DICE
DEPARTMENT OF INTERDISCIPLINARY COURSES IN ENGINEERING

CHITKARA UNIVERSITY

Introduction To IoT – Unit 3

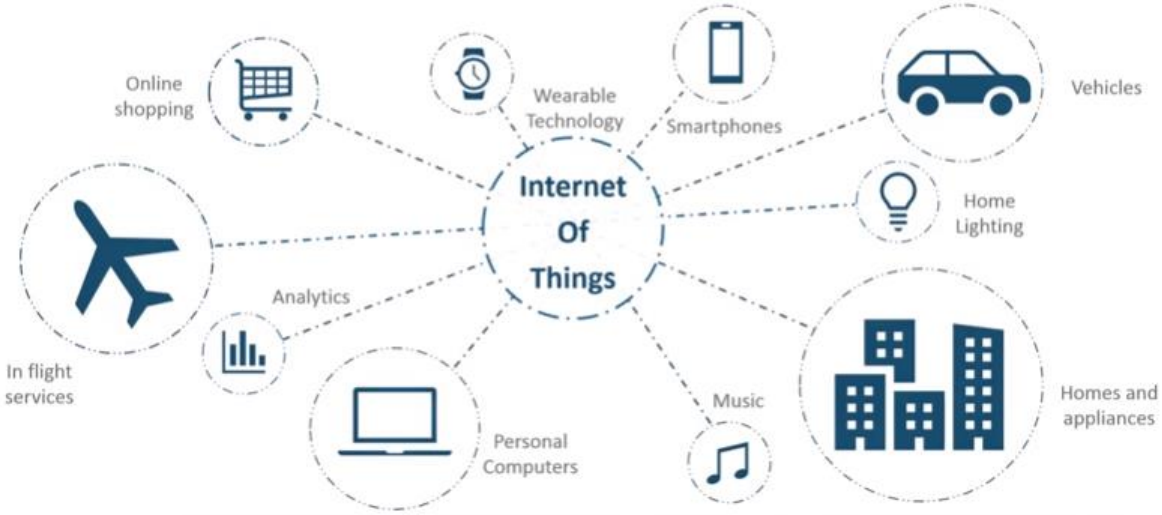
What is IoT

- IoT stands for Internet of Things.
- It refers to the interconnectedness of physical devices, such as appliances and vehicles, that are embedded with software, sensors, and connectivity which enables these objects to connect and exchange data.
- This technology allows for the collection and sharing of data from a vast network of devices, creating opportunities for more efficient and automated systems.

IoT is network of interconnected computing devices which are embedded in everyday objects, enabling them to send and receive data.

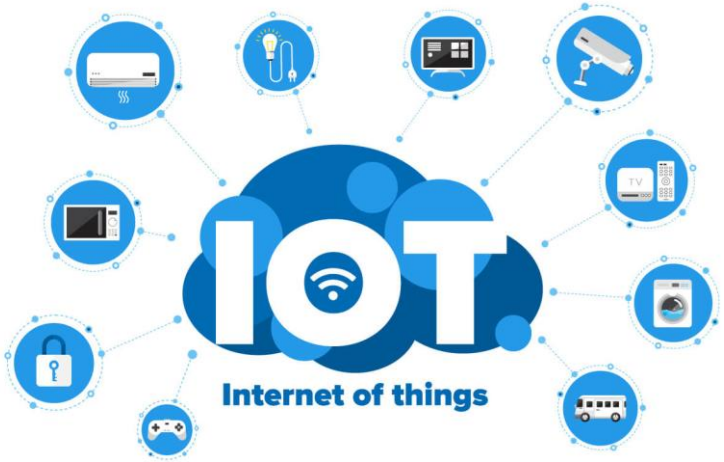
What is IoT

- Connecting everyday things embedded with electronics, software, and sensors to internet enabling to collect and exchange data without human interaction called as the Internet of Things (IoT).



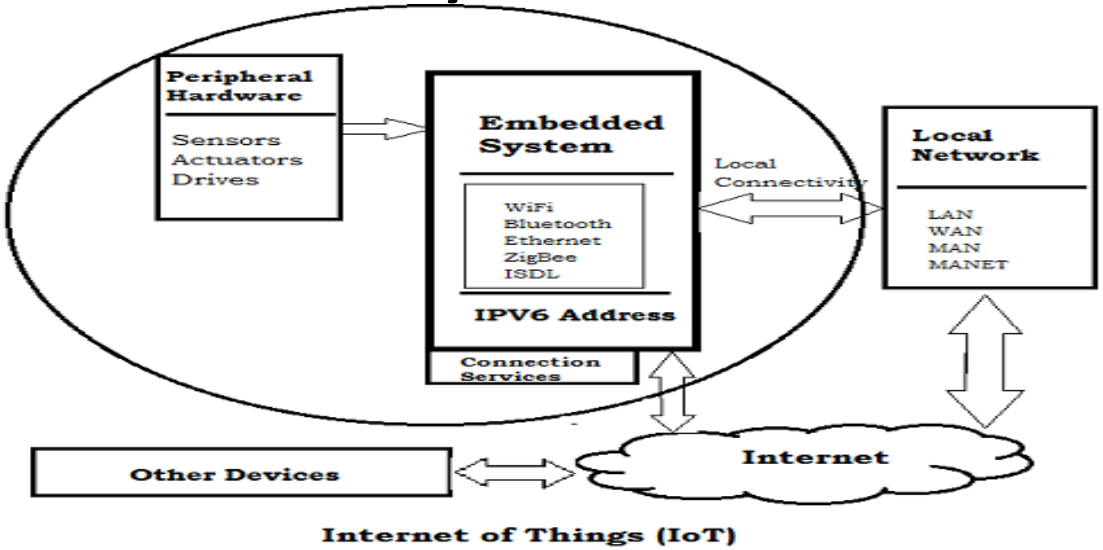
Thing?

A “Thing” in the context of IOT, is a entity or physical object that has unique identifier, an embedded system and ability to transfer data over the network.



These devices collect useful data with the help of various existing technologies and then autonomously flow the data between other devices

Embedded Systems in IoT

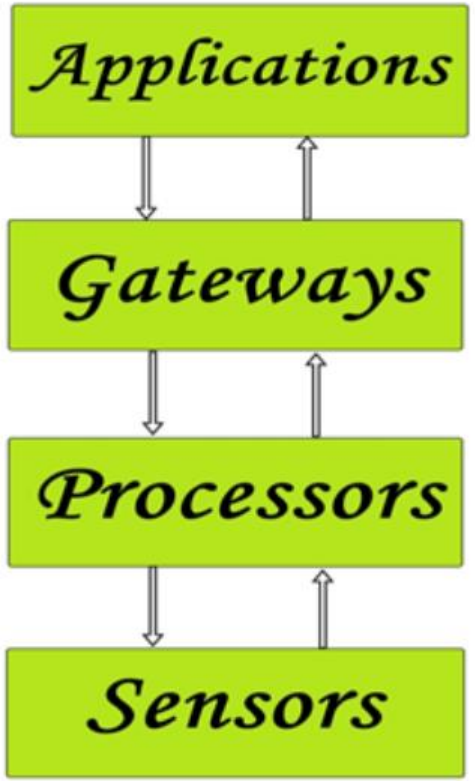


An embedded device system generally runs as a single application. these devices can connect through the internet connection, and able communicate through other network devices

Building Blocks of IoT

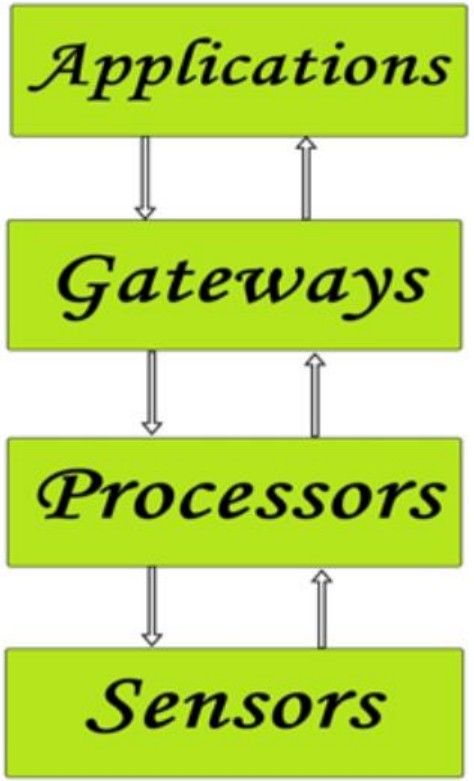
Four things form the basic building blocks of the IoT system –sensors, processors, gateways, and applications.

- **Sensors**
- These form the front end of the IoT devices. These are the so-called “Things” of the system. Their main purpose is to collect data from its surroundings (sensors) or give out data to their surroundings (actuators).
- These have to be uniquely identifiable devices with a unique IP address so that they can be easily identifiable over a large network.
- These have to be active in nature which means that they should be able to collect real-time data. These can either work on their own (autonomous in nature) or can be made to work by the user depending on their needs (user-controlled).
- Examples of sensors are gas sensor, water quality sensor, moisture sensor, etc.



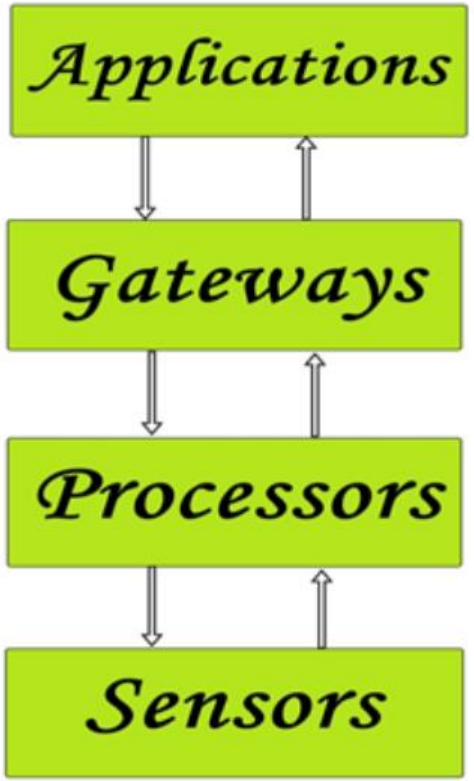
Building Blocks of IoT

- **Processors**
- Processors are the brain of the IoT system. Their main function is to process the data captured by the sensors and process them so as to extract valuable data from the enormous amount of raw data collected. In a word, we can say that it gives intelligence to the data.
- Processors mostly work on a real-time basis and can be easily controlled by applications. These are also responsible for securing the data – that is performing encryption and decryption of data.
- Embedded hardware devices, microcontrollers, etc are the ones that process the data because they have processors attached to it.



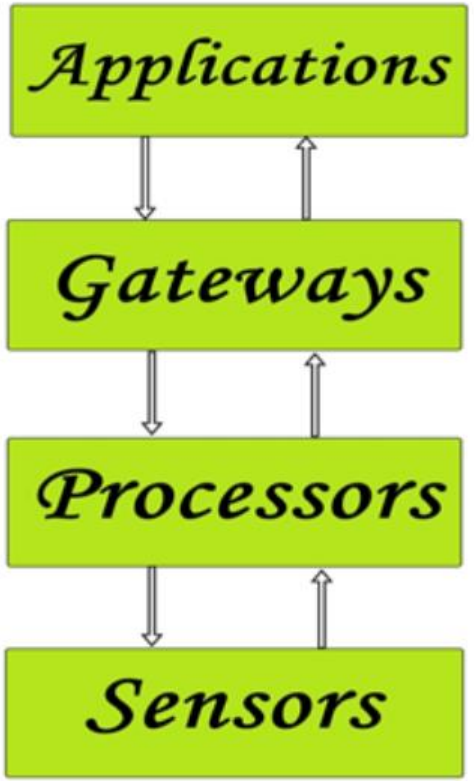
Building Blocks of IoT

- **Gateways**
- Gateways are responsible for routing the processed data and sending it to proper locations for its (data) proper utilization.
- In other words, we can say that the gateway helps in to and fro communication of the data. It provides network connectivity to the data. Network connectivity is essential for any IoT system to communicate.
- LAN, WAN, PAN, etc are examples of network gateways.



Building Blocks of IoT

- **Applications**
- Applications form another end of an IoT system. Applications are essential for the proper utilization of all the data collected.
- These cloud-based applications are responsible for rendering effective meaning to the data collected. Applications are controlled by users and are a delivery point of particular services.
- Examples of applications are home automation apps, security systems, industrial control hubs, etc.



Characteristics of IoT

1. Connectivity

Connectivity is an important requirement of the IoT infrastructure. Things of IoT should be connected to the IoT infrastructure. Anyone, anywhere, anytime can connect, this should be guaranteed at all times. For example, the connection between people through Internet devices like mobile phones, and other gadgets, also a connection between Internet devices such as routers, gateways, sensors, etc.

2. Intelligence and Identity

The extraction of knowledge from the generated data is very important. For example, a sensor generates data, but that data will only be useful if it is interpreted properly. Each IoT device has a unique identity. This identification is helpful in tracking the equipment and at times for querying its status.

3. Scalability

The number of elements connected to the IoT zone is increasing day by day. Hence, an IoT setup should be capable of handling the massive expansion. The data generated as an outcome is enormous, and it should be handled appropriately.

4. Dynamic and Self-Adapting (Complexity)

IoT devices should dynamically adapt themselves to changing contexts and scenarios. Assume a camera meant for surveillance. It should be adaptable to work in different conditions and different light situations (morning, afternoon, and night).

Characteristics of IoT

5. Architecture

IoT Architecture cannot be homogeneous in nature. It should be hybrid, supporting different manufacturers ‘ products to function in the IoT network. IoT is not owned by anyone engineering branch. IoT is a reality when multiple domains come together.

6. Safety

There is a danger of the sensitive personal details of the users getting compromised when all his/her devices are connected to the internet. This can cause a loss to the user. Hence, data security is the major challenge. Besides, the equipment involved is huge. IoT networks may also be at risk. Therefore, equipment safety is also critical.

7. Self Configuring

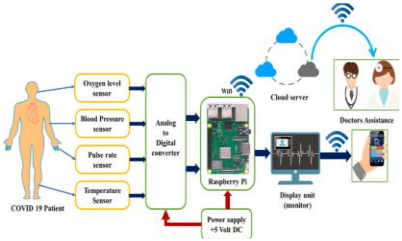
This is one of the most important characteristics of IoT. IoT devices are able to upgrade their software in accordance with requirements with a minimum of user participation. Additionally, they can set up the network, allowing for the addition of new devices to an already-existing network.

8. Interoperability

IoT devices use standardized protocols and technologies to ensure they can communicate with each other and other systems. Interoperability is one of the key characteristics of the Internet of Things (IoT). It refers to the ability of different IoT devices and systems to communicate and exchange data with each other, regardless of the underlying technology or manufacturer.

