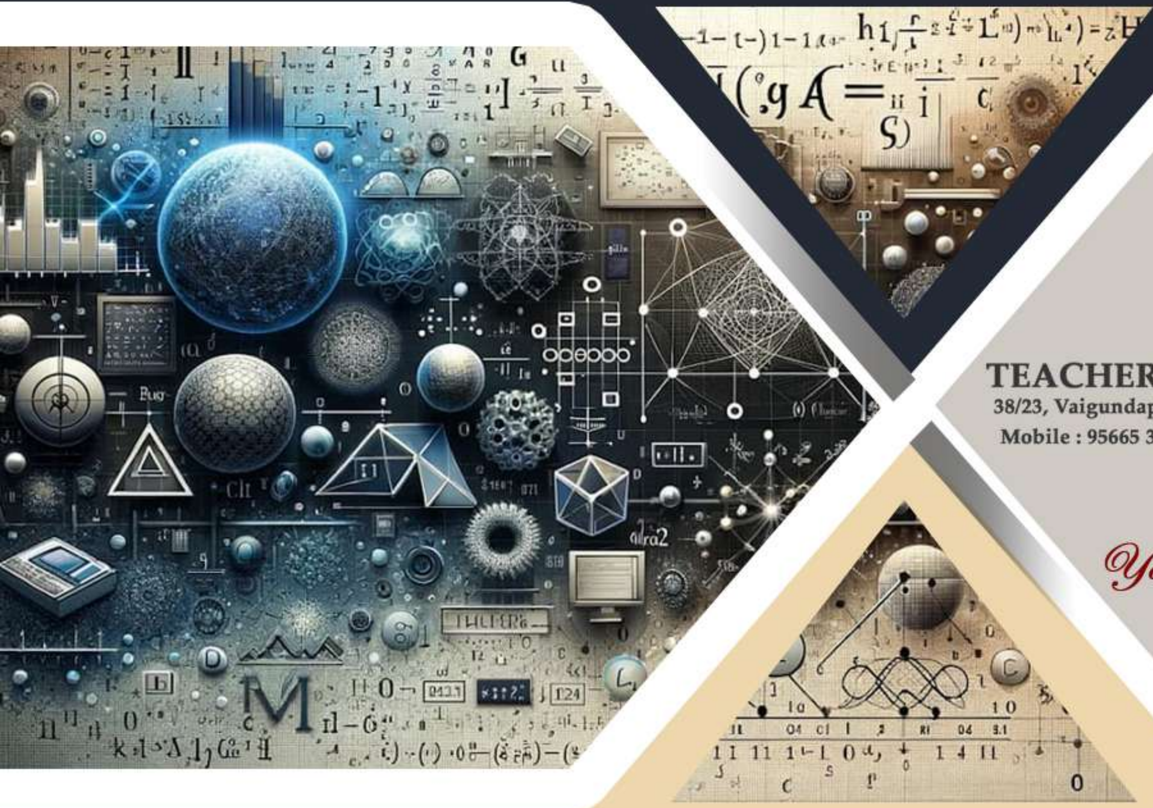
COLLEGE TRB
Computer Science

2024-2025



TEACHER'S CARE PUBLICATION
38/23, Vaigundaperumal Koil Street,Kancheepuram- 631502
Mobile : 95665 35080, 9786269980 Land Line : 044-2723 5080

Your Success is Our Goal...

UNIT-1
Discrete Structures and Optimization

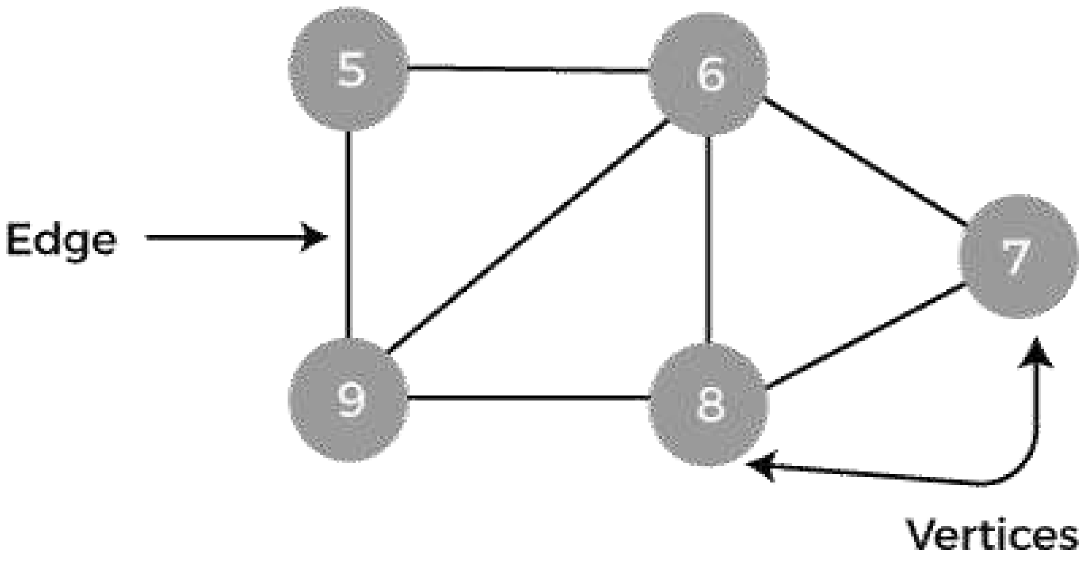
TEACHER'S CARE ACADEMY

KANCHIPURAM

COMPUTER SCIENCE

UNIT – I

DISCRETE STRUCTURES & OPTIMIZATION



COMPETITIVE EXAM

FOR

ASSISTANT PROFESSOR TRB

2024 - 25

Computer Science Unit I - Discrete Structures and Optimization

Teacher's Care Academy

Syllabus

Mathematical Logic: Propositional and Predicate Logic, Propositional Equivalences, Normal Forms, Predicates and Quantifiers, Nested Quantifiers, Rules of inference.

Sets and Relations: Set Operations, Representation and Properties of Relations, Equivalence Relations, Partial Ordering.

Counting, Mathematical Induction and Discrete Probability: Basics of Counting, Pigeonhole Principle, Permutations and Combinations, Inclusion-Exclusion Principle, Mathematical Induction, Probability, Baye's Theorem.

Group Theory: Groups, Subgroups, Semi Groups, Product and Quotients of Algebraic Structures, Isomorphism, Homomorphism, Automorphism, Rings, Integral Domains, Fields, Applications of Group Theory.

Graph Theory: Simple Graph, Multigraph, Weighted Graph, Paths and Circuits, Shortest Paths in Weighted Graphs, Eulerian Paths and Circuits, Hamiltonian Paths and Circuits, Planar graph, Graph Coloring, Bipartite Graphs, Trees and Rooted Trees, Prefix Codes, Tree Traversals, Spanning Trees and Cut-Sets.

Boolean Algebra: Boolean Functions and its Representation, Simplifications of Boolean Functions.

Optimization: Linear Programming - Mathematical Model, Graphical Solution, Simplex and Dual Simplex Method, Sensitivity Analysis; Integer Programming, Transportation and Assignment Models, PERT - CPM: Diagram Representation, Critical Path Calculations, Resource Levelling, Cost Consideration in Project Scheduling.

Contents

Chapter 1

Mathematical Logic

Logic is the science dealing with the methods of learning. Logic which uses a symbolic language to express its principles in precise and unambiguous terms is known as mathematical logic. One reason for this is that all efforts at the verification of algorithms inevitably the notation and methods of logic. Logic, among other things, have provided the theoretical basis for many areas of computer science such as digital logic design, automata theory and computability, and artificial intelligence etc. One component of logic is *proposition calculus* which deals with statements with values true and false and is concerned with analysis of propositions. And the other part is *predicate calculus* which deals with the predicates which are propositions containing variables.

1.1 Propositions

A number of words making a complete grammmatical structure having a sense and meaning and also meant an assertion in logic or mathematics is called a *sentence*. This assertion may be of two types - declarative and non-declarative. A *proposition* or *statement* is a declarative sentence that is either true or false.

For example, “Three plus three equals six” and “Three plus three equals seven” are both statements, the first because it is true and the second because it is false. Similarly “$x+y>1$” is not a statement because for some values of x and y the sentence is true, whereas for others it is false. For instance, if $x=1$ and $y=2$, the sentence is true, if $x=-3$ and $y=1$, this is false. The truth or falsity of a statement is called its *truth value*. Since only two possible truth values are admitted this logic is sometimes called *two-valued logic*. Questions, exclamations and commands are not propositions.
It is customary to represent simple statements by letters $p, q, r, \cdots$ known as *proposition variables*. Propositional variables can only assume two values, true or false. There are also two *propositional constants*, T and F, that represent true or false, respectively. If p denotes the proposition “The sun sets in the east”, then instead of saying “The sun sets in the east” is false, one can simply say the value of p is F.

(a)	The sun rises in the west.	A false statement.
(b)	$2 + 4 = 6$.	A true statement.
(c)	Do you speak Hindi?	It is a question and not a declarative statement. Hence it is not a statement.
(d)	Close the door.	It is a command. Hence it is not a statement.
(e)	What a hot day!	It is an exclamation. Hence it is not a statement.
(f)	We shall have chicken for dinner.	It is a statement since it is either true or false but not both, although one has to wait until dinner to find out if it is true or false.

1.2 Compound Proposition

A proposition consisiting of only a single propositional variable or a single propositional constant is called an *atomic*(primary, primitive) proposition or simply proposition; that is, they can not be further subdivided. A proposition obtained from the combinations of two or more propositions by means of logical operators or connectives of two or more propositions or by negating a single proposition is referred to *molecular or composite or compound proposition*.

Connectives

The words or phrases (or symbols) used to form compound propositions are called connectives. There are five basic connectives called

- Negation,
- Conjunction,
- Disjunction,
- Implication or Conditional, and
- Equivalence or Biconditional.

***Example* 1.2.1:** Let p: Paris is in France. Then the negation of p is the statement

$$\sim p: \text{ It is not the case that Paris is in France.}$$

Symbol used	Connective word	Nature of the compound statement formed by the connective	Symbolic form	Negation
$\sim, \neg, N$	not	Negation	$\sim p$	$\sim(\sim p) = p$
$\wedge$	and	Conjunction	$p \wedge q$	$(\sim p) \vee (\sim q)$
$\vee$	or	Disjunction	$p \vee q$	$(\sim p) \wedge (\sim q)$
$\Rightarrow, \rightarrow$	if, $\cdots$ then	Implication (or) Conditional	$p \Rightarrow q$	$p \wedge (\sim q)$
$\Leftrightarrow, <->$	if and only if	Equivalence (or) Bi-conditional	$p \Leftrightarrow q$	$[p \wedge (\sim q)] \vee [q \wedge (\sim p)]$

Normally it is written as

$$\sim p: \text{ Paris is not in France.}$$

Statement p	$\sim p$
All students are intelligent.	Some students are not intelligent. There exists a student who is not intelligent. At least one student is not intelligent.
No student is intelligent.	Some students are intelligent.

Remark 1.2.1. It is to be noted that "not" is not a connective, since it does not join two statements and $\sim p$ is not really a compound statement. However, "not" is a unary operation for the collection of statements, and $\sim p$ is a statement if p is considered a statement.

***Example* 1.2.2:** From the conjunction of p and q for each of the following.

1. p: Ram is healthy, q: He has blue eyes.
2. p: It is cold, q: It is raining.
3. p: $5x + 6 = 26$, q: $x > 3$.

Solution.

1. $p \wedge q$: Ram is healthy and he has blue eyes.

2. $p \wedge q$: It is cold and raining.

3. $p \wedge q$: $5x + 6 = 26$ and $x > 3$.

***Example* 1.2.3:** Assign a truth value to each of the following statements.

$5 < 5 \vee 5 < 6$	True, since one of its components *viz.* $5 < 6$ is true.
$5 \times 4 = 21 \vee 9 + 7 = 17$	False, since both of its components are false.
$6 + 4 = 10 \vee 0 > 2$	True, one of its components *viz.* $6 + 4 = 10$ is true.

1.3 Proposition and Truth Tables

p	q	$p \wedge q$		p	q	$p \vee q$		p	$\sim p$
T	T	T		T	T	T		T	F
T	F	F		T	F	T		F	T
F	T	F		F	T	T			
F	F	F		F	F	F			

Table 1.1: Truth tables for the three propositional connectives

***Example* 1.3.1:** Construct a truth table for each compound proposition.

1. $p \wedge (\sim q \vee q)$

2. $\sim (p \vee q) \vee (\sim p \wedge \sim q)$

Solution.

p	q	$\sim q$	$\sim q \vee q$	$p \wedge (\sim q \wedge q)$
T	T	F	T	T
T	F	T	T	T
F	T	F	T	F
F	F	T	T	F

Table 1.2: 1. Truth table for $p \wedge (\sim q \vee q)$

p	q	$\sim p$	$\sim q$	$p \wedge q$	$\sim (p \wedge q)$	$(\sim p \wedge \sim q)$	$\sim (p \vee q) \vee (\sim p \wedge \sim q)$
T	T	F	F	T	F	F	F
T	F	F	T	T	F	F	F
F	T	T	F	T	F	F	F
F	F	T	T	F	T	T	T

Table 1.3: 2. Truth table for $\sim (p \vee q) \vee (\sim p \wedge \sim q)$

Definition 1.3.1. If two propositions $P(p, q, \cdots)$ and $Q(p, q, \cdots)$ where $p, q, \cdots$ are propositional variables have the same truth values in every possible case, the propositions are called logically equivalent or simply equivalent, and denoted by

$$P(p, q, \cdots) \equiv Q(p, q, \cdots).$$

It is always permissible, and sometimes desiarable to replace a given proposition by an equivalent one.

1.3.1 Algebra of Propositions

1. Idempotent laws.
   (a) $p \vee p \equiv p$,
   (b) $p \wedge p \equiv p$.
2. Associative laws.
   (a) $(p \vee q) \vee r \equiv p \vee (q \vee r)$,
   (b) $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$.
3. Commutative laws.
   (a) $p \vee q \equiv q \vee p$,
   (b) $p \wedge q \equiv q \wedge p$.
4. Distributive laws.
   (a) $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$,
   (b) $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$.

5. Identity laws.
   (a) $p\vee$ F $\equiv p$,
   (b) $p\vee$ T $\equiv$ T,
   (c) $p\wedge$ T $\equiv p$,
   (d) $p\wedge$ F $\equiv$ F.
6. Complement laws.
   (a) $p\vee \sim p \equiv$ T,
   (b) $p\wedge \sim p \equiv$ F,
   (c) $\sim$ T $\equiv$ F,
   (d) $\sim$ F $\equiv$ T.
7. Involution law. $\sim(\sim p) \equiv p$.
8. DeMorgan's laws.
   (a) $\sim(p \vee q) \equiv \sim p\wedge \sim q$,
   (b) $\sim(p \wedge q) \equiv \sim p\vee \sim q$.
9. Absorption laws.
   (a) $p \vee (p \wedge q) \equiv p$,
   (b) $p \wedge (p \vee q) \equiv p$.

1.3.2 Conditional Proposition

If p and q are proposition, the compound proposition "if p then q" denoted by $p \Rightarrow q$ is called a *conditional proposition* or *implication* and the connective is the *conditional connective*. The proposition p is called *antecedent* or *hypothesis*, and the proposition q is called the *consequent* or *conclusion*.

***Example* 1.3.2:** 1. If tomorrow is Sunday, then today is Saturday.

Here p: "Tomorrow is Sunday" is antecedent, q: "Today is Saturday" is consequent.

2. If it rains, hten I will carry an umbrella.

Here p: "It rains" is antecedent, q: "I will carry an umbrella" is consequent.

The connective if $\cdots$ then can also be read as follows.

1. p implies q.
2. p is sufficient for q.

3. p only if q.
4. q is necessary for p.
5. q if p.
6. q follows from p.
7. q is a consequence of p.

The truth table for implication is given in the following table.

p	q	$p \Rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Table 1.4: Truth table for the implication $p \Rightarrow q$.

***Example* 1.3.3:** Which of the following propositions are true and which are false?

(a) If the earth is round then the earth travels around the sun.

(b) If Alexander Graham Bell invented telephone, then tigers have wings.

(c) If tigers have wings, then RDX is dangerous.

Solution.

(a) True. Let p: The earth is round, q: The earth travels round the sun. Here p is true and q is true and hence the conditional proposition is true.

(b) False. Let p: Alexander Graham Bell invented telephone, q: Tigers have wings. Here p is true and q is false and hence the conditional proposition is false.

(c) True. Let p: Tigers have wings, q: RDX is dangerous. Here p is false and q is true and hence the conditional proposition is true.

***Example* 1.3.4:** Construct truth table for

1. $p \vee \sim q \Rightarrow p$,
2. $(\sim (p \wedge q) \vee r) \Rightarrow \sim p$.

Solution.

p	q	$\sim q$	$p\vee \sim q$	$p\vee \sim q \Rightarrow p$
T	T	F	T	T
T	F	T	T	T
F	T	F	F	T
F	F	T	T	F

p	q	r	$(p \wedge q)$	$\sim (p \wedge q)$	$(\sim (p \wedge q) \vee r)$	$(\sim (p \wedge q) \vee r) \Rightarrow \sim p$
T	T	T	T	F	T	F
T	T	F	T	F	F	T
T	F	T	F	T	T	F
T	F	F	F	T	T	F
F	T	T	F	T	T	T
F	T	F	F	T	T	T
F	F	T	F	T	T	T
F	F	F	F	T	T	T

1.3.3 Converse, Contrapositive and Inverse

There are some related implication that can be formed from $p \Rightarrow q$. If $p \Rightarrow q$ is an implication, then the converse of $p \Rightarrow q$ is the implication $q \Rightarrow p$, the contrapositive of $p \Rightarrow q$ is the implication $\sim q \Rightarrow \sim p$ and the inverse of $p \Rightarrow q$ is $\sim p \Rightarrow \sim q$.

The truth table of the four propositions follow:

		Conditional	Converse	Inverse	Contrapositive
p	q	$p \Rightarrow q$	$q \Rightarrow p$	$\sim p \Rightarrow \sim q$	$\sim q \Rightarrow \sim p$
T	T	T	T	T	T
T	F	F	T	T	F
F	T	T	F	F	T
F	F	T	T	T	T

***Problem* 1.3.1:** Prove that if x^2 is divisible by 4, then x is even.

Proof. Let p and q be the propositions such that

$$p : x^2 \text{ is divisible by } 4 \quad \text{and} \quad q : x \text{ is even.}$$

The implication is of the form $p \Rightarrow q$. The contrapositive is $\sim q \Rightarrow \sim p$, which states in words:

If x is odd, then x^2 is not divisible by 4.

The proof of contrapositive is easy. Since x is odd, one can write $x = 2k+1$, for some integer k. Hence

$$x^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 4(k^2 + k + 1/4).$$

Since $k^2 + k$ is an integer, $k^2 + k + 1/4$ is not an integer; therefore x^2 is not divisible by 4. □

1.3.4 Biconditional Statement

If p and q are statements, then the compound statement p if and only if q, denoted by $p \Leftrightarrow q$ is called a biconditional statement and the connective if and only if is the biconditional connective. The biconditional statement $p \Leftrightarrow q$ can also be stated as "p is a necessary and sufficient condition for q" or as "p implies q and q implies p".

p	q	$p \Leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

1.3.5 Negation of Compound Statements

Negation of Conjunction: A conjunction $p \wedge q$ consists of two sub-statements p and q both of which exist simultaneously. Therefore, the negation of the conjunction would mean the negation of at least one of the two sub-statements. Thus, we have "the negation of a conjunction $p \wedge q$ is the disjunction of the negation of p and the negation of q". Equivalently, we write

$$\sim (p \wedge q) \equiv \sim p \vee \sim q.$$

In order to prove the above equivalence, we prepare the following table.

