Course Code	Course Title	L	T	P	C
BECE351L	IoT Fundamentals	1	0	2	2
Pre-requisite	Nil	Syllabus version			
		1.0			

Course Objectives

1. To impart knowledge on the infrastructure, communication and networking technologies of IoT.
2. To analyse, design and develop Industrial IoT solutions.
3. To develop IoT architecture for use cases under discussion.

Course Outcomes

1. To focus on the technologies that enable IoT and to interpret the different components in IoT architecture.
2. Comprehend the concepts of edge computing and edge enabled solutions for real-time industrial applications.
3. Envision the IoT architecture models and the protocol stack for the design and development of IoT applications on different platforms.
4. Interpret the security threats and to design a resilient IoT Architecture.
5. Program the controller and sensors as part of IoT
6. Assess different Internet of Things technologies and their applications
7. To apply the concepts of Internet of Things in the real world scenarios

Module:1	Essentials of Internet of Things	2 hours

IoT Emergence, Definition and Characteristics of IoT, Impact of IoT on business and society, IoT product development life cycle, IoT enabling Technologies, Applications.

Module:2	Architecture Reference Model	2 hours

Domain Model, Information Model, Functional Model – Communication and security model, SOA based architecture.

Module:3	Protocol Suite	2 hours

Physical layer, Link layer -BLE, LoRAWAN, Network layer, Transport layer, Application Layer protocols - MQTT, CoAP – Communication Models.

Module:4	Edge Computing	2 hours

Introduction to Edge/Fog computing, Front end Edge Devices, Gateway, Edge ML for Industry automation.

Module:5	Security Engineering	2 hours

IoT Attacks and Security Challenges, Threat and Mitigating Threats to IoT Systems, Privacy concerns - Access control, Lightweight Cryptography, Privacy in IoT

Module:6	IoT Platforms for Usecase Development	2 hours

Open source IoT platforms and services, Communication API's- REST, Websocket, Scalability of IoT Solutions

Module:7	IoT Verticals	1 hours

Roadmap for developing complete IoT solutions; Smart Cities, Healthcare,

Agriculture and Farming		
Module:8	**Contemporary Issues**	**2 hours**
Guest lectures from Industry and R&D organizations.		
	Total Lecture hours:	**15 hours**

Text Book(s)

1. Arshdeep Bahga, Vijay Madisetti, "Internet of Things: A hands-on Approach", University Press, 2015.
2. Ammar Rayes,Samer Salam, " Internet of Things from Hype to Reality- A road to Digitization" Second Edition, Springer, ISBN 978-3-319-99515-1.
3. Rajkumar Buyya,Amir Vahid " Internet of Things Principles and Paradigms", Elsevier, 2016.

Reference Books

1. Brian Russell, Drew Van "Practical Internet of Things Security "Packt Publishing, ISBN 978-1-78588-963-9 ,2016.
2. Adrian McEwen & Hakim Cassimally, "Designing the Internet of Things", Wiley, 2017 , Second Edition.

Mode of Evaluation: Continuous Assessment Test, Digital Assignment, Quiz and Final Assessment Test

List of Experiments (Indicative)

1. IoT based soil health Monitoring
2. Air Quality monitoring system
3. Smart Parking System using an appropriate IoT visualization services
4. IoT based Healthcare and fitness monitoring
5. Real-time environmental weather prediction
6. IoT enabled accident prevention and detection system
7. Smart Street light system
8. Plant health monitoring using a suitable IoT platform and services
9. Build a web based application to automate the door that unlocks itself using facial recognition.
10. Intelligent Traffic light control system for ambulance services

Total Hours			**30 Hours**
Mode of assessment: Continuous assessment and FAT			
Recommended by Board of Studies	28.02.2023		
Approved by Academic Council	No. xx	Date	DD-MM-YYYY

Module:1 Essentials of Internet of Things

1.1 IoT Emergence

- Internet of Things (IoT) is a concept which enables communication between internetworking devices and applications, whereby physical objects or 'things' communicate through the Internet.
- The concept of IoT began with things classified as identity communication devices. Radio Frequency Identification Device (RFID) is an example of an identity communication device. Things are tagged to these devices for their identification in future and can be tracked, controlled and monitored using remote computers connected through the Internet.
- The concept of IoT enables, for example, GPS-based tracking, controlling and monitoring of devices; machine-to-machine (M2M) communication; connected cars; communication between wearable and personal devices and Industry 4.0.

1.2 Definition and Characteristics of IoT

- **"Internet of Things means a network of physical things (objects) sending, receiving, or communicating information using the Internet or other communication technologies and network just as the computers, tablets and mobiles do, and thus enabling the monitoring, coordinating or controlling process across the Internet or another data network."**
- **"Internet of Things is the network of physical objects or 'things' embedded with electronics, software, sensors and connectivity to enable it to achieve greater value and service by exchanging data with the manufacturer, operator and/or other connected devices. Each thing is uniquely identifiable through its embedded computing system but is able to interoperate within the existing Internet infrastructure."**

Example: Streetlights in a city can be made to function like living entities through sensing and computing using tiny embedded devices that communicate and interact with a central control-and-command station through the Internet. Assume that each light in a group of 32 streetlights comprises a sensing, computing and communication circuit. Each group connects to a group-controller (or coordinator) through Bluetooth or ZigBee. Each controller further connects to the central command-and-control station through the Internet. The station receives information about each streetlight in each group in the city at periodic intervals. The information received is related to the functioning of the 32 lights, the faulty lights, about the presence or absence of traffic in group vicinity, and about the ambient conditions, whether cloudy, dark or normal daylight. The station remotely programs the group controllers, which automatically take an appropriate action as per the conditions of traffic and light levels. It also directs remedial actions in case a fault develops in a light at a specific location. Thus, each group in the city is controlled by the 'Internet of streetlights'.

Characteristics of IoT

Internet of Things refers to the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, and network connectivity, allowing them to collect and exchange data.

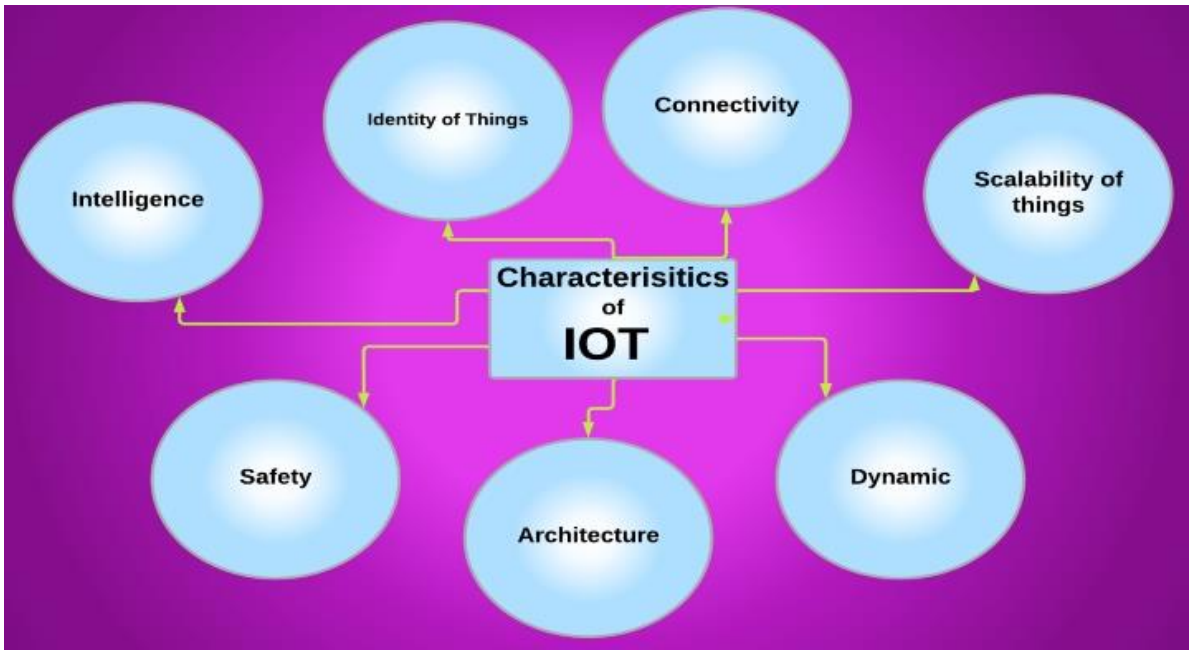


Fig.1.1 Characteristics of IoT

The IoT enables these devices to interact with each other and with the environment and enables the creation of smart systems and services.

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).

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.

- Interoperability is critical for the success of IoT, as it enables different devices and systems to work together seamlessly and provides a seamless user experience. Without interoperability, IoT systems would be limited to individual silos of data and devices, making it difficult to share information and create new services and applications.

- To achieve interoperability, IoT devices, and systems use standardized communication protocols and data formats. These standards allow different devices to understand and process data in a consistent and reliable manner, enabling data to be exchanged between devices and systems regardless of the technology used.

9. Embedded Sensors and Actuators: Embedded sensors and actuators are critical components of the Internet of Things (IoT). They allow IoT devices to interact with their environment and collect and transmit data.

- Sensors are devices that can detect changes in the environment, such as temperature, light, sound, or movement. In IoT systems, sensors are embedded into devices, allowing them to collect data about the environment.
- Actuators are devices that can interact with the environment, such as turning on lights, opening or closing doors, or controlling the speed of a motor. In IoT systems, actuators are embedded into devices, allowing them to perform actions based on data collected by sensors.
- Together, sensors and actuators allow IoT devices to collect data about the environment, process that data, and take action based on the results. This makes it possible to automate a wide range of processes and tasks, such as home automation, energy management, and predictive maintenance.

- In order to ensure that sensors and actuators can communicate with each other and with other devices and systems, they use standardized communication protocols, such as Bluetooth Low Energy (BLE), Zigbee, or Wi-Fi.

Examples of standards used in IoT

- **MQTT (Message Queuing Telemetry Transport):** MQTT (Message Queuing Telemetry Transport**)** is a publish/subscribe communication protocol used for IoT device communication.
- **CoAP (Constrained Application Protocol):** CoAP (Constrained Application Protocol) is a lightweight communication protocol for IoT devices with limited resources.
- **Bluetooth Low Energy (BLE):** Bluetooth Low Energy is a wireless communication technology used for IoT devices with low power consumption requirements.
- **Wi-Fi:** A wireless communication technology used for IoT devices that require high data transfer rates.
- **Zigbee:** A low-power, low-cost wireless communication technology used for IoT devices.
- In addition to communication protocols, IoT systems may also use data formats such as JSON or XML to ensure that data can be exchanged and processed consistently across different systems.

10. Autonomous operation: Autonomous operation is made possible by advances in artificial intelligence, machine learning, and cloud computing, which enable IoT devices and systems to process and analyze large amounts of data in real time and make decisions based on that data.

- Overall, the autonomous operation is an important characteristic of IoT systems, allowing them to deliver new and innovative services and applications that can improve efficiency, reduce costs, and enhance the user experience. IoT devices are designed to operate autonomously, without direct human intervention, making it possible to automate a wide range of processes and tasks.

11. Data-driven: Data-driven is a key characteristic of the Internet of Things (IoT). IoT devices and systems collect vast amounts of data from sensors and other sources, which can be analyzed and used to make data-driven decisions.

- In IoT systems, data is collected from embedded sensors, actuators, and other sources, such as cloud services, databases, and mobile devices. This data is used to gain insights into the environment, improve operational efficiency, and make informed decisions.

12. Security: Security is a critical concern for the Internet of Things (IoT), as IoT devices and systems handle sensitive data and are connected to critical infrastructure. The increasing number of connected devices and the amount of data being transmitted over the Internet make IoT systems a prime target for cyberattacks. To secure IoT systems, multiple layers of security are necessary, including physical security, network security, and data security.

13. Ubiquity: Ubiquity refers to the widespread and pervasive presence of the Internet of Things (IoT) devices and systems in our daily lives. The goal of IoT is to create a seamless and interconnected world where devices and systems can communicate and share data seamlessly and transparently. Ubiquity is achieved through the widespread deployment of IoT devices, such as sensors, actuators, and other connected devices, as well as the development of IoT networks and infrastructure to support communication and data exchange.

14. Context Awareness: Context awareness refers to the ability of Internet of Things (IoT) devices and systems to understand and respond to the environment and context in which they are operating. This is achieved through the use of sensors and other technologies that can detect and collect data about the environment. Context awareness is a critical aspect of IoT, as it enables IoT devices and systems to make decisions and take actions based on the context in which they are operating.

Components of IoT

- **Device:** An IoT Device has capabilities like unique identity, sensing, actuating, and monitoring. Wireless sensors, applications, actuators, and computer devices are examples of IoT devices.

These are linked to a specific object that communicates through the internet, allowing data to be transferred between objects or people automatically and without the need for human interference.

- **Resource:** Every IoT has a software module for the entry, processing, and storage of sensor data. They are therefore used to control actuators connected to devices.
- **Controller Service:** It acts as a connector between web service and device. The controller service sends data from the system to the web service and receives commands for controlling the device from the application (via web services).
- **Database:** It is the repository for all data provided by IoT devices.
- **Web Service:** Web services connect the IoT computer, application, database, and research components. Web services may be implemented either using HTTP and REST concepts (REST service) or the WebSocket protocol (WebSocket service).
- **Analysis Component:** It retrieves data from the IoT device's database and turns it into useful information. This module analyses data, produce results and presents them in a user-friendly format using various algorithms.
- **Application:** IoT applications have a user-friendly interface to track and manage different IoT device aspects. Users will access the Monitor system and the data generated.

1.3 Impact of IoT on business and society

Impact will IoT have in the Society?

The impact of the Internet of Things on the economy is expected to be significant. Here are some of the ways in which IoT is likely to affect the economy:

- **Job creation:** The development and deployment of IoT technology is expected to create new job opportunities in areas such as software development, data analysis, and engineering.
- **Increased productivity:** IoT can help businesses optimize their operations and reduce waste, leading to increased productivity and profitability.
- **New business models:** IoT can enable businesses to develop new products and services, such as smart home devices, wearable technology, and connected cars.
- **Improved customer experience:** IoT can help businesses improve their customer experience by providing personalized and convenient services, such as remote monitoring and control of devices.
- **Cost savings:** IoT can help businesses reduce their costs by improving their supply chain management, reducing downtime, and optimizing their energy usage.
- **New revenue streams:** IoT can create new revenue streams for businesses by enabling them to sell data collected from connected devices to other businesses, such as insurance companies.

Business

The Internet of Things (IoT) has had a significant impact on society in several ways:

1. Improved efficiency: The IoT has made it possible to automate processes and connect devices, leading to increased efficiency in industries such as manufacturing, logistics, and healthcare.
2. Enhanced convenience: IoT has made it possible for people to remotely control devices such as thermostats, lighting, and security systems, making their lives more convenient.
3. Better health outcomes: IoT has enabled the creation of wearable devices that monitor vital signs and provide real-time data to healthcare professionals, helping them to provide better care.
4. Enhanced safety and security: IoT can help monitor and respond to emergency situations more quickly and effectively, improving public safety. For example, smart home security systems can provide remote monitoring and control, while smart city technology can detect and respond to natural disasters, traffic congestion, and other emergencies.
5. Environmental benefits: IoT can help reduce waste and conserve energy by optimizing resource utilization and reducing emissions. For example, smart home technology can reduce energy usage and smart city technology can optimize public transportation, reducing traffic and air pollution.

6. New business opportunities: IoT has created new business opportunities in areas such as smart homes, wearable technology, and industrial automation, leading to job creation and economic growth.

1.4 IoT product development life cycle

1. • INCEPTION: UNDERSTANDING, GATHERING, AND REALIZING THE POSSIBILITY.
2. • DESIGN: TRANSFORMATION OF THE IDEA INTO A CIRCUIT DESIGN.
3. • REVIEW: REVIEWING THE CIRCUIT DESIGN AND FUNCTIONALITY AT EVERY POINT.
4. • IOT PROTOTYPE: BUILDING A PROOF OF CONCEPT.
5. • MANUFACTURING: FABRICATION OF THE PRINTED CIRCUIT BOARD AND IT'S ASSEMBLY.
6. • TESTING
7. • APPLICATION BUILDING (Software)
8. • INTEGRATING THE HARDWARE WITH SOFTWARE
9. • TESTING AND VALIDATING THE INTEGRATED PRODUCT
10. • DEPLOYMENT

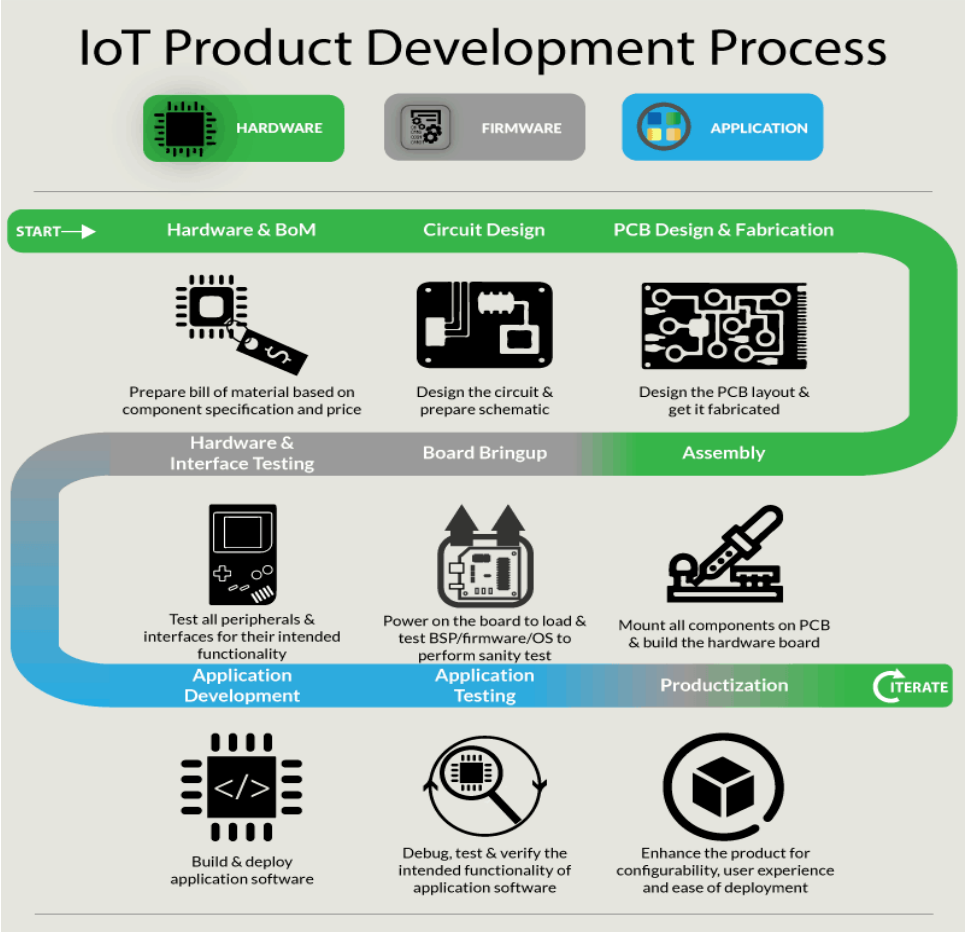


Fig.1.2 IoT Product Development Cycle

- **Data collection:** The first step to developing an IoT solution is understanding the needs and demands of the manufacturer. Hence, the developers collect as much information as possible from the client regarding the expectations from the project.
  It mainly consists of the details of the power and size requirements of the product. The developers analyze the information provided, arrive at an idea, and plan to depend on the functionality of the IoT device.
- **Design:** After the customer brainstorms the requirements of the product, here comes some engineering. The engineers convert the idea into a prototype by developing a circuit design for the product.
  Designing a circuit requires various software knowledge and algorithms to arrive at an appropriate solution for the product based on the real-world market. Some important factors in the process are range, battery life, and product cost.
  The best solution based on the cost and performance ratio is selected and implemented in the final project for the IoT device.
- **Review:** Once the most appropriate circuit design is formulated, the developers must continuously make necessary changes. It is possible by reviewing the circuit design and functionality throughout the project.
  The developers can make changes to the layout, schematics, algorithms, or infrastructure of the project to arrive at a reliable solution with the highest performance and cost efficiency.
- **Prototyping:** Here comes the stage where the circuit design implementation is carried out. The developers come forward with a proof of concept for the IoT solution by building the actual product by combining the hardware and software components.
  Considering the cost/performance ratio and form factor required by the customer, the developers test the various components, such as sensors, simulators, embedded boards, modules, etc. Minimizing the error at the end of prototyping is the main aim of this step.
- **Validation:** Testing and validating the final prototype are essential steps of the IoT lifecycle. Here the hardware component of the prototype is tested under different parameters, such as amplitude, magnitude, voltage, power consumption, temperature, etc. Once validated, the product is all set to be manufactured.
- **Manufacturing and Maintenance:** The final prototype is forwarded to the manufacturer. The manufacturing step involves the assembly of the various components of the circuit design and gives life to the initial idea. Once the product is manufactured, it requires maintenance from time to time to stay in touch with technological developments. Hence, the engineers keep upgrading it to newer versions from time to time.

1.5 IoT enabling Technologies and Applications

IoT(internet of things) enabling technologies are

1. Wireless Sensor Network
2. Cloud Computing

3. Big Data Analytics
4. Communications Protocols
5. Embedded System

1. Wireless Sensor Network(WSN) :

A **WSN** comprises distributed devices with sensors which are used to monitor the environmental and physical conditions. A **wireless sensor network** consists of end nodes, routers and coordinators. End nodes have several sensors attached to them where the data is passed to a coordinator with the help of routers. The coordinator also acts as the gateway that connects WSN to the internet.

Example –

- Weather monitoring system
- Indoor air quality monitoring system
- Soil moisture monitoring system
- Surveillance system
- Health monitoring system

2. Cloud Computing :

It provides us the means by which we can access applications as utilities over the internet. Cloud means something which is present in remote locations.

With Cloud computing, users can access any resources from anywhere like databases, webservers, storage, any device, and any software over the internet.

Characteristics –

1. Broad network access
2. On demand self-services
3. Rapid scalability
4. Measured service
5. Pay-per-use

Provides different services, such as –

- **IaaS (**Infrastructure as a service**)**
  Infrastructure as a service provides online services such as physical machines, virtual machines, servers, networking, storage and data center space on a pay per use basis. Major IaaS providers are Google Compute Engine, Amazon Web Services and Microsoft Azure etc.
  Ex : Web Hosting, Virtual Machine etc.
- **PaaS (**Platform as a service**)**
  Provides a cloud-based environment with a very thing required to support the complete life cycle of building and delivering West web based (cloud) applications – without the cost and complexity of buying and managing underlying hardware, software provisioning and hosting. Computing platforms such as hardware, operating systems and libraries etc. Basically, it provides a platform to develop applications.
  Ex : App Cloud, Google app engine
- **SaaS (**Software as a service**)**
  It is a way of delivering applications over the internet as a service. Instead of installing and maintaining software, you simply access it via the internet, freeing yourself from

complex software and hardware management.

SaaS Applications are sometimes called web-based software on demand software or hosted software.

SaaS applications run on a SaaS provider's service and they manage security availability and performance.

Ex : Google Docs, Gmail, office etc.

3. Big Data Analytics :

It refers to the method of studying massive volumes of data or big data. Collection of data whose volume, velocity or variety is simply too massive and tough to store, control, process and examine the data using traditional databases.

Big data is gathered from a variety of sources including social network videos, digital images, sensors and sales transaction records.

Several steps involved in analyzing big data –

1. Data cleaning
2. Munging
3. Processing
4. Visualization

Examples –

- Bank transactions
- Data generated by IoT systems for location and tracking of vehicles
- E-commerce and in Big-Basket
- Health and fitness data generated by IoT system such as a fitness bands

4. Communications Protocols :

They are the backbone of IoT systems and enable network connectivity and linking to applications. Communication protocols allow devices to exchange data over the network. Multiple protocols often describe different aspects of a single communication. A group of protocols designed to work together is known as a protocol suite; when implemented in software they are a protocol stack.

They are used in

1. Data encoding
2. Addressing schemes

5. Embedded Systems :

It is a combination of hardware and software used to perform special tasks.

It includes microcontroller and microprocessor memory, networking units (Ethernet Wi-Fi adapters), input output units (display keyword etc.) and storage devices (flash memory).

It collects the data and sends it to the internet.

Embedded systems used in

Examples –

1. Digital camera
2. DVD player, music player
3. Industrial robots
4. Wireless Routers etc.

Module:2 Architecture Reference Model

IOT Reference Model

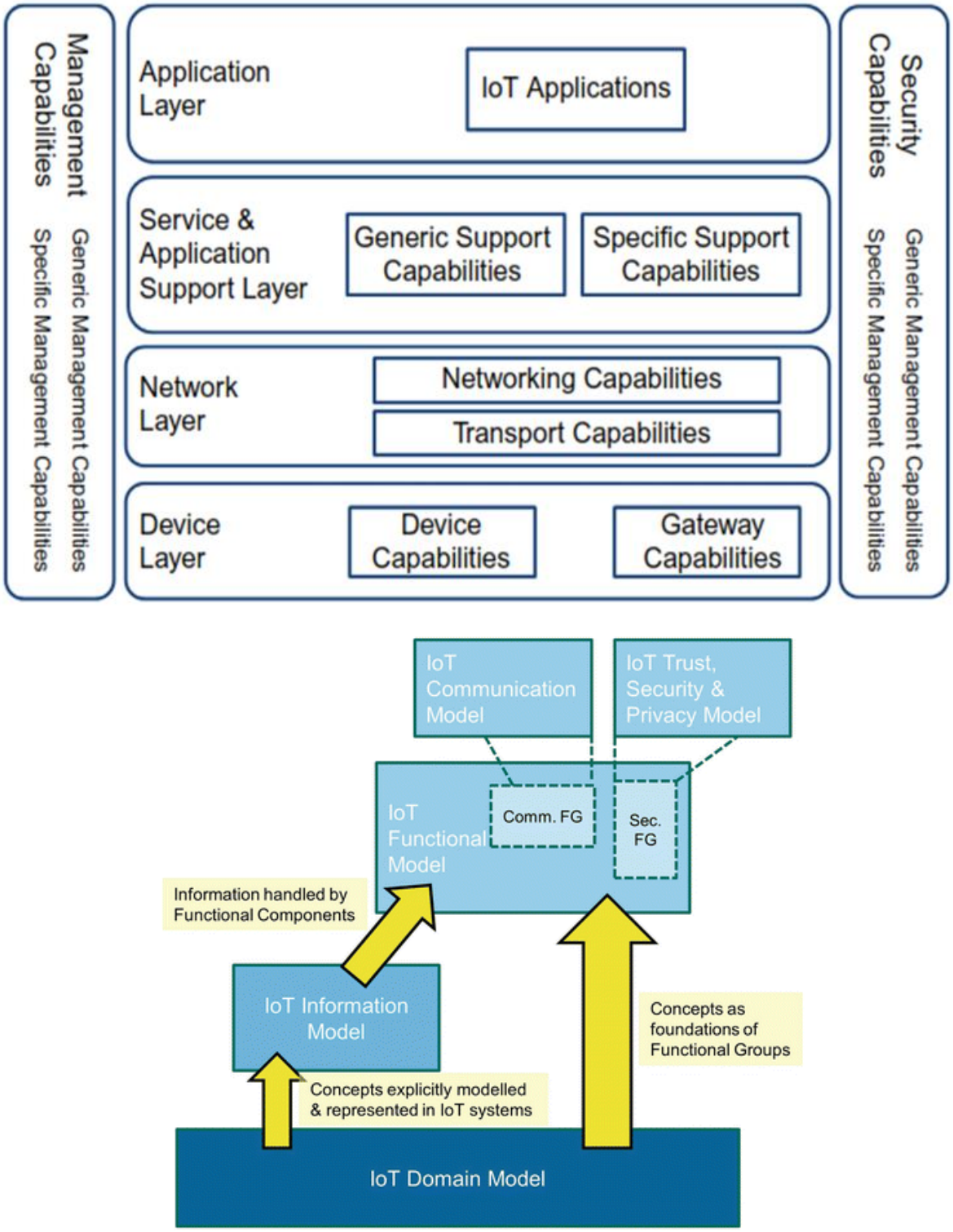


Fig.2.1 IoT Reference Model

An ARM consists of two main parts: a Reference model and a Reference Architecture. A reference model describes the domain using a number of sub-model.

2.1 Domain Model

A model that is a base of any reference model, that creates a model for any specific domain and in this case, it is an IoT domain model. The domain model captures the basic attributes of the main concepts and the relationship between these concepts. A domain model also serves as a tool for human communication between people working in the domain in question and between people who work across different domains.

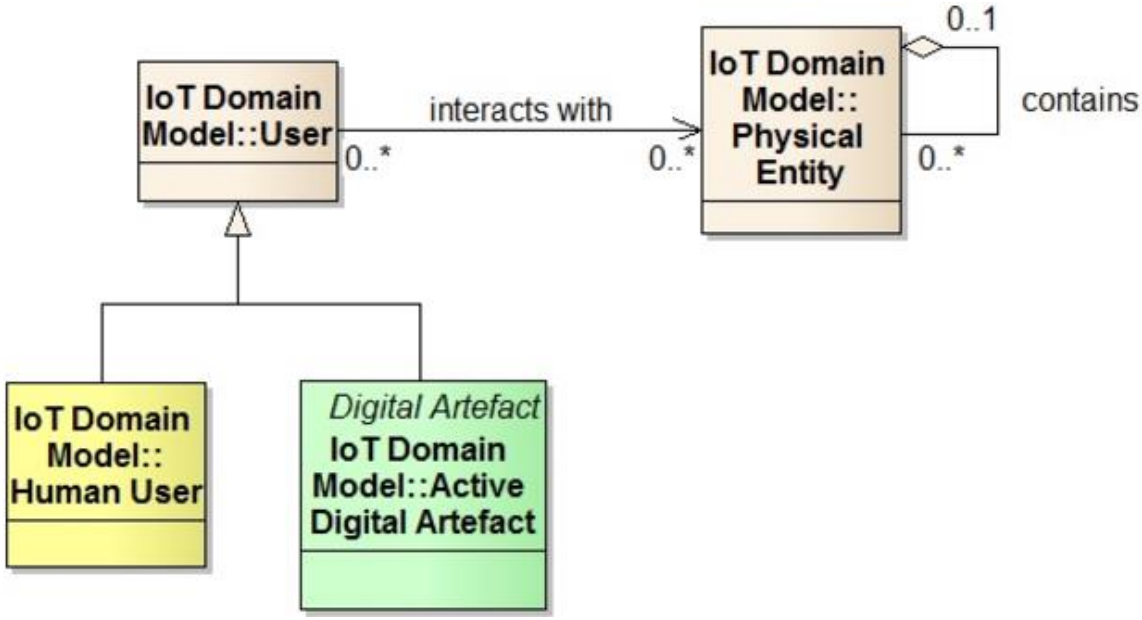


Fig.2.2 IoT Domain Model

In the IOT Domain model, devices are technical artifacts that behave as interfaces between the digital (Virtual Entity) and physical (Physical Entity) world. Therefore, devices must have capabilities (like storage, computation & communication) to operate in the digital as well as physical world. Also resources available in devices also play a very critical role in overall operation.

I. Device Capabilities:

1. Communication related capabilities are covered under **Communication Model** that is a sub-model of the Domain model.

- Type of data exchange (like identifier, identifier + data, sensor data or commands) is supported by device.
- Support communication topology (like network, peer to peer etc.). It affects energy consumption, data collection frequency, and the amount of data transmitted. Location of Resources (on-device or on network) are also impacted based on communication capabilities.
- It also affects the Security features.

2. Computation Capabilities of devices have huge impacts on security features, and power resources.

3. Storage capabilities of devices are also impacted as it determines firmware or software running on devices.

II. Resource

Resources are software components that provide special functionality. Please note that hardware are not considered as resources.

Example: Actuation, Storage Resources, processing information or services on cloud/network etc.

Here, Actuation allows to get information and also changes in digital or physical entities.

- Resources are of two types:

1. On-Device resources like sensor data retrieval or actuator to control the digital or physical world, storage with limited capacity.
2. Network Resources like services on cloud to perform large data processing like data aggregation, computation or storage in cloud.

III. Services

IoT Services are technical services that define standard interfaces and hide complexity of accessing a variety of heterogeneous Resources.

Following services are classified based on their level of abstractions:

1. **Resource level Services**:

These services are for on device resources or network resources to provide functionality and also handles following quality aspects:

- Dependability
- Security
- Resilience(availability)
- Scalability & Timeliness

2. **Virtual Entity level Services**:

These services can be associated with a single or multiple virtual entity that gives access to attributes to read and update the values.

3. **Integrated Services**:

These services are composition of Resource level and Virtual entity level services.

IV. Physical Entities

Physical entity can be a living being or any other item like car, store, logistic items, electronics appliances etc.

In IoT domain model, physical entities are identified by two ways:

1. **Primary Identification:** Based on natural features of entity like camera having sensor
2. **Secondary Identification:** Tags or labels based identification, like RFID tags or barcodes

USE CASE:

The IoT is a support infrastructure for enabling objects and places in the physical world to have a corresponding representation in the digital world. The Devices are physical artefacts with which the physical and virtual worlds interact. Devices as mentioned before can also be Physical Entities for certain types of applications, such as management applications when the interesting entities of a system are the Devices themselves and not the surrounding environment.

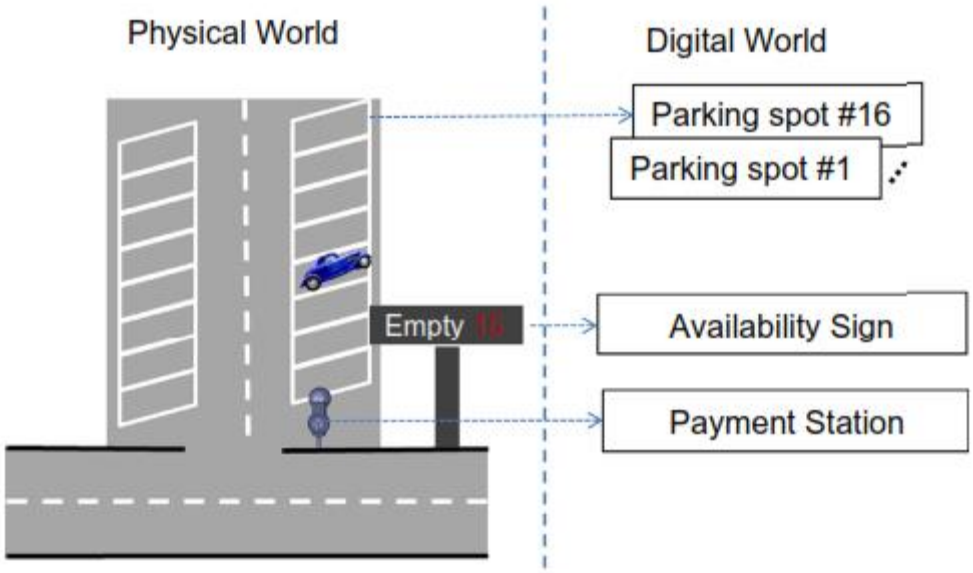


Fig.2.3 IoT Domain Model-Use Case

For the IoT Domain Model, three kinds of Device types are the most important:

1. Sensors:

- These are simple or complex Devices that typically involve a transducer that converts physical properties such as temperature into electrical signals.
- These Devices include the necessary conversion of analog electrical signals into digital signals, e.g. a voltage level to a 16-bit number, processing for simple calculations, potential storage for intermediate results, and potentially communication capabilities to transmit the digital representation of the physical property as well receive commands.
- A video camera can be another example of a complex sensor that could detect and recognise people.

2. Actuators:

- These are also simple or complex Devices that involve a transducer that converts electrical signals to a change in a physical property (e.g. turn on a switch or move a motor).
- These Devices also include potential communication capabilities, storage of intermediate commands, processing, and conversion of digital signals to analog electrical signals.

3. Tags:

- Tags in general identify the Physical Entity that they are attached to. In reality, tags can be Devices or Physical Entities but not both, as the domain model shows.
- An example of a Tag as a Device is a Radio Frequency Identification (RFID) tag, while a tag as a Physical Entity is a paper-printed immutable barcode or Quick Response (QR) code.
- Either electronic Devices or a paper-printed entity tag contains a unique identification that can be read by optical means (bar codes or QR codes) or radio signals (RFID tags).
- The reader Device operating on a tag is typically a sensor, and sometimes a sensor and an actuator combined in the case of writable RFID tags.

2.2 Information Model

Virtual Entity in the IoT Domain Model is the "Thing" in the Internet of Things, the IoT information model captures the details of a Virtual Entity- centric model. Similar to the IoT Domain Model, the IoT Information Model is presented using Unified Modelling Language (UML) diagrams.

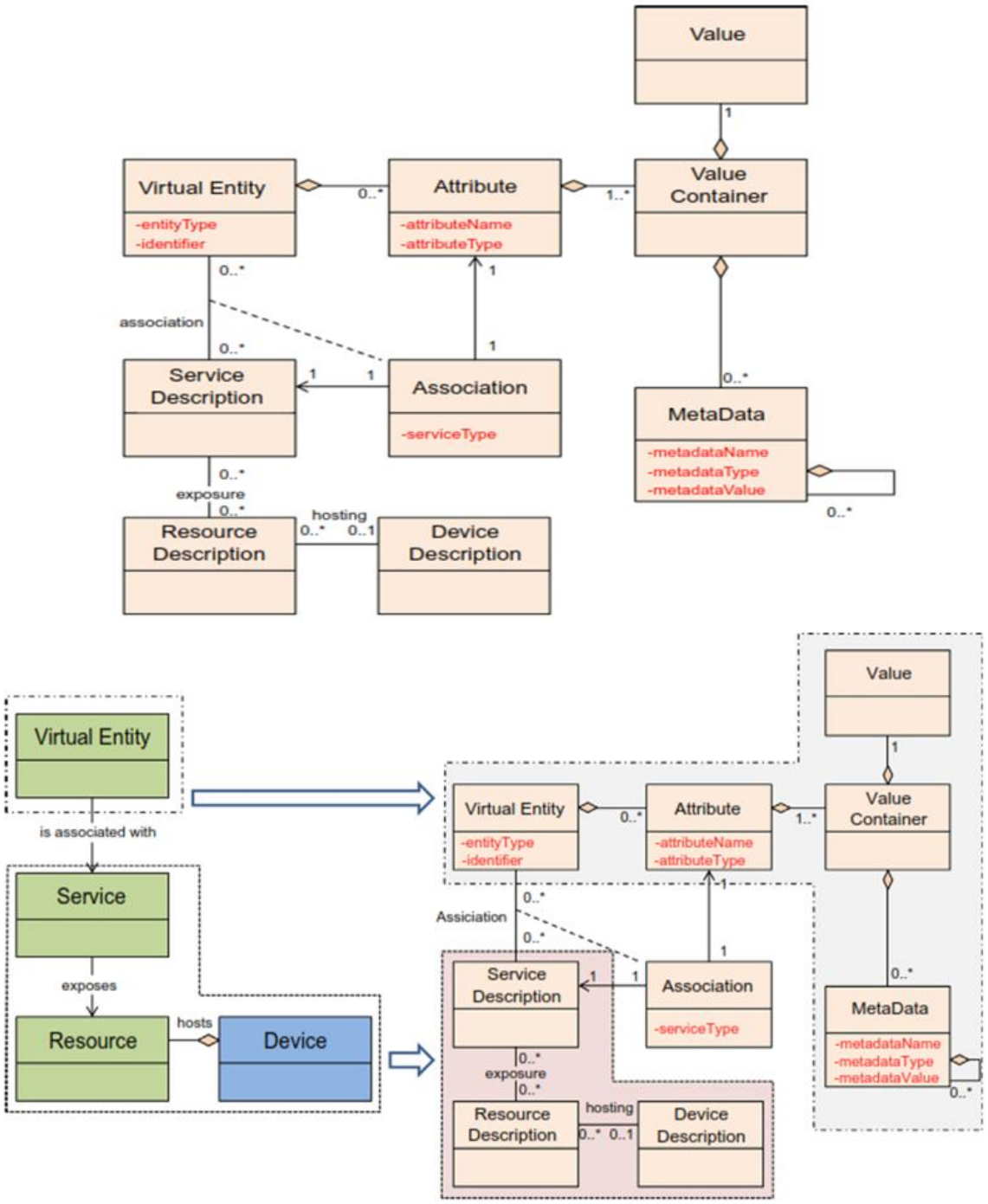


Fig.2.4 IoT Information Model

An IoT information model is an abstract, formal representation of IoT thing types that often include their properties, relationships, and the operations that can be performed on them. A thing information model needs to specify what the thing is, what it can do, thing properties and

their values, and ways to interact with it. Some specifications provide only definition of thing types and their properties and they tend to be referred to as data models.

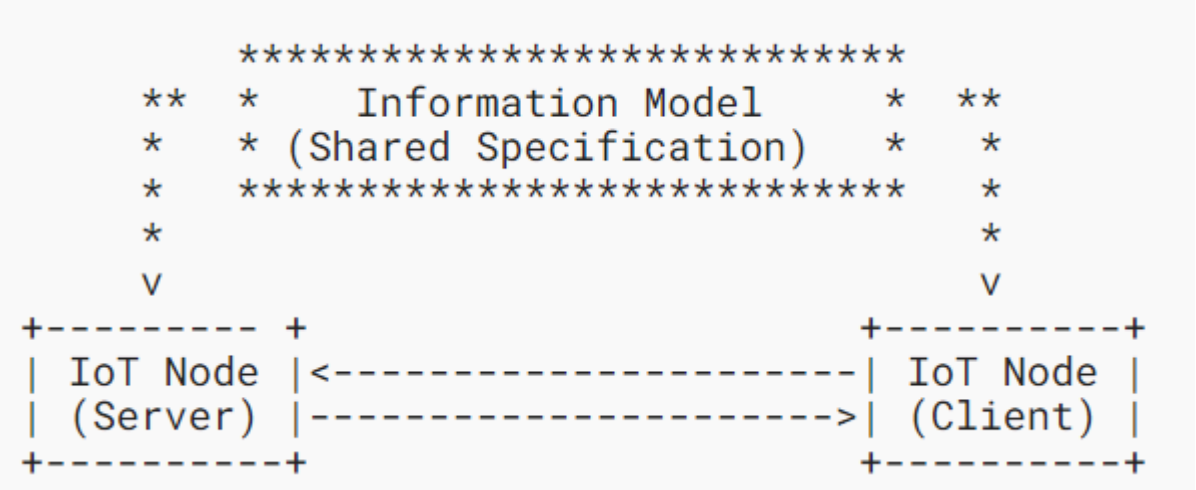


Fig.2.5 Shared Information Model

- All communicating parties need to use the same conceptual model of things that exist in their domain. A common way to accomplish this is by having all parties use the same specification of an IoT information model.
- One of the primary tasks of an IoT information model is to facilitate semantic interoperability, i.e. a shared understanding of what the data means. It is used by the IoT server to provide the compliant software abstraction of the thing, and by IoT clients to properly interpret it and be able to process the data accordingly.
- Following the Internet principles and practice, clients and servers exchange payloads and rely on an information-model specification for proper encoding and interpretation of the exchanged messages. The two endpoints are otherwise decoupled, i.e. they may be developed independently of each other and operate on different platforms and runtime environments. In this context, endpoint refers to software agents that are acquiring (producing) or processing (consuming) data.
- The IoT server internally implements the thing description as a cyber-world touch point for each particular instance using the format of the information model defined by the shared specification. It also needs to implement the actions associated with the device when requested by the authorized parties. Depending on the system configuration and hardware capability, an IoT server may physically reside on the thing or on a proxy device, such as a directly attached gateway.
- IoT clients are consumers of thing/server data and generators of actuation commands. IoT clients are commonly software agents associated with or acting on behalf of services and applications. A client can access server thing descriptions that it is authorized to and use them to issue queries and optionally command messages. The client uses the shared information-model definition to properly format requests and interpret responses that it receives. IoT clients may reside at in at the edge - such as a peer thing node or edge gateway, at some intermediate point in the system hierarchy such as a fog node, or in the cloud.
- Since objects are modeling real-world things, there are some implied semantics defined by the intrinsic nature of the thing that is being modeled. For example, a temperature

sensor is commonly understood to measure temperature of something based on its characteristics and placement, such as the ambient air temperature or water temperature in a heating/cooling pipe.

Element	Description
Object Type	Physical thing being modeled (device)
Properties, Attributes	Object attributes, data, metadata
Interactions	Ways to interact with object, actions, events
Links	To other objects, compositions and collections

Fig.2.6 Structure of IoT Information Model

Software objects that represent and model IoT things are often structured in ways that follow the common practices and terminology in object-oriented (OO) programming. They are also referred to by other names, such as thing descriptions and smart objects. Fig.2.6 illustrates general the form of object-oriented IoT information model representations.
In addition to using the same information model, in order to interoperate IoT endpoints also need to use the common set of protocols and serialization mechanisms, compatible naming and addressing conventions and operate in the common security perimeter.

A working IoT system needs to provide a common operational environment for nodes to communicate. Some of its components may be defined by standards and other specified by convention or a particular system implementation. Environmental components needed for functional interoperability generally include:

- Data (information) model
- Payload serialization
- Protocol bindings
- Naming and addressing, discovery
- Security
- Device management and provisioning

Payload serialization refers to the common agreement on the format in which messages appear "on the wire", i.e. are formatted by the sending node and parsed by the receiving one. Variants of JSON are commonly used for this purpose, although more compact forms of serializations, such as the binary CBOR may be favoured for constrained nodes and networks.