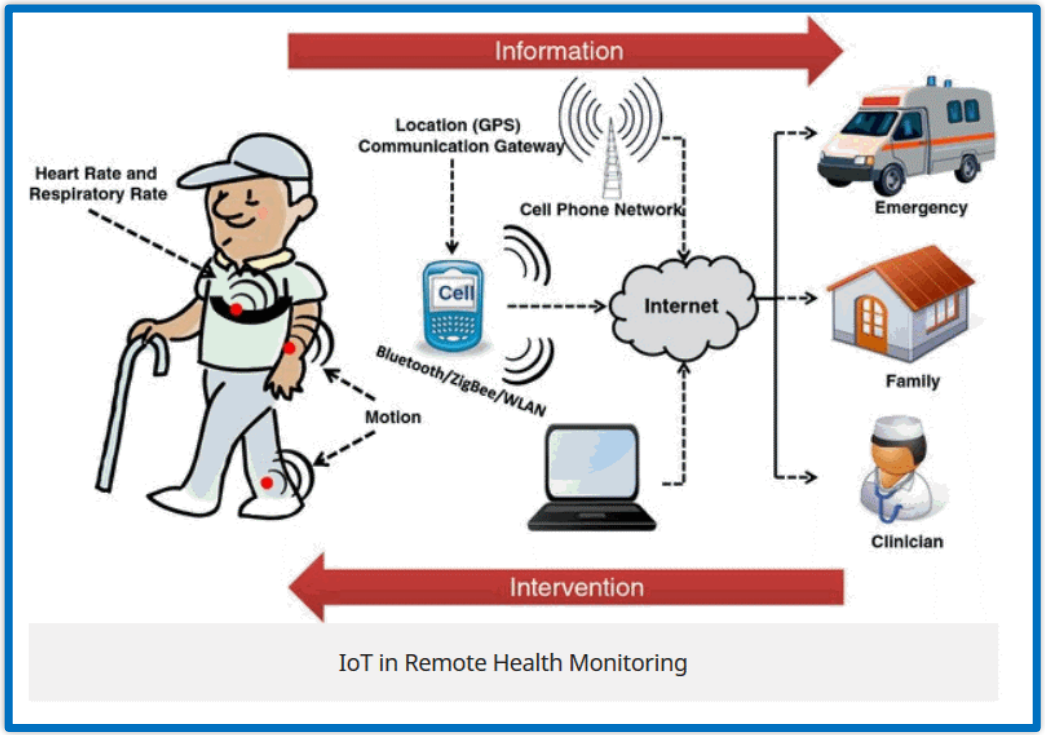
Case Studies and Applications

- Example 1: Smart Cities
  - Real-time traffic management
  - Environmental monitoring
- Example 2: Healthcare
  - Remote patient monitoring
  - Real-time diagnostics
- Example 3: Industrial IoT
  - Predictive maintenance
  - Real-time quality control



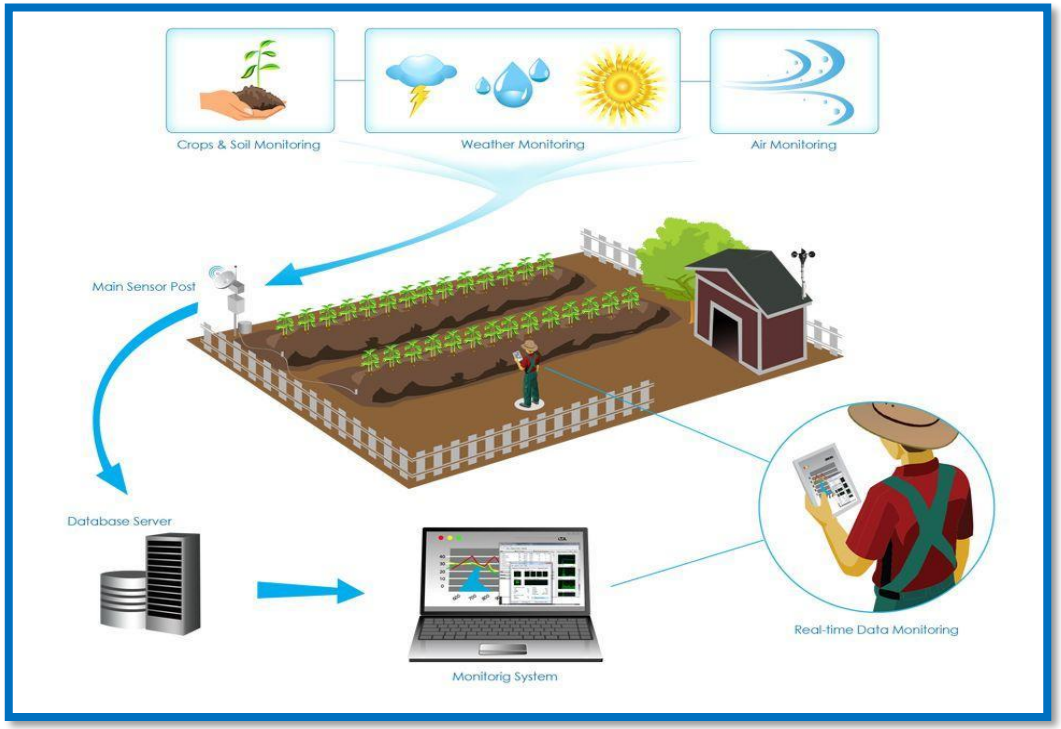

Health center
Smart ships
Smart home
Smart sea transport
Maritime industry (IoS)
IoT applications
eHealth
Smart ports
Other application domains
Agriculture sector
E-Learning
Power system
Education sector
Smart transportation

IoT in Health

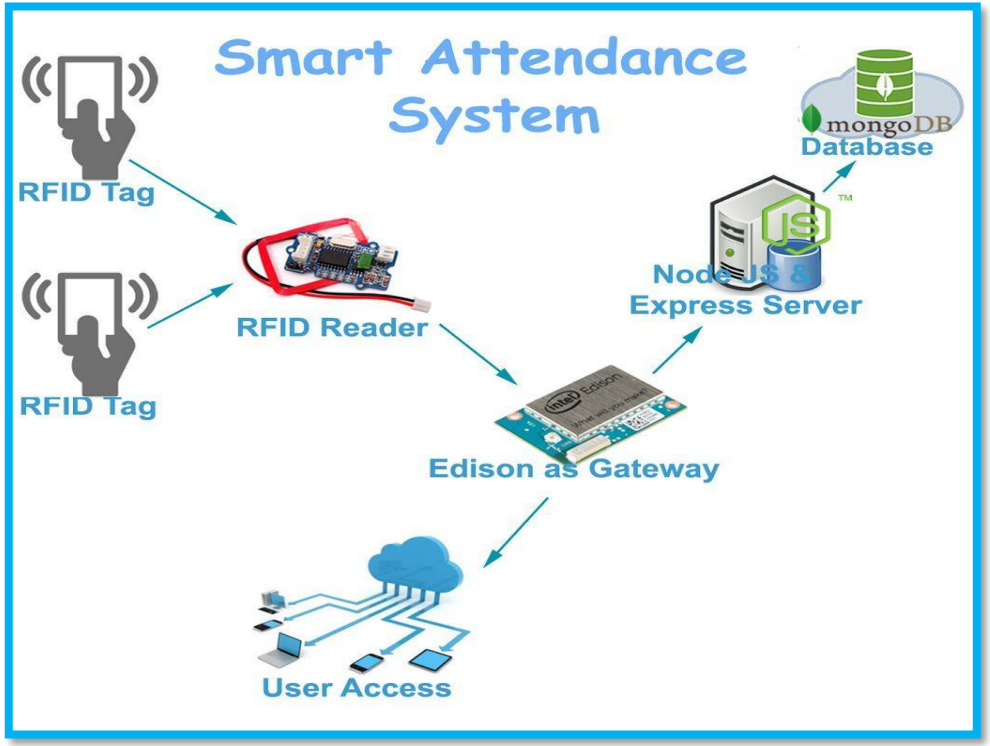


IoT in Remote Health Monitoring

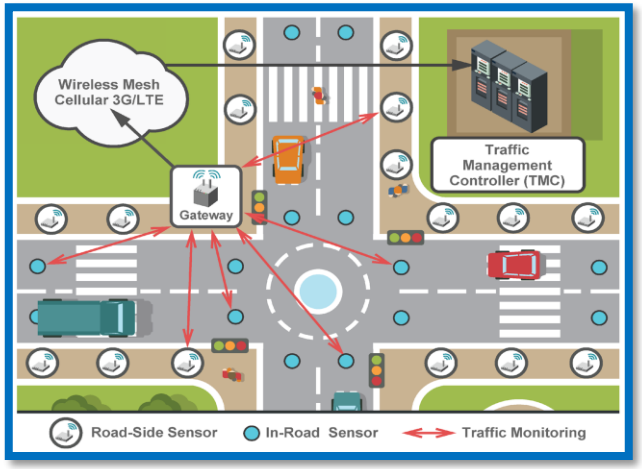
IoT in Agriculture



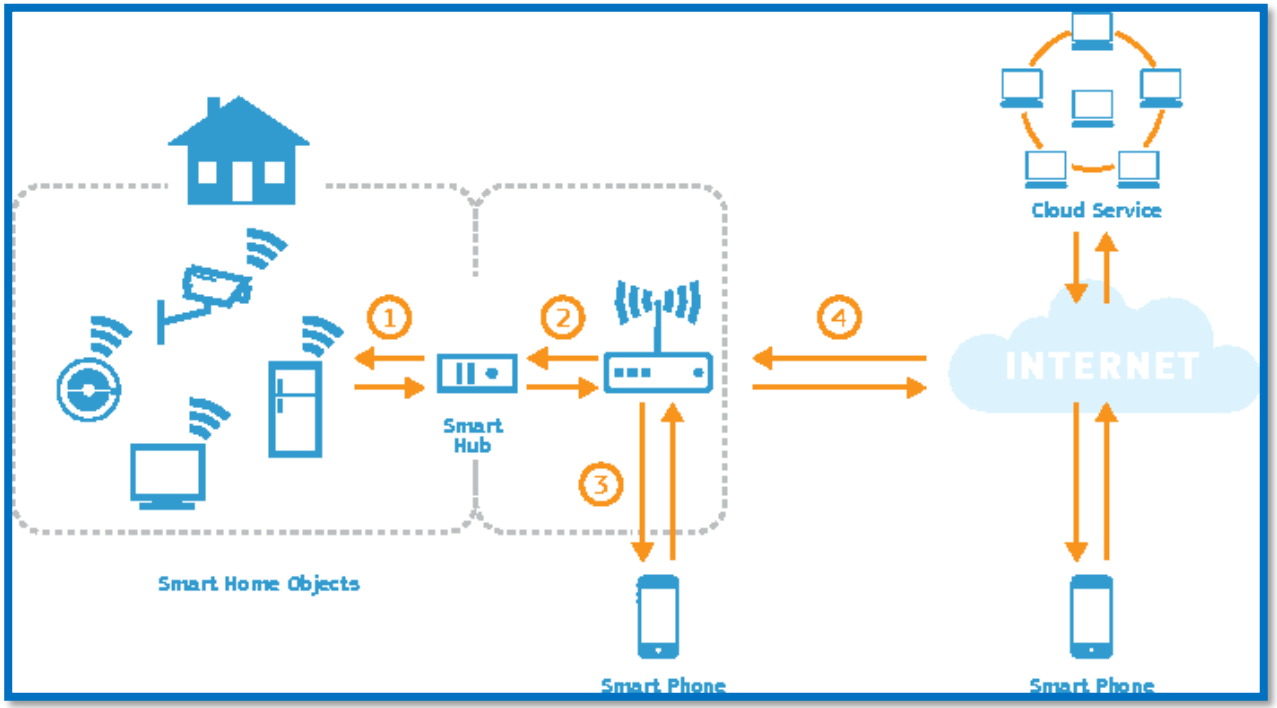
IoT in Education



IoT in Traffic Control



IoT in Smart Home

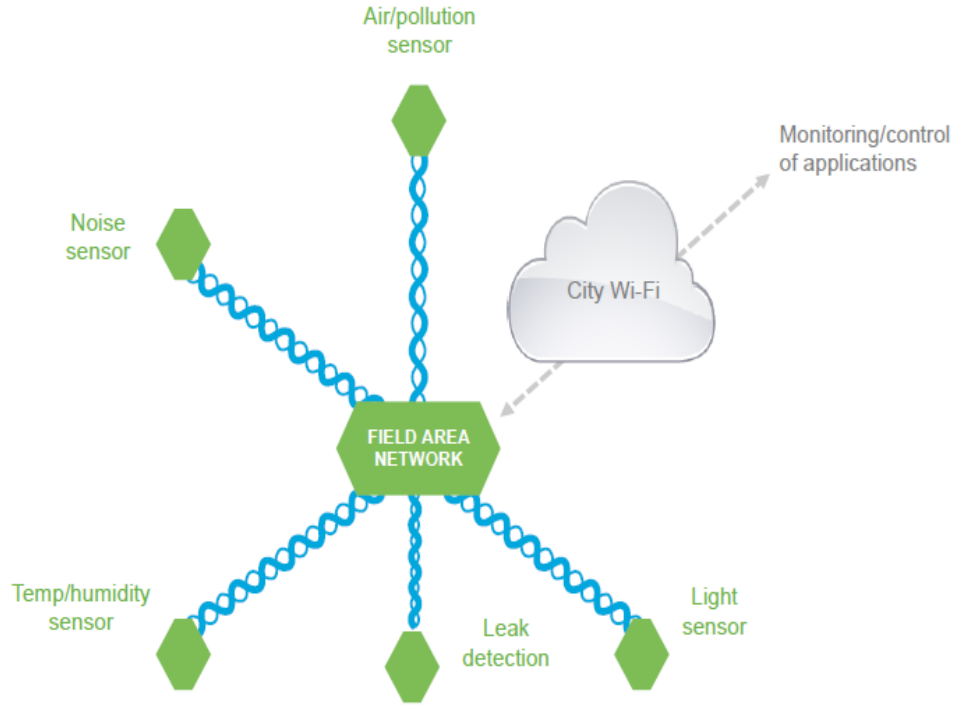


IoT in Pollution Tracking

Installation of environment sensors: air, light, humidity, noise, etc.

Benefits include:

- Leverages parking sensor infrastructure
- Provides valuable data for improving analytics applications and forecasting



Industrial IoT (IIOT)

- Industrial IoT (IIoT) focusses on the use of cyber-physical systems to monitor the physical factory processes and make data-based automated decisions.
- While the physical systems are made the intelligent using IoT, the real-time communication, and cooperation both with each other and with humans is established via the wireless web
- IIoT brings in the concept of ‘a *connected factory leads to a smart factory*’.

IIOT in Manufacturing

1. **Digital/connected factory**: IoT enabled machinery can transmit operational information to the partners like original equipment manufacturers and to field engineers.
2. **Facility management:** The use of IoT sensors in manufacturing equipment enables condition-based maintenance alerts.
3. **Production flow monitoring:** IoT in manufacturing can enable the monitoring of production lines starting from the refining process down to the packaging of final products.
4. **Inventory management:** IoT applications permit the monitoring of events across a supply chain.

IIOT in Manufacturing

5. **Plant Safety and Security:** IoT combined big data analysis can improve the overall workers' safety and security in the plant. .
6. **Quality control:** IoT sensors collect aggregate product data and other third-party syndicated data from various stages of a product cycle.
7. **Packaging Optimization**: By using IoT sensors in products and/or packaging, manufacturers can gain insights into the usage patterns and handling of product from multiple customers.
8. **Logistics and Supply Chain Optimization**: The Industrial IoT (IIoT) can provide access to real-time supply chain information by tracking materials, equipment, and products as they move through the supply chain.