Negation of Disjunction: A disjunction $p \wedge q$ consists of two sub-statements p and q which are such that either p or q or both exist. Therefore, the negation of the disjunction would mean the negation of both p and q simultaneously.

p	q	$p \wedge q$	$\sim (p \wedge q)$	$\sim p$	$\sim q$	$\sim p \vee \sim q$
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

The negation of a disjunction $p \wedge q$ is the conjunction of the negation of p and the negation of q. Equivalently, we write

$$\sim (p \vee q) \equiv \sim p \wedge \sim q.$$

In order to prove the above equivalence, we prepare the following table.

p	q	$\sim p$	$\sim q$	$\sim p \wedge \sim q$	$p \wedge q$	$\sim (p \wedge q)$
T	T	F	F	F	T	F
T	F	F	T	F	T	F
F	T	T	F	F	T	F
F	F	T	T	T	F	T

Negation of Implication: If p and q are two statements, then

$$\sim (p \Rightarrow q) \equiv p \wedge \sim q.$$

In order to prove the above equivalence, we prepare the following table.

p	q	$p \Rightarrow q$	$\sim (p \Rightarrow q)$	$\sim q$	$p \wedge \sim q$
T	T	T	F	F	F
T	F	F	T	T	T
F	T	T	F	F	F
F	F	T	F	T	F

Negation of Biconditional: If p and q are two statements, then

$$\sim (p \Leftrightarrow q) \equiv p \Leftrightarrow \sim q \equiv \sim p \Leftrightarrow q.$$

In order to prove the above equivalence, we prepare the following table.

Derived Connectives

1. **NAND**: It means negation of conjunction of two statements. Assume p and q be two propositions. NAND of p and q is a proposition which is false when both p and q are true

p	q	$p \Leftrightarrow q$	$\sim (p \Leftrightarrow q)$	$\sim p$	$\sim p \Leftrightarrow q$	$\sim q$	$p \Leftrightarrow \sim q$
T	T	T	F	F	F	F	F
T	F	F	T	F	T	T	T
F	T	F	T	T	T	F	T
F	F	T	F	T	F	T	F

p	q	$p \uparrow q$
T	T	F
T	F	T
F	T	T
F	F	T

Table 1.5: Truth table for NAND

otherwise true. It is denoted by $p \uparrow q$ and $q \uparrow p \equiv \sim (p \wedge q)$.

2. **NOR**: It means negation of disjunction of two statements. Assume p and q be two propositions. Nor p and q is a proposition which is true when both p and q are false, otherwise false. It is denoted by $p \downarrow q$ and $p \downarrow q \equiv \sim (p \wedge q)$.

p	q	$p \downarrow q$
T	T	F
T	F	F
F	T	F
F	F	T

Table 1.6: Truth table for NOR

Note that

(i) $\sim p \equiv p \downarrow p$

(ii) $p \wedge q \equiv (p \downarrow p) \downarrow (q \downarrow q)$

(iii) $p \vee q \equiv (p \downarrow q) \downarrow (p \downarrow q)$

3. **XOR (Exclusive OR)**: Assume p and q be two proposition. The exclusive OR (XOR) of p and q, denoted by $p \oplus q$ is the proposition that is true when exactly one of p and q is

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Table 1.7: Truth table for XOR

true but not both and is false otherwise.
Properties of Exclusive OR

(i) $p \oplus q \equiv q \oplus p$ (Commutative)

(ii) $(p \oplus q) \oplus r \equiv p \oplus (q \oplus r)$ (Associative)

(iii) $p \wedge (q \oplus r) \equiv (p \wedge q) \oplus (p \wedge r)$ (Distributive)

1.3.6 Tautologies and Contradictions

A compound proposition that is always true for all possible truth values of its variables or, in other words contain only T in the last column of its truth table is called a *tautology*. A compound proposition that is always false for all possible values of its variables or, in other words contain only F in the last column of its truth table is called a *contradiction*. Finally a proposition that is neither a tautology nor a contradiction is called a *contingency*.

Propositions like

(a) The professor is either a woman or a man,

(b) People either like watching TVs or they don't

are always true and are called tautologies.

Propositions like

(a) x is prime and x is an even integer greater than 8,

(b) All men are good and all men are bad

are always false and are called contradictions.

***Example* 1.3.5:** Prove that the following propositions are tautology

(a) $p \vee \sim p$

(b) $\sim (p \wedge q) \vee q$

(c) $p \Rightarrow (p \vee q)$

Solution.

(a) The truth table of the given proposition is shown below. Since the truth value is TRUE for all possible values of the propositional variables which can be seen in the last column of the table, the given proposition is a tautology.

p	$\sim p$	$p \vee \sim p$
T	F	T
F	T	T

(b) We construct the truth table for the expression in question. It can be seen that for any possible assignment of p and q, the expression $\sim (p \wedge q) \vee q$ is true, which establishes that it is a tautology.

p	q	$p \wedge q$	$\sim (p \wedge q)$	$\sim (p \wedge q) \vee q$
T	T	T	F	T
T	F	F	T	T
F	T	F	T	T
F	F	F	T	T

(c) We construct the truth table of the given expression. It can be seen from the last column of the truth table that the expression is true for all possible assignments of p and q. Hence the proposition is a tautology.

p	q	$p \vee q$	$p \Rightarrow (p \vee q)$
T	T	T	T
T	F	T	T
F	T	T	T
F	F	F	F

1.4 Normal Forms

By comparing truth tables, one determine whether two logical expressions P and Q are equivalent. But the process is very tedious when the number of variable increases. A better method is to transform the expression P and Q to some standard forms of expressions P' and Q' such

COLLEGE TRB
Computer Science
2024-2025

TEACHER'S CARE PUBLICATION
38/23, Vaigundaperumal Koil Street,Kancheepuram- 631502
Mobile : 95665 35080, 9786269980 Land Line : 044-2723 5080

Your Success is Our Goal...

UNIT-5
Software Engineering

tcaexamguide@gmail.com
Teachers Care Academy
www.tcaexamguide.com
tcaexamguide

9360268118
7639967359
6369036346
9566535080

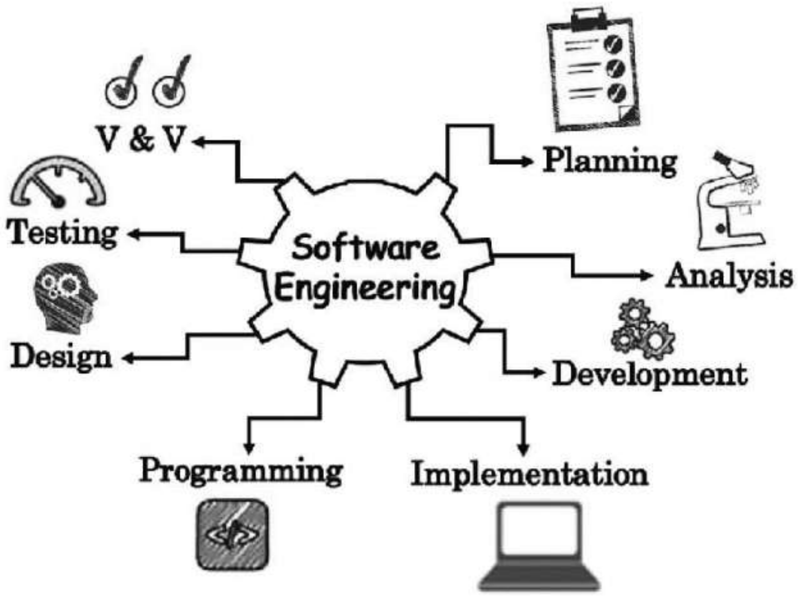
TEACHER'S CARE ACADEMY

KANCHIPURAM

COMPUTER SCIENCE

UNIT – V

SOFTWARE ENGINEERING



COMPETITIVE EXAM

FOR

ASSISTANT PROFESSOR TRB

2024 - 25

INDEX

ASSISTANT PROFESSOR TRB

COMPUTER SCIENCE – 2024-25

UNIT - V

SOFTWARE ENGINEERING

CHAPTER – I - SOFTWARE PROCESS MODELS

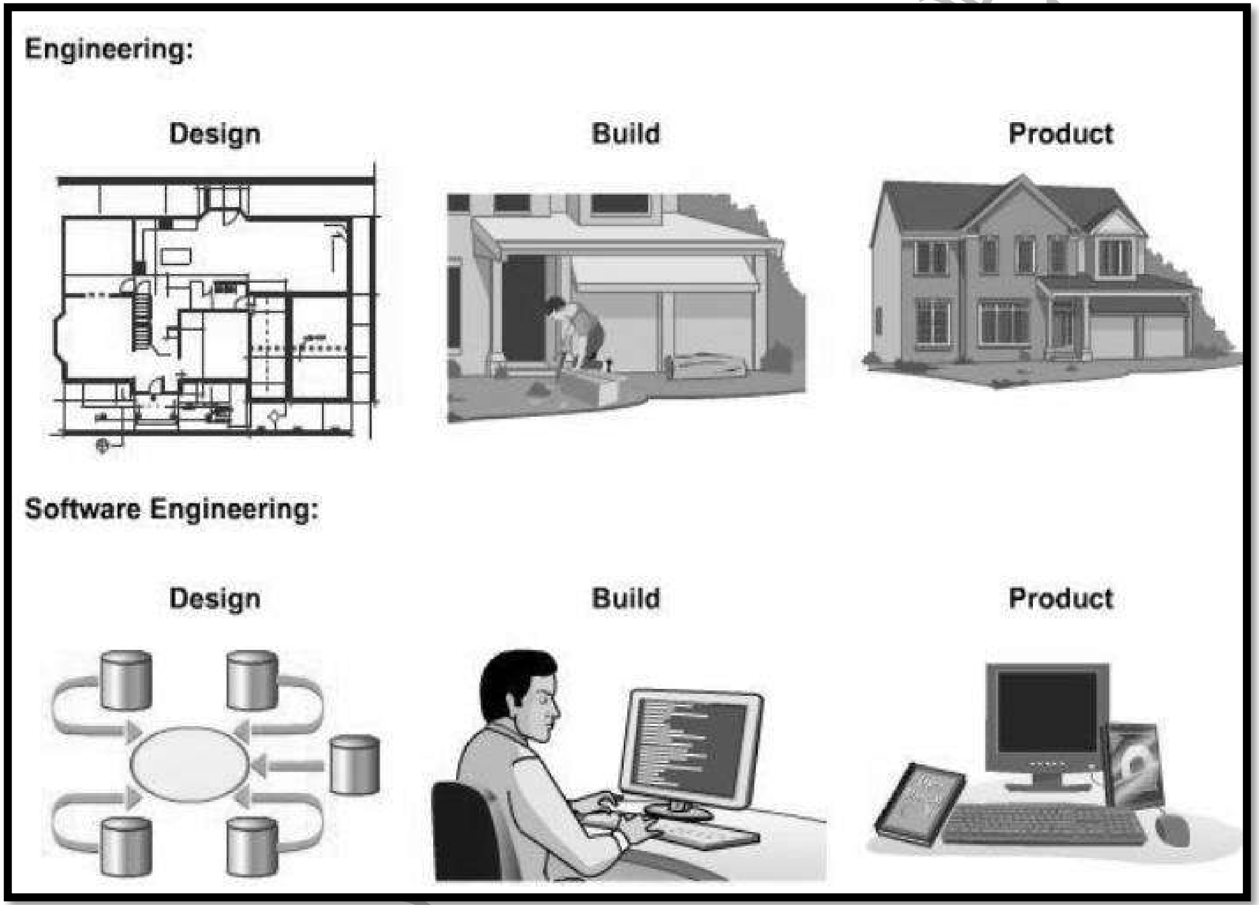
1.1. WHAT IS SOFTWARE ENGINEERING?

- Software is defined as computer programs, procedures, rules and possibly associated documentation and data pertaining to the operation of a computer based systems.
- Computer software is often divided into two categories:
  1. System software. This software includes the operating system and all utilities that enable the computer to function.
  2. Application software. These consist of programs that do real work for users. For example, word processors, spreadsheets, and database management systems fall under the category of applications software.

Definition:

- Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, i.e., the application of engineering to software.
- Software Engineering deals with cost-effective solutions to practical problems by applying scientific knowledge in building software artifacts in the service of mankind.

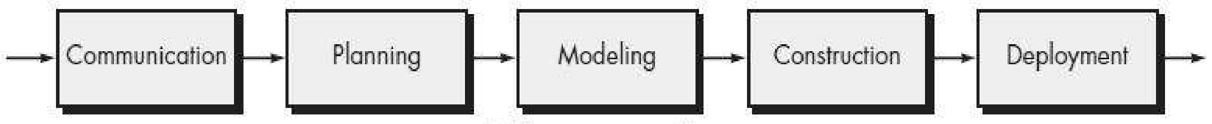
- Software Engineering is the application of methods and scientific knowledge to create practical cost-effective solutions for the design, construction, operation and maintenance of software.
- Software Engineering is a discipline whose aim is the production of fault free software that satisfies the user's needs and that is delivered on time and within budget**. (NET-JUNE-2012)**



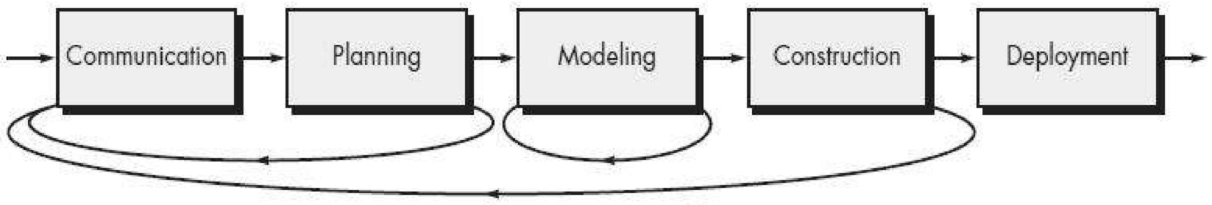
1.2. SOFTWARE PROCESS, GENERIC PROCESS MODEL

- A *process* is a collection of activities, actions, and tasks that are performed when some work product is to be created.
- An *activity* strives to achieve a broad objective (e.g., communication with stakeholders) and is applied regardless of the application domain, size of the project, complexity of the effort, or degree of rigor with which software engineering is to be applied.
- An *action* (e.g., architectural design) encompasses a set of tasks that produce a major work product (e.g., an architectural design model).
- A *task* focuses on a small, but well-defined objective (e.g., conducting a unit test) that produces a tangible outcome.

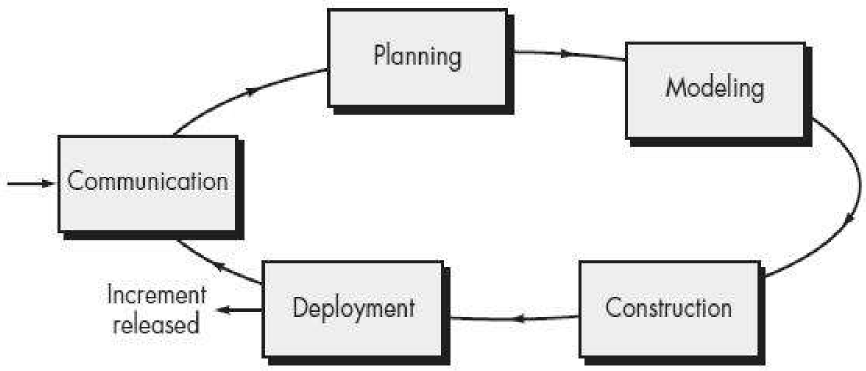
- A *process framework* establishes the foundation for a complete software engineering process by identifying a small number of *framework activities* that are applicable to all software projects, regardless of their size or complexity.
- In addition, the process framework encompasses a set of *umbrella activities* that are applicable across the entire software process.
- A generic process framework for software engineering encompasses five activities: **(NET-JUNE-2011)**
  1. **Communication:** Before any technical work can commence, it is critically important to communicate and collaborate with the customer. The intent is to understand stakeholders' objectives for the project and to gather requirements that help define software features and functions.
  2. **Planning:** Any complicated journey can be simplified if a map exists. A software project is a complicated journey, and the planning activity creates a "map" that helps guide the team as it makes the journey. The map - called a *software project plan* - defines the software engineering work by describing the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule.
  3. **Modeling:** Whether you're a landscaper, a bridge builder, an aeronautical engineer, a carpenter, or an architect, we work with models every day. We create a "sketch" of the thing so that you'll understand the big picture - what it will look like architecturally, how the constituent parts fit together, and many other characteristics.
  4. **Construction:** This activity combines code generation (either manual or automated) and the testing that is required uncovering errors in the code.
  5. **Deployment:** The software is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.
- In addition, a set of umbrella activities project tracking and control, risk management, quality assurance, configuration management, technical reviews, and others are applied throughout the process. This aspect is called *process flow.* It describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time and is illustrated in following figure.



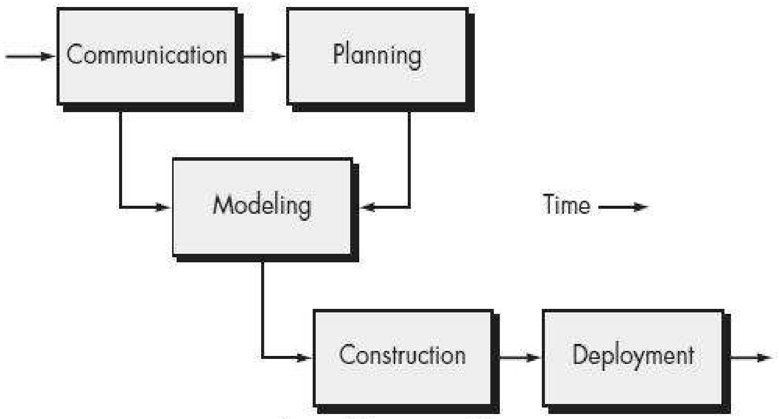
(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow



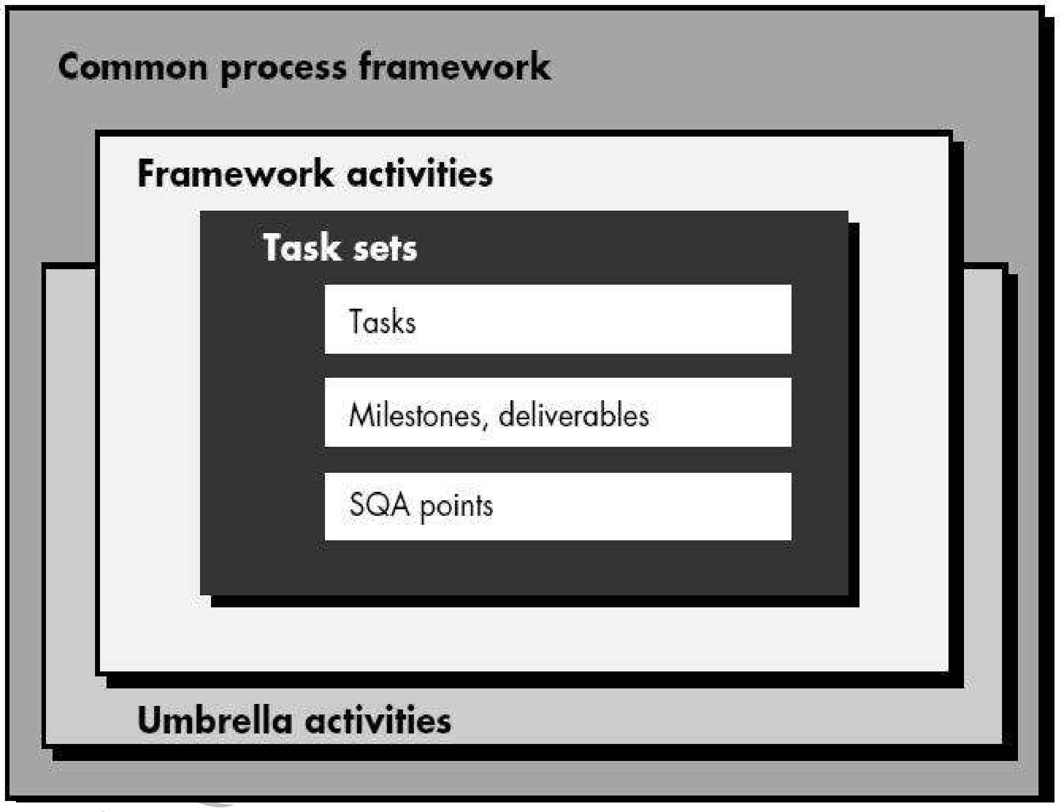
(d) Parallel process flow

1.2.1. Framework Activities

- Software engineering process framework activities are complemented by a number of *umbrella activities.* In general, umbrella activities are applied throughout a software project and help a software team manage and control progress, quality, change, and risk. Typical umbrella activities include:
    1. **Software project tracking and control:** allows the software team to assess progress against the project plan and take any necessary action to maintain the schedule.
    2. **Risk management:** assesses risks that may affect the outcome of the project or the quality of the product.