- Protocol bindings specify which protocols are supported and perhaps the port numbers on which they operate. In addition to the common Internet protocols, more compact versions, such as CoAP may be favored for constrained nodes or segments of the network.

- Naming and addressing refer to the mechanisms used to access nodes in the system. They usually resolve to network IP addresses, but may also include higher-level mapping constructs such as URLs and URIs.

- Discovery provides means to identify other nodes in the system to communicate with or to form groupings of interest. They can include discovery through network scanning to identify nodes. This is usually combined with mechanisms and conventions to access machine-readable descriptions of the nature and capabilities of nodes at actively used addresses. Another approach is to create directories where nodes get registered during the activation process and are discovered by queries.

- A working IoT system typically implements security requirements, conventions, and protocols for node authentication, access authorization, and secure communication. In order to access each other and to communicate, all nodes must adhere to the rules and use the commonly prescribed system procedures. Security system usually maintains and updates security postures and policies of IoT nodes and monitors their activity to detect and deal with anomalous behaviors such as intrusions and probing.

- Device management, like security, is part of the control plane of an IoT system. It is often implemented to keep the system operational, track states of its components - such as active or inactive - and often manages distribution and application of software and firmware updates.

2.3 Functional Model

The IoT Functional Model aims at describing mainly the Functional Groups (FG) and their interaction with the ARM, while the Functional View of a Reference Architecture describes the functional components of an FG, interfaces, and interactions between the components. The Functional View is typically derived from the Functional Model in conjunction with high-level requirements.

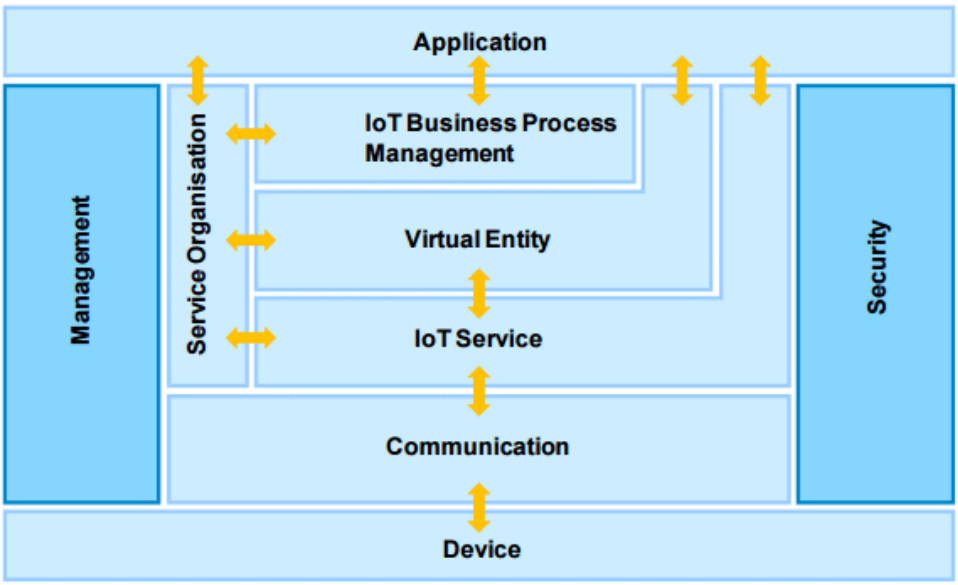


Fig.2.7 IoT Functional Model

- **Device functional group:** The Device FG contains all the possible functionality hosted by the physical Devices that are used for increment the Physical Entities. This Device functionality includes sensing, actuation, processing, storage, and identification components, the sophistication of which depends on the Device capabilities.
- **Communication functional group:** The Communication FG abstracts all the possible communication mechanisms used by the relevant Devices in an actual system in order to transfer information to the digital world components or other Devices.
- ***IoT Service functional group:*** The IoT Service FG corresponds mainly t the Service class from the IoT Domain Model, and contains single IoT Services exposed by Resources hosted on Devices or in the Network (e.g. processing or storage Resources).
- **Virtual Entity functional group:** The Virtual Entity FG corresponds to the Virtual Entity class in the IoT Domain Model, and contains the necessary functionality to manage associations between Virtual Entities with themselves as well as associations between Virtual Entities and related IoT Services, i.e. the Association objects for the IoT Information Model. Associations between Virtual Entities can be static or dynamic depending on the mobility of the Physical Entities related to the corresponding Virtual Entities.
- I**oT Service Organization functional group:** The purpose of the IoT Service Organisation FG is to host all functional components that support the composition and orchestration of IoT and Virtual Entity services. Moreover, this FG acts as a service hub between several other functional groups such as the IoT Process Management FG when, for example, service requests from Applications or the IoT Process Management are directed to the Resources implementing the necessary Services.
- **IoT Process Management functional group:** The IoT Process Management FG is a collection of functionalities that allows smooth integration of IoT-related services (IoT Services, Virtual Entity Services, Composed Services) with the Enterprise (Business) Processes.
- **Management functional group:** The Management **functional group** (FG) includes the necessary functions for enabling fault and performance monitoring of the system, configuration for enabling the system to be flexible to changing User demands, and accounting for enabling subsequent billing for the usage of the system. Support functions such as management of ownership, administrative domain, rules and rights of functional components, and information stores are also included in the Management FG.
- **Security functional group:** The Security FG contains the functional components that ensure the secure operation of the system as well as the management of privacy. The Security FG contains components for Authentication of Users (Applications, Humans), Authorisation of access to Services by Users, secure communication (ensuring integrity and confidentiality of messages) between entities of the system such as Devices, Services, Applications, and last but not least, assurance of privacy of sensitive information relating to Human Users.

- **Application functional group:** The Application FG is just a placeholder that represents all the needed logic for creating an IoT application. The applications typically contain custom logic tailored to a specific domain such as a Smart Grid

2.4 Communication Model

IoT devices are found everywhere and will enable circulatory intelligence in the future. For operational perception, it is important and useful to understand how various IoT devices communicate with each other. Communication models used in IoT have great value. The IoTs allow people and things to be connected any time, any space, with anything and anyone, using any network and any service.

Types of Communication Model :

1. **Request & Response Model –** This model follows a client-server architecture.

- The **client**, when required, requests the information from the server. This request is usually in the encoded format.
- This model is stateless since the data between the requests is not retained and each request is independently handled.
- The server Categories the request, and fetches the data from the database and its resource representation. This data is converted to response and is transferred in an encoded format to the client. The client, in turn, receives the response.
- On the other hand — In **Request-Response** communication model client sends a request to the server and the server responds to the request. When the server receives the request it decides how to respond, fetches the data retrieves resources, and prepares the response, and sends it to the client.

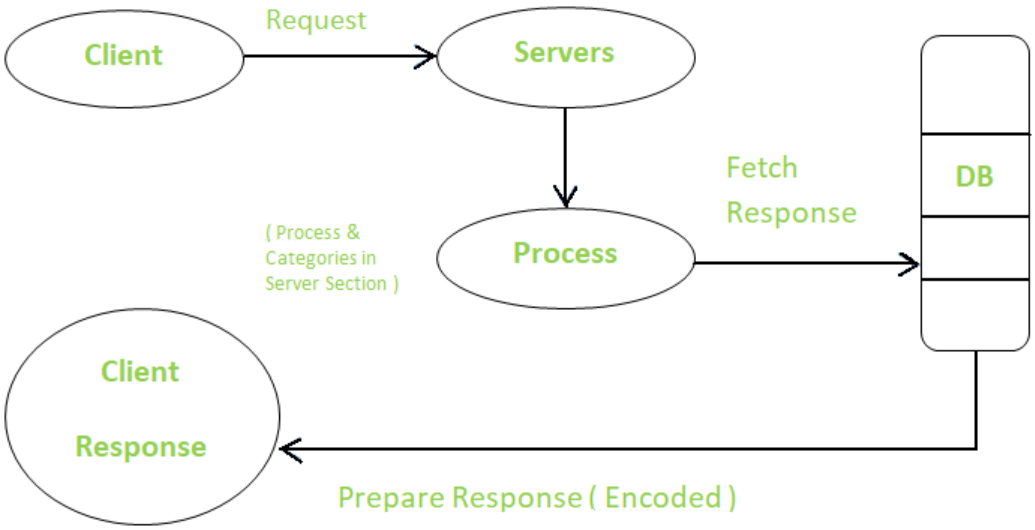


Fig.2.8 Request & Response Model

2. Publisher-Subscriber Model –

This model comprises three entities: Publishers, Brokers, and Consumers.

- **Publishers** are the source of data. It sends the data to the topic which are managed by the broker. They are not aware of consumers.
- **Consumers** subscribe to the topics which are managed by the broker.
- Hence, **Brokers** responsibility is to accept data from publishers and send it to the appropriate consumers. The broker only has the information regarding the consumer to which a particular topic belongs to which the publisher is unaware of.

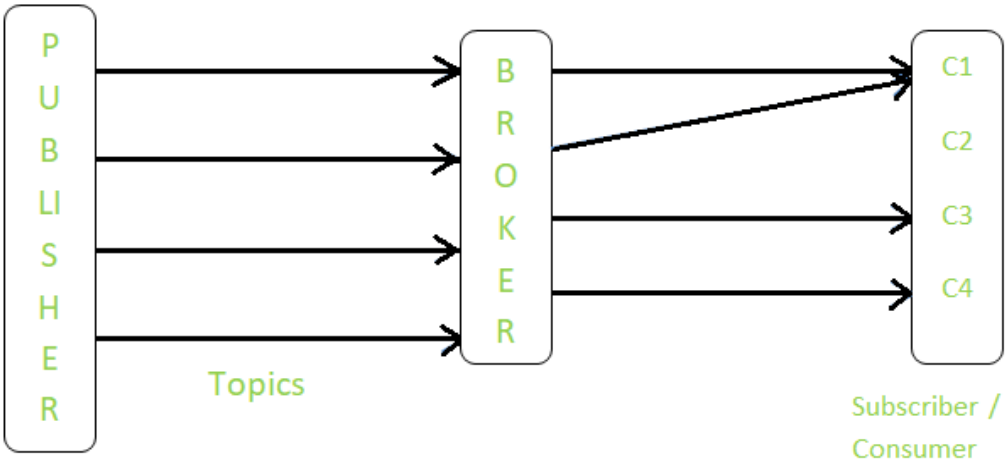


Fig.2.9 Publisher-Subscriber Model

3. Push-Pull Model –

The push-pull model constitutes data publishers, data consumers, and data queues.

- **Publishers** and **Consumers** are not aware of each other.
- Publishers publish the message/data and push it into the queue. The consumers, present on the other side, pull the data out of the queue. Thus, the queue acts as the buffer for the message when the difference occurs in the rate of push or pull of data on the side of a publisher and consumer.
- **Queues** help in decoupling the messaging between the producer and consumer. Queues also act as a buffer which helps in situations where there is a mismatch between the rate at which the producers push the data and consumers pull the data.

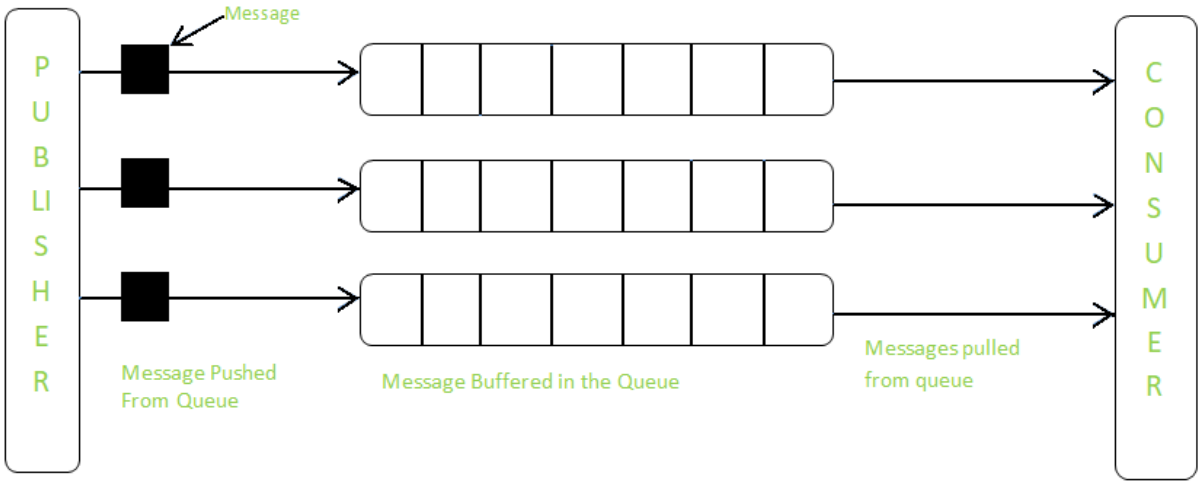


Fig.2.10 Push-Pull Model

4. Exclusive Pair –

- **Exclusive Pair** is the bi-directional model, including full-duplex communication among client and server. The connection is constant and remains open till the client sends a request to close the connection.

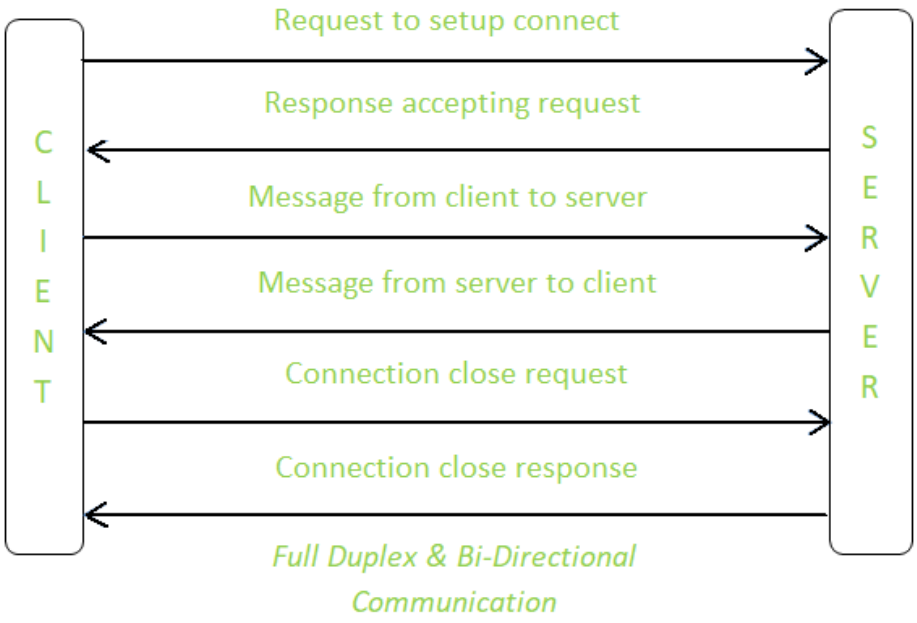


Fig.2.11 Exclusive Pair Model

- The **Server** has the record of all the connections which has been opened.
- This is a state-full connection model and the server is aware of all open connections.
- WebSocket based communication API is fully based on this model.

2.5 Security model

- The Internet of Things (IoT) refers to the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, and connectivity, allowing these objects to connect, interact, and exchange data.
- Security has become a major concern with the increasing number of devices in our daily lives. As these devices collect and store sensitive information, they are vulnerable to cyberattacks, data breaches, and hacking.
- An IoT security model refers to the set of security measures and protocols that protect devices, networks and systems from cyber-attacks and data breaches. The purpose of an IoT security model is to ensure the confidentiality, integrity, and availability of data transmitted between devices, as well as to ensure the privacy and security of end-users.

Fig.2.12 IOT Security Model components

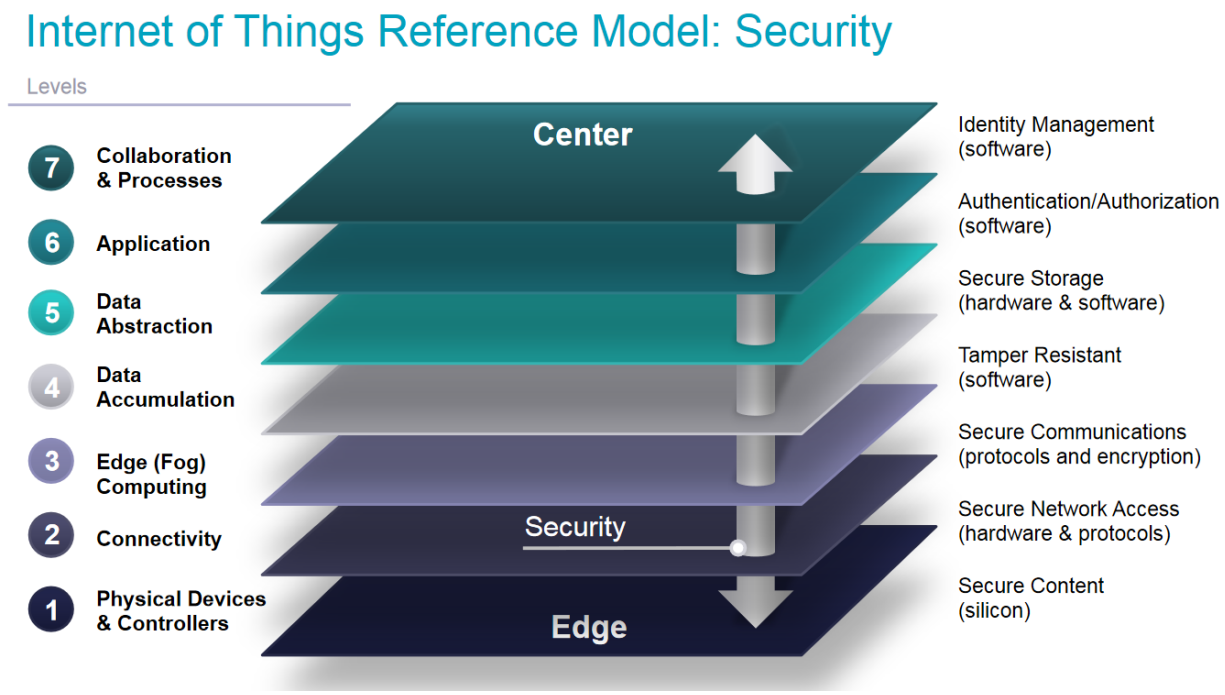


Fig.2.13 IOT Security Model

There are several components to an IoT security model, including device security, network security, cloud security, application security and end-to-end security. Each of these components must be secured to provide a comprehensive and effective security solution.

- **Device Security**
  Device security refers to the security measures implemented at the device level to protect devices from cyber-attacks and data breaches. This includes using secure boot processes, secure firmware updates, strong passwords and encryption. Additionally, device security also involves ensuring that devices are not susceptible to denial of service (DoS) attacks and can detect and respond to potential security threats.
- **Network Security**
  Network security is the second component of an IoT security model and refers to the security measures implemented to protect the communication between IoT devices and systems. This includes the use of secure protocols such as HTTPS, SSL and TLS, and the implementation of firewalls, intrusion detection systems, and access control systems. Additionally, network security also involves ensuring that network segmentation is implemented to prevent unauthorised access to sensitive data, and that network traffic is monitored for suspicious activity.
- **Cloud Security**
  Cloud security is the third component of an IoT security model and refers to the security measures implemented to protect the data stored in the cloud. This includes using encryption, access control systems, and data backup and recovery systems. Additionally, cloud security also involves ensuring that cloud providers implement best practices for data protection, such as regularly performing security audits and implementing strict access control policies.

- **Application Security**
  Application security is the fourth component of an IoT security model and involves securing the applications that run on the devices. This includes ensuring that the applications are protected from malware and other security threats and that they are updated regularly to address any security vulnerabilities. This can be achieved by using secure coding practices, implementing access controls, and using anti-malware solutions.
- **End-to-End Security**
  End-to-end security is the fifth component of an IoT security model and involves securing the entire IoT system, from the devices to the cloud and back. This includes ensuring that all system components are secure and that the transmitted data is protected. This can be achieved by implementing security policies and procedures, conducting regular security audits, and using security solutions such as VPNs.
  In addition to these security layers, it is crucial to consider privacy protection in security. This involves protecting the sensitive information that devices collect as personal and financial data. This can be achieved through secure data storage and privacy-enhancing technologies such as anonymisation and differential privacy.
  Finally, organisations must have a comprehensive security policy in place for devices. This policy should address the various security layers outlined above, including device security, network security, cloud security, application security, end-to-end security, and privacy protection. The policy should also outline the roles and responsibilities of all stakeholders involved in the security process, including device manufacturers, service providers, and end-users.

Challenges in IoT security Model:

The IoT security model faces several challenges that can impact its effectiveness in securing IoT devices and the data they collect and transmit. Some of these challenges include:

- **Complexity**: IoT devices are becoming increasingly complex, with a growing number of components and systems that must be secured. This complexity makes it difficult to implement a comprehensive security solution that can effectively protect all aspects of the IoT system
- **Inadequate security measures**: Many IoT devices need more adequate security measures, such as encryption, firewalls, and intrusion detection systems. This leaves these devices vulnerable to attack and exploitation by malicious actors.
- **Outdated software**: Many IoT devices run on outdated software that the manufacturer no longer supports. This makes it difficult to apply security updates or patches to these devices, leaving them vulnerable to attack.
- **Limited processing power**: Many IoT devices have limited processing power, memory, and storage capacity, making it difficult to run traditional security software on them. This leaves these devices vulnerable to attack, as attackers can exploit known vulnerabilities in these devices to gain access to sensitive data or control over the device.
- **Poorly designed protocols**: The protocols used to communicate between IoT devices and the internet may be poorly designed, making them vulnerable to exploitation. For

example, some protocols may use unencrypted communications, making it easy for attackers to intercept and manipulate the data being transmitted.

- **Lack of visibility**: It can be challenging to monitor and manage the security of IoT devices, as they are often deployed in remote or inaccessible locations. This makes it challenging to detect and respond to security threats promptly.
- **User behaviour**: The security of IoT devices is also dependent on the behaviour of the users. For example, users may use weak passwords, neglect to apply software updates or be careless with the data they share online, putting their devices and data at risk.
- **Interoperability**: As the number of IoT devices continues to grow, there is a need for interoperability between these devices, which can make it difficult to implement a consistent security approach across the entire IoT ecosystem.
- **Lack of security standards**: The IoT ecosystem consists of a vast number of devices from different manufacturers, each with its security standards. This makes it difficult to have a unified security approach for the entire system.
- **Poor testing:** Most IoT developers do not prioritise security, and perform effective vulnerability testing to identify weaknesses in IoT systems.

How to protect IoT systems and devices

1. Introduce IoT security during the design phase

Enabling security by default is critical, as well as providing the most recent operating systems and using secure hardware. IoT developers should, however, be mindful of cybersecurity vulnerabilities throughout each stage of development -- not just the design phase. The car key hack, for instance, can be mitigated by placing the FOB in a metal box, or away from one's windows and hallways.

2. Public Key Infrastructure (PKI) and digital certificates

Public Key Infrastructure (PKI) is an excellent way to secure the client-server connections between multiple networked devices. Using a two-key asymmetric cryptosystem, PKI is able to facilitate the encryption and decryption of private messages and interactions using digital certificates. These systems help to protect the clear text information input by users into websites to complete private transactions. E-commerce wouldn't be able to operate without the security of PKI.

3. Network security

Networks provide a huge opportunity for threat actors to remotely control others' IoT devices. Because networks involve both digital and physical components, on-premises IoT security should address both types of access points. Protecting an IoT network includes ensuring port security, disabling port forwarding and never opening ports when not needed; using antimalware, firewalls and intrusion detection systems/intrusion prevention systems; blocking unauthorized IP (Internet Protocol) addresses; and ensuring systems are patched and up to date.

4. API security

APIs are the backbone of most sophisticated websites. They allow travel agencies, for example, to aggregate flight information from multiple airlines into one location. Unfortunately, hackers can compromise these channels of communication, making API security necessary for

protecting the integrity of data being sent from IoT devices to back-end systems and ensuring only authorized devices, developers and apps communicate with APIs.

- **Network access control.** NAC can help identify and inventory IoT devices connecting to a network. This will provide a baseline for tracking and monitoring devices.
- **Segmentation.** IoT devices that need to connect directly to the internet should be segmented into their own networks and have restricted access to the enterprise network. Network segments should be monitoring for anomalous activity, where action can be taken, should an issue be detected.
- **Security gateways.** Acting as an intermediary between IoT devices and the network, security gateways have more processing power, memory and capabilities than the IoT devices themselves, which provides them the ability to implement features such as firewalls to ensure hackers cannot access the IoT devices they connect.
- **Patch management/continuous software updates.** It is critical to provide the means of updating devices and software either over network connections or through automation. Having a coordinated disclosure of vulnerabilities is also important for updating devices as soon as possible. Consider end-of-life strategies as well.
- **Training.** IoT and operational system security are new to many existing security teams. It is critical for security staff to keep up to date with new or unknown systems, learn new architectures and programming languages and be ready for new security challenges. C-level and cybersecurity teams should receive regular cybersecurity training to keep up with modern threats and security measures.
- **Integrating teams.** Along with training, integrating disparate and regularly siloed teams can be useful. For example, having programing developers work with security specialists can help ensure the proper controls are added to devices during the development phase.
- **Consumer education.** Consumers must be made aware of the dangers of IoT systems and provided steps to stay secure, such as updating default credentials and applying software updates. Consumers can also play a role in requiring device manufacturers to create secure devices and refusing to use those that don't meet high-security standards.

2.6 SOA based architecture

- 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.
- A Service-Oriented Architecture or SOA is a design pattern which is **designed to build distributed systems that deliver services to other applications through the protocol. It is only a concept and not limited to any programming language or platform.**
- 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.

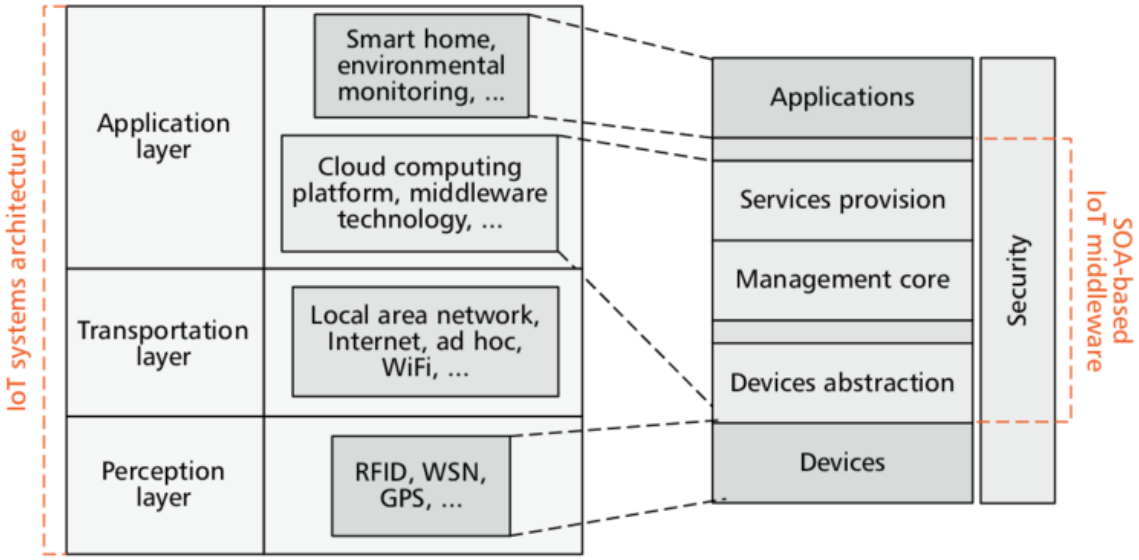


Fig.2.14 IoT systems architecture and SOA-based IoT middleware

The different characteristics of SOA are as follows :

- Provides interoperability between the services.
- Provides methods for service encapsulation, service discovery, service composition, service reusability and service integration.
- Facilitates QoS (Quality of Services) through service contract based on Service Level Agreement (SLA).
- Provides loosely couples services.
- Provides location transparency with better scalability and availability. Ease of maintenance with reduced cost of application development and deployment.

There are two major roles within Service-oriented Architecture:

1. **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.

2. **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.

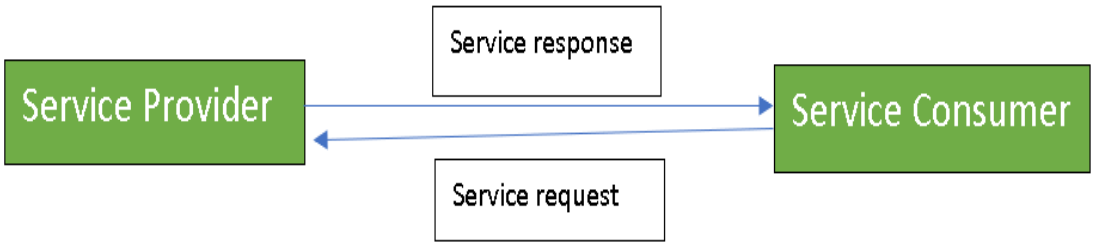


Fig.2.15 Major roles within Service-oriented Architecture

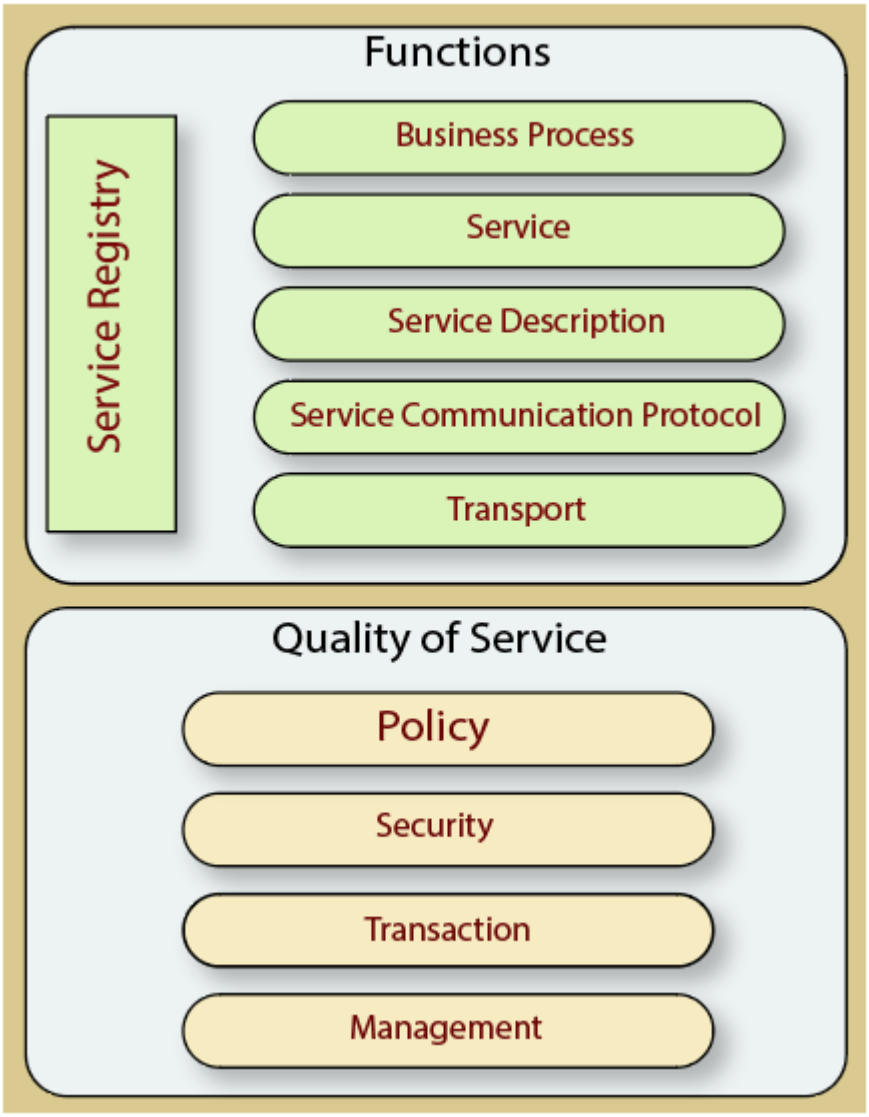


Fig.2.16 Components of SOA

Components of SOA:

1. **Standardized service contract:** Specified through one or more service description documents.
2. **Loose coupling:** Services are designed as self-contained components, maintain relationships that minimize dependencies on other services.
3. **Abstraction:** A service is completely defined by service contracts and description documents. They hide their logic, which is encapsulated within their implementation.
4. **Reusability:** Designed as components, services can be reused more effectively, thus reducing development time and the associated costs.
5. **Autonomy:** Services have control over the logic they encapsulate and, from a service consumer point of view, there is no need to know about their implementation.
6. **Discoverability:** Services are defined by description documents that constitute supplemental metadata through which they can be effectively discovered. Service discovery provides an effective means for utilizing third-party resources.
7. **Composability:** Using services as building blocks, sophisticated and complex operations can be implemented. Service orchestration and choreography provide a solid support for composing services and achieving business goals.

Functional aspects

The functional aspect contains:

- Transport - It transports the service requests from the service consumer to the service provider and service responses from the service provider to the service consumer.
- Service Communication Protocol - It allows the service provider and the service consumer to communicate with each other.
- Service Description - It describes the service and data required to invoke it.
- Service - It is an actual service.
- Business Process - It represents the group of services called in a particular sequence associated with the particular rules to meet the business requirements.
- Service Registry - It contains the description of data which is used by service providers to publish their services.

Quality of Service aspects

The quality of service aspects contains:

- Policy - It represents the set of protocols according to which a service provider make and provide the services to consumers.
- Security - It represents the set of protocols required for identification and authorization.
- Transaction - It provides the surety of consistent result. This means, if we use the group of services to complete a business function, either all must complete or none of the complete.

- Management - It defines the set of attributes used to manage the services.

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

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 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.
4. SOA helps maintain museums a virtualized storage pool for their information and content.

Module:3 Protocol Suite

- The wireless hardwares used in IoT are the **nodes** and the **base stations**. Nodes or **clients** are the devices that connect to the base stations. Base stations are the **routers** or **gateways**.
- The ability to interact with each other and resolve common problems is what separates Iot devices from traditional computers. These interactions are only possible if there is a medium or means of communication in the IoT ecosystem. The IoT protocols are thus a common "language" that allows devices to interact with other Iot devices. The IoT protocols lay down standards that are adopted in every Iot ecosystem for proper functioning and to avoid security threats.
- There is no universal protocol that helps in communicating with all IoT devices yet. The OMA Lightweight protocol manages devices easily and offers adaptive solutions.

TCP/IP Model	IoT Model
Application Layer	HTTPS, XMPP, CoAP, MQTT, AMQP
Transport Layer	TCP, UDP
Internet Layer	IPv6, LoWPAN, RPL
Network Access and Physical Layer	IEEE 802.15.4, WiFi(802.11 a/b/g/n), Ethernet(802.3), CDMA, GSM, LTE

Fig.3.1 TCP/IP model and IOT Model similarity

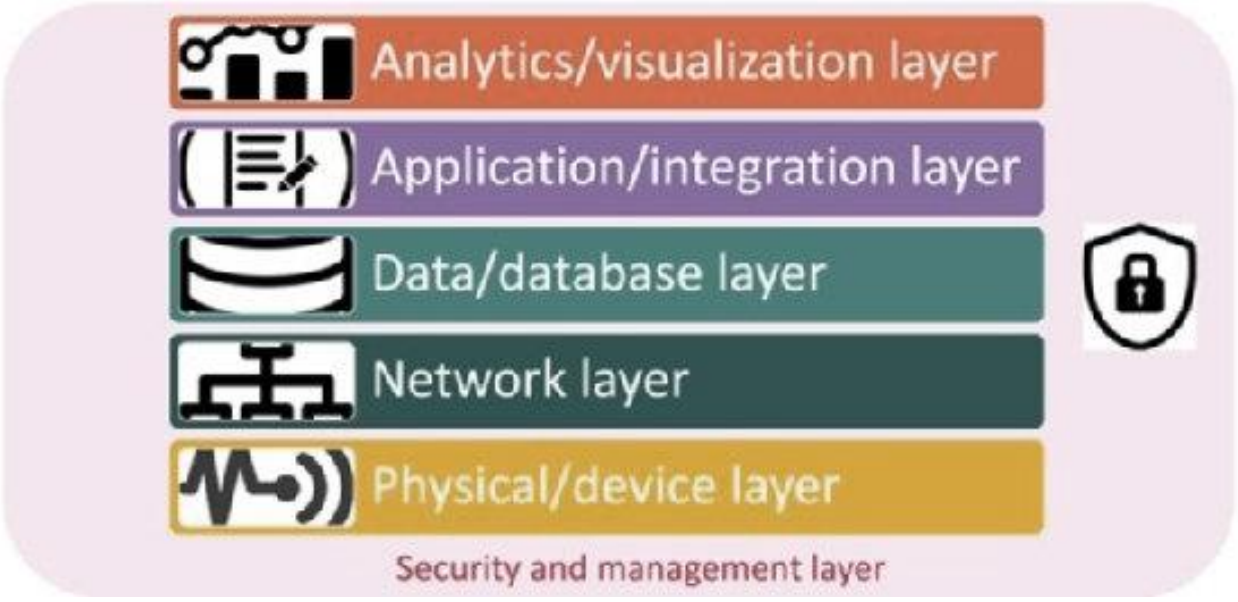


Fig.3.2 IOT Protocol Model

1. **Physical/device layer.** This includes the sensors, actuators, and other smart devices and connected devices that constitute the physical layer and device layer. These smart devices capture data (sensors), take action (actuators) or, sometimes, both.
2. **Network layer.** This comprises the network devices and communications types and protocols, e.g., 5G, Wi-Fi, Bluetooth, etc. Although many IoT architectures rely on general-purpose network layers, there is an increasing trend to move to dedicated IoT-specific networks.
3. **Data/database layer.** This also includes the database platform layer. There are a range of databases used for IoT architectures, and many organizations spend a fair amount of time selecting and architecting the right IoT databases for them.
4. **Analytics/visualization layer.** This layer comprises the analytics layer, visualization layer and perception layer. In essence, this layer's focus is on analyzing the data provided by IoT and providing it to users and applications to make sense of.
5. **Application/integration layer.** This is the layer of applications and platforms that work together to deliver functionality from the IoT infrastructure to the business. In other words, the application layer, platform layer and integration layer are what provides the business value from the IoT infrastructure. The processing layer and business layer are all part of the larger application/integration layer.
6. **Security and management layer.** As the name implies, this layer encompasses both the security layer and the management layer. Strictly speaking, this is not a layer as it connects with all the other layers to provide security and management.

Three Layer Vs. Five Layer protocol suite

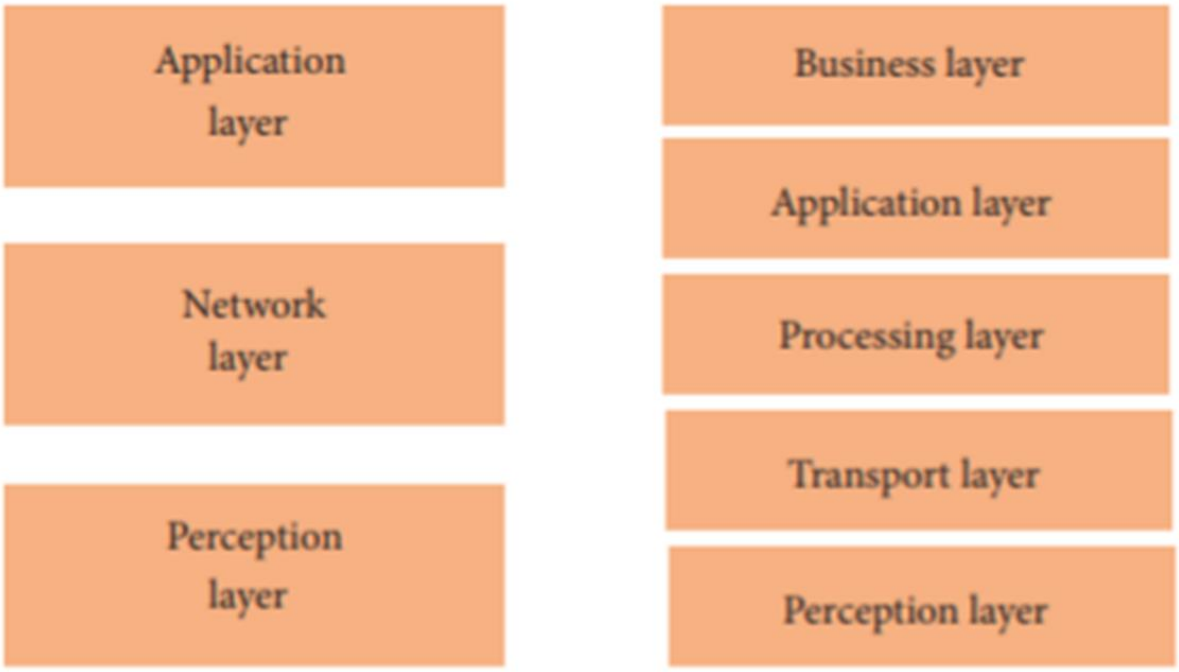


Fig.3.3 IOT Three vs Five layer protocol

Three Layer protocol suite

- The perception layer is the physical layer, which has *sensors for sensing and gathering information about the environment*. It *senses some physical parameters or identifies other smart objects in the environment*.
- The network layer is responsible *for connecting to other smart things, network devices, and servers*. Its features are also *used for transmitting and processing sensor data*. (BLE, Wi-Fi, LoRAWAN)
- The application layer is responsible for *delivering application specific services to the user*. It defines *various applications* in which the Internet of Things can be *deployed, for example, smart homes, smart cities, and smart health*. b

Five Layer protocol suite

- The perception layer is the physical layer, which has *sensors for sensing and gathering information about the environment*. It *senses some physical parameters or identifies other smart objects in the environment*.
- The transport layer transfers the *sensor data from the perception layer to the processing layer and vice versa through networks* such as wireless, 3G, LAN, Bluetooth, RFID, and NFC.
- The processing layer is also known as *the middleware layer*. It *stores, analyzes, and processes huge amounts of data that comes from the transport layer. It can manage and provide a diverse set of services to the lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules*
- The application layer is responsible for *delivering application specific services to the user*. It defines *various applications* in which the Internet of Things can be *deployed, for example, smart homes, smart cities, and smart health*.
- The business layer manages the *whole IoT system, including applications, business and profit models, and users' privacy*.
    - This *Layer also has the responsibility to design, analyze, implement, evaluate, and monitor* the requirements of the IoT system.
    - This layer can use *big data analysis to support decision-making activities*.
    - Performs a *comparison of obtained versus expected outputs to enhance the quality of service*s.

3.1 Physical layer

It is the base of IoT and consists of physical components such as sensors, power networks, and embedded systems. The physical layer is responsible for capturing data and physical events from the physical environment. This layer handles the physical connections between objects, sensors, and other physical device.

Functions of Physical Layer:

1. Representation of Bits: The physical layer takes the responsibility of transmitting individual bits from one node to another via a physical medium. It specifies the procedure for

encoding bits, such as how many volts should represent a 0 bit and a 1 bit in the case of electrical signals.

2. Data Rate: The data rate is maintained by the function of Physical Layer. The number of bits sent per second is referred to as the data rate. It is determined by a variety of factors, including:

- Bandwidth: The physical constraint of the underlying media.
- Encoding: The number of levels used for signaling.
- Error rate: Incorrect information reception due to noise.

3. Synchronization: The function of physical layer includes bit synchronization. The sender and receiver are bit-synchronized. This is accomplished by including a clock. This clock is in charge of both the sender and the receiver. Synchronization is achieved at the bit level in this manner.

4. Interface: The transmission interface between devices and the transmission medium is defined by the physical layer. When considering the standards, it is common, but not required, that the physical layer be divided into two:

- Physical Medium (PM) layer: The physical layer's lowest sublayer.
- Transmission Convergence (TC) layer: The high sublayer of the physical layer.

5. Line Configuration: The function of physical layer includes connecting devices to the medium or line configuration. Line configuration, also known as a connection, is the method by which two or more devices are connected to a link. A dedicated link connects two devices in a point-to-point configuration. A device can be a computer, a printer, or any device capable to send and receive data.

6. Topologies: The physical layer specifies how different computing devices in a network should be connected to one another. A network topology is a configuration by which computer systems or network devices are linked to one another. Topologies can define both the physical and logical aspects of a network. Mesh, Star, Ring, and Bus topologies are required for device connectivity.

7. Transmission Modes: The physical layer specifies the transmission direction between two devices. Transmission mode refers to the method that is used to transfer data from a device to another device. The physical layer in the OSI model primarily determines the direction of data travel required to reach the receiver system or node. Transmission modes are classified into three types: Simplex mode, Half-duplex mode and Full-duplex mode.