DEPARTMENT OF INTERDISCIPLINARY COURSES IN ENGINEERING

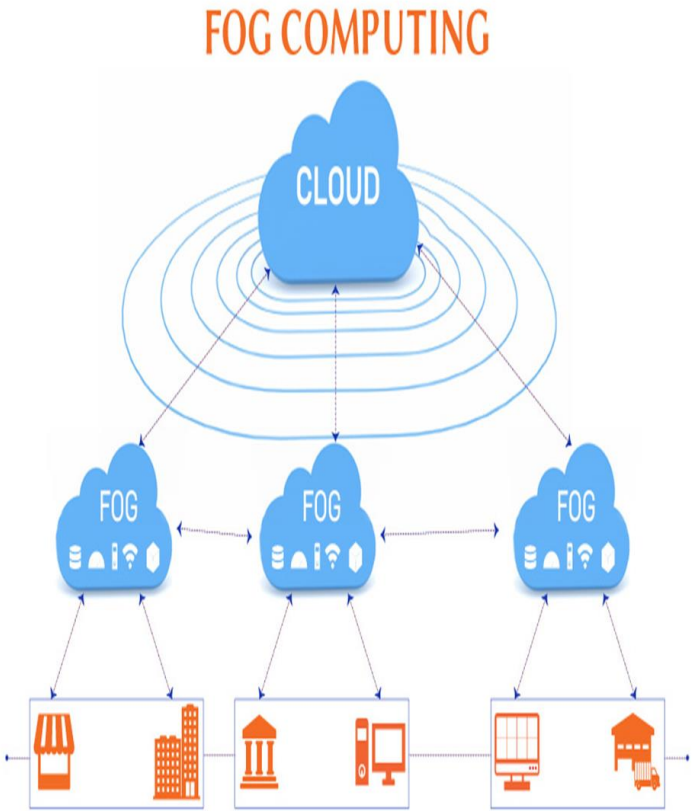
CHITKARA UNIVERSITY

Real-Time Analytics in IoT and Fog Computing

Layered Architecture, SOA, and API-Oriented Architecture

Fog Computing

- What is IoT: Network of physical devices embedded with electronics, software's and sensors which enable these objects to connect and exchange data.
- Fog Computing: Distributed computing infrastructure that brings computation, storage and networking closer to the source of data, at the edge of the network.
- Challenge: Traditional cloud-based IoT solutions struggle with latency, bandwidth, and security.
- Solution: Fog computing addresses these challenges by processing data closer to the source.



Fog Computing

Edge Computing

Computation takes place at the edge of a device's network, which is known as edge computing. That means a computer is connected with the network of the device, which processes the data and sends the data to the cloud in real-time. That computer is known as "edge computer" or "edge node".

Example of Edge computing:

- Autonomous vehicle edge computing devices collect data from cameras and sensors on the vehicle, process it, and make decisions in milliseconds, such as self-parking cars.
- In order to accurately assess a patient's condition and foresee treatments, data is processed from a variety of edge devices connected to sensors and monitors.

Fog Computing

Fog computing is an extension of cloud computing. It is a layer in between the edge and the cloud. **When edge computers send huge amounts of data to the cloud, fog nodes receive the data and analyze what's important. Then the fog nodes transfer the important data to the cloud to be stored and delete the unimportant data or keep them with themselves for further analysis.** In this way, fog computing saves a lot of space in the cloud and transfers important data quickly.

Fog Computing

Why is Fog Computing Used?

•**To improve latency and performance:** Because fog nodes are often deployed at the network edge, closer to the IoT devices themselves, they can substantially reduce the processing time and enhance performance for applications that demand low latency.

•**To improve decision-making:** It can help improve decision-making in real-time as fog computing allows for real-time data collection and analysis from IoT devices.

•**To reduce costs:** Fog computing can also help reduce costs associated with data storage and analysis. This is because, by bringing computation and data storage closer to the network edge, fog computing reduces the amount of data that needs to be transmitted back to a central location for processing.

Fog Computing

Components of Fog Computing:

Some of the key components of Fog computing

Edge devices: These are the devices located at the edge of the network, closest to the data source. Edge devices include sensors, PLCs (programmable logic controllers), and gateway routers.

Data processing: Data processing is done locally on edge devices rather than sent to a central location for processing. The result is improved performance and reduced latency.

Data storage: Edge devices can store data locally rather than sending it to a central location for storage. This improves security and privacy, as well as reduces latency.

Connectivity: Fog computing requires high-speed connectivity between edge devices and the rest of the network. This is achieved through wired or wireless means.

Why Is Fog Computing Beneficial for IoT?

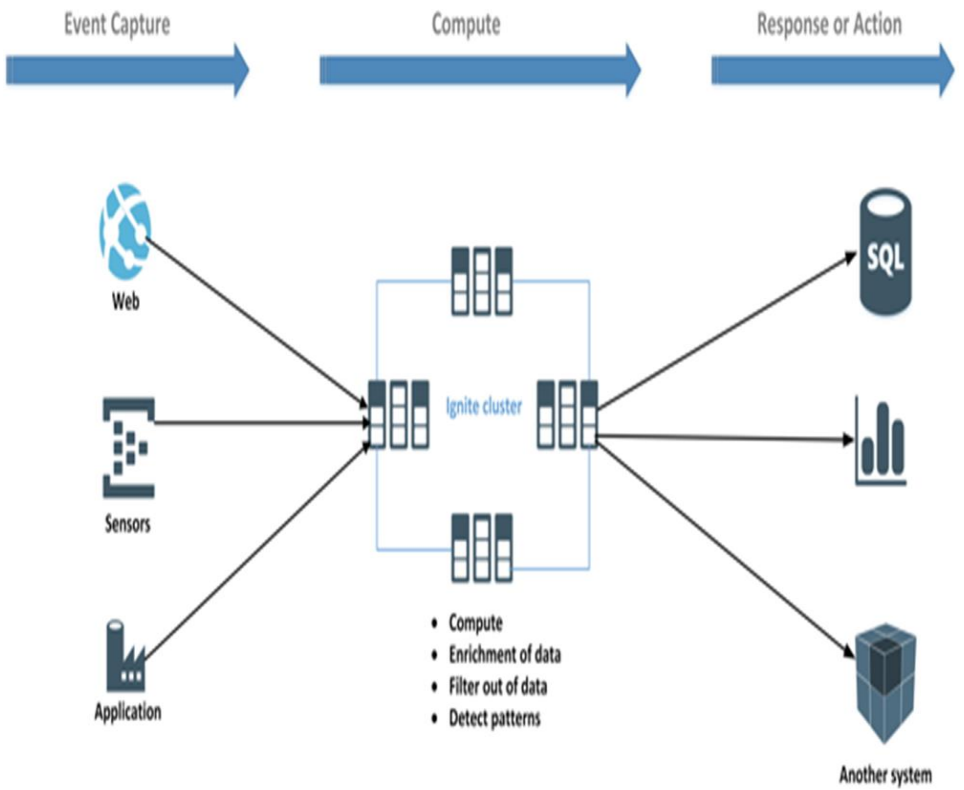
The internet of things (IoT) is a system of interconnected devices, sensors, and software components that share data and information. The power of the IoT comes from its ability to collect and analyze massive volumes of data from various sources. This data can be used to improve efficiency, optimize operations and make better decisions.

- **Fog computing in IoT is a decentralized computing model that brings computation and data storage closer to the edge of the network.**

- **In other words, fog computing moves processing power and data storage away from centralized server farms and into local networks where IoT devices are located.**

Real time analytics

- Real-time analytics in IoT refers to the process of analysing data generated by connected devices as it arrives, without any delay.
- IoT systems collect vast amounts of streaming data from millions of devices (sensors, wearables, industrial machines, etc.).
- Through Real-time analytics process this data instantly, allows for timely insights and actions.
- It’s crucial for applications where immediate decisions impact safety, efficiency, or performance.



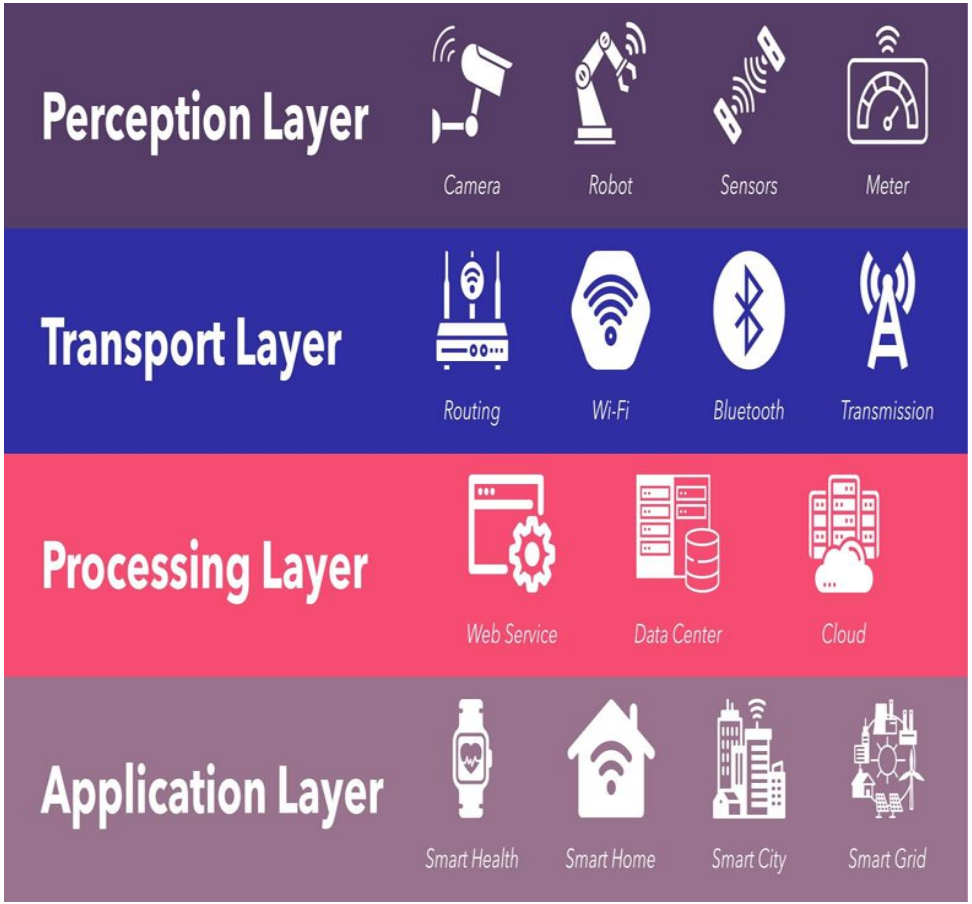
The Need for Real-Time Analytics

- Data Volume: IoT generates massive amounts of data.
- Data Velocity: Data needs to be processed immediately to derive insights.
- Data Variety: Data comes in various formats and structures.
- Benefits of Real time analytics:

a) Improved decision-making.
b) Enhanced operational efficiency.
c) Predictive maintenance
d) Fraud detection
e) Personalized experiences

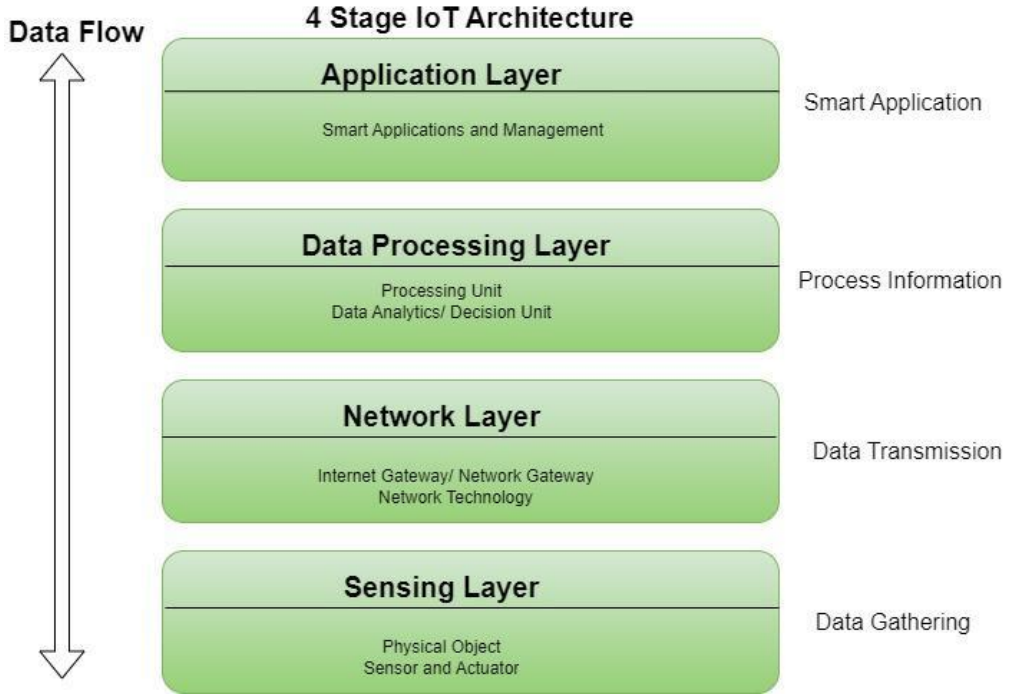
IoT Layered Architecture

- Perception Layer: Sensors, actuators, and devices collect data from the physical world.
- Transport Layer: Connectivity infrastructure (Wi-Fi, cellular, Bluetooth) for data transmission.
- Processing Layer: Data aggregation, processing, and storage (fog and cloud).
- Application Layer: Value-added services and applications based on processed data.



IoT Layered Architecture

- Perception Layer (**Sensing Layer**): Sensors, actuators, and devices collect data from the physical world.
- Transport Layer (**Network Layer**): Connectivity infrastructure (Wi-Fi, cellular, Bluetooth) for data transmission.
- Processing Layer (**Data Processing Layer**): Data aggregation, processing, and storage (fog and cloud).
- Application Layer: Value-added services and applications based on processed data.



