
IoT

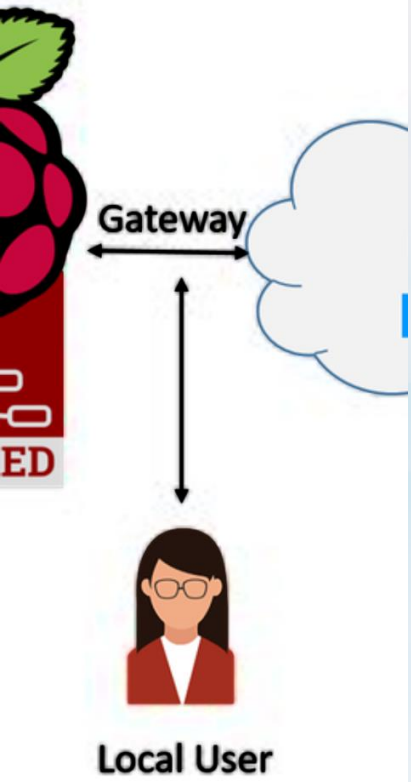
Introduction to IoT

Internet of Things (IoT) is a network of interconnected devices that communicate and share data with each other.

It encompasses a wide range of devices, from everyday appliances to industrial equipment, and has the potential to revolutionize how we interact with technology.

These devices are embedded with sensors, actuators, and communication technology, enabling them to collect, exchange, and act upon data.

The goal of IoT is to create a network where these devices can seamlessly communicate and collaborate to provide enhanced functionalities, automation, and data-driven insights.



Local User

What is IoT?

1 **Interconnected Devices**

IoT refers to the connection and communication between devices, allowing them to exchange data and perform automated tasks.

2 **Data Sharing**

It involves the seamless sharing and utilization of data generated by sensors and devices for improved efficiency and decision-making.

3 **Remote Monitoring and Control**

IoT enables remote monitoring and control of devices, leading to enhanced convenience and management of resources.

History of IoT

1

Early Concepts

The concept of interconnected devices and smart objects dates back to the early 1980s, with visionary thinkers foreseeing a future characterized by digital connectivity.

2

Emergence of Standards

In the late 1990s and early 2000s, industry organizations and consortia began developing standards and protocols for IoT, setting the stage for its widespread adoption.

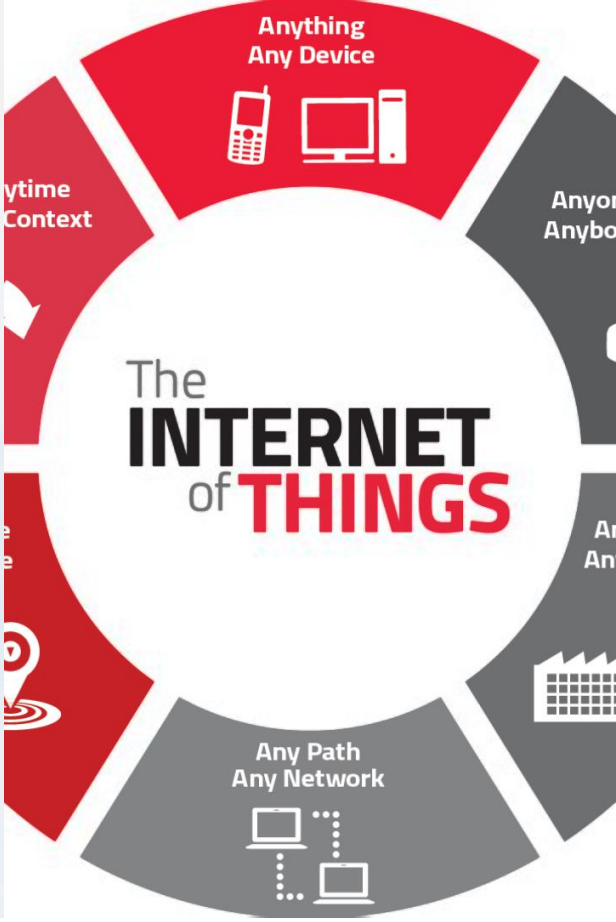
3

Integration into Daily Life

IoT's integration into everyday life accelerated with the advent of smartphones and affordable, high-speed internet, paving the way for smart homes and cities.

IoT Enabling Technologies

IoT Enabling Technologies refer to the foundational elements that make the Internet of Things possible. These include wireless communication protocols, sensor technologies, cloud and edge computing, data analytics, security, privacy, blockchain, and artificial intelligence integration.



Wireless Communication Protocols for IoT

Wi-Fi

Wi-Fi is widely used for IoT applications due to its high data transfer rates and availability in most settings.

Bluetooth

Bluetooth is ideal for low-power, short-range communication between IoT devices.

Zigbee

Zigbee is designed for low-data-rate, low-power applications such as home automation and smart lighting.