3. **Software quality assurance:** defines and conducts the activities required to ensure software quality.
4. **Technical reviews:** assesses software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity.
5. **Measurement:** defines and collects process, project, and product measures that assist the team in delivering software that meets stakeholders' needs; can be used in conjunction with all other framework and umbrella activities.
6. **Software configuration management:** manages the effects of change throughout the software process.
7. **Reusability management:** defines criteria for work product reuse and establishes mechanisms to achieve reusable components.
8. Work product preparation and production: encompasses the activities required to create work products such as models, documents, logs, forms, and lists.

- A software process can be characterized as shown below:



- A ***common process framework*** is established by defining a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.
- A number of *task sets* - each a collection of software engineering work tasks, project milestones, work products, and quality assurance points - enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team.

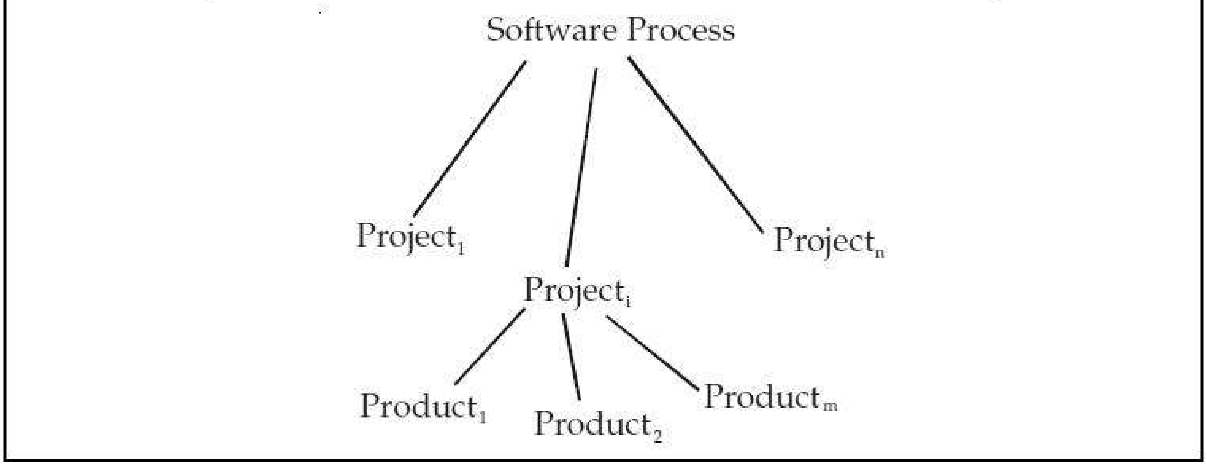
- A set of activities whose goal is the development or evolution of software. Generic activities in all software processes are:
    1. ***Specification:*** what the system should do and its development constraints.
    2. ***Development:*** production of the software system.
    3. ***Validation:*** checking that the software is what the customer wants.
    4. ***Evolution:*** changing the software in response to changing demands.

Software Process Model

- A simplified representation of a software process, presented from a specific perspective. *Example:* Process perspectives are:
    - Workflow perspective - sequence of activities
    - Data-flow perspective - information flow
    - Role/action perspective – who does what
- Software engineering approach pivots around the concept of process. A process means "a particular method of doing something, generally involving a number of steps or operations." In software engineering, the phrase software process refers to the method of developing software.
- A software process is a set of activities, together with ordering constraints among them, such that if the activities are performed properly and in accordance with the ordering constraints, the desired result is produced.
- The desired result is high-quality software at low cost. Clearly, if a process does not scale up and cannot handle large software projects or cannot produce good-quality software, it is not a suitable process.
- Major software development organizations typically have many processes executing simultaneously. Many of these do not concern software engineering, though they do impact software development. These could be considered non-software engineering process models.
- Business process models, social process models, and training models, are all examples of processes that come under this. These processes also affect the software development activity but are beyond the purview of software engineering.
- Software Process framework is a set of guidelines, concepts and best practices that describes high level processes in software engineering. It does not talk about how these processes are carried out and in what order.
- Each software development project starts with some needs and is expected to end with some software that satisfies those needs.

- A software process specifies the abstract set of activities that should be performed to go from user needs to the final product.
- The actual act of executing the activities for some specific user needs is a software project. And all the outputs that are produced while the activities are being executed are the products.
- One can view the software process as an abstract type, and each project is done using that process as an instance of this type.
- In other words, there can be many projects for a process and there can be many products produced in a project.
- This relationship is shown in below figure:

Relation between Processes, Projects and Products



- The sequence of activities specified by the process is typically at an abstract level because they have to be usable for a wide range of projects. Hence, "implementing" them in a project is not straightforward. To clarify this, let us take the example of book writing. A process for book writing on a subject will be something like this:
    1. Set objectives of the book – audience, marketing etc.
    2. Outline the contents.
    3. Research the topics covered.
    4. Do literature survey.
    5. Write and/or compile the content.
    6. Get the content evaluated by a team of experts.
    7. Proof read the book.
    8. Make corrections, if any.
    9. Get the book typeset.
    10. Print the book.

11. Bind the book.

- Overall, the process specifies activities at an abstract level that are not project-specific.

1.2.2. Task Set and Process Patterns

- Task set is a collection of software engineering work tasks, milestones and deliverables that must be accomplished to complete a particular software project.
- Task sets are different for different types of projects
- Most organizations encounter following types of projects
  - Concept development projects
    - Explore some new business concept or application of new technology
  - New application development projects
    - Undertaken as a consequence of specific customer request
  - Application Enhancement projects
    - Involve modification to functions, performance or interfaces (observable by end-user) in existing software
  - Application maintenance projects
    - That correct, adapt or extend existing software in ways that may not be obvious to end user
  - Reengineering projects
    - Undertaken for rebuilding an existing system in whole or part
- Process patterns can be defined as the set of activities, actions, work tasks or work products and similar related behaviour followed in a software development life cycle.
- Process patterns can be more easily understood by dividing it into terms: "Process", which means the steps followed to achieve a task and "patterns", which means the recurrence of same basic features during the lifecycle of a process. Thus in a more universal term process patterns are common or general solution for a complexity.
- Typical Examples are:
  1. Customer communication (a process activity).
  2. Analysis (an action).
  3. Requirements gathering (a process task).
  4. Reviewing a work product (a process task).
  5. Design model (a work product).
- Patterns can be defined at any level of abstraction. A pattern might be used to

describe a problem (and solution) associated with a complete process model (e.g., prototyping).

- In other situations, patterns can be used to describe a problem (and solution) associated with a framework activity (e.g., planning) or an action within a framework activity (e.g., project estimating).
- Ambler has proposed a template for describing a process pattern:
- Pattern Name. The pattern is given a meaningful name describing it within the context of the software process (e.g., Technical Reviews).
- Forces. The environment in which the pattern is encountered and the issues that make the problem visible and may affect its solution.
- Type. The pattern type is specified. Ambler suggests three types:

1. ***Stage pattern***—defines a problem associated with a framework activity for the process. Since a framework activity encompasses multiple actions and work tasks, a stage pattern incorporates multiple task patterns that are relevant to the stage. This pattern would incorporate the task pattern Requirements Gathering and others.

2. ***Task pattern***—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice.

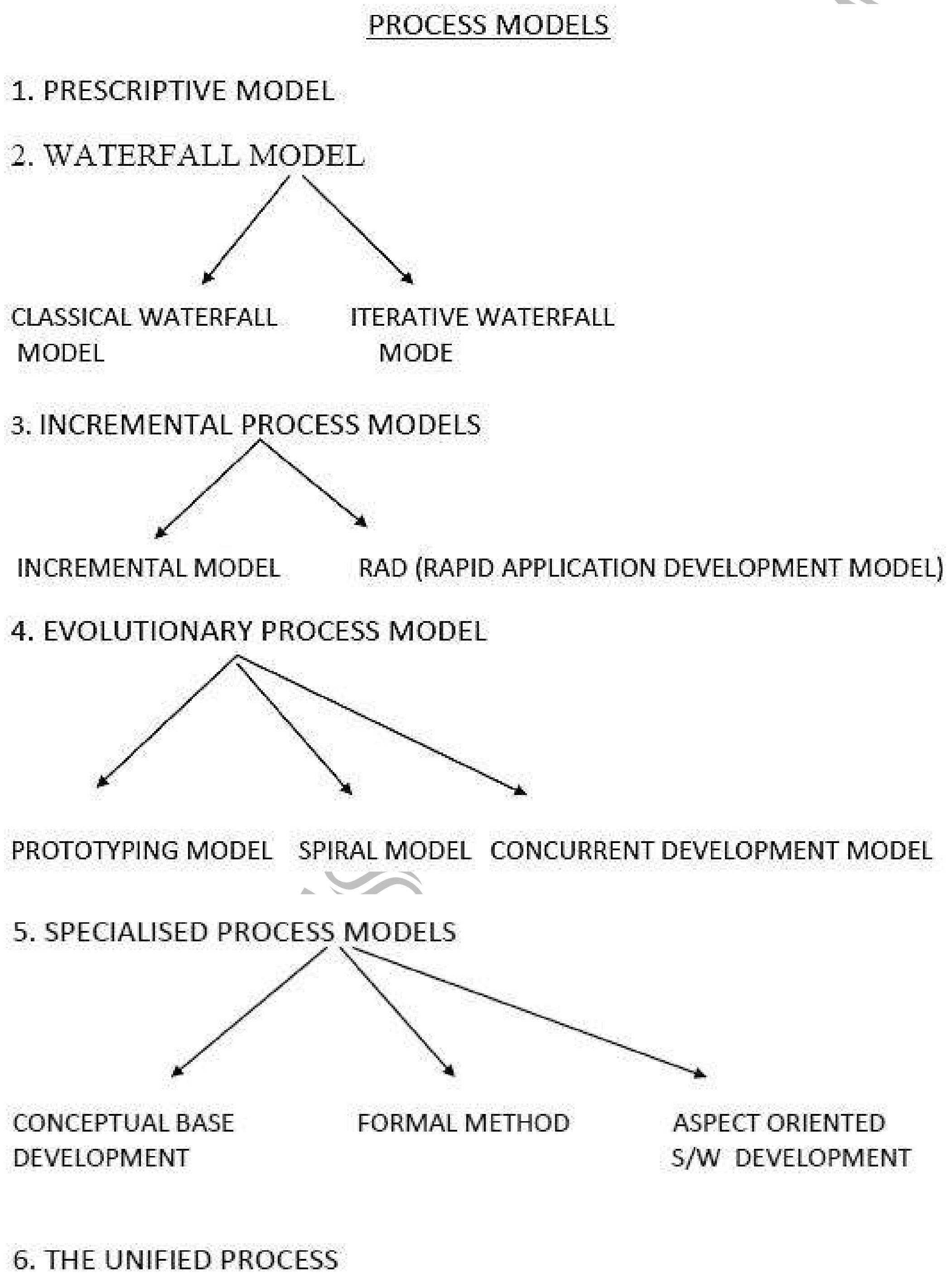
3. ***Phase pattern***—define the sequence of framework activities that occurs within the process, even when the overall flow of activities is iterative in nature. An example of a phase pattern might be Spiral Model or Prototyping.

1.2.3. Process Model Types

- A software process model is a simplified description of a software process, which is presented from a particular perspective.
- Models, by their very nature, are simplifications, so a software process model is an abstraction of the actual process, which is being described.
- Process models may include activities, which are part of the software process, software products and the roles of people involved in software engineering. Some examples of the types of software process model that may be produced are:
  1. ***A workflow model:*** This shows the sequence of activities in the process along with their inputs, outputs and dependencies. The activities in this model represent human actions.
  2. ***A dataflow or activity model:*** This represents the process as a set of activities each of which carries out some data transformation. It shows how the input to the process such as a specification is transformed to an output such as a design.

The activities here may be at a lower level than activities in a workflow model. They may represent transformations carried out by people or by computers.

3. ***A role/action model:*** This represents the roles of the people involved in the software process and the activities for which they are responsible.


PROCESS MODELS
1. PRESCRIPTIVE MODEL
2. WATERFALL MODEL
CLASSICAL WATERFALL MODEL
ITERATIVE WATERFALL MODE
3. INCREMENTAL PROCESS MODELS
INCREMENTAL MODEL
RAD (RAPID APPLICATION DEVELOPMENT MODEL)
4. EVOLUTIONARY PROCESS MODEL
PROTOTYPING MODEL
SPIRAL MODEL
CONCURRENT DEVELOPMENT MODEL
5. SPECIALISED PROCESS MODELS
CONCEPTUAL BASE DEVELOPMENT
FORMAL METHOD
ASPECT ORIENTED S/W DEVELOPMENT
6. THE UNIFIED PROCESS

- There are a number of different general models or paradigms of software development:
  - ***The waterfall approach:*** This takes the above activities and represents them as separate process phases such as requirements specification, software design, implementation, testing and so on. After each stage is defined it is 'signed off' and development goes on to the following stage.

- ***Evolutionary development:*** This approach interleaves the activities of specification, development and validation. An initial system is rapidly developed from very abstract specifications. This is then refined with customer input to produce a system which satisfies the customer's needs. The system may then be delivered. Alternatively, it may be reimplemented using a more structured approach to produce a more robust and maintainable system.
- ***Formal transformation:*** This approach is based on producing a formal mathematical system specification and transforming this specification, using mathematical methods to a program. These transformations are 'correctness preserving'. This means that you can be sure that the developed program meets its specification.
- ***System assembly from reusable components:*** This technique assumes that parts of the system already exist. The system development process focuses on integrating these parts rather than developing them from scratch.

1.2.4. Characteristics of Software Model

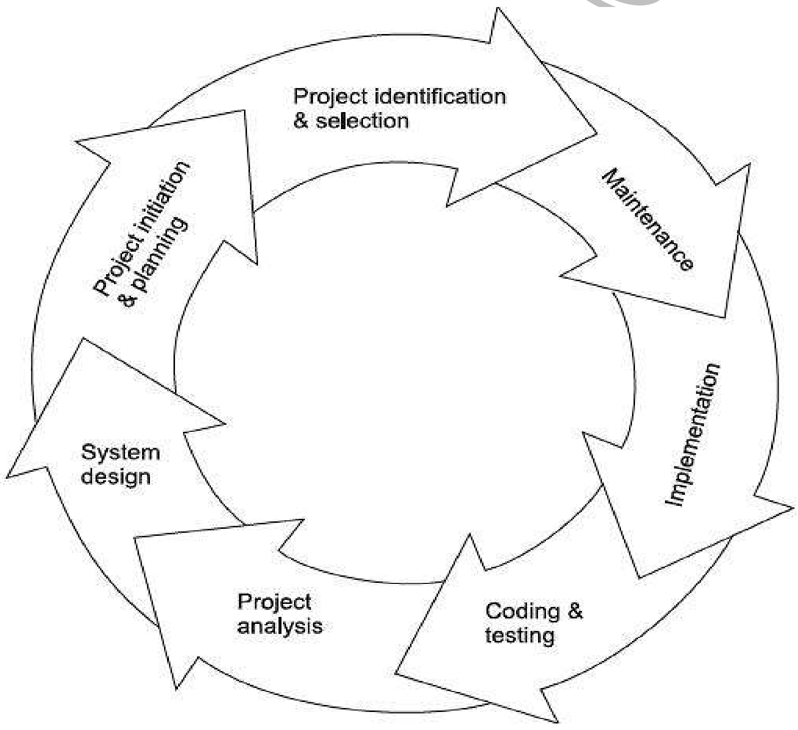
- Software engineering can be defined as: "Software engineering is the branch of engineering in which the software is developed in a manner that it will have all the properties like maintenance, extensibility and is within time and budget and also satisfies all the requirements given by user." Producing software does not mean producing just software but it means to develop the software in a cost effective manner. Characteristics of well engineered software are:
- ***Efficiency*****:** Software is said to be well-organized if it uses the available resources in the most efficient manner. The software should be able to offer a quick response in the least processing time using the resources at minimum level.
- Resources refer to the memory and processor utilization. The software should efficiently perform what the user demanded and give appropriate response in each case i.e. the output given is accurate or not.
- ***Maintainability*****:** This characteristic of the software is important for both the software engineer and the user. If the change is to be required in the software then the change leads to the change in the software so that it performs in accordance with the user requirement. The software engineer has to respond very fast if there is any change in the user requirements. Changes should be performed like this that it will not affect the overall integrity of the software.
- ***On-time*****:** The software should be developed on-time. If the software is urbanized late then it is of no use. A good engineer always develops the software on-time.

- ***Within Budget*****:** Some of the software gets swarming. Overrun does not mean that the cost of the software exceeds the limit given by user. But, it means that the software cost is out of control. So, the software should be developed in such a manner that it will not overrun and the software being developed is within budget.
- ***Functionality*****:** The software system is developed in a manner that it performs the entire task perfectly for which it is developed. The software should respond correctly as the user wants.
- ***Adaptability*****:** The software should be adaptable. Software is desired to be adaptable all the changes efficiently. The software should easily adopt all the changes in the software with no change in the efficiency of the software.
- ***Dependability*****:** It is the ability of the software that should not cause any physical or economic damage in the event of system failure. It includes a range of characteristics like: Reusability, security, and safety.
- ***Usability*****:** Software becomes usable if it does not call for extra efforts to be learned. Usability increases with good documentation provided along with the software. In software operations a lot depends on the quality of user manual. If software satisfies all the above characteristics then it is said to be good software or the software is well engineered.

1.3. PROCESS LIFE-CYCLE MODELS

- The software-development life-cycle is used to facilitate the development of a large software product in a systematic, well-defined, and cost-effective way.
- An information system goes through a series of phases from conception to implementation. This process is called the Software-Development Life-Cycle. Various reasons for using a life-cycle model include:
  1. Helps to understand the entire process
  2. Enforces a structured approach to development
  3. Enables planning of resources in advance
  4. Enables subsequent controls of them
  5. Aids management to track progress of the system
- The software development life-cycle consists of several phases and these phases need to be identified along with defining the entry and exit criteria for every phase. A phase can begin only when the corresponding phase-entry criteria are satisfied. Similarly, a phase can be considered to be complete only when the corresponding exit criteria are satisfied. If there is no clear indication of the entry and exit for every phase, it becomes very difficult to track the progress of the project.

- The software development life-cycle can be divided into 5-9 phases, i.e., it must have a minimum of five phases and a maximum of nine phases. On average it has seven or eight phases. These are:
  1. Project initiation and planning/Recognition of need/Preliminary investigation
  2. Project identification and selection/Feasibility study
  3. Project analysis
  4. System design
  5. Coding
  6. Testing
  7. Implementation
  8. Maintenance



Software-Development Life-Cycle

1. **Recognition of Need**. Recognition of need is nothing but the problem definition. It is the decision about problems in the existing system and the 2-impetus for system change. The first stage of any project or system-development life-cycle is called the preliminary investigation.
   - This investigation provides the organization's computer steering committee and any project team a set of terms or references for more detailed work. This is carried out by a senior manager and will result in a study proposal.
   - At this stage the need for changes in the existing system are identified and shortcomings of the existing system are detected. These are stated clearly providing the basis for the initial or feasibility study.

COLLEGE TRB
Computer Science

2024-2025

TEACHER'S CARE PUBLICATION
38/23, Vaigundaperumal Koil Street,Kancheepuram- 631502
Mobile : 95665 35080, 9786269980 Land Line : 044-2723 5080

Your Success is Our Goal...

