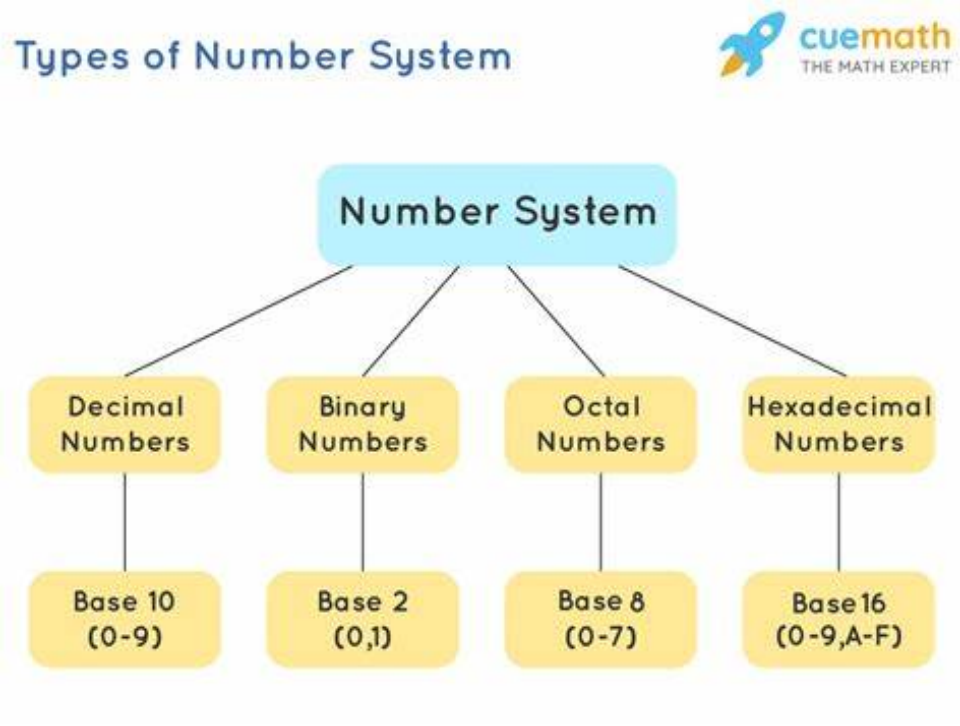
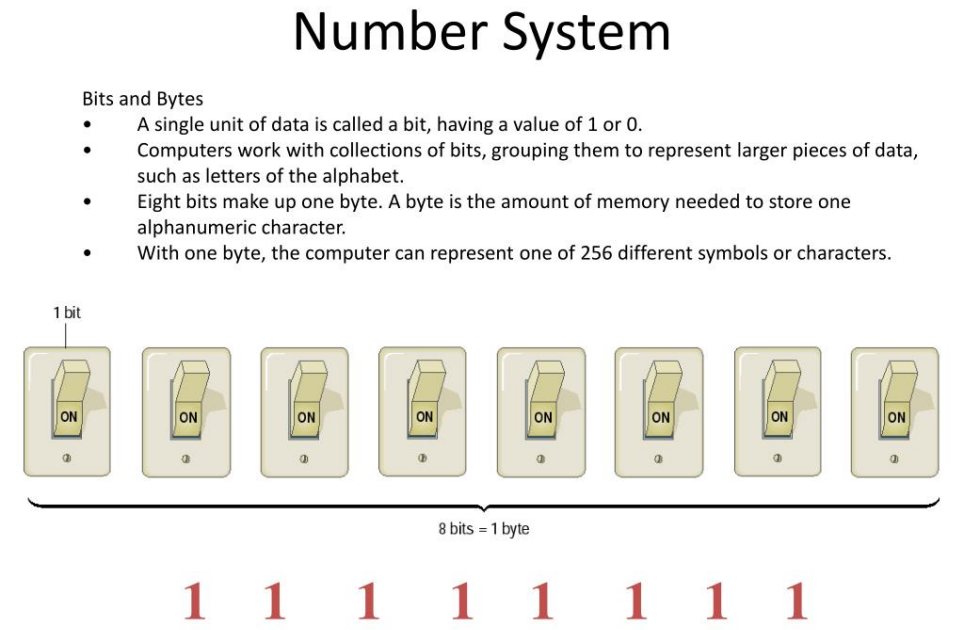
Computer Concept

Number System



Binary Number System



Base Conversion

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

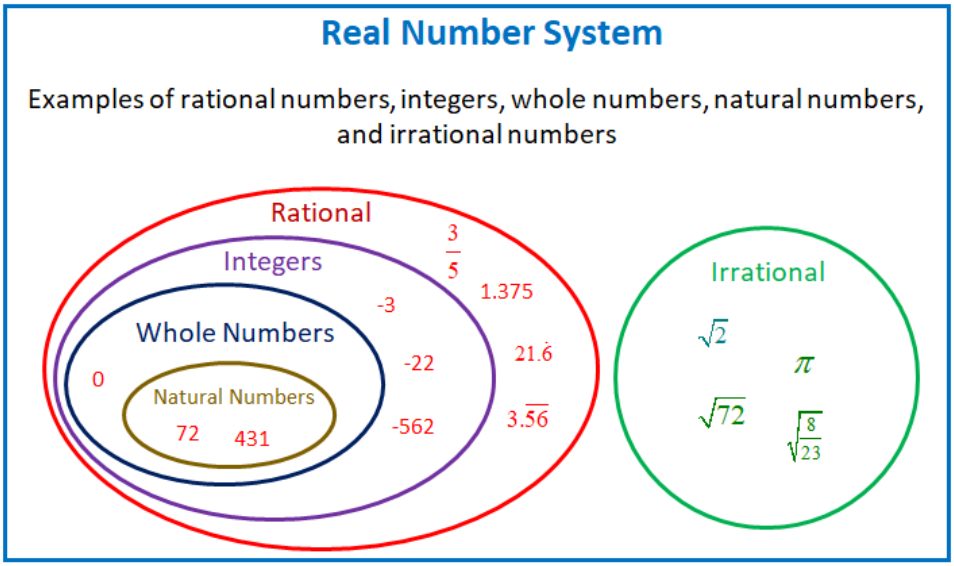
Binary, Hex, and Octal Conversions

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

1. 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.
2. 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.
3. 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.

Real Number System



Complements

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 $(10^n-1)-N$

•$99999\ldots\ldots - N$

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

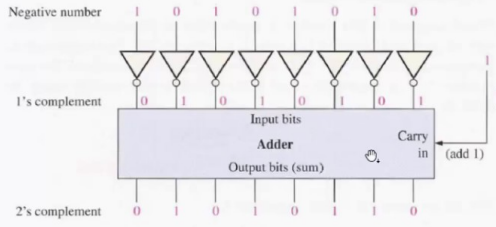
2.5) 1 'S and 2'S Complements of Binary Numbers

Finding the 2 's Complement

The 2's complement of a binary number is found by adding 1t o the LSB of the 1's complement.

2's complement = (1's complement) + 1

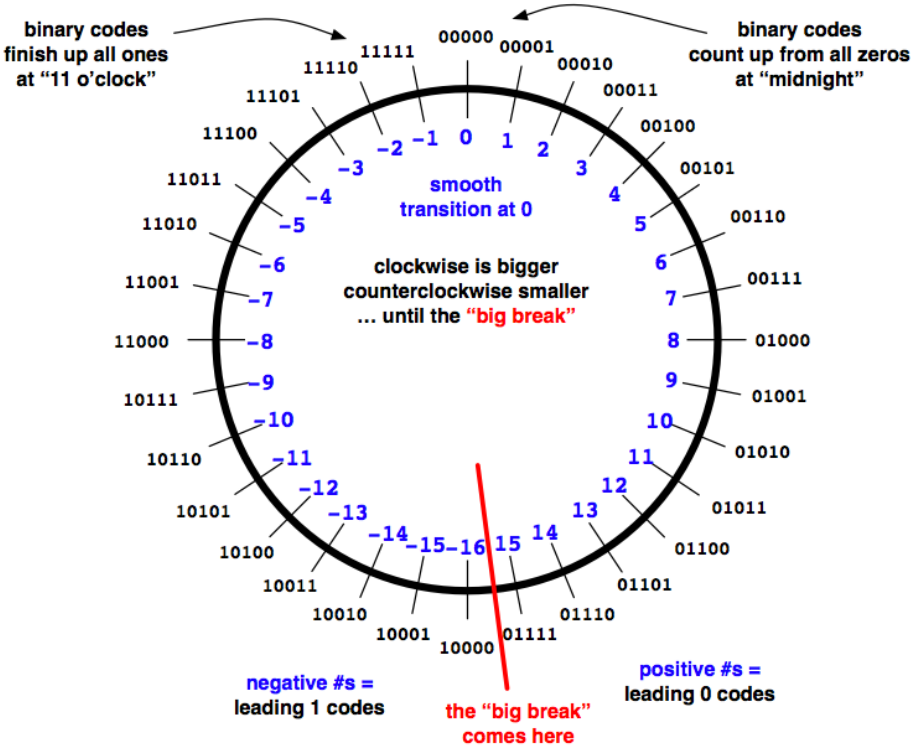
Example 17: Find the 2's Complement of 10110010



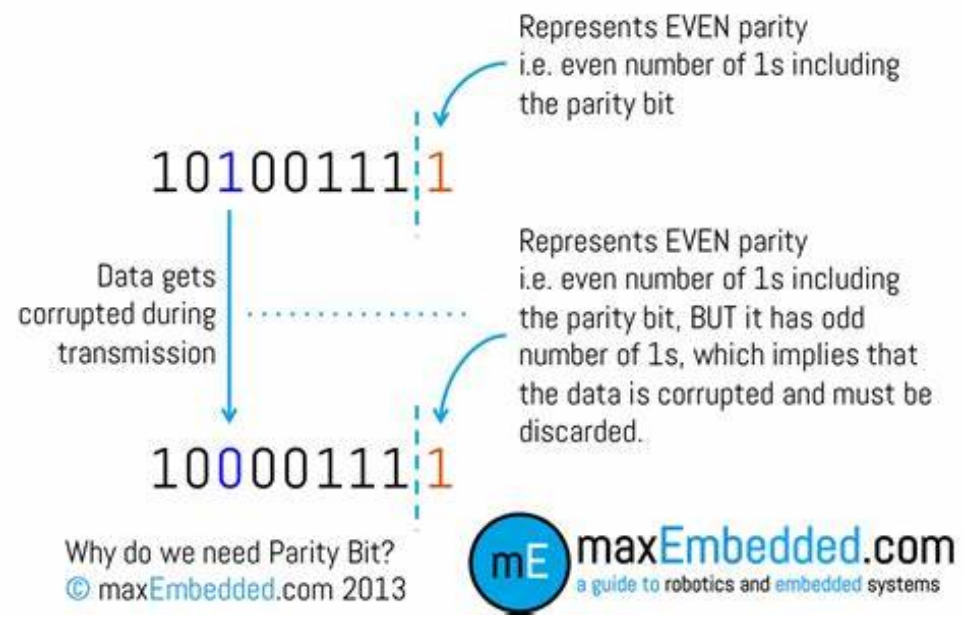
10110010 Binary number
01001101 1's complement
+ 1 Add 1
01001110 2's complement

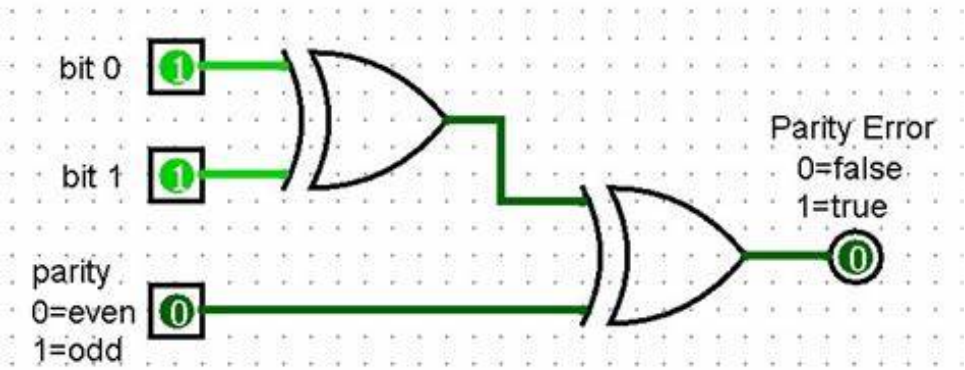
Source: Digital Fundamentals by Thomas L. Floyd.

27


binary codes
finish up all ones
at "11 o'clock"
binary codes
count up from all zeros
at "midnight"
smooth
transition at 0
clockwise is bigger
counterclockwise smaller
... until the "big break"
negative #s =
leading 1 codes
positive #s =
leading 0 codes
the "big break"
comes here

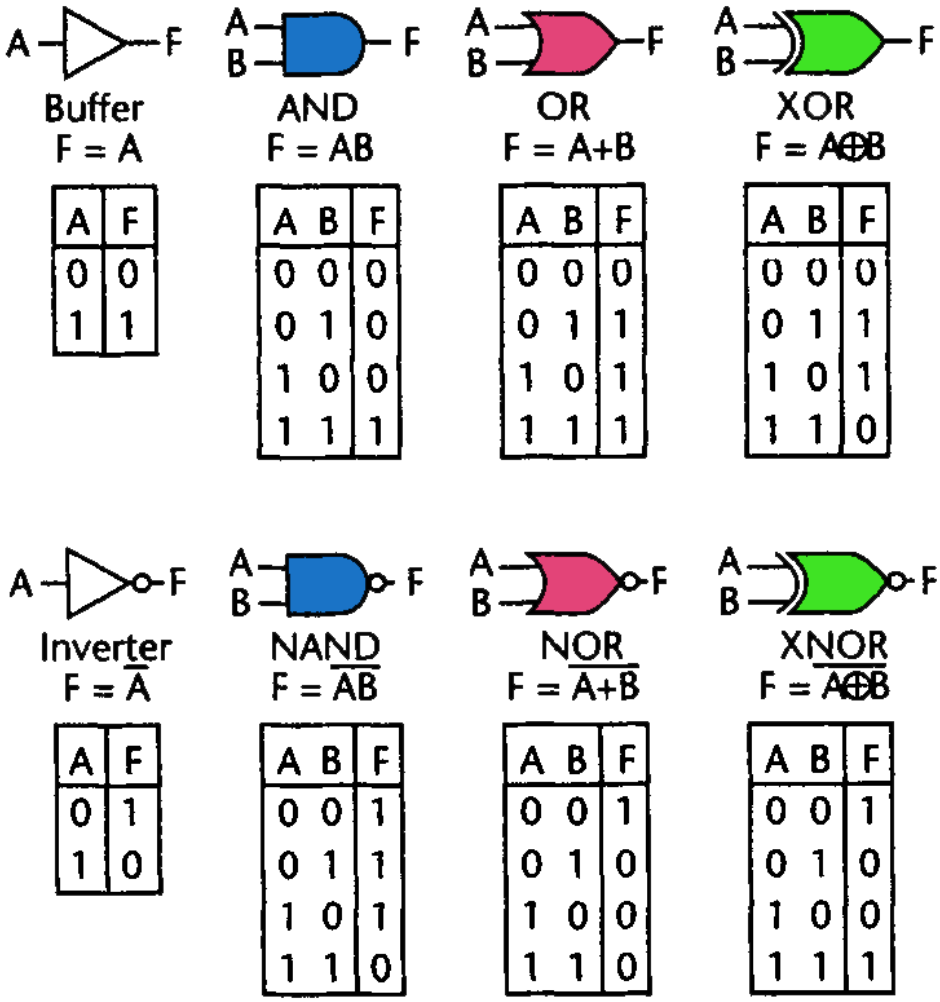
Parity Bit


Represents EVEN parity
i.e. even number of 1s including
the parity bit
10100111 1
Data gets
corrupted during
transmission
Represents EVEN parity
i.e. even number of 1s including
the parity bit, BUT it has odd
number of 1s, which implies that
the data is corrupted and must be
discarded.
10000111 1
Why do we need Parity Bit?
© maxEmbedded.com 2013
maxEmbedded.com
a guide to robotics and embedded systems


bit 0
bit 1
parity
0=even
1=odd
Parity Error
0=false
1=true

Logic Design

Logic Gates



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	1

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

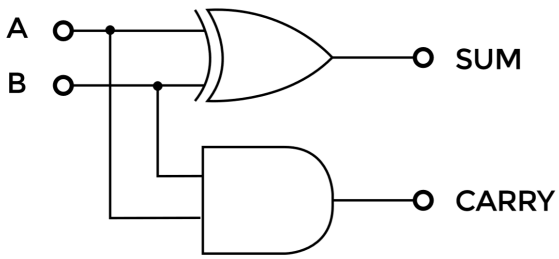
A	F
0	1
1	0

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

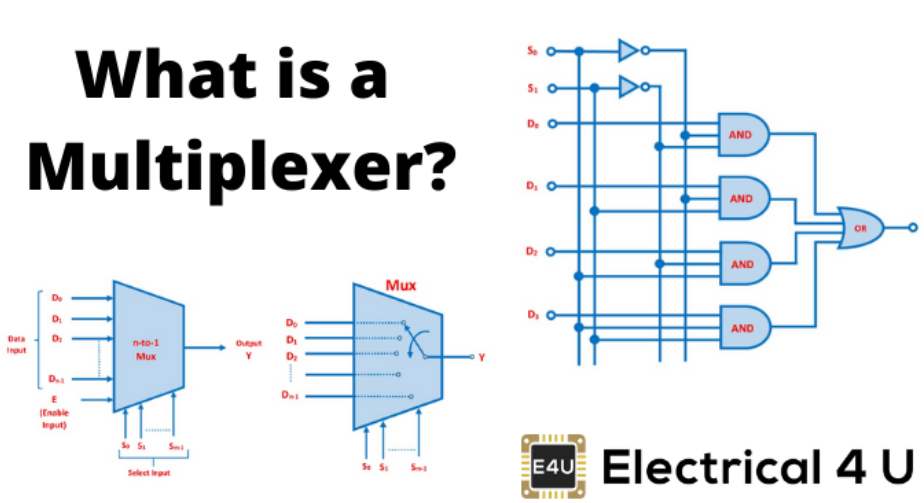
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Half-Adder

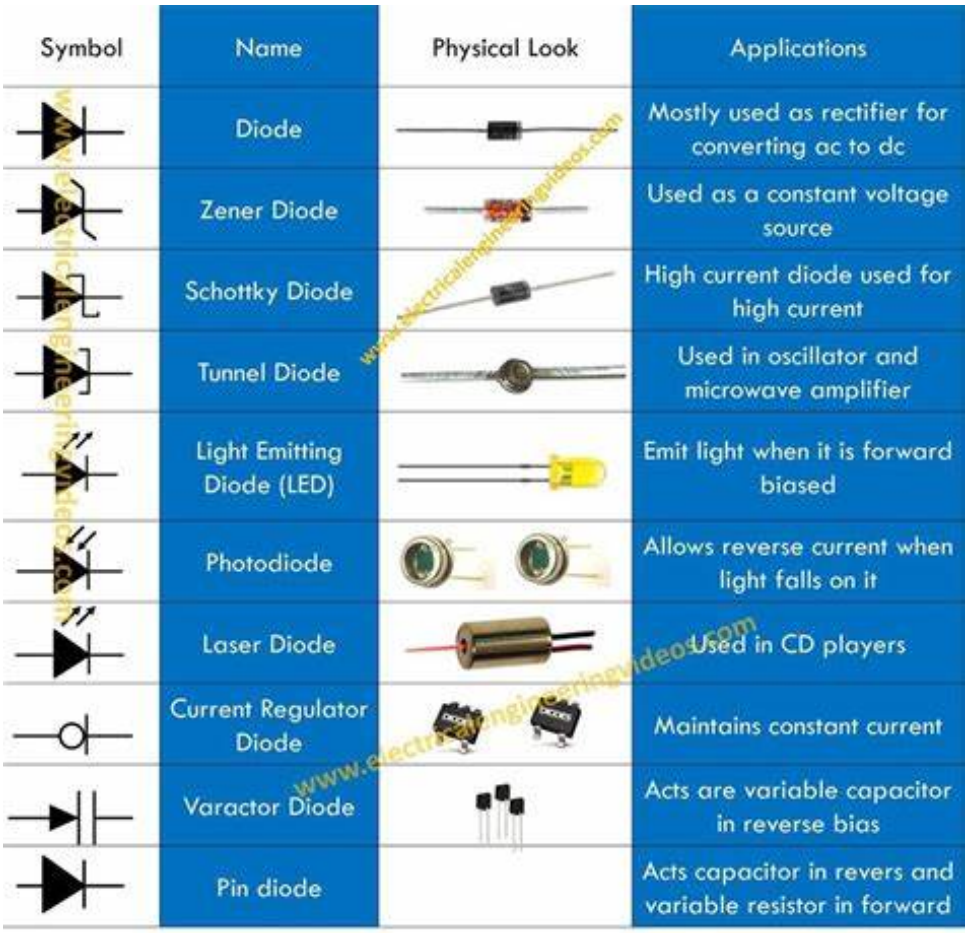


A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Multiplexer



Diodes



Data Presenstation and Processing

Machine Language and Assembly 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:

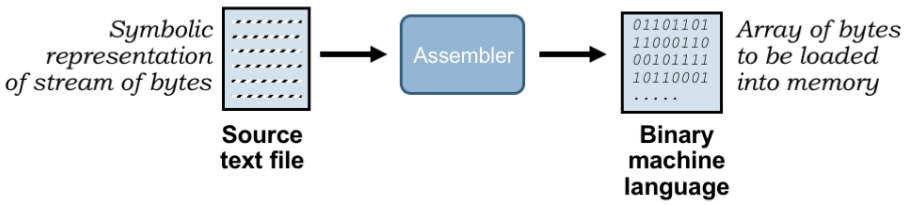
000000	10001	10010	01000	00000	100000
op	rs	rt	rd	shamt	funct

- **Can you guess what the field names stand for?**

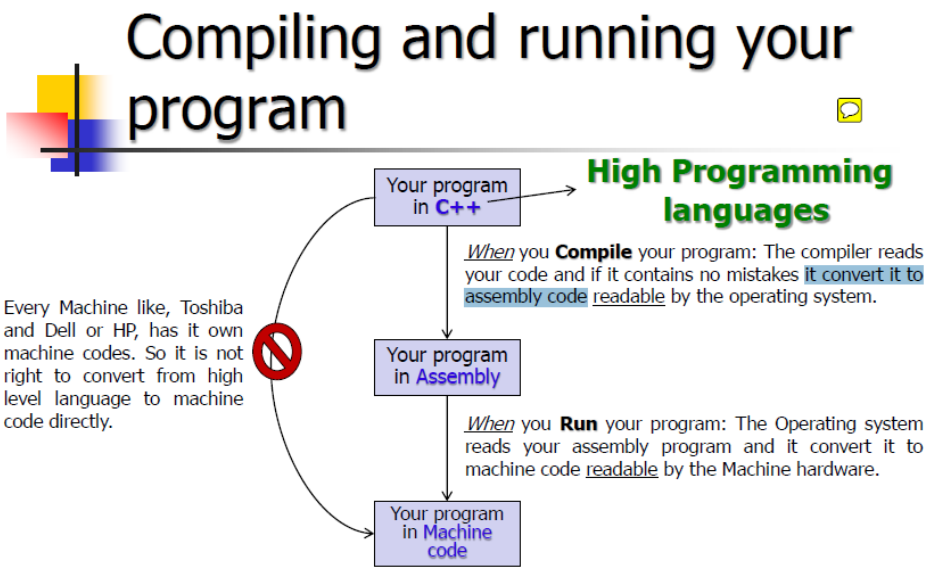
op = Basic operation of the instruction: opcode
rs = The first register source operand
rt = The second register source operand
rd = The register destination operand
shamt = shift amount
funct = function code

25

Assembly Language



- Abstracts bit-level representation of instructions and addresses
- We'll learn UASM ("microassembler"), built into BSim
- Main elements:
  - Values
  - Symbols
  - Labels (symbols for addresses)
  - Macros



ASCII

ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

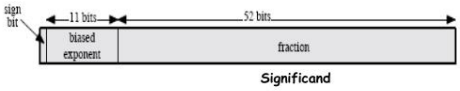
Precision floating-point

Double Precision Floating Point Numbers
IEEE Standard

64 bit Double Precision Floating Point Numbers are stored as:

S EEEEEEEEEEE FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

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 = 11111111111 in binary (11 bits)

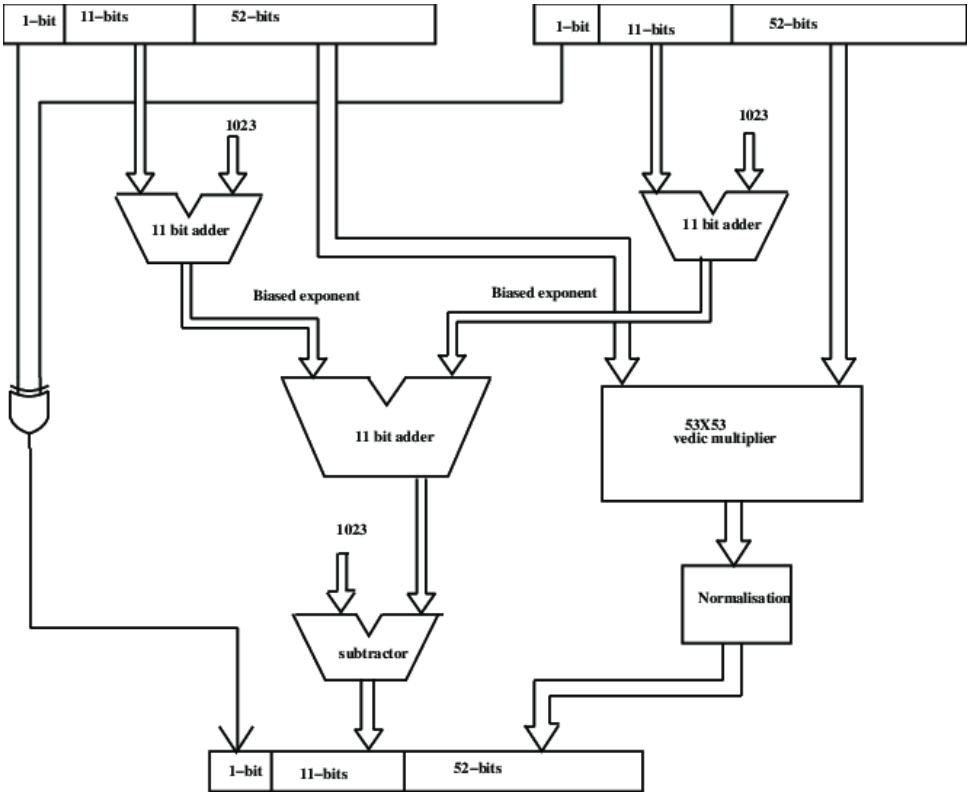
Biased Exponent - Cont'd

- 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

$$(-1)^S \times (1.F)_2 \times 2^{E-\text{Bias}}$$

$$(-1)^S \times (1.f_1f_2f_3f_4\ldots)_2 \times 2^{E-\text{Bias}}$$

$$(-1)^S \times (1 + f_1\times 2^{-1} + f_2\times 2^{-2} + f_3\times 2^{-3} + f_4\times 2^{-4}\ldots)_2 \times 2^{E-\text{Bias}}$$



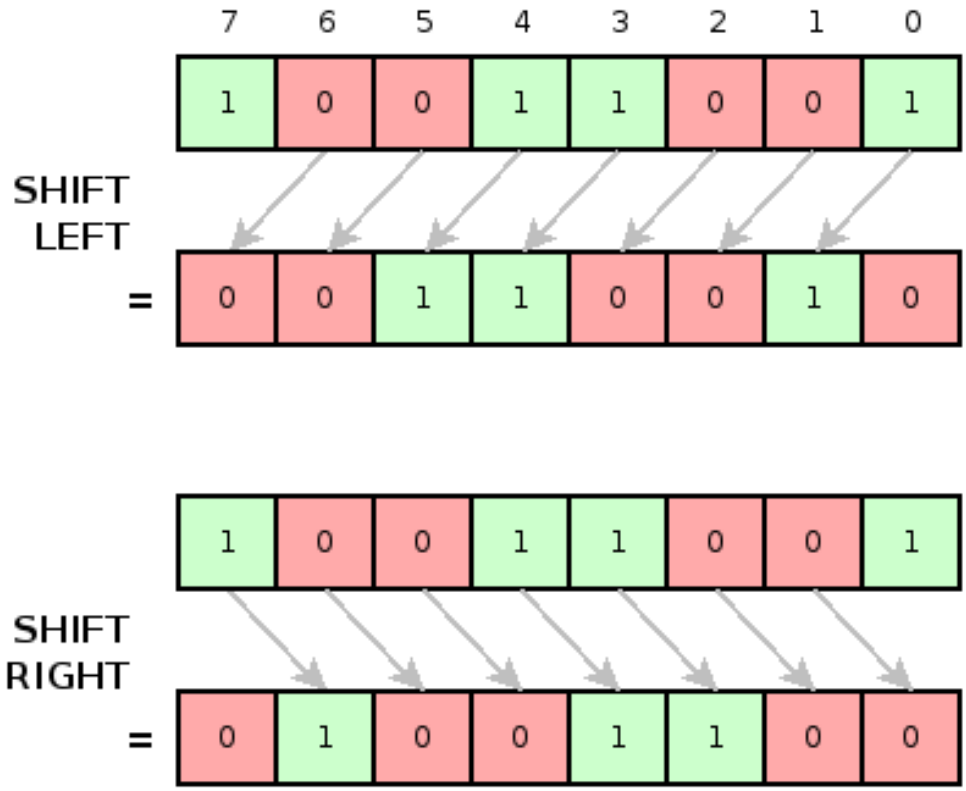
Bitwise Operation

Bitwise Operators

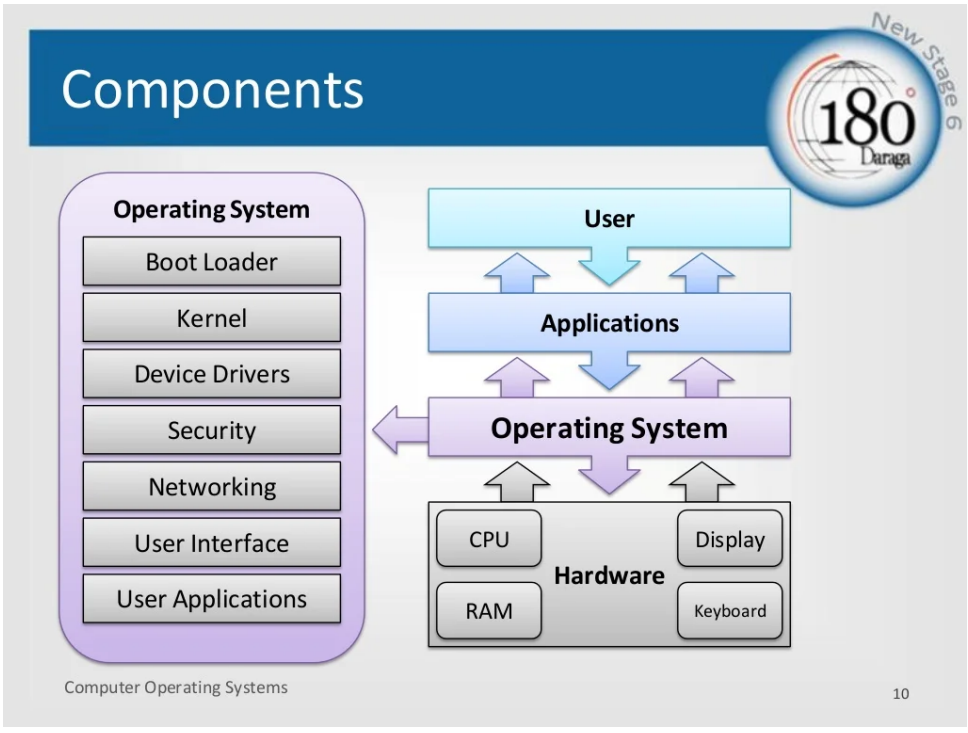
int a = 10, b = 2 for all examples below

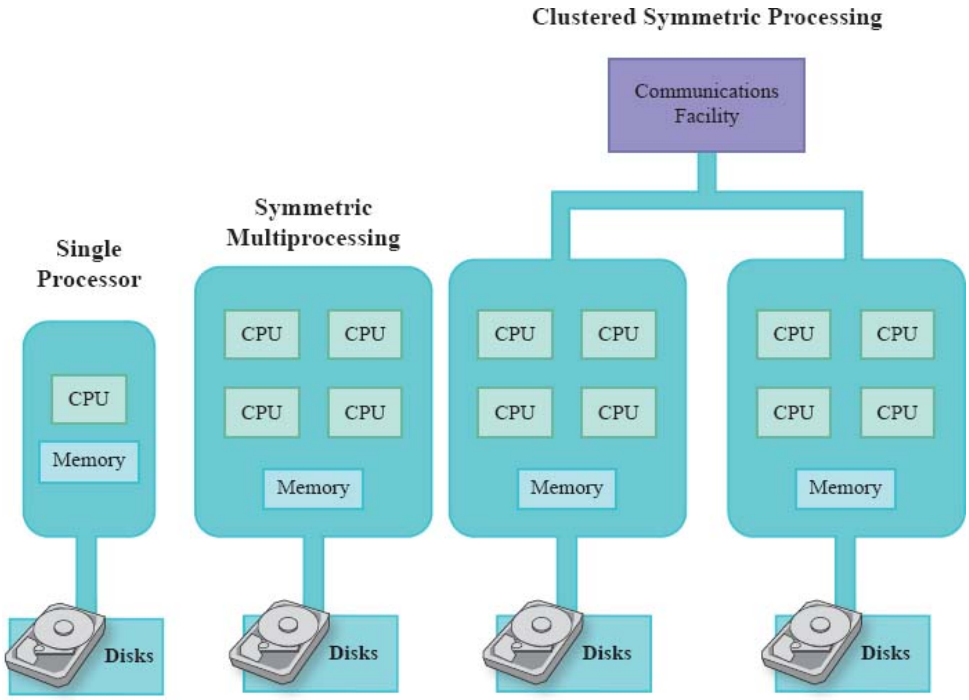
Operator	Meaning	Example	Result
~	Bitwise unary NOT	~a	-11
&	Bitwise AND	a&b	2
|	Bitwise OR	a|b	10
^	Bitwise Ex-OR	a^b	8
>>	Shift right	a>>1	5
>>>	Shift right zero fill	a>>>1	5
<<	Shift left	a<<1	20
&=	Bitwise AND assignment	a &= b	2
|=	Bitwise OR assignment	a |= b	10
^=	Bitwise Ex-OR assignment	a ^=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

Startertutorials.com

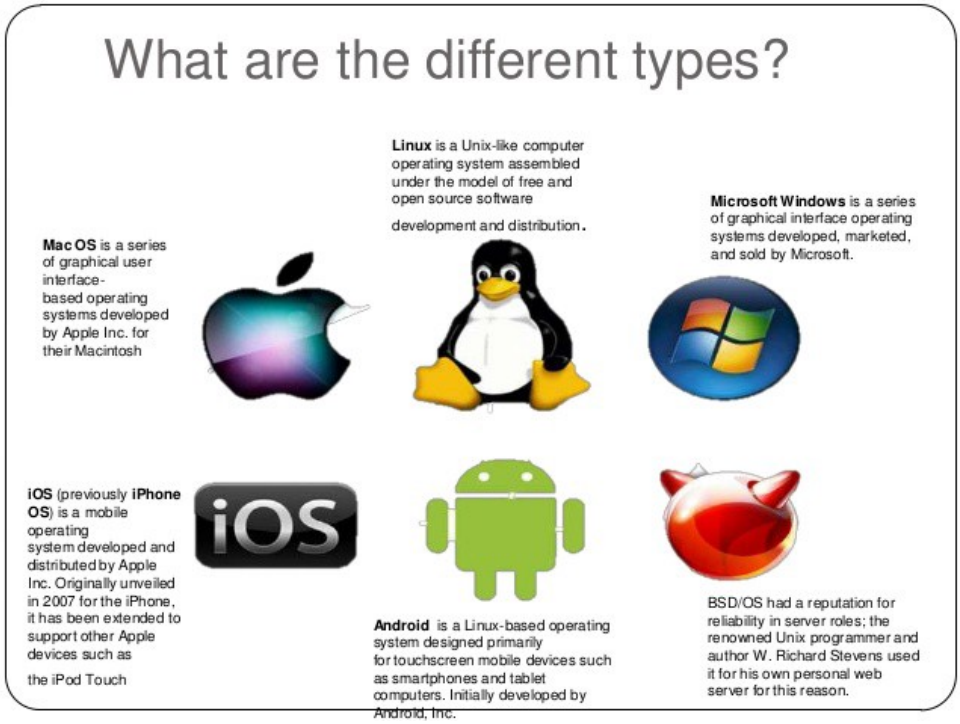


Operating System

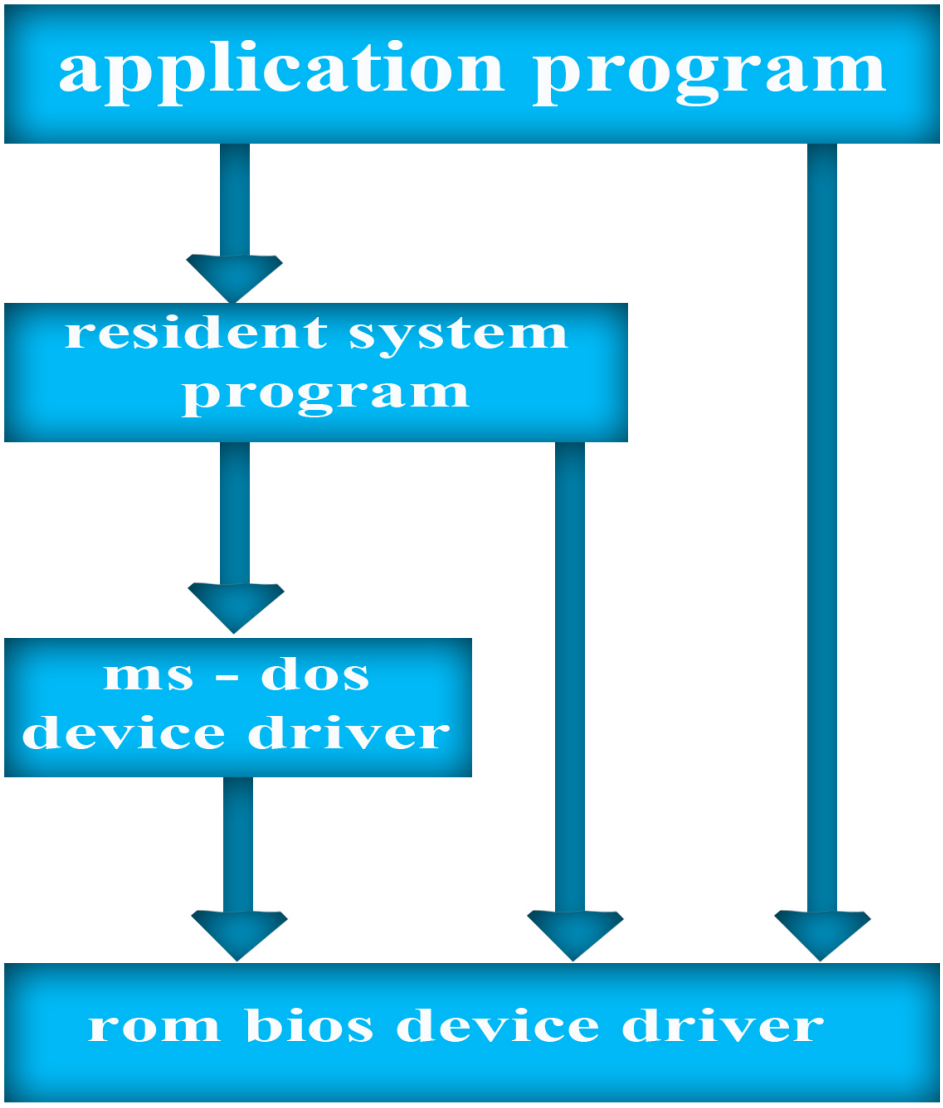




Types of OS



OS Structure



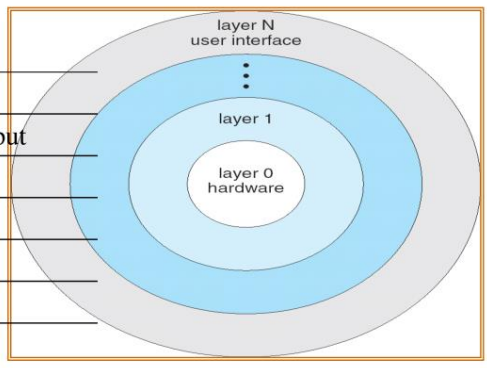
ms dos system structure

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



6

CIS410

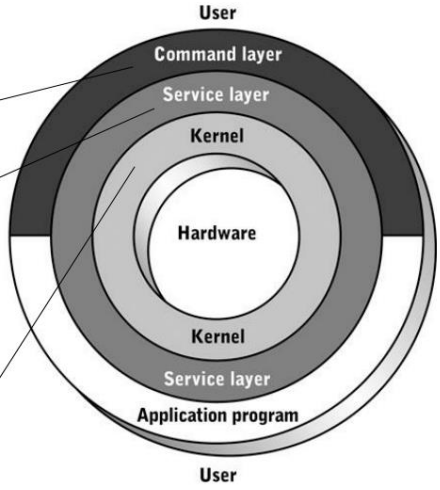
Operating System Layers

Figure 11-3 ►
Operating system layers (shaded)

User's interface to OS

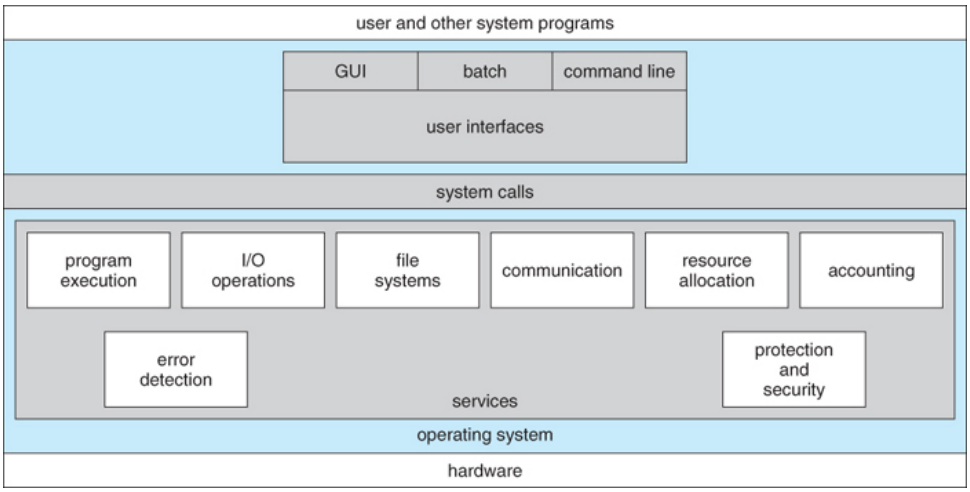
Contains set of functions executed by application programs and command layer

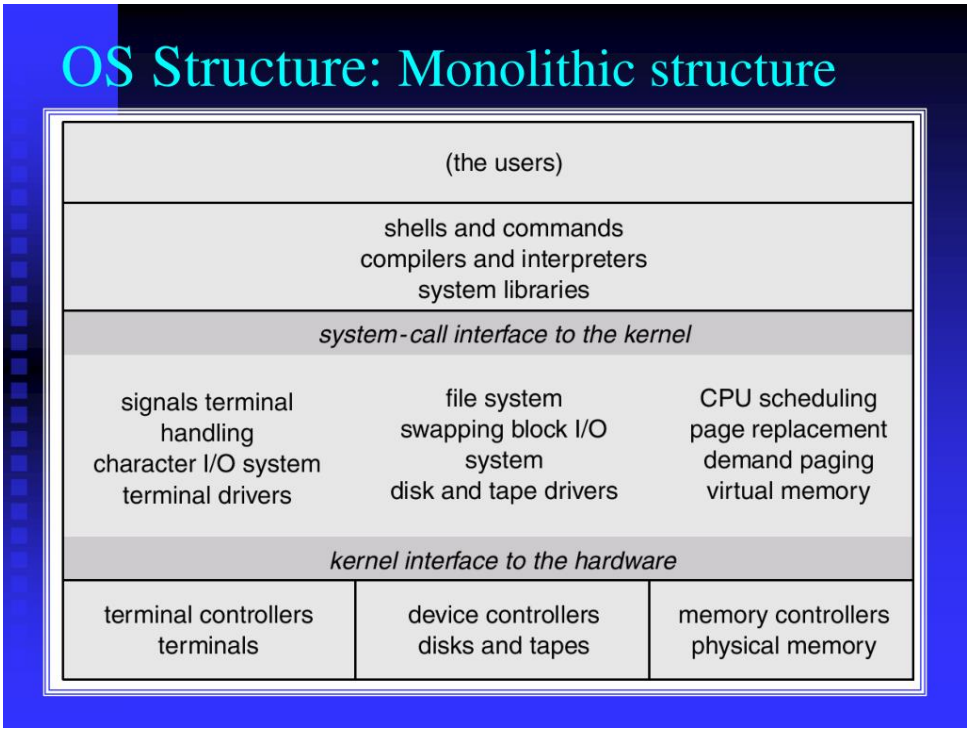
Manages resources; interacts directly with computer hardware



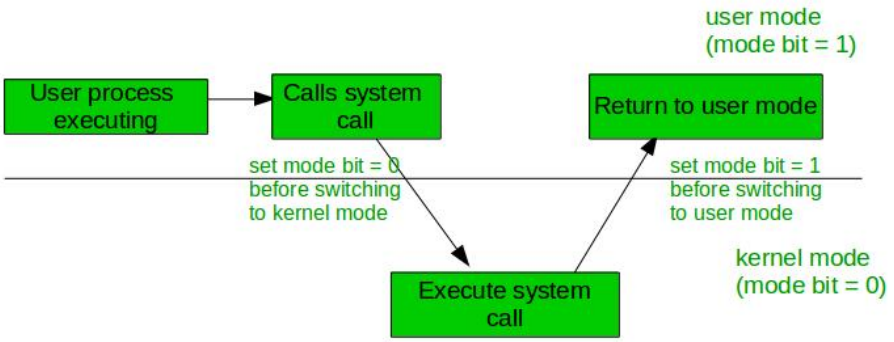
Hardware and Software Architecture

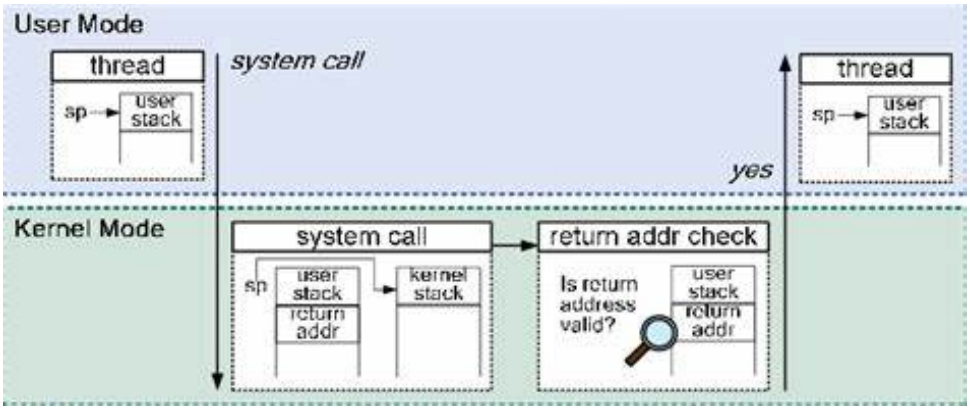
16



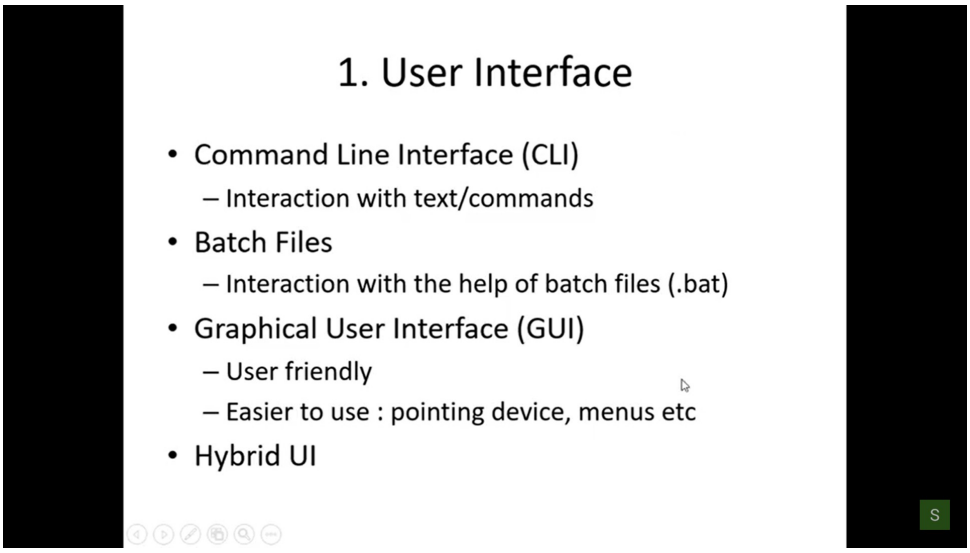


Dual Mode Operation

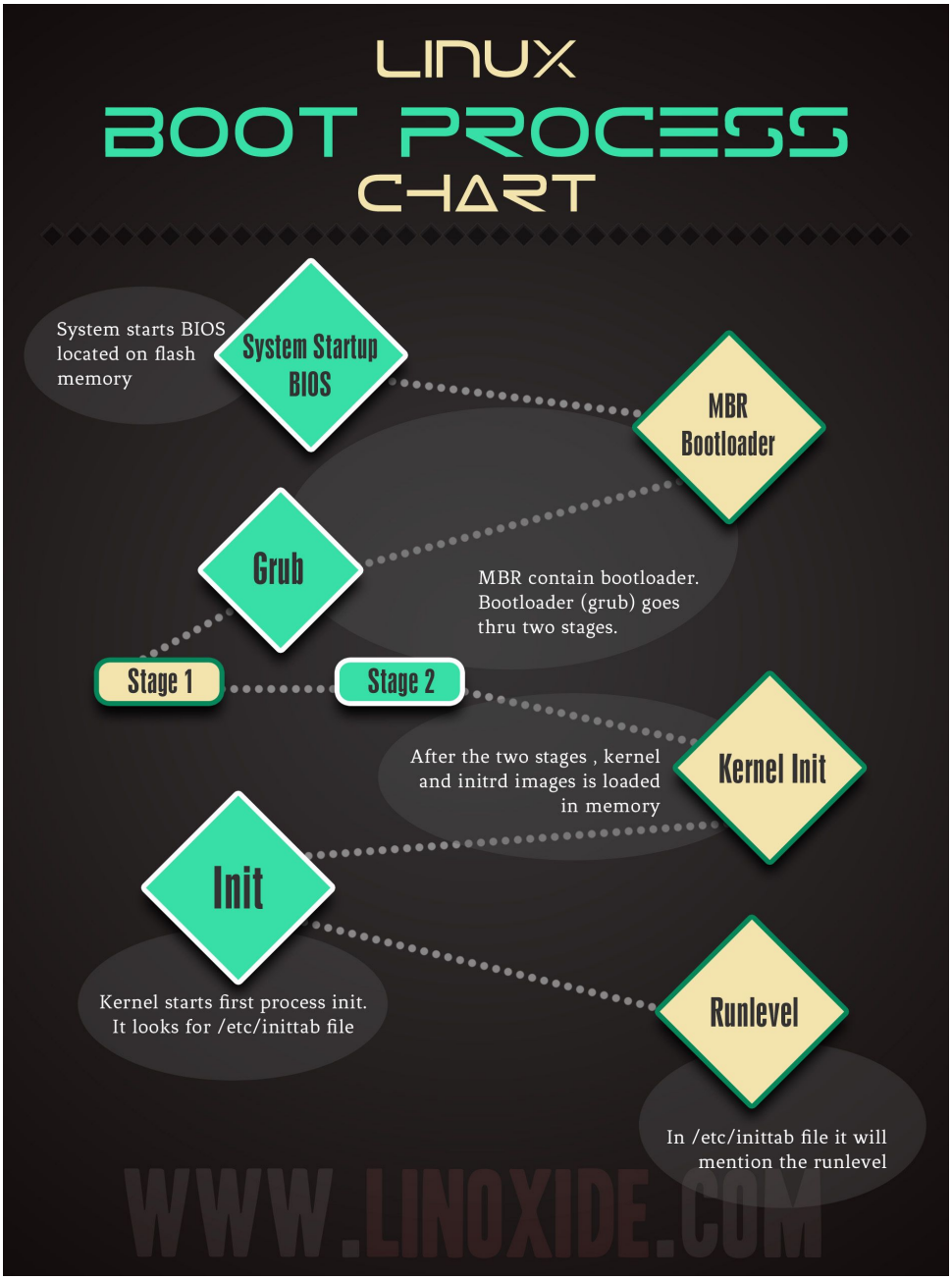


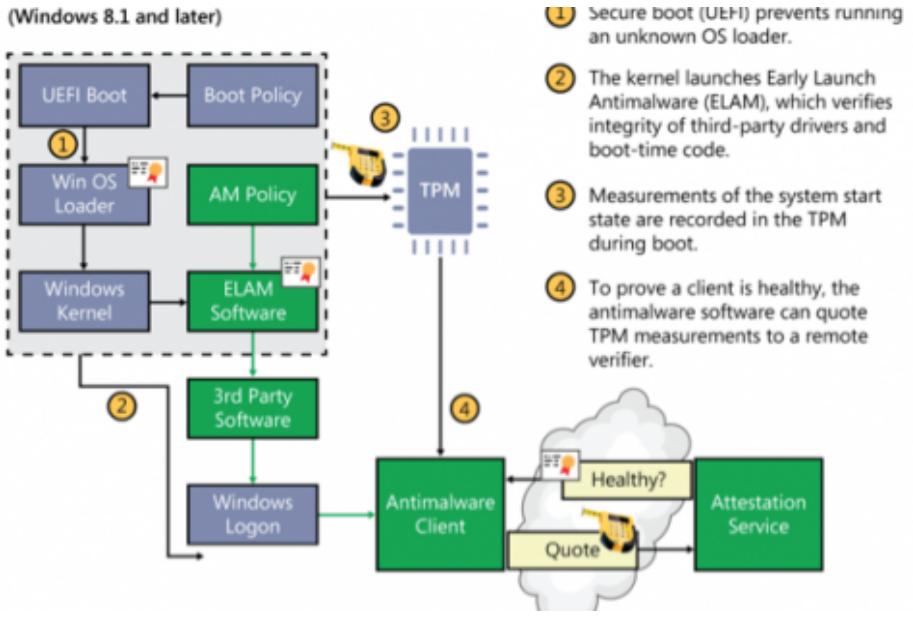


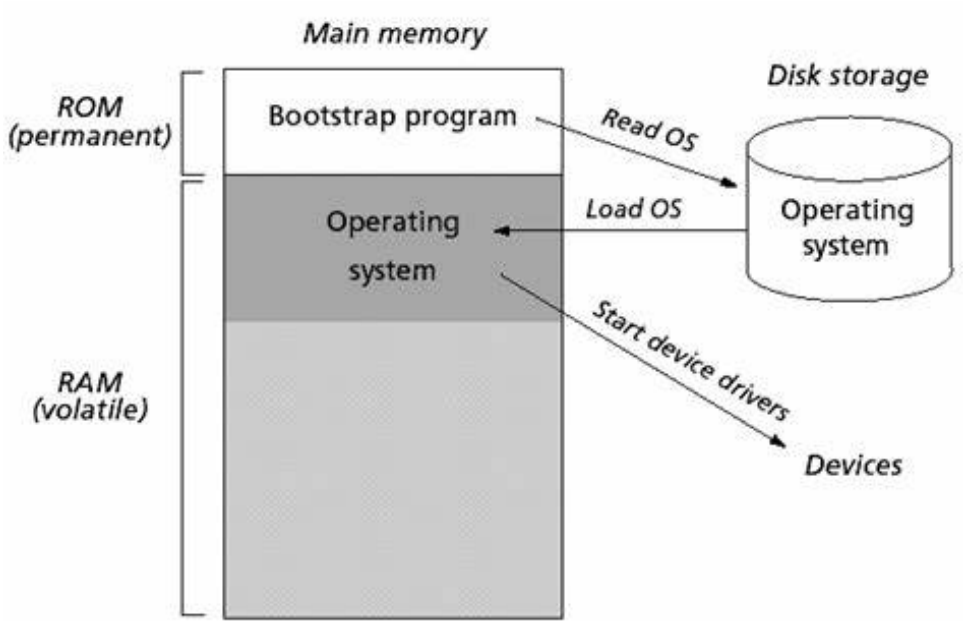
User Interface

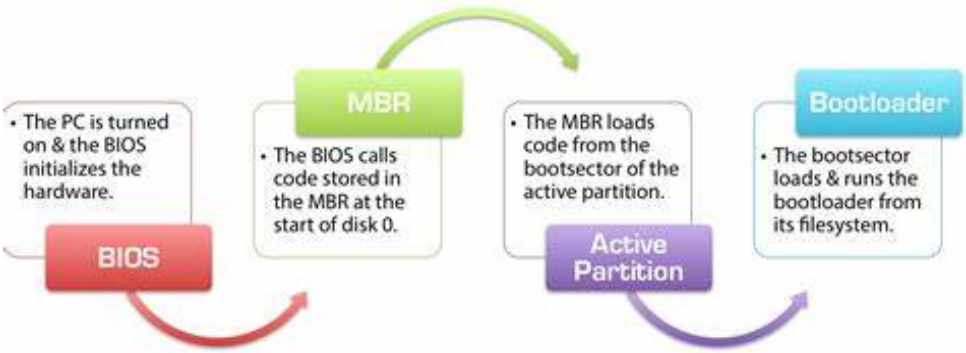


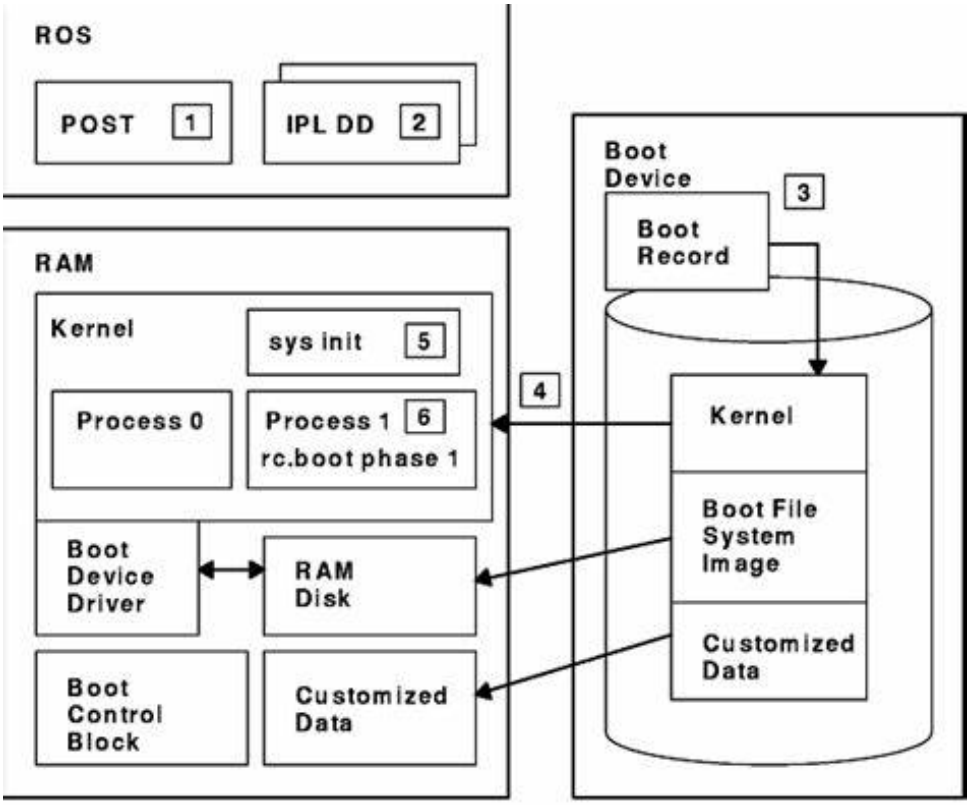
System Boot


LINUX
BOOT PROCESS
CHART
System starts BIOS located on flash memory
System Startup BIOS
MBR Bootloader
Grub
MBR contain bootloader. Bootloader (grub) goes thru two stages.
Stage 1
Stage 2
After the two stages , kernel and initrd images is loaded in memory
Kernel Init
Init
Kernel starts first process init. It looks for /etc/inittab file
Runlevel
In /etc/inittab file it will mention the runlevel
WWW.LINOXIDE.COM


(Windows 8.1 and later)
UEFI Boot
Boot Policy
Win OS Loader
AM Policy
TPM
Windows Kernel
ELAM Software
3rd Party Software
Windows Logon
Antimalware Client
Healthy?
Quote
Attestation Service
1 Secure boot (UEFI) prevents running an unknown OS loader.
2 The kernel launches Early Launch Antimalware (ELAM), which verifies integrity of third-party drivers and boot-time code.
3 Measurements of the system start state are recorded in the TPM during boot.
4 To prove a client is healthy, the antimalware software can quote TPM measurements to a remote verifier.