3.2 Link layer –BLE

- The link layer includes the protocols that define communication between local (on-link) network nodes which fulfill the purpose of maintaining link states between the local nodes, such as the local network topology, and that usually use protocols that are based on the framing of packets specific to the link types.
- The Bluetooth low energy is commonly known as BLE or Bluetooth 4.0 and marketed as Bluetooth Smart. Bluetooth Special Interest Group (Bluetooth SIG) designed and marketed this wireless personal area network technology. BLE is an intelligent and low-power version of the classic Bluetooth wireless technology. The use of BLE in today's products have increased drastically because of some important benefits over the classic Bluetooth technology:

    - Lower implementation costs
    - Multi-vendor interoperability
    - Enhanced range
    - Much improved pairing speed

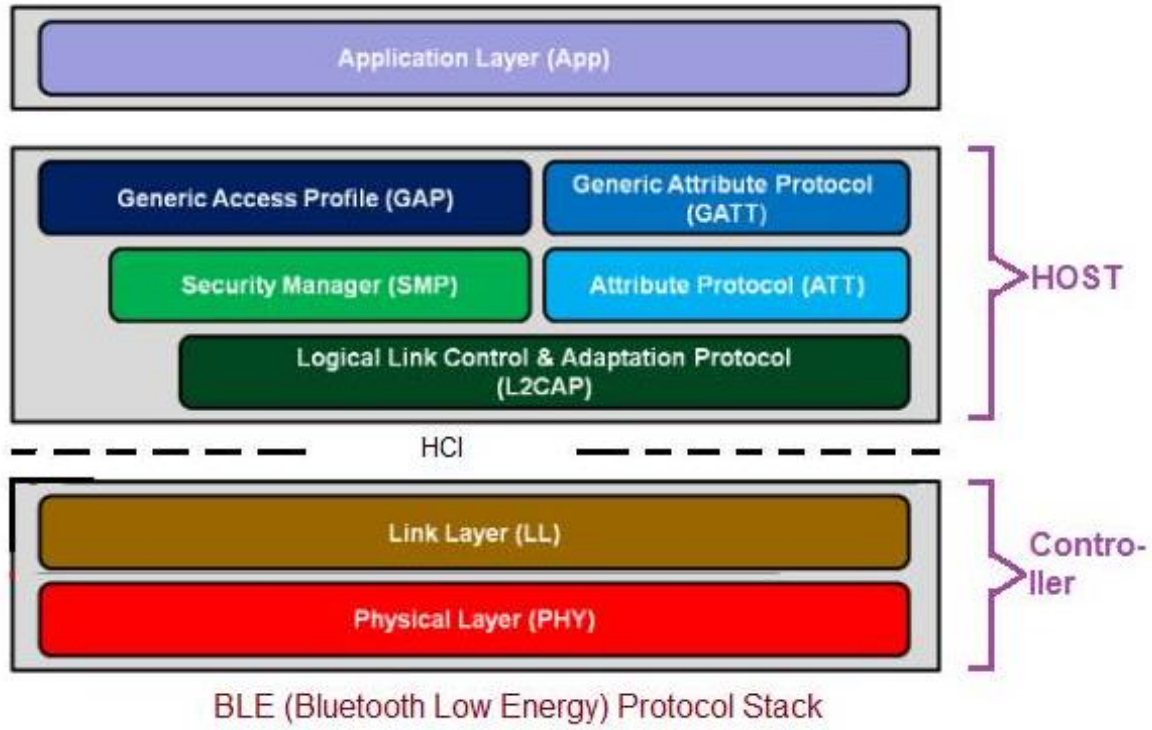


Fig.3.4 BLE architecture

BLE is used in battery-powered devices developed for industries like healthcare, fitness, beacons, security, and home industries. The devices usually transmit a small amount of data as bursts at 1 Mbps speed. However, it is not suitable for sending data in real-time. Some of the features of BLE are:

- Location Detection Accuracy: BLE is used as beacons for micro-locating.

- Proximity Detection: The proximity data provided by BLE is much more accurate than Wi-Fi, but not very accurate either.
- Power Consumption: BLE is a low-power low latency technology.

- Bluetooth low energy (BLE) is a short-range communication network protocol with PHY (physical layer) and MAC (Medium Access Control) layer.
    - HCI: It provides communication between controller and host through standard interface types. This HCI layer can be implemented either using API or by interfaces such as UART/SPI/USB. Standard HCI commands and events are defined in the bluetooth specifications.
      L2CAP: This layer offers data encapsulation services to upper layers. This allows logical end to end data communication.
    - SMP : This security Manager layer provides methods for device pairing and key distributions. It offers services to other protocol stack layers in order to securely connect and exchange data between BLE devices.
    - **GAP:** This layer directly interfaces with application layer and/or profiles on it. It handles device discovery and connection related services for BLE device. It also takes care of initiation of security features.
      GATT: This layer is service framework which specifies sub-procedures to use ATT. Data communications between two BLE devices are handled through these sub-procedures. The applications and/or profiles will use GATT directly.

      • **ATT:** This layer allows BLE device to expose certain pieces of data or attributes.

Application Layer :

• The BLE protocol stack layers interact with applications and profiles as desired. Application interoperability in the Bluetooth system is accomplished by Bluetooth profiles.

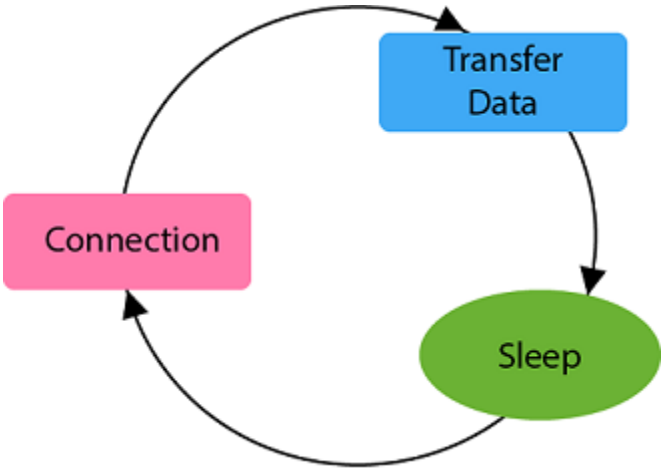


Fig.3.5 BLE

• The profile defines the vertical interactions between the layers as well as the peer-to-peer

interactions of specific layers between devices.
• A profile composed of one or more services to address particular use case. A service consists of characteristics or references to other services.
• Any profiles/applications run on top of GAP/GATT layers of BLE protocol stack. It handles device discovery and connection related services for the BLE device.

It is designed for low-power devices which uses less data. BLE always remain in sleep mode except when the connection between devices is initiated and data transmission occurs, due to this it conserves power of the device. Bluetooth low energy follows the master/slave architecture and offers two types of frames that are adverting and data frames. Slave node sent the advertising frame to discover one or more dedicated advertisement channels. Master nodes sense this advertisement channels to find slaves and connect them.

3.3 LoRAWAN

- The LoRaWAN protocol is a Low Power Wide Area Networking (LPWAN) communication protocol that functions on LoRa.
- LoRaWAN is a low-power, wide area networking protocol built on top of the LoRa radio modulation technique. It wirelessly connects devices to the internet and manages communication between end-node devices and network gateways.
- Usage of LoRaWAN in industrial spaces and smart cities is growing because it is an affordable long-range, bi-directional communication protocol with very low power consumption — devices can run for ten years on a small battery. It uses the unlicensed ISM (Industrial, Scientific, Medical) radio bands for network deployments.
- The LoRaWAN specification is open so anyone can set up and operate a LoRa network.
- LoRa is a wireless audio frequency technology that operates in a license-free radio frequency spectrum. LoRa is a physical layer protocol that uses spread spectrum modulation and supports long-range communication at the cost of a narrow bandwidth. It uses a narrow band waveform with a central frequency to send data, which makes it robust to interference.
- An end device can connect to a network with LoRaWAN in two ways:
    - Over-the-air Activation (OTAA): A device has to establish a network key and an application session key to connect with the network.
    - Activation by Personalization (ABP): A device is hardcoded with keys needed to communicate with the network, making for a less secure but easier connection.

LoRa: LoRa is a radio modulation technique that is essentially a way of manipulating radio waves to encode information using a chirped (chirp spread spectrum technology), multi-symbol format. LoRa as a term can also refer to the systems that support this modulation technique or the communication network that IoT applications use.

The main advantages of LoRa are its long-range capability and its affordability. A typical use case for LoRa is in smart cities, where low-powered and inexpensive internet of things devices

(typically sensors or monitors) spread across a large area send small packets of data sporadically to a central administrator.

Spreading Factor (SF): *The chirp spread spectrum technology uses so-called "chirps," which are signals with a frequency that moves up or down (up-chirp or down-chirp respectively) at different speeds. The spreading factor (SF) determines the speed of a chirp.*
A high SF means a broadcast has higher range and penetration, at the cost of increased power consumption. A lower SF is faster and transmits more data at the same bandwidth and time.

Why LoRaWAN?

- **Ultra low power** - LoRaWAN end devices are optimized to operate in low power mode and can last up to 10 years on a single coin cell battery.
- **Long range** - LoRaWAN gateways can transmit and receive signals over a distance of over 10 kilometers in rural areas and up to 3 kilometers in dense urban areas.
- **Deep indoor penetration** - LoRaWAN networks can provide deep indoor coverage, and easily cover multi floor buildings.
- **License free spectrum** - You don't have to pay expensive frequency spectrum license fees to deploy a LoRaWAN network.
- **Geolocation**- A LoRaWAN network can determine the location of end devices using triangulation without the need for GPS. A LoRa end device can be located if at least three gateways pick up its signal.
- **High capacity** - LoRaWAN Network Servers handle millions of messages from thousands of gateways.
- **Public and private deployments** - It is easy to deploy public and private LoRaWAN networks using the same hardware (gateways, end devices, antennas) and software (UDP packet forwarders, Basic Station software, LoRaWAN stacks for end devices).
- **End-to-end security**- LoRaWAN ensures secure communication between the end device and the application server using AES-128 encryption.
- **Firmware updates over the air** - You can remotely update firmware (applications and the LoRaWAN stack) for a single end device or group of end devices.
- **Roaming**- LoRaWAN end devices can perform seamless handovers from one network to another.
- **Low cost** - Minimal infrastructure, low-cost end nodes and open source software.
- **Certification program**- The LoRa Alliance certification program certifies end devices and provides end-users with confidence that the devices are reliable and compliant with the LoRaWAN specification.
- **Ecosystem**- LoRaWAN has a very large ecosystem of device makers, gateway makers, antenna makers, network service providers, and application developers.

<u>Architecture of LoRaWan</u>

- LoRaWAN networks are deployed in a **star-of-stars** topology.

- A typical LoRaWAN network consists of the following elements.

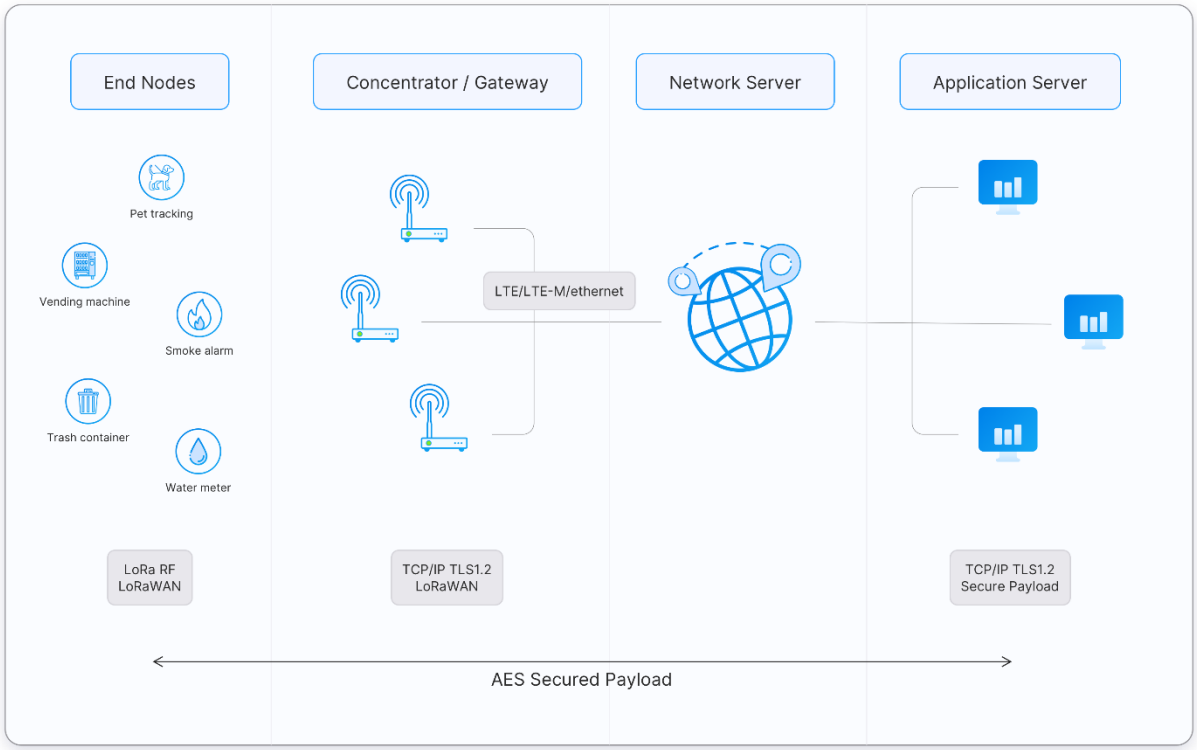


Fig.3.6 A typical LoRaWAN network architecture

- **End Devices** - sensors or actuators send LoRa modulated wireless messages to the gateways or receive messages wirelessly back from the gateways. A LoRaWAN end device can be a sensor, an actuator, or both. They are often battery operated. These end devices are wirelessly connected to the LoRaWAN network through gateways using LoRa RF modulation.
- **Gateways** - receive messages from end devices and forward them to the Network Server. Each gateway is registered (using configuration settings) to a LoRaWAN network server. A gateway receives LoRa messages from end devices and simply forwards them to the LoRaWAN network server. Gateways are connected to the Network Server using a **backhaul** like Cellular (3G/4G/5G), WiFi, Ethernet, fiber-optic or 2.4 GHz radio links.
    - LoRaWAN gateways can be categorized into indoor (picocell) and outdoor (macrocell) gateways. Indoor gateways are cost-effective and suitable for providing coverage in places like deep-indoor locations (spaces covered by multiple walls), basements, and multi-floor buildings. These gateways have internal antennas or external ‘pigtail’ antennas. However depending on the indoor physical environment some indoor gateways can receive messages from sensors located several kilometers away.
    - Outdoor gateways provide a larger coverage than the indoor gateways. They are suitable for providing coverage in both rural and urban areas. These gateways can be mounted on cellular towers, the rooftops of very tall buildings, metal pipes (masts) etc. Usually an outdoor gateway has an external antenna (i.e. Fiberglass antenna) connected using a coaxial cable.

- **Network Server** - a piece of software running on a server that manages the entire network. The Network Server manages gateways, end-devices, applications, and users in the entire LoRaWAN network.
  - The network server provides three key functions:
    - Device authentication and authorization
    - Network management and optimization
    - Interaction with higher-level application servers
    -

A typical LoRaWAN Network Server has the following features.

- Establishing secure 128-bit AES connections for the transport of messages between end-devices and the Application Server (end-to-end security).
- Validating the authenticity of end devices and integrity of messages.
- Deduplicating uplink messages.
- Selecting the best gateway for routing downlink messages.
- Sending ADR commands to optimize the data rate of devices.
- Device address checking.
- Providing acknowledgements of confirmed uplink data messages.
- Forwarding uplink application payloads to the appropriate application servers
- Routing uplink application payloads to the appropriate Application Server.
- Forwarding Join-request and Join-accept messages between the devices and the join server
- Responding to all MAC layer commands.

- **Application servers** - a piece of software running on a server that is responsible for securely processing application data. The Application Server processes application-specific data messages received from end devices. It also generates all the application-layer downlink payloads and sends them to the connected end devices through the Network Server. A LoRaWAN network can have more than one Application Server. The collected data can be interpreted by applying techniques like machine learning and artificial intelligence to solve business problems.
- **Join Server** - A piece of software running on a server that processes join-request messages sent by end devices. The Join Server assists in secure device activation, root key storage, and session key generation. The join procedure is initiated by the end device by sending the Join-request message to the Join Server through the Network Server. The Join-server processes the Join-request message, generates session keys, and transfers NwkSKey and AppSKey to the Network server and the Application server respectively.

End devices communicate with nearby gateways and each gateway is connected to the network server. LoRaWAN networks use an ALOHA based protocol, so end devices don't need to peer with specific gateways. Messages sent from end devices travel through all gateways within range. These messages are received by the Network Server. If the Network Server has received

multiple copies of the same message, it keeps a single copy of the message and discards others. This is known as message deduplication.

Characteristics of LoRaWAN technology

- Long range communication up to 10 miles in line of sight.
- Long battery duration of up to 10 years. For enhanced battery life, you can operate your devices in class A or class B mode, which requires increased downlink latency.
- Low cost for devices and maintenance.
- License-free radio spectrum but region-specific regulations apply.
- Low power but has a limited payload size of 51 bytes to 241 bytes depending on the data rate. The data rate can be 0,3 Kbit/s – 27 Kbit/s data rate with a 222 maximal payload size.

Security in LoRA

- Any communication technology dealing with many connected nodes need robust end-to-end security. LoRa achieve this by implementing security at two different layers:
- One for the network
- One for the application
- Network security ensures authenticity of the node in the network and application security ensures operator does not have access to end user's application data.
- LoRa uses AES (Advanced Encryption Standard) security keys.
- To achieve the required levels of security for LoRa networks, several layers of encryption have been employed:
    - Unique Network key (EUI64) and ensure security on network level
    - Unique Application key (EUI64) ensure end to end security on application level
    - Device specific key (EUI128)

Use cases of LoRaWAN

- **Vaccine cold chain monitoring** - LoRaWAN sensors are used to ensure vaccines are kept at appropriate temperatures in transit.
- **Animal conservation** - Tracking sensors manage endangered species such as Black Rhinos and Amur Leopards.
- **Dementia patients** - Wristband sensors provide fall detection and medication tracking.
- **Smart farms**- Real time insights into crop soil moisture and optimized irrigation schedule reduce water use up to 30%.
- **Water conservation**- Identification and faster repair of leaks in a city's water network.
- **Food safety**- Temperature monitoring ensures food quality maintenance.
- **Smart waste bins** - Waste bin level alerts sent to staff optimize the pickup schedule.
- **Smart bikes**- Bike trackers track bikes in remote areas and dense buildings.
- **Airport tracking** - GPS-free tracking monitors vehicles, personnel, and luggage.

- **Efficient workspaces** - Room occupancy, temperature, energy usage and parking availability monitoring.
- **Cattle health** - Sensors monitor cattle health, detect diseases and forecast calves delivery time.
- **LoRa in space** - Satellites to provide LoRaWAN-based coverage worldwide.

3.4 Network layer

The network layer in IoT is mainly divided into two parts. The routing layer sends packages from origin to destination and the encapsulation layer is largely responsible for creating packets.

1. RPL Protocol: RPL stands for *Routing Protocol for Low-Power and Lossy Network*. It is a distance-vector protocol that supports a variety of Data Link Protocols. RPL builds a **Destination Oriented Directed Acyclic Graph (DODAG)** which has only one route from each leaf node to the root. All the traffic in this DODAG is routed through the root. Initially, each node sends a DODAG Information Object (DIO) announcing them self as a root. This information travels in the network, and complete DODAG is gradually built. When a new node wants to join the network, it sends a DODAG Information Solicitation (DIS) request and root responds back with a DAO Acknowledgment (DAO-ACK) confirming the join.

2. CORPL Protocol: CORPL protocol is the extension of the **RPL protocol**, which is termed as **cognitive RPL**. This network protocol is designed for cognitive networks and uses DODAG topology. CORPL protocol makes two new modifications in the RPL protocol. It uses opportunistic forwarding to forward a packet between the nodes. Each node of CORPL protocol keeps the information of forwarding set rather than parents only maintaining it. Each node updates its changes to its neighbor using DIO messages. On the basis of this updated message, each node frequently updates its neighbor for constant forwarder set.

3. CARP Protocol: CARP (Channel-Aware Routing Protocol) is a distributed routing protocol. It is designed for underwater communication. It has lightweight packets so that it can be used for Internet of Things (IoT). It performs two different functionalities: network initialization and data forwarding. CARP protocol does not support previously collected data. Hence, it is not beneficial for those IoT or other application where data is changed frequently. The upgradation of CARP is done in E-CARP which overcomes the limitation of CARP. The E-CARP allows the sink node to save previously received sensory data.

4. 6LoWPAN: The 6LoWPAN protocol refers to IPv6 Low Power Personal Area Network which uses a lightweight IP-based communication to travel over low data rate networks. It has limited processing ability to transfer information wirelessly using an internet protocol. So, it is mainly used for home and building automation. The 6LoWPAN protocol operates only within the 2.4 GHz frequency range with 250 kbps transfer rate. It has a maximum length of 128-bit header packets.

6LowPAN Security Measure

Security is a major issue for 6LowPAN communication Protocol. There are several attacks issues at the security level of 6LoWPAN which aim is to direct destruction of the network. Since it is the combination of two systems, so, there is a possibility of attack from two sides

that targets all the layer of the 6LoWPAN stack (Physical layer, Data link layer, Adaptation layer, Network layer, Transport layer, Application layer).
Properties of 6LowPAN protocol

- **Standard:** RFC6282
- **Frequency:** Used over a variety of other networking media including Bluetooth Smart (2.4GHz) or ZigBee or low-power RF (sub-1GHz)

3.5 Transport layer

- The transport layer is a 4th layer from the top.
- The main role of the transport layer is to provide the communication services directly to the application processes running on different hosts.

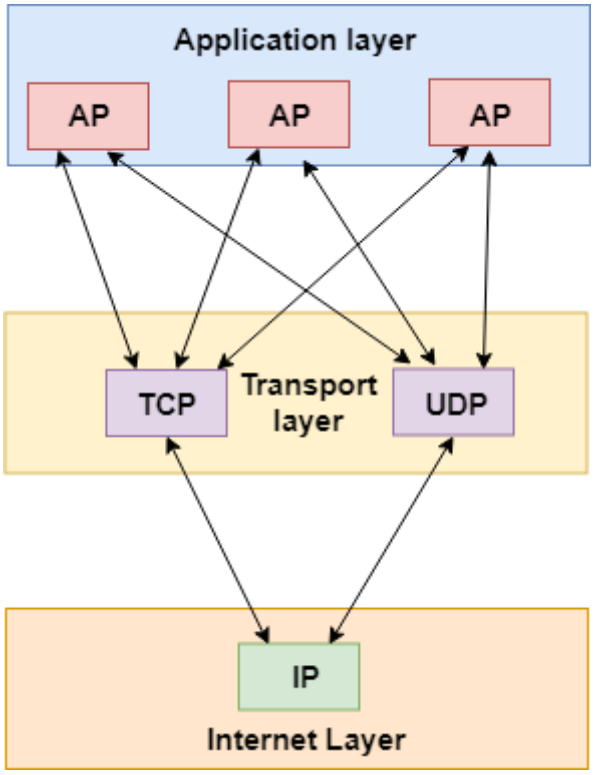


Fig.3.7 ATransport layer

- The transport layer provides a logical communication between application processes running on different hosts. Although the application processes on different hosts are not physically connected, application processes use the logical communication provided by the transport layer to send the messages to each other.
- The transport layer protocols are implemented in the end systems but not in the network routers.
- A computer network provides more than one protocol to the network applications. For example, TCP and UDP are two transport layer protocols that provide a different set of services to the network layer.

- All transport layer protocols provide multiplexing/demultiplexing service. It also provides other services such as reliable data transfer, bandwidth guarantees, and delay guarantees.
- Each of the applications in the application layer has the ability to send a message by using TCP or UDP. The application communicates by using either of these two protocols. Both TCP and UDP will then communicate with the internet protocol in the internet layer. The applications can read and write to the transport layer. Therefore, we can say that communication is a two-way process.

Services provided by the Transport Layer

The services provided by the transport layer are similar to those of the data link layer. The data link layer provides the services within a single network while the transport layer provides the services across an internetwork made up of many networks. The data link layer controls the physical layer while the transport layer controls all the lower layers.

The services provided by the transport layer protocols can be divided into five categories:

- End-to-end delivery
- Addressing
- Reliable delivery
- Flow control
- Multiplexing

End-to-end delivery: The transport layer transmits the entire message to the destination. Therefore, it ensures the end-to-end delivery of an entire message from a source to the destination.

Reliable delivery: The transport layer provides reliability services by retransmitting the lost and damaged packets.

The reliable delivery has four aspects:

- Error control
- Sequence control
- Loss control
- Duplication control

Error Control

- The primary role of reliability is **Error Control**. In reality, no transmission will be 100 percent error-free delivery. Therefore, transport layer protocols are designed to provide error-free transmission.

- The data link layer also provides the error handling mechanism, but it ensures only node-to-node error-free delivery. However, node-to-node reliability does not ensure the end-to-end reliability.
- The data link layer checks for the error between each network. If an error is introduced inside one of the routers, then this error will not be caught by the data link layer. It only detects those errors that have been introduced between the beginning and end of the link. Therefore, the transport layer performs the checking for the errors end-to-end to ensure that the packet has arrived correctly.

Sequence Control

- The second aspect of the reliability is sequence control which is implemented at the transport layer.
- On the sending end, the transport layer is responsible for ensuring that the packets received from the upper layers can be used by the lower layers. On the receiving end, it ensures that the various segments of a transmission can be correctly reassembled.

Loss Control

Loss Control is a third aspect of reliability. The transport layer ensures that all the fragments of a transmission arrive at the destination, not some of them. On the sending end, all the fragments of transmission are given sequence numbers by a transport layer. These sequence numbers allow the receiver?s transport layer to identify the missing segment.

Duplication Control

Duplication Control is the fourth aspect of reliability. The transport layer guarantees that no duplicate data arrive at the destination. Sequence numbers are used to identify the lost packets; similarly, it allows the receiver to identify and discard duplicate segments.

Flow Control

Flow control is used to prevent the sender from overwhelming the receiver. If the receiver is overloaded with too much data, then the receiver discards the packets and asking for the retransmission of packets. This increases network congestion and thus, reducing the system performance. The transport layer is responsible for flow control. It uses the sliding window protocol that makes the data transmission more efficient as well as it controls the flow of data so that the receiver does not become overwhelmed. Sliding window protocol is byte oriented rather than frame oriented.

Multiplexing

The transport layer uses the multiplexing to improve transmission efficiency.

Multiplexing can occur in two ways:

- **Upward multiplexing:** Upward multiplexing means multiple transport layer connections use the same network connection. To make more cost-effective, the transport layer sends several transmissions bound for the same destination along the same path; this is achieved through upward multiplexing.
- **Downward multiplexing:** Downward multiplexing means one transport layer connection uses the multiple network connections. Downward multiplexing allows the transport layer to split a connection among several paths to improve the throughput. This type of multiplexing is used when networks have a low or slow capacity.

Addressing

- According to the layered model, the transport layer interacts with the functions of the session layer. Many protocols combine session, presentation, and application layer protocols into a single layer known as the application layer. In these cases, delivery to the session layer means the delivery to the application layer. Data generated by an application on one machine must be transmitted to the correct application on another machine. In this case, addressing is provided by the transport layer.

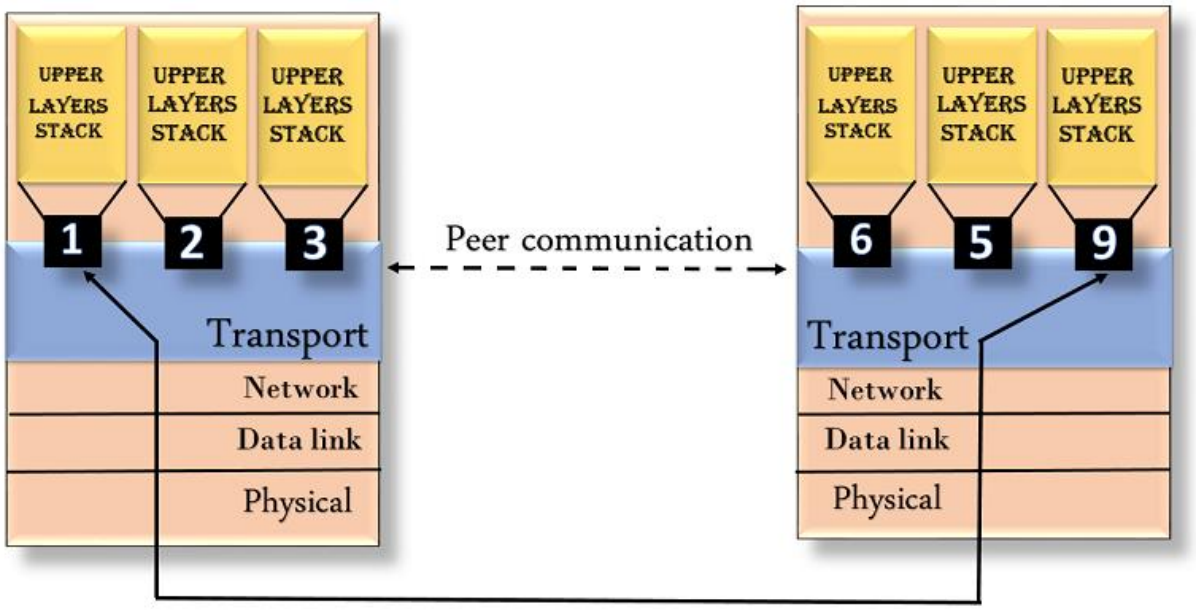


Fig.3.8 Addressing in Transport layer

- The transport layer provides the user address which is specified as a station or port. The port variable represents a particular TS user of a specified station known as a Transport Service access point (TSAP). Each station has only one transport entity.

- The transport layer protocols need to know which upper-layer protocols are communicating.

3.6 Application Layer protocols –MQTT

MQTT is a standards-based messaging protocol, or set of rules, used for machine-to-machine communication. Smart sensors, wearables, and other Internet of Things (IoT) devices typically have to transmit and receive data over a resource-constrained network with limited bandwidth. These IoT devices use MQTT for data transmission, as it is easy to implement and can communicate IoT data efficiently. MQTT supports messaging between devices to the cloud and the cloud to the device.

Why is the MQTT protocol important?

The MQTT protocol has become a standard for IoT data transmission because it delivers the following benefits:

- **Lightweight and efficient:** MQTT implementation on the IoT device requires minimal resources, so it can even be used on small microcontrollers. For example, a minimal MQTT control message can be as little as two data bytes. MQTT message headers are also small so that you can optimize network bandwidth.
- **Scalable:** MQTT implementation requires a minimal amount of code that consumes very little power in operations. The protocol also has built-in features to support communication with a large number of IoT devices. Hence, you can implement the MQTT protocol to connect with millions of these devices.
- **Reliable:** Many IoT devices connect over unreliable cellular networks with low bandwidth and high latency. MQTT has built-in features that reduce the time the IoT device takes to reconnect with the cloud. It also defines three different quality-of-service levels to ensure reliability for IoT use cases— at most once (0), at least once (1), and exactly once (2).
- **Secure:** MQTT makes it easy for developers to encrypt messages and authenticate devices and users using modern authentication protocols, such as OAuth, TLS1.3, Customer Managed Certificates, and more.
- **Well-supported:** Several languages like Python have extensive support for MQTT protocol implementation. Hence, developers can quickly implement it with minimal coding in any type of application.

Characteristics of MQTT

The MQTT has some unique features which are hardly found in other protocols. Some of the features of an MQTT are given below:

- It is a machine to machine protocol, i.e., it provides communication between the devices.
- It is designed as a simple and lightweight messaging protocol that uses a publish/subscribe system to exchange the information between the client and the server.
- It does not require that both the client and the server establish a connection at the same time.

- It provides faster data transmission, like how WhatsApp/messenger provides a faster delivery. It's a real-time messaging protocol.
- It allows the clients to subscribe to the narrow selection of topics so that they can receive the information they are looking for.

Components of MQTT

MQTT implements the publish/subscribe model by defining clients and brokers as below.

MQTT client: An MQTT client is any device from a server to a microcontroller that runs an MQTT library. If the client is sending messages, it acts as a publisher, and if it is receiving messages, it acts as a receiver. Basically, any device that communicates using MQTT over a network can be called an MQTT client device.

MQTT broker: The MQTT broker is the backend system which coordinates messages between the different clients. Responsibilities of the broker include receiving and filtering messages, identifying clients subscribed to each message, and sending them the messages. It is also responsible for other tasks such as:

- **Authorizing and authenticating MQTT clients**
- **Passing messages to other systems for further analysis**
- **Handling missed messages and client sessions**

MQTT connection: Clients and brokers begin communicating by using an MQTT connection. Clients initiate the connection by sending a *CONNECT* message to the MQTT broker. The broker confirms that a connection has been established by responding with a *CONNACK* message. Both the MQTT client and the broker require a TCP/IP stack to communicate. Clients never connect with each other, only with the broker.

Architecture of MQTT

The MQTT protocol works on the principles of the publish/subscribe model. In traditional network communication, clients and servers communicate with each other directly. The clients request resources or data from the server, then the server processes and sends back a response. However, MQTT uses a publish/subscribe pattern to decouple the message sender (publisher) from the message receiver (subscriber). Instead, a third component, called a message broker, handles the communication between publishers and subscribers. The broker’s job is to filter all incoming messages from publishers and distribute them correctly to subscribers. The broker decouples the publishers and subscribers as below:

- **Space decoupling-** *The publisher and subscriber are not aware of each other's network location and do not exchange information such as IP addresses or port numbers.*
- **Time decoupling-** *The publisher and subscriber don't run or have network connectivity at the same time.*
- **Synchronization decoupling-***Both publishers and subscribers can send or receive messages without interrupting each other. For example, the subscriber does not have to wait for the publisher to send a message.*

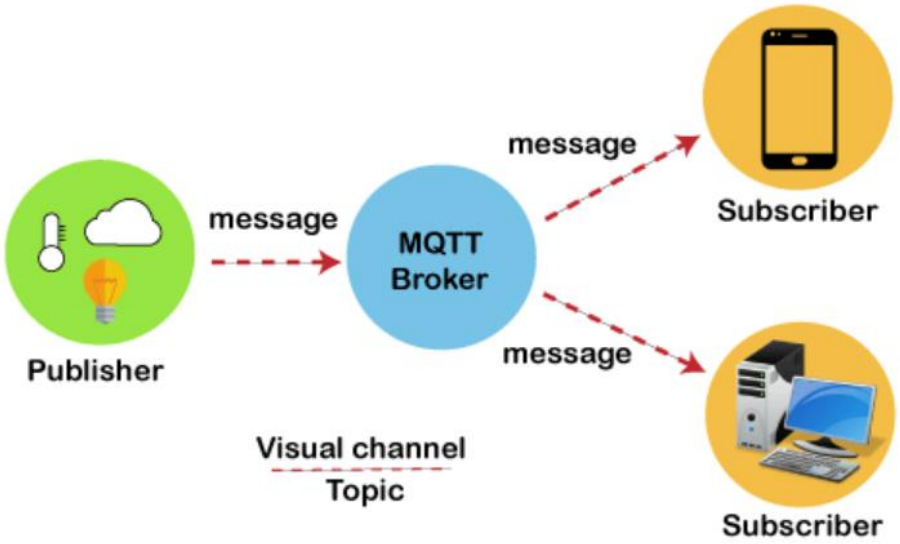


Fig.3.9 MQTT Architecture

- **Message**
- **Client**
- **Server or Broker**

Message

The message is the data that is carried out by the protocol across the network for the application. When the message is transmitted over the network, then the message contains the following parameters:

1. Payload data
2. Quality of Service (QoS)
3. Collection of Properties
4. Topic Name

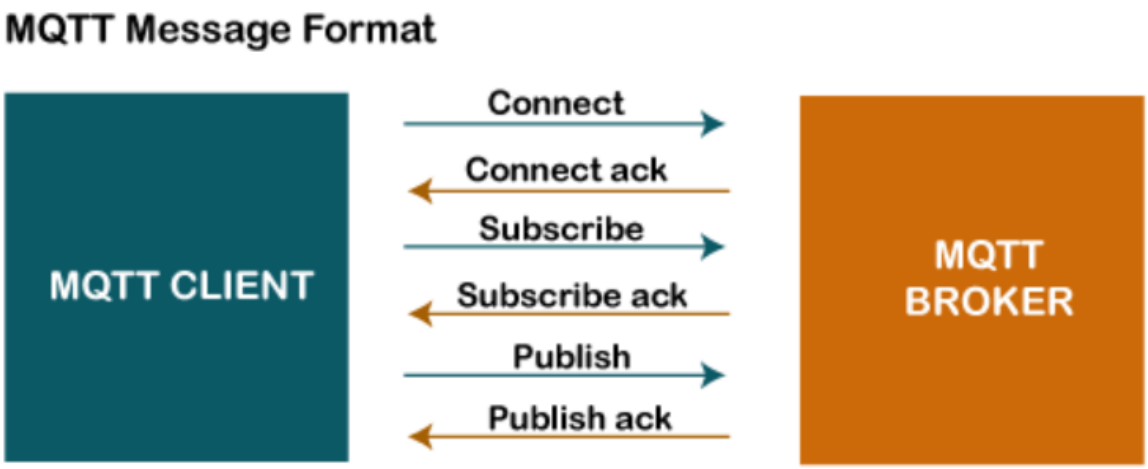


Fig.3.10 MQTT Message Format

Client

In MQTT, the subscriber and publisher are the two roles of a client. The clients subscribe to the topics to publish and receive messages. In simple words, we can say that if any program or device uses an MQTT, then that device is referred to as a client. A device is a client if it opens the network connection to the server, publishes messages that other clients want to see, subscribes to the messages that it is interested in receiving, unsubscribes to the messages that it is not interested in receiving, and closes the network connection to the server.

In MQTT, the client performs two operations:

In MQTT, the client performs two operations:

Publish: When the client sends the data to the server, then we call this operation as a publish.

Subscribe: When the client receives the data from the server, then we call this operation a subscription.

Server

The device or a program that allows the client to publish the messages and subscribe to the messages. A server accepts the network connection from the client, accepts the messages from the client, processes the subscribe and unsubscribe requests, forwards the application messages to the client, and closes the network connection from the client.

- MQTT's publish/subscribe (pub/sub) communication strategy, which aims to maximize bandwidth utilization, is a substitute for traditional consumer architecture directly interacting with an endpoint. However, in the pub/sub paradigm, the client who transmits the news (the publisher) is separated from the customers receiving the information (or the subscribers). Since neither the writers nor the customers communicate with each other immediately, their interactions in them are handled by third parties called brokers.
- Publishers and subscribers are two types of MQTT clients, depending on whether the client is publishing or getting messages. One can combine the two features in a single MQTT client. A publish is when a device (or client) wishes to send information to the server (or broker). In the middle, there is a central server or broker that acts as a mediator. Each incoming message is filtered by the broker, who then sends them to the appropriate customers.

- Customers include both publishers and subscribers. Subscribers sign up for messages that are relevant to them. Topic-based, content-based, or type-based filtering are all possibilities. Topics could be arranged in a logical order. The publisher and subscribers do not need to identify one another when using the publish/subscribe structure. For setup and synchronization, they are not reliant on one another. Message distribution is one-to-many and may scale in terms of customer base without putting publishers under undue strain.
- One of the ways MQTT reduces communications is by using a well-defined, compact message structure. Every message contains a two-byte fixed header. The introduction of an optional title enhances the message's quantity. The payload of the message is restricted to 256 MB. Network administrators can select among both limiting data transmission and maximizing reliability using three different quality of service (QoS) levels which include;

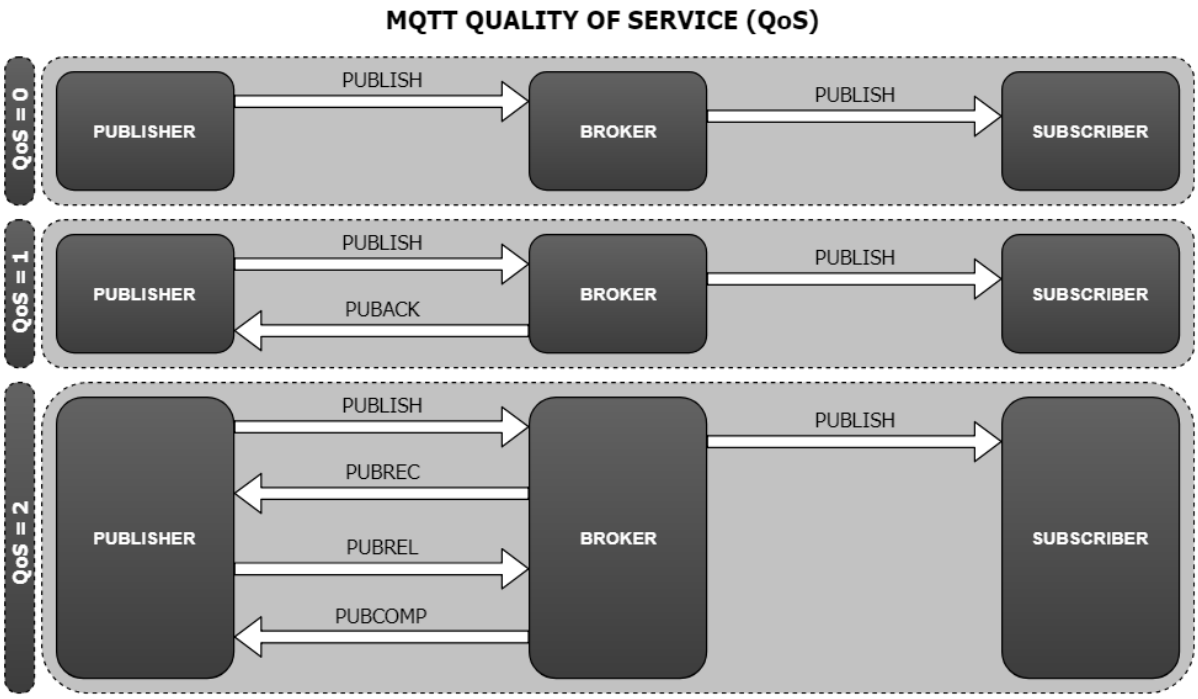


Fig.3.11 MQTT QOS

- **QoS 0**: Provides the bare minimum of data transfer. Every message is transmitted to a user only once at this stage, with no verification. There is no means of knowing whether the recipients received the message. "Fire and forget" or "at the very most once distribution" are terms used to describe this strategy. Messages are not saved for delivery to disconnected clients who rejoin later since this level presumes delivery is accomplished.
- **QoS 1**: This broker tries to send the message and then checks for the user's approval reply. When no verification is received in a certain amount of time, the message is resent. If the broker does not get the user's confirmation on time, the

message may be sent to the user multiple times using this approach. “At least once delivery” is a term used to describe this.

- **QoS 2**: A four-step handshake is used by the client and broker to ensure that information is received and only once. This is referred to as “exactly once delivery” on occasion. QoS 0 may be the optimal option when communications are reliable but constrained. QoS 2 is the optimum solution in cases where communications are inconsistent but links aren’t as resource-constrained.

QoS 1 provides a solution that combines the best of both environments, but it requires that the program receiving the data knows how to handle duplicates. For disconnected users with persistent sessions, both QoS 1 and QoS 2 messages are retained or queued.

3.7 CoAP – Communication Models

CoAP, which stands for Constrained Application Protocol, is a lightweight protocol designed for machine-to-machine communication in constrained IoT devices and networks. It is built upon the principles of REST (Representational State Transfer) and operates over UDP (User Datagram Protocol) or DTLS (Datagram Transport Layer Security) for secure communication. It enables resource-constrained devices to communicate and interact with resources in a simple and scalable manner, making it a popular choice for IoT applications.

- A generic framework for resource-oriented applications targeting constrained nodes and networks.
- Defines simple and flexible ways to manipulate sensors and actuators for data or device management.

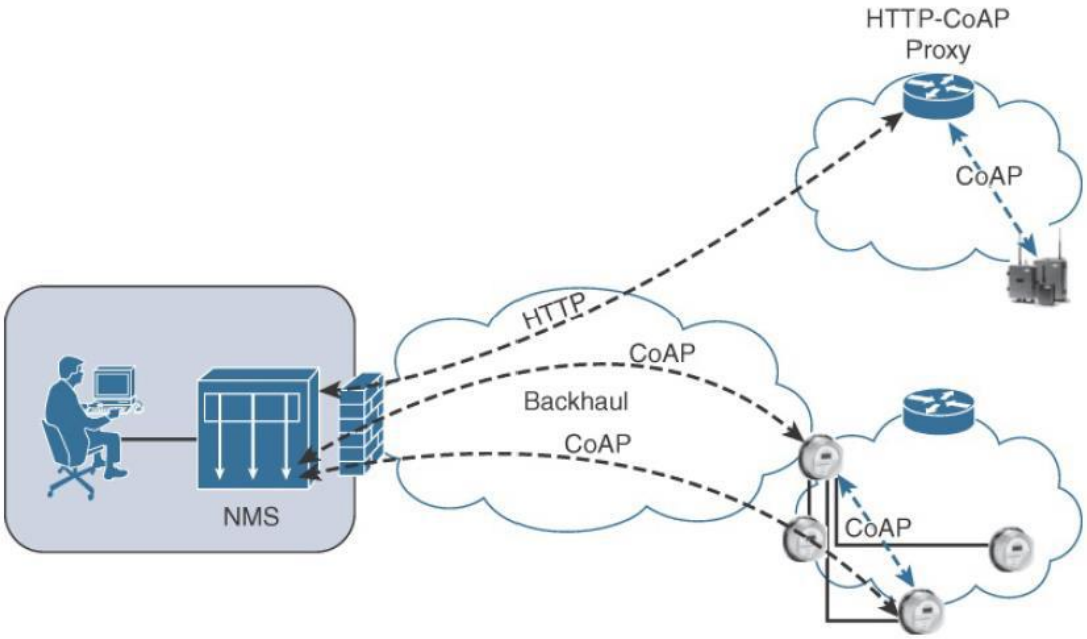


Fig.3.12 CoAP Communication Protocol

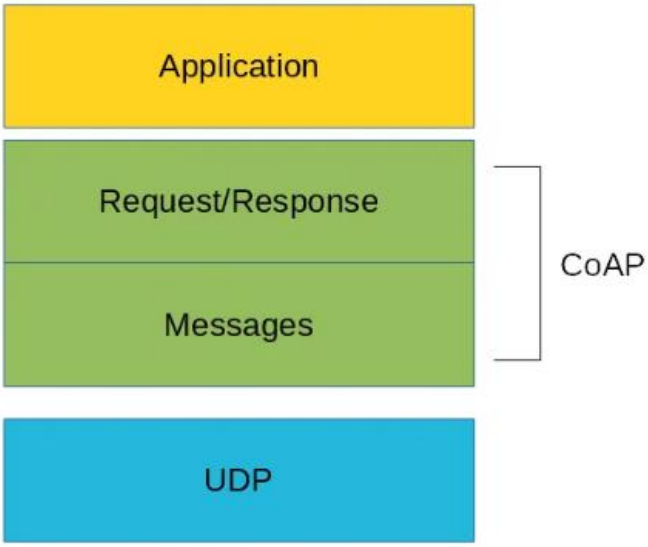


Fig.3.12 CoAP Communication Protocol-layers

CoAP (Constrained Application Protocol) is a machine-to-machine (M2M), specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things (IoT). The CoAP is not only designed to work with devices on the same constrained network, but also with ones over different networks, offering interoperability. The CoRE (Constrained RESTful Environments Working Group) group has designed CoAP with the following features in mind:

- Overhead and parsing complexity.
- URI and content-type support.
- Support for the discovery of resources provided by known CoAP services.
- Simple subscription for a resource, and resulting push notifications.
- Simple caching based on max-age.

