F = \bar{A \oplus B}$																																																			
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>F</th></tr> <tr> <td>0</td><td>1</td></tr> <tr> <td>1</td><td>0</td></tr> </table>	A	F	0	1	1	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	1
A	F																																																					
0	1																																																					
1	0																																																					
A	B	F																																																				
0	0	1																																																				
0	1	1																																																				
1	0	1																																																				
1	1	0																																																				
A	B	F																																																				
0	0	1																																																				
0	1	0																																																				
1	0	0																																																				
1	1	0																																																				
A	B	F																																																				
0	0	1																																																				
0	1	0																																																				
1	0	0																																																				
1	1	1																																																				

- AND Gate

### AND Gate Resistor-Transistor Logic Circuit



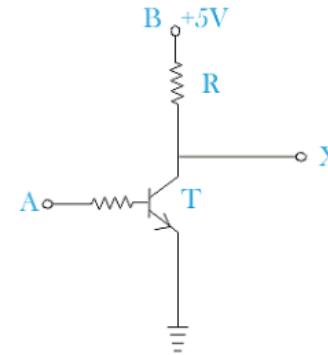
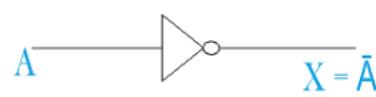
- OR Gate



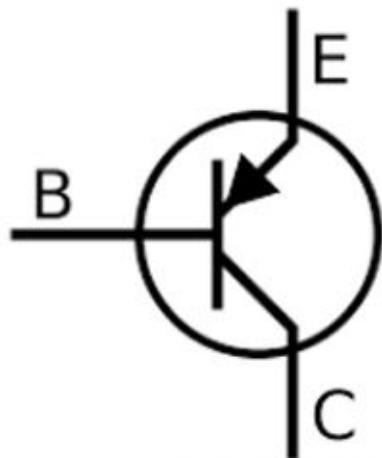
- NOT Gate

# What is a NOT Gate?

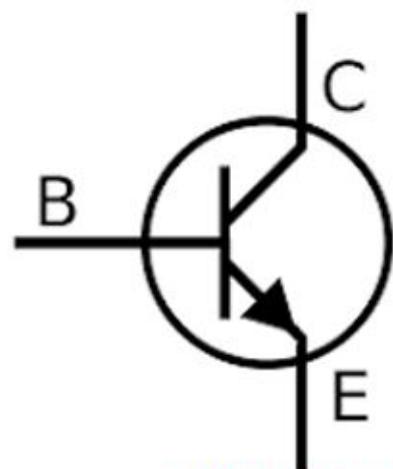
A	X = $\bar{A}$
0	1
1	0



**Electrical 4 U**



**PNP BJT**



**NPN BJT**

- XOR Gate

## XOR GATE Truth Table



**BOOLEAN EXPRESSION**

$$\begin{aligned} & A \cdot \overline{B} + \overline{A} \cdot B \\ & (A + B) \cdot (\overline{A} + \overline{B}) \end{aligned}$$

$C = A \oplus B$

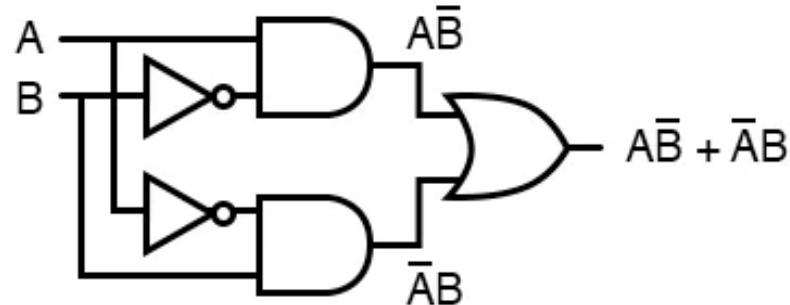
Output      Input1      Input2

INPUT		OUTPUT
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

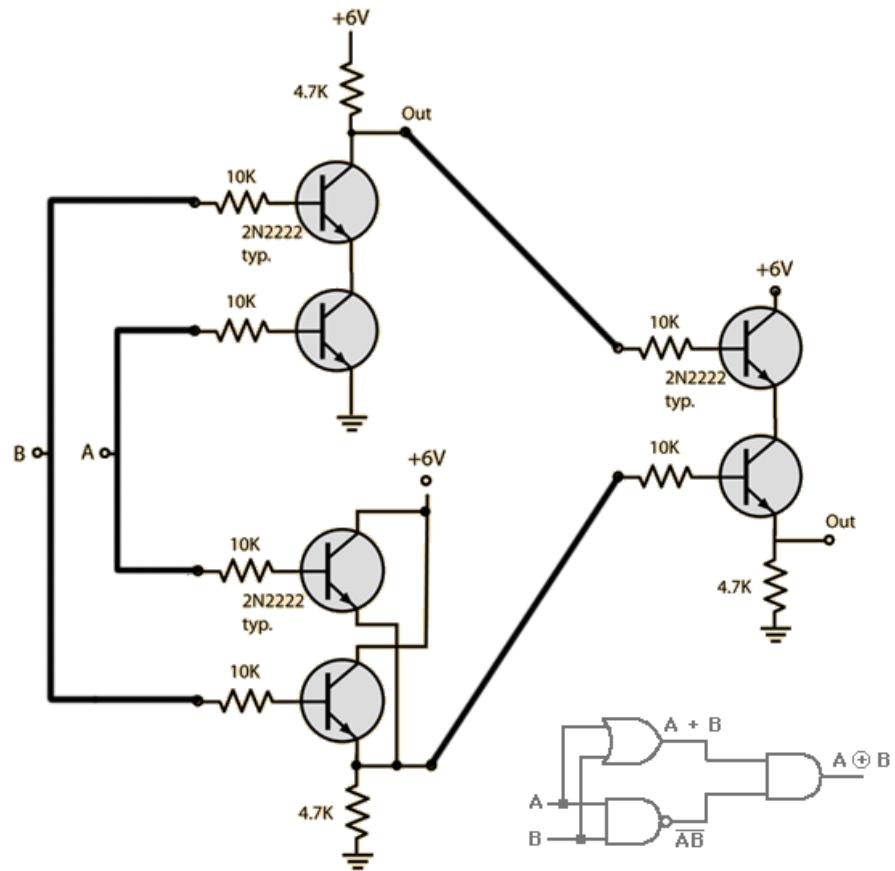
ProjectIoT123.com



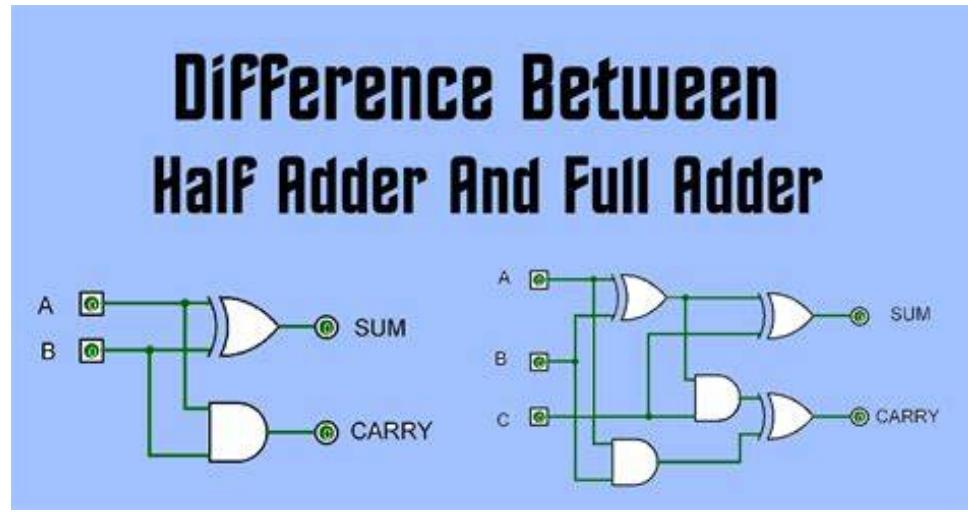
. . . is equivalent to . . .



$$A \oplus B = A\overline{B} + \overline{A}B$$



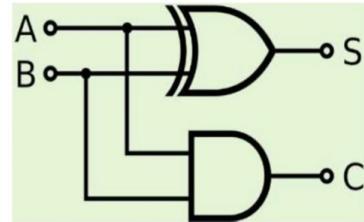
- **Half-Adder vs Full-Adder**



- **Half-Adder**

Half Adder and Full Adder

#### **Logic Diagram:**



$$S = A \oplus B \quad = \quad S = \bar{A}B + A\bar{B}$$

$$S = \bar{A}B + A\bar{B}$$

**C=A.**

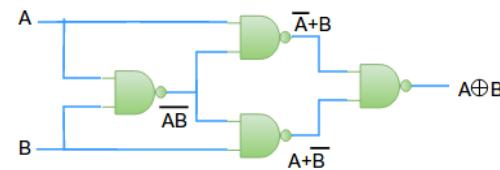
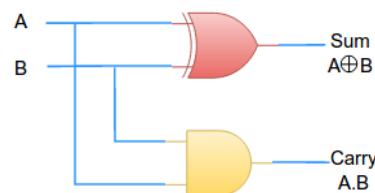
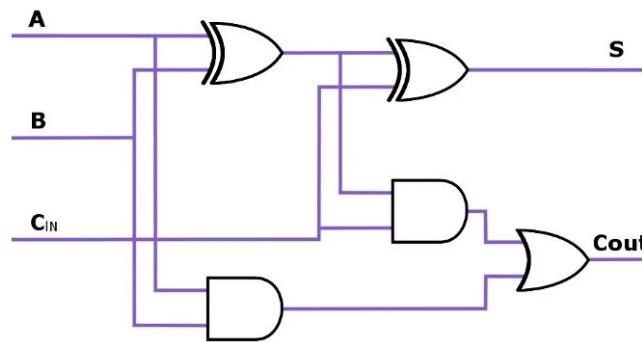
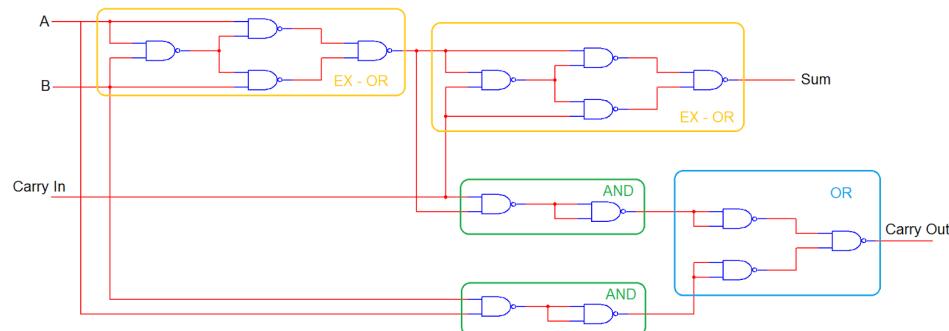
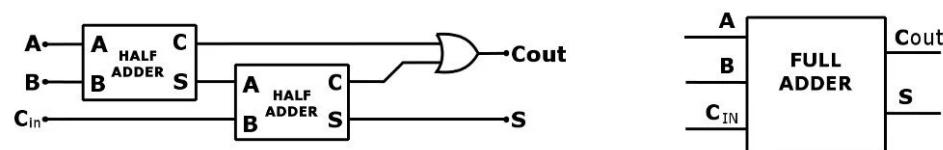


Figure-1 : a)Half Adder b)XOR implementation using NAND gates

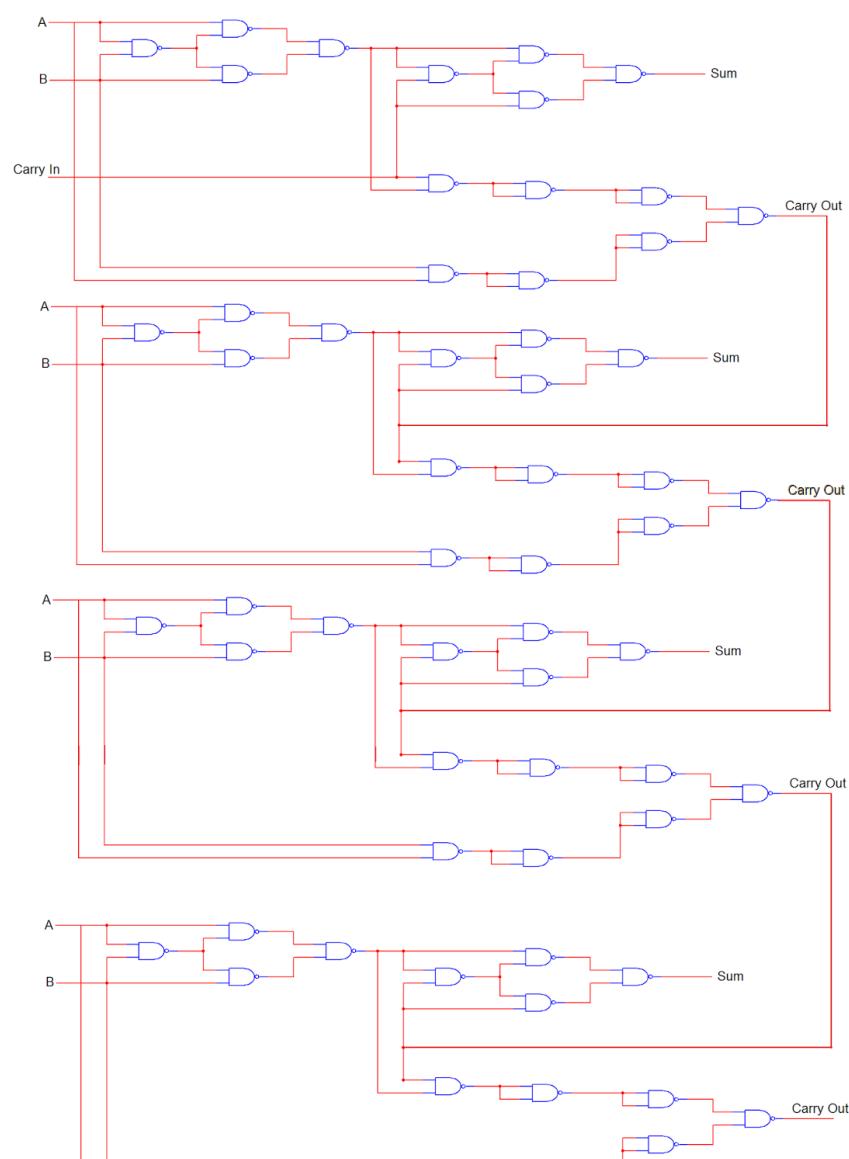
- Full Adder



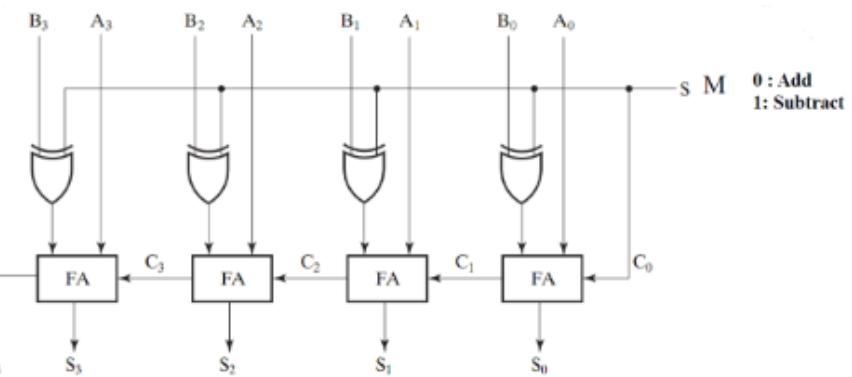
Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



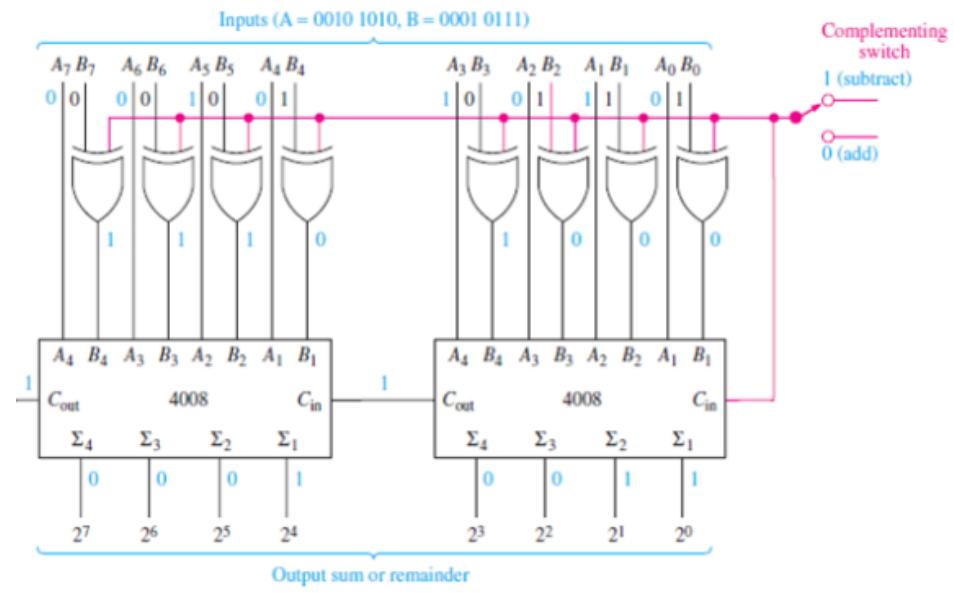
- **4 Bit Full Adder**



- **Subtract**

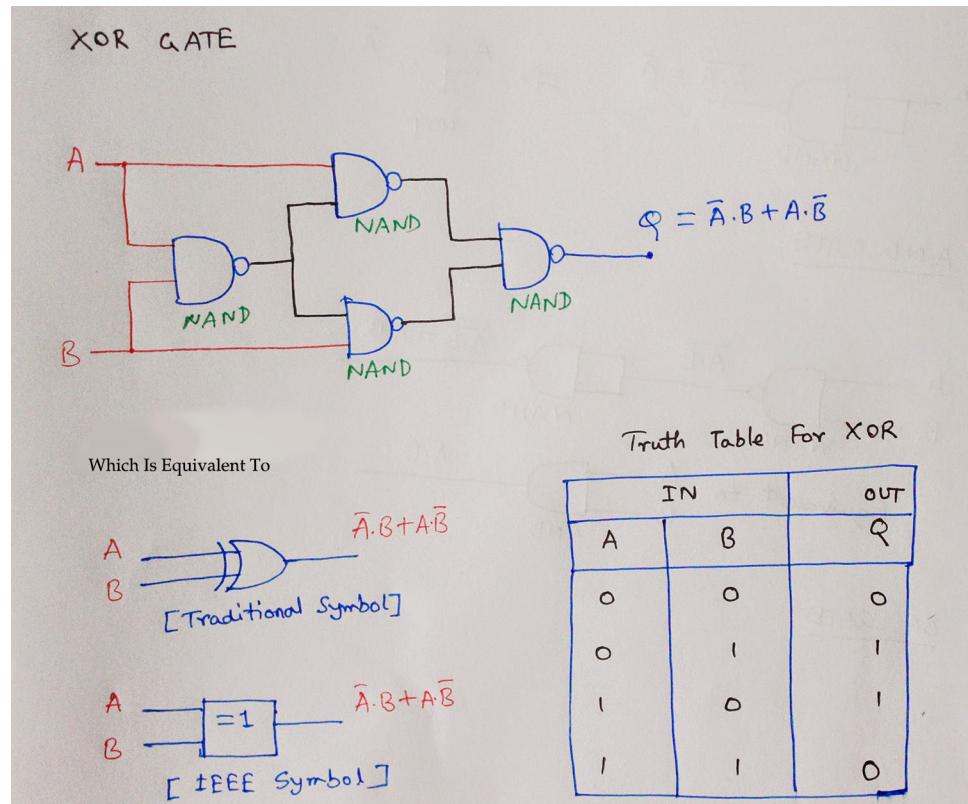


**Fig. 1** Modular design of 4-bit adder/subtractor



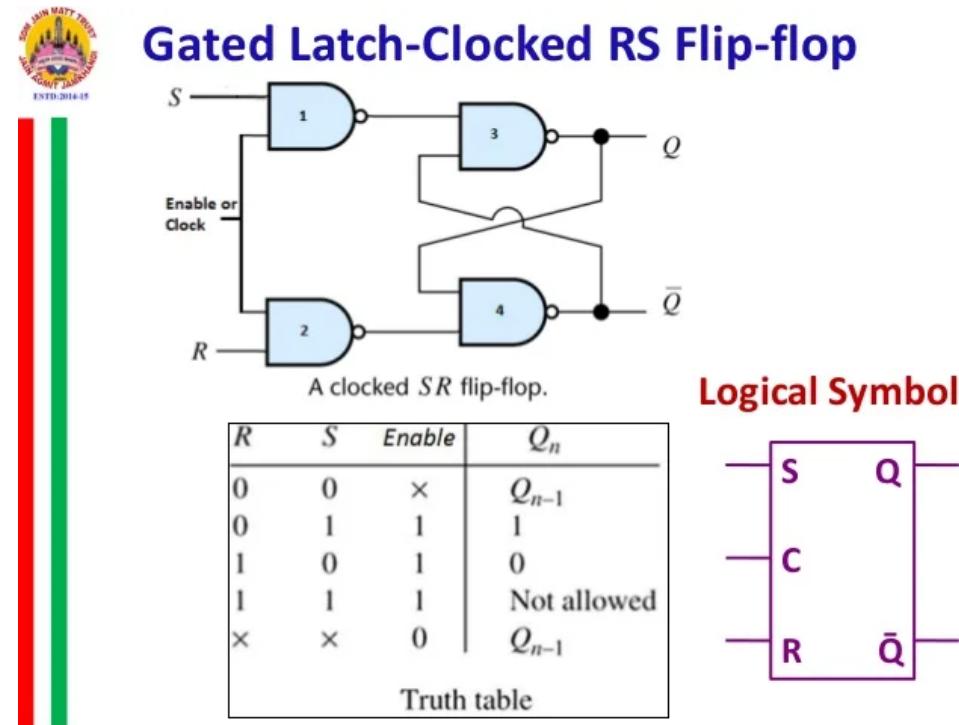
- NAND Gate

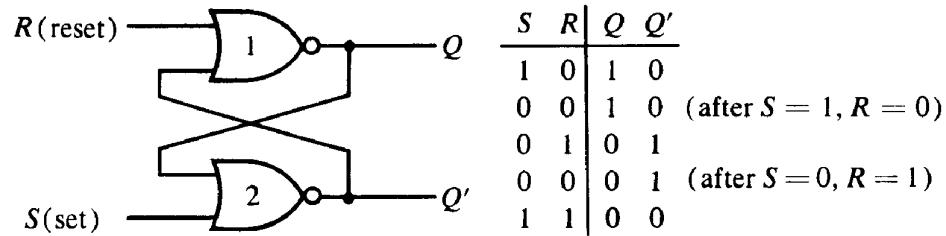
- XOR and NAND



## Circuits as Memory

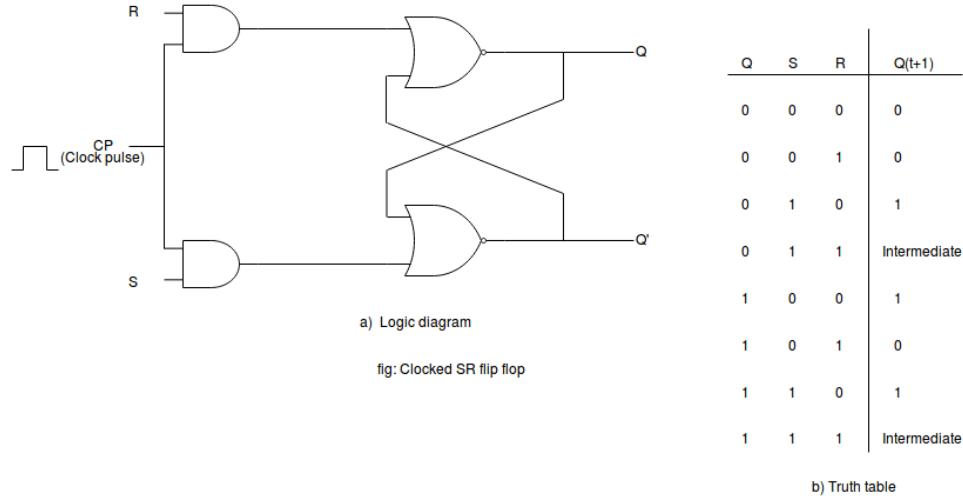
- SR Flip Flop





(a) Logic diagram

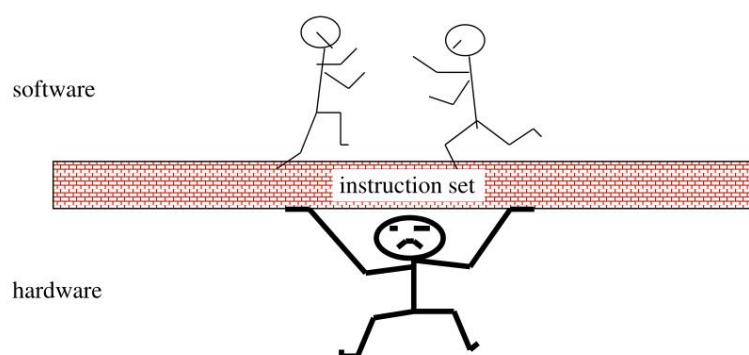
(b) Truth table



## Computer Organization and Architecture

### Instruction Set Architecture - Interface of S/W and H/w

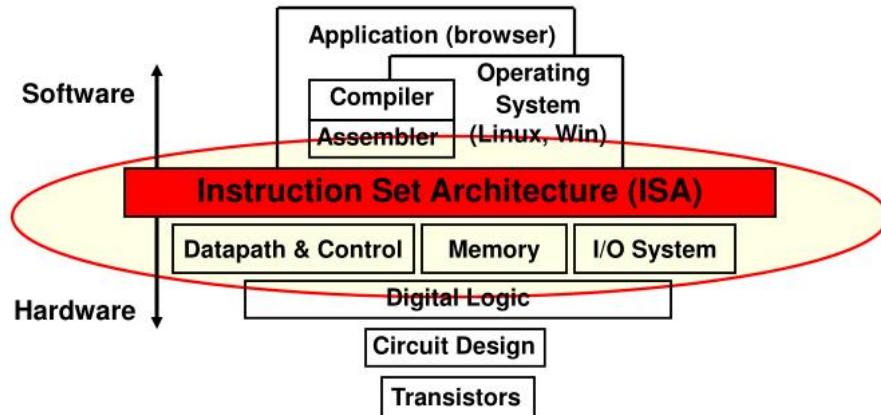
#### Instruction Set Architecture: Critical Interface



- Properties of a good abstraction
  - ◆ Lasts through many generations (portability)
  - ◆ Used in many different ways (generality)
  - ◆ Provides **convenient** functionality to higher levels
  - ◆ Permits an **efficient** implementation at lower levels

Computer Architecture- 8

# The Instruction Set Architecture



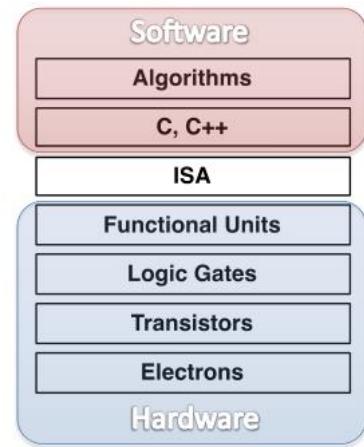
ISA

## Instruction Set Architecture

- The computer ISA defines all the *programmer-visible* components and operations of the computer
  - Memory organization
    - address space -- how many locations can be addressed?
    - addressability -- how many bits per location?
  - Register set
    - how many? what size? how are they used?
  - Instruction set
    - opcodes
    - data types
    - addressing modes
- ISA provides all information needed for someone that wants to write a program in **machine language** (or translate from a high-level language to machine language).

# Instruction Set Architecture (ISA)

- ISA is the interface provided by the hardware to the software
  - Defines the available:
    - instructions
    - registers
    - addressing modes
    - memory architecture
    - interrupt and exception handling
    - external I/O
  - Syntax defined by assembly language
    - Symbolic representation of the machine instructions
  - Examples: x86, ARM, HCS12



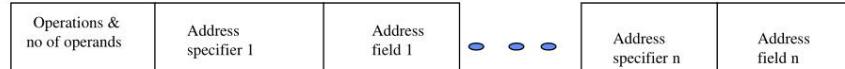
Mike Holenderski, m.holenderski@tue.nl

12



**TU/e** Technische Universiteit  
Eindhoven University of Technology

## Three Examples of Instruction Set Encoding



Variable: VAX (1-53 bytes)

Operation	Address field 1	Address field 2	Address field3
-----------	-----------------	-----------------	----------------

Fixed: DLX, MIPS, PowerPC, SPARC

Operation	Address Specifier	Address field
-----------	-------------------	---------------

Operation	Address Specifier 1	Address Specifier 2	Address field
-----------	---------------------	---------------------	---------------

Operation	Address Specifier	Address field 1	Address field 2
-----------	-------------------	-----------------	-----------------

Hybrid : IBM 360/370, Intel 80x86

**EECC551 - Shaaban**

#20 Lec # 2 Fall 2000 9-12-2000

## CICS vs RICS

## CISC vs. RISC

### Complex Instruction Set Computer

- Many instructions
  - e.g., 75-100
- Many instructions are macro-like
  - Simplifies programming
- Most microcontrollers are based on CISC concept
  - e.g., PDP-11, VAX, Motorola 68k
  - PIC is an exception

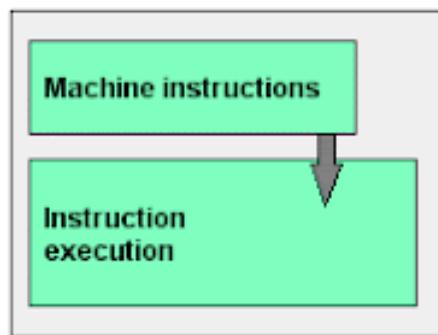
### Reduced Instruction Set Computers

- Few instructions
  - e.g., 30-40
- Smaller chip, smaller pin count, & very low-power consumption
  - Simple but fast instructions
- Harvard architecture, instruction pipelining
- Industry trend for microprocessor design
  - e.g., Intel Pentium, PIC

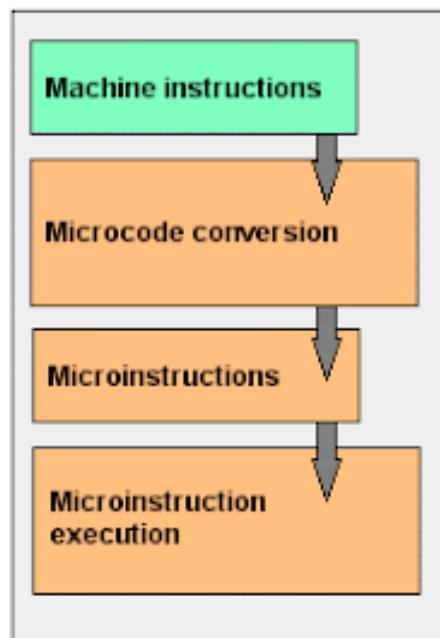
24

From Computer Desktop Encyclopedia  
© 1998 The Computer Language Co., Inc.

### RISC

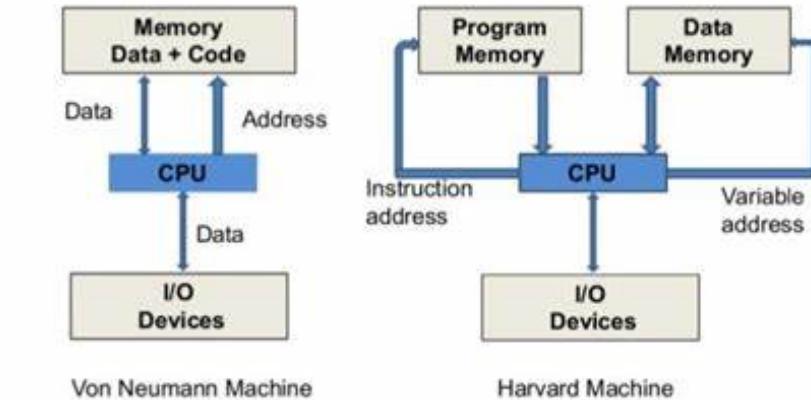


### CISC



## Memory Architecture

## Von Neumann vs. Harvard Architecture



6

Memory & Timing model

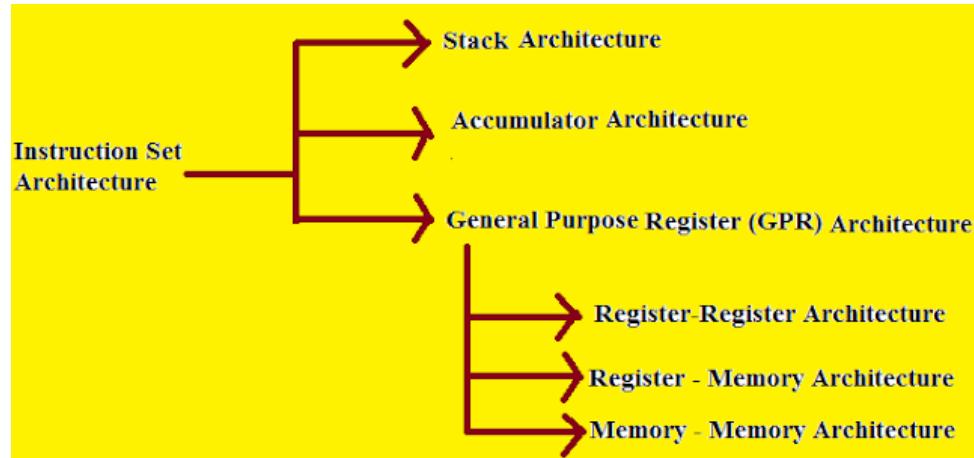
ISA: STORAGE RESOURCES

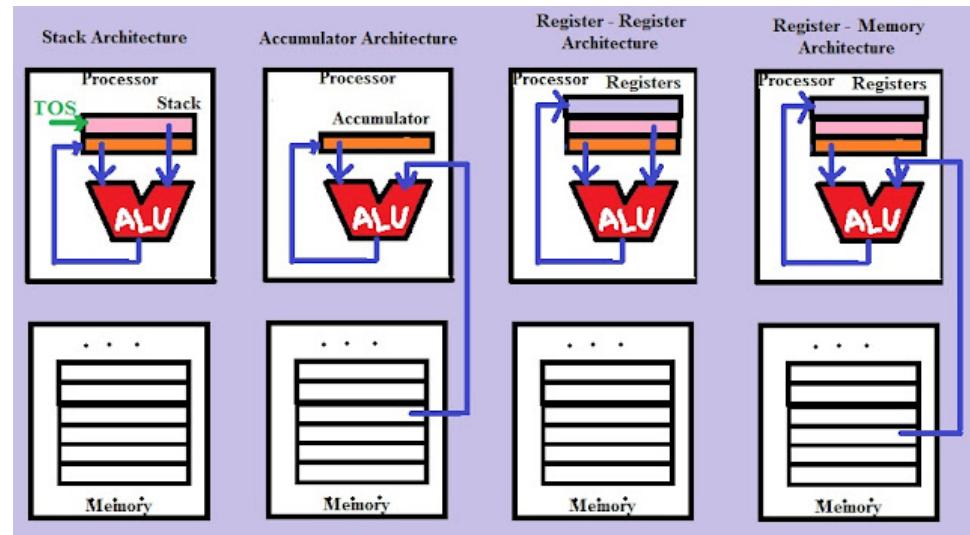
- "Harvard architecture": Separate instruction and data memories
- Permit use of **single clock cycle per instruction** implementation
- Due to use of "cache" in modern computer architectures, it is a fairly **realistic model**

The diagram shows the components of a Harvard architecture system:

- Instruction memory:  $2^{15} \times 16$
- Program counter (PC)
- Data memory:  $2^{15} \times 16$
- Register file:  $8 \times 16$

## Classification of ISA (or) Types of ISA

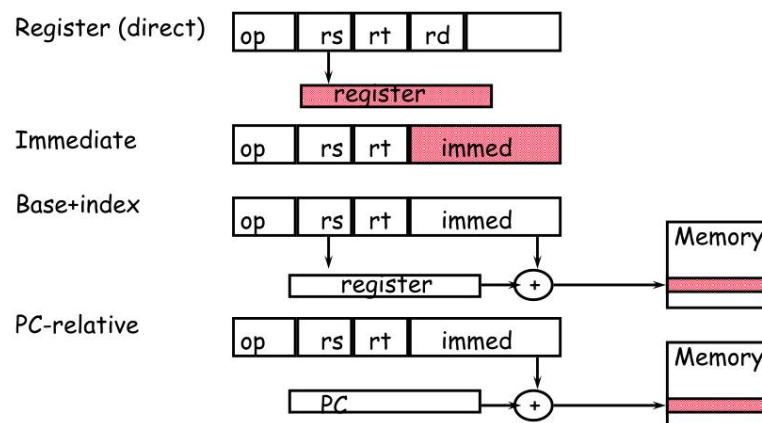




## ISA - MIPS Instruction Format and Addressing Model

### Example: MIPS Instruction Formats and Addressing Modes

- All instructions 32 bits wide

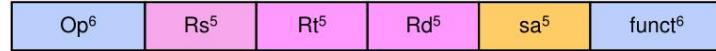


## Instruction Formats

❖ All instructions are 32-bit wide. Three instruction formats:

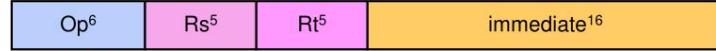
❖ **Register (R-Type)**

- ◊ Register-to-register instructions
- ◊ Op: operation code specifies the format of the instruction



❖ **Immediate (I-Type)**

- ◊ 16-bit immediate constant is part in the instruction



❖ **Jump (J-Type)**

- ◊ Used by jump instructions



	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
<b>R:</b>	op	rs	rt	rd	shamt	funct
<b>I:</b>	op	rs	rt	address / immediate		
<b>J:</b>	op	target address				

op: basic operation of the instruction (opcode)

rs: first source operand register

rt: second source operand register

rd: destination operand register

shamt: shift amount

funct: selects the specific variant of the opcode (function code)

address: offset for load/store instructions (+/-2<sup>15</sup>)

immediate: constants for immediate instructions

## 5.6 Addressing Modes

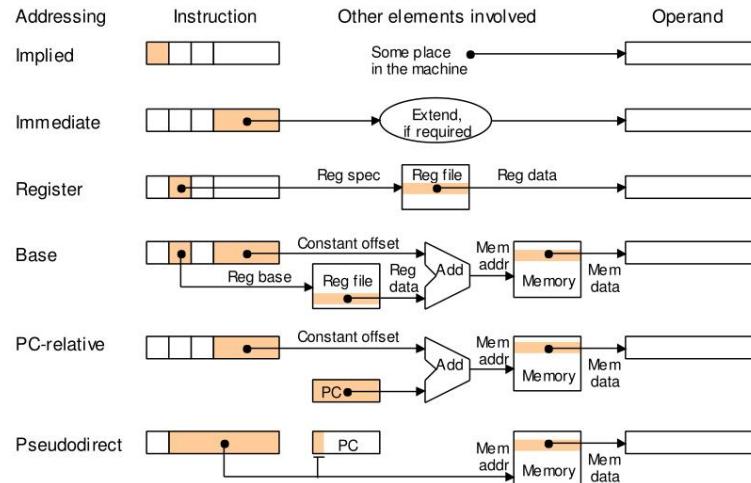
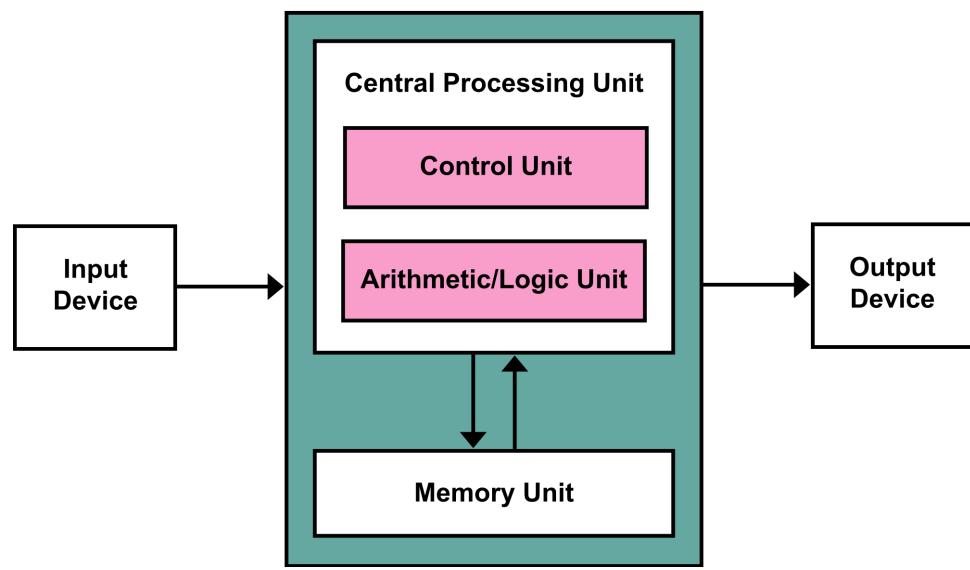


Figure 5.11 Schematic representation of addressing modes in MiniMIPS.

Computer Architecture, Instruction-Set Architecture

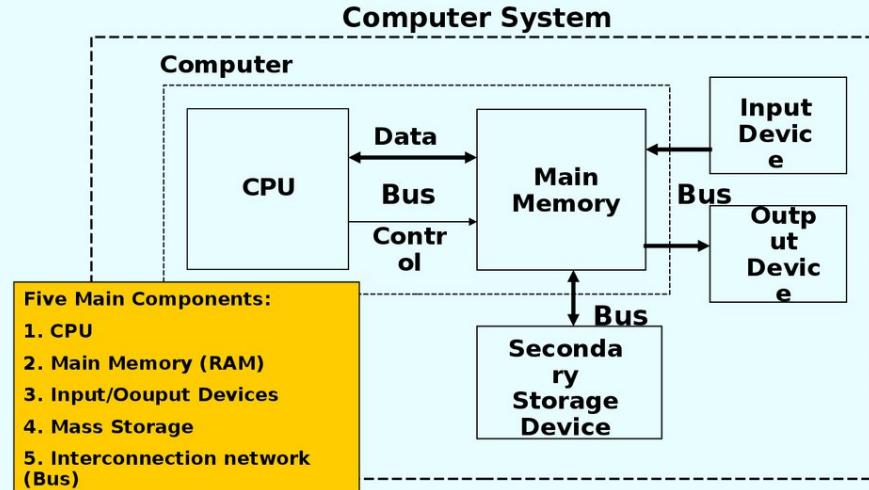
Slide 22

## Processor - Von Neumann Architecture

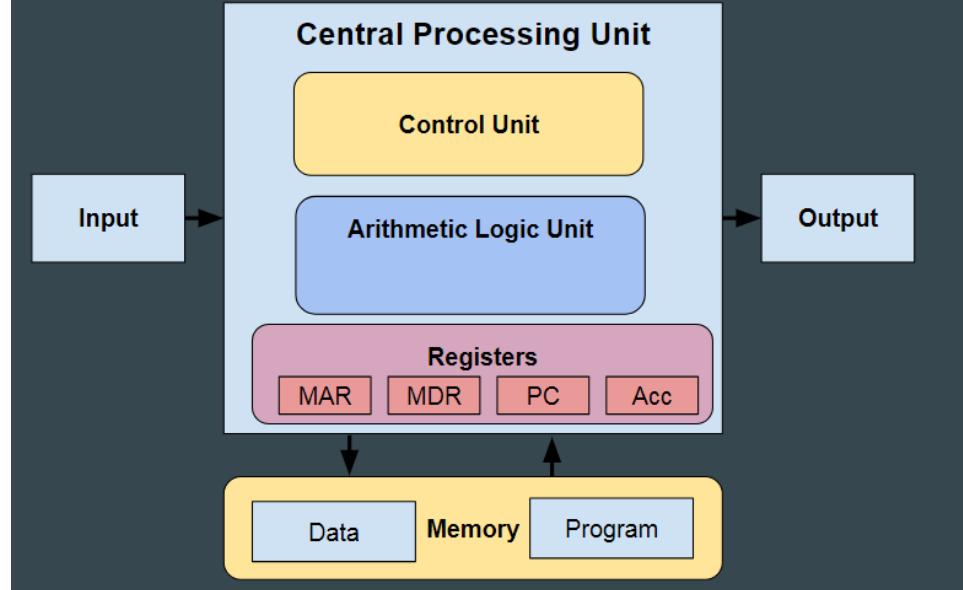


## von Neumann Architecture

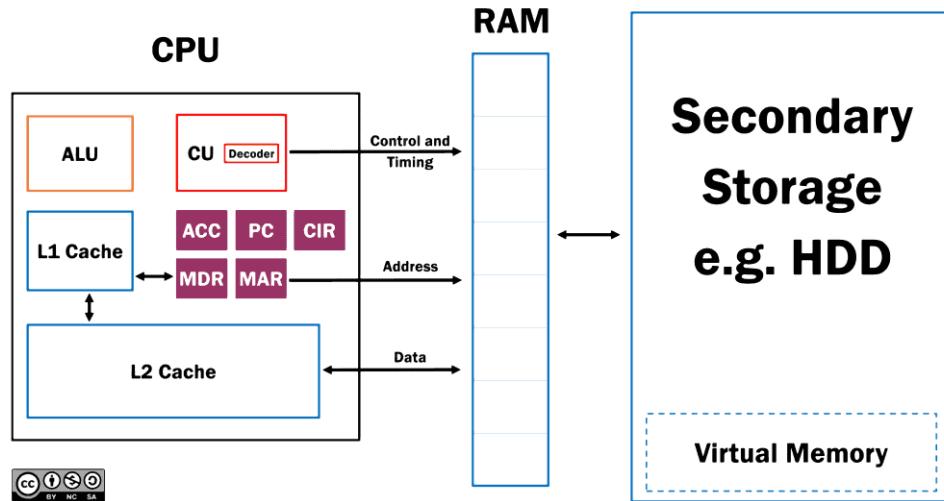
- A more complete view of the computer *system* architecture that integrates interaction (human or otherwise) consists of:



## Von Neumann Architecture Diagram



## Computer Systems - Von Neumann Architecture



## Processor - Fetch-Execute Cycle

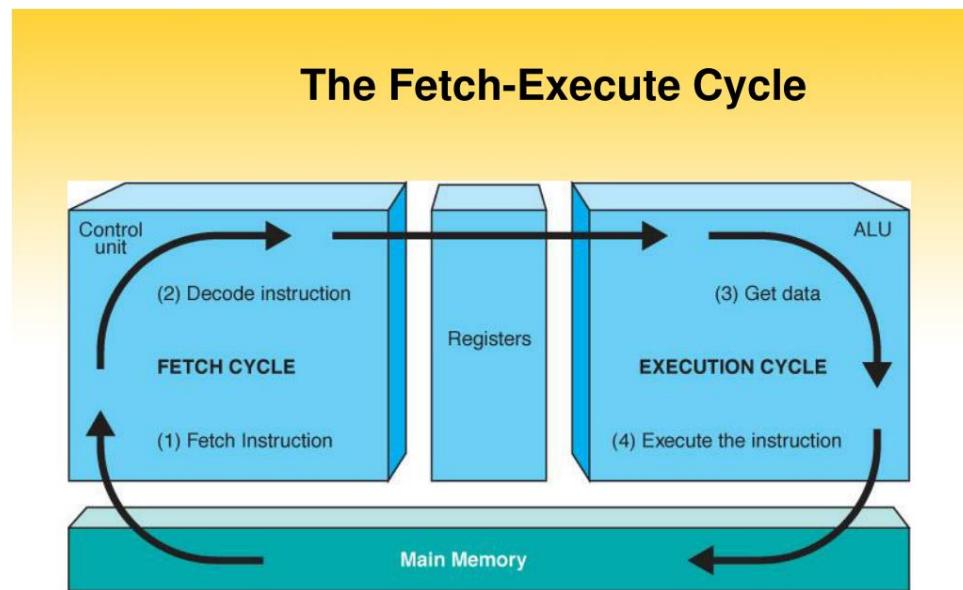
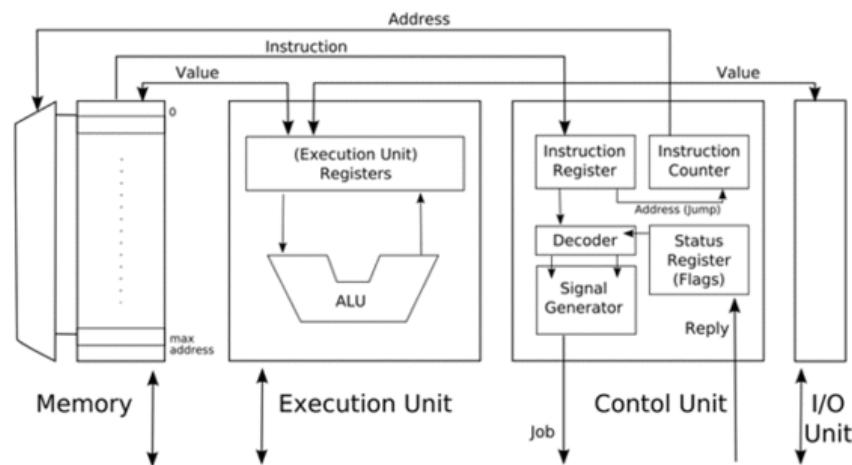
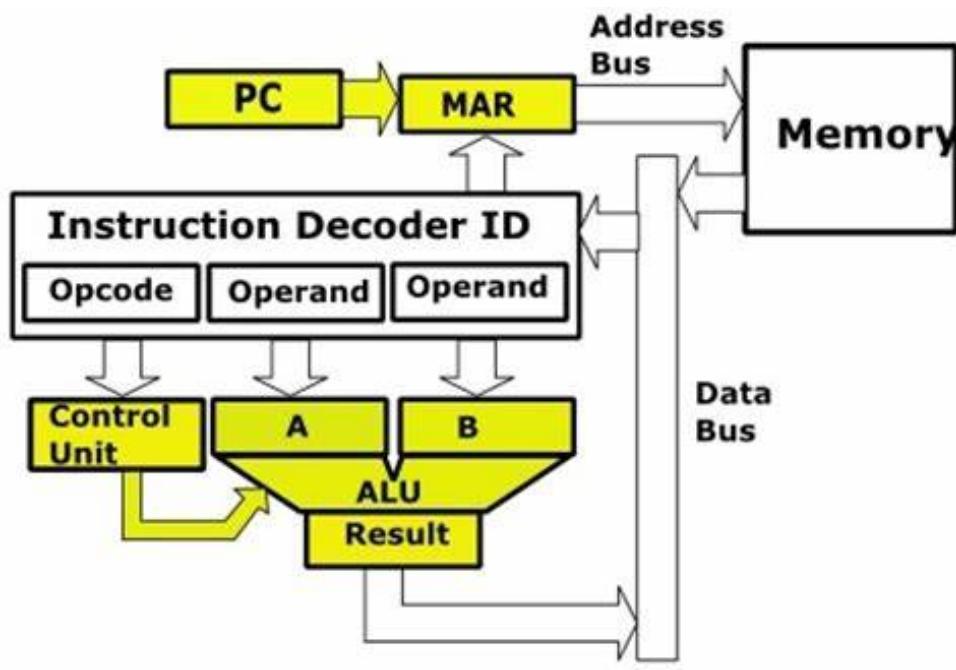
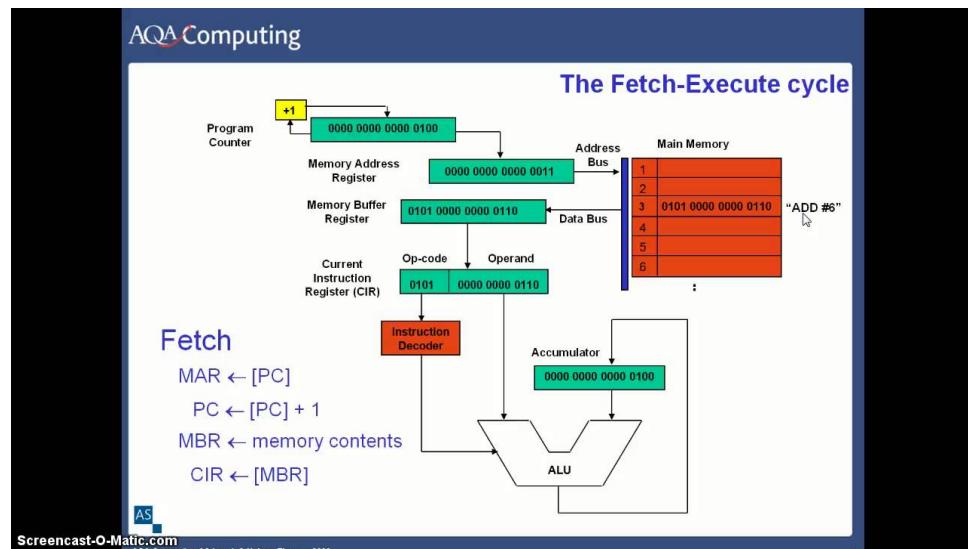
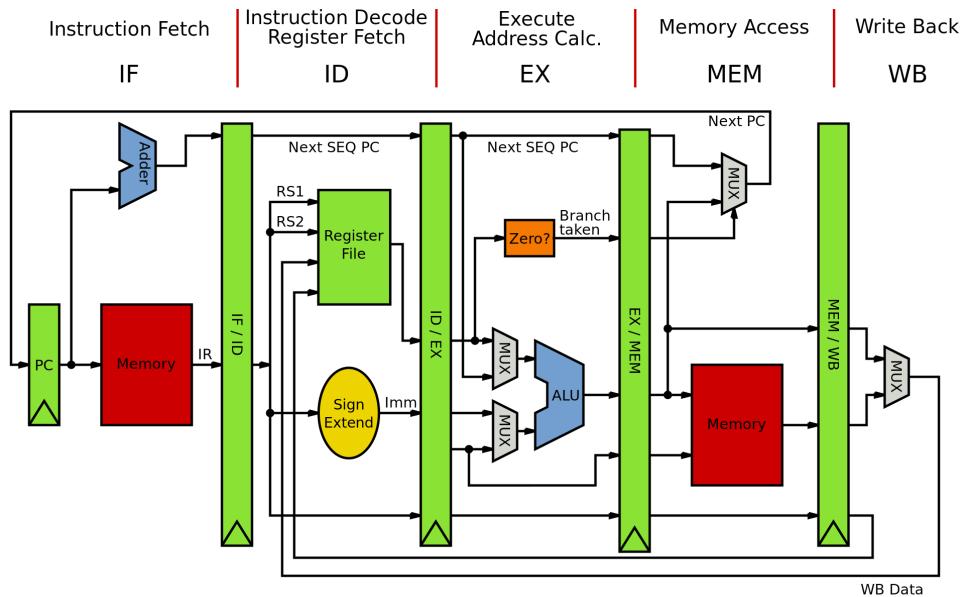


Figure 5.3 The Fetch-Execute Cycle

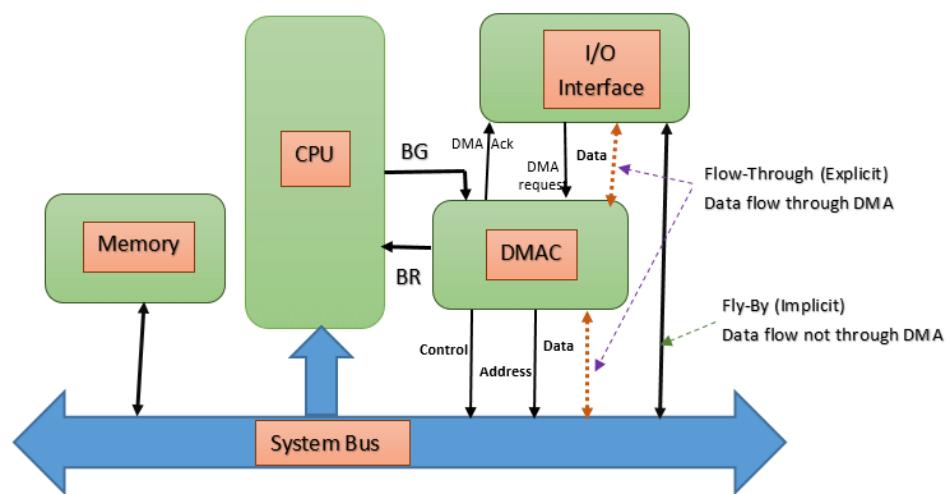
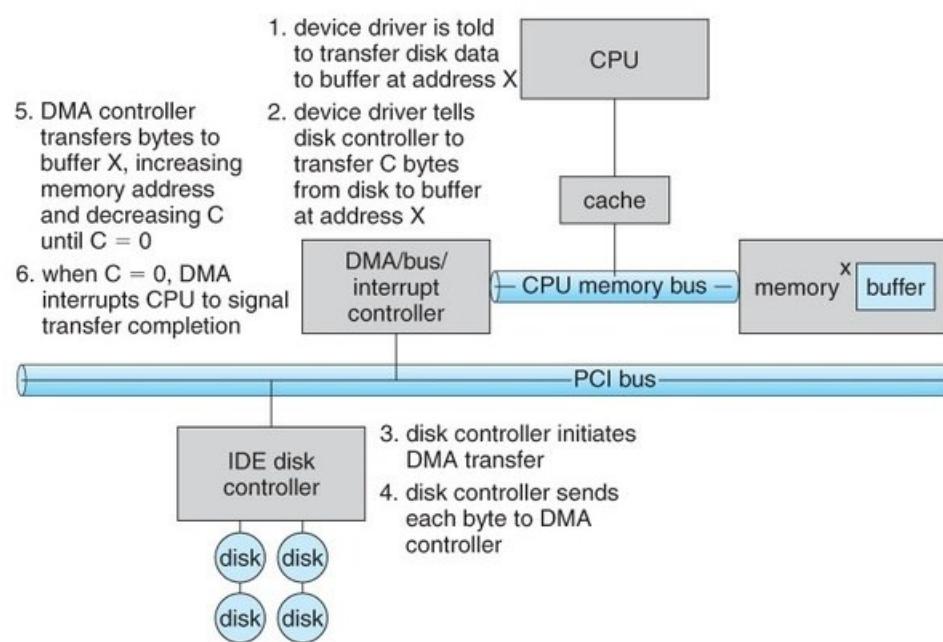
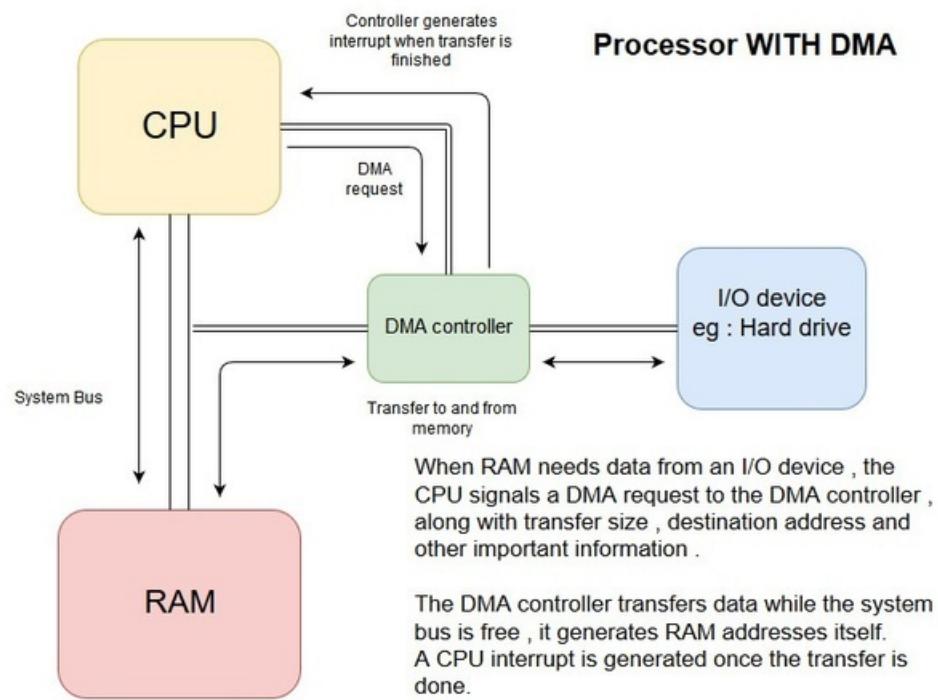
17

© 2011 Jones and Bartlett Publishers, LLC (www.jbpub.com)

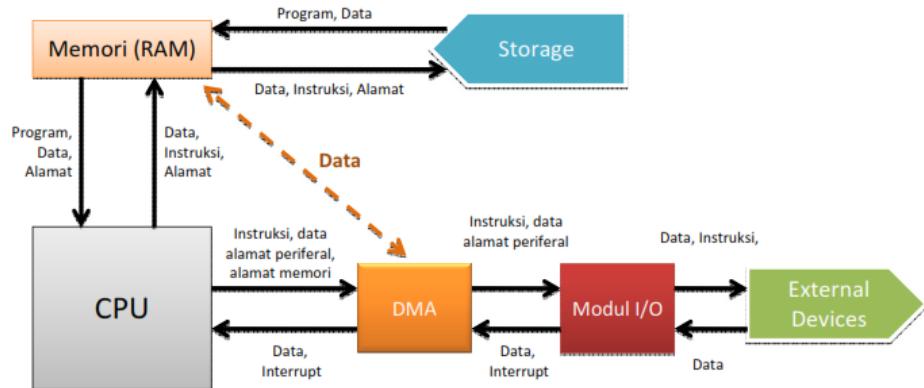




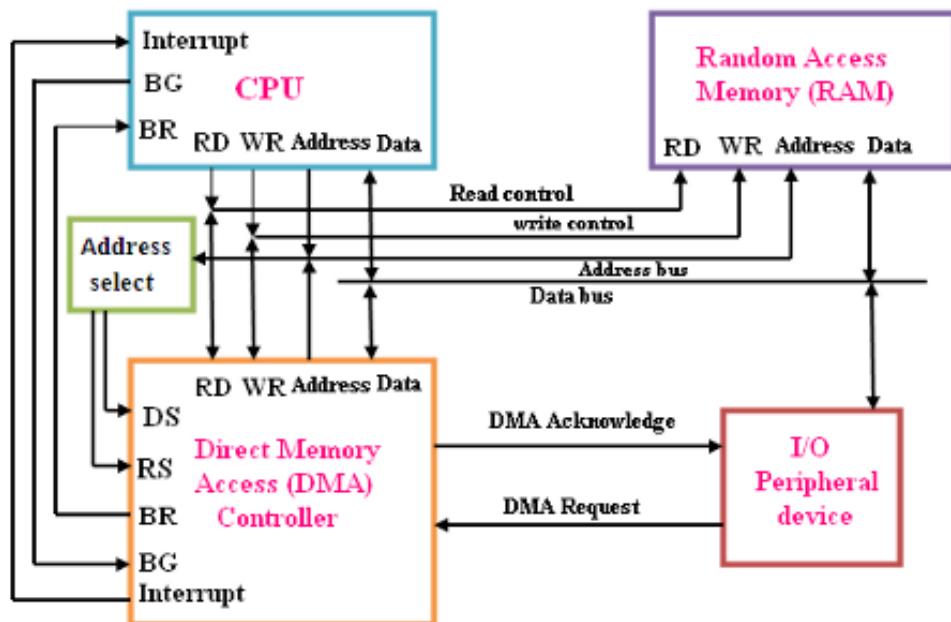
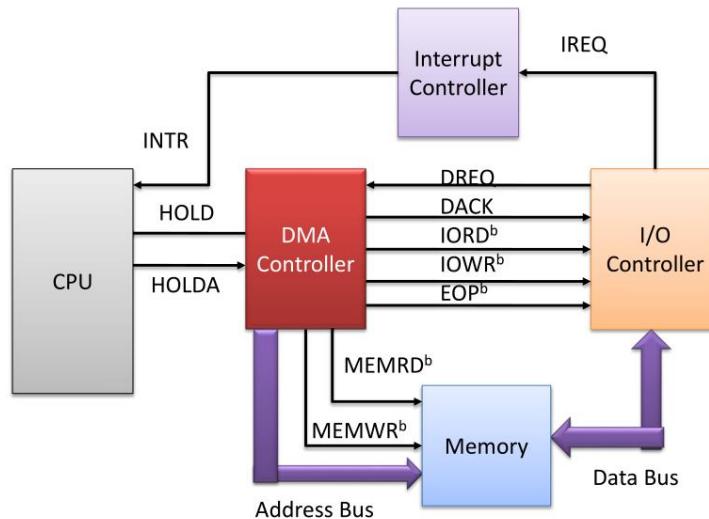
## Data Transfer - DMA



## Direct Memory Access



## DMA Controller



# Operating System

## Types of OS

### What are the different types?

**Mac OS** is a series of graphical user interface-based operating systems developed by Apple Inc. for their Macintosh



**Linux** is a Unix-like computer operating system assembled under the model of free and open source software development and distribution.



**Microsoft Windows** is a series of graphical interface operating systems developed, marketed, and sold by Microsoft.



**iOS** (previously iPhone OS) is a mobile operating system developed and distributed by Apple Inc. Originally unveiled in 2007 for the iPhone, it has been extended to support other Apple devices such as the iPod Touch



**Android** is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially developed by Android, Inc.

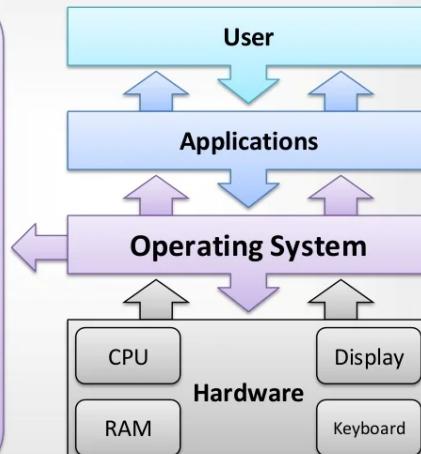


**BSD/OS** had a reputation for reliability in server roles; the renowned Unix programmer and author W. Richard Stevens used it for his own personal web server for this reason.



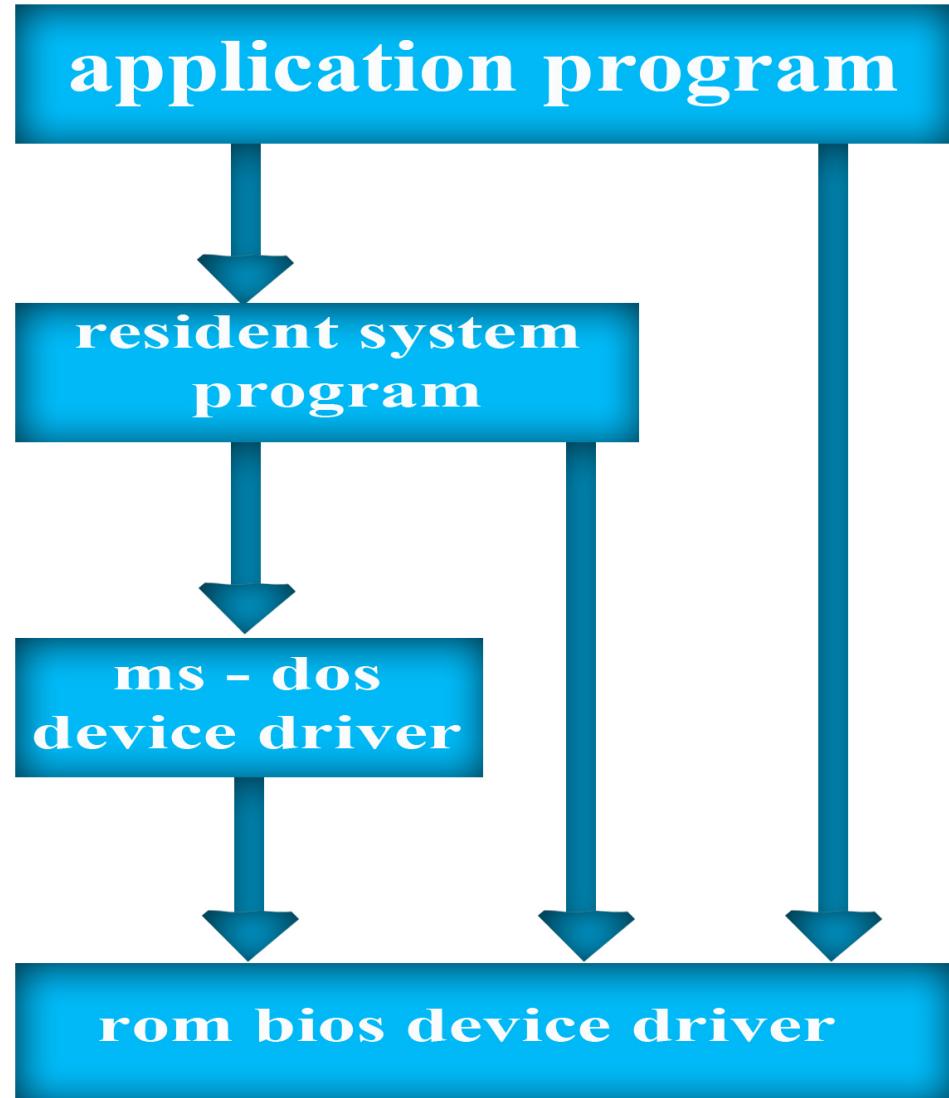
## OS Structure

### Components



Computer Operating Systems

10



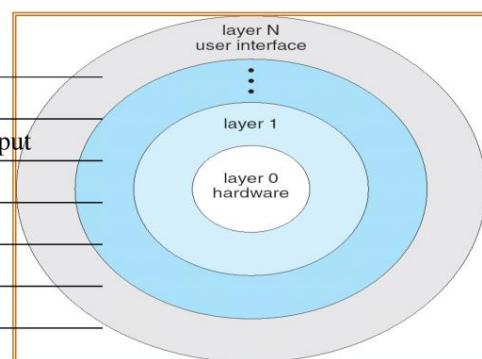
## ms dos system structure

### Layered Structure of the THE OS

- A layered design was first used in THE operating system.

Its six layers are as follows:

- |   |
|---|
| layer 5: user programs                  |
| layer 4: buffering for input and output |
| layer 3: Process management             |
| layer 2: memory management              |
| layer 1: CPU scheduling                 |
| layer 0: hardware                       |



# Operating System Layers

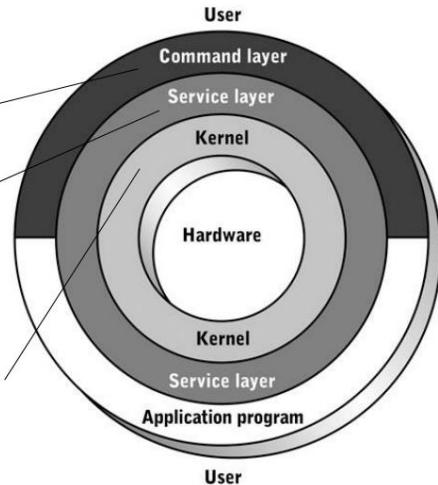
Figure 11-3 ►

Operating system layers (shaded)

User's interface to OS

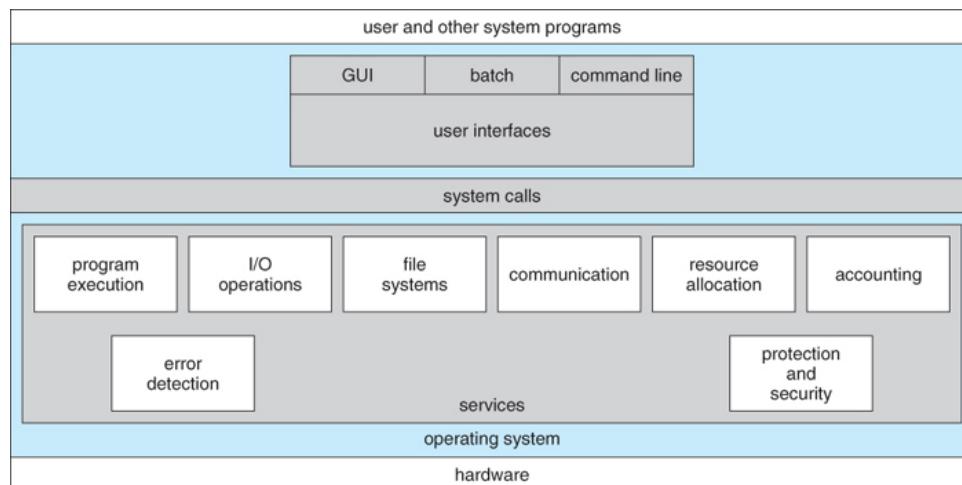
Contains set of functions executed by application programs and command layer

Manages resources; interacts directly with computer hardware

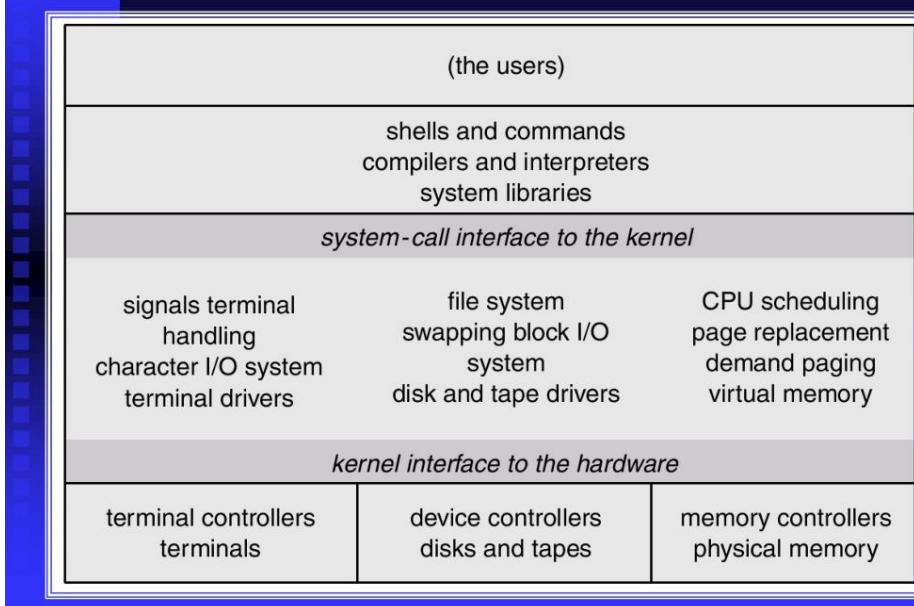


Hardware and Software Architecture

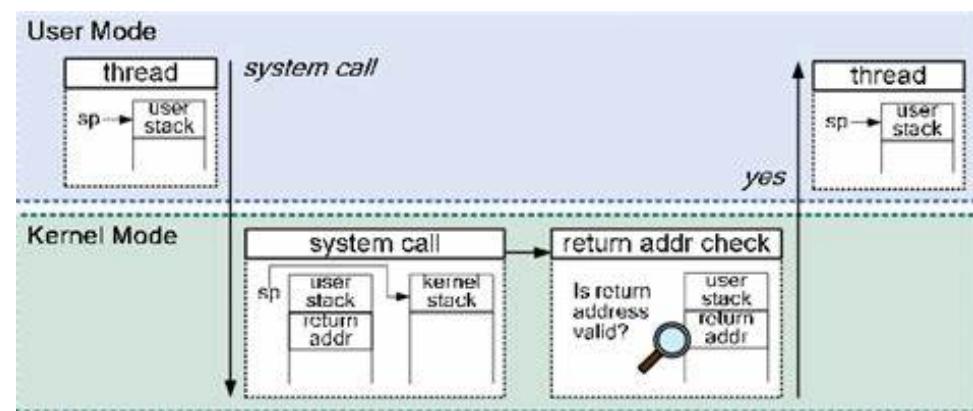
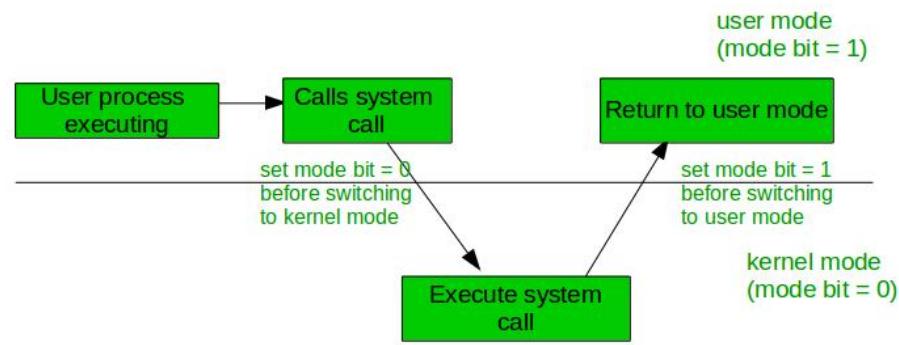
16



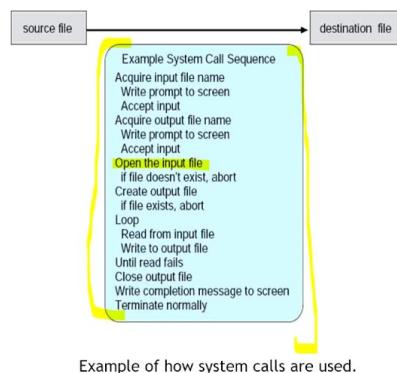
## OS Structure: Monolithic structure



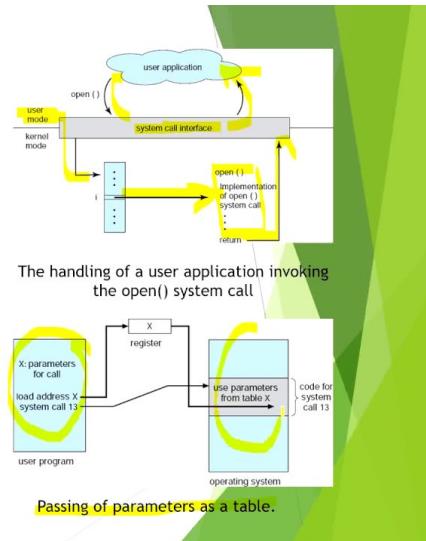
## OS - Dual Mode Operation



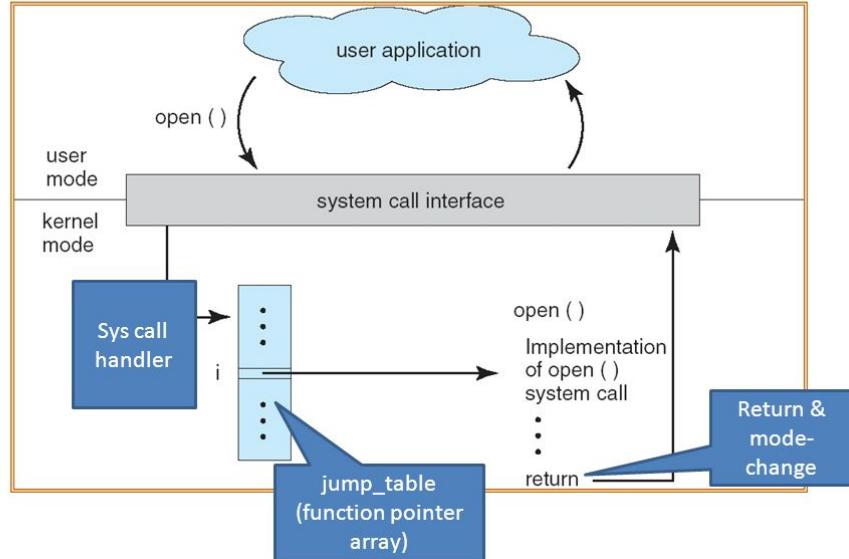
## SYSTEM CALLS



Example of how system calls are used.



# API – System Call – OS Relationship





## Types of System Calls

- s Process control**
  - q Load, execute, create process, wait, etc.
  - q Differs between single-tasking and multi-tasking.
- s File management**
  - q Create/delete file, open/close, read/write, etc.
- s Device management**
  - q Read, write, reposition, attach/detach device, etc.
- s Information maintenance.**
  - q Get time/date/process/file, set time/date/process/file, etc.
- s Communications**
  - q Send/receive messages , create/delete communication, etc.
  - q Two models for IPC (interprocess communication): messages-passing and shared-memory.

Operating System Concepts – 7<sup>th</sup> Edition, Jan 14, 2005
2.21
Silberschatz, Galvin and Gagne ©2005

## EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

	<b>Windows</b>	<b>Unix</b>
<b>Process Control</b>	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
<b>File Manipulation</b>	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
<b>Device Manipulation</b>	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
<b>Information Maintenance</b>	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
<b>Communication</b>	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
<b>Protection</b>	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

## API VERSUS SYSTEM CALL

API	SYSTEM CALL
A set of protocols, routines, functions that programmers use to develop software to facilitate interaction between distinct systems	A programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on
Helps to exchange data between various systems, devices and applications	Allows a program to access services from the kernel of the operating system

Visit [www.PEDIAA.com](http://www.PEDIAA.com)

# SYSTEM CALL VERSUS LIBRARY CALL

SYSTEM CALL	LIBRARY CALL
A request by the program to the kernel to enter kernel mode to access a resource	A request made by the program to access a function defined in a programming library
The mode changes from user mode to kernel mode	There is no mode switching
Not portable	Portable
Execute slower than library calls	Execute faster than system calls
System calls have more privileges than library calls	Library calls have less privileges than system calls
fork() and exec() are some examples for system calls	fopen(), fclose(), scanf() and printf() are some examples for library calls

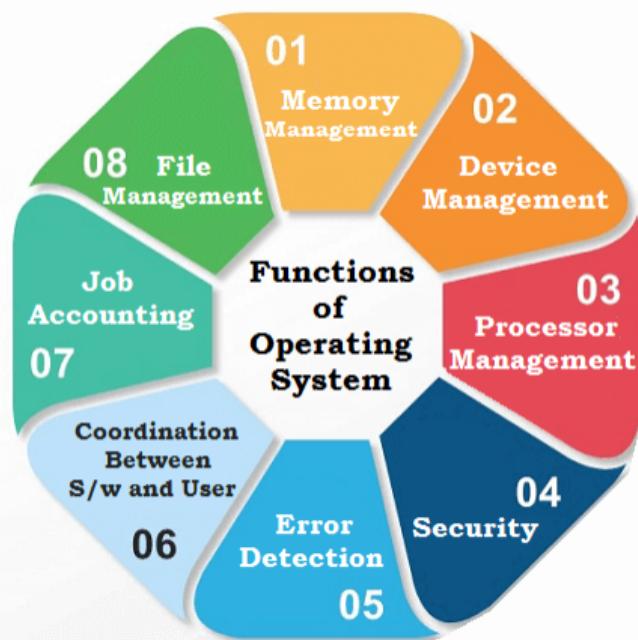
Visit [www.PEDIAA.com](http://www.PEDIAA.com)

# OPERATING SYSTEM VERSUS APPLICATION SOFTWARE

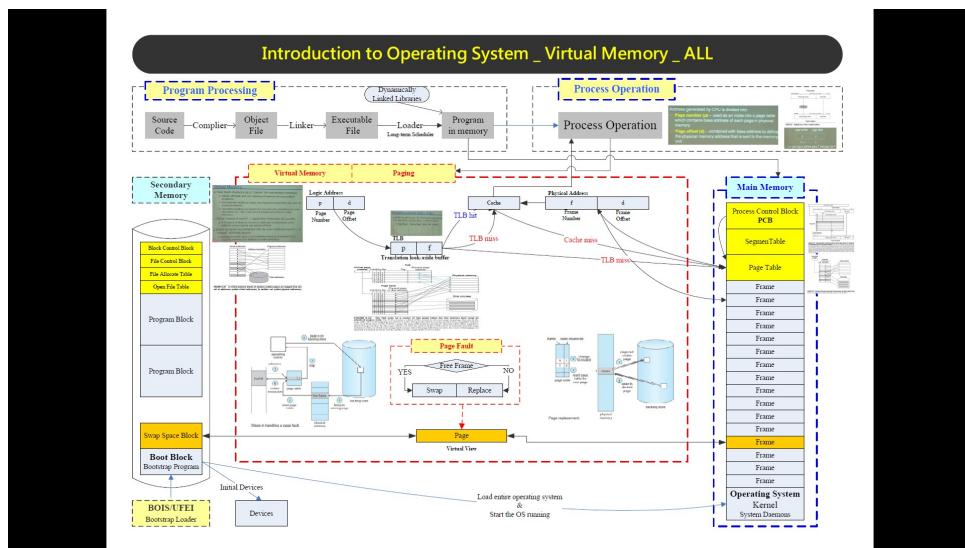
OPERATING SYSTEM	APPLICATION SOFTWARE
A system software that manages computer hardware and software resources and provides common services for computer programs	A software designed to perform a group of coordinated functions, tasks or activities for the benefit of the user
Works as the interface between the user and hardware, performs process management, memory management, task scheduling, hardware device controlling and many more	Performs a single specific task
Developed using C, C++, Assembly languages	Developed using Java, Visual Basic, C, C++
Boots up when the user switches on the computer and runs till he switches off the machine	Runs only when the user requests to run the application
Necessary for the proper functioning of the computer	Cannot be installed without an operating system
Ex: Windows, Unix, Linux, DOS	Ex: Word, Spreadsheet, Presentation, Multimedia tools, Database Management Systems

Visit [www.PEDIAA.com](http://www.PEDIAA.com)

## Functions of OS

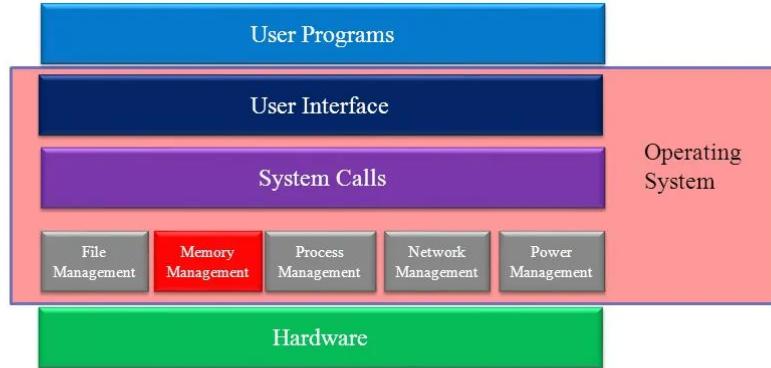


## Memory Management

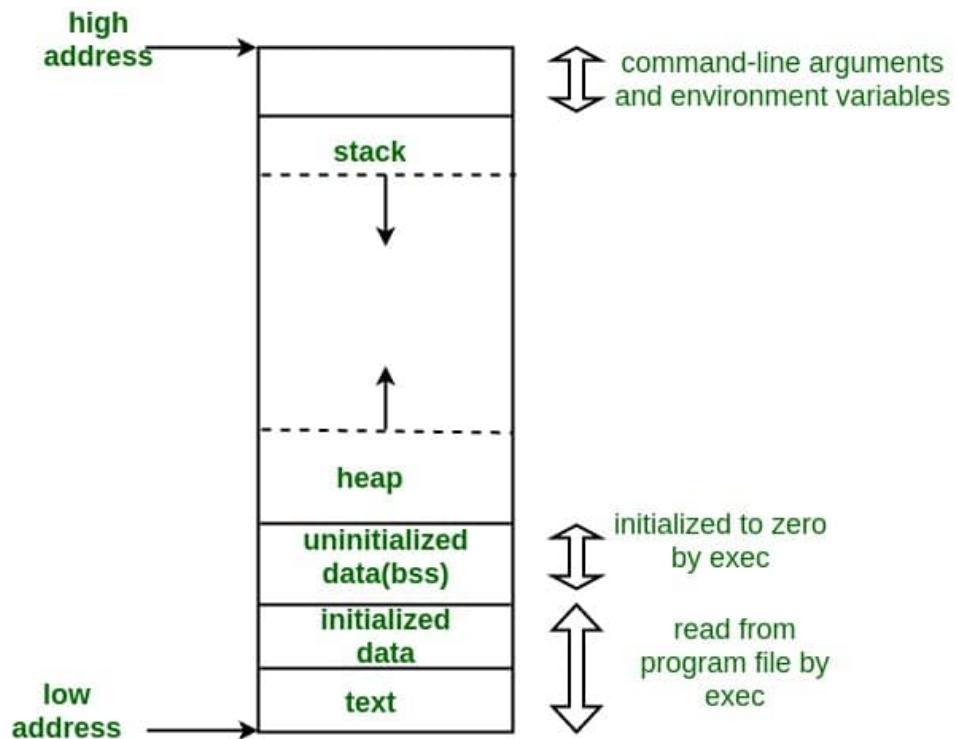


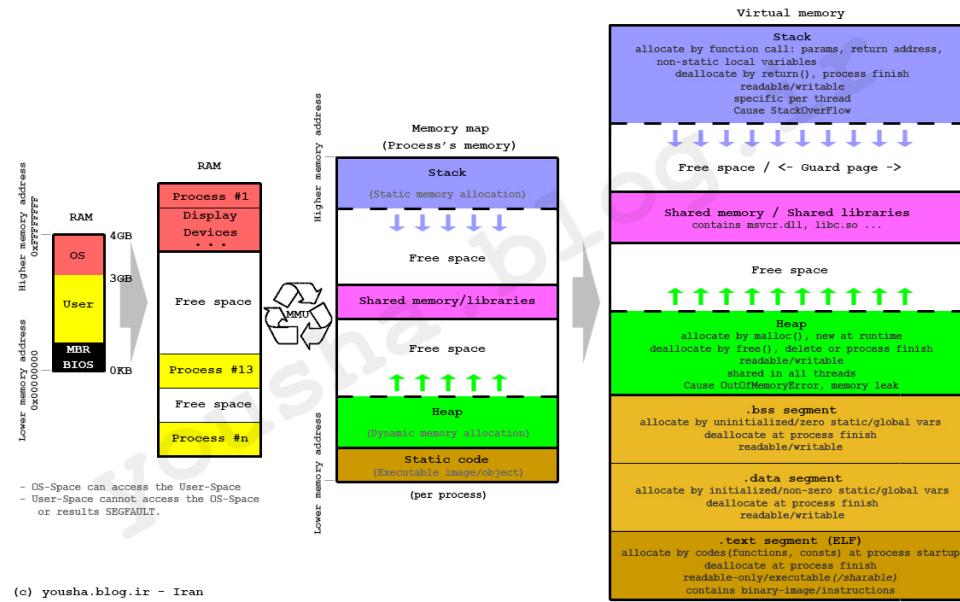


# Memory Management in OS

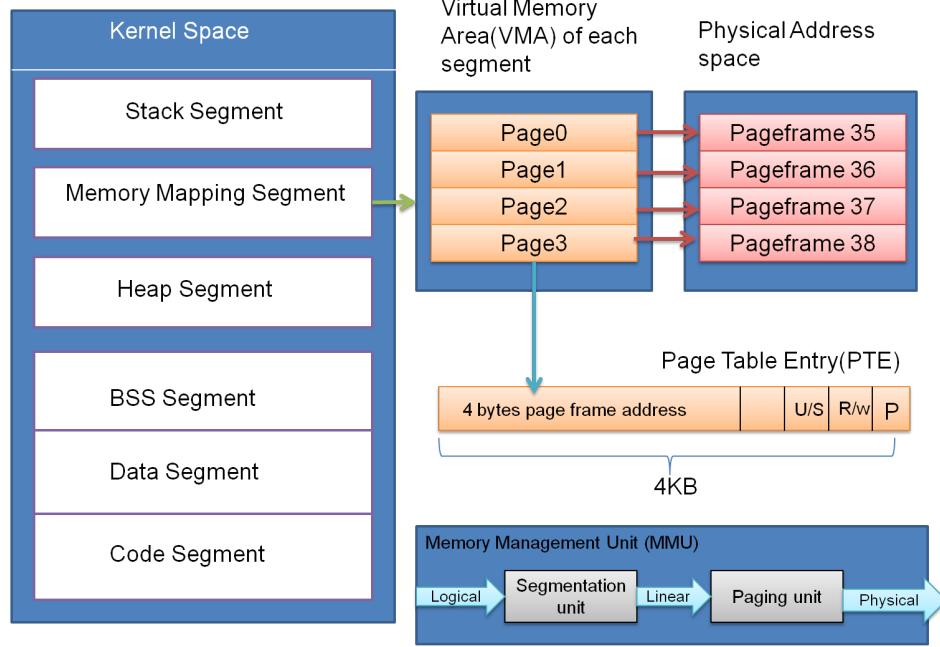


5





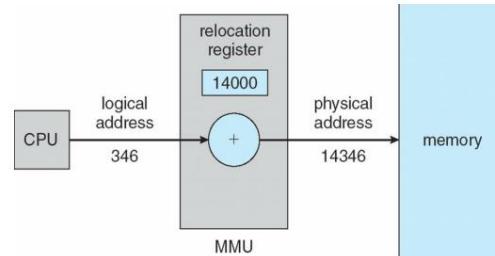
### Process Virtual Memory Layout





## Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with *logical addresses*; it never sees the *real* physical addresses



Dynamic relocation using a relocation register



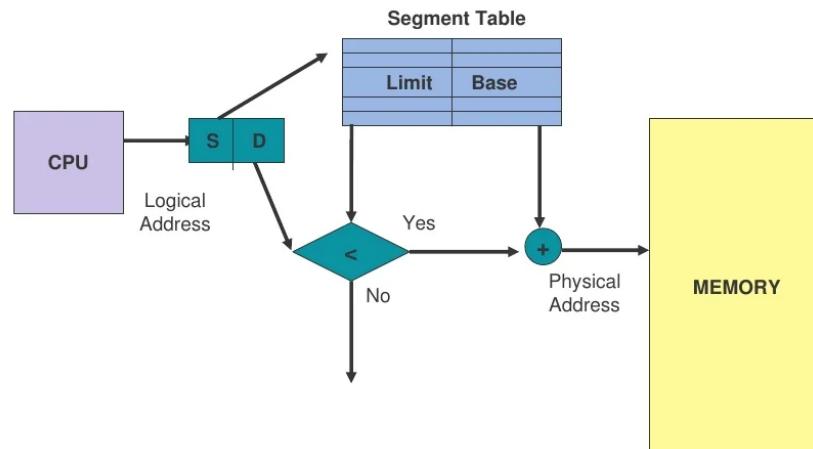
Operating System Concepts with Java – 8<sup>th</sup> Edition

8.8

Silberschatz, Galvin and Gagne ©2009

## MEMORY MANAGEMENT Segmentation

**HARDWARE** -- Must map a dyad (segment / offset) into one-dimensional address.



8: Memory Management

31

# MEMORY MANAGEMENT

## PAGING

Permits a program's memory to be physically noncontiguous so it can be allocated from wherever available. This avoids fragmentation and compaction.