Sensor Technologies for IoT

1 **Temperature Sensors**

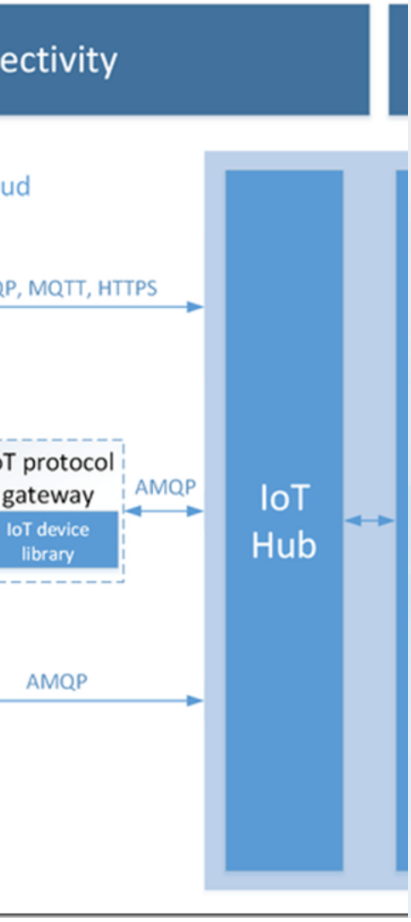
Used for monitoring temperature variations in industrial and residential environments.

2 **Pressure Sensors**

Measure pressure changes in industrial equipment, pipeline systems, and HVAC applications.

3 **Proximity Sensors**

Commonly employed in robotics, automotive safety systems, and touch-activated devices.



Cloud Computing and IoT

Scalability

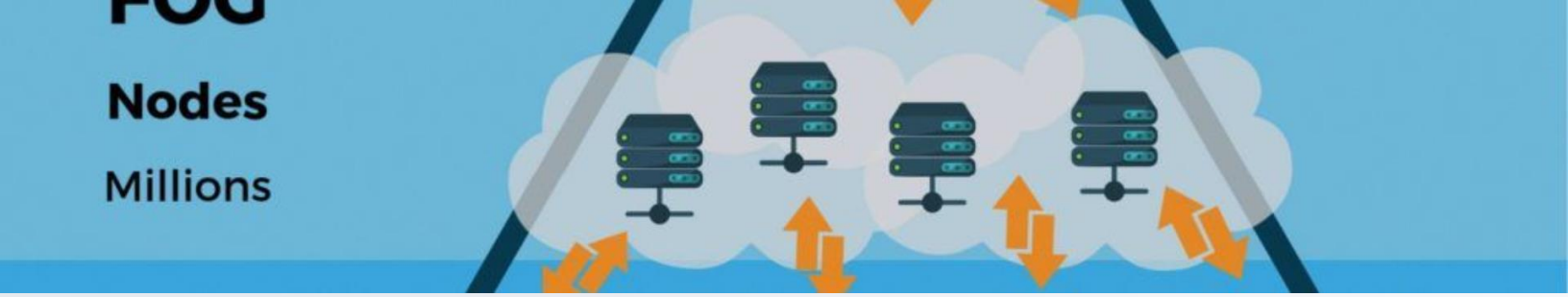
Cloud computing allows IoT systems to scale resources according to demand, ensuring seamless operation.

Data Storage

Cloud storage provides a secure and configurable environment for storing and analyzing IoT data.

Flexibility

IoT devices can leverage the elasticity of cloud infrastructure to adapt to changing conditions.



Edge Computing and IoT

1

2

3

Data Processing

Edge computing enables real-time data processing and analysis at or near the data source.

Latency Reduction

Minimizes latency in IoT applications by processing data locally, leading to faster responses.

Bandwidth Efficiency

Reduces the need for bandwidth by processing and filtering data before transmission to the cloud.

Data Analytics and IoT

3

Data Visualization

Enhances decision-making and provides actionable insights from IoT-generated data.

7

Predictive Maintenance

Anticipates equipment failures and reduces downtime through predictive analytics.

2K

Big Data Handling

Processes and analyzes large volumes of IoT data to extract valuable information and patterns.

Artificial Intelligence and IoT

1

Machine Learning Models

Utilized to analyze and make decisions based on IoT-generated data, leading to autonomous autonomous actions.

2

Natural Language Processing

Enables IoT devices to understand and respond to human language inputs, enhancing user enhancing user interaction.

3

Computer Vision

Employs visual data analysis for object recognition, surveillance, and human activity monitoring monitoring in IoT environments.

Future Trends in IoT Enabling Technologies

1

5G Integration

5G networks will enable faster and more reliable communication, unlocking new IoT applications and use cases.

2

Edge AI Adoption

Increasing integration of AI capabilities at the edge for real-time decision-making and local processing.