VOLUME-1

UNIT-6
Data Structures and Algorithms

tcaexamguide@gmail.com
Teachers Care Academy
www.tcaexamguide.com
tcaexamguide

9360268118
7639967359
6369036346
9566535080

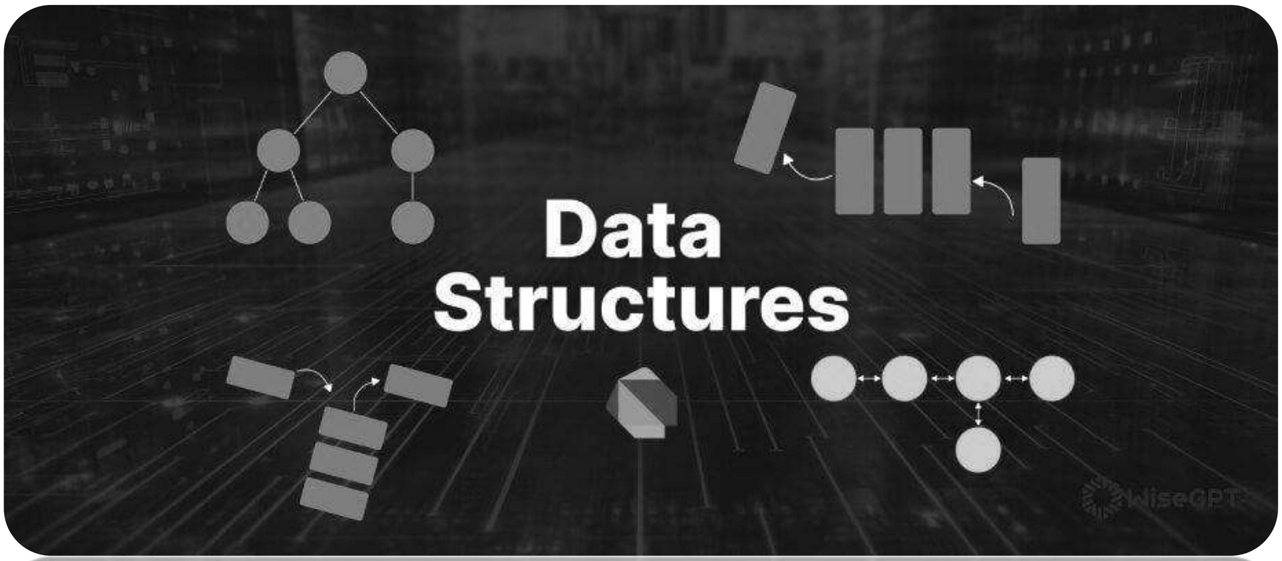
TEACHER'S CARE ACADEMY

KANCHIPURAM

COMPUTER SCIENCE

UNIT – VI - DATA STRUCTURES & ALGORITHMS

(Volume -1)



COMPETITIVE EXAM

For

COLLEGE TRB - CS – 2024-25

UNIT-VI DATA STRUCTURES & ALGORITHMS
(VOLUME-1)

UNIT VI: DATA STRUCTURES & ALGORITHMS

SYLLABUS

Data Structures: Arrays and their Applications; Sparse Matrix, Stacks, Queues, Priority Queues, Linked Lists, Trees, Forest, Binary Tree, Threaded Binary Tree, Binary Search Tree, AVL Tree, B Tree, B+ Tree, B* Tree, Data Structure for Sets, Graphs, Sorting and Searching Algorithms; Hashing.

Performance Analysis of Algorithms and Recurrences: Time and Space Complexities; Asymptotic Notation, Recurrence Relations.

Design Techniques: Divide and Conquer; Dynamic Programming, Greedy Algorithms, Backtracking, Branch and Bound.

Lower Bound Theory: Comparison Trees, Lower Bounds through Reductions.

Graph Algorithms: Breadth-First Search, Depth-First Search, Shortest Paths, Maximum Flow, Minimum Spanning Trees.

Complexity Theory: P and NP Class Problems; NP-completeness and Reducibility.

Selected Topics: Number Theoretic Algorithms, Polynomial Arithmetic, Fast Fourier Transform, String Matching Algorithms.

Advanced Algorithms: Parallel Algorithms Algorithms, Randomized Algorithms.

BOOKS TO STUDY:

1) Classic Data Structures - D.Samanta
2) Data Structures made simple - Sathish Jain, Shashi Singh.
3) Data Types and structures - Gotlieb, C.C. and L.R.Gotlieb.
4) Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran, "Fundamentals of Computer Algorithms", 2nd Edition, University Press, 2008.
5) T. H. Cormen, C. L. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms", MIT Press

UNIT VII - DATA STRUCTURES & ALGORITHMS

INTRODUCTION

WHAT IS MEANT BY A DATA?

❖ Data is a Single (or) a set of values. (Or) Facts and statistics collected together for reference or analysis

WHAT IS MEANT BY DATA STRUCTURE?

- It is a logical or mathematical model of a particular organization of data.

(Or)

- Data Structure is a specialized format for organizing and storing data so that it can be accessed and worked with in appropriate ways to make a program efficient.
- **Data Structure = Organized Data + Allowed Operations.**

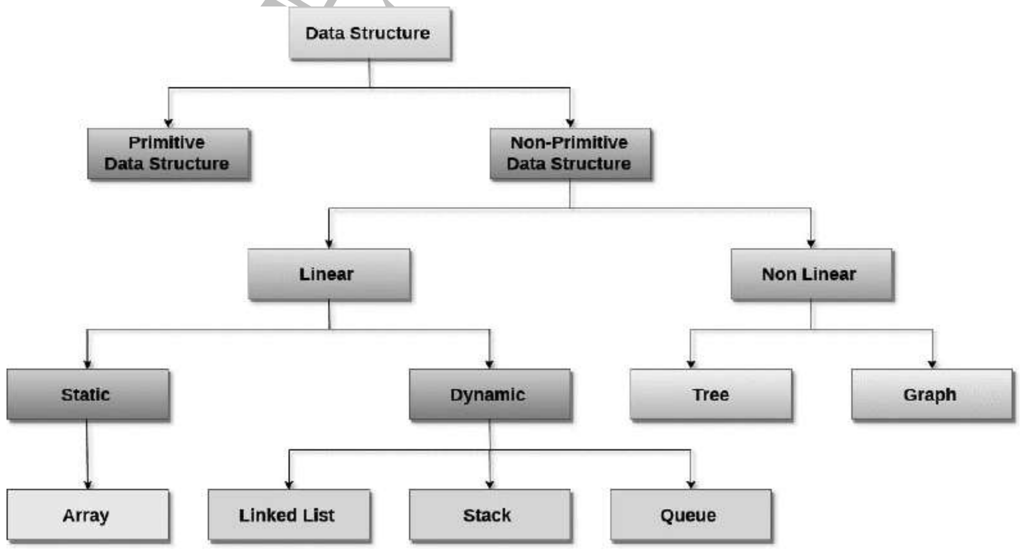
APPLICATIONS OF DATA STRUCTURE

Areas	Type of Data Structure
Operating System	Arrays and Tables
Data Base Management System	Array, Tables, B – Tress
Compiler Design	Hash Tables (look up an identifier)
Hierarchical Data Model	Trees

Categories of data structures:

- Two types:

1) Linear data structure → Single generic type **(UGC NET 2012)**
2) on-linear data structure → Multiple Individual type



1. ARRAY AND THEIR APPLICATIONS

- An array is a collection of items stored at contiguous memory locations.
- The idea is to store multiple items of the same type together.

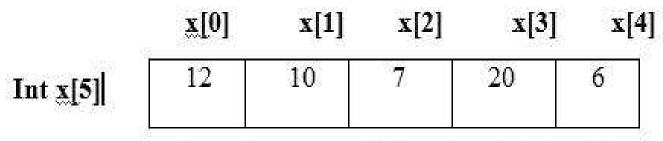


Fig : An array of size 5 containing integers

- This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array).
- Array data structure is typically used to implement hash table **(UGC NET 2023)**

1.1. ARRAYS TERMINOLOGY:

Size:

- Number of elements in an array is called the size of the array. Also called as length or dimension.

Type:

- Type of an away represents the kind of data type. Ex: int, string

Base:

- Base of an array is address of memory location where the first element in the array is located.

Range of index:

- Indices of array elements any charge can be referenced by subscript like Ai or A[i], this subscript is known as index. Index is always as integer value. Every element is identified by a subscripted or indexed variable
- Ex:
  - ✓ Int A[100]; The range of index is from 0 to 9
  - ✓ A:Array[-5….19] of integer: The Points of the rage is -5,-4,-3,….18,19.
  - ✓ Here L is the Lower Bound.
  - ✓ If the range of index varies from L…U then the size of the away can be calculated as Size(A)=U-L+1.

Word:

- It denotes the size of an element. In memory location computer can store an element of word size w. This word size varies from machine to machine such as 1 byte to 8 bytes.

1.2. OPERATIONS ON ARRAY

- The common operations can be performed on an array are
    - ✓ **Traversing-processing each element in the array.**
    - ✓ **Sorting -Organizing the elements in some order.**
    - ✓ **Searching -Finding the location of an element with a given value.**
    - ✓ **Insertion - Adding a new element.**
    - ✓ **Deletion -Removing an element.**
    - ✓ **Merging -Combining two arrays into a single array.**
- Although searching, and traversal of an array is an easy job, insertion and deletion is time consuming. The elements need to be shifted down before insertion and shifted up after deletion.

1. Traversing:

This operation is used visiting all elements in an array.

- Example: Array 'a' contains the following elements:

5	3	4	8	7			

The result of traversing is:

5

3

4

8

7

The algorithm for traversing is as follows:

1) Read the Array elements.
2) Display the elements of the array.

Algorithms: Traverse-array ( )

Input: An array A with elements

Output: According to process ( )

Steps:

```
1.  i=L                          //  start from first location L
2.  while i<=U do                // U upper bound
        1. Process (A[i])
```

```
2. i=i+1                    // move to next position
3. End while
4. Stop
```

Here process ( ) is an procedure which when called for an element can perform an action

2. Sorting:

❖ This operation if performed on an array will sort it in a specified order. The algorithm is used to store the elements of an integer array in ascending order or descending order.

Example:

Before Sort						
10	2	3	4	6	9	1

After Sort						
1	2	3	4	6	9	10

Algorithm steps:

1) Read the array elements.
2) Set the first position by comparing the first position of the array element with other array element. If the value is smaller than the first position element then swap the elements.
3) Set the second position by comparing the elements. If the value is smaller than the second position element then swap the elements.
4) Set the other positions likewise.

Input: An array with integer data

Output: An array with sorted element in an order according to ORDER ( )

Steps:

```
1. i= U
2. While i>= L do
    1. j=L                              // start comparing from first
    2. While j< i do
        1. If ORDER (A[j, A[j+1]) = FALSE          // if A[j] and A[j+1]
           are not in order
            1. Swap (A[j], A[j+1])          // Interchange the
               elements
        2. Endif
```

```
            3.  j=j+1                                   // Go to next statement
        3. Endwhile
        4. i=i-1
3.  Endwhile
4.  Stop
```

❖ Here order ( ) is a procedure to test whether two elements are in order and SWAP ( ) is a procedure to interchange the elements between two consecutive locations.

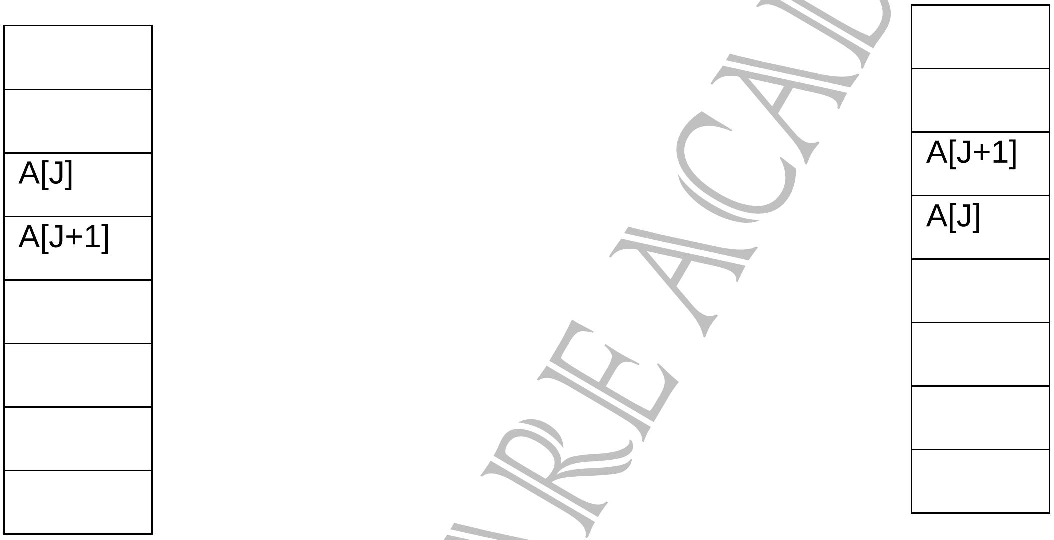


Fig: Swapping of two elements in an array

<u>3. Searching:</u>

❖ This operation is applied to search an element of interest in an array

Example:

4	5	6	8	9		

Element to search: 8

The given element is present in the position: 4

Algorithm steps:

1) Read the element to search.
2) Compare the element to the array elements.
3) If it matches then, display the position of the array.
4) Otherwise compare the entire array.
5) If match not found display the message "search is unsuccessful, key is not in the array "

Algorithm: Search_array(key)

Input: Key is the element to be searched

Output: Index of key in A or a message on failure

Steps:

```
1. i=L, found=0, location=0                    // found=0 indicates search is not
finished and unsuccessful
2. While (i<=U) and (found =0) do
        1. if compare(A[i], key)= true then
                1. Found=1
                2. Location =i
        2. Else
                1. i=i+1
        3. End if
3. End while
4. If found=0 then
        1. Print "search is unsuccessful; key is not in the array "
5. Else
        1. Print "search is successful: key is in the array at location ", location
6. End if
7. Return (location)
8. Stop--
```

4. Insertion:

- This operation is used to insert an element into an array provided that the array is not full.

Example:

Array

4	5	6	8	9		

Insert -2 at position: 3

4	5	6	8	9		

Shifting one position right

4	5	-2	6	8	9	

After insertion

Fig: Insertion of an element

Algorithm steps:

1) Read the number to insert and the position to insert.
2) Shift the numbers from the specified position, one place to the right from their existing position.
3) Place the number at the vacant place.

Algorithm: insert (key, location)

Input: key is the item; location is the index of the element where it is to be stored.

Output: array enriched with key

Steps:

```
1. if A[U] # NULL then
        1. print " Array is full , no insertion possible"
        2. Exit
2. Else
        1. i = U
        2. While i> location do
                1. A[i+1]=A[i]
                2. i = i-1
        3. End while
        4. A[location] =key
        5. U=U+1
3. End if
4. Stop
```

5. Deletion:

This operation is used to delete a particular element from an array. The element will be deleted by overwriting it with its subsequent element and this subsequent element then is to be deleted.

Example: Delete the element in the position: 3

Before deletion:

4	5	6	8	9		

After deletion:

4	5	8	9	0	0

Algorithm steps:

1) Read the position to delete.
2) Shift the numbers placed after the position, where the number is to be deleted.
3) Leave the last position blank.

Algorithm: delete (key)

Input: key is the element to be deleted.

Output: slimed array without key

Steps:

```
1. i = search_array (a, key)
2. if i=0 then
      1. print " key is not found, no deletion"
      2. Exit
3. Else
         1. While i< U do
                  1. A[i] = A[i+1]
                  2. i = i+1
         3. End while
4. End if
5. A[U] = NULL
6. U=U-1
7. Stop
```

6. Merging:

- Merging is an important operation when we need to compact the lements from two different arrays into a single array.

  Rules:

  1) Copy all the elements of first array into a new array(third array)
  2) Copy the second array into the third array after the position ,at which the last elements of the first array copied.

  Types

  - Merging can be done in two ways:
    - Merging without sorting.

⇨ Merging with sorting.

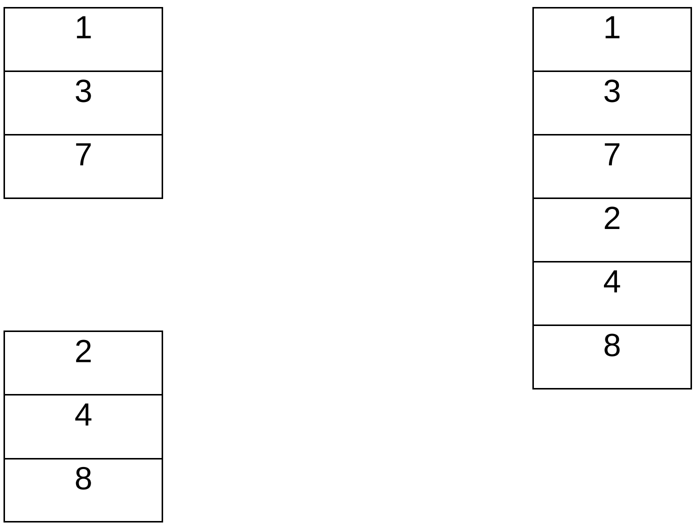


Fig: merging of A1 and A2 to A

Algorithm: merge (A1, A2: A)

Input: Two arrays A1 [L1…U1], A2 [l2…U2]

Output: Result array A [L…U] , where L=L1 and U=U1+(U2-L2+1) when A1 is append after A2

Steps:

```
1. i1=L1, i2=L2;                      // initialization of variables
2. L=L1, U=U1+U2 –L2 +1               // initialization of lower and upper
   bounds of an array
3. i=L
4. Allocate memory for a[L…U]
5. while i1<U do                      // to copy array A1 into the first
   part of A
        1. A[i] = A1[i1]
        2. i=i+1, i1=i1+1
6. End while
7. While i2<=U2 do                    // To copy the array A2 into last part of
   A
        1. A[i] = A2[i2]
        2. i=i+1, i2=i2+1
8. End while
9. Stop
```

COMPLEXITY ANALYSIS OF OPERATIONS ON ARRAYS

Time Complexity

1.3. TYPES OF ARRAYS

- ❖ Types of arrays depend upon the number of dimensions of an array.
- ❖ The count of indices or subscripts required to access one element of an array define the dimensions of an array.

Types of arrays

1) One-dimensional Array
2) Two-dimensional Array
3) Three-dimensional Array

- ❖ Two dimensional and three dimensional arrays are also called multi-dimensional arrays.

1. One-dimensional Array

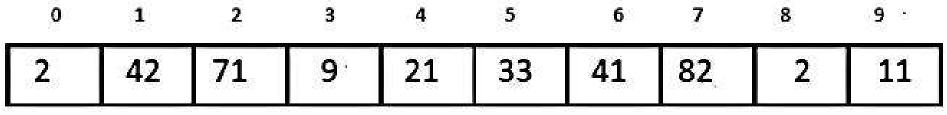
- ❖ In a one-dimensional array the elements are stored in contiguous memory

Operation	Best Case	Average Case	Worst Case
Traversal	O(1)	O(n)	O(n)
Insertion	O(1)	O(n)	O(n)
Deletion	O(1)	O(n)	O(n)
Search	O(1)	O(n)	O(n)
Update	O(1)	O(1)	O(1)

locations where each element is accessed by using a single index value. It is a linear data structure storing all the elements in sequence.

2. Two-dimensional Array

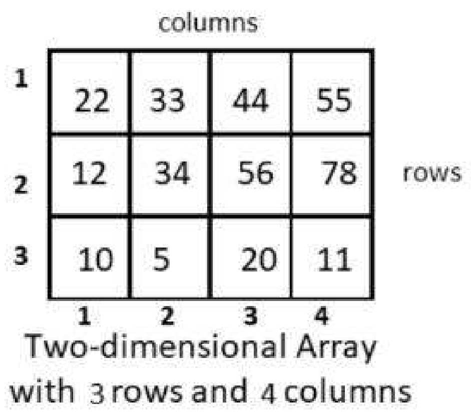
- ❖ In types of arrays, a two dimensional array is a tabular representation of data where elements are stored in rows and columns.



One Dimensional Array

- ❖ A two dimensional array is actually a collection of M X N elements which has M rows and N columns. To access any element in a two-dimensional array two subscripts are required for defining position of an element in a specific row and column.

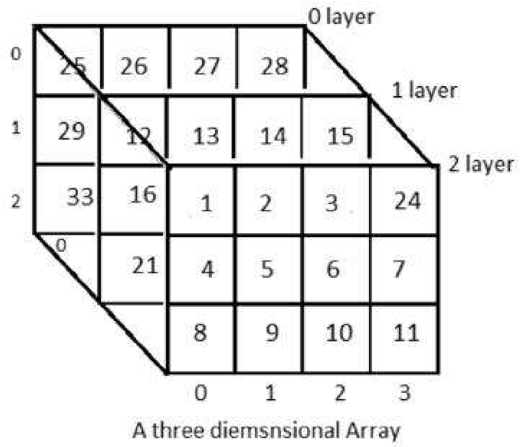
- ❖ The first index is for row number and second is for column index. In the example shown the first index values row=2 column=3 is used to access element 56.



Two-dimensional Array with 3 rows and 4 columns

3. Three-dimensional Array

- ❖ In types of arrays, a three-dimensional array is an extension to the two dimensional array with addition of depth.
- ❖ It can be seen as a cube that has rows, columns and depth as third dimension. To access any element in a three-dimensional array three subscripts are required for position of element in a specific row, column and depth.
- ❖ The first index is for depth (dimension or layer), second is for row index and third is for column. In the example shown the index values (2,0,3) is used to access element 24.