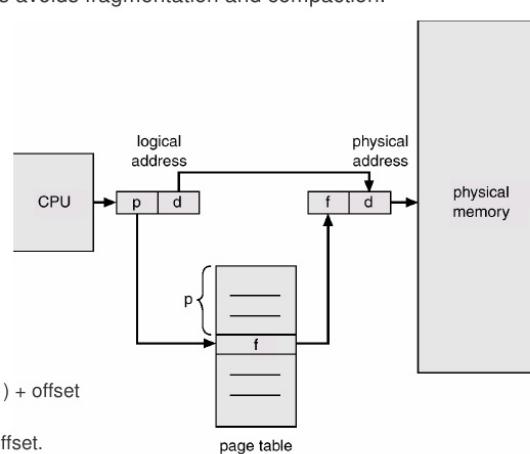
Frames = physical blocks  
Pages = logical blocks

Size of frames/pages is defined by hardware (power of 2 to ease calculations)

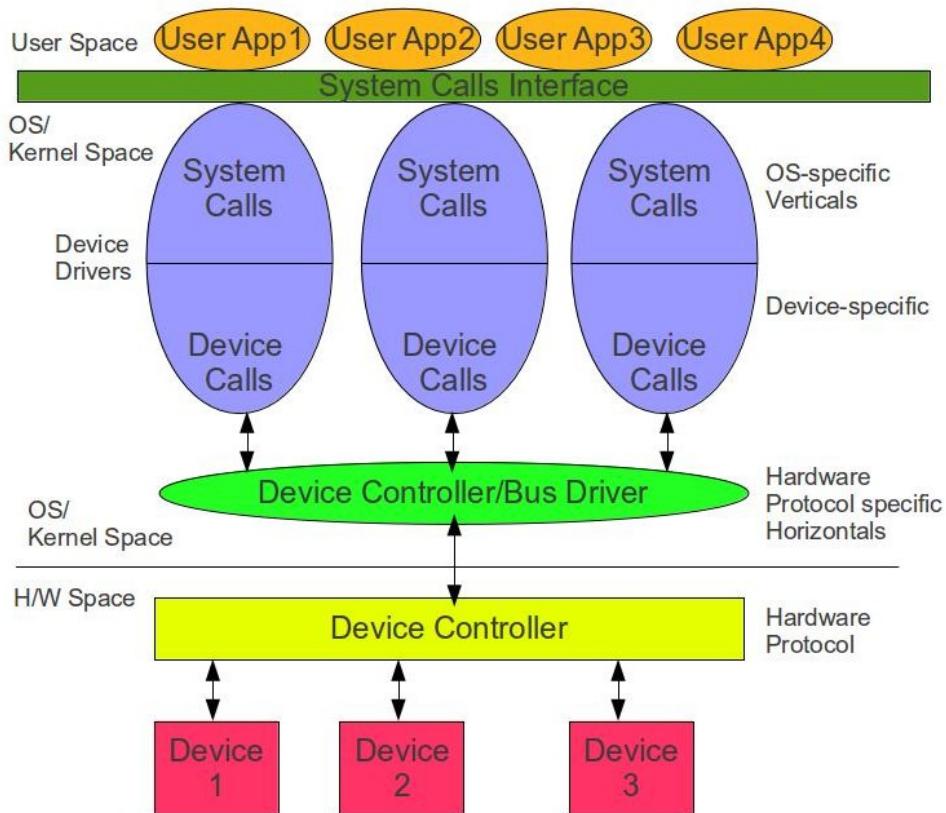
### HARDWARE

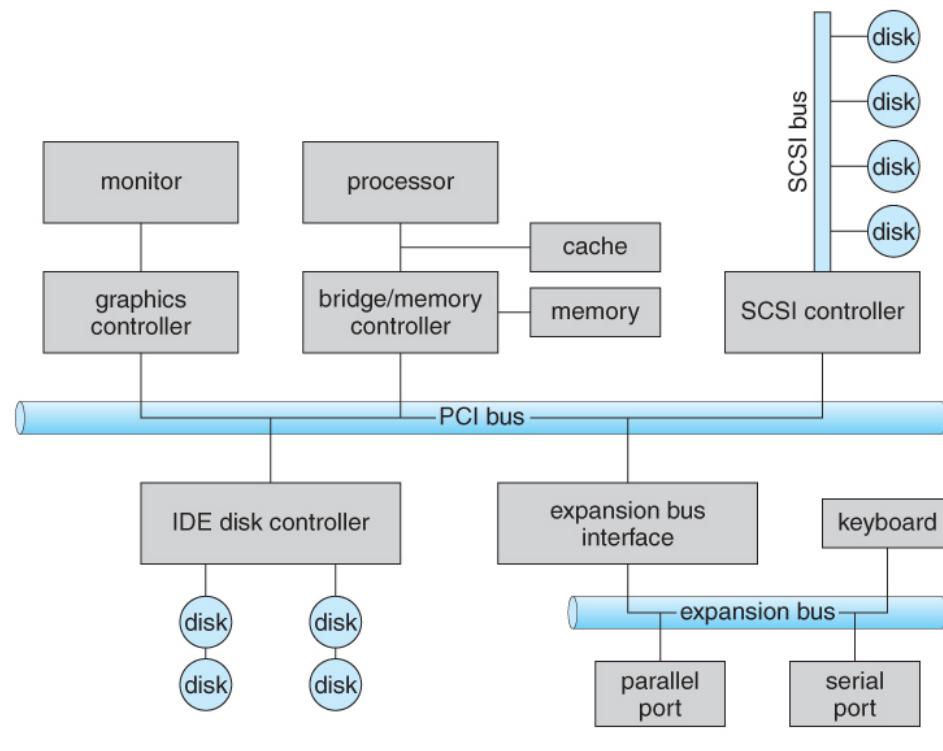
An address is determined by:

page number ( index into table ) + offset  
---> mapping into --->  
base address ( from table ) + offset.

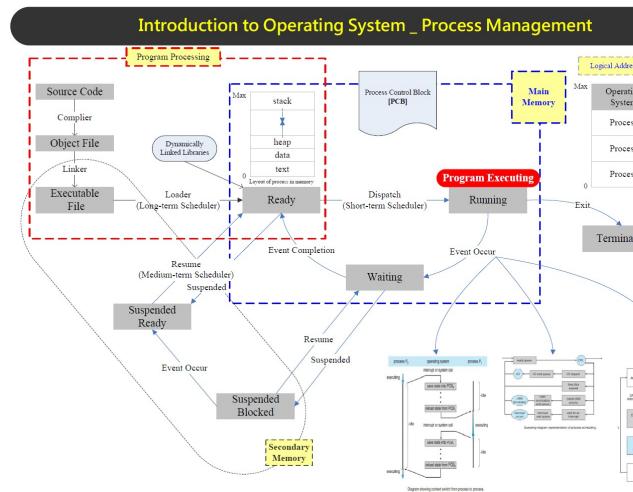


## - Device Management

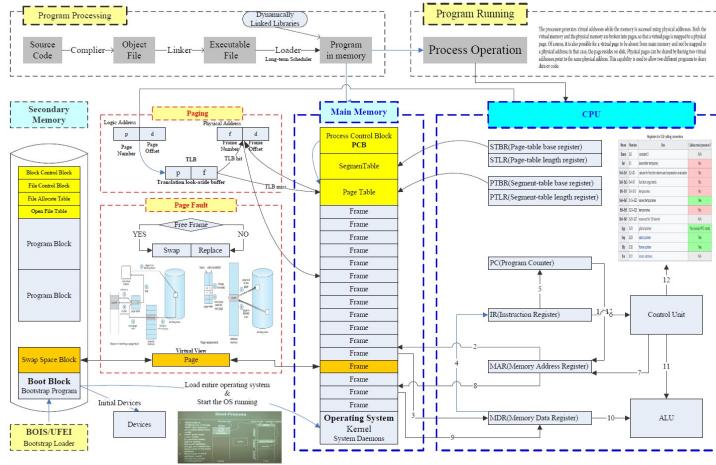




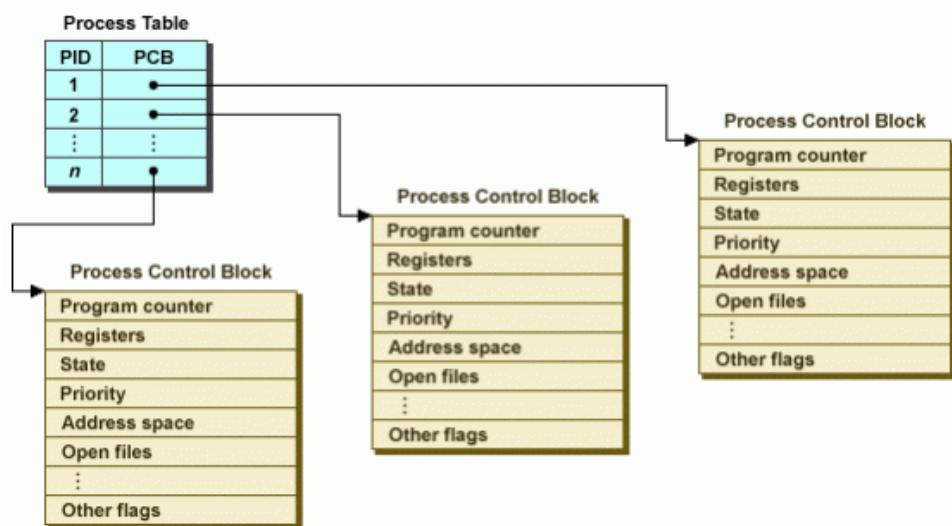
## Process Management



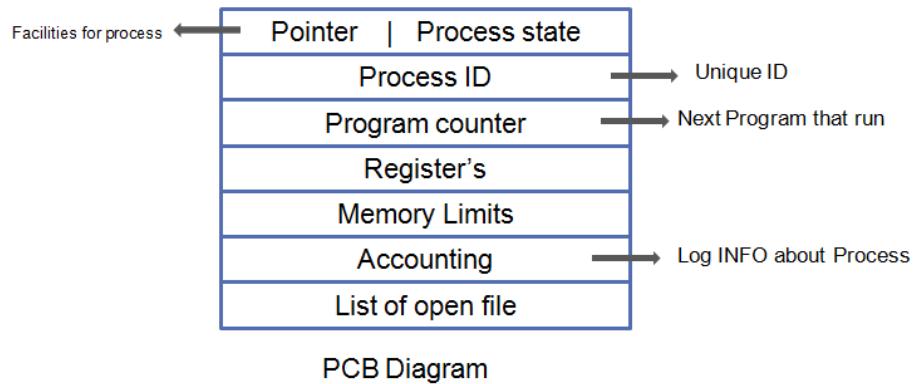
### Introduction to Operating System \_ Program Executing \_ ALL



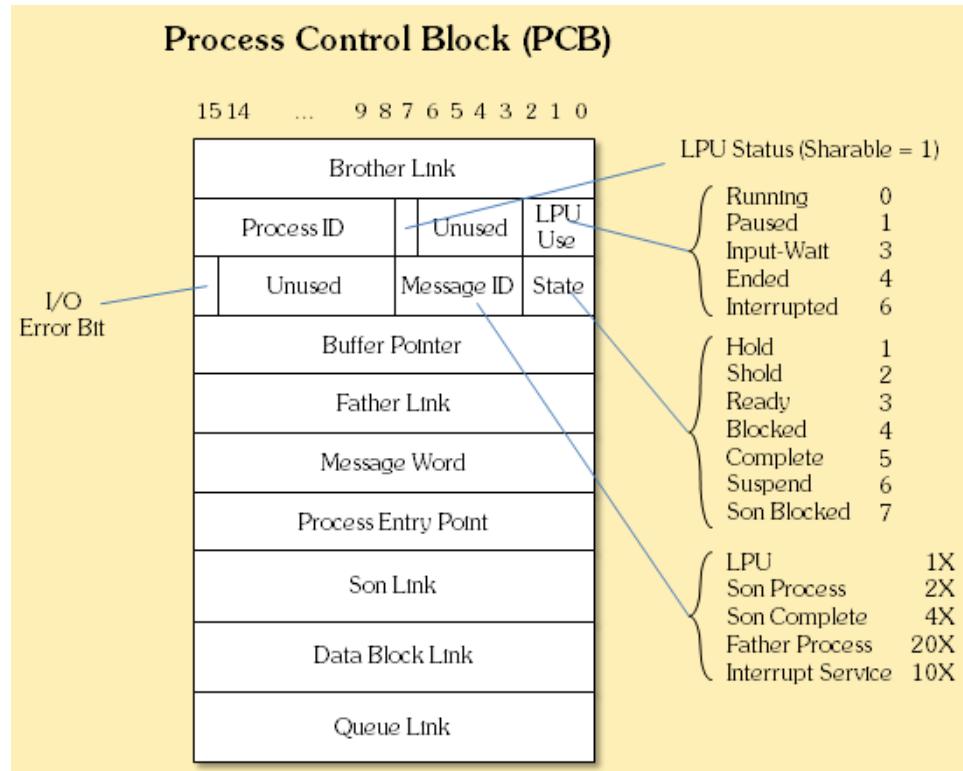
### - PCB(Process Control Block)



## PROCESS CONTROL BLOCK (PCB)

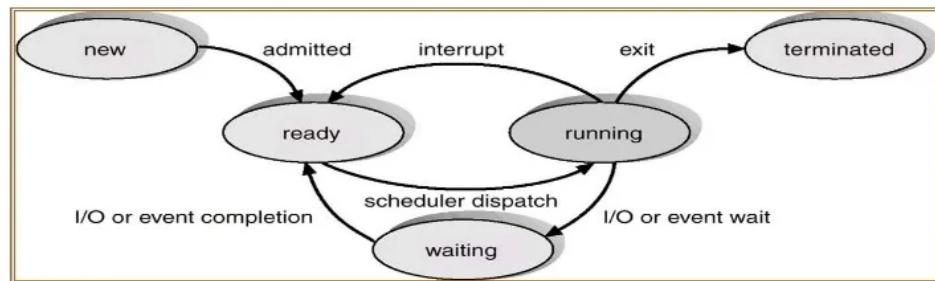


PCB Diagram



### - Process State Flow

# Process Management

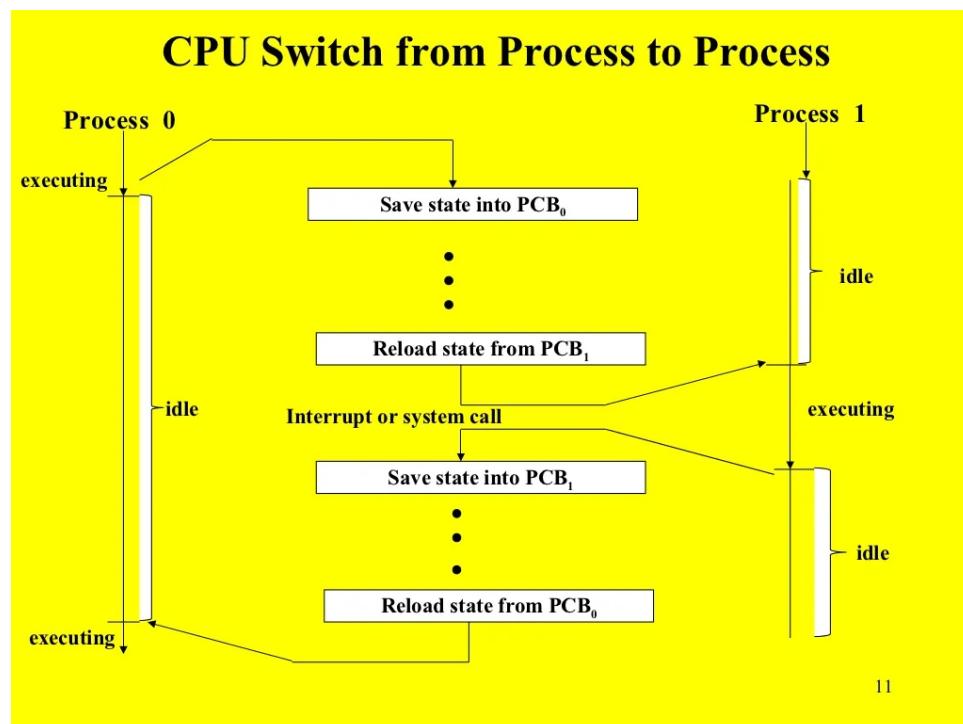


## Process States

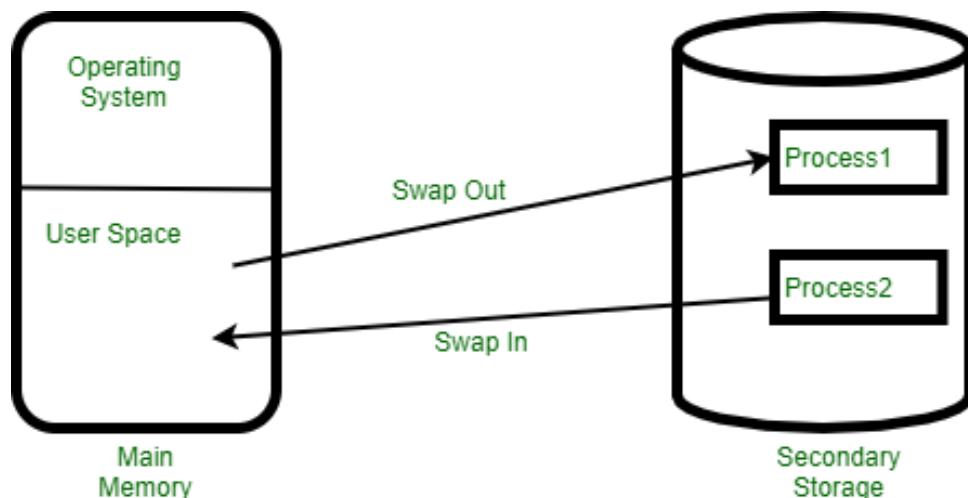
- New- The process is being created.
- Running- Instructions are being executed.
- Waiting- The process is waiting for some event to occur.
- Ready- The process is waiting to be assigned to a processor.
- Terminated- The process has finished execution.

Process Control Block- contains all information associated with a specific process like process state, program counter, CPU registers and info regarding CPU scheduling algorithms, memory, I/O and accounting.

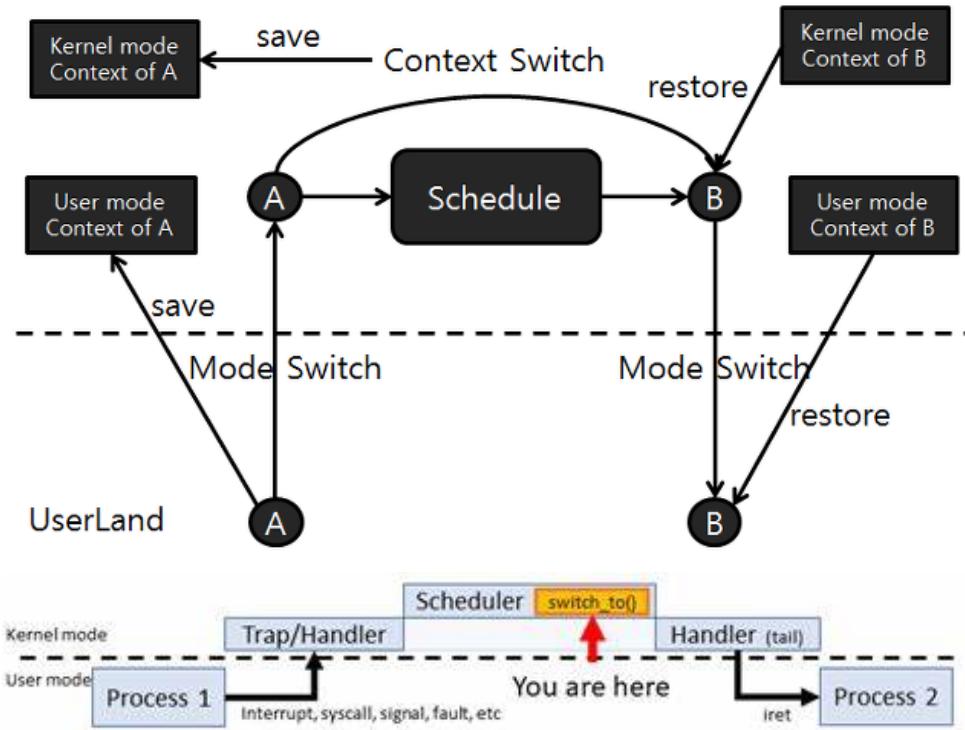
## - Context Switch and SWAP



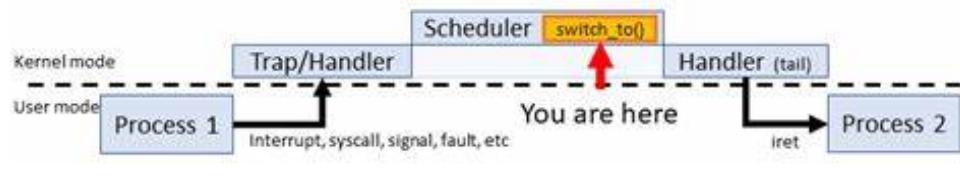
11

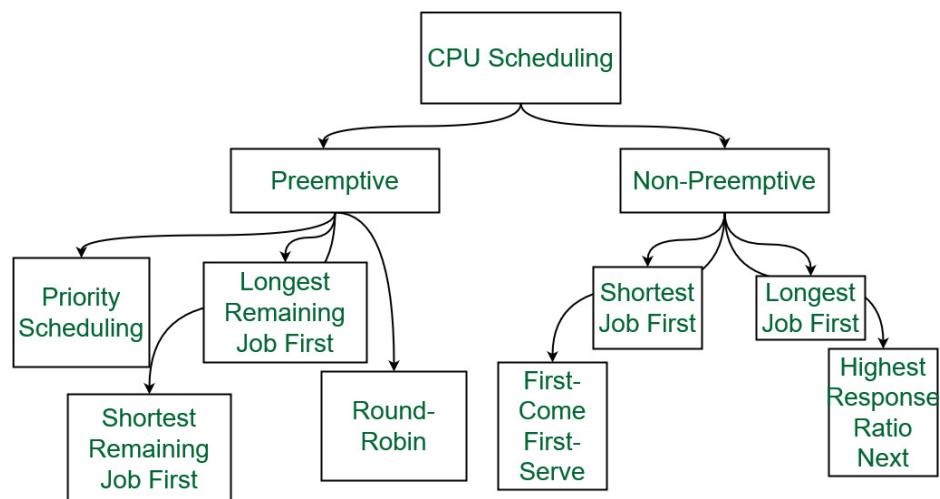
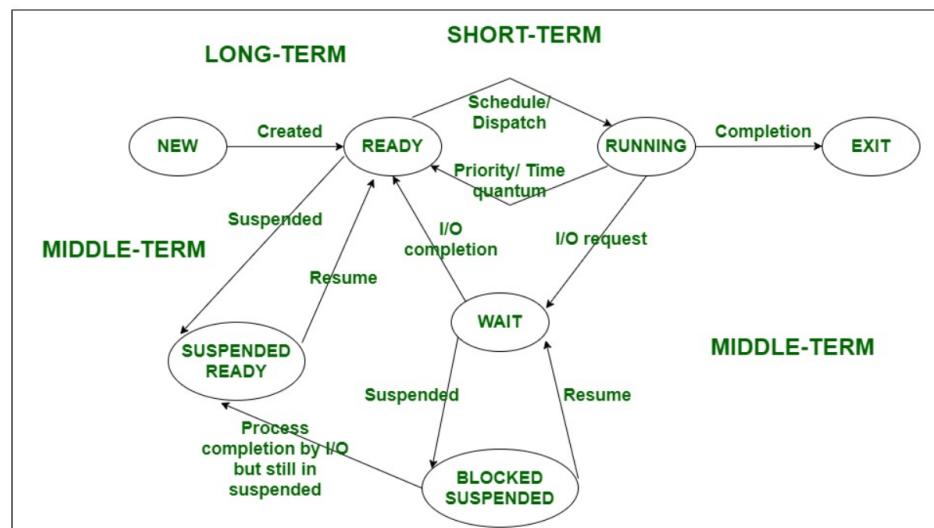
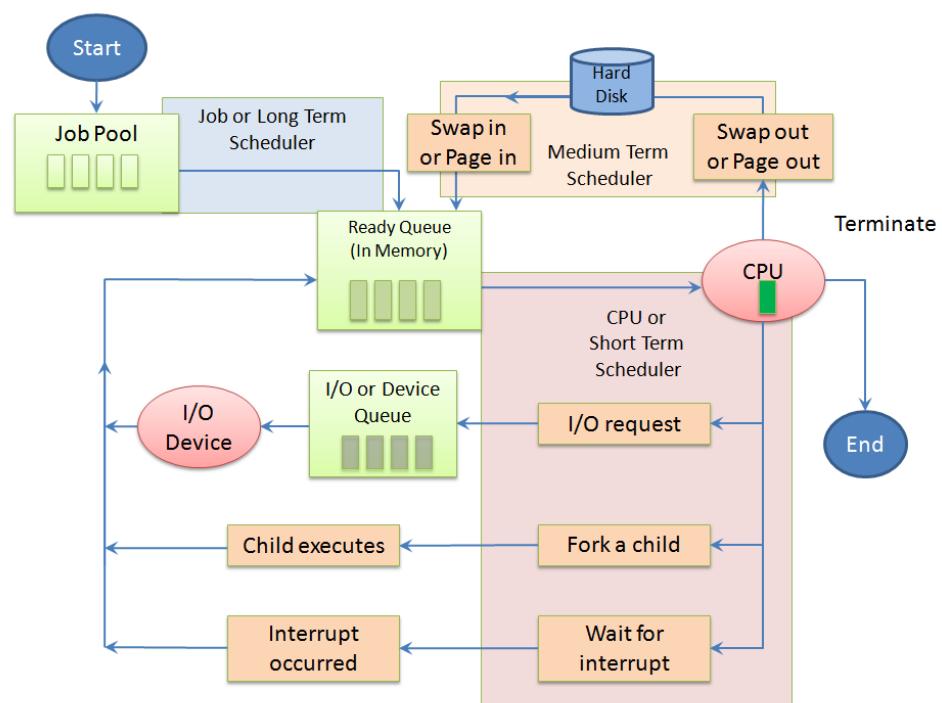


## KernelLand



## - Scheduling





## CPU SCHEDULING CRITERIA

- CPU Utilization: Percent of time that the CPU is busy executing a process.
- Throughput: Number of processes executed per unit time.
- Turnaround Time: The interval of time between submission of a process and its completion.
- Waiting Time: The amount of time the process spends in the ready queue waiting for the CPU.
- Response Time: The time between submission of requests and first response to the request.

CPU Scheduling (40 points)

Process	Burst Time	Priority	Arrival Time
P1	12	3	0
P2	6	4	2
P3	4	1	4
P4	18	2	6

Table 1: Process Information.

Consider the processes described in Table 1.

*Questions:* What is the average waiting time of those processes for each of the following scheduling algorithms? (Draw a Gantt chart for each algorithm.)

- (a) First Come First Serve (FCFS)
- (b) Non-preemptive Shortest Job First (NP-SJF)
- (c) Preemptive Shortest Job First (P-SJF)
- (d) Priority Scheduling
- (e) Round Robin, with the following assumptions:

Assumption (1). The scheduling time quantum is 5 time units.

Assumption (2). If a new process arrives at the same time as the time slice of the executing process expires, the OS puts the executing process in the ready queue, followed by the new process.

**Question 3:** Consider the following set of processes, their arrival times, length of the CPU burst time given in milliseconds:

Process	Arrival Time	Burst Time	Priority
P1	1	12	3
P2	2	3	1
P3	2	16	4
P4	3	10	2
P5	7	1	1

Let us schedule the execution of these processes using the following scheduling algorithms: First Come First Served (FCFS), Shortest Job First (SJF), Round Robin (RR), and a new type of scheduling algorithm called non-preemptive priority scheduling. The following are the assumptions:

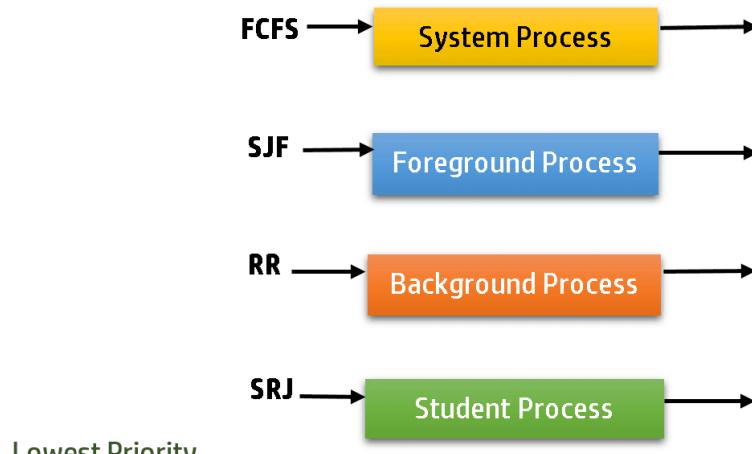
- 1) Larger priority number implies higher priority
- 2) Assume the quantum (aka time slice) of 2 for RR algorithm
- 3) Non-Preemptive means once scheduled, a process cannot be preempted (i.e. it runs to completion).

**3a [10 points]** What is the response (completion) time of each process for each of the scheduling algorithms. Write the times in the table below.

	FCFS	SJF	Priority	RR
P1				
P2				
P3				
P4				
P5				

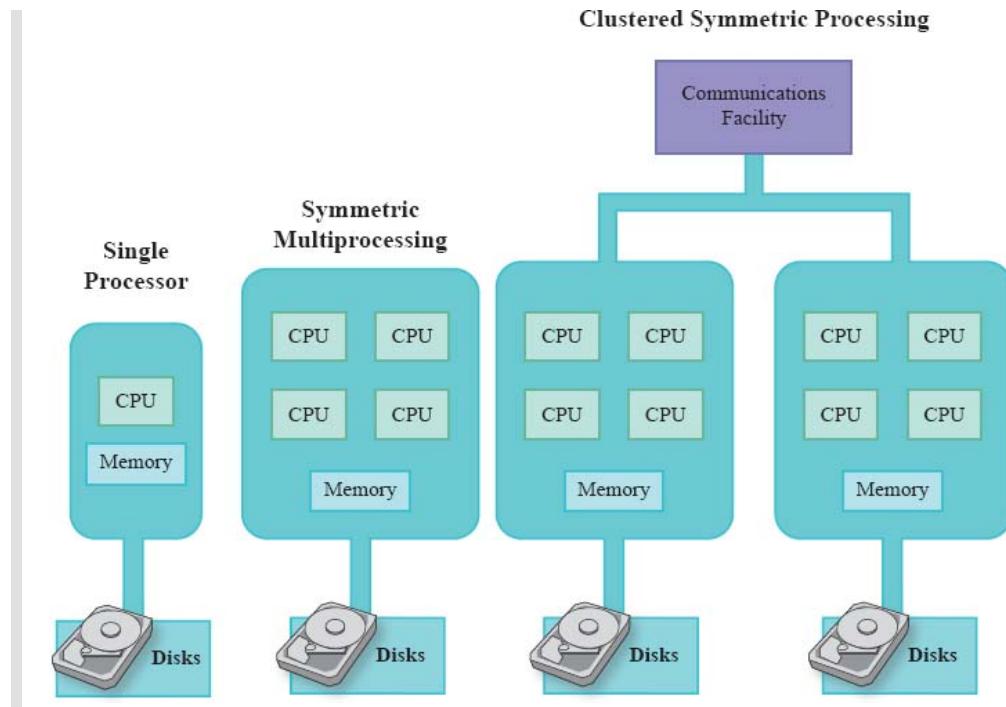
**3b [10 points]** Which of the above methods results in the longest average wait time. Show calculations and explain.

### Highest Priority



created by Notes\_Jam

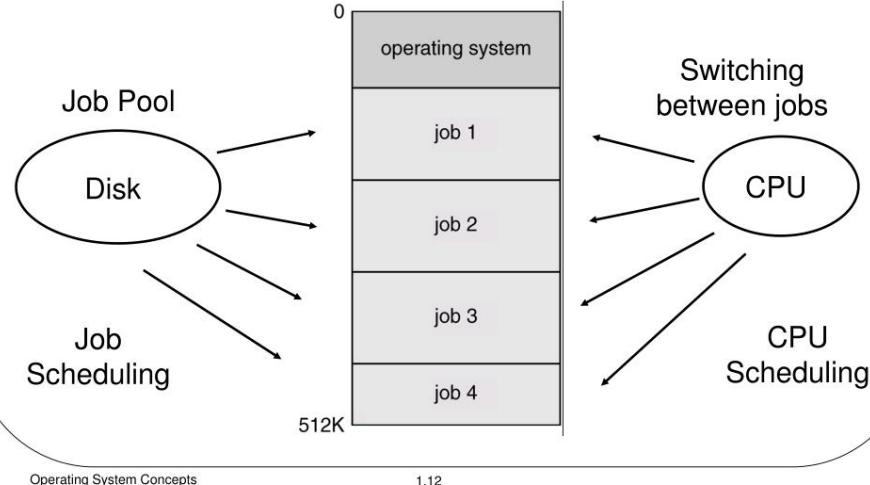
### - SP vs MP vs Cluster

**Clustered Symmetric Processing**

- Multiprogramming

## Multiprogramming Batch Systems

Multiprogramming: several jobs are kept in main memory at the same time, and the CPU is multiplexed among them which requires memory management and protection.



Operating System Concepts

1.12

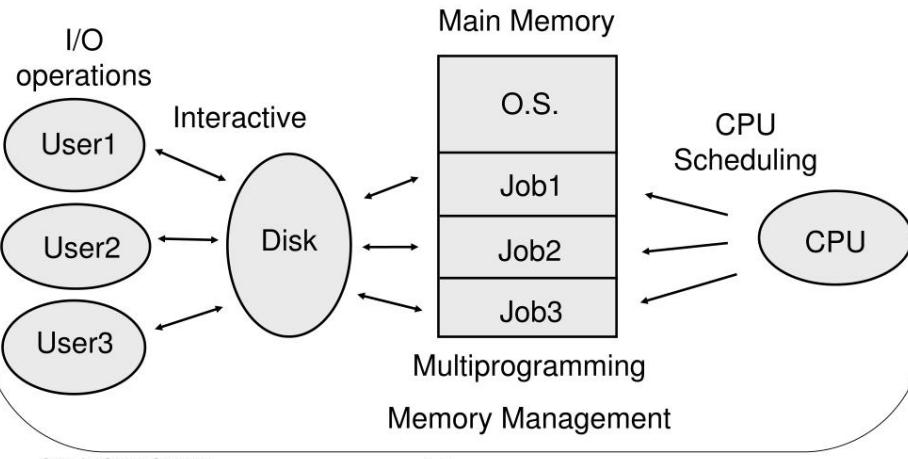
## MULTIPROGRAMMING VERSUS MULTITASKING

Multiprogramming	Multitasking
In multiprogramming, multiple processes run concurrently at the same time on a single processor.	Multitasking is when more than one task is executed at a single time utilizing multiple CPUs
It is based on the concept of context switching.	It is based on the concept of time sharing.
Multiple programs reside in the main memory simultaneously to improve CPU utilization so that CPU doesn't sit idle for a long time.	It enables execution of multiple tasks and processes at the same time to increase CPU performance.
It utilizes single CPU for execution of processes.	It utilizes multiple CPUs for task allocation.
It takes more time to execute the processes.	It takes less time to execute the tasks or processes.
The idea is to reduce the CPU idle time for as long as possible.	The idea is to allow multiple processes to run simultaneously via time sharing.

- Time Sharing

## Time-Sharing Systems – Interactive Computing

A time-sharing system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer.

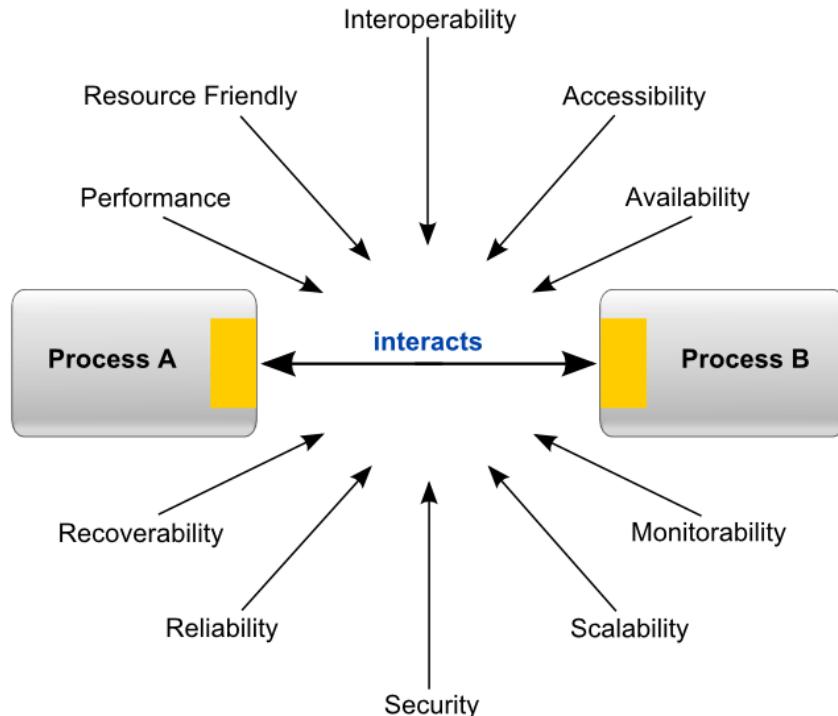


Operating System Concepts

1.15

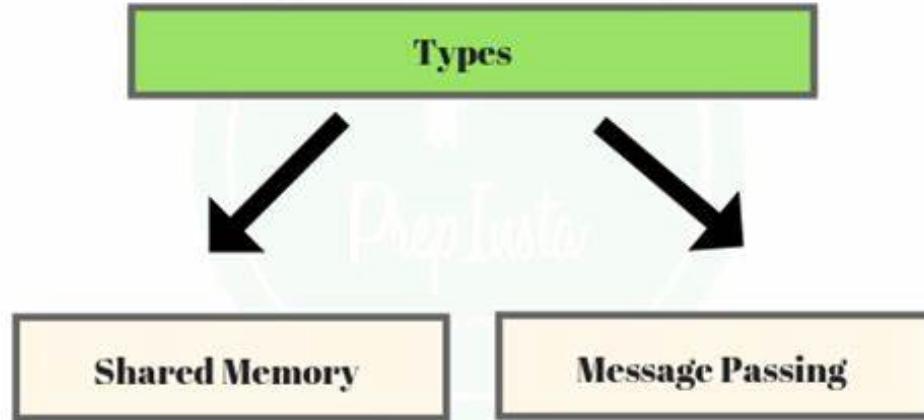
## Process Communication

### Interprocess Communication



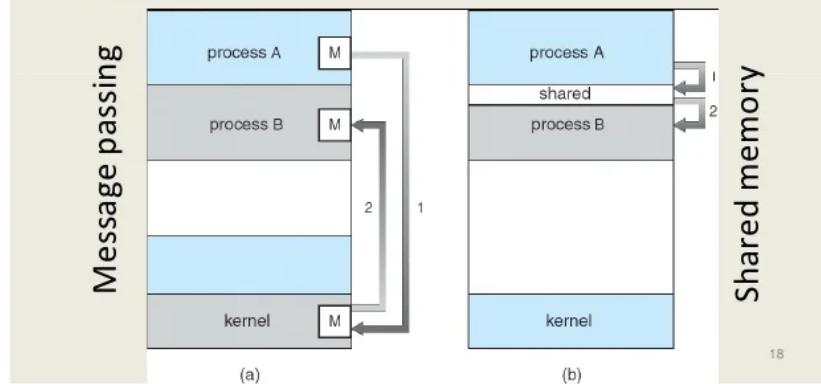


# Inter-Process Communication Types



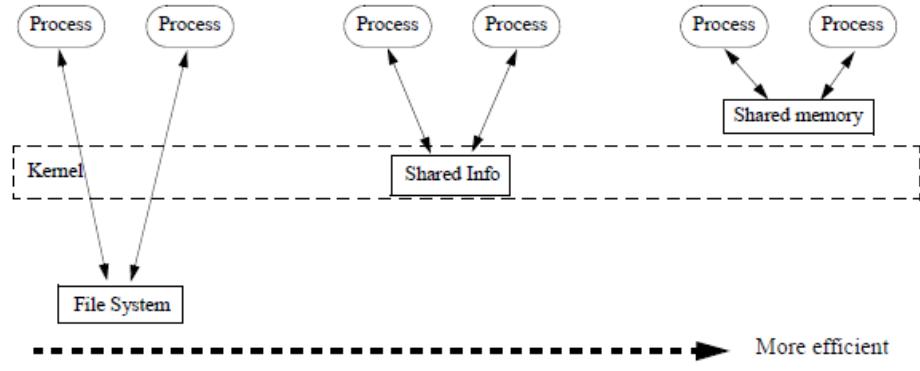
## 4. Interprocess Communication Contd...

- Two fundamental models of Interprocess communication
- **Shared memory**
  - a region of memory that is shared by cooperating processes is established then exchange information takes place by reading and writing data to the shared region
- **Message passing**
  - communication takes place by means of messages exchanged between the cooperating processes



## IPC Mechanisms

IPC classified by implementation mechanisms.

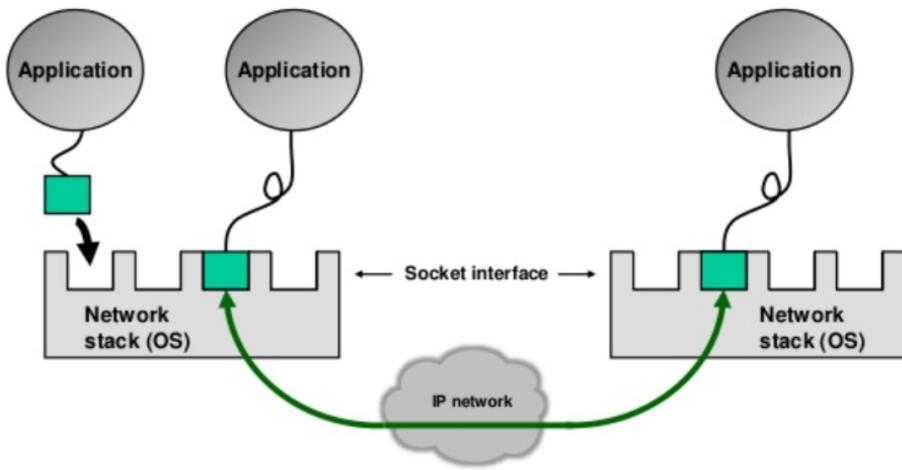


Examples:

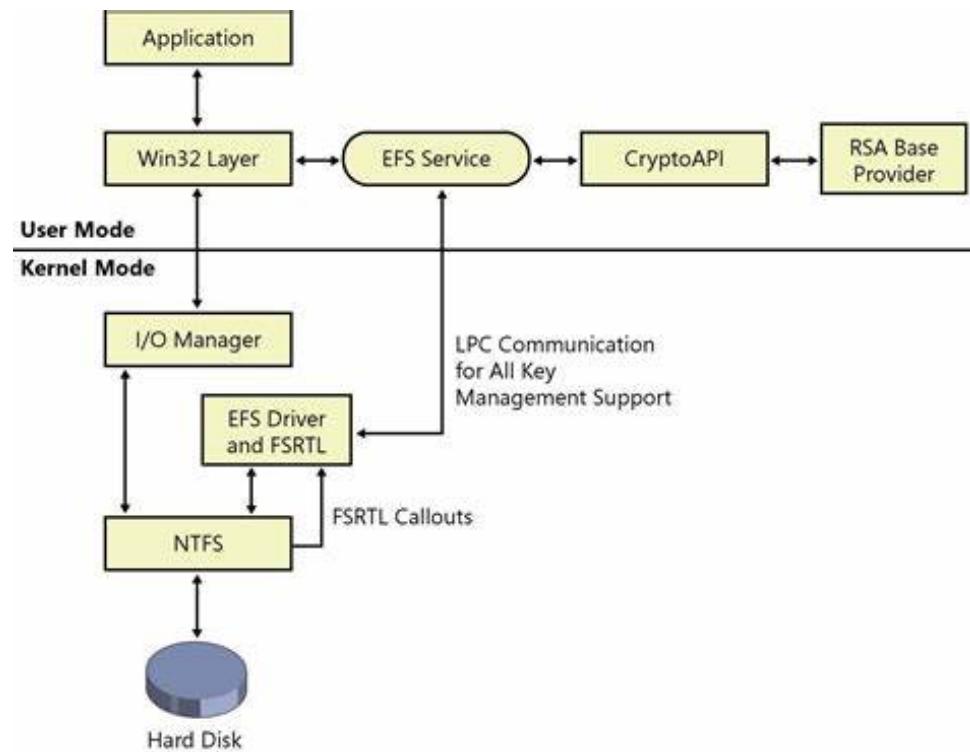
files

semaphore, socket,  
pipe, message queue,  
signal

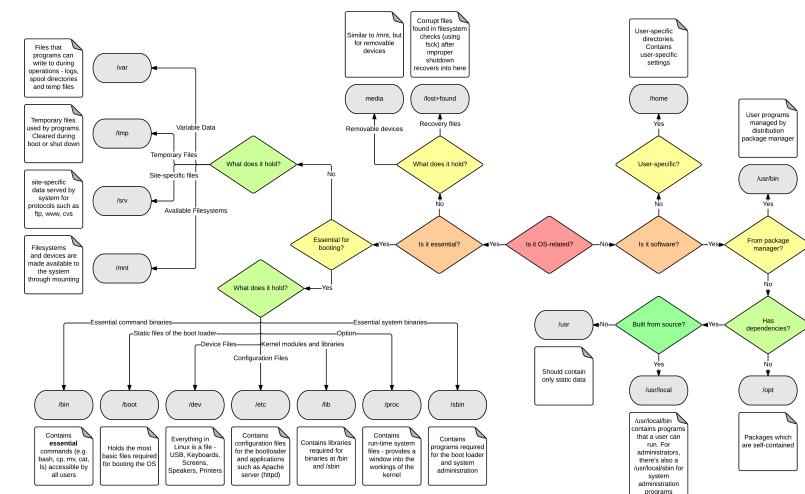
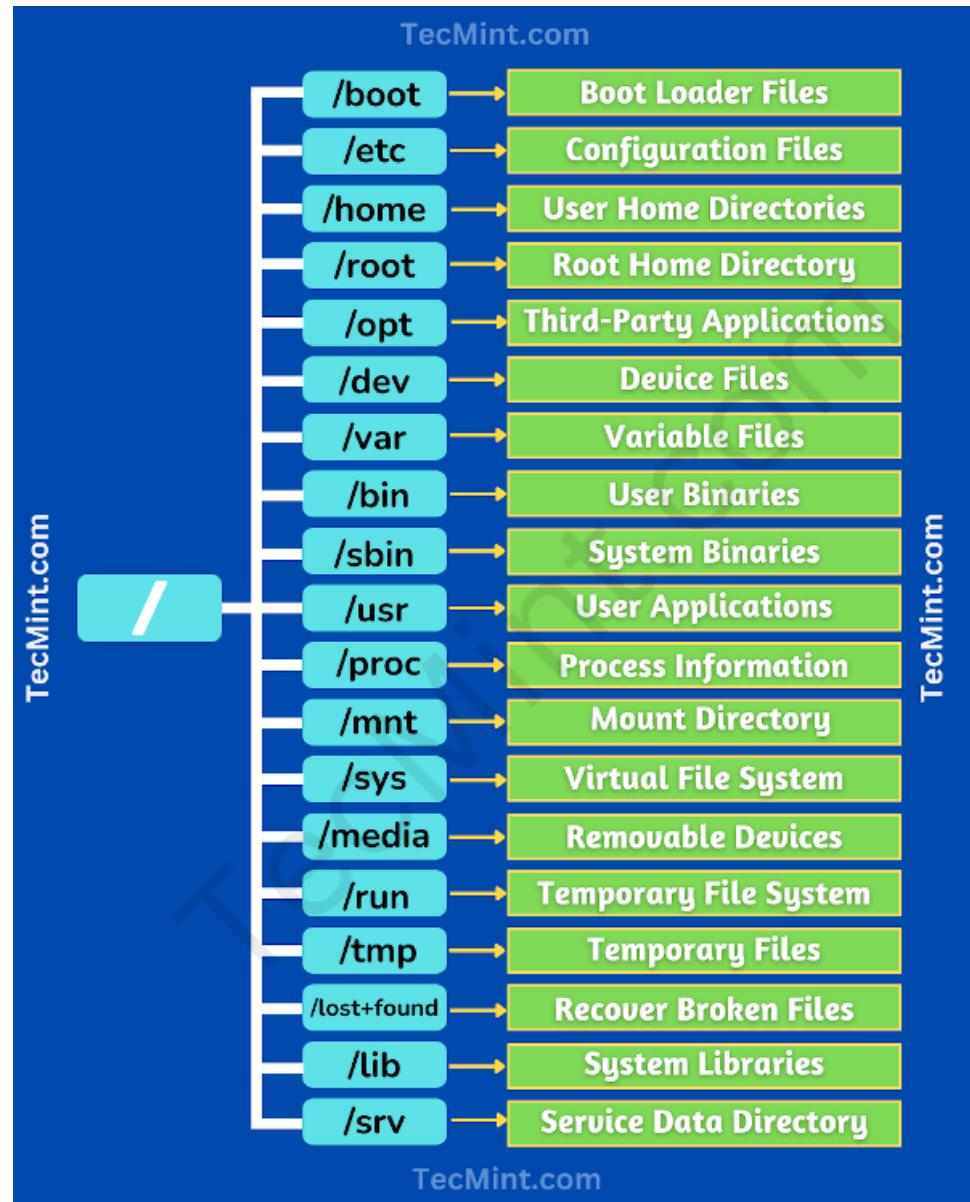
shared memory

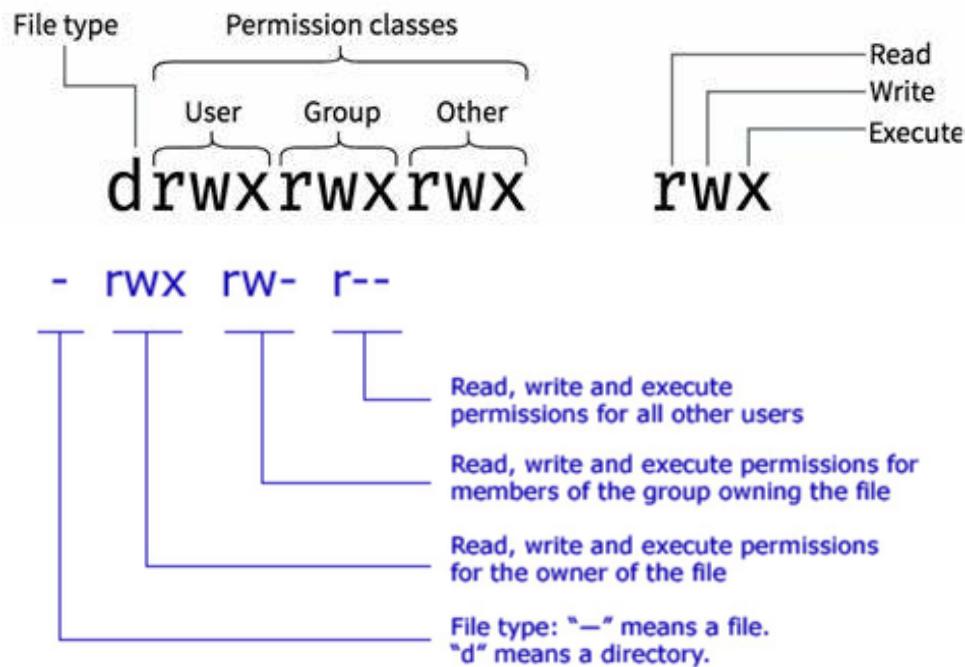
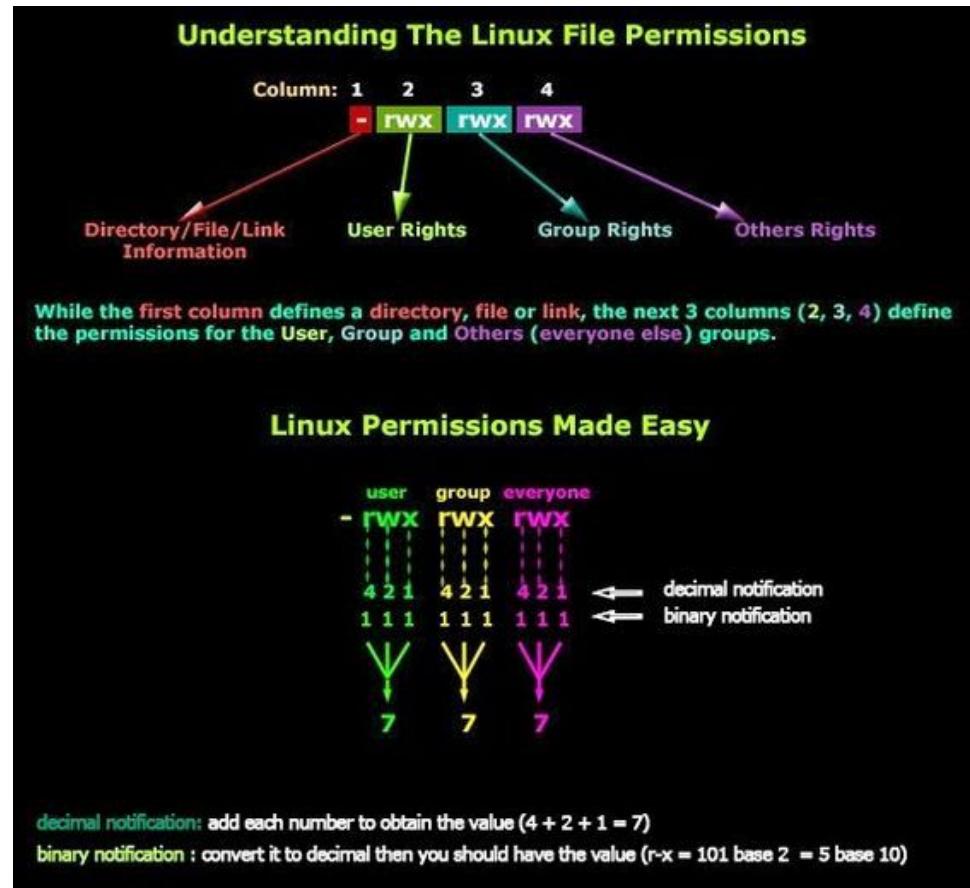


- File Management



## File Structure in Linux





## File Attributes

## File Attributes

A file's attributes vary from one operating system to another but typically consist of these:

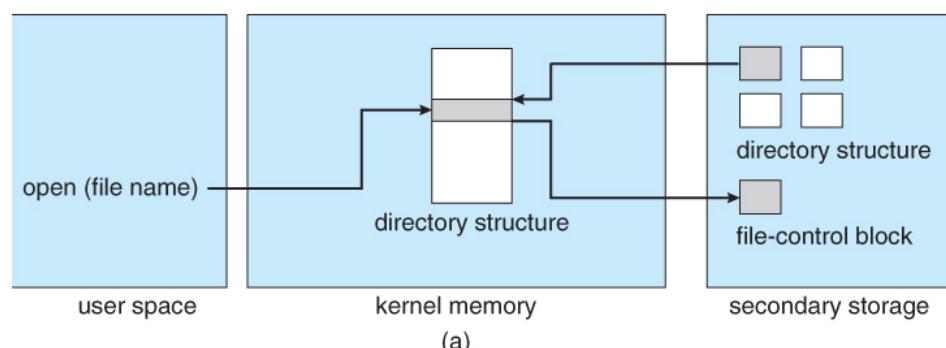
- ▶ **Name** – only information kept in human-readable form
- ▶ **Identifier** – unique tag (number) identifies file within file system
- ▶ **Type** – needed for systems that support different types
- ▶ **Location** – pointer to file location on device
- ▶ **Size** – current file size
- ▶ **Protection** – controls who can do reading, writing, executing
- ▶ **Time, date, and user identification** – data for protection, security, and usage monitoring
- ▶ Information about files are kept in the **directory structure**, which is maintained on the disk. Typically, a directory entry consists of the file's name and its unique identifier.