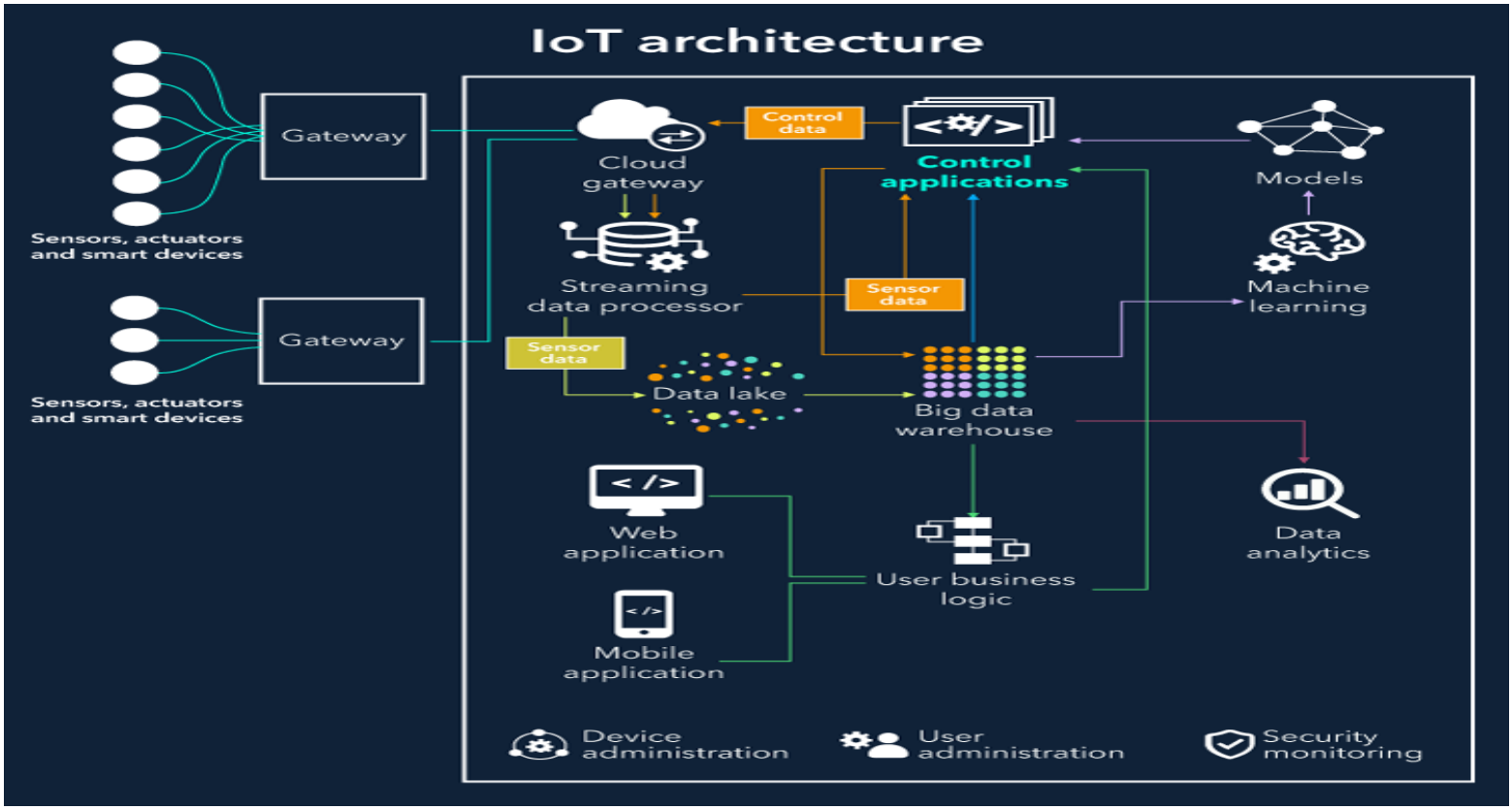
3

Quantum Computing Impact

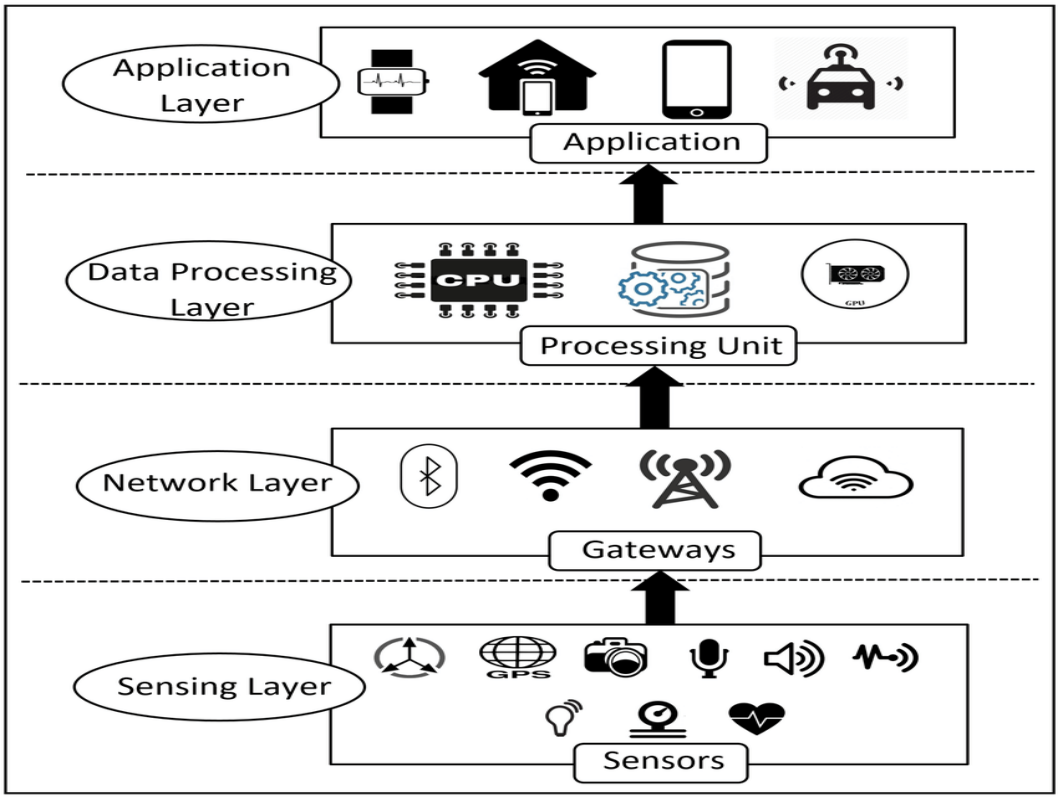
Quantum computing advancements will revolutionize the processing power and capabilities of IoT ecosystems.

IoT architecture

The Internet of Things (IoT) architecture refers to the framework of interconnected physical devices and software systems. It encompasses the structure, protocols, and standards that enable these devices to communicate and exchange data.


IoT architecture
Gateway
Sensors, actuators and smart devices
Gateway
Sensors, actuators and smart devices
Cloud gateway
Control data
Control applications
Models
Streaming data processor
Sensor data
Machine learning
Sensor data
Data lake
Big data warehouse
Web application
User business logic
Data analytics
Mobile application
Device administration
User administration
Security monitoring

Layers of IOT Architecture



Layers of IOT Architecture

Perception Layer (Sensing):

Devices/Things: These are the physical objects or devices embedded with sensors and actuators. Sensors collect data from the environment, and actuators perform actions based on the received data.

Sensors and Actuators: Sensors capture information from the physical world, such as temperature, humidity, motion, etc. Actuators enable devices to perform actions in response to the collected data.

Network Layer:

Connectivity: This layer focuses on the communication protocols and technologies that facilitate data exchange between IoT devices. Common communication protocols include Wi-Fi, Bluetooth, Zigbee, RFID, and cellular networks.

Middleware Layer:

Data Processing and Communication: This layer handles data processing, message brokering, and communication between devices. It may include protocols for secure data transmission and middleware solutions that facilitate interoperability between different devices and platforms.

Application Layer:

Applications and Services: This layer involves the development of applications and services that leverage the data collected by IoT devices. It includes user interfaces, analytics, and business logic that provide value to end-users or businesses.

Layers of IOT Architecture

Business Layer (Enterprise):

Business Logic and Integration: This layer integrates IoT data into broader business processes. It includes the logic and rules that govern how data is used to make decisions, optimize processes, and deliver business value.

Security and Privacy Layer:

Security Measures: Given the potential vulnerabilities in IoT systems, security is a critical concern. This layer includes measures such as encryption, authentication, access control, and secure update mechanisms to protect data and devices.

Cloud and Edge Computing Layer:

Cloud Services: Many IoT systems leverage cloud computing for scalable storage, processing, and analytics of large volumes of data. Edge computing, on the other hand, involves processing data closer to the source, reducing latency and bandwidth requirements.

End-User Layer:

User Interfaces: This layer provides interfaces for end-users to interact with IoT devices and applications. It could be web interfaces, mobile apps, or other means through which users can monitor and control IoT devices.

Regulatory and Compliance Layer:

Regulatory Compliance: This layer ensures that the IoT system complies with relevant regulations and standards. It addresses issues related to data privacy, security, and industry-specific compliance requirements.

Overview of oneM2M standard

1

Unified Approach

oneM2M is a global standard for M2M and IoT systems, providing a common common foundation for diverse applications and industries.

2

Interoperability

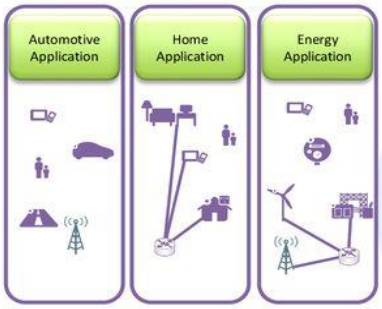
It ensures seamless integration between various devices and platforms, promoting promoting interoperability and scalability in IoT deployments.