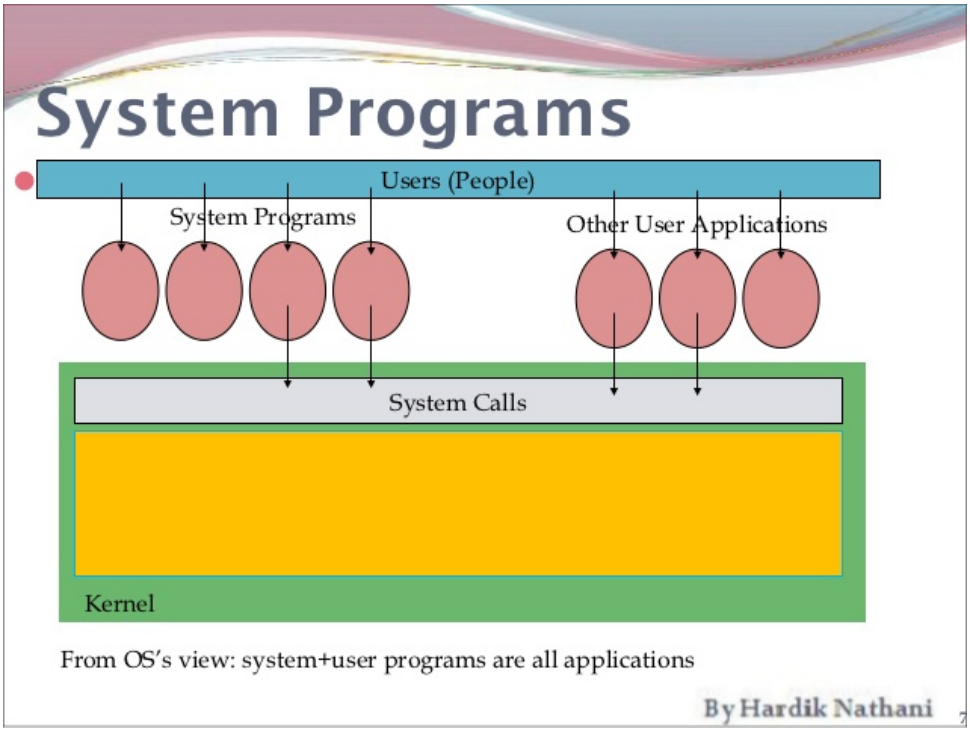


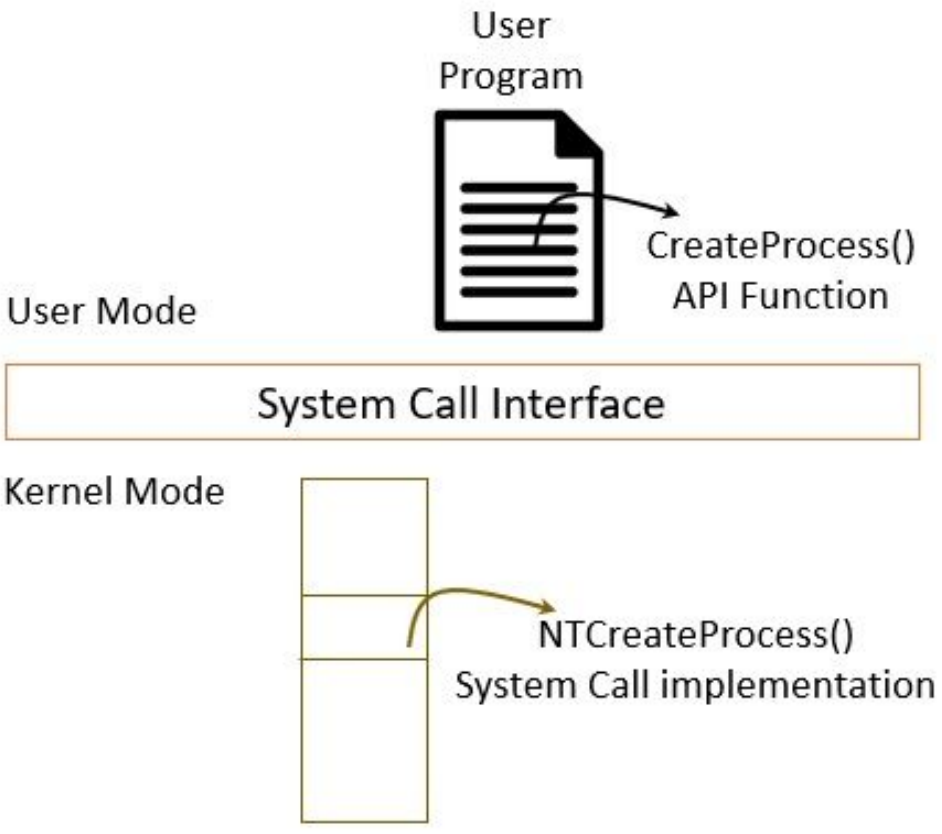


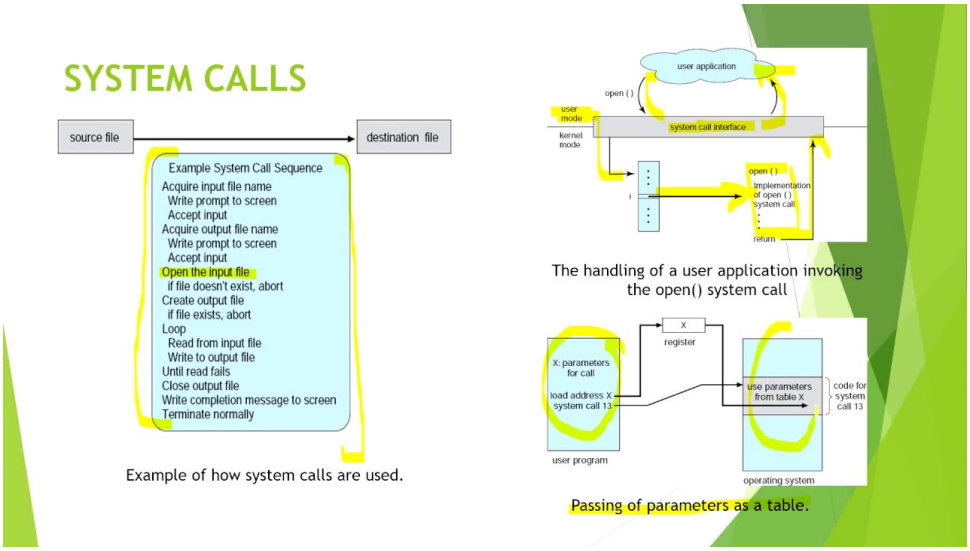


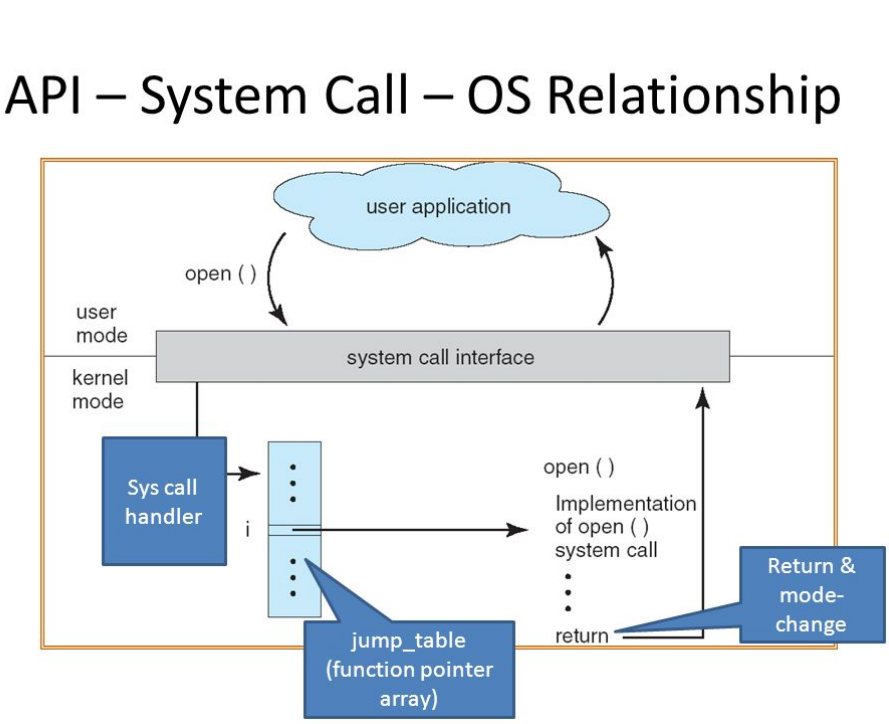
ROS Kernel Init Phase

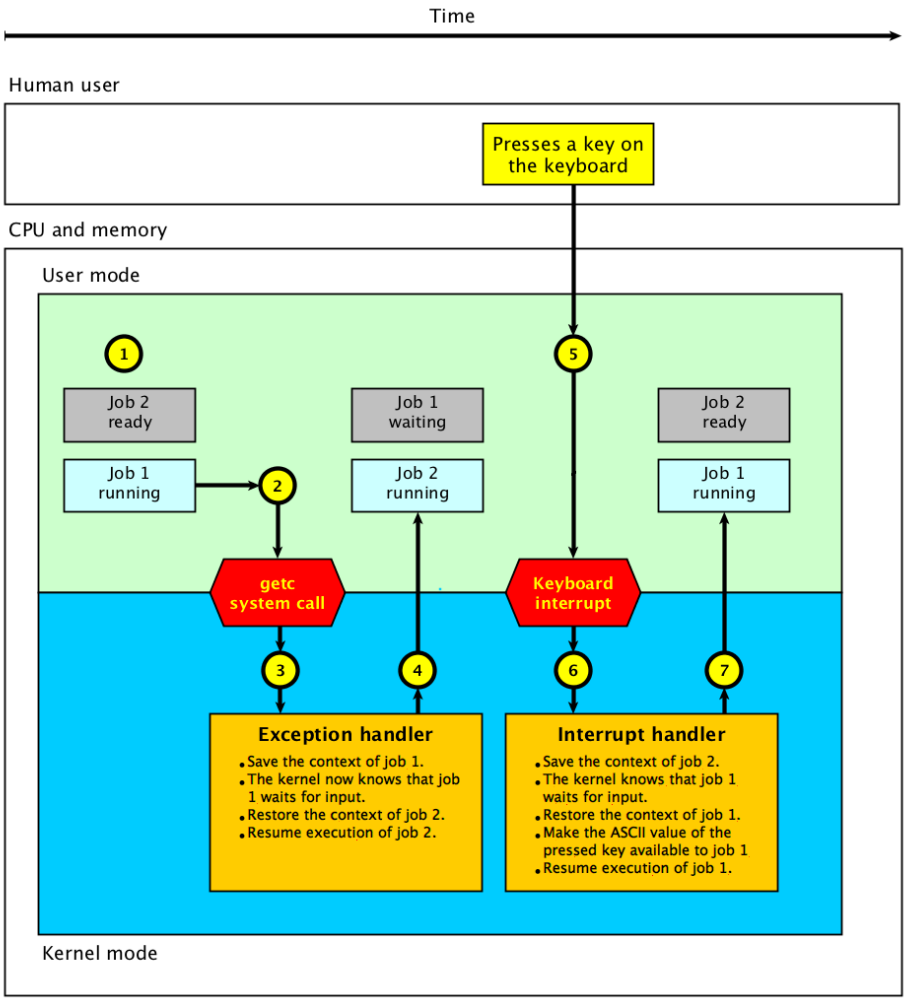
System Call

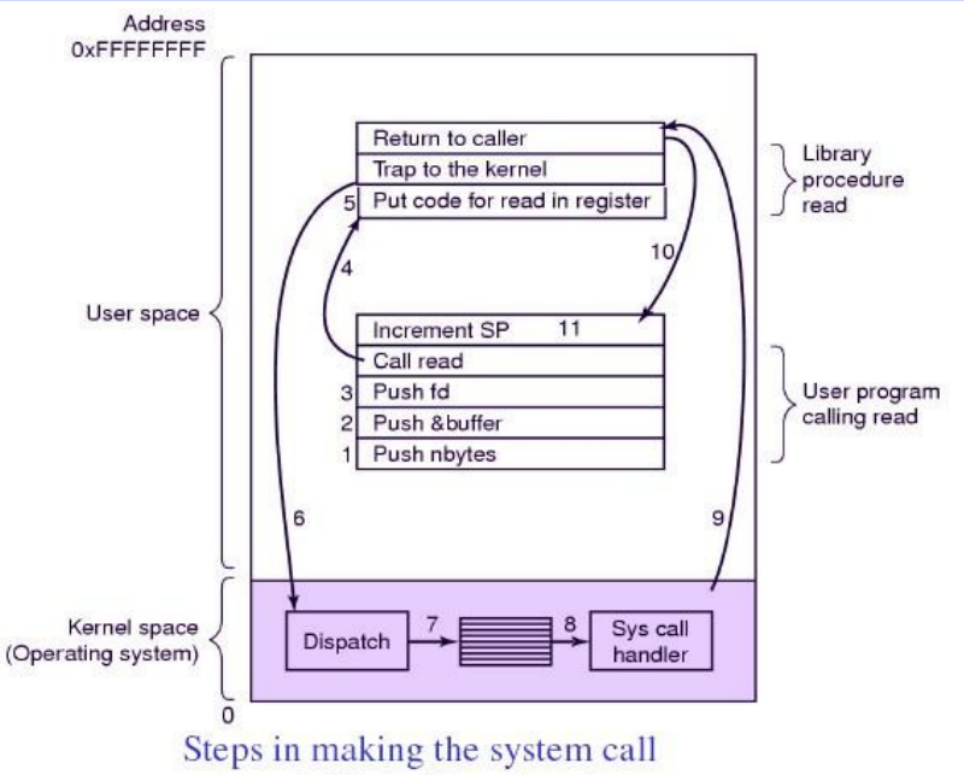

System Programs
Users (People)
System Programs
Other User Applications
System Calls
Kernel
From OS's view: system+user programs are all applications
By Hardik Nathani


User
Program
CreateProcess()
API Function
User Mode
System Call Interface
Kernel Mode
NTCreateProcess()
System Call implementation

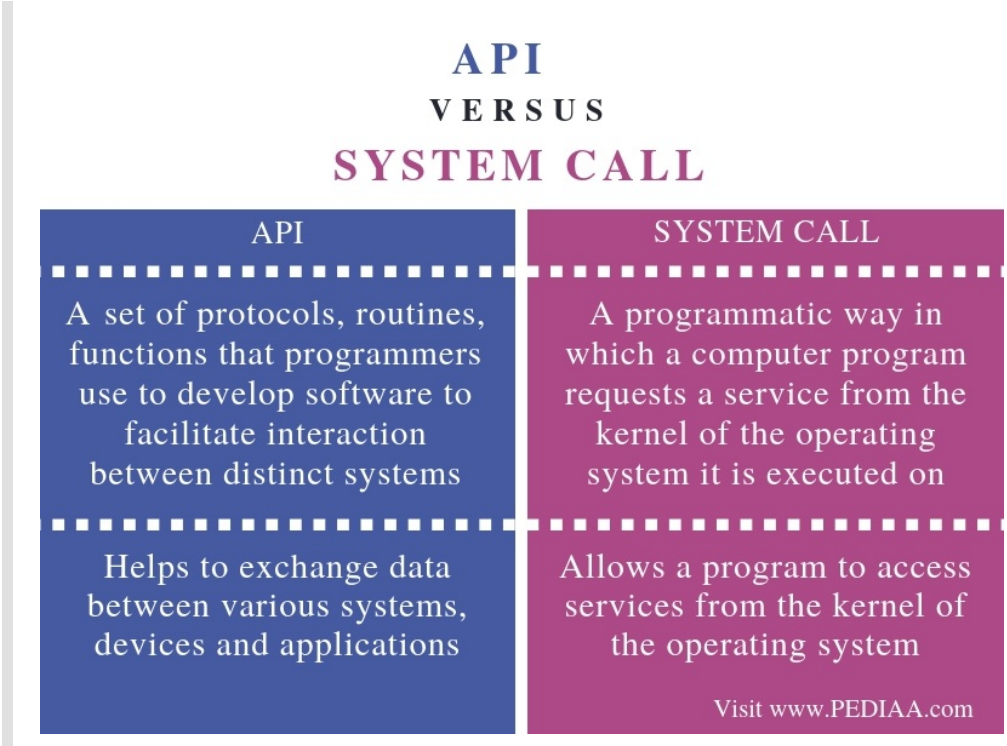

SYSTEM CALLS
source file
destination file
Example System Call Sequence
Acquire input file name
Write prompt to screen
Accept input
Acquire output file name
Write prompt to screen
Accept input
Open the input file
if file doesn't exist, abort
Create output file
if file exists, abort
Loop
Read from input file
Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally
Example of how system calls are used.
user application
open ( )
user mode
kernel mode
system call interface
i
open ( )
Implementation of open ( ) system call
return
The handling of a user application invoking the open() system call
X
register
X: parameters for call
load address X
system call 13
user program
use parameters from table X
code for system call 13
operating system
Passing of parameters as a table.


API – System Call – OS Relationship
user application
open ( )
user mode
kernel mode
system call interface
Sys call handler
i
open ( )
Implementation of open ( ) system call
return
Return & mode-change
jump_table (function pointer array)

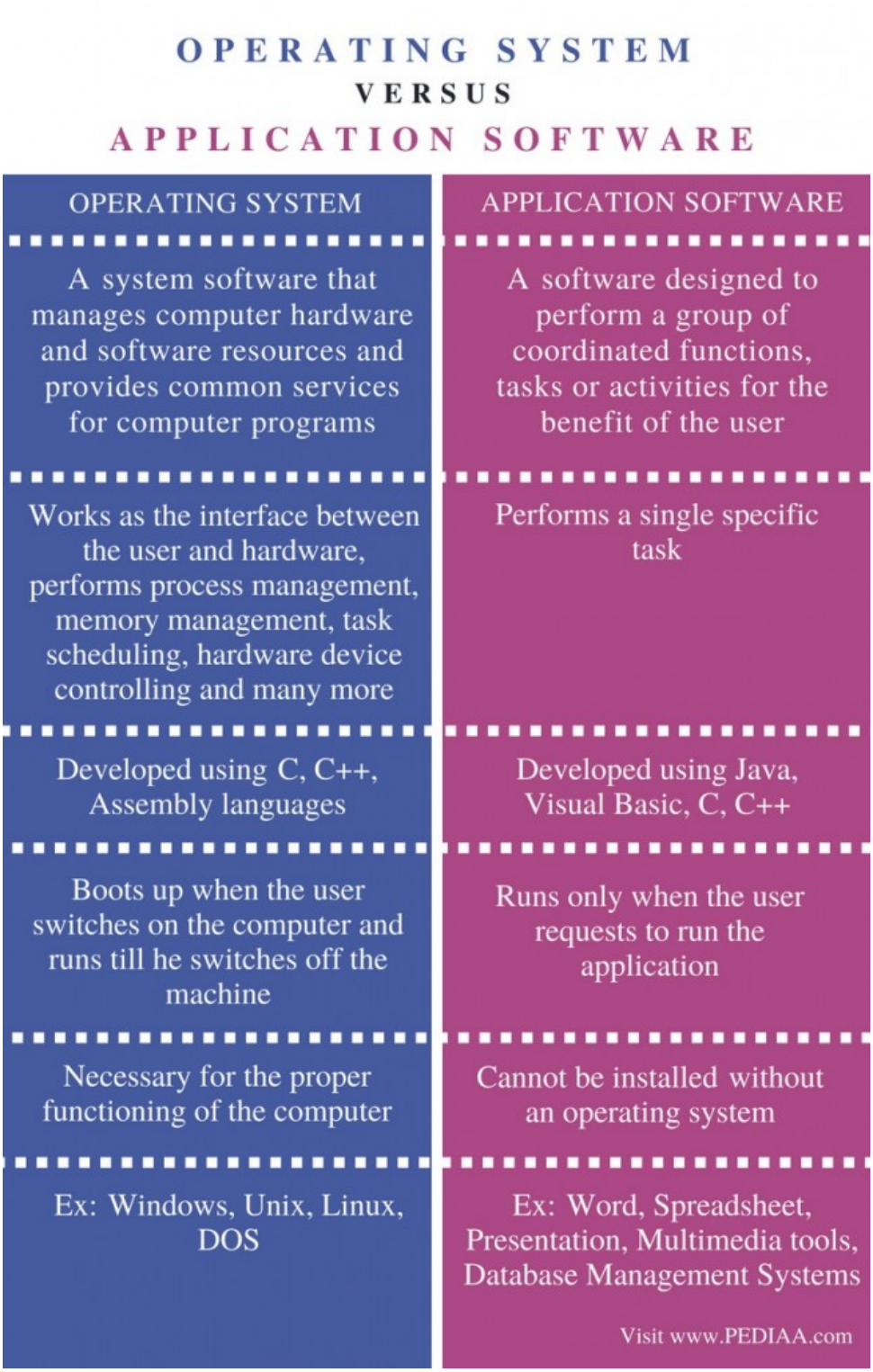




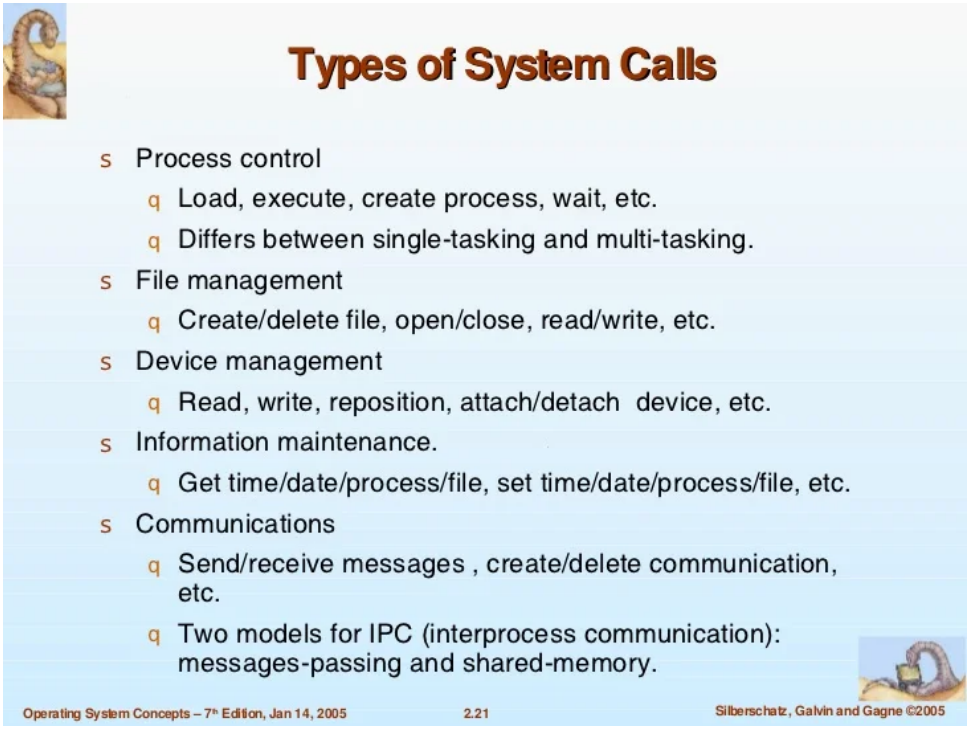
Steps in making the system call


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


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 prinf() are some examples for library calls
Visit www.PEDIAA.com



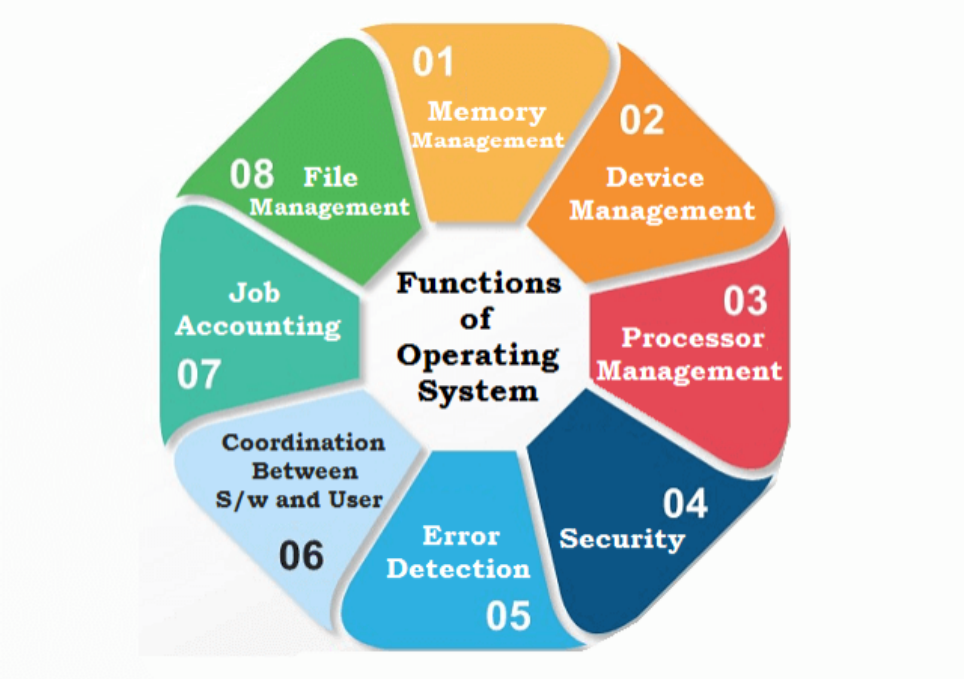
Type of System Call



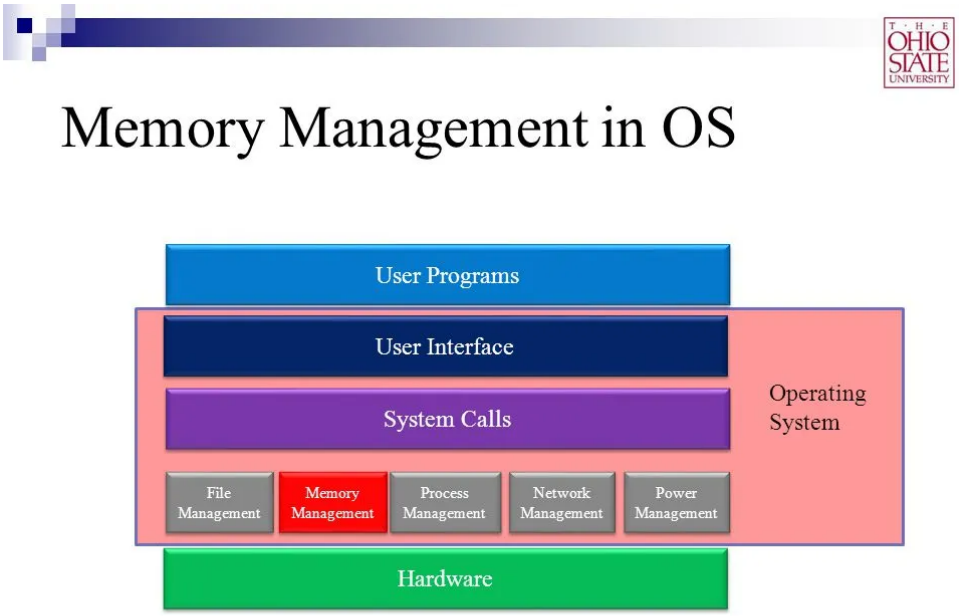
EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

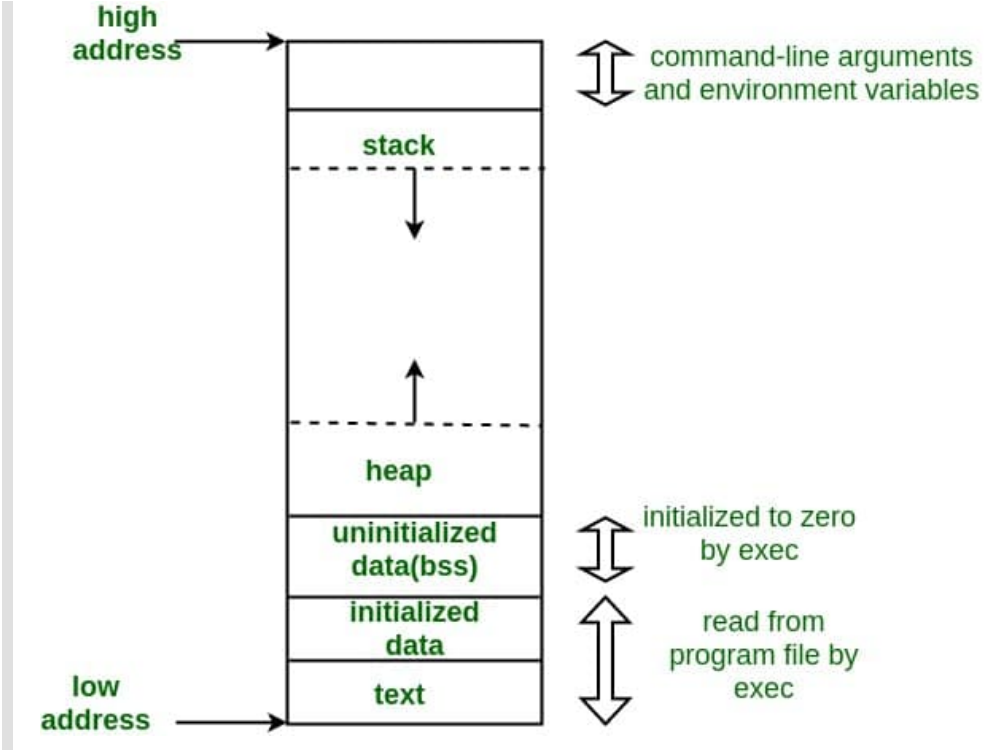
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information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

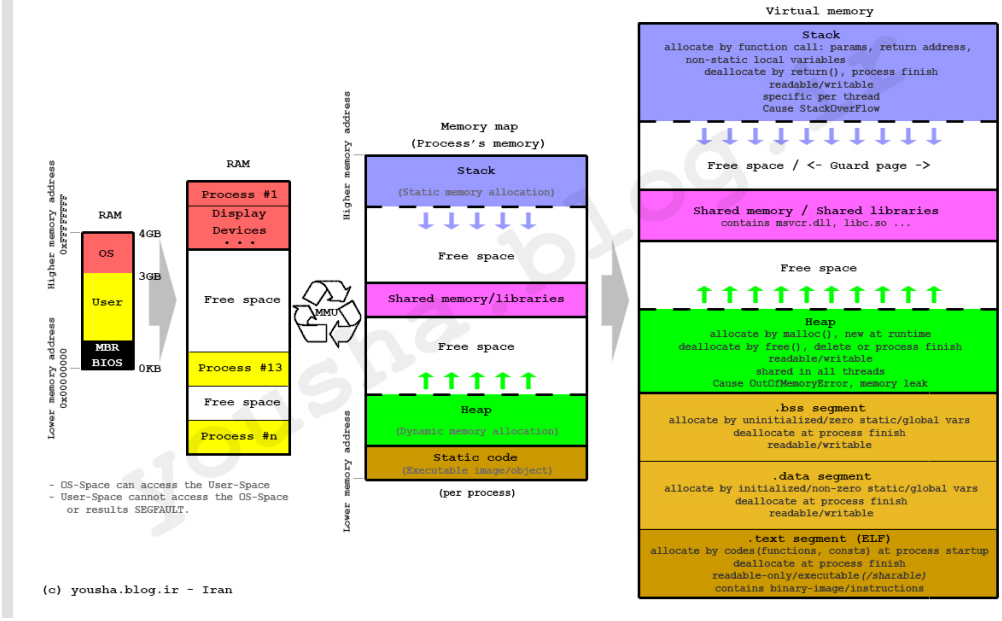
Functions of OS

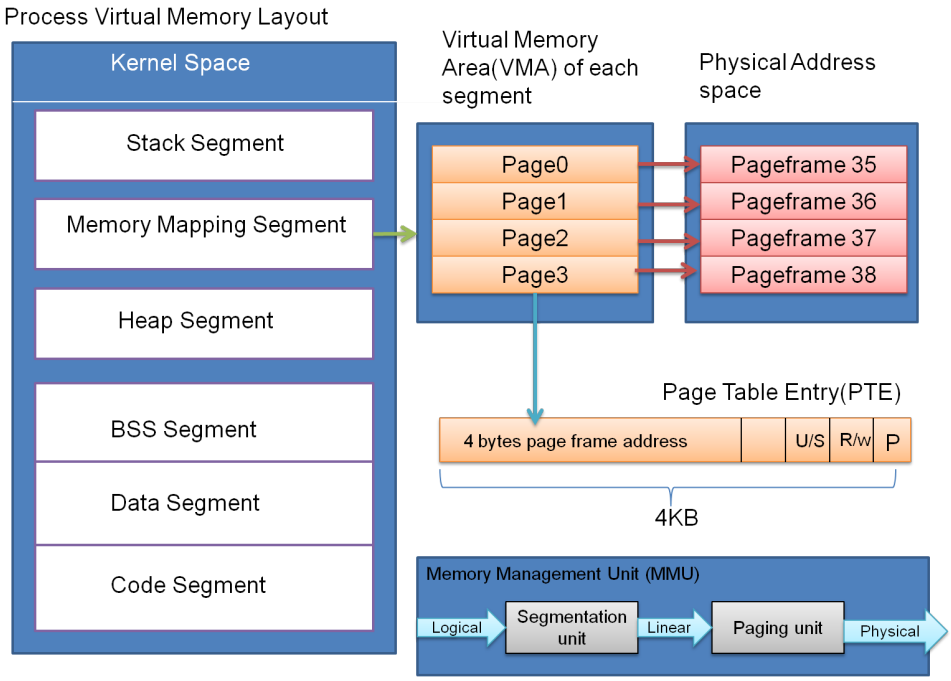


- Memory Management



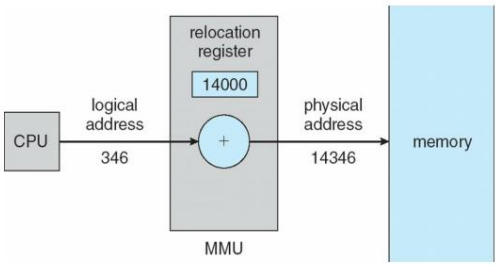

high address
command-line arguments and environment variables
stack
heap
uninitialized data(bss)
initialized to zero by exec
initialized data
read from program file by exec
text
low address


Virtual memory
Stack
allocate by function call: params, return address, non-static local variables
deallocate by return(), process finish
readable/writable
specific per thread
Cause StackOverFlow
Free space / <- Guard page ->
Shared memory / Shared libraries
contains msvcr.dll, libc.so ...
Free space
Heap
allocate by malloc(), new at runtime
deallocate by free(), delete or process finish
readable/writable
shared in all threads
Cause OutOfMemoryError, memory leak
.bss segment
allocate by uninitialized/zero static/global vars
deallocate at process finish
readable/writable
.data segment
allocate by initialized/non-zero static/global vars
deallocate at process finish
readable/writable
.text segment (ELF)
allocate by codes(functions, consts) at process startup
deallocate at process finish
readable-only/executable(/sharable)
contains binary-image/instructions
Memory map
(Process's memory)
Higher memory address
Lower memory address
Stack
(Static memory allocation)
Free space
Shared memory/libraries
Free space
Heap
(Dynamic memory allocation)
Static code
(Executable image/object)
(per process)
RAM
Process #1
Display Devices
Free space
Process #13
Free space
Process #n
MMU
RAM
4GB
3GB
0KB
OS
User
MBR
BIOS
- OS-Space can access the User-Space
- User-Space cannot access the OS-Space or results SEGFAULT.
(c) yousha.blog.ir - Iran



Memory-Management Unit (MMU)

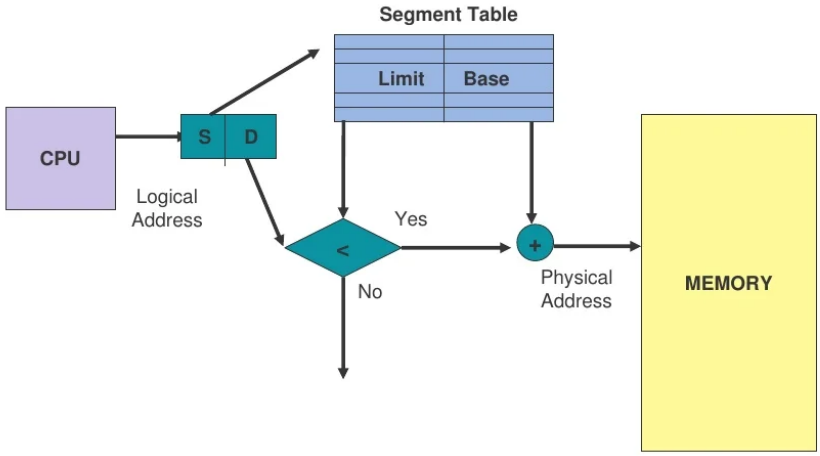
- Hardware device that maps **virtual** to **physical** address
- In MMU scheme, the value in the relocation register **is added to** every address generated by a user process at the time it is sent to memory
- The user program deals with *logical* addresses; it never sees the *real* physical addresses



Dynamic relocation using a relocation register

MEMORY MANAGEMENT Segmentation

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



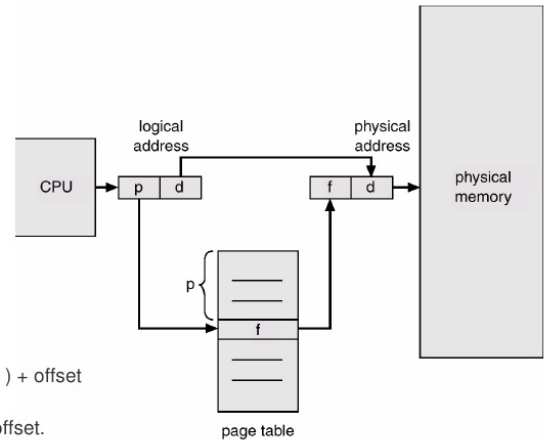
8: Memory Management 31

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)

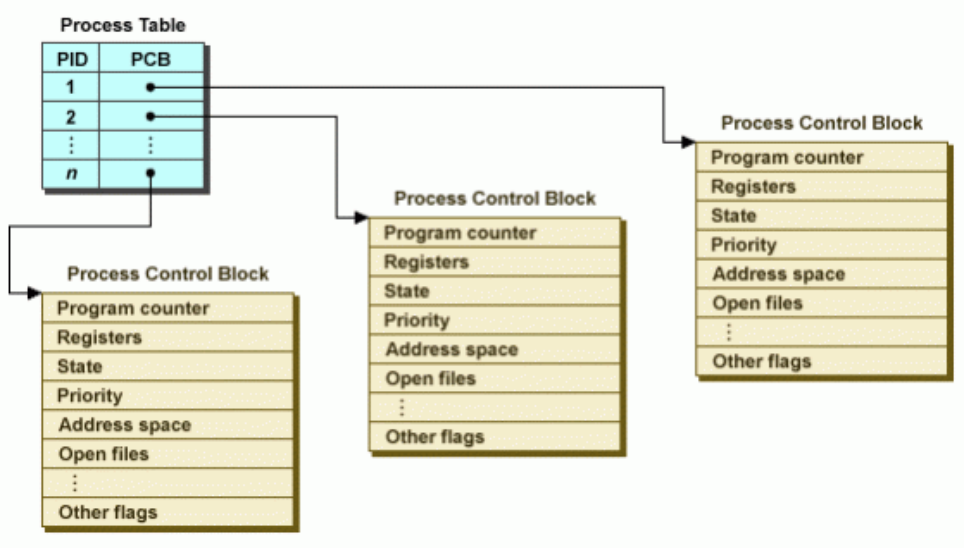


HARDWARE
An address is determined by:

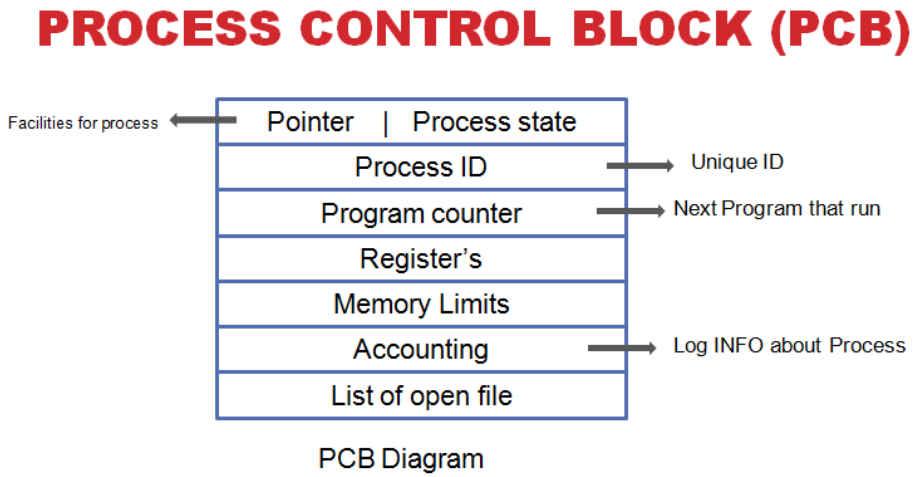
page number (index into table) + offset
---> mapping into --->
base address (from table) + offset.

8: Memory Management 22

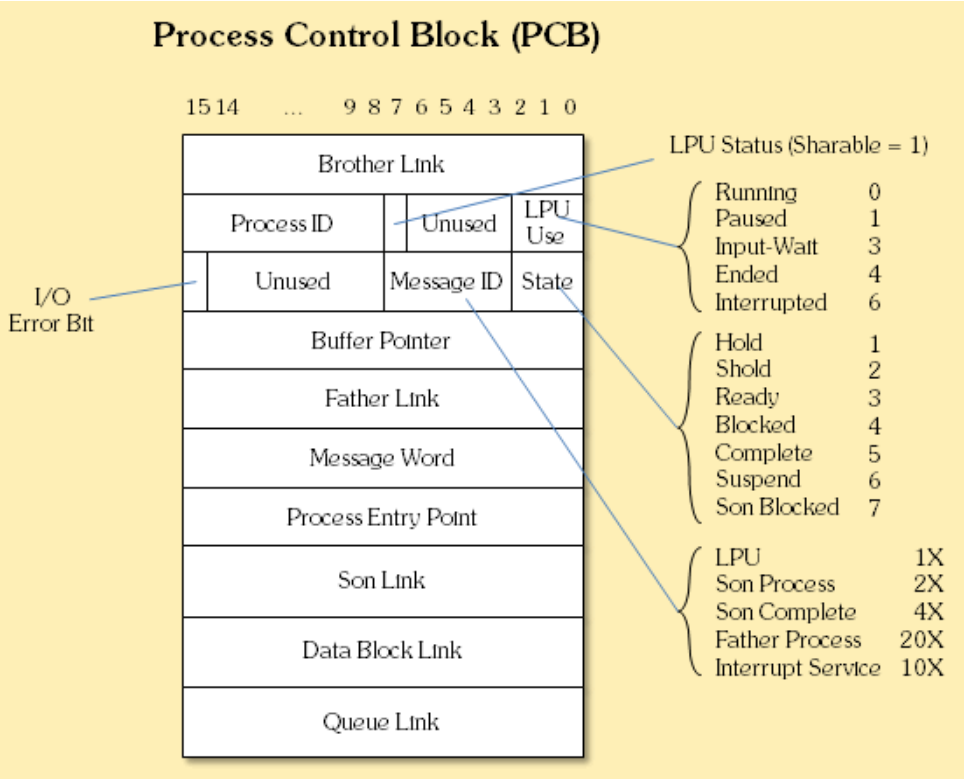
- Process Management



PCB(Process Control Block)

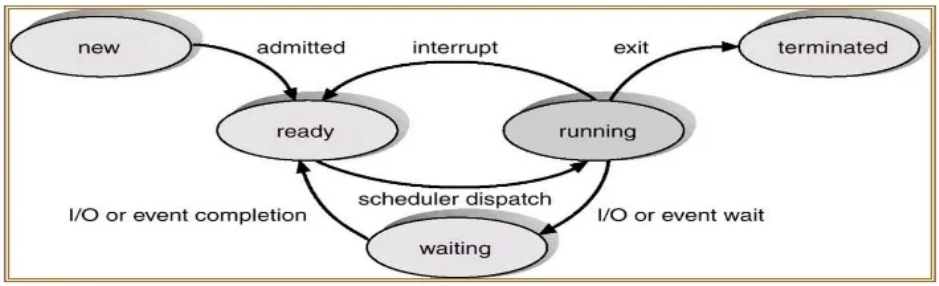


PCB Diagram



Process Switch and State Flow

Process Management



Process States

•New- The process is being created.

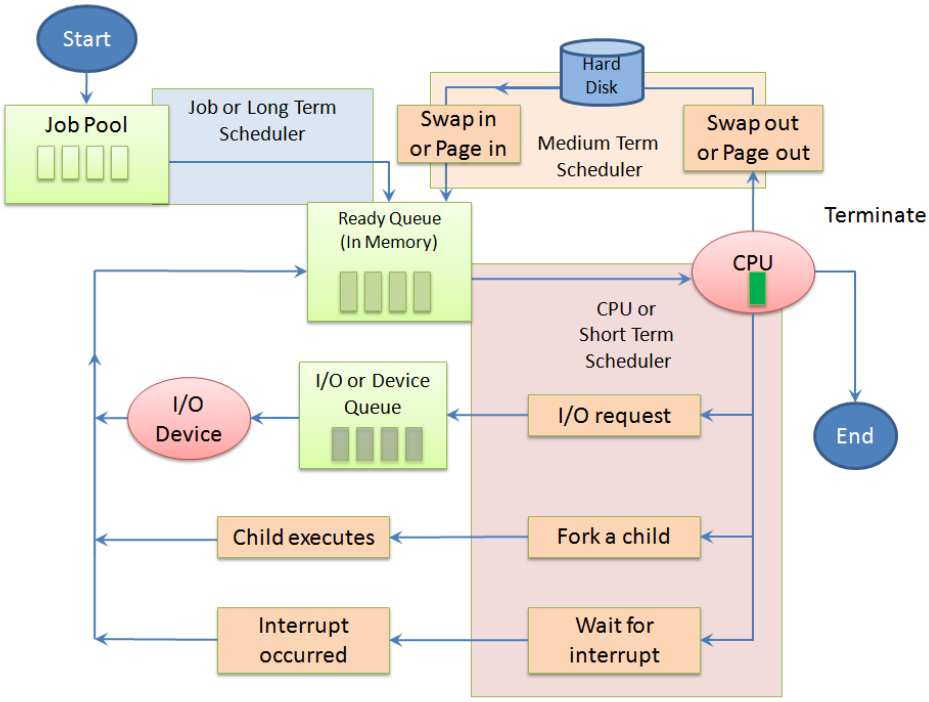
•Running- Instructions are being executed.

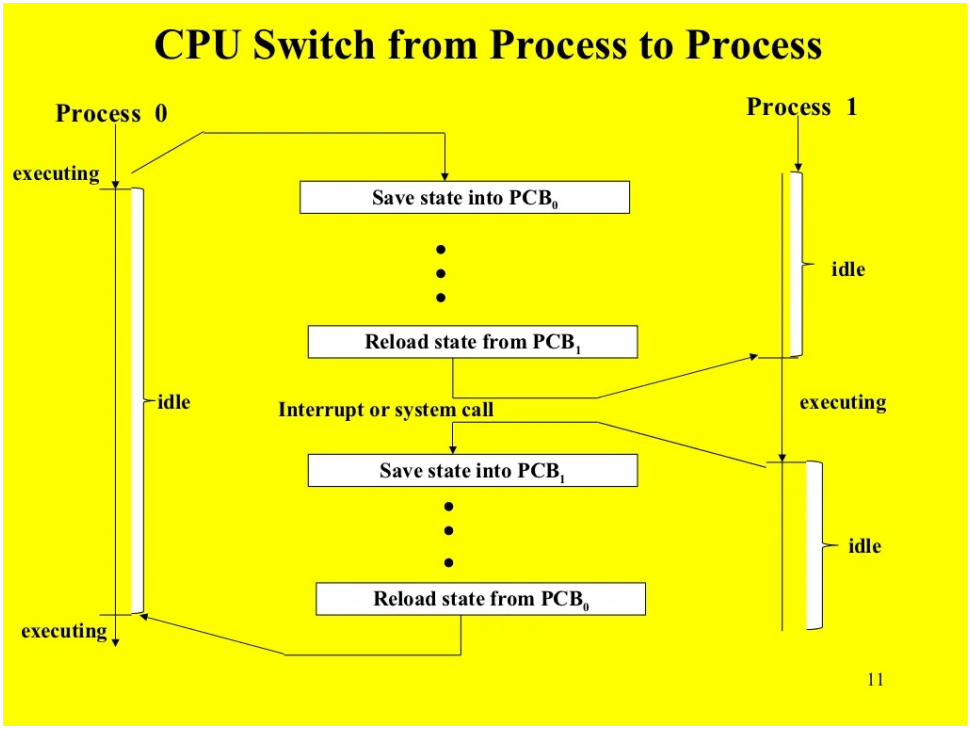
•Waiting- The process is waiting for some event to occur.

•Ready- The process is waiting to be assigned to a processor.

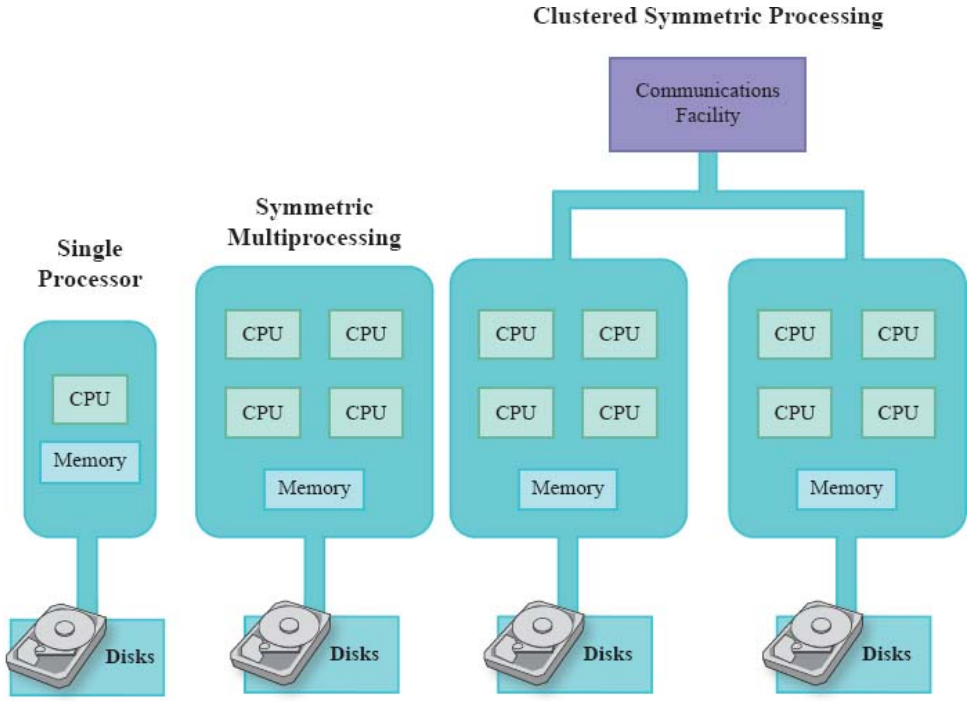
•Terminated- The process has finished execution.

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.

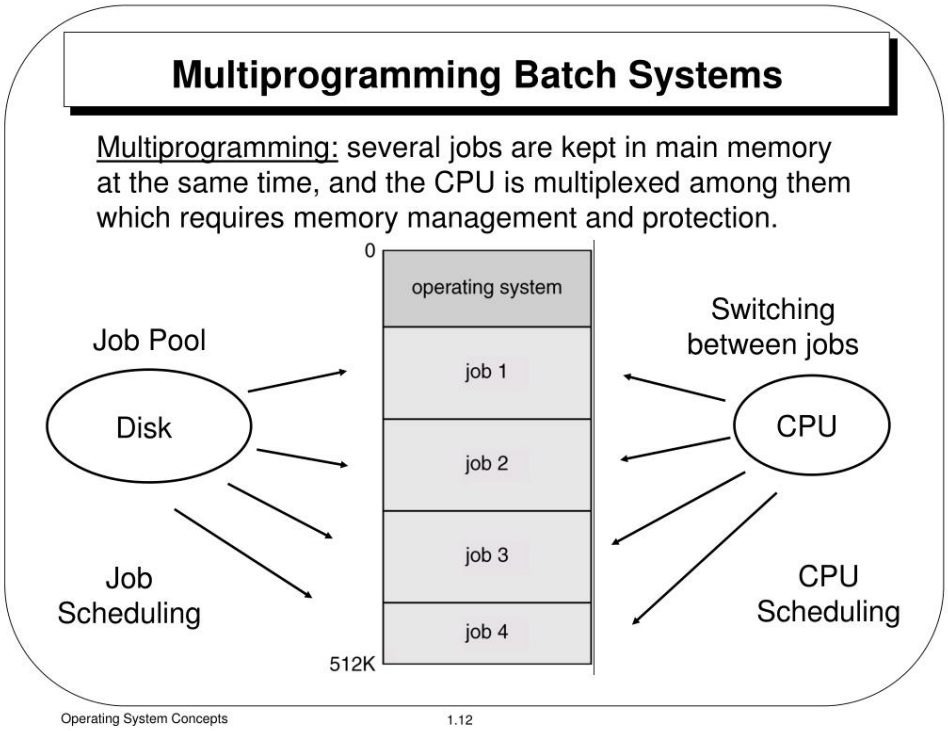




SP vs MP vs Cluster



Multiprogramming

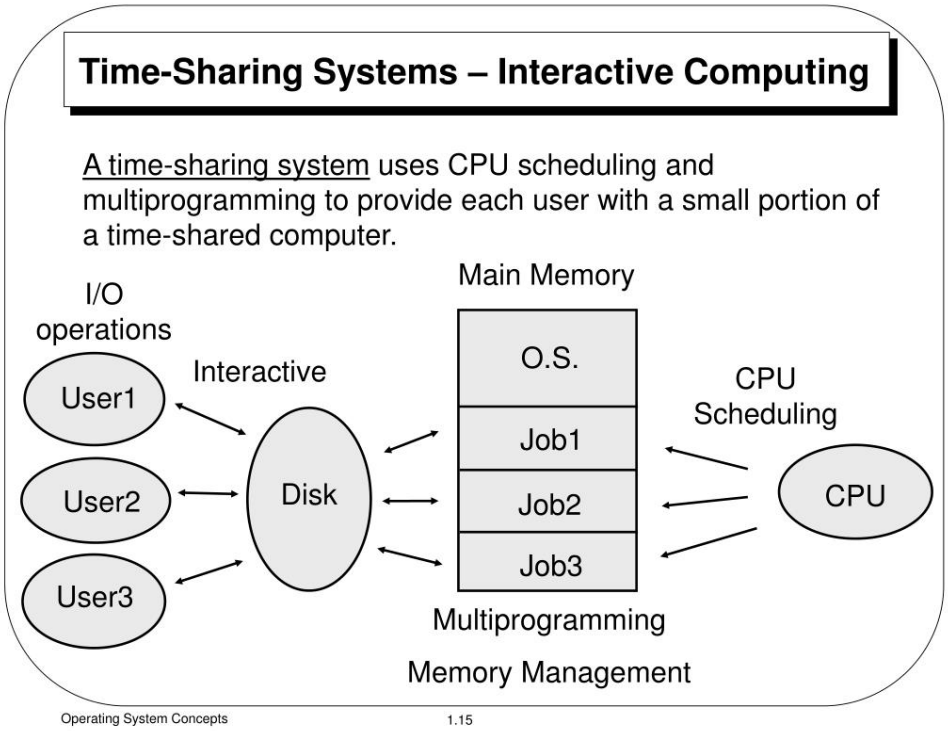


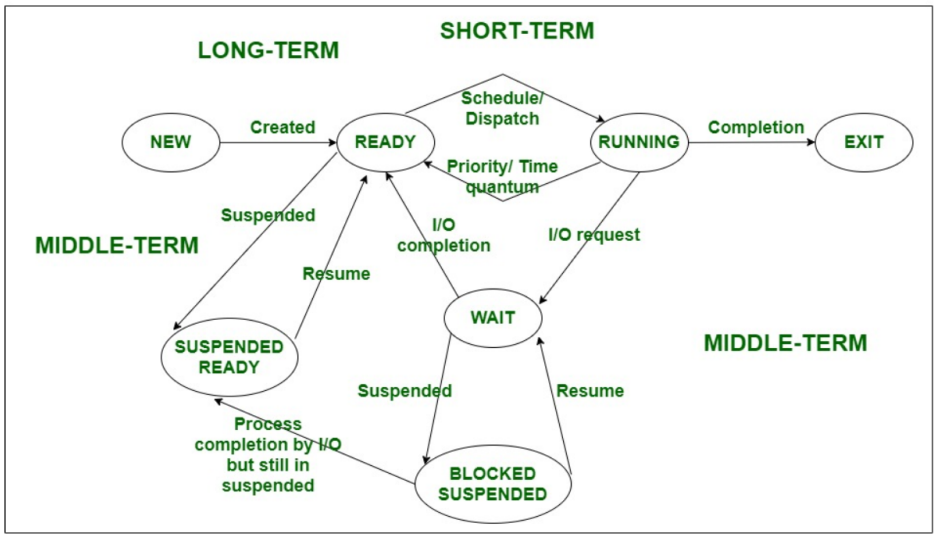
MULTIPROGRAMMING VERSUS MULTITASKING

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.