IoT Layered Architecture

A five-layer IoT architecture can also be used

1. Perception Layer
2. Transport layer
3. Processing layer
4. Application layer
5. Business layer

Service Oriented Architecture (SOA)

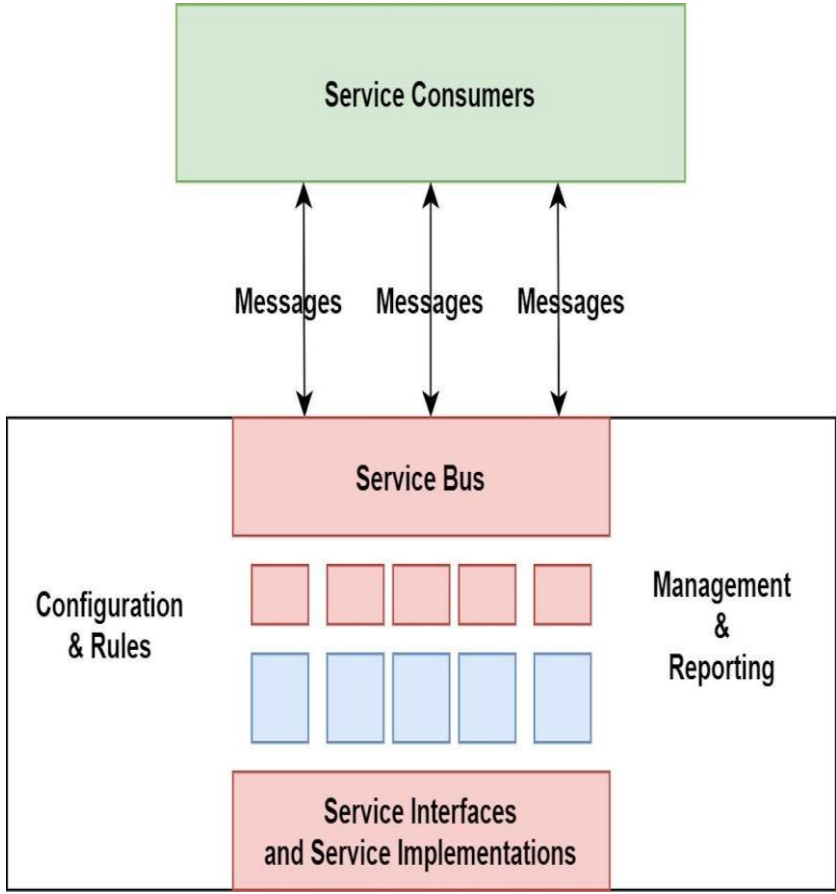
Service-Oriented Architecture (SOA) is a stage in the evolution of application development and/or integration. **It defines a way to make software components reusable using the interfaces.**

Formally, SOA is an architectural approach in which **applications make use of services available in the network**. In this architecture, **services are provided to form applications**, through a network call over the internet. It uses common communication standards to speed up and streamline the service integrations in applications. **Each service in SOA is a complete business function in itself.** The services are published in such a way that it makes it easy for the developers to assemble their apps using those services.

- SOA **allows users to combine a large number of facilities from existing services** to form applications.
- SOA **encompasses a set of design principles** that structure system development and provide means for integrating components into a coherent and decentralized system.
- SOA-based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.

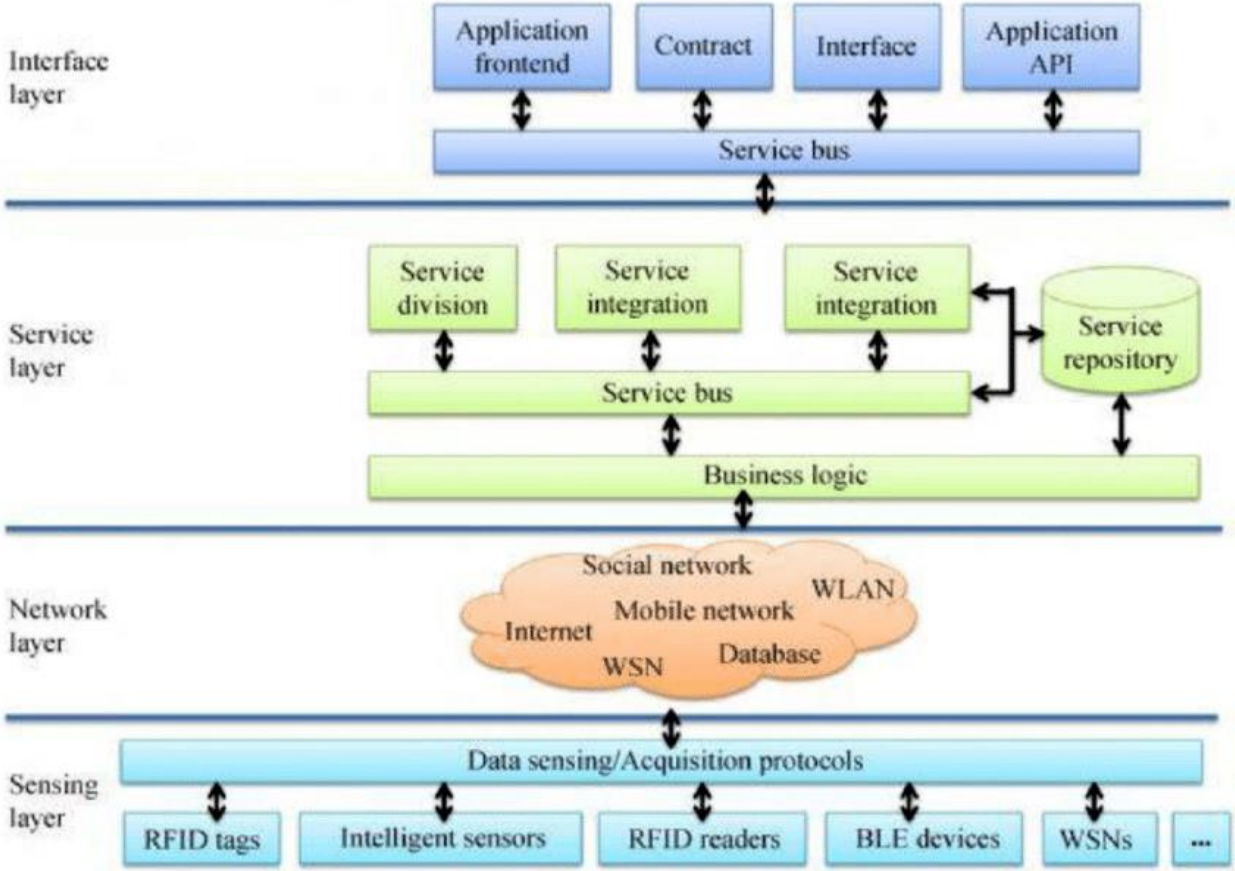
SOA-Based Architecture

1. **Definition**:
   1. SOA(Service oriented Architecture) defines a way to make software components reusable using interfaces.
   2. It allows applications to use services available in the network.
2. **Key Points**:
   1. **Services**: A service is a discrete unit of functionality that can be accessed remotely and acted upon independently. For example, retrieving a credit card statement online.
   2. **Independence**: SOA aims to be independent of vendors, products, and technologies.
   3. **Integration**: SOA is commonly used for system integration.



SOA-Based Architecture in IoT

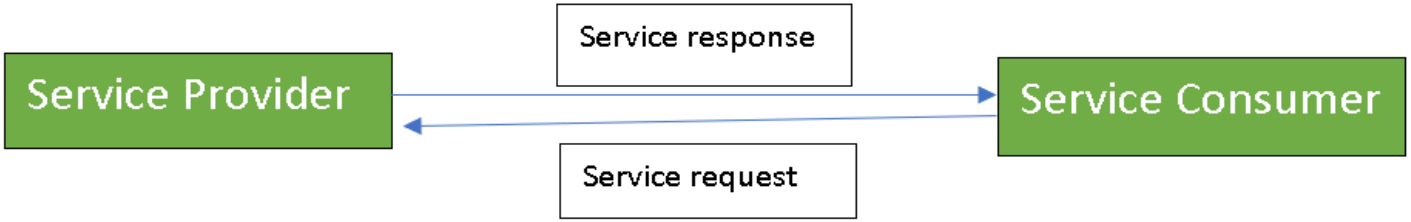
- Sensing layer is integrated with available hardware objects to sense the status of things
- Network layer is the infrastructure to support over wireless or wired connections among things.
- Service layer is to create and manage services required by users or applications
- Interfaces layer consists of the interaction methods with users or applications



SOA-Based Architecture

There are two major roles within Service-oriented Architecture:

1. **Service provider**
2. **Service consumer**



Service provider: The service provider is the **maintainer of the service and the organization** that makes available one or more services for others to use. **To advertise services**, the **provider can publish them** in a registry, together **with a service contract** that **specifies the nature of the service**, **how to use it, the requirements for the service, and the fees charged.**

Service consumer: The service consumer can locate the service metadata in the registry and develop the required client components to bind and use the service.

SOA-Based Architecture

A good example of a SOA-based system is a set of customer services, like **CRM, ERP, Product Information Management System (PIM).**

CRM: **Customer Relationship Management** refers to the process of tracking and handling all interactions between a business and its customers. **You can track forms, emails, text messages, social posts, quotes, and even purchases in one location using the platform. e.g. Semrush, Saleforce, Hubspot**

Semrush CRM

Semrush CRM enables you to:

•Track and manage new and existing customers
•Maintain and organize key customer information
•Set and manage client tasks
•Track multiple Semrush projects for a specific client
•Create Client Portals for closer client collaboration

Salesforce

Salesforce is one of the original SaaS CRM tool and is a common choice for enterprise companies with large customer bases.

Software as a service (SaaS) allows users to connect to and use cloud-based apps over the Internet.

HubSpot

HubSpot is a popular CRM platform that combines a variety of hubs and integrations for a customized experience.

ERP (**Enterprise Resource Planning**) system is **a comprehensive software platform that integrates various business processes and functions into a single, unified system.**

e.g. For example, when a customer contacts support with a query about an order, the representative can instantly access the order's status, including any delays or issues in the supply chain, and communicate this information accurately to the customer.

Benefits of SOA Architecture

Faster Time to Market:

Developers can reuse services across different business processes, saving time and costs.

Assembling applications becomes faster with SOA compared to building everything from scratch.

Efficient Maintenance:

SOA promotes smaller, modular services.

Updating, debugging, and modifying individual services is easier than dealing with large monolithic code blocks.

Changes to one service do not impact the overall functionality of the entire business process.

Greater Adaptability:

SOA is adaptable to technological advancements.

You can modernize applications efficiently and cost-effectively.

For example, older electronic health record systems can be integrated into newer cloud-based applications.

Interoperability:

Each service in SOA includes description documents specifying functionality and terms.

Any client system can run a service, regardless of the underlying platform or programming language.

Loose coupling ensures changes in one service do not affect others.

Reduced Redundancy:

SOA reduces redundancy by reusing services.

It increases usability, maintainability, and overall value.

Advantages of SOA Architecture

Advantages of SOA

Service reusability: In SOA, applications are made from existing services. Thus, services can be reused to make many applications.

Easy maintenance: As services are independent of each other they can be updated and modified easily without affecting other services.

Platform independent: SOA allows making a complex application by combining services picked from different sources, independent of the platform.

Availability: SOA facilities are easily available to anyone on request.

Reliability: SOA applications are more reliable because it is easy to debug small services rather than huge codes

Scalability: Services can run on different servers within an environment, this increases scalability of messages.

Disadvantages of SOA:

High overhead: A validation of input parameters of services is done whenever services interact, this decreases performance as it increases load and response time.

High investment: A huge initial investment is required for SOA.

Complex service management: When services interact they exchange messages to tasks. The number of messages may go in millions. It becomes a cumbersome task to handle a large number of messages.

Practical Application of SOA Architecture

Practical applications of SOA: SOA is used in many ways around us whether it is mentioned or not.

1. SOA infrastructure is **used by many armies and air forces** to deploy situational awareness systems.
2. SOA is **used to improve healthcare delivery**.
3. Nowadays many apps are games and they use inbuilt functions to run. For example, an app might need GPS so it uses the inbuilt GPS functions of the device. This is SOA in mobile solutions.

API-Oriented Architecture

What Is API Architecture ?

API architecture refers to the process of developing a software interface that exposes backend data and application functionality for use in new applications.

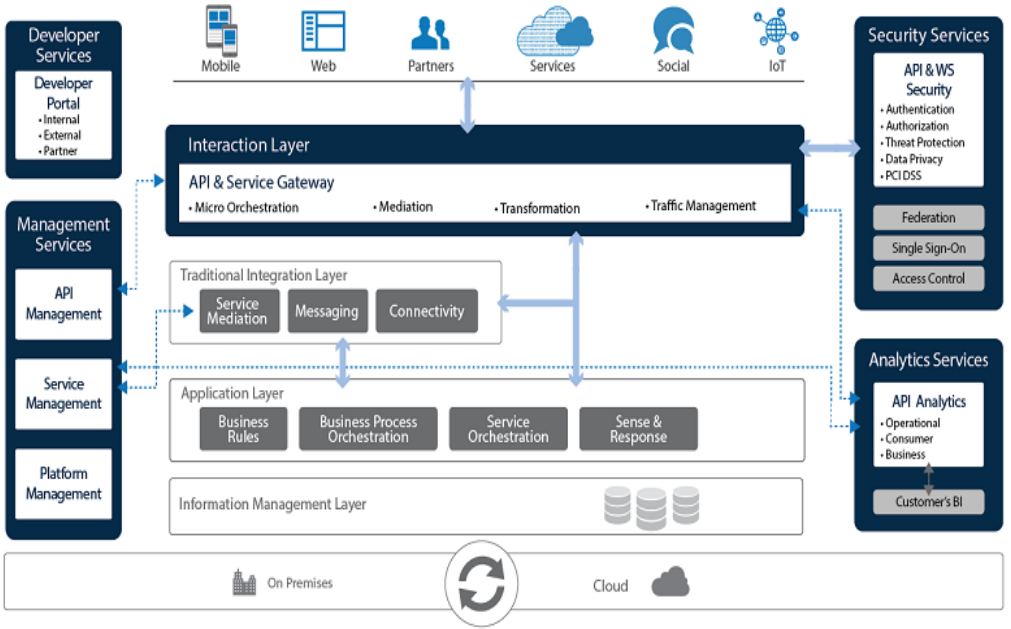
4 Layers of API Architecture

Information Management Layer: Modern digital organizations run on giant data repositories. You need a steady diet of advanced database systems to store and manage all of it. All of your applications need a reliable, high-performance data layer. And you find yourself needing increasingly advanced (or simplified) data storage systems.

Application Layer: This speaks for itself. It's where the applications that run your organization live. You might want to replace these applications with more modern alternatives built in microservices. But this is likely not realistic in many cases.

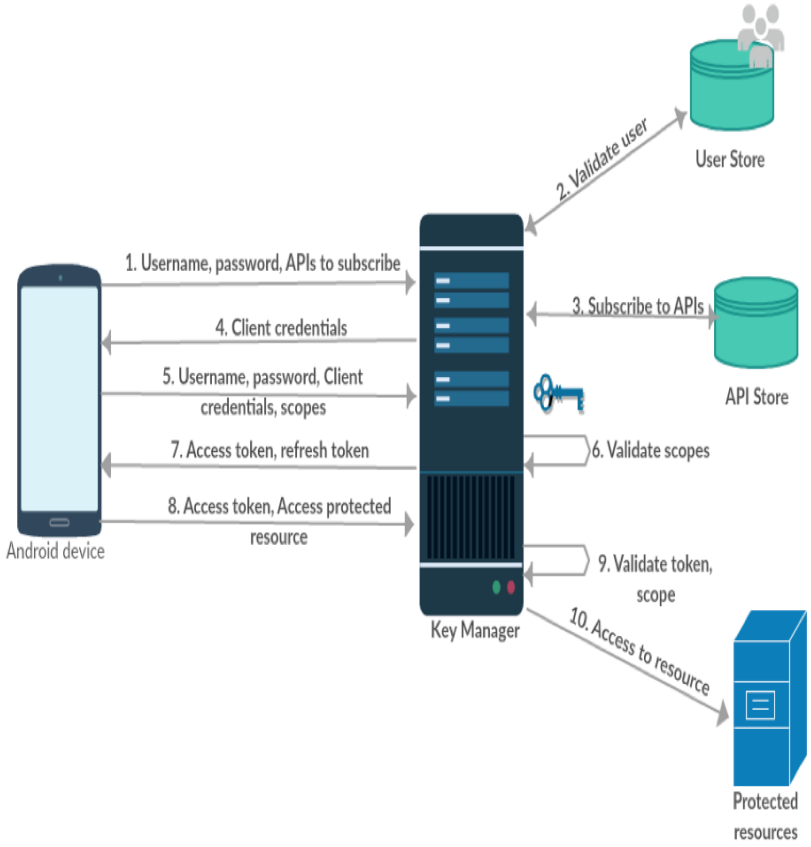
Integration Layer: The integration layer enhances interoperability by managing the integration of various systems and services.