SHASHI KS

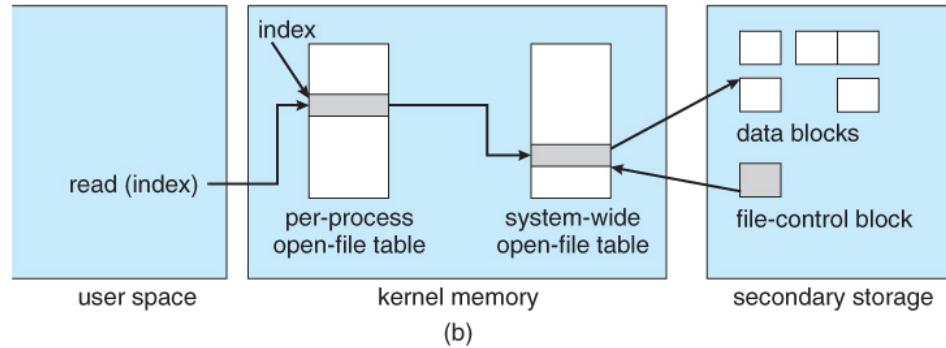
## File Type

File type	Usual extension	Function
Executable	exe,com,bin	Read to run machine language program
Object	obj,o	Compiled,machine language not linked
Source code	C,java,pas,asm,a	Source code in various languages
Batch	bat,sh	Commands to the command interpreter
Text	txt,doc	Textual data,documents
Word processor	Wp,tex,rrf,doc	Various word processor formats
Archive	arc,zip,tar	Related files grouped into one file compressed

## File Access



(a)



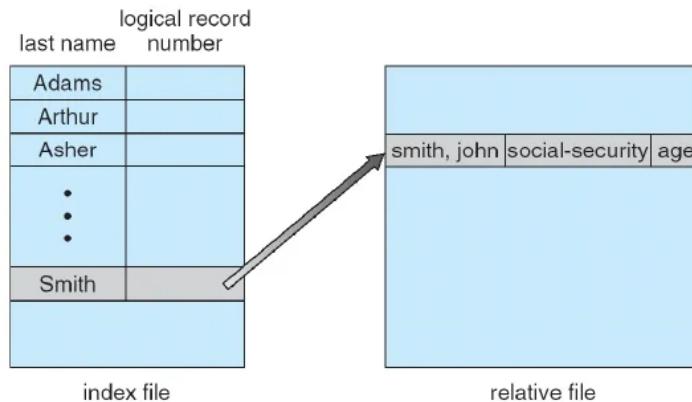
(b)

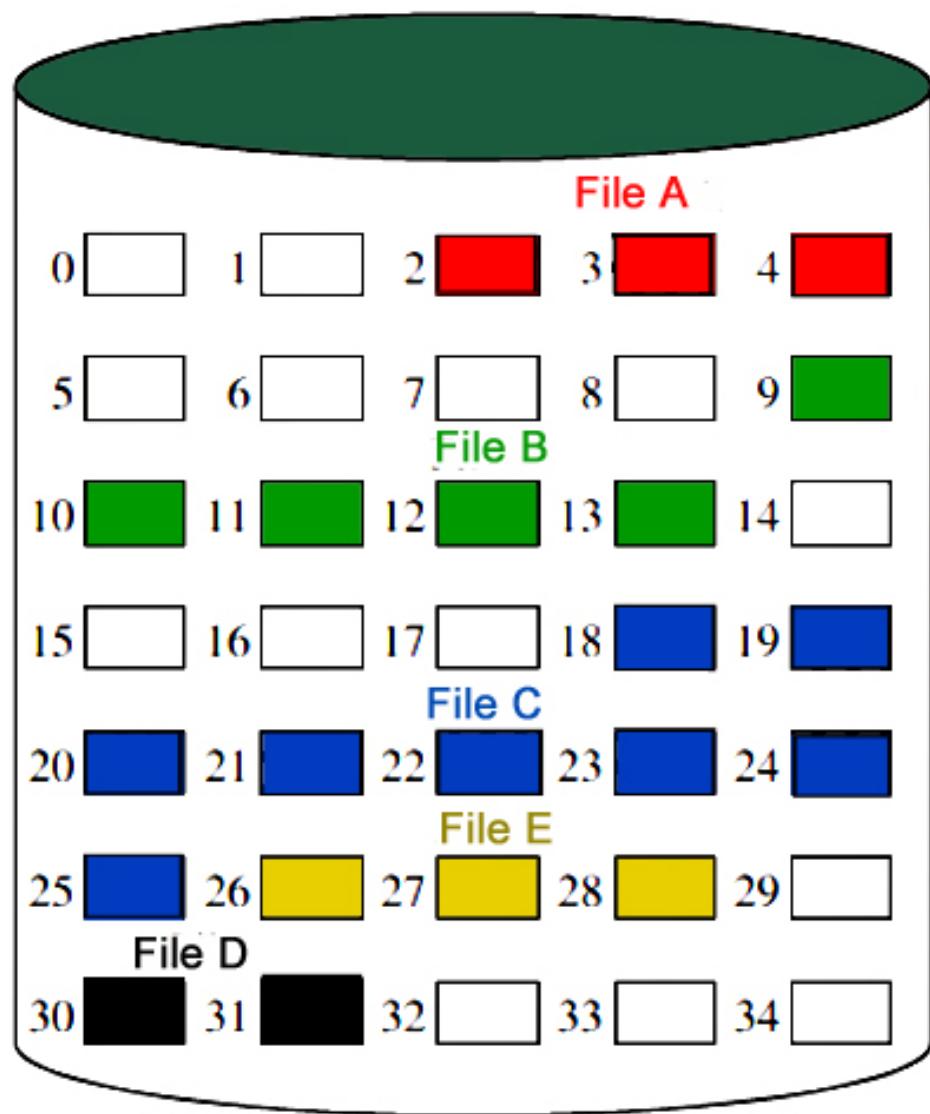
## 2. Access Methods

Contd...

### 2.3 Other Access Methods

- Built on top of a direct-access method
- Example of Index and Relative Files





File allocation table

File name	Start block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

## Disk Scheduling Algorithm



## Selecting a Disk-Scheduling Algorithm

- Simulation results
  - low load --> SCAN
  - medium to heavy load --> C-SCAN
- influenced by the file-allocation method
  - contiguously allocated file
  - a linked or indexed file
- the location of directories and index blocks
  - placing the directories halfway between the inner and outer edge of the disk
  - placing the directories at either end

SunMoon University

19

## Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

16

## Disk Scheduling Algorithms

Selection according to requestor		
<b>RSS</b>	<b>Random scheduling</b>	<b>For analysis &amp; simulation</b>
<b>FIFO</b>	<b>First in first out</b>	<b>Fairest of them all</b>
<b>Priority</b>	<b>Priority by process</b>	<b>No disk optimization</b>
<b>LIFO</b>	<b>Last in first out</b>	<b>Max locality &amp; resource</b>
Selection according to requested item		
<b>SSTF</b>	<b>Shortest service time first</b>	<b>High utilization, small queues</b>
<b>SCAN</b>	<b>Back and forth over disk</b>	<b>Better service distribution</b>
<b>C-SCAN</b>	<b>One way with fast return</b>	<b>Lower service variability</b>
<b>N-step-SCAN</b>	<b>SCAN of N records at a time</b>	<b>Service guarantee</b>
<b>FSCAN</b>	<b>NsS w/N=queue at beginning of SCAN cycle</b>	<b>Load sensitive</b>

### Example

Trace the policies FIFO, SSTF, SCAN, C-SCAN and FSCAN for the following disk requests. Each I/O request on a track takes 5 time units. At time 0, the disk starts reading track 10, and the read/write head was moving to the larger track number direction .

Time	0	1	2	3	6	7
Request to access track ..	10	19	3	14	12	9

	Track access order	Average seek length
FIFO	10,19,3,14,12,9	$(9+16+11+2+3)/5 = 8.2$
SSTF	10,14,12,9,3,19	$(4+2+3+6+16)/5 = 6.2$
SCAN	10,14,19,12,9,3	$(4+5+7+3+6)/5 = 5$
C-SCAN	10,14,19,3,9,12	$(4+5+16+6+3)/5 = 6.8$
FSCAN	10,14,19,3,9,12	$(4+5+16+6+3)/5 = 6.8$



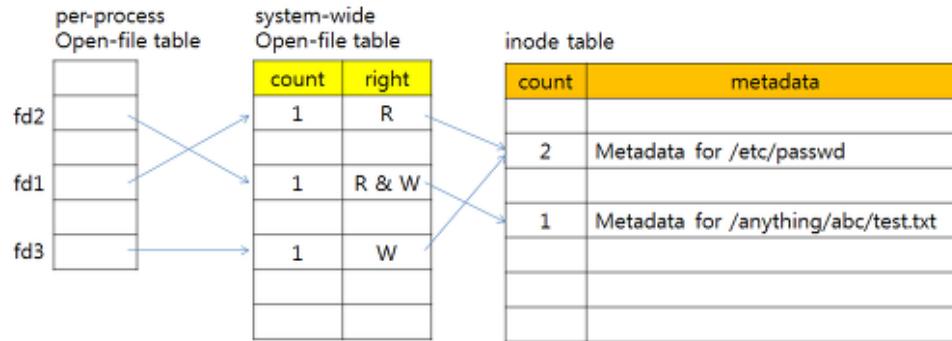
# Disk Scheduling Algorithms

Table 11.2 Comparison of Disk Scheduling Algorithms

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length		Average seek length		Average seek length		Average seek length	
55.3		27.5		27.8		35.8	

38

## File Operations



File operation	Declaration & Description
<b>fopen() - To open a file</b>	<p>Declaration: FILE *fopen (const char *filename, const char *mode)</p> <p>fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist.</p> <pre>FILE *fp; fp=fopen ("filename", "mode"); Where, fp - file pointer to the data type "FILE". filename - the actual file name with full path of the file. mode - refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.</pre>
<b>fclose() - To close a file</b>	<p>Declaration: int fclose(FILE *fp);</p> <p>fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below.</p> <pre>fclose (fp);</pre>
<b>fgets() - To read a file</b>	<p>Declaration: char *fgets(char *string, int n, FILE *fp)</p> <p>fgets function is used to read a file line by line. In a C program, we use fgets function as below.</p> <pre>fgets (buffer, size, fp); where, buffer - buffer to put the data in. size - size of the buffer fp - file pointer</pre>
<b>fprintf() - To write into a file</b>	<p>Declaration:</p> <pre>int fprintf(FILE *fp, const char *format, ...);</pre> <p>fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below.</p> <pre>fprintf (fp, "some data"); or fprintf (fp, "text %d", variable_name);</pre>

File operation	Declaration & Description
<b>fopen() - To open a file</b>	<p>Declaration: FILE *fopen (const char *filename, const char *mode)</p> <p>fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist.</p> <pre data-bbox="817 316 1091 370">FILE *fp; fp=fopen ("filename", "mode");</pre> <p>Where,</p> <p>fp - file pointer to the data type "FILE".</p> <p>filename - the actual file name with full path of the file.</p> <p>mode - refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.</p>
<b>fclose() - To close a file</b>	<p>Declaration: int fclose(FILE *fp);</p> <p>fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below.</p> <pre data-bbox="906 574 1006 601">fclose (fp);</pre>
<b>fgets() - To read a file</b>	<p>Declaration: char *fgets(char *string, int n, FILE *fp)</p> <p>fgets function is used to read a file line by line. In a C program, we use fgets function as below.</p> <pre data-bbox="858 698 1049 752">fgets (buffer, size, fp);</pre> <p>where,</p> <p>buffer - buffer to put the data in.</p> <p>size - size of the buffer</p> <p>fp - file pointer</p>
<b>fprintf() - To write into a file</b>	<p>Declaration:</p> <pre data-bbox="596 844 1309 929">int fprintf(FILE *fp, const char *format, ...);</pre> <p>fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below. fprintf (fp, "some data"); or</p> <pre data-bbox="787 900 1112 927">fprintf (fp, "text %d", variable_name);</pre>

File operation	Declaration & Description
<b>fopen() - To open a file</b>	<p>Declaration: FILE *fopen (const char *filename, const char *mode)</p> <p>fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist.</p> <pre data-bbox="817 1197 1091 1251">FILE *fp; fp=fopen ("filename", "mode");</pre> <p>Where,</p> <p>fp - file pointer to the data type "FILE".</p> <p>filename - the actual file name with full path of the file.</p> <p>mode - refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.</p>
<b>fclose() - To close a file</b>	<p>Declaration: int fclose(FILE *fp);</p> <p>fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below.</p> <pre data-bbox="890 1462 1006 1489">fclose (fp);</pre>
<b>fgets() - To read a file</b>	<p>Declaration: char *fgets(char *string, int n, FILE *fp)</p> <p>fgets function is used to read a file line by line. In a C program, we use fgets function as below.</p> <pre data-bbox="858 1603 1049 1630">fgets (buffer, size, fp);</pre> <p>where,</p> <p>buffer - buffer to put the data in.</p> <p>size - size of the buffer</p> <p>fp - file pointer</p>
<b>fprintf() - To write into a file</b>	<p>Declaration:</p> <pre data-bbox="596 1749 1309 1834">int fprintf(FILE *fp, const char *format, ...);</pre> <p>fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below. fprintf (fp, "some data"); or</p> <pre data-bbox="787 1799 1112 1825">fprintf (fp, "text %d", variable_name);</pre>

## - OS Security

## Precautions for OS Security

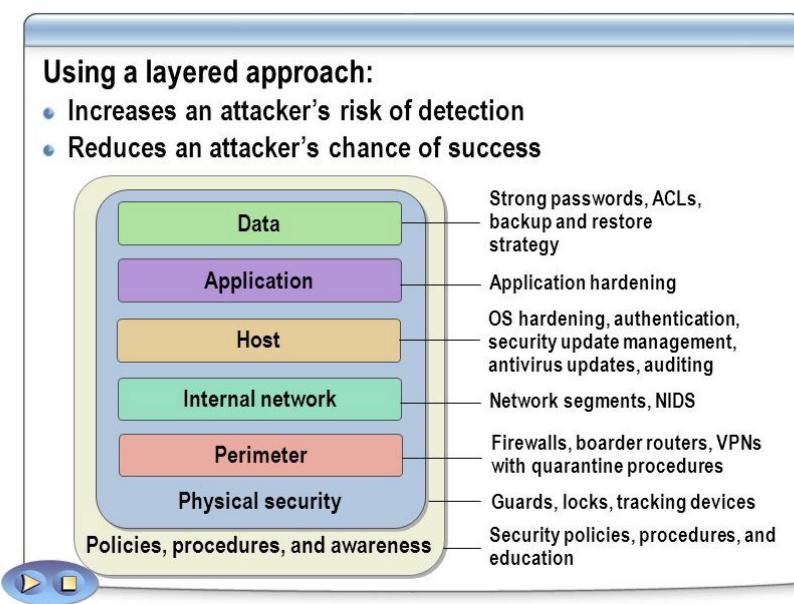
To Secure Our System and to avoid the Security breaches we must take some precautions in our OS.

So we have to ensure of the Shown Security Components:-

- Bios Security
- User Accounts Security
- Data Security
- Antivirus Security
- Firewall



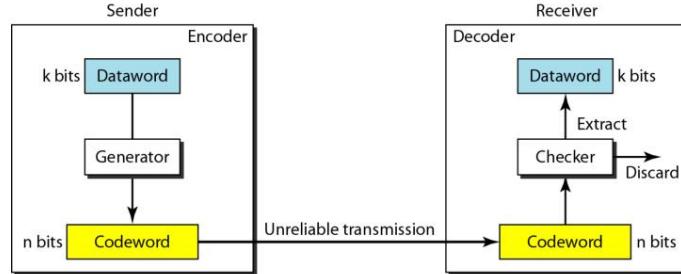
## Understanding Defense-in-Depth



## Error Detection and Correction

## Error Detection

- A receiver can detect a change if the original codeword if
  - The receiver has a list of valid codewords, and
  - The original codeword has changed to an invalid one.



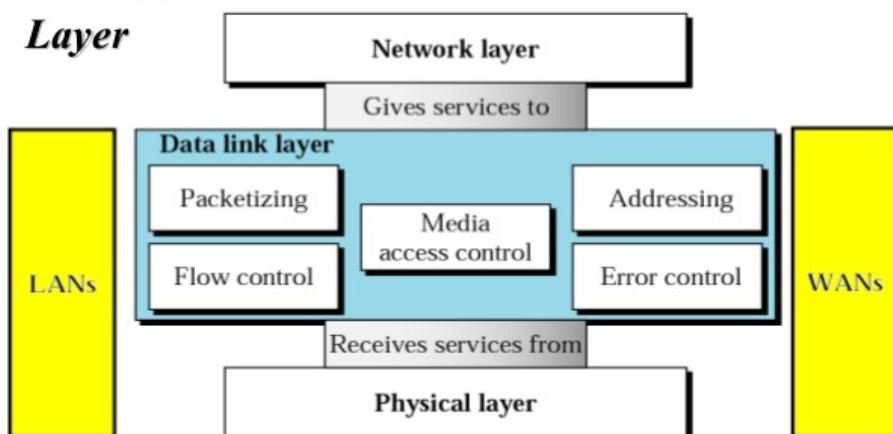
10.12

Er. M.S.Kuthar

## Error Detection and Correction

**Data can be corrupted during transmission.  
Some applications require that errors be detected and corrected.**

### **Data Link Layer**



## Error Detection & Correction

### ■ Error Detection

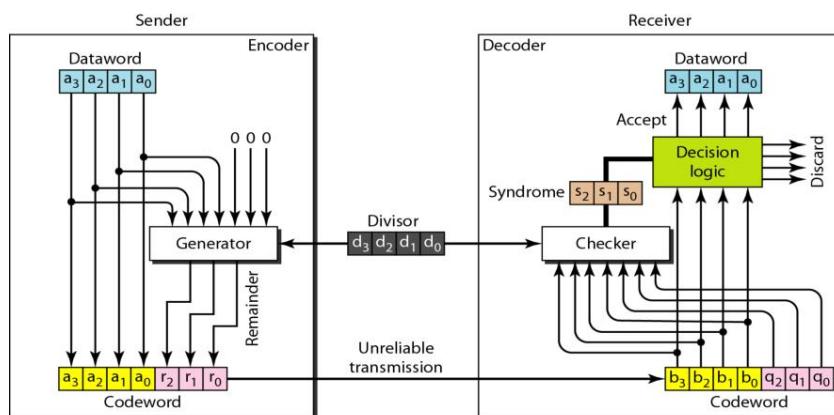
- Check if any error has occurred
- Don't care the number of errors
- Don't care the positions of errors

### ■ Error Correction

- Need to know the number of errors
- Need to know the positions of errors
- More complex
- The number of errors and the size of the message are important factors in error correction

## Error Detection & Correction

Figure 1.6 CRC encoder and decoder



## 1. Resource Allocation

- In case of multiple users accessing same resource
- In case of multiple processes accessing same resource
- Example:
  - Multiple users may be accessing same resource (say printer), then OS allocates the printer based on some algo (like FIFO for ex)
  - CPU Scheduling algos (FIFO, SJF etc)



S

## 2. Accounting

- Statistics related to the resource usage by the users – how much resource each user consumes, what all resources a user uses etc.
- Can be used for billing purposes
- Can also help in reconfiguring system to improve computing services

S

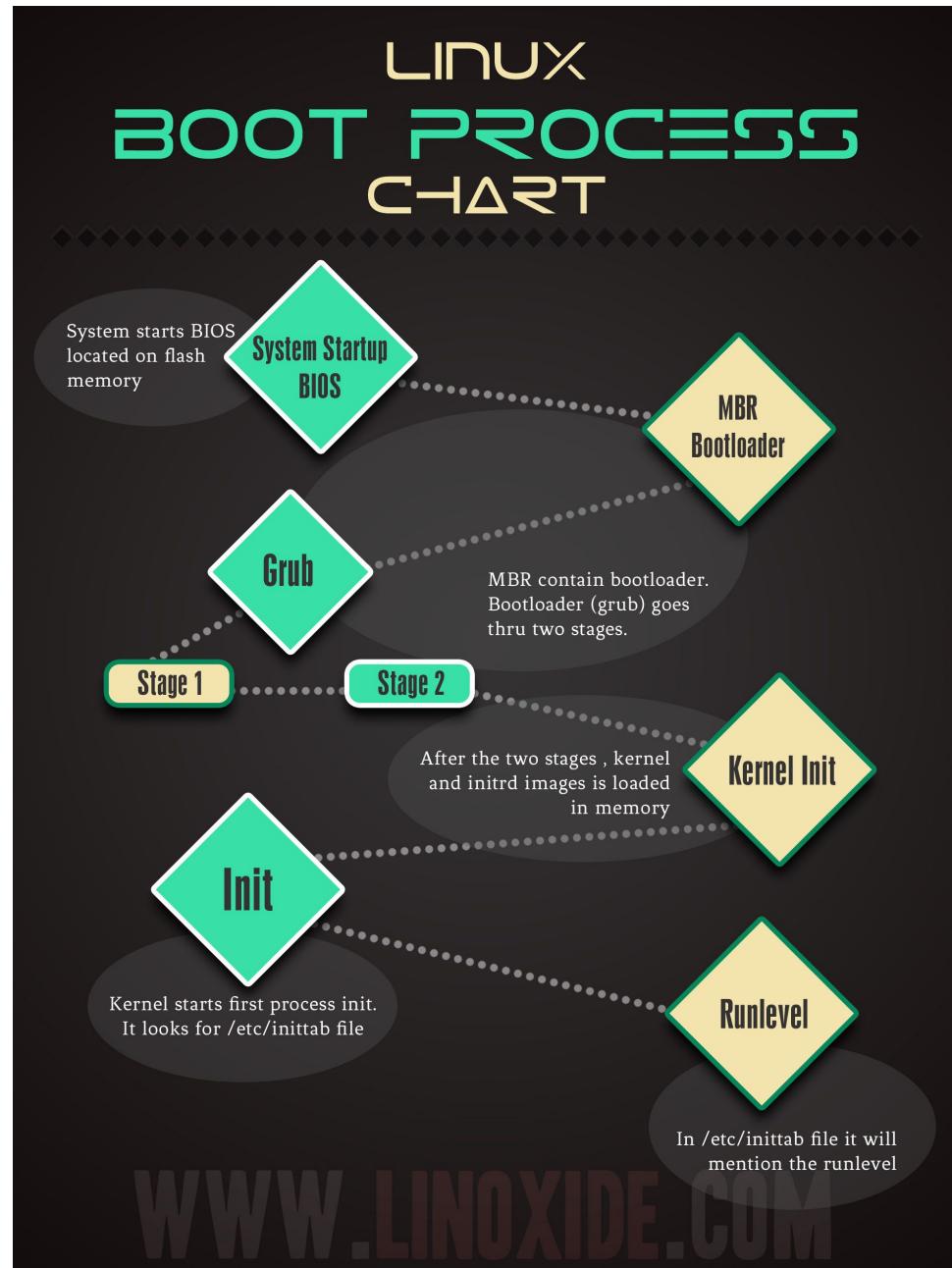
## 3. Protection and Security

- Protection – access to system should be controlled
  - Multiuser systems should not allow an unauthorized user to access content
  - No interference between 2 processes
- Security – from external world
  - To access a system, there must be some kind of authorization (like a password)

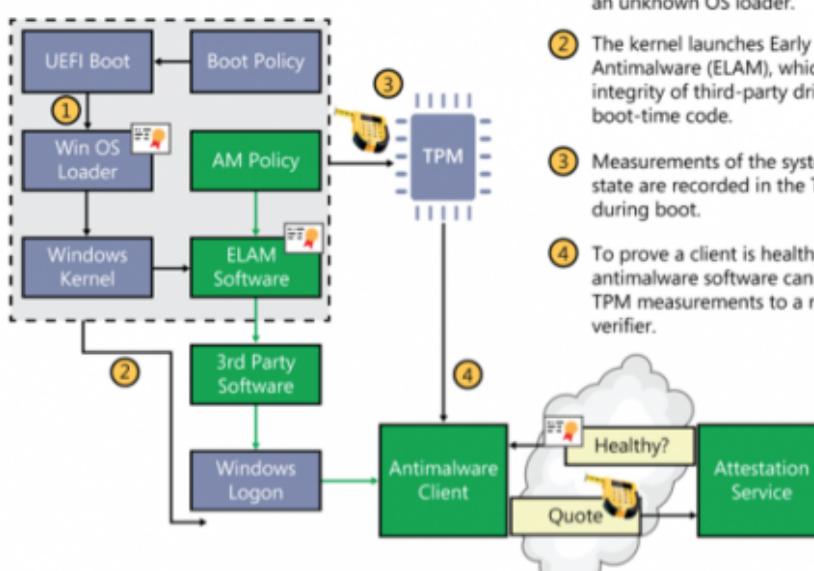


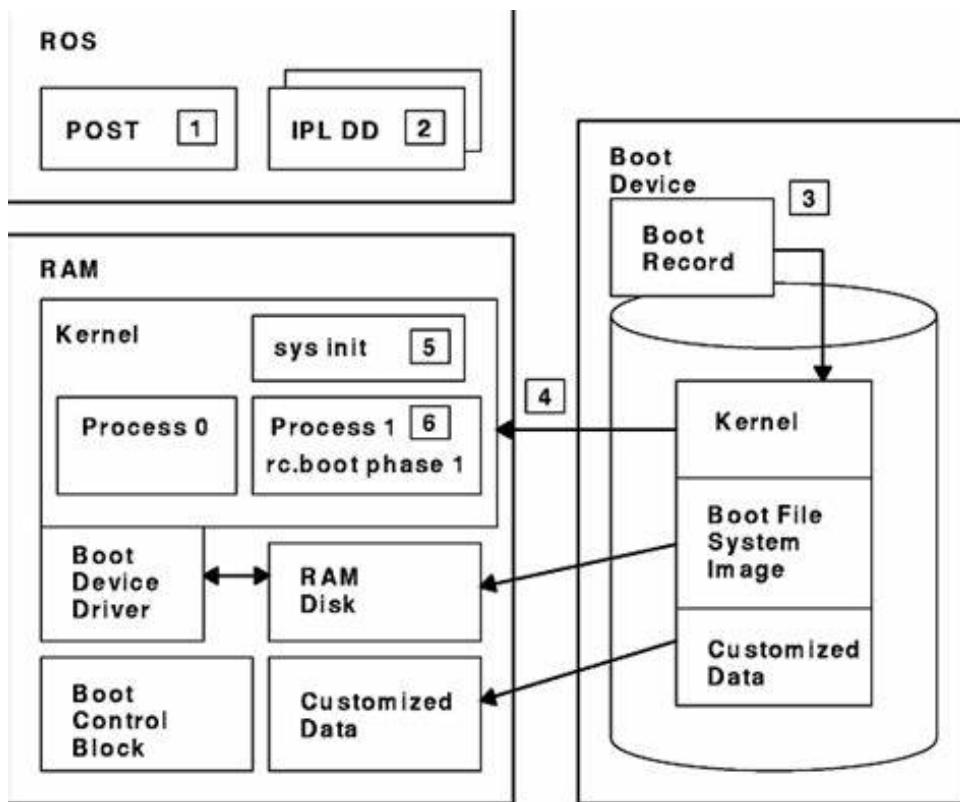
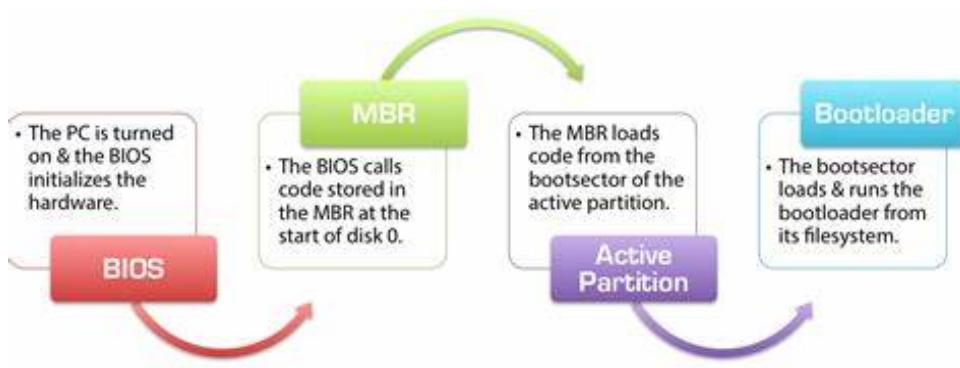
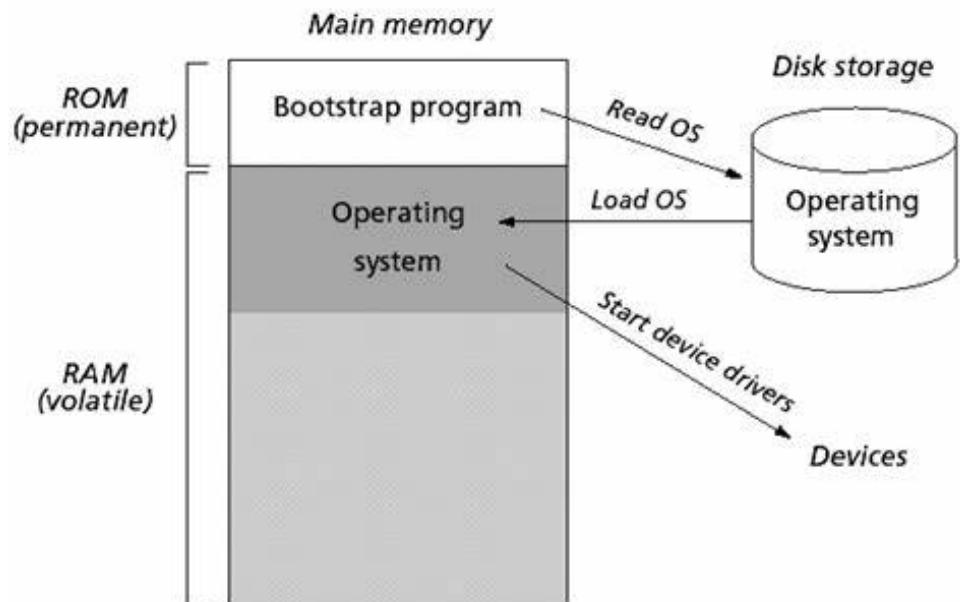
S

# System Boot



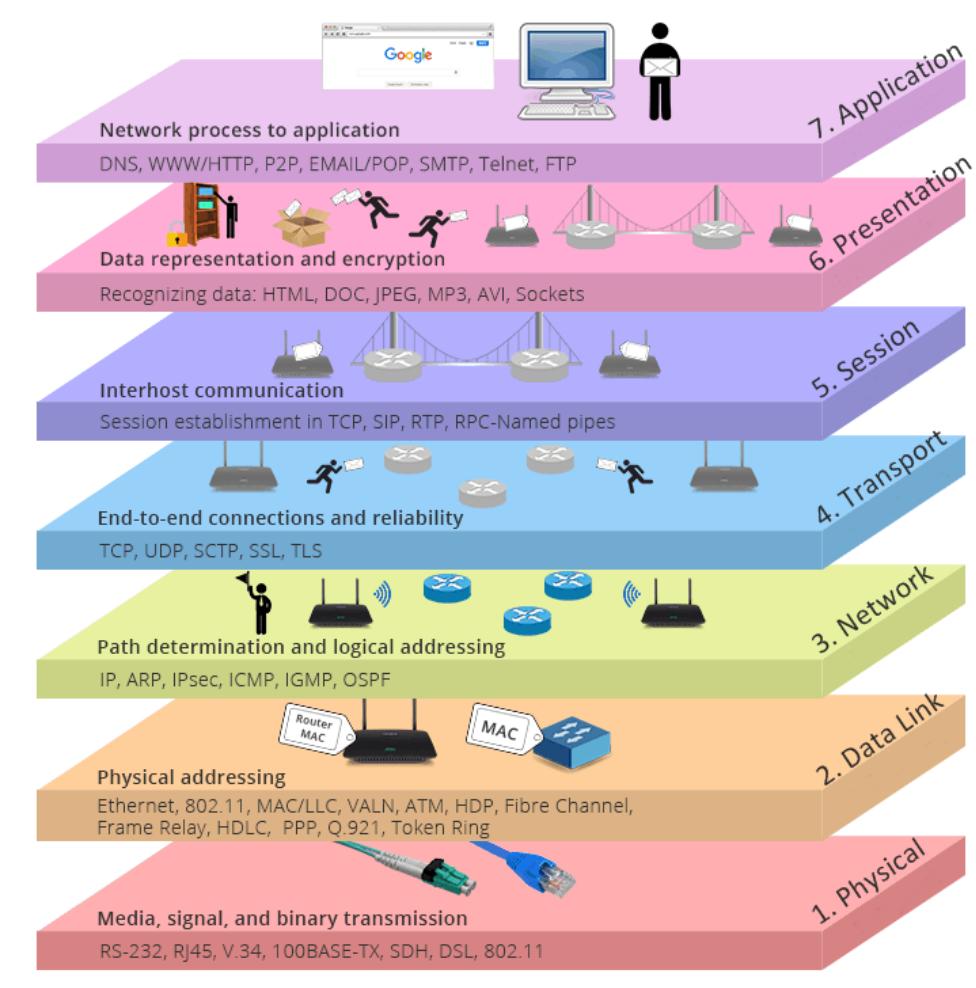
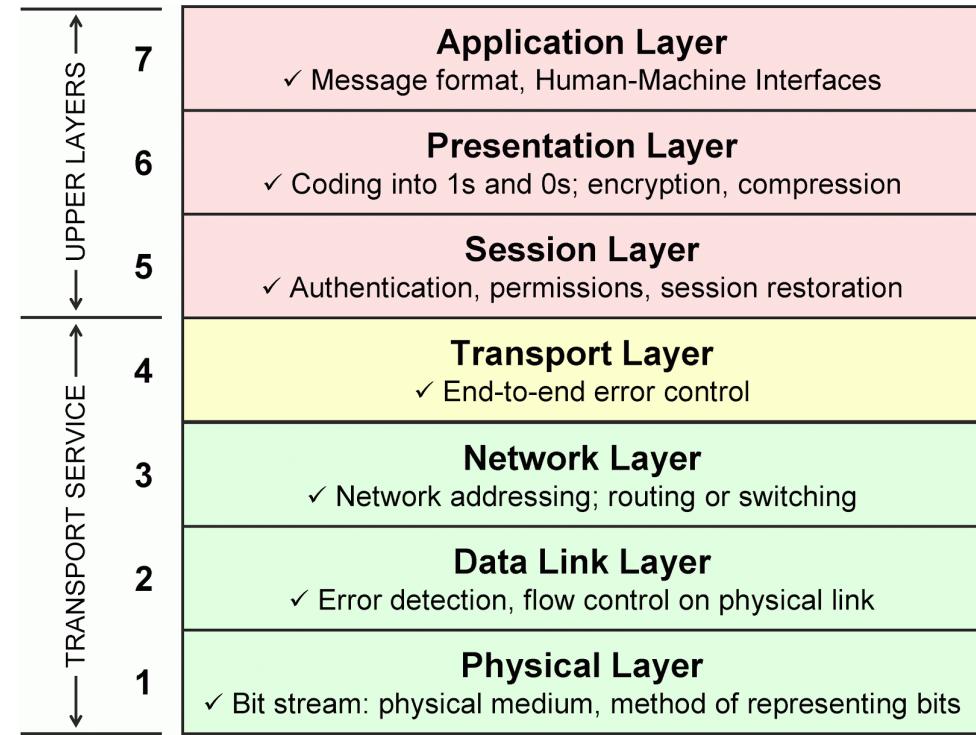
(Windows 8.1 and later)



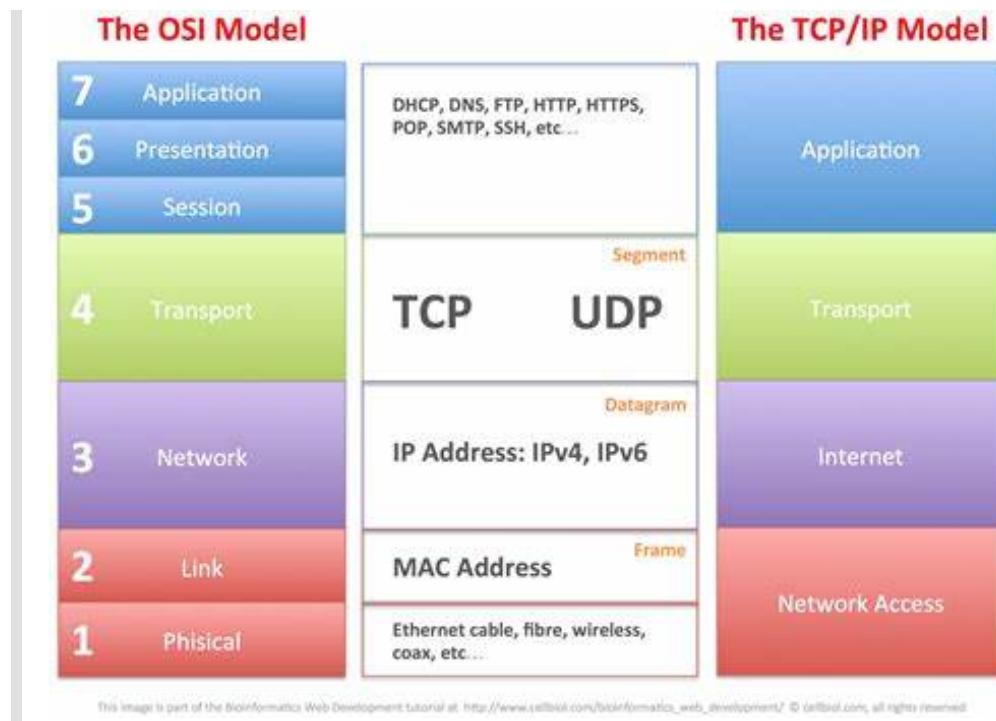


# Network

## Network OSI Model

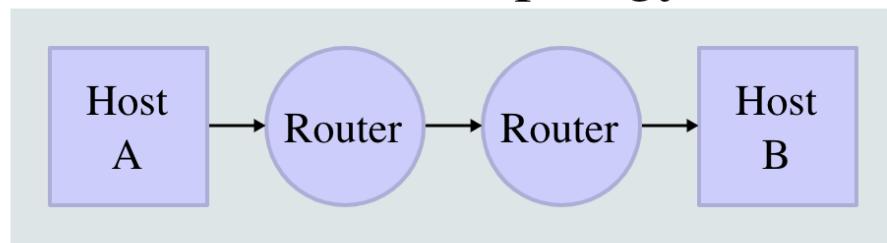


# TCP/IP



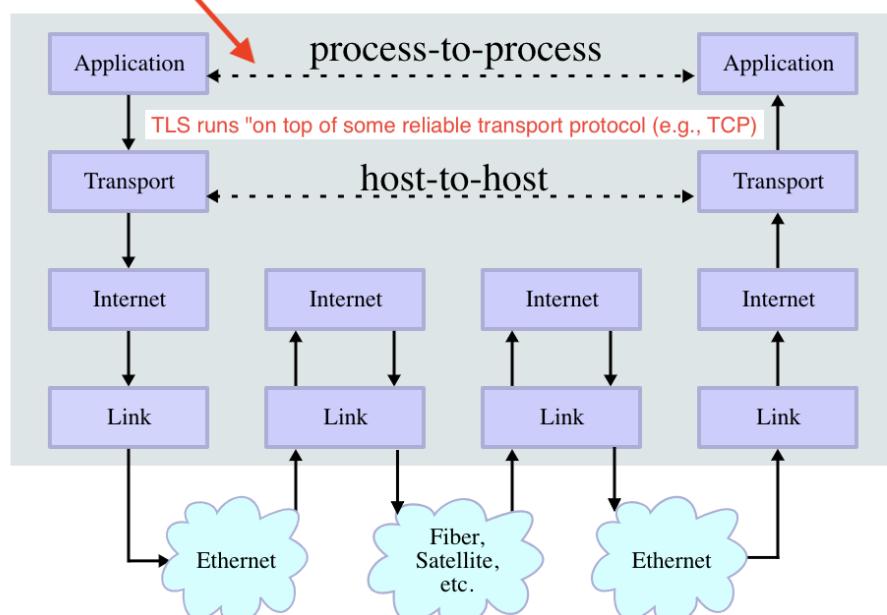
## Data Flow

### Network Topology

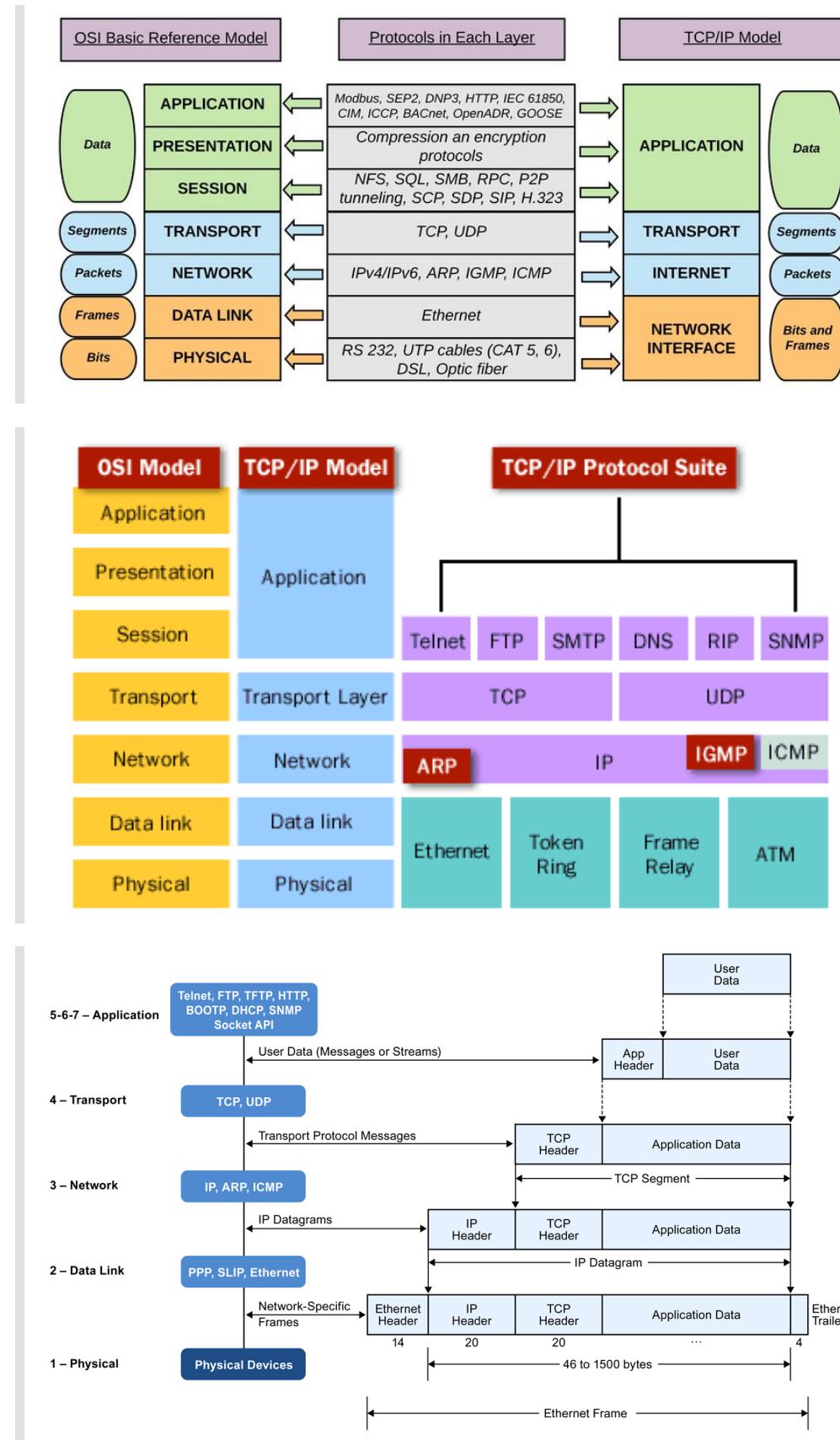


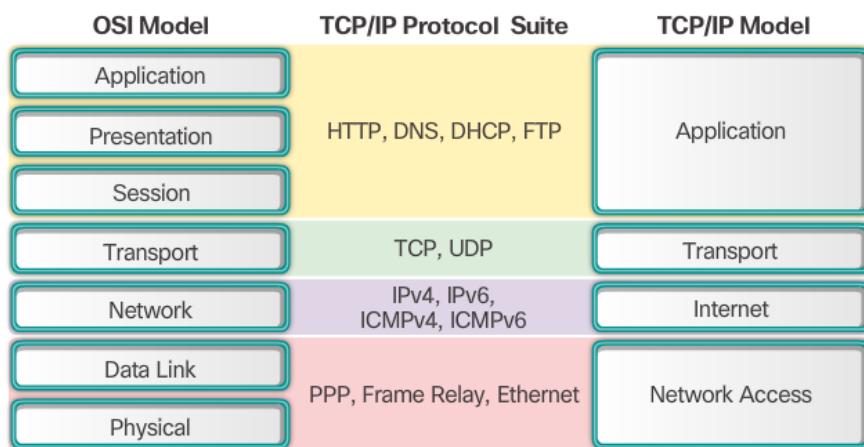
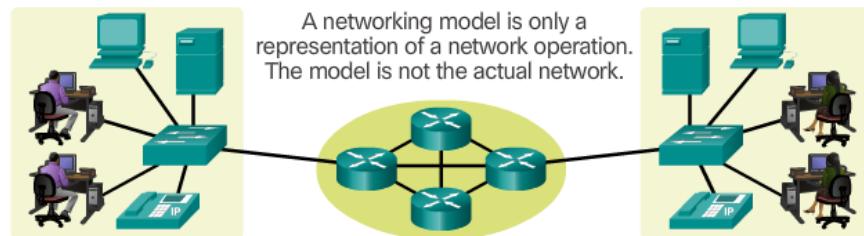
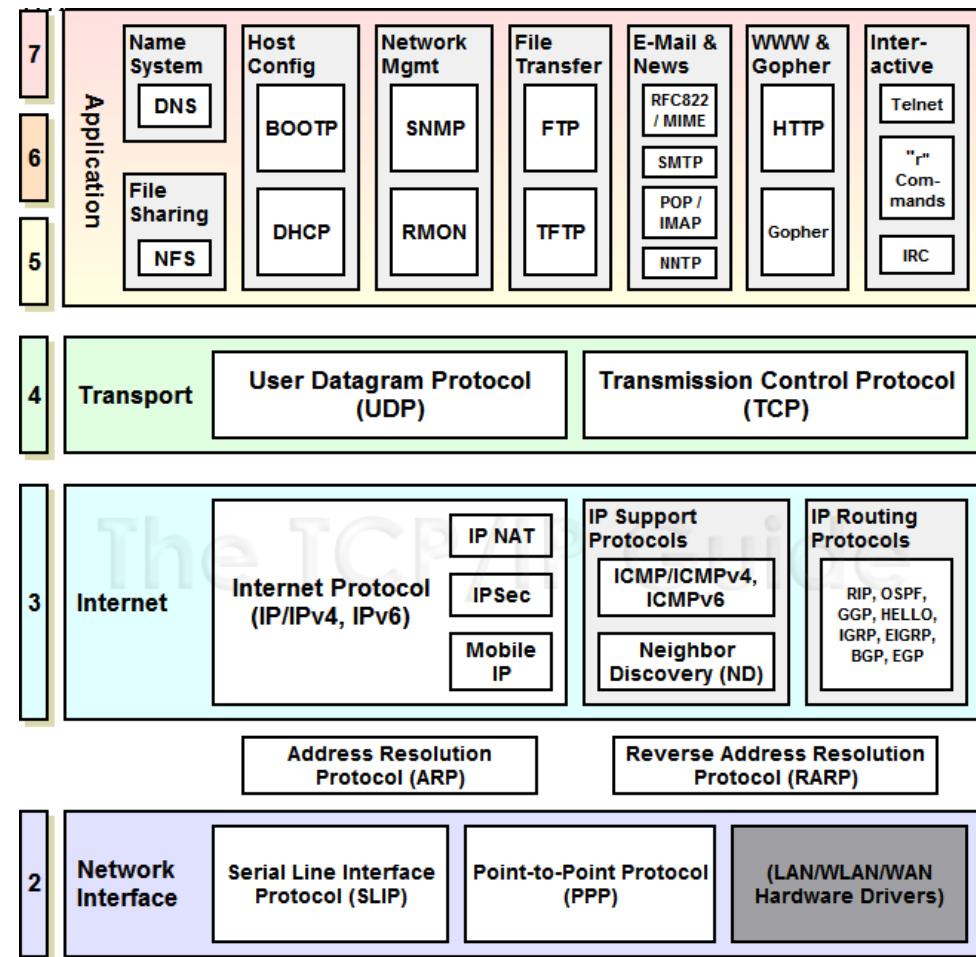
TLS