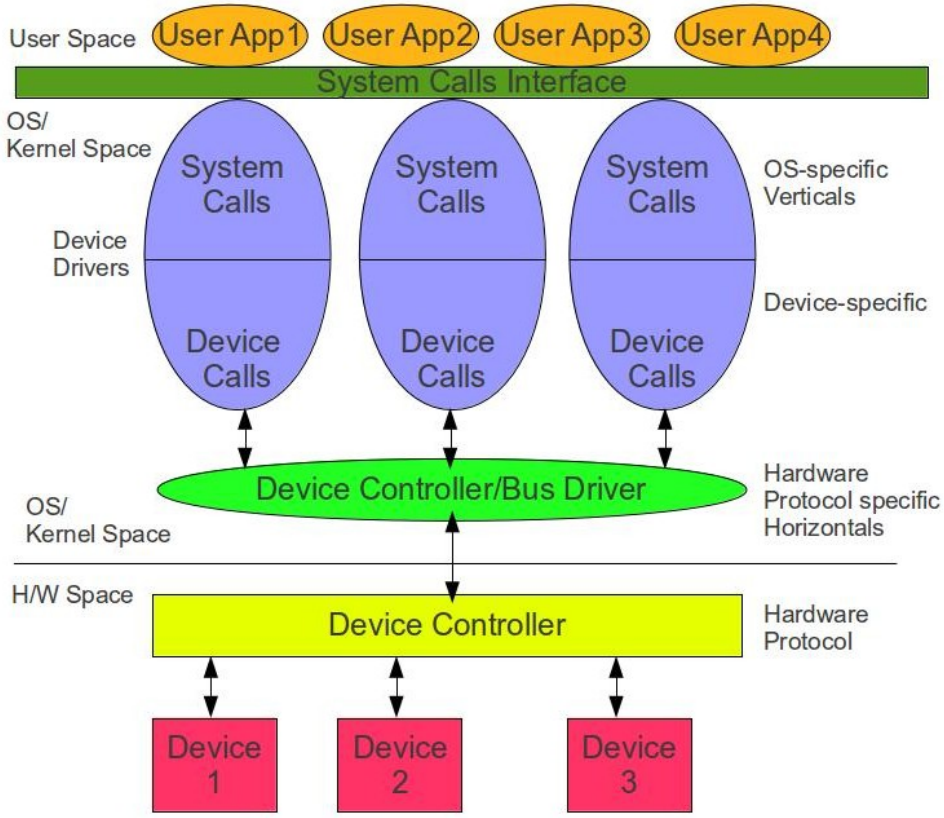
DB Difference Between.net

Time Sharing

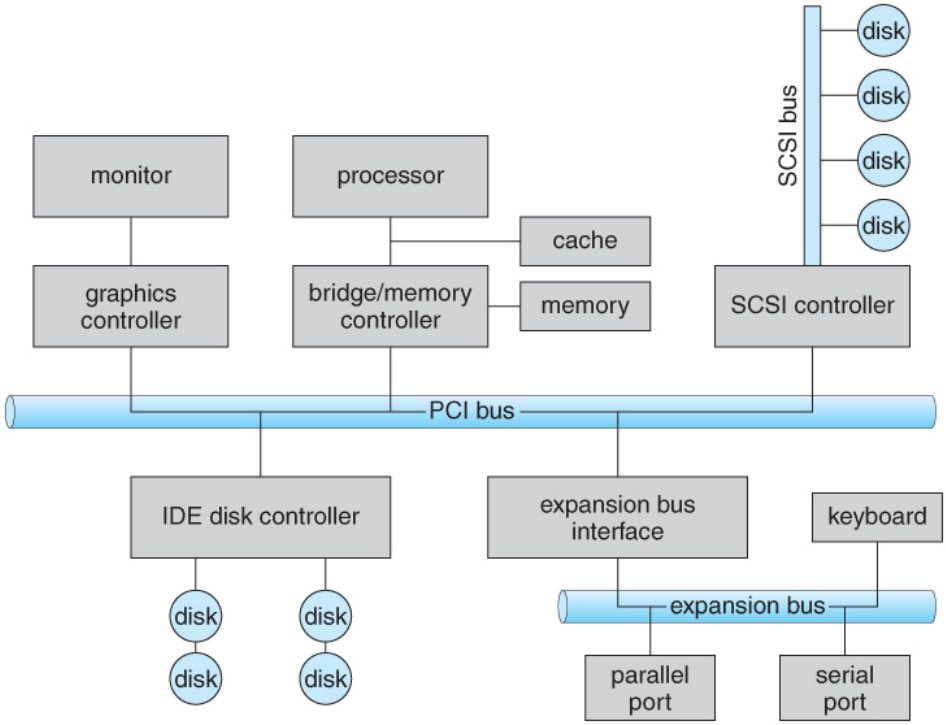




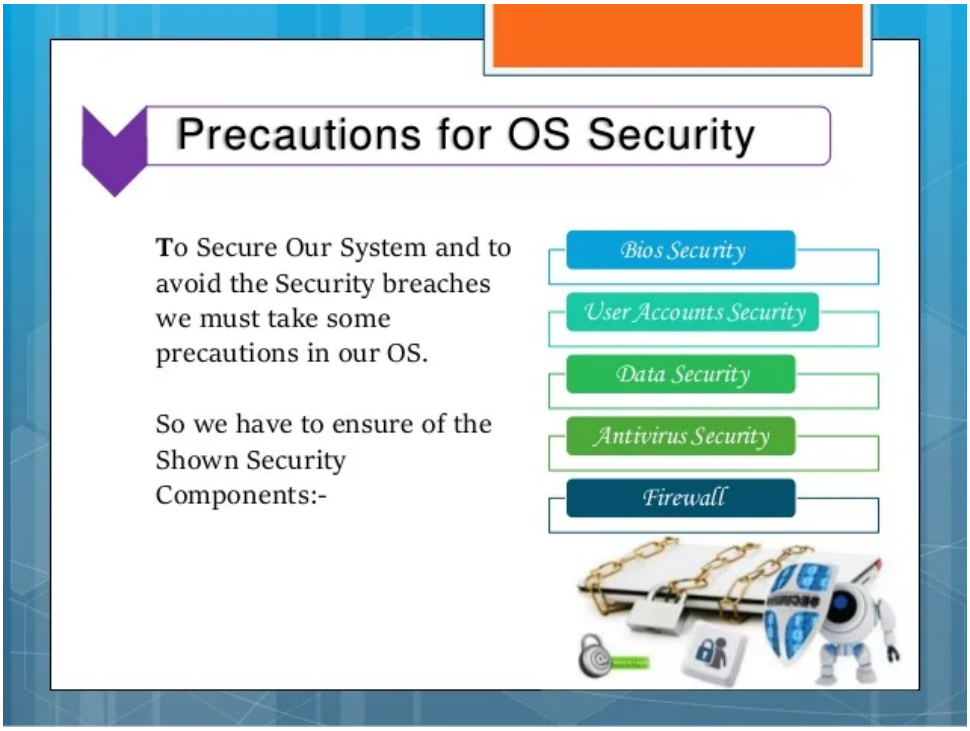
- Device Mangement

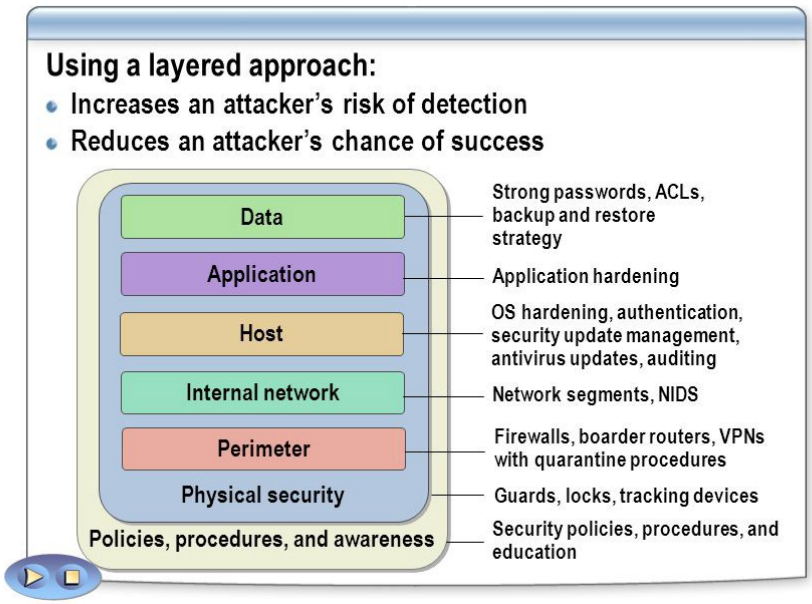


- I/O Mangement

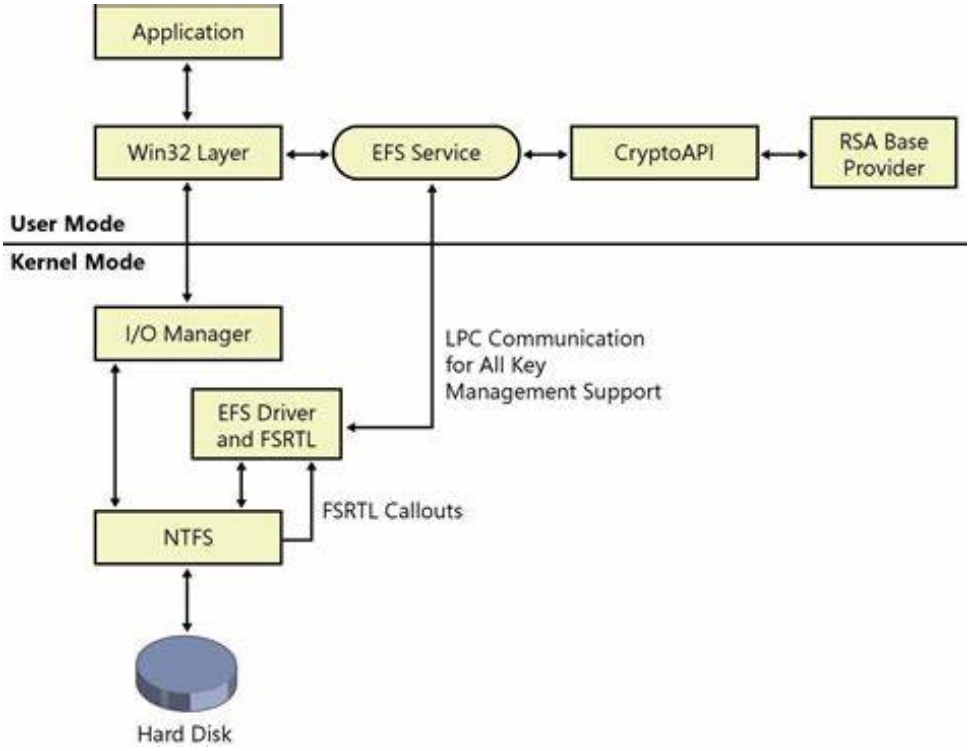


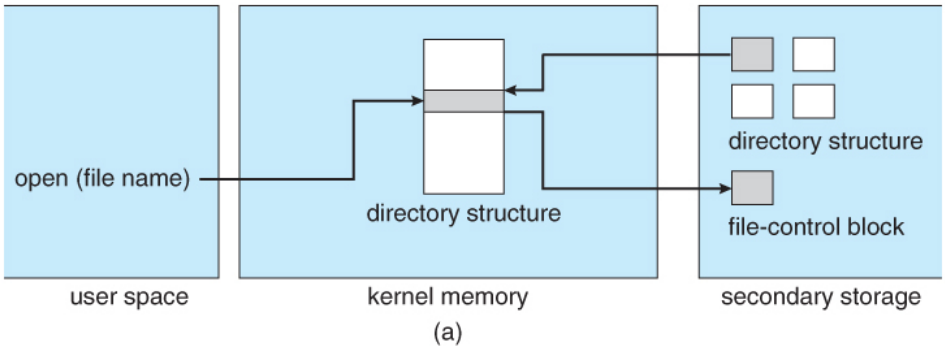
- OS Security



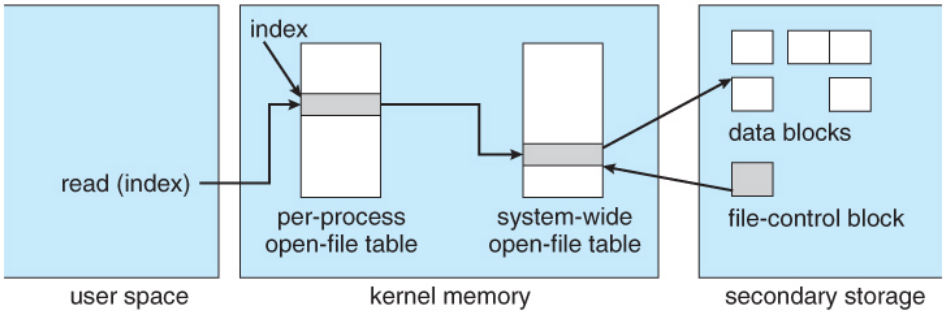


- File Management



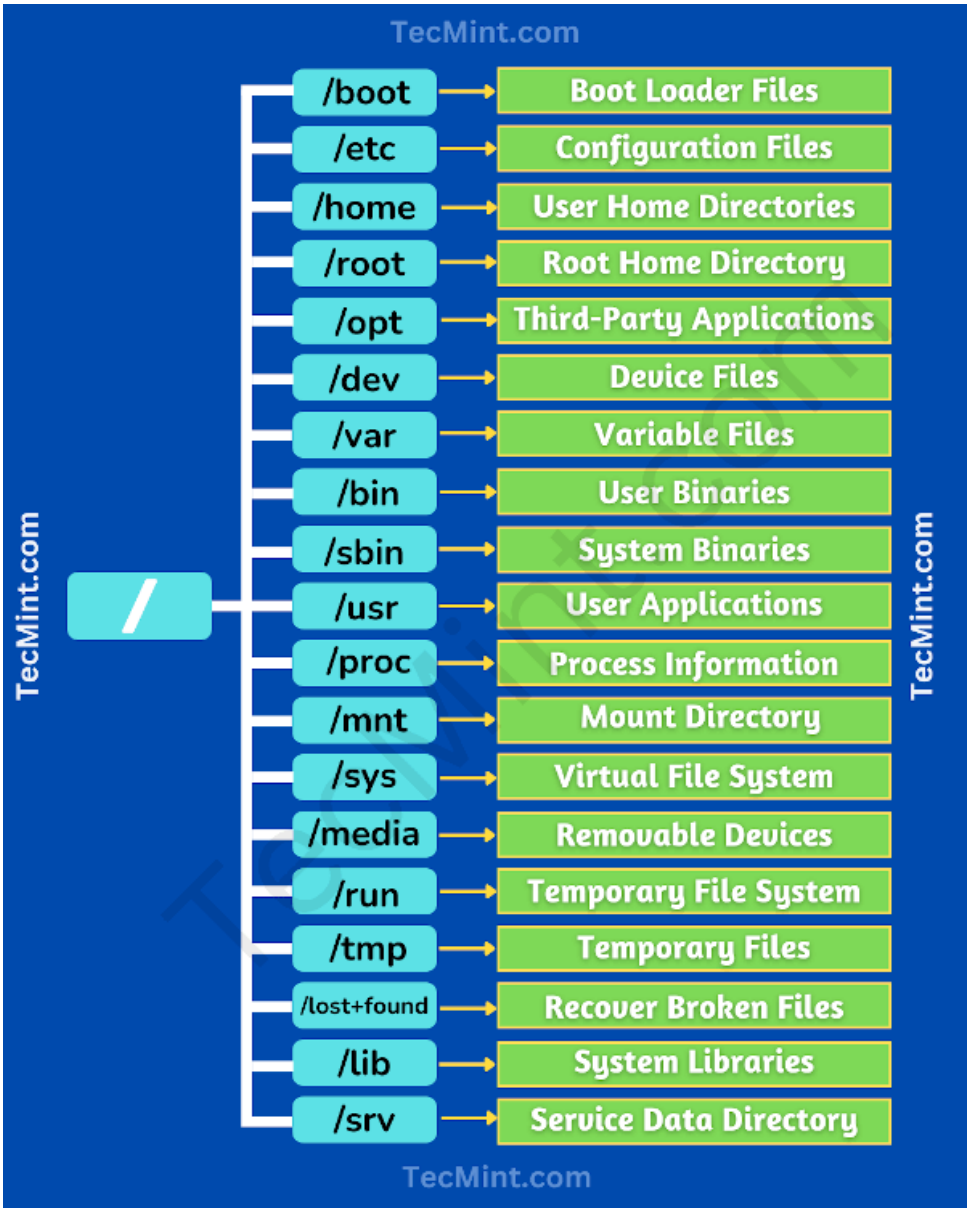


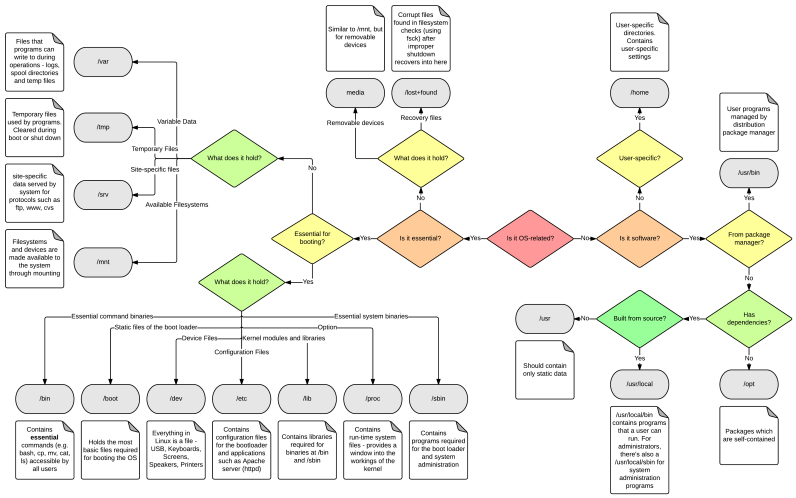
(a)



(b)

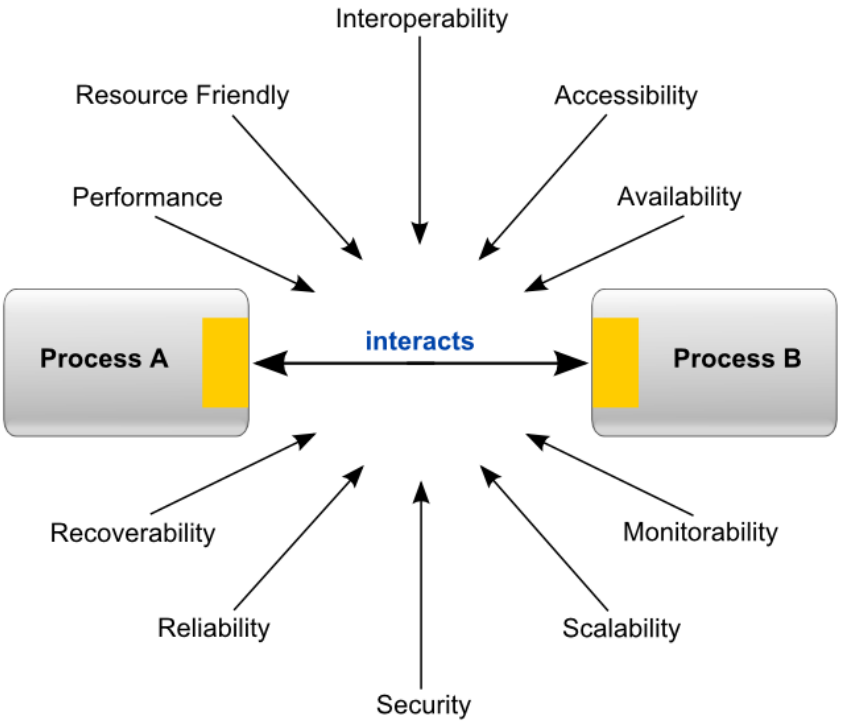
File type	Usual extension	Function
Executable	exe,com,bin	Read to run machine language program
Object	obj,o	Compiled,machine language not linked
Source code	C,java,pas,asm,a	Source code in various languages
Batch	bat,sh	Commands to the command interpreter
Text	txt,doc	Textual data,documents
Word processor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



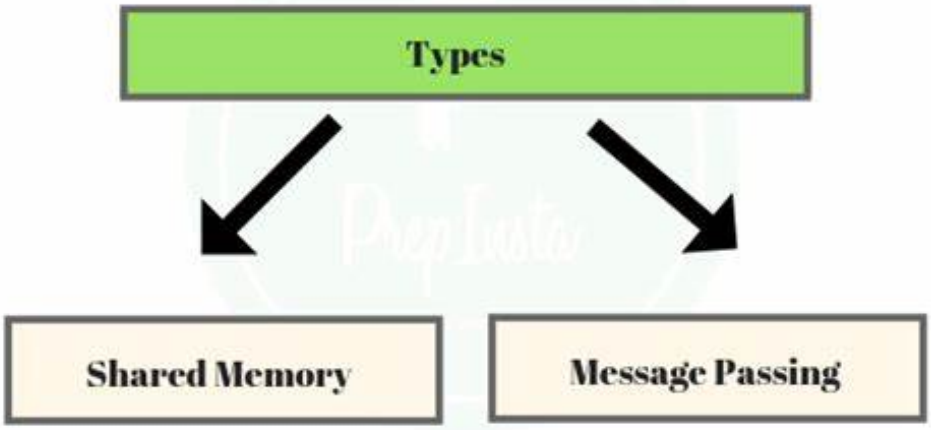


Process Communication

Interprocess Communication

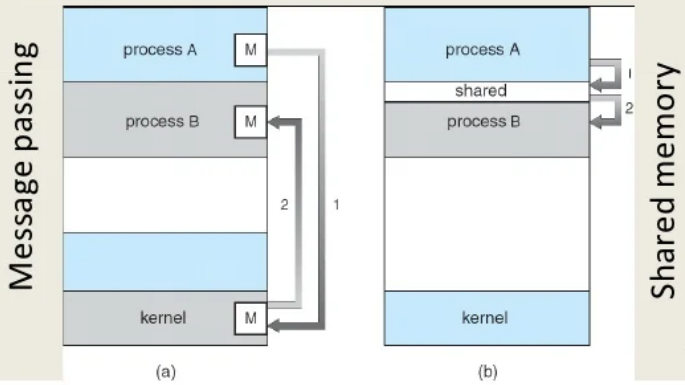


Inter-Process Communication Types



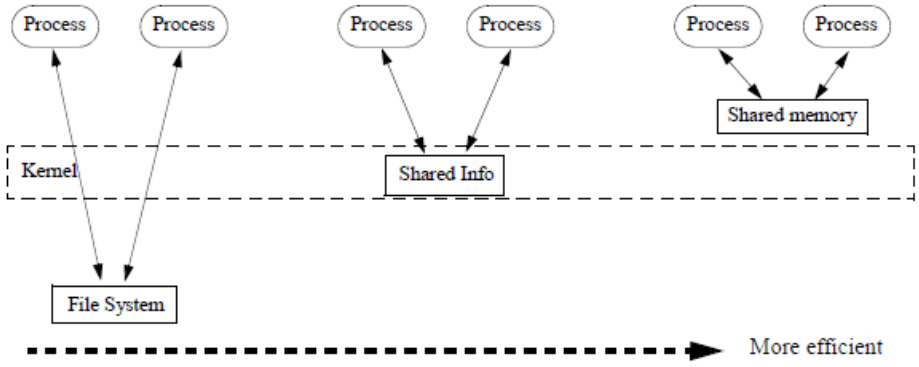
4. Interprocess Communication Contd...

- 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.

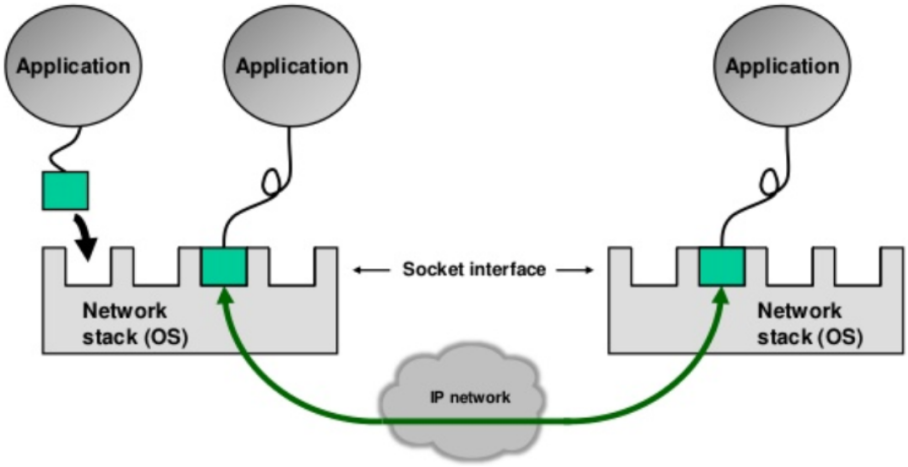


Examples:

files

semaphore, socket, pipe, message queue, signal

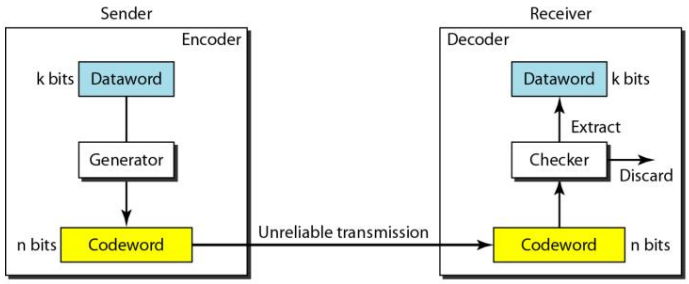
shared memory



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.

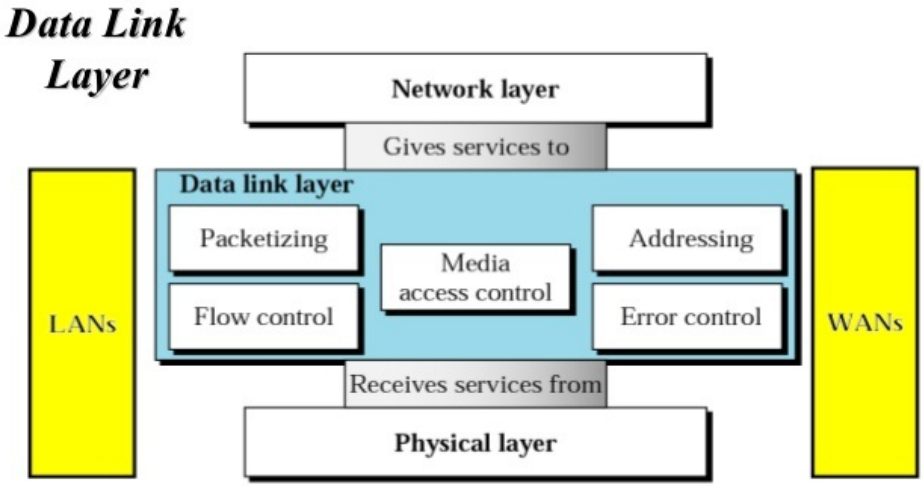


10.12

Er. M.S.Kuthar

Error Detection and Correction

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

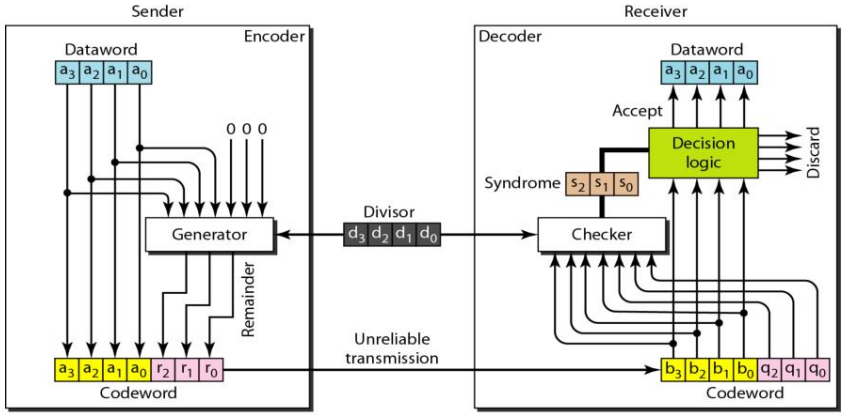


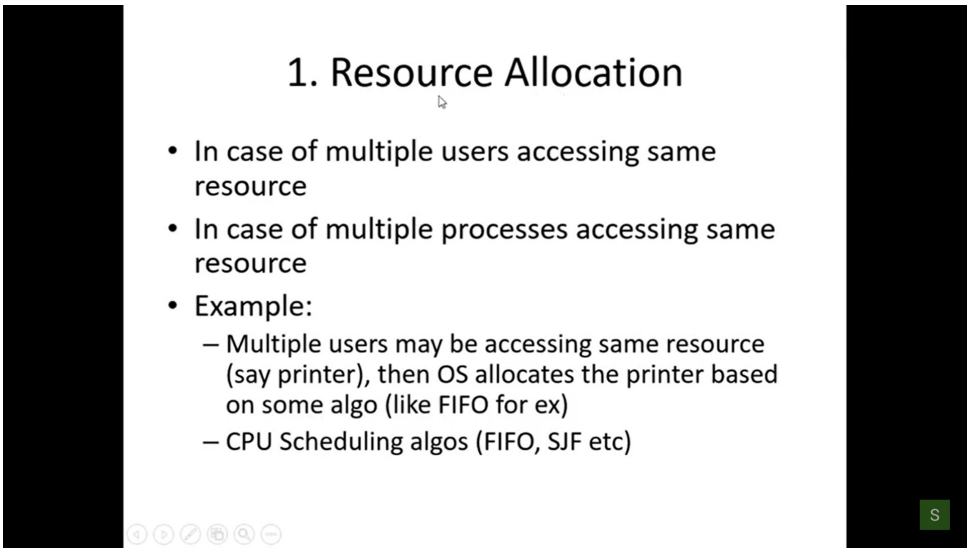
BCS Error Detection & Correction

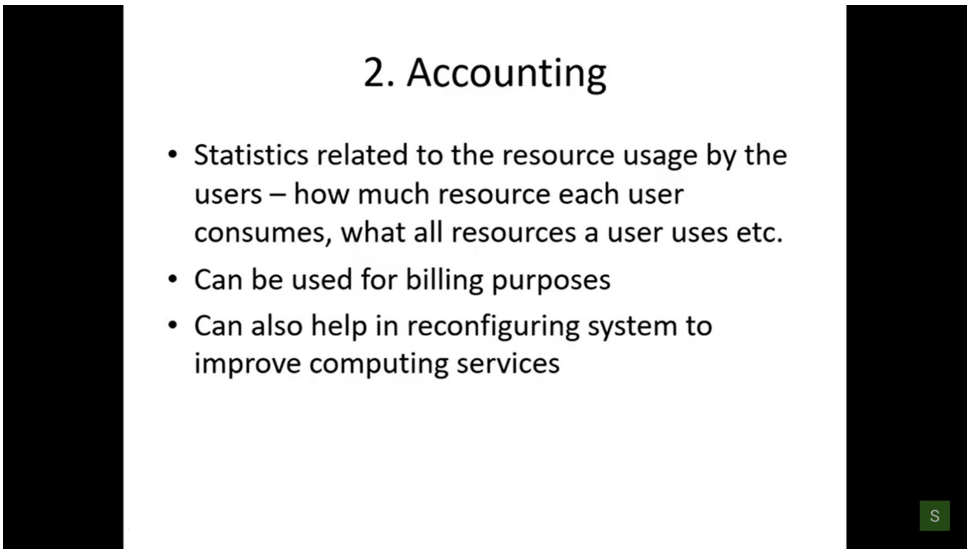
- Error Detection
  - Check if any error has occurred
  - Don't care the number of errors
  - Don't care the positions of errors
- Error Correction
  - Need to know the number of errors
  - Need to know the positions of errors
  - More complex
  - The number of errors and the size of the message are important factors in error correction

BCS Error Detection & Correction

Figure 1.6 CRC encoder and decoder

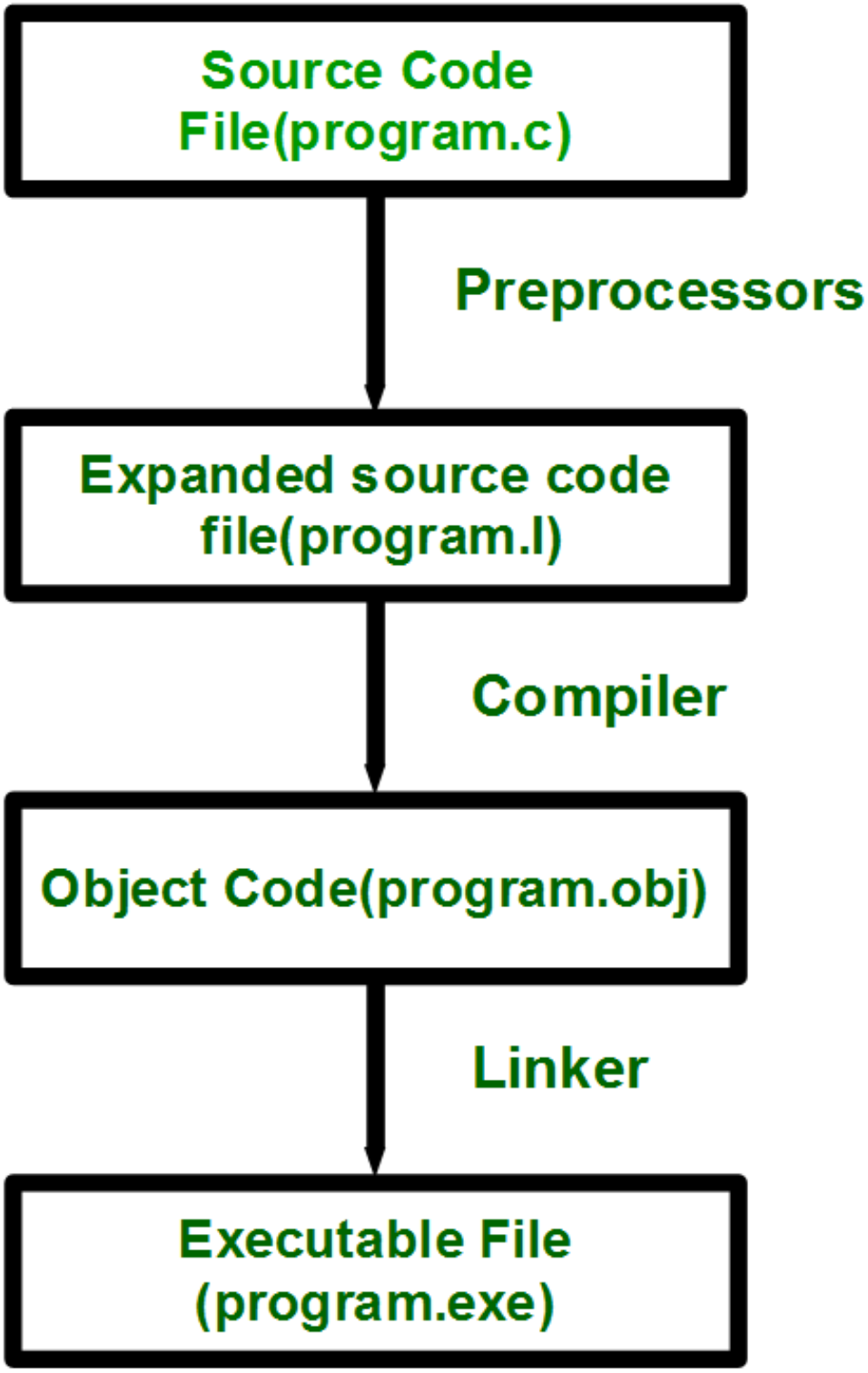


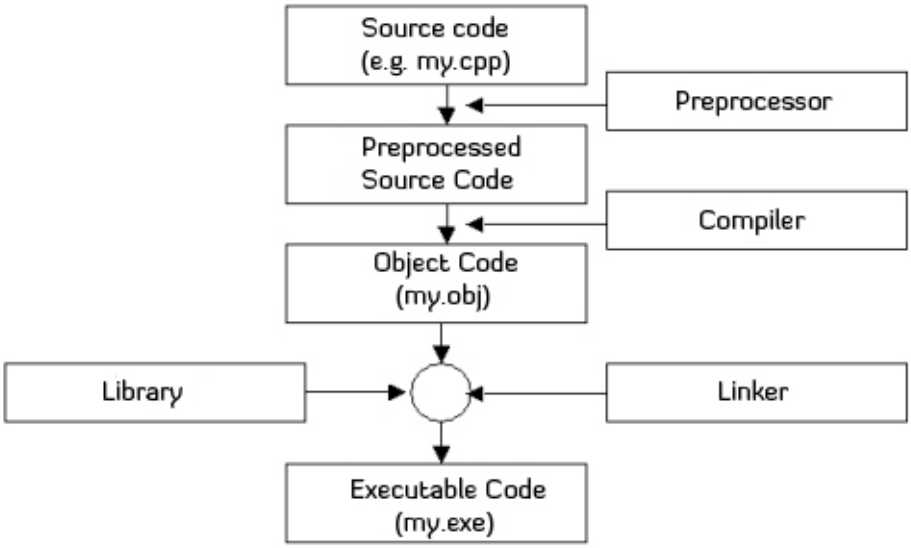




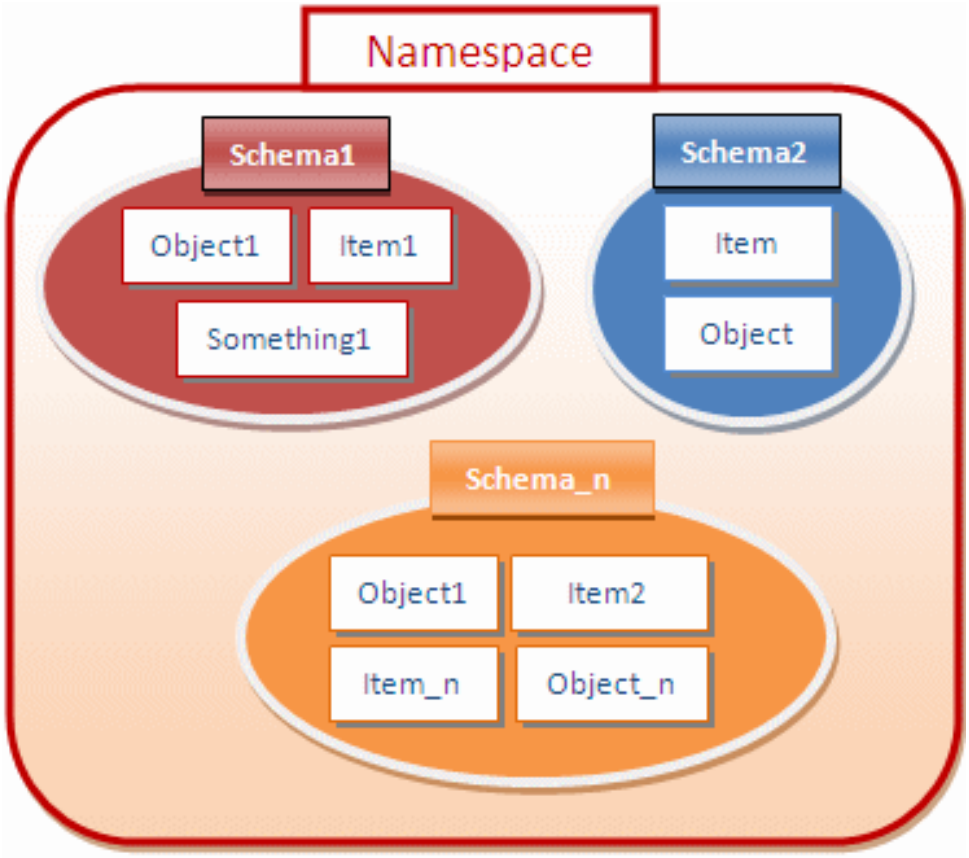
Programming Language - C & C++

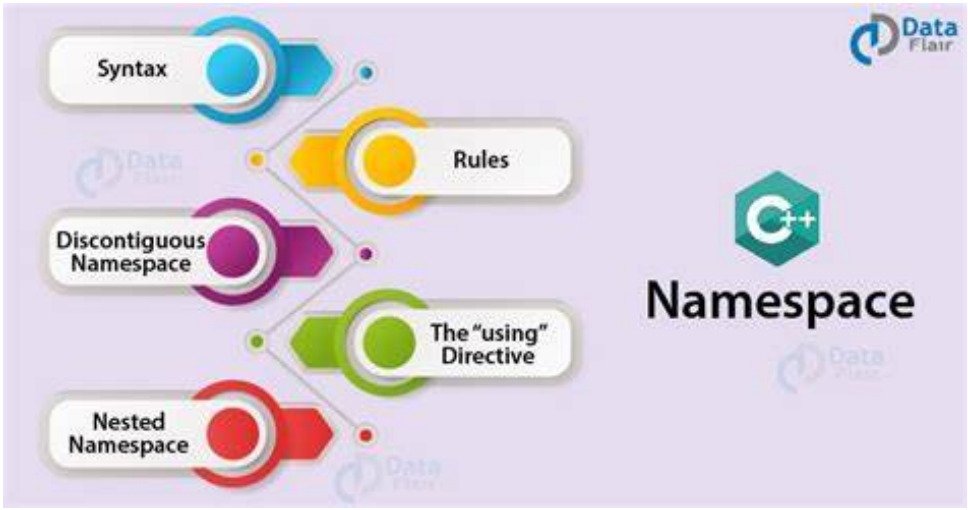
Preprocessors


Source Code
File(program.c)
Preprocessors
Expanded source code
file(program.I)
Compiler
Object Code(program.obj)
Linker
Executable File
(program.exe)

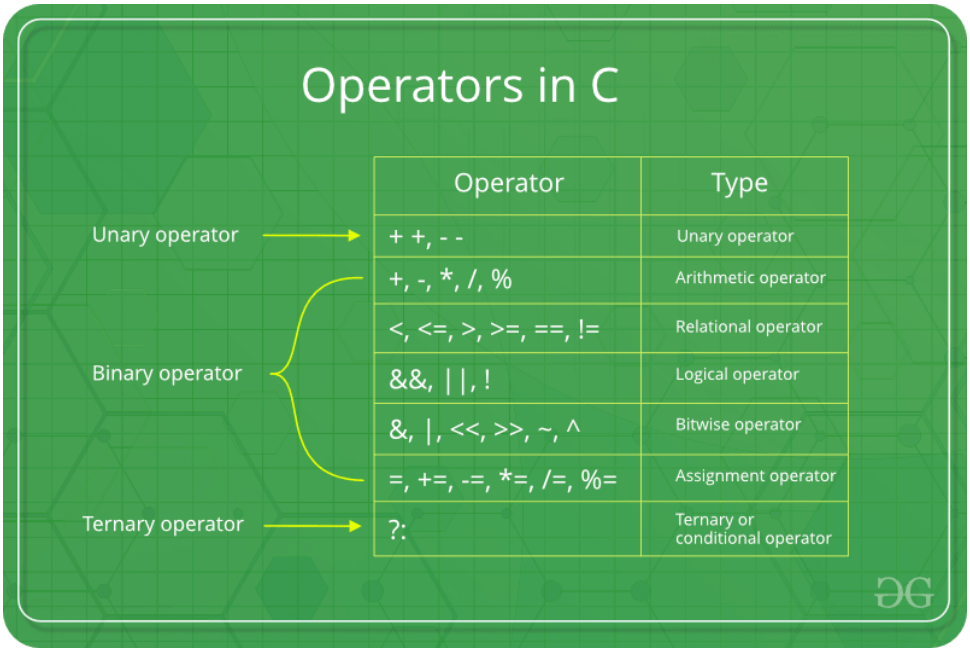

Source code
(e.g. my.cpp)
Preprocessor
Preprocessed
Source Code
Compiler
Object Code
(my.obj)
Library
Linker
Executable Code
(my.exe)

Namespace


Namespace
Schema1
Object1
Item1
Something1
Schema2
Item
Object
Schema_n
Object1
Item2
Item_n
Object_n



Operations in C

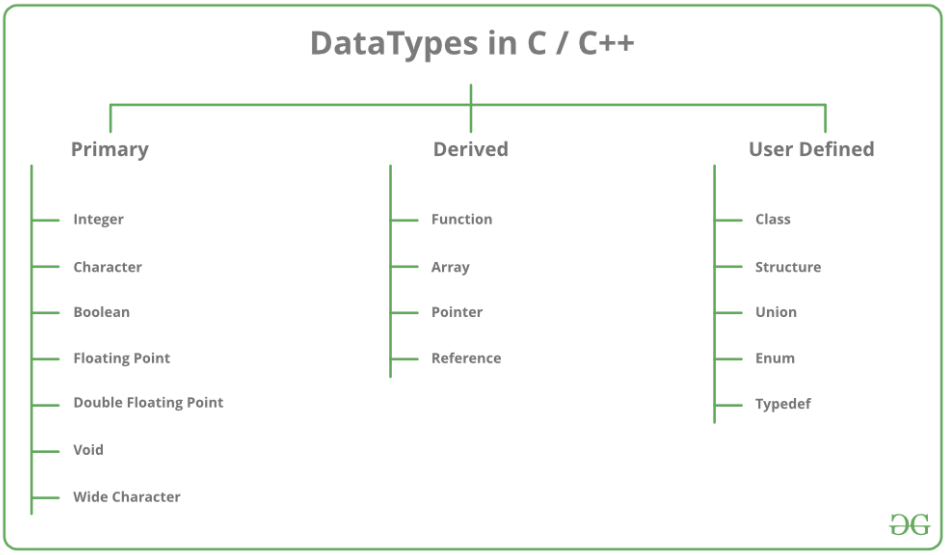


Priority of Operations

	Operator Precedence
1	! Logical not (Highest)
2	( ) Parenthesis
3	*, /, %
4	+, -
5	>, >=, <, <=
6	==, !=
7	&& (AND)
8	|| (OR)
9	= (Lowest)

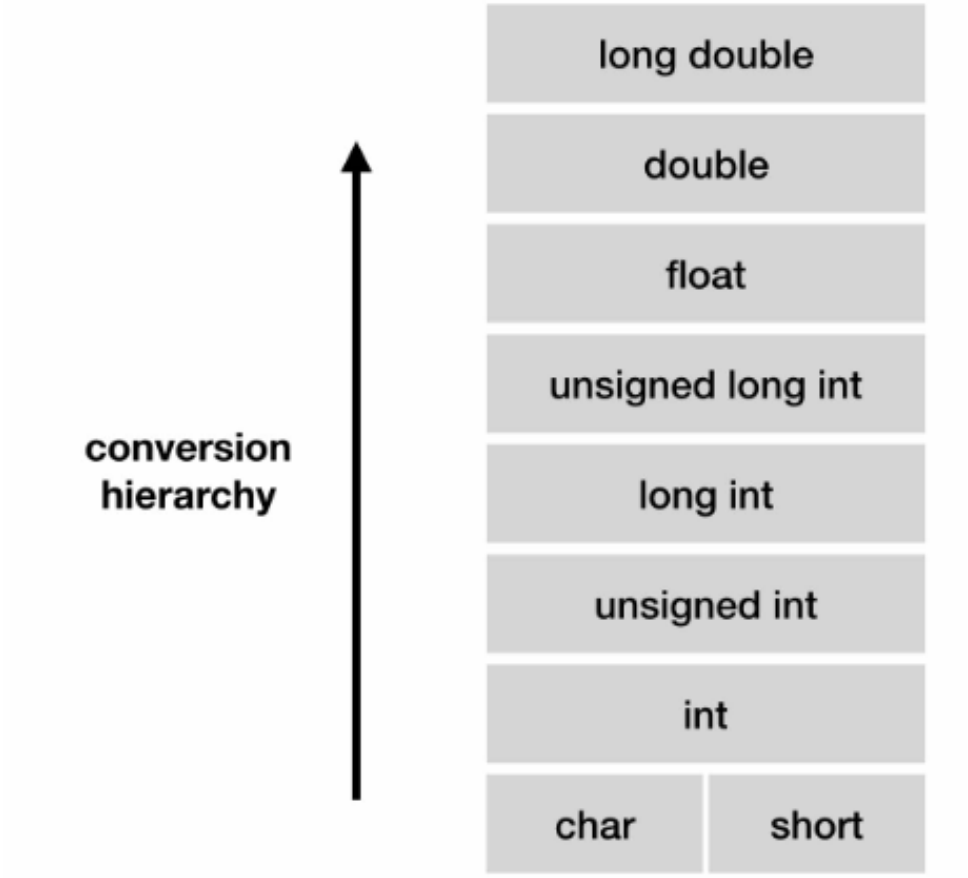
Signs of operations	Name of operation, explanation	Associativity
`() [] . ->`	Primary	From left to right
`+ - ~ ! * & ++ --` `sizeof(type)` `(type cast)`	Unary	From right to left
`* / %`	Multiplicative, arithmetical, binary	From left to right
`+ -`	Additive, arithmetical, binary	From left to right
`>> <<`	Shift	From left to right
`< > <= >=`	Relation	From left to right
`== !=`	Relation	From left to right
`&`	Bitwise "AND", logical, binary	From left to right
`^`	Bitwise XOR, logical, binary	From left to right
`|`	Bitwise logical "OR", logical, binary	From left to right
`&&`	Logical "AND", binary	From left to right
`||`	Logical "OR", binary	From left to right
`?:`	Conditional, ternary	From right to left
`= *= /= %= += -=` `<<= >>= &= |= ^=`	Simple and complex assignment	From right to left
`,`	Sequential computation	From left to right

Data Type



Key word	Size in bytes	Interpretation	Possible values
bool	1	boolean	true and false
unsigned char	1	Unsigned character	0 to 255
char (or signed char)	1	Signed character	-128 to 127
wchar_t	2	Wide character (in windows, same as unsigned short)	0 to 2^{16}-1
short (or signed short)	2	Signed integer	-2^{15} to 2^{15} - 1
unsigned short	2	Unsigned short integer	0 to 2^{16}-1
int (or signed int)	4	Signed integer	-2^{31} to 2^{31}-1
unsigned int	4	Unsigned integer	0 to 2^{32} - 1
Long (or long int or signed long)	4	signed long integer	-2^{31} to 2^{31}-1
unsigned long	4	unsigned long integer	0 to 2^{32} -1
float	4	Signed single precision floating point (23 bits of significand, 8 bits of exponent, and 1 sign bit.)	$3.4*10^{-38}$ to $3.4*10^{38}$ (both positive and negative)
long long	8	Signed long long integer	-2^{63} to 2^{63} -1
unsigned long long	8	Unsigned long long integer	0 to 2^{64} -1
double	8	Signed double precision floating point(52 bits of significand, 11 bits of exponent, and 1 sign bit.)	$1.7*10^{-308}$ to $1.7*10^{308}$ (both positive and negative)
long double	8	Signed double precision floating point(52 bits of significand, 11 bits of exponent, and 1 sign bit.)	$1.7*10^{-308}$ to $1.7*10^{308}$ (both positive and negative)

Casting



Introduction to C++

C++ Casting Operators

- **static_cast<type> (expr):**
- **static_cast operator** is used to convert a given expression to the specified type. For example, it can be used to cast a base class pointer into a derived class pointer.

```
int main() {
    int a = 31;
    int b = 3;
    float x = a/b;
    float y = static_cast<float>(a)/b;
    cout << "Output without static_cast = " << x << endl;
    cout << "Output with static_cast = " << y << endl;
}
```

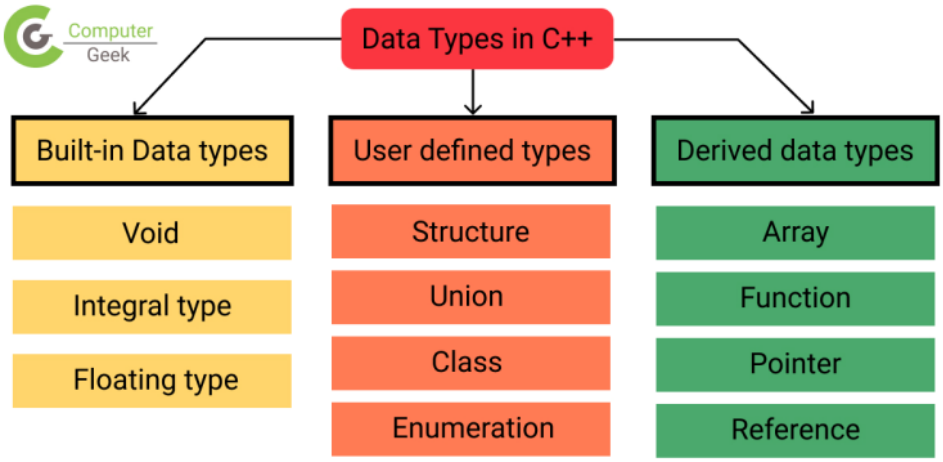
In this type casting example, using the static_cast to an integer as "float" returns a float value.

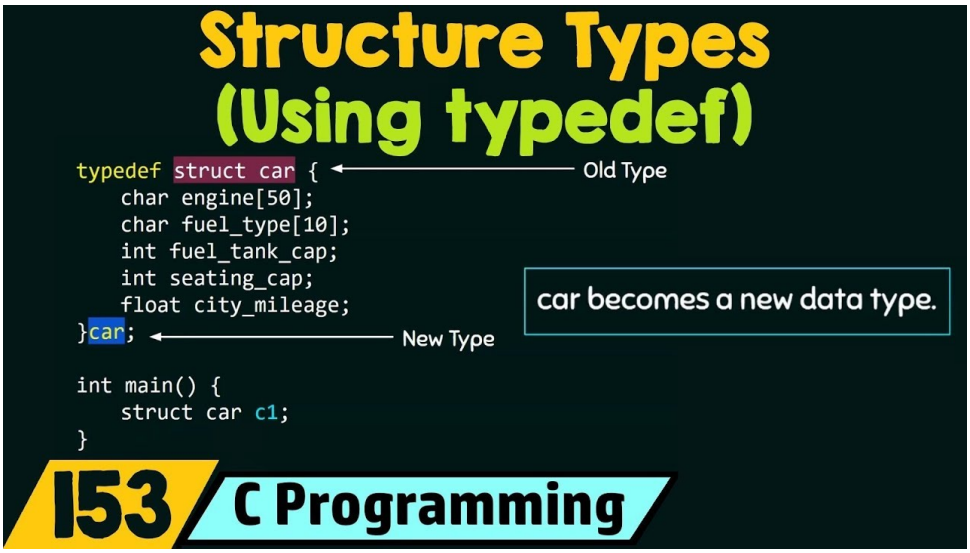
Lecture Slides By Adil Aslam

Explicit type conversion

- C++ casts
  - `static_cast` between 2 related types (int/float, int/enum, 2 pointers in class hierarchy)
  - `reinterpret_cast` between 2 unrelated types (int/ptr, pointers to 2 unrelated classes)
  - `const_cast` cast away constness
  - `dynamic_cast` used for polymorphic types Run-time type info (RTTI)
- Avoid casts, but use these instead of C casts
  - e.g., compiler can perform minimal checking for `static_cast`, none for `reinterpret_cast`

Struct in C and Class in C++





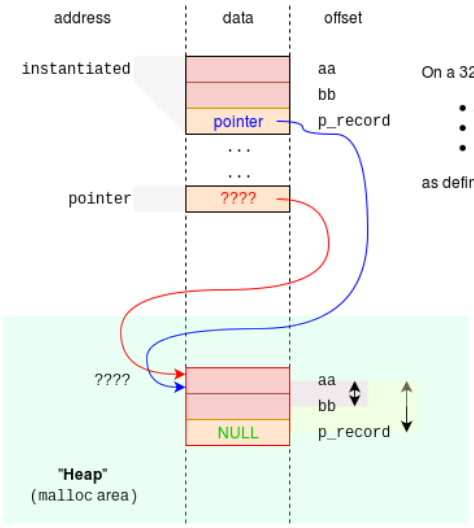
Structure

Structures within Structures

1. C define a variable of structure type as a member of other structure type.
2. The syntax to define the structure within structure is

```
struct <struct_name>{
    <data_type> <variable_name>;
    struct <struct_name>
            { <data_type> <variable_name>;
            ……..}<struct_variable>;
    <data_type> <variable_name>;
};
```

Use of `struct` in C



Offsets

On a 32-bit architecture (32-bit 'int's):

- aa = 0
- bb = 4
- p_record = 8

as defined by `typedef`.

Code

```
typedef struct record_t
  {
  int aa;
  int bb;
  struct record_t *p_record;
  } record;

record instantiated;
record *pointer;

...

pointer = malloc(sizeof(record));
pointer->p_record = NULL;
instantiated.p_record = pointer;
```

```
class myClass {
  public:
    int myNum;
  private:
    string myStr;
  protected:
    float myFloat;
```

Data and Access Modifiers

Access Modifiers are used to assign the accessibility to the class members.

```
  public:
    myClass() {
      ...
    }

    myClass(int x, string y){
        ...
    }
```

Constructors

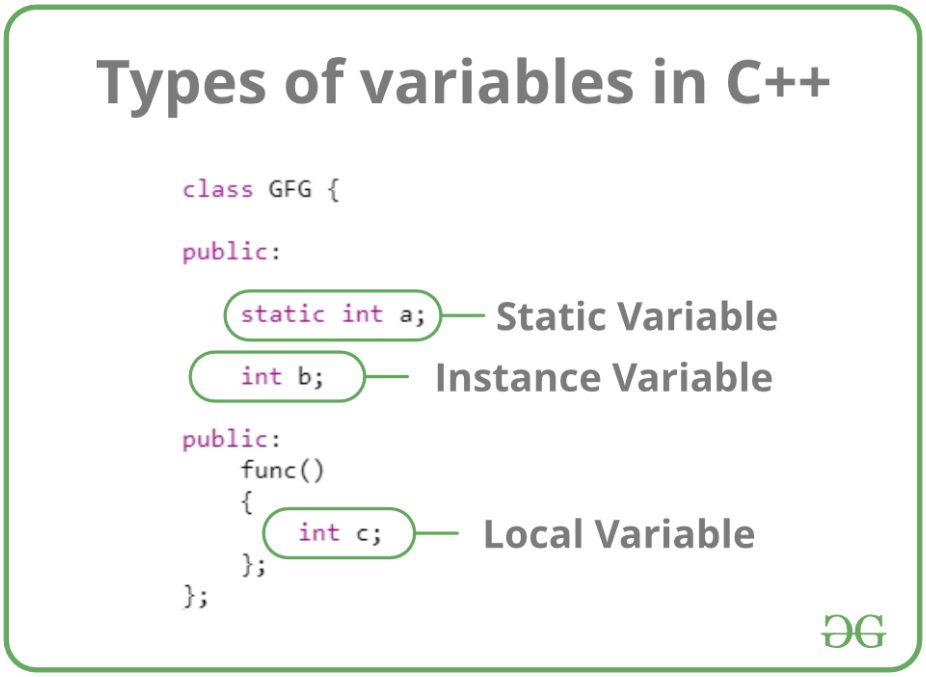
A constructor is a member function which initialises objects of a class.

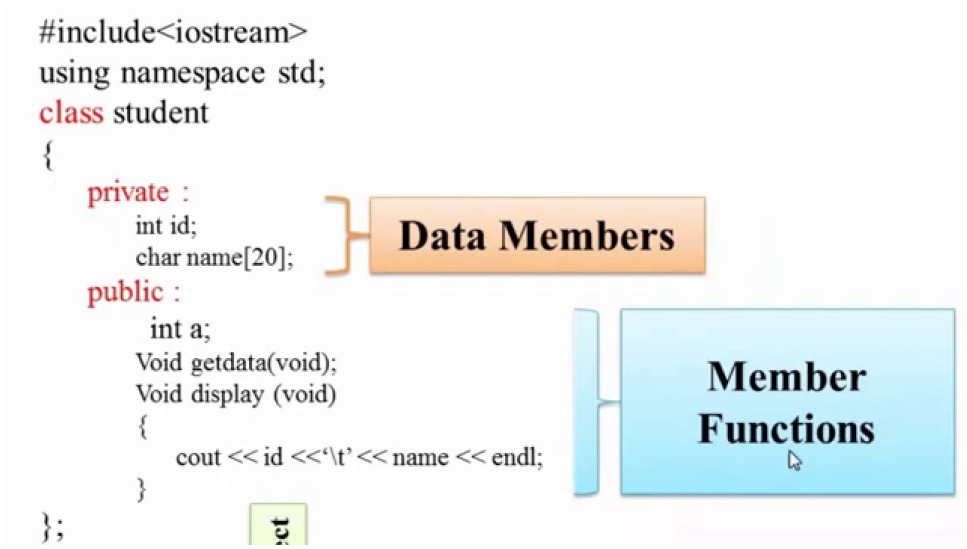
```
  private:
    void myMethodOne() {
      ...
    }

  protected:
    void myMethodTwo() {
        ...
    }
};
```

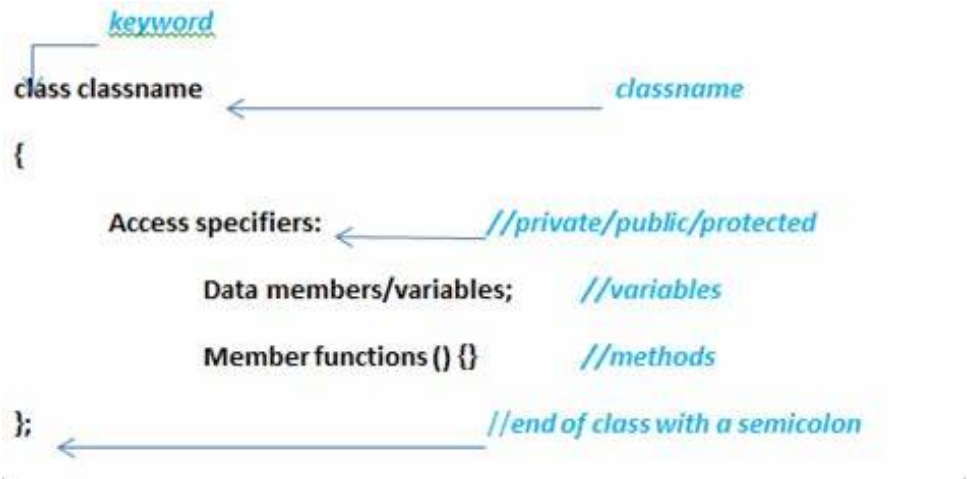
Methods

Methods are functions that perform operations on the class data.

enjoyalgorithms.com

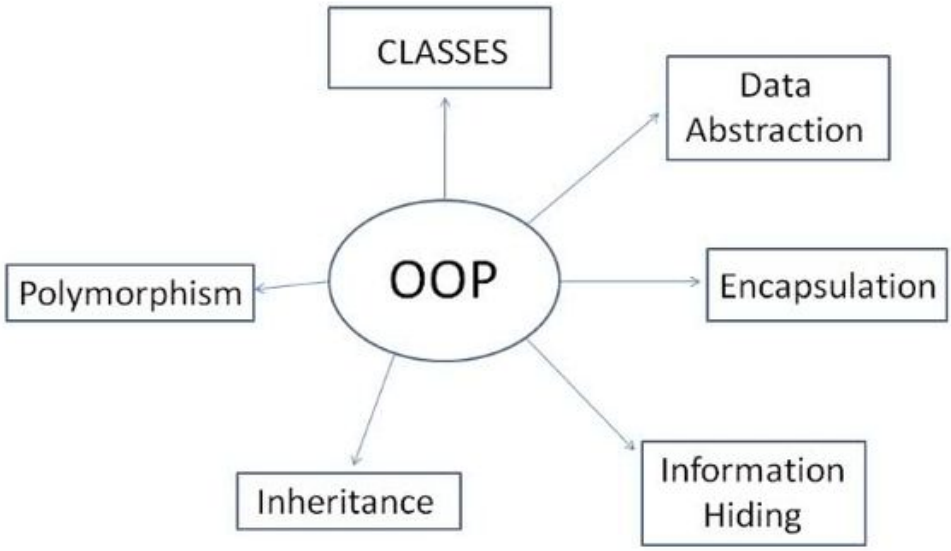

Types of variables in C++
class GFG {
public:
static int a;
Static Variable
int b;
Instance Variable
public:
func()
{
int c;
Local Variable
};
};


#include<iostream>
using namespace std;
class student
{
private :
int id;
char name[20];
Data Members
public :
int a;
Void getdata(void);
Void display (void)
{
cout << id <<'\t' << name << endl;
}
};
Member Functions


keyword
class classname
classname
{
Access specifiers:
//private/public/protected
Data members/variables;
//variables
Member functions () {}
//methods
};
//end of class with a semicolon

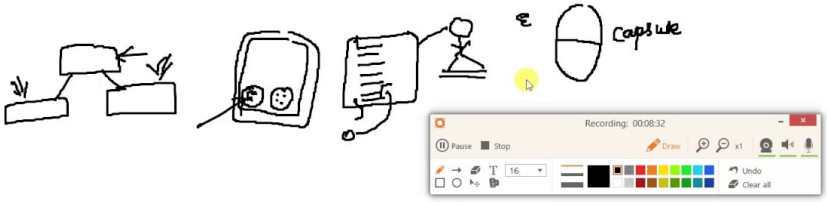
Structure	Class
It is a value type.	It is a reference type.
Its object is created on the stack memory.	Its object is created on the heap memory.
It does not support inheritance.	It supports inheritance.
The member variable of structure cannot be initialized directly.	The member variable of class can be initialized directly.
It can have only parameterized constructor.	It can have all the types of constructor and destructor.

OOP



Four Pillars of Object Oriented Programming.

- **Inheritance:** Process of creating new classes from existing classes.
- **Abstraction:** Refers to providing only essential information and hiding the background details from the end user.
- **Encapsulation:** Binding together of data and functions that manipulate that data and keep both safe from outside interference and misuse.
- **Polymorphism:** Refers to one interface and many forms.



Data Abstraction

What is Data Abstraction?