Like HTTP, CoAP is based on the wildly successful REST protocol model: Servers make resources available under a URL, and clients access these resources using methods such as GET, PUT, POST, and DELETE. The link between CoAP and HTTP are also defined. But the difference is that CoAP uses the UDP protocol. This allows proxies to be built providing access to CoAP resources via HTTP in a uniform way. The code footprint for CoAP is very less and thus finds its way into microcontrollers with as low as 10 KB of RAM and 100 KB of code space. CoAP has a good security feature too. CoAP's default choice of DTLS (Datagram Transport Layer Security) parameters is equivalent to 3072-bit RSA keys.

The salient features of CoAP are

1. Lightweight: CoAP is designed to be lightweight and efficient, making it suitable for resource-constrained devices with limited processing power, memory, and bandwidth. It uses compact binary encoding and has a small message overhead, reducing the payload size and conserving network resources.

2. Request-Response Model: CoAP follows a request-response model similar to HTTP. Clients send CoAP requests to servers, which respond with CoAP messages. CoAP supports various request methods such as GET, POST, PUT, and DELETE, allowing clients to interact with resources on the server.

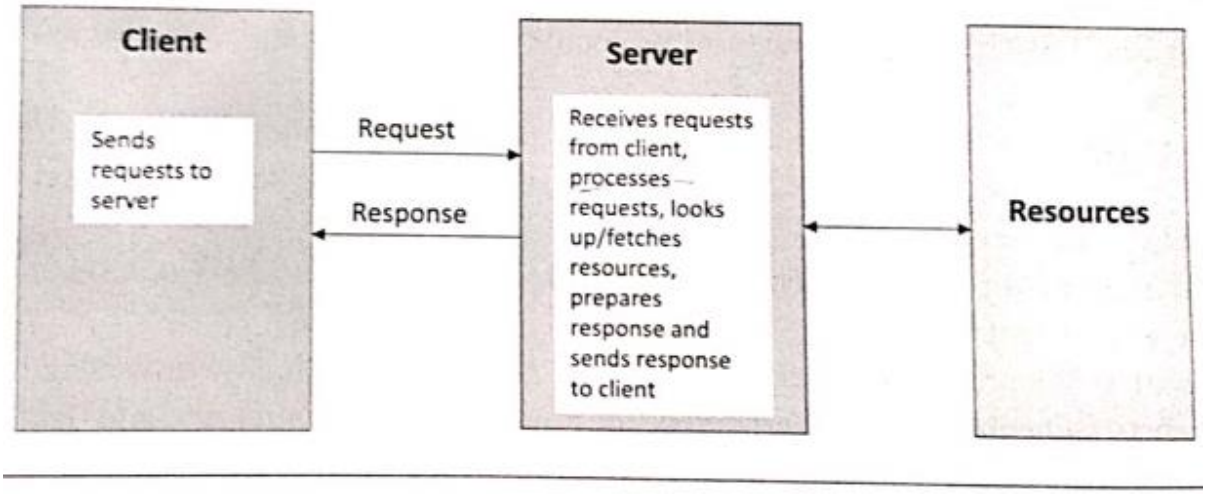


Fig.3.13 Request-Response Communication Model

3. RESTful Interaction: CoAP employs a RESTful architecture, allowing clients to access and manipulate resources using uniform resource identifiers (URIs). It supports standard HTTP-like methods and status codes, making it familiar and easy to integrate with existing web-based systems.
4. Reliable and Unreliable Messaging: CoAP provides both reliable and unreliable messaging options. It offers a Confirmable message type for reliable transmission, where the recipient acknowledges the receipt of the message. It also supports Non-Confirmable messages for one-way, fire-and-forget communication, which is useful for scenarios where reliability is not critical.
5. Resource Discovery: CoAP includes a resource discovery mechanism, allowing clients to discover available resources on a CoAP server. The CoAP server exposes a /.well-known/core resource that provides a list of all available resources and their corresponding URIs, making it easier for clients to navigate the server's resources.
6. Caching and Proxying: CoAP supports caching of responses, which helps reduce network traffic and improve performance. It also includes proxying capabilities, enabling CoAP messages to be forwarded through intermediate CoAP proxies to reach resources on different networks.
7. Security: CoAP provides built-in security features to protect data and communication. It can operate over DTLS, which adds a secure layer on top of UDP. DTLS provides encryption, authentication, and integrity checks, ensuring secure communication between CoAP devices.

Messaging in CoAP

- CoAP can run over IPv4 or IPv6.
- CoAP communications across an IoT infrastructure can take various paths.
- Connections can be between devices located on the same or different constrained networks or between devices and generic Internet or cloud servers, all operating over IP.

- Through the exchange of asynchronous messages, a client requests an action via a method code on a server resource.
- A uniform resource identifier (URI) localized on the server identifies this resource.
- The server responds with a response code that may include a resource representation.
- The CoAP request/response semantics include the methods GET, POST, PUT, and DELETE.

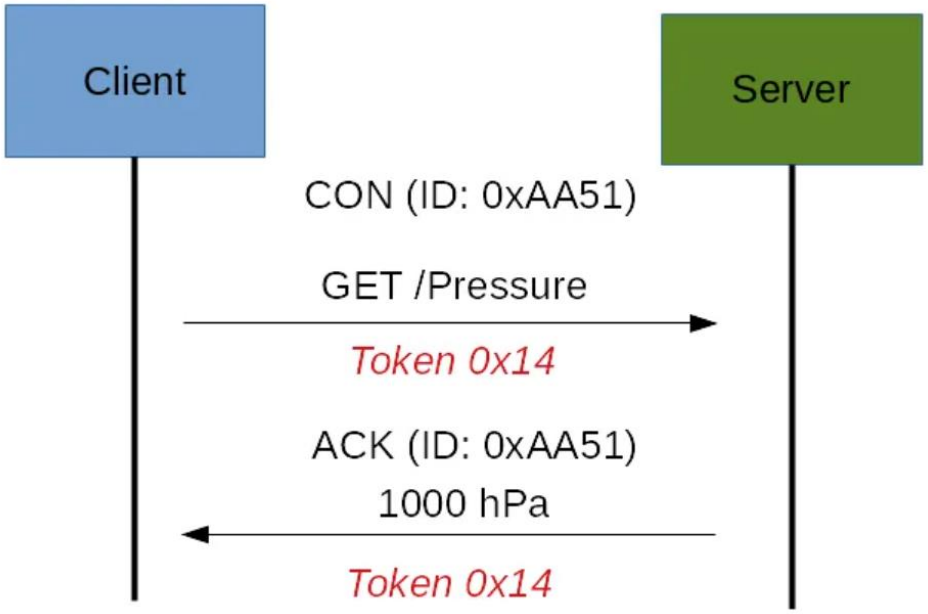


Fig.3.14 Messaging in CoAP

Features of CoAP

- Web protocol used in M2M with constrained requirements-As it shares great similarities with HTTP, developers face bare minimum difficulties while using it.
- Asynchronous message exchange
- Low overhead and very simple to parse
- Proxy and caching capabilities
- Integration-friendly protocol and can be paired easily with applications using cross-protocol proxies.
- Designed to handle such huge mode amounts with full perfection while keeping the overheads under control.
- It can operate on tons of microcontrollers while using the least possible resources.
- RAM space as low as 10KiB and code space as 100 KiB is enough for CoAP.

Advantages

- Will fit in microcontrollers with as low as 10 KB of RAM and 100 KB of code space.
- Follows REST protocol model rendering it easy to implement.
- It has reduced power requirements as it operates over UDP.
- Has smaller packet size, leading to faster communication (even smaller than MQTT).

Disadvantages

- As it uses UDP, the reliability is in question.
- Different standards of CoAP are still evolving.

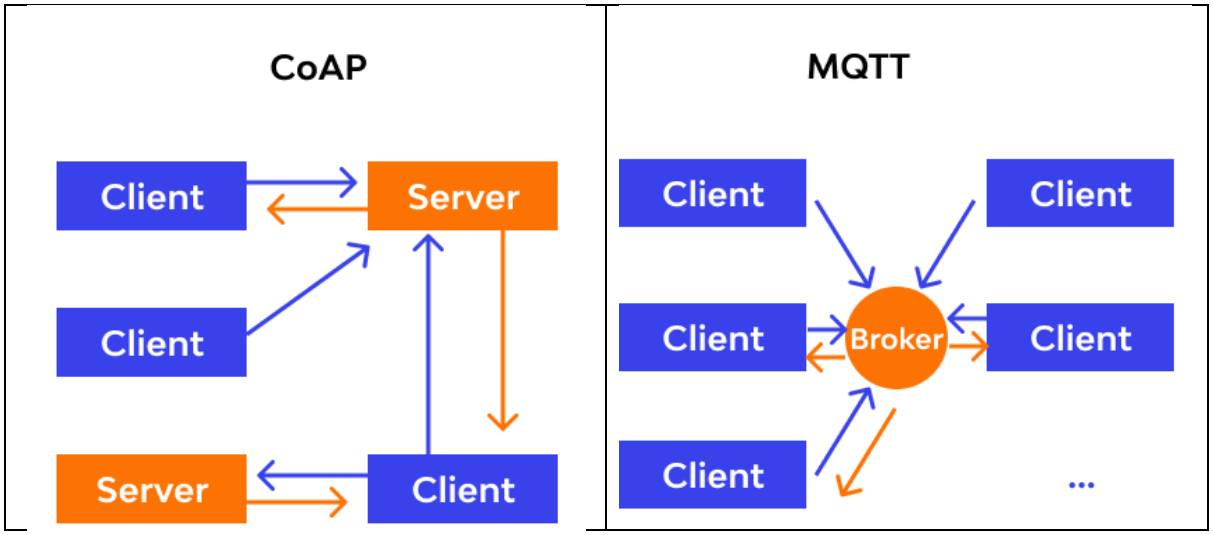
Applications:

CoAP is commonly used in various IoT applications, such as smart home automation, industrial monitoring and control, asset tracking, and environmental monitoring. Its lightweight nature and support for constrained devices make it an ideal choice for IoT deployments. CoAP has several applications in various domains, particularly in IoT scenarios where constrained devices and networks are involved. Here are some highlighted applications and use cases where CoAP can be utilized:

1. Smart Home Automation: CoAP can be used to connect and control smart devices within a home environment. It enables seamless communication between smart devices such as thermostats, lighting systems, security cameras, and appliances, allowing users to remotely monitor and manage their home automation systems.
2. Industrial Monitoring and Control: CoAP finds application in industrial environments for monitoring and controlling devices and processes. It enables real-time data acquisition and control of sensors, actuators, and machinery on the factory floor, facilitating efficient industrial automation and process optimization.
3. Asset Tracking: CoAP can be employed for tracking and monitoring assets in logistics, transportation, and supply chain management. By equipping assets with CoAP-enabled tracking devices, their location, status, and other relevant information can be transmitted to a central system, enabling efficient asset management and logistics operations.
4. Environmental Monitoring: CoAP can be utilized for environmental monitoring applications such as air quality monitoring, water quality monitoring, and weather stations. Constrained sensor nodes equipped with CoAP can collect and transmit environmental data to a central server, facilitating real-time monitoring and analysis for environmental studies and decision-making.
5. Smart Agriculture: CoAP can be applied in agriculture for monitoring and managing various aspects of farming. It enables the connection and communication between agricultural sensors, irrigation systems, weather stations, and farm equipment, allowing farmers to make informed decisions regarding irrigation, pest control, and crop management.
6. Healthcare and Wearable Devices: CoAP can be used in healthcare applications for connecting and exchanging data with wearable devices, health monitors, and medical equipment. It enables remote monitoring of patients, collection of vital signs, and timely transmission of health-related data to healthcare providers for analysis and decision-making.

CoAP vs. MQTT

Parameters	MQTT	CoAP
Header Size	It has a 2 bytes sized header.	It has a 4 bytes sized header.
Reliability	There are three quality of service levels: delivery not guaranteed, delivery confirmation, and delivery double confirmation.	It takes care of confirmable messages, non-confirmable messages, acknowledgements, and re-transmissions.
Abbreviation	Message Query Telemetry Transport.	Constrained Application Protocol.
Effectiveness	Effectiveness in LNN is low.	Effectiveness in LNN is excellent.
Implementation	Although easy to implement, it is hard to add extensions.	There are a few existing libraries and support.
Messaging Mode	This uses only Asynchronous mode of messaging.	This uses both Synchronous and Asynchronous.
Message Labeling	It does not add labels to the messages.	It is provided by adding labels to the messages.
Communication Type	It uses the Publish-Subscribe model.	It uses the Request-Response model.
Persistence Support	It supports live data communication.	It does not offer any such support.
Communication Model	It follows the "many-many" or "M:M" model.	It follows the "one-one" or "1:1" model.
Transport Layer Protocol	This mainly uses TCP.	This mainly uses UDP.
Protocol Maturity & Stability	It is more mature and stable.	It is slightly unstable and immature.



Comparison of Application IoT Protocols

	MQTT	AMQP	HTTP	CoAP
Abstraction	Pub/Sub	Pub/Sub	Request/Reply	Request/Reply
Architecture	Brokered	P2P or Brokered	P2P	P2P
QoS	3	3	Provided by TCP	Confirmable and no Confirmable
Interoperability	Partial	Yes	Yes	Yes
Real-time	No	No	No	No
Transports	TCP	TCP	TCP	UDP
Subscription Control	hierarchical matching	Exchanges, Queues and bindings	N/A	support for Multicast addressing msgs.
Data Serialization	Undefined	AMQP type system or user defined	No	Configurable
Dynamic Discovery	No	No	No	Yes
Security	SSL	TLS	SSL/TLS	DTLS

Module:4 Edge Computing

1. Introduction to Edge/Fog computing

A. Edge Computing

- Edge Computing brings the decentralization of networks.
- Edge Computing is a distributed computing system that allows to bring computation of data and storage too close to the source (where data is required). It brings computing as much close as possible so as to minimize the bandwidth, improve response time, and use of latency. Instead of locating the data at a centralized place, the concept of edge computing believes in distributing the computing process of the data. However, cloud computing and IoT are faster plus efficient, but edge computing is a faster computing method.
- The objective of Edge Computing is to improve the network technology by moving the computation of data close to the edge of the network and away from the data centers. Such a process exploits network gateways or smart objects for performing tasks and provide services on behalf of the cloud.
- This technology increases the efficient usage of bandwidth by analyzing the data at the edges itself, unlike the cloud which requires the transfer of data from the IOT requiring large bandwidth, making it useful to be used in remote locations with minimum cost.
- It allows smart applications and devices to respond to data almost at the same time which is important in terms of business and self-driving cars.
- It has the ability to process data without even putting it on a public cloud, this ensures full security.
- Data might get corrupt while on an extended network thus affecting the data reliability for the industries to use.
- Edge computation of data provides a limitation to the use of cloud.

Why Edge Computing

Edge Computing is a new type of technology that will not only save time but also save the cost of servicing and other charges too.

- Through edge computing, it allows smart applications and devices for responding to data very quickly as soon as it is created and thus removing the lag time.
- Edge Computing also enables data stream acceleration that includes real-time processing of data without latency use. Data Stream acceleration is, however, critical for self-driving cars type of technologies and provides equal and vital benefits for the businesses.
- Efficient processing of data at large scale by allowing processing close to the source, and it saves the use of internet bandwidth also. Thus, it reduces the cost and enables effective accessibility to the applications in remote locations.

- The ability of edge computing to provide services and processing data at the furthest distance makes a secured layer for the sensitive data without keeping it into a public cloud.
- Computing tasks demand suitable architectures, and the architecture that suits one type of computing task doesn't necessarily fit all types of computing tasks. Edge computing has emerged as a viable and important architecture that supports distributed computing to deploy compute and storage resources closer to -- ideally in the same physical location as -- the data source. In general, distributed computing models are hardly new, and the concepts of remote offices, branch offices, data center colocation and cloud computing have a long and proven track record. But decentralization can be challenging, demanding high levels of monitoring and control that are easily overlooked when moving away from a traditional centralized computing model. Edge computing has become relevant because it offers an effective solution to emerging network problems associated with moving enormous volumes of data that today's organizations produce and consume.

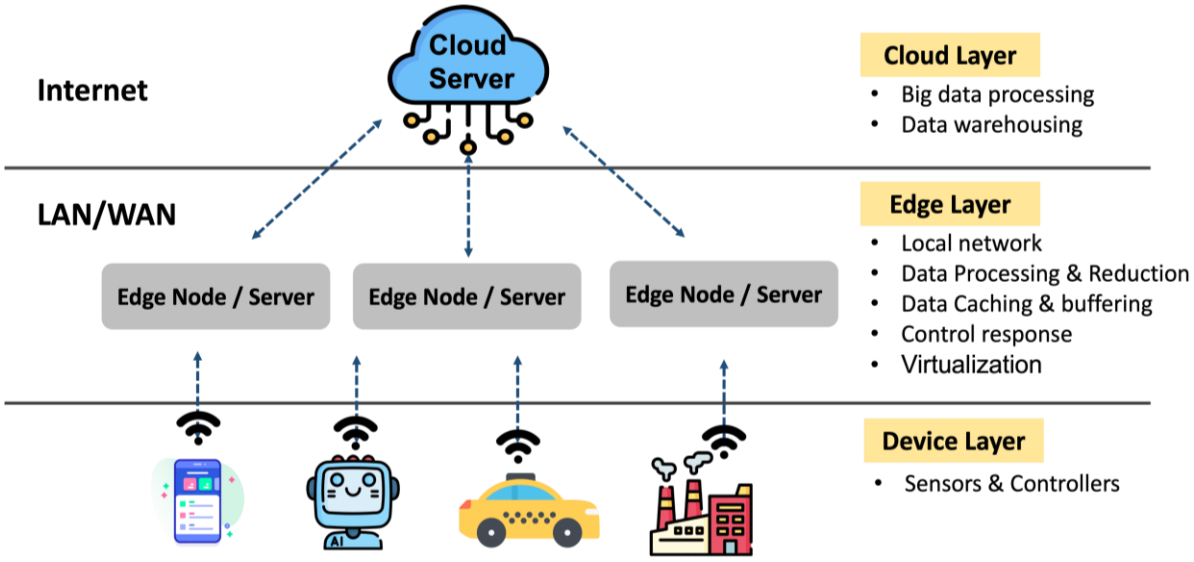
Edge Computing Architecture

Edge computing is the process of storing individual user data as near to the source as possible. This could be at the periphery, or the edge, of an associated network. Many opportunities arise for the positioning of servers at the edge of a network, as well as data storage and processors. Due to the fact this data is closer to the user, it means it minimizes latency. This is a distributed I.T. network that ultimately leverages the minimal distance required. It increases the speed of service, and this generates the associated heightening of value.

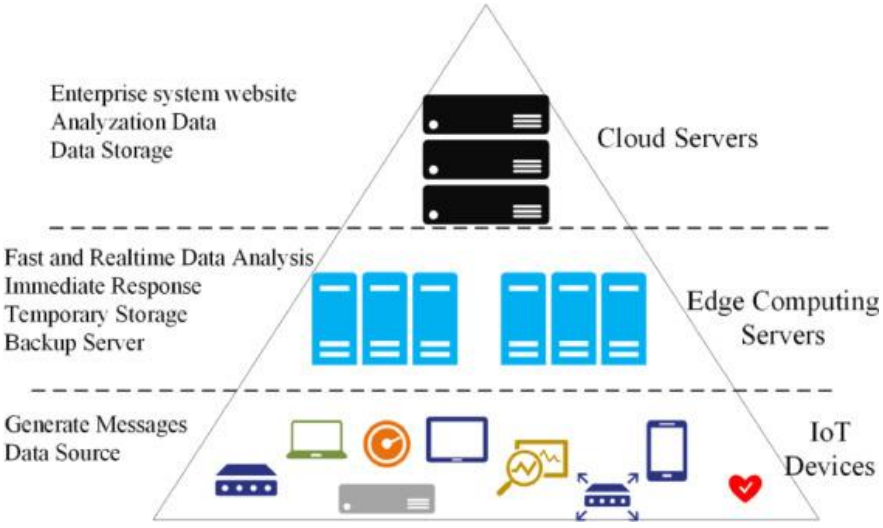
- Location is the primary concern in edge computing. Due to the rapid rise of devices and the amount of data being transferred through the Internet, traditional data centers are struggling to keep up. Therefore, the focus is now being targeted to the infrastructure’s logical edge, relocating resources to the point of data generation. In essence, instead of data traveling to the data center, the data center is repositioned closer to the data.
- However, closer does not necessarily mean physically closer, it means closer in terms of the network and routing. Depending on the number of service providers a business utilizes, such as the cloud, etc., there could be many systems all potentially able to be the edge. Nevertheless, storage and servers are set up where the data is.
- All this requires is a small amount of computing setup to operate a remote LAN. Computing gear is applied to the network and protected against environmental factors in various ways. When the data is processed, the data stream is normalized and analyzed for business intelligence. The results of this are the only pieces of data that are rerouted back to the main data center.
- In a traditional setting, data is produced on a user's computer or any other client application. It is then moved to the server through channels like the internet, intranet, LAN, etc., where the data is stored and worked upon. This remains a classic and proven approach to client-server computing. However, the exponential growth in the volume

of data produced and the number of devices connected to the internet has made it difficult for traditional data center infrastructures to accommodate them. **The concept of edge computing is simple - instead of getting the data close to the data center, the data center is brought close to the data. The storage and computing resources from the data center are deployed as close as possible (ideally in the same location) to where the data is generated.**

A typical edge computing architecture as shown in Fig. 4.1 can be divided into three layers: The cloud layer, or the layer that is responsible for processing and storing all data; The edge layer, or the layer that handles the data processing near real time; And the device layer, or the layer that is in charge of detecting and performing simple processing.



(a) Generalised Edge Computing Architecture



(b) 3.5-tier container-based edge computing architecture

Fig. 4.1 (a) and (b) Edge computing architecture

- **Cloud Layer:** Although edge computing was introduced to address network congestion and latency problems commonly found in cloud computing, cloud computing in fact still plays an important role in the entire edge computing architecture. The cloud computing and edge computing complement one another. Through the edge layer described in the next section, the entire system determines if data needs to be processed in the cloud layer. If that is the case, edge servers will pass data to the cloud layer for complex processing. On the other hand, edge servers will also pass a part or critical data to the cloud layer for storage and comprehensive analysis. This also demonstrates the integration between both the cloud and edge layers.
- **Edge Layer:** This layer mainly consists of edge servers, and when compared to the cloud layer, the edge layer contains edge servers that are larger in quantity and more vastly deployed. Therefore, through distributed edge computing, the edge layer can process data that is closer to the data source and address latency problems found in cloud computing. The edge layer can be considered the core in the entire edge computing architecture. After data from the device layer is analyzed and processed in the edge layer, data is transmitted to the cloud layer for subsequent processing and analysis. Data which cannot be processed in the edge layer can be sent to and analyzed in the cloud layer to ensure data integrity.
- **Device Layer:** Amongst the three layers, the device layer contains the most devices. Ranging from devices that are as small as our mobile phones or computers to ones that are as large as buses and factories, these devices are all examples of components in the device layer. Through their sensors, devices in the device layer collect and capture data used to help products achieve the purposes they are designed for. Equipment in a hospital collecting vital signs of patients and autonomous vehicles capturing data of other nearby vehicles are all such examples. Although components in the cloud and edge layers possess better computing power, the devices in the device layer can still perform data analyses, processing and storage tasks which require negligible computing power, as well as process data closest to the data source in almost real-time.

Security Risks and its measures:

Edge computing has many benefits, including lower latency, higher security, and improved efficiency. However, there are also several security risks and challenges associated with this technology.

- **Data Breach:** One of the biggest security risks with edge computing is data breaches. When data is stored locally on devices instead of in a central location, it becomes much easier for hackers to access it. Hackers can target individual devices or networks of devices to gain access to sensitive information. In addition, if data is sent over the internet to another device for processing, it could be intercepted by third parties.
- **Frequent Updates:** Another challenge associated with edge computing is ensuring that all devices are properly updated with security patches. Because each device has its processor and memory, it can be difficult to push updates out to all devices promptly.

This can leave some devices vulnerable to attack even after patches have been released for other devices.

- Scalability: Finally, one of the biggest challenges facing edge computing is scalability. As more and more devices are connected to the internet and begin generating data, it becomes increasingly more work to manage and process all of that data centrally.
- Cloud Adoption Risk: Organizations must be aware of these risks when considering adopting cloud-based applications and services that use edge computing. They should consider implementing additional security controls to protect data and systems and ensure that their staff are trained on the proper handling of data at the edge. In addition, they should work with their cloud service providers to ensure that appropriate measures are in place to mitigate these risks.
- Edge and IoT security risks: The rise of the Internet of Things (IoT) has created a new category of devices when left unsecured become easy targets for hackers. Edge devices are increasingly being used to collect and process data and are particularly vulnerable. These devices are often located in remote or difficult-to-reach locations, making them difficult to secure and manage. In addition, they typically have limited processing power and storage, making them more susceptible to attacks.

How to secure Edge Computing?

As edge computing becomes more prevalent, it is essential to consider the security risks and challenges associated with it. Let's understand how to secure edge computing.

- **Keep your devices and data safe:** Protect your devices connected to the internet and keep them up-to-date with the latest security patches. In addition, be sure to encrypt your data at rest and in transit.
- **Secure your network connection:** Ensure that your network connection is secure using a VPN or other encryption methods. This will help to prevent eavesdropping and data theft.
- **Implement security measures at the edge device level:** Implementation of several security measures at the edge device level, such as firewalls, intrusion detection/prevention systems, and access control mechanisms. By implementing these measures, you can help to protect your data and devices from attack.
- **Protect against IoT security risks:** The best way to protect against these risks is to implement a comprehensive security strategy that includes physical and logical security measures. Physical security measures should be implemented to secure edge devices and prevent unauthorized access. Logical security measures, such as encryption and authentication, should be used to protect data collected by edge devices.
- **Monitor and log all edge activities:** As edge computing becomes more popular, securing the devices connected to the network is essential. One way to do this is to monitor and log all activity relating to operations and configuration. This will help to identify any potential security risks and allow you to take steps to mitigate them.

Advantages of Edge computing

1. Faster response time.
2. Security and Compliance.
3. Cost-effective Solution.
4. Reliable Operation with Intermittent Connectivity.
5. Reduced latency

Limitations of Edge Computing

1. **Complexity:** Setting up and maintaining edge computing systems can be challenging, especially if there are many devices or a vast geographic region involved.
2. **Limited resources:** Edge devices frequently have constrained processing, storage, and bandwidth, which can restrict their capacity to carry out specific activities.
3. **Dependence on connectivity:** In order for edge computing to work correctly, connectivity is required. If the connection is lost, the system may not be able to work.
4. **Security Concern:** Edge devices may be susceptible to security risks such malware, hacking, and physical interference.

Challenges in Edge computing

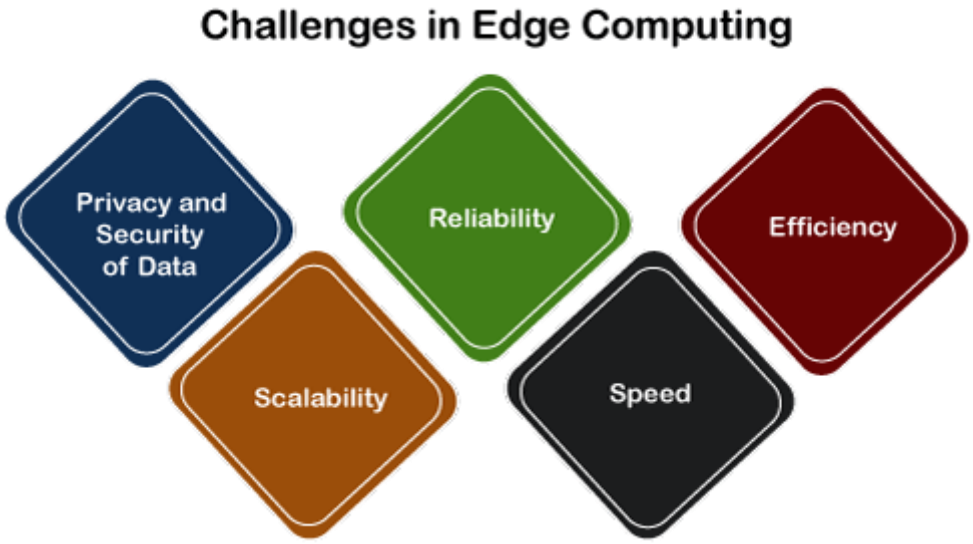


Fig. 4.2 Edge Computing Challenges

- **Privacy and Security of Data:** It is the new change and enhancement in technology, and so there should be change and enhancement in the privacy and security feature also. According to advanced security schemes of cloud computing, "different encryption mechanisms should be introduced because few encryption methods are being used for encrypting the data, but before reaching to the cloud, data transit may take place between different distributed nodes that are connected via the internet". Also, it needs a shift to decentralized infrastructure from the centralized, top-down model.

- **Scalability:** Edge Computing is based on a distributed network, and scalability becomes a challenge for such a distributed network-facing several issues. These issues are:
    1. **Heterogeneity of Devices:** A focus should be present on the heterogeneity of those devices that have its own different energy and performance constraints.
    2. Devices that have extensively dynamic condition and reliability of the connections in comparison to the robust infrastructure of the cloud data centers. In addition to this, the increase in the security requirements affects and slows down the scaling factor of edge computing as it may bring more latency between the nodes that communicate with each other.
- **Reliability:** Such a feature is a very challenging task for every technology and so for edge computing. For handling service from certain failovers, its management is required, which is very crucial. As edge computing relies on a distributed network, so if a single nodes fail or unable to reach, still the users must be able to avail the service without any disturbance. Also, the edge computing must be able to alert the user about the failure node and must provide actions to recover from the failure. For this, each device should maintain the network topology of the whole distributed system that will make the detection of errors and its recovery easy to process. Other than this, the connection technology in use may provide different reliability levels and data accuracy produced at the edge that could be unreliable because of the environmental conditions.
- **Speed:** Edge computing should be able to provide speed services to the end-users as it brings analytical, computational resources near to the source (end users), and it leads to fast communication. Such a modern system will automatically outperform the cloud computing system, which is a traditional system. So, maintenance of good speed is also a challenging task for edge computing.
- **Efficiency:** The efficiency of the edge computing becomes better because the availability of the analytical tools is too close to the end-users, and due to this, AI tools and analytical tools which are sophisticated can possibly execute on the edge of the system. Such a platform improves and increases operational efficiency and thus provides several benefits to the system.

Edge Computing Use-Cases

It has numerous varieties, with numerous IT experts seeing it as a development of the conveyed 'lights out' server farm idea. Regardless of how savvy the end-point is; all Edge approaches share similar engineering.

Center information center(s) with satellite areas store and cycle information and cooperate with end-focuses.

Edge comprises organization doors, server farms, and everything IoT.

The motivation behind the Edge is to convey dispersed application administrations, give knowledge to the end-point, speed up execution from the center data frameworks or gather and forward data from the Edge end-point sensors and regulators.

The shortfall of a concurred and acknowledged Edge processing definition requested we make our own subsequent in three distinct kinds of purpose cases −

- **Remote 'Lights Out' Edge Server,** farms can be a little hardware rack in different far-off areas or numerous enormous server farms. It is the most different, non-standard Edge climate. It requires new hierarchical models, modern programming application designs, and a high degree of reflection to the picture, conveying low touch control and the capacity to scale and deal with a heterogenous blend of gear.
- **Holder IT Edges,** is where combined frameworks reside. This climate comprises an answering stack including at least one of the accompanying; servers, operating system, stockpiling, organization, and improved power and cooling to help all the hardware in the contained climate. The compartments are exceptionally normalized notwithstanding, customization is accessible to suit explicit Edge prerequisites with choices for extra parts.
- **Internet of Things (IoT),** where profoundly accessible processors empower constant investigation for applications that can hardly hold on to decide. IoT end-directs go on toward getting more brilliant with a more remarkable capacity to work freely and settle on choices without routine correspondence with center stage.

B. Fog computing

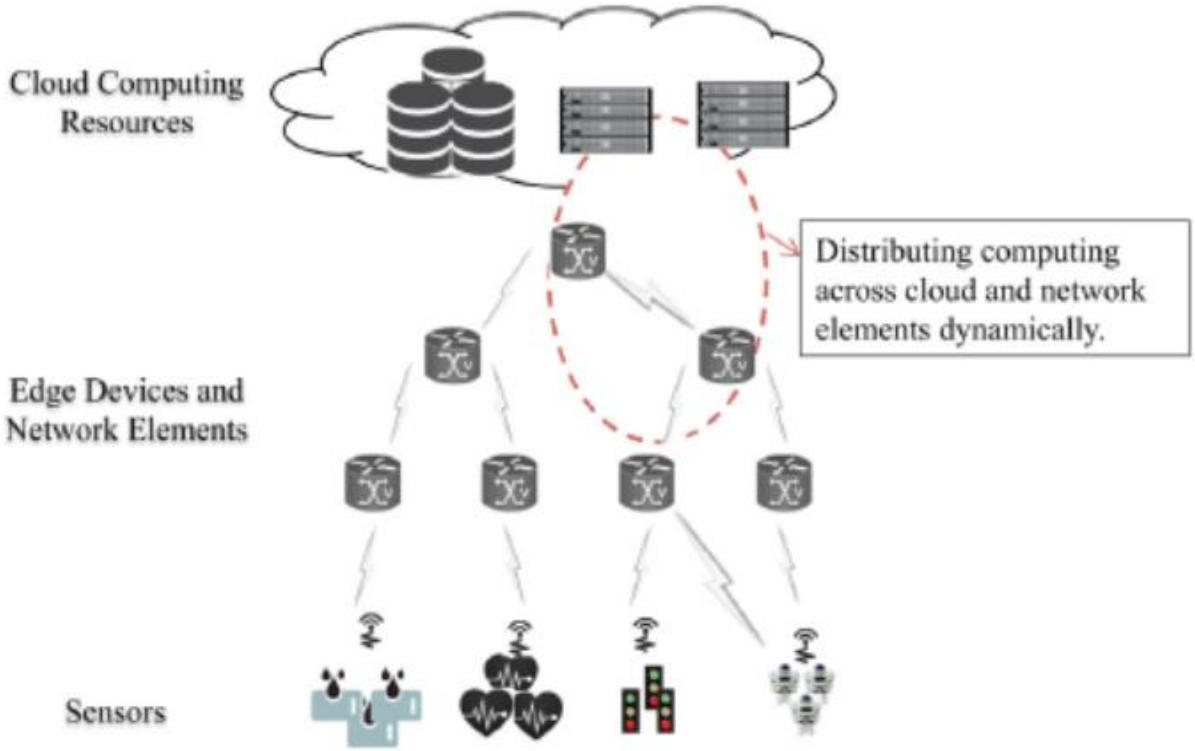


Fig. 4.3 Fog Computing is a Distributed Computing Paradigm That Extends the Cloud Services to the Edge of the Network

- Internet of Things (IoT) environments consist of loosely connected devices that are connected through heterogeneous networks. In general, the purpose of building such environments is to collect and process data from IoT devices in order to mine and detect patterns, or perform predictive analysis or optimization, and finally make smarter

decisions in a timely manner. Data in such environments can be classified into two categories:

• Little Data or Big Stream: transient data that is captured constantly from IoT smart devices

• Big Data: persistent data and knowledge that is stored and archived in centralized cloud storage.

- Processing of a large magnitude of data volume is enabled by on-demand scalability of Clouds. However, when data sources are distributed across multiple locations and low latency is indispensable, in-cloud data processing fails to meet the requirements.
- An alternative paradigm that is capable of bringing the computation to more computationally capable devices that are geographically closer to the sensors than to the clouds, and that have connectivity to the Internet. Such devices, which are at the *edge of the network* and therefore referred to as *edge devices,* can build local views of data flows and can aggregate data to be sent to the cloud for further offline analysis. To this end, Fog computing has emerged.
- Fog computing as a distributed computing paradigm that fundamentally extends the services provided by the cloud to the edge of the network. It facilitates management and programming of compute, networking, and storage services between data centers and end devices. Fog computing essentially involves components of an application running both in the cloud as well as in edge devices between sensors and the cloud in smart gateways, routers, or dedicated fog devices. Fog computing supports mobility, computing resources, communication protocols, interface heterogeneity, cloud integration, and distributed data analytics to address requirements of applications that need low latency with a wide and dense geographical distribution.

Architecture of Fog computing

Fog computing is emerging as an attractive solution to the problem of data processing in IoT. It relies on devices on the edge of the network that have more processing power than the end devices, and are nearer to these devices than the more powerful cloud resources, thus reducing latency for applications.

- Fig. 4.4 shows the reference architecture for fog computing. In the bottommost layer lays the end devices (sensors), as well as edge devices and gateways. This layer also includes apps that can be installed in the end devices to enhance their functionality. Elements from this layer use the next layer, the network, for communicating among themselves, and between them and the cloud. The next layer contains the cloud services and resources that support resource management and processing of IoT tasks that reach the cloud. On top of the cloud layer lays the resource management software that manages the whole infrastructure and enables quality of Service to Fog Computing applications. Finally, the topmost layer contains the applications that leverage fog computing to deliver innovative and intelligent applications to end users.

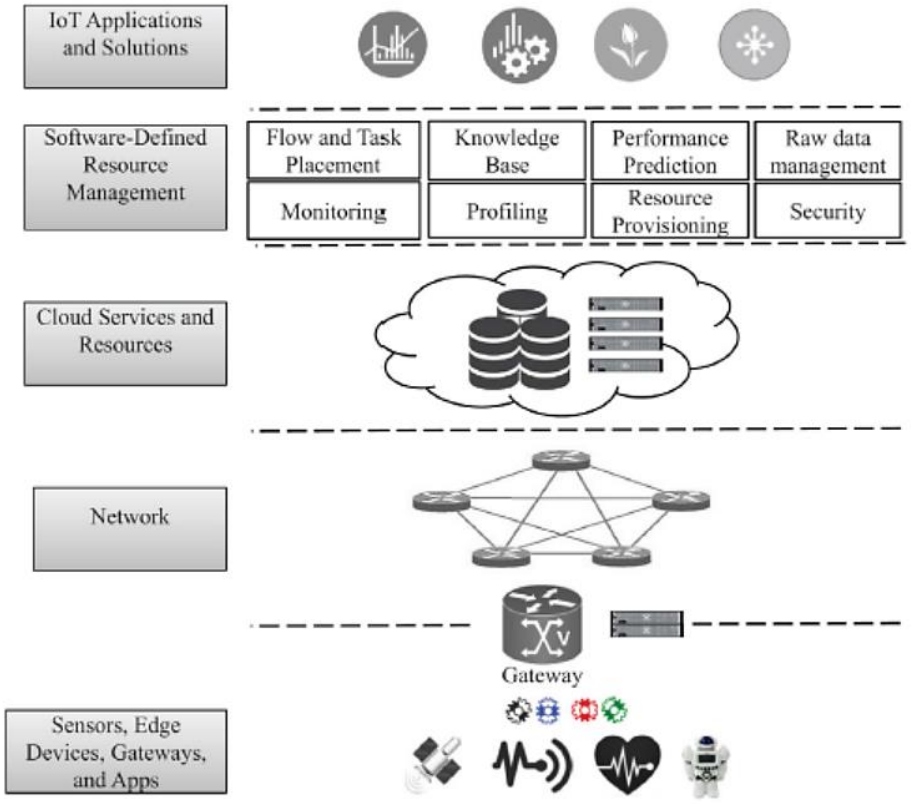


Fig. 4.4 Fog Computing Reference Architecture

- Software-Defined Resource Management layer implements many middleware like services to optimize the use of the cloud and Fog resources on behalf of the applications. The goal of these services is to reduce the cost of using the cloud at the same time that performance of applications reach acceptable levels of latency, by pushing task execution to Fog nodes. This is achieved with a number of services working together, as follows.
    - **Flow and task placement:** This component keeps track of the state of available cloud, Fog, and network resources (information provided by the Monitoring service) to identify the best candidates to hold incoming tasks and flows for execution. This component communicates with the Resource-Provisioning service to indicate the current number of flows and tasks, which may trigger new rounds of allocations if deemed too high.
    - **Knowledge Base:** This component stores historical information about application demands and resource demands that can be leveraged by other services to support their decision-making process.
    - **Performance Prediction:** This service utilizes information of the Knowledge-Base service to estimate the performance of available cloud resources. This information is used by the Resource- Provisioning service to decide the amount of resources to be provisioned, in times where there are a large number of tasks and flows in use or when performance is not satisfactory.
    - **Raw-Data Management:** This service has direct access to the data sources and provides views from the data for other services. Sometimes, these views can be

obtained by simple querying (eg, SQL or NOSQL REST APIs), whereas other times more complex processing may be required (eg, MapReduce). Nevertheless, the particular method for generation of the view is abstracted away from other services.

- **Monitoring:** This service keeps track of the performance and status of applications and services, and supplies this information to other services as required.
- **Profiling:** This service builds resource- and application-profiles based on information obtained from the Knowledge Base and Monitoring services.
- **Resource Provisioning:** This service is responsible for acquiring cloud, Fog, and network resources for hosting the applications. This allocation is dynamic, as the requirements of applications, as well as the number of hosted applications, changes over time. The decision on the number of resources is made with the use of information provided by other services (such as Profiling, Performance Prediction, and Monitoring), and user requirements on latency, as well as credentials managed by the Security service. For example, the component pushes tasks with low-latency requirements to edge of network as soon as free resources are available.
- **Security:** This service supplies authentication, authorization, and cryptography, as required by services and applications.

Security Risks and its measures:

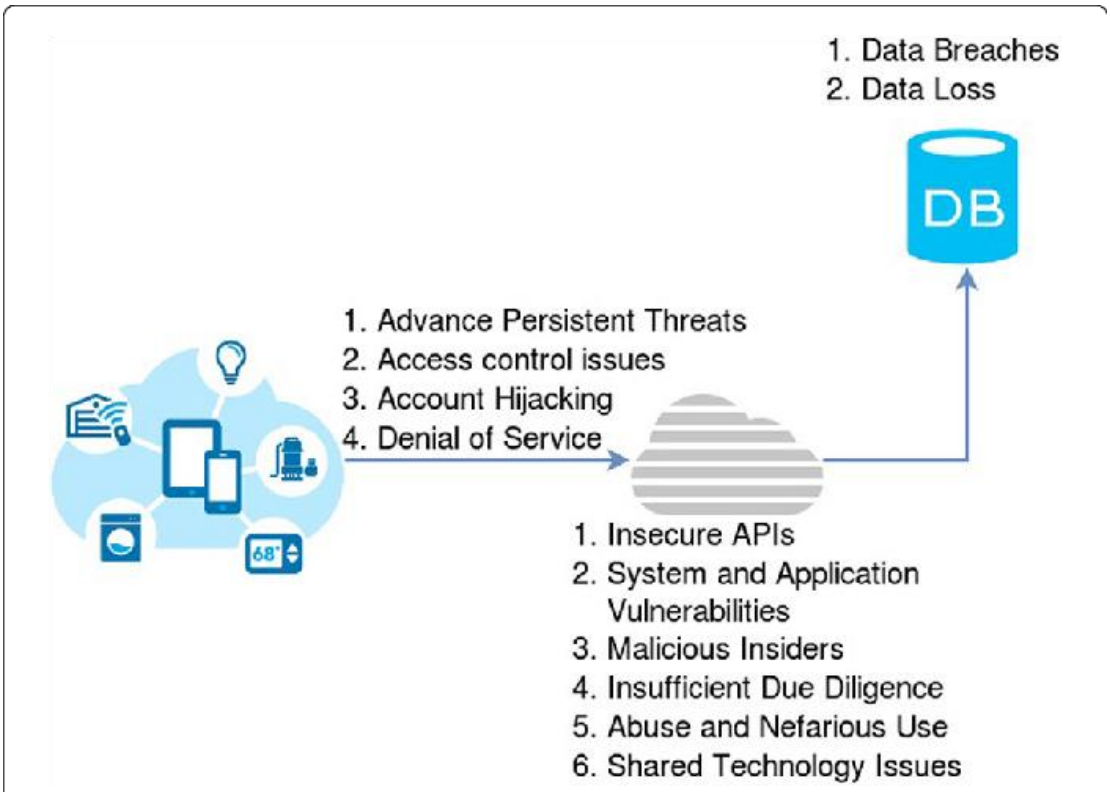


Fig.4.5 Potential security issues of Fog Platform inherited from Cloud computing

1. Security Risk-01: Increased attack surface due to the distributed nature of the technology.

Solution: Organizations must implement robust security measures at each level of the fog computing architecture. This includes securing the individual devices and systems that make up the fog network, as well as the connections between them. One potential solution is to use encryption and authentication protocols to ensure that only authorized devices can access the network and that data transmitted between devices is protected from eavesdropping or tampering.