### Data Flow



# Protocol

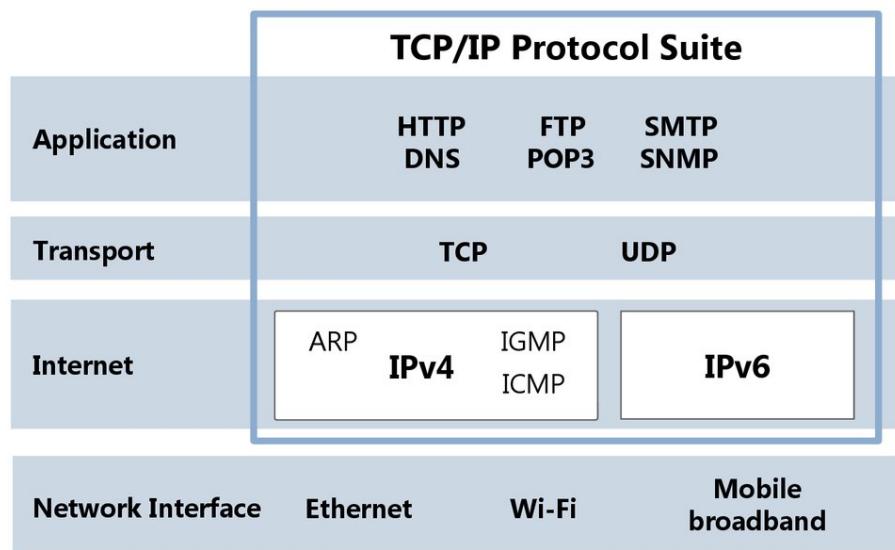




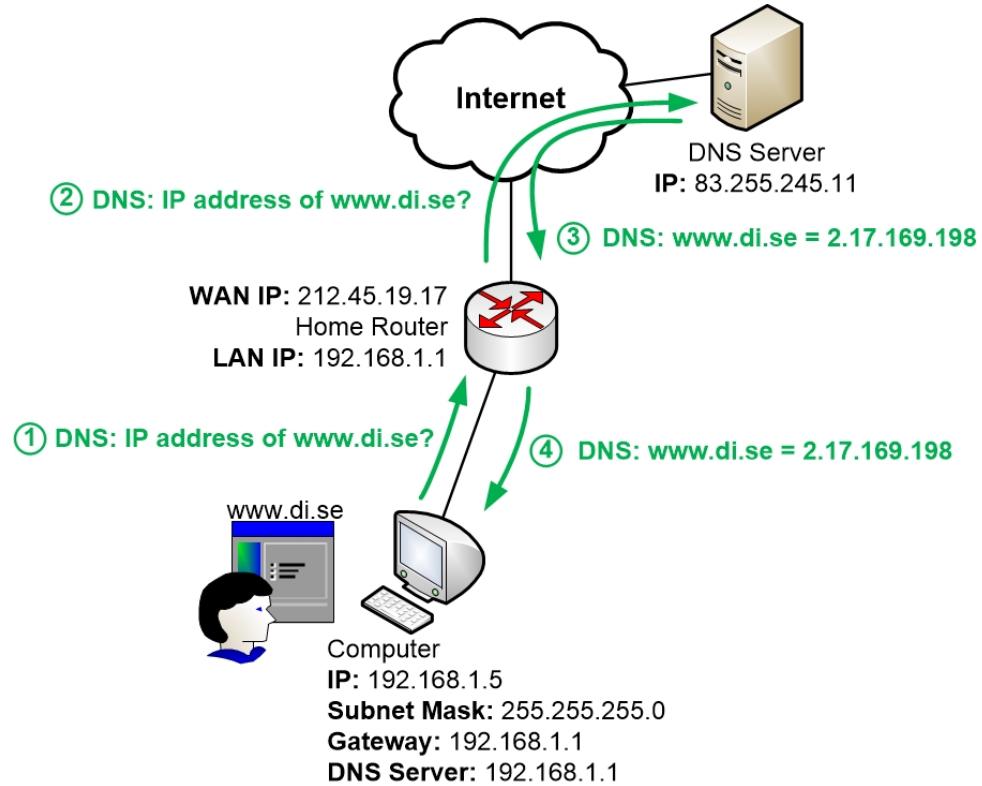
Application	File Transfer	Web Browser	Email	Remote Login	Name Resolution	IP Address
Presentation	FTP TFTP	HTTP	SMTP IMAP POP3	Telnet	DNS	DHCP
Session						
Transport	Transmission Control Protocol TCP				User Datagram Protocol UDP	
Network	Internet Protocol IP				ARP ICMP	
Data Link	Ethernet	Token Ring		FDDI	WAN Protocols	

Computer Network Basic

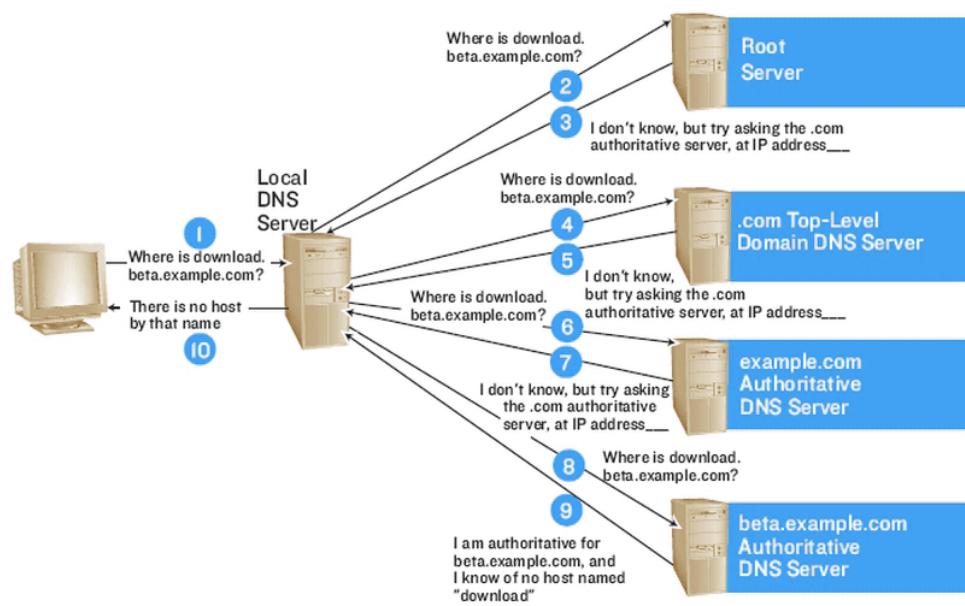
## The TCP/IP Protocol Suite

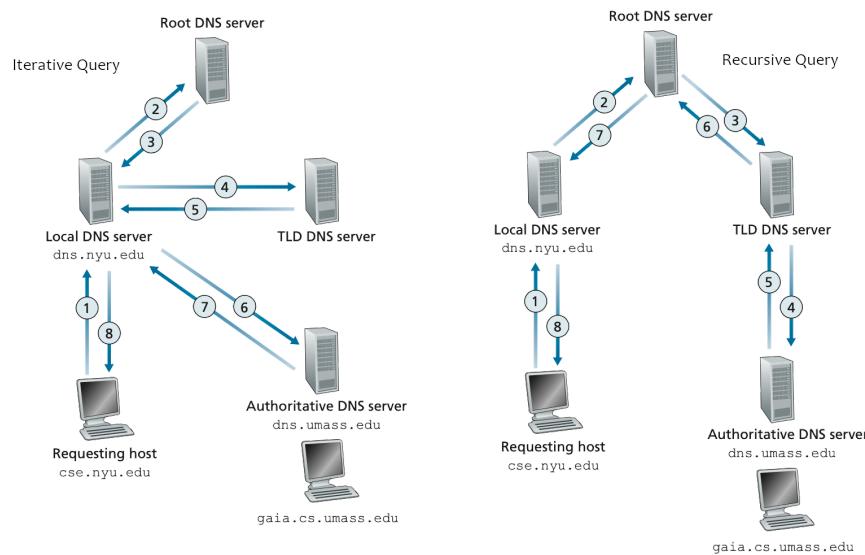
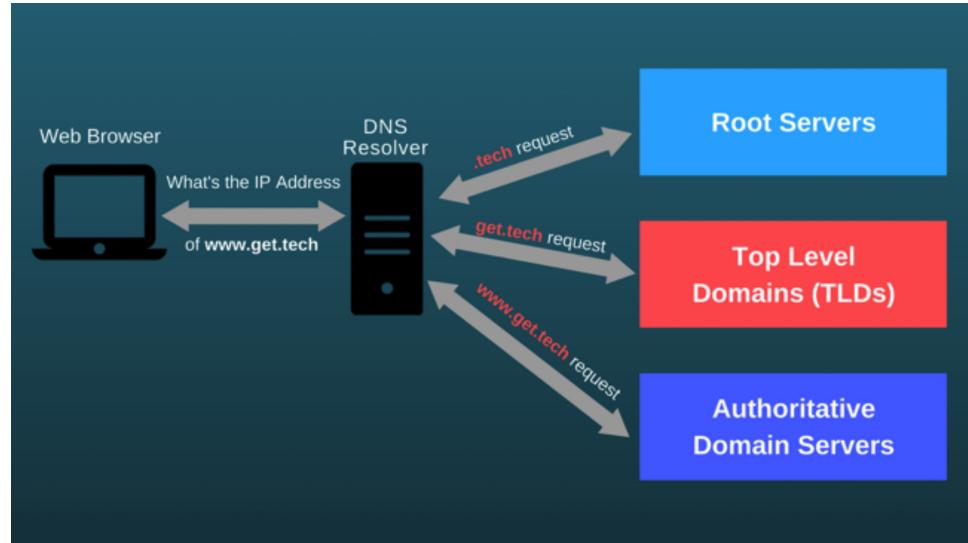


## DNS(Domain Name System)



## HOW DNS WORKS





## Port

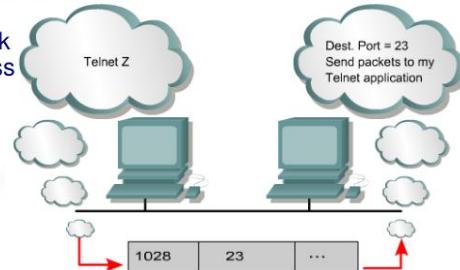
Port #	Application Layer Protocol	Type	Description
20	FTP	TCP	File Transfer Protocol - data
21	FTP	TCP	File Transfer Protocol - control
22	SSH	TCP/UDP	Secure Shell for secure login
23	Telnet	TCP	Unencrypted login
25	SMTP	TCP	Simple Mail Transfer Protocol
53	DNS	TCP/UDP	Domain Name Server
67/68	DHCP	UDP	Dynamic Host
80	HTTP	TCP	HyperText Transfer Protocol
123	NTP	UDP	Network Time Protocol
161,162	SNMP	TCP/UDP	Simple Network Management Protocol
389	LDAP	TCP/UDP	Lightweight Directory Authentication Protocol
443	HTTPS	TCP/UDP	HTTP with Secure Socket Layer

TCP/UDP Port Numbers					
7 Echo	554 RTSP	2745 Bagle.H	6891-6901	Windows Live	
19 Chargen	546-547 DHCPv6	2967 Symantec AV	6970	Quicktime	
20-21 FTP	560 rmonitor	3050 Interbase DB	7212	GhostSurf	
22 SSH/SCP	563 NNTP over SSL	3074 XBOX Live	7648-7649	CU-SeeMe	
23 Telnet	587 SMTP	3124 HTTP Proxy	8000	Internet Radio	
25 SMTP	591 FileMaker	3127 MyDoom	8080	HTTP Proxy	
42 WINS Replication	593 Microsoft DCOM	3128 HTTP Proxy	8086-8087	Kaspersky AV	
43 WHOIS	631 Internet Printing	3222 GLBP	8118	Privoxy	
49 TACACS	636 LDAP over SSL	3260 iSCSI Target	8200	VMware Server	
53 DNS	639 MSDP (PIM)	3306 MySQL	8500	Adobe ColdFusion	
67-68 DHCP/BOOTP	646 LDP (MPLS)	3389 Terminal Server	8767	TeamSpeak	
69 TFTP	691 MS Exchange	3689 iTunes	8866	Bagle.B	
70 Gopher	860 iSCSI	3690 Subversion	9100	HP JetDirect	
79 Finger	873 rsync	3724 World of Warcraft	9101-9103	Bacula	
80 HTTP	902 VMware Server	3784-3785 Ventrilo	9119	Mxit	
88 Kerberos	989-990 FTP over SSL	4333 mSQL	9800	WebDAV	
102 MS Exchange	993 IMAP4 over SSL	4444 Blaster	9898	Dabber	
110 POP3	995 POP3 over SSL	4664 Google Desktop	9988	Rbot/Spybot	
113 Ident	1025 Microsoft RPC	4672 eMule	9999	Urchin	
119 NNTP (Usenet)	1026-1029 Windows Messenger	4899 Radmin	10000	Webmin	
123 NTP	1080 SOCKS Proxy	5000 UPnP	10000	BackupExec	
135 Microsoft RPC	1080 MyDoom	5001 Slingbox	10113-10116	NetIQ	
137-139 NetBIOS	1194 OpenVPN	5001 iperf	11371	OpenPGP	
143 IMAP4	1214 Kazaa	5004-5005 RTP	12035-12036	Second Life	
161-162 SNMP	1241 Nessus	5050 Yahoo! Messenger	12345	NetBus	
177 XDMCP	1311 Dell OpenManage	5060 SIP	13720-13721	NetBackup	
179 BGP	1337 WASTE	5190 AIM/ICQ	14567	Battlefield	
201 AppleTalk	1433-1434 Microsoft SQL	5222-5223 XMPP/Jabber	15118	Dipnet/Oddbob	
264 BGMP	1512 WINS	5432 PostgreSQL	19226	AdminSecure	
318 TSP	1589 Cisco VQP	5500 VNC Server	19638	Ensim	
381-383 HP Openview	1701 L2TP	5554 Sasser	20000	Usermin	
389 LDAP	1723 MS PPTP	5631-5632 pcAnywhere	24800	Synergy	
411-412 Direct Connect	1725 Steam	5800 VNC over HTTP	25999	Xfire	
443 HTTP over SSL	1741 CiscoWorks 2000	5900+ VNC Server	27015	Half-Life	
445 Microsoft DS	1755 MS Media Server	6000-6001 X11	27374	Sub7	
464 Kerberos	1812-1813 RADIUS	6112 Battle.net	28960	Call of Duty	
465 SMTP over SSL	1863 MSN	6129 DameWare	31337	Back Orifice	
497 Retrospect	1985 Cisco HSRP	6257 WinMX	33434+	traceroute	
500 ISAKMP	2000 Cisco SCCP	6346-6347 Gnutella	Legend		
512 rexec	2002 Cisco ACS	6500 GameSpy Arcade			
513 rlogin	2049 NFS	6566 SANE			
514 syslog	2082-2083 cPanel	6588 AnalogX			
515 LPD/LPR	2100 Oracle XDB	6665-6669 IRC			
520 RIP	2222 DirectAdmin	6679/6697 IRC over SSL			
521 RIPng (IPv6)	2302 Halo	6699 Napster			
540 UUCP	2483-2484 Oracle DB	6881-6999 BitTorrent			

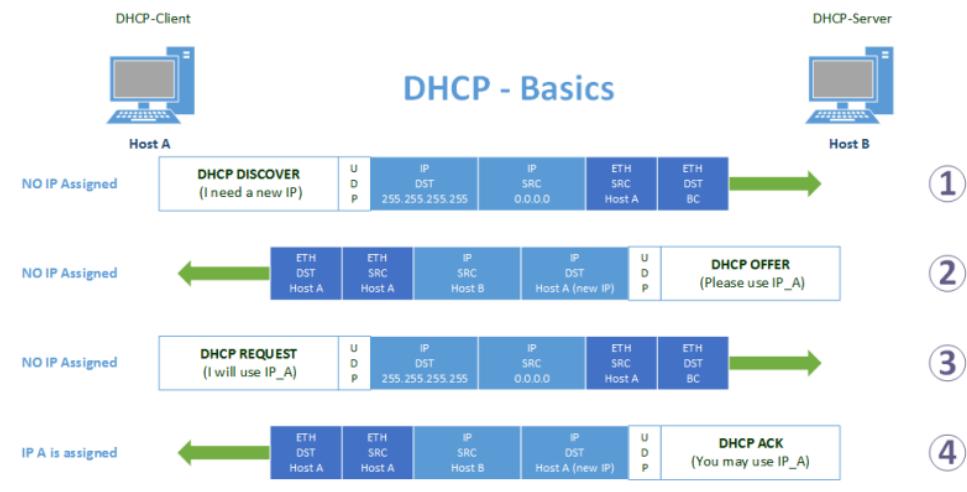
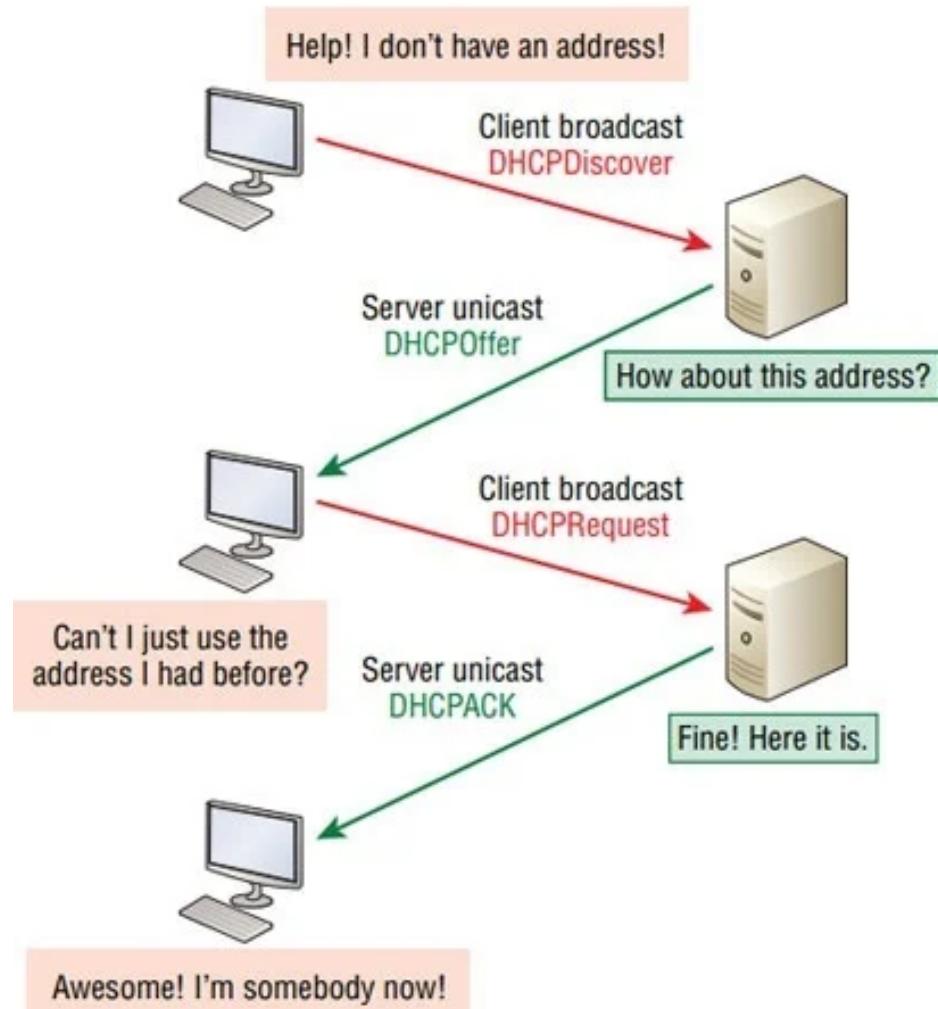
IANA port assignments published at <http://www.iana.org/assignments/port-numbers>

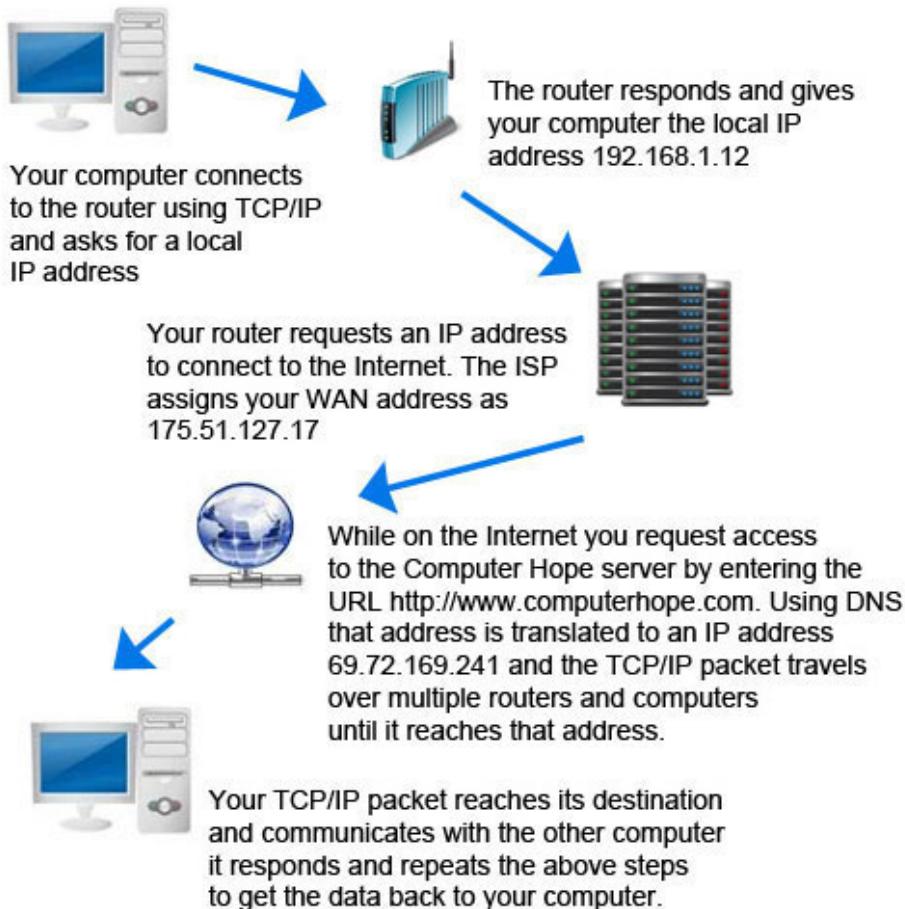
## Transport Layer Ports

- Port numbers are used to keep track of different **conversations** that cross the network at the same time.
- Port numbers identify which upper layer service is needed, and are needed when a host communicates with a server that uses multiple services.
- Both TCP and UDP use port numbers to pass to the upper layers.
- Port numbers have the following **ranges**:
  - 0-255 used for public applications, 0-1023 also called **well-known ports**, regulated by IANA (Internet assigned numbers authority).
  - Numbers from 255-1023 are assigned to marketable applications
  - 1024 through 49151 Registered Ports, not regulated.
  - 49152 through 65535 are Dynamic and/or Private Ports .



## DHCP





ComputerHope.com

## APIPA

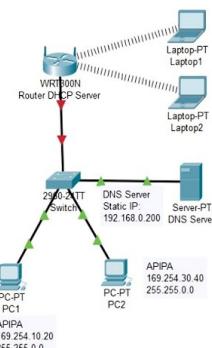
**MANA NETWORK**  
All about EDUCATION

### APIPA Class B Private IP v4 Address

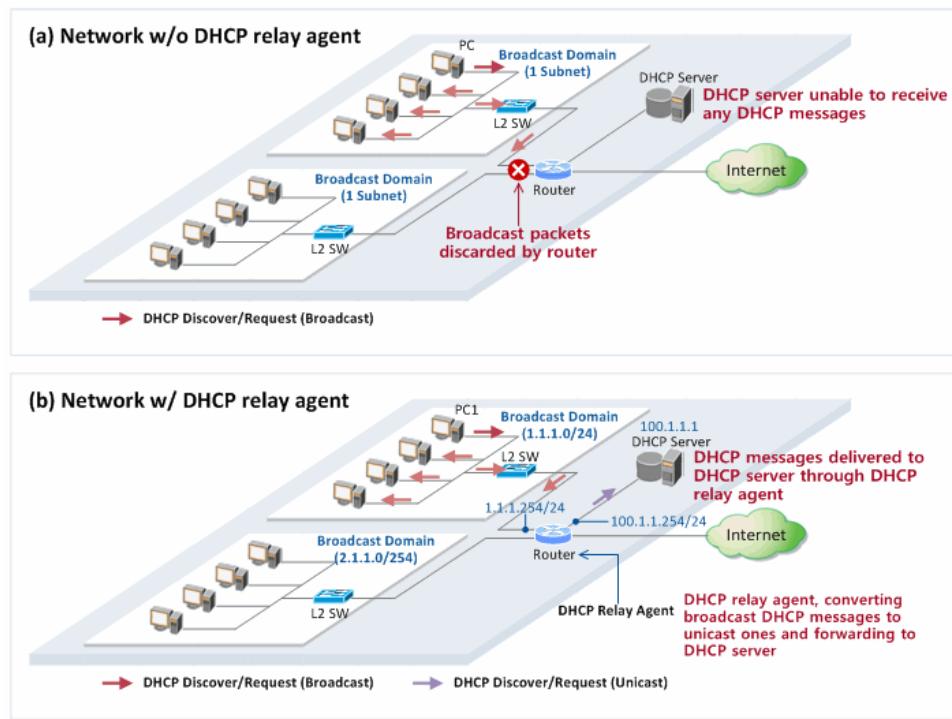
#### ❖ Automatic Private IP Address

169.254. x. x

- ❑ When connection between **DHCP Server & N/W device (Switch)** goes **DOWN**, **APIPA** addresses are **AUTOMATICALLY** created on **END User Devices** like Desktops, PCs, Laptops, Printers etc.
- ❑ **END User Devices** who have **APIPA** addresses can **ONLY** communicate **INSIDE** the own **LOCAL N/W**
- ❑ **APIPA** addresses **DO NOT** go out of their **OWN N/W**
- ❑ **APIPA** addresses are **NOT ROUTABLE**
- ❑ If **APIPA** addresses are seen on **END Devices** than this is a **INTERNAL N/W** problem
- ❑ Check the **MEDIA or CABLE** between **DHCP server (Router) & N/W device (Switch)** inside **LOCAL N/W**

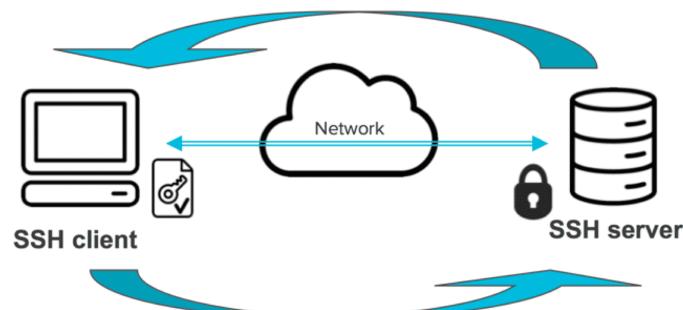


## DHCP Relay = IP Helper

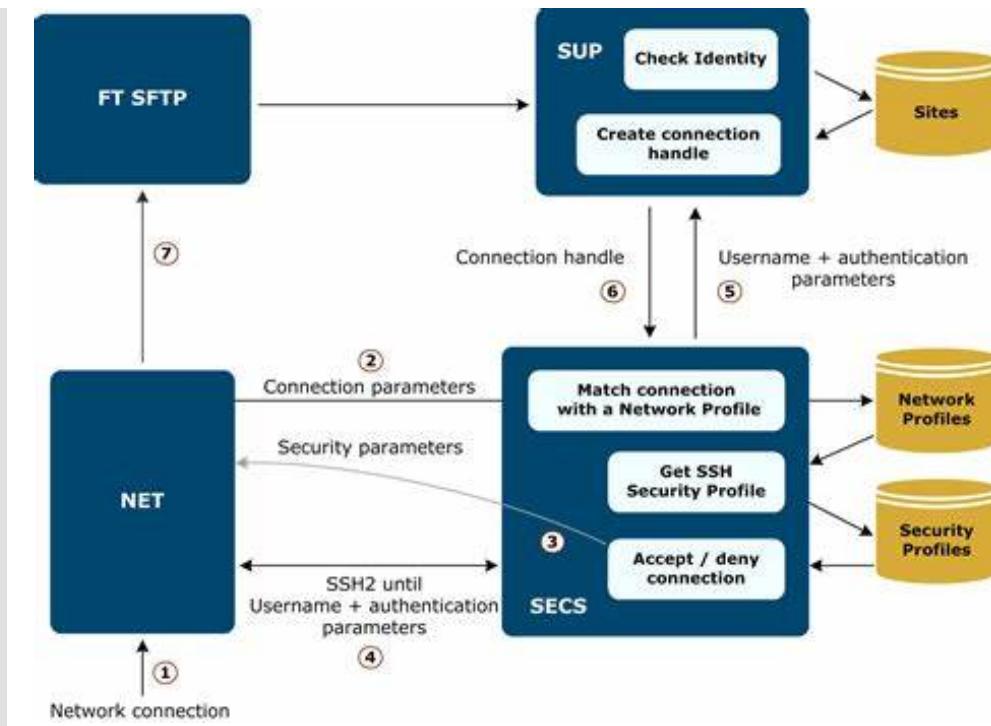


## SSH(Secure Shell)

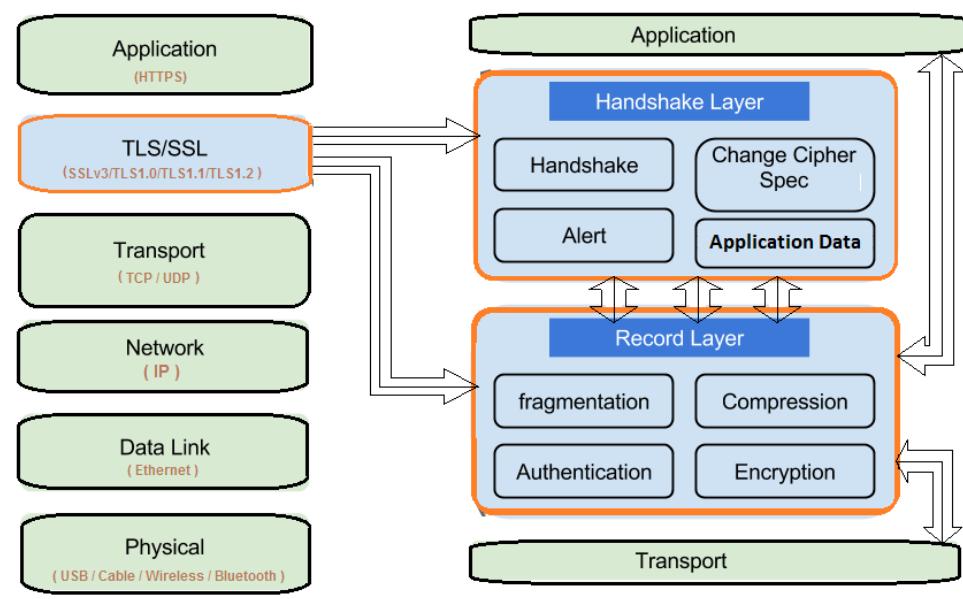
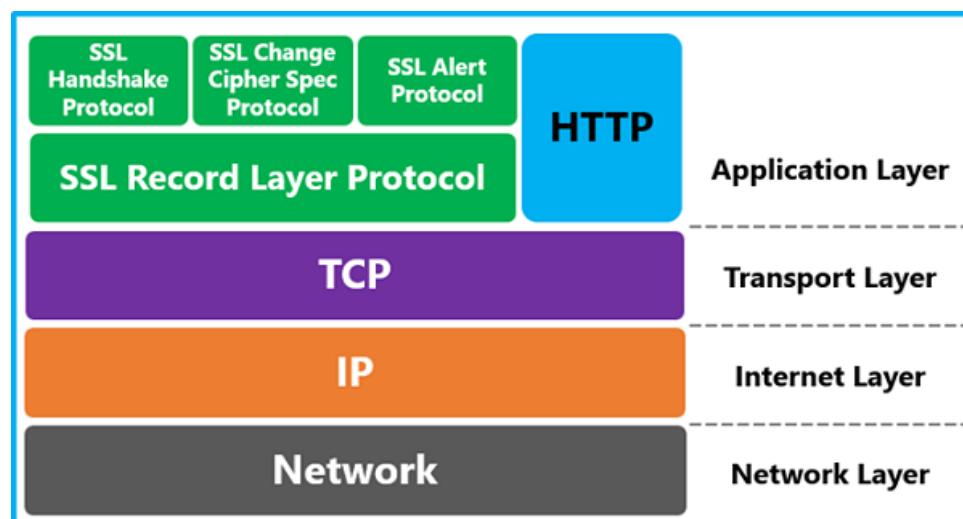
- 1) **Server authentication:**  
Server proves its identity to the client

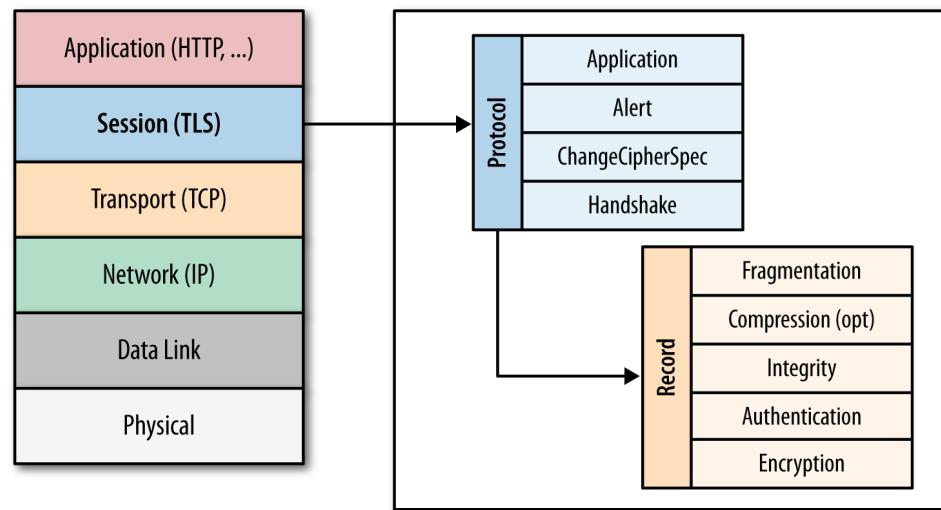


- 2) **User authentication:**  
Client proves user's identity to the server

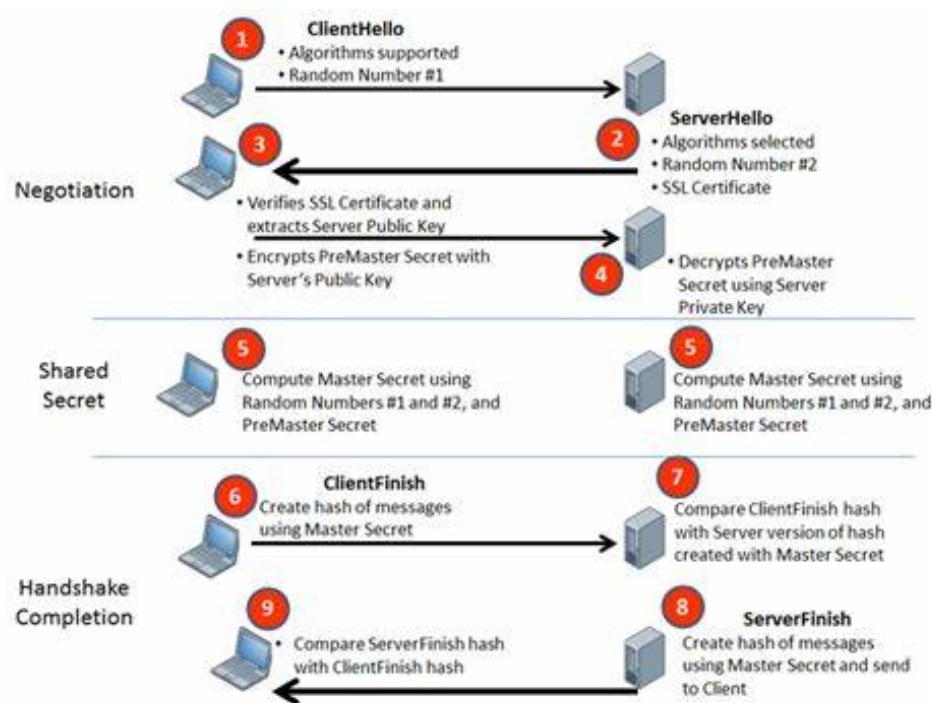
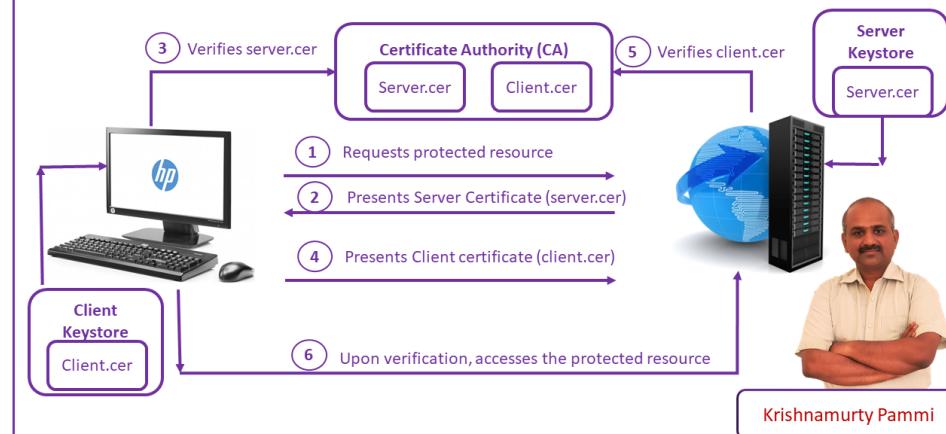


## SSL(Secure Sockets Layers)

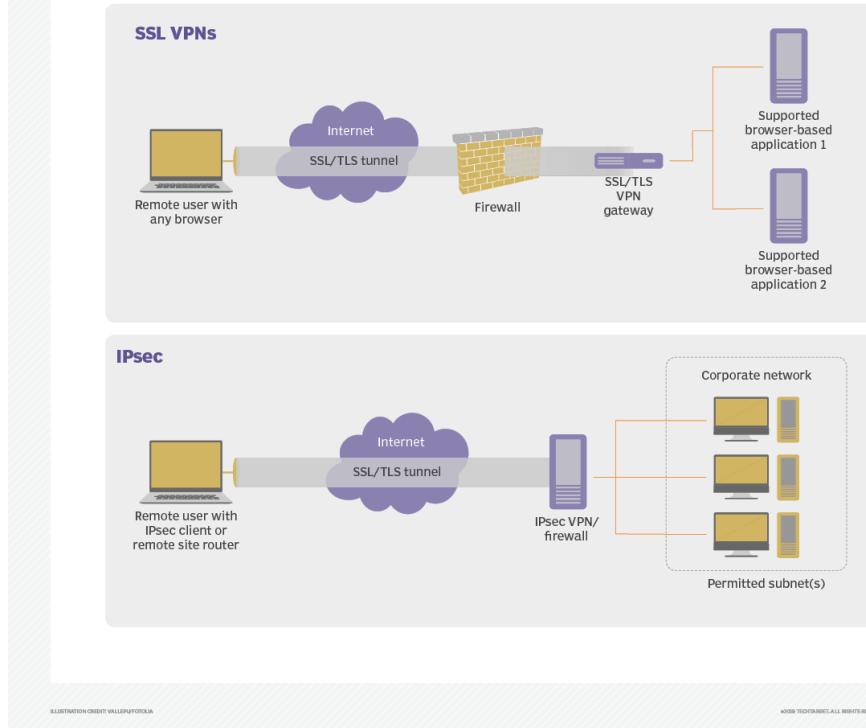




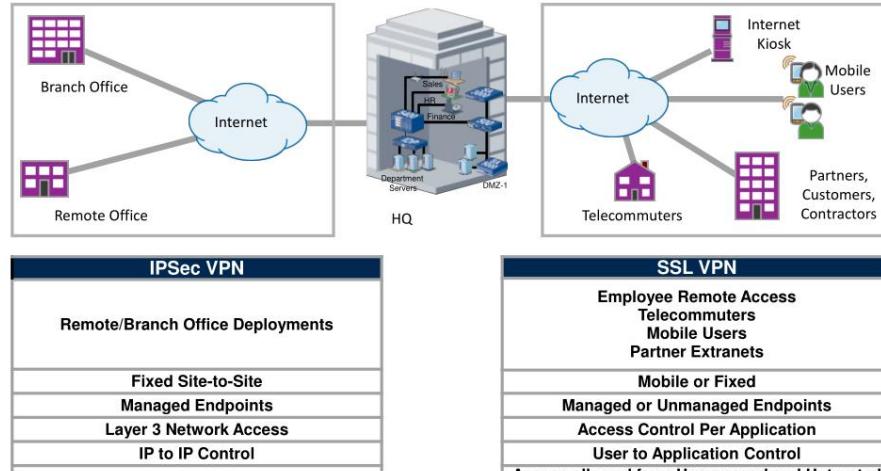
## Web Security: Secure Socket Layer (SSL)



## SSL/TLS VPNs vs. IPsec VPNs

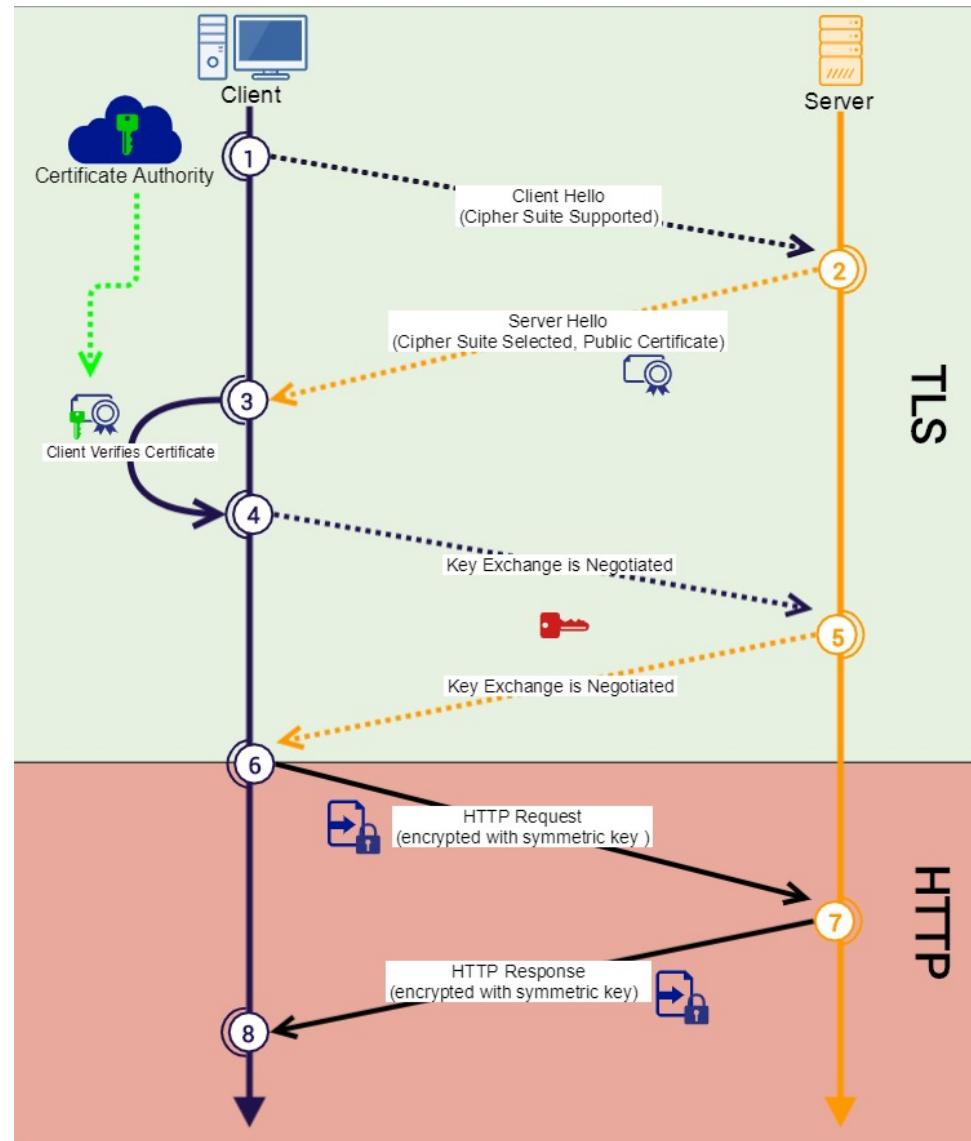


## IPSec VPN vs. SSL VPN



**ASU** Ira A. Fulton  
Schools of Engineering  
ARIZONA STATE UNIVERSITY

TLS



## SSL vs TSL

**S S L****V E R S U S****T L S****SSL**

Standard security protocol for establishing an encrypted link between a web server and a browser