Interaction Layer: The interaction layer is where the applications and services used by your customers, partners, and employees interact with your business applications and data.



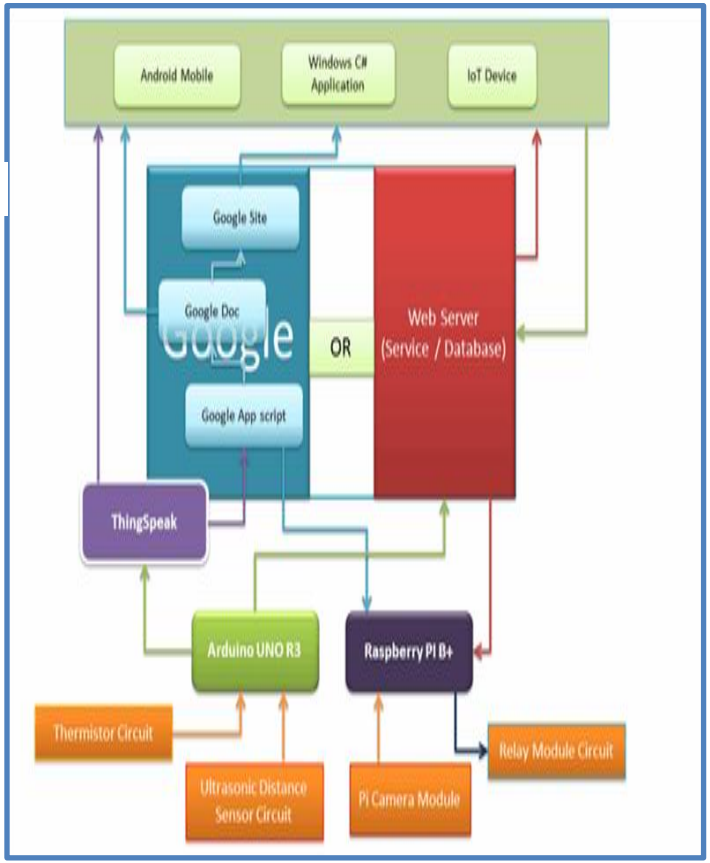
API-Oriented Architecture

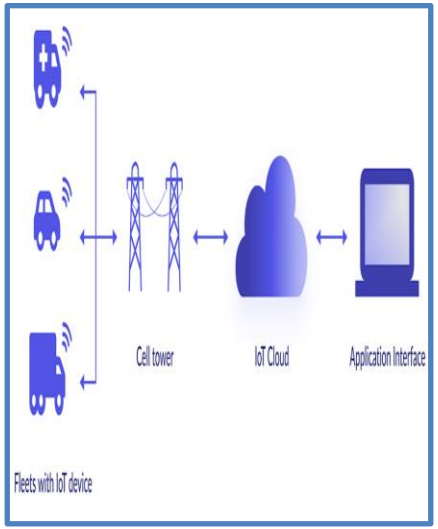
- API (Application Programming Interface) is a tool that software developers use to gather and transfer data between different applications or computers
- Role in IoT:
  - **Gathering Data:** APIs allow communication between connected devices and applications, enabling data collection from the device.
  - **Instructing Devices:** APIs can instruct a connected device to perform specific actions remotely.
- Since IoT devices can be located anywhere globally, APIs make it possible to access and utilize device data from a distance.

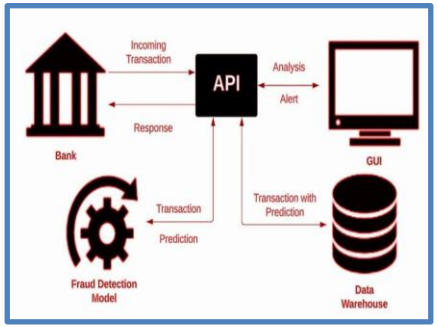


How API and IoT work together

- APIs enable users to programmatically access information about their devices and make decisions based on that data.
- Examples of API use in IoT:
  - Fleet Management: Activating devices, billing, and reporting are all done via APIs.
  - Fraud Prevention: Users can create alerts if a device is moved or if a SIM card is stolen.
  - Custom Applications: Developers leverage APIs to integrate IoT data into their own tools or applications.







Types of API Architectures

APIs in IoT are highly diverse to support IoT development by all means. The key API types are as mentioned next.

SOAP: (Simple Object Access Protocol)

SOAP APIs are crucial for IoT devices development as **they make a communication bridge between the servers and the clients.** The API supports only XML-based data transfer. SOAP is mostly used for applications that require a high level of security – like payments.

REST: (Representational State Transfer)

IoT REST APIs useful for HTTP (Hypertext transfer protocol) data transmission and for empowering IoT devices to stay associated with the rest of the world. These APIs are driven by architectural principles and boast features like interface simplicity, instant resources identification during the request, and manipulation of particular interfaces.

JSON and XML: (JavaScript Object Notation) and (Extensible Markup Language)

A bit older than SOAP APIs, JSON and XML IoT APIs are based on a simple approach and consume limited bandwidth. JSON and XML are two of the formats in which data is delivered. JSON is a lightweight, human-readable format that can be used with any programming language. Most public web services use REST APIs with JSON.

Extensible Markup Language (XML) lets you define and store data in a shareable manner. XML supports information exchange between computer systems such as websites, databases, and third-party applications. Predefined rules make it easy to transmit data as XML files over any network because the recipient can use those rules to read the data accurately and efficiently.

The important role of APIs in the IoT

What makes API crucial for IoT and any other device/software:

- The fact that they support effective utilization of pre-existing functions to ensure smooth software processing while keeping developers free from the need of reprogramming again and again.

- The world of IoT is too complex demanding continual contact between multiple agents involved. API usage makes the task achievable.

- IoT APIs serve as amazing technical development resources as unmatched flexibility is granted.

- Speaking of cyber security, APIs are essential as developers can use them to gain control over access requests.

IoT Challenges

1. **IoT Security**:
   1. Connecting numerous devices to the internet introduces security risks. Each connected device becomes a potential entry point for hackers.
   2. Challenges include managing device updates, securing communication (especially over unsecured Wi-Fi networks), and detecting vulnerabilities.
2. **Device Compatibility**:
   1. IoT ecosystems involve diverse devices from various manufacturers. Ensuring seamless compatibility between them can be challenging.
   2. Interoperability standards are crucial to enable devices to work together effectively.
3. **Bandwidth Constraints**:
   1. As the number of connected devices grows, network bandwidth becomes a limitation.
   2. Efficient data transmission and load balancing are necessary to prevent congestion.


IoT
SECURITY
CHALLENGES
1
Lack of Physical Security
2
Lack of Visibility
3
Data Privacy
4
Botnet Attacks
5
Ransomware

IoT Challenges

Security, privacy and data sharing issues

- Because IoT devices are closely connected, all a hacker has to do is exploit one vulnerability to manipulate all the data, rendering it unusable. And manufacturers that don't update their devices regularly -- or at all -- leave them vulnerable to cybercriminals.
- However, hackers aren't the only threat to the internet of things; privacy is another major concern for IoT users. For instance, companies that make and distribute consumer IoT devices could use those devices to obtain and sell users' personal data.

IoT Challenges

Interoperability

Because IoT encompasses a wide array of devices, sensors, and platforms from various manufacturers and platform providers. Ensuring these devices can communicate and work together seamlessly is a significant challenge and can be costly for device certification. Interoperability standards such as MQTT and CoAP help, but organizations must still plan for integration, compatibility testing, and potential vendor lock-in.

IoT Challenges

Interoperability

Interoperability refers to the ability of different systems, devices, or components to work together seamlessly and exchange data effectively.

Because IoT encompasses a wide array of devices, sensors, and platforms from various manufacturers and platform providers. Ensuring these devices can communicate and work together seamlessly is a significant challenge and can be costly for device certification. Interoperability standards such as MQTT and CoAP help, but organizations must still plan for integration, compatibility testing, and potential vendor lock-in.

IoT Challenges

Scalability

Scalability refers to the ability of a system to handle increasing workloads or numbers of users without a significant decline in performance.

IoT deployments are typically expected to scale from a handful of devices to thousands or even millions. Verifying the infrastructure can handle this scalability without compromising performance is a significant challenge. Issues such as data overload, network congestion, and efficient data processing and storage solutions need to be addressed early in the deployment process.

IoT Challenges

Reliability

Reliability refers to the ability of a system to perform its intended function consistently and without failure over time.

Ensuring 24/7 uptime, especially for IoT devices stationed in remote or challenging environments, presents unique challenges in IoT. Organizations must develop robust maintenance and remote monitoring strategies to address issues promptly and minimize downtime.

IoT Challenges

Power Management

Battery powered and low-power IoT devices require effective power management to provide long-lasting operation and minimize maintenance needs. This challenge is even more significant in remote or inaccessible locations, where power conservation becomes essential for optimal device performance and reliability.

Cost Management

While IoT promises efficiency gains, the initial investment can be significant. Balancing the cost of deployment with the expected benefits requires careful planning and cost analysis. Organizations must consider hardware, software, connectivity, and ongoing operational expenses

IOT Communication

- The main goal of IoT is to **enable communication between devices that are not traditional computers**.
- To interact with **gateways, programs, servers and cloud platforms**, IoT devices **need to have connections** as it allows devices to transmit and receive data across a network.
- IoT allows these objects to send and receive data over a network. There are **various protocols** in IOT established to offer this communication.
- There are mainly two types of protocols: **network and data.**

IOT Communication (Data) Protocols

Some of the different IoT data protocols, namely:

- **Message Queue Telemetry Transport (MQTT)**
- **HyperText Transfer Protocol (HTTP)**
- **Constrained Application Protocol (CoAP)**
- **Data Distribution Service (DDS)**
- **WebSocket**
- **Advanced Message Queue Protocol (AMQP)**
- **Extensible Messaging and Presence Protocol (XMPP)**
- **OPC Unified Architecture (OPC UA)**

Message Queue Telemetry Transport(MQTT)

Designed to be lightweight, so it can **work in very low bandwidth** networks, MQTT allows communication between nodes in both reliable and unreliable networks.

Figure: MQTT's publish/subscribe architecture.

A use case of MQTT is in a smart factory where there are temperature sensors installed along with the production plant. The installed **sensors will connect to the MQTT broker** and will **publish the data within sensor topics**, as follows: **sensors/temperature**

Afterward, the **MQTT clients**, which can be of several types and quantities, **will subscribe to the same topic in order to read the temperature data.**

HyperText Transfer Protocol (HTTP)

This protocol has been the origin of data communication for the World Wide Web (WWW), so logically it is being used in the IoT world. However, **it is not optimized** for it, because of the following:

- The HTTP is **made for two systems communicating to each other at a time**, not more, so it is **time and energy-consuming to connect several sensors** to get information.
- The HTTP is **unidirectional**, made for one system (client) to be sending one message to another one (server). This makes it quite hard to escalate an IoT solution.
- Power consumption: HTTP relies on Transmission Control Protocol (TCP), which requires a lot of computing resources, so it is **not suitable for battery-powered applications.**

Constrained Application Protocol (CoAP)

- CoAP is a **web transfer protocol** to be **used with limited networks** with **low bandwidth and low availability**. It follows a **client/server architecture** and is built similarly to HTTP, supporting the REST model: servers make resources available with an URL, and clients can make requests of types GET, POST, PUT and DELETE.

- The CoAP communication links are 1:1 and UDP-based, so the delivery is not guaranteed. CoAP is made to work in highly congested networks, where nodes do not have a lot of intelligence and are not always working.

- URL: Uniform Resource Locator
- UDP: User Datagram Protocol

Data Distribution Service (DDS):

- It is similar to MQTT.
- It follows a publish-subscribe methodology, with the main difference being that there are no brokers.
- It means that all publishers (i.e., temperature sensors) and subscribers (i.e., mobile phones) are all connected to the same network.
- This network is known as Global Data Space (GDS) and it interconnects each node with all the other ones to avoid bottlenecks.

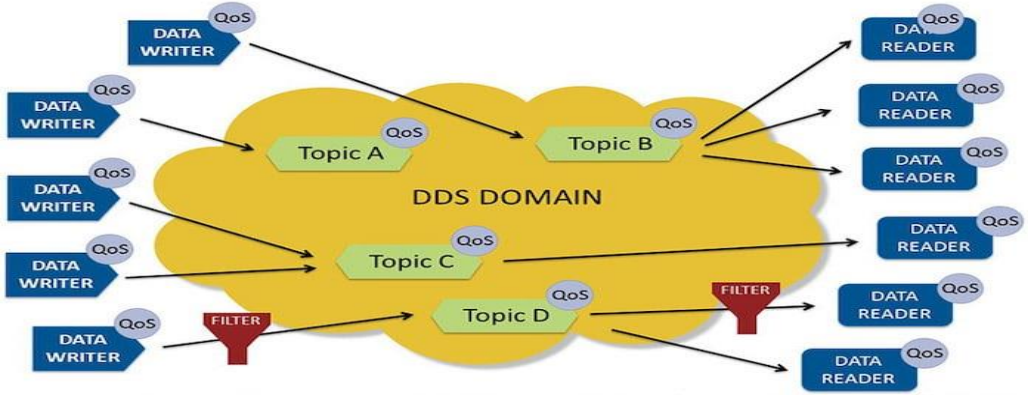


Figure: A DDS Global Data Space.

- Furthermore, any node can leave or join the network, since they are dynamically discovered.

WebSocket

WebSocket:

- Linked to the HTTP protocol, the WebSocket technology establishes a TCP connection between a browser and a server, and then both of them exchange information until the connection is closed.

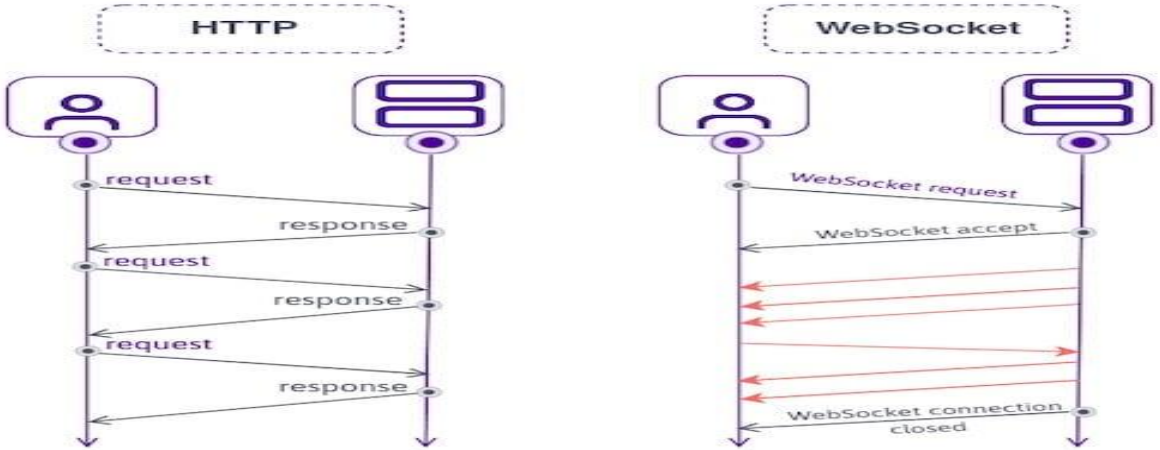


Figure: Comparison between HTTP and WebSocket.