- A data abstraction is a simplified view of an object that
  - includes only features one is interested in.
  - while hides away the unnecessary details.
- In programming languages, a data abstraction becomes an abstract data type (ADT) or a user-defined type.
- In OOP, it is implemented as a class

Data Abstraction in C++ (continued)

(11.14)

- **Constructors**
  - functions to initialize the data members of instances
  - may also allocate storage if part of the object is heap-dynamic
  - can include parameters to provide parameterization of the objects
  - implicitly called when an instance is created
    - » can be explicitly called
  - name is the same as the class name
- **Destructors**
  - functions to cleanup after an instance is destroyed; usually just to reclaim heap storage
  - implicitly called when the object's lifetime ends
    - » can be explicitly called
  - name is the class name, preceded by a tilda (~)

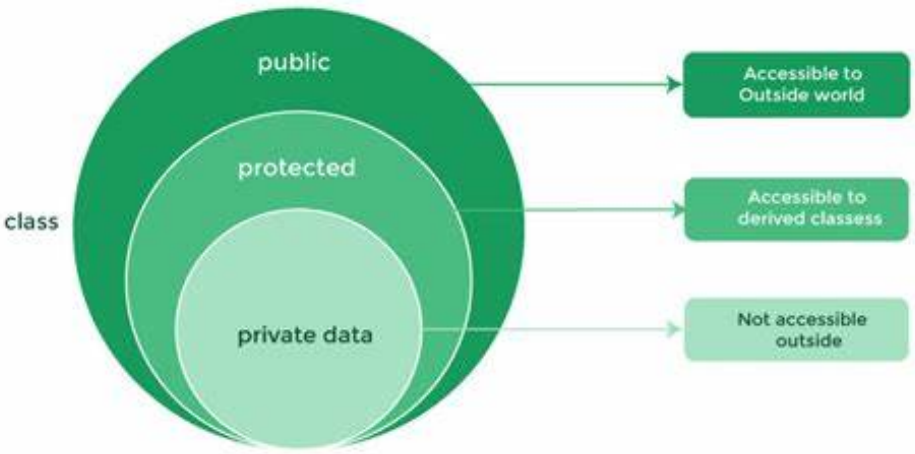
Encapsulation

Access Control

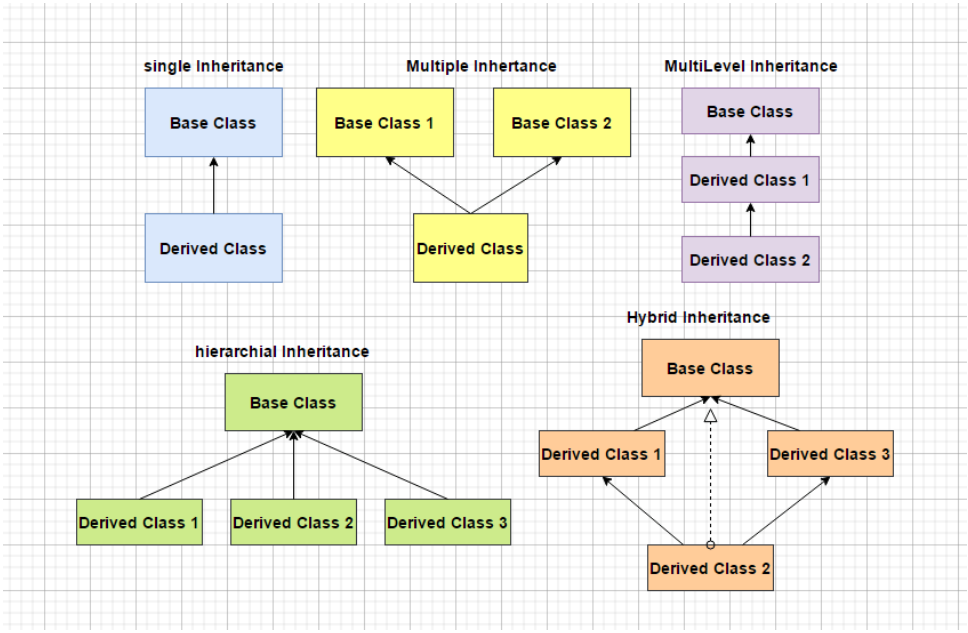
- We use the **private section** of the struct to **encapsulate** our data type as in our ADTs in C
- But if the struct is defined in the header file, doesn't this **break encapsulation**?
  - No! Encapsulation is not about *hiding information* from the user, it's about ensuring that he uses our ADT **in a safe way**, which we achieve by enabling him to use only its public interface
- However, unlike ADTs in C style, we do not get **compile-time** encapsulation
  - In other words, changing the private part of a struct will require **recompiling files** that use the ADT

Encapsulation

- **Encapsulation** is the packing of data and functions into a single component. The features of **encapsulation** are supported using classes in most object-oriented programming languages, although other alternatives also exist.
- Encapsulation is:
  - A language mechanism for restricting access to some of the object's components. (public, private, protected)
  - A language construct that facilitates the bundling of data with the methods (or other functions) operating on that data.

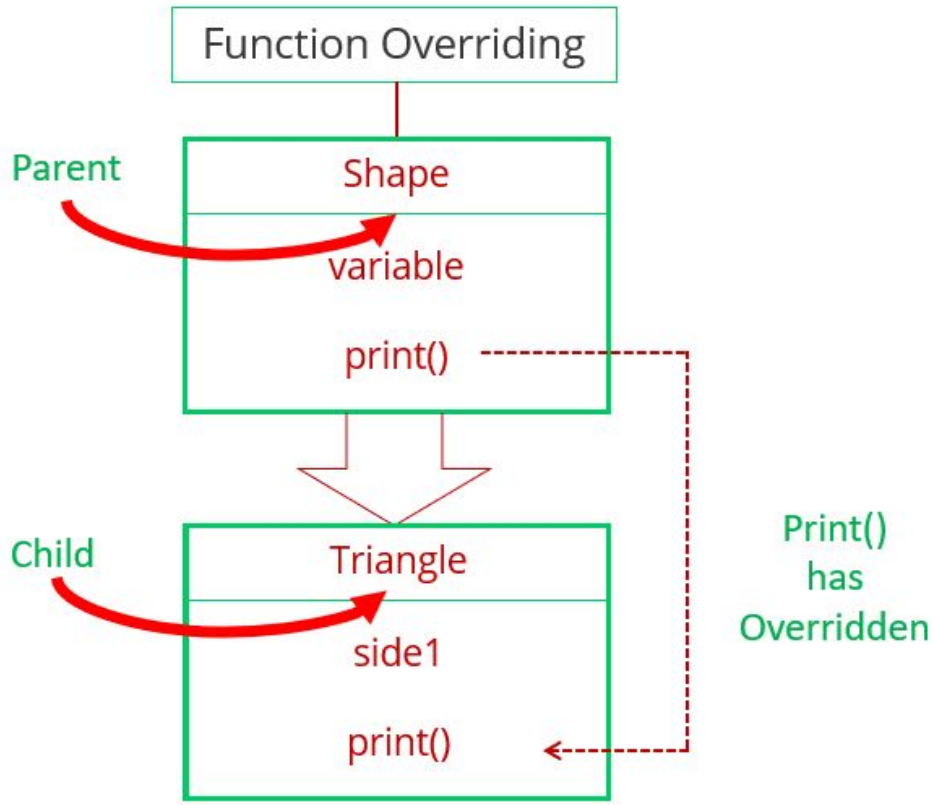


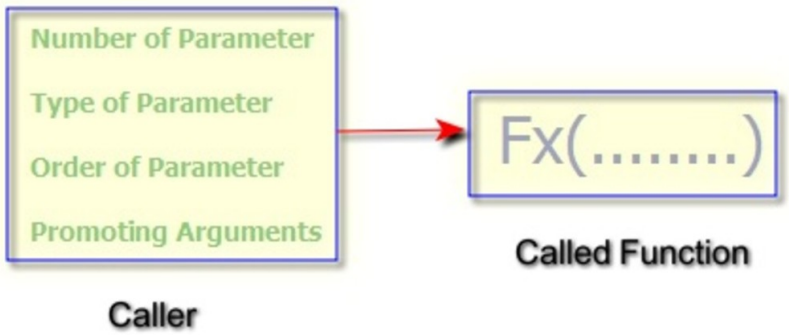
Inheritance



Base-class member-access specifier	Type of inheritance		
	`public` inheritance	`protected` inheritance	`private` inheritance
`public`	`public` in derived class. Can be accessed directly by member functions, `friend` functions and nonmember functions.	`protected` in derived class. Can be accessed directly by member functions and `friend` functions.	`private` in derived class. Can be accessed directly by member functions and `friend` functions.
`protected`	`protected` in derived class. Can be accessed directly by member functions and `friend` functions.	`protected` in derived class. Can be accessed directly by member functions and `friend` functions.	`private` in derived class. Can be accessed directly by member functions and `friend` functions.
`private`	Hidden in derived class. Can be accessed by member functions and `friend` functions through `public` or `protected` member functions of the base class.	Hidden in derived class. Can be accessed by member functions and `friend` functions through `public` or `protected` member functions of the base class.	Hidden in derived class. Can be accessed by member functions and `friend` functions through `public` or `protected` member functions of the base class.

Overriding vs Overloading

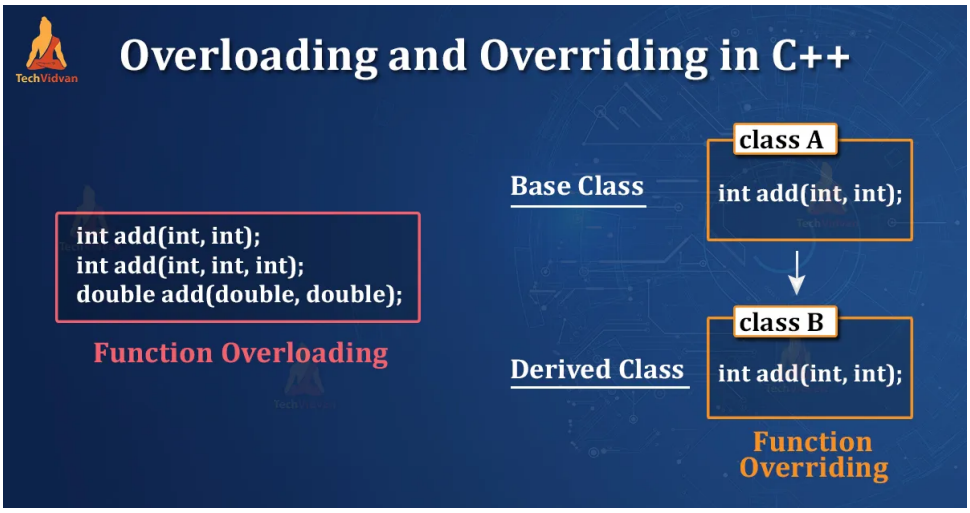

Function Overriding
Parent
Shape
variable
print()
Child
Triangle
side1
print()
Print()
has
Overridden


Number of Parameter
Type of Parameter
Order of Parameter
Promoting Arguments
Caller
Fx(.....)
Called Function

Overloading in C++

- What is overloading
  - Overloading means assigning multiple meanings to a function name or operator symbol
  - It allows multiple definitions of a function with the same name, but different signatures.
- C++ supports
  - Function overloading
  - Operator overloading

3 Ritika sharma

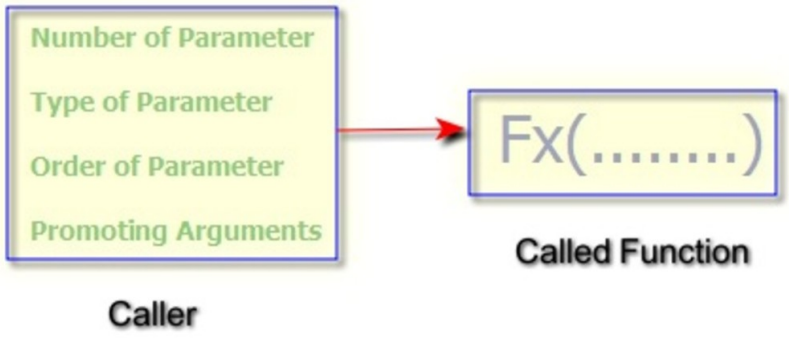


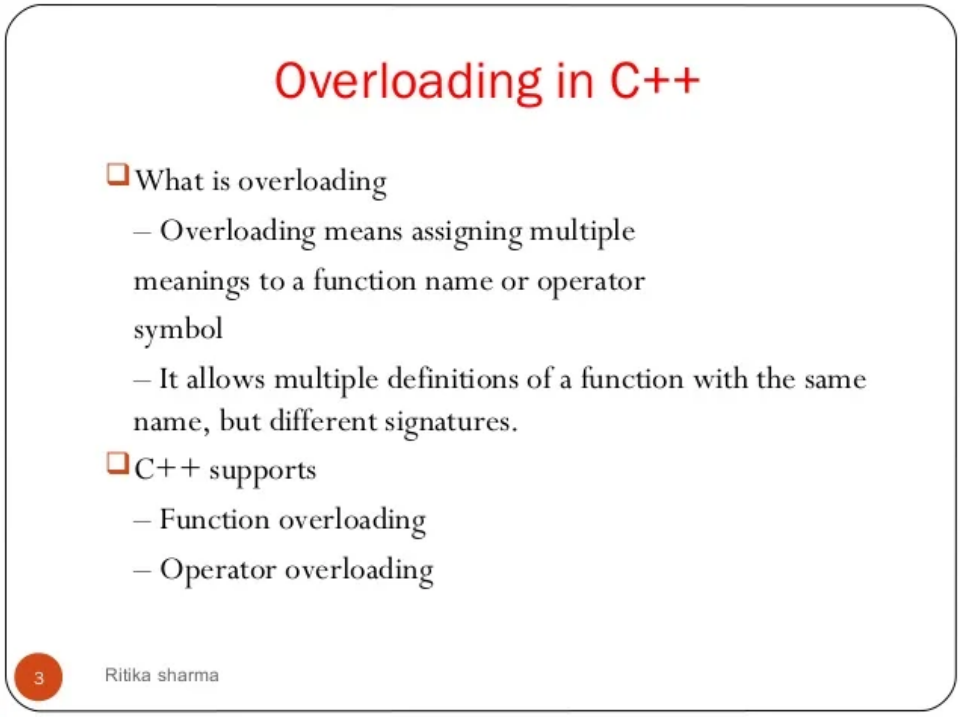
Overloading vs. Overriding

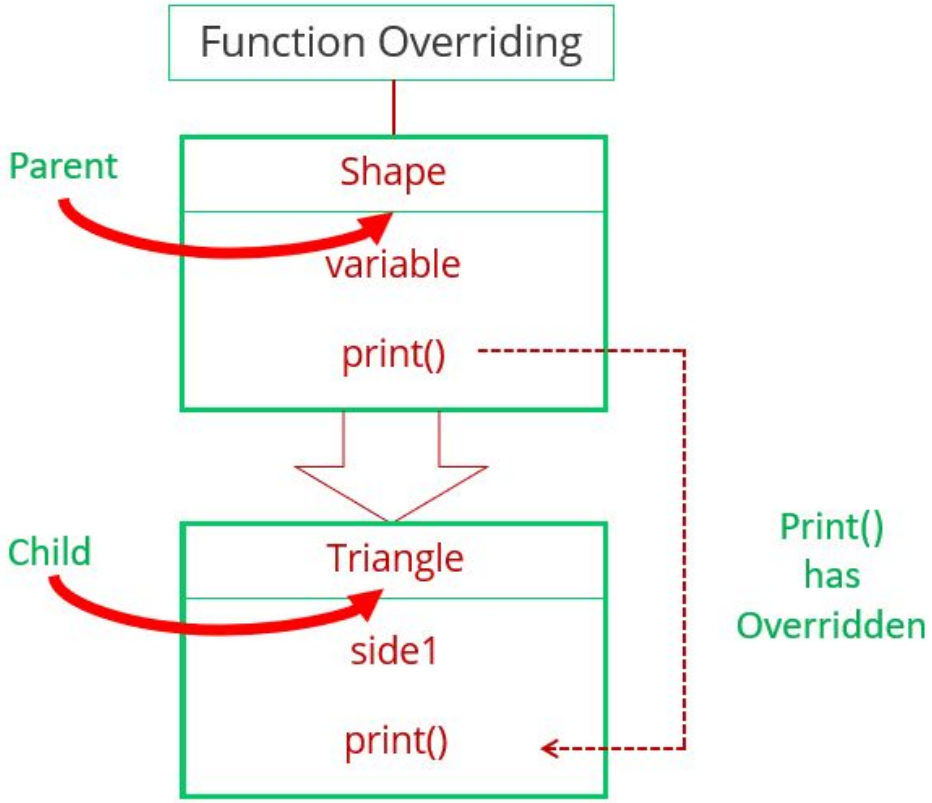
- Overriding a base class member function is similar to overloading a function or operator
  - But for overriding, definitions are distinguished by their scopes rather than by their signatures
- C++ can distinguish method definitions according to either static or dynamic type
  - Depends on whether a method is virtual or not
  - Depends on whether called via a reference or pointer vs. directly on an object
  - Depends on whether the call states the scope explicitly (e.g., `Foo::baz();`)

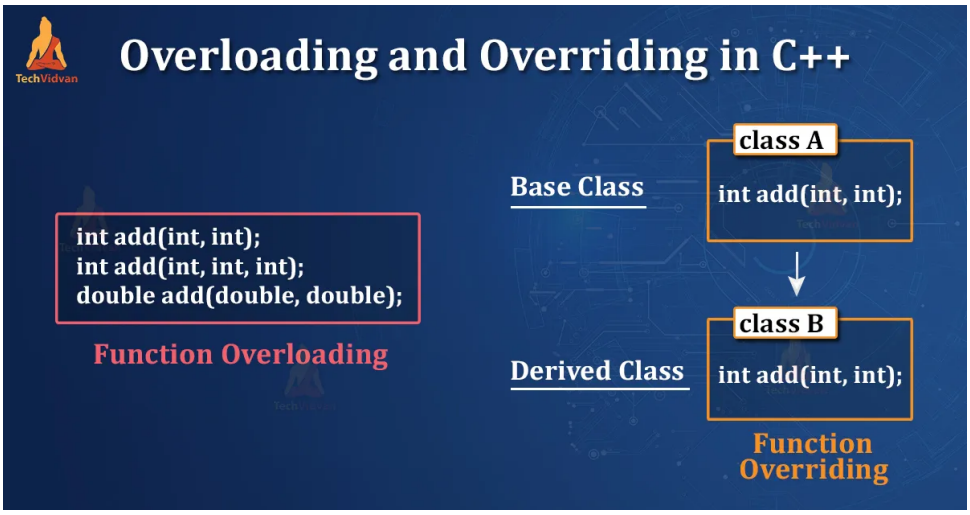
CSE 332: C++ Overloading

Overloading vs Overriding






Function Overriding
Parent
Shape
variable
print()
Child
Triangle
side1
print()
Print()
has
Overridden


TechVidvan
Overloading and Overriding in C++
int add(int, int);
int add(int, int, int);
double add(double, double);
Function Overloading
Base Class
class A
int add(int, int);
Derived Class
class B
int add(int, int);
Function
Overriding

Overloading vs. Overriding

- Overriding a base class member function is similar to overloading a function or operator
  - But for overriding, definitions are distinguished by their scopes rather than by their signatures
- C++ can distinguish method definitions according to either static or dynamic type
  - Depends on whether a method is virtual or not
  - Depends on whether called via a reference or pointer vs. directly on an object
  - Depends on whether the call states the scope explicitly (e.g., `Foo::baz();`)

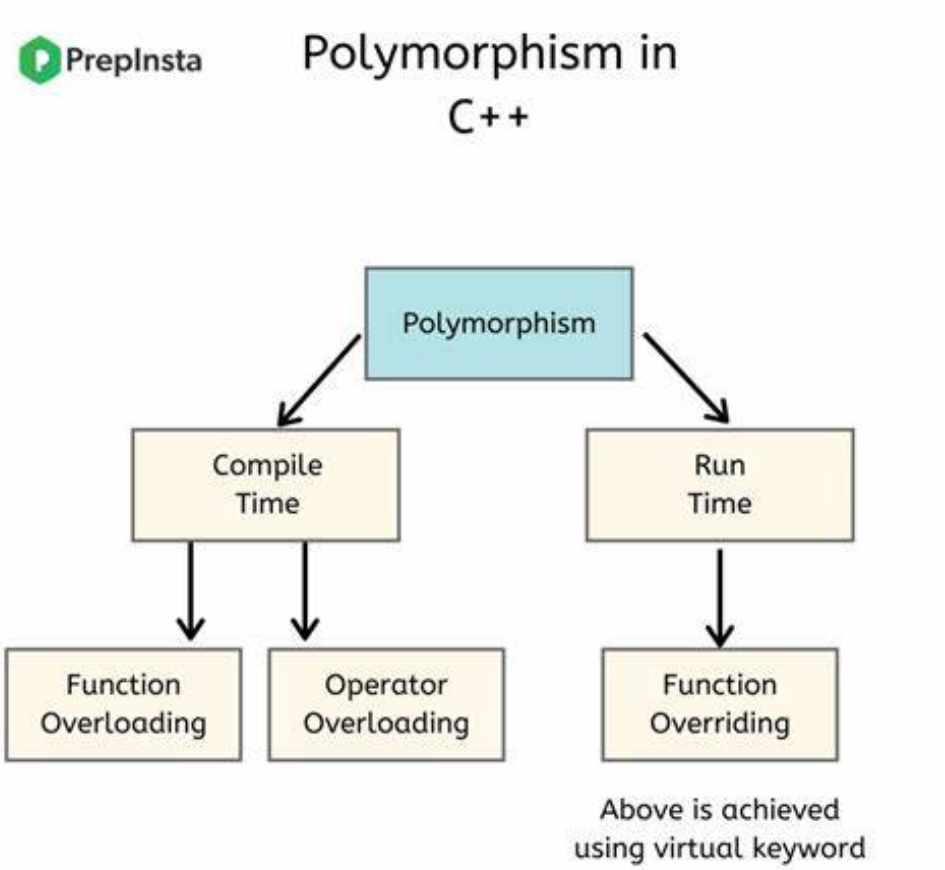
CSE 332: C++ Overloading

Polymorphism

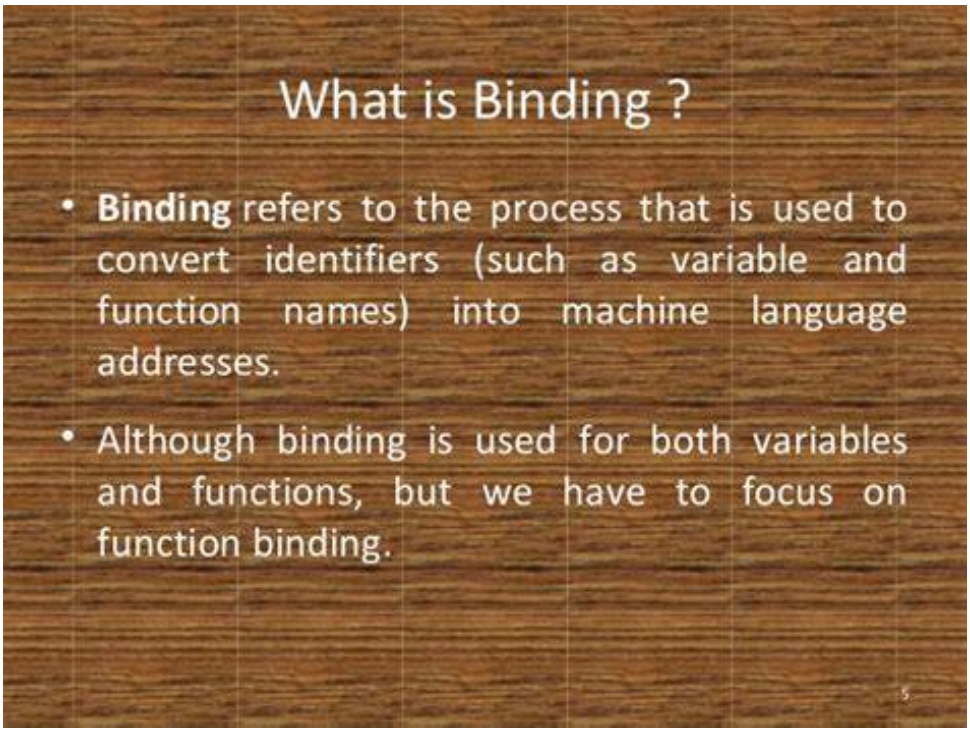
Overview of C++ Polymorphism

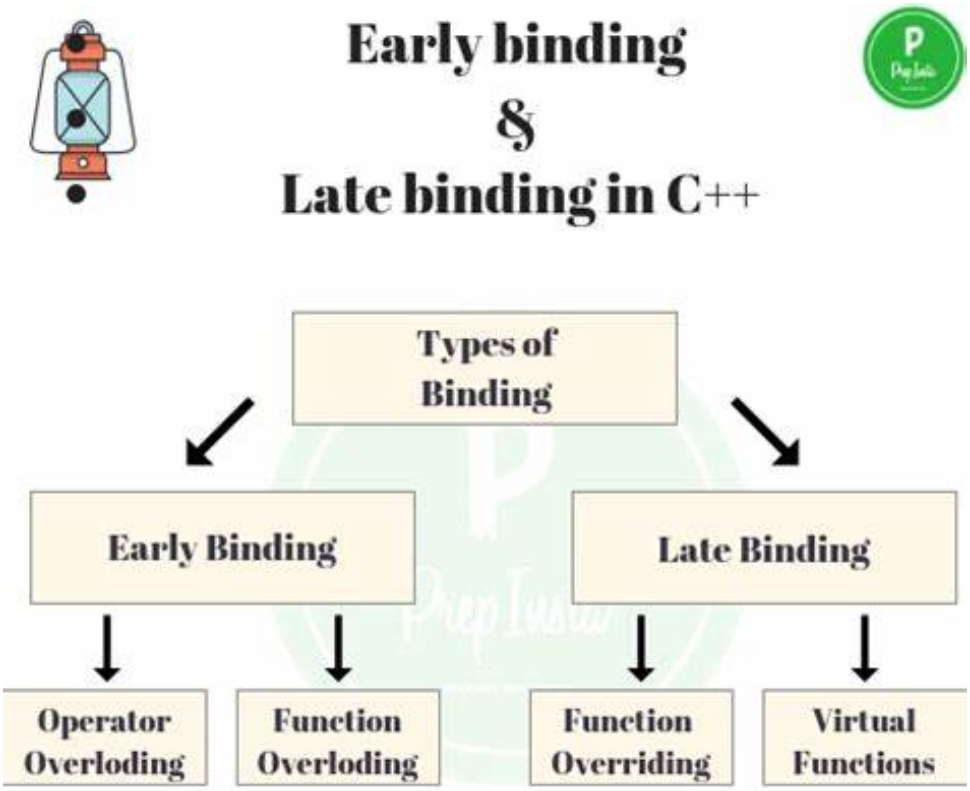
- Two main kinds of types in C++: native and user-defined
  - "User" defined types: declared classes, structs, unions
    - including types provided by the C++ standard libraries
  - Native types are "built in" to the C++ language itself: int, long, float, …
  - A typedef creates a new type *name* for another type (type aliasing)
- Public inheritance creates sub-types
  - Inheritance only applies to user-defined classes (and structs)
  - A publicly derived class is-a subtype of its base class
  - Known as "inheritance polymorphism"
- Template parameters also induce a subtype relation
  - Known as "interface polymorphism"
  - We'll cover how this works in depth, in later sessions
- Liskov Substitution Principle (for both kinds of polymorphism)
  - if S is a subtype of T, then wherever you need a T you can use an S

CSE 332: C++ Polymorphism

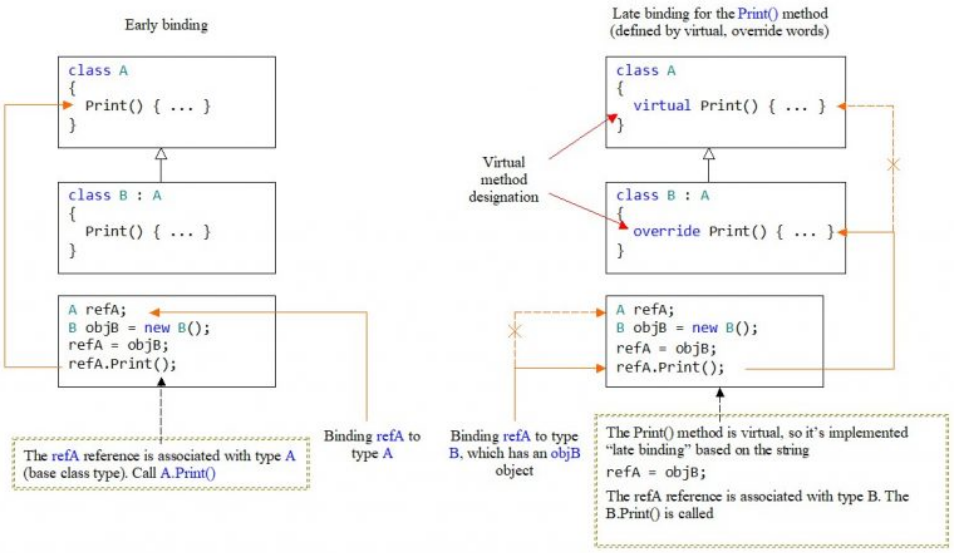


Early Binding and Late Binding

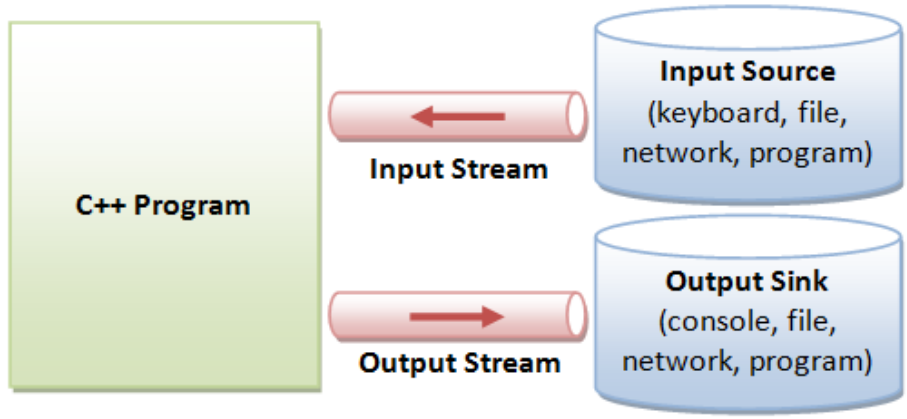




Sr. No.	Early Binding	Late Binding
1.	Early binding is the process of linking a function with an object during the compilation process.	Late binding is a run-time polymorphism with method overriding.
2.	Static binding is another name for early binding.	Dynamic binding is another name for late binding.
3.	Early binding happens at compile time.	Late binding happens at run time.
4.	Execution speed is faster in early binding.	Execution speed is lower in late binding.
5.	The class information is used by Early binding to resolve method calls.	The object is used by Late binding to resolve method calls.



File Stream

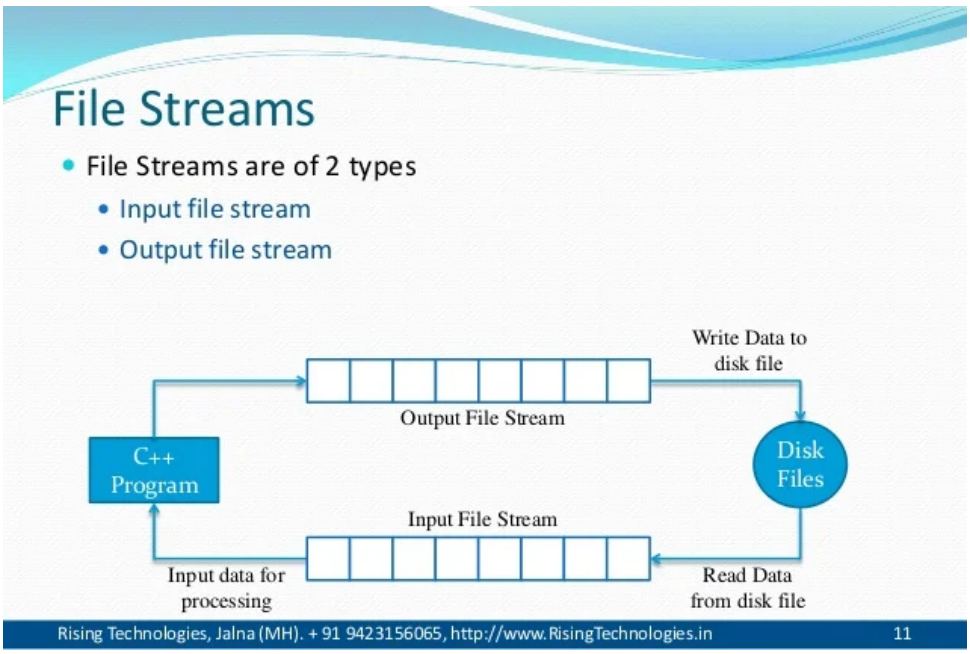


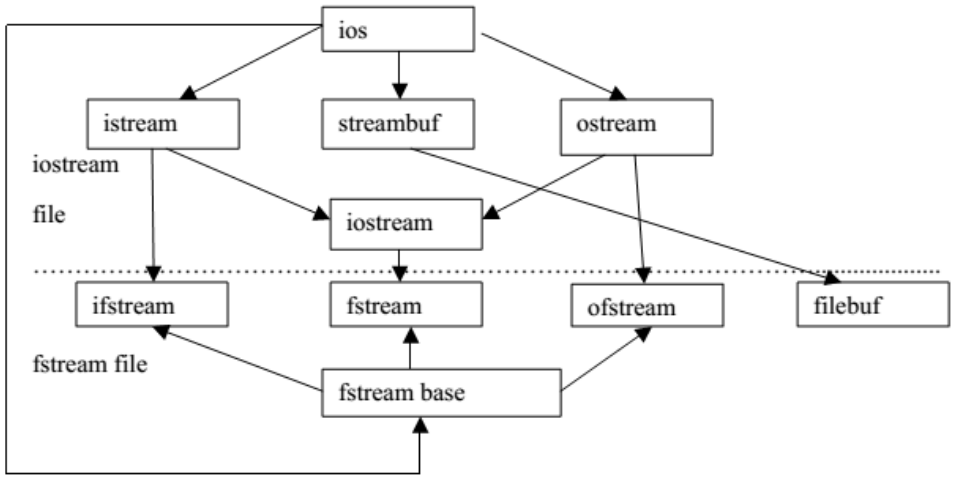
Internal Data Formats:

- Text: char, wchar_t
- int, float, double, etc.

External Data Formats:

- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)





Stream classes for file operations

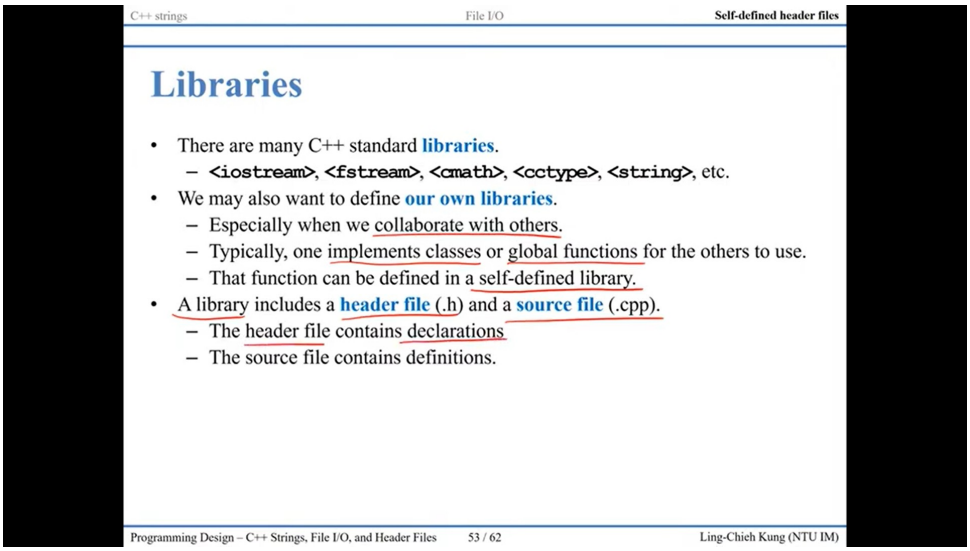
Object Oriented Programming in C++

File Opening Mode

File Mode Parameter	Meaning
ios::app	Append mode. All output to that file to be appended to the end.
ios::ate	Open a file for output and move the read/write control to the end of the file.
ios::binary	file open in binary mode
ios::in	open file for reading only
ios::out	open file for writing only
ios::nocreate	open fails if the file does not exist
ios::noreplace	open fails if the file already exist
ios::trunc	delete the contents of the file if it exist

Lecture Slides By Adil Aslam

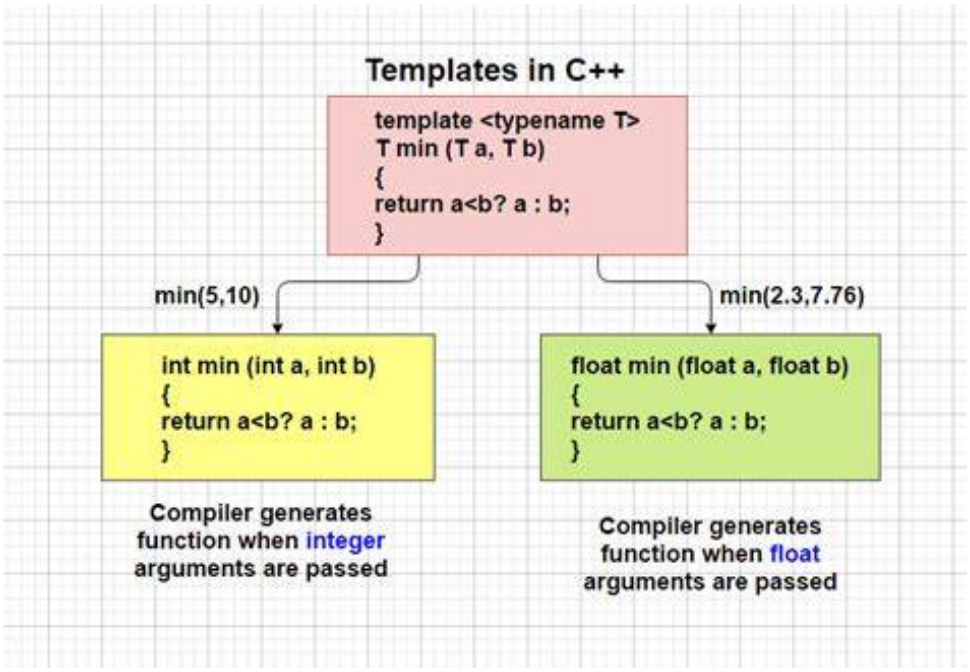
Self-defined Library



Including a header and a source file

- When your main program also wants to include a self-defined source file, the include statement needs not be changed.
  - `#include "myMax.h"`
- We add a source file myMax.cpp.
  - In the source file, we **implement** those functions declared in the header file.
  - The main file names of the header and source files can be different.
- The two source files (main.cpp and myMax.cpp) must be **compiled together**.
  - Each environment has its own way.

Templates



Template declaration

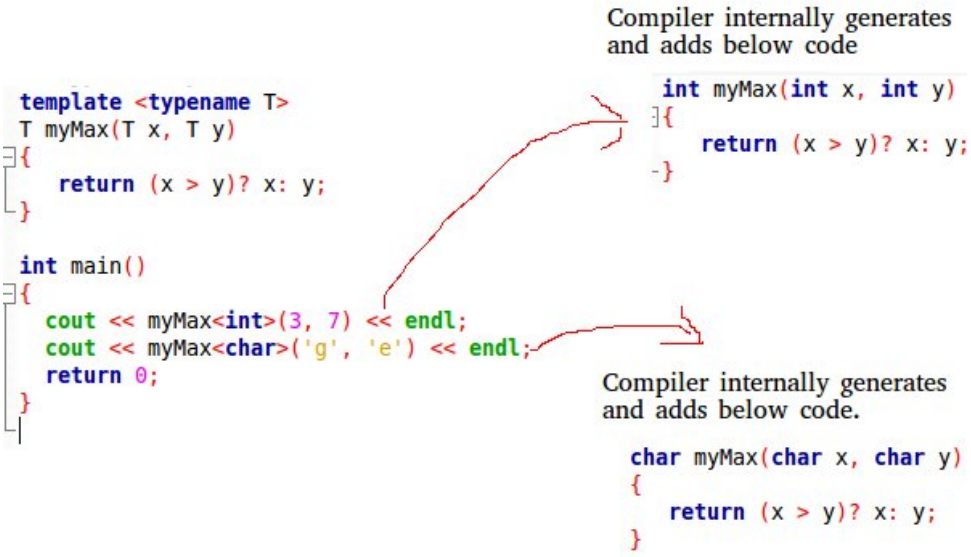
- To declare a type parameter, use the keywords `template` and `typename`.

```
template<typename T>
class TheClassName
{
  // T can be treated as a type inside the class definition block
};
```

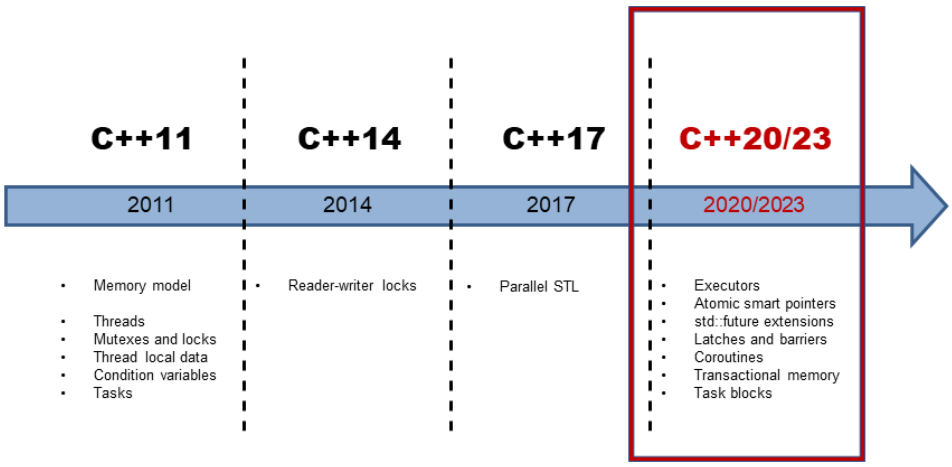
  - Some old codes write **`class`** instead of **`typename`**. Both are fine.
- We then do this to all member functions:

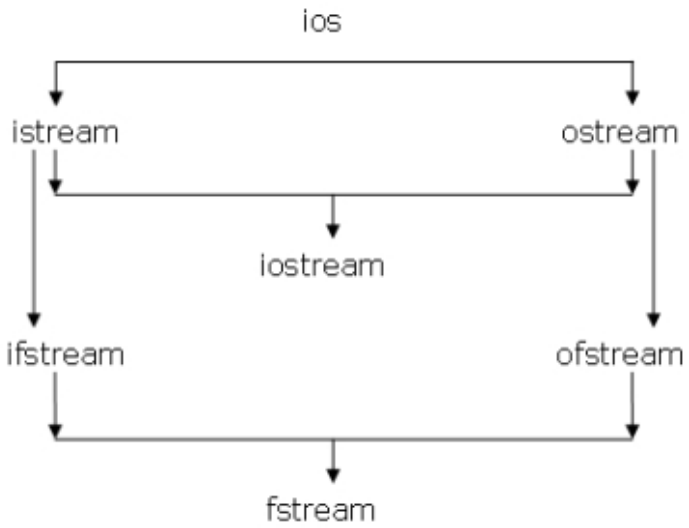
```
template<typename T>
T TheClassName<T>::f(T t)
{
  // t is a variable whose type is T
};
```

```
template<typename T>
void TheClassName<T>::f(int i)
{
  // follow the rule even if T is not used
};
```

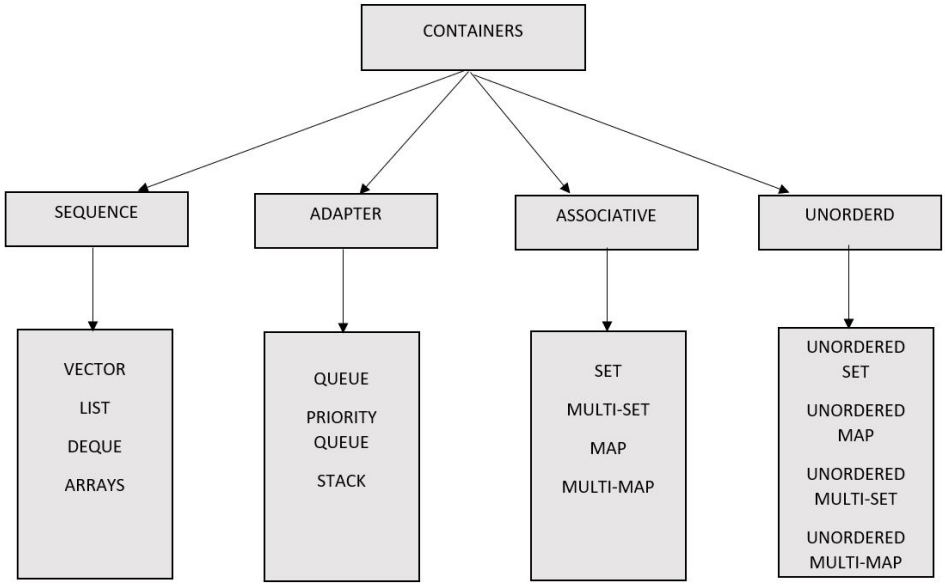


C++ SL(Strandard Library)





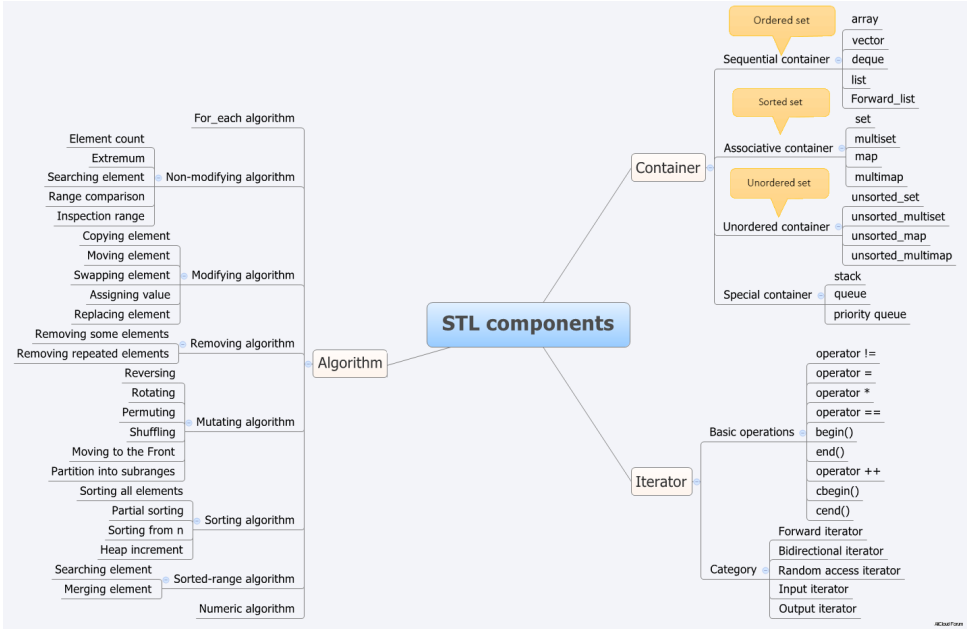
C++ STL(Strandard Template Library)

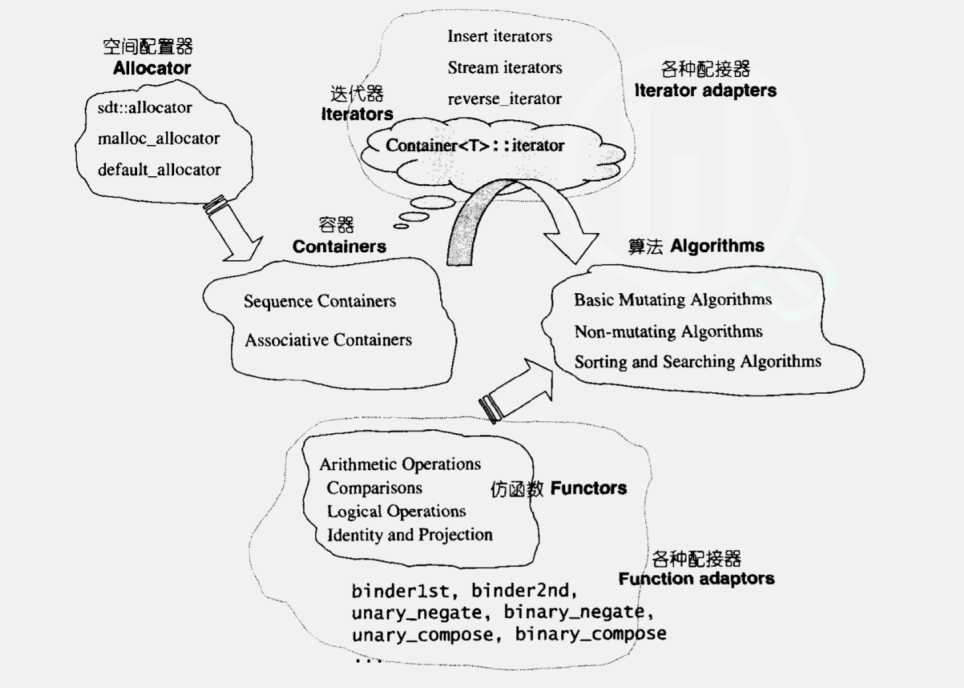


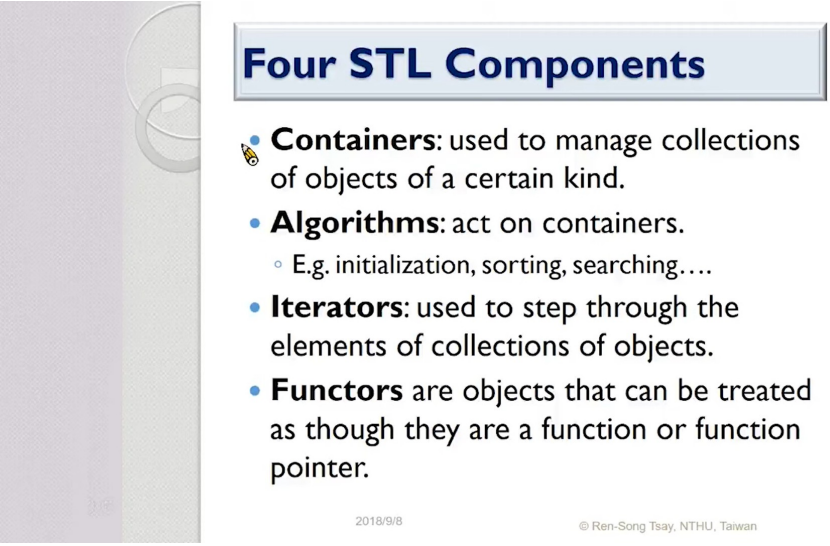
Intro to the C++ Standard Template Library (STL)

- The STL is a collection of related software elements
  - Containers
    - Data structures: store values according to a specific organization
  - Iterators
    - Variables used to give flexible access to the values in a container
  - Algorithms
    - Functions that use iterators to access values in containers
    - Perform computations that modify values, or creates new ones
  - Function objects
    - Encapsulate a function as an object, use to modify an algorithm
- The STL makes use of most of what we've covered
  - Extensive use of function and class templates, concepts
- The STL makes use of several new ideas too
  - typedefs, traits, and associated types

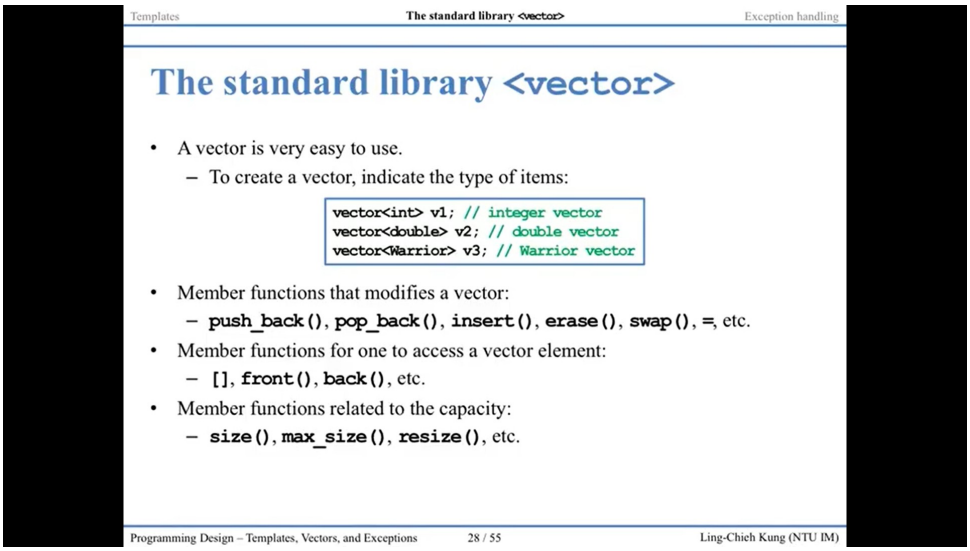
CSE 332: C++ STL containers

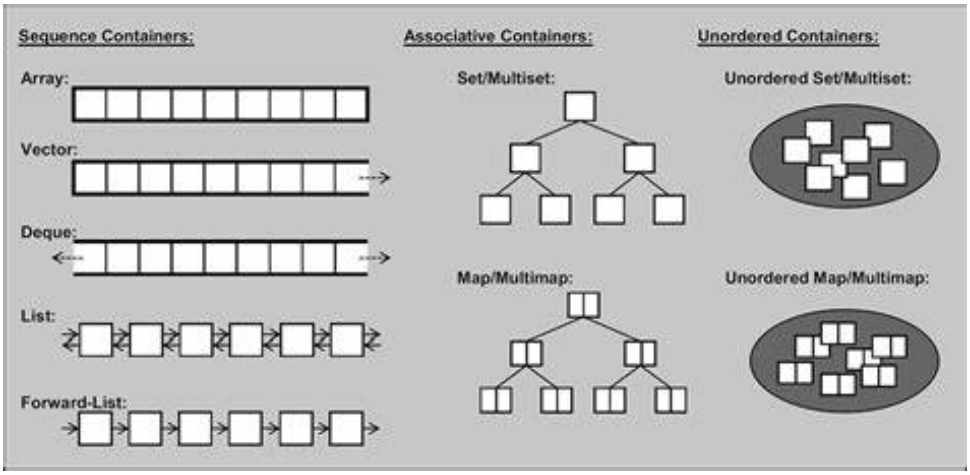



空间配置器
Allocator
sdt::allocator
malloc_allocator
default_allocator
迭代器
Iterators
Insert iterators
Stream iterators
reverse_iterator
Container<T>::iterator
各种配接器
Iterator adapters
容器
Containers
Sequence Containers
Associative Containers
算法 Algorithms
Basic Mutating Algorithms
Non-mutating Algorithms
Sorting and Searching Algorithms
Arithmetic Operations
Comparisons
Logical Operations
Identity and Projection
仿函数 Functors
各种配接器
Function adaptors
binder1st, binder2nd,
unary_negate, binary_negate,
unary_compose, binary_compose
...