3

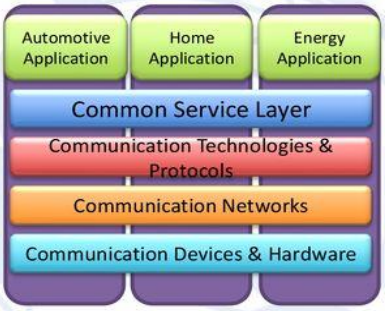
Flexibility

With support for different communication technologies and underlying networks, it networks, it offers flexibility in implementing IoT solutions.

oneM2M Architecture approach



Currently developed solutions are similar are vertically integrated, with limited integration of data models (Zigbee, DLMS for smart meters, etc.).



Horizontal framework, Restful API
Objects represented as resource
Access control policy to access resource

IoT will be based on ontologies (formal description of concepts and relationships, e.g. W3C Semantic Sensor Network) as well as big data frameworks

IoT ready

TOMORROW
IoT enabled

M2M

Machine-to-Machine (M2M) communication is a subset of Internet of Things (IoT) that involves direct communication between devices without human intervention. M2M architecture enables devices to exchange information and perform actions based on the data they receive. Here's a simplified overview of the architecture for M2M communication within an IoT system:

Device/Thing Layer:

Sensors and Actuators: Devices or things are equipped with sensors to collect data from the environment and actuators to perform actions based on that data.

Device Communication Layer:

Connectivity: This layer handles the communication protocols and technologies that enable devices to directly exchange data with each other. Common communication protocols for M2M include MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), and HTTP/HTTPS.

Middleware Layer:

Data Processing and Communication: Middleware facilitates communication between devices and helps in processing the data. It may include protocols for efficient and reliable message transfer, data encoding/decoding, and addressing.

M2M

- **Application Layer:**
  - **Applications and Services:** This layer involves applications that leverage the data exchanged between devices. It includes business logic, data analysis, and services that provide value to end-users or other devices.
- **Security Layer:**
  - **Security Measures:** Security is crucial in M2M communication to ensure data integrity and confidentiality. This layer includes encryption, authentication, and access control mechanisms to secure the communication between devices.
- **Device Management Layer:**
  - **Configuration and Monitoring:** This layer is responsible for managing the lifecycle of devices. It includes functionalities such as device registration, configuration, monitoring, and over-the-air (OTA) updates.
- **Cloud and Edge Computing Layer:**
  - **Cloud Services:** M2M data can be processed in the cloud for storage, analytics, and additional services. Edge computing may also be employed for real-time processing closer to the devices to reduce latency and bandwidth usage.
- **End-User Layer:**
  - **User Interfaces:** While M2M communication often operates without direct human intervention, there may be user interfaces for administrators or end-users to monitor and manage the devices or access relevant information.

Benefits of using oneM2M in IoT deployments

1

Scalability

oneM2M allows for the efficient scaling of IoT deployments, ensuring adaptability to changing demands and growth.

2

Resource Optimization

It optimizes resource usage, leading to cost efficiencies and improved improved performance in in IoT systems.

3

Interoperable Ecosystem

Enables the creation of an an interconnected ecosystem, fostering collaboration and innovation across different different IoT applications. applications.

IoT World Forum (IoTWF)

Global Platform

IoTWF serves as a global platform for industry leaders, innovators, and experts to discuss the latest latest trends and advancements in IoT.

Knowledge Exchange

It facilitates meaningful exchanges of knowledge, best practices, and strategies for leveraging IoT technologies.

IoT World Forum Reference Model

Levels

7 **Collaboration & Processes**
(Involving People & Business Processes)

6 **Application**
(Reporting, Analytics, Control)

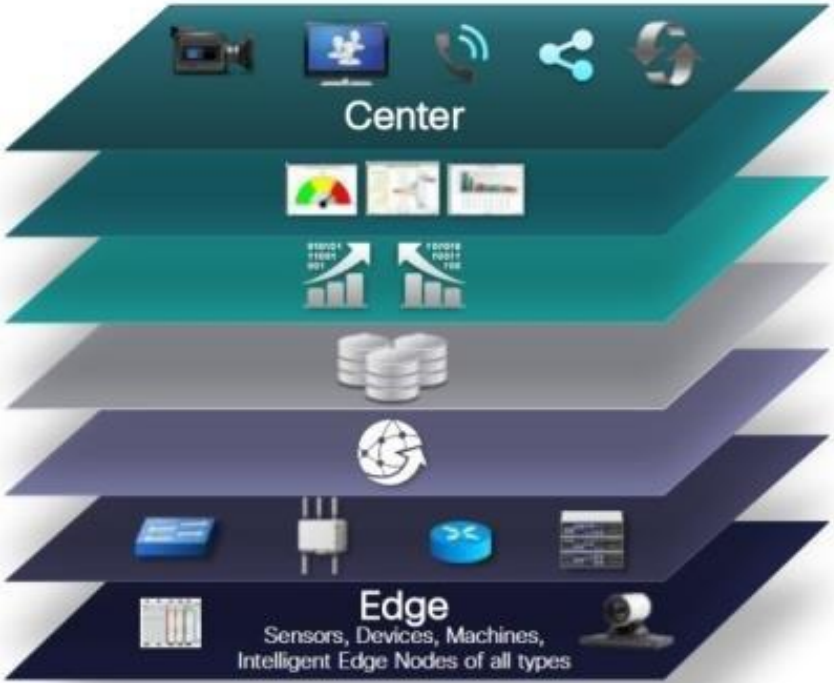
5 **Data Abstraction**
(Aggregation & Access)