A three diemsnsional Array

1.4. APPLICATIONS OF ARRAYS IN DATA STRUCTURES

- **Storing and accessing data:** Arrays are used to store and retrieve data in a specific order. For example, an array can be used to store the scores of a group of students, or the temperatures recorded by a weather station.
- **Sorting:** Arrays can be used to sort data in ascending or descending order. Sorting algorithms such as bubble sort, merge sort, and quicksort rely heavily on arrays.

- **Searching:** Arrays can be searched for specific elements using algorithms such as linear search and binary search.
- **Matrices:** Arrays are used to represent matrices in mathematical computations such as matrix multiplication, linear algebra, and image processing.
- **Stacks and queues:** Arrays are used as the underlying data structure for implementing stacks and queues, which are commonly used in algorithms and data structures.
- **Graphs:** Arrays can be used to represent graphs in computer science. Each element in the array represents a node in the graph, and the relationships between the nodes are represented by the values stored in the array.
- **Dynamic programming:** Dynamic programming algorithms often use arrays to store intermediate results of sub problems in order to solve a larger problem.
- **Image processing:** Arrays can be used to represent and process images. Each element in the array represents a pixel in the image, and operations can be performed on the array to manipulate the image.
- **Numerical computations:** The application of an array is extensive in numerical computations, such as in linear algebra and signal processing. For example, a matrix can be represented as a two-dimensional array, and operations like matrix multiplication can be performed efficiently using arrays.
- **Games and simulations:** Arrays can be used to represent game boards, game pieces, and game states. They are also used in simulations to store and manipulate data over time.

1.5. ADVANTAGES & DISADVANTAGES OF ARRAYS IN DATA STRUCTURES

Advantages

- **Efficient access:** Arrays offer fast and efficient access to elements because each element can be accessed directly through its index. This makes array traversal quick and straightforward.
- **Versatility:** Arrays can be used to store any type of data like integers, characters, and strings. They can also be used to store user-defined data types, such as structures and classes.
- **Flexibility:** Arrays are used to implement other data structures like stacks, queues, trees, graphs, etc.
- **Easy to remember:** Arrays represent multiple data items of the same type using a single name. Therefore, it's easier to remember an array name than remembering the names of several variables.

Disadvantages

- **Fixed-size:** The size of an array is fixed at the time of its creation, which means that once the array is created, its size cannot be changed. This can be a limitation in situations where the size of the data is not known in advance.
- **Memory wastage:** There will be a wastage of memory if we store less number of elements than the declared size because there is static memory allocation in arrays.
- **Inefficient insertion and deletion:** Arrays store data in contiguous memory locations, which makes deletion and insertion very difficult to implement. All the elements after insertion or deletion must be shifted to accommodate the new element or fill in the gap. This can be a time-consuming process, especially for large arrays.
- **Homogeneous data:** Arrays can only store elements of the same data type, which can be a limitation in situations where the user needs to store data of different types.
- **No built-in support for dynamic resizing:** While some programming languages provide built-in support for dynamic resizing of arrays, many do not. In those cases, the developer may have to implement their own resizing logic, which can be complex and error-prone.

EXCERICE 1: (PART-A)

1) What is Data Structure?

A) Address of the variable B) Subset of all variables

C) The memory representation of data D) The type of the variable

2) Which of the following is a collection of heterogeneous elements?

A) Array B) Structure C) Stack D) Queue

3) __________is a linear data structure

A) Tree B) Array C) Graph D) None of these.

4) The Smallest element of an array index is __________

A) Smallest Bound B) Lower Bound

C) First Bound D) Higher Bound

5) Two Dimensional Array are also called a

A) Table Array B) Matrix Array

C) Both A & B D) None of above

6) An Array of n elements will be declared in c as

A) Array [n+1] B) Array [n-1]

C) Array [n] D) Array

7) The First Element of a 0 based array can be accessed by

A) Array [0] B) Array [n-1]

C) Array D) Both A & C

8) Which of the following data structure can't store the nonhomogeneous data elements?

A) Arrays B) Stacks

C) Records D) None of the above

9) In an array range specifies__________

A) Scope of the Array B) Number of the Elements in the Array

C) The Group of the Array D) Size-1 of the array

10) Which of the following data structures are indexed structures?

A) Linear arrays B) Linked lists

C) Both (A) & (B) D) None of above

2.SPARSE MATRIX

2.1. WHAT IS A MATRIX?

- A matrix can be defined as a two-dimensional array having 'm' rows and 'n' columns. A matrix with m rows and n columns is called m × n matrix. It is a set of numbers that are arranged in the horizontal or vertical lines of entries.
- For example:

$$A = \overset{\xrightarrow{\text{Row (m)}}}{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}} \Big\downarrow \text{Columns (n)}$$

2.2. WHAT IS SPARSE MATRIX?

- A matrix is a two-dimensional data object made of m rows and n columns, therefore having total m x n values. If most of the elements of the matrix have 0 value, then it is called a sparse matrix.
- There are the following benefits of using the sparse matrix -

COLLEGE TRB
Computer Science
2024-2025

TEACHER'S CARE PUBLICATION
38/23, Vaigundaperumal Koil Street,Kancheepuram- 631502
Mobile : 95665 35080, 9786269980 Land Line : 044-2723 5080

Your Success is Our Goal...

VOLUME-1

UNIT-8
Data Communication and Computer Networks

TEACHER'S CARE ACADEMY

KANCHIPURAM

UNIT – VIII

DATA COMMUNICATION AND COMPUTER NETWORKS

COMPETITIVE EXAM

FOR

CLG-TRB-2024 – 2025

TEACHER'S CARE ACADEMY, KANCHIPURAM

TNPSC-TRB- COMPUTER SCIENCE -TET COACHING CENTER

HEAD OFFICE: NO. 38/23, VAIGUNDA PERUMAL KOIL, SANNATHI STREET, KANCHIPURAM – 1. CELL: 9566535080

B.Off 2: 65C, Thillai Ngr(West), 4th Cross St, Trichy – 620018
B.Off 3: Vijiyaraghavachariar Memorial Hall(Opp to Sundar Lodge), Salem

Trichy : 76399 67359 Salem : 93602 68118

UNIT-8: DATA COMMUNICATION AND COMPUTER NETWORKS

SYLLABUS

Data Communication: Components of a Data Communication System, Simplex, Half- Duplex and Duplex Modes of Communication; Analog and Digital Signals; Noiseless and Noisy Channels; Bandwidth, Throughput and Latency; Digital and Analog Transmission; Data Encoding and Modulation Techniques; Broadband and Baseband Transmission; Multiplexing, Transmission Media, Transmission Errors, Error Handling Mechanisms.

Computer Networks: Network Topologies, Local Area Networks, Metropolitan Area Networks, Wide Area Network, Wireless Networks, Internet.

Network Models: Layered Architecture, OSI Reference Model and its Protocols; TCP/IP Protocol Suite, Physical, Logical, Port and Specific Addresses; Switching Techniques.

Functions of OSI and TCP/IP Layers: Framing, Error Detection and Correction; Flow and Error Control; Sliding Window Protocol, HDLC, Multiple Access – CSMA/CD, CSMA/CA, Reservation, Polling, Token Passing, FDMA, CDMA, TDMA, Network Devices, Backbone Networks, Virtual LANs.

IPv4 Structure and Address Space; Classful and Classless Addressing; Datagram, Fragmentation and Checksum; **IPv6** Packet Format, Mapping Logical to Physical Address (ARP), Direct and Indirect Network Layer Delivery; Routing Algorithms, TCP, UDP and SCTP Protocols; Flow Control, Error Control and Congestion Control in TCP and SCTP.

World Wide Web (WWW): Uniform Resource Locator (URL), Domain Name Service (DNS), Resolution–Mapping Names to Addresses and Addresses to Names; Electronic Mail Architecture, SMTP, POP and IMAP; TELNET and FTP.

Network Security: Malwares, Cryptography and Steganography; Secret-Key Algorithms, Public-Key Algorithms, Digital Signature, Virtual Private Networks, Firewalls.

Mobile Technology: GSM and CDMA; Services and Architecture of GSM and Mobile Computing; Middleware and Gateway for Mobile Computing; Mobile IP and Mobile Communication Protocol; Communication Satellites, Wireless Networks and Topologies; Cellular Topology, Mobile Adhoc Networks, Wireless Transmission and Wireless LANs; Wireless Geolocation Systems, GPRS and SMS.

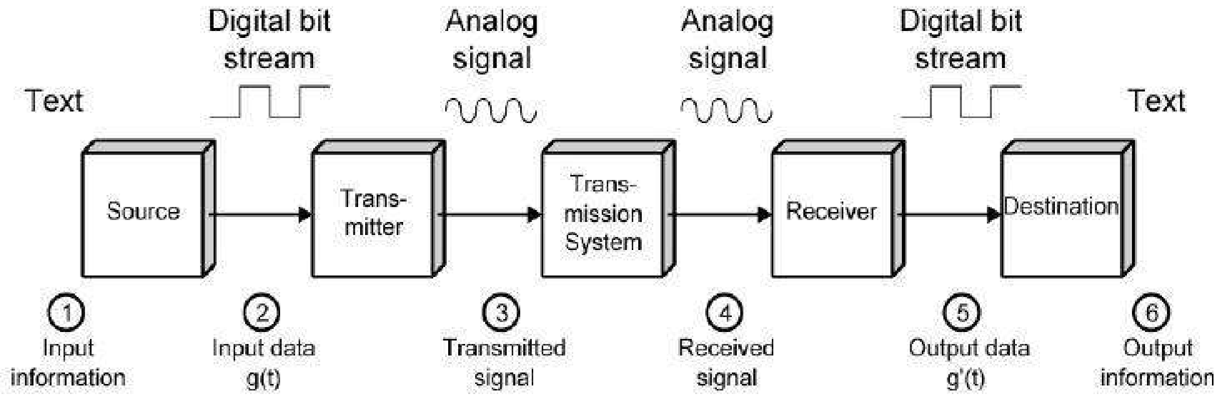
Cloud Computing and IoT: SaaS, PaaS, IaaS, Public and Private Cloud; Virtualization, Virtual Server, Cloud Storage, Database Storage, Resource Management, Service Level Agreement, Basics of IoT.

CHAPTER 1

DATA COMMUNICATION

1.1 COMPONENTS OF DATA COMMUNICATION SYSTEM

- Sending or Receiving information, such as speaking, writing, telephone lines , computers or using some other medium is communication.
- The **communication system** basically deals with the transmission of information from one point to another using the well-defined steps which are carried out in sequential manner. The system for data transmission makes use of the sender and destination address, In this other so many elements are also there that allows it to transfer data from one set of point to another set of point after dividing the **elements of communication system** in groups and these interface elements acts as the main **component for data communication**.



- A data communication system comprises several components that work together to facilitate the exchange of data between two or more devices. These components ensure that data is transmitted accurately, efficiently, and securely. The main components of a data communication system include:

1. Message:

- The message is the information or data that needs to be transmitted from the source to the destination.
- It can take various forms, such as text, numbers, images, audio, video, or any other type of digital data.

2. Sender/Transmitter:

- The sender or transmitter is the device that originates and sends the message.
- It converts the message into electrical, electromagnetic, or optical signals suitable for transmission over the communication channel.

3. Receiver:

- The receiver is the device that receives the transmitted signals from the sender.
- It converts the received signals back into the original message format for interpretation by the destination device.

4. Communication Channel/Medium:

- The communication channel or medium is the physical or logical pathway through which the signals travel from the sender to the receiver.
- It can be wired (e.g., twisted-pair cables, coaxial cables, optical fibers) or wireless (e.g., radio waves, microwaves, infrared).
- The choice of communication channel depends on factors such as distance, data rate, cost, and environmental conditions.

5. Protocol:

- A protocol is a set of rules, conventions, and standards that governs how data is transmitted and received between devices in a network.
- It defines parameters such as data format, error detection and correction, flow control, and addressing.
- Protocols ensure interoperability and reliable communication between different devices and systems.

6. Encoder/Decoder:

- Encoders and decoders are responsible for converting the message into a format suitable for transmission and then back into its original form upon reception.
- They may perform functions such as data compression, encryption, modulation, and demodulation.

7. Modem (Modulator-Demodulator):

- A modem is a device that modulates digital signals into analog signals for transmission over analog communication channels and demodulates analog signals back into digital signals upon reception.
- It enables communication between digital devices over analog communication networks such as telephone lines.

8. Switching and Routing Equipment:

- Switches and routers are network devices that direct data traffic between multiple devices and networks.
- They ensure that data packets are delivered to their intended destinations efficiently and securely.

- ➢ Switches operate at the data link layer (Layer 2) of the OSI model, while routers operate at the network layer (Layer 3).

9. Multiplexers/Demultiplexers:

- ➢ Multiplexers combine multiple signals or data streams into a single transmission channel, allowing for more efficient use of bandwidth.
- ➢ Demultiplexers separate the combined signals back into their original individual signals upon reception.

10. Terminal Equipment:

- ➢ Terminal equipment includes devices such as computers, terminals, printers, scanners, and other end-user devices that generate or consume data.
- ➢ They interact with the communication system to send or receive messages.
- ➢ These components work together to enable effective communication and data exchange between devices across various communication networks and systems. Each component plays a specific role in the transmission, reception, processing, and delivery of data within the communication system.

1.2. MODES OF COMMUNICATION (TRANSMISSION MODE)

- ➢ There are several modes of communication that define how data is transmitted between two communicating parties. These modes determine the direction of data flow and the interaction between the sender and receiver. The most common modes of communication include:

1. Simplex Mode:

- ➢ In simplex mode, communication occurs in only one direction, either from the sender to the receiver or from the receiver to the sender.
- ➢ The sender or transmitter can only transmit data, while the receiver can only receive data.
- ➢ Examples of simplex mode include television broadcasting and keyboard input to a computer.

2. Half-Duplex Mode:

- ➢ In half-duplex mode, communication can occur in both directions, but not simultaneously.
- ➢ A communication channel is shared between the sender and receiver, allowing both parties to transmit and receive data.
- ➢ However, only one party can transmit at a time, while the other party listens or receives.
- ➢ Walkie-talkies and two-way radios operate in half-duplex mode.

3. Full-Duplex Mode:

- In full-duplex mode, communication occurs in both directions simultaneously.
- Each party can transmit and receive data independently without any interference.
- Dedicated channels are used for transmitting and receiving, allowing for simultaneous communication.
- Examples of full-duplex mode include telephone conversations and most modern computer networks.

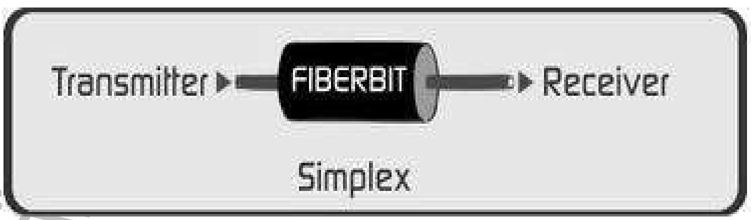
4. Asynchronous Mode:

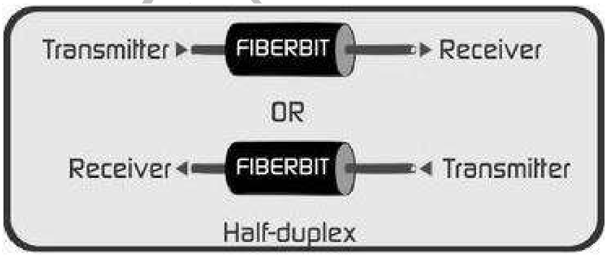
- In asynchronous mode, data transmission is not synchronized with a common clock signal between the sender and receiver.
- Each data character is preceded by start and stop bits to indicate the beginning and end of the data transmission.
- Asynchronous communication is commonly used in serial communication interfaces, such as RS-232.

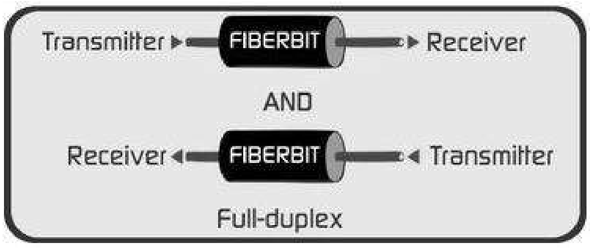
5. Synchronous Mode:

- In synchronous mode, data transmission is synchronized with a common clock signal shared between the sender and receiver.
- Data is transmitted in blocks or frames, with each frame preceded by synchronization bits or headers.
- Synchronous communication allows for higher data rates and more efficient bandwidth utilization.
- Examples of synchronous communication protocols include Ethernet and SONET/SDH.

- transfer speed, reliability, and cost considerations.





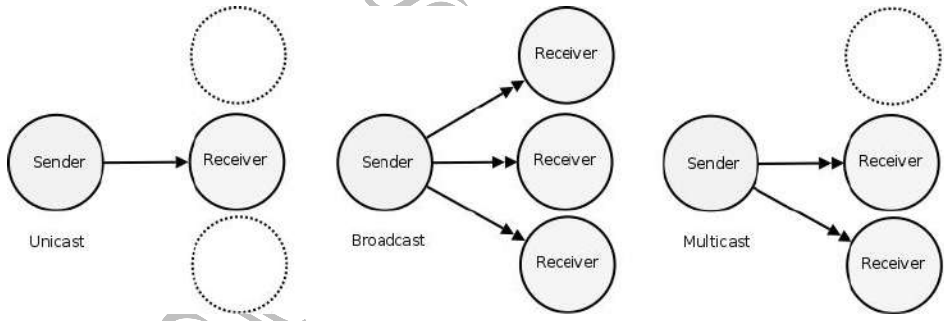


BASIS FOR COMPARISON	SIMPLEX	HALF DUPLEX	FULL DUPLEX
Direction of Communication	Communication is unidirectional.	Communication is two-directional but, one at a time.	Communication is two directional and done simultaneously.
Send/Receive	A sender can send data but, cannot receive.	A sender can send as well as receive the data but one at a time.	A sender can send as well as receive the data simultaneously.
Performance	The half-duplex and full duplex yields better performance than the Simplex.	The full duplex mode yields higher performance than half duplex.	Full duplex has better performance as it doubles the utilization of bandwidth.
Example	Keyboard and monitor.	Walkie-Talkies.	Telephone.

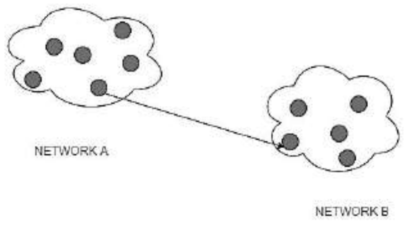
1.3. WAY OF COMMUNICATION

➢ The way of communication can be either of the following

- Unicast (one to one communication)
- Broadcast (One to all communication)
- Multicast (One to many Communication)



1. Unicast



➢ This type of information transfer is useful when there is a participation of single sender and single recipient. So, in short, you can term it as a one-to-one transmission.

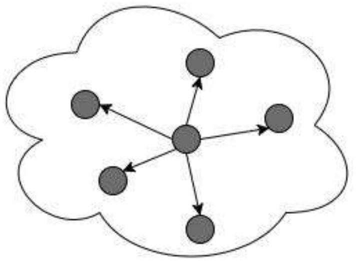
➢ For example, a device having IP address 10.1.2.0 in a network wants to send the traffic stream (data packets) to the device with IP address 20.12.4.2 in the other network, then unicast comes into the picture. This is the most common form of data transfer over the networks.

2. Broadcast

Broadcasting transfer (one-to-all) techniques can be classified into two types :

a. Limited Broadcasting

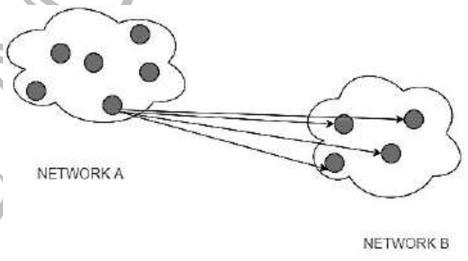
- Suppose you have to send stream of packets to all the devices over the network that you reside, this broadcasting comes handy.



- For this to achieve, it will append 255.255.255.255 (all the 32 bits of IP address set to 1) called as **Limited Broadcast Address** in the destination address of the datagram (packet) header which is reserved for information transfer to all the recipients from a single client (sender) over the network.

b. Direct Broadcasting

- This is useful when a device in one network wants to transfer packet stream to all the devices over the other network.



- This is achieved by translating all the Host ID part bits of the destination address to 1, referred as **Direct Broadcast Address** in the datagram header for information transfer.
- This mode is mainly utilized by television networks for video and audio distribution. One important protocol of this class in Computer Networks is **Address Resolution Protocol (ARP)** that is used for resolving IP address into physical address which is necessary for underlying communication.