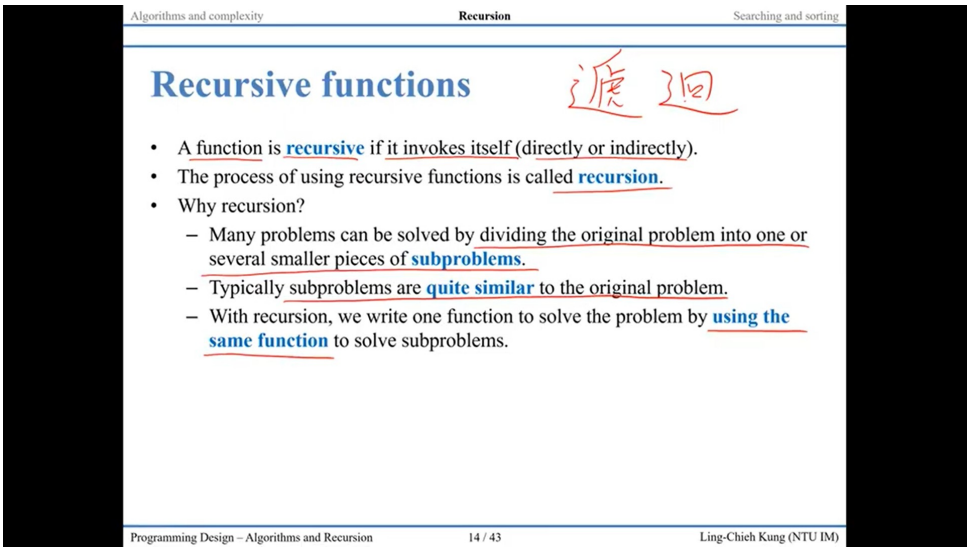

Four STL Components
Containers: used to manage collections of objects of a certain kind.
Algorithms: act on containers.
E.g. initialization, sorting, searching….
Iterators: used to step through the elements of collections of objects.
Functors are objects that can be treated as though they are a function or function pointer.
2018/9/8
© Ren-Song Tsay, NTHU, Taiwan

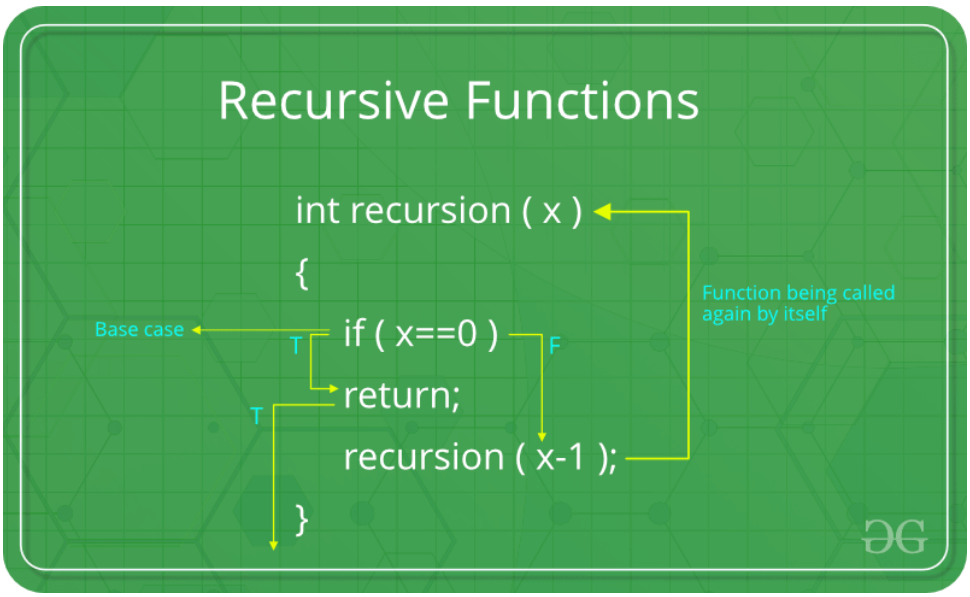
Vector

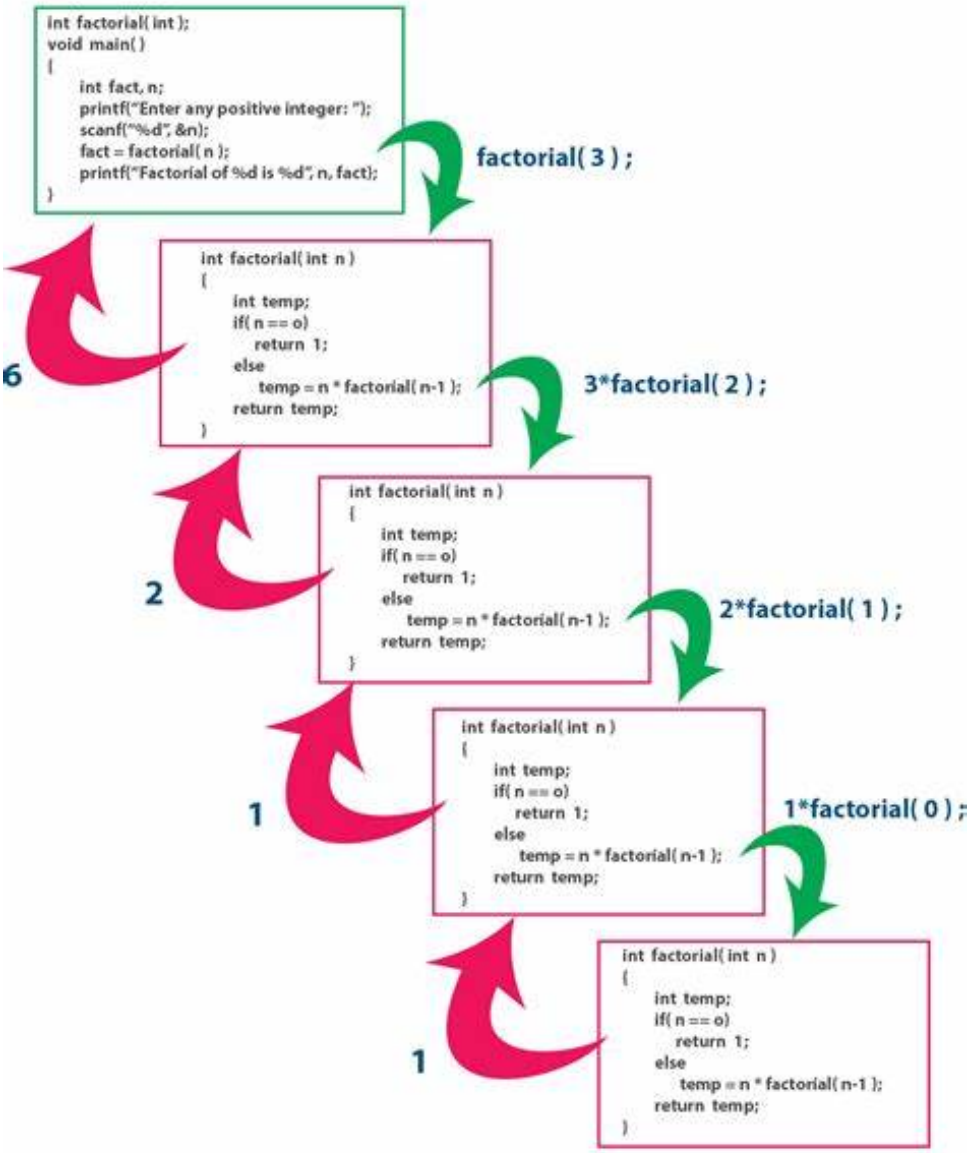

Templates
The standard library <vector>
Exception handling
The standard library <vector>
A vector is very easy to use.
To create a vector, indicate the type of items:
vector<int> v1; // integer vector
vector<double> v2; // double vector
vector<Warrior> v3; // Warrior vector
Member functions that modifies a vector:
push_back(), pop_back(), insert(), erase(), swap(), =, etc.
Member functions for one to access a vector element:
[], front(), back(), etc.
Member functions related to the capacity:
size(), max_size(), resize(), etc.
Programming Design – Templates, Vectors, and Exceptions 28 / 55 Ling-Chieh Kung (NTU IM)

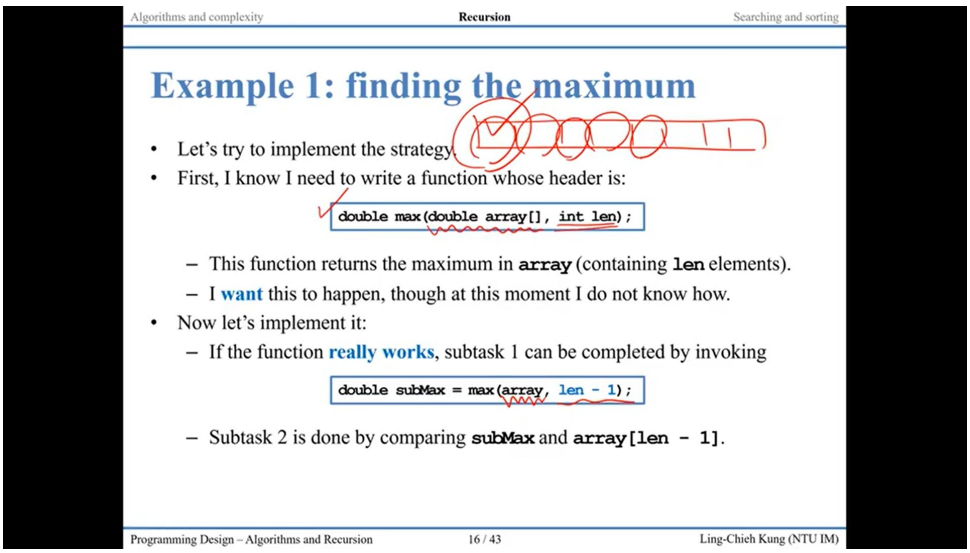


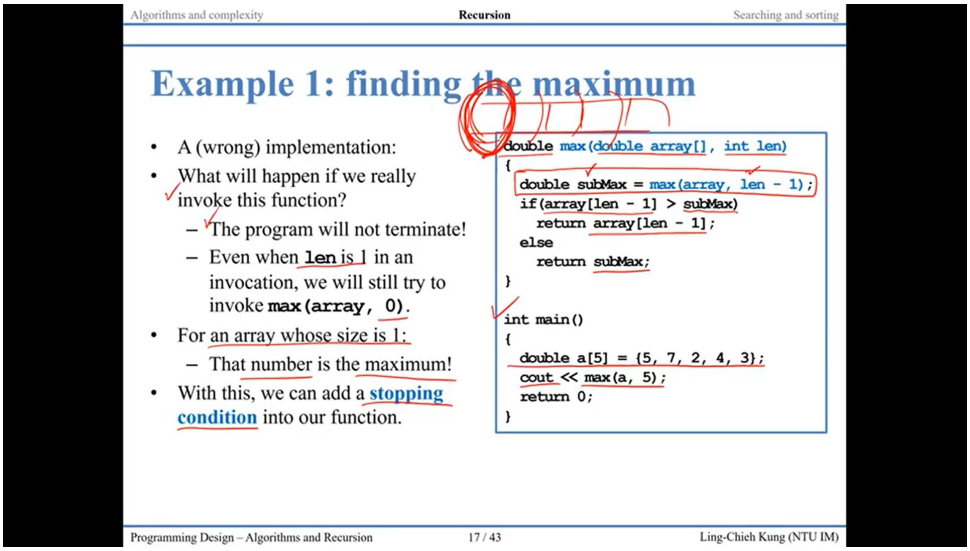
Recursive Function

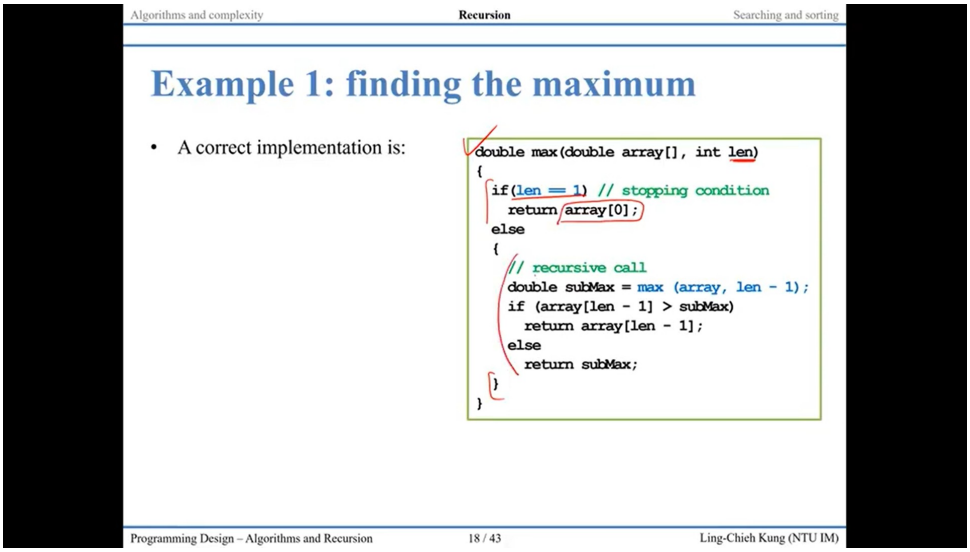




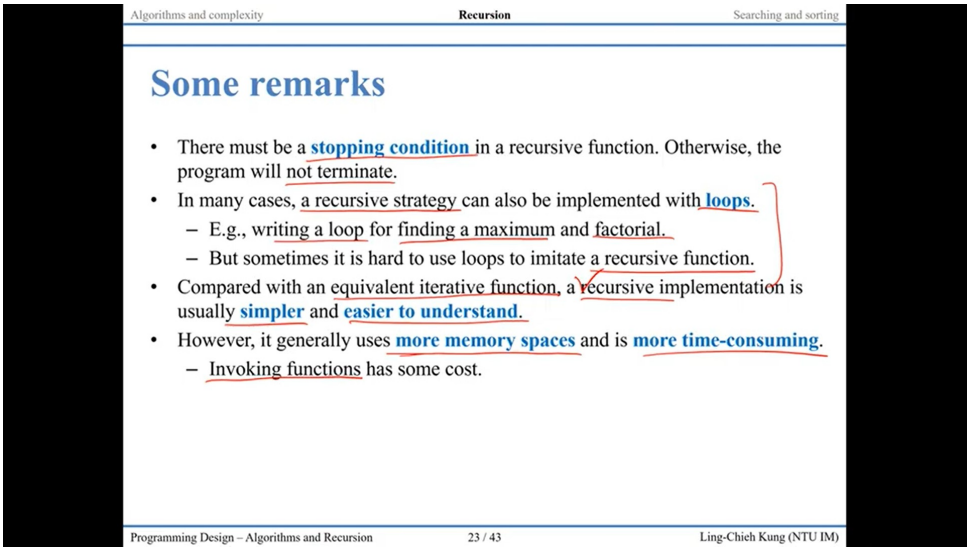

int factorial(int);
void main( )
{
int fact, n;
printf("Enter any positive integer: ");
scanf("%d", &n);
fact = factorial(n);
printf("Factorial of %d is %d", n, fact);
}
factorial(3);
int factorial(int n)
{
int temp;
if(n == 0)
return 1;
else
temp = n * factorial(n-1);
return temp;
}
6
3*factorial(2);
2
2*factorial(1);
1
1*factorial(0);
1

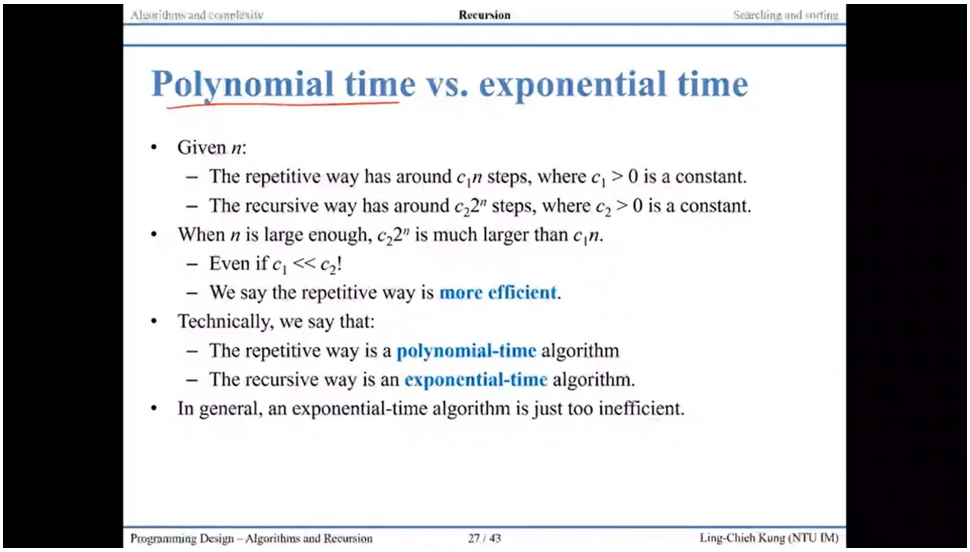

Algorithms and complexity
Recursion
Searching and sorting
Example 1: finding the maximum
Let's try to implement the strategy.
First, I know I need to write a function whose header is:
double max(double array[], int len);
This function returns the maximum in array (containing len elements).
I want this to happen, though at this moment I do not know how.
Now let's implement it:
If the function really works, subtask 1 can be completed by invoking
double subMax = max(array, len - 1);
Subtask 2 is done by comparing subMax and array[len - 1].
Programming Design – Algorithms and Recursion
16 / 43
Ling-Chieh Kung (NTU IM)





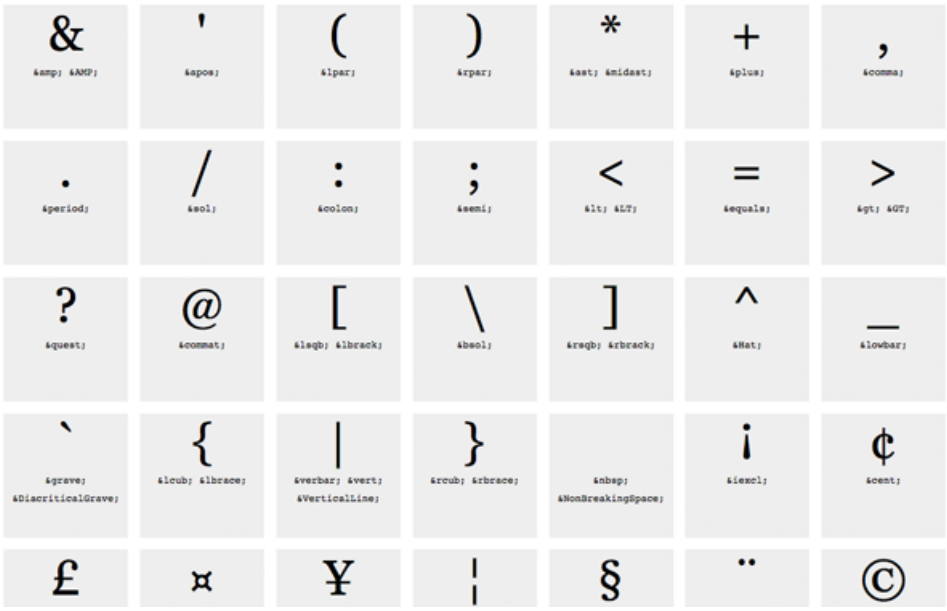
In [ ]:
```
#include <iostream>
using namespace std;

double findMax(double array[], int len);

int main(){
    double a[5] = {5, 7, 2, 4, 3};
    cout << "The max item is " << findMax(a, 5) << endl;

    return 0;
}

double findMax(double array[], int len){
 if (len == 1){
        return array[0];
 }
 else {
    double subMax = findMax(array, len-1);
    if (array[len-1] > subMax){
        return array[len -1];
    }
    else {
        return subMax;
    }
 }
}
```





Programming Language - HTML

Special HTML character entities



Latex Language

LaTeX Mathematical Symbols

The more unusual symbols are not defined in base LaTeX (NFSS) and require `\usepackage{amssymb}`

1 Greek and Hebrew letters

α	\alpha	κ	\kappa	ψ	\psi	F	\digamma	Δ	\Delta	Θ	\Theta
β	\beta	λ	\lambda	ρ	\rho	ε	\varepsilon	Γ	\Gamma	Υ	\Upsilon
χ	\chi	μ	\mu	σ	\sigma	$\varkappa$	\varkappa	Λ	\Lambda	Ξ	\Xi
δ	\delta	ν	\nu	τ	\tau	φ	\varphi	Ω	\Omega		
ϵ	\epsilon	o	o	θ	\theta	ϖ	\varpi	Φ	\Phi	$\aleph$	\aleph
η	\eta	ω	\omega	υ	\upsilon	ϱ	\varrho	Π	\Pi	$\beth$	\beth
γ	\gamma	ϕ	\phi	ξ	\xi	ς	\varsigma	Ψ	\Psi	$\daleth$	\daleth
ι	\iota	π	\pi	ζ	\zeta	ϑ	\vartheta	Σ	\Sigma	$\gimel$	\gimel

2 LaTeX math constructs

$\frac{abc}{xyz}$	\frac{abc}{xyz}	$\overline{abc}$	\overline{abc}	$\overrightarrow{abc}$	\overrightarrow{abc}
f'	f'	$\underline{abc}$	\underline{abc}	$\overleftarrow{abc}$	\overleftarrow{abc}
$\sqrt{abc}$	\sqrt{abc}	$\widehat{abc}$	\widehat{abc}	$\overbrace{abc}$	\overbrace{abc}
$\sqrt[n]{abc}$	\sqrt[n]{abc}	$\widetilde{abc}$	\widetilde{abc}	$\underbrace{abc}$	\underbrace{abc}

3 Delimiters

$\vert$	\vert	$\{$	\{	$\lfloor$	\lfloor	$/$	/	$\Uparrow$	\Uparrow	$\llcorner$	\llcorner
$\vert$	\vert	$\}$	\}	$\rfloor$	\rfloor	$\backslash$	\backslash	$\uparrow$	\uparrow	$\lrcorner$	\lrcorner
$\Vert$	\Vert	$\langle$	\langle	$\lceil$	\lceil	$[$	[	$\Downarrow$	\Downarrow	$\ulcorner$	\ulcorner
$\Vert$	\Vert	$\rangle$	\rangle	$\rceil$	\rceil	$]$	]	$\downarrow$	\downarrow	$\urcorner$	\urcorner

Use the pair `\left`s_1 and `\right`s_2 to match height of delimiters s_1 and s_2 to the height of their contents, e.g.,
`\left|` *expr* `\right|` `\left\{` *expr* `\right\}` `\left\Vert` *expr* `\right.`

4 Variable-sized symbols (displayed formulae show larger version)

$\sum$	\sum	$\int$	\int	$\biguplus$	\biguplus	$\bigoplus$	\bigoplus	$\bigvee$	\bigvee
$\prod$	\prod	$\oint$	\oint	$\bigcap$	\bigcap	$\bigotimes$	\bigotimes	$\bigwedge$	\bigwedge
$\coprod$	\coprod	$\iint$	\iint	$\bigcup$	\bigcup	$\bigodot$	\bigodot	$\bigsqcup$	\bigsqcup

5 Standard Function Names

Function names should appear in Roman, not Italic, e.g.,

Correct: `\tan(at-n\pi)` $\longrightarrow \tan(at - n\pi)$
Incorrect: `tan(at-n\pi)` $\longrightarrow tan(at - n\pi)$

arccos	\arccos	arcsin	\arcsin	arctan	\arctan	arg	\arg
cos	\cos	cosh	\cosh	cot	\cot	coth	\coth
csc	\csc	deg	\deg	det	\det	dim	\dim
exp	\exp	gcd	\gcd	hom	\hom	inf	\inf
ker	\ker	lg	\lg	lim	\lim	lim inf	\liminf
lim sup	\limsup	ln	\ln	log	\log	max	\max
min	\min	Pr	\Pr	sec	\sec	sin	\sin
sinh	\sinh	sup	\sup	tan	\tan	tanh	\tanh

Programming Language - JavaScript

JavaScript Reserved Words

Keywords or Reserved Words

var	delete	for	let	break
super	void	case	do	static
function	new	switch	while	interface
catch	else	if	package	finally
this	with	class	enum	default
implements	private	throw	yield	typeof
const	export	import	protected	return
true	continue	extends	in	instanceof
public	try	debugger	false	

www.geekyshows.com

bitwise operation Javascript

Bitwise operators in JavaScript

Syntax	Name	Type
&	Bitwise AND	binary
|	Bitwise OR	binary
^	Bitwise XOR (exclusive OR)	binary
~	Bitwise NOT	unary
<<	Left shift	binary
>>	Right shift	binary
>>>	Zero-fill right shift	binary

Data Structure in C++

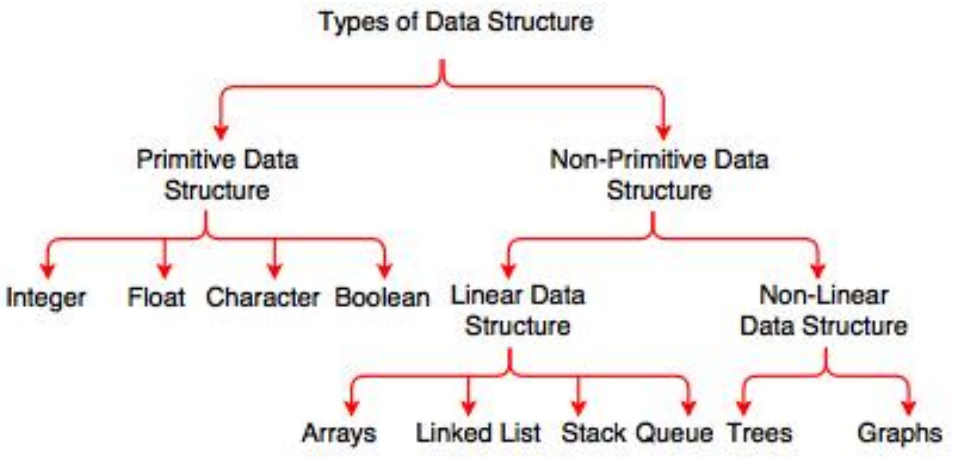


Fig. Types of Data Structure

C++ Performance Evaluation - Space Complexity and Time Complexity

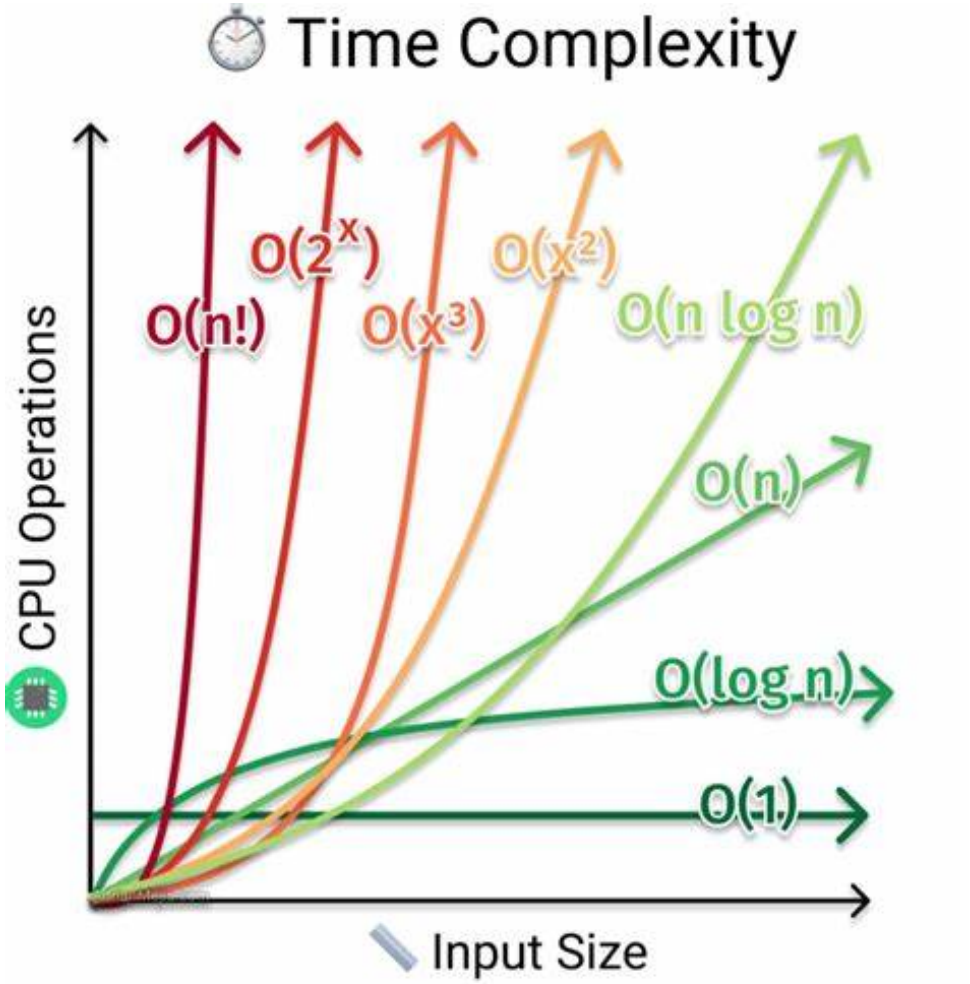
- We want to define time taken by an algorithm without depending on the implementation details.
- because
- A given algorithm will take different amounts of time on the same inputs depending on such factors as:
  - Processor speed;
  - Instruction set,
  - Disk speed,
  - Brand of compiler and etc.
- The way around is to estimate efficiency of each algorithm asymptotically.
  - **Time Complexity:** Running time of the program as a function of the size of input
  - **Space Complexity:** Amount of computer memory required during the program execution, as a function of the input size

Mr. Nitin M Shivale (Asst. Prof JSPM'S BSIOTR)

Space Complexity

- When memory was expensive we focused on making programs as space efficient as possible and developed schemes to make memory appear larger than it really was (virtual memory and memory paging schemes)
- Space complexity is still important in the field of embedded computing (hand held computer based equipment like cell phones, palm devices, etc)

3



Container	Insertion	Access	Erase	Find	Persistent Iterators
vector / string	Back: O(1) or O(n) Other: O(n)	O(1)	Back: O(1) Other: O(n)	Sorted: O(log n) Other: O(n)	No
deque	Back/Front: O(1) Other: O(n)	O(1)	Back/Front: O(1) Other: O(n)	Sorted: O(log n) Other: O(n)	Pointers only
list / forward_list	Back/Front: O(1) With iterator: O(1) Index: O(n)	Back/Front: O(1) With iterator: O(1) Index: O(n)	Back/Front: O(1) With iterator: O(1) Index: O(n)	O(n)	Yes
set / map	O(log n)	-	O(log n)	O(log n)	Yes
unordered_set / unordered_map	O(1) or O(n)	O(1) or O(n)	O(1) or O(n)	O(1) or O(n)	Pointers only
priority_queue	O(log n)	O(1)	O(log n)	-	-

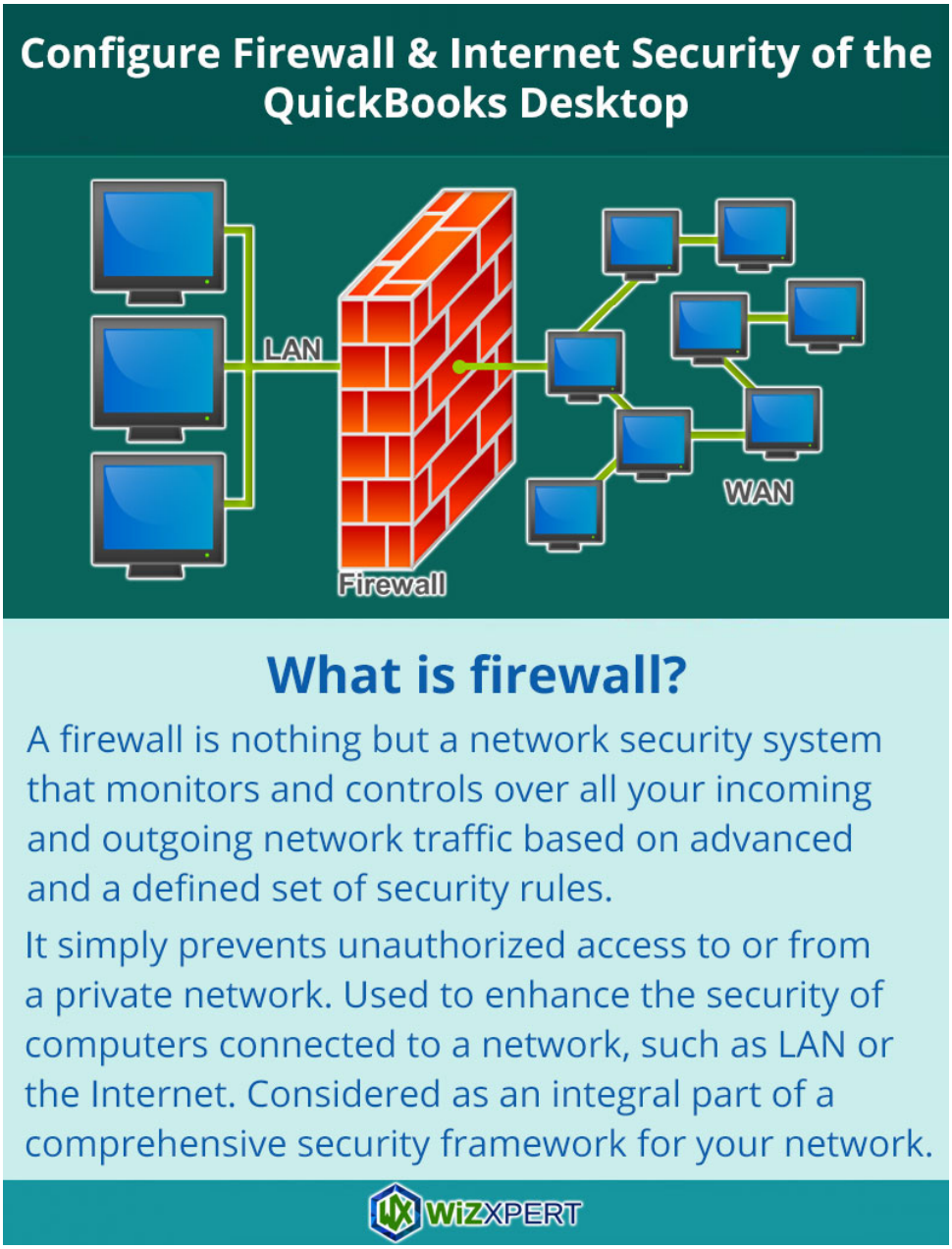
Data Structures

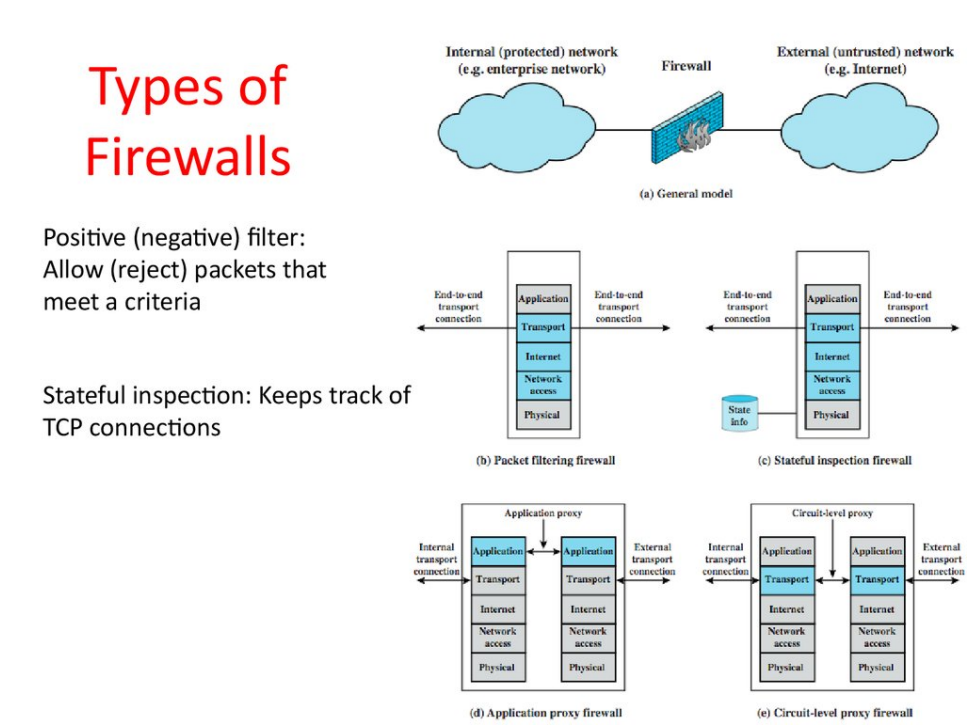
Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Indexing	Search	Insertion	Deletion	Indexing	Search	Insertion	Deletion	
Basic Array	O(1)	O(n)	-	-	O(1)	O(n)	-	-	O(n)
Dynamic Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	-	O(1)	O(1)	O(1)	-	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	-	O(log(n))	O(log(n))	O(log(n))	-	O(n)	O(n)	O(n)	O(n)
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	-	O(log(n))	O(log(n))	O(log(n))	-	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)

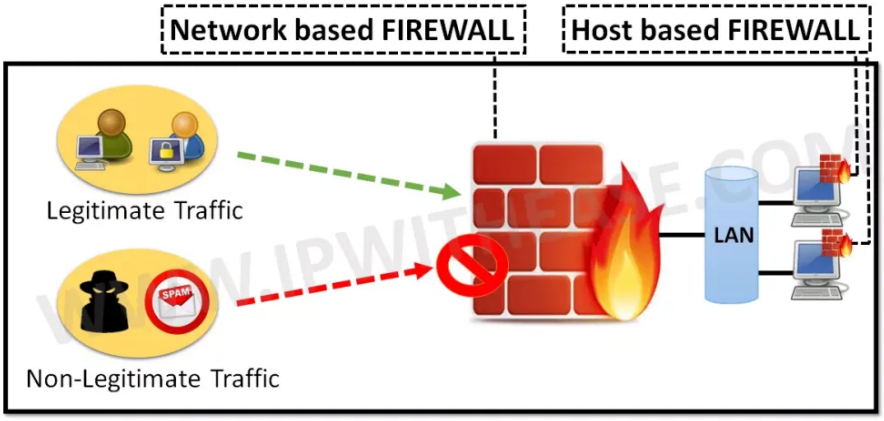
Sorting Algorithm	Time Complexity			Space Complexity
	Best Case	Average Case	Worst Case	Worst Case
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Merge Sort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)

Network

Firewall


Configure Firewall & Internet Security of the QuickBooks Desktop
LAN
Firewall
WAN
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.
WiZXPERT

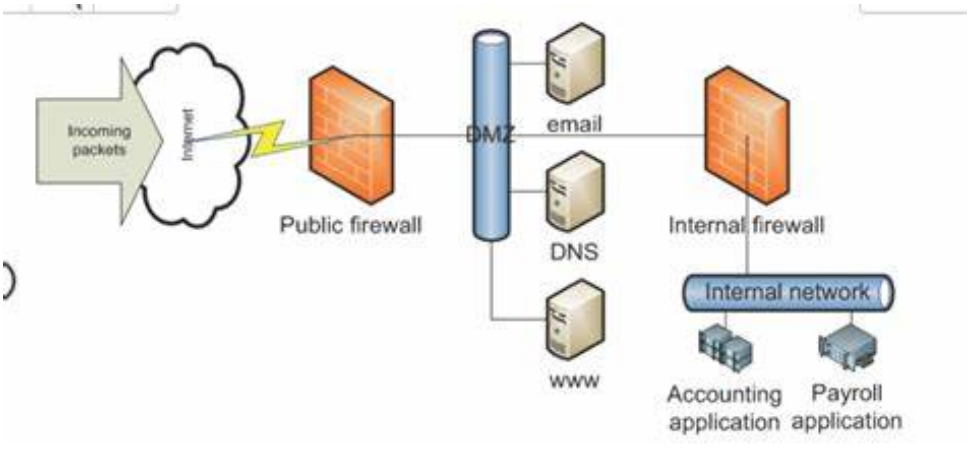

Types of Firewalls
Positive (negative) filter: Allow (reject) packets that meet a criteria
Stateful inspection: Keeps track of TCP connections
Internal (protected) network (e.g. enterprise network)
Firewall
External (untrusted) network (e.g. Internet)
(a) General model
End-to-end transport connection
Application
Transport
Internet
Network access
Physical
State info
(b) Packet filtering firewall
(c) Stateful inspection firewall
Application proxy
Internal transport connection
External transport connection
(d) Application proxy firewall
Circuit-level proxy
(e) Circuit-level proxy firewall

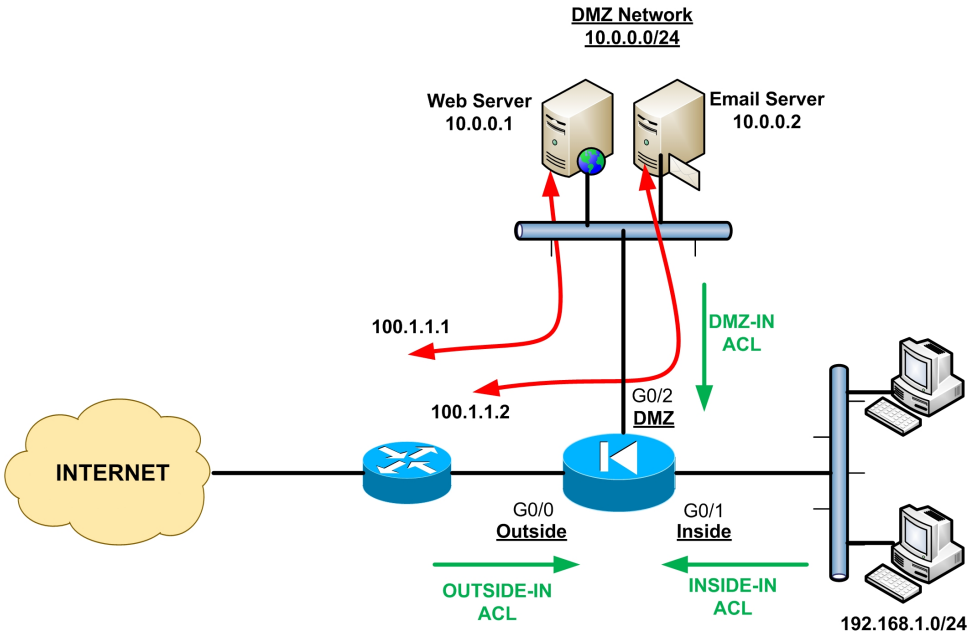


ACL (Access Control List)

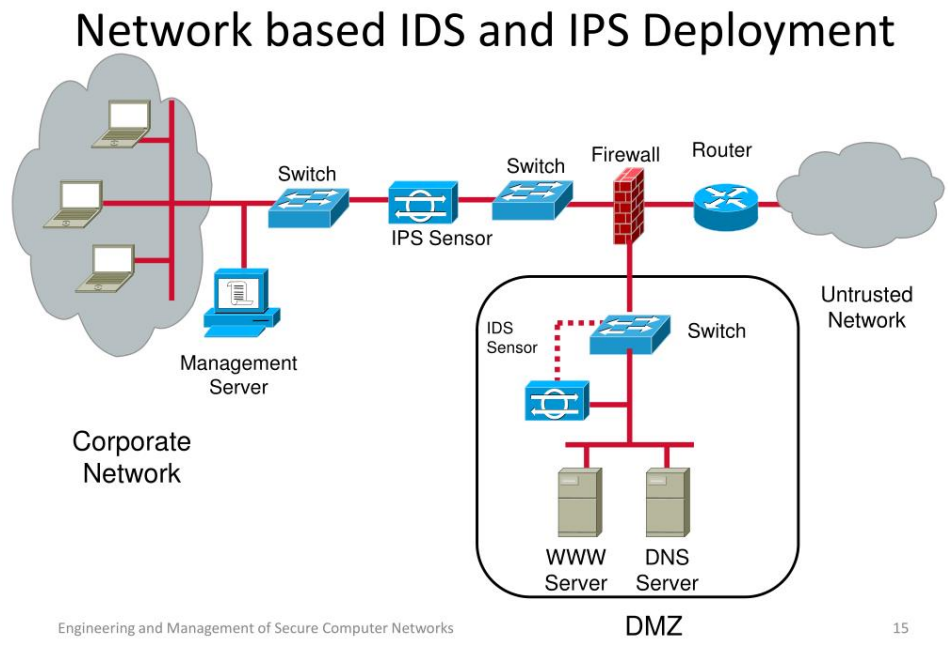
Parameter	ACL	Firewall
Asset Type	Feature on Layer 3 devices and Firewalls	Hardware or Software
Stateful/Stateless inspection	Performs stateless inspection	Performs Stateful inspection
Scope wrt OSI	Upto Layer 4	Upto Layer 7
Security	Low	High
Intrusion detection	Not possible	Possible
Target deployment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

DMZ



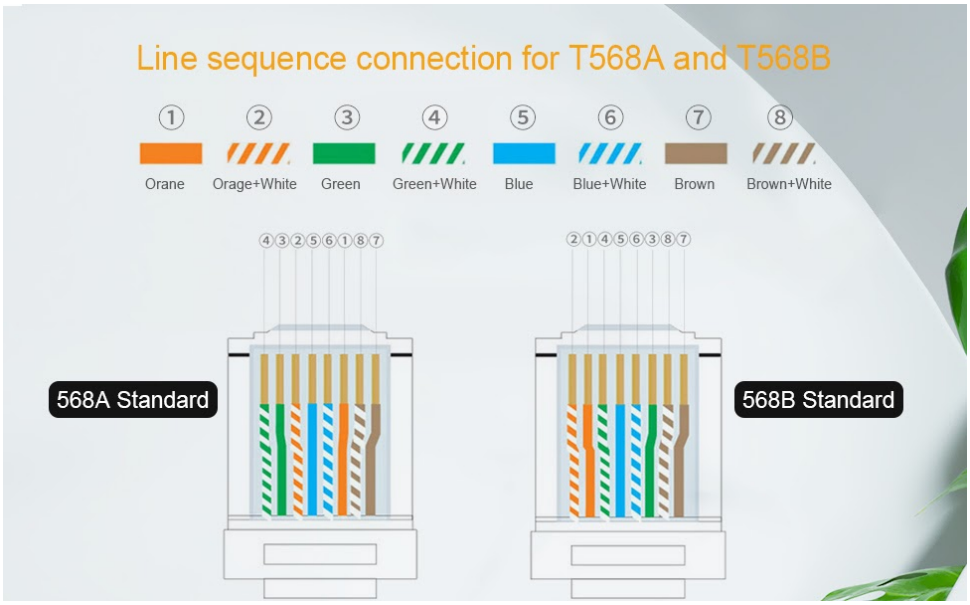


IPS/IDS

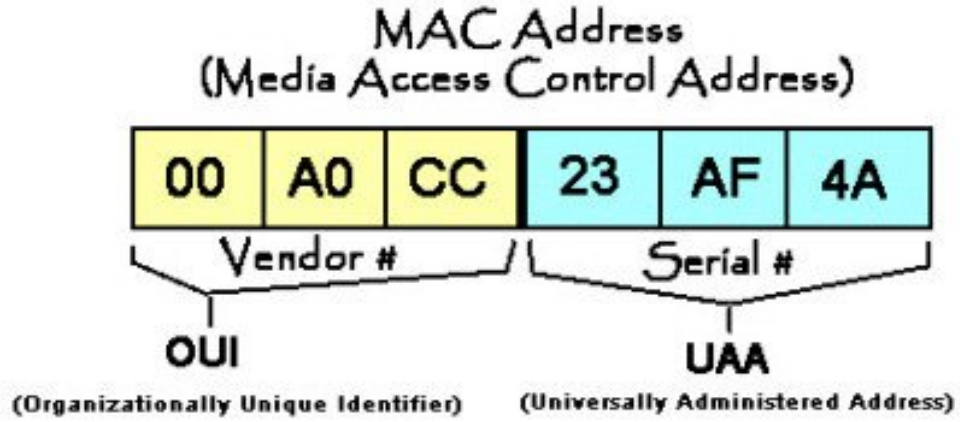


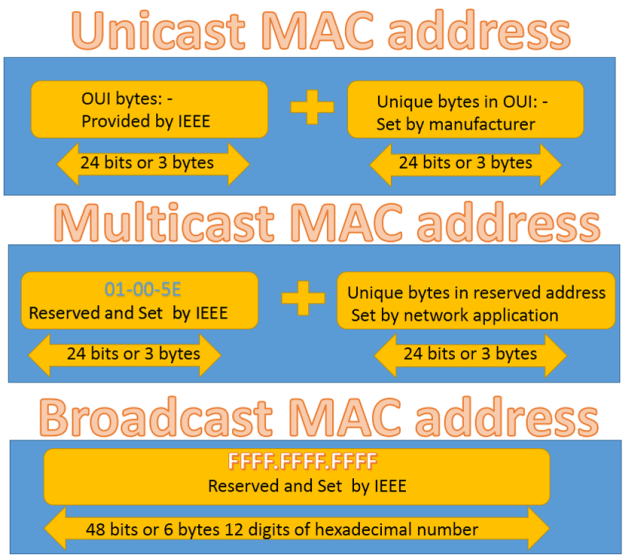
Cable

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



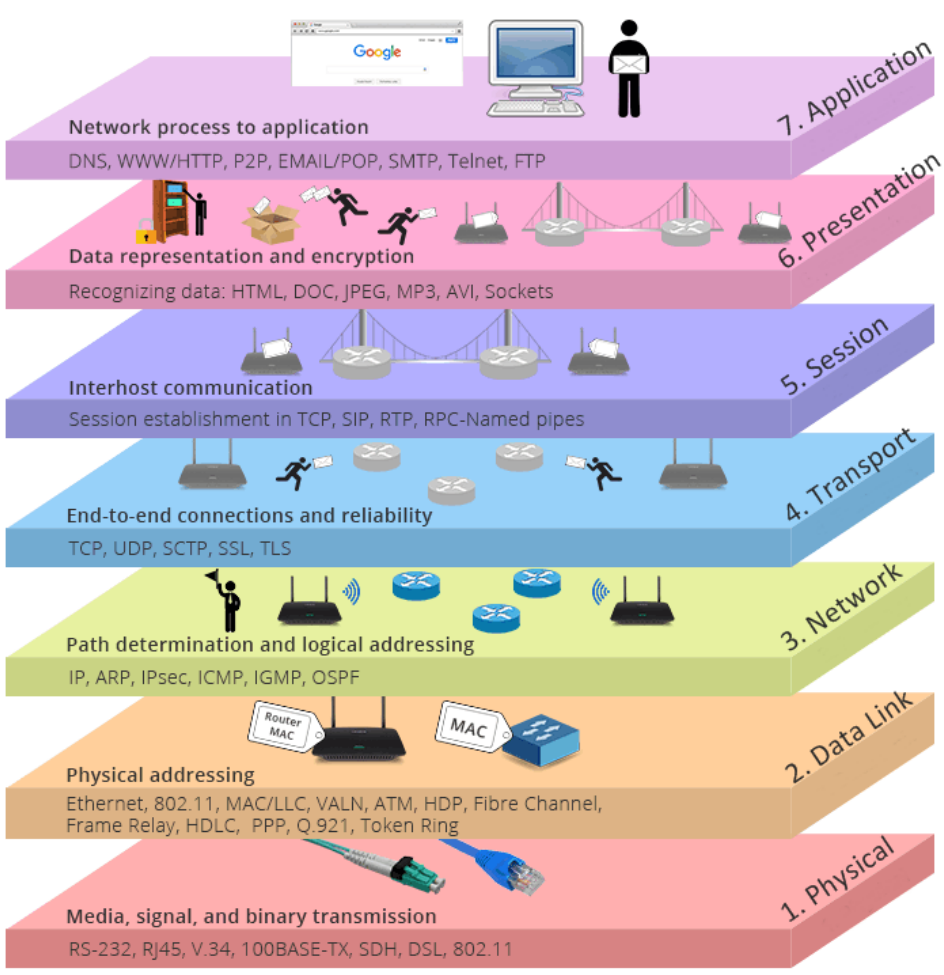
MAC Address

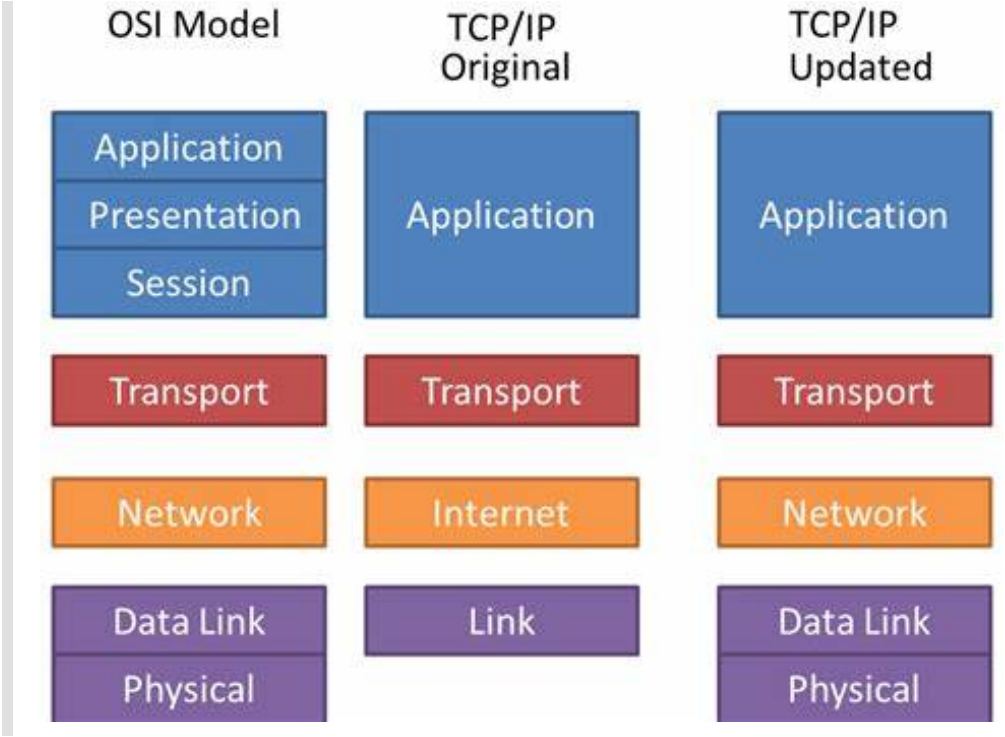


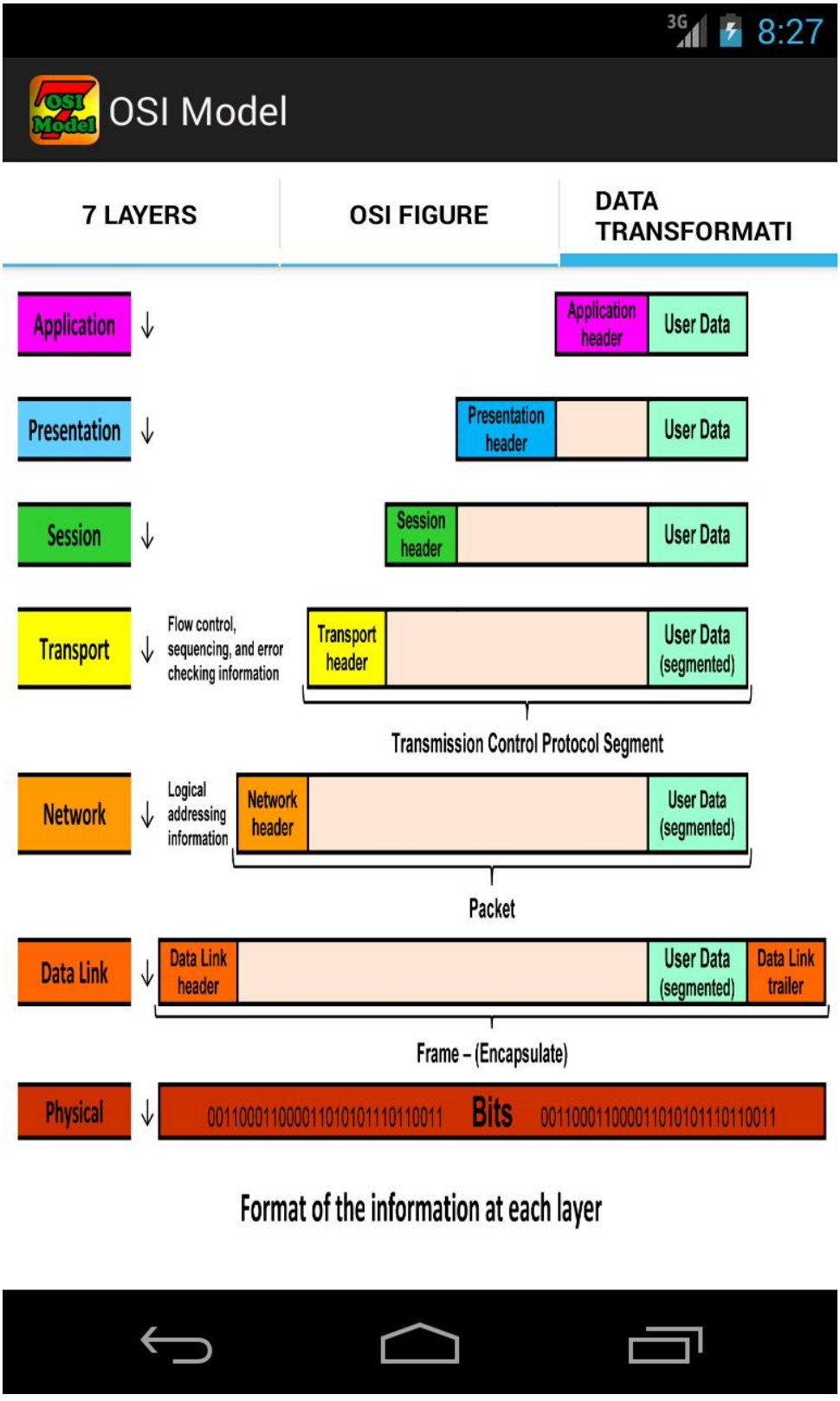


OSI Model

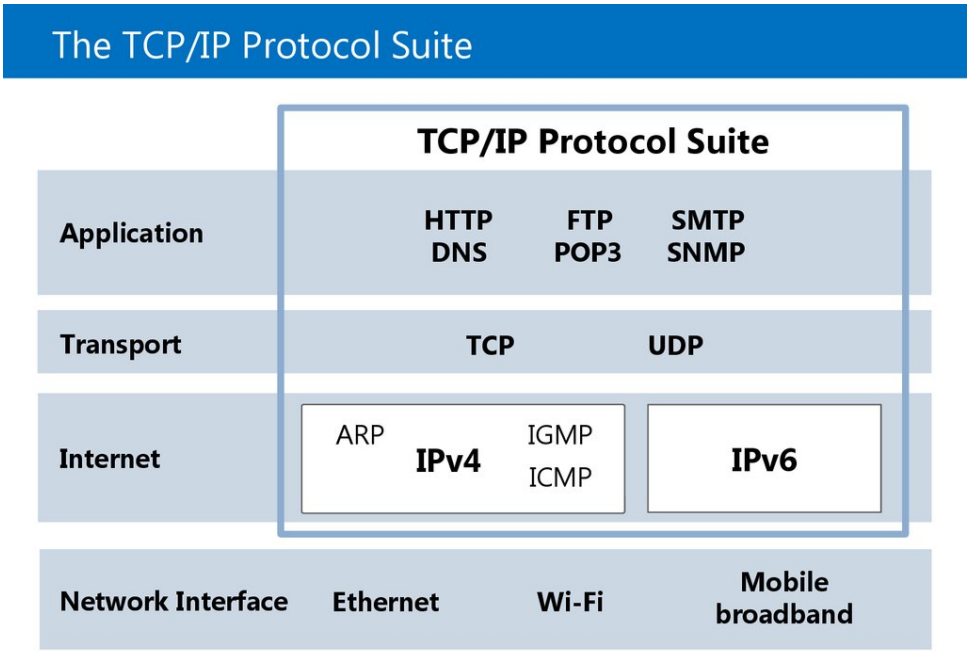
UPPER LAYERS	7	**Application Layer** ✓ Message format, Human-Machine Interfaces
	6	**Presentation Layer** ✓ Coding into 1s and 0s; encryption, compression
	5	**Session Layer** ✓ Authentication, permissions, session restoration
TRANSPORT SERVICE	4	**Transport Layer** ✓ End-to-end error control
	3	**Network Layer** ✓ Network addressing; routing or switching
	2	**Data Link Layer** ✓ Error detection, flow control on physical link
	1	**Physical Layer** ✓ Bit stream: physical medium, method of representing bits

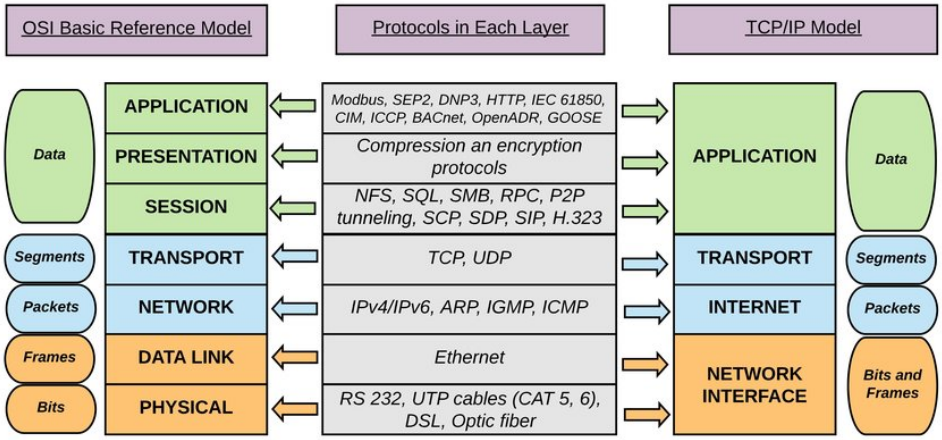


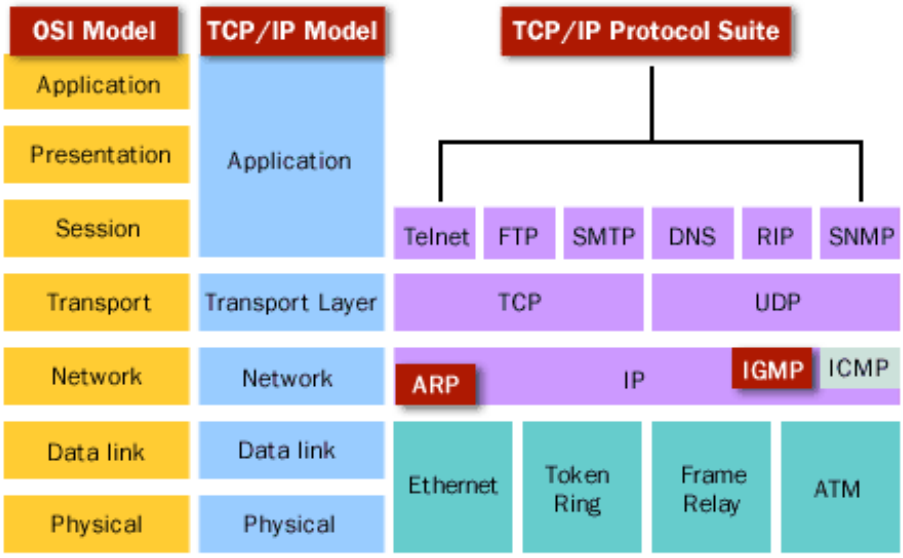

OSI Model
TCP/IP Original
TCP/IP Updated
Application
Presentation
Session
Application
Application
Transport
Transport
Transport
Network
Internet
Network
Data Link
Physical
Link
Data Link
Physical

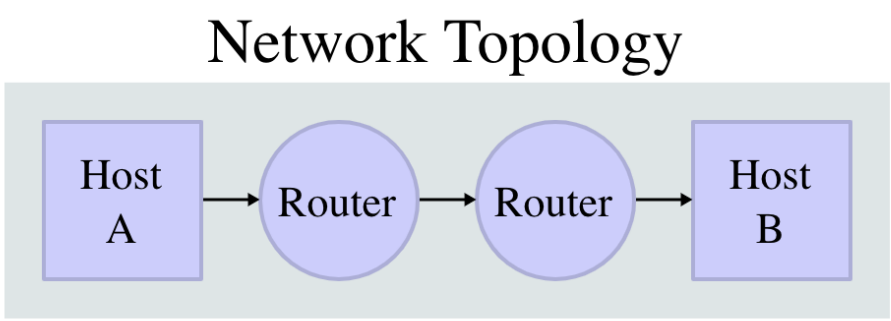

8:27
OSI Model
7 LAYERS
OSI FIGURE
DATA TRANSFORMATI
Application
Application header
User Data
Presentation
Presentation header
User Data
Session
Session header
User Data
Transport
Flow control, sequencing, and error checking information
Transport header
User Data (segmented)
Transmission Control Protocol Segment
Network
Logical addressing information
Network header
User Data (segmented)
Packet
Data Link
Data Link header
User Data (segmented)
Data Link trailer
Frame – (Encapsulate)
Physical
00110001100001101010111010011 Bits 00110001100001101010111010011
Format of the information at each layer

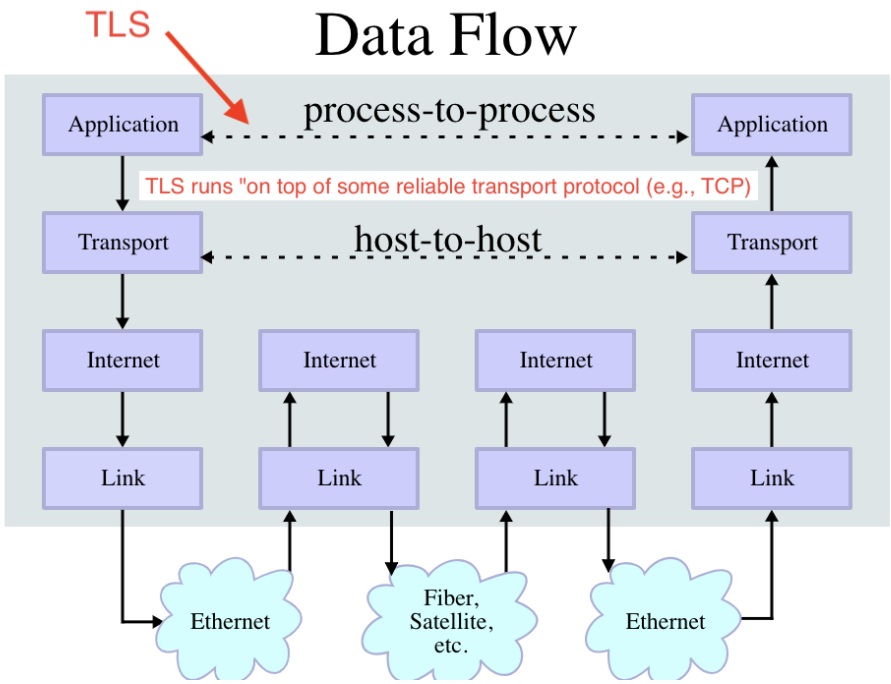
Protocol


The TCP/IP Protocol Suite
TCP/IP Protocol Suite
Application
HTTP
DNS
FTP
POP3
SMTP
SNMP
Transport
TCP
UDP
Internet
ARP
IPv4
IGMP
ICMP
IPv6
Network Interface
Ethernet
Wi-Fi
Mobile broadband