4 **Data Accumulation**
(Storage)

3 **Edge Computing**
(Data Element Analysis & Transformation)

2 **Connectivity**
(Communication & Processing Units)

1 **Physical Devices & Controllers**
(The "Things" in IoT)



Key highlights and objectives of IoTWF

Networking Opportunities

IoTWF provides ample opportunities for networking and establishing valuable connections with industry peers and IoT experts.

Thought Leadership

It promotes thought leadership by showcasing showcasing innovative IoT solutions and facilitating insightful discussions on industry industry challenges and opportunities.

The primary objective is to drive the development and adoption of IoT technologies across diverse sectors.

Case studies showcasing successful IoT implementations using oneM2M

1

Smart Agriculture

Implementing oneM2M standards led to enhanced crop monitoring, irrigation irrigation control, and improved yield in agricultural operations.

2

Smart Cities

Seamless integration and interoperability of IoT devices facilitated efficient traffic management, waste management, and public safety.

Alternatives to oneM2M for IoT deployments

LoRaWAN

LoRaWAN offers long-range connectivity suitable for IoT applications such as smart metering and asset tracking.

MQTT Protocol

It's a lightweight messaging protocol ideal for scenarios where low power consumption and bandwidth efficiency are crucial.

LoRaWAN

LoRaWAN provides exceptional long-range coverage, allowing seamless data transmission over vast distances, making it the ideal solution for applications requiring extensive coverage in remote areas where other connectivity technologies cannot span.

LoRaWAN is a Media Access Control (MAC) layer protocol built on top of LoRa modulation. It is a software layer which defines how devices use the LoRa hardware, for example when they transmit, and the format of messages.

The LoRaWAN protocol is developed and maintained by the LoRa Alliance. The first LoRaWAN specification was released in January 2015.

Bandwidth vs. Range

LoRaWAN is suitable for transmitting small size payloads (like sensor data) over long distances. LoRa modulation provides a significantly greater communication range with low bandwidths than other competing wireless data transmission technologies.

Why is LoRaWAN so awesome?

- **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

LoRaWAN

LoRaWAN use cases

Here are a few great LoRaWAN use cases provided by Semtech, to give you some insight into how LoRaWAN can be applied:

- **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

LoRaWAN

LoRa Alliance

The LoRa Alliance® is an open, non-profit association established in 2015.
It supports development of the LoRaWAN protocol and ensures interoperability of all LoRaWAN products and technologies. Today, the LoRa Alliance has over 500 members around the globe.
The LoRa Alliance provides LoRaWAN certification for end devices. Certified end devices provide users with confidence that the end device is reliable and compliant with the LoRaWAN specification.

Certification is only available for device manufacturers that are members of the LoRa Alliance. Once certified, the manufacturer can use the LoRaWAN Certified mark with the product.