3. Multicast

- In multicasting, one/more senders and one/more recipients participate in data transfer traffic. In this method traffic recline between the boundaries of unicast (one-to-one) and broadcast (one-to-all).
- Multicast lets server's direct single copies of data streams that are then simulated and routed to hosts that request it. IP multicast requires support of some other protocols like **IGMP (Internet Group Management Protocol), Multicast routing** for its working. Also in Classful IP addressing **Class D** is reserved for multicast groups.

4. Point-to-Point Communication:

- In point-to-point communication, data is transmitted between two individual nodes, typically over a dedicated communication link.
- Examples include serial communication between two devices using cables or wireless communication between two mobile devices.

5. Client-Server Communication:

- In client-server communication, one or more client devices request services or resources from a central server.
- The server responds to client requests by providing the requested data or performing the requested tasks.
- This model is prevalent in networked environments, such as web browsing, email services, and file sharing.

6. Peer-to-Peer (P2P) Communication:

- Peer-to-peer communication enables direct interaction between individual nodes without the need for a central server.
- Each node can act as both a client and a server, exchanging data and resources with other peers on the network.
- P2P networks are commonly used for file sharing, distributed computing, and decentralized applications.

7. Inter-Process Communication (IPC):

- IPC facilitates communication between different processes or threads running on the same or different devices within a network.
- Methods such as pipes, sockets, shared memory, and message passing are used for IPC in operating systems and distributed computing environments.

8. Remote Procedure Call (RPC):

- RPC allows a process to execute code or invoke procedures on a remote system as if they were local.
- It abstracts the complexities of network communication, enabling seamless interaction between distributed components.
- RPC is widely used in client-server applications, distributed systems, and remote administration tasks.

CHAPTER 1.1, 1.2 & 1.3 MCQ

1. Which component is responsible for converting digital signals to analog signals for transmission over analog communication channels?

A) Modem B) Router C) Switch D) Gateway

2. Which component of data communication is responsible for encoding data into signals suitable for transmission?

A) Sender B) Receiver C) Transmission medium D) Modem

3. Which communication mode is best suited for applications where real-time bidirectional communication is required?

(A) Simplex (B) Half Duplex (C) Full Duplex (D) Multiplexing

4. Which communication mode does not require a common clock signal between the sender and receiver?

(A) Asynchronous (B) Synchronous (C) Full Duplex (D) Simplex

5. Which communication mode is more suitable for high-speed data transfer applications that require precise timing and synchronization?

(A) Asynchronous (B) Synchronous (C) Full Duplex (D) Simplex

6. Which type of communication is often used for sending important network control messages, such as routing updates and network management commands?

(A) Unicast (B) Multicast (C) Broadcast (D) AnyCast

7. Which type of communication is more bandwidth-efficient when transmitting data to multiple recipients located in different parts of the network?

(A) Unicast (B) Multicast (C) Broadcast (D) AnyCast

8. Which communication mode is commonly used in applications where data transfer rates are relatively low and timing precision is not critical?

(A) Asynchronous (B) Synchronous (C) Full Duplex (D) Simplex

9. Which of the following is an example of a point-to-point communication link?

(A) Sending an email to multiple recipients

(B) Making a phone call from one phone to another

(C) Broadcasting a message on social media

(D) Sharing a file on a network drive

10. Which of the following is an example of client-server communication?

(A) Sending an email to multiple recipients

(B) Broadcasting a message on social media

(C) Accessing a website hosted on a remote server

(D) Sharing files on a peer-to-peer network

11. What is the primary characteristic of a Peer-to-Peer (P2P) network?

(A) Centralized control (B) Client-server architecture

(C) Decentralized architecture (D) Hierarchical structure

12. Communication between a computer and a keyboard involves __________ transmission

(A) Simplex (B) Automatic (C) Full-duplex (D) Half-duplex

13. Communication channel is shared by all the machines on the network in

(A) Multicast network (B) Broadcast network

(C) Unicast network (D) None of these

14. In a peer to peer network, who controls the devices?

(A) First computer (B) Last computer

(C) No device controls (D) All devices control

15. Choose the correct statements about a Client-Server networking model.

(A) Server software is costly to acquire and renew

(B) Dedicated admin people are required to monitor and grant resources to normal computers or nodes. So this client-server network is highly secure.

(C) The hardware of a Server PC is costly

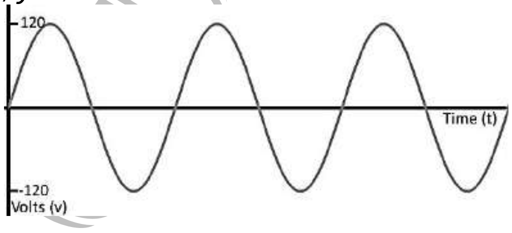
(D) All the above

1.4. SIGNAL

- A signal is an electromagnetic or electrical current that carries data from one system or network to another.
- In electronics, a signal is often a time-varying voltage that is also an electromagnetic wave carrying information, though it can take on other forms, such as current.
- There are two main types of signals used in electronics: analog and digital signals.

a.) Analog Signal

- Because a signal varies over time, it's helpful to plot it on a graph where time is plotted on the horizontal, *x*-axis, and voltage on the vertical, *y*-axis.
- Looking at a graph of a signal is usually the easiest way to identify if it's analog or digital; a time-versus-voltage graph of an analog signal should be **smooth** and **continuous**.



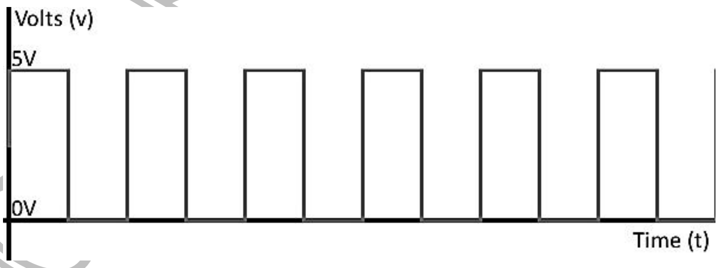
- While these signals may be limited to a **range** of maximum and minimum values, there are still an infinite number of possible values within that range.

For example:

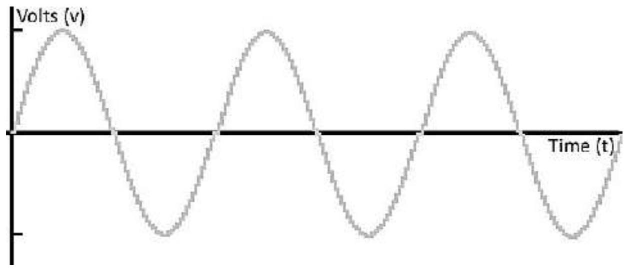
- The analog voltage coming out of your wall socket might be clamped between -120V and +120V, but, as you increase the resolution more and more, you discover an infinite number of values that the signal can actually be (like 64.4V, 64.42V, 64.424V, and infinite, increasingly precise values).

b.) Digital Signals

- Digital signals must have a finite set of possible values.
- The number of values in the set can be anywhere between two and a-very-large-number-that's-not-infinity.
- Most commonly digital signals will be one of **two values** -- like either 0V or 5V.
- Timing graphs of these signals look like **square waves**.



- Or a digital signal might be a discrete representation of an analog waveform. Viewed from afar, the wave function below may seem smooth and analog, but when you look closely there are tiny discrete **steps** as the signal tries to approximate values:



- That's the big difference between analog and digital waves. Analog waves are smooth and continuous, digital waves are stepping, square, and discrete.

c.) Key Differences:

- An analog signal is a continuous signal whereas Digital signals are time separated signals.
- Analog signal is denoted by sine waves while It is denoted by square waves
- Analog signal uses a continuous range of values that help you to represent information on the other hand digital signal uses discrete 0 and 1 to represent information.
- The analog signal bandwidth is low while the bandwidth of the digital signal is high.
- Analog instruments give considerable observational errors whereas Digital instruments never cause any kind of observational errors.
- Analog hardware never offers flexible implementation, but Digital hardware offers flexibility in implementation.
- Analog signals are suited for audio and video transmission while Digital signals are suited for Computing and digital electronics.

d.) Characteristics Of Analog Signal

Here, are essential characteristics of Analog Signal

- These types of electronic signals are time-varying
- Minimum and maximum values which is either positive or negative.
- It can be either periodic or non-periodic.
- Analog Signal works on continuous data.
- The accuracy of the analog signal is not high when compared to the digital signal.
- It helps you to measure natural or physical values.
- Analog signal output form is like Curve, Line, or Graph, so it may not be meaningful.

e.) Characteristics of Digital Signals

Here, are essential characteristics of Digital signals

- Digital signal are continuous signals
- This type of electronic l signals can be processed and transmitted better compared to analog signal.
- Digital signals are versatile, so it is widely used.
- The accuracy of the digital signal is better than that of the analog signal.

f.) Difference Between Analog and Digital Signal

Here are important differences between Analog and Digital Signal:

Analog	Digital
An analog signal is a continuous signal that represents physical measurements.	Digital signals are time separated signals which are generated using digital modulation.
It is denoted by sine waves	It is denoted by square waves

It uses a continuous range of values that help you to represent information.	Digital signal uses discrete 0 and 1 to represent information.
Temperature sensors, FM radio signals, Photocells, Light sensor, Resistive touch screen are examples of Analog signals.	Computers, CDs, DVDs are some examples of Digital signal.
The analog signal bandwidth is low	The digital signal bandwidth is high.
Analog signals are deteriorated by noise throughout transmission as well as write/read cycle.	Relatively a noise-immune system without deterioration during the transmission process and write/read cycle.
Analog hardware never offers flexible implementation.	Digital hardware offers flexibility in implementation.
It is suited for audio and video transmission.	It is suited for Computing and digital electronics.
Processing can be done in real-time and consumes lesser bandwidth compared to a digital signal.	It never gives a guarantee that digital signal processing can be performed in real time.
Analog instruments usually have s scale which is cramped at lower end and gives considerable observational errors.	Digital instruments never cause any kind of observational errors.
Analog signal doesn't offer any fixed range.	Digital signal has a finite number, i.e.,0 and 1.

g.) Advantages of Analog Signals

Here, are pros/benefits of Analog Signals

- Easier in processing
- Best suited for audio and video transmission.
- It has a low cost and is portable.
- It has a much higher density so that it can present more refined information.
- Not necessary to buy a new graphics board.
- Uses less bandwidth than digital sounds
- Provide more accurate representation of a sound
- It is the natural form of a sound.

h.) Advantages of Digital Signals

Here, are pros/advantages of Digital Signals:

- Digital data can be easily compressed.

- Any information in the digital form can be encrypted.
- Equipment that uses digital signals is more common and less expensive.
- Digital signal makes running instruments free from observation errors like parallax and approximation errors.
- A lot of editing tools are available
- You can edit the sound without altering the original copy
- Easy to transmit the data over networks

i.) Disadvantages of Analog Signals

Here are cons/drawback of Analog Signals:

- Analog tends to have a lower quality signal than digital.
- The cables are sensitive to external influences.
- The cost of the Analog wire is high and not easily portable.
- Low availability of models with digital interfaces.
- Recording analog sound on tape is quite expensive if the tape is damaged
- It offers limitations in editing
- Tape is becoming hard to find
- It is quite difficult to synchronize analog sound
- Quality is easily lost
- Data can become corrupted
- Plenty of recording devices and formats which can become confusing to store a digital signal
- Digital sounds can cut an analog sound wave which means that you can't get a perfect reproduction of a sound
- Offers poor multi-user interfaces

j.) Disadvantage of Digital Signals

- Sampling may cause loss of information.
- A/D and D/A demands mixed-signal hardware
- Processor speed is limited
- Develop quantization and round-off errors
- It requires greater bandwidth
- Systems and processing is more complex.

1.4.1. DIGITAL AND ANALOG TRANSMISSION

- Digital and analog transmission are two methods used to transmit data over communication channels. These methods differ in how they represent and transmit data signals. Here's a comparison between digital and analog transmission:

a. Digital Transmission:

- **Representation**: Digital transmission uses discrete, binary signals (0s and 1s) to represent data. Each binary digit (bit) represents a specific voltage level or symbol.

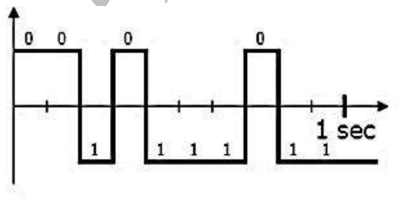
- **Signal Characteristics:** Digital signals are characterized by distinct voltage levels or symbols, typically represented by square waveforms. These signals have well-defined thresholds for interpreting 0s and 1s.
- **Noise Immunity:** Digital signals are less susceptible to noise and interference compared to analog signals. They can be regenerated and cleaned up using repeaters and error correction techniques, resulting in improved signal integrity.
- **Bandwidth Efficiency:** Digital transmission allows for higher data transmission rates and greater bandwidth efficiency compared to analog transmission. Multiple digital signals can be transmitted simultaneously over the same communication channel using techniques like multiplexing.
- **Examples:** Ethernet, Fiber Optic Communication, Digital Subscriber Line (DSL), Satellite Communication.

b. Analog Transmission:

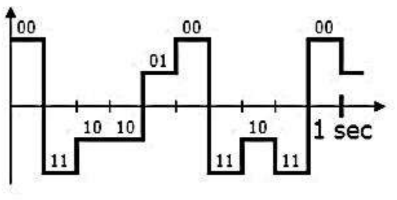
- **Representation:** Analog transmission uses continuous, varying signals to represent data. These signals can take on an infinite number of values within a specified range.
- **Signal Characteristics:** Analog signals have continuously varying voltage levels or waveforms that represent the original data. They can be affected by noise, distortion, and attenuation during transmission.
- **Noise Sensitivity:** Analog signals are more susceptible to noise and distortion compared to digital signals. Noise and interference can degrade the quality of the signal and result in errors during data transmission.
- **Bandwidth Usage:** Analog transmission typically requires more bandwidth to transmit the same amount of information compared to digital transmission. This limits the data transmission rates and efficiency of analog communication systems.
- **Examples:** Analog Telephone Systems (POTS), AM/FM Radio Broadcasting, Analog Television (TV) Broadcasting.
- In summary, digital transmission offers advantages such as noise immunity, higher data rates, and greater bandwidth efficiency, making it the preferred choice for modern communication systems. However, analog transmission is still used in certain applications where continuous signals are required or legacy systems are in place.

1.4.2. BIT RATE AND BAUD RATE

- Bit rate and baud rate are not always the same.
- The bit rate is the number of bits transmitted per second,
- The baud rate is the number of signal units transmitted per second and one signal unit is able to represent one or more bits.



Baud = 10
Bit rate = 10 bps



Baud = 10
Bit rate = 20 bps

- Therefore, baud rate is always less than or equal to the bit rate but never greater.

- **Bit Rate:** The number of bits transmitted per second
  - Bit rate = baud rate * number of bits per single unit
  - Ex: if 10 baud/Sec and there are 2 bits/baud what is bit rate?
  - 10*2=20 bits/Sec
- **Baud Rate:** Number of Signal unit transmitted per second.
  - Baud rate= bit rate / number of bits per single unit
  - Ex: if 20 bits/Sec and the 2 bits per baud what is baud rate?
  - 20/2=10 baud/Sec

Example:

1. If there are 4 bits per baud and there are 40 pits transferred per second , what will be the baud rate? 40/4=10 Baud/sec
2. If there are 10 baud per second and each baud contains 4 bit then what is bit rate? 10*4=40 bps
3. If there are 10 baud per second and the bit rate is 40 bps then how many signal elements are there? 40/10=4 bits/baud.

Example:

1.) An analog signal has a bit rate of 8000 bps and a baud rate of 1000. Then analog signal has _____ signal elements and carry _____ data elements in each signal.

(A) 256, 8 bits (B) 128, 4 bits (C) 256, 4 bits (D) 128, 8 bits

Answer: (A)

Explanation:

- Analog signal has a bit rate of 8000 bps and a baud rate of 1000. So, each signal will clearly carry bit rate / baud rate bits. i.e. 8000 / 1000 = 8 bits and 2^8 = 256 signal. So, option (A) is correct.

CHAPTER 1.4 MCQ

1. The speech signal is obtained after

(A) Analog to digital conversion

(B) Digital to analog conversion

(C) Modulation (D) Quantization

2. What type of data is typically represented by analog signals?

(A) Text documents

(B) Images and graphics

(C) Audio and video

(D) Binary code

3. What type of data is typically represented by digital signals?

(A) Analog sound waves

(B) Continuous video streams

(C) Text documents and numerical data

(D) Analog voltage levels

4. Which of the following statements is true regarding noise immunity?

(A) Analog signals are more immune to noise than digital signals

(B) Digital signals are more immune to noise than analog signals

(C) Analog and digital signals have equal immunity to noise

(D) Noise does not affect either analog or digital signals

5. Before data can be transmitted, they must be transformed to ________.

(A) periodic signals

(B) electromagnetic signals

(C) aperiodic signals

(D) low-frequency sine waves

6. A periodic signal completes one cycle in 0.001 s. What is the frequency?

(A) 1 Hz (B) 100 Hz

(C) 1 KHz (D) 1 MHz

7. In a frequency-domain plot, the horizontal axis measures the ________.

(A) peak amplitude (B) frequency

(C) phase (D) slope

8. If the bandwidth of a signal is 5 KHz and the lowest frequency is 52 KHz, what is the highest frequency?

(A) 5 KHz (B) 10 KHz

(C) 47 KHz (D) 57 KHz

9. What is the bandwidth of a signal that ranges from 1 MHz to 4 MHz?

(A) 4 MHz (B) 1 KHz

(C) 3 MHz (D) None of the above

10. As frequency increases, the period ________.

(A) decreases (B) increases

(C) remains the same (D) doubles

11. ________ is a type of transmission impairment in which the signal loses strength due to the resistance of the transmission medium.

(A) Attenuation (B) Distortion

(C) Noise (D) Decibel

12. ________ is a type of transmission impairment in which the signal loses strength due to the different propagation speeds of each frequency that makes up the signal.

(A) Attenuation (B) Distortion

(C) Noise (D) Decibel

13. ________ is a type of transmission impairment in which an outside source such as crosstalk corrupts a signal.

(A) Attenuation (B) Distortion

(C) Noise (D) Decibel

14. A _________ sine wave is not useful in data communications; we need to send a _______ signal.

(A) composite; single-frequency

(B) single-frequency; composite

(C) single-frequency; double-frequency

(D) none of the above

15. Frequency and period are ______

(A) inverse of each other

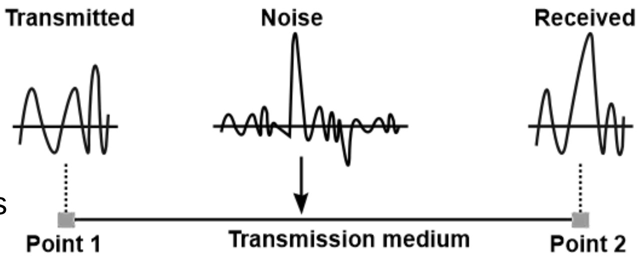
(B) proportional to each other

(C) the same

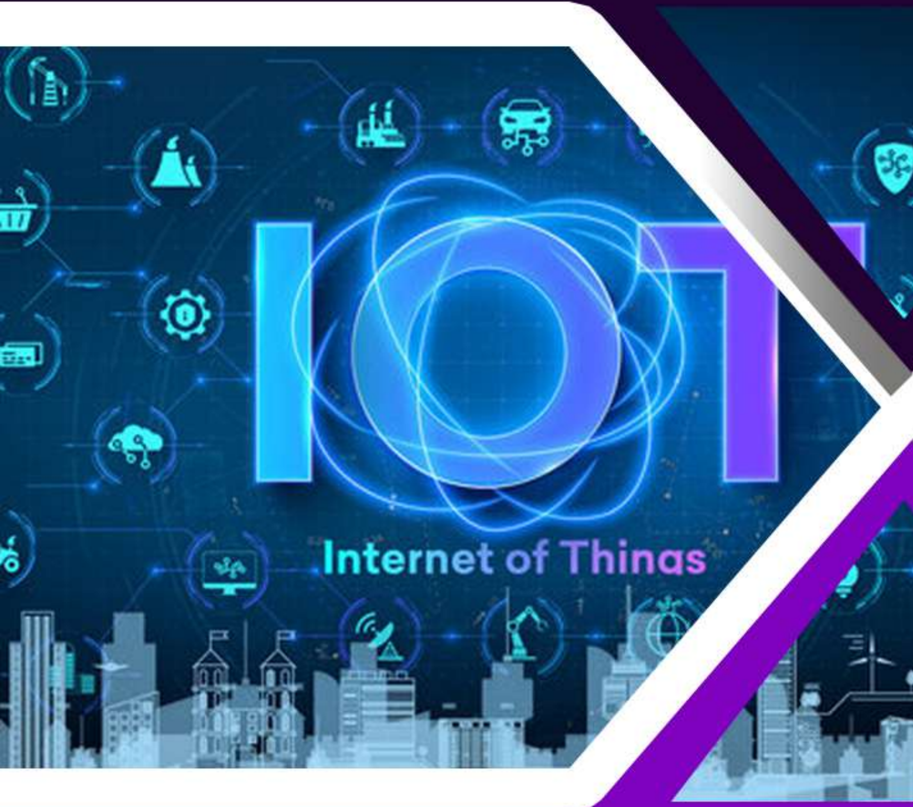
(D) none of the above

1.5. NOISE

- In computer networks, noise refers to any unwanted or random interference that disrupts the transmission of data between devices. It can occur in both wired and wireless communication channels and can arise from various sources, including electromagnetic interference, signal attenuation, crosstalk, and environmental factors.