- Although this protocol can be seen as an improvement of the HTTP connection, the WebSocket is still very overloaded and heavy for IoT applications.

Advanced Message Queue Protocol (AMQP):

In the beginning, AMQP **was not initially created for IoT applications**, but **for banking environments**. AMQP accepts **publish/subscribe architectures**, as well as **request/response types**. It is **TCP-based**, so **delivery is guaranteed**, as well as acknowledgment, which makes this protocol **reliable**, with the consequent overhead message reliability.
Compared to MQTT, **AMQP offers two Quality of Service levels**:

- **At most once**: the sender does not wait until having an acknowledgment from the receiver to delete a message.
- **At least once**: for each message, the sender will receive an acknowledgment from the receiver before deleting the message. In a case where the acknowledgment is lost, the message is re-sent.
- **Exactly once**: the messages are sent only once. It requires special coordination between the sender and the receiver.

Extensible Messaging and Presence Protocol (XMPP):

- It is **based on Extensible Markup Language (XML)** and in the past, it was known as **Jabber**. It is an **open-source, decentralized, secure protocol to exchange XML messages**.

- A characteristic factor of XMPP is its addressing method and how nodes are identified. It **uses a Jabber ID with the format jabberID@domain.com**, which **allows two nodes to interchange information regardless of the distance between them.**

OPC Unified Architecture (OPC UA):

- It is a standard **made for industrial communication**, strongly oriented to **guarantee interoperability** between manufacturers, operating systems, and programming languages. The OPC foundation reported that many industrial vendors are currently (as of 2022) adopting the OPC UA as the open standard.

- All in all, the OPC UA is a transport-agnostic protocol, so **it supports both** of the previously used architectures: **request/response** (such as WebSocket or HTTP), as well as the **publish/subscribe** (such as the MQTT).

- OPC: Open Platform Communications

IOT Communication (Network) Protocols

Bluetooth	Zigbee	Bluetooth Low Energy	Wi-Fi
Z-Wave	Cellular	Sigfox	RFID
Ethernet	Near Field Communication (NFC)	LPWAN	LoRaWAN

Bluetooth

- It's effective for **sending high-speed data for distances of up to 10 meters**. Small amounts of data from wearables or sensors can be transmitted effectively using Bluetooth.
- More than **one third of IoT devices contain Bluetooth** connectivity.
- Bluetooth is a form of wireless technology **used for device communication** and to **make personal area networks(PANs)**.
- Bluetooth technology uses the **2.4 GHz ISM** (industrial, scientific, and medical) band
- The latest version is the bluetooth5 and it has features such as high range, speed and data broadcasting.
- It has some empowering features that support IoT devices. Bluetooth low energy (BLE) supports devices that require less power and ideal for IoT enabled projects.

Zigbee

- Zigbee is a **WLAN** and a wireless technology that aims to support **extremely low power devices.**
- It supports these kinds of devices and **makes it possible to connect them to the internet**.
- It is an **open global standard** and **works on IEEE 802.15.4** physical radio standards.
- IoT devices do not require extra functionality and Zigbee **is an ideal protocol** for transferring data from one communication point to another.
- Zigbee makes data flow easy. It is used to **send small amounts of data using very less power** which is why it is used in machine to machine communication(M2M) and IoT.
- Zigbee devices operate in the 2.4 GHz spectrum, although some can use 784 MHz (China), 868 MHz (Europe), and 915 MHz (US)

Bluetooth Low Energy (BLE)

- BLE is a Bluetooth that **uses less power**.
- It is **designed to support** the internet of things.
- BLE is energy efficient and **offers better connectivity** compared to other forms of technology such **as Zigbee or LoRa**.
- BLE **fits the need of data transfers as that is the only function** that takes place in IoT sensors.
- BLE is used in the **making of smartwatches**, **medical devices**, **fitness trackers**, **beacons** and **home automation devices.**
- BLE consumes **less power** and has **less bandwidth**.

Wi-Fi

- Wi-fi is a form of **local area network** for wireless communication.
- It is a better option for data transfers as **it easily fits into a variety of standards.**
- It plays an **important role in IoT communication and intercommunication** with other cellular networks such as Bluetooth.
- Wi-fi supports **high bandwidth** and **low latency**.
- It’s common in households and workplaces due to its low operating costs and works in the frequency of **2.4 GHz** and **5 GHz**.

Z-Wave

- Z-wave is a **wireless messaging protocol** to communicate between various IoT devices.
- It is useful especially **in home automation to connect appliances** in smart homes. Z-wave **offers a two-way mode of communication empowered with mesh networking** and **message received acknowledgment**.
- Z-wave is a **low-cost technology** that eliminates issues caused by Wi-fi and Bluetooth. Z-Wave **operates on a lower radio frequency** than Wi-Fi and Zigbee, which **reduces the likelihood of interference** from other wireless devices and directly improves device reliability. It uses **865 MHz** to **926 MHz** bands.
- The **network of Z-wave** includes the internet of things devices and control called **the primary hub.**
- When z-hub receives a message via a smartphone or tablet, it sends this message to the relevant smart home appliance.

RFID

- **Radio frequency identification system** is a technology that supports the identification of objects via radio waves.
- By connecting the RFID reader to the terminal of the internet, **users can identify, monitor and track the object with tags.**
- RFID is **fast, dependent and does not require physical interaction** between the user and the tagged item.
- It **identifies IoT objects by tagging and labelling** them. The **tag or label replaces the object.**
- RFID is **useful to identify, track and monitor remote IoT objects** with time and location.
- Low Frequency (LF) 125-135 KHz, High Frequency (HF) 13.56 MHz, Ultra High Frequency (UHF) 868-930 MHz

Cellular

- Cellular **connects the devices to anything and everything** without the need for smartphones or gateway.
- This means it **connects devices directly to the base station** without any intermediaries.
- These **connections are always present** even in remote areas.
- Cellular IoT makes it possible to construct less power-consuming devices that connect to the internet which was not possible before.
- There are cellular communication subtypes like **Long-Term Evolution Machine Type Communication (LTE-M) and NarrowBand-Internet of Things (NB-IoT)** that offer more data capacity and less power consumption. These subtypes are designed to enable IoT devices that require a long battery life or are used in places where conventional 4G technology is challenging to access.

Sigfox

- Sigfox was the first to introduce **Low-Power WAN (LPWAN)** technologies in the development of IoT projects. It **uses a low-power wide area network** to intercommunicate between IoT devices via the internet. It supports long-distance communication for sending and receiving small messages.
- For **machine-to-machine (M2M) applications**, Sigfox is the **best long-range network since it uses less power** than other long-range networks. This makes it an **excellent option for connecting remote devices that must operate on batteries** for extended periods without being charged.
- Sigfox operates in the **industrial, scientific and medical (ISM)** bands using an ultra-narrow band (UNB) technology, with a specialized infrastructure.

Ethernet

- Ethernet is a **communication standard** that was developed in the early 80s to network between local devices and computers.
- The local environment is labeled as a local area network(LAN).
- LAN creates a common environment for devices to receive and share information among one another.
- **Ethernet however offers a wired form of communication**. Since it does not offer wireless communication, the set becomes a bit costly and is not the ideal option for IoT communication.

Near Field Communication (NFC)

- NFC stands for near-field communication.
- **It is a wireless technology for short-distance communication.**
- However, the **NFC-enabled devices must be in close proximity to each other** so that they can communicate via radio waves.
- **One of the devices should be an active device such as a smartphone or tablet and the other device can be passive such as an NFC tag.**
- The active devices require an external power supply while the passive devices do not.
- NFC is only limited to the **frequency band of 13.56 MHz**
- Effective working distance: NFC (**less than 10 cm, so it has high security**)

LPWAN

- LPWAN stands for **low power wide area network.**
- It **offers wireless communication between devices** that consume less amounts of power.
- It connects these devices to the internet to send and receive messages from devices within the same network.
- **Some examples of LPWAN are Sigfox and LoRa.**
- LPWAN allows communication across a minimum of 500 meters, consumes little power and enables long-distance connectivity.

LoRaWAN

- **Low Range Wide Area Network** or LoRaWAN is a wide area network protocol.
- It was constructed to connect objects to the internet and to act as a mode of communication between these objects.
- These objects could be **home automation devices, smart cars, thermostats and so on.**
- Millions of **low-power, low-memory devices** are used in smart cities, where this communication protocol is mostly employed.

What is Cloud Computing?

- **Cloud Computing** means **storing and accessing the data and programs on remote servers that are hosted on the internet** instead of the computer's hard drive or local server.
- Cloud computing is **also referred to as Internet-based computing**, it is a technology where the **resource is provided as a service through the Internet to the user**.
- The data that is stored **can be files, images, documents**, or any other storable document.

The following are some of the Operations that can be performed with Cloud Computing

- Storage, backup, and recovery of data
- Delivery of software on demand
- Development of new applications and services
- Streaming videos and audio

What is Cloud Computing?

- Cloud computing helps users in easily accessing computing resources like storage, and processing over internet rather than local hardware. How cloud computing works, explained as:

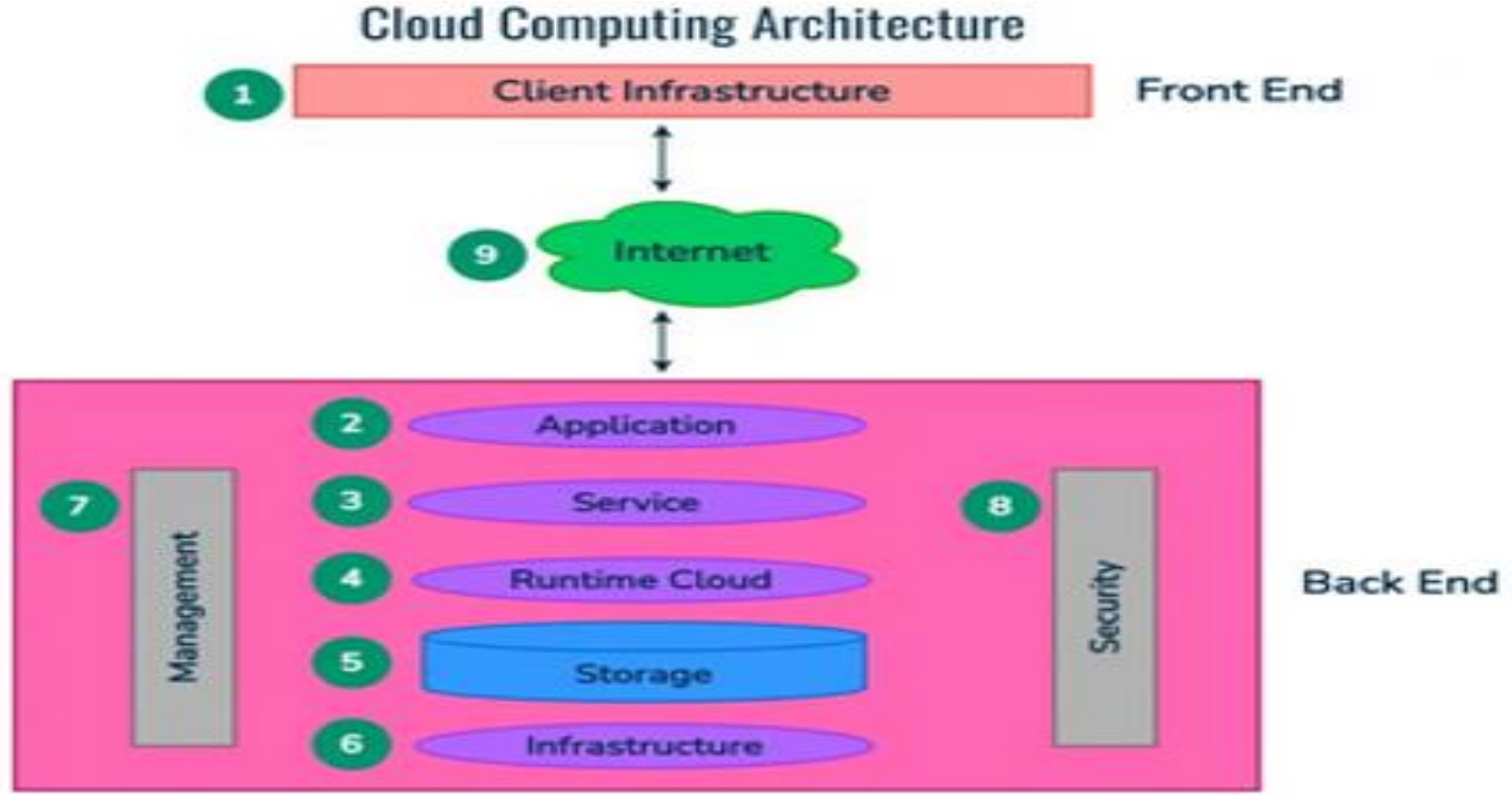
- **Infrastructure:** Cloud computing depends on **remote network servers** hosted on internet for **store, manage, and process the data**.
- **On-Demand Acess:** Users can **access cloud services and resources** based on-demand. They can scale up or down, without having to invest for physical hardware.
- **Types of Services:** Cloud computing offers various benefits such **as cost saving, scalability, reliability and accessibility**. It **reduces capital expenditures, improves efficiency.**

Architecture of Cloud Computing

Cloud computing architecture refers to the components and sub-components required for cloud computing. These components typically refer to:

- **Front end (Fat client, Thin client)**
- **Back-end platforms (Servers, Storage)**
- **Cloud-based delivery and a network (Internet, Intranet, Intercloud)**

Architecture of Cloud Computing



Architecture of Cloud Computing

1. Front End (User Interaction Enhancement):

The User Interface of Cloud Computing consists of 2 sections of clients. **The Thin clients are the ones that use web browsers facilitating portable and lightweight accessibilities** and others are known as **Fat Clients that use many functionalities** for offering a **strong user experience**.

e.g. Microsoft Office tools such as **Word, Excel, and PowerPoint are other examples of fat client applications**. A common **thin client definition is a computer that uses resources housed inside a central server** as opposed to a hard drive.

2. Back-end Platforms (Cloud Computing Engine)

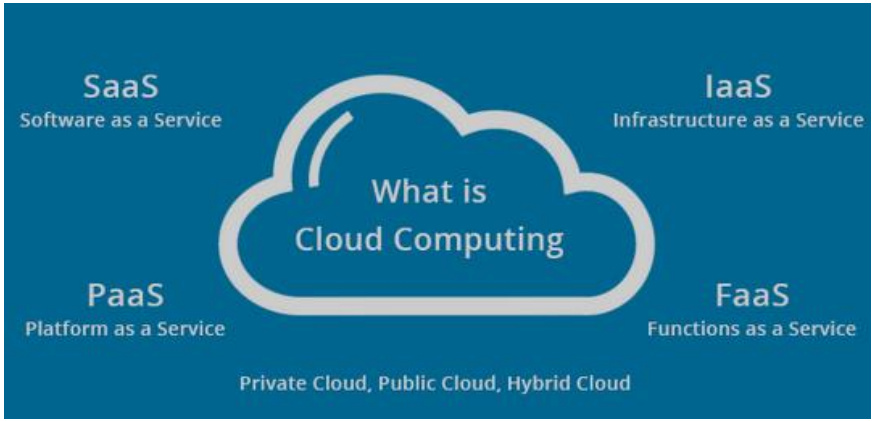
The core of cloud computing is made at **back-end platforms** with several servers for storage and processing computing. Management of **Applications logic is managed through servers and effective data handling is provided by storage**. The combination of **these platforms at the backend offers the processing power, and capacity to manage and store data** behind the cloud.