Attack category	Possible threats	Possible solutions	Impact
Virtualization issues	Hypervisor attacks VM-based attacks Weak or no Logical Segregation Side channel attacks Privilege Escalation Service abuse Privilege escalation attacks Inefficient resource policies	Multi-factor Authentication Intrusion Detection System User data isolation Attribute/identity based encryption Role-Based Access Control model User-based permissions model Process isolation	As all services and VMs are executing in a virtualized environment, its compromise will have adverse effect on all Fog services, data and users
Web security issues	SQL injection Cross-site scripting Cross-site request forgery Session/Account hijacking Insecure direct object references Malicious redirections Drive-by attacks	Secure code Find and patch vulnerabilities Regular software updates Periodic auditing Firewall Anti-virus protection Intrusion Prevention System	Exposure of sensitive information, attacker can become legitimate part of network, and enable malicious applications to install
Internal/external communication issues	Man-in-the-Middle attack Inefficient rules/policies Poor access control Session/Account hijacking Insecure APIs and services Application vulnerabilities Single-point of failure	Encrypted communication Mutual/Multi-factor authentication Partial encryption Isolating compromised nodes Certificate pinning Limiting number of connections Transport layer security (TLS)	Attacker can acquire sensitive information by eavesdropping and get access to unauthorized Fog resources
Data security related issues	Data replication and sharing Data altering and erasing attacks Illegal data access Data ownership issues Low attack tolerance Malicious Insiders Multi-tenancy issues Denial of Service attacks	Policy enforcement Security inside design architecture Encryption Secure key management Obfuscation Data Masking Data classification Network monitoring	High probability of illegal file and database access, where attacker can compromise both user and Fog system's data
Wireless security issues	Active impersonation Message replay attacks Message distortion issues Data loss Data breach Sniffing attacks Illegal resource consumption	Authentication Encrypted communication Key management service Secure routing Private network Wireless security protocols	Vulnerable wireless access points can compromise communication privacy, consistency, accuracy, availability and trustworthiness
Malware protection	Virus Trojans Worms Ransomware Spyware Rootkits Performance reduction	Anti-malware programs Intrusion Detection System Rigorous data backups Patching vulnerabilities System restore points	Malware infected nodes will lower the performance of the entire Fog platform, allow back-doors to the system and corrupt/damage data permanently

Fig.4.6 Summary of potential security issues found in Fog applications

2. **Security Risk-02:** Potential for data leakage or loss –As the data is processed and stored closer to the source, there is an increased risk of unauthorized access or accidental exposure. This is particularly concerning for organizations that handle sensitive information, such as financial data or personal information.

Solution: organizations should implement strong access controls and data protection measures, such as encryption and data masking. Additionally, regular audits and monitoring can help identify potential vulnerabilities and ensure that security measures are functioning as intended.

3. **Security Risk-03:** Privacy, as the technology often involves the collection and processing of personal data.

Solution: Organizations should implement strict data handling policies and ensure that any personal data collected is anonymized or aggregated to protect individual privacy. Additionally, organizations should be transparent about their data collection practices and provide users with the ability to opt-out of data collection if desired.

4. **Security Risk-04:** Fog computing introduces new challenges in terms of compliance with data protection regulations such as the General Data Protection Regulation (GDPR) in the European Union.

Solution: Organizations should work closely with legal and compliance experts to develop a comprehensive understanding of the applicable regulations and implement appropriate measures to ensure adherence. This may include implementing data protection by design principles, conducting regular risk assessments, and maintaining detailed records of data processing activities.

While fog computing offers significant benefits in terms of efficiency and performance, it also introduces new security challenges that must be addressed to ensure the safety and integrity of data and systems. By implementing robust security measures, such as encryption, authentication, and access controls, organizations can mitigate these risks and reap the benefits of this innovative technology. Additionally, by being proactive in addressing privacy concerns and compliance with data protection regulations, organizations can build trust with users and stakeholders, further enhancing the value of fog computing.

RESEARCH DIRECTIONS AND ENABLERS:

To realize the full potential of the Fog paradigm, researchers and practitioners need to address the following major challenges,

- **PROGRAMMING MODELS:** Computation offloading has been an active area of research in the mobile computing domain, with most of the proposals offloading workloads to the cloud. Since offloading to the cloud may not always be possible or reasonable, an adaptive Mobile Edge Computing (MEC) programming framework named CloudAware is proposed recently, which offloads tasks to edge devices, thus facilitating the development of elastic and scalable edge-based mobile applications. The types of components that an MEC application should be broken into, so that the offloading decision is simplified. The framework offloads tasks, with objectives to: (1) speed up computation, (2) save energy, (3) save bandwidth, or (4) provide low latency.
- **SECURITY AND RELIABILITY:** Enforcing security protocols over a distributed system such as a fog is one of the most important challenges in its realization. Calling authentication at various levels of fog nodes is the main security challenge.

Authentication solutions based on Public Key Infrastructure may prove beneficial for this problem. Trusted execution environment (TEE) techniques are potential solutions to this authentication problem in fog computing as well. Measurement-based methods may also be used to detect rogue devices and hence reduce authentication cost. The research challenges in policy management for fog computing, and a policy-driven security-management approach, including policy analysis and its integration with a fog-computing paradigm. Such an approach is critical for supporting secure sharing and data reuse in heterogeneous Fog environments. Since fog computing is realized by the integration of a large number of geographically distributed devices and connections, reliability is one of the prime concerns when designing such a system. For a reliable fog paradigm it is essential to plan for failure of individual sensors, network, service platform, and the application. To this end, the current reliability protocols for WSNs can be applied. They majorly deal with packet reliability and event reliability. The most basic facts about sensors, in general, are not that they are expensive, but that their readings can be affected by noise; in this case the information accuracy problem can be resolved by redundancy.

- **RESOURCE MANAGEMENT:** Fog devices are often network devices equipped with additional storage and compute power. However, it is difficult for such devices to match the resource capacity of traditional servers, let alone the cloud. Hence a judicious management of resources is essential for an efficient operation of a fog-computing environment. Tactical cloudlet refers to the scenario when cloudlets serve as fog devices in order to provide infrastructure to offload computation, provide forward data-staging for a mission, perform data filtering to remove unnecessary data from streams intended for dismounted users, and serve as collection points for data heading for enterprise repositories. Tasks running on cloudlets are executed on Service virtual-machine. The various policies for provisioning virtual-machine on cloudlets, each policy having a unique implication on payload sent to cloudlet, application-ready time, and client energy spent. In addition, mechanisms for cloudlet discovery and application execution have also been laid out.
- **ENERGY MINIMIZATION:** Since fog environments involve the deployment of a large number of fog nodes, the computation is essentially distributed and can be less energy-efficient than the centralized-cloud-model of computation. Hence the reduction of energy consumption in fog computing is an important challenge. The trade-off between power consumption and delay in a fog-computing system can be solved by modelling the power consumption and delay functions for the fog system and formalize the problem of allocating workloads between the fog and cloud.

Advantages of Fog computing

- **Reduction of network traffic:** Fog computing provides a platform for filter and analysis of the data generated by these devices close to the edge, and for generation of local data views. This drastically reduces the traffic being sent to the cloud.

- **Suitable for IoT tasks and queries:** With the increasing number of smart devices, most of the requests pertain to the surroundings of the device. Hence, such requests can be served without the help of the global information present at the cloud.
- **Low-latency requirement**
- **Scalability:** Even with virtually infinite resources, the cloud may become the bottleneck if all the raw data generated by end devices is continually sent to it. Since fog computing aims at processing incoming data closer to the data source itself, it reduces the burden of that processing on the cloud, thus addressing the scalability issues arising out of the increasing number of endpoints.

Limitations of Fog Computing:

Edge and fog computing addresses three principal network limitations: bandwidth, latency and congestion or reliability.

- **Bandwidth.** Bandwidth is the amount of data which a network can carry over time, usually expressed in bits per second. All networks have a limited bandwidth, and the limits are more severe for wireless communication. This means that there is a finite limit to the amount of data -- or the number of devices -- that can communicate data across the network. Although it's possible to increase network bandwidth to accommodate more devices and data, the cost can be significant, there are still (higher) finite limits and it doesn't solve other problems.
- **Latency.** Latency is the time needed to send data between two points on a network. Although communication ideally takes place at the speed of light, large physical distances coupled with network congestion or outages can delay data movement across the network. This delays any analytics and decision-making processes, and reduces the ability for a system to respond in real time. It even cost lives in the autonomous vehicle example.
- **Congestion.** The internet is basically a global "network of networks." Although it has evolved to offer good general-purpose data exchanges for most everyday computing tasks such as file exchanges or basic streaming the volume of data involved with tens of billions of devices can overwhelm the internet, causing high levels of congestion and forcing time-consuming data retransmissions. In other cases, network outages can exacerbate congestion and even sever communication to some internet users entirely - making the internet of things useless during outages.

Applications of Fog computing

- **Health Care:** Smart sensor-based healthcare infrastructure
- **Augmented Reality:** Augmented Brain Computer Interaction Game based on Fog Computing and Linked Data. When a person plays the game, raw streams of data collected by EEG sensors are generated and classified to detect the brain state of the player. Brain-state classification is among the most computationally heavy signal-processing tasks, but this needs to be carried out in real time. The system employs both fog and cloud servers, a combination that enables the system to perform continuous

real-time brain-state classification at the fog servers, while the classification models are tuned regularly in the cloud servers, based on the EEG readings collected by the sensors. Another example is the wearable Cognitive Assistance system based on Google Glass devices that assist people with reduced mental acuity.

- **Caching and Preprocessing:** Users connect to the Internet through fog boxes, hence each HTTP request made by a user goes through a fog device. The fog device performs a number of optimizations that reduces the amount of time the user has to wait for the requested webpage to load. Apart from generic optimizations like caching HTML components, reorganizing webpage composition, and reducing the size of web objects, edge devices also perform optimizations that take user behavior and network conditions into account. For example, in case of network congestion, the edge device may provide low resolution graphics to the user in order to reach acceptable response times. Furthermore, the edge device can also monitor the performance of the client machines, and, depending on the browser rendering times, send graphics of an appropriate resolution.

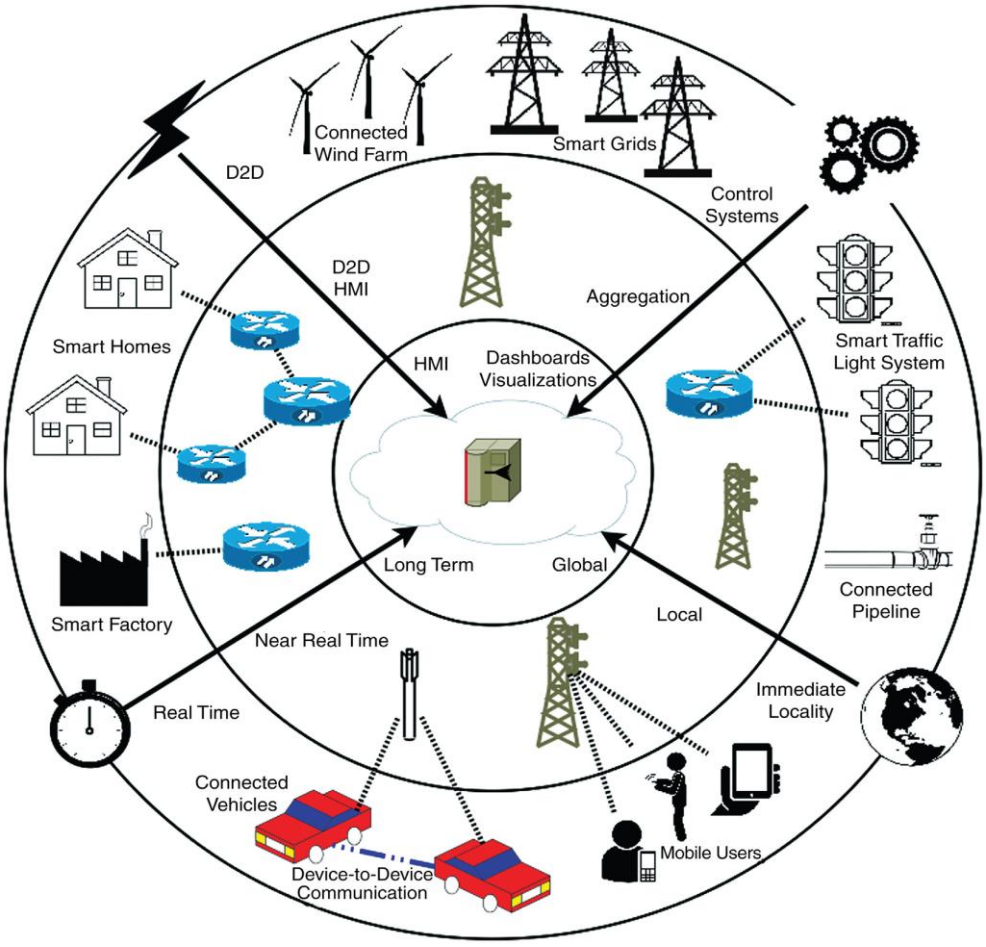


Fig.4.7 Applications of Fog Computing

Edge vs. Fog Computing

Edge Computing	Cloud Computing
It is good to be used for those organizations that have a limited budget to invest in financial resources. So, mid-level organizations can use edge computing.	It is generally recommended for processing and managing a high volume of data that is complex and massive enough. Thus, such organizations that deal with huge data storage use cloud computing.
It can use different programming languages on different platforms, each having different runtime.	Cloud Computing works for one target platform using one programming language only.
Security in edge computing needs tight and robust plans such as advanced authentication methods, network security, etc.	It does not need high and advanced security methods.
It processes time-sensitive data.	It process that data that is not driven by time, i.e., not time-driven.
It processes data at remote locations and uses the Decentralization approach.	It processes and deals with data at centralized locations by using a centralized approach.
Organizations can indulge in edge computing with the existing IoT devices, advance them, and use them. There is no need to purchase new devices.	For advancement, existing IoT devices need to be exchanged with the new ones that will cost more money and time.
Edge Computing is the upcoming future.	Cloud Computing is the currently existing technology.

Charactersitics between IOT, Edge, Fog Computing

Characteristic	IoT	Edge	Fog	Cloud
Deployment	Distributed	Distributed	Distributed	Centralised
Components	Physical devices	Edge nodes	Fog nodes	Virtual resources
Location awareness	Aware	Aware	Aware	Not aware
Computational	Limited	Limited	Limited	Unlimited
Storage	Very limited	Limited	Limited	Unlimited
Data	Source	Process	Process	Process
Distance to data source	The source	The nearest	Near	Far
Response time	No response time	The fastest	Fast	Slow
Nodes count	The largest	Very large	Large	Small

Case Study

A smart city is one of the key use-cases of IoT, which in itself is a combination of a variety of use cases, ranging from smart traffic management to energy management of buildings. In this section, we present a case study on smart traffic management and show that employing fog computing improves the performance of the application in terms of response time and bandwidth consumption. A smart traffic management system can be realized by a set of stream queries executing on data generated by sensors deployed throughout the city. Typical examples of such queries are real-time calculation of congestion (for route planning), and detection of traffic incidents. In this case study, we compare the performance of a query DETECT_TRAFFIC_INCIDENT on fog infrastructure versus the typical cloud implementation.

In the query, the sensors deployed on the roads send the speed of each crossing vehicle to the query processing engine. The operator "Average Speed Calculation" calculates the average speed of vehicles from the sensor readings over a given timeframe and sends this information to the next operator. The operator "Congestion Calculation" calculates the level of congestion in each lane based on the average speed of vehicles in that lane. The operator "Incident Detection," based on the average level of congestion, detects whether or not an incident has occurred. This query was simulated on both a fog-based as well as a cloud-based stream query-processing engine.

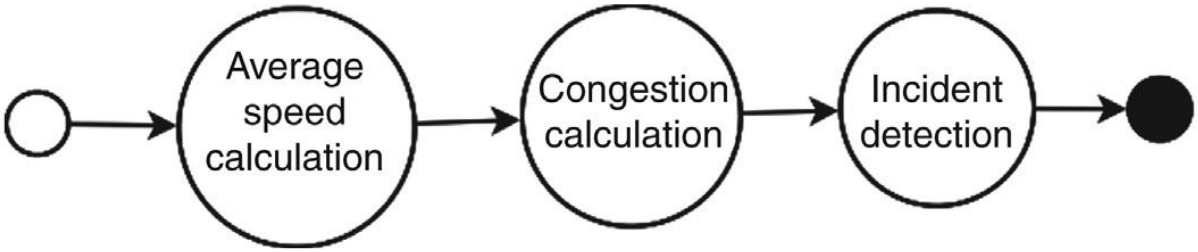


Fig.4.8 Dag of Query for Incident Detection

The network topology used for the simulation was a hierarchical topology of fog devices. The leaves of the treelike topology are the edge devices (gateways) and the cloud is located at the root. Intermediate nodes in the tree represent intermediate network devices between the cloud and the edge, which are able to host applications by utilizing their nascent compute, network, and storage capacity. Each fog device has both an associated CPU capacity and an associated uplink network bandwidth, which shall be utilized for running fog applications on them. Traffic data fed to simulation i.e.a road-traffic simulator. Induction loops were inserted on the road that measured the speed of vehicles, and the information was sent to the query processing engine.

The simulation environment was implemented in CloudSim by extending the basic entities in the original simulator. Fog devices were realized by extending the Datacenter class, whereas stream operators were modeled as a VM in CloudSim. Furthermore, tuples that were executed by the stream operators were realized by extending Cloudlets. Fog devices are entities with only one host, whose resources it can provide for running applications. Each tuple has an associated CPU and network cost for processing it.

PERFORMANCE EVALUATION:

Average Tuple Delay: Average tuple delay is the amount of time (on average) that a tuple takes to be processed. Once operators are placed on fog devices the end-to-end tuple delay falls much below the delay of in-cloud processing, as data are processed closer to the sources. However, it is worth mentioning that if the operators are not placed optimally, resource contention in edge devices can cause more delay.
Core Network Usage: Considerably lower number of tuples traversing the core network once, compared to the traditional cloud-based stream processing. Thus, running the query on the edge devices reduces the workload coming to the cloud for processing and also reduces the network usages considerably. However, this reduction in network resource usage and end-to-end latency is only possible if a placement algorithm is in position to push operators downward when enough capacity is available in edge devices.

2. Front end Edge Devices and Gateway

Edge Computing Layers: The three fundamental layers involved in an edge computing setup consist of sensors or edge devices, edge gateway, and the cloud or central server. As the data travels through these layers, each layer also enables decision-making.

- **Sensors:** Smart sensors or edge devices have an embedded microprocessor that collects vital measurements from a variety of sensors it is connected with. These devices can collect data measurements such as time-stamp, hours of operation, connectivity, calibration conformance and a host of other micro-operations. It can even operate autonomously as long as it has power with the ability to sync up data if connectivity was lost, providing continuous data assurance. The Smart Sensors can even provide local control outputs in various forms for alarms to actuation.

- **Edge Gateway:** An edge gateway sits between the edge devices and the cloud. It is the central repository for the edge devices data as well as synchronizing with another edge gateway. The edge gateway is also the gatekeeper for all the edge devices connected to it granting them secure authentication and provisioning. Only higher-order data processing is transmitted to the cloud for modeling and analytics. Edge gateways are set up to run independently of the cloud while providing many of the benefits of the cloud. More than one edge gateway can be deployed in a large factory setting, each working on specific data metrics that can ultimately be synchronized and unified at the cloud. Then the heavy data crunching can be computed without impacting any local edge devices and gateways.

- **Cloud:** Cloud in an interconnected network of virtual servers and web services hosted on the internet. It is where the higher-order data from the edge gateways get stored, processed and analyzed.

Edge Devices

- An edge device is any piece of hardware that controls data flow at the boundary between two networks. Edge devices are comprised of hardware that performs the two essential functions of providing physical connectivity and enabling traffic between networks. A full range of edge device functions might include the transmission, routing, processing, monitoring, filtering, translation and storage of data between networks.
- Some common functions of edge devices are the transmission, routing, processing, monitoring, filtering, translation and storage of data passing between networks. Edge devices are used by enterprises and service providers. For networks that use different protocols, edge devices also provide traffic translation in addition to connectivity.
- Traditional edge devices include edge routers, routing switches, firewalls, multiplexers, and other wide area network (WAN) devices.
- Intelligent edge devices have built-in processors with onboard analytics or artificial intelligence capabilities. Such devices might include sensors, actuators, and IoT gateways. With the ability to process certain amounts of data directly on intelligent edge devices, rather than uploading, processing and storing data on the cloud, companies can create efficiencies and reduce costs.
- While edge devices are primarily used by enterprises and service providers to connect an internal local area network (LAN) to the Internet or an external wide area network (WAN), edge devices and edge computing have evolved to provide more advanced capabilities. Such capabilities might include wireless access points (APs), security through wireless APs or virtual private network (VPN) servers, dynamic host configuration protocol (DHCP) services, and domain name system (DNS) services.
- While edge devices provide speed, cost savings, and expanded access to networks and resources, there are several considerations to be made when employing this technology. Some challenges with edge devices include the responsibility of registering and managing various edge devices, ensuring data security, and preventing bottlenecks in network traffic.
- Cloud computing and the internet of things (IoT) have elevated the role of edge devices, ushering in the need for more intelligence, computing power and advanced services at the network edge. This concept, where processes are decentralized and occur in a more logical physical location, is referred to as edge computing.

Edge Devices Use case

Although the primary function of edge devices is to provide connectivity between disparate networks, the edge has evolved to increasingly support advanced services. These might include the following:

- **Wireless capabilities.** Wireless access points (APs) act as edge devices since they typically provide wireless clients with access to the wired network.

- **Security functions.** Edge devices, such as wireless APs or virtual private network (VPN) servers, commonly include integrated security capabilities designed to block a malicious user or device connection.
- **Dynamic Host Configuration Protocol (DHCP) services.** DHCP is a service that is commonly used in conjunction with edge devices, such as wireless APs and VPNs. When a client connects to such edge devices, the client requires an Internet Protocol (IP) address that is local to the network that it's accessing. The DHCP server provides the required IP address. In many cases, DHCP services are integrated into edge devices.
- **Domain name system (DNS) services.** When an external client accesses a network through an edge device, the client must be able to resolve fully qualified domain names on the network. When the DNS services lease an IP address to the client, it typically also points the client to a DNS server that provides name resolution services for the network.

Cloud computing and IoT have demonstrated the value for pushing intelligence to the periphery of the network. If an enterprise has thousands of IoT devices, it's not efficient for them all to try to communicate with the same resource at once. Edge devices can collect, process and store data in a more distributed fashion and closer to endpoints -- hastening response times, reducing latency and conserving network resources.

Edge Devices

- **Edge Routers:** Edge routers connect multiple packet transmission networks. They manage data traffic to an IP address so multiple devices can operate using a single internet connection. For a Local Area Network (LAN), a router is responsible for several devices within a specific geography.
- **WAN Devices:** Wide Area Network (WAN) devices extend over large areas and can even be global. WAN devices allow organizations to work seamlessly across the enterprise regardless of location.
- **Routing Switches:** Routing switches are used in situations like industrial IoT. They can perform many of the same functions as a router but also allow connections across different devices.
- **Firewalls:** Firewalls monitor all network traffic for security. They contain rules-based functionality and can be configured to block data traffic deemed harmful by the rule set.
- **Multiplexers:** Multiplexers merge data from multiple data sources over a single signal. They're integrated access devices (IAD) that help automation systems and advanced IoT networks perform efficiently.

Intelligent Edge Devices

Intelligent edge devices are used in advanced IoT and industrial IoT to connect the entire system's onboard processing or analytics capabilities. This functionality enables smart factories and IIoT platforms to facilitate advanced automation and analytics.

Intelligent edge devices include:

- **Sensors:** Sensors measure a condition or event, trigger action, and route data to the next destination. Sensor types are almost limitless and include GPS, motion, optical, temperature, humidity, vibration, and more.
- **Actuators:** Actuators act as the physical connectivity bridge between electronic devices like sensors and the physical movement required on a machine. Actuators take the sensor signals and instructions from intelligent edge devices and trigger actions using electric, air, or hydraulic power.
- **IoT Gateways:** IoT gateways connect multiple sensors and other devices to cloud computing platforms for analytics, computing, processing, and storage.
- **M2M Devices:** M2M devices connect equipment or machines to transfer data and facilitate automation. M2M stands for **Machine to Machine** communication. It is a direct communication system between the devices using wired or wireless communications channels without any human interaction. It collects the data and shares it with other connected devices. It is a technology that allows devices without the use of the internet to connect between devices. Various applications, such as defense, monitoring and tracking, production and facility management, are provided by M2M communications.

How an Edge Device work?

- An edge device is essentially just a bridge between two networks which works by being near the source of the data it manages. Devices like embedded or added sensors act as the device layer, acquiring data and sending it to the edge computing device or a cloud computing platform.
- If the data is informational or transactional, the device may send it directly to the cloud for storage. Or, based on parameters, it may send it to an edge device for processing and instructions and then forward the instructions to an actuator to trigger a response. Devices in M2M systems can be linked to perform tasks sent from an edge computing device or an edge network.
- In large connected systems, data flow may be controlled by a router. Using extended access to devices within a system, a combination of sensors, actuators, routers, switches, and edge computing devices can be controlled and accessed locally or over WAN to provide visibility and the capability to act over long distances.
- These can be two on-premises networks, but an edge device can also be used for cloud connectivity. The important thing to keep in mind is that the two networks are otherwise not connected to one another and might have major architectural differences. For example, at one time, it was common for organizations to use Systems Network Architecture (SNA) networks for 3270 communications in mainframe environments. As personal computers (PCs) and other devices became more prevalent, however, edge devices were used to tie Ethernet networks -- or other network types -- to existing networks.

- Regardless of the use case, there are two basic things that an edge device must do. First, the edge device must provide physical connectivity to both networks. The second thing that it must do is allow traffic to traverse the two networks when necessary. Depending on the nature of the edge device, this might mean simply forwarding an IP packet. In the case of architecturally dissimilar networks, however, the edge device might need to perform protocol translation.

Benefits of Edge Devices

- Improved Performance: In an extensive centralized system, data flow may be constrained by too much traffic across a narrow bandwidth. The vast volume of data in systems like an advanced machine data platform can overwhelm resources. Edge computing and edge computing infrastructure create a local edge layer where a great deal of functionality for automation and response can be conducted near the source, reducing latency and bandwidth constraints. Built-in processors and onboard analytics do much of the work that would've been sent to the cloud. Advanced capabilities allow network entry across traditional cables, cellular, or wireless access points at the network edge.
- Empowering AI and ML: Edge computing creates a decentralized edge network so that advanced cloud-based artificial intelligence (AI) and machine learning (ML) platforms can focus on the volume of data requiring the most complex analysis and insight. Edge computing extends the reach of AI and ML to the edge near the data source, making the entire system more efficient.
- Better Regulatory and Compliance Performance: Many industries, such as medical, pharmaceutical, defense, and aerospace, have tight regulation and compliance requirements. Edge computing facilitates the quick and efficient update and analysis of data at the edge so users have the most relevant insights. This functionality may be as simple as sensors that perform precise measurements for pharmaceuticals. They can also be tied to expiry parameters and fully traceable supply chains. Or, they can be tasked to identify quality and strength requirements for military and aerospace parts.
- Greater Interoperability: Many providers offer plug-and-play devices that enable interoperable legacy equipment and software. It's possible to create a system where analog and digital assets communicate in a single platform with cloud-based analytics and advanced edge networks.

Use cases of Edge devices:

M2M Devices: Machine to machine (M2M) is a comprehensive concept that can be used to describe any technology that allows networked devices to exchange information and inform actions automatically without human interference. M2M communication devices system is standalone network equipment that uses point-to-point communication between machines, sensors and hardware. Today, M2M is among the fastest-growing technologies because it can connect millions of devices within a single network.

Working of M2M:

The purpose of its technology is to use sensors to acquire data and transmit it to the network. Unlike SCADA or other remote monitoring tools, M2M systems typically use public networks and access methods, making them more cost-effective.

The M2M system includes hardware and software, and each hardware device communicates data through the network. In addition, the M2M system includes a server host and an intelligent expert system, and the data transmitted between devices can be processed through the server host expert system.

- A device or group of devices capable of replying to requests for data contained within those devices or capable of transmitting data contained within those devices autonomously.
- A communications link to connect the device or group of devices to a computer server or another device.
- A software agent, process, or interface by which the data can be analyzed, reported, and/or acted upon.
- Software Intelligence.

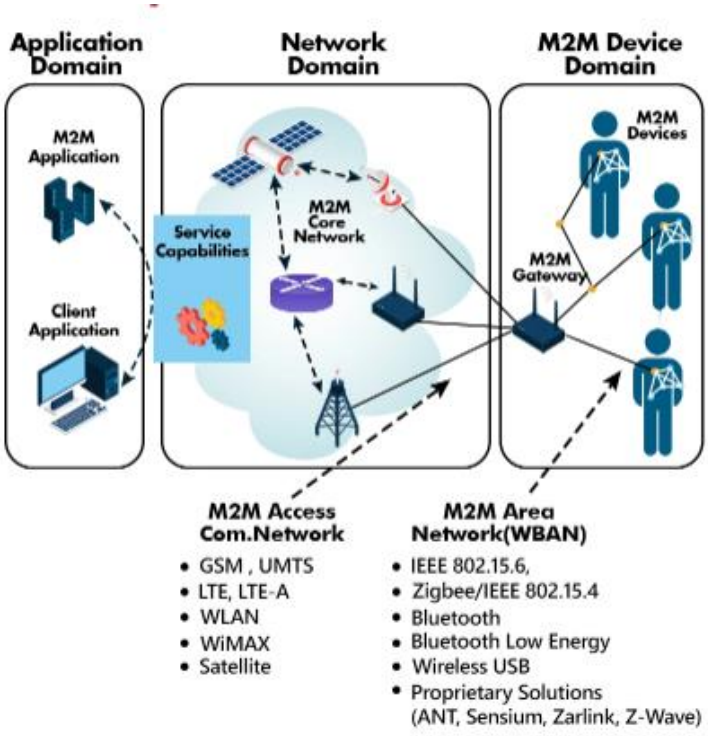


Fig.4.10 M2M Architecture

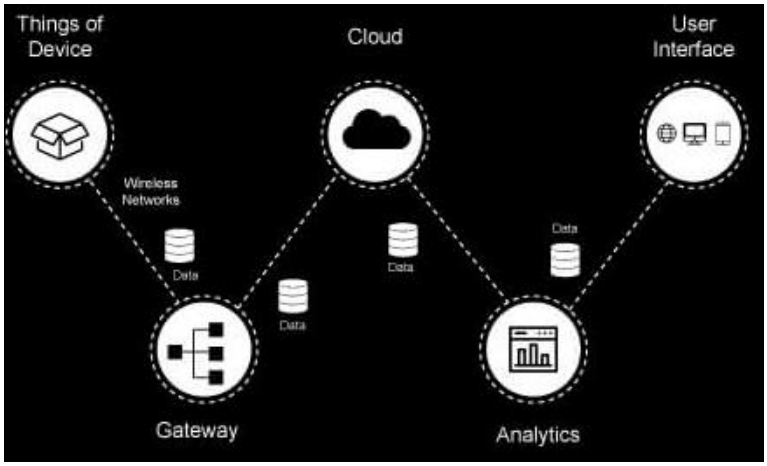


Fig.4.11 M2M communication

- The main components of an M2M system include sensors, RFID, Wi-Fi, or cellular communication links, etc., and autonomous software programmed to help network devices interpret data and make decisions. These M2M applications transform the data and then trigger pre-programmed automated operations.
- M2M technology's purpose is to collect the data from sensors and transmit them to the cloud. It uses a mobile network for transmission purposes, which makes it a cost-effective technology. The working of M2M is that the data is collected from the device sensors, sent to the cloud via mobile towers.
- The interaction between systems is carried out by artificial intelligence and machine learning, enabling them to make their autonomous decisions. It includes a wireless network in the industrial IoT, and since it is wireless, more applications can be connected.
- M2M virtually enables any sensor to communicate, and the systems monitor themselves and automatically respond. Any two or more machines wired or wirelessly can communicate with one another. The data is stored and translated to help in automatic actions. A
- vending machine can alert thM2M can be used to capture and transmit data according to specific applications through multiple wireless technologies (LTE, WiFi, BLE, ZigBee, RFID). Often more than one wireless technology is used in each application or solution, and different wireless technologies are used for data capture and data transfer. Through these tools, you can connect mobile or stationary objects, such as trains, household appliances, medical equipment, displays, etc., and complete related tasks, including sensing or data capture, wireless data transport, and information management systems.
- The advantages of M2M are that it can change the original market model by reducing equipment maintenance and downtime to reduce costs and by proactively monitoring and repairing equipment before it breaks down or only when needed, thereby improving customer service, etc.e distributor's network when a particular item runs out of stock and sends a refill.

Key Features of M2M Communication

- M2M consumes lower power as compared to its rival technologies.
- It acts as a network operator and provides packet-switched services
- It is capable of detecting events with the help of its monitoring abilities
- This allows for a delay in the transfer of data
- It allows for the specification to time to send and receive data
- It sends an alert call or wake up call to awaken devices when the device enters an unknown premises
- This continuously receives and sends minimal amounts of data

XMPP:

- XMPP (Extensible Messaging and Presence Protocol) is often used for data package exchanges in real time. Its specifications are described in the RFC 3920 and RFC 3921 standards. To supporting the XML format, it is perfect for use in IoT solutions. Networks based on the XMPP protocol are decentralized to ensure their high fault tolerance.
- For security, XMPP networks often employ the help of the SASL and TLS protocols. Besides this, some XMPP-based IoT solutions use PGP/GPG for encryption purposes.
- XMPP can be used as the basis for extremely flexible solutions. The XMPP Software Foundation supports numerous extensions, including ones dedicated to distributing resources, working with IoT devices and monitoring their state, etc.
- It is an open set of rules for streaming XML elements in order to swap messages and presence information in close to real-time. The XMPP protocol is based on the typical client server architecture, in which the XMPP client uses the XMPP server with the TCP socket.
- XMPP provides a general framework for messaging across a network, offering a multitude of applications beyond traditional instant messaging (IM) and the distribution of presence data. It enables the discovery of services residing locally or across a network, as well as finding out about the availability of these services.
- XMPP is well-matched for cloud computing where virtual machines, networks and firewalls would otherwise present obstacles to alternative service discovery and presence-based solutions. Cloud computing and storage systems rely on diverse forms of communication over multiple levels, including not only messaging between systems to relay state but also the migration of the distribution of larger objects, like storage or virtual machines. Along with validation and in-transit data protection, it can be useful at many levels and may prove ideal as an extensible middleware or a message-oriented middleware (MOM) protocol.

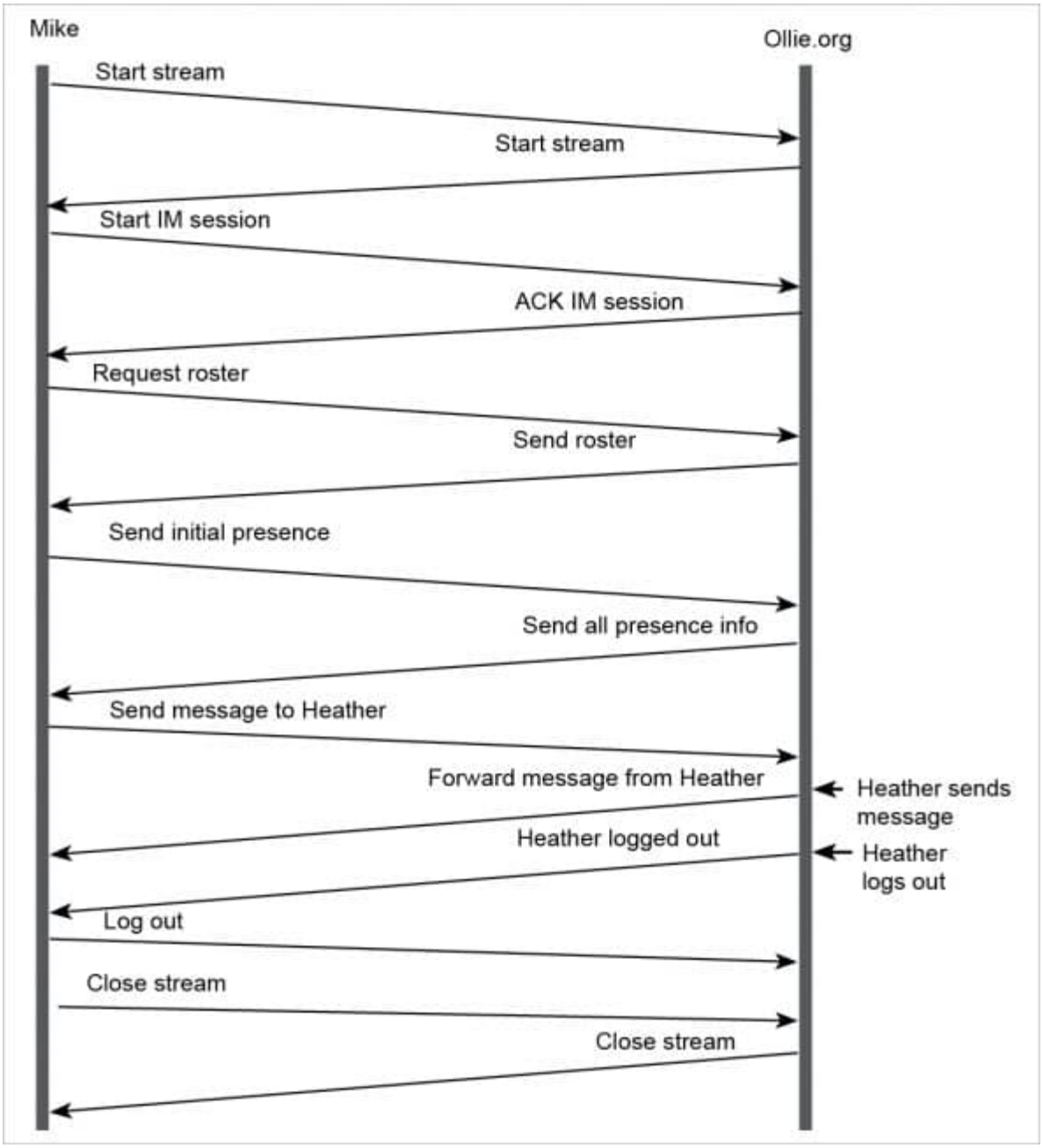


Fig.4.12 XMPP IM conversation

AMQP (Advanced Message Queuing Protocol):

- AMQP enables encrypted and interoperable messaging between organizations and applications. The protocol is used in client/server messaging and in IoT device management.
- AMPQ is efficient, portable, multichannel and secure. The binary protocol offers authentication and encryption by way of SASL or TLS, relying on a transport protocol such as TCP. The messaging protocol is fast and features guaranteed delivery with acknowledgement of received messages. AMPQ works well in multi-client environments and provides a means for delegating tasks and making servers handle immediate requests faster. Because AMPQ is a streamed binary messaging

system with tightly mandated messaging behavior, the interoperability of clients from different vendors is assured.

- AMQP allows for various guaranteed messaging modes specifying a message be sent:
    - At-most-once(sent one time with the possibility of being missed).
    - At-least-once (guaranteeing delivery with the possibility of duplicated messages).
    - Exactly-once (guaranteeing a one-time only delivery).

3. Edge computing Gateway

- An ***edge gateway processes data from edge devices, sending back relevant data, providing network translation between networks using different protocols and ultimately reducing bandwidth needs***.
- An edge IoT gateway is a computer that sits on the edge of a network, close to IoT devices, for the purpose of collecting data. It might consolidate, filter, analyze and/or perform computational tasks on the data before sending it to the cloud. The proximity of the IoT edge gateway to the data source is good for latency and efficiency and it reduces traffic on the network.
- Gateway name certainly fits, since it ***connects sensors and nodes at one end, provides one or multiple local functions, and extends bidirectional communications to the cloud***. Interestingly, edge devices often serve as gateways.

Architecture of Edge IoT gateway:

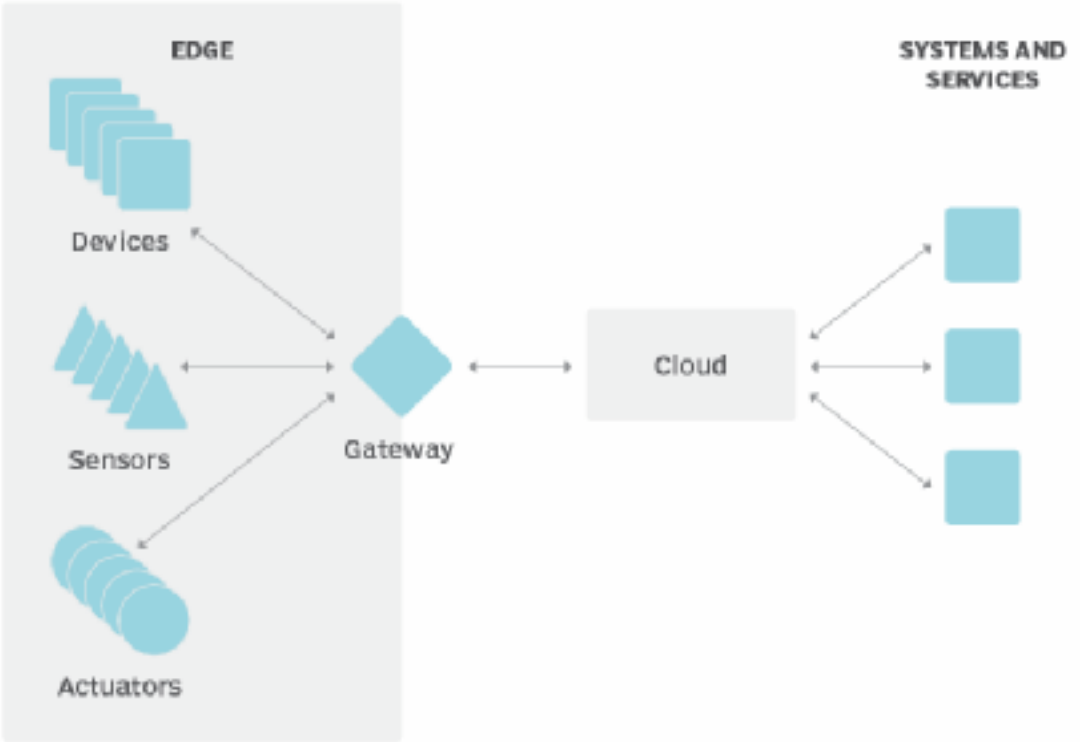


Fig.4.13 Gateways in IoT

- IoT gateways provide the glue that enables communication between IoT devices, their operators, and the cloud. They provide a way to gather, manage, process, and access data before it's passed to other devices, networks, or storage architectures. By using IoT gateways, administrators are able to gain control over where and how data is stored and processed.
- Devices like embedded or added sensors act as the device layer, acquiring data and sending it to the edge computing device or a cloud computing platform.
- A combination of sensors, actuators, routers, switches, and edge computing devices can be controlled and accessed locally or over WAN to provide visibility and the capability to act over long distances.
- If the data is informational or transactional, the device may send it directly to the cloud for storage. Or, based on parameters, it may send it to an edge device for processing and instructions and then forward the instructions to an actuator to trigger a response.

What does an IoT gateway do?

- 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 devices to the cloud. Now, it's common for a gateway device to handle both inbound and outbound traffic. Outbound traffic streams are used for sending IoT data to the cloud, while inbound traffic is used for device management tasks, such as updating device firmware.
- Some IoT gateways do more than just route traffic. A gateway device can sometimes be used to preprocess that data locally at the edge before sending it to the cloud. In doing so, the device might deduplicate, summarize or aggregate data as a way of reducing the volume of data that must be forwarded to the cloud. This can have a big effect on response times and network transmission costs.
- Some IoT gateways do more than just route traffic. A gateway device can sometimes be used to preprocess that data locally at the edge before sending it to the cloud. In doing so, the device might deduplicate, summarize or aggregate data as a way of reducing the volume of data that must be forwarded to the cloud. This can have a big effect on response times and network transmission costs.

How does an IoT gateway work?

- A simple IoT gateway functions similarly to a Wi-Fi router. An IoT system connects to the gateway using a Wi-Fi connection and then the gateway routes the IoT device data to the cloud. More often, though, IoT gateways are far more complex.
- One reason why an IoT gateway tends to be more complex than a Wi-Fi router is that there are several different protocols used by IoT devices. Some of these protocols include Z-Wave, BACnet, Bluetooth Low Energy and Zigbee. As such, an IoT gateway might need to support a variety of protocols to service all the IoT devices in an organization.