COLLEGE TRB
Computer Science
2024-2025



TEACHER'S CARE PUBLICATION
38/23, Vaigundaperumal Koil Street,Kancheepuram- 631502
Mobile : 95665 35080, 9786269980 Land Line : 044-2723 5080

Your Success is Our Goal...

VOLUME-1

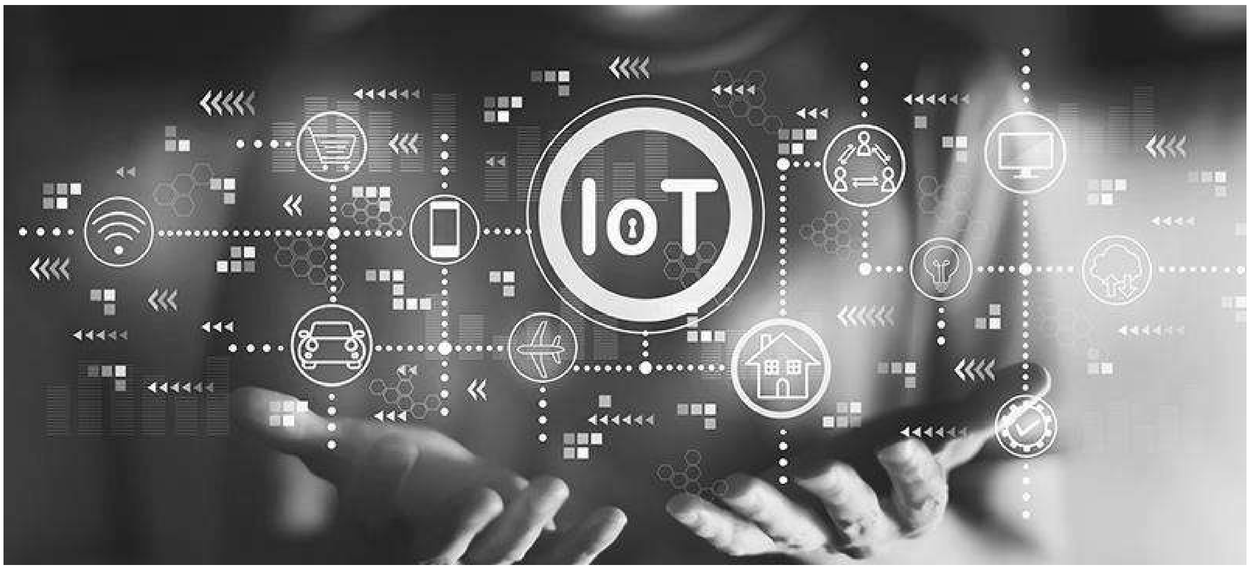
UNIT-10
Internet of Things

TEACHER'S CARE ACADEMY

KANCHIPURAM

UNIT – X

Internet of Things



COMPETITIVE EXAM

FOR

CLG-TRB-2024 – 2025

UNIT-10:

Internet of Things

UNIT-8: DATA COMMUNICATION AND COMPUTER NETWORKS

SYLLABUS

Data Communication: Components of a Data Communication System, Simplex, Half- Duplex and Duplex Modes of Communication; Analog and Digital Signals; Noiseless and Noisy Channels; Bandwidth, Throughput and Latency; Digital and Analog Transmission; Data Encoding and Modulation Techniques; Broadband and Baseband Transmission; Multiplexing, Transmission Media, Transmission Errors, Error Handling Mechanisms.

Computer Networks: Network Topologies, Local Area Networks, Metropolitan Area Networks, Wide Area Network, Wireless Networks, Internet.

Network Models: Layered Architecture, OSI Reference Model and its Protocols; TCP/IP Protocol Suite, Physical, Logical, Port and Specific Addresses; Switching Techniques.

Functions of OSI and TCP/IP Layers: Framing, Error Detection and Correction; Flow and Error Control; Sliding Window Protocol, HDLC, Multiple Access – CSMA/CD, CSMA/CA, Reservation, Polling, Token Passing, FDMA, CDMA, TDMA, Network Devices, Backbone Networks, Virtual LANs.

IPv4 Structure and Address Space; Classful and Classless Addressing; Datagram, Fragmentation and Checksum; **IPv6** Packet Format, Mapping Logical to Physical Address (ARP), Direct and Indirect Network Layer Delivery; Routing Algorithms, TCP, UDP and SCTP Protocols; Flow Control, Error Control and Congestion Control in TCP and SCTP.

World Wide Web (WWW): Uniform Resource Locator (URL), Domain Name Service (DNS), Resolution–Mapping Names to Addresses and Addresses to Names; Electronic Mail Architecture, SMTP, POP and IMAP; TELNET and FTP.

Network Security: Malwares, Cryptography and Steganography; Secret-Key Algorithms, Public-Key Algorithms, Digital Signature, Virtual Private Networks, Firewalls.

Mobile Technology: GSM and CDMA; Services and Architecture of GSM and Mobile Computing; Middleware and Gateway for Mobile Computing; Mobile IP and Mobile Communication Protocol; Communication Satellites, Wireless Networks and Topologies; Cellular Topology, Mobile Adhoc Networks, Wireless Transmission and Wireless LANs; Wireless Geolocation Systems, GPRS and SMS.

Cloud Computing and IoT: SaaS, PaaS, IaaS, Public and Private Cloud; Virtualization, Virtual Server, Cloud Storage, Database Storage, Resource Management, Service Level Agreement, Basics of IoT.

FUNDAMENTALS OF IoT

PREFACE

10.2.a Introduction to Internet of Things

The Internet of things (IoT) is the inter-networking of physical devices, vehicles (also referred to as “connected devices” and “smart devices”), buildings, and other items embedded with electronics, software, sensors, actuators, and network connectivity which enable these objects to collect and exchange data.

10.2.b. Characteristics:

- ✓ **Things-related services:** The IoT is capable of providing thing-related services within the constraints of things, such as privacy protection and semantic consistency between physical things and their associated virtual things
- ✓ **Connectivity:** Things in I.O.T. should be connected to the infrastructure, without connection nothing makes sense.
- ✓ **Intelligence:** Extraction of knowledge from the generated data is important, sensor generate data and this data and this data should be interpreted properly.
- ✓ **Scalability:** The no. of things getting connected to the I.O.T. infrastructure is increased day by day. Hence, an IOT setup shall be able to handle the massive expansion.
- ✓ **Unique Identity:** Each IOT device has an I.P. address. This identity is helpful in tracking the equipment and at times to query its status.
- ✓ **Dynamic and Self-Adapting:** The IOT device must dynamically adopt itself to the changing context. Assume a camera meant for surveillance, it may have to work in different conditions and at different light situations (morning, afternoon, night).
- ✓ **Heterogeneity:** The devices in the IoT are heterogeneous as based on different hardware platforms and networks. They can interact with other devices different networks.
- ✓ **Safety:** Having got all the things connected with the Internet possess a major threat, as our personal data is also there and it can be tampered with, if proper safety measures are not taken.

10.2.c. Application areas of IoT:

- ✓ **Smart Home:** The smart home is one of the most popular applications of IoT. The cost of
- ✓ owning a house is the biggest expense in a homeowner’s life. Smart homes are promised to
- ✓ save the time, money and energy.
- ✓ **Smart cities:** The smart city is another powerful application of IoT. It includes smart surveillance, environment monitoring, automated transformation, urban security, smart traffic management, water distribution, smart healthcare etc.

- ✓ **Wearables:** Wearables are devices that have sensors and software installed which can collect data about the user which can be later used to get the insights about the user. They must be energy efficient and small sized.
- ✓ **Connected cars:** A connected car is able to optimize its own operation, maintenance as well as passenger’s comfort using sensors and internet connectivity.
- ✓ **Smart retail:** Retailers can enhance the in-store experience of the customers using IoT. The shopkeeper can also know which items are frequently bought together using IoT devices.
- ✓ **Smart healthcare:** People can wear the IoT devices which will collect data about user's health. This will help users to analyze themselves and follow tailor-made techniques to combat illness. The doctor also doesn't have to visit the patients in order to treat them.

10.2.d IoT Categories

IOT can be classified into two categories:

1. **Consumer IoT(CIOT):** The Consumer IoT refers to the billions of physical personal devices, such as smartphones, wearables, fashion items and the growing number of smart home appliances, that are now connected to the internet, collecting and sharing data.

- ✓ A Consumer IoT network typically entails few consumer devices, each of which has a limited lifetime of several years.
- ✓ The common connectivity used in this kind of solutions are Bluetooth, WiFi, and ZigBee. These technologies offer short-range communication, suitable for applications deployed in limited spaces such as houses, or small offices.

2. **industrial internet of things (IIoT):** It refers to interconnected sensors, instruments, and other devices networked together with computers' industrial applications, including manufacturing and energy management. This connectivity allows for data collection,

exchange, and analysis, potentially facilitating improvements in productivity and efficiency as well as another economic ben.

10.2.e Baseline technologies

There are various baseline technologies that are very closely related to IOT, They include: Machine-to-Machine (M2M), Cyber-Physical Systems (CPS), Web Of Things (WOT)

a) Machine-to-Machine (M2M):

- Machine-to-Machine (M2M) refers to networking of machines (or devices) for the purpose of remote monitoring and control and data exchange.
- An M2M area network comprises of machines (or M2M nodes) which have embedded network modules for sensing, actuation and communicating various communication protocols can be used for M2M LAN such as ZigBee, Bluetooth, M- bus, Wireless M-Bus etc., These protocols provide connectivity between M2M nodes within an M2M area network.
- The communication network provides connectivity to remote M2M area networks.

- The communication network provides connectivity to remote M2M area network.
- The communication network can use either wired or wireless network (IP based).
- While the M2M are networks use either proprietary or non-IP based communication protocols, the communication network uses IP-based network. Since non-IP based protocols are used within M2M area network, the M2M nodes within one network cannot communicate with nodes in an external network.
- To enable the communication between remote M2M are network, M2M gateways are used

b) Cyber-Physical systems:

- ✓ Cyber-Physical Systems (CPS) are integrations of computation, networking, and physical processes. Embedded computers and networks monitor and control the physical processes, with feedback loops where physical processes affect computations and vice versa.
- ✓ In cyber-physical systems, physical and software components are deeply intertwined, able to operate on different spatial and temporal scales, exhibit multiple and distinct behavioural modalities, and interact with each other in ways that change with context.

c) Web of Things: web of things is a term used to describe approaches, software architectural style of programming patterns that allow real world objects to be part of WWW. The major portion of the WoT specification is the Thing Description. Thing is an abstract representation of a physical or virtual entity. A Thing Description includes the metadata and interfaces of a Thing in a standardized way, with the aim to make the Thing able to communicate with other Things in a heterogeneous world.

10.2.e SENSOR

Sensor is a device used for the conversion of physical events or characteristics into the electrical signals. This is a hardware device that takes the input from environment and gives to the system by converting it.

For example, a thermometer takes the temperature as physical characteristic and then converts it into electrical signals for the system.

Characteristics of Sensors

1. Range: It is the minimum and maximum value of physical variable that the sensor can sense or measure. For example, a Resistance Temperature Detector (RTD) for the measurement of temperature has a range of -200 to 800oC.

2. Span: It is the difference between the maximum and minimum values of input. In above example, the span of RTD is 800 – (-200) = 1000oC.

3. Accuracy: The error in measurement is specified in terms of accuracy. It is defined as the difference between measured value and true value. It is defined in terms of % of full scale or % of reading.

4. Precision: It is defined as the closeness among a set of values. It is different from accuracy.

5.Linearity: Linearity is the maximum deviation between the measured values of a sensor from ideal curve.

6.Hysteresis: It is the difference in output when input is varied in two ways- increasing and decreasing.

7. Resolution: It is the minimum change in input that can be sensed by the sensor.

8. Reproducibility: It is defined as the ability of sensor to produce the same output when same input is applied.

9. Repeatability: It is defined as the ability of sensor to produce the same output every time when the same input is applied and all the physical and measurement conditions kept the same including the operator, instrument, ambient conditions etc.

10. Response Time: It is generally expressed as the time at which the output reaches a certain percentage (for instance, 95%) of its final value, in response to a step change of the input.

Classification of sensors:

Sensors based on the power requirement sensor is classified into two types: Active Sensors, Passive Sensors.

- ✓ **Active Sensors:** Does not need any external energy source but directly generates an electric signal in response to the external.
- ✓ Example: Thermocouple, Photodiode, Piezoelectric sensor.
- ✓ **Passive Sensors:** The sensors require external power called excitation signal. Sensors modify the excitation signal to provide output.
- ✓ Example: Strain gauge.
- ✓ Sensors based on output sensor is classified into two types: Analog Sensors, Digital Sensors.
- ✓ **Analog Sensors**
  - Analog Sensors produces a continuous output signal or voltage which is generally proportional to the quantity being measured.
  - Physical quantities such as Temperature, speed, Pressure, Displacement, Strain etc. are all analog quantities as they tend to be continuous in nature.
  - For example, the temperature of a liquid can be measured using a thermometer or thermocouple (e.g. in geysers) which continuously responds to temperature changes as the liquid is heated up or cooled down.
- ✓ **Digital Sensors**
  - Digital Sensors produce discrete output voltages that are a digital representation of the quantity being measured.
  - Digital sensors produce a binary output signal in the form of a logic "1" or a logic "0" ("ON" or "OFF).
  - Digital signal only produces discrete (non-continuous) values, which may be output as a signal "bit" (serial transmission), or by combing the bits to produce a signal "byte" output (parallel transmission).

Based on type of data measured sensor is classified into two types: Scalar Sensors and Vector Sensors.

✓ **Scalar Sensors**

- Scalar Sensors produce output signal or voltage which generally proportional to the magnitude of the quantity being measured.
- Physical quantities such as temperature, color, pressure, strain, etc. are all scalar quantities as only their magnitude is sufficient to convey an information.
- For example, the temperature of a room can be measured using thermometer or thermocouple, which responds to temperature changes irrespective of the orientation of the sensor or its direction.

✓ **Vector Sensors**

- Vector Sensors produce output signal or voltage which generally proportional to the magnitude, direction, as well as the orientation of the quantity being measured.
- Physical quantities such as sound, image, velocity, acceleration, orientation, etc. are all vector quantities, as only their magnitude is not sufficient to convey the complete information.
- For example, the acceleration of a body can be measured using an accelerometer, which gives the components of acceleration of the body with respect to the x,y,z coordinate axes.

10.2.f ACTUATOR

Actuator is a device that converts the electrical signals into the physical events or characteristics. It takes the input from the system and gives output to the environment. For example, motors and heaters are some of the commonly used actuators.

Types of Actuators

1. Hydraulic Actuators: Hydraulic actuators operate by the use of a fluid-filled cylinder with a piston suspended at the centre. Commonly, hydraulic actuators produce linear movements, and a spring is attached to one end as a part of the return motion. These actuators are widely seen in exercise equipment such as steppers or car transport carriers.

2. Pneumatic Actuators: Pneumatic actuators are one of the most reliable options for machine motion. They use pressurized gases to create mechanical movement. Many companies prefer pneumatic-powered actuators because they can make very precise motions, especially when starting and stopping a machine. Examples of equipment that uses pneumatic actuators include: Bus brakes, Exercise machines, Vane motors, Pressure sensors

3.Electric Actuators: Electrical actuators, as you may have guessed, require electricity to work. Well-known examples include electric cars, manufacturing machinery, and robotics

equipment. Similar to pneumatic actuators, they also create precise motion as the flow of electrical power is constant.

4.Thermal and Magnetic Actuators: Thermal and magnetic actuators usually consist of shape memory alloys that can be heated to produce movement. The motion of thermal or magnetic actuators often comes from the Joule effect, but it can also occur when a coil is placed in a static magnetic field. The magnetic field causes constant motion called the Laplace-Lorentz force. Most thermal and magnetic actuators can produce a wide and powerful range of motion while remaining lightweight.

5.Mechanical Actuators: Some actuators are mostly mechanical, such as pulleys or rack and pinion systems. Another mechanical force is applied, such as pulling or pushing, and the actuator will leverage that single movement to produce the desired results. For instance, turning a single gear on a set of rack and pinions can mobilize an object from point A to point B. The tugging movement applied on the pulley can bring the other side upwards or towards the desired location.

6. Soft Actuators: Soft actuators (e.g., polymer based) are designed to handle fragile objects like fruit harvesting in agriculture or manipulating the internal organs in biomedicine.

They typically address challenging tasks in robotics. Soft actuators produce flexible motion due to the integration of microscopic changes at the molecular level into a macroscopic deformation of the actuator materials.

10.2.f IOT COMPONENTS

Four fundamental components of IoT system, which tells us how IoT works.

i. Sensors/Devices

- First, sensors or devices help in collecting very minute data from the surrounding environment. All of this collected data can have various degrees of complexities ranging from a simple temperature monitoring sensor or a complex full video feed.
- A device can have multiple sensors that can bundle together to do more than just sense things. For example, our phone is a device that has multiple sensors such as GPS, accelerometer, camera but our phone does not simply sense things.

ii. Connectivity

- Next, that collected data is sent to a cloud infrastructure but it needs a medium for transport. The sensors can be connected to the cloud through various mediums of communication and
- transports such as cellular networks, satellite networks, Wi-Fi, Bluetooth, wide-area networks (WAN), low power wide area network and many more.

iii. Data Processing

- Once the data is collected and it gets to the cloud, the software performs processing on the acquired data.
- This can range from something very simple, such as checking that the temperature reading on devices such as AC or heaters is within an acceptable range. It can sometimes also be very complex, such as identifying objects (such as intruders in your house) using computer vision on video.

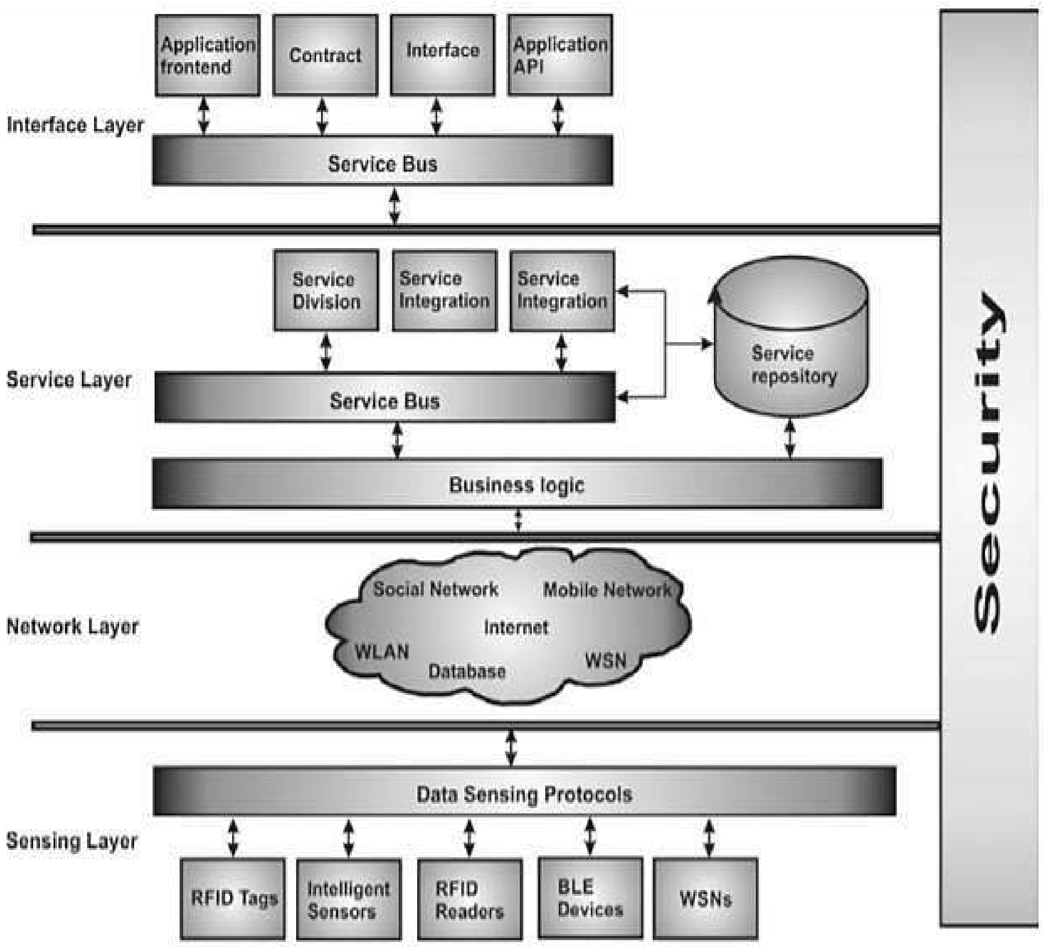
iv. User Interface

- Next, the information made available to the end-user in some way. This can achieve by triggering alarms on their phones or notifying through texts or emails.