Introduced in the year 1994 by Netscape Communications

Stands for Secure Socket Layer

Not as secure as TSL

Comparatively less complex

**TLS**

Protocol that provides communication security between client/server applications that communicate with each other over the interne

Introduced in 1999 by Internet Engineering Task Force (IETF)

Stands for Transport Layer Security

More secure

A complex protocol

Visit [www.PEDIAA.com](http://www.PEDIAA.com)

**TCP vs UDP****TCP Vs UDP Communication**

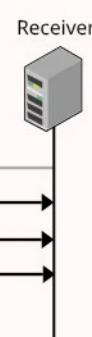
Sender

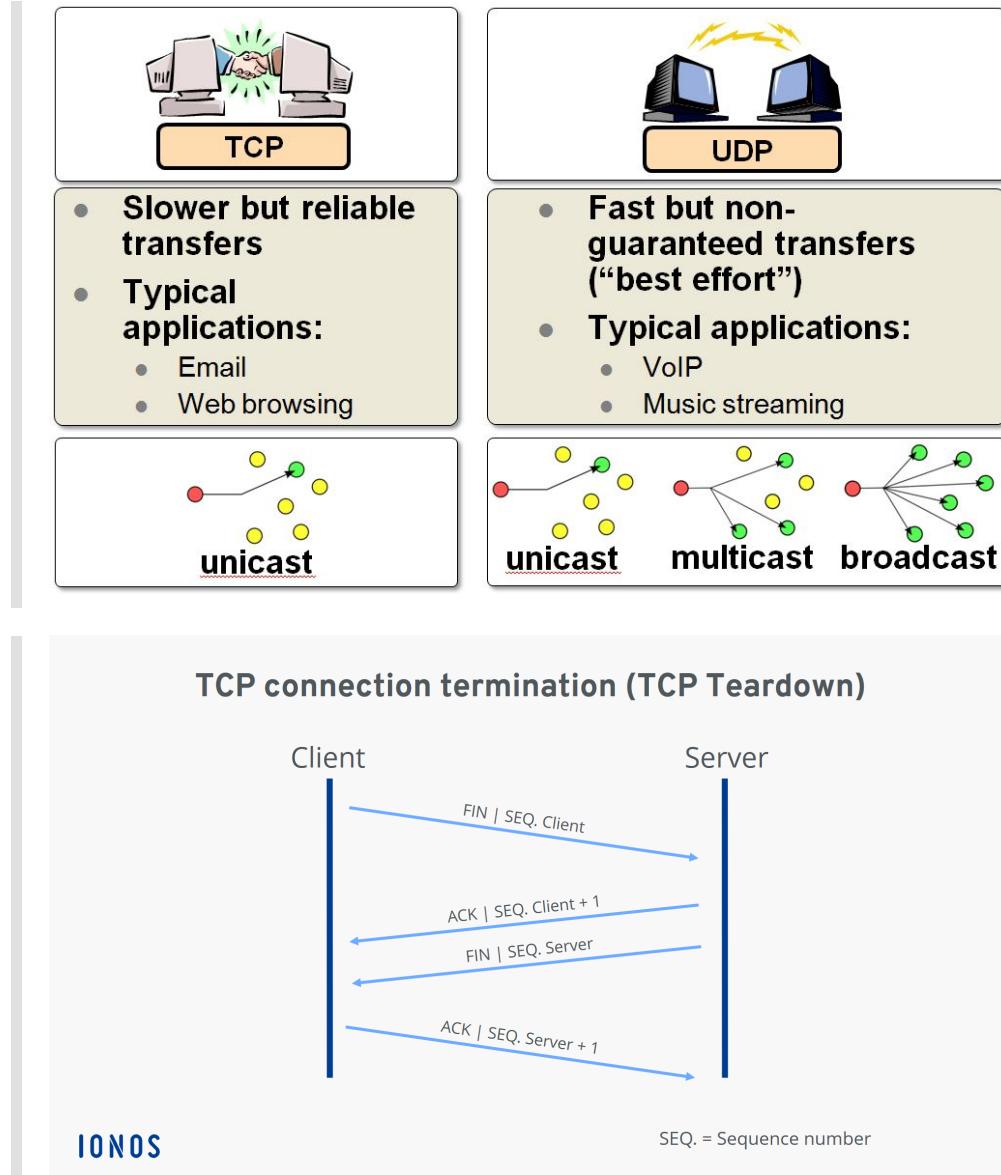
**TCP**

Receiver

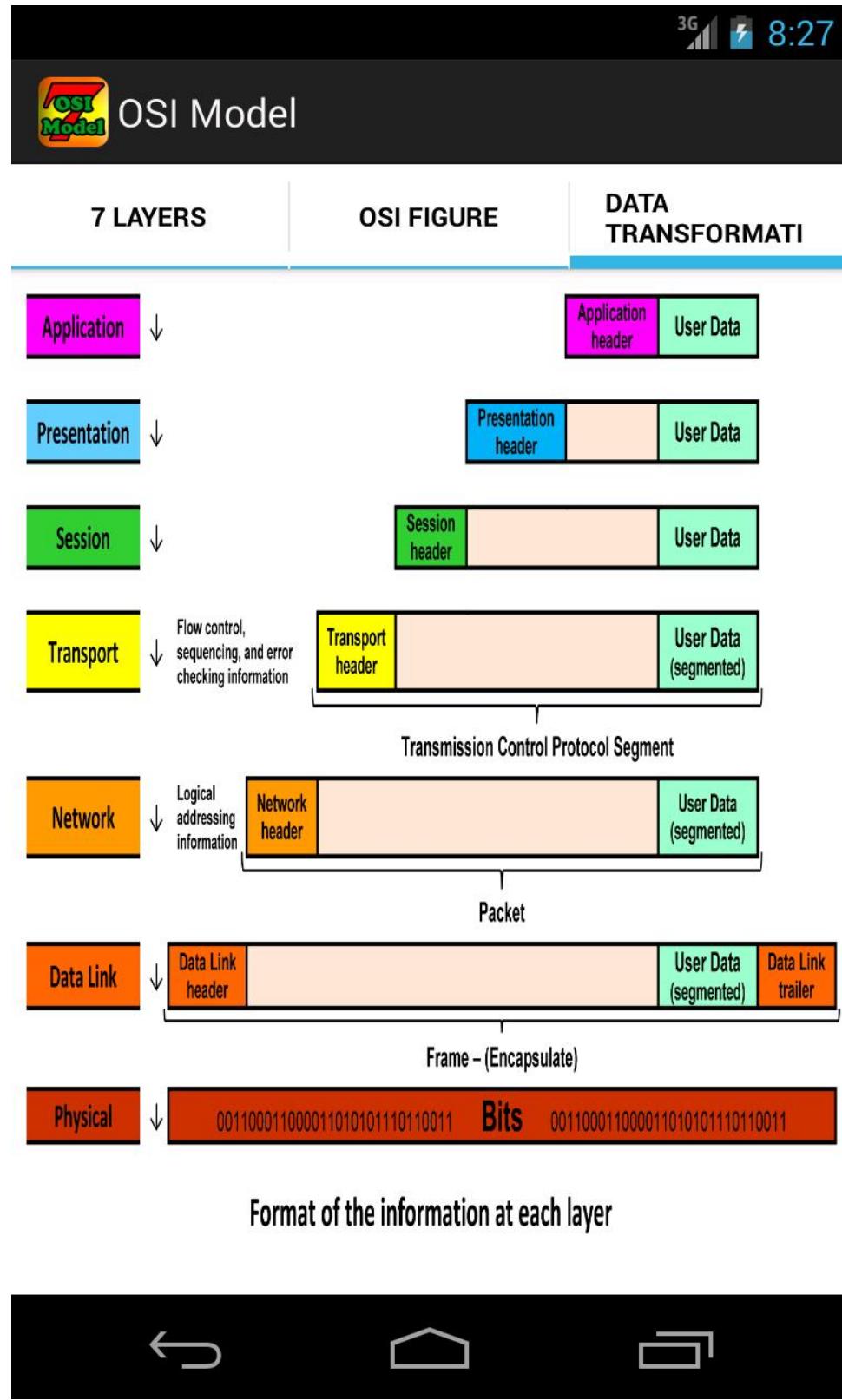


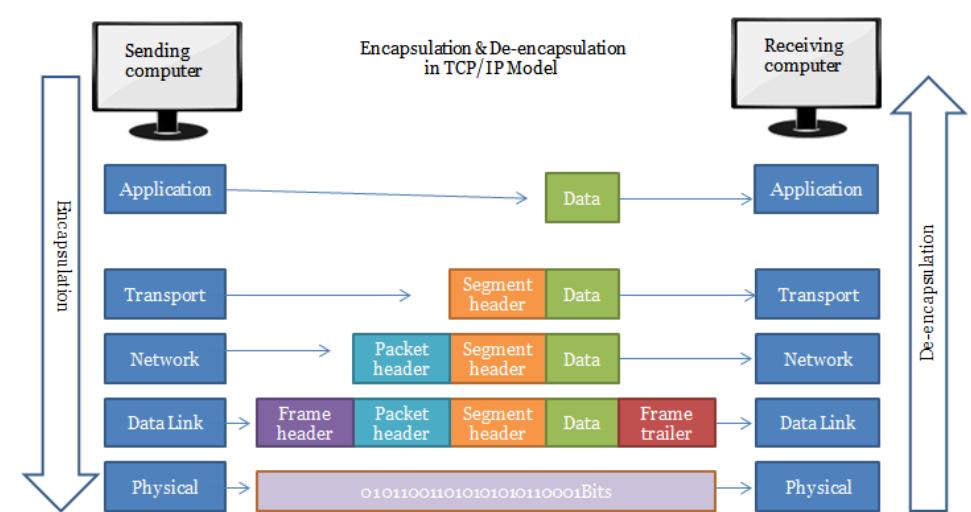
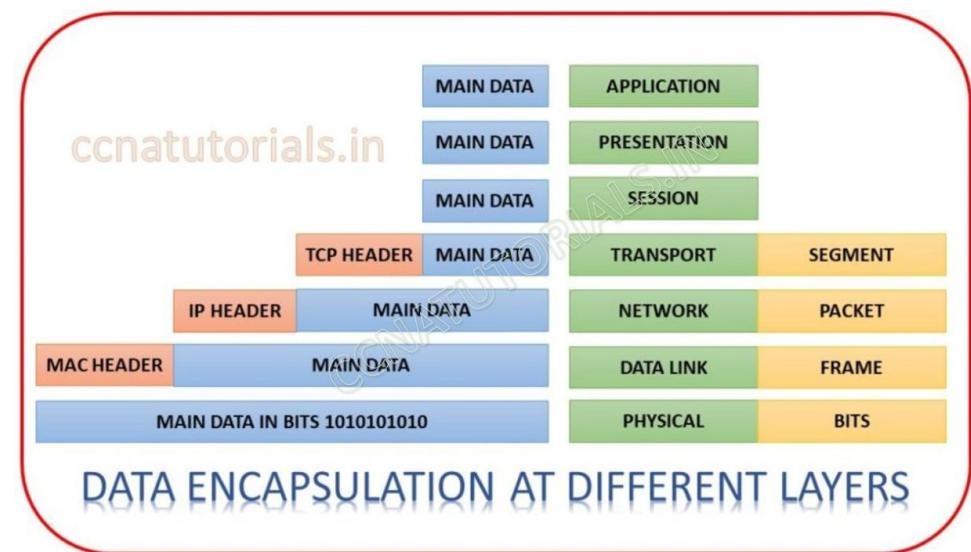
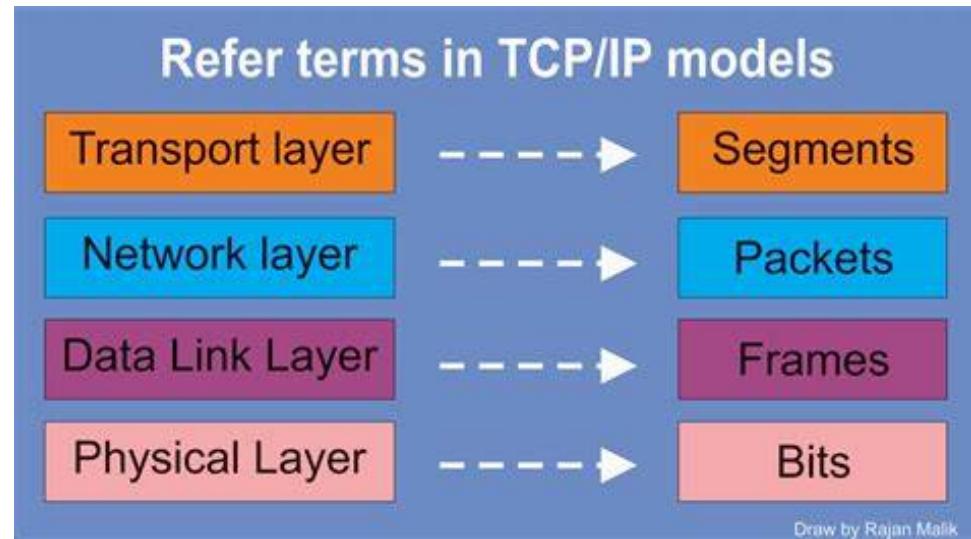
Sender

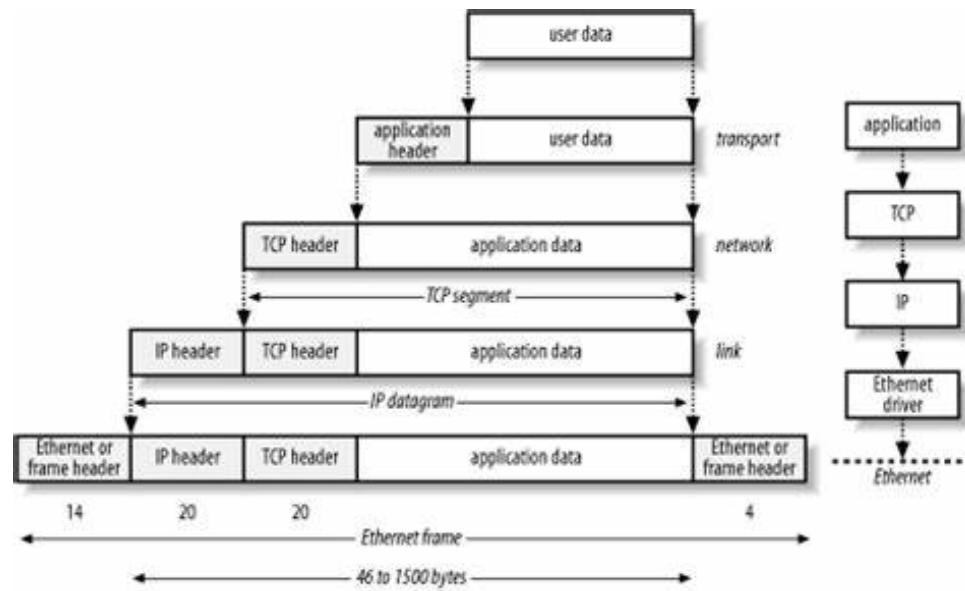
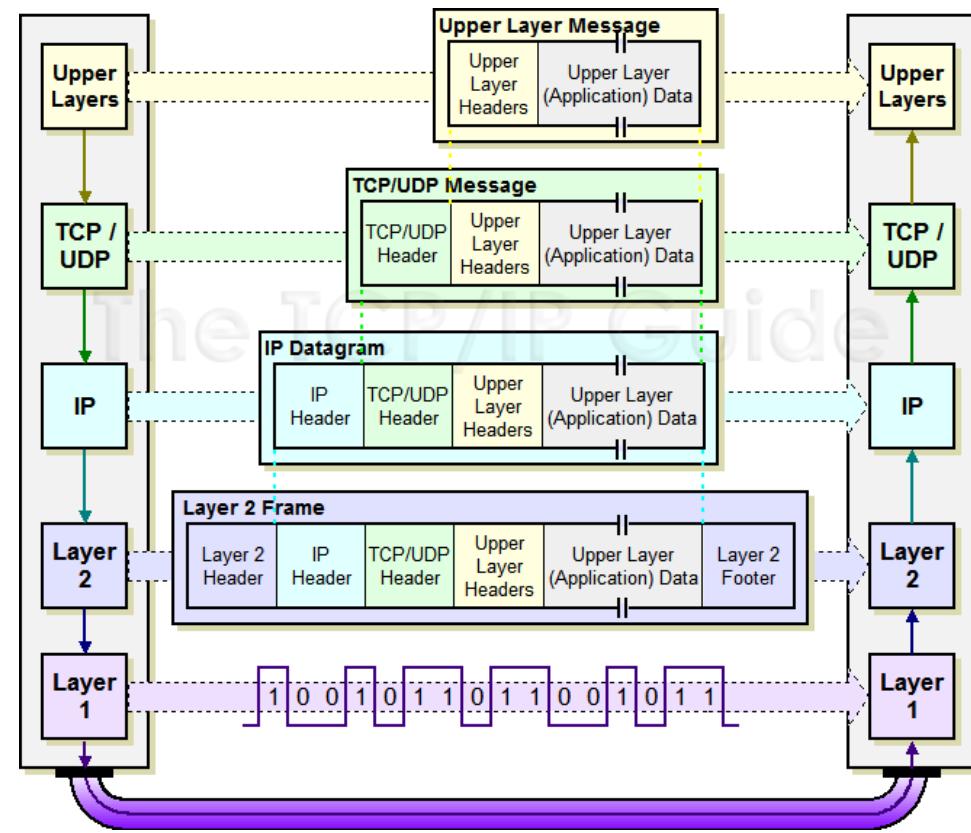
**UDP**

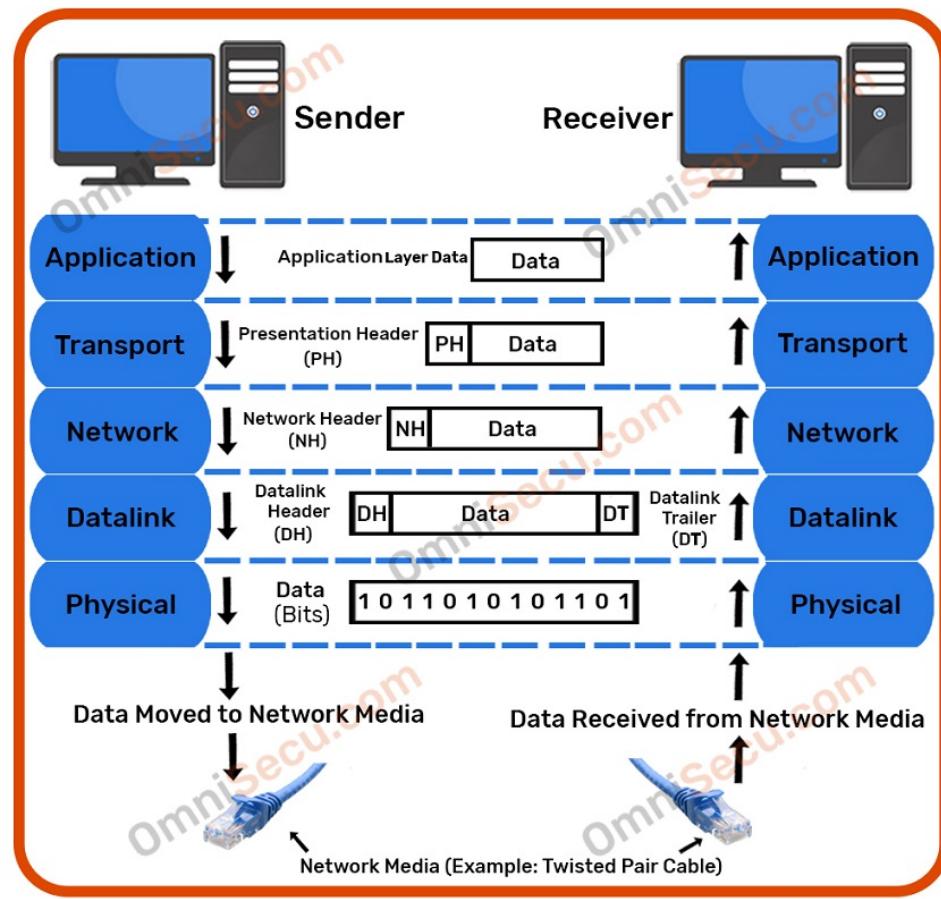


## Encapsulation



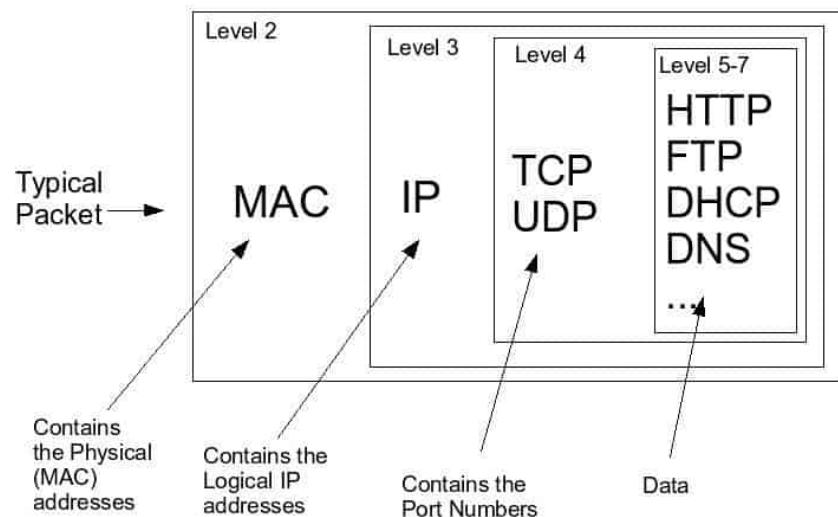




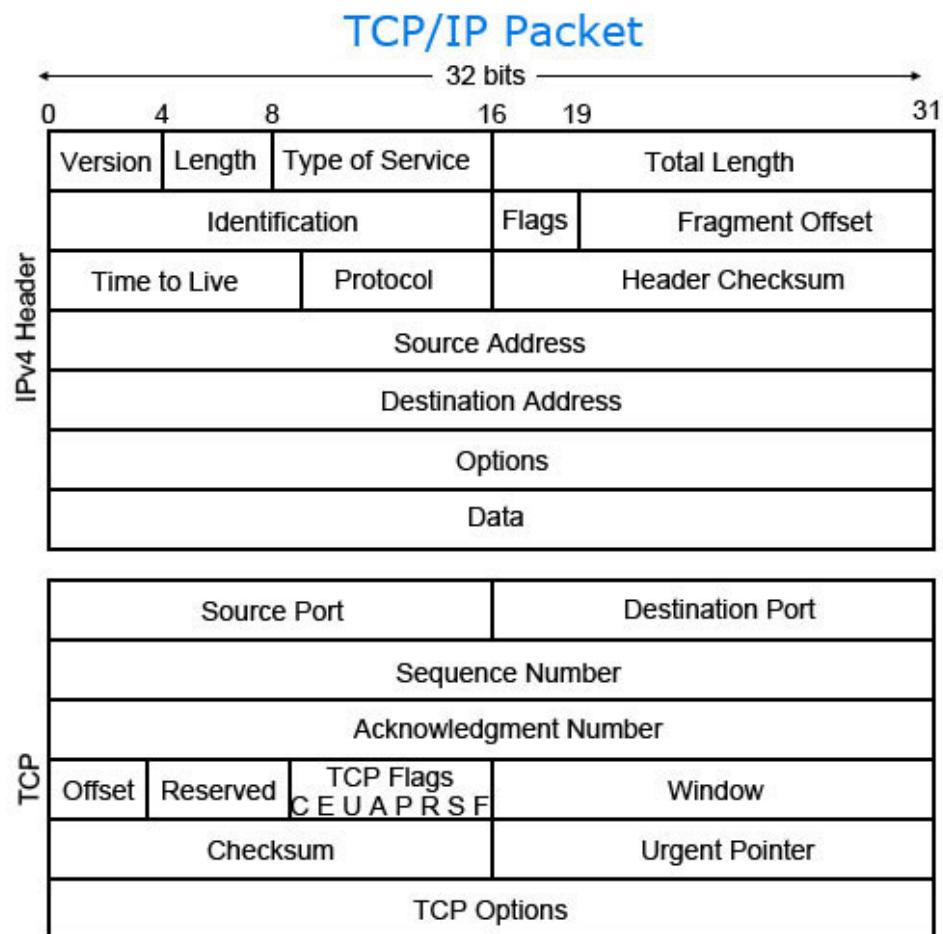
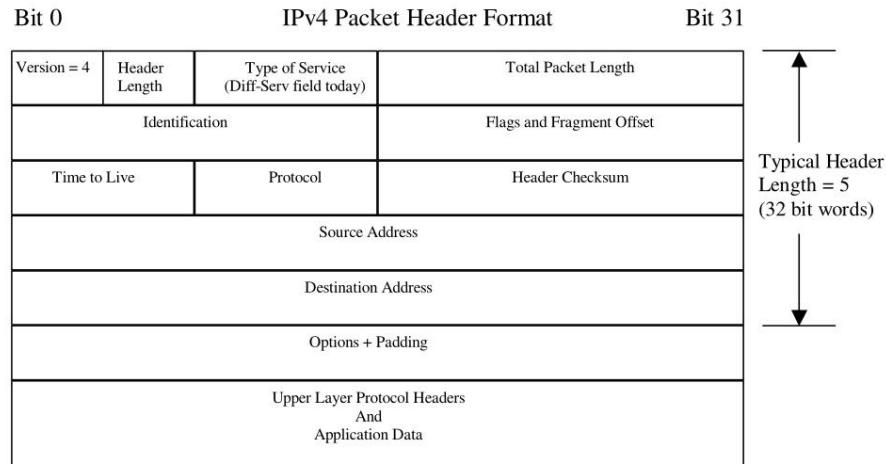


© OmniSecu.com

## Packet



# IP Packet Format



ComputerHope.com

## Packet Switch

# How TCP/IP Works

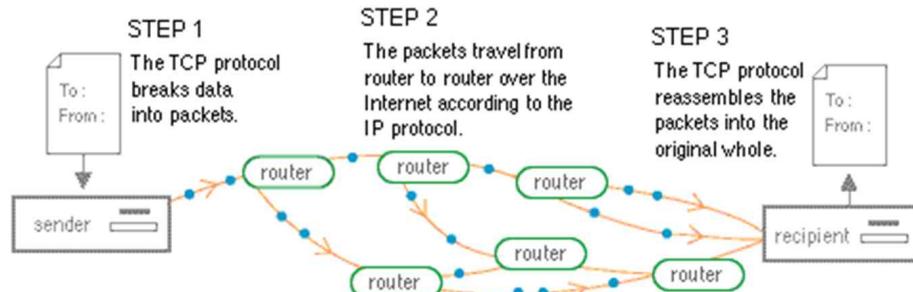


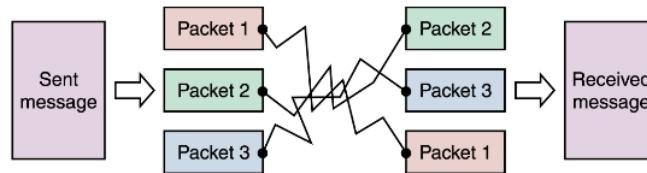
Figure 2. How data travels over the Net.

Dr. Vinton Cerf

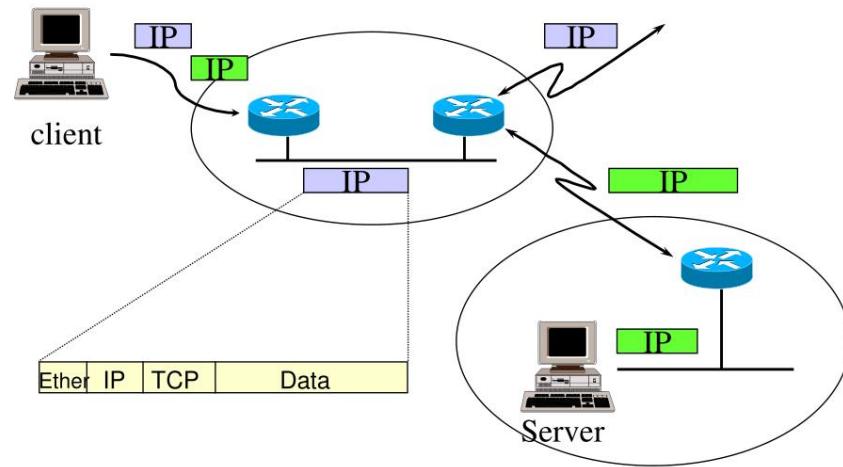


## Packet Switching

- To improve the efficiency of transferring information over a shared communication line, messages are divided into fixed-sized, numbered **packets**
- Network devices called routers are used to direct packets between networks

Figure 15.4  
Messages sent  
by packet  
switchingMessage is divided  
into packetsPackets are sent over the Internet  
by the most expedient routePackets are reordered  
and then reassembled

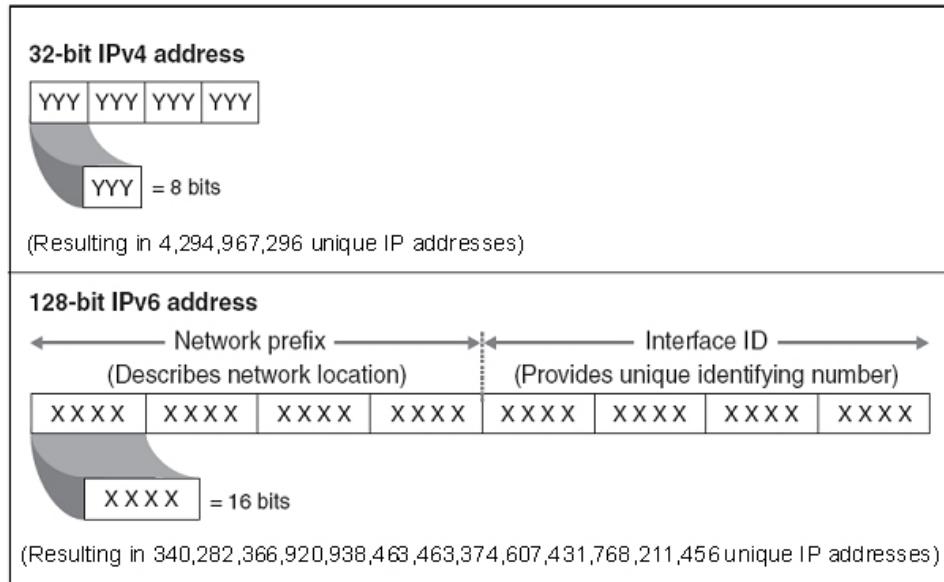
# Packet Switch Network



## IP4 vs IP6

IPv4 Header				IPv6 Header				
Version	IHL	Type of Service	Total Length		Version	Traffic Class	Flow Label	
Identification			Flags	Fragment Offset	Payload Length		Next Header	Hop Limit
Time to Live	Protocol	Header Checksum			Source Address			
Destination Address				Destination Address				
Options		Padding						
<b>Legend</b> <ul style="list-style-type: none"> <li>Yellow Box: Field's Name Kept from IPv4 to IPv6</li> <li>Red Box: Fields Not Kept in IPv6</li> <li>Blue Box: Name and Position Changed in IPv6</li> <li>Teal Box: New Field in IPv6</li> </ul>								

Figure 1: Comparison of IPv6 and IPv4 Address Scheme



Source: GAO.

## Differences Between IPv4 and IPv6

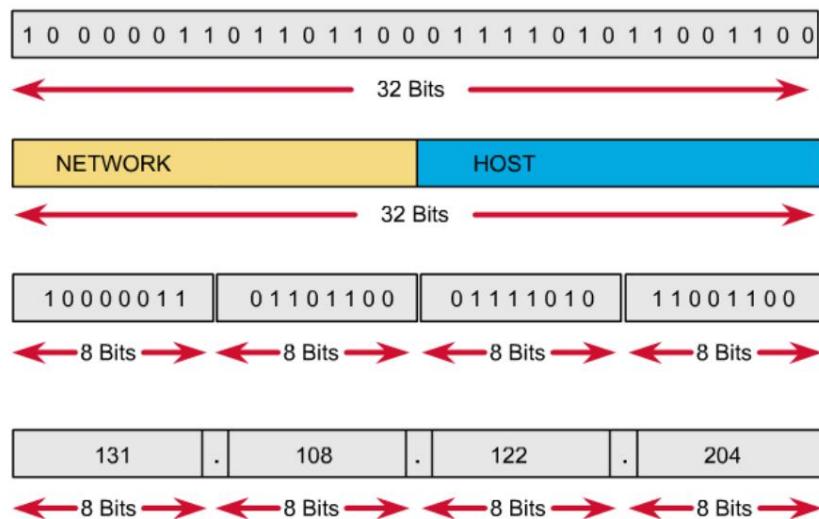
Feature	IPv4	IPv6
Fragmentation	Performed by routers and sending host	Performed only by sending host
Address Resolution	Broadcast ARP Request frames	Multicast Neighbor Solicitation messages
Manage multicast group membership	IGMP	Multicast listener discovery
Router Discovery	ICMP Router Discovery (optional)	ICMPv6 Router Solicitation and Router Advertisement (required)
DNS host records	A records	AAAA records
DNS reverse lookup zones	IN-ADDR.ARPA	IP6.ARPA
Minimum packet size	576 bytes	1280 bytes

### IPv4/IPv6 Differences

	IPv4	IPv6
<b>Address</b>	32 bits (4 bytes) 12:34:56:78	128 bits (16 bytes) 1234:5678:9abc:defo:1234:5678:9abc:defo
<b>Packet size</b>	576 bytes required, fragmentation optional	1280 bytes required without fragmentation
<b>Packet fragmentation</b>	Routers and sending hosts	Sending hosts only
<b>Packet header</b>	Does not identify packet flow for QoS handling	Contains Flow Label field that specifies packet flow for QoS handling
	Includes a checksum	Does not include a checksum
	Includes options up to 40 bytes	Extension headers used for optional data
<b>DNS records</b>	Address (A) records, maps host names	Address (AAAA) records, maps host names
	Pointer (PTR) records, IN-ADDR.ARPA DNS domain	Pointer (PTR) records, IP6.ARPA DNS domain
<b>Address configuration</b>	Manual or via DHCP	Stateless address autoconfiguration (SLAAC) using Internet Control Message Protocol version 6 (ICMPv6) or DHCPv6
<b>IP to MAC resolution</b>	broadcast ARP	Multicast Neighbor Solicitation
<b>Local subnet group management</b>	Internet Group Management Protocol (IGMP)	Multicast Listener Discovery (MLD)
<b>Broadcast</b>	Yes	No
<b>Multicast</b>	Yes	Yes
<b>IPSec</b>	optional, external	required

## IP Address

### IP address format

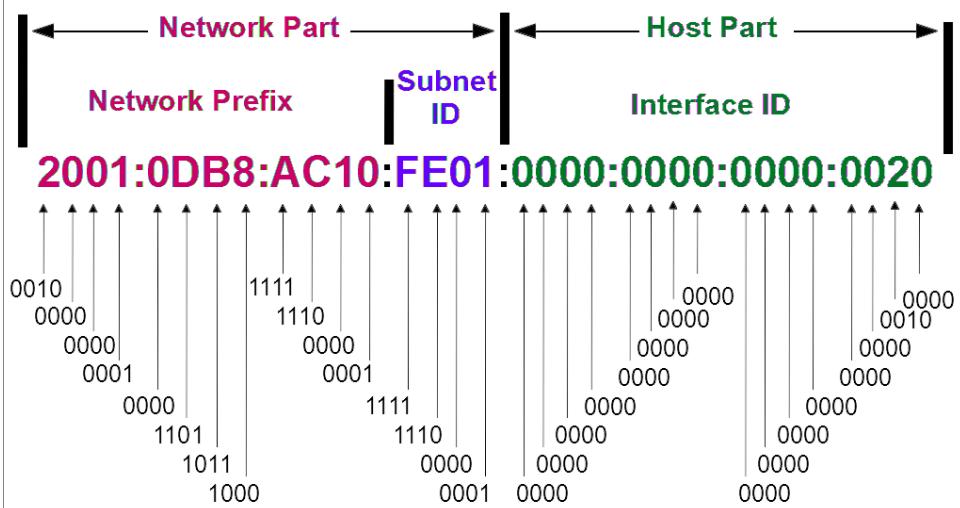


Class	Private Address Ranges
Class A	10.0.0.0 – 10.255.255.255
Class B	172.16.0.0 – 172.31.255.255
Class C	192.168.0.0 – 192.168.255.255
Loopback	127.0.0.0 – 127.255.255.255 (127.0.0.1)

## IPv6 Address Structure

128 Bits, Expressed in Hex (Hexadecimal) with 3 parts

This is the usual breakdown but it can be broken down in other ways



## IPv6 Address Notation

One Hex digit = 4 bits

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

**2001:0DB8:AAAA:1111:0000:0000:0000:0100/64**

1      2      3      4      5      6      7      8  
**2001 : 0DB8 : AAAA : 1111 : 0000 : 0000 : 0000 : 0100** / 16 bits    16 bits

- IPv6 addresses are 128-bit addresses represented in:
  - Eight 16-bit segments or “hextets” (not a formal term)
  - Hexadecimal (non-case sensitive) between 0000 and FFFF

## IP Range

### Global Addresses

Class	First Octet value	Range	No. of Network	No. of Hosts / Network
A	<u>0</u> 0000000 – <u>0</u> 1111111 (0 – 127)	1.0.0.1 – 126.255.255.254	126	$2^{24} - 2$
B	<u>1</u> 0000000 – <u>1</u> 0111111 (128 – 191)	128.1.0.1 – 191.255.255.254	16000	65000
C	<u>1</u> 1000000 – <u>1</u> 1011111 (192 – 223)	192.0.1.1 – 223.255.255.254	2 Million	254
D	<u>1</u> 1100000 – <u>1</u> 1101111 (224 – 239)	224.0.0.0 – 239.255.255.255		Multicast addresses
E	<u>1</u> 1110000 – <u>1</u> 1111111 (240 – 255)	240.0.0.0 – 254.255.255.254		Future use

In class A **127.0.0.1 – 127.255.255.255** addresses are reserved for loopback & diagnostic purpose.

### IP Address Ranges

IP Address Class	First Octet Binary Value	First Octet Decimal Value	Possible Number of Hosts
Class A	1-126	<u>0</u> 0000001 to <u>0</u> 1111110*	16,777,214
Class B	128-191	<u>1</u> 0000000 to <u>1</u> 0111111	65,534
Class C	192-223	<u>1</u> 1000000 to <u>1</u> 1011111	254

\*127 (01111111) is a Class A address reserved for loopback testing and cannot be assigned to a network.



## Private IP ranges

- Often it is necessary to connect devices to the network, but not to the internet. RFC 1918 manages the private IP addresses that cannot appear on the internet, but are reserved for private use.

- Private IP ranges managed by IANA:

Class	From	To	No. Of hosts
1 x A class	10.0.0.0	10.255.255.255	$2^{24} = 16.777.216$
16 x B class	172.16.0.0	172.31.255.255	$2^{20} = 1.048.576$
256 x C class	192.168.0.0	192.168.255.255	$2^{16} = 65.536$

- example:

- 192.168.1.0/24 (mask: 255.255.255.0 | 256 hosts) - 256 networks
- 172.17.0.0/16 (mask: 255.255.0.0 | 65.536 hosts) 256 networks

### What Do You Need To Know About Private IP Address?

#### Class

#### Private Address Ranges

Class A

10.0.0.0 – 10.255.255.255

Class B

172.16.0.0 – 172.31.255.255

Class C

192.168.0.0 – 192.168.255.255

Loopback

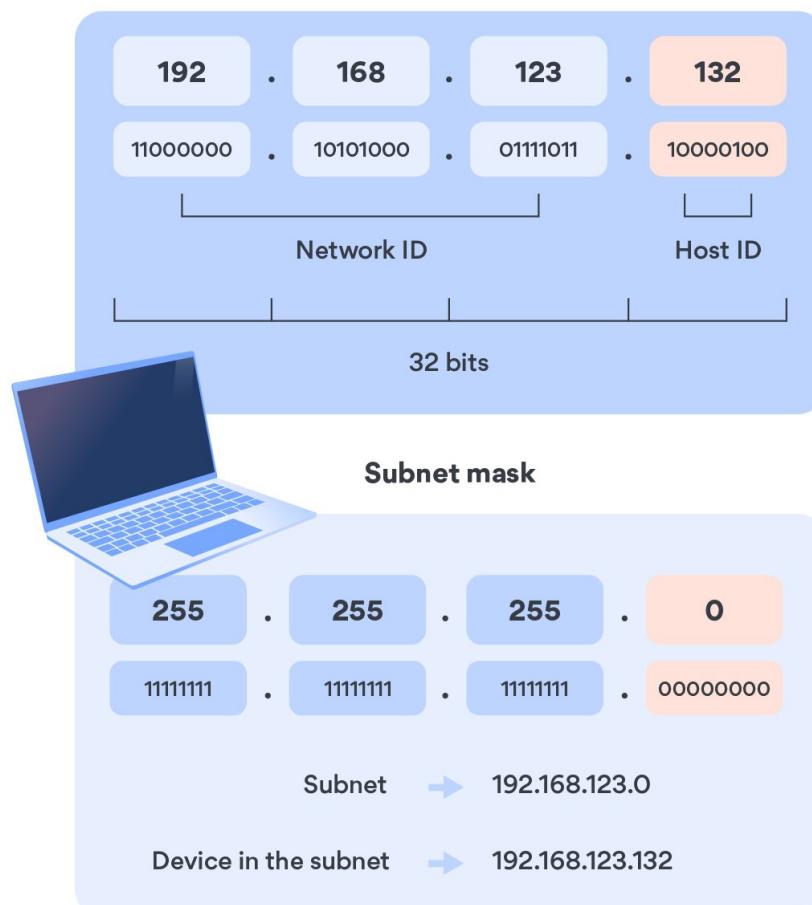
127.0.0.0 – 127.255.255.255  
(127.0.0.1)

## Subnet Mask

## **Subnet Mask**

Suffix	Hosts	32-Borrowed=CIDR	$2^{\text{Borrowed}}$ = Hosts	Binary=> dec = Suffix
.255	1	/32	0	11111111
.254	2	/31	1	11111110
.252	4	/30	2	11111100
.248	8	/29	3	11111000
.240	16	/28	4	11110000
.224	32	/27	5	11100000
.192	64	/26	6	11000000
.128	128	/25	7	10000000

## IP address explained



## The Default Subnet Masks (no subnets)

	1st octet	2nd octet	3rd octet	4th octet
<b>Class A</b>	<b>Network</b>	<b>Host</b>	<b>Host</b>	<b>Host</b>
<b>Class B</b>	<b>Network</b>	<b>Network</b>	<b>Host</b>	<b>Host</b>
<b>Class C</b>	<b>Network</b>	<b>Network</b>	<b>Network</b>	<b>Host</b>
<b>Class A or /8</b>	<b>11111111</b>	<b>00000000</b>	<b>00000000</b>	<b>00000000</b>
<b>Class B or /16</b>	<b>11111111</b>	<b>11111111</b>	<b>00000000</b>	<b>00000000</b>
<b>Class C or /24</b>	<b>11111111</b>	<b>11111111</b>	<b>11111111</b>	<b>00000000</b>

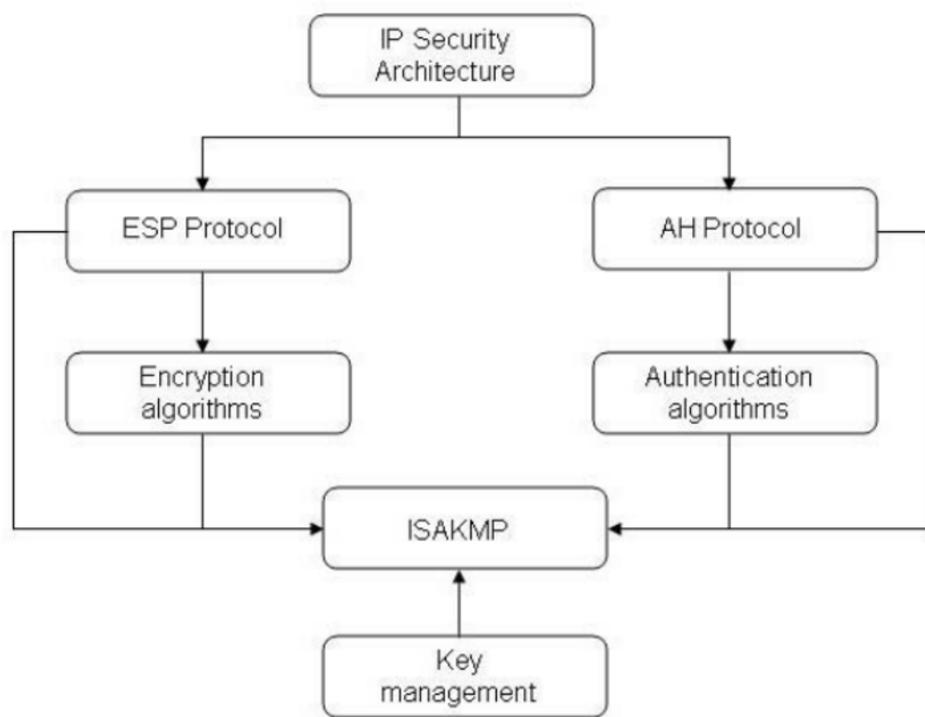
- A "1" bit in the subnet mask means that the corresponding bit in the IP address should be read as a network number
- A "0" bit in the subnet mask means that the corresponding bit in the IP address should be read as a host bit.
- /n "slash" tells us how many "1" bits are in the subnet mask.

## The Subnet Mask

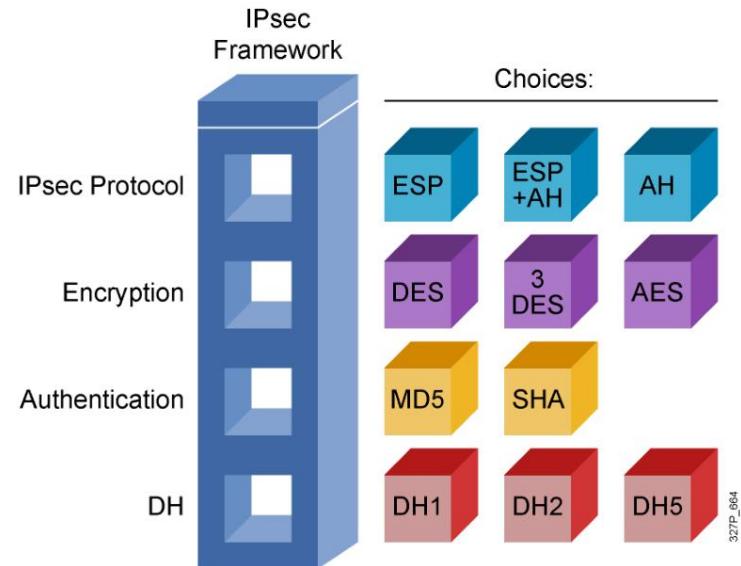
- Subnet Mask:
  - Let's not forget about the subnet mask.
    - Each class has a **default or "natural"** subnet mask based on the default number of bits used for the network and host portion.