3. Cloud-Based Delivery and Network

On-demand **access to the computer and resources is provided over the Internet, Intranet, and Intercloud**. The **Internet comes with global accessibility**, the **Intranet helps in internal communications of the services within the organization** and the **Intercloud enables interoperability across various cloud services**. This dynamic network connectivity ensures an essential component of cloud computing architecture on guaranteeing easy access and data transfer.

Cloud Computing Services

1. Infrastructure as a Service (IaaS)
2. Platform as a Service (PaaS)
3. Software as a Service (SaaS)
4. Function as a Service (FaaS)



Cloud Computing Services

1. **Infrastructure as a Service (IaaS):** Typically IaaS is a service **where infrastructure is provided as outsourcing to enterprises** such as networking equipment, devices, database, and web servers. It is also known as Hardware as a Service (HaaS).
2. **Platform as a Service (PaaS):** PaaS is a category of cloud computing that **provides a platform and environment to allow developers to build applications and services over the internet.** PaaS services are hosted in the cloud and accessed by users simply via their web browser.

Cloud Computing Services

3. **Software as a Service (SaaS):** Software-as-a-Service (SaaS) is a way of delivering services and applications over the Internet. **Instead of installing and maintaining software, we simply access it via the Internet**, freeing ourselves from the complex software and hardware management. **It removes the need to install and run applications on our own computers**. The various companies providing Software as a service are Cloud9 Analytics, Salesforce.com, Cloud Switch, Microsoft Office 365, Big Commerce, Eloqua, dropBox, and Cloud Tran.

4. **Function as as Service (FaaS):** It provides a platform for its users or customers to **develop, compute, run and deploy the code or entire application as functions**. It allows the user to entirely develop the code and update it at any time without worrying about the maintenance of the underlying infrastructure. The developed code can be executed with response to the specific event. It is also as same as PaaS.

Cloud Computing Characteristics

The following are the characteristics of Cloud Computing:

Scalability: With Cloud hosting, it is **easy to grow and shrink the number and size of servers based on the need**. This is done by either increasing or decreasing the resources in the cloud. This ability to alter plans due to fluctuations in business size and needs is a superb benefit of cloud computing, especially when experiencing a sudden growth in demand.

Save Money: An advantage of cloud computing is the **reduction in hardware costs**. Instead of purchasing in-house equipment, **hardware needs are left to the vendor**. For companies that are growing rapidly, new hardware can be large, expensive, and inconvenient. Cloud computing alleviates these issues because resources can be acquired quickly and easily.

Reliability: Rather than being hosted on one single instance of a physical server, hosting is delivered on a virtual partition that draws its resource, such as disk space, from an extensive network of underlying physical servers. If one server goes offline it will have no effect on availability, as the virtual servers will continue to pull resources from the remaining network of servers.

Physical Security: The underlying **physical servers are still housed within data centers** and so benefit from the security measures that those facilities implement to prevent people from accessing or disrupting them on-site.

Advantages

- Cost Efficiency
- Flexibility and Scalability
- Collaboration and Accessibility
- Automatic Maintenance and Updates

Disadvantages

- Security Concerns
- Server Down Problems
- Dependency on Internet Connectivity
- Cost Management Complexity

Big Data

- Big data **refers to the data which is huge in size** and also increasing rapidly with respect to time.
- Big data **includes structured data, unstructured data as well as semi-structured data**. Big data can not be stored and processed in traditional data management tools, it needs specialized big data management tools.
- It refers to complex and large data sets having 5 V's **volume, velocity, veracity, value and variety** information assets. It includes data storage, data analysis, data mining and data visualization.

Examples of the sources where big data is generated includes **social media data**, **e-commerce data**, **weather station data, IoT Sensor data** etc.

Big Data

Characteristics of Big Data :

- Variety of Big data – Structured, unstructured, and semi structured data
- Velocity of Big data – Speed of data generation
- Volume of Big data – Huge volumes of data that is being generated
- Value of Big data – Extracting useful information and making it valuable
- Variability of Big data – Inconsistency which can be shown by the data at times.

Types of Big Data

1. Structured

Structured data are defined as the data which can be stored, processed and accessed in a fixed format. Structured data has a fixed schema and thus can be processed easily. We can use SQL to manage structured data.

Example of Structured Data: Data stored in RDBMS.

2. Semi-Structured

Semi-Structured data are the data that do not have any formal structure like table definition in RDBMS, but they have some organizational properties like markers and tags to separate semantic elements thus, making it easier for analysis.

Example of Semi-Structured Data: XML (Extensible Markup Language) files or JSON (JavaScript Object Notation) documents.

3. Unstructured:

Unstructured data have unknown form or structure and cannot be stored in RDBMS. We cannot analyze unstructured data until they are transformed into a structured format. 80 % of the data generated by the organizations are unstructured.

Example of Unstructured Data: Text files, multimedia contents like audio, video, images, etc.

Big Data Examples

- The New York Stock Exchange (NYSE) produces one terabyte of new trade data every day.
- Each day 500 million tweets are sent.
- Amazon, in order to recommend products, on average, handles more than 15 million+ customer clickstreams per day.
- Walmart an American Multinational Retail Corporation handle about 1 million+ customer transactions per hour.
- 65 billion+ messages are sent on Whatsapp every day.
- A single Jet engine generates more than 10 terabytes of data in-flight time of 30 minutes.
- On average, everyday 294 billion+ emails are sent.
- Modern cars have close to 100 sensors for monitoring tire pressure, fuel level, etc. , thus generating a lot of sensor data.
- Facebook stores and analyzes more than 30 Petabytes of data generated by the users each day.
- YouTube users upload about 48 hours of video every minute of the day.

Big Data Applications

Big Data finds applications in many domains in various industries. The article enlisted some of the applications in brief.

1. Bank and Finance: In the banking and Finance sectors, it helps in detecting frauds, managing risks, and analyzing abnormal trading.

2. Agriculture: In agriculture sectors, it is used to increase crop efficiency. It can be done by planting test crops to store and record the data about crops' reaction to different environmental changes and then using that stored data for planning crop plantation accordingly.

3. Advertising and Marketing: Advertising agencies use Big Data to understand the pattern of user behavior and collect information about customers' interests.

4. Media and Entertainment: Media and Entertainment industries are using big data analysis to target the interested audience. They now understand the kind of advertisements that attract a customer as well as the most appropriate time for broadcasting the advertisements to seek maximum attention.

IoT Threats

There are three broad categories of threats: *Capture, Disrupt, and Manipulate*.

- *Capture* threats are related to **capturing** the system or information.
- *Disrupt* threats are related to **denying, destroying, and disrupting** the system.
- *Manipulate* threats are related to **manipulating the data, identity, time-series data**, etc.

IoT Threats

Active and Passive Threats

- Some of the well-known active threats are as follows:
- ***Masquerading***: an entity pretends to be a different entity. This includes masquerading other objects, sensors, and users.
- ***Man-in-the-middle***: when the attacker secretly relays and possibly alters the communication between two entities that believe that they are directly communicating with each other.
- ***Replay attacks***: when an intruder sends some old (authentic) messages to the receiver. In the case of a broadcast link or beacon, access to previous transmitted data is easy.
- ***Denial-of-Service (DoS) attacks***: when an entity fails to perform its proper function or acts in a way that prevents other entities from performing their proper functions.

- Active threats such as masquerading, replay attacks, DoS attacks are, in general, comparatively easy in an IoT environment.

IoT Threats

Active and Passive Threats

- The simplest type of passive threats in the IoT is that of **eavesdropping or monitoring of transmissions** with a **goal to obtain information** that is being transmitted. It is also referred to as capture attacks. Capture attacks are designed to gain control of physical or logical systems or to gain access to information or data items from these systems.
- The physical distribution of the IoT objects and systems provide attackers with great opportunity to gain control of these systems.
- The distribution of smart objects, sensors, and systems results in self-advertisements, beacons, and mesh communications, providing attackers greater opportunity to intercept or intercede in information transmission within the environment.

Vulnerabilities in IoT Systems

1. Weak, guessable, or hardcoded passwords

- Weak, default, and hardcoded passwords are the **easiest way for attackers to compromise IoT devices** and **launch large-scale botnets**, and other malware.
- Managing passwords in a distributed IoT ecosystem is a time-consuming and difficult responsibility, especially since IoT devices are managed over-the-air.

2. Insecure network services

Unneeded or insecure network services running on the device itself, especially those exposed to the internet, that compromise the

- confidentiality
- integrity/authenticity
- availability of information
- allow unauthorized remote control.

Vulnerabilities in IoT Systems

3. Insecure ecosystem interfaces

Insecure web, backend API, cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. Common issues include

- a lack of authentication/authorization
- weak encryption
- a lack of input and output filtering.

4. Lack of secure update mechanism: This includes

- lack of firmware validation on device
- lack of secure delivery (un-encrypted in transit)
- lack of anti-rollback mechanisms
- lack of notifications of security changes due to updates.

Vulnerabilities in IoT Systems

5. **Use of insecure or outdated components**

Use of insecure software components/libraries that could allow the device to be compromised. This includes

- insecure customization of operating system platforms
- use of third-party software or hardware components from a compromised supply chain.

6. **Insufficient privacy protection**: User's personal information stored on the device or in the ecosystem that is used insecurely, improperly, or without permission.

7. **Insecure data transfer and storage**: Lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.

8. **Lack of device management**: Lack of security support on devices deployed in production, including asset management, update management, systems monitoring, and response capabilities.

Vulnerabilities in IoT Systems

9. **Insecure default settings**: Devices or systems shipped with insecure default settings lack the ability to make the system more secure by restricting operators from modifying configurations.

10. **Lack of physical hardening**: Lack of physical hardening measures, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device.

Botnet: a network of private computers infected with malicious software and controlled as a group without the owners' knowledge, e.g. to send spam.

Anti-rollback mechanisms: Prevents the execution of previous versions of authenticated firmware that might have security flaws.

Network and Transport Layer Challenges

- The IPSec uses the concept of a **Security Association (SA),** defined as the set of algorithms and parameters (such as keys) used to encrypt and authenticate a particular flow in one direction.
- To establish a SA, IPSec can be preconfigured (specifying a preshared key, hash function, and encryption algorithm) by the IPSec **Internet Key Exchange** (IKE) protocol.
- The IKE protocol uses **asymmetric cryptography**, **which is computationally heavy for resource-constrained devices.**
- To address this issue**, IKE extensions using lighter algorithms should be used.**
- **Data overhead is another problem** for IPSec implementations in IoT environments. This is introduced by the extra header encapsulation of IPSec AH (Authentication Header) and/or Encapsulating Security Payload (ESP), and can be mitigated by using header compression.

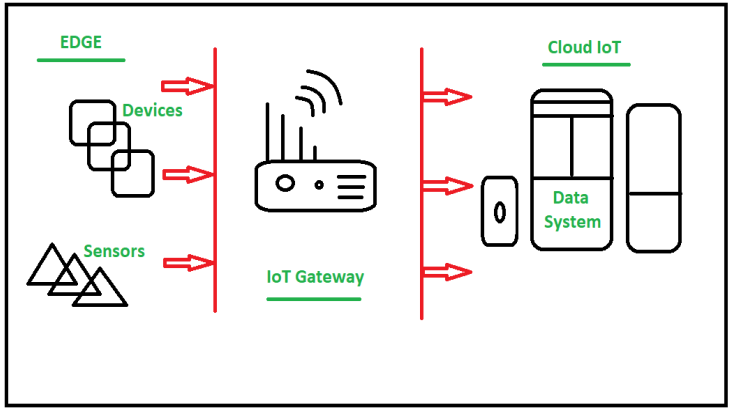
- Cryptography **is** a technique of securing communication by converting plain text into ciphertext
- An Authentication Header (AH) is a security protocol in IPSec that **ensures the integrity of packet headers and data, provides user authentication, and offers optional replay protection and access protection**. It does not encrypt any part of the packets.
- The difference between ESP and the Authentication Header (AH) protocol is that **ESP provides encryption**, while both protocols provide authentication, integrity checking, and replay protection.

- CoAP proposes to use the DTLS protocol to provide end-to-end security in IoT systems.
- The DTLS protocol provides a security service similar to TLS, but on top of UDP. This is highly suitable for IoT environments due to its usage of UDP as transport protocol.
- This results in avoidance of problems from the use of TCP in network-constrained scenarios that are caused due to the extremely variable transmission delay and loss links.
- DTLS headers are too long to fit in a single IEEE 802.15.4 MTU (Maximum Transmission Unit).
- 6LoWPAN provides header compression mechanisms to reduce the size of upper layer headers. 6LoWPAN header compression mechanisms can be used to compress the security headers as well.

- **Datagram Transport Layer Security** (*DTLS*) is a communications *protocol* providing security to datagram-based applications**.**
- ***CoAP (Constrained Application Protocol)*** is an application-layer protocol that is intended for use in resource-constrained Internet devices
- **Transport Layer Security**, or TLS, is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet. A primary use case of TLS is encrypting the communication between web applications and servers, such as web browsers loading a website.
- **User Datagram Protocol (UDP)** is a communications protocol for time-sensitive applications like gaming, playing videos, or Domain Name System (DNS) lookups.
- **Transmission Control Protocol**, is a communications standard for delivering data and messages through networks

IoT GATEWAYS AND SECURITY

- Gateway acts as a **medium to open up connections between the cloud and controller**(sensors/devices) in Internet of Things (IoT).
- Gateway serves **as the connection point** between the cloud and IoT devices, such as controllers, sensors and smart devices.
- With the help of gateways, it is possible to establish **device-to-device** or **device-to-cloud** communication.
- A gateway can be a **typical hardware device** or **software program**.
- IoT gateways act as a **central hub**, connecting IoT devices to the cloud.



IoT GATEWAYS

- Organizations can use gateways to connect IoT devices **for data processing as well as to monitor and manage IoT devices.**
- An IoT gateway acts **as a network router**, routing data between IoT devices and the cloud. Early on, most gateway devices only sent traffic in one direction: from the IoT device to the cloud. Now **it's common for gateway devices to handle both inbound and outbound traffic**. Outbound traffic streams are used to send IoT data to the cloud, while inbound traffic is used for device management tasks, such as updating firmware.
- Some IoT gateways do more than just route traffic, they **can also preprocess data locally at the edge** before sending it to the cloud. In doing so, **the device might duplicate, summarize or aggregate data as a way of reducing the volume of data** that must be forwarded to the cloud. This can **improve response times** and **reduce network transmission costs**.

Key functionalities of IoT Gateway

- Establishing communication bridge
- Provides additional security.
- Performs data aggregation.
- Pre processing and filtering of data.
- Provides local storage as a cache/ buffer.
- Data computing at edge level.
- Ability to manage entire device.
- Device diagnostics.
- Adding more functional capability.
- Verifying protocols.