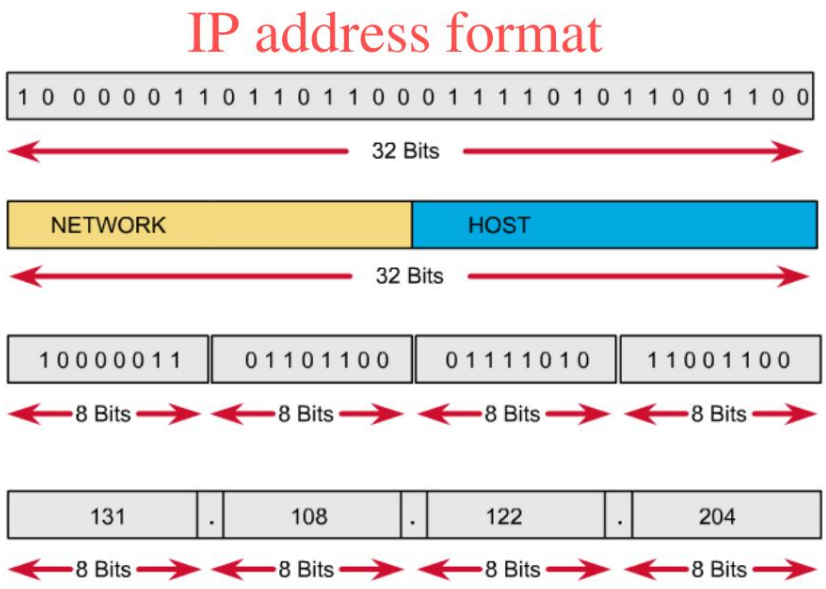

OSI Basic Reference Model
Protocols in Each Layer
TCP/IP Model
Data
Segments
Packets
Frames
Bits
APPLICATION
PRESENTATION
SESSION
TRANSPORT
NETWORK
DATA LINK
PHYSICAL
Modbus, SEP2, DNP3, HTTP, IEC 61850, CIM, ICCP, BACnet, OpenADR, GOOSE
Compression an encryption protocols
NFS, SQL, SMB, RPC, P2P tunneling, SCP, SDP, SIP, H.323
TCP, UDP
IPv4/IPv6, ARP, IGMP, ICMP
Ethernet
RS 232, UTP cables (CAT 5, 6), DSL, Optic fiber
APPLICATION
TRANSPORT
INTERNET
NETWORK INTERFACE
Data
Segments
Packets
Bits and Frames


OSI Model
TCP/IP Model
TCP/IP Protocol Suite
Application
Presentation
Session
Transport
Network
Data link
Physical
Application
Transport Layer
Network
Data link
Physical
Telnet
FTP
SMTP
DNS
RIP
SNMP
TCP
UDP
ARP
IP
IGMP
ICMP
Ethernet
Token Ring
Frame Relay
ATM


Network Topology
Host A
Router
Router
Host B


TLS
Data Flow
Application
process-to-process
Application
TLS runs "on top of some reliable transport protocol (e.g., TCP)
Transport
host-to-host
Transport
Internet
Internet
Internet
Internet
Link
Link
Link
Link
Ethernet
Fiber, Satellite, etc.
Ethernet

IP Address

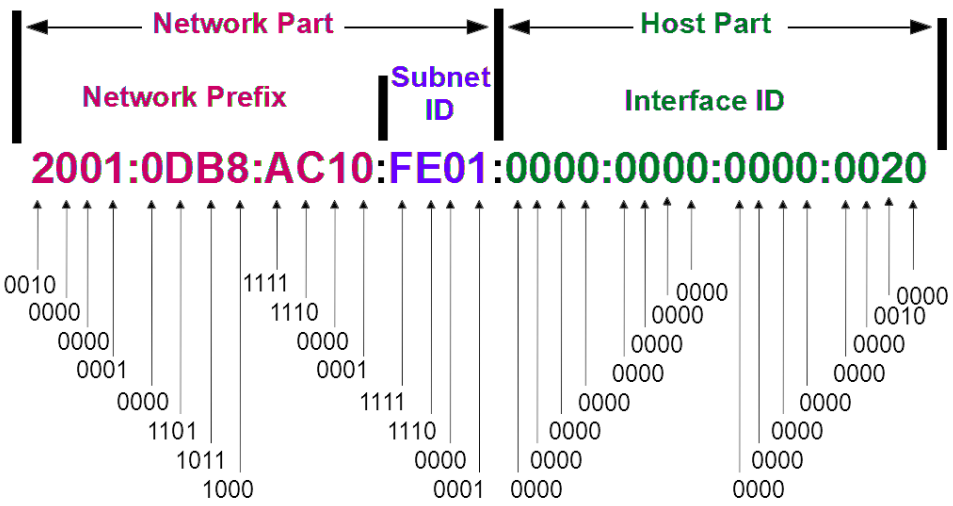

IP address format
1 0 0 0 0 0 1 1 0 1 1 0 1 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 1 0 0
32 Bits
NETWORK
HOST
32 Bits
10000011
01101100
01111010
11001100
8 Bits
8 Bits
8 Bits
8 Bits
131
.
108
.
122
.
204
8 Bits
8 Bits
8 Bits
8 Bits

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

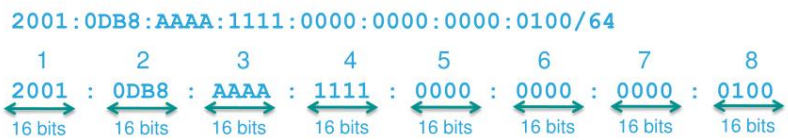
This is the usual breakdown but it can be broken down in other ways



IPv6 Address Notation

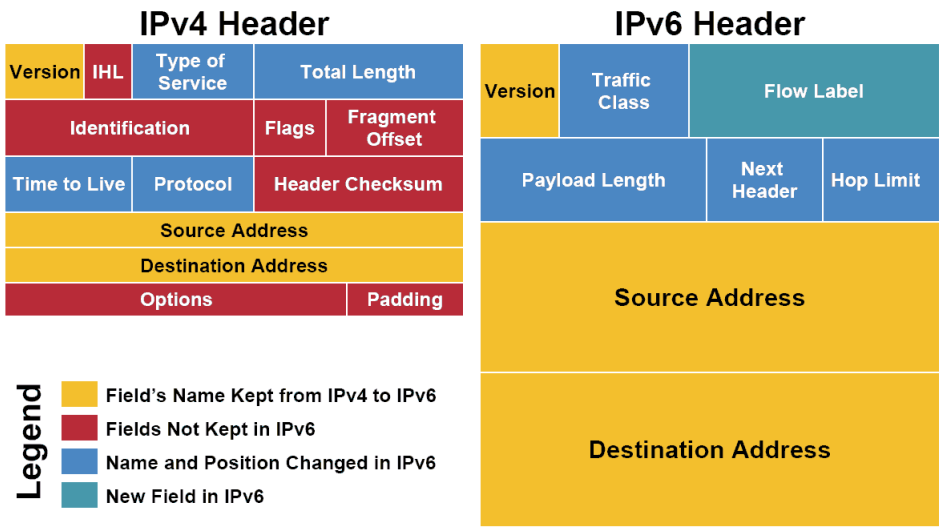
One Hex digit = 4 bits

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

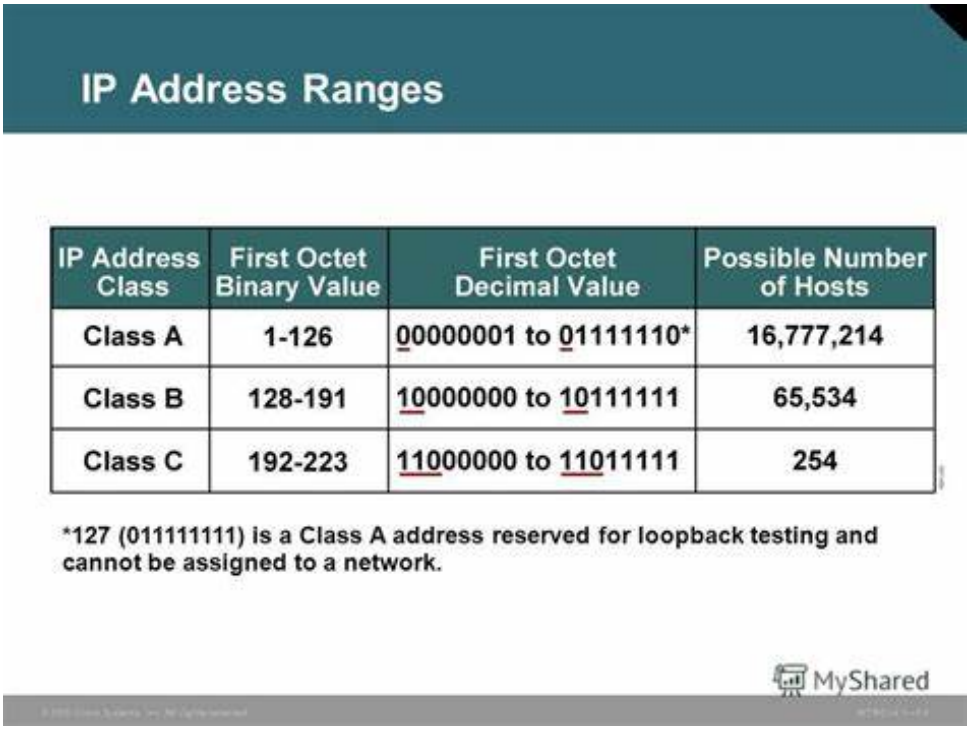


- IPv6 addresses are 128-bit addresses represented in:
  - Eight 16-bit segments or “hextets” (not a formal term)
  - Hexadecimal (non-case sensitive) between 0000 and FFFF

© 2013 Cisco and/or its affiliates. All rights reserved. Cisco Public 5


IPv4 Header
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Padding
IPv6 Header
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address
Legend
Field's Name Kept from IPv4 to IPv6
Fields Not Kept in IPv6
Name and Position Changed in IPv6
New Field in IPv6

IP Range

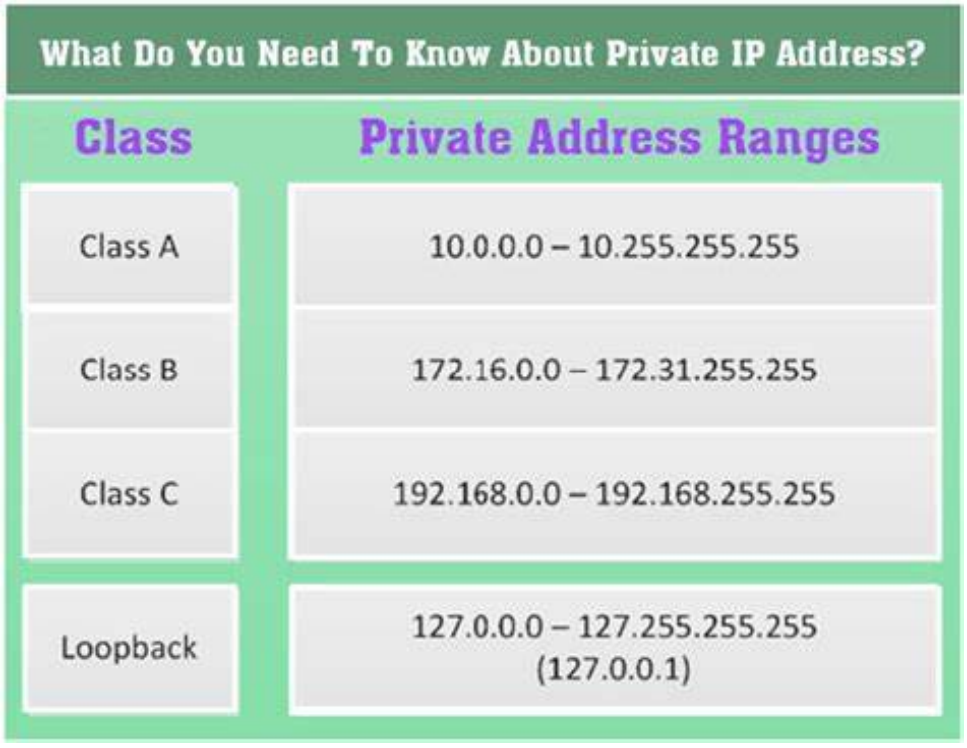

IP Address Ranges
IP Address Class
First Octet Binary Value
First Octet Decimal Value
Possible Number of Hosts
Class A
1-126
00000001 to 01111110*
16,777,214
Class B
128-191
10000000 to 10111111
65,534
Class C
192-223
11000000 to 11011111
254
*127 (011111111) is a Class A address reserved for loopback testing and cannot be assigned to a network.
MyShared

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

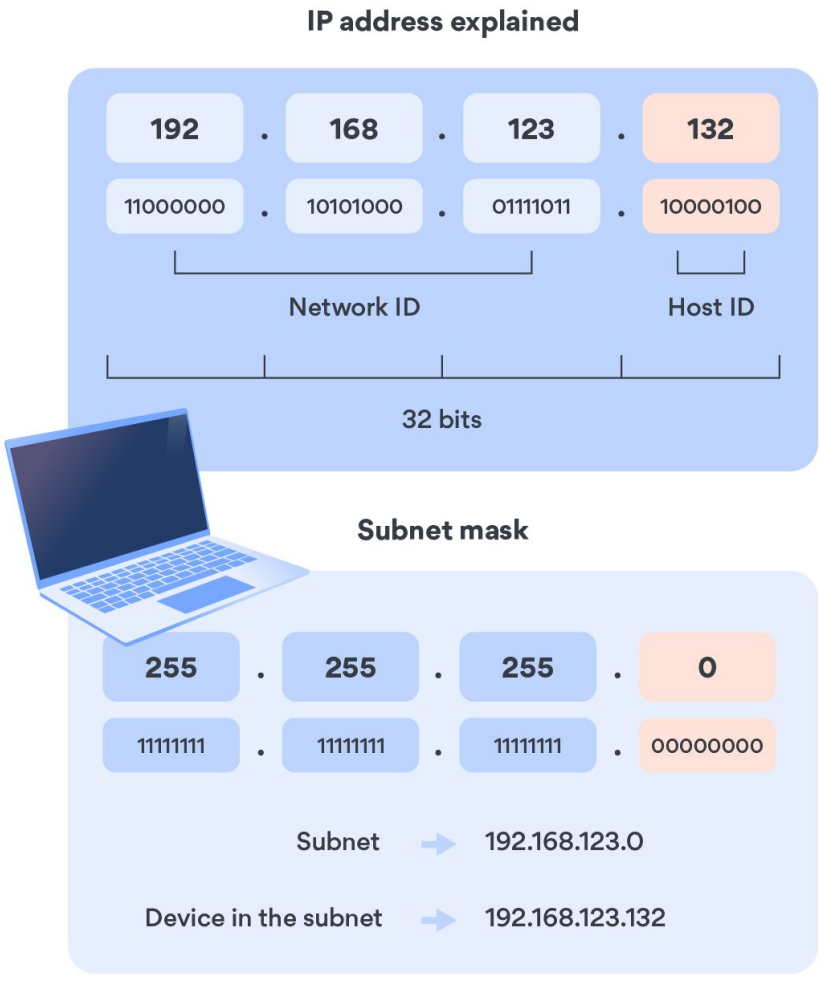
- example:
  - 192.168.1.0/24 (mask: 255.255.255.0 | 256 hosts) - 256 networks
  - 172.17.0.0/16 (mask: 255.255.0.0 | 65.536 hosts) 256 networks

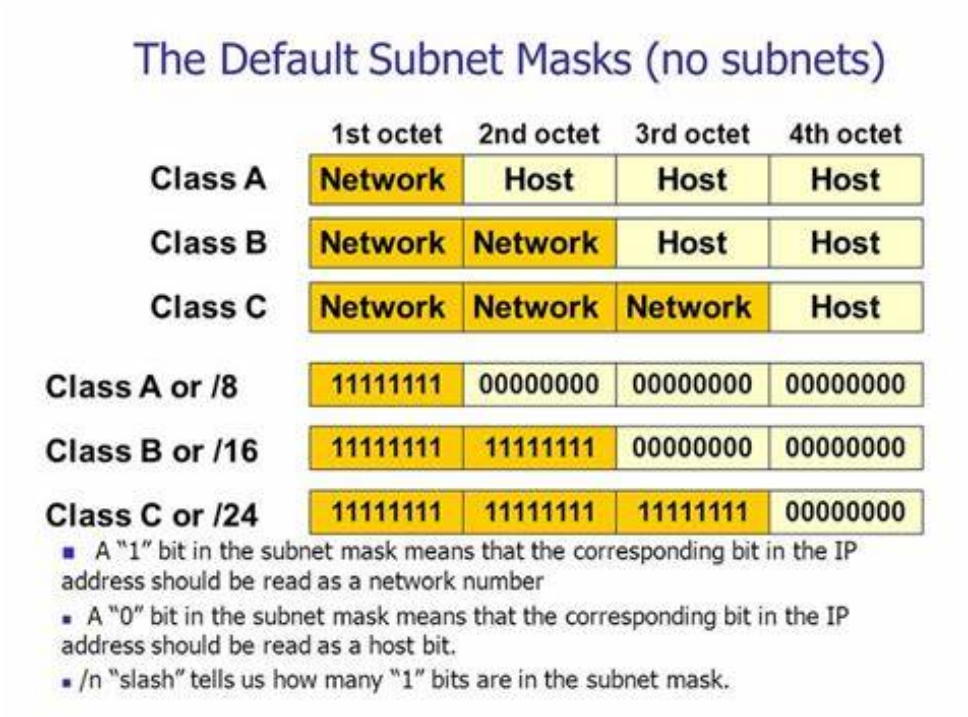


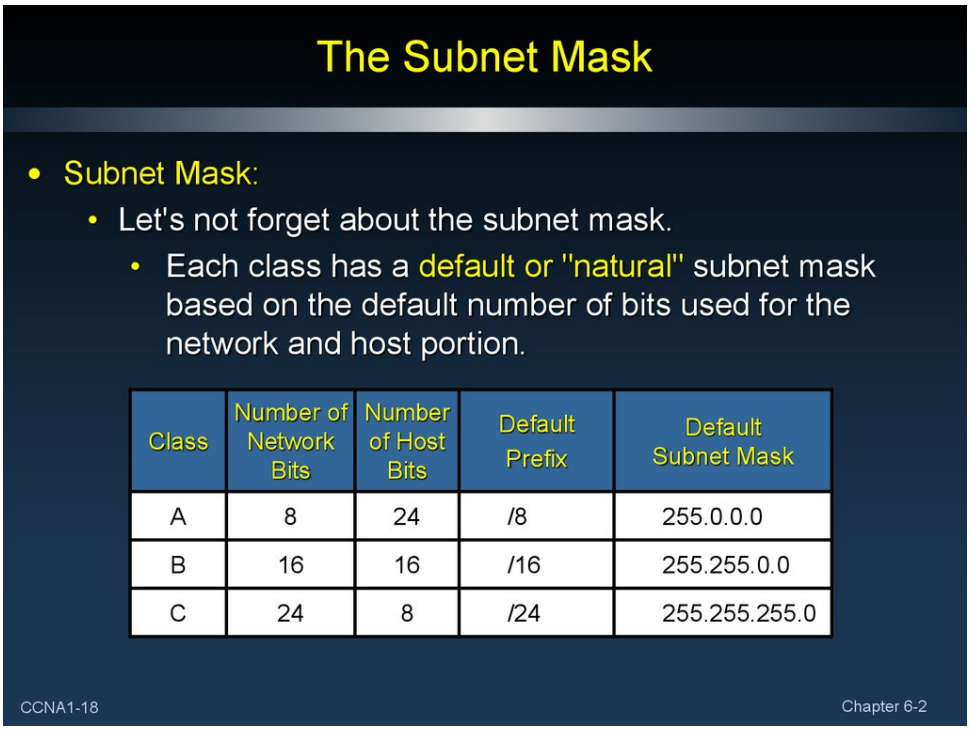
Subnet Mask

Subnet Mask

Suffix	Hosts	32-Borrowed=CIDR	2^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



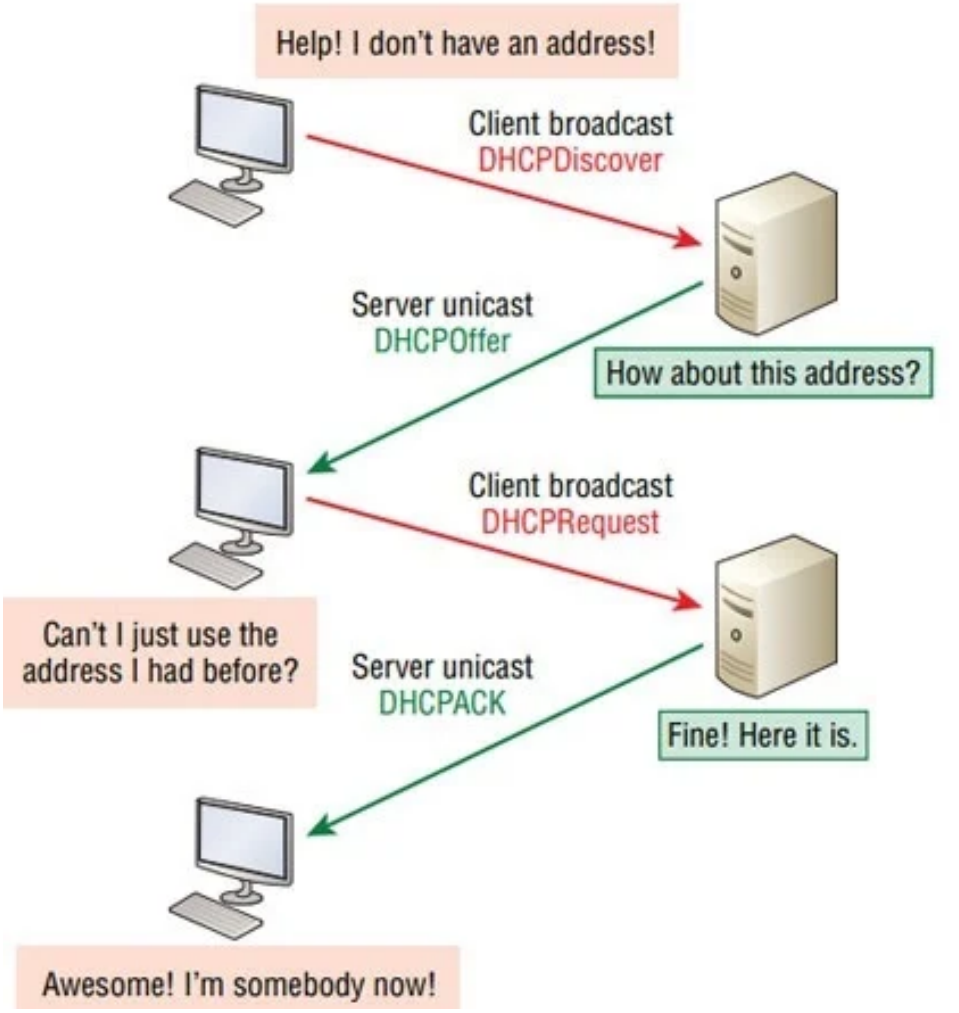


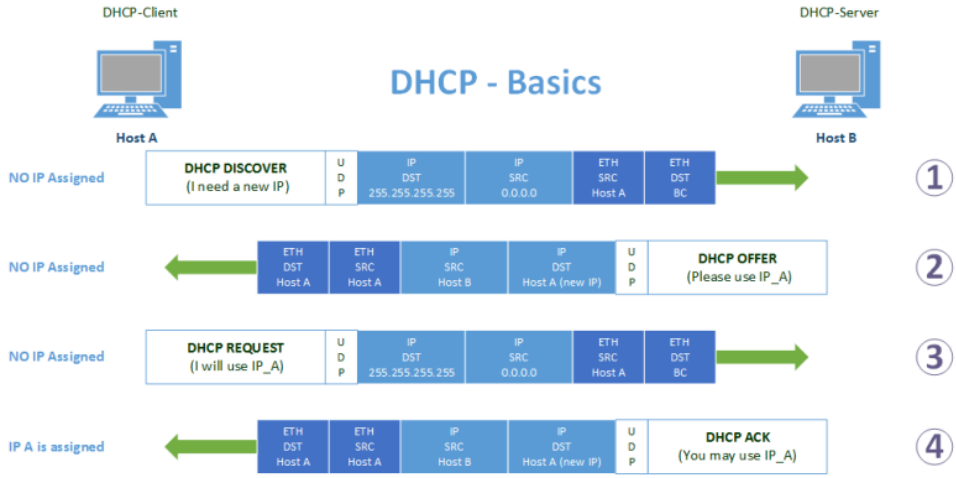


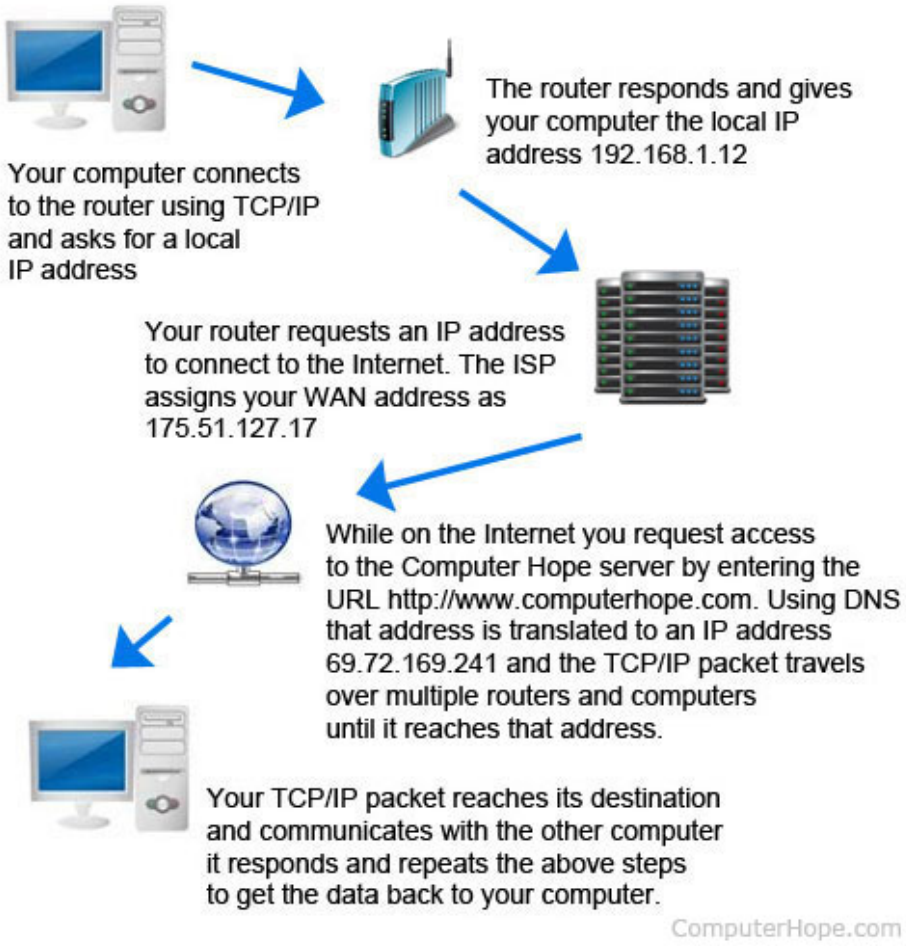
IP Addressing Methods

Dynamic IP

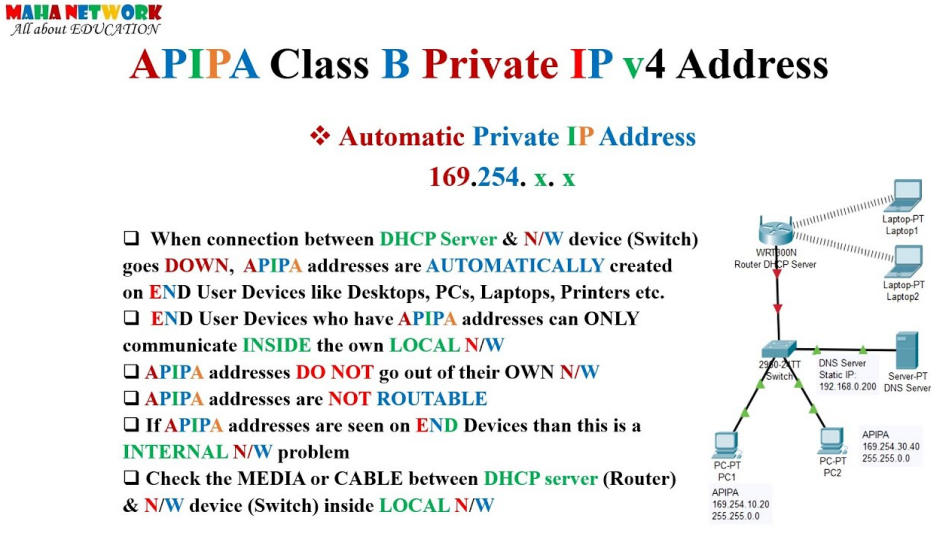
DHCP


Help! I don't have an address!
Client broadcast
DHCPDiscover
Server unicast
DHCPOffer
How about this address?
Client broadcast
DHCPRequest
Can't I just use the address I had before?
Server unicast
DHCPACK
Fine! Here it is.
Awesome! I'm somebody now!

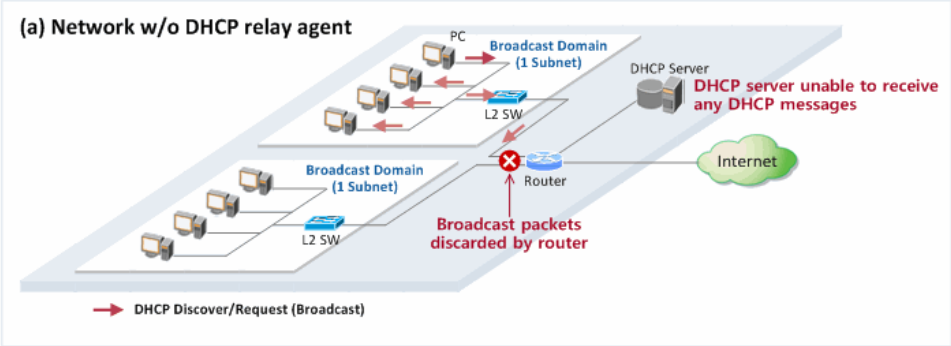

DHCP-Client
DHCP-Server
DHCP - Basics
Host A
Host B
NO IP Assigned
DHCP DISCOVER
(I need a new IP)
UDP
IP DST 255.255.255.255
IP SRC 0.0.0.0
ETH SRC Host A
ETH DST BC
1
NO IP Assigned
ETH DST Host A
ETH SRC Host A
IP SRC Host B
IP DST Host A (new IP)
UDP
DHCP OFFER
(Please use IP_A)
2
NO IP Assigned
DHCP REQUEST
(I will use IP_A)
UDP
IP DST 255.255.255.255
IP SRC 0.0.0.0
ETH SRC Host A
ETH DST BC
3
IP A is assigned
ETH DST Host A
ETH SRC Host A
IP SRC Host B
IP DST Host A (new IP)
UDP
DHCP ACK
(You may use IP_A)
4

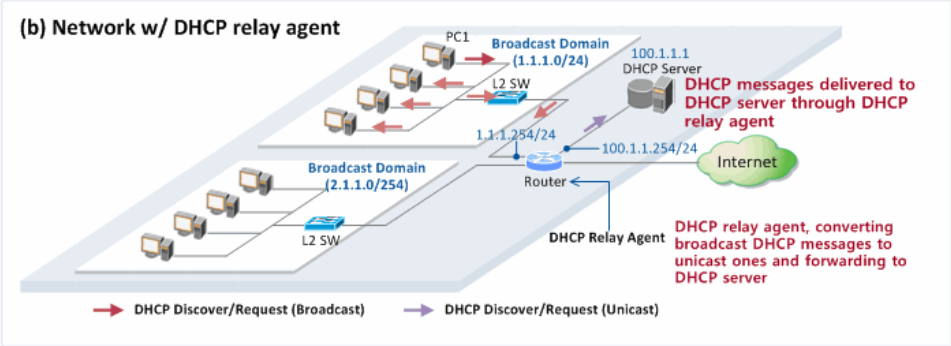


APIPA



DHCP Relay = IP Helper

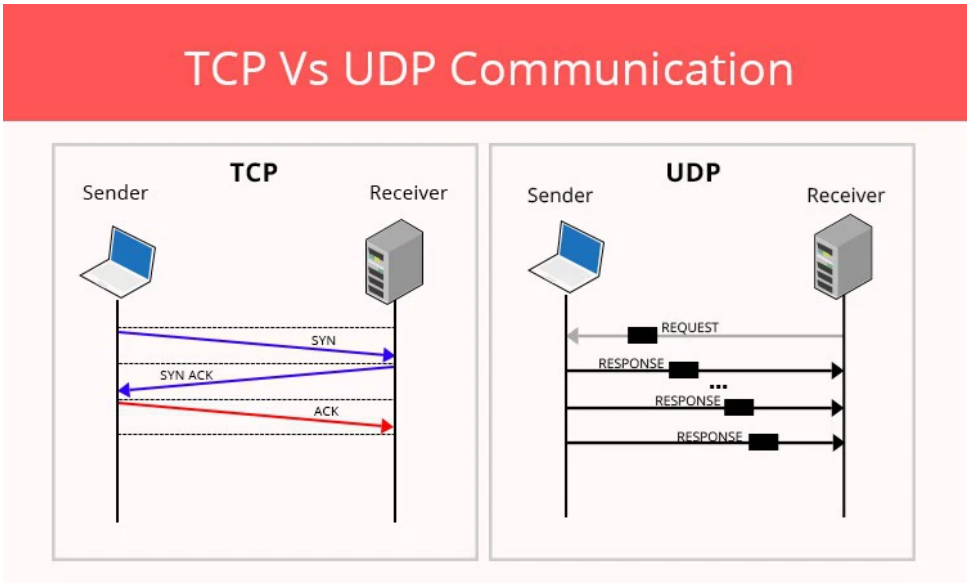


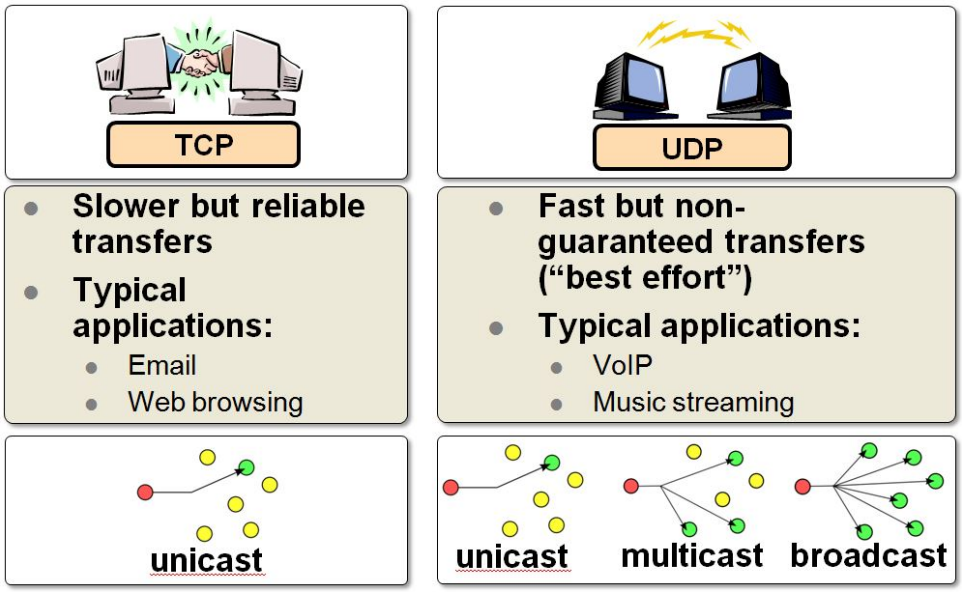


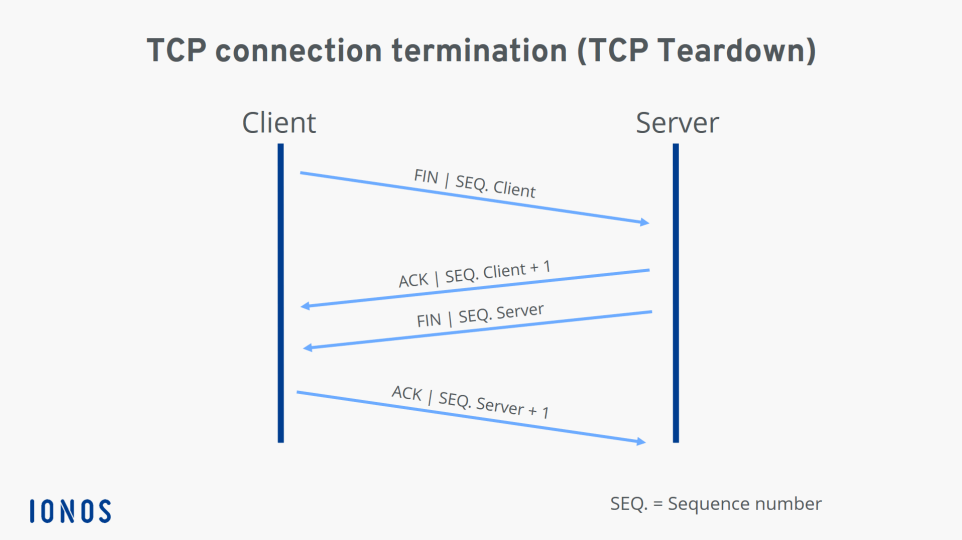
ARP(Address Resolution Protocl) / RARP

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

TCP vs UDP



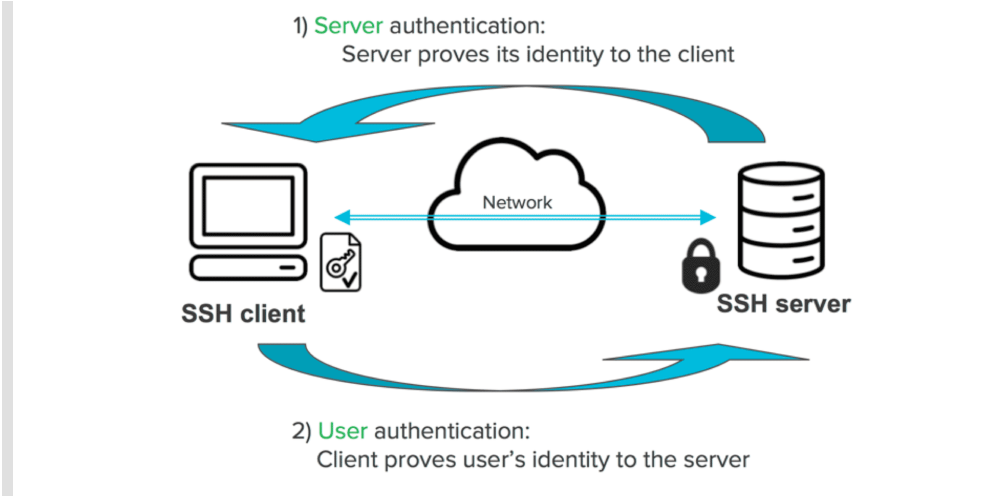


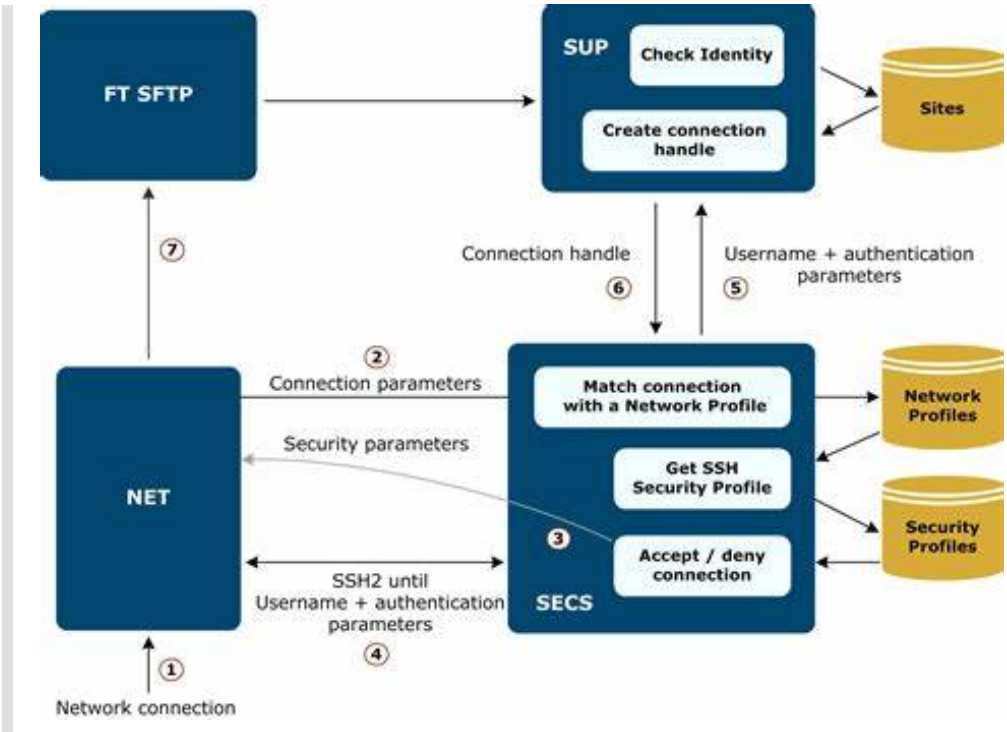


Ports

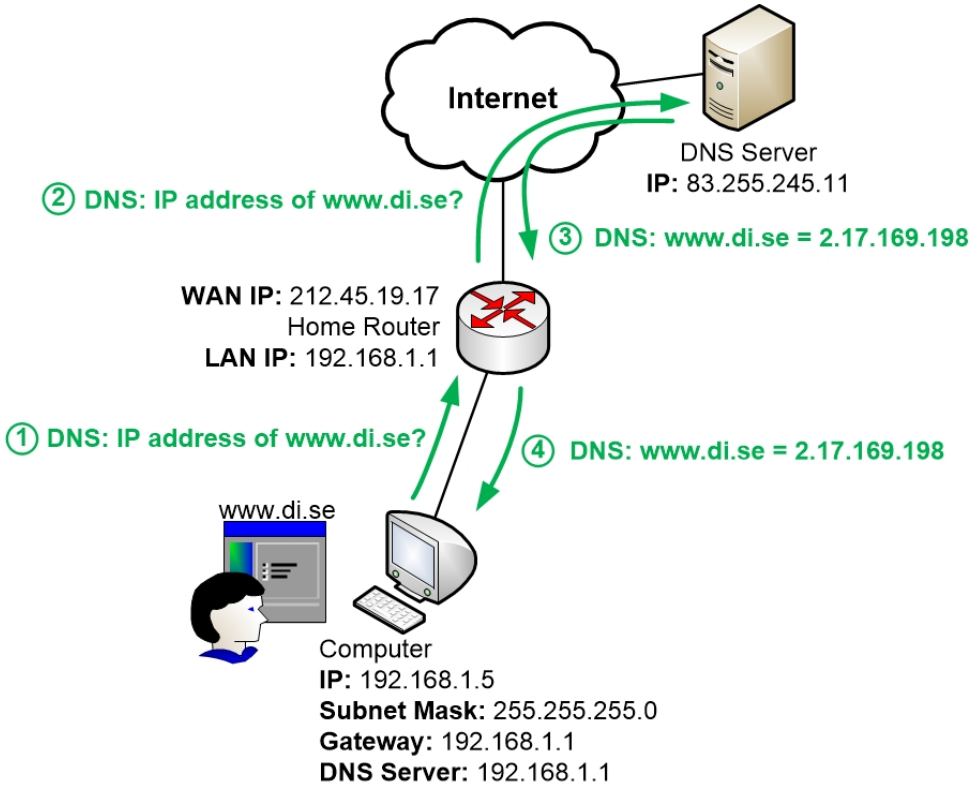
Port Number	Transport Protocol	Service Name	RFC
20, 21	TCP	File Transfer Protocol (FTP)	RFC 959
22	TCP and UDP	Secure Shell (SSH)	RFC 4250-4256
23	TCP	Telnet	RFC 854
25	TCP	Simple Mail Transfer Protocol (SMTP)	RFC 5321
53	TCP and UDP	Domain Name Server (DNS)	RFC 1034-1035
67, 68	UDP	Dynamic Host Configuration Protocol (DHCP)	RFC 2131
69	UDP	Trivial File Transfer Protocol (TFTP)	RFC 1350
80	TCP	HyperText Transfer Protocol (HTTP)	RFC 2616
110	TCP	Post Office Protocol (POP3)	RFC 1939
119	TCP	Network News Transport Protocol (NNTP)	RFC 8977
123	UDP	Network Time Protocol (NTP)	RFC 5905
135-139	TCP and UDP	NetBIOS	RFC 1001-1002
143	TCP and UDP	Internet Message Access Protocol (IMAP4)	RFC 3501
161, 162	TCP and UDP	Simple Network Management Protocol (SNMP)	RFC 1901-1908, 3411-3418
179	TCP	Border Gateway Protocol (BGP)	RFC 4271
389	TCP and UDP	Lightweight Directory Access Protocol	RFC 4510
443	TCP and UDP	HTTP with Secure Sockets Layer (SSL)	RFC 2818
500	UDP	Internet Security Association and Key Management Protocol (ISAKMP) / Internet Key Exchange (IKE)	RFC 2408 - 2409
636	TCP and UDP	Lightweight Directory Access Protocol over TLS/SSL (LDAPS	RFC 4513
989/990	TCP	FTP over TLS/SSL	RFC 4217

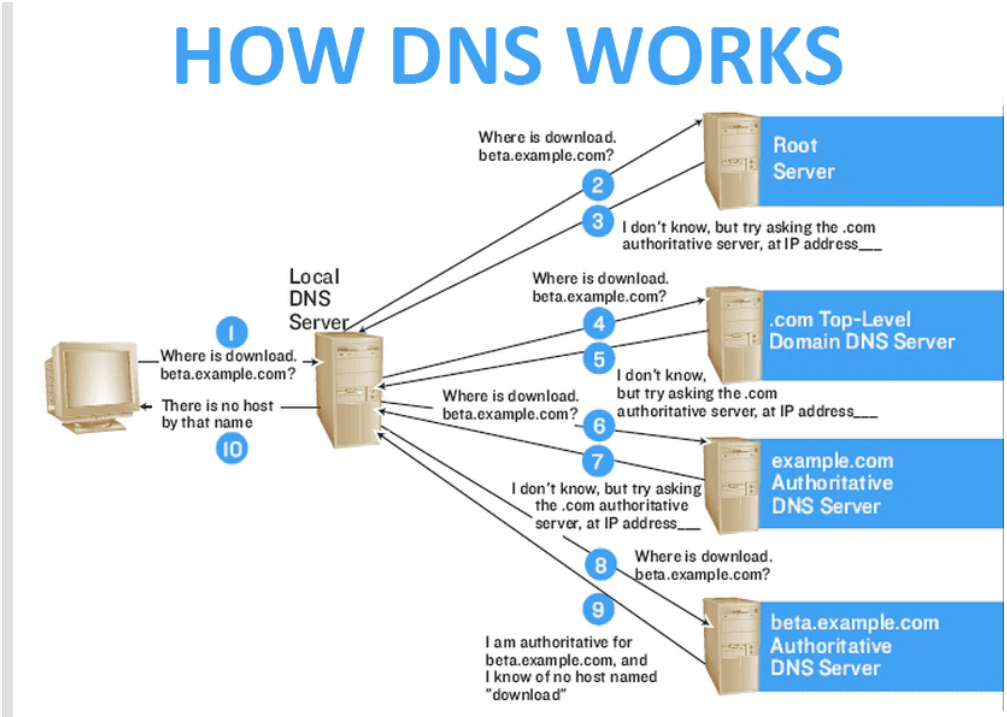
SSH(Secure Shell)

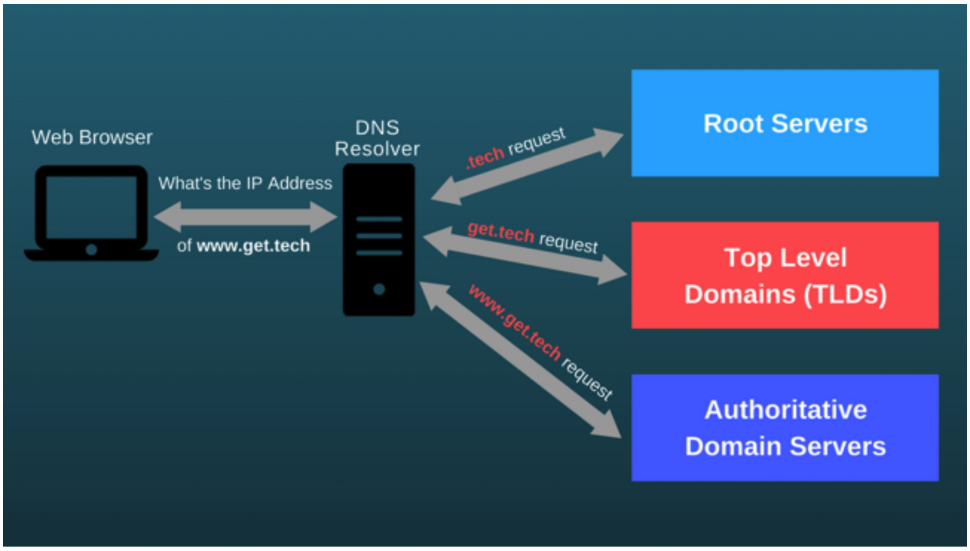


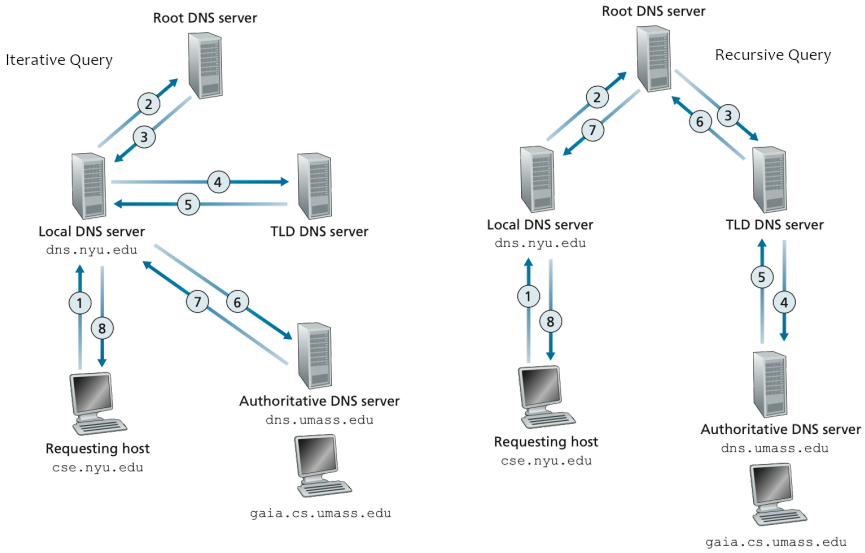


DNS(Domain Name System)

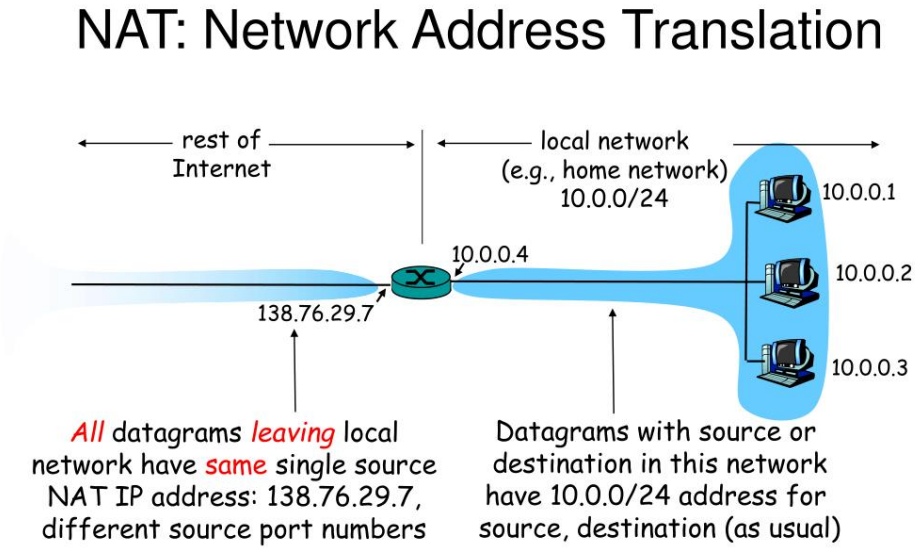

Internet
DNS Server
IP: 83.255.245.11
② DNS: IP address of www.di.se?
③ DNS: www.di.se = 2.17.169.198
WAN IP: 212.45.19.17
Home Router
LAN IP: 192.168.1.1
① DNS: IP address of www.di.se?
④ DNS: www.di.se = 2.17.169.198
www.di.se
Computer
IP: 192.168.1.5
Subnet Mask: 255.255.255.0
Gateway: 192.168.1.1
DNS Server: 192.168.1.1

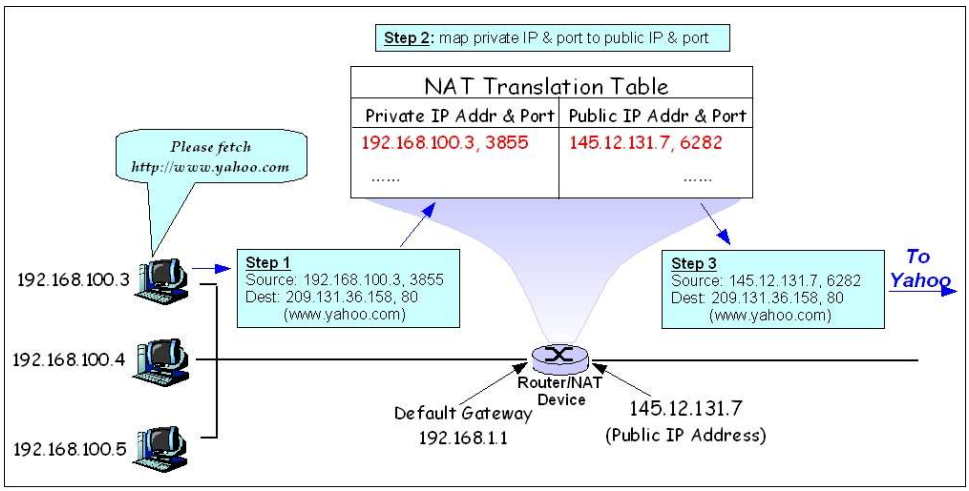

HOW DNS WORKS
Where is download. beta.example.com?
Root Server
I don't know, but try asking the .com authoritative server, at IP address___
Local DNS Server
Where is download. beta.example.com?
.com Top-Level Domain DNS Server
I don't know, but try asking the .com authoritative server, at IP address___
Where is download. beta.example.com?
There is no host by that name
Where is download. beta.example.com?
example.com Authoritative DNS Server
I don't know, but try asking the .com authoritative server, at IP address___
Where is download. beta.example.com?
beta.example.com Authoritative DNS Server
I am authoritative for beta.example.com, and I know of no host named "download"

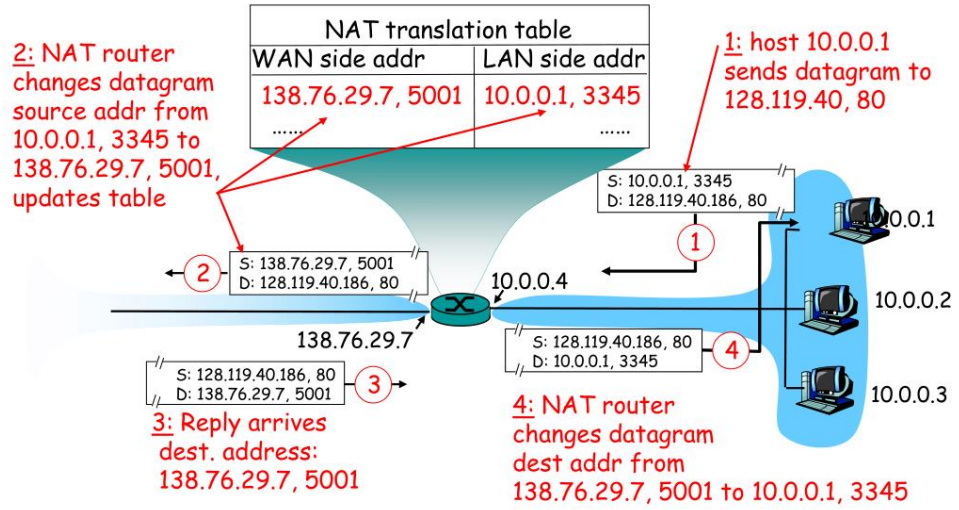


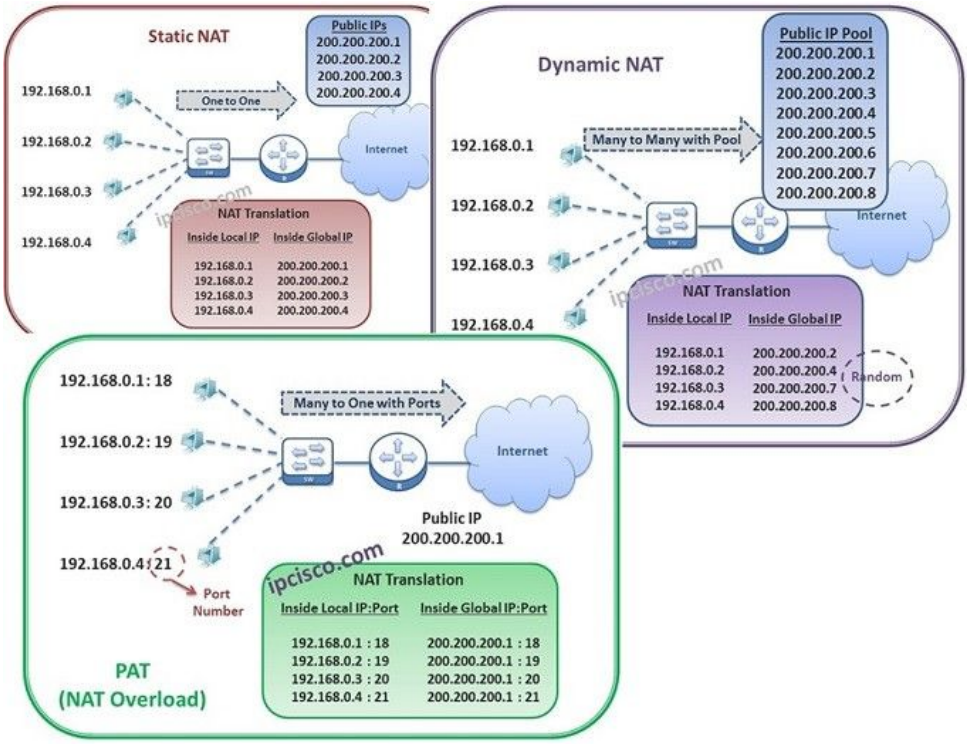


NAT(Network Address Translation)

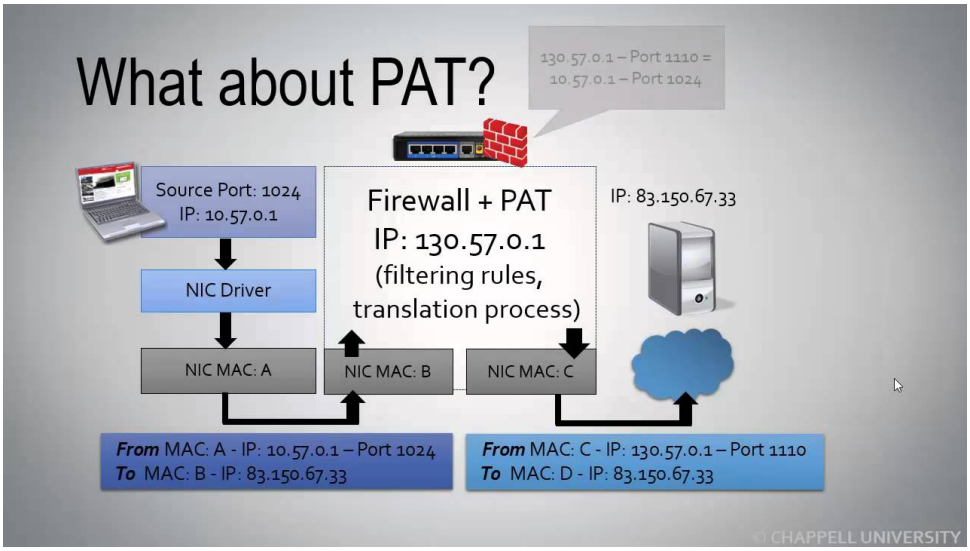


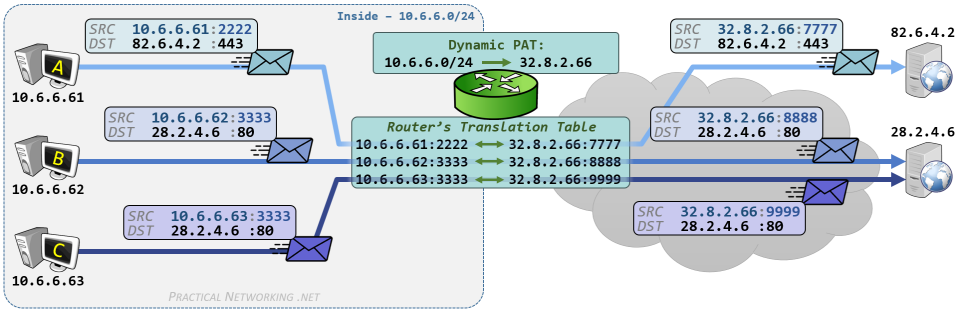

Step 2: map private IP & port to public IP & port
NAT Translation Table
Private IP Addr & Port
Public IP Addr & Port
192.168.100.3, 3855
145.12.131.7, 6282
Please fetch http://www.yahoo.com
192.168.100.3
192.168.100.4
192.168.100.5
Step 1
Source: 192.168.100.3, 3855
Dest: 209.131.36.158, 80
(www.yahoo.com)
Step 3
Source: 145.12.131.7, 6282
Dest: 209.131.36.158, 80
(www.yahoo.com)
To Yahoo
Router/NAT Device
Default Gateway
192.168.1.1
145.12.131.7
(Public IP Address)