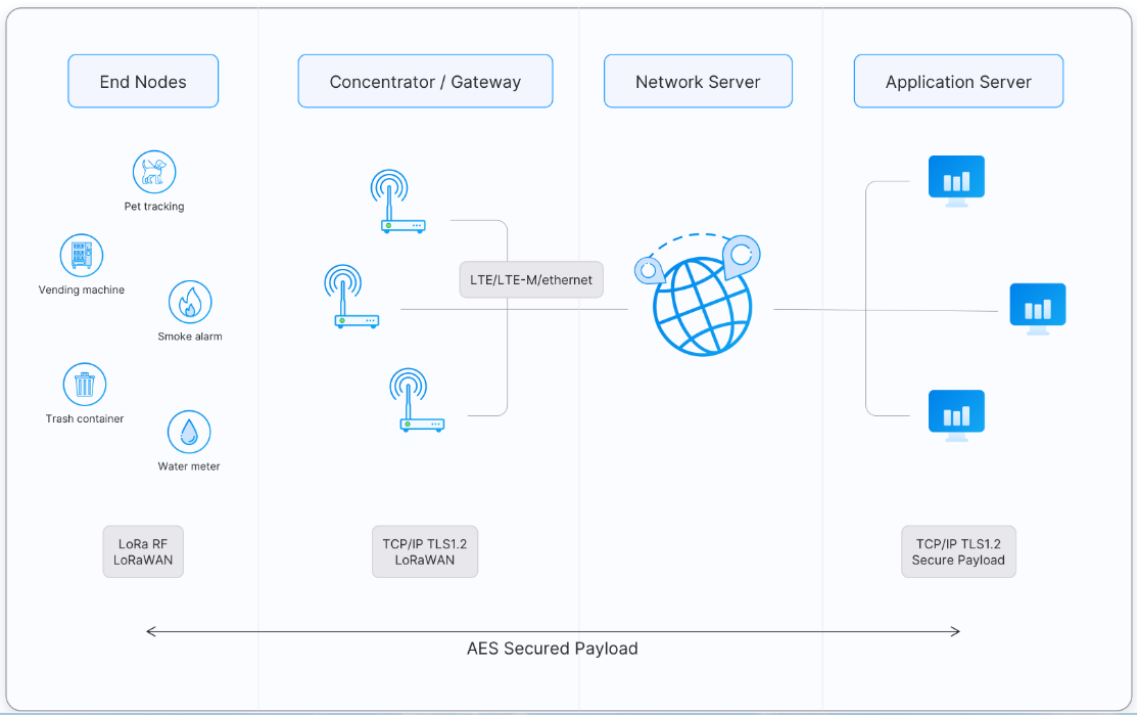
LoRaWAN is now an ITU standard.

As announced by the LoRa Alliance® on December 7, 2021, LoRaWAN® is officially approved as a standard for Low Power Wide Area Networking (LPWAN) by the International Telecommunication Union (ITU)

LoRaWAN

LoRaWAN Architecture

LoRaWAN networks are deployed in a **star-of-stars** topology.
A typical LoRaWAN network consists of the following elements.



LoRaWAN

•**End Devices** - sensors or actuators send LoRa modulated wireless messages to the gateways or receive messages wirelessly back from the gateways. .
•**Gateways** - receive messages from end devices and forward them to the Network Server.
•**Network Server** - a piece of software running on a server that manages the entire network.
•**Application servers** - a piece of software running on a server that is responsible for securely processing application data.
•**Join Server** - a piece of software running on a server that processes join-request messages sent by end devices (The Join Server is not shown in the above figure).
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.

LoRaWAN

End devices

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. The following figure shows an end device that consists of sensors like temperature, humidity, and fall detection

LoRaWAN

Gateways

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.

Types of LoRaWAN Gateways

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.

The following figure shows The Things Indoor gateway designed to be directly plugged into an AC power outlet.

Figure: The Things Indoor gateway

LoRaWAN

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. If you are good at hacking electronic products, you can convert some indoor gateways to outdoor gateways using water/dust proof enclosures and adding external antennas.
The following figure shows a LoRaWAN outdoor gateway. It has connectors for connecting external LoRaWAN, 3G/4G, and GPS antennas.

Figure: Tektelic Enterprise Outdoor Gateway

LoRaWAN

Network Server

The Network Server manages gateways, end-devices, applications, and users in the entire LoRaWAN network.

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

LoRaWAN

Application Server

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

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. The Join Server was first introduced with LoRaWAN v1.1. It is also availabe in LoRaWAN v1.0.4

LoRaWAN

Message Types

In this chapter, you will learn about different message types used in LoRaWAN 1.0.x and 1.1. These message types are used to transport MAC commands and application data. The Things Fundamentals Certification exam expects you should have basic knowledge on the following topics with regards to the message types:

•Uplink and downlink messages.
•MAC Message types and their uses.
•Sending MAC commands in the FOpts field.
•Sending MAC commands and application data in the FRMPayload field.
•Keys used to encrypt each field that carries MAC Commands and application data.
•Keys used to calculate the Message Integrity Code (MIC) of each message.

Uplink and Downlink Messages

LoRa messages can be divided into uplink and downlink messages based on the direction they travel.

Uplink messages - Uplink messages are sent by end devices to the Network Server relayed by one or many gateways. If the uplink message belongs to the Application Server or the Join Server, the Network server forwards it to the correct receiver.