Working of IoT Gateway

- Receives data from sensor network.
- Performs Pre processing, filtering and cleaning on unfiltered data.
- Transports into standard protocols for communication.
- Sends data to cloud.

IoT GATEWAYS & SECURITY

- A simple IoT gateway **functions similarly to a Wi-Fi router**. More often, though, **IoT gateways are far more complex.**
- **IoT also comes with some security risks**, as it increases an organization's attack surface.
- IoT gateways should sit between IoT devices and the internet as well as have integrated security functions. These functions, such as **tamper detection, encryption and hardware random number generation,** should protect the IoT devices from being attacked.
- Likewise, **gateway filtering technology can monitor, manage and secure data transfers** through authenticated traffic **using packet filtering or physical network signal filtering**.
- To further improve IoT gateway security, organizations can do the following:
  - **Use only authenticated IoT gateways.**
  - **Perform security assessments before implementation.**
  - **Keep gateway software updated.**
  - **Regularly review gateway access.**
  - **Include gateways in security audits.**
  - **Use a separate network for IoT gateways and devices.**

IoT ROUTING ATTACKS

- Threats **arising due to the physical nature of IoT devices** can be mitigated by appropriate **physical security safeguards**, whereas **secure communication protocols and cryptographic algorithms** are the only way of coping with the fact that they **arise due to IoT devices communicating** with each other and the external world.
- IoT devices can **either run the standard TCP/IP protocol stack**, if their computational and power resources allow, **or can run adaptions which are optimized for lower computational and power consumption.**
- Some examples of routing attacks include: **Sinkhole attacks, Selective forwarding attacks, Blackhole attacks, Sybil attacks, and DDoS/DoS attacks.**
- Some techniques for detecting routing attacks include: Intrusion detection systems (IDS), Hybrid IDS, and One-class SVM.

Sinkhole Attack

- An attacker **launches an attack on a network by adding bogus nodes.**
- A Sinkhole Attack is an internal assault in which an **impacted node (sinkhole) attempts to attract network traffic by propagating bogus routing updates**.
- The **objective of this attack is to reroute traffic from a certain region across a malicious node** that appears to be highly appealing to the surrounding nodes.

Selective Forwarding Attack

- A Selective Forwarding Attack is a security threat wherein **an attacker manipulates network traffic by selectively forwarding or dropping packets.**
- The **attacker gains control over one or more nodes** in the network and **selectively forwards packets to some nodes while dropping or delaying packets to others.** This disrupts the communication between the nodes and can compromise the security of the network.
- The attacker **can selectively drop packets containing critical information, leading to data loss and the disruption of the communication between nodes**. This type of attack can be **particularly harmful in WSNs and IoT networks** where **nodes have limited resources** and rely on efficient and reliable communication. To ensure network security and reliability, effective defense mechanisms must be developed to detect and prevent selective forwarding attacks.

Sybil Attack

- A Sybil Attack is a type of attack in which **a malicious attacker creates multiple fake identities or nodes in a network to gain control and manipulate the network.**
- In the context of RPL, a Sybil attack involves a malicious node creating multiple fake identities or nodes that can participate in the network and influence the routing process. These **fake nodes can cause routing loops, interfere with routing decisions, and lead to network congestion**, which can ultimately **result in denial of service (DoS) attacks**.
- This type of attack is **particularly dangerous in peer-to-peer networks and distributed systems where trust is critical**.

Blackhole Attack

- A Blackhole Attack is a security threat that **disrupts the routing process by selectively forwarding or dropping network packets.**
- The **attacker pretends to have the shortest path to the destination node**, causing **legitimate nodes to forward packets to the attacker** instead of the actual destination.
- Once the **attacker has control over the packets**, they can be **selectively dropped or forwarded to other nodes**, resulting in severe damage to the network. The **consequences of a Blackhole Attack** can be significant, including **communication disruption, data loss, and network failures**.
- In RPL, which is commonly used in IoT networks, **a blackhole attack can be particularly damaging** since many IoT devices have limited resources and rely on reliable and efficient communication to function properly.

Wormhole Attacks

- A wormhole is **an out-of-band connection between two nodes** using wired or wireless links.

- Wormholes can be **used to forward packets faster than via normal paths**. A **wormhole created by an attacker and combined with another attacks**, such as **sinkhole**, is a serious security threat.

AUTHORIZATION MECHANISMS

- The present day services that run over the Internet, such as **popular social media applications**, have **faced and handled privacy-related problems** when dealing with personal and protected data that might be made accessible to third parties.
- In the future, the **IoT applications will face similar issues**, and others that may be unique to the domain.
- The **OAuth (Open Authorization) protocol** has been defined to **solve the problem of allowing authorized third parties to access personal user data.**
- **OAuth2.0** is an authorization framework **that allows a third party to access a resource owned by a resource owner without giving unencrypted credentials to the third party.**
- For example, assume that **a healthcare sensor or mobile app wants to access a Facebook profile** to post status updates. There is **no need to provide the Facebook credentials to the app**; instead, the user logs into Facebook, and as a result the **app is authorized to use Facebook on the user's behalf**. The **user can also revoke this authorization any time** by deleting the privilege in the Facebook settings.

AUTHORIZATION MECHANISMS

The OAuth 2.0 protocol defines the following four roles:

- **Resource Owner:** It is an **entity capable of granting access to a protected resource**. When the **resource owner is a person, it is referred to as an end user**. In the above example, this could be the end user of the healthcare device.
- **Resource Server (Service Provider, SP):** It is the server **hosting the protected resources**, **capable of accepting and responding to protected resource requests using access tokens**. In the example, this is the **Facebook server.**
- **Client (Service Consumer, SC):** It is **the application making protected resource requests** on behalf of the resource owner and with its authorization. In this case, it is the **healthcare sensor or mobile application**.
- **Authorization Server:** It is the **server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization**. In this example, it would be the **Facebook authorization server**

Lightweight Cryptography

- The Internet of Things (IoT) will soon penetrate every aspect of human life.
- **Several threats and vulnerabilities** are present due to the different devices and protocols used in an IoT system.
- **Conventional cryptographic algorithms cannot** run efficiently and are unsuitable for resource-constrained devices in IoT.
- Hence, a recently developed area of cryptography, **known as lightweight cryptography**, has been introduced, and over the years, numerous lightweight algorithms have been suggested.

Lightweight Cryptography

- The LWC algorithms: categorized into two groups, **based on the number of keys** being used.
- Algorithms using the **same key** (secret key or private key) for encryption and decryption are called **symmetric algorithms**.
- Algorithms using **different keys** (private and public keys), one for encryption and the other for decryption, are called **asymmetric algorithms**.
- Any LWC algorithm should satisfy four conditions: **confidentiality, integrity, authentication, and non-repudiation** for accepting it as a perfectly secure algorithm.
- Symmetric algorithms provide **confidentiality, integrity, and authentication** security services.
- In contrast, asymmetric algorithms provide **confidentiality and integrity** by using the receiver's public key and **authentication** (in the form of a digital signature) and **non-repudiation** by using the sender's private key.
- The **downside** of **symmetric algorithms** is the **key distribution**, and for **asymmetric algorithms**, it is the **usage of large keys** (number of bits) for encryption and decryption.

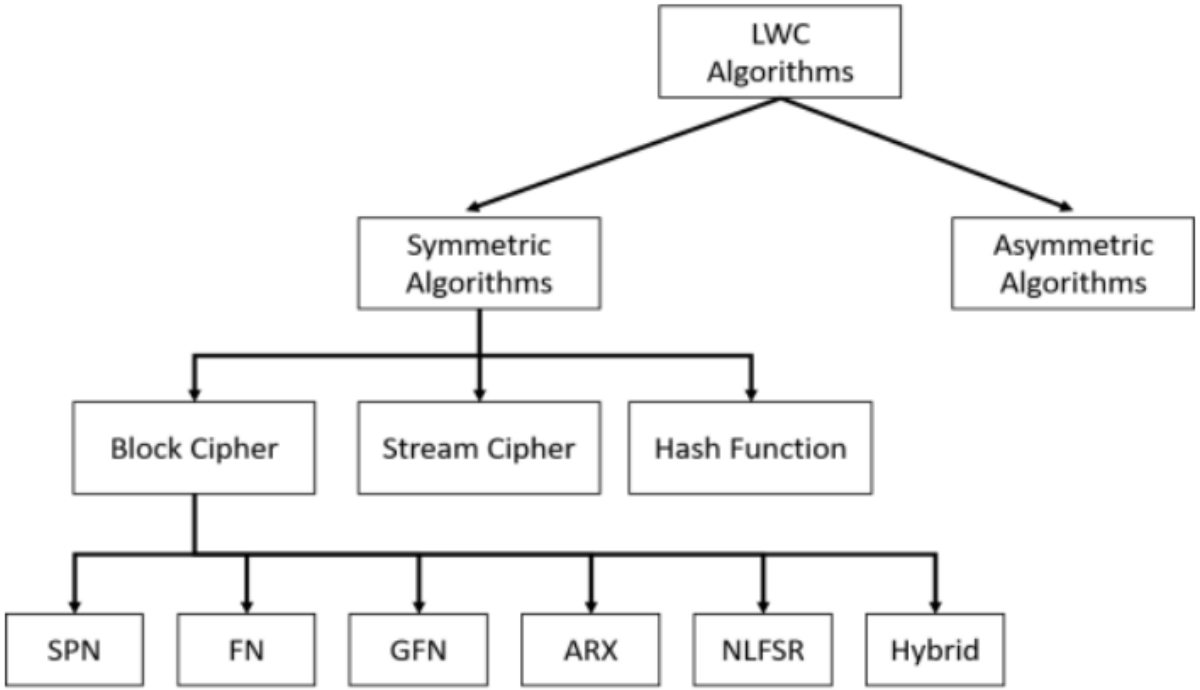
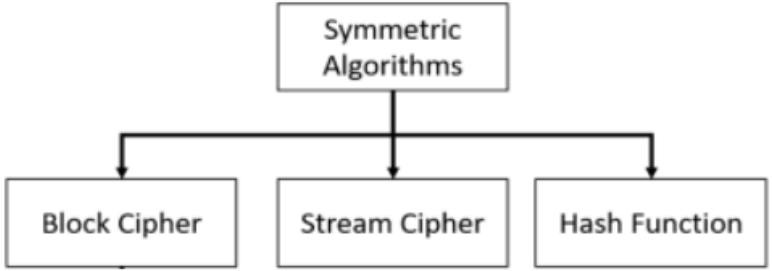


Fig. Classification of LWC algorithms

Symmetric Key LWC Algorithms

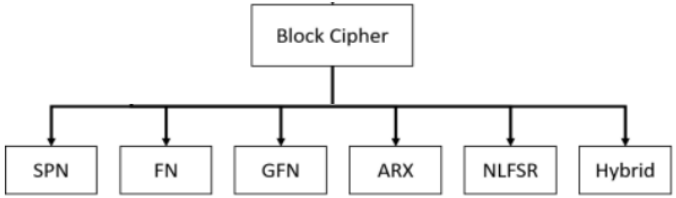
- The symmetric algorithms can be further divided into **block** and **stream ciphers** based on how the plaintext is processed.
- A block cipher **processes the plaintext as fixed-size blocks** and generates the ciphertext.
- On the other hand, a stream cipher **takes the plaintext as bits and performs operations on the bits** to generate ciphertext.
- Another category of symmetric algorithms is a **hash function**, which **takes a variable length text and generates a fixed size output, also called hash or digest**. A hash is a one-way function; the generated hash cannot be used to get the original plaintext. A hash is generally used for enforcing integrity.



Block Cipher & Stream Cipher

- The two fundamental operations performed by a block cipher are **confusion** and **diffusion**.
- The **confusion** operation **removes the dependencies between the key and the generated ciphertext** to prevent statistical analysis attacks.
- The **diffusion** operation **removes the dependencies between the plaintext and ciphertext** to prevent statistical analysis and guessing attacks.
- **Block cipher uses both confusion and diffusion** operations.
- **Stream cipher only uses confusion operations**. So, the security of a block cipher is more than a stream cipher.

Symmetric Key LWC Algorithms



- A block cipher internally uses a specific structure for achieving security.
- Different structures used: **Substitution-Permutation Network** (SPN), **Feistel Network** (FN), **General Feistel Network** (GFN), **Add-RotateXOR** (ARX), **Non-Linear Feedback Shift Register** (NLFSR), and **Hybrid structure**
- **SPN**: Processes the data by performing different operations using S-boxes (Substitution Boxes) and permutation tables.
- **FN**: divides the text into equal halves, and on one half, it applies diffusion. Also, at the beginning of each round, it swaps the two halves.
- **GFN**: divides the plaintext into sub-blocks and performs operations on the sub-blocks. It also applies a cyclic shift proportional to the number of created sub-blocks.
- **ARX**: uses addition, rotation, and XOR operations for encryption and decryption.
- **NLFSR**: uses the basic building blocks of a stream cipher.
- **Hybrid structure**: uses a combination of the structures mentioned above based on the application's requirements.

Tiny Encryption Algorithm (TEA)

- Block cipher & FN-based algorithm
- Requires **1350 bytes of ROM** and only **13 bytes of RAM.**
- uses a large number of iterations, rather than a complicated program, in order to avoid preset tables and long setup times
- uses **128-bit key**
- TEA family: XTEA (extended TEA), XXTEA, Block TEA, Speed TEA, and Tiny TEA
- uses very simple operations (addition, XORing, shifts), and has a very small code size; ideal candidate for small objects and wireless sensors

Scalable Encryption Algorithm (SEA)

- Block cipher & FN-based algorithm
- implementation in assembly language
- on-the-fly derivation of keys,
- efficient encryption and decryption.
- SEA requires 2800 bytes of ROM and only 24 bytes of RAM.

PRESENT

- PRESENT is an ultra-lightweight block cipher algorithm based on a Substitution-Permutation Network (SPN) structure.
- PRESENT has been designed to be extremely compact and efficient in hardware.
- It operates on 64-bit blocks and with keys of either 80 or 128 bits.
- It is for use in situations where low-power consumption and high chip efficiency are desired, thus making it of particular interest for constrained environments.

HIGHT

- The HIGh security and lightweigHT (HIGHT) encryption algorithm is a generalized Feistel network with a block size of 64 bits, 128-bit keys.
- HIGHT was designed with an eye on low-resource hardware performance.
- HIGHT uses very simple operations, such as XORing, addition and bitwise rotation

Asymmetric LWC Algorithms

- Public-key (asymmetric) cryptography requires the use of a **public-key** and a **private key**. Asymmetric cryptography requires higher processing and long keys (at least 1024 bits for RSA) to be used.
- Alternative public-key cryptographic schemes, such as ECC, might **require shorter keys** to be used in order to achieve the same security than RSA keys. However, because of these reasons, **symmetric cryptography is preferred** in terms of **processing speed, computational effort, and size of transmitted messages.**
- Lightweight cryptography algorithms are suitable for environments that do not have stringent security requirements and where the constraints on available hardware and power budget cannot be relaxed.

Asymmetric LWC Algorithms

Elliptic Curve Cryptography (ECC)

- **provides higher security** and a **better performance** than the first generation public-key techniques (RSA and Diffie-Hellman).
- the most interesting public-key cryptographic family for embedded environments
- can reach the same security level as RSA with much shorter keys
- computationally lighter operations, like addition and multiplication, rather than exponentiation