NAT: Network Address Translation
NAT translation table
WAN side addr
LAN side addr
138.76.29.7, 5001
10.0.0.1, 3345
2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table
1: host 10.0.0.1 sends datagram to 128.119.40, 80
S: 10.0.0.1, 3345
D: 128.119.40.186, 80
10.0.0.1
10.0.0.2
10.0.0.3
10.0.0.4
138.76.29.7
S: 138.76.29.7, 5001
D: 128.119.40.186, 80
S: 128.119.40.186, 80
D: 138.76.29.7, 5001
S: 128.119.40.186, 80
D: 10.0.0.1, 3345
3: Reply arrives dest. address: 138.76.29.7, 5001
4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

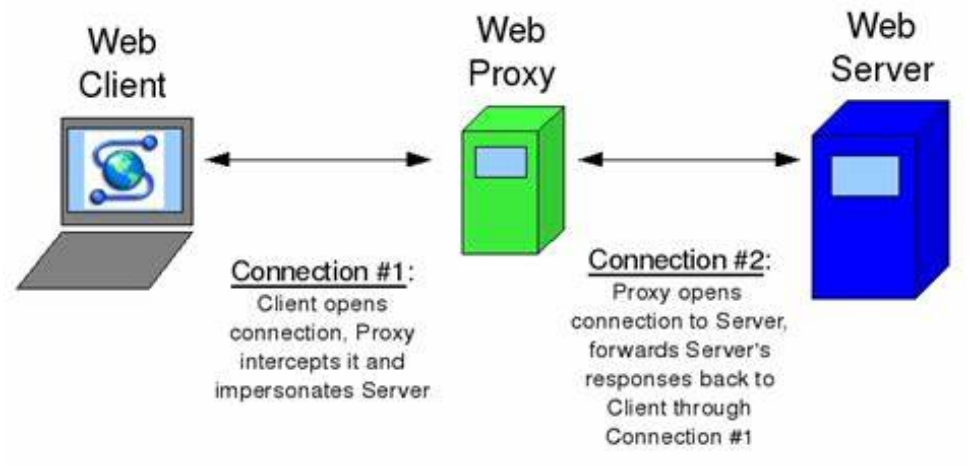

Static NAT
Public IPs
200.200.200.1
200.200.200.2
200.200.200.3
200.200.200.4
192.168.0.1
192.168.0.2
192.168.0.3
192.168.0.4
One to One
Internet
NAT Translation
Inside Local IP
Inside Global IP
Dynamic NAT
Public IP Pool
200.200.200.1
200.200.200.2
200.200.200.3
200.200.200.4
200.200.200.5
200.200.200.6
200.200.200.7
200.200.200.8
Many to Many with Pool
Random
192.168.0.1: 18
192.168.0.2: 19
192.168.0.3: 20
192.168.0.4: 21
Many to One with Ports
Public IP
200.200.200.1
Port Number
Inside Local IP:Port
Inside Global IP:Port
192.168.0.1 : 18
192.168.0.2 : 19
192.168.0.3 : 20
192.168.0.4 : 21
200.200.200.1 : 18
200.200.200.1 : 19
200.200.200.1 : 20
200.200.200.1 : 21
PAT
(NAT Overload)
ipcisco.com

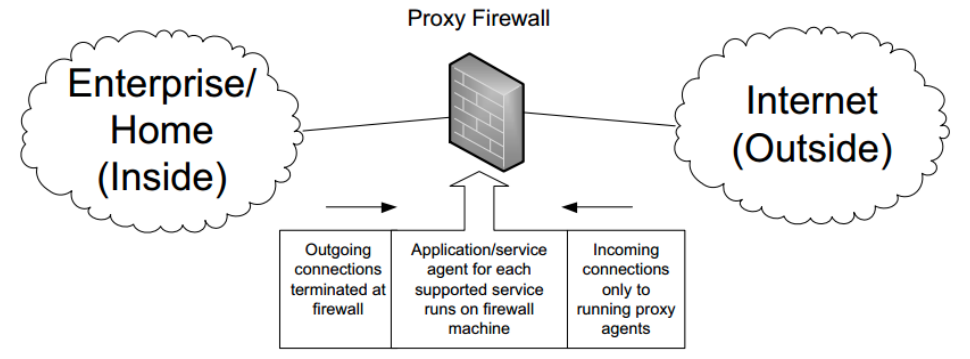
PAT(Port Address Translation)





Proxy





Proxy Cache

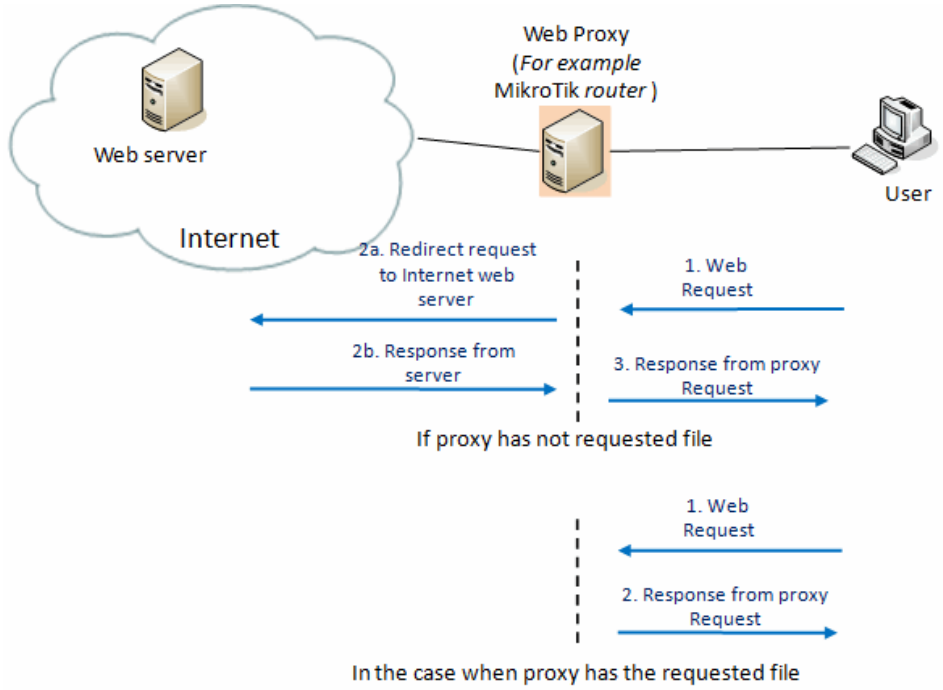
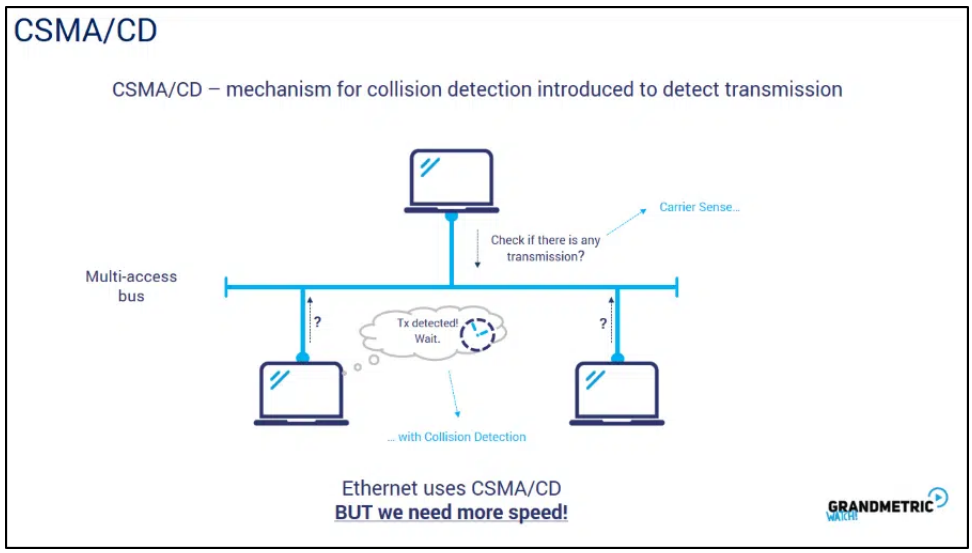
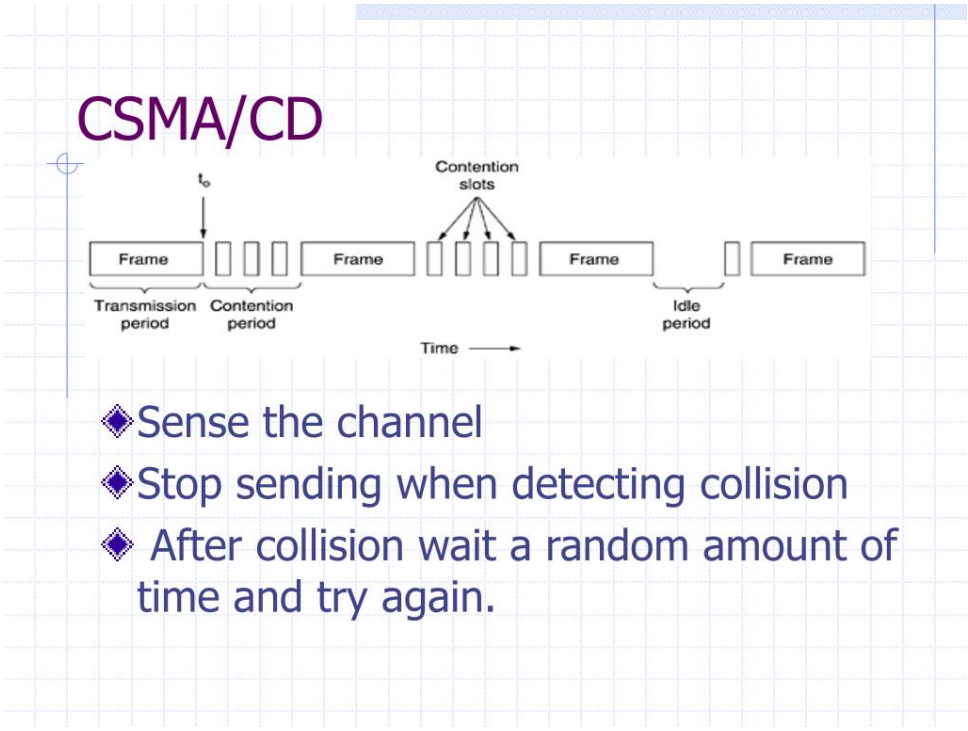
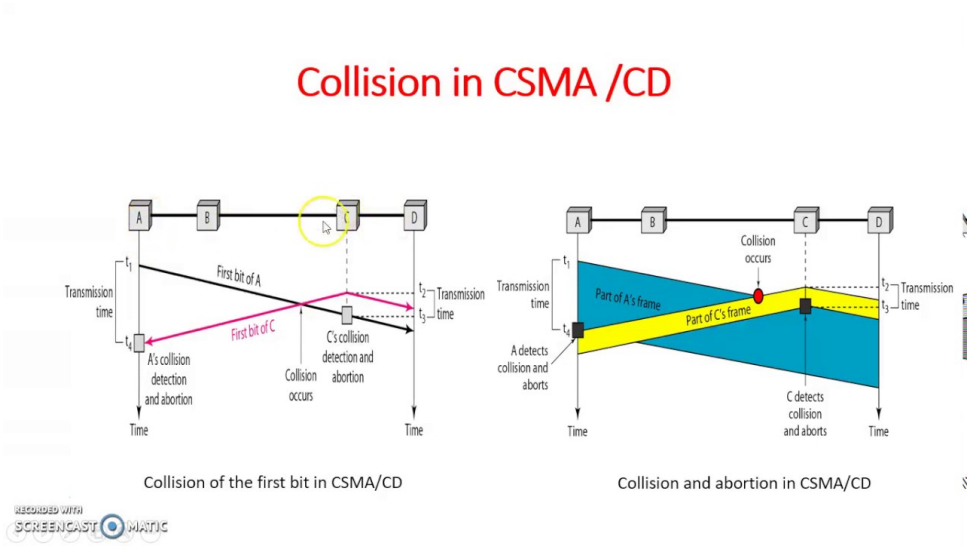


Figure 10.1. Web proxy basic operation scheme

CSMA/CD



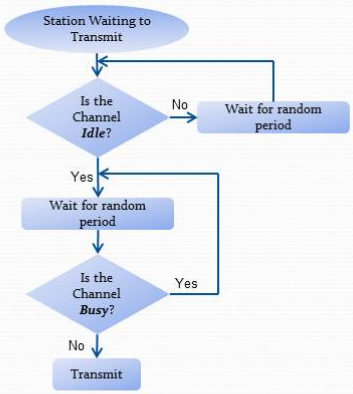

CSMA/CD
Contention slots
Frame
Frame
Frame
Frame
Transmission period
Contention period
Idle period
Time
Sense the channel
Stop sending when detecting collision
After collision wait a random amount of time and try again.


Collision in CSMA /CD
A
B
C
D
Transmission time
First bit of A
First bit of C
A's collision detection and abortion
Collision occurs
C's collision detection and abortion
Transmission time
Time
Collision occurs
Part of A's frame
Part of C's frame
A detects collision and aborts
C detects collision and aborts
Collision of the first bit in CSMA/CD
Collision and abortion in CSMA/CD

CSMA/CA

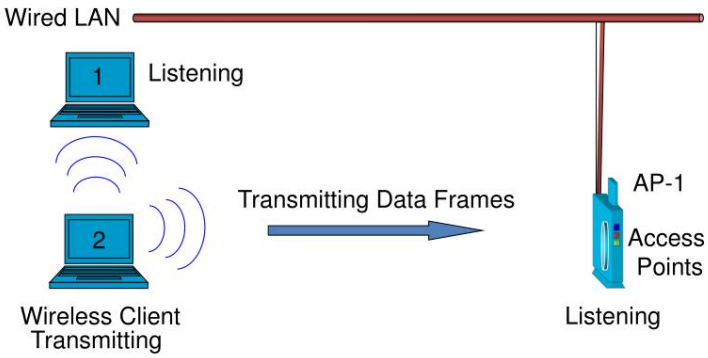
CSMA/CA

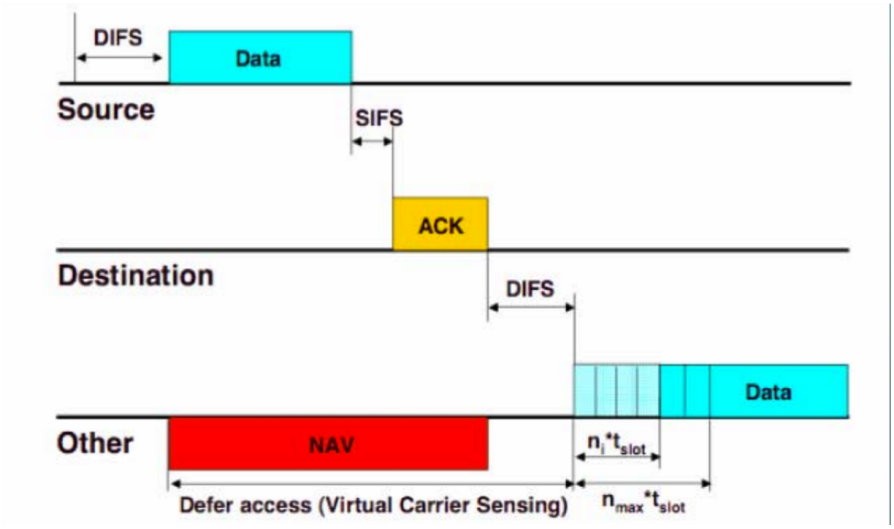
- CSMA/CA is a wireless network multiple access method in which:
  - A carrier sensing scheme is used.
  - A node wishing to transmit data has to first listen to the channel for a predetermined amount of time whether or not another node is transmitting on channel within the wireless range. If the channel 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 explained later in the presentation



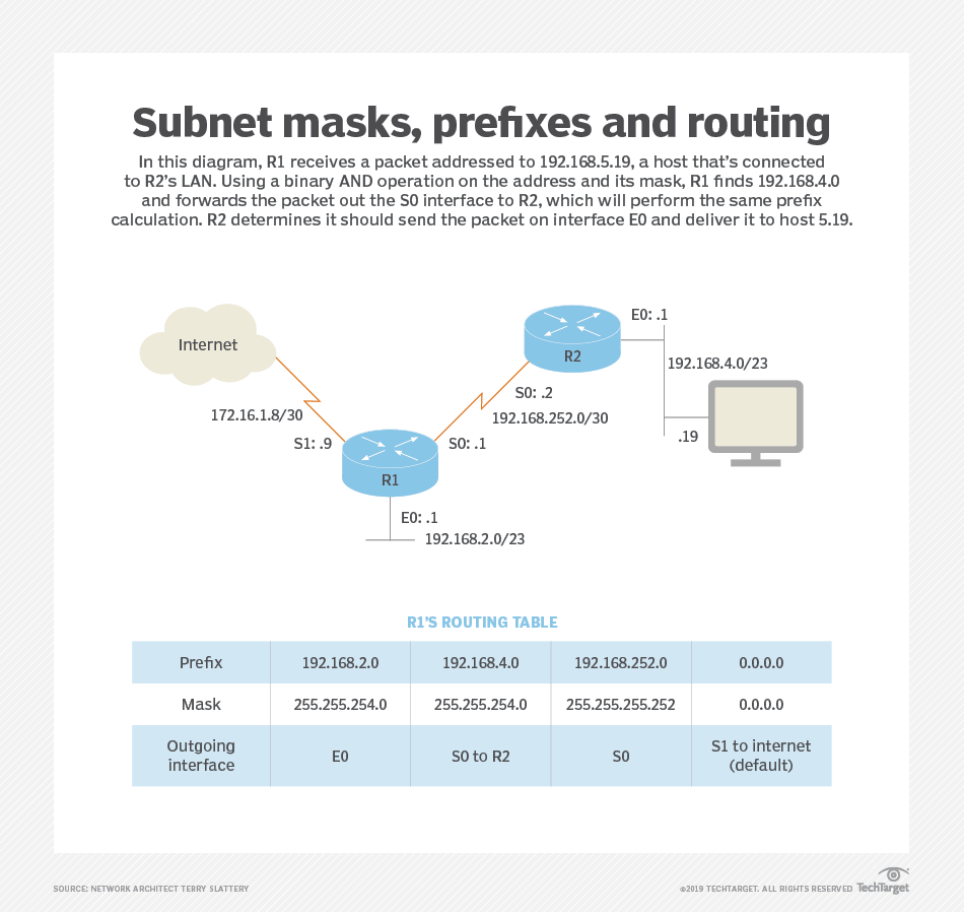
CSMA/CA Collision Handling

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

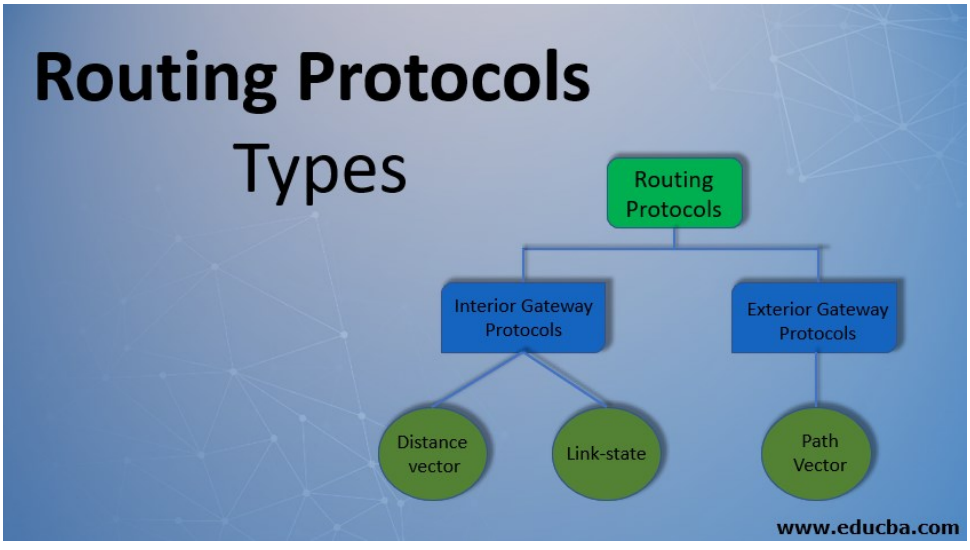


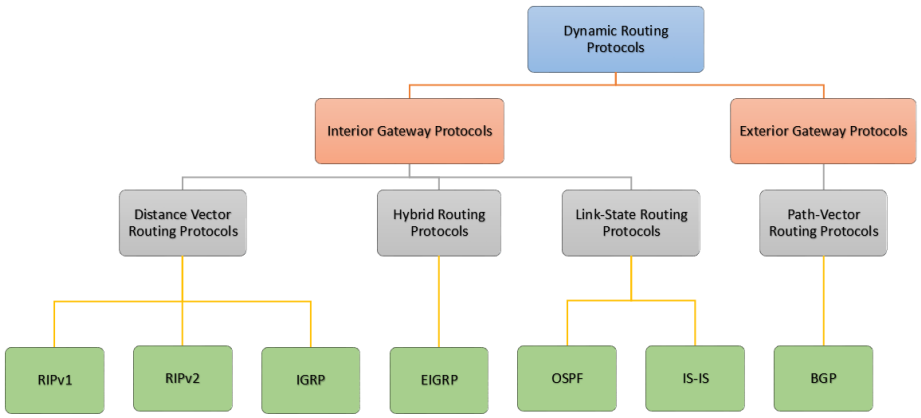


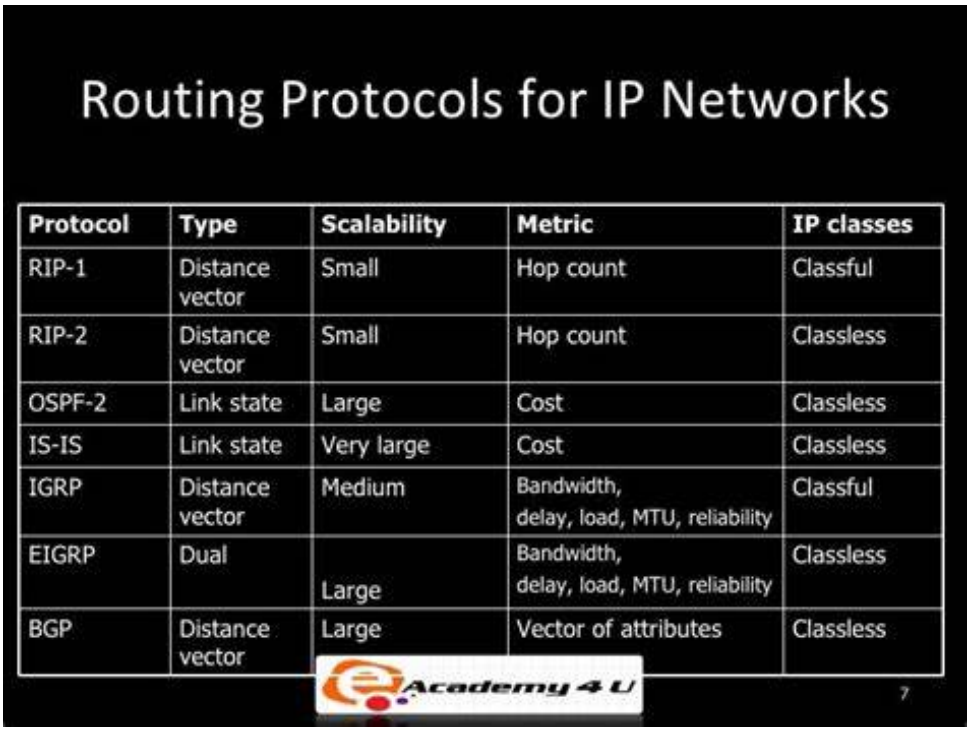
Routing Table



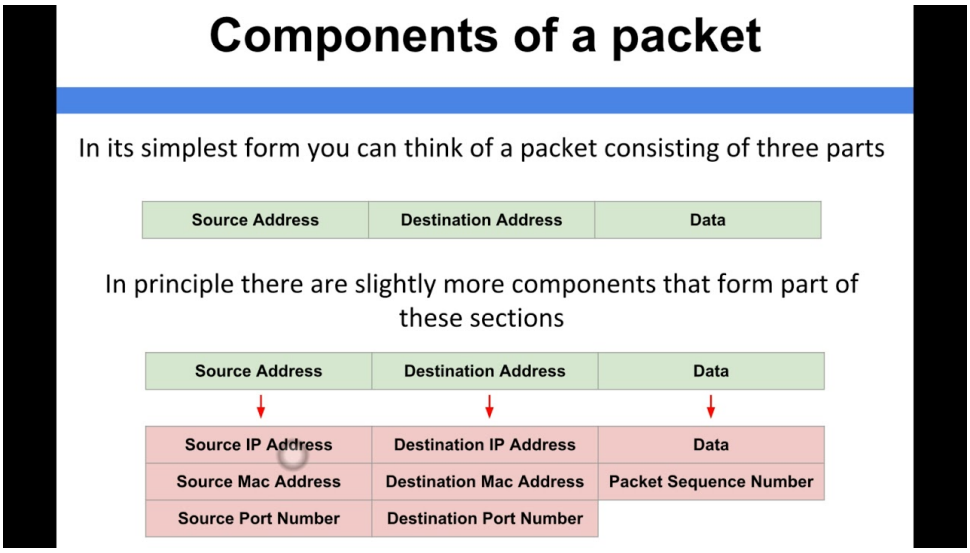
Routing Protocol

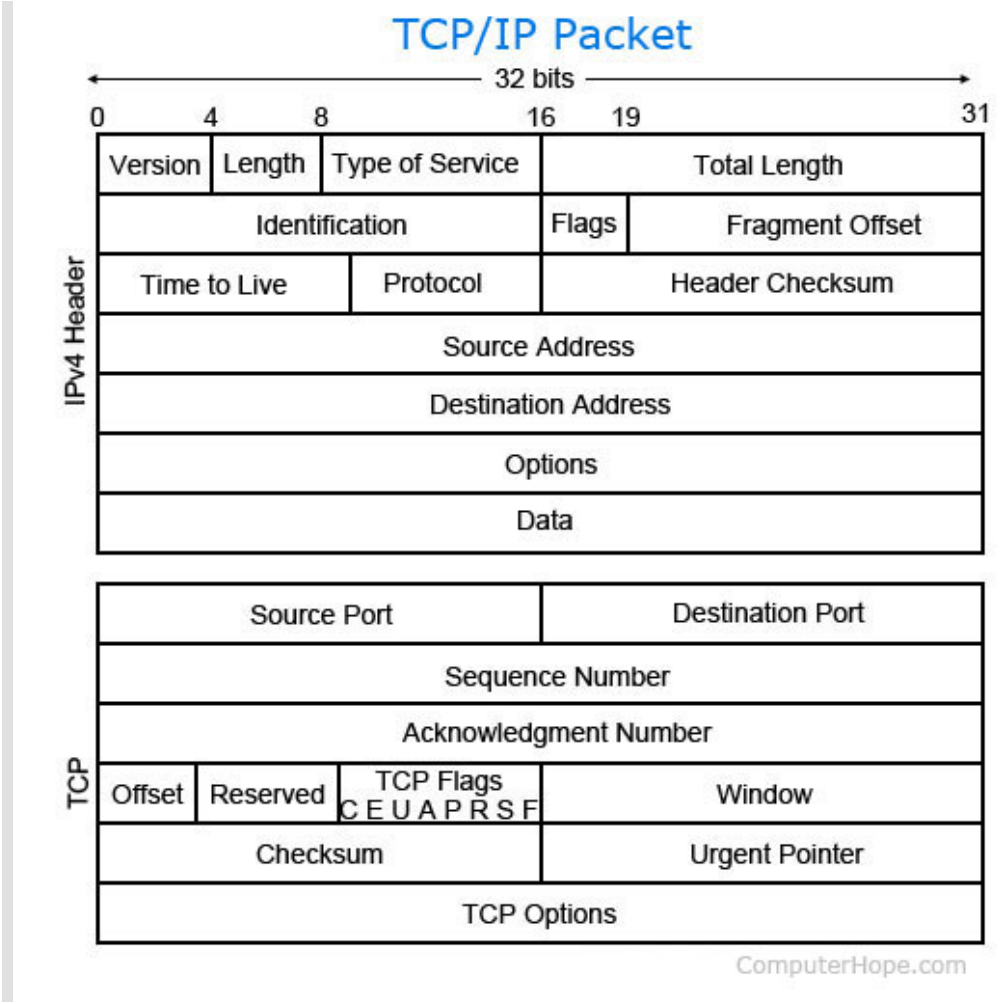


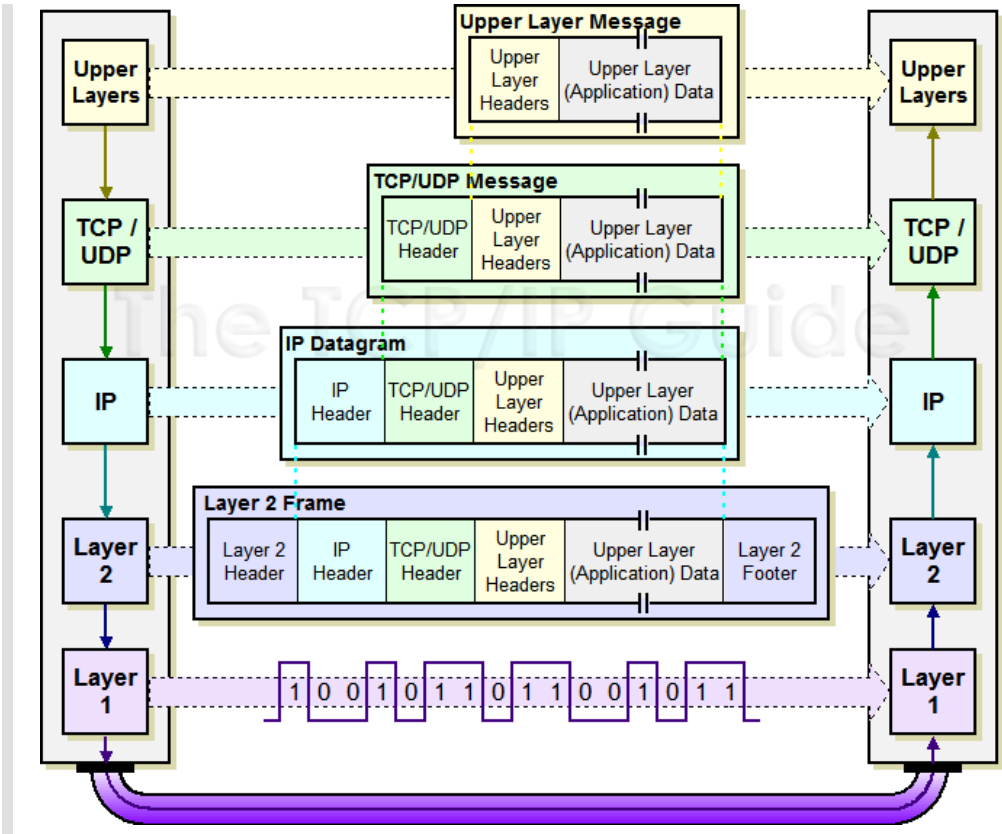

Dynamic Routing Protocols
Interior Gateway Protocols
Exterior Gateway Protocols
Distance Vector Routing Protocols
Hybrid Routing Protocols
Link-State Routing Protocols
Path-Vector Routing Protocols
RIPv1
RIPv2
IGRP
EIGRP
OSPF
IS-IS
BGP


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
Academy 4 U
7

Package


Components of a packet
In its simplest form you can think of a packet consisting of three parts
Source Address | Destination Address | Data
In principle there are slightly more components that form part of these sections
Source Address | Destination Address | Data
Source IP Address | Destination IP Address | Data
Source Mac Address | Destination Mac Address | Packet Sequence Number
Source Port Number | Destination Port Number


TCP/IP Packet
32 bits
0
4
8
16
19
31
IPv4 Header
Version
Length
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Data
TCP
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Offset
Reserved
TCP Flags
C E U A P R S F
Window
Checksum
Urgent Pointer
TCP Options
ComputerHope.com


Upper Layers
TCP / UDP
IP
Layer 2
Layer 1
Upper Layer Message
Upper Layer Headers
Upper Layer (Application) Data
TCP/UDP Message
TCP/UDP Header
IP Datagram
IP Header
Layer 2 Frame
Layer 2 Header
Layer 2 Footer
1 0 0 1 0 1 1 0 1 1 0 0 1 0 1 1
The TCP/IP Guide

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

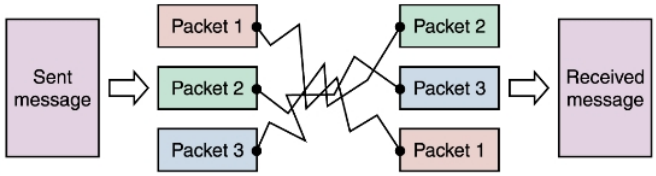
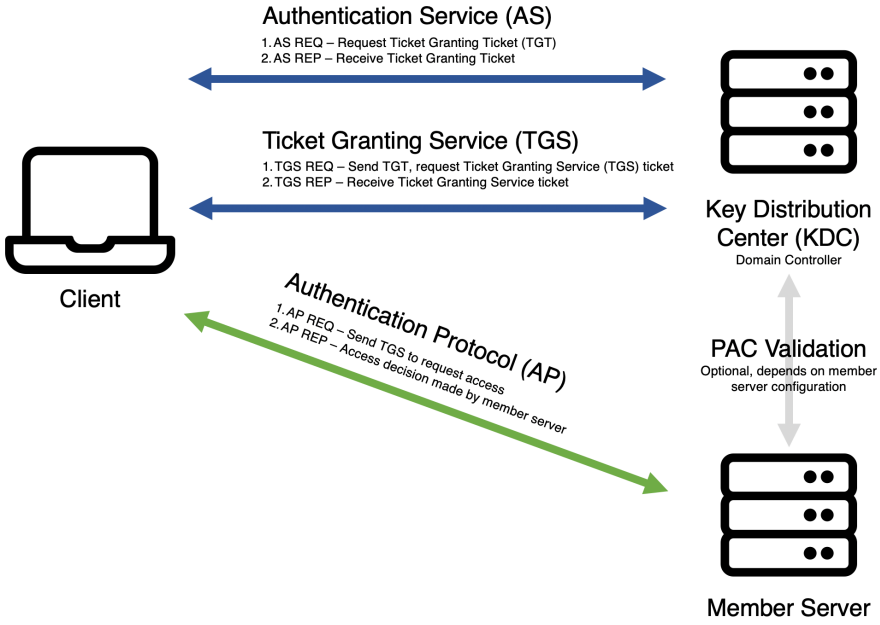


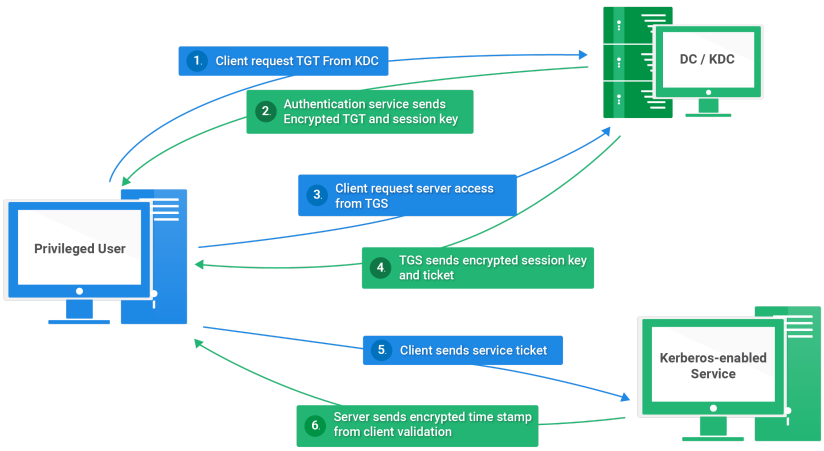
Figure 15.4 Messages sent by packet switching

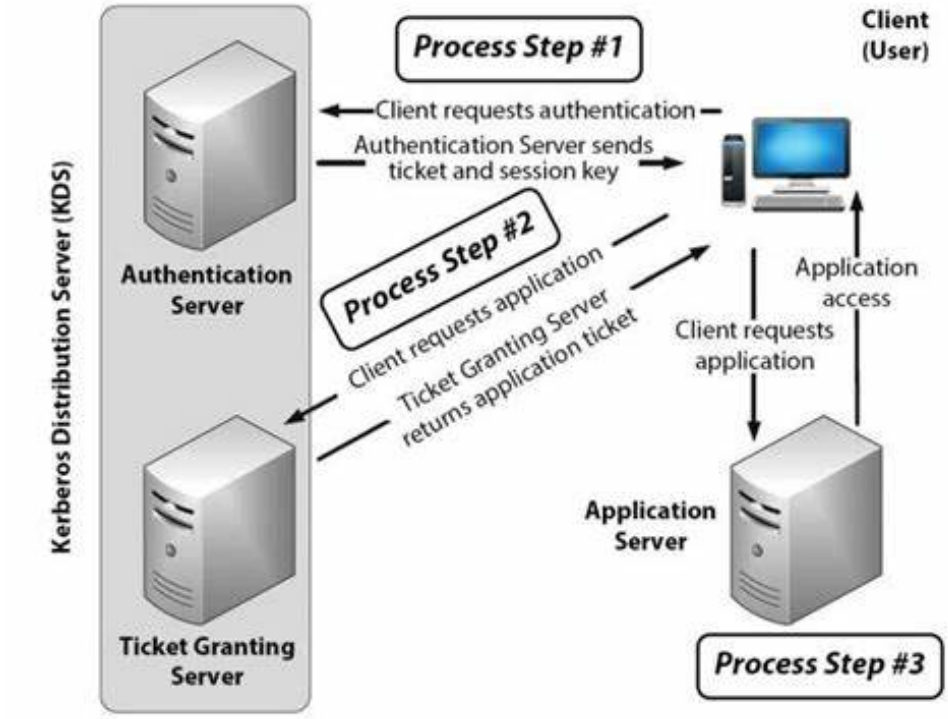
Message is divided into packets | Packets are sent over the Internet by the most expedient route | Packets are reordered and then reassembled

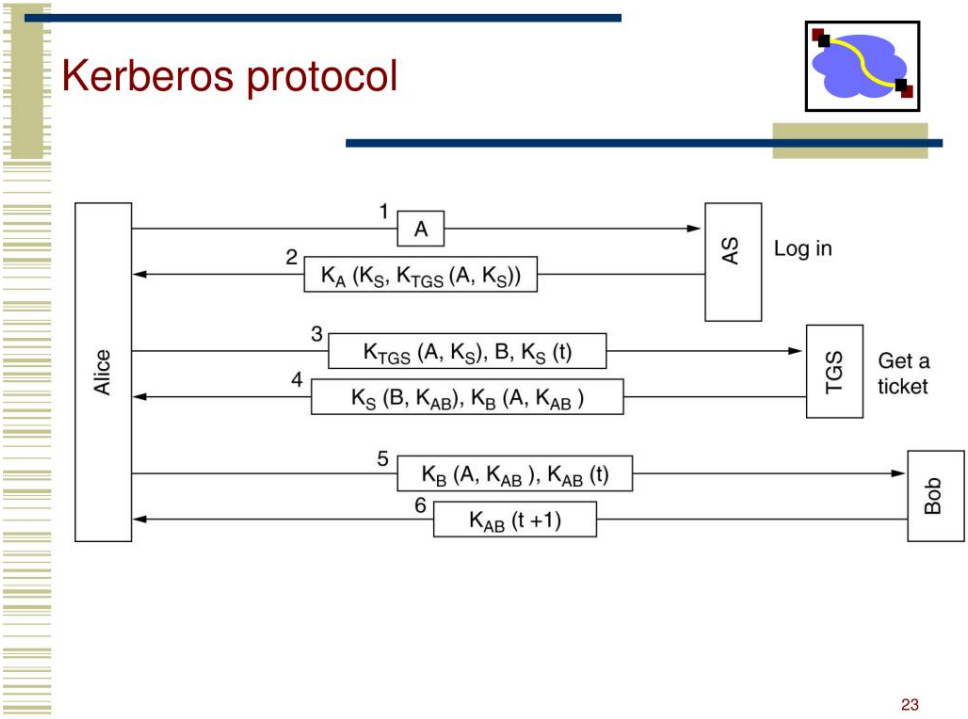
11/27/06 Hofstra University - CSC005 15-18

Kerberos

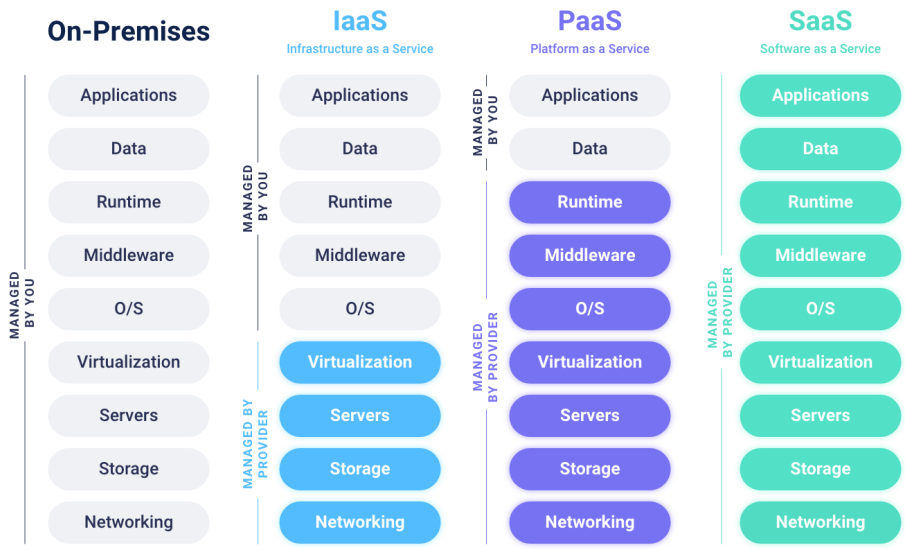




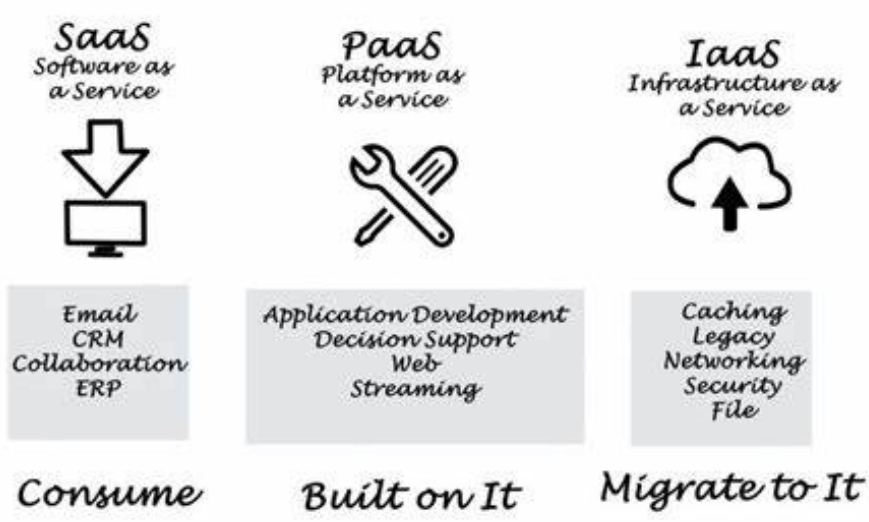


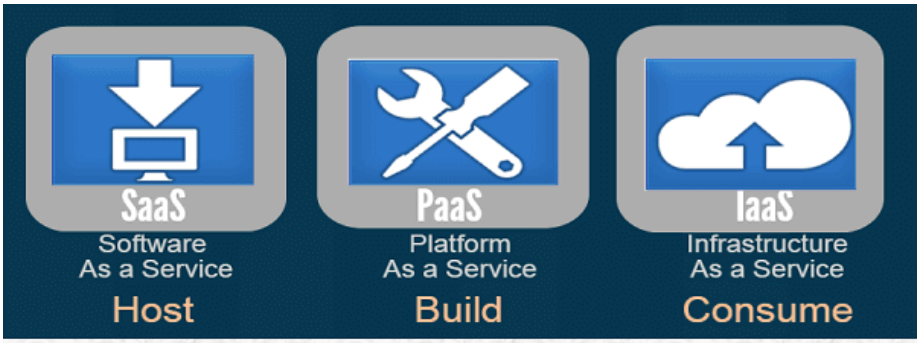


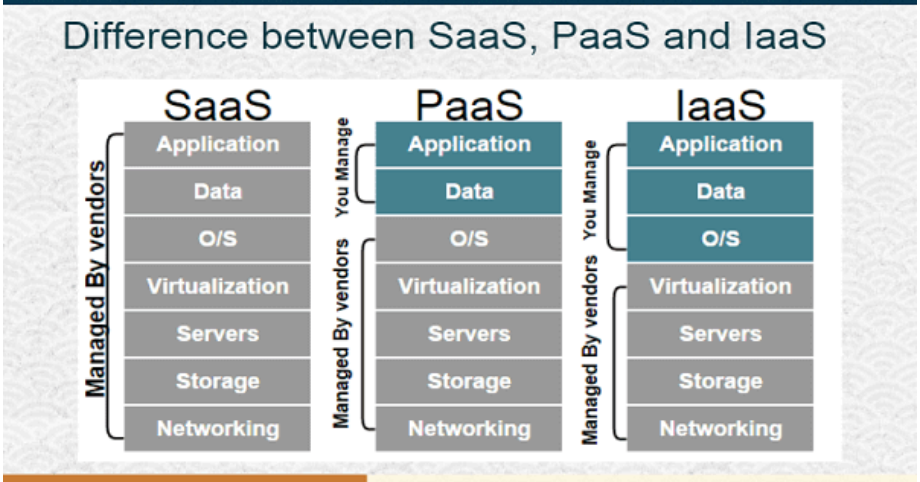
Iaas Paas Saas

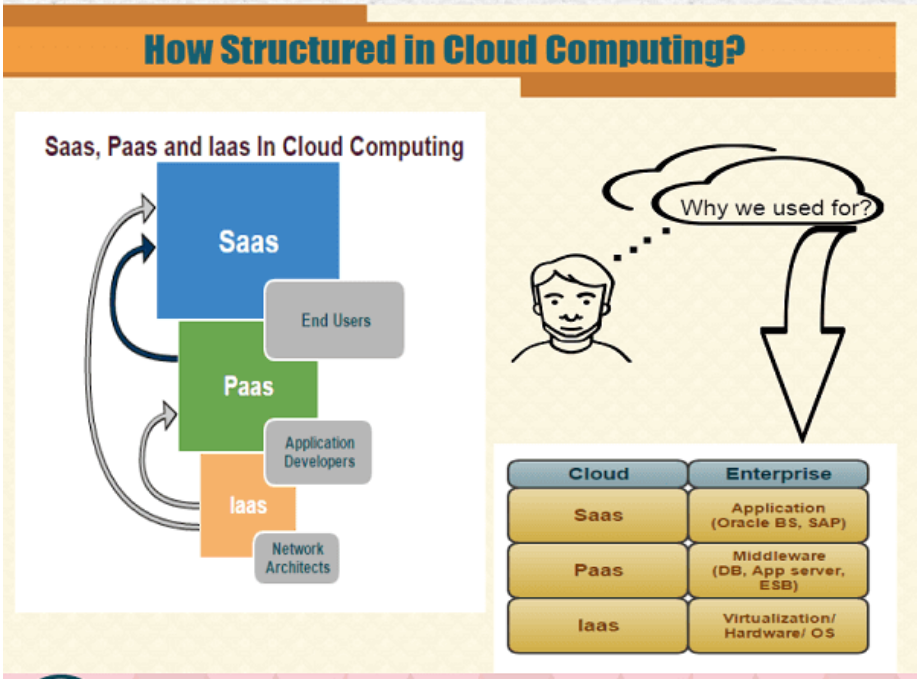

On-Premises
IaaS
Infrastructure as a Service
PaaS
Platform as a Service
SaaS
Software as a Service
MANAGED BY YOU
MANAGED BY PROVIDER
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking

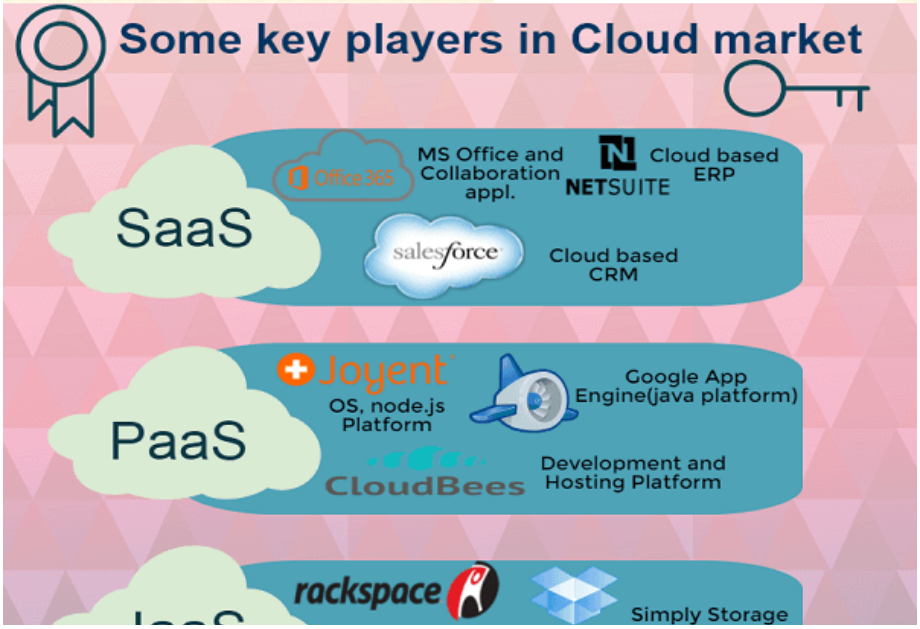

DaaS
SaaS
PaaS
IaaS
Complete Business Solution
Specific Business Software
App Developers
Network Architects


SaaS
Software as a Service
PaaS
Platform as a Service
IaaS
Infrastructure as a Service
Email
CRM
Collaboration
ERP
Application Development
Decision Support
Web
Streaming
Caching
Legacy
Networking
Security
File
Consume
Built on It
Migrate to It


SaaS
Software As a Service
Host
PaaS
Platform As a Service
Build
IaaS
Infrastructure As a Service
Consume

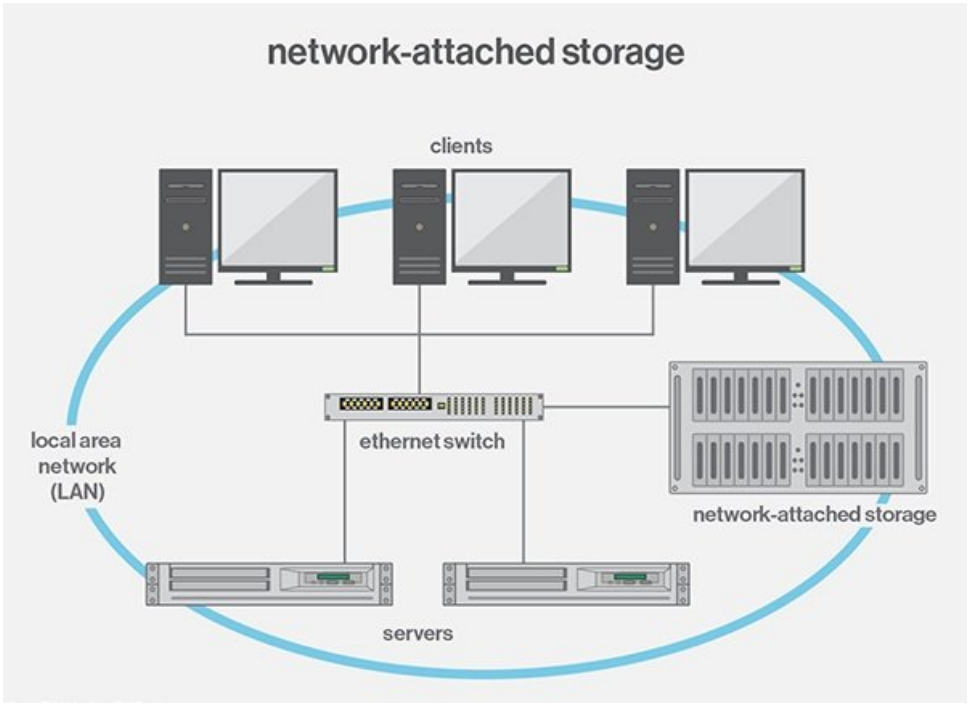

Difference between SaaS, PaaS and IaaS
SaaS
PaaS
IaaS
Application
Data
O/S
Virtualization
Servers
Storage
Networking
Managed By vendors
You Manage

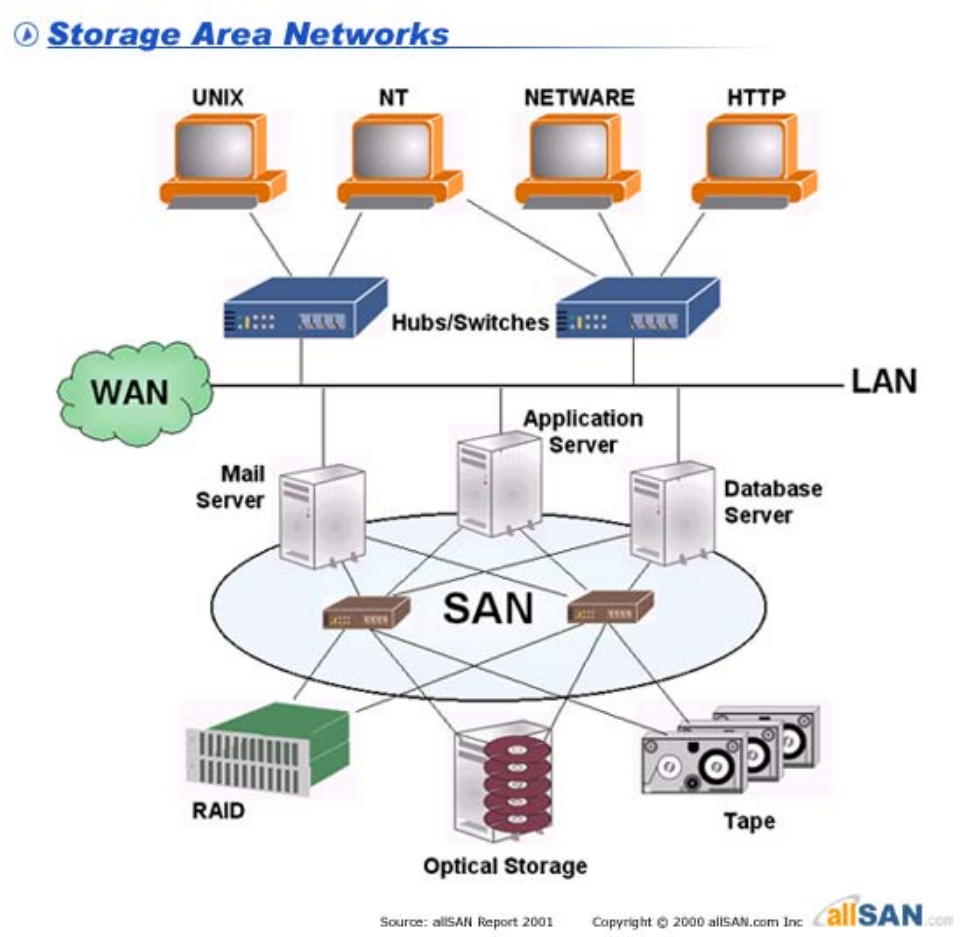

How Structured in Cloud Computing?
Saas, Paas and Iaas In Cloud Computing
Saas
End Users
Paas
Application Developers
Iaas
Network Architects
Why we used for?
Cloud
Enterprise
Saas
Application (Oracle BS, SAP)
Paas
Middleware (DB, App server, ESB)
Iaas
Virtualization/ Hardware/ OS

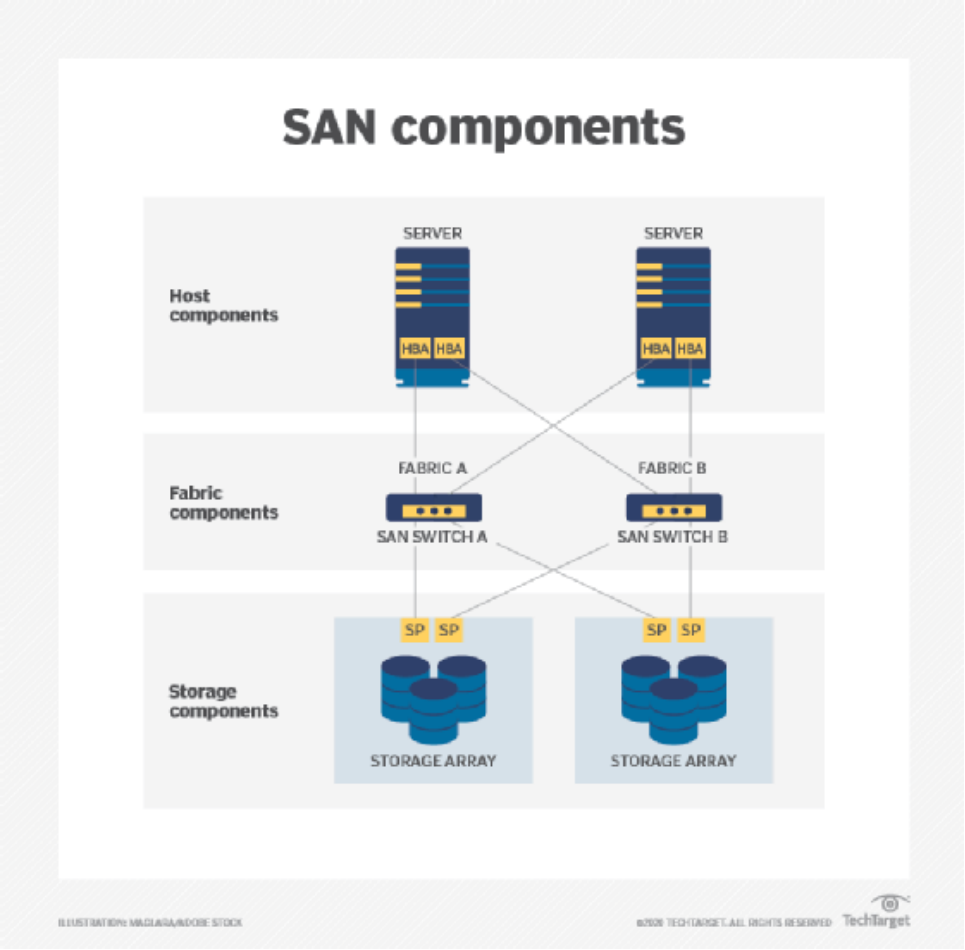

Some key players in Cloud market
SaaS
Office 365
MS Office and Collaboration appl.
NETSUITE
Cloud based ERP
salesforce
Cloud based CRM
PaaS
Joyent
OS, node.js Platform
Google App Engine(java platform)
CloudBees
Development and Hosting Platform
rackspace
Simply Storage

ASSIGNMENT Help
changing the way we learn

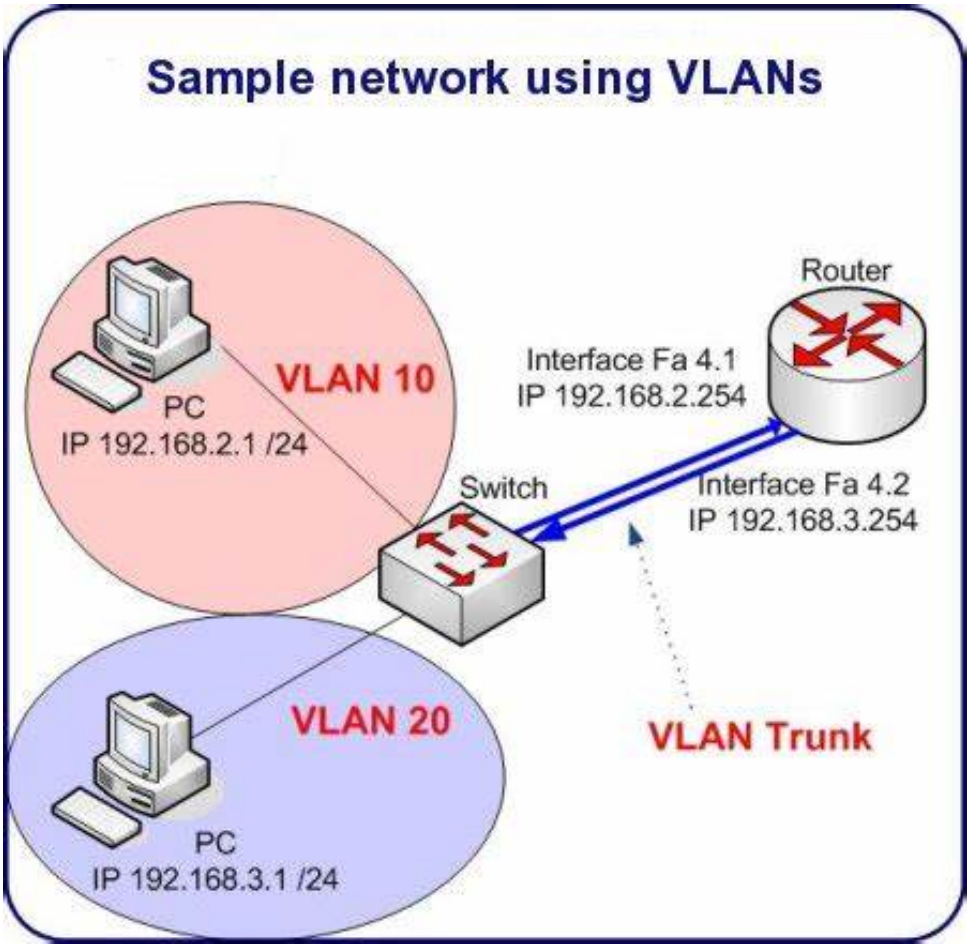
NAS vs SAN



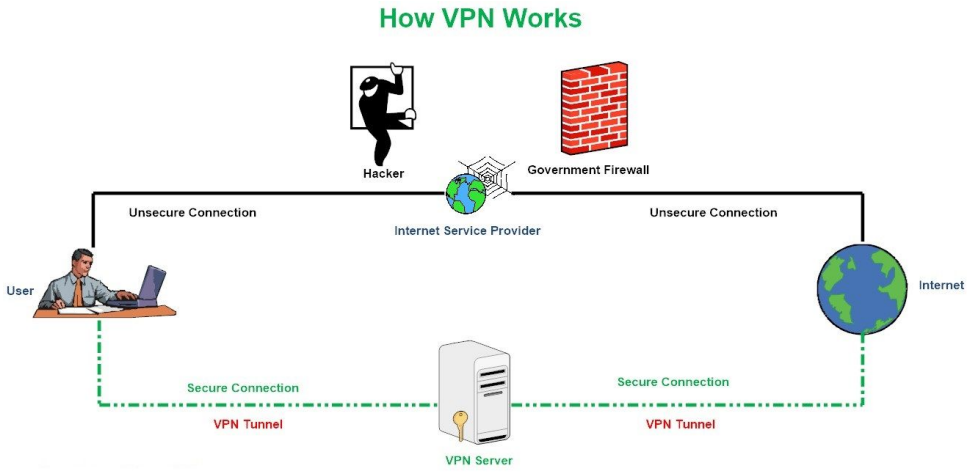

Storage Area Networks
UNIX
NT
NETWARE
HTTP
Hubs/Switches
WAN
LAN
Application Server
Mail Server
Database Server
SAN
RAID
Optical Storage
Tape
Source: allSAN Report 2001
Copyright © 2000 allSAN.com Inc
allSAN