- Also, a user sometimes might also have an interface through which they can actively check in on their IOT system. For example, a user has a camera installed in his house, he might want to check the video recordings and all the feeds through a web server.

10.2.g Service Oriented Architecture of IoT

SOA can also use to support IoT as a main contributing technology in devices or heterogeneous systems.



Service-Oriented Architecture for IoT technologies.

1.Sensing Layer: IoT can be defined as a worldwide interconnected network, where things or devises are controlled remotely. Interconnected things or devices are become easier, as more and more things are furnished with sensors and RFID technologies.

2.Networking Layer: Networking Layer is responsible to connect all device or things together so that they can able to share the information with each other over the Internet. Moreover, network layer also collects data and information from the present IT infrastructure for example ICT systems, power grids, business systems, healthcare systems, and transportation systems.

3. Service Layer: This layer depends upon the technology used on the middleware layer which is responsible for functionalities incorporate between applications and services in IoT. This middleware technology also provides a cost-effective and efficient platform for IoT and this platform including software and hardware components which can be reused when needed.

4. Interface Layer: The core responsibility of the interface layer has also simplified the interconnection and management of things. Interface specific profile can be defined as the subset of services that support interaction with the application used in a network

10.2.h Challenges for IoT

1. Security: Security is the most significant challenge for the IoT. Increasing the number of connected devices increases the opportunity to exploit security vulnerabilities, as do poorly designed devices, which can expose user data to theft by leaving data streams inadequately

protected and in some cases people's health and safety can be put at risk.

2. Privacy: The IoT creates unique challenges to privacy, many that go beyond the data privacy issues that currently exist. Much of this stems from integrating devices into our environments without us consciously using them. This is becoming more prevalent in consumer devices, such as tracking devices for phones and cars as well as smart televisions.

3. Scalability: Billions of internet-enabled devices get connected in a huge network, large volumes of data are needed to be processed. The system that stores, analyses the data from these IoT devices needs to be scalable.

4. Interoperability: Technological standards in most areas are still fragmented. These technologies need to be converged. Which would help us in establishing a common framework and the standard for the IoT devices. As the standardization process is still

lacking, interoperability of IoT with legacy devices should be considered critical. This lack of interoperability is preventing us to move towards the vision of truly connected everyday interoperable smart objects.

5. Bandwidth: Connectivity is a bigger challenge to the IoT than you might expect. As the size of the IoT market grows exponentially, some experts are concerned that bandwidth- intensive IoT applications such as video streaming will soon struggle for space on the IoT's

current server-client model.

6. Standards: Lack of standards and documented best practices have a greater impact than just limiting the potential of IoT devices. Without standards to guide manufacturers, developers sometimes design products that operate in disruptive ways on the Internet without much regard to their impact. If poorly designed and configured, such devices can have negative consequences for the networking resources they connect to and the broader Internet.

7. Regulation: The lack of strong IoT regulations is a big part of why the IoT remains a severe security risk, and the problem is likely to get worse as the potential attack surface

expands to include ever more crucial devices. When medical devices, cars and children's toys are all connected to the Internet, it's not hard to imagine many potential disaster scenarios unfolding in the absence of sufficient regulation.

10.2.i IOT Networking

- ✓ **IoT Node:** These are machines, things or computers Connected to other nodes inside a LAN
- ✓ via the IoT LAN, May be sometimes connected to the internet through a WAN directly
- ✓ **IoT LAN:** It is Local, Short range Comm, May or may not connect to Internet, Building or
- ✓ Organization wide
- ✓ **IoT WAN:** Connection of various network segments, Organizationally and geographically wide, Connects to the internet
- ✓ **IoT Gateway:** A router connecting the IoT LAN to a WAN to the Internet, can implement several LAN and WAN, Forwards packets between LAN and WAN on the IP layer
- ✓ **IoT Proxy:** Performs active application layer functions between IoT nodes and other entities

➢ **Gateway Prefix Allotment:**

- One of the strategies of address conservation in IoT is to use local addresses which exist uniquely within the domain of the gateway. These are represented by the circles in this slide.
- The network connected to the internet has routers with their set of addresses and ranges.
- These routers have multiple gateways connected to them which can forward packets from the nodes, to the Internet, only via these routers. These routers assign prefixes to gateways under them, so that the gateways can be identified with them.

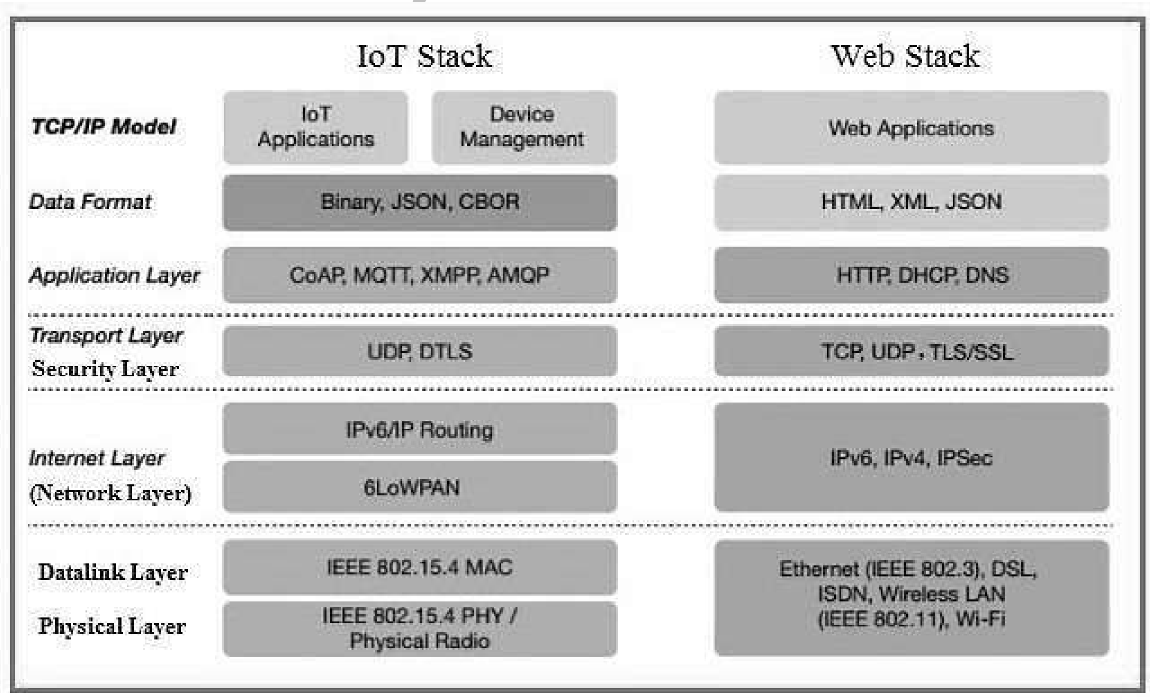
➢ **Impact of Mobility on Addressing**

- o The network prefix changes from 1 to 2 due to movement, making the IoT LAN safe from changes due to movements.
- o IoT gateway WAN address changes without change in LAN address. This is achieved using ULA.
- o The gateways assigned with prefixes, which are attached to a remote anchor point by using various protocols such as Mobile IPv6, and are immune to changes of network prefixes.
- o This is achieved using LU. The address of the nodes within the gateways remains unchanged as the gateways provide them with locally unique address and the change

➢ **in gateway's network prefix doesn't affect them.**

- o Sometimes, there is a need for the nodes to communicate directly to the internet. This is achieved by tunnelling, where the nodes communicate to a remote anchor point instead of channelling their packets through the router which is achieved by using tunnelling protocols such as IKEv2:internet key exchange version 2

- **Multihoming**
  - Multihoming is the practice of connecting a host or a computer network to more than one network. This can be done in order to increase reliability or performance or to reduce cost. There are several different ways to perform multihoming.
- **Host multihoming:** A single host may be connected to multiple networks. For example, a mobile phone might be simultaneously connected to a WiFi network and a 3G network, and a desktop computer might be connected to both a home network and a VPN. A multihomed host usually is assigned multiple addresses, one per connected network.
- **Classical multihoming:** In classical multihoming a network is connected to multiple providers, and uses its own range of addresses (typically from a Provider Independent (PI) range). The network's edge routers communicate with the providers using a dynamic routing protocol, typically BGP, which announces the network's address range to all providers. If one of the links fails, the dynamic routing protocol recognizes the failure within seconds or minutes, and reconfigures its routing tables to use the remaining links, transparently to the hosts.
- **Multihoming with multiple addresses:** In this approach, the network is connected to multiple providers, and assigned multiple address ranges, one for each provider. Hosts are assigned multiple addresses, one for each provider.
- **Deviation from regular Web**



	IoT Stack		Web Stack
TCP/IP Model	IoT Applications	Device Management	Web Applications
Data Format	Binary, JSON, CBOR		HTML, XML, JSON
Application Layer	CoAP, MQTT, XMPP, AMQP		HTTP, DHCP, DNS
Transport Layer Security Layer	UDP, DTLS		TCP, UDP, TLS/SSL
Internet Layer (Network Layer)	IPv6/IP Routing		IPv6, IPv4, IPSec
	6LoWPAN		
Datalink Layer	IEEE 802.15.4 MAC		Ethernet (IEEE 802.3), DSL, ISDN, Wireless LAN (IEEE 802.11), Wi-Fi
Physical Layer	IEEE 802.15.4 PHY / Physical Radio		

Features	IoT Stack	Web Stack
Function or application	It is used in constrained network having low power, low bandwidth and low memory requirements.	It is used in non-constrained network having no limits on power/BW/memory.
Size of data to be transported	tens of bytes	hundreds or thousands of bytes
Data format	It uses CBOR (Concise Binary Object Representation) format as IoT is used for tiny messages. CBOR is based on JSON though CBOR uses binary encoding while JSON uses text encoding.	It uses HTML, XML and JSON formats.
Application Layer	It uses CoAP protocol at application layer.	It uses HTTP protocol at application layer.
Transport layer	It uses UDP which is faster due to smaller header size compare to TCP. It is lighter protocol compare to TCP.	It uses TCP which is connection oriented and slower compare to UDP.
Security layer	It uses DTLS (Datagram Transport Layer Security) protocol for security.	It uses TLS/SSL protocols for the same.
Internet layer	It uses 6LoWPAN to convert large IPv6 packets into small size packets to be carried on wireless medium as per bluetooth, zigbee etc. standards. It does fragmentation and reassembly. It also does header compression to reduce packet size.	It does not require protocols like 6LoWPAN. Fragmentation and reassembly is taken care by transport layer (i.e. TCP) itself.
Datalink or MAC layer	It will have MAC layer as per IoT wireless technology used viz. bluetooth, zigbee, zwave etc. It takes care of medium access control and resource allocation and management.	It will have MAC layer as per LAN or WLAN or DSL or ISDN technologies.

Physical layer and Radio Frequency (RF) layer	It will have physical layer (baseband) as per IoT wireless technologies viz. bluetooth, zigbee, zwave etc. It uses frequencies as per cellular or indoor wireless technologies and country wide allocations for the same.	It will have PHY layer as per LAN or WLAN or DSL or ISDN technologies.

10.2.j IoT identification and Data protocols

IPv4: IP version four addresses are 32-bit integers which will be expressed in dotted decimal notation.

Example- 192.0.2.126 could be an IPv4 address.

Characteristics of IPv4

- IPv4 could be a 32-Bit IP Address.
- IPv4 could be a numeric address, and its bits are separated by a dot.
- The number of header fields is twelve and the length of the header field is twenty.
- It has Unicast, broadcast, and multicast style of addresses.
- IPv4 supports VLSM (Virtual Length Subnet Mask).
- IPv4 uses the Post Address Resolution Protocol to map to the MAC address.
- RIP may be a routing protocol supported by the routed daemon.
- Networks ought to be designed either manually or with DHCP.
- Packet fragmentation permits from routers and causing host.

IPv4 Datagram Header

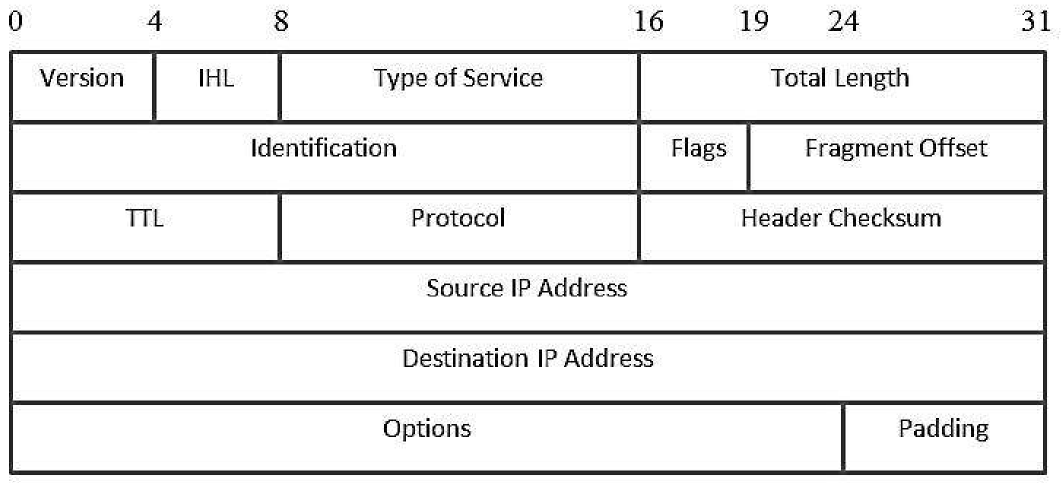


Fig: IPv4 Frame Format

- ✓ **Version:** This field indicates the version number of the IP packet so that the revised version can be distinguished from the previous version. The current IP version is 4.
- ✓ **Internet Header Length (IIHL):** It specifies the length of the IP header in unit 32 bits. In case of no option present in the IP header, IHL will have a value of 5. So, if the value of IHL is more than 5 then the length of the option field can be easily calculated.
- ✓ **Type of Service:** This field specifies the priority of the packets based on delay, throughput, reliability and cost requirements. Three bits are assigned for priority level and four bits for specific requirements (delay, throughput, reliability and cost).
- ✓ **Total Length:** This field specifies the number of bytes of the IP packet including header and data. As 16 bits are assigned to this field, the maximum length of the packet is 65635 bytes.
- ✓ **Identification:** The identification field is used to identify which packet a particular fragment belongs to so that fragments for different packets don't get mixed up.

Flags:

The flag field has three bits:

1. Unused bit
2. Don't fragment (DF) bit
3. More fragment (MF) bit

- ✓ **Fragment Offset:**The fragment offset field identifies the location of the fragment in a packet. The value measures the offset in a unit of 8 bytes, between the beginning of the packet to be fragmented and the beginning of the fragment.
- ✓ **Time to live (TTL):**This field is used to indicate the amount of time in seconds a packet is allowed to remain in the network.
- ✓ **Protocol:**This field specifies the protocol that is to receive the IP data at the destination host.
- ✓ **Header Checksum:**This field verifies the integrity of the header of the IP packet. The integrity of the data part is left to the upper layer protocols. The checksum is generated by the source and it is sent along with the frame header to the next router.
- ✓ **Source IP address & Destination IP address:**
- ✓ These two fields contain the IP addresses of the source and destination hosts respectively.
- ✓ **Options:**Options fields are rarely used to include special features such as security level, the route to be taken and time stamp at each router. It is used in RSVP.
- ✓ **Padding:**This field is used to make the header a multiple of 32-bit words**.**

10.2.k IPv6

- Internet Protocol version 6 (IPv6) is also known as Internet Protocol next generation (IPng). It also accommodates more feature to meet the global requirement of growing Internet.

- To allocate a sufficient number of network address, IPv6 allows 128 bits of IP address separated into 8 sections of 2 bytes each. Unlike IPv4 where the address is represented as dotted-decimal notation, IPv6 uses hexadecimal numbers and colon (“:”) is used as a delimiter between the sections.

Example: IPv6 address may be like this: FA20:B120: 6230:0000:0000: CE12:0006: ABDF

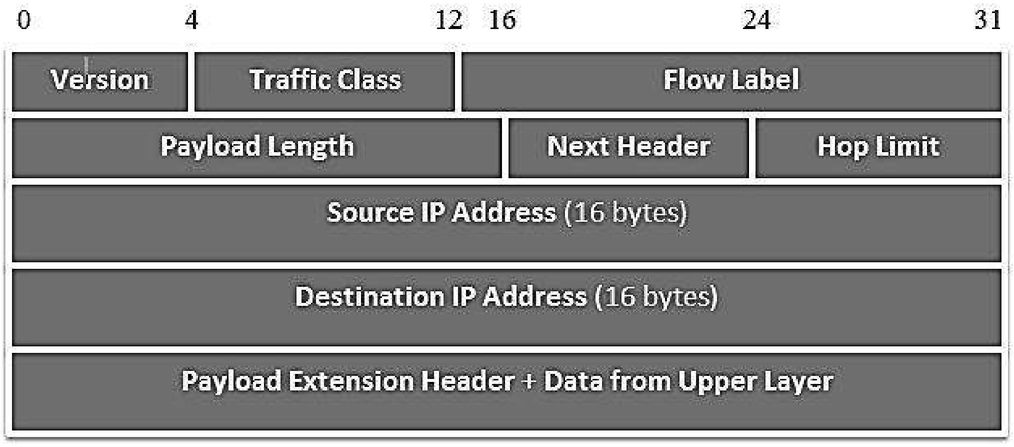

0
4
12
16
24
31
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source IP Address (16 bytes)
Destination IP Address (16 bytes)
Payload Extension Header + Data from Upper Layer

Fig: IPv6 Packet Format

- ✓ **Version:** This field is 4 bits long and it defines the version of the IP packet. The value of it for IPv6 is 6 and IPv4 its value is 4. During the transition period from IPv4 to IPv6, the routers will be able to distinguish the two versions of the IP packets.
- ✓ **Traffic Class:** This field is 20 bits long and it is used to distinguish between the different requirements for real-time delivery services.
- ✓ **Flow Label:** This field is 20 bits long and it is used to allow the source and destination nodes to set up a pseudo connection with particular properties and requirements. It is designed to provide special handling of a particular flow of data.
- ✓ **Payload Length:** It is of 2 bytes length and signifies the number of bytes that follow the 40 bytes base header. It is the length of the IP datagram excluding the base header.
- ✓ **Next Header:** This field is of 1 bye length and it defines one of the extension headers that follow the base header. The extension headers also contain this field to indicate the next header. if this is the last IP header then Next header field tells which of the transport protocols (TCP or UDP) the packet is to be passed.
- ✓ **Hop Limit:** This field contains 1 byte and it signifies the maximum number of hops a packet can travel. The time to live field in the IPv4 header did the same task, except that in IPv4 it was counted in time and in IPv6 it is counted in terms of the number of routers.
- ✓ **Source Address:** It is 16 bytes long and contains the IP address of the source machine to the network interface.
  - ✓ **Destination Address:** It is 16 bytes long and usually contains the IP address of the ultimate destination machine to the network interface. In case of specific routing, it may contain the IP address of the next router.