- Another reason why IoT gateways can be more complex than Wi-Fi routers is because IoT gateways might need to locally cache data in case of an internet outage or in case the gateway is flooded with more data than it can handle.
- Additionally, IoT gateways often support failover clustering or the ability to scale out to support increasingly large workloads.
- Some IoT devices produce vast quantities of data. This can be a problem if an organization has a significant number of devices in its IoT ecosystem and tries to send the data from all those devices to the cloud. The IoT devices could potentially deplete the organization's available internet bandwidth, while also incurring large cloud storage costs.
- One use for gateways is creating a communication link between an IoT environment and the cloud.
- IoT gateways are important for managing and securing IoT devices, and they might also help an organization to reduce its IoT-related internet bandwidth consumption

4. Edge ML for Industry automation

- Edge AI is the class of ML architecture in which the AI algorithms process the data on the edge of the network (the place where data is generated, i.e., locally) instead of sending it to the cloud.
- Edge machine learning (edge ML) is the process of running machine learning algorithms on computing devices at the periphery of a network to make decisions and predictions as close as possible to the originating source of data. It is also referred to as edge artificial intelligence or edge AI.
- In traditional machine learning, we often find large servers processing heaps of data collected from the Internet to provide some benefit, such as predicting what movie to watch next or to label a cat video automatically. By running machine learning algorithms on edge devices like laptops, smartphones, and embedded systems (such as those found in smartwatches, washing machines, cars, manufacturing robots, etc.), we can produce such predictions faster and without the need to transmit large amounts of raw data across a network.
- The nature of the edge architecture makes it a perfect fit for reducing the inefficiencies in the existing systems.
- **Operational Efficiency-** Significantly reduces latency, enhancing the real-time decision-making capabilities.
- **Enhanced Security-** It increases the level of security in terms of data privacy through local processing. Data is no longer shared in a centralized cloud.
- **Decentralization of Workloads-** Decentralization of the Processing makes all the distributed systems efficient and self-sustained.
- Machine learning offers the ability to create further advancements in automation by introducing models that can make predictions or decisions without human intervention.

Due to the complex nature of many machine learning algorithms, the traditional integration of IoT and ML involves sending raw sensor data to a central server, which performs the necessary inference calculations to generate a prediction.

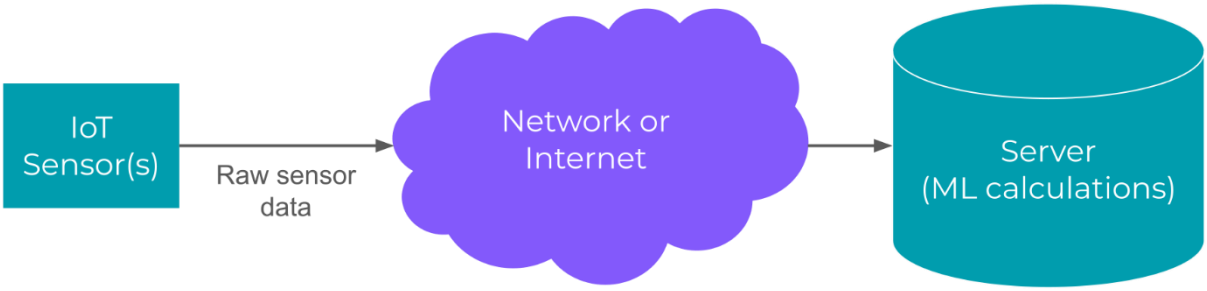


Fig.4.14 Data flow diagram for machine learning with connected sensors

For low volumes of raw data and complex models, this configuration may be acceptable. However, there are several potential issues that arise:

- Transmitting large sensor data, such as images, may hog network bandwidth
- Transmitting data also requires power
- The sensors require constant connection to the server to provide near real time ML.

To counter the need to transmit large amounts of raw data across networks, data storage and some computations can be accomplished on devices closer to the user or sensor, known as the "edge." Computations.

Advances in hardware and machine learning have paved the way for running deep ML models efficiently on edge devices. Complex tasks, such as object detection, natural language processing, and model training, still require powerful computers. In these cases, raw data is often collected and sent to a server for processing.
However, performing ML on low-power devices offers a variety of benefits:

- Less network bandwidth is spent on transmitting raw data
- While some information may need to be transmitted over a network (e.g. inference results), less communication often means reduced power usage
- Prediction results are available immediately without the need to send them across a network
- Inference can be performed without a connection to a network
- User privacy is ensured, as data is only stored long enough to perform inference (not including data collected for model training)

Edge ML includes personal computers, smartphones, and embedded systems. As a result, embedded ML, also known as tinyML, is a subset of edge ML that focuses on running machine learning algorithms on embedded systems, such as microcontrollers and headless single board computers.

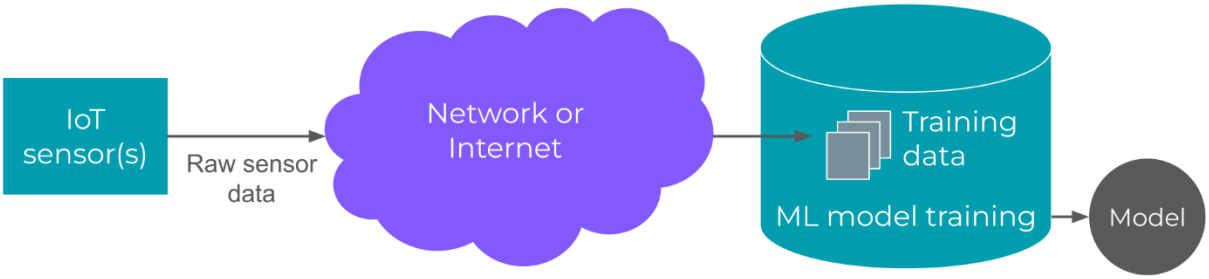


Fig.4.15 Machine learning model training in the cloud using data from IoT sensors

Once we have a trained model, which is just a mathematical model (in the form of a software library), we can deploy it to our smart sensor or other edge device. We can write firmware or software using the model to gather new raw sensor readings, perform inference, and take some action based on those inference results.

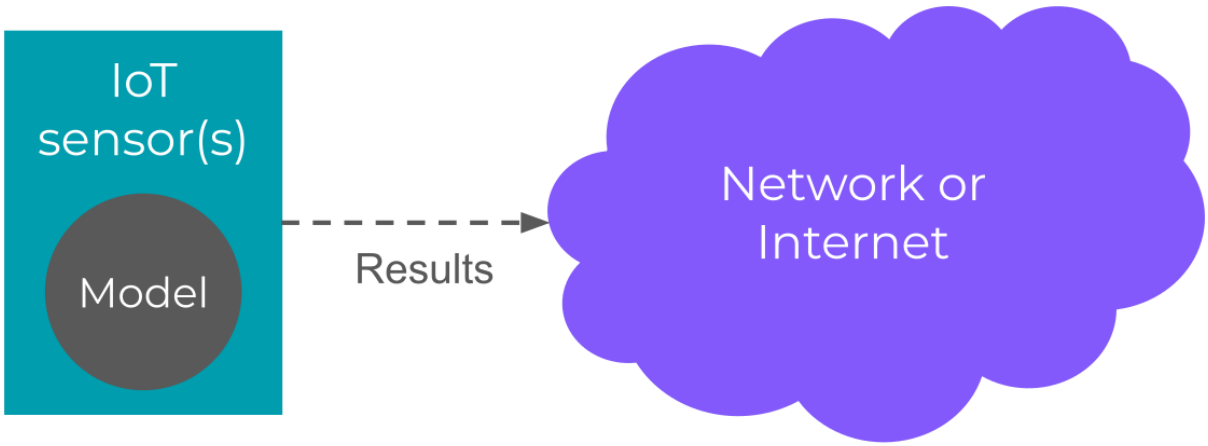


Fig.4.16 How a machine learning model is deployed to an edge device

ML models are not perfect. They provide a generalization of the training data. In other words, the model is only as good as the data used to train it. As a result, machine learning (and the subsequent field of data-driven engineering) will not replace traditional programming. However, it nicely complements other types of software engineering, and it opens new possibilities for solving difficult problems.

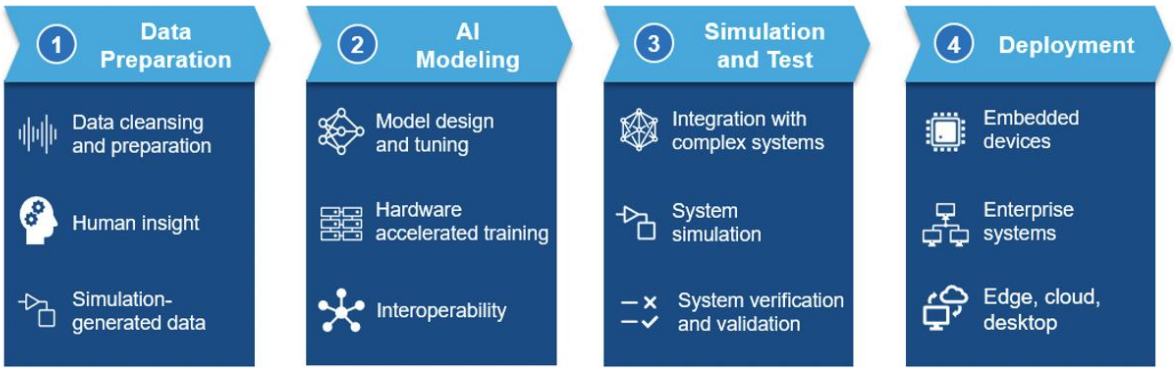


Fig.4.16 Edge ML for Industrial Automation

- **Data Ingestion and Storage of the Data :** The real-time sensor data from the monitoring system of machines into the pipeline. The stream of the data will be stored locally and securely as no cloud processing is involved here.
- **Processing:** This stage of the pipeline will process the data according to the need of the pre-trained models.
- **Analysis:** Here the data will be analyzed by the models, and they will give the results.
- **Results:** This stage will compile the results. And after these Results, the Response can be taken by the stakeholders.

Use cases of Edge AI

- Predictive Maintenance refers to the ability to pre-emptively detect the failure of machines using machine learning predictive algorithms.
    - Predictive Maintenance has been in the industry for some time, but it has also been difficult to implement.
- Condition-based Monitoring: Challenges in fetching the data from their machines, processes, and system. Whether the streams are sent to the cloud and Then processing is done.
    - If some initial filtering can be done, then only useful data streams can be utilized in the cloud or locally, this can be achieved with edge ai near the data generation streams.
- Precision monitoring and controlling system: Use a large amount of data Machine learning algorithms.
    - Edge Computing is a perfect fit for it as it can collect, aggregate, and filter the data used by the AI/ML algorithms.

Applications:

Agriculture

- Automatically identifying irrigation requirements
- ML-powered robots for recording crop trials

Smart buildings

- Smart HVAC systems that can adapt to the number of people in a room
- Security sensors that listen for the unique sound signature of glass breaking

Environment conservation

- Smart grid monitoring that looks for early faults in power lines
- Wildlife tracking

Health and fitness

- Portable medical devices that can identify diseases from images

- Tiny wearables that track your sleep and workouts

Human-computer interaction (HCI)

- Keyword spotting and wake word detection to control household appliances
- Gesture control as assistive technology

Industry

- Safety systems that automatically detect the presence of hard hats
- Predictive maintenance that identities faults in machinery before larger problems arise

Edge computing in Health Care

Monitoring devices (e.g. glucose monitors, health tools and other sensors) are either not connected, or where they are, large amounts of unprocessed data from devices would need to be stored on a 3rd party cloud. This presents security concerns for healthcare providers. An edge on the hospital site could process data locally to maintain data privacy. A key driver of digital health will be leveraging capabilities at the edge. Edge computing has been rapidly gaining traction and tangible edge deployments have been increasing significantly.

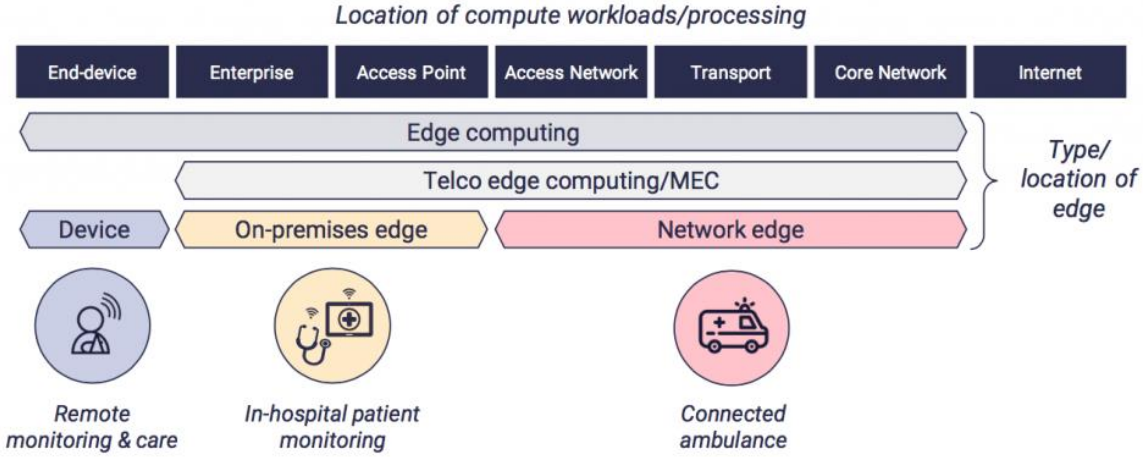


Fig.4.17 Edge Computing in Health care

The three key use cases which could leverage edge computing to help drive the transformation of the healthcare industry and the move towards digital health are as follows:

1. Connected ambulance

In current emergency service systems, paramedics are typically only able to brief emergency doctors, and patients can only receive the diagnostic procedures they require, once the ambulance transporting the patient has arrived at the hospital. This can create inefficiencies and delays in terms of hand over times and patient transferals to the correct wards, meaning

ambulances are hindered from getting back into the field and patient diagnoses and treatments are slowed down. In an emergency situation, seconds and minutes can be essential.

The low latency, mobility and data processing capabilities of edge computing at a network edge (alongside 5G) can enable faster and more accurate diagnosis and treatment by paramedics on-site, as well as more granular information at the hospital on the status and location of incoming patients.

This can primarily be achieved through:

- Live streaming of processed patient data, such as heart rate from sensors and monitors or live video feeds from first responder body cams to the hospital (enabled through 5G). This gives hospital staff a heads up on incoming patients' status as well as initial information on potential treatment processes.
- Analysis of patient information and vitals (such as blood pressure, heart rate) at the edge for real-time diagnosis and recommendations by first responders.
- Augmented reality glasses (rendered at the edge) to display information about patient history and complex treatment protocols to empower paramedics
- Haptic-enabled diagnostic tools to enable remote diagnostics by specialists (e.g. remote ultrasounds)

Telcos such as *Vodafone* and *BT* are exploring the potential value of this use case, developing PoCs to deliver the service via 5G and MEC (multi-access edge computing).

2. In-hospital patient monitoring

- Currently, monitoring devices (e.g. glucose monitors, health tools and other sensors) are either not connected, or where they are, large amounts of unprocessed data from such devices would need to be stored on a 3rd party cloud. This presents some level of security concerns of healthcare providers, and limits adoption of such use cases.
- An on-premise edge on the hospital site could process data locally to maintain data privacy and compliance, compiling information from multiple sources within the hospital and extracting relevant information. Edge enables right-time notifications to practitioners of unusual patient trends or behaviors (through analytics/AI), creation of 360-degree view patient dashboards for full visibility, and sending relevant data to be stored securely in a cloud system.
- This creates significant resource efficiency for clinicians, increasing productivity and decreasing cost per patient.

Fig.4.18 Connecting Sensor network to cloud data center through gateway

Examples:

- *Google DeepMind/Google Health*: early prediction of Acute Kidney Injury – a life threatening condition which, when caught early, can be regulated
- *CareAI*: platform for acute and post-acute patient monitoring to enhance quality of care and improve operational efficiency
- *AWS Outposts*: AWS is targeting the healthcare industry with its on-premise edge technology stack, Outposts. They are looking to bring the power of AWS core cloud to the hospital/research site, mitigating (some) of the security concerns healthcare providers may have around data privacy

3. Remote monitoring and care

- Life expectancy is increasing globally and populations are increasing in wealth. These two factors coupled together leads to an increase in treatment of chronic, non-communicable diseases (e.g. diabetes, chronic obstructive pulmonary disease (COPD), cardiac diseases). This adds pressure to healthcare and long-term social care as demands on medical professionals' time increase with a move towards continuous care and chronic disease management.
- Furthermore, especially in developing or heavily rural markets (e.g. Australia), there are many patients who physically cannot access the care they need because travel to medical infrastructure is either prohibitively far or expensive.
- Remote patient monitoring could help alleviate some of this strain, as well as improve care for an increasingly elderly population and increase patient access to healthcare. IoT, e-health devices and patient wearable can provide real time visibility of a person's status providing insight, alarms and alerts (missed medication, increased blood glucose levels, severe fall, heart rate monitoring etc.) to care workers and clinicians.
- However, it is often perceived as unsafe to run these applications in the cloud due to the sensitivity of patient data, decreasing adoption of these solutions. Additionally, some patient monitoring devices may leverage video to, for example, track if patients have fallen down/any significant changes in patient behavior – but sending unprocessed video is extremely bandwidth intensive.
- Processing data at the edge (e.g. on the camera, on a solution specific tablet, or on the network edge) can ensure the protection of sensitive data transmitted between patient and healthcare provider. The low latency aspects can also enable real-time altering which, in the case of a potential stroke, could be essential. Edge can also reduce the cost and strain on bandwidth by processing data close to the source and limiting the amount which is sent to healthcare providers/stored in a secure location.

Module:5 Security Engineering

1. IoT Attacks and Security Challenges

- The internet of things (IoT) is the vast network of connected physical objects (i.e., things) that exchange data with other devices and systems via the internet. While it refers to the actual devices, IoT is commonly used as an overarching term to describe a highly-distributed network that combines connectivity with sensors and lightweight applications, which are embedded into tools and devices. These are used to exchange data with other devices, applications, and systems for everything from smart plugs and power grids to connected cars and medical devices.
- IoT security is an umbrella term that covers the strategies, tools, processes, systems, and methods used to protect all aspects of the internet of things. Included in IoT security is the protection of the physical components, applications, data, and network connections to ensure the availability, integrity, and confidentiality of IoT ecosystems.
- Internet of Things (IoT) promises to make our lives more convenient by turning each physical object in our surrounding environment into a smart object that can sense the environment, communicate with the remaining smart objects, perform reasoning, and respond properly to changes in the surrounding environment.
- However, the conveniences that the IoT brings are also associated with new security risks and privacy issues that must be addressed properly. Ignoring these security and privacy issues will have serious effects on the different aspects of our lives.
- IoT Security is the act of securing Internet devices and the networks they're connected to from threats and breaches by protecting, identifying, and monitoring risks all while helping fix vulnerabilities from a range of devices that can pose security risks to your business.
- Security challenges abound, because of the high volume of flaws regularly discovered in IoT systems. Robust IoT security includes all facets of protection, including hardening components, monitoring, keeping firmware updated, access management, threat response, and remediation of vulnerabilities. IoT security is critical as these systems are sprawling and vulnerable, making them a highly-targeted attack vector. Securing IoT devices from unauthorized access ensures that they do not become a gateway into other parts of the network or leak sensitive information.
- IoT security vulnerabilities are found in everything from vehicles and smart grids to watches and smart home devices. For example, researchers found webcams that could be easily hacked to gain access to networks and smartwatches containing security vulnerabilities that allowed hackers to track the wearer's location and eavesdrop on conversations.

The Importance of IoT Security

- IoT is widely believed to be one of the most significant security vulnerabilities that impact nearly everyone—consumers, organizations, and governments. For all of the convenience and value derived from IoT systems, the risks are unparalleled. The

importance of IoT security cannot be overstated, as these devices provide cybercriminals with a vast and accessible attack surface.

- IoT security provides the vital protections needed for these vulnerable devices. Developers of IoT systems are known to focus on the functionality of the devices and not on security. This amplifies the importance of IoT security and for users and IT teams to be responsible for implementing protections.

IOT Security Requirements

1. Confidentiality: ensures that the exchanged messages can be understood only by the intended entities.
2. Integrity: ensures that the exchanged messages were not altered/tampered by a third party.
3. Authentication: ensures that the entities involved in any operation are who they claim to be. A masquerade attack or an impersonation attack usually targets this requirement where an entity claims to be another identity.
4. Availability: ensures that the service is not interrupted. Denial of service attacks target this requirement as they cause service disruption.
5. Authorization: ensures that entities have the required control permissions to perform the operation they request to perform.
6. Freshness: ensures that the data is fresh. Replay attacks target this requirement where an old message is replayed in order to return an entity into an old state.
7. Non-repudiation: ensures that an entity cannot deny an action that it has performed.
8. Forward Secrecy: ensures that when an object leaves the network, it will not understand the communications that are exchanged after its departure.
9. Backward Secrecy: ensures that any new object that joins the network will not be able to understand the communications that were exchanged prior to joining the network.

IOT Three Domain Architecture:

The three-domain architecture that we consider in our security analysis are

- **IoT Sensing Domain**: This domain is made up of all the smart objects that have the capability to sense the surrounding environment and report the sensed data to one of the devices in the fog domain. The smart objects in the sensing domain are expected to change their location over time.
- **Fog Domain:** This domain consists of a set of fog devices that are located in areas that are highly populated by many smart objects. Each fog device is allocated a set of smart objects where the allocated objects report their sensed data to the fog device. The fog device performs operations on the collected data including aggregation, preprocessing, and storage. Fog devices are also connected with each other in order to manage the communication among the smart objects and in order to coordinate which fog device will be responsible for handling which object as objects change their location over time. Each fog device is also connected to one or multiple servers in the cloud domain.
- **Cloud Domain:** This domain is composed of a large number of servers that host the applications that are responsible for performing the heavy-computational processing operations on the data reported from the fog devices.

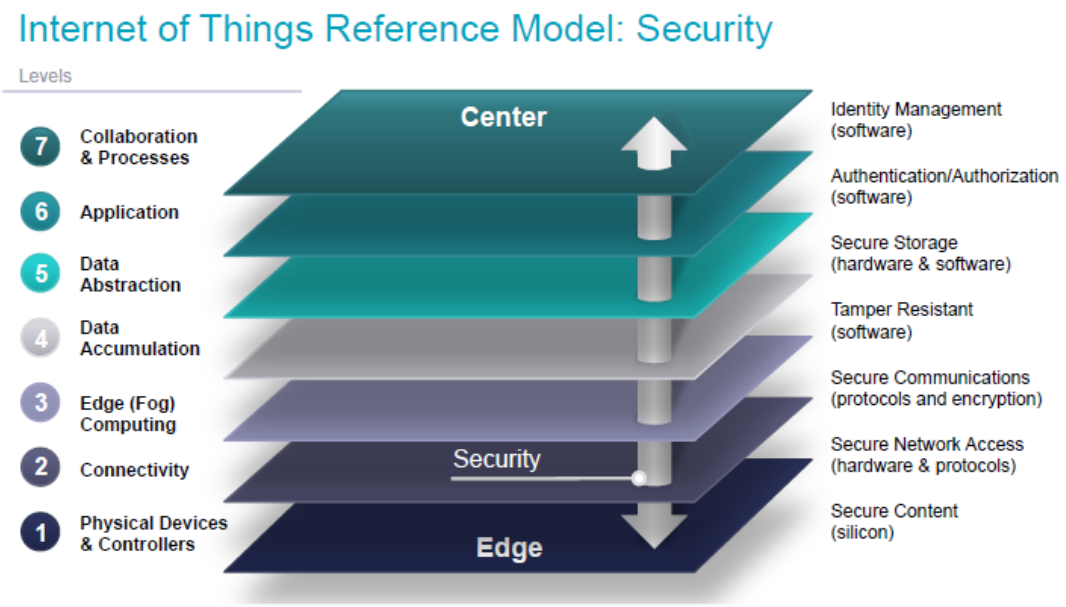


Fig.5.1 IOT Reference Model Security

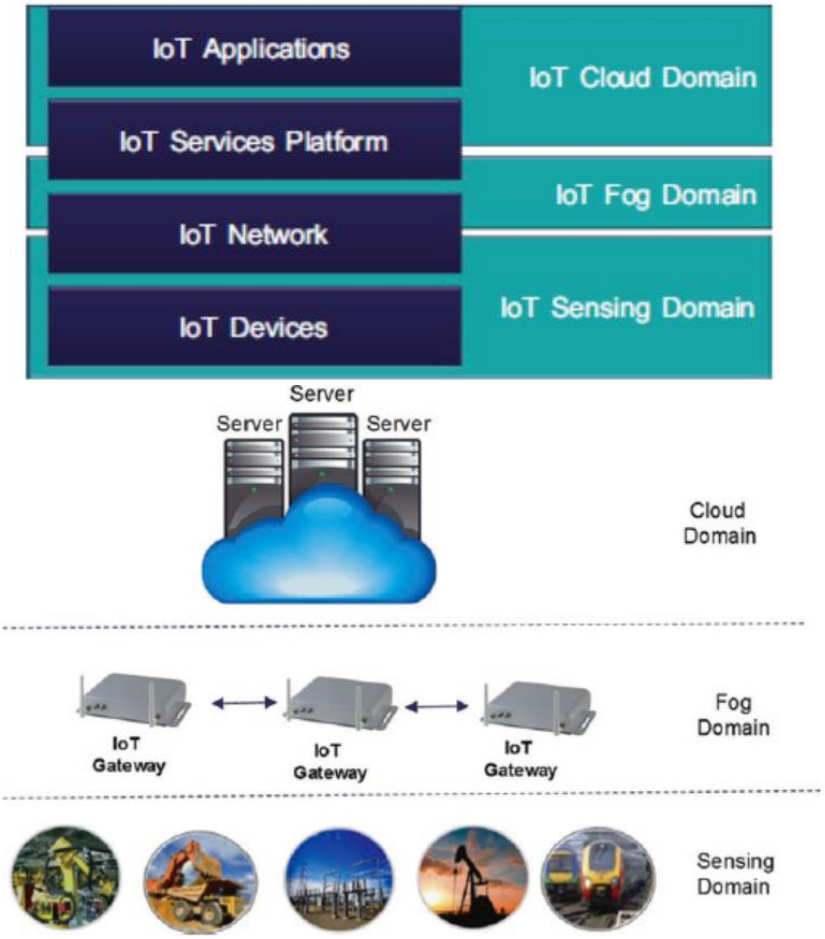


Fig.5.2 Mapping of IoT domains

Cloud Domain Attacks and Countermeasures

Each IoT application is dedicated one or multiple virtual machines (VMs) where each VM is assigned to one of the servers in the cloud data center and gets allocated certain amount of CPU and memory resources in order to perform certain computing tasks. The cloud data center is made up of thousands of servers where each server has certain CPU, memory, and storage capacities, and thus each server has a limit on the number of VMs that it can accommodate. The servers in the cloud data center are virtualized which allows multiple VMs to be assigned to the same server as long as the server has enough resource capacity to support the resource requirements of each hosted VM.

Attack	Vulnerability Reason	Security Violation	Countermeasures
Hidden-Channel Attack	Shared hardware components (e.g. cache) among the server's VMs	Confidentiality	- Hard Isolation - Cache Flushing - Noisy Data Access Time - Limiting Cache Switching Rate
VM Migration attacks	- VM Migration software bugs - VM Migration is performed without authentication - Memory pages copied in clear	Confidentiality Integrity Availability	- Server authentication - Encrypting migrated memory pages
Theft-of-Service Attack	Periodic sampling of VMs' used resources	Availability Non-Repudiation	-Fine-grain sampling using high precision clocks - Random sampling
VM Escape Attack	Hypervisor software bugs	Confidentiality Availability Integrity	- Add an isolation domain between the hypervisor and hardware
Insider Attacks	Lack of trust in cloud administrators	Confidentiality Integrity	- Homomorphic Encryption - Secret storage through data chopping and permutation based on a secret key

Fig.5.3 Cloud Domain Attacks and Countermeasures

Fog Domain Attacks and Countermeasures

- The fog device performs different operations on the collected data which include data aggregation, data preprocessing, and data storage. The fog device may also perform some reasoning operations on the collected data. After processing and aggregating the collected data, the fog device forwards these data to the cloud domain.
- Each fog device provides computing resources to be used by the IoT smart objects that are located close to the fog device. These computing resources are virtualized in order to allow the connected objects to share the computing resources that are offered by the fog device where each object or set of connected objects are allocated a virtual machine that performs the necessary data processing operations.
- Although the fog domain is highly similar to the cloud domain, there are three key differences that distinguish fog devices from cloud servers:
    - **Location:** Unlike cloud servers which are usually located far from smart objects, fog devices are placed in areas with high popular access and thus are

placed close to the smart objects. This placement plays an important role in giving the fog devices the ability to respond quickly to changes in the reported data. This also gives the fog devices the ability to provide location-aware services as smart objects connect to the closest fog device, and thus each fog device knows the location of the objects connected to it.

- **Mobility:** Since the location of the smart object may change over time, then the Virtual Machines created to handle those objects at the fog domain must be moved from a fog device into another, in order to keep the processing that is performed in the fog device close to the object that is generating data.
- **Lower Computing Capacity:** The fog devices that are installed in a certain location are expected to have a lower computing capacity when compared to capacities offered by cloud data centers as the latter are made of thousands of servers.

The security threats that are specific to the fog domain are

- **Authentication and Trust Issues:** The fact that fog devices do not require a large facility space or a high number of servers compared to cloud data centers will encourage many small and less-known companies to install virtualized fog devices in dense areas and to offer these computing resources to be rented by the smart objects that are near the installed fog devices. Unlike cloud data centers which are offered by well-known companies, fog devices are expected to be owned by multiple and less-known entities. An important security concern that needs then to be taken into account when assigning a smart object to a fog device is to authenticate first the identity of the owner of the fog device.
- **Higher Migration Security Risks:** Although VM migration is common in both the cloud and the fog domains, there is an important difference between the migration in the cloud domain and that in the fog domain. While the migrated VMs in the cloud domain are carried over the cloud data center's internal network, the migrations from a fog device into another are carried over the Internet. Thus there is a higher probability that the migrated VMs get exposed to compromised network links or network routers when moving a VM from a fog device into another. This makes it vital to encrypt the migrated VM and to authenticate the VM migration messages that are exchanged among the fog devices.
- **Higher Vulnerability to DoS Attacks:** Since fog devices have lower computing capacities, this makes them a low-hanging-fruit for denial of service (DoS) attacks where attackers can easily overwhelm fog devices when compared to the cloud data centers, where a huge number of servers that have high computing capacity are available.
- **Additional Security Threats Due to Container Usage:** In order to provide the computing needs for a larger number of connected objects, the fog device may use containers rather than VMs to allocate the resource demands for each connected object. The low overhead of containers allows larger number of objects to be served by the fog device. However, sharing the same operating system among the containers dedicated for objects that belong to different users raises serious security concerns as the

opportunities for data leakage and for hijacking the fog device increase significantly. The industry needs to address these gaps in container security to enable IoT applications at scale.

- **Privacy Issues:** We mentioned before that each smart object will be connected to one of the fog devices that are close to it. This means that the fog device can infer the location of all the connected smart objects. This allows the fog device to track users or to know their commuting habits which may break the privacy of the users carrying those objects. New mechanisms should be developed in order to make it harder for fog devices to track the location of the smart objects over time. Furthermore, the advancement in wireless signal processing has made it possible now to identify the presence of humans and track their location, their lip movement, and their heartbeats by capturing and analyzing the wireless signals that are exchanged between the sensing objects and the fog domain. This advancement makes it possible for any entity to install a reception device close to your home that analyzes the wireless signals that are emitted from your home in order to spy on your daily activities.

Sensing Domain Attacks and Countermeasures

The sensing domain contains all the smart objects, where each object is equipped with a number of sensors that allow the object to perceive the world. The smart object is also supplied with a communication interface that allows it to communicate with the outer world. The smart object reports the sensed data to one of the fog devices in the fog domain. This is done by either creating a direct connection with the fog device if the smart object is directly connected by wires or has the wireless transmission capability to reach that fog device or in a multi-hop fashion where the smart object relies on other smart objects that lie along the path to the fog device to deliver the sensed data.

Attack	Target OSI layer	Vulnerability reason	Security violation	Countermeasures
Jamming attack	Physical Data link	Shared wireless channel	Availability	Frequency hopping Spread spectrum Directional antennas Jamming detection techniques
Vampire attack	Data link Network	Limited battery lifetime	Availability Freshness	Rate limitation Drop packets with a source route that contains a loop Monitor whether or not the forwarded packets are making progress toward their destination
Selective-forwarding attack	Network	Limited transmission capability	Availability	Increase transmission range Path redundancy Choose certain intermediate objects as checkpoints to acknowledge received packets
Sinkhole attack	Network	Limited transmission capability	Confidentiality Availability	Analyze the collected routing information from multiple objects

Fig.5.4 Sensing Domain Attacks and Countermeasures

IOT Security Challenges

IoT has unique characteristics and constraints when it comes to designing efficient defensive mechanisms against cybersecurity threats that can be summarized by

1. Multiple Technologies: IoT combines multiple technologies such as radio-frequency identification (RFID), wireless sensor networks, cloud computing, virtualization, etc. Each of these technologies has its own vulnerabilities. The problem with the IoT paradigm is that one must secure the chain of all of those technologies as the security resistance of an IoT application will be judged based on its weakest point which is usually referred to by Achilles' heel.

2. Multiple Verticals: The IoT paradigm will have numerous applications (also called verticals) that span eHealth, industrial, smart home gadgets, smart cities, etc. The security requirements of each vertical are quite different from the remaining verticals.

3. Scalability: According to Cisco, 26.3 billion smart devices will be connected to the Internet by 2020. This huge number makes scalability an important issue when it comes to developing efficient defensive mechanisms. None of the previously proposed centralized defensive frameworks can work anymore with the IoT paradigm, where the focus must be switched to finding practical decentralized defensive security mechanisms. An IoT solution needs to scale cost-effectively, potentially to hundreds of thousands or even millions of endpoints.

4. Availability: Availability refers to characteristic of a system or subsystem that is continuously operational for a desirably long period of time. It is typically measured relative to "100% operational" or "never failing." A widely held but difficult-to- achieve standard of availability for a system or product is known as "five 9 s" (available 99.999% of the time in a given year) availability. Security plays a major rule in high availability as network administrators often hesitate to use needed threat-response technology functions for fear that such functions will take down critical systems. Even a simple port scan causes some IoT devices to stop working, and the cost of downtime can far exceed the cost of remediating all but the most severe incidents. In some instances, network administrators would rather have no cybersecurity protection rather than risk an outage due to a false positive. This leaves them blind to threats within their control networks. Companies often add redundancy to their systems so that failure of a component does not impact the entire system.

5. Big Data: Not only the number of smart objects will be huge, but also the data generated by each object will be enormous as each smart object is expected to be supplied by numerous sensors, where each sensor generates huge streams of data over time. This makes it essential to come up with efficient defensive mechanisms that can secure these large streams of data.

6. Resource Limitations: The majority of IoT end devices have limited resource capabilities such as CPU, memory, storage, battery, and transmission range. This makes those devices a low-hanging-fruit for denial of service (DoS) attacks.

7. Remote Locations: In many IoT verticals (e.g., smart grid, railways, roadsides), IoT devices, epically sensors, will be installed in unmanned locations that are difficult to reach. Attackers can interfere with these devices without being seen. Cyber and physical security monitoring systems must be installed in safe-guarded location, operate in extreme environmental

conditions, fit in small spaces, and operate remotely for routine updates and maintenance avoiding delayed and expensive visits by network technicians.
8. *Mobility*: Smart objects are expected to change their location often in the IoT paradigm. This adds extra difficulties when developing efficient defensive mechanisms in such dynamic environments.
9. *Delay-Sensitive Service*: The majority of IoT applications are expected to be delay-sensitive, and thus one should protect the different IoT components from any attack that may degrade their service time or may cause a service disruption.

2. Threat and Mitigating Threats to IoT Systems

Security Concerns	Application & Interface Layer	Service Support Layer	Network Layer	Device Layer
Insecure web interface	✓	✓	✓	
Insufficient authentication/authorization	✓	✓	✓	✓
Insecure network services		✓	✓	
Lack of transport encryption		✓	✓	
Privacy concerns		✓	✓	✓
Insecure cloud interface	✓			
Insecure mobile interface	✓		✓	✓
Insecure security configuration	✓	✓	✓	
Insecure software/firmware	✓		✓	
Poor physical security			✓	✓

Fig.5.5 Top Ten Vulnerabilities in IoT

IoT Security issues

As noted above, IoT devices were not built with security in mind. This results in myriad IoT security challenges that can lead to disastrous situations. Unlike other technology solutions, few standards and rules are in place to direct IoT security. In addition, most people do not understand the inherent risks with IoT systems. Nor do they have any idea about the depth of IoT security challenges. Among the many IoT security issues are the following:

- *Lack of visibility*
  Users often deploy IoT devices without the knowledge of IT departments, which makes it impossible to have an accurate inventory of what needs to be protected and monitored.
- *Limited security integration*
  Because of the variety and scale of IoT devices, integrating them into security systems ranges from challenging to impossible.
- *Open-source code vulnerabilities*
  Firmware developed for IoT devices often includes open-source software, which is prone to bugs and vulnerabilities.
- *Overwhelming data volume*
  The amount of data generated by IoT devices makes data oversight, management, and protection difficult.
- *Poor testing*
  Because most IoT developers do not prioritize security, they fail to perform effective vulnerability testing to identify weaknesses in IoT systems.
- *Unpatched vulnerabilities*
  Many IoT devices have unpatched vulnerabilities for many reasons, including patches not being available and difficulties accessing and installing patches.
- *Vulnerable APIs*
  APIs are often used as entry points to command-and-control centers from which attacks are launched, such as SQL injection, distributed denial of service (DDoS), man-in-the-middle (MITM), and breaching networks
- *Weak passwords*
  IoT devices are commonly shipped with default passwords that many users fail to change, giving cyber criminals easy access. In other cases, users create weak passwords that can be guessed.

Addressing IoT Security Challenges

A holistic approach is required to implement and manage IoT security effectively. It must encompass a variety of tactics and tools as well as take into consideration adjacent systems, such as networks.

Three key capabilities for a robust IoT security solution are the ability to:

1. *Learn*
   Take advantage of security solutions that provide network visibility to learn what the ecosystem encompasses at what the risk profiles are for each group of IoT devices.
2. *Protect*
   Monitor, inspect, and enforce IoT security policies commiserate with activities at different points in the infrastructure
3. *Segment*
   In the same way that networks are segmented, use segmentation based on policy groups and risk profiles to segment IoT systems.

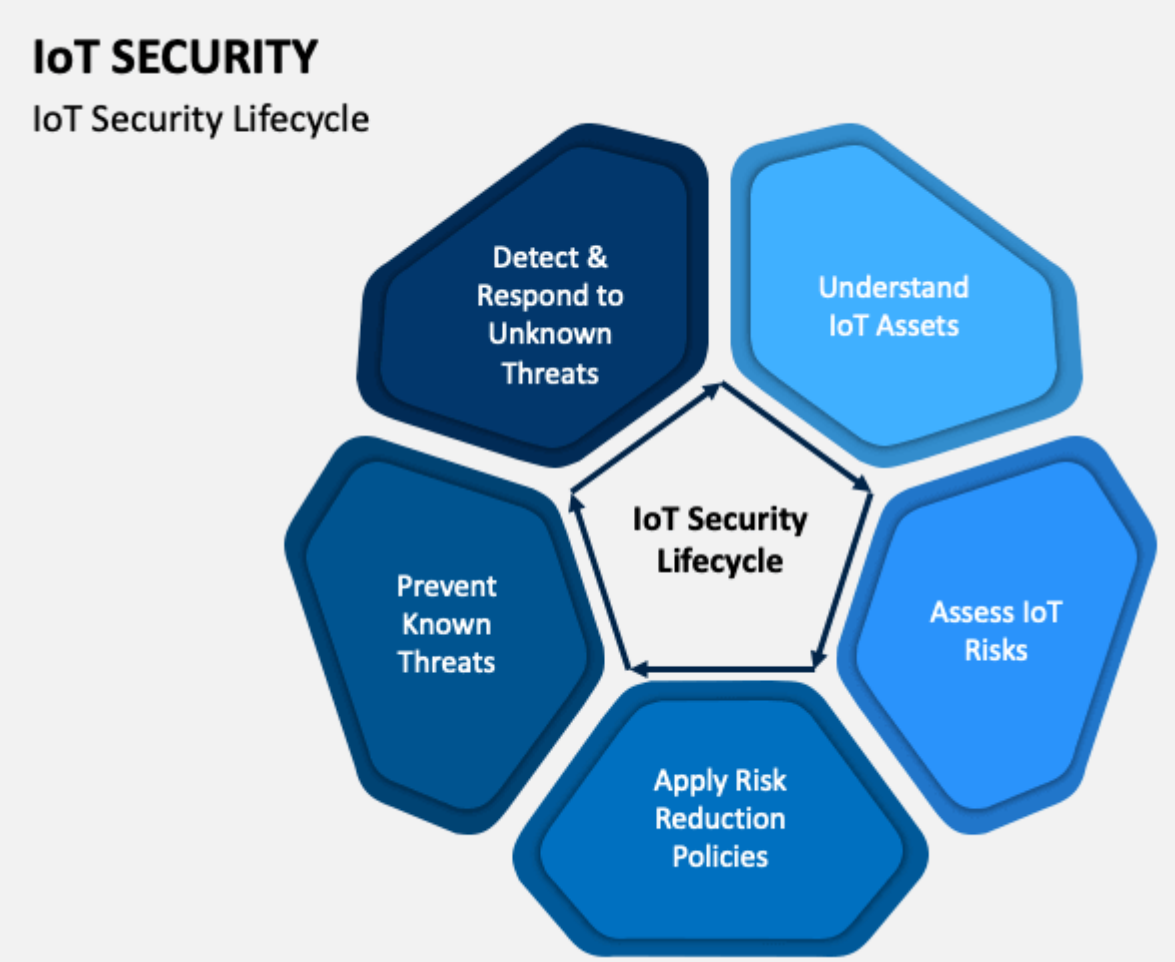


Fig.5.6 IoT Security Life cycle

Specific features required for securing IoT devices include the following:

- API security
- Broader and deep IoT device inventory
- Continuous software updates
- DNS filtering
- Education and training staff, vendors, and partners
- Encryption for data at rest and in transit
- Honeypot decoy programs
- Multi-factor authentication
- Network security
- Network traffic monitoring analysis
- Password management
- Patch management
- Security gateways
- Unauthorized IoT device scans

Enhance IoT Security to Realize Increased Benefits

IoT devices are increasingly being used by individuals and across the enterprise. They are not only here to stay, but proliferating exponentially in more and more forms. The result is increasing complexity, which hampers efforts to manage IoT systems security successfully.

IoT security challenges range from deflecting malicious insiders to defending against nation-state attacks. Because of the inherent vulnerability of IoT devices and the scale of their deployment, attacks continue to grow in scale and scope.

Securing IoT devices is well worth the investment despite the IoT security challenges. The value realized with IoT devices can only be increased with enhanced security to be on par with other technology. It will mitigate risks and increase rewards.

IoT Security Best Practices

The very first step in securing IoT is knowing what is connected. This includes using a device identification and discovery tool that automates three critical IoT security functions.

1. Automatically and continuously detects, profiles, and classifies IoT devices on the network
2. Maintains a real-time inventory of devices
3. Provides relevant risk insights for each of these asset classes by continuously monitoring across attack vectors.

By following these industry best practices for IoT security and adopting leading-edge solutions, you can understand, manage, and secure your complete asset inventory, including IoT.

Mitigating Threats to IoT Systems:

MUD: Manufacturer Usage Descriptor (MUD) is an embedded software standard defined by the IETF (RFC 8520) to help reduce the vulnerability surface of IoT devices by employing network policy (whitelisting approach). It aims to reduce the scope of malware injection and hijacking of over-the-air firmware updates. It also addresses the scenario of devices that are no longer being actively maintained by their original manufacturer.

- MUD enables IoT device manufacturers to advertise formal device specifications, including the intended communication patterns for a given device when connected to the network. The network can then leverage this advertised intent, or profile, to formulate a tailored and context-specific access control policy, to guarantee that the device communicates only within the specified parameters. This way the network behavior of the device, in any operating environment, can be locked down and verified rigorously.
- In this context, MUD becomes the delegated identifier and authoritative enforcer of policy for IoT devices on the network. MUD works by enabling networks to automatically permit each IoT device to send and receive only the traffic it requires to perform as intended while blocking unauthorized communication with the device.
- The MUD solution consists of three key components
    - A unique identifier, in the form of a Universal Resource Locator (URL), that an IoT device advertises when it connects to the network.

- An Internet hosted profile file that this URL points to. This file contains an abstracted policy that describes the level of communication access which the IoT device needs to perform its intended functionality.
- A core process that receives the URL from the IoT Device, retrieves the profile file from the MUD File Server, and establishes the appropriate access control policies in the network to restrict the communication patterns for that IoT device.