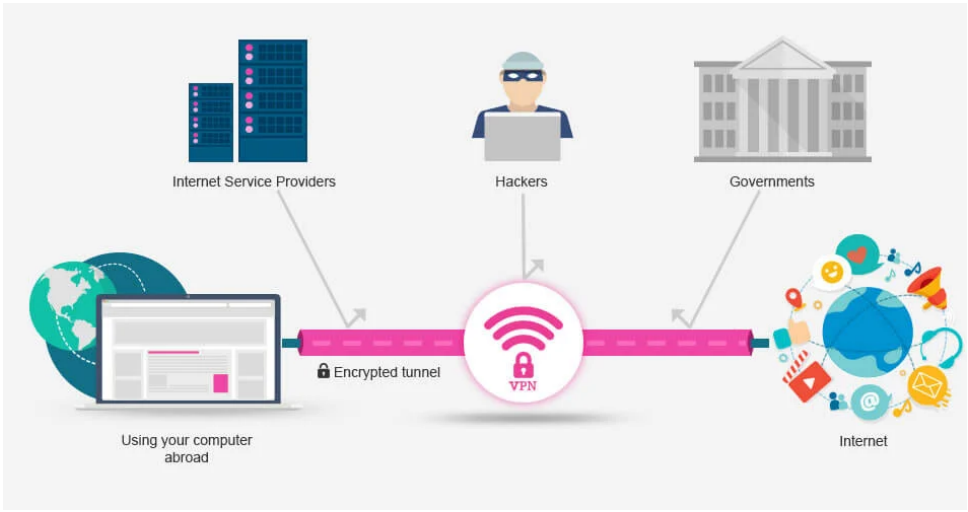

SAN components
Host components
SERVER
SERVER
HBA HBA
HBA HBA
Fabric components
FABRIC A
FABRIC B
SAN SWITCH A
SAN SWITCH B
Storage components
SP SP
SP SP
STORAGE ARRAY
STORAGE ARRAY
TechTarget

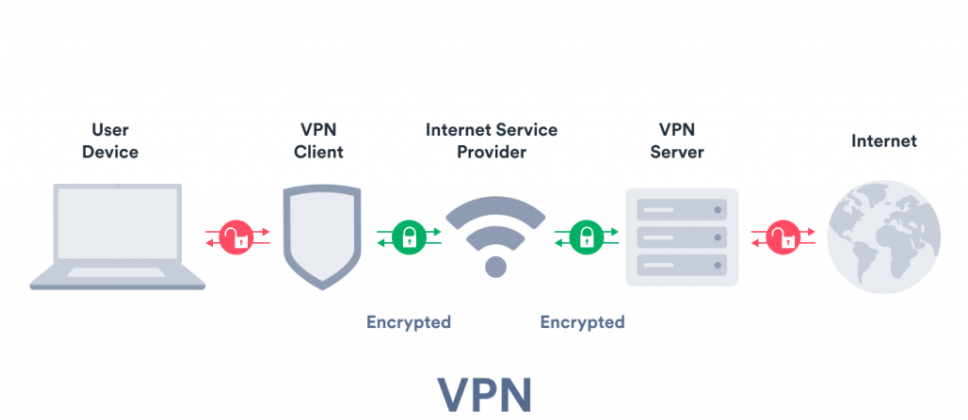
VLAN

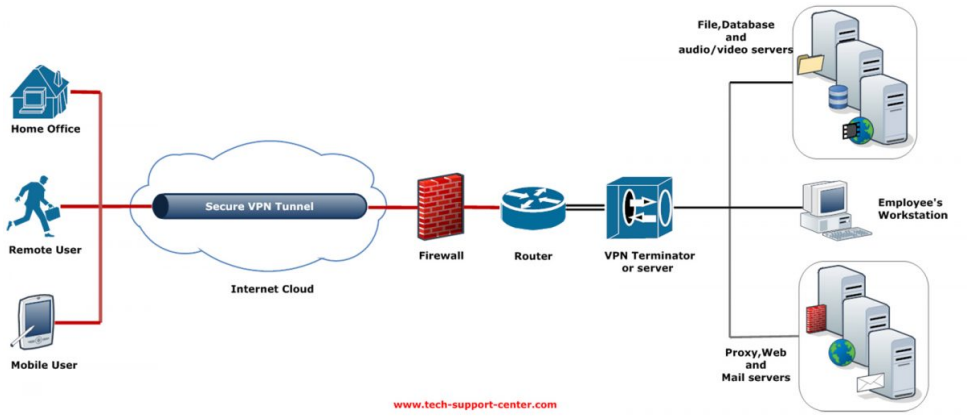

Sample network using VLANs
Router
Interface Fa 4.1
IP 192.168.2.254
VLAN 10
PC
IP 192.168.2.1 /24
Switch
Interface Fa 4.2
IP 192.168.3.254
VLAN 20
VLAN Trunk
PC
IP 192.168.3.1 /24

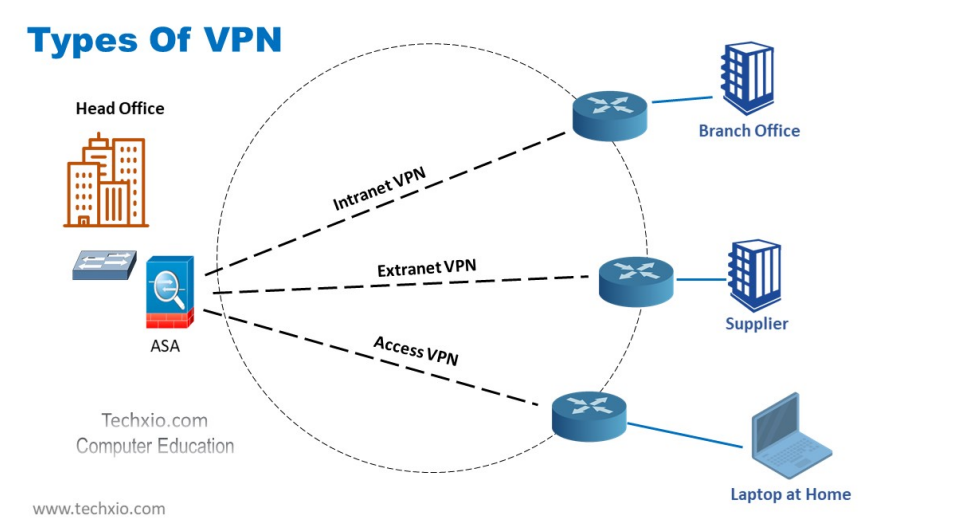
VPN


How VPN Works
Hacker
Government Firewall
Unsecure Connection
Internet Service Provider
Unsecure Connection
User
Internet
Secure Connection
VPN Tunnel
Secure Connection
VPN Tunnel
VPN Server

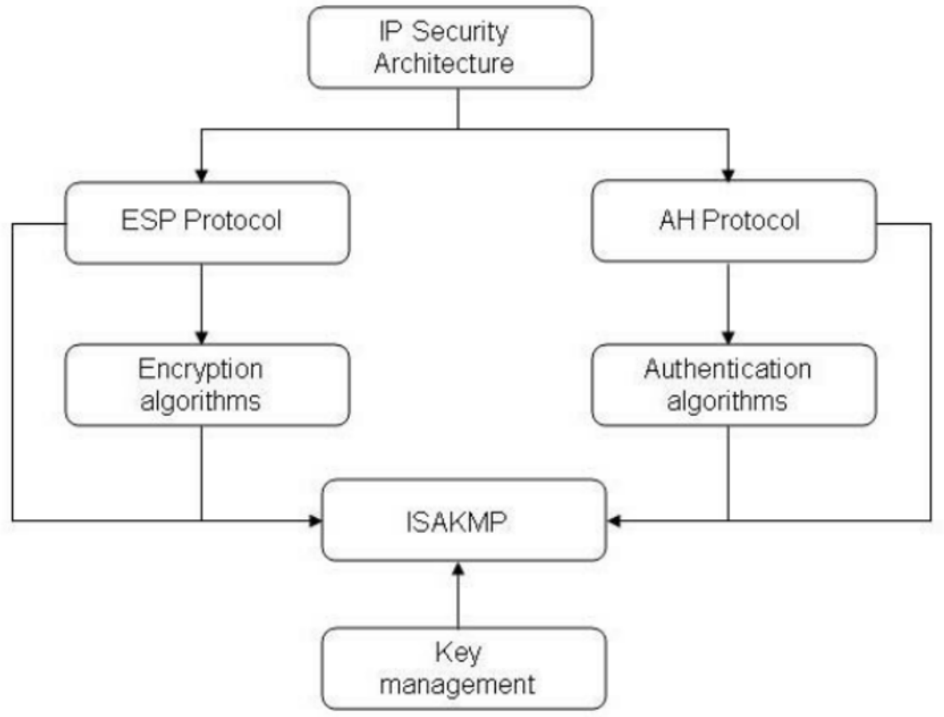

Internet Service Providers
Hackers
Governments
Encrypted tunnel
VPN
Using your computer abroad
Internet

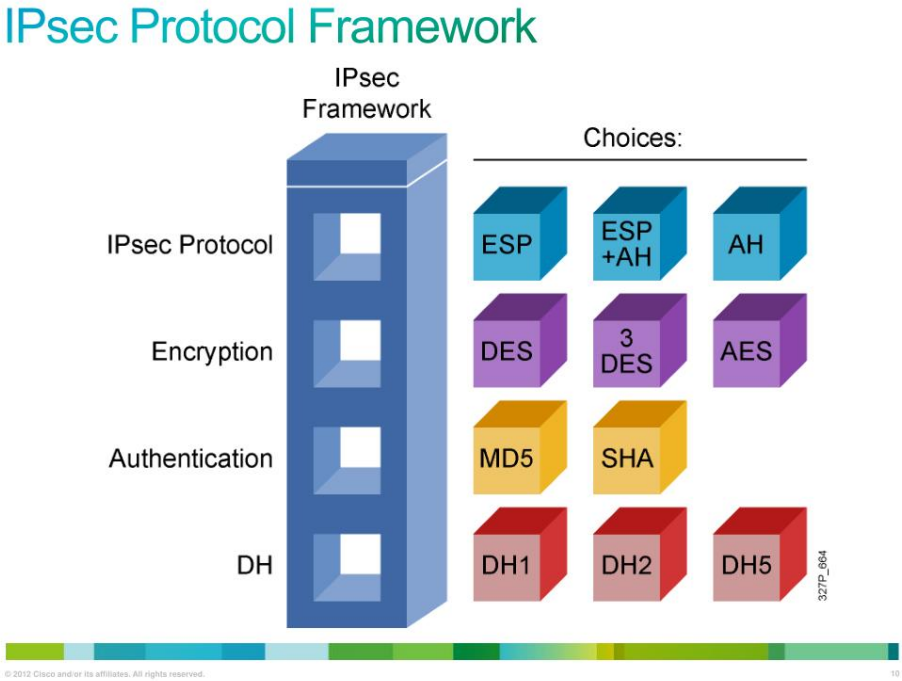

User Device
VPN Client
Internet Service Provider
VPN Server
Internet
Encrypted
Encrypted
VPN

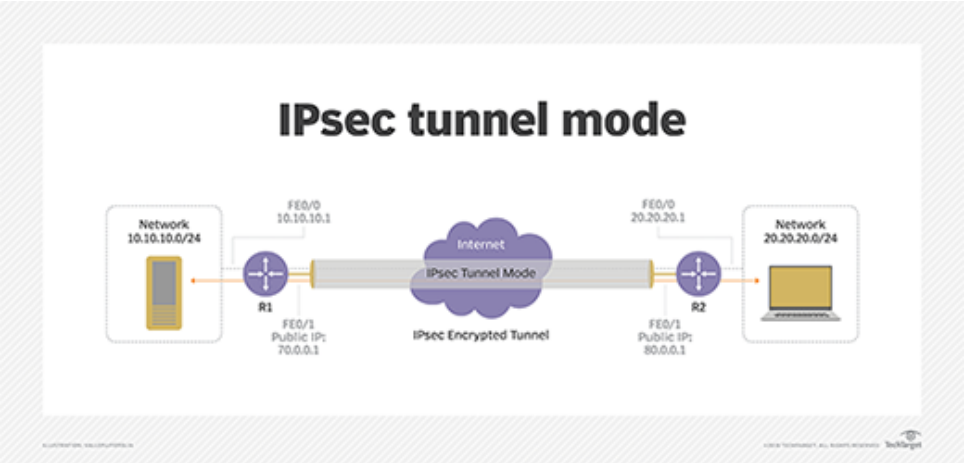

Home Office
Remote User
Mobile User
Secure VPN Tunnel
Internet Cloud
Firewall
Router
VPN Terminator or server
File,Database and audio/video servers
Employee's Workstation
Proxy,Web and Mail servers
www.tech-support-center.com


Types Of VPN
Head Office
ASA
Intranet VPN
Extranet VPN
Access VPN
Branch Office
Supplier
Laptop at Home
Techxio.com
Computer Education
www.techxio.com

IPSec


IP Security Architecture
ESP Protocol
AH Protocol
Encryption algorithms
Authentication algorithms
ISAKMP
Key management


IPsec Protocol Framework
IPsec Framework
Choices:
IPsec Protocol
Encryption
Authentication
DH
ESP
ESP +AH
AH
DES
3 DES
AES
MD5
SHA
DH1
DH2
DH5
327P_664
© 2012 Cisco and/or its affiliates. All rights reserved.
10

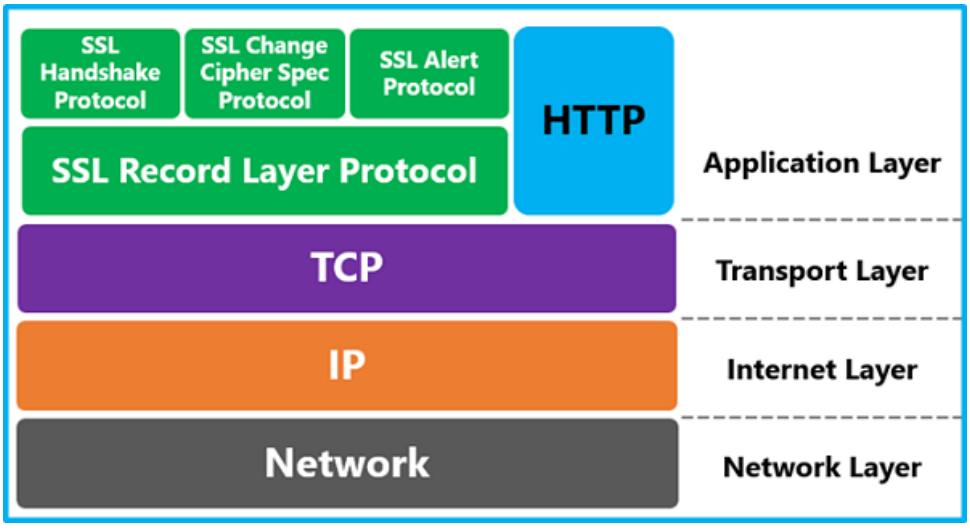


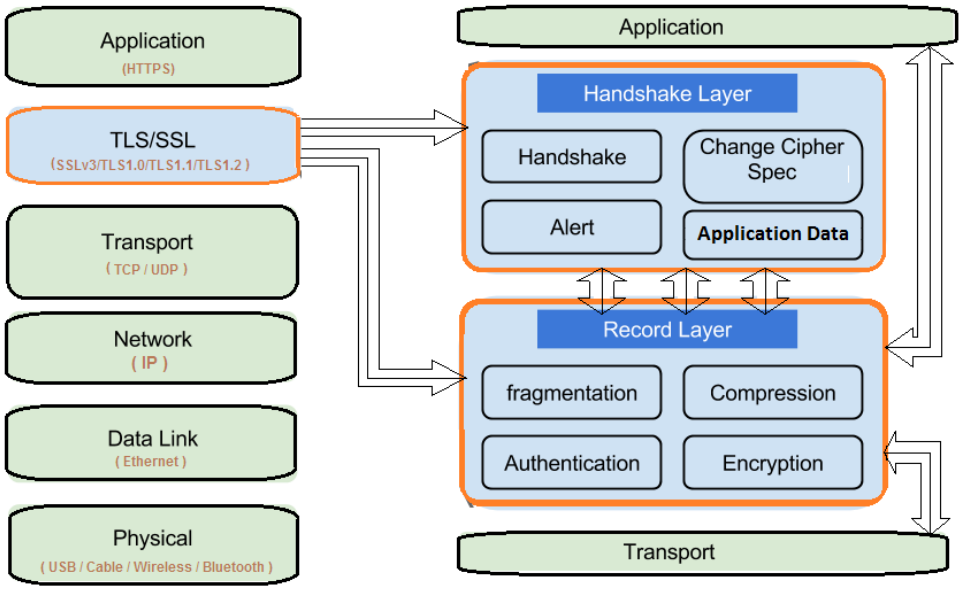
IPsec: Network Layer Security

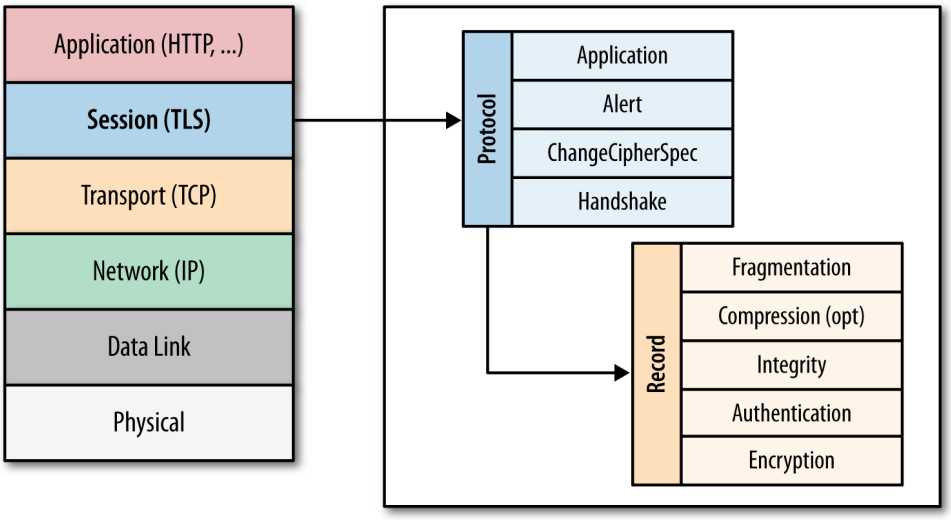
- network-layer secrecy:
  - sending host encrypts the data in IP datagram
  - TCP and UDP segments; ICMP and SNMP messages.
- network-layer authentication
  - destination host can authenticate source IP address
- two principal protocols:
  - authentication header (AH) protocol
  - encapsulation security 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

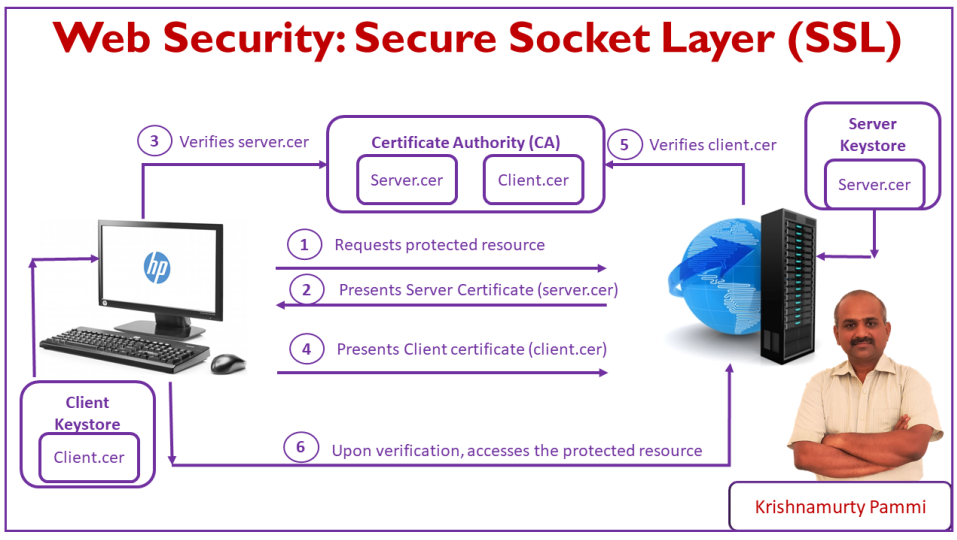
8: Network Security 8-1

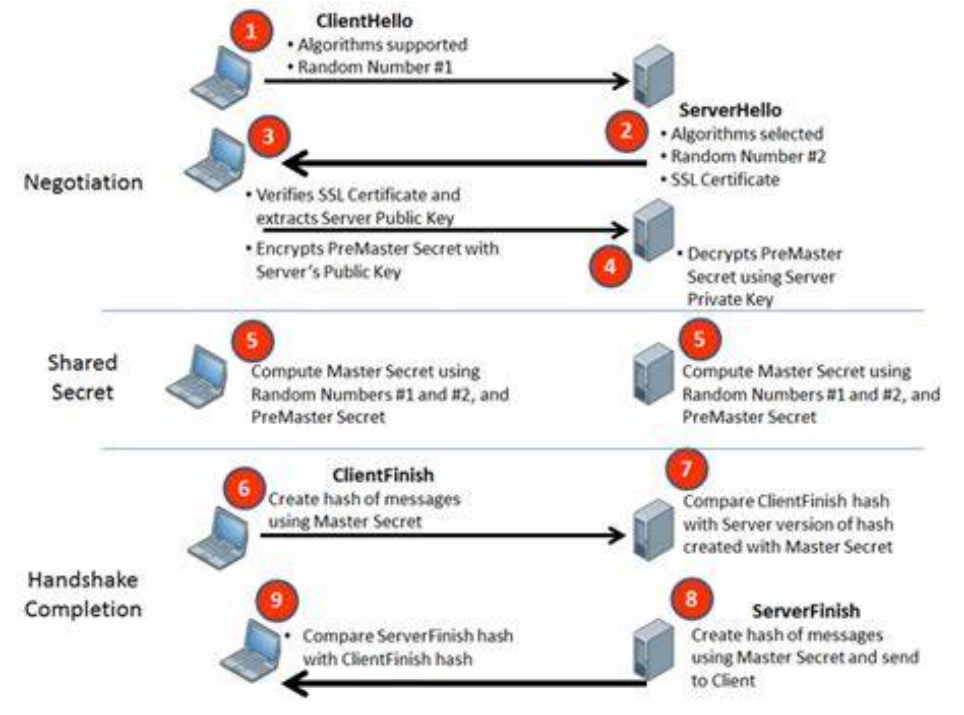
SSL(Secure Sockets Layers)

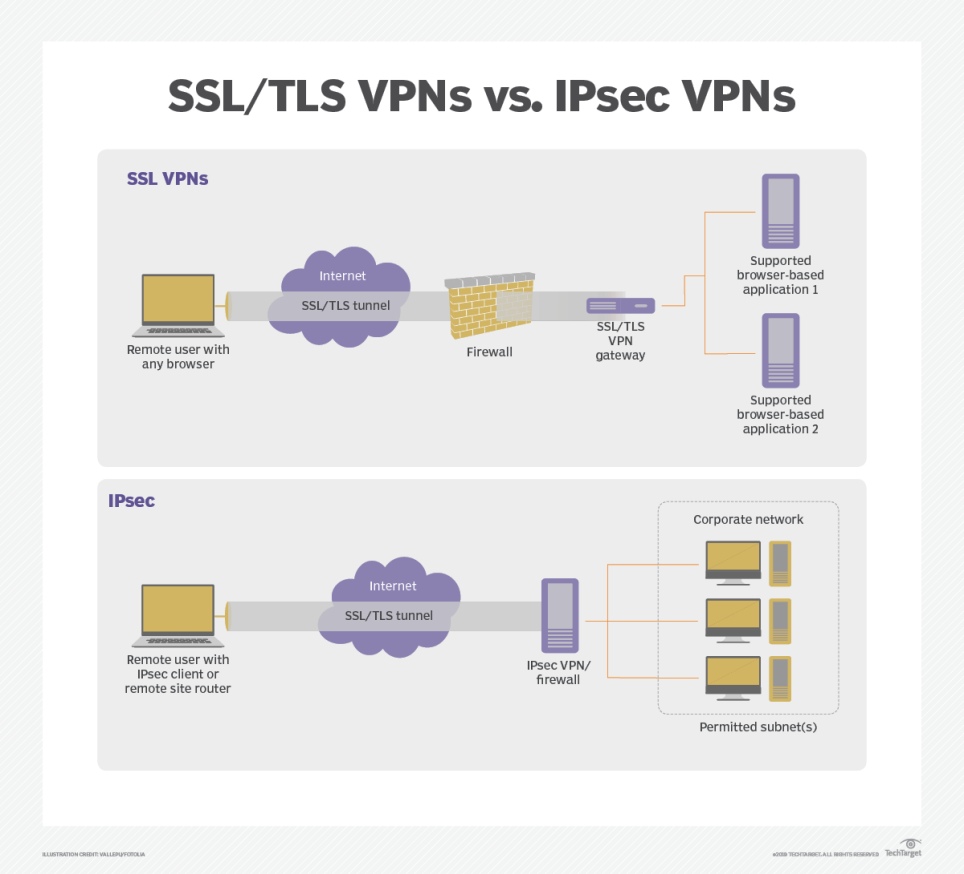


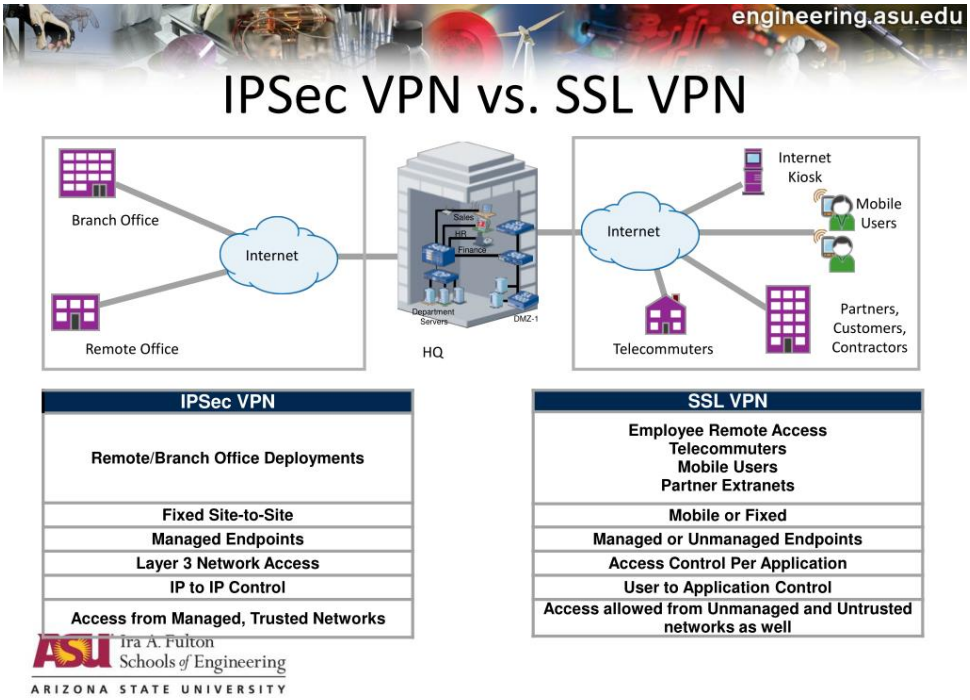

Application
(HTTPS)
TLS/SSL
(SSLv3/TLS1.0/TLS1.1/TLS1.2)
Transport
(TCP / UDP)
Network
(IP)
Data Link
(Ethernet)
Physical
(USB / Cable / Wireless / Bluetooth)
Application
Handshake Layer
Handshake
Change Cipher Spec
Alert
Application Data
Record Layer
fragmentation
Compression
Authentication
Encryption
Transport


Application (HTTP, ...)
Session (TLS)
Transport (TCP)
Network (IP)
Data Link
Physical
Protocol
Application
Alert
ChangeCipherSpec
Handshake
Record
Fragmentation
Compression (opt)
Integrity
Authentication
Encryption

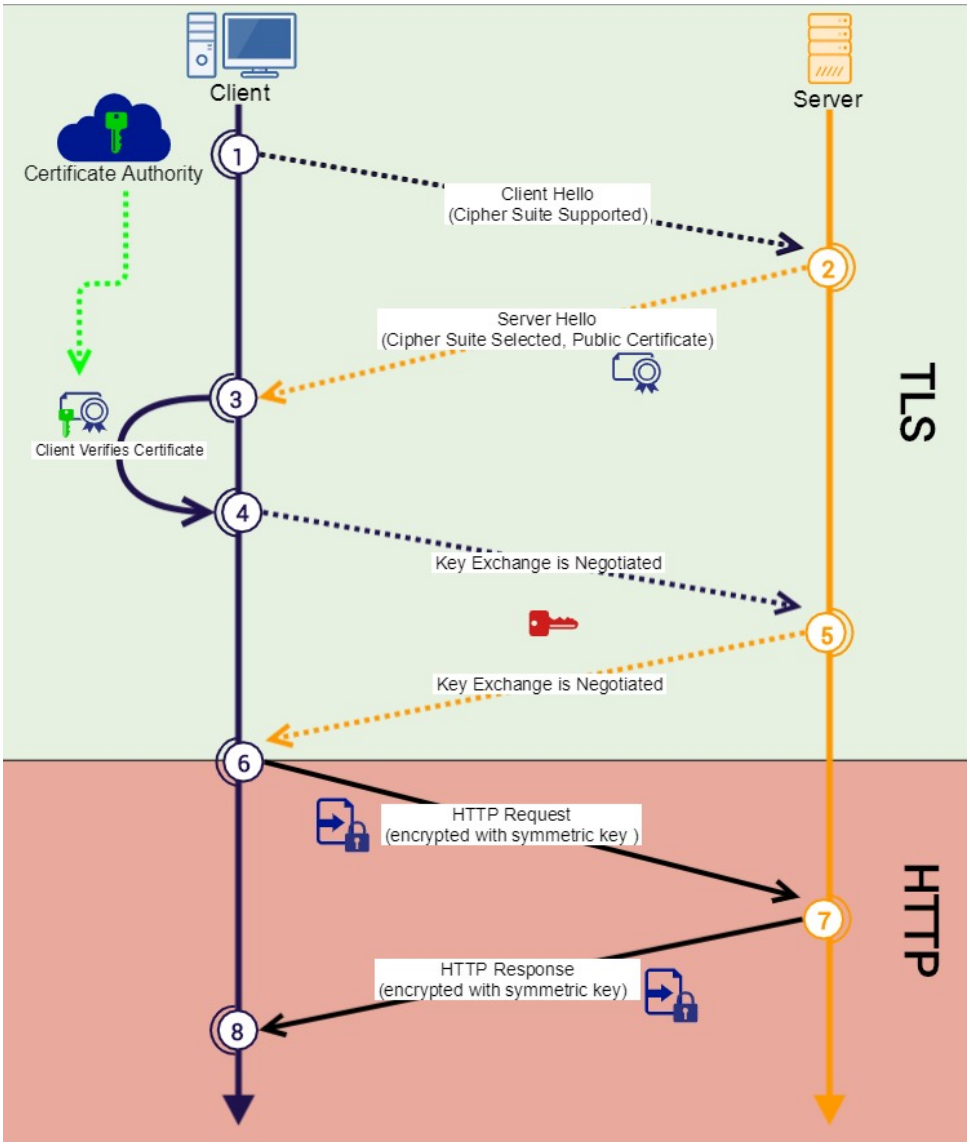

Web Security: Secure Socket Layer (SSL)
3 Verifies server.cer
Certificate Authority (CA)
Server.cer
Client.cer
5 Verifies client.cer
Server Keystore
Server.cer
1 Requests protected resource
2 Presents Server Certificate (server.cer)
4 Presents Client certificate (client.cer)
Client Keystore
Client.cer
6 Upon verification, accesses the protected resource
Krishnamurty Pammi


ClientHello
• Algorithms supported
• Random Number #1
ServerHello
• Algorithms selected
• Random Number #2
• SSL Certificate
Negotiation
• Verifies SSL Certificate and extracts Server Public Key
• Encrypts PreMaster Secret with Server's Public Key
• Decrypts PreMaster Secret using Server Private Key
Shared Secret
Compute Master Secret using Random Numbers #1 and #2, and PreMaster Secret
Compute Master Secret using Random Numbers #1 and #2, and PreMaster Secret
ClientFinish
Create hash of messages using Master Secret
Compare ClientFinish hash with Server version of hash created with Master Secret
Handshake Completion
Compare ServerFinish hash with ClientFinish hash
ServerFinish
Create hash of messages using Master Secret and send to Client

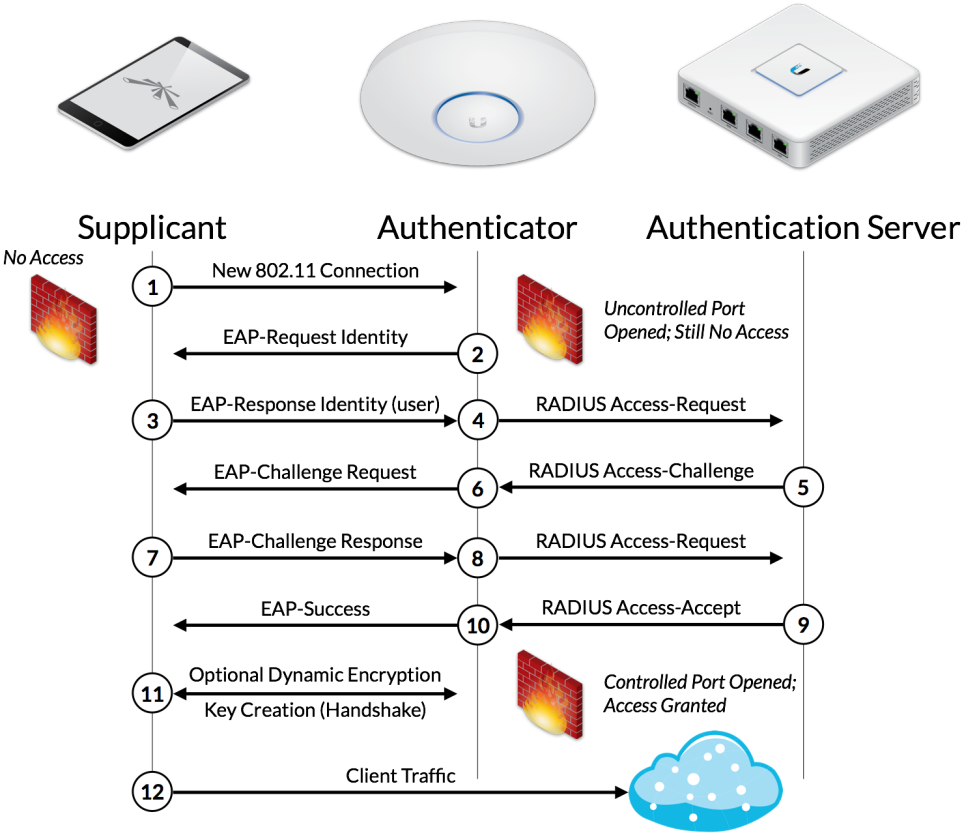

SSL/TLS VPNs vs. IPsec VPNs
SSL VPNs
Internet
SSL/TLS tunnel
Remote user with any browser
Firewall
SSL/TLS VPN gateway
Supported browser-based application 1
Supported browser-based application 2
IPsec
Internet
SSL/TLS tunnel
Remote user with IPsec client or remote site router
IPsec VPN/ firewall
Corporate network
Permitted subnet(s)

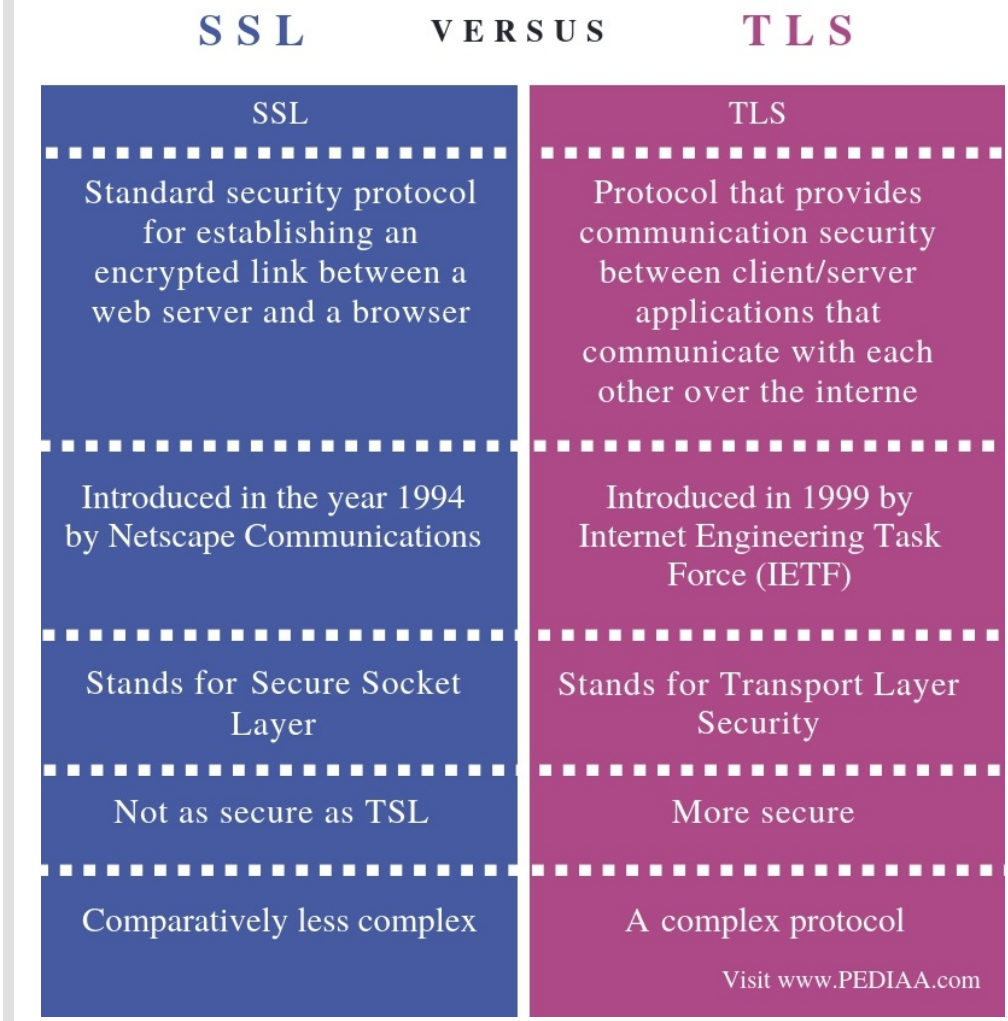

engineering.asu.edu
IPSec VPN vs. SSL VPN
Branch Office
Internet
Remote Office
HQ
Internet
Internet Kiosk
Mobile Users
Telecommuters
Partners, Customers, Contractors
IPSec VPN
Remote/Branch Office Deployments
Fixed Site-to-Site
Managed Endpoints
Layer 3 Network Access
IP to IP Control
Access from Managed, Trusted Networks
SSL VPN
Employee Remote Access
Telecommuters
Mobile Users
Partner Extranets
Mobile or Fixed
Managed or Unmanaged Endpoints
Access Control Per Application
User to Application Control
Access allowed from Unmanaged and Untrusted networks as well
Ira A. Fulton Schools of Engineering
ARIZONA STATE UNIVERSITY

TLS

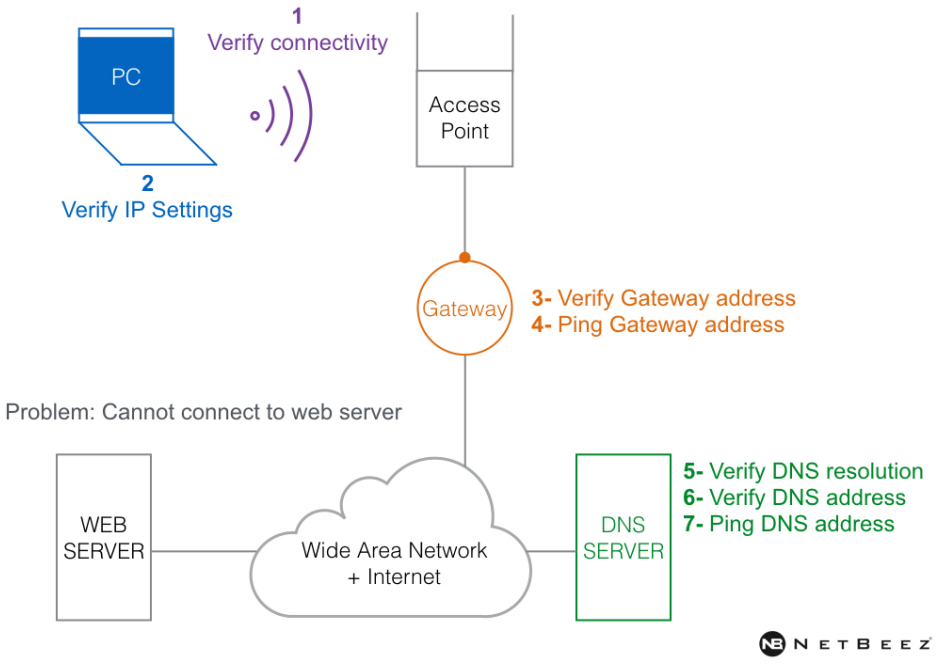

Client
Server
Certificate Authority
1
Client Hello
(Cipher Suite Supported)
2
Server Hello
(Cipher Suite Selected, Public Certificate)
3
Client Verifies Certificate
4
Key Exchange is Negotiated
5
Key Exchange is Negotiated
6
HTTP Request
(encrypted with symmetric key)
7
HTTP Response
(encrypted with symmetric key)
8
TLS
HTTP

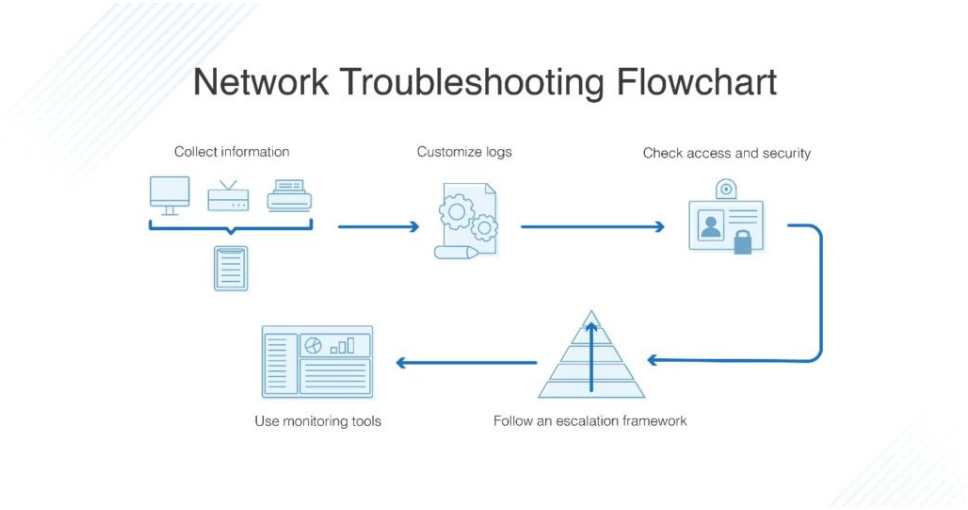
802.1X Authentication (EAP & RADIUS)

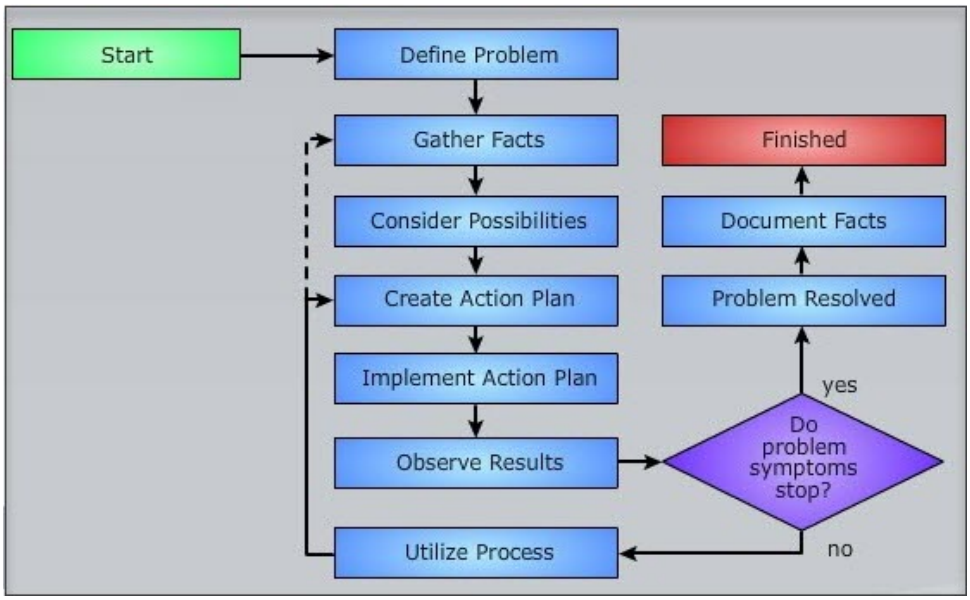


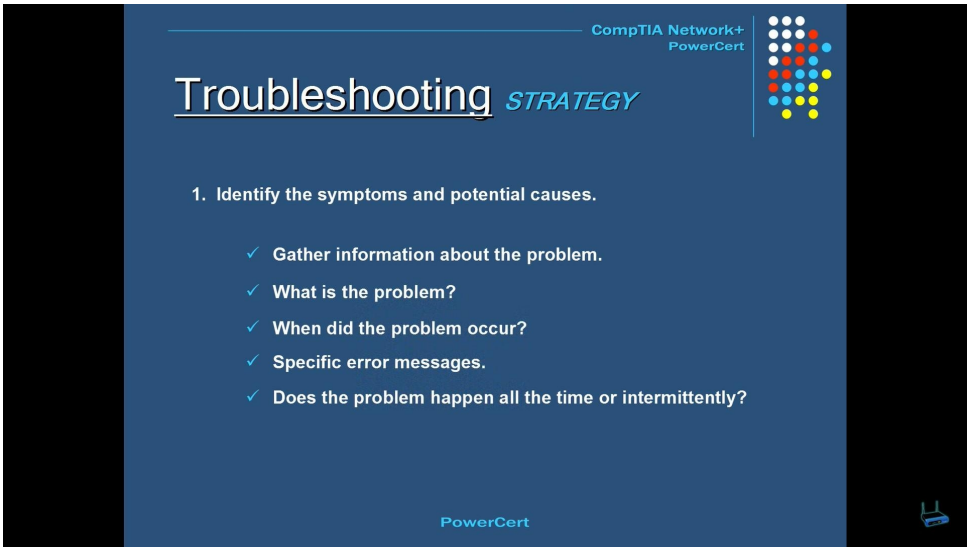


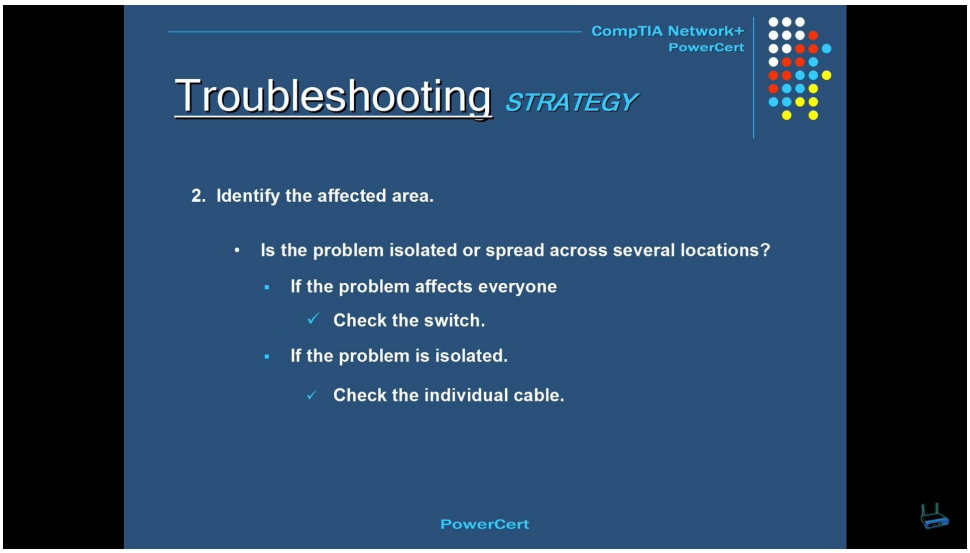
Network Troubleshooting

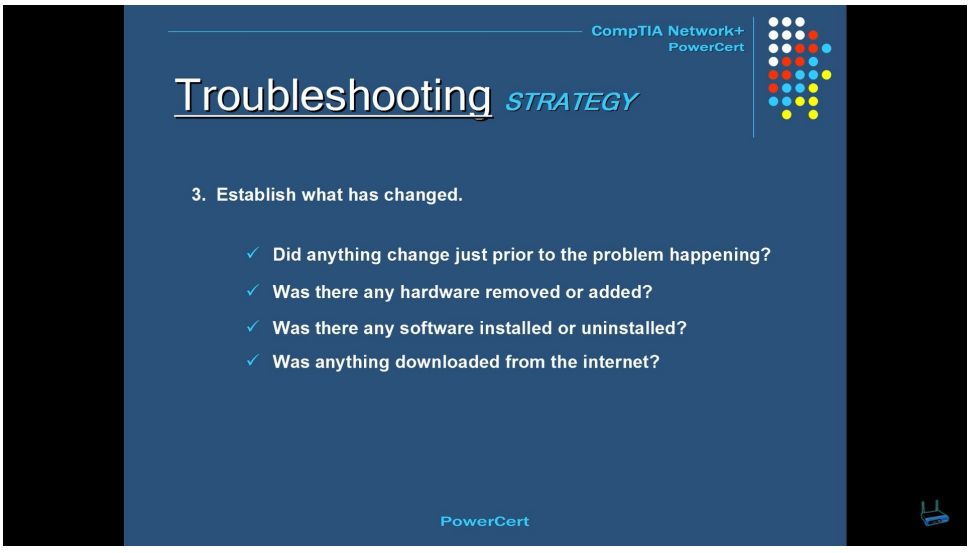


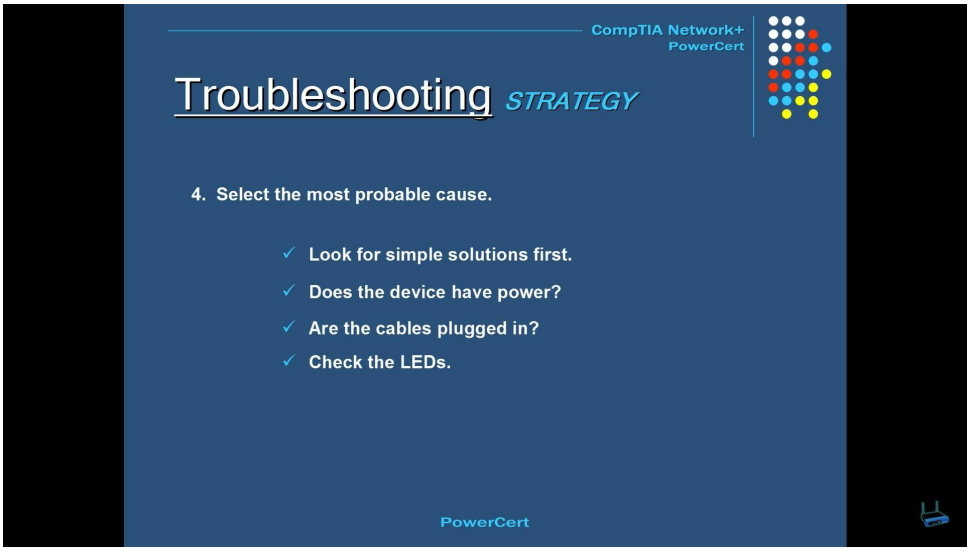

Network Troubleshooting Flowchart
Collect information
Customize logs
Check access and security
Use monitoring tools
Follow an escalation framework

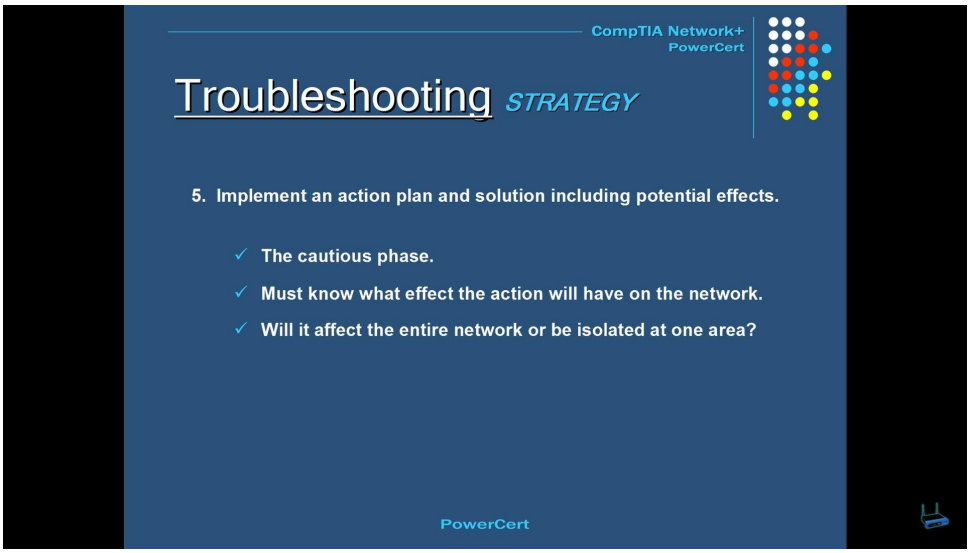

Start
Define Problem
Gather Facts
Consider Possibilities
Create Action Plan
Implement Action Plan
Observe Results
Utilize Process
Do problem symptoms stop?
yes
no
Problem Resolved
Document Facts
Finished

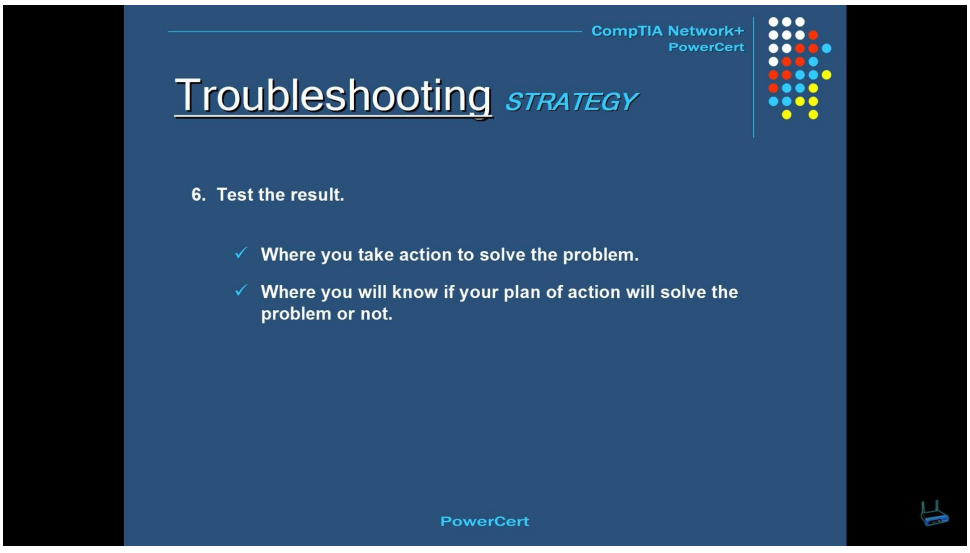

CompTIA Network+
PowerCert
Troubleshooting STRATEGY
1. Identify the symptoms and potential causes.
✓ Gather information about the problem.
✓ What is the problem?
✓ When did the problem occur?
✓ Specific error messages.
✓ Does the problem happen all the time or intermittently?
PowerCert

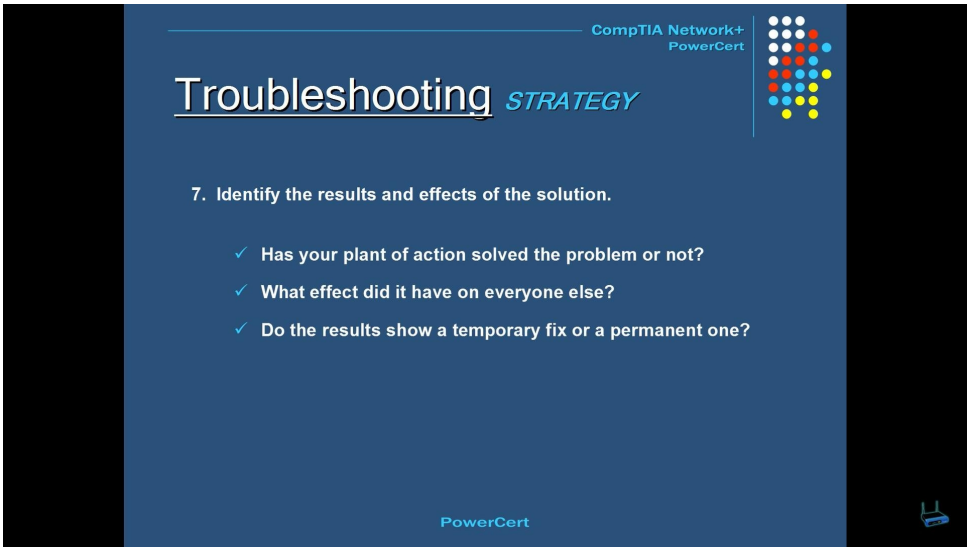

CompTIA Network+
PowerCert
Troubleshooting STRATEGY
2. Identify the affected area.
• Is the problem isolated or spread across several locations?
▪ If the problem affects everyone
✓ Check the switch.
▪ If the problem is isolated.
✓ Check the individual cable.
PowerCert

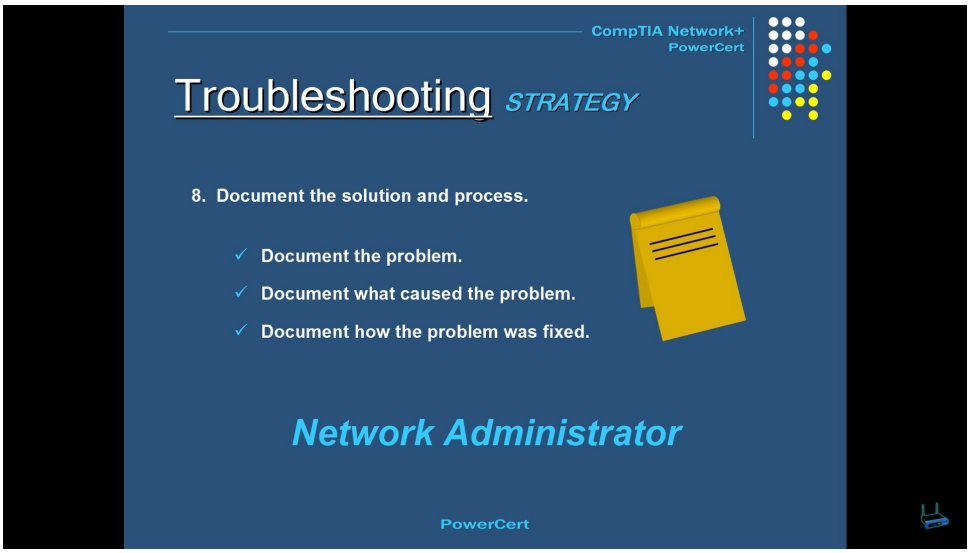

CompTIA Network+
PowerCert
Troubleshooting STRATEGY
3. Establish what has changed.
✓ Did anything change just prior to the problem happening?
✓ Was there any hardware removed or added?
✓ Was there any software installed or uninstalled?
✓ Was anything downloaded from the internet?
PowerCert


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


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


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


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

-- Memo End --