Class	Number of Network Bits	Number of Host Bits	Default Prefix	Default Subnet Mask
A	8	24	/8	255.0.0.0
B	16	16	/16	255.255.0.0
C	24	8	/24	255.255.255.0

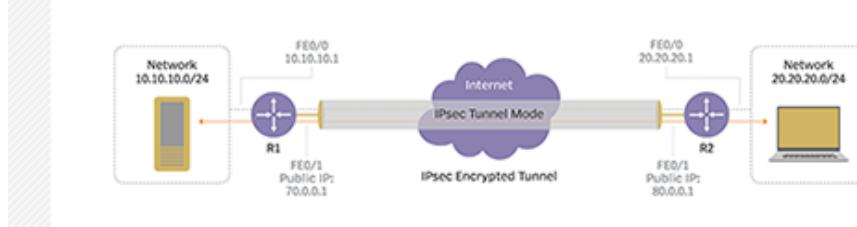
## IPSec



## IPsec Protocol Framework



## IPsec tunnel mode



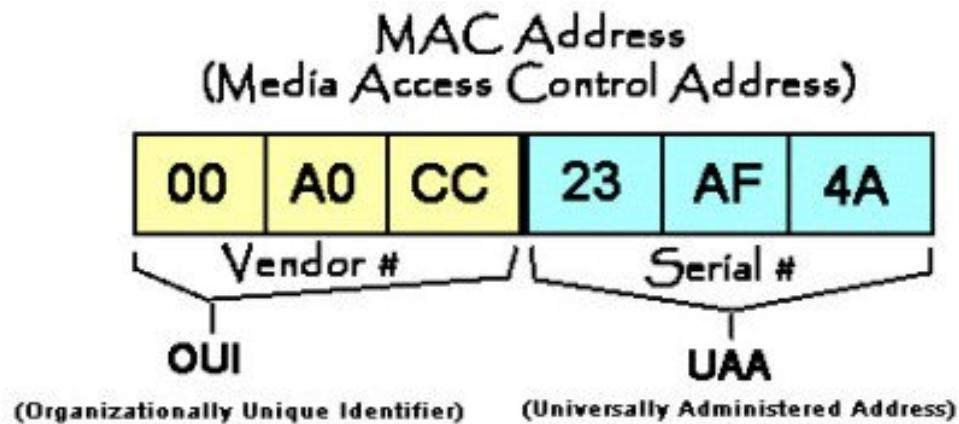
## IPsec: Network Layer Security

- **network-layer secrecy:**
  - sending host encrypts the data in IP datagram
  - TCP and UDP segments; ICMP and SNMP messages.
- **network-layer authentication**
  - destination host can authenticate source IP address
- **two principal protocols:**
  - authentication header (AH) protocol
  - encapsulation security payload (ESP) protocol
- **for both AH and ESP, source, destination handshake:**
  - create network-layer logical channel called a security association (SA)
- **each SA unidirectional.**
- **uniquely determined by:**
  - security protocol (AH or ESP)
  - source IP address
  - 32-bit connection ID

8: Network Security

8-1

## MAC Address - DataLink Layer



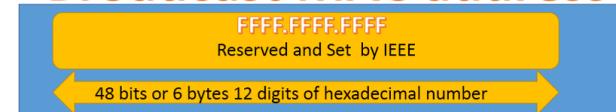
### Unicast MAC address



### Multicast MAC address

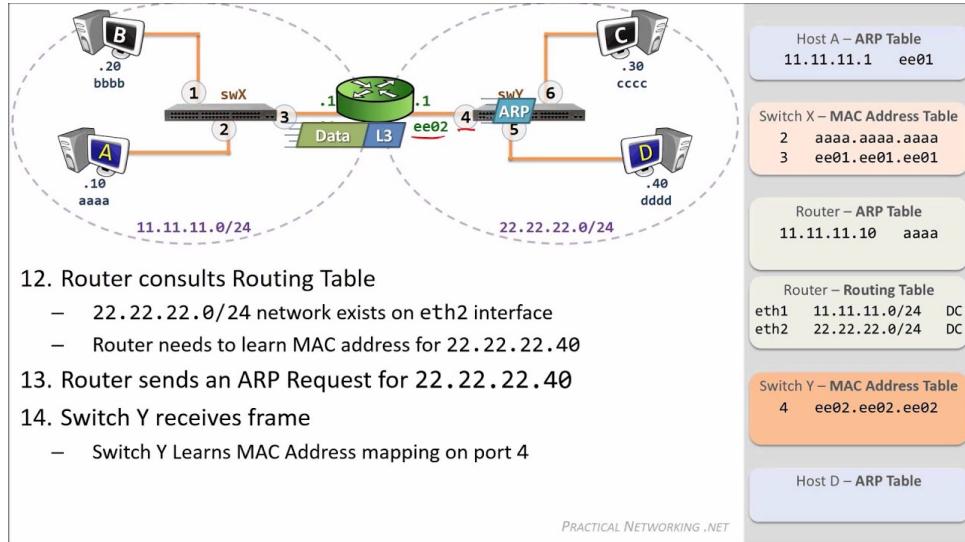


### Broadcast MAC address



# ARP(Address Resolution Protocol) / RARP (ReverseAddress Resolution Protocol)

- ARP: resolve IP Address to MAC Address
- RARP: resolve MAC Address to IP Address

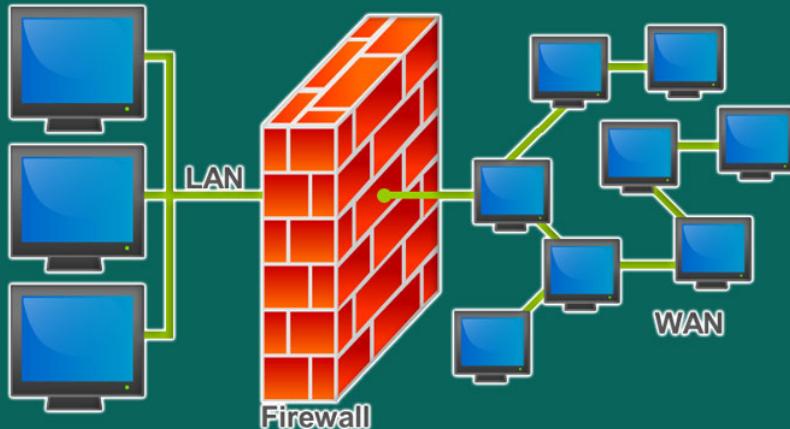


12. Router consults Routing Table
  - 22.22.22.0/24 network exists on eth2 interface
  - Router needs to learn MAC address for 22.22.22.40
13. Router sends an ARP Request for 22.22.22.40
14. Switch Y receives frame
  - Switch Y Learns MAC Address mapping on port 4

## Network Equipment

### Firewall

# Configure Firewall & Internet Security of the QuickBooks Desktop



## What is firewall?

A firewall is nothing but a network security system that monitors and controls over all your incoming and outgoing network traffic based on advanced and a defined set of security rules.

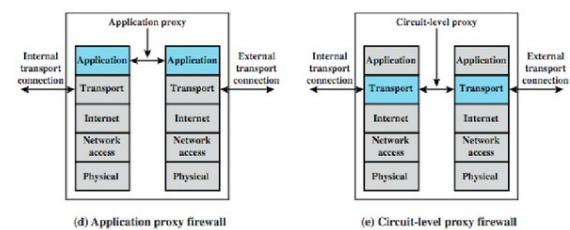
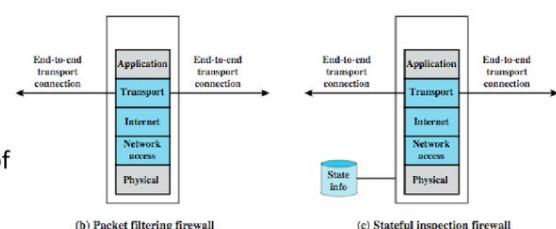
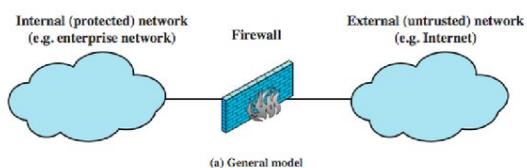
It simply prevents unauthorized access to or from a private network. Used to enhance the security of computers connected to a network, such as LAN or the Internet. Considered as an integral part of a comprehensive security framework for your network.

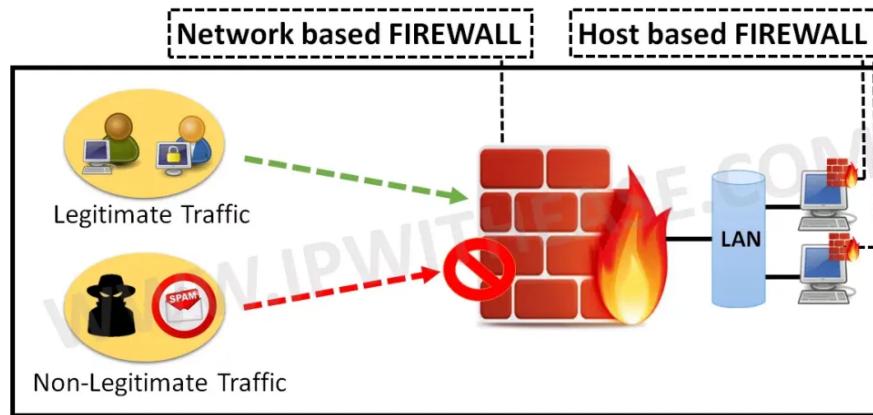


## Types of Firewalls

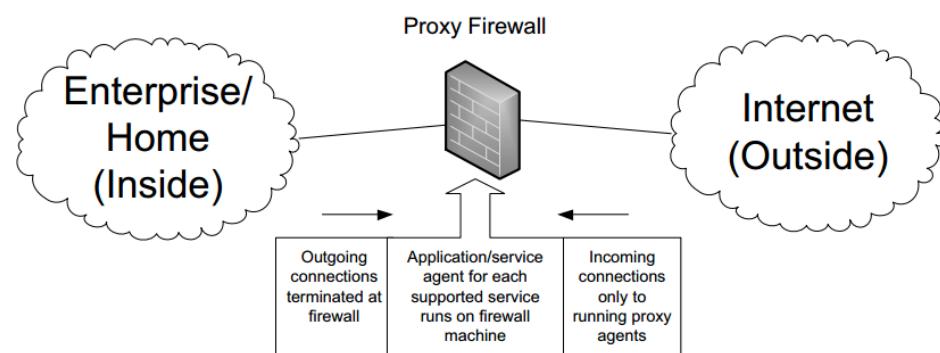
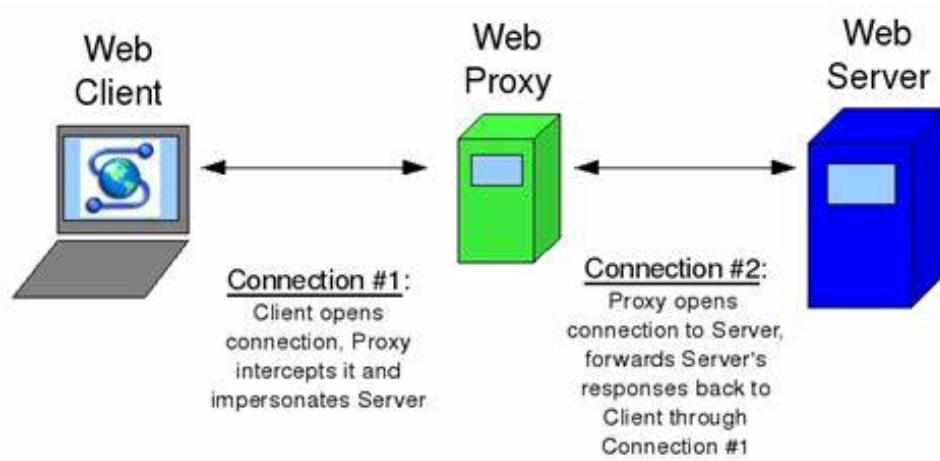
**Positive (negative) filter:**  
Allow (reject) packets that meet a criteria

**Stateful inspection:** Keeps track of TCP connections





## Proxy



## Proxy Cache

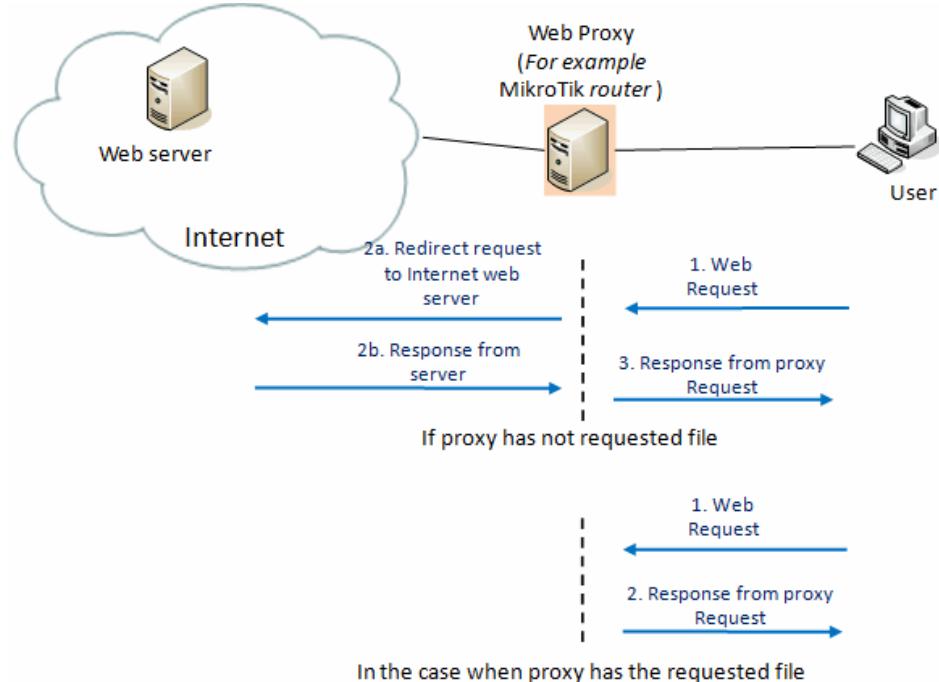
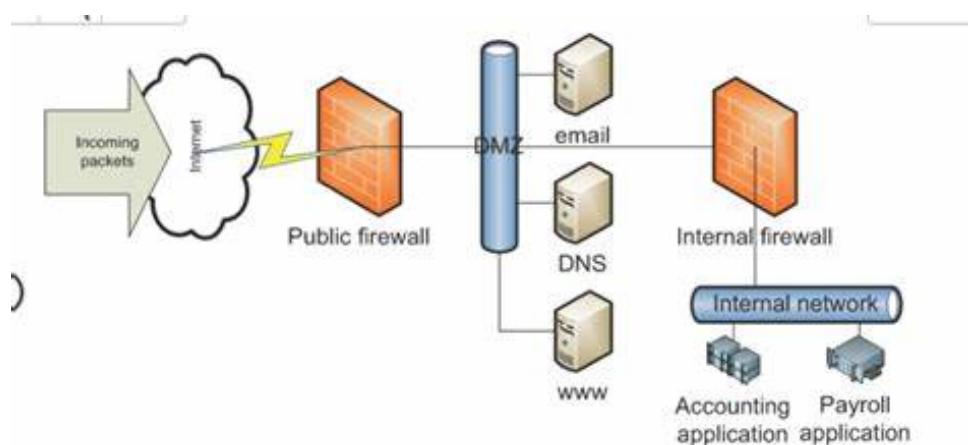


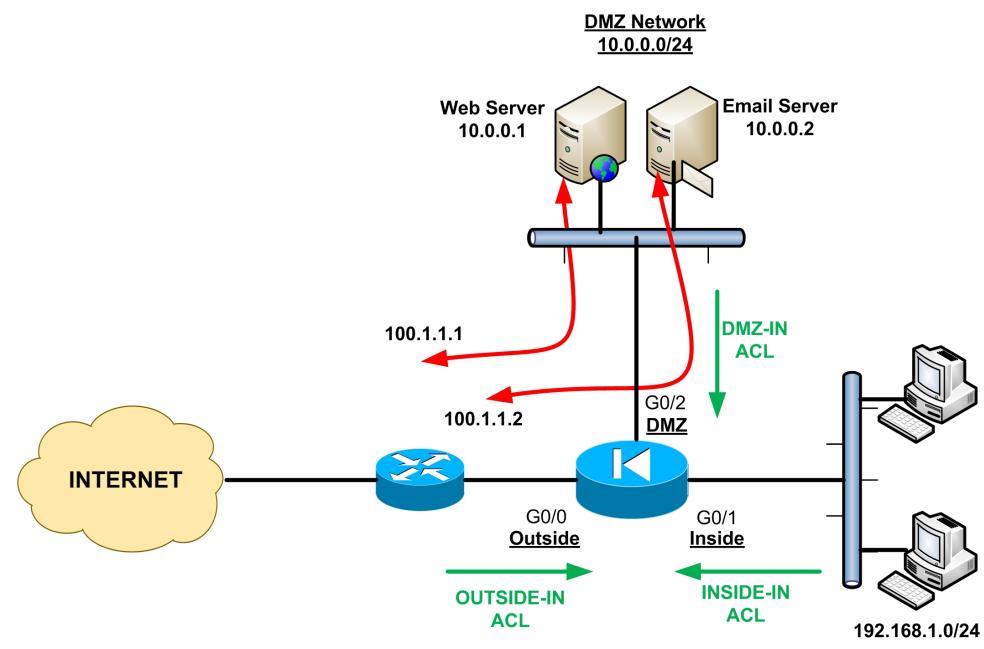
Figure 10.1. Web proxy basic operation scheme

## ACL (Access Control List)

Parameter	ACL	Firewall
Asset Type	Feature on Layer 3 devices and Firewalls	Hardware or Software
Stateful/Stateless inspection	Performs stateless inspection	Performs Stateful inspection
Scope wrt OSI	Upto Layer 4	Upto Layer 7
Security	Low	High
Intrusion detection	Not possible	Possible
Target deployment	Setups requiring low level of security	Setups requiring higher level of security

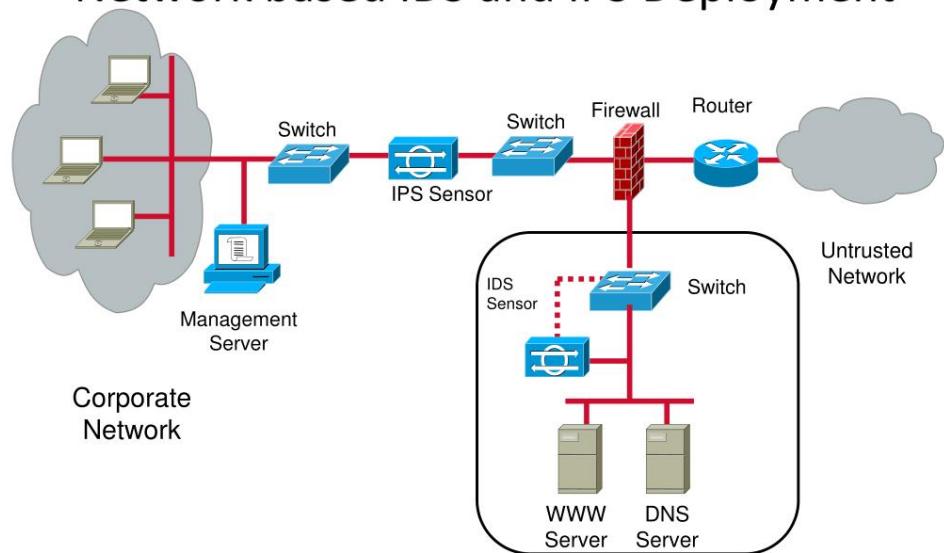
## DMZ





## IPS/IDS

### Network based IDS and IPS Deployment



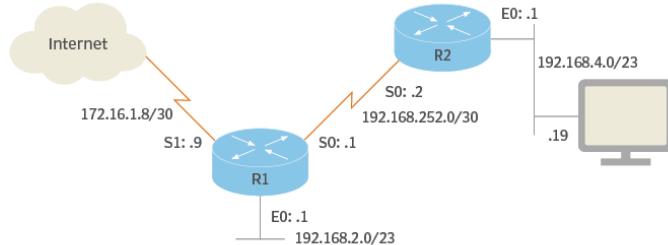
Engineering and Management of Secure Computer Networks

15

## Routing Table

## Subnet masks, prefixes and routing

In this diagram, R1 receives a packet addressed to 192.168.5.19, a host that's connected to R2's LAN. Using a binary AND operation on the address and its mask, R1 finds 192.168.4.0 and forwards the packet out the S0 interface to R2, which will perform the same prefix calculation. R2 determines it should send the packet on interface E0 and deliver it to host 5.19.



R1'S ROUTING TABLE

Prefix	192.168.2.0	192.168.4.0	192.168.252.0	0.0.0.0
Mask	255.255.254.0	255.255.254.0	255.255.255.252	0.0.0.0
Outgoing interface	E0	S0 to R2	S0	S1 to internet (default)

SOURCE: NETWORK ARCHITECT TERRY SLATTERY

©2019 TECHTARGET. ALL RIGHTS RESERVED TechTarget

## Routing Protocol

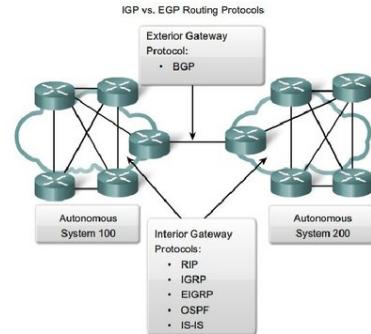
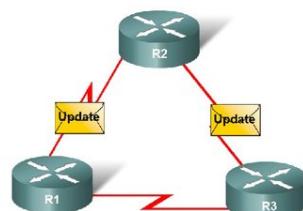
### Dynamic IP Routing Protocols

Routing Protocols learn and **dynamically** share information about the networks connected to each other therefore these protocols are called **dynamic protocols**.

There are quite many dynamic routing protocols for routing IP packets. The most common protocols are:

- **RIP** (Routing Information Protocol);
- **IGRP** (Interior Gateway Routing Protocol);
- **EIGRP** (Enhanced Interior Gateway Routing Protocol);
- **OSPF** (Open Shortest Path First);
- **IS-IS** (Intermediate System-to-Intermediate System) (*pronounced "i-s i-s" or more commonly "Eye-Sis"*);
- **BGP** (Border Gateway Protocol).

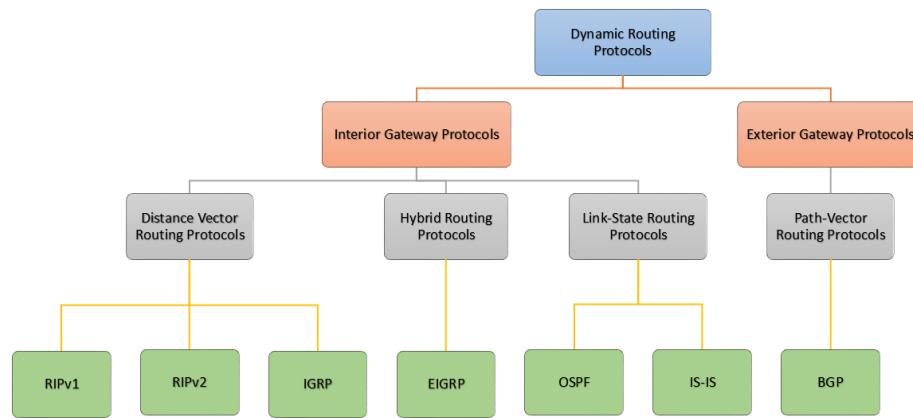
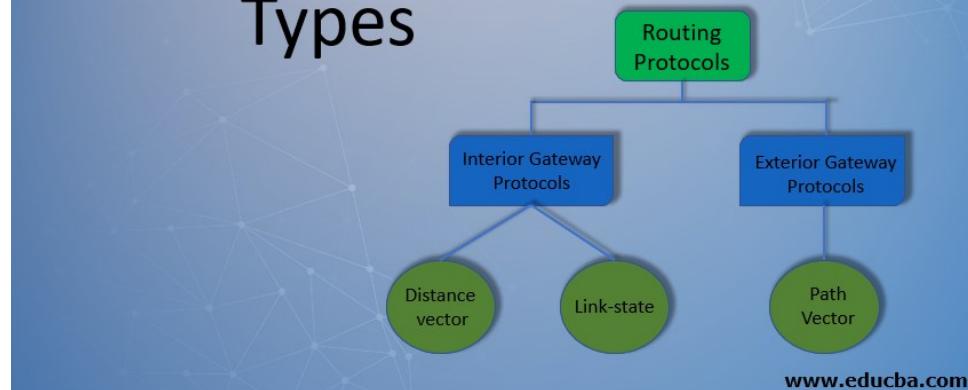
Routers Dynamically Pass Updates



16

# Routing Protocols

## Types



## Routing Protocols for IP Networks

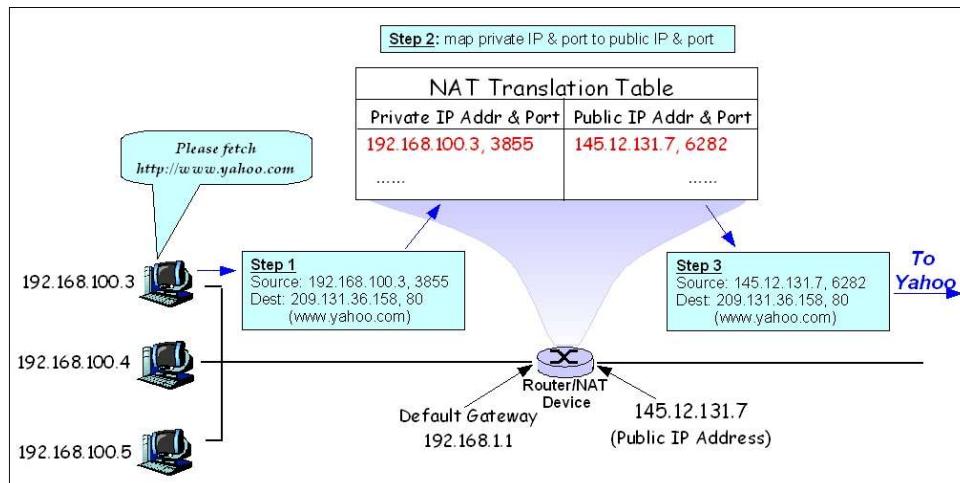
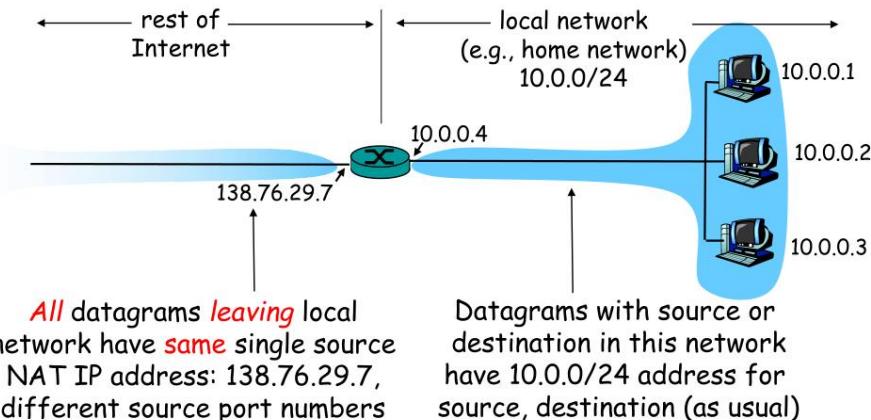
Protocol	Type	Scalability	Metric	IP classes
RIP-1	Distance vector	Small	Hop count	Classful
RIP-2	Distance vector	Small	Hop count	Classless
OSPF-2	Link state	Large	Cost	Classless
IS-IS	Link state	Very large	Cost	Classless
IGRP	Distance vector	Medium	Bandwidth, delay, load, MTU, reliability	Classful
EIGRP	Dual	Large	Bandwidth, delay, load, MTU, reliability	Classless
BGP	Distance vector	Large	Vector of attributes	Classless



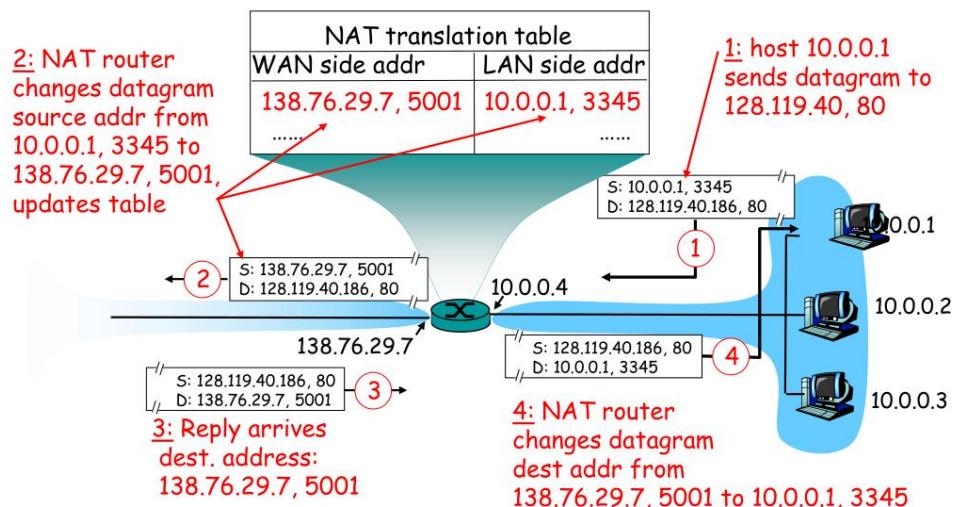
7

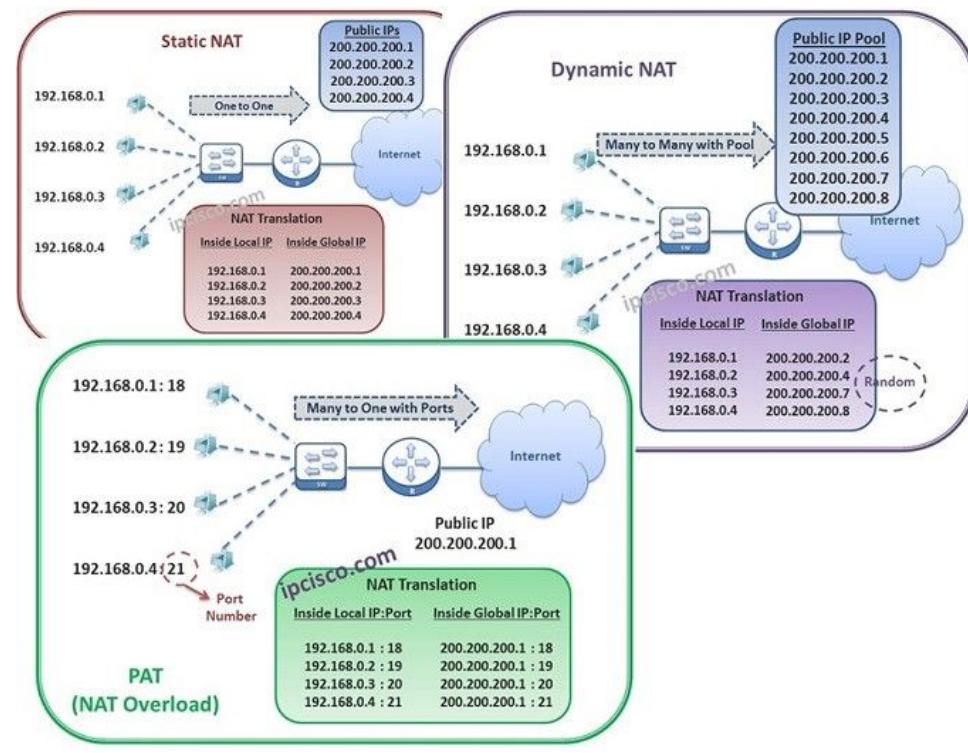
## NAT(Network Address Translation)

# NAT: Network Address Translation

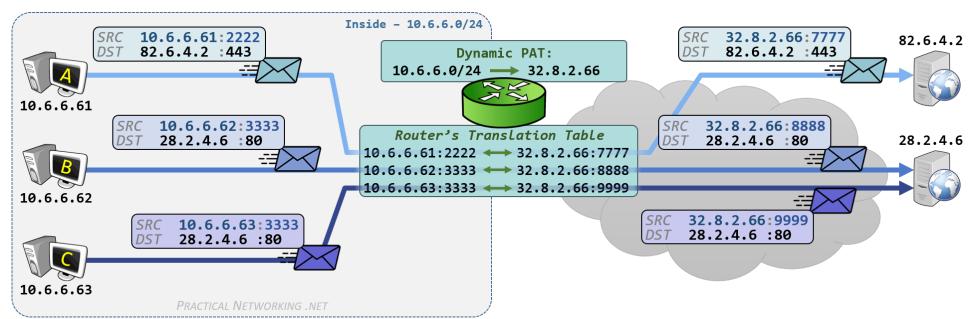
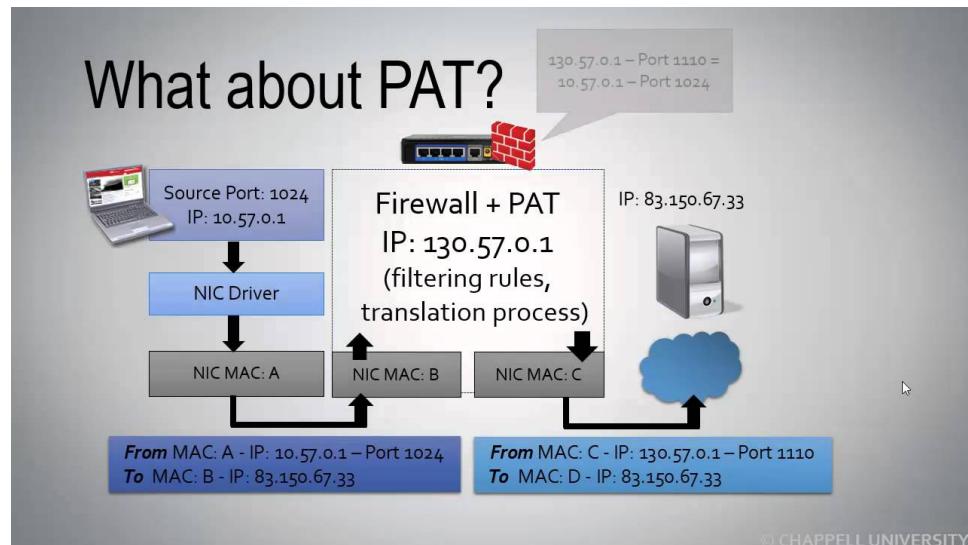


# NAT: Network Address Translation





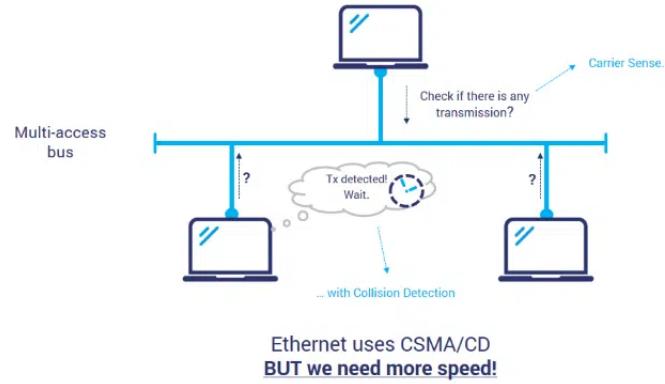
## PAT (Port Address Translation)



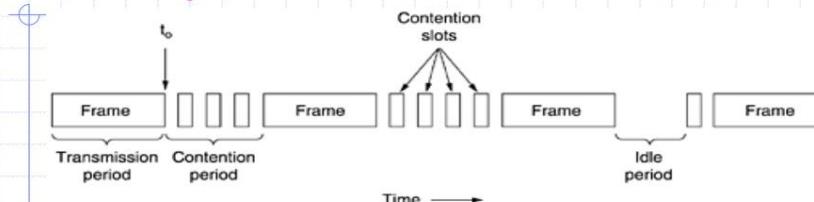
## CSMA/CD

## CSMA/CD

CSMA/CD – mechanism for collision detection introduced to detect transmission

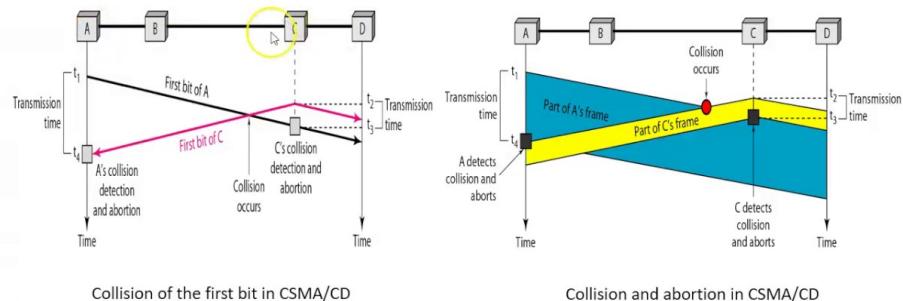


## CSMA/CD



- ◆ Sense the channel
- ◆ Stop sending when detecting collision
- ◆ After collision wait a random amount of time and try again.

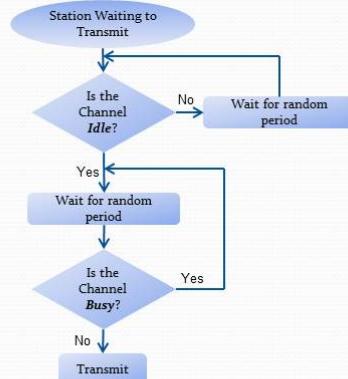
## Collision in CSMA /CD



## CSMA/CA

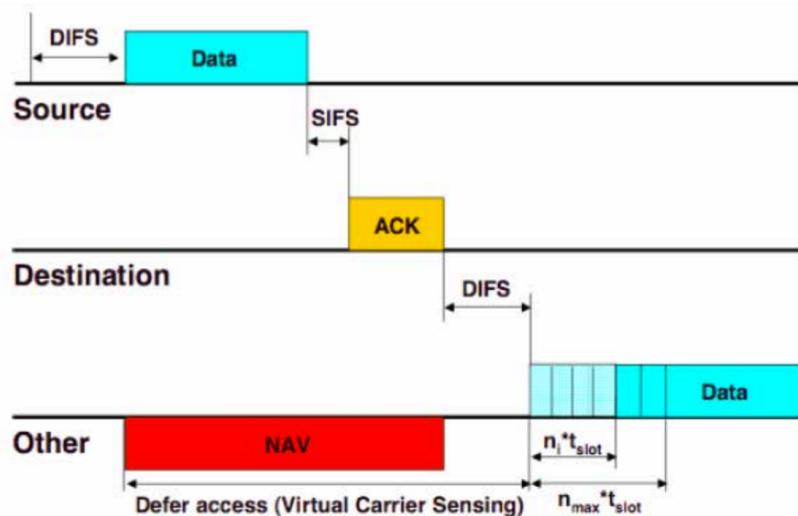
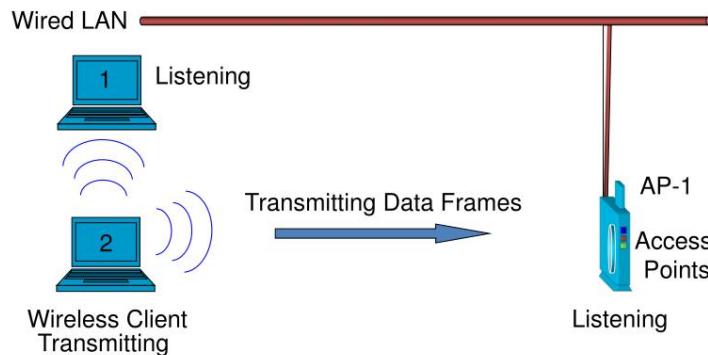
# CSMA/CA

- CSMA/CA is a wireless network multiple access method in which:
  - A carrier sensing scheme is used.
  - A node wishing to transmit data has to first listen to the channel for a predetermined amount of time whether or not another node is transmitting on channel within the wireless range. If the channel is sensed “idle”, then the node is permitted to begin the transmission process. If the channel is sensed as “busy”, the node defers its transmission for a random period of time.
  - State of channel “Idle” or “Busy” is based on CS mechanism, which will be explained later in the presentation

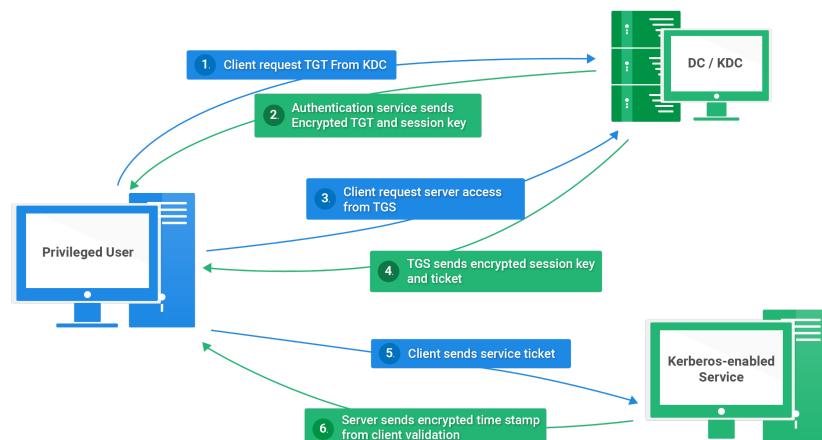
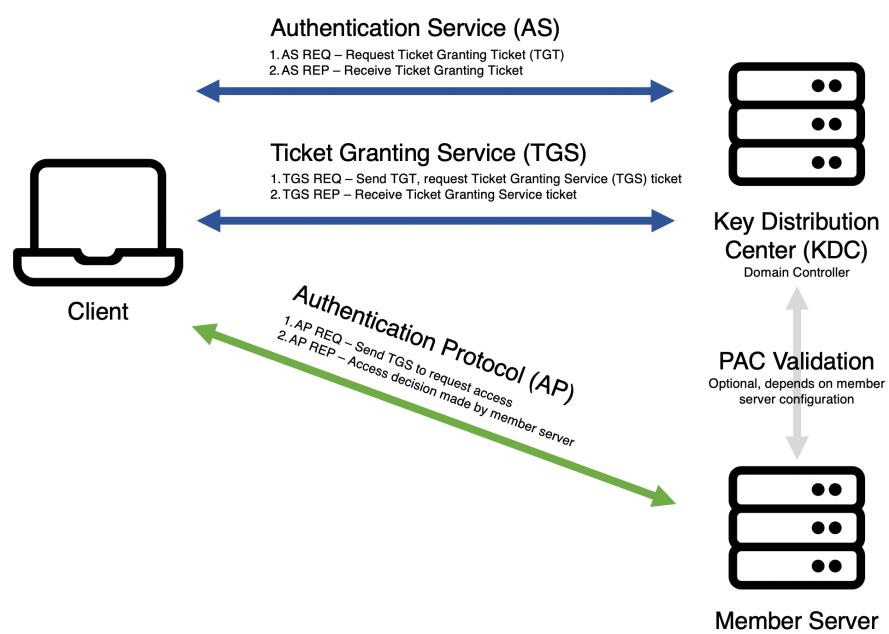


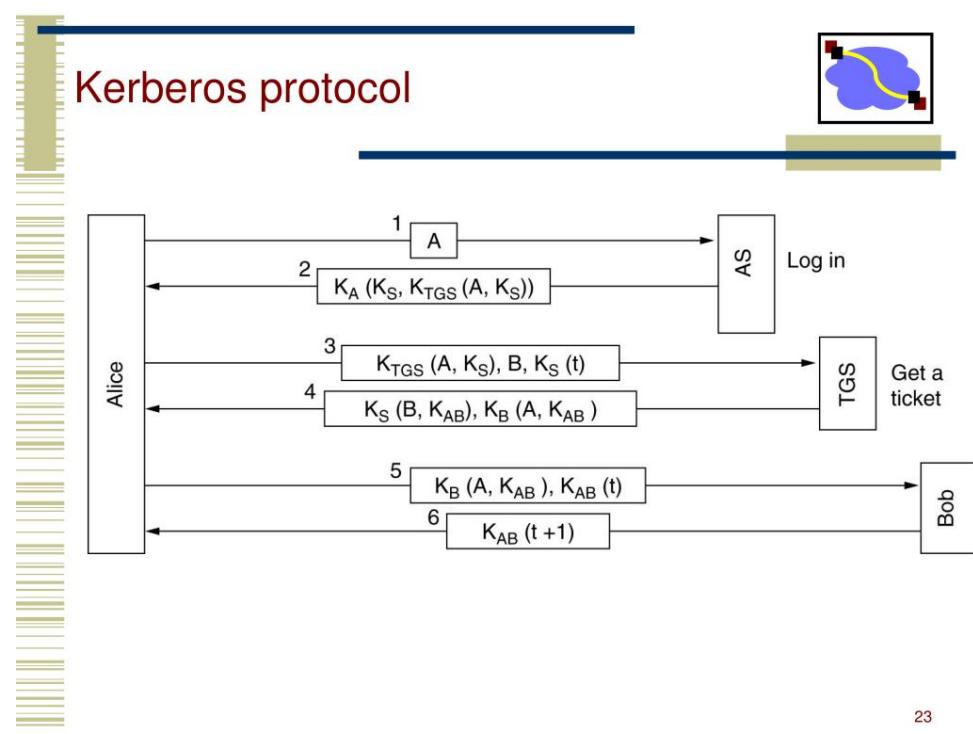
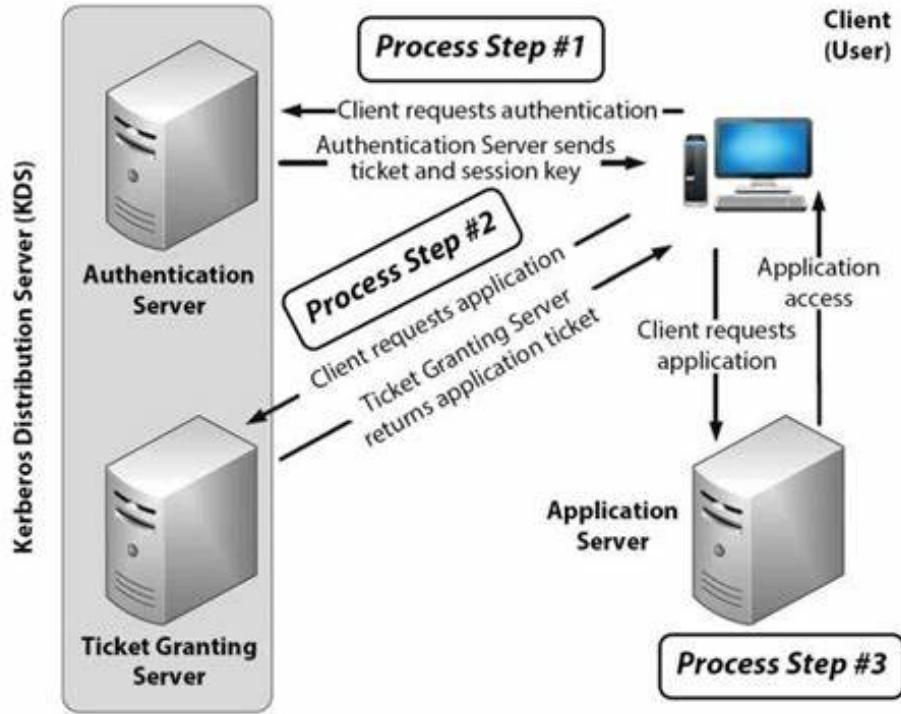
## CSMA/CA Collision Handling

- 802.11 standard employs half-duplex radios-radios capable of transmission or reception-but not both simultaneously