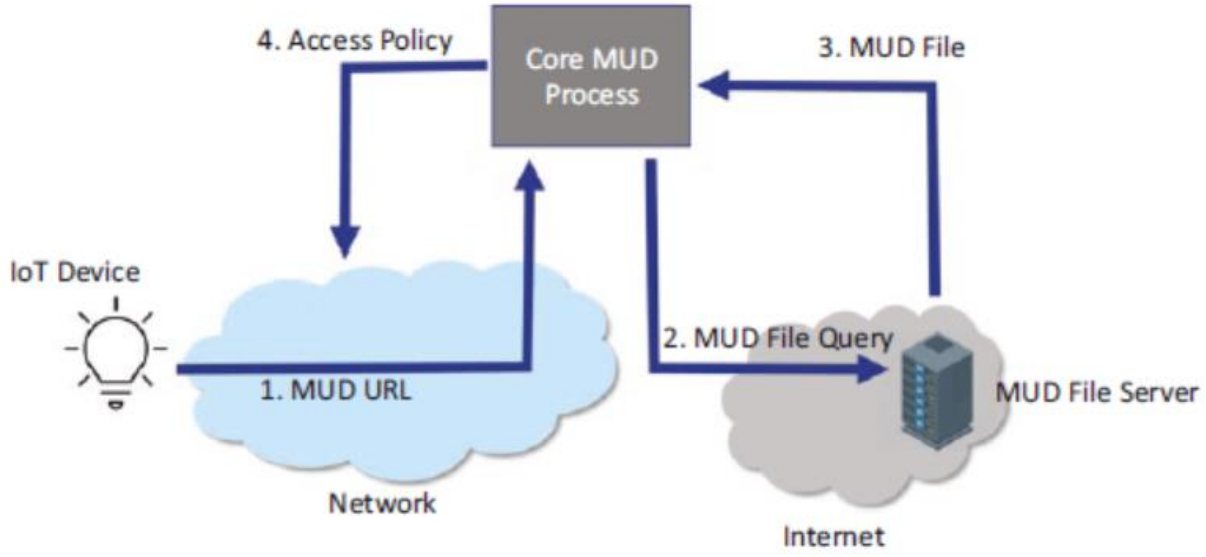


Fig.5.7 MUD

Device Identifier Composition Engine (DICE)

- Device Identifier Composition Engine (DICE) is a collection of hardware and software mechanisms for cryptographic IoT device identity, attestation, and data encryption.
  IoT devices that perform encryption use a private key called a Unique Device Secret (UDS) in order to secure their operation. It is possible for an attacker to leak this key by compromising the code running on the chip. Having access to the private key can enable the attacker to impersonate the device and even to replace its firmware. Therefore, it is paramount to prevent the disclosure of the UDS. The key to
- DICE is its ability to break up the boot process for any device into layers and to combine unique secrets and a measure of integrity for each of these layers. This way, if malware is present at any stage of the boot process, the device is automatically re-keyed and secrets protected by the legitimate keys remain safe.
- DICE implements three measures to secure the UDS:
- Power-on Latch: The power-on latch locks read access to the UDS before early boot-code transfers control to subsequent execution layers.
- Cryptographic One-way Functions: A cryptographic one-way function computes a hash of the UDS to store in RAM so that in the event of RAM disclosure by compromised code, the original UDS is safe.

- Tying Key Derivation to Software Identity: To prevent compromise of the device by attempts to modify the early boot-code, the cryptographic one-way function uses a measurement of the boot code as input together with the UDS. The function outputs a key called the Compound Device Identifier (CDI) taking both the UDS and early boot code hash as input (optionally taking the hardware state and configuration as input as well). This process ensures that modification of early boot code generates a new key so that the UDS is secure.
- DICE is predicated upon a hardware root of trust for measurement. It works by organizing the boot into layers and creating secrets unique to each layer and configuration based on UDS.
- If a different code or configuration is loaded, at any point in the boot chain, the secrets will be different. Each software layer keeps the secret it receives completely confidential. If a vulnerability exists and a secret is leaked, it patches the code automatically and creates a new secret, effectively re-keying the device. In other words, when malware is present, the device is automatically re-keyed and secrets are protected.
- DICE provides strong device identity, attestation of device firmware and security policy, and safe deployment and verification of software updates. The latter are often a source of malware and other attacks. Another key benefit for device manufacturers is that they are no longer required to maintain databases of unique secrets.

Botnets: A botnet is a typically large collection of networked computers (bots) that are under remote control from some malicious third party over the Internet. Usually, these computers would have been compromised by an outside attacker who controls aspects of their functionality without the owners' consent or knowledge.

3. Security Threats for IoT, Countermeasures Based on Encryption

- The biggest security-related threat of IoT systems from the traditional IT systems is that even using devices for data collection from the real world can become a target of cyberattacks.
- For example, the purpose of applying IoT to a plant is to significantly improve the productivity and maintainability by collecting data from a large number of sensors installed in production equipment, by analyzing it and performing autonomous control in real time.
- If sensor data should be falsified during this process, incorrect analysis results would be induced and erroneous control would result due to such an occurrence having the potential of leading to major damage.
- Moreover, since measurement data and control commands are trade secrets associated with the know-how of production and management, preventing leakages is also important from the viewpoint of competitiveness. Even if there is no problem at present, it is necessary to consider the effect of threats that might become evident in the future.
- Applying encryption to sensor devices means the implementation of data protection for confidentiality and integrity, which can be an effective countermeasure against the threats.

- Lightweight cryptography has the function of enabling the application of secure encryption, even for devices with limited resources.

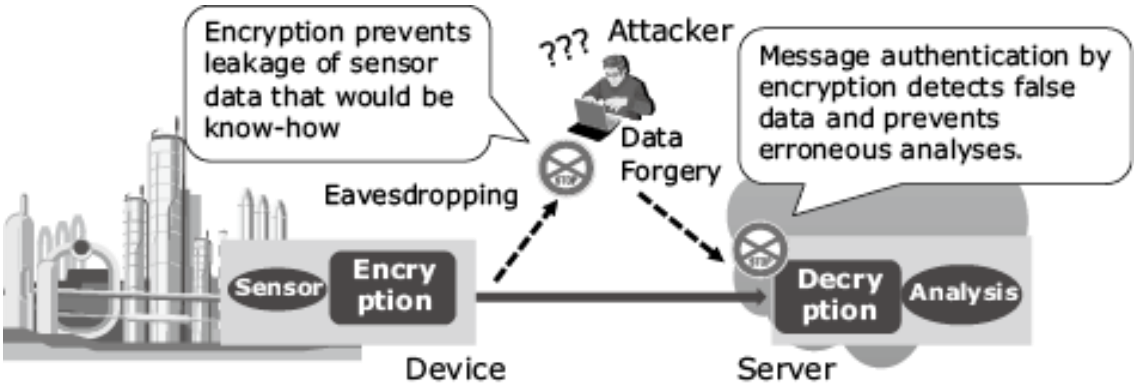


Fig.5.8 Encryption-based countermeasure against attack on data collection

Encryption is already applied as standard on the data link layer of communication systems such as the cellphone. Even in such a case, encryption in the application layer is effective in providing end-to-end data protection from the device to the server and to ensure security independently from the communication system. Then encryption must be applied at the processor processing the application and on unused resources and hence should desirably be as lightweight as possible.

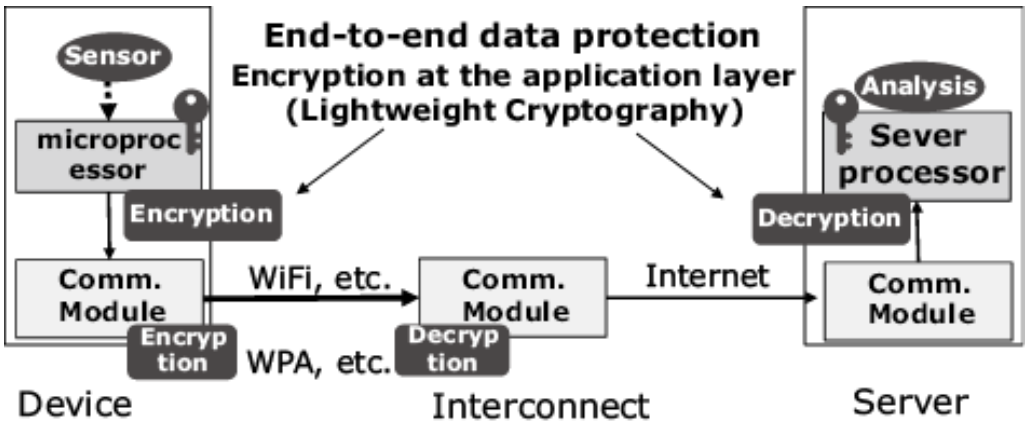


Fig.5.9 Example of lightweight cryptography applications

4. Lightweight Cryptography

The following factors on the implementation are required for lightweight cryptography.

- Size (circuit size, ROM/RAM sizes)
- Power
- Power consumption
- Processing speed (throughput, delay)

The first factor determining the possibility of implementation in a device is the size. Power is especially important with the RFID and energy harvesting devices while the power consumption is important with battery-driven devices. A high throughput is necessary for devices with large data transmissions such as a camera or a vibration sensor, while a low delay is important for the real-time control processing of a car-control system, etc.

Since the power is greatly dependent on the hardware such as the circuit size or the processor in use, the size becomes the reference point for the lightness of the encryption method and also for the power. The power consumption is dependent on the processing speed because of the execution time, so the number of computations that determines the processing speed becomes the index of the lightness. The throughput depends greatly on the parallel processing capability.

With regard to security, since encryption is the technological point of origin of the overall system security the lightweight cryptography needs to adopt a method that is evaluated as having a sufficient security level of modern cryptography. Even when the block length and/or secret key length are set shorter than for the standard cryptography by prioritizing the ease of implementation (such as via 64-bit block and 80-bit secret key, for example,) it is still required to correctly apply a proven method.

- Cryptography can roughly be divided into symmetric key and public key (asymmetric key) cryptographies. The symmetric key cryptography uses the same secret key for encryption and decryption.
- With the processing that is relatively lightweight, it is used in data encryption and authentication. On the other hand, public key cryptography uses a secret key in decryption and a public key different from the secret key in encryption, and it is quite difficult to guess the secret key from the public key.
- The computational complexity of the public key cryptography is typically as high as more than 1,000 times that of the symmetric key cryptography, but this technology is used in sharing the secret key used in symmetric key cryptography and the digital signature, thanks to the asymmetrical property.
- With a system such as a plant or car- control system, it may be possible to embed the secret keys shared by the devices in advance. In such a case, secure and efficient data protection can be implemented using symmetric key cryptography alone. On the other hand, with a system that performs encrypted communications dynamically with unspecified parties such as an inter-vehicle communication system, the use of public key cryptography is effective.

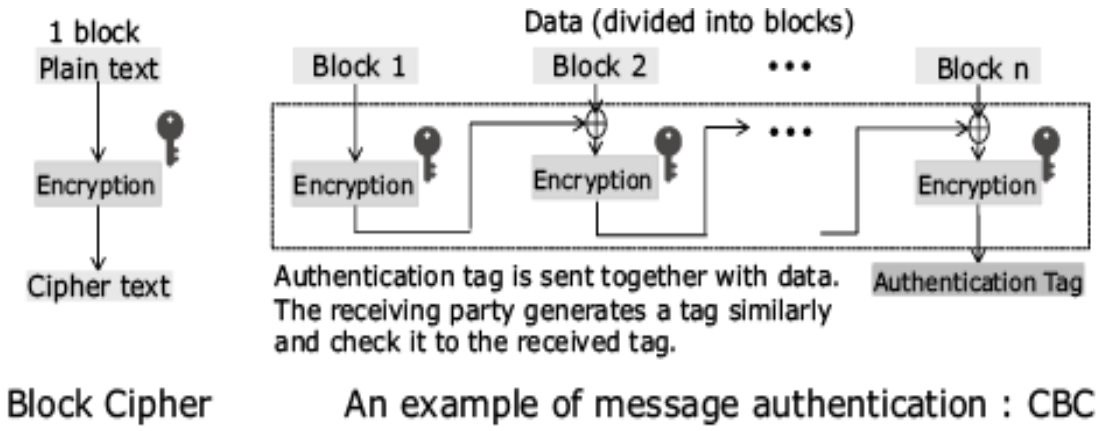


Fig.5.10 An Example of block cipher mode of operation

- We focus mainly on the symmetric key cryptography that can be widely applied to devices that are subject to severe resource restrictions. The symmetric key cryptography consists of core functions such as block or stream ciphers (cryptographic primitives) and methods to apply the core function to a packet called the block cipher mode of operation for encryption and/or authentication. Fig. 5.10 shows an example of the block cipher mode of operation used for the authentication (called CBC-MAC: cipher block chaining message authentication code). To render a cryptography lightweight, it is required to improve the efficiency of the block cipher mode of operation as well as the cryptographic primitives.

5. Privacy Concerns in IOT:

- Internet of Things privacy is the special considerations required to protect the information of individuals from exposure in the IoT environment, in which almost any physical or logical entity or object can be given a unique identifier and the ability to communicate autonomously over the Internet or similar network.
- As endpoints (things) in the IoT environment transmit data autonomously, they also work in conjunction with other endpoints and communicate with them. Interoperability of things is essential to the IoT's functioning so that, for example, networked elements of a home work together smoothly.
- The data transmitted by a given endpoint might not cause any privacy issues on its own. However, when even fragmented data from multiple endpoints is gathered, collated and analyzed, it can yield sensitive information.
- The idea of networking appliances and other objects is relatively new, especially in terms of the global connectivity and autonomous data transfer that are central to the Internet of Things. As such, security has not traditionally been considered in product design, which can make even everyday household objects points of vulnerability.

Key Privacy Issues:

- **Collection, use and disclosure of IoT data:**
    - The data collected from IoT devices generally comes from sensors including microphones, accelerometers and thermometers. Data from sensors such as these is often highly detailed and precise. This granularity allows additional information to be easily created through machine learning inferences and other analysis techniques that can yield results that would not be possible with coarser data.
    - In addition, devices with multiple sensors, or multiple devices in close proximity, can combine their data in a process known as sensor fusion, which allows for more accurate and specific inferences that would not be possible with data from a single sensor.
    - If private organisations that provide IoT devices or services can access IoT data, there is a risk that they could use or disclose personal information for purposes that are not in the public interest, such as for profiling, targeted advertising or sale of the data to data brokers.
    - IoT devices can also allow practices that were previously only possible online to occur in physical spaces.
- **De-identification of IoT data:** The data collected by large IoT ecosystems like smart cities can be valuable for a range of purposes such as research or informing policy decisions. A common way to maximise the value of this data is to make it publicly available online. However, it is generally impermissible for datasets that include personal information to be made publicly available. The simplest way to ensure personal information is not included in a dataset is to allow individuals to remain anonymous by never collecting information that can identify them.
- **Consent:** Consent is a common basis for organisations to use and disclose personal information. However, valid consent generally requires more than getting a user to click 'I agree'. Meaningful consent has five elements: capacity, voluntary, current, specific and informed. The IoT challenges each of these elements.
- **Capacity:** An individual must be capable of giving consent for it to be valid. A common reason that an individual does not have the capacity to consent is because they are a minor. For IoT devices targeted to children, such as smart toys, or devices designed to monitor children, such as IoT tracking wristbands, an authorised representative such as a parent or guardian may consent on behalf of the child.
- **Voluntary:** Consent must be freely given in order to be meaningful. It must be a genuine choice. If an individual must choose between giving consent or not being able to use a device they have purchased, then that consent may not be voluntary. If accepting terms and conditions is a prerequisite to using an IoT device and refusing those terms will result in an inoperable device, it is likely not a genuine choice and therefore not meaningful consent.
- **Current:** Consent cannot be assumed to last indefinitely. The seamless and unobtrusive nature of IoT devices often makes it easy for people to forget they are there or what they are doing. Smart devices such as watches, doorbells, and billboards can blend into

the background as they collect, use and share personal information. Additionally, the way that an IoT device operates can also change over time.

- **Specific:** Consent must be specific to an identified purpose. It is not possible for an individual to provide meaningful consent to their personal information being used for a vague or broad purpose. If an IoT device provides unspecific information, users can develop misconceptions about what happens to personal information collected by the device and can be surprised when they discover what they actually agreed to.
- **Informed:** An individual must have full knowledge of all relevant facts for their consent to be meaningful. This includes, but is not limited to:
- what information will be collected, used or disclosed;
- the purpose for collecting the information; and
- who the information will be shared with and what they will do with
- The complex interactions between the functions of an IoT device – and the interactions between different devices, organisations and third parties – can make it difficult for users to develop mental models for visualising how their devices operate, what information they collect, and how they use and disclose that information.
- **Dependency on vendors:** Organisations and individuals who use IoT devices are often dependant on the vendors or manufacturers of those devices to handle security and privacy issues through the delivery of software or firmware updates to fix security vulnerabilities. Sometimes they are reliant upon vendors to ensure that collected personal information is sufficiently de-identified before it is shared. However, vendors often focus on specific parts of IoT ecosystems and will not necessarily consider how those ecosystems function holistically.
- **Interoperability:** The rapid expansion of the IoT in recent years has led to the development of many different kinds of devices, Application Programming Interfaces (APIs) infrastructure, data formats, standards and frameworks. An API is a way for a computer to communicate with another computer, or for a person to interrogate or instruct a computer and get a result. This has caused significant interoperability issues, in that devices, software and data from one vendor often do not work with devices, software and data from other vendors.
- **Managing IoT devices:** When management options are not centralised or interoperable, the resources required to manage devices increases as the number and diversity of devices increases. If an organisation had thousands of devices from dozens of manufacturers, it would be near impossible to effectively manage them individually. This issue can also apply to consumer devices, where it is common for devices to be managed by smartphone apps. If a person owns 10 IoT devices, they may require 10 different apps to manage them, likely leading to those devices being effectively unmanaged.
- **Accountability**
- **Transparency**

Module:6 IoT Platforms for Usecase Development

1. Open source IoT platforms and services

- An IoT platform is the middleware and the infrastructure that enables end-users to interact with smart objects. They function as the software bridge between the hardware and application layers.
- The IoT platform orchestrates the movement of data between the IoT device and the IoT application. Further, it provides application-level capabilities for humans to interact with the IoT system.
- IoT platforms are a type of middleware, a mediator between the hardware and application layers.
- In heterogeneous IoT environments, platforms support integration between third-party apps, sensors, legacy equipment, and other connected devices, allowing organizations to manage all activities in the same straightforward manner.
- Managing a growing IoT ecosystem brings various challenges as organizations transition from small-scale pilots to fully-fledged IoT deployments.
- Thousands of IoT endpoints, the data they generate, the analyses of IoT data, integrations with the cloud as well as other systems have to be managed and maintained.
- Typical concerns include multiprotocol connectivity, interoperability, device discovery and device management, customization, scalability, data management, privacy and security concerns, and in some cases, the cloud service and automatic context detection.
- IoT platforms help in such scenarios as they abstract the hardware while simplifying deployment.
- Improve resiliency, maximize scalability, increase reliability, reduce cost, and minimize latency.
- IoT platform solutions vary considerably, they generally offer the following capabilities:
    - Connect devices, sensors, machines, etc.
    - Handle multiple types of communications protocols.
    - Secure devices, data, and networks.
    - Capture, analyze, and organize data collected from various devices.
    - Integrate IoT systems with existing business processes like inventory management, CRMs, etc.
    - Industrial IoT platforms should reduce costs and associated risks of developing and maintaining applications.

Why IOT platform?

- Managing a growing IoT ecosystem brings various challenges as organizations transition from small-scale pilots to fully-fledged IoT deployments.
- Thousands of IoT endpoints, the data they generate, the analyses of IoT data, integrations with the cloud as well as other systems have to be managed and maintained.
- Typical concerns include multiprotocol connectivity, interoperability, device discovery and device management, customization, scalability, data management, privacy and

security concerns, and in some cases, the cloud service and automatic context detection.

- IoT platforms help in such scenarios as they abstract the hardware while simplifying deployment.
- Improve resiliency, maximize scalability, increase reliability, reduce cost, and minimize latency.

Types of IoT Platforms

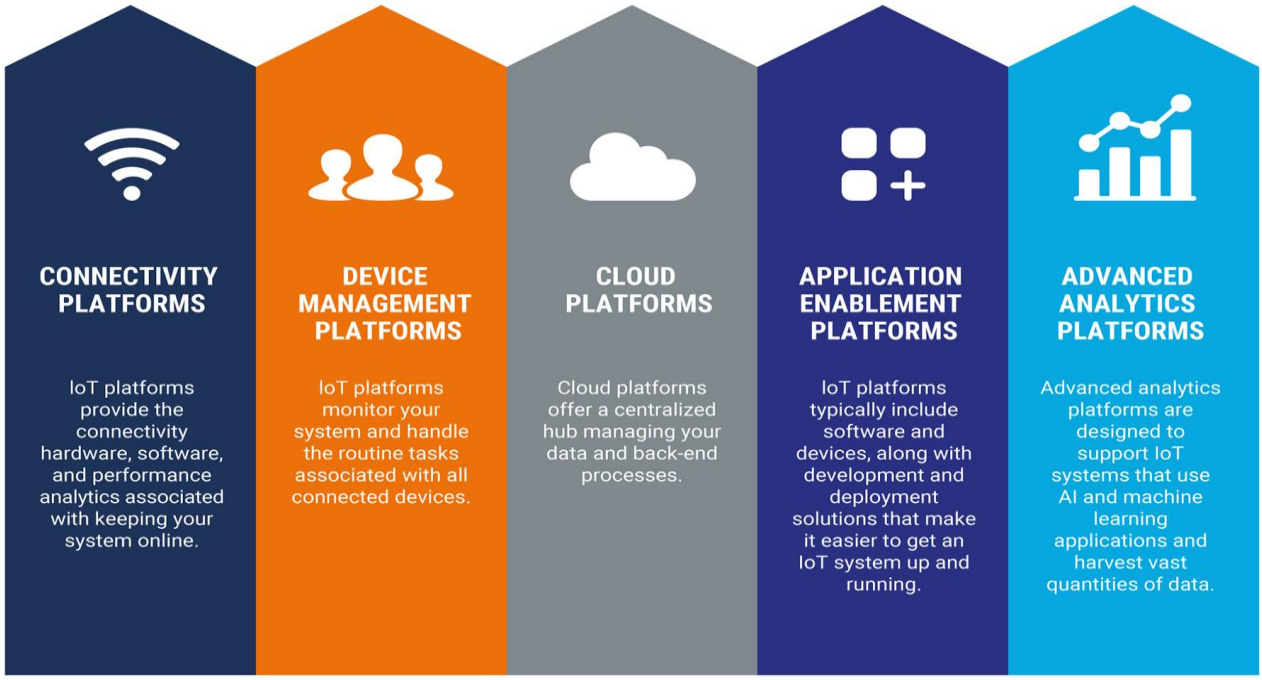


Fig.6.1 Types of IoT platform

- *End-to-end IoT platforms* facilitate the handling of millions of simultaneous device connections by providing hardware, software, connectivity, security and device management tools. In addition, these platforms provide OTA (over the air) firmware updates, cloud connectivity and device management to facilitate device monitoring.
- *Connectivity platforms* provide cheap and low power connectivity solutions via Wi-Fi and cellular technology.
- *Cloud platforms* reduce the complexity of building complex network stacks and provide back-end facilities for device monitoring.
- *Data platforms* based on IoT platforms provide numerous tools for data routing, and facilitate management and visualisation of data using data analytics tools.

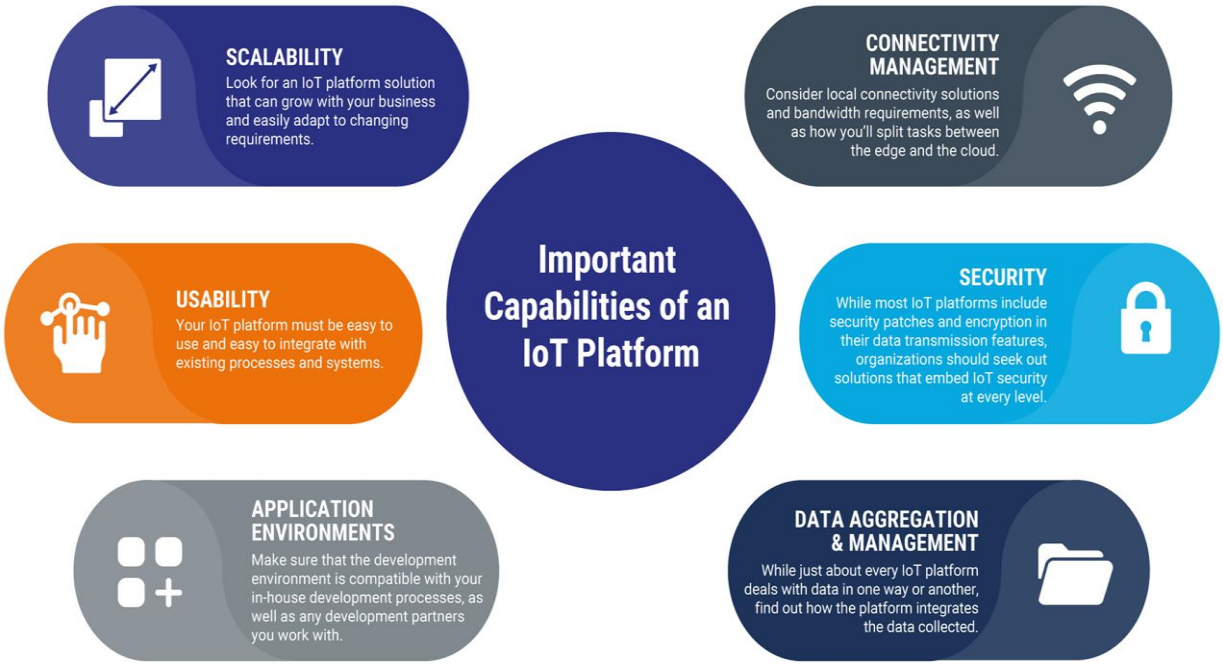


Fig.6.2 Capabilities of IoT platform

How to validate an effective IoT platform

- *Scalability:* The IoT platform should be scalable enough to accommodate growing needs without and efficient backup to prevent all sorts of errors.
- *Reliability:* The IoT platform should handle all sorts of failover and have disaster recovery options to keep the overall system up and running without any issues.
- *Customisation:* IoT platforms should support lots of customisations and be well integrated with cloud services. They should consist of APIs, and have extensive libraries as well as strong integration with other platforms to enhance the core functionality of the system with the programmer's customised code.
- *Operations:* The IoT platform's operations should not be hidden; rather the end user should be greeted with a centralised interface which gives all the details with regard to system statistics, hardware information and the modules or services.
- *Protocols:* The most important and foremost protocol to facilitate IoT is MQTT. The IoT platform should support the MQTT protocol to provide all sorts of communication from sensors to the cloud. Almost all cloud service providers use MQTT. These include Microsoft Azure, Amazon AWS, Google Cloud, etc. The IoT platform should support the API over WebSockets, REST and CoAP.
- ***Hardware support:*** Hardware facilitates low-end devices by using MQTT or CoAP protocols and, overall, the systems should be flexible and powerful enough to run programs written in any language.
- ***Cloud technology:*** The IoT platform should be flexible enough to support all operations without any hiccups, whether in the cloud, on premise or hybrid.
- ***Technical support:*** IoT platforms should have strong back-end technical support, and service providers should ensure regular updation of the platform with all the basic upgrades, security patches and bug fixes.

- ***Systems architecture:*** An IoT platform should deploy production-grade, well-supported frameworks, tools and languages, making the platform easy to operate and flexible enough to support all sorts of custom implementations.
- ***Security:*** This is another significant parameter for an IoT platform. It is very important to facilitate encrypted communication and the platform should have authorisation capabilities. It should be secure enough to defend the end user data from all sorts of backdoors, DDoS attacks, website vulnerabilities, malware and even rootkits.
- ***Operational expenses:*** When evaluating an IoT platform, the end user must consider parameters like pay per node, pay per active device, pay per message, the price of premium features, the price of support, etc.

2. Communication API's- REST and Websocket

- An API is an interface used by programs to access an application.
- It enables a program to send commands to another program and receive replies from the app.
- IoT APIs are the interface points between an IoT device and the Internet and/or other network components.

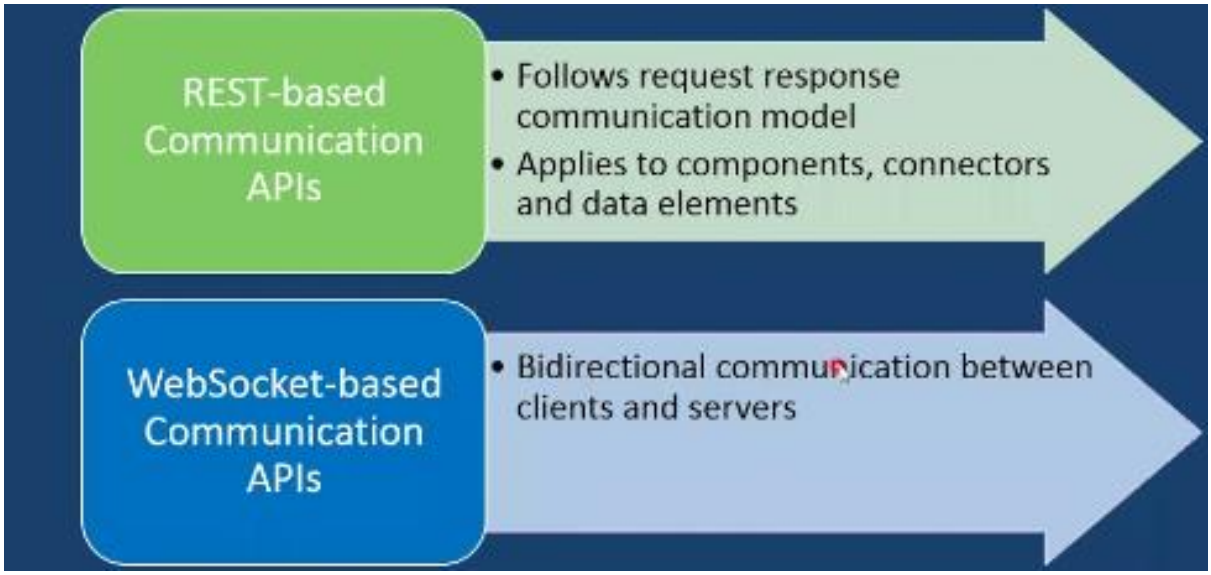


Fig.6.3 Communication API's

REST-based Communication APIs

- Representational state transfer (REST) is a set of architectural principles by which you can design Web services the Web APIs that focus on the system's resources and how resource states are addressed and transferred.
- URIs(example:- example.com/api/tasks) are used to depict resources in the RESTful web service.
- Client tries to access these resources via URIs using commands like GET, PUT, POST, DELETE and so on that are defined by HTTP.
- In response, the server responds with a JSON object or XML file.
- The REST APIs follow the request-response model.

- REST APIs follow the request response communication model.
- The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system.
- The motivation for REST was to capture the characteristics of the Web which made the Web successful.
    - URI Addressable resources
    - HTTP Protocol
    - Make a Request – Receive Response – Display Response
- Exploits the use of the HTTP protocol beyond HTTP POST and HTTP GET HTTP PUT, HTTP DELETE

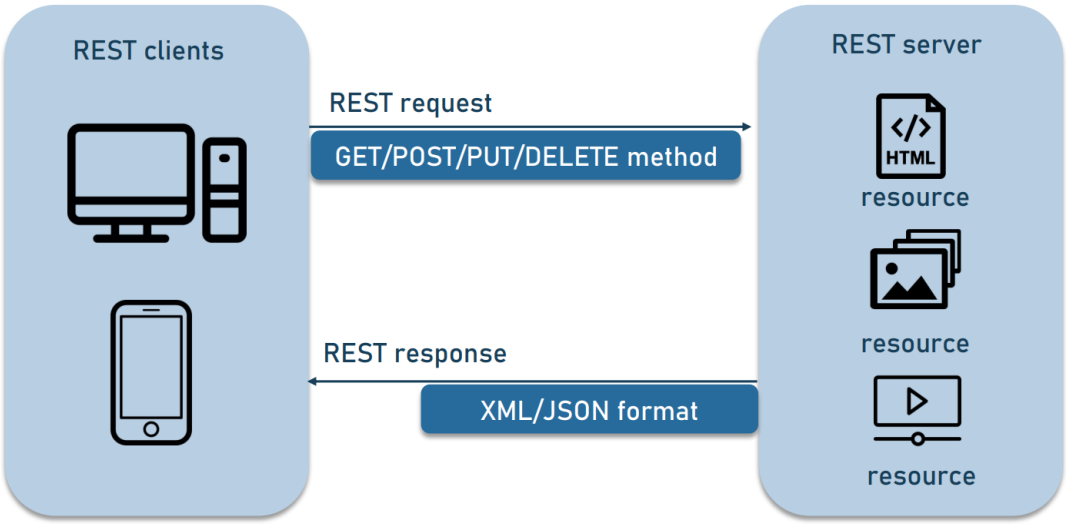


Fig.6.4 Rest based API's

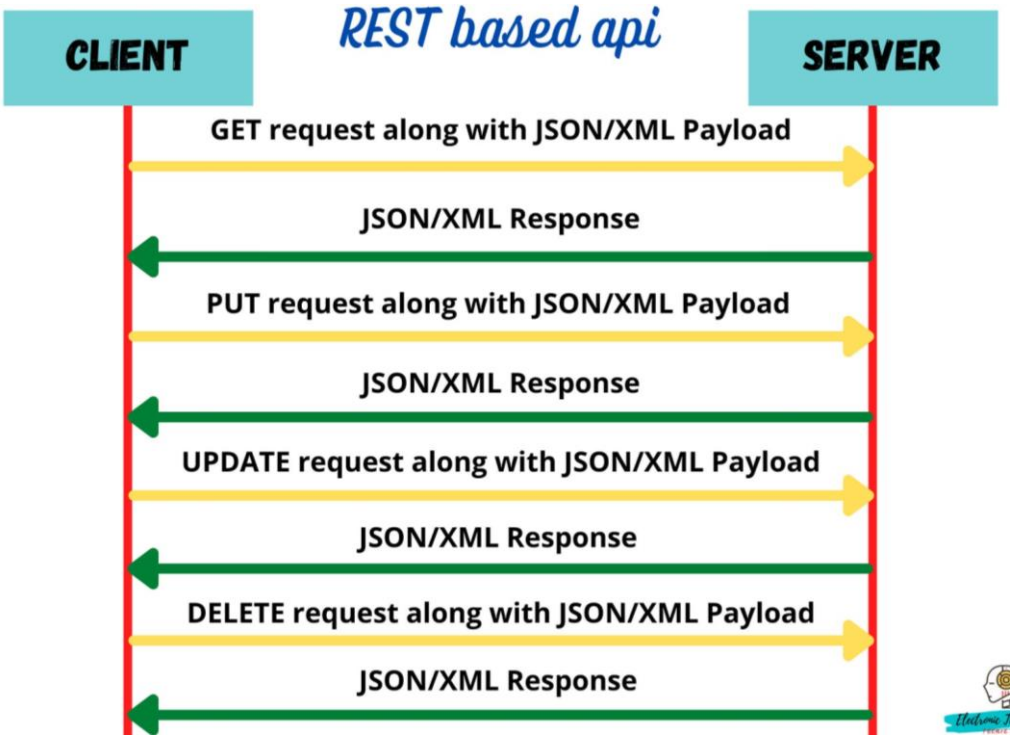


Fig.6.5 Rest based API's-Communication

Constraints in Rest based API

- Client-Server: The principle behind client-server constraint is the separation of concerns. Separation allows client and server to be independently developed and updated.
- Stateless: Each request from client to server must contain all the info. Necessary to understand the request, and cannot take advantage of any stored context on the server.
- Cache-able: Cache constraint requires that the data within a response to a request be implicitly or explicitly labeled as cache-able or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.
- Layered System: constraints the behavior of components such that each component cannot see beyond the immediate layer with which they are interacting.
- User Interface: constraint requires that the method of communication between a client and a server must be uniform.
- Code on Demand: Servers can provide executable code or scripts for clients to execute in their context. This constraint is the only one that is optional.

Advantages

- Simplicity: REST APIs are relatively simple to design and implement, making them a popular choice for building APIs for web applications.
- Flexibility: REST APIs can be used to support a wide range of applications and services, from simple web applications to complex enterprise systems.
- Caching: REST APIs can leverage caching to improve performance and reduce server load.
- Stateless: REST APIs are stateless, meaning that each request is processed independently of any previous requests, making them easy to scale and distribute.

Disadvantages

- Limited real-time support: REST APIs do not support real-time communication between the server and client, making them less suitable for applications that require real-time updates.
- Performance overhead: REST APIs require more overhead than WebSocket APIs, as each request and response must contain all the necessary information to complete the request.
- Complexity: REST APIs can be complex to design and implement for large, distributed systems.

Websocket-based Communication APIs

- WebSocket protocol is a TCP-based network protocol. It defines how data is exchanged between networks.
- Reliable and efficient used by all clients.
- TCP establishes communication between two endpoints, which are referred to as sockets. This allows data to be transferred in both directions.
- Two-way connections like WebSocket (sometimes written as Web Socket) allow data to be exchanged in both directions simultaneously. The advantage of this is that data can be called up more quickly.
- WebSocket in particular enables direct communication between a web application and a WebSocket server.
- WebSocket is useful **when fast connections.**
- Examples include **live chats** on support websites, news tickers, stock tickers, **messaging apps,** and real-time games. For most companies today, standard connection requests are no longer sufficient.
- **Social media** sites can benefit from WebSocket too, because it allows live interaction and instant messaging between users.
- WebSocket is a logical choice for any application that requires a high-speed, low-latency connection.

Architecture of Web socket

- Websocket APIs enable bi-directional and duplex communication between customers and servers.
- Unlike REST, There is no need to set up a connection every now and then to send messages between a client and a server.

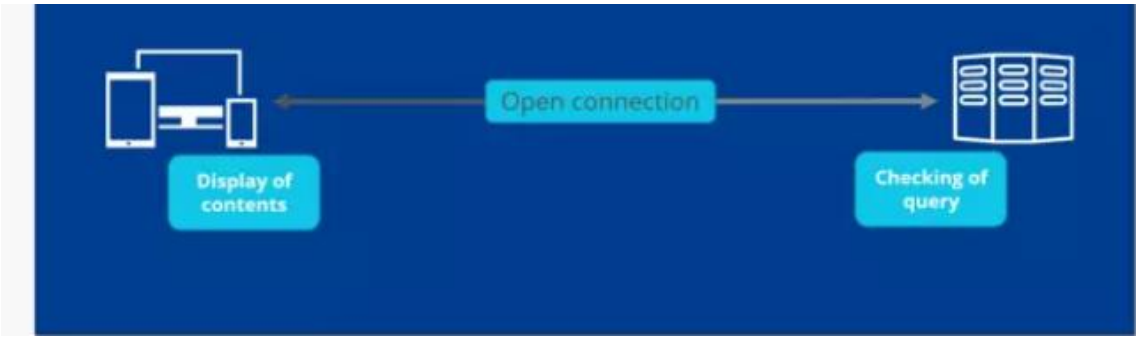


Fig.6.6 Websocket based API's-Communication

- It works on the principle of the exclusive pair model.

- Once a connection is set up, there is a constant exchange of messages between the client and the server.
- All we need is to establish a dedicated connection to start the process. The communication goes on unless the connection is terminated.
- It is a stateful type.
- Due to onetime dedicated connection setup, there is less overhead, lower traffic and less latency and high throughput.
- Web socket is the most suitable IoT Communication APIs for IoT System.
- WebSocket APIs allow bidirectional, full duplex communication between clients and servers.
- WebSocket APIs follow the exclusive pair communication model.

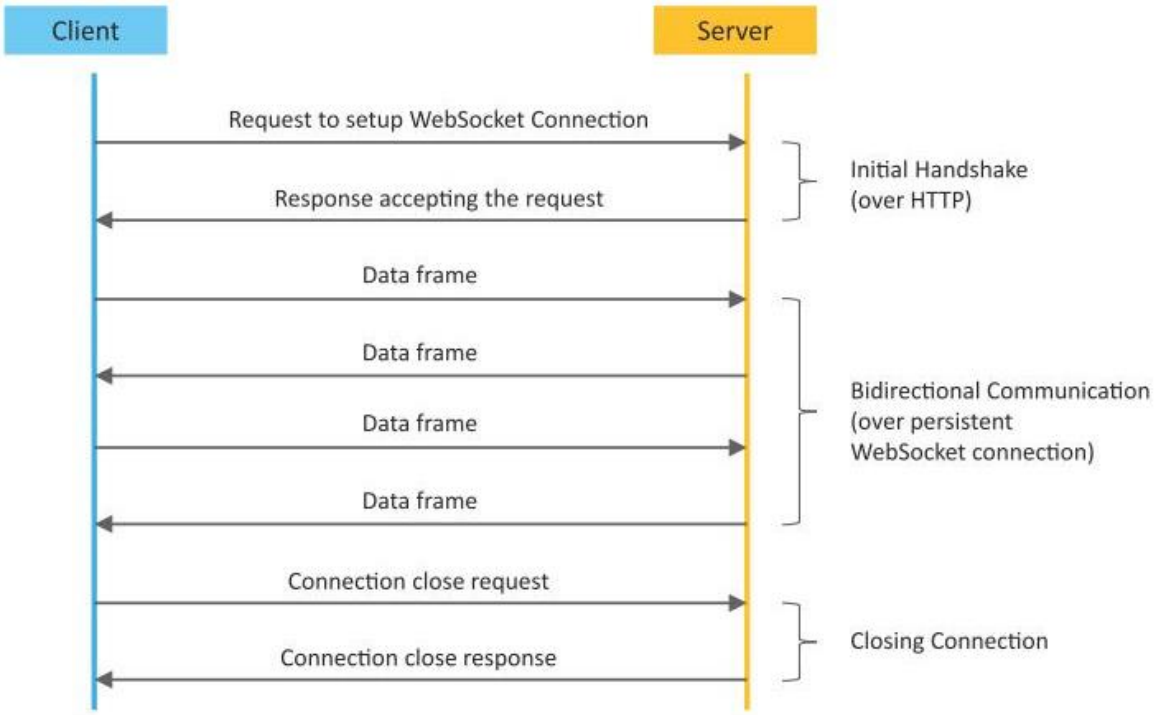


Fig.6.7 Websocket Protocol

- Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server.
- Once the connection is setup it remains open until the client sends a request to close the connection.
- Client and server can send messages to each other after connection setup.

Advantages:

- Real-time communication: WebSocket APIs enable real-time communication between the server and client, making them ideal for applications that require real-time updates.
- Efficiency: WebSocket APIs are more efficient than REST APIs for real-time applications, as they use a persistent connection to enable bidirectional communication.
- Scalability: WebSocket APIs are highly scalable, as they can support thousands of connections per server.
- Reduced overhead: WebSocket APIs have lower overhead than REST APIs, as they use a single connection to transmit data.
- With conventional HTTP connections, problems may arise because the client always loads the entire HTML page.
- AJAX technology was developed to address this issue. However, this still wasn't a perfect solution, because it only allows one-way communication, which leads to considerable delays that are no longer compatible with today's applications, in particular live chats.
- By establishing a two-way data connection, WebSocket enables direct contact with the browser, which reduces loading times.
- In a live support chat, this means messages are displayed instantly.

Disadvantages:

- Complexity: WebSocket APIs are more complex to design and implement than REST APIs, requiring additional programming skills and knowledge.
- Security: WebSocket APIs can be vulnerable to security threats if not properly secured.
- Compatibility: WebSocket APIs are not supported by all browsers, requiring fallback mechanisms for older browsers.

Comparison between Websocket and Rest Protocol

The basis Of Comparison	WebSocket	REST
HTTP	The use of HTTP occurs in the initial connection.	HTTP is a common protocol in RESTful web services.
Communication	Bi-directional in nature.	Uni-directional in nature.
Nature	Socket-based concept.	Resources-based concept, rather than commands.
Scenario	Real-time chat application.	Lots of getting requests.
Dependency	Rely on IP address and port number.	Based on the HTTP protocol and uses HTTP methods to relay data.
Cost	The cost of communication is lower.	The cost of communication is comparatively higher than WebSocket.
Performance	Better with high loads.	Great for occasional communication.
State	WebSocket is a stateful protocol.	REST is based on HTTP, which is a stateless protocol.

3. Scalability of IoT Solutions

- Scalability is one of the most crucial criteria you should look for when considering an enterprise IoT solution.
- The capability of a system to manage an increasing quantity of work by adding extra resources; remains a sticking point for many developers attributable to issues inherent to IoT technology.

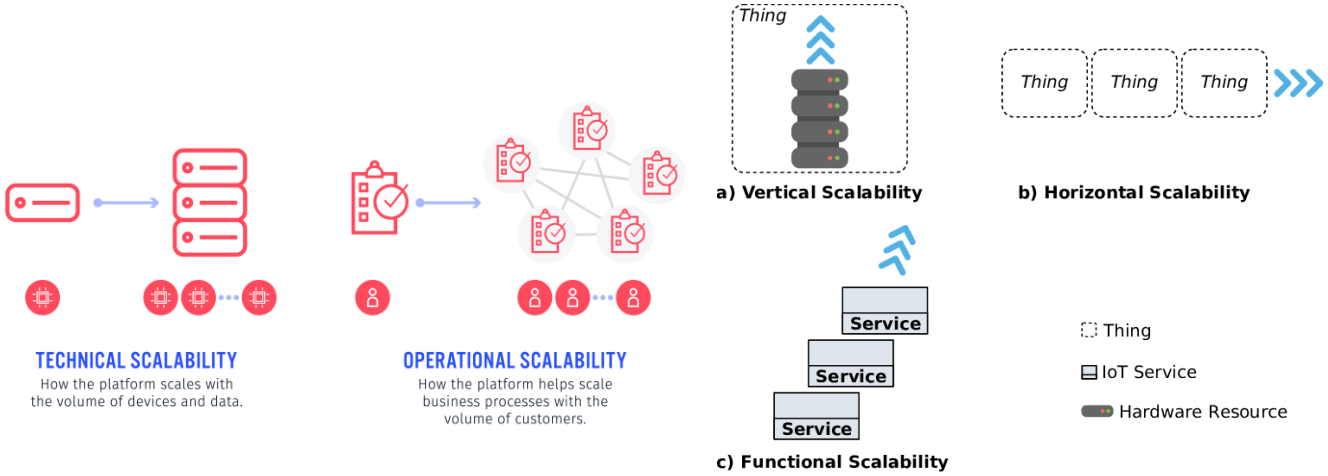


Fig.6.8 Scalability of IoT solutions-Types

- IoT scalability refers to the ability to go from prototype to production in a seamless way.
- In IoT deployments, the ability to scale up and down depending on the amount of data their fleet of devices refers to Scalability of IOT.
- To have an optimum scalable IOT solution, the infrastructure, cloud, and connectivity layers can all scale with requirements.
- That software-layer scalability is critical, but because IoT encompasses hardware, the ability to quickly iterate on your hardware design, deploy that hardware around the world on different cellular networks, and stay within the bounds of the certification requirements is equally important.
- This process often presents challenges because it can involve multiple parts: different vendors, pieces of hardware, and cellular carriers. Scaling into new geographic areas usually involves finding new carriers and seeking regional certifications, draining time and resources in the process.

Why Scalability Issues?

1. Lack of a planning phase:

- When launching an IoT project, there's a requirement for a solid Proof of Concept (POC) and planning phase.

- Since IoT projects come with a range of technological and organizational issues, many aspects need to be addressed at the planning stage.
- However, these features are often overlooked or neglected since they appear trivial at first.

2. Compatibility with future devices:

- Even the tiniest devices may modify how they interact with IoT communication protocols.
- This implies that the procedure for adding or replacing new appliances in five years may be radically different from now.
- This would involve modifying the architecture, the software, or the protocols themselves.

3. Budgetary constraints:

Scalability In IoT: IoT developers might be resistant to spending money now to prevent difficulties in the future.

Scalability Challenges:

- Network Security: The expansion in the volume of IoT devices is accompanied by an urgent need to secure the network against malicious attacks. We must define new protocols and incorporate encryption algorithms to enable high throughput.
- Privacy: Ensuring the anonymity and individuality of IoT users must be critical for any IoT provider. And this will only get more challenging as more IoT devices enter an ever-expanding network.
- Governance: Without a proper governance system for trust management between the user and provider, a breach of confidence between entities is imminent. This is one of the leading research challenges in IoT scalability.
- Access Control: With the low bandwidth between the IoT device and the internet, low power usage, and distributed architecture, access control will be a challenge. So, conventional access control systems for admins and end-users must be refurbished as and when new IoT scalability challenges arrive.
- Big Data Generation: IoT systems make programmed judgments based on categorized data compiled from multitudinous sensors. As data volume increases disproportionately with an increasing number of devices, scaling will present the challenge of large silos of Big Data. Determining the relevance of this data will require unprecedented computing power.

Tips for IoT at Scale

1. Start Small and Build Out from There: Firstly, if you're looking to scale responsibly, it may be a good idea to limit your growth over time.

- If you start your project with a few manageable devices, periodically adding one or two over time, you won't have to change your entire infrastructure and it won't take much time to accommodate them.

2. Use Simple Architecture: Choosing the right architecture for your project from the very beginning leads to way fewer problems in the future. Moreover, it's important to choose an option that has the future in mind. If you start your project with a few manageable devices, periodically adding one or two over time, you won't have to change your entire infrastructure and it won't take much time to accommodate them.

3. Use a Decentralized AEP Platform: Another way to effectively scale up your IoT project is by using a decentralized IoT Application Enablement Platform (AEP).

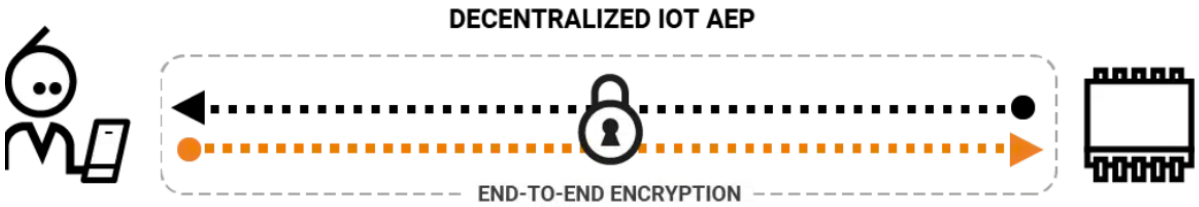


Fig.6.9 Decentralized AEP Platform

Module:7 IoT Verticals

1. Roadmap for developing complete IoT solutions

Smart Systems: Improve operational performance through seamless system integration.

Smart Buildings: Monitor and manage multiple eco-friendly properties in real-time.

Industrial IoT: Reduce human error, improve process accuracy, and reduce overall operational costs.

Transportation & Logistics: Enhance in-transit visibility while accelerating delivery time and operational efficiency.

Smart Oil & Gas: Eliminate on-site visits and increase regulatory compliance, reliability and efficiency.

Telecom: Develop new revenue streams and cross sell services while enhancing customer loyalty.

Fig.7.1 IoT Verticals

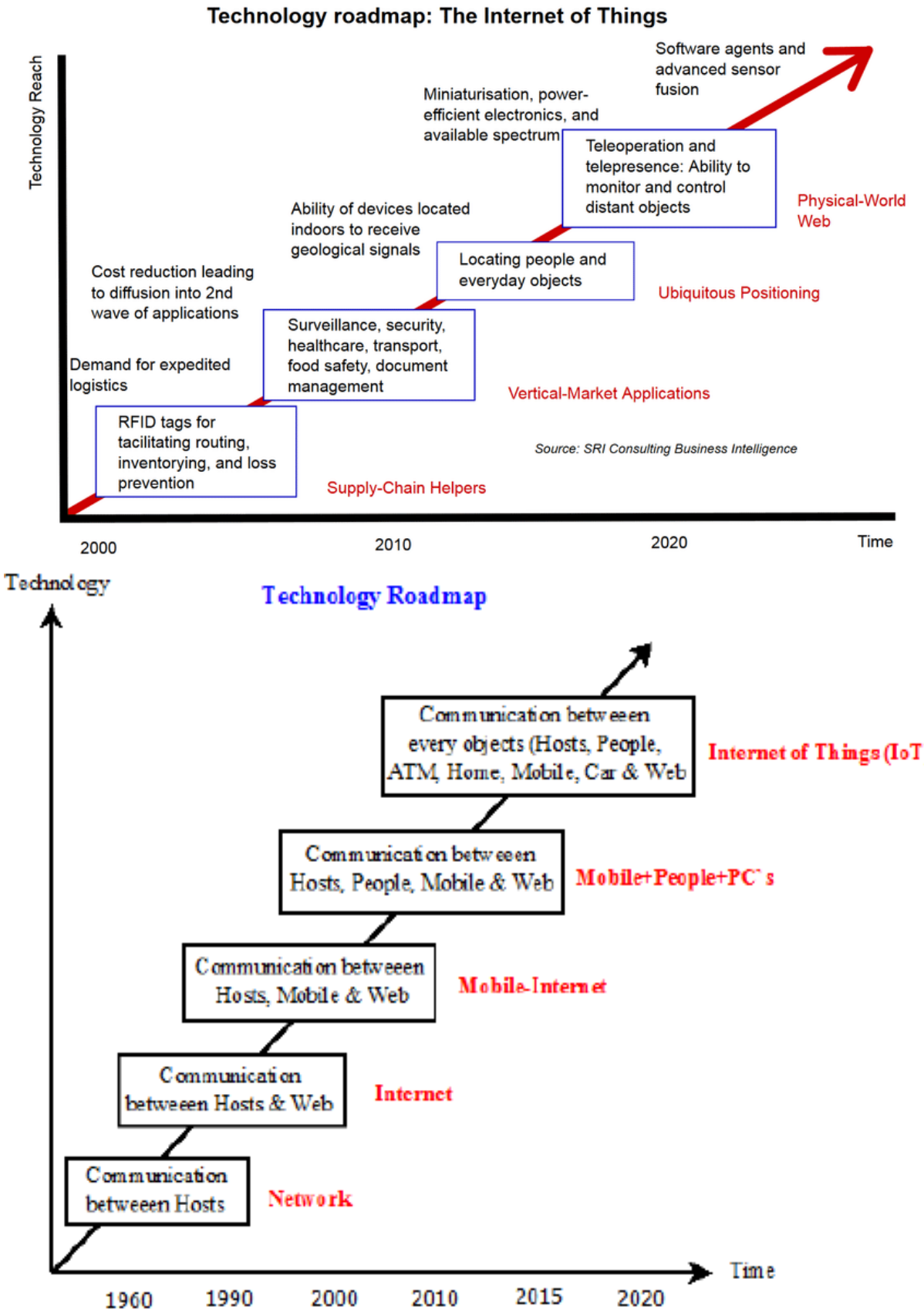


Fig.7.2 IoT Roadmap

2. Smart Cities

Smart city technology is increasingly being used to improve public safety, from monitoring areas of high crime to improving emergency preparedness with sensors. For example, smart sensors can be critical components of an early warning system before droughts, floods, landslides or hurricanes.

Smart buildings are also often part of a smart city project. Legacy infrastructure can be retrofitted and new buildings constructed with sensors to not only provide real time space management and ensure public safety, but also to monitor the structural health of buildings. Sensors can detect wear and tear, and notify officials when repairs are needed. Citizens can help in this matter, notifying officials through a smart city application when repairs are needed in buildings and other public infrastructure, such as potholes. Sensors can also be used to detect leaks in water mains and other pipe systems, helping reduce costs and improve the efficiency of public workers.

Need for Smart Cities:

- The primary goal of a smart city is to create an urban environment that yields a high quality of life to its residents while also generating overall economic growth. Therefore, a major advantage of smart cities is their ability to facilitate an increased delivery of services to citizens with less infrastructure and cost.
- As the population within cities continues to grow, it becomes necessary for these urban areas to accommodate the increasing population by making more efficient use of their infrastructure and assets. Smart city applications can enable these improvements, advance city operations and improve the quality of life among residents.
- Smart city applications enable cities to find and create new value from their existing infrastructure. The improvements facilitate new revenue streams and operational efficiencies, helping governments and citizens save money.
- Smart city technologies also bring efficiencies to urban manufacturing and urban farming, including job creation, energy efficiency, space management and fresher goods for consumers.

How a smart city works

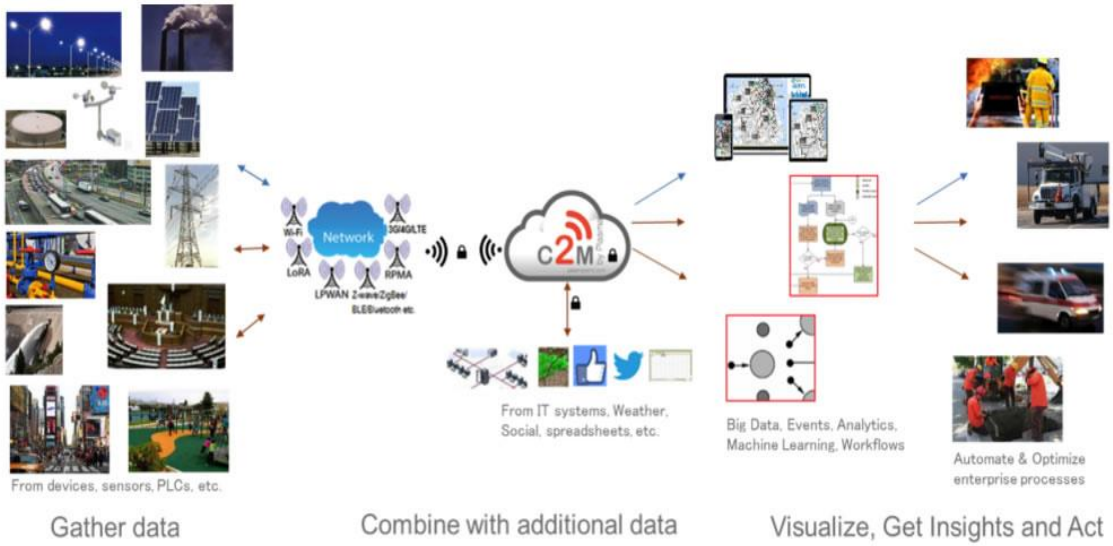


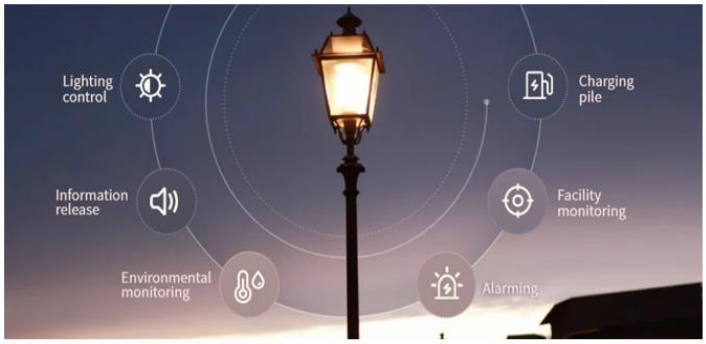
Fig.7.3 IoT Generalised Architecture used in Smart Cities

Smart cities utilize their web of connected IoT devices and other technologies to achieve their goals of improving the quality of life and achieving economic growth. Successful smart cities follow four steps:

1. Collection - Smart sensors throughout the city gather data in real time.
2. Analysis - Data collected by the smart sensors is assessed in order to draw meaningful insights.
3. Communication - The insights that have been found in the analysis phase are communicated with decision makers through strong communication networks.
4. Action - Cities use the insights pulled from the data to create solutions, optimize operations and asset management and improve the quality of life for residents.

Smart Cities Use cases

A smart street light system incorporates a cluster of streetlight lamps that can communicate with each other and provide lighting data to a local concentrator. It allows facility managers to remotely control street lights while keeping track of electrical power consumption in the lamps and in the driving circuits.



Advantages:

1.With The Help Of IoT, Street Lights Can Switch ON And OFF Automatically.
2.Maintenance Of Street Lights Using IoT Is Quite Less Which Leads To Cost Reduction.
3.Street Lights Using IoT Will Also Reduce Light Pollution.
4.Power Consumption Is Quite Low In These Street Lights Using IoT Which Also Leads To Energy Conservation.
5.No Large Manpower Is Required To Maintain These Street Lights Using IoT.

- An IoT based smart parking system, also known as a connected parking system, is a centralized management system that allows drivers to use a smartphone app to search for and reserve a parking spot.
- The system's hardware features sensors that detect available parking slots and communicate this information to all drivers in the area. This data is updated in real-time, which means drivers never have to worry about not finding an available space.
- In addition to helping drivers find a spot, the system also sends alerts about peak times and peak prices. The goal of these alerts is to help save drivers money while also reducing congestion.

How IoT Can Solve The Existing Parking Management Problem

01 Improving Car Control And Safety

02 Monitoring The Parking Space In Real-Time

03 Anticipating The Traffic Flow By Examining Parking Patterns

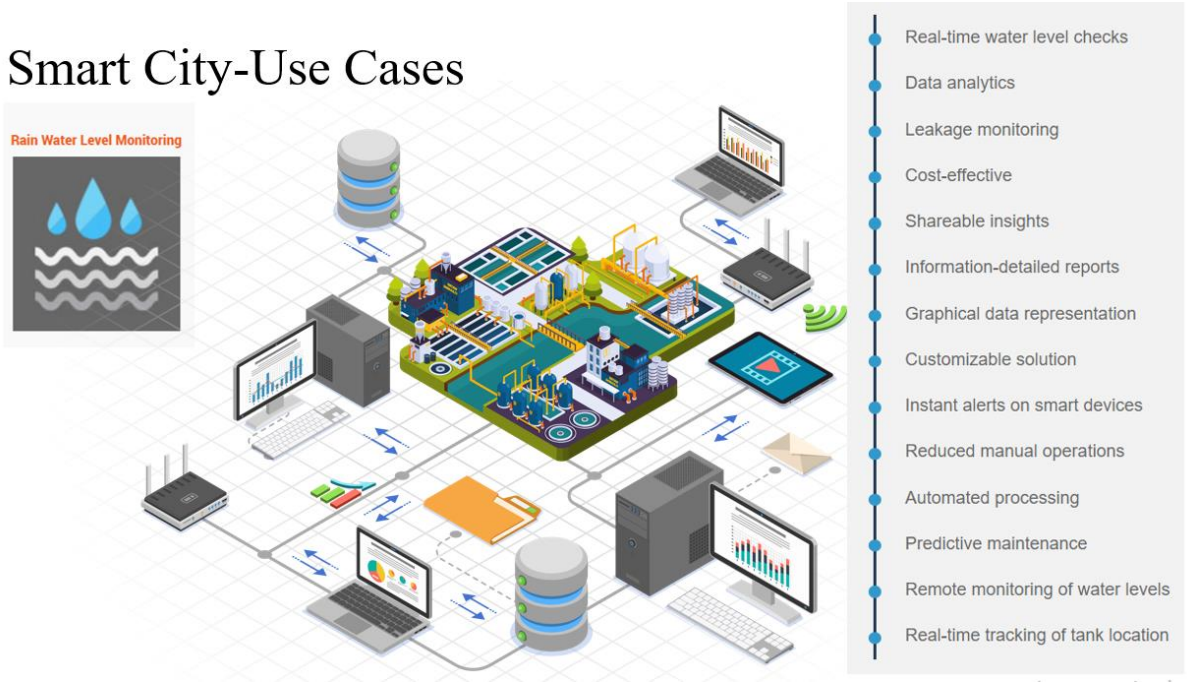
04 Making The Most Of Time And Space In A Congested Urban Environment

Smart City-Use Cases-Outfall Remote sensing

- Hazardous waste from sewage pipelines cause a risk of contaminating beaches, lakes and underground water tables. This poses a major environmental and health risk. Monitoring sewage leakage is a critical challenge faced by every city.
- To implement the monitoring of the sewage leaks, there is a need for a sensor grid with sensors for measuring level, water conductivity & accelerometer sensors.

Smart City-Use Cases

- IoT-based water level monitoring provides automatic detection of liquid levels from differently sized tanks or storage containers.
- It is a state-of-the-art system specially designed to inform the users about the real status of the liquid levels.
- It is meticulously designed to benefit the industrialists with IoT technology and improve the overall business productivity.
- IoT-based water level monitoring provides real-time autonomous detection of water levels and takes appropriate action based on the levels including overflowing, water depletion, and water usage.
- Deploying an autonomous system to keep a real-time check upon the water levels provides an effective solution to water-related challenges.



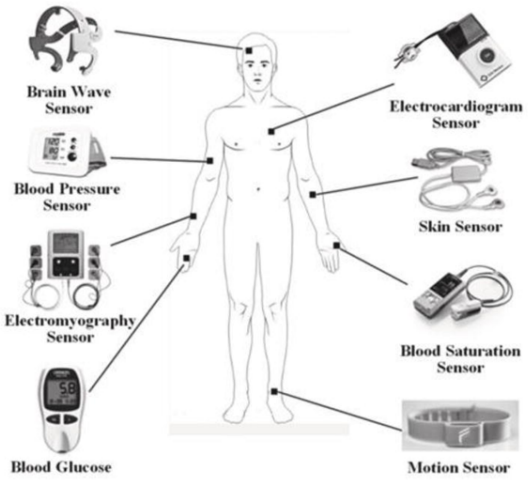
3. Healthcare

- Internet of Things (IoT)-enabled devices have made remote monitoring in the healthcare sector possible, unleashing the potential to keep patients safe and healthy, and empowering physicians to deliver superlative care.
- It has also increased patient engagement and satisfaction as interactions with doctors have become easier and more efficient.
- Furthermore, remote monitoring of patient's health helps in reducing the length of hospital stay and prevents re-admissions.
- IoT also has a major impact on reducing healthcare costs significantly and improving treatment outcomes.

Fig.7.4 IoT in Healthcare

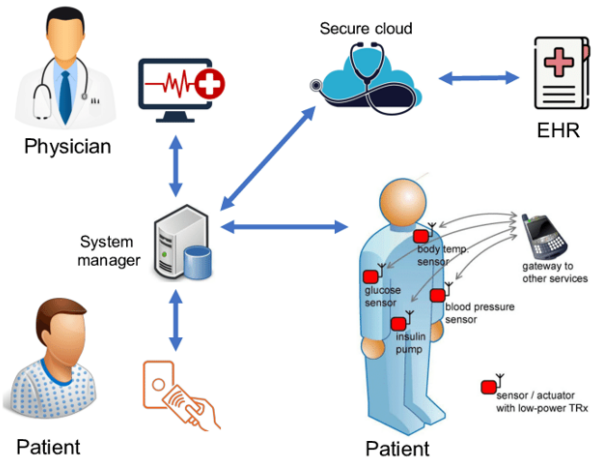
IoT in Health care

- **IoT for Patients** - Devices in the form of wearables like fitness bands and other wirelessly connected devices like blood pressure and heart rate monitoring cuffs, glucometer etc. give patients access to personalized attention.
- These devices can be tuned to remind calorie count, exercise check, appointments, blood pressure variations and much more.
- Enabling constant tracking of health conditions.
- This has a major impact on people living alone and their families.
- On any disturbance or changes in the routine activities of a person, alert mechanism sends signals to family members and concerned health providers.



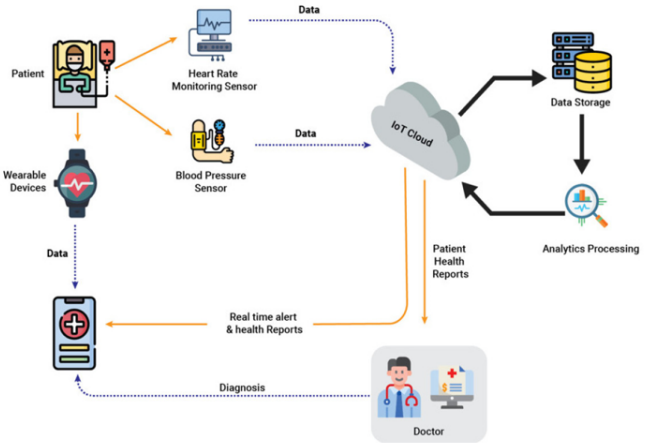
IoT in Health care

- **IoT for Physicians** - By using wearables and other home monitoring equipment embedded with IoT, physicians can keep track of patients' health more effectively.
- They can track patients' adherence to treatment plans or any need for immediate medical attention.
- IoT enables healthcare professionals to be more watchful and connect with the patients proactively.
- Data collected from IoT devices can help physicians identify the best treatment process for patients and reach the expected outcomes.



IoT in Health care

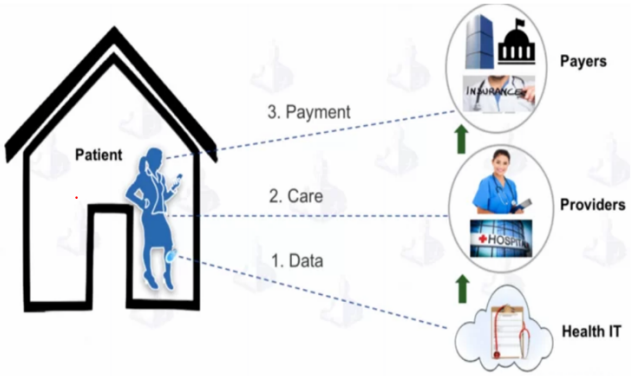
- **IoT for Hospitals** - Apart from monitoring patients' health, there are many other areas where IoT devices are very useful in hospitals.
- IoT devices tagged with sensors are used for tracking real time location of medical equipment like wheelchairs, defibrillators, nebulizers, oxygen pumps and other monitoring equipment.



- Deployment of medical staff at different locations can also be analyzed real time.
- The spread of infections is a major concern for patients in hospitals.
- IoT-enabled hygiene monitoring devices help in preventing patients from getting infected.

IoT in Health care

- **IoT for Health Insurance Companies –** There are numerous opportunities for health insurers with IoT-connected intelligent devices.
- Insurance companies can leverage data captured through health monitoring devices for their underwriting and claims operations.



- This data will enable them to detect fraud claims and identify prospects for underwriting.
- IoT devices bring transparency between insurers and customers in the underwriting, pricing, claims handling, and risk assessment processes.

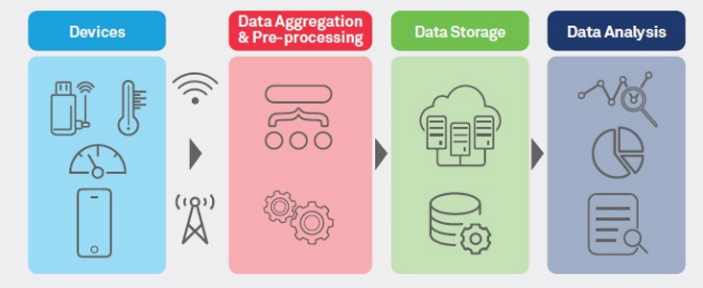
IoT in Health care

Redefining Healthcare–

Step 1: First step consists of deployment of interconnected devices that includes sensors, actuators, monitors, detectors, camera systems etc. These devices collect the data.

Step 2: Usually, data received from sensors and other devices are in analog form, which need to be aggregated and converted to the digital form for further data processing.

Step 3: Once the data is digitized and aggregated, this is pre-processed, standardized and moved to the data center or Cloud.



- **Step 4**: Final data is managed and analyzed at the required level. Advanced Analytics, applied to this data, brings actionable business insights for effective decision-making.
- IoT is redefining healthcare by ensuring better care, improved treatment outcomes and reduced costs for patients, and better processes and workflows, improved performance and patient experience for healthcare providers.

Advantages of IoT in Health care

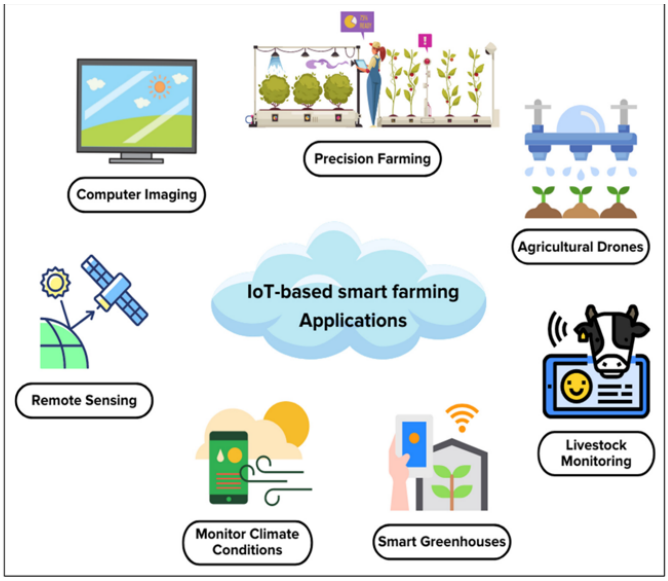
- Cost Reduction: IoT enables patient monitoring in real time, thus significantly cutting down unnecessary visits to doctors, hospital stays and re-admissions
- Improved Treatment: It enables physicians to make evidence-based informed decisions and brings absolute transparency
- Faster Disease Diagnosis: Continuous patient monitoring and real time data helps in diagnosing diseases at an early stage or even before the disease develops based on symptoms
- Proactive Treatment: Continuous health monitoring opens the doors for providing proactive medical treatment
- Drugs and Equipment Management: Management of drugs and medical equipment is a major challenge in a healthcare industry. Through connected devices, these are managed and utilized efficiently with reduced costs
- Error Reduction: Data generated through IoT devices not only help in effective decision making but also ensure smooth healthcare operations with reduced errors, waste and system costs

4. Agriculture and Farming

- **Smart agriculture**, on the other hand, is mostly used to denote the application of IoT solutions in agriculture.
- By using IoT sensors to collect environmental and machine metrics, farmers can make informed decisions, and improve just about every aspect of their work – from livestock to crop farming.
- For example, by using smart agriculture sensors to monitor the state of crops, farmers can define exactly how many pesticides and fertilizers they have to use to reach optimal efficiency. The same applies to the smart farming definition.

Challenges in Modern Agricultural Industry

- Lack of workforce and manpower
- Environmental challenges and global warming
- Requirement of large manual intervention
- Lack of proper monitoring
- Challenges in analyzing the large scale unstructured data



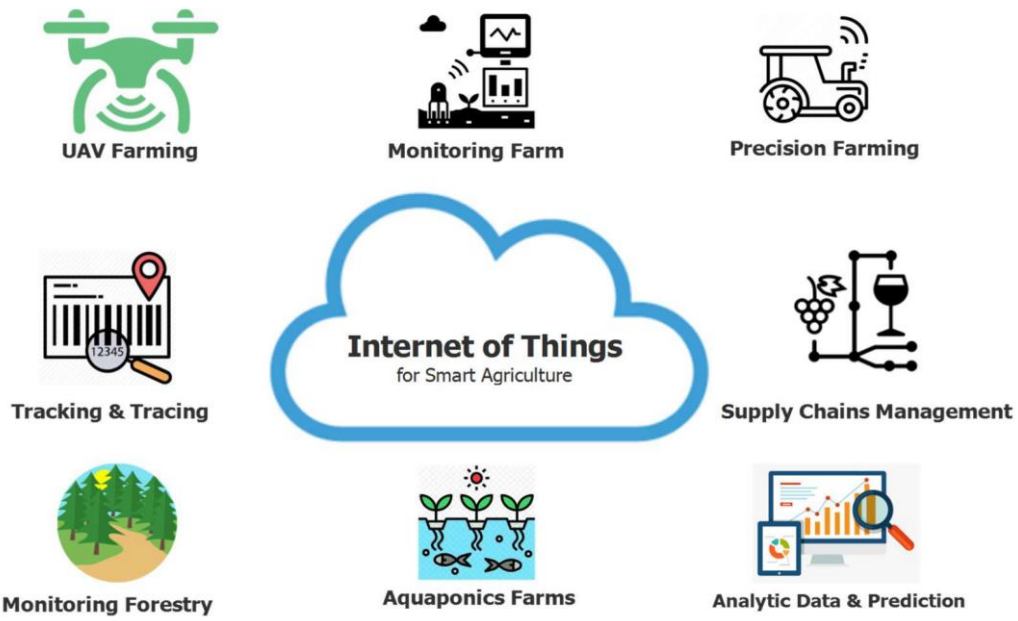
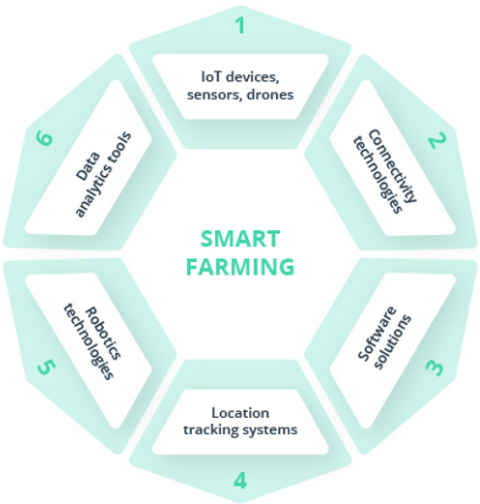


Fig.7.4 IoT in Agriculture and Farming

Benefits of smart farming

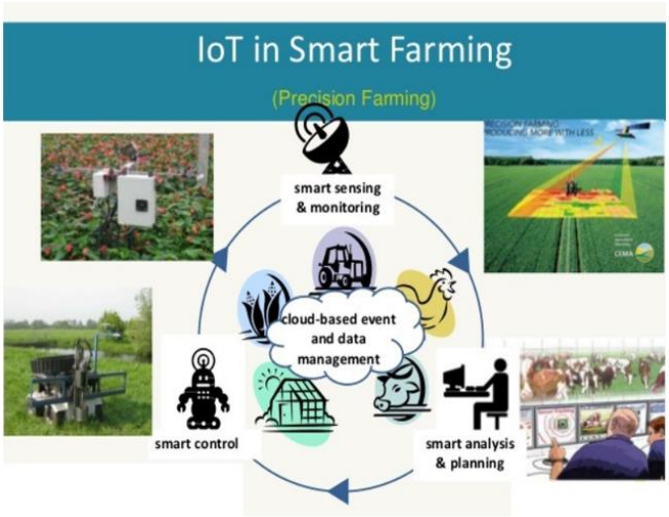
Smart farming Features

- **Data, tons of data, collected by smart agriculture sensors,** e.g. weather conditions, soil quality, crop's growth progress or cattle's health. This data can be used to track the state of your business in general as well as staff performance, equipment efficiency, etc.
- **Better control over the internal processes and, as a result, lower production risks**. The ability to foresee the output of your production allows you to plan for better product distribution. If you know exactly how much crops you are going to harvest, you can make sure your product won't lie around unsold.
- **Cost management and waste reduction for the increased control over the production**. Being able to see any anomalies in the crop growth or livestock health, you will be able to mitigate the risks of losing your yield.



Smart farming Features

- **Increased business efficiency through process automation**. By using smart devices, you can automate multiple processes across your production cycle, e.g. irrigation, fertilizing, or pest control.
- **Enhanced product quality and volumes**. Achieve better control over the production process and maintain higher standards of crop quality and growth capacity through automation.



- **Reduced environmental footprint.** Automation also carries environmental benefits. Smart farming technologies can cut down on the use of pesticides and fertilizer by offering more precise coverage, and thus, reduce greenhouse gas emissions.

IoT use cases in agriculture

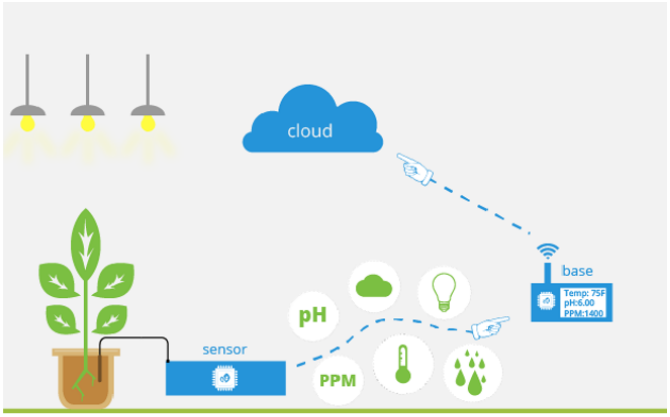
Monitoring of climate conditions

- The most popular smart agriculture gadgets are weather stations, combining various smart farming sensors. Located across the field, they collect various data from the environment and send it to the cloud. The provided measurements can be used to map the climate conditions, choose the appropriate crops, and take the required measures to improve their capacity (i.e. precision farming).
- Some examples of such agriculture IoT devices are all METEO, Smart Elements, and Pycno.

IoT use cases in agriculture

Greenhouse automation

- Typically, farmers use manual intervention to control the greenhouse environment. The use of IoT sensors enables them to get accurate real-time information on greenhouse conditions such as lighting, temperature, soil condition, and humidity.
- In addition to sourcing environmental data, weather stations can automatically adjust the conditions to match the given parameters.



- Specifically, greenhouse automation systems use a similar principle.
- For instance, Farmapp and Growlink are also IoT agriculture products offering such capabilities among others.

IoT use cases in agriculture

Crop management

- One more type of IoT product in agriculture and another element of precision farming are crop management devices.
- Just like weather stations, they should be placed in the field to collect data specific to crop farming; from temperature and precipitation to leaf water potential and overall crop health.
- Thus, you can monitor your crop growth and any anomalies to effectively prevent any diseases or infestations that can harm your yield.
- Arable and Semios can serve as good representations of how this use case can be applied in real life.

Cattle monitoring and management

- Just like crop monitoring, there are IoT agriculture sensors that can be attached to the animals on a farm to monitor their health and log performance. Livestock tracking and monitoring help collect data on stock health, well-being, and physical location.
- For example, such sensors can identify sick animals so that farmers can separate them from the herd and avoid contamination. Using drones for real-time cattle tracking also helps farmers reduce staffing expenses. This works similarly to IoT devices for petcare.
- For example, SCR by Allflex and Cowlar use smart agriculture sensors (collar tags) to deliver temperature, health, activity, and nutrition insights on each individual cow as well as collective information about the herd.

Precision farming

- Also known as precision agriculture, precision farming is all about efficiency and making accurate data-driven decisions.
- By using IoT sensors, farmers can collect a vast array of metrics on every facet of the field microclimate and ecosystem: lighting, temperature, soil condition, humidity, CO2 levels, and pest infections.
- This data enables farmers to estimate optimal amounts of water, fertilizers, and pesticides that their crops need, reduce expenses, and raise better and healthier crops.
- For example, **CropX** builds IoT soil sensors that measure soil moisture, temperature, and electric conductivity enabling farmers to approach each crop's unique needs individually. Combined with geospatial data, this technology helps create precise soil maps for each field. **Mothive** offers similar services, helping farmers reduce waste, improve yields, and increase farm sustainability.

Agricultural drones

- Perhaps one of the most promising agritech advancements is the use of agricultural drones in smart farming.
- Also known as UAVs (unmanned aerial vehicles), drones are better equipped than airplanes and satellites to collect agricultural data.
- The data collected from these drones are sent back to the server to be used for analysis and decision-making.
- Apart from surveillance capabilities, drones can also perform a vast number of tasks that previously required human labor: planting crops, fighting pests and infections, agriculture spraying, crop monitoring, etc.
- DroneSeed, for example, builds drones for planting trees in deforested areas.

Predictive analytics for smart farming

- Precision agriculture and predictive data analytics go hand in hand. While IoT and smart sensor technology are a goldmine for highly relevant real-time data, the use of data analytics helps farmers make sense of it and come up with important predictions: crop harvesting time, the risks of diseases and infestations, yield volume, etc.
- Data analytics tools help make farming, which is inherently highly dependent on weather conditions, more manageable, and predictable.
- For example, the **Crop Performance** platform helps farmers access the volume and quality of yields in advance, as well as their vulnerability to unfavorable weather conditions, such as floods and drought. It also enables farmers to optimize the supply of water and nutrients for each crop and even select yield traits to improve quality.

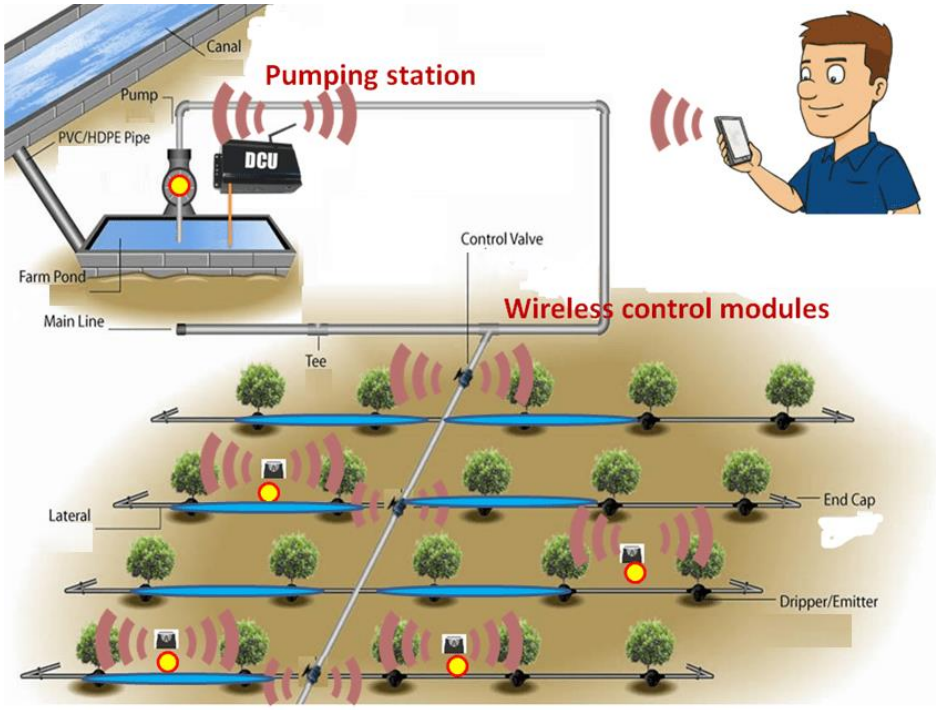


Fig.7.5 IoT in smart Farming

End-to-end farm management systems

- A more complex approach to IoT products in agriculture can be represented by the so-called farm productivity management systems. They usually include a number of

agriculture IoT devices and sensors, installed on the premises as well as a powerful dashboard with analytical capabilities and in-built accounting/reporting features.

- This offers remote farm monitoring capabilities and allows you to streamline most of the business operations. Similar solutions are represented by FarmLogs and Cropio.
- In addition to the listed IoT agriculture use cases, some prominent opportunities include vehicle tracking (or even automation), storage management, logistics, etc.

Robots and autonomous machines

- Robotic innovations also offer a promising future in the field of autonomous machines for agricultural purposes.
- Some farmers already use automated harvesters, tractors, and other machines and vehicles that can operate without a human controlling it. Such robots can complete repetitive, challenging, and labor-intensive tasks.
- For instance, modern agrobots include automated tractors that can work on assigned routes, send notifications, start work at planned hours, etc. Such tractors are driverless and cut farmers' labor costs. Bear Flag Robotics is one company that works on such technology at the moment.

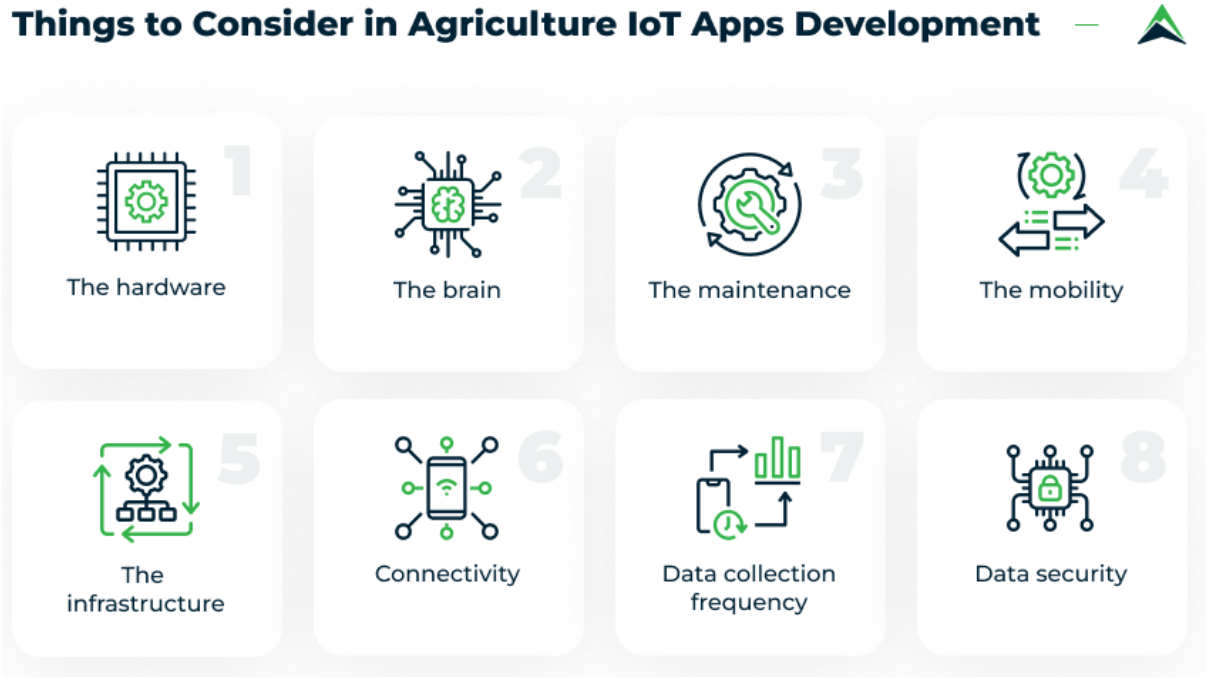


Fig.7.6 Key aspects to be considered for IoT in agriculture

Challenges to Building the Internet of Things Platform

- A unified solution that can be integrated with different types of Internet of Things devices.
- The most common challenge for the Internet of Things in agriculture is connectivity. Every area doesn't have proper internet connectivity.
- The second most common challenge for the Internet of Things based Advanced Agriculture is the lack of awareness among consumers.
- Due to various service providers, it becomes really difficult to maintain interoperability between different IoT systems.
- A scalable solution that can be integrated with thousands of IoT devices for large farms.

Solutions for Building IoT based Intelligent Agriculture

- Smart Farming has enabled farmers to reduce waste and enhance productivity with the help of sensors (light, humidity, temperature, soil moisture, etc.) and automation of irrigation systems.
- Further, with the help of these sensors, farmers can monitor the field conditions from anywhere.
- Internet of Things-based Advanced Farming is highly efficient when compared with the conventional approach.
- The applications of intelligent Agriculture solutions not only target conventional, large farming.
- With operations, but could also be new levers to uplift other growing or common trends in agriculture like organic farming, family farming (complex or small spaces, particular cattle and/or cultures, preservation of specific or high-quality varieties, etc.), and enhance highly transparent Farming.
- In Azure, IoT Hub acts as the central unit which allows connecting, monitoring and managing millions of devices using bi-directional messaging. It supports AMQP, MQTT, and HTTP protocols for communication with IoT devices.
- It also helps their own protocol gateway Azure IoT Protocol Gateway, in case a device doesn't support AMQP, MQTT, or HTTP.