# Kerberos

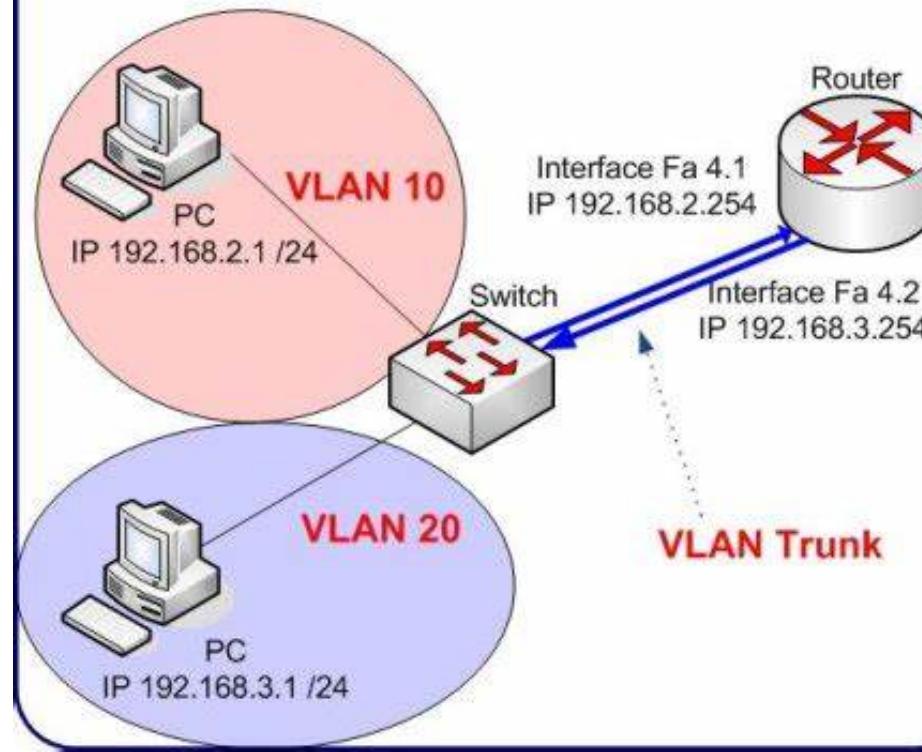




23

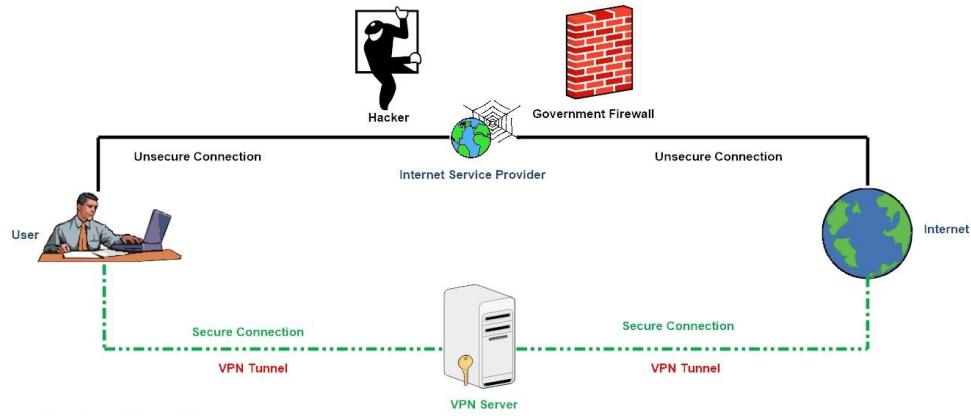
## VLAN

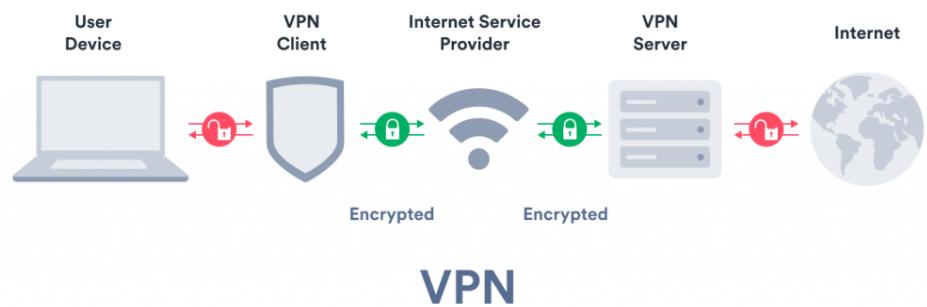
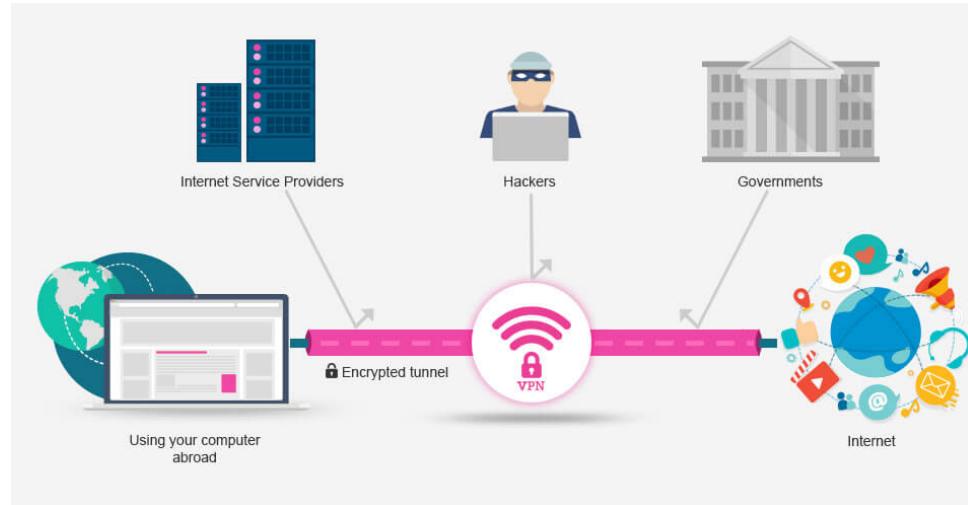
## Sample network using VLANs



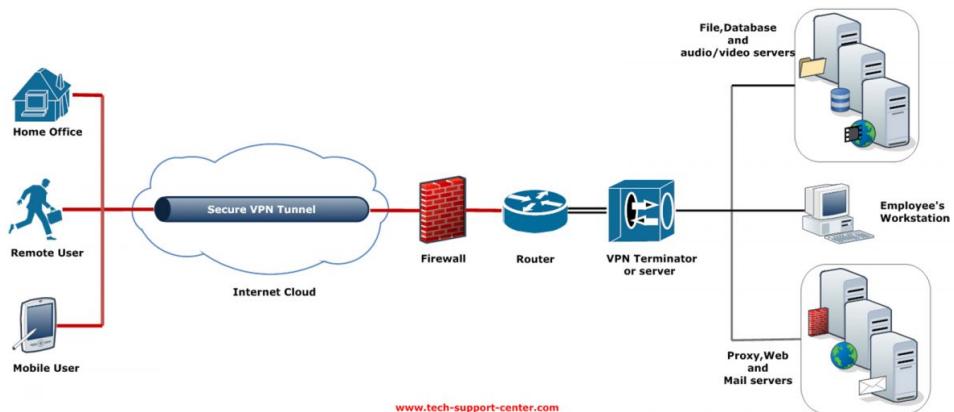
## VPN

### How VPN Works

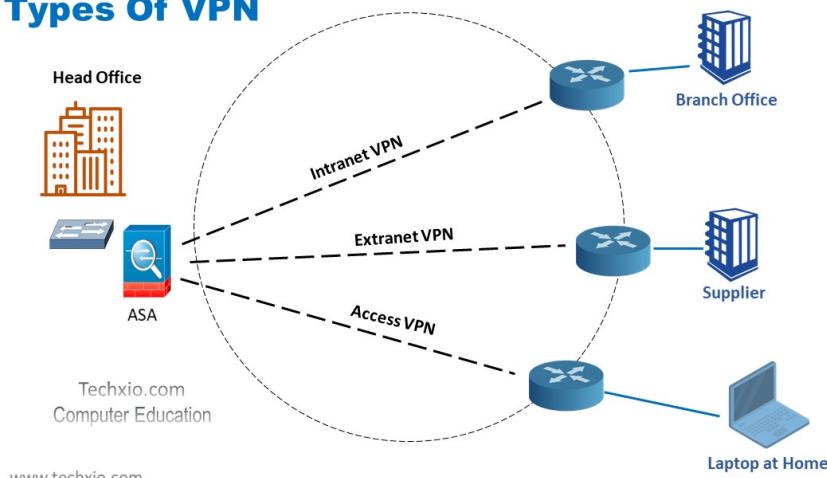




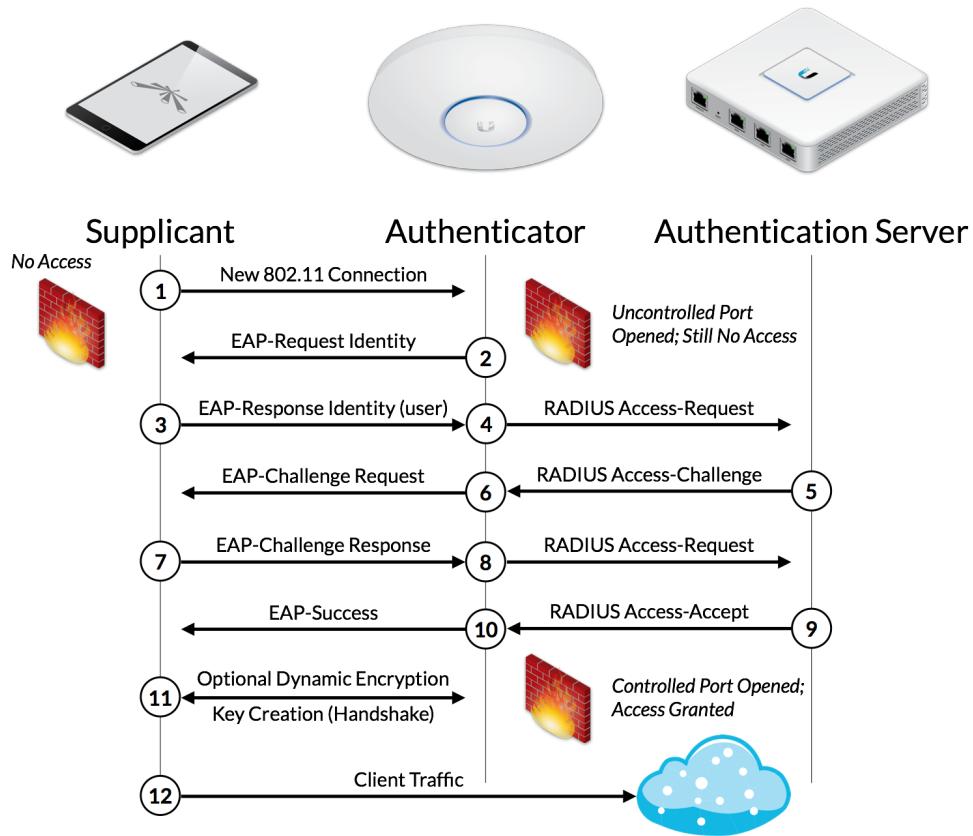
## VPN



## Types Of VPN

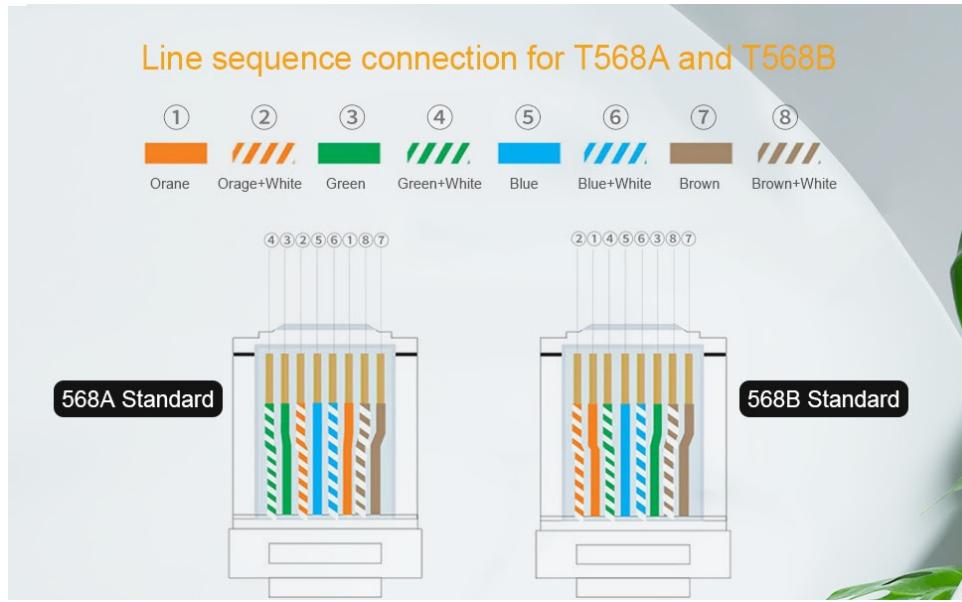


# 802.1X Authentication (EAP & RADIUS)

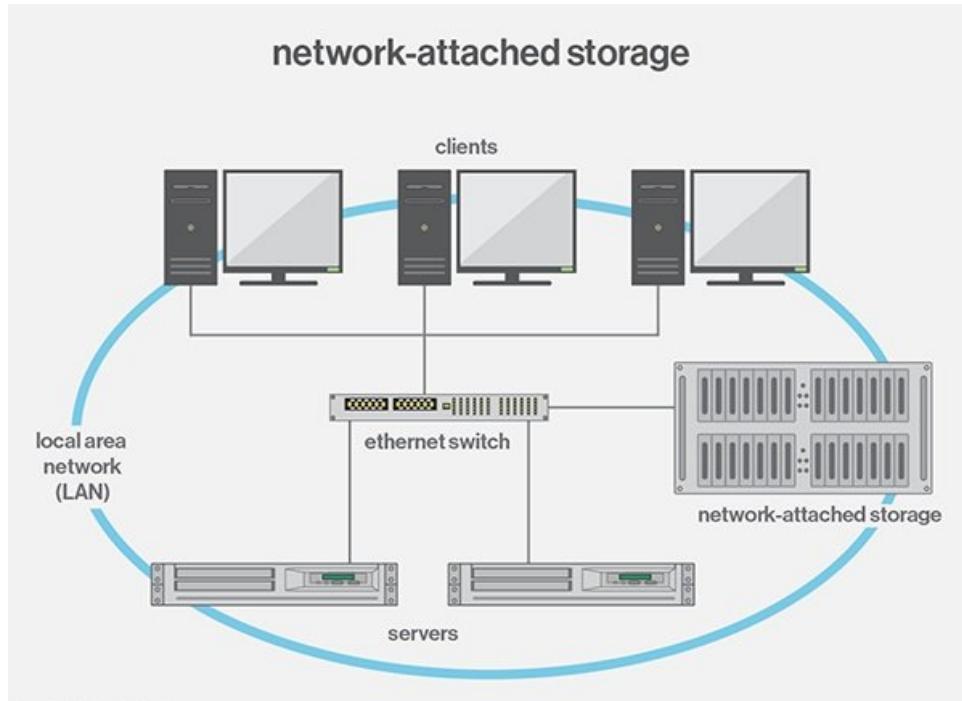


## Cable

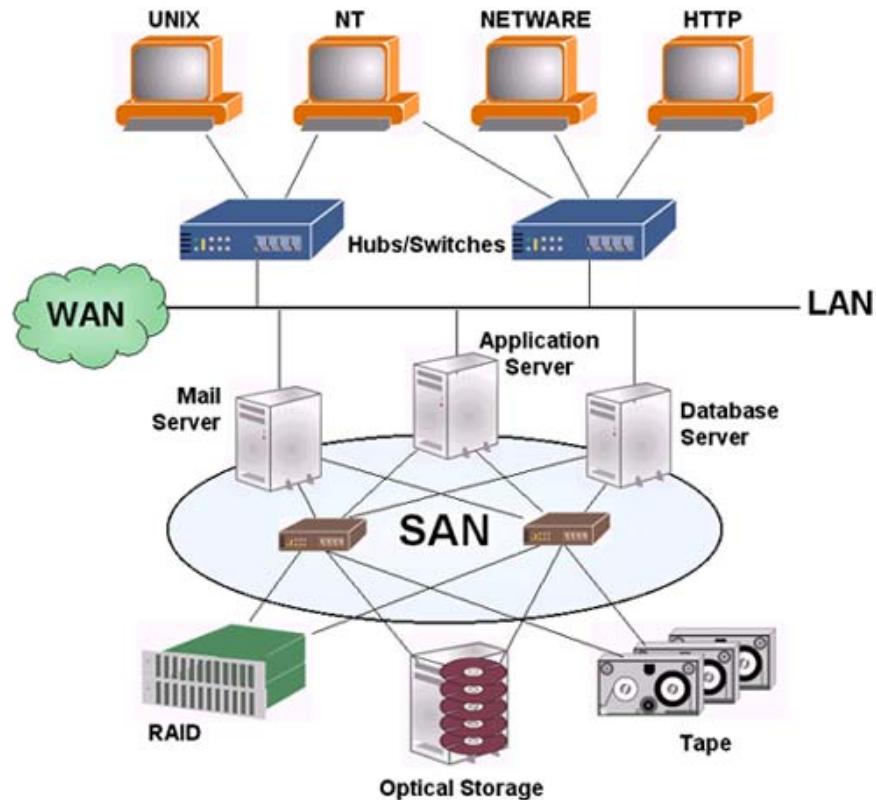
Category	Standard Bandwidth	Max Data Rate	Shielding
<b>Cat5e</b>	100MHz (up to 350)	1000Mbps	UTP or STP
<b>Cat6</b>	250MHz (up to 550)	1000Mbps	UTP or STP
<b>Cat6A</b>	500MHz (up to 550)	10Gbps	UTP or STP
<b>Cat7</b>	600MHz	10Gbps	Shielded only
<b>Cat8</b>	2000MHz	25Gbps or 40Gbps	Shielded only



## NAS vs SAN



## ① Storage Area Networks



Source: allSAN Report 2001

Copyright © 2000 allSAN.com Inc



## SAN components

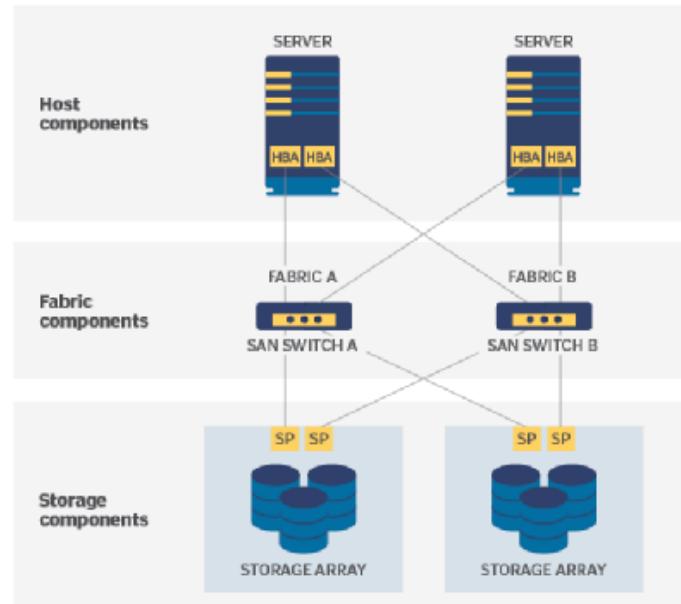
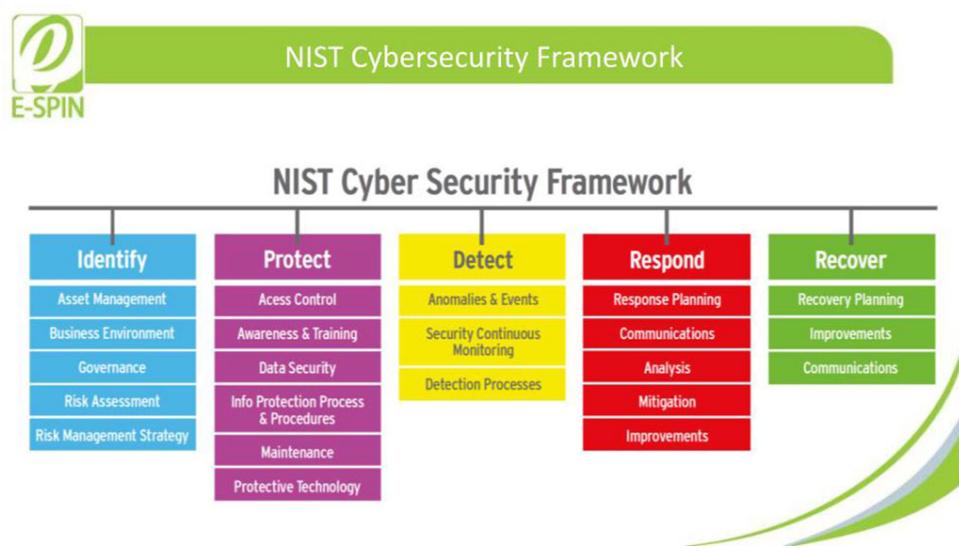
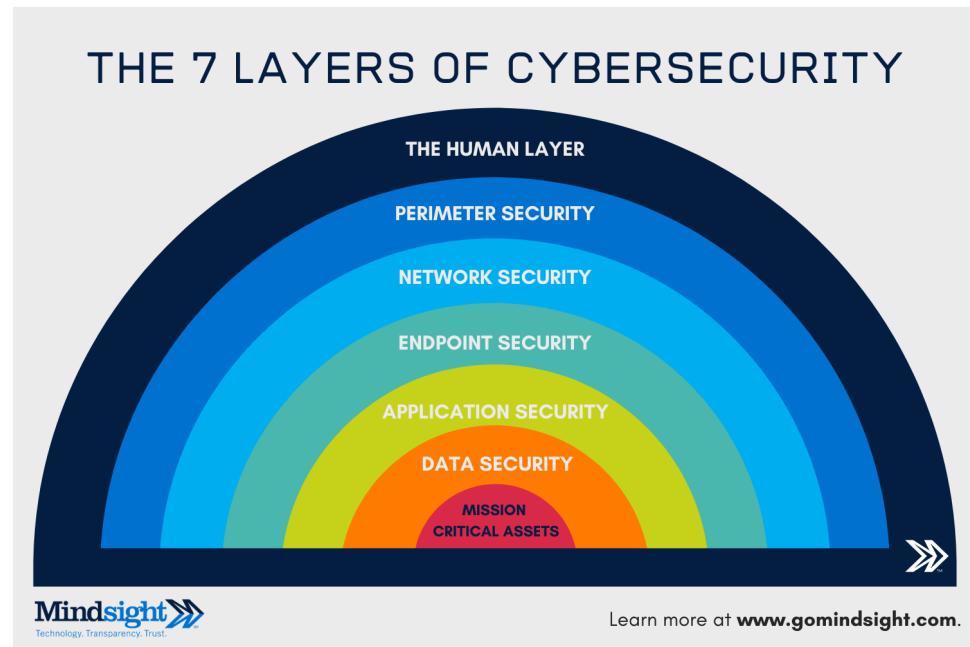
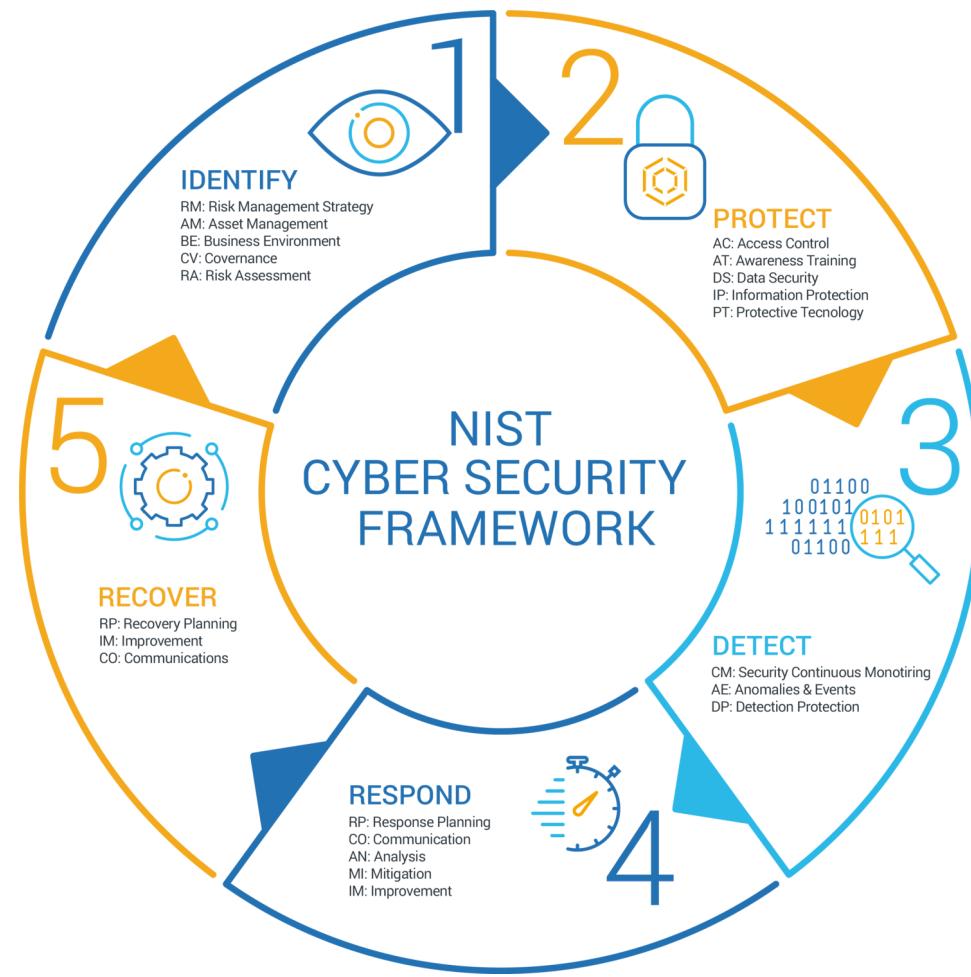


ILLUSTRATION: MAGLARA/ANDOE STOCK

©2006 TECHTARGET. ALL RIGHTS RESERVED.  
TechTarget

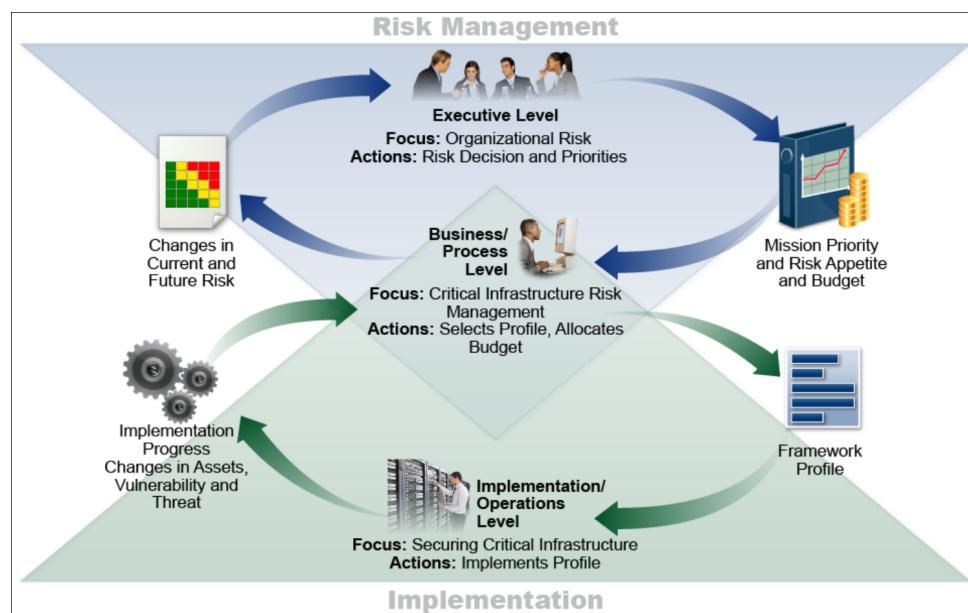
# CyberSecurity

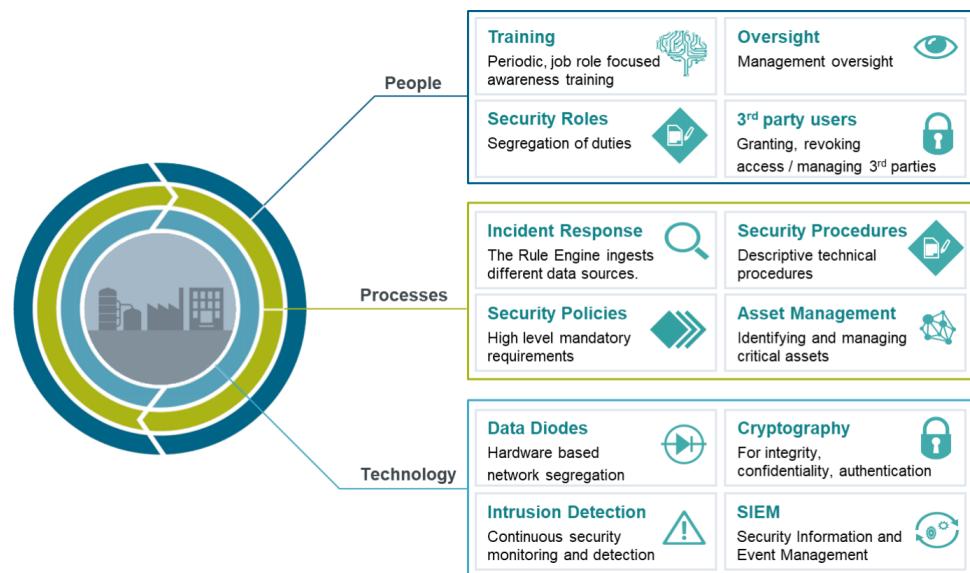




## Risk Management

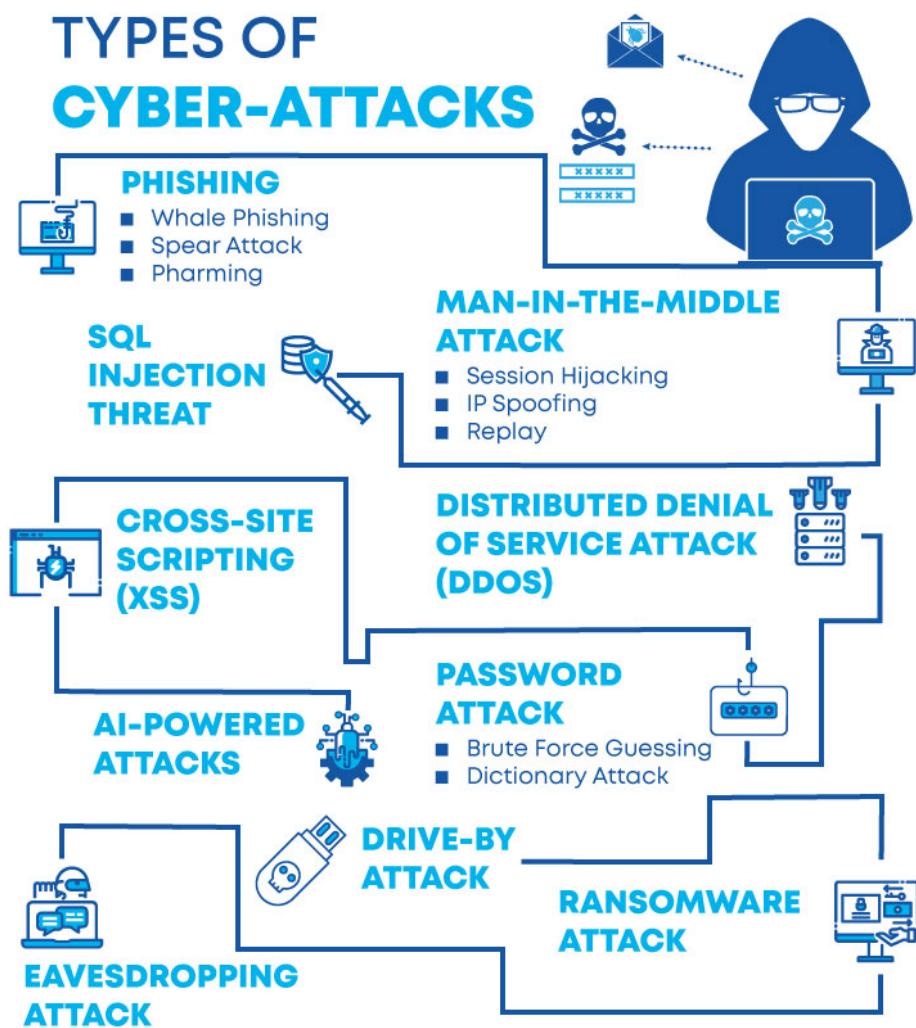
# BEST CYBERSECURITY PRACTICES

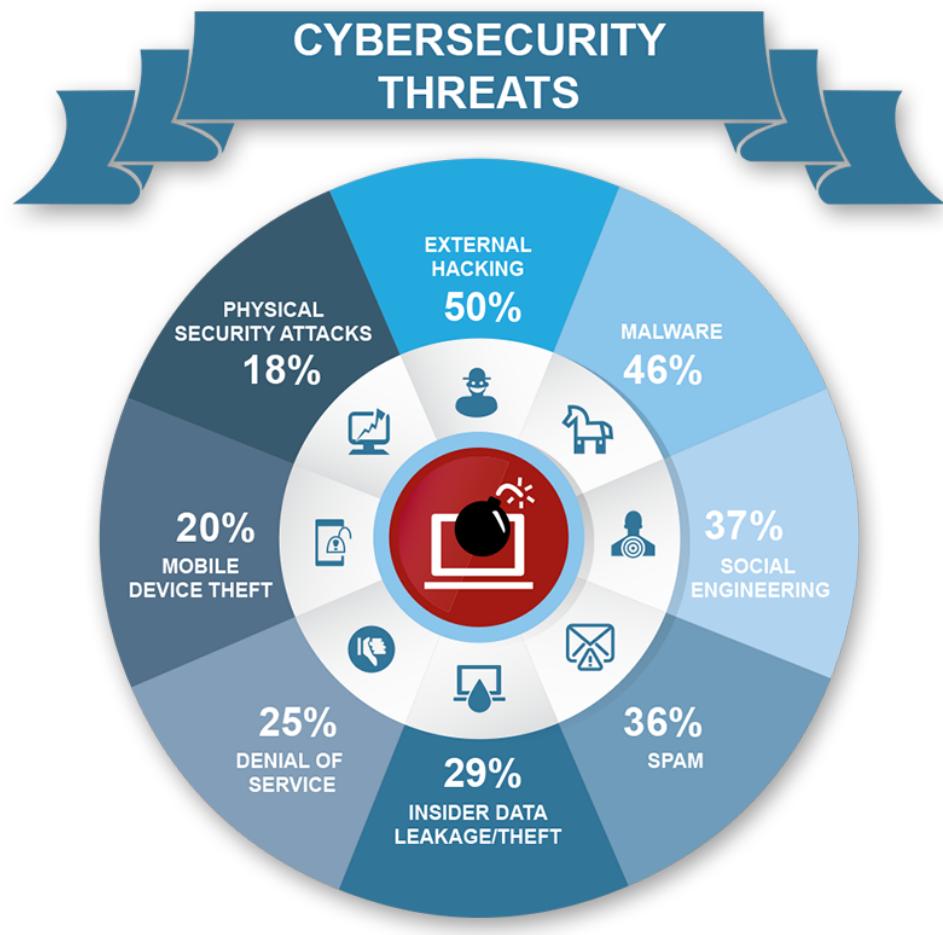




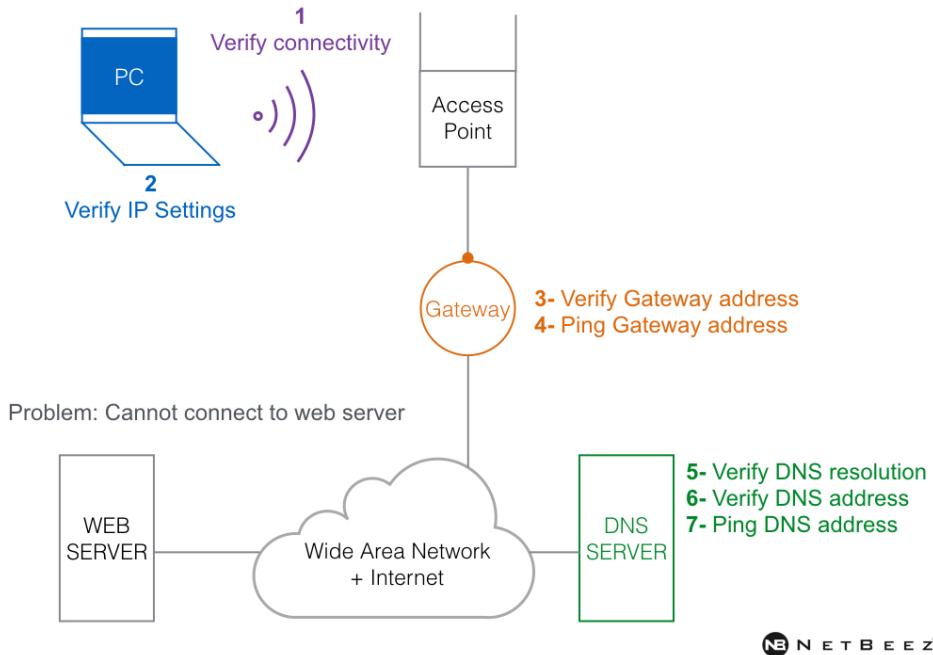


## Cybersecurity Attacks



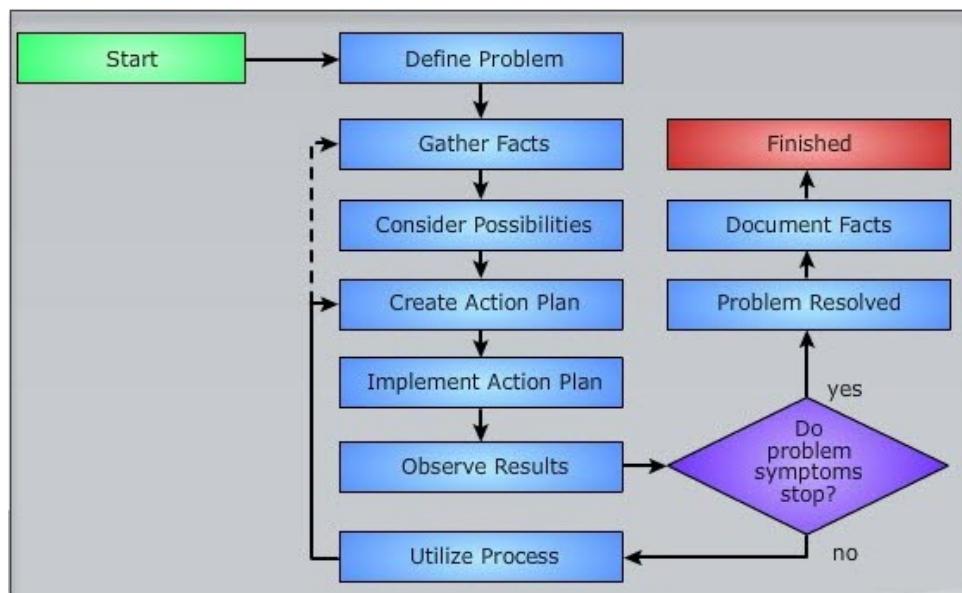
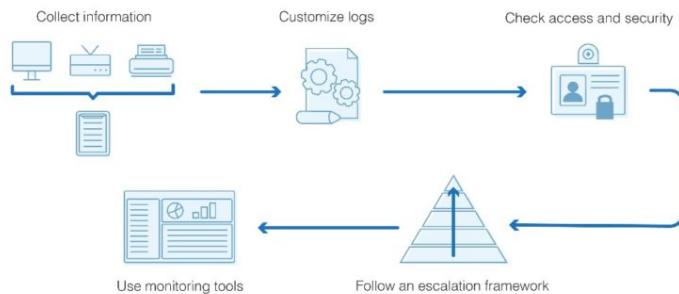


## Network Troubleshooting



NETBEEZ

### Network Troubleshooting Flowchart



## Troubleshooting Strategy

CompTIA Network+  
PowerCert



**Troubleshooting *STRATEGY***

1. Identify the symptoms and potential causes.

- ✓ Gather information about the problem.
- ✓ What is the problem?
- ✓ When did the problem occur?
- ✓ Specific error messages.
- ✓ Does the problem happen all the time or intermittently?

PowerCert



CompTIA Network+  
PowerCert



**Troubleshooting *STRATEGY***

2. Identify the affected area.

- Is the problem isolated or spread across several locations?
  - If the problem affects everyone
    - ✓ Check the switch.
  - If the problem is isolated.
    - ✓ Check the individual cable.

PowerCert



CompTIA Network+  
PowerCert



**Troubleshooting *STRATEGY***

3. Establish what has changed.

- ✓ Did anything change just prior to the problem happening?
- ✓ Was there any hardware removed or added?
- ✓ Was there any software installed or uninstalled?
- ✓ Was anything downloaded from the internet?

PowerCert



CompTIA Network+  
PowerCert

## Troubleshooting *STRATEGY*

4. Select the most probable cause.

- ✓ Look for simple solutions first.
- ✓ Does the device have power?
- ✓ Are the cables plugged in?
- ✓ Check the LEDs.

PowerCert

PowerCert

CompTIA Network+  
PowerCert

## Troubleshooting *STRATEGY*

5. Implement an action plan and solution including potential effects.

- ✓ The cautious phase.
- ✓ Must know what effect the action will have on the network.
- ✓ Will it affect the entire network or be isolated at one area?

PowerCert

PowerCert

CompTIA Network+  
PowerCert

## Troubleshooting *STRATEGY*

6. Test the result.

- ✓ Where you take action to solve the problem.
- ✓ Where you will know if your plan of action will solve the problem or not.

PowerCert

PowerCert

CompTIA Network+  
PowerCert



**Troubleshooting *STRATEGY***

7. Identify the results and effects of the solution.

- ✓ Has your plant of action solved the problem or not?
- ✓ What effect did it have on everyone else?
- ✓ Do the results show a temporary fix or a permanent one?

PowerCert



CompTIA Network+  
PowerCert



**Troubleshooting *STRATEGY***

8. Document the solution and process.

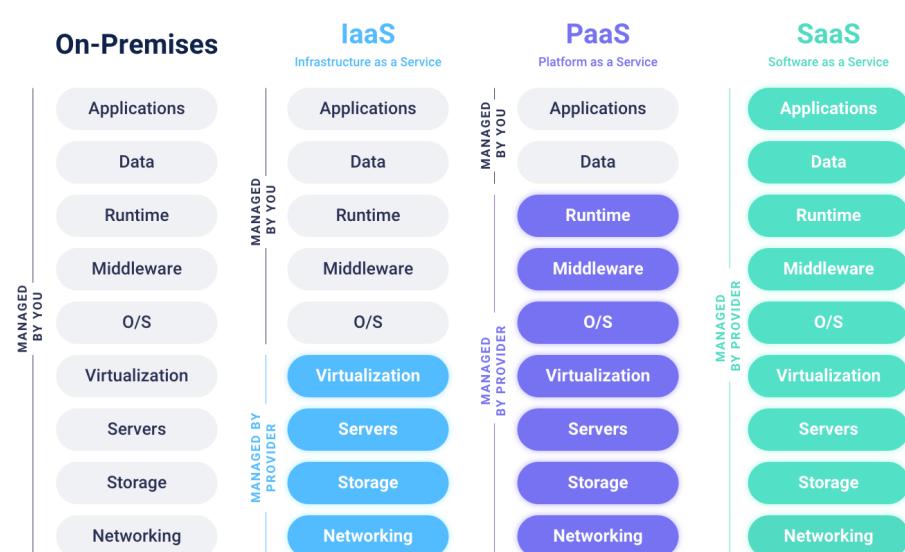
- ✓ Document the problem.
- ✓ Document what caused the problem.
- ✓ Document how the problem was fixed.

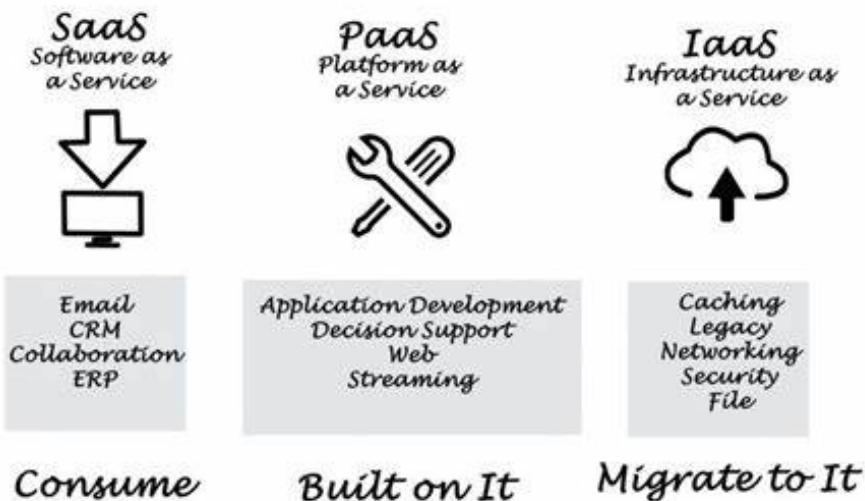
**Network Administrator**

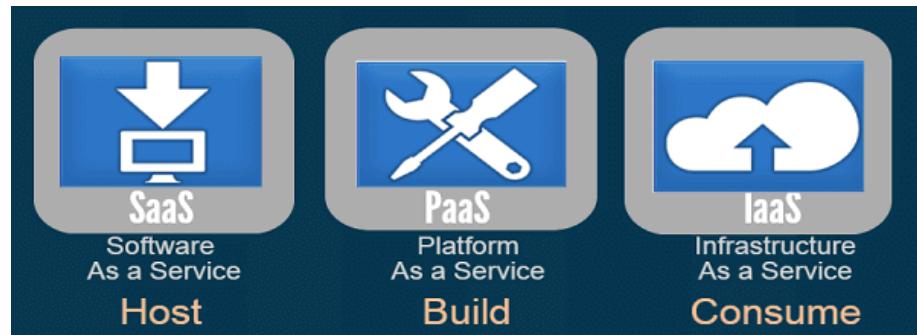
PowerCert



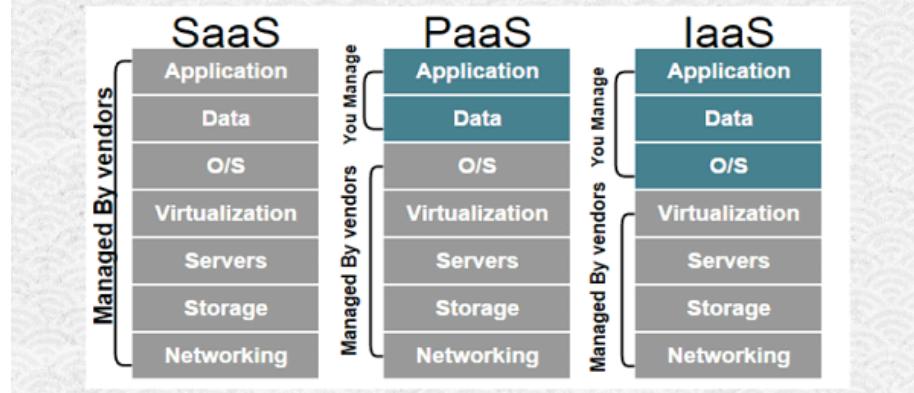
## Cloud Computing - IaaS Paas Saas



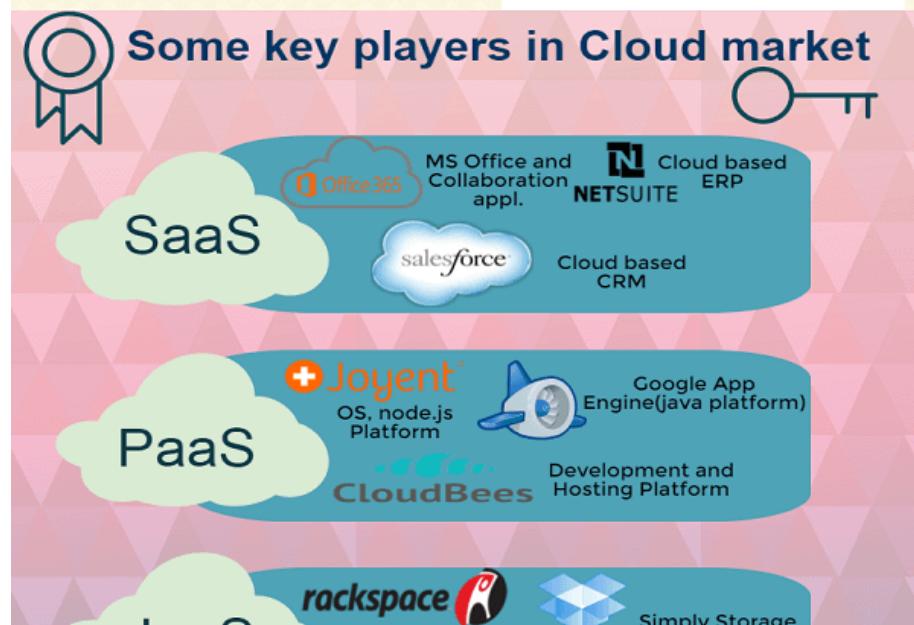
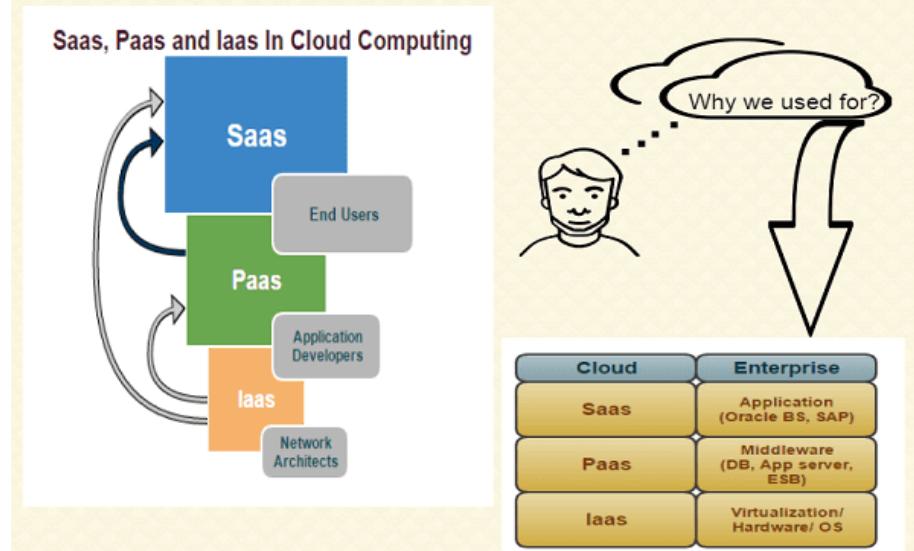




### Difference between SaaS, PaaS and IaaS



### How Structured in Cloud Computing?





-- Memo End --