Downlink messages - Each downlink message is sent by the Network Server to only one end device and is relayed by a single gateway. This includes some messages initiated by the Application Server and the Join Server too

LoRaWAN

MAC Message Types

LoRaWAN defines several MAC message types.
The following table presents MAC message types that can be found in LoRaWAN 1.0.x and 1.1

LoRaWAN 1.0.x	LoRaWAN 1.1	Description
Join-request	Join-request	An uplink message, used by the over-the-air activation (OTAA) procedure
Join-accept	Join-accept	A downlink message, used by the over-the-air activation (OTAA) procedure
Unconfirmed Data Up	Unconfirmed Data Up	An uplink data frame, confirmation is not required
Unconfirmed Data Down	Unconfirmed Data Down	A downlink data frame, confirmation is not required
Confirmed Data Up	Confirmed Data Up	An uplink data frame, confirmation is requested
Confirmed Data Down	Confirmed Data Down	A downlink data frame, confirmation is requested
RFU	Rejoin-request	1.0.x - Reserved for Future Usage 1.1 - Uplink over-the-air activation (OTAA) Rejoin-request

LoRaWAN

Join-request, Rejoin-request, and Join-accept messages

In LoRaWAN 1.0.x, **two** MAC message types are used by the Over-The-Air-Activation (OTAA) procedure:

•Join-request
•Join-accept

In LoRaWAN 1.1, **three** MAC message types are used by the Over-The-Air-Activation (OTAA) procedure and for roaming purposes:

•Join-request
•Join-accept
•Rejoin-request

Join-request

The Join-request message is always initiated by an end device and sent to the Network Server. In LoRaWAN versions **earlier** than 1.0.4 the Join-request message is forwarded by the Network Server to the Application Server. In LoRaWAN 1.1 and 1.0.4+, the Network Server forwards the Join-request message to the device's Join Server. The Join-request message is not encrypted.

Join-accept

In LoRaWAN versions **earlier** than 1.0.4 the Join-accept message is generated by the Application Server. In LoRaWAN 1.1 and 1.0.4+ the Join-accept message is generated by the Join Server. In both cases the message passes through the Network Server. Then the Network Server routes the Join-accept message to the correct end-device. The Join-accept message is encrypted as follows.

•In LoRaWAN 1.0, the Join-accept message is encrypted with the AppKey.
•In LoRaWAN 1.1, the Join-accept message is encrypted with different keys

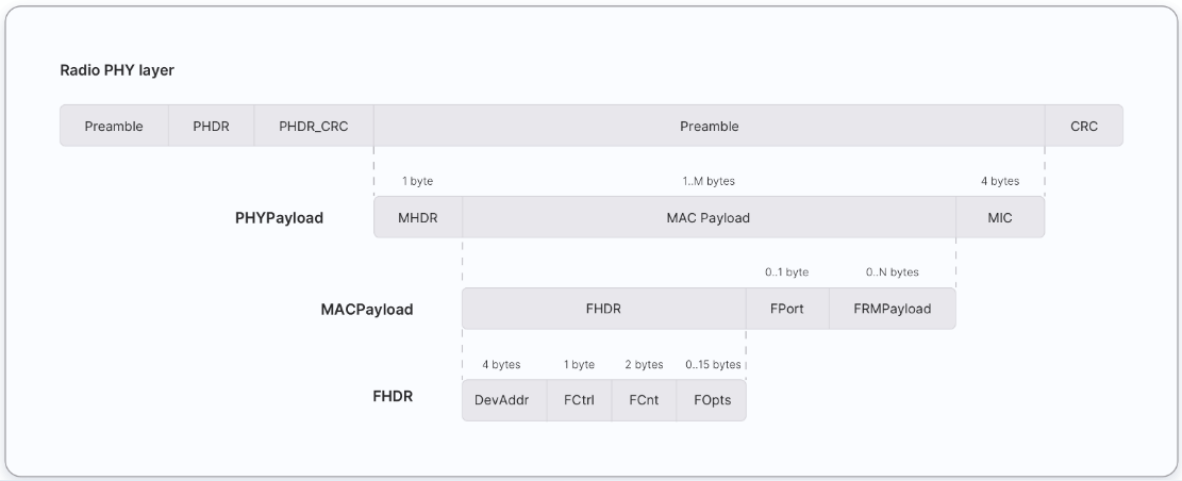
Rejoin-request

The Rejoin-request message is always initiated by an end device and sent to the Network Server. There are three types of Rejoin-request messages: Type 0, 1, and 2. These message types are used to initialize the new session context for the end device. For the Rejoin-request message, the network replies with a Join-accept message

LoRaWAN

Data Messages

There are 4 data message types used in both LoRaWAN 1.0.x and 1.1. These data message types are used to transport both MAC commands and application data which can be combined together in a single message. Data messages can be confirmed or unconfirmed. Confirmed data messages must be acknowledged by the receiver whereas unconfirmed data messages do not need to be acknowledged by the receiver.
A data message is constructed as shown below:



MQTT protocol

MQTT stands for **Message Queuing Telemetry Transport**.
MQTT is a machine to machine internet of things connectivity protocol.
It is an extremely lightweight and publish-subscribe messaging transport protocol.
This protocol is useful for the connection with the remote location where the bandwidth is a premium.
These characteristics make it useful in various situations, including constant environment such as for communication machine to machine and internet of things contexts.
It is a publish and subscribe system where we can publish and receive the messages as a client.
It makes it easy for communication between multiple devices.
It is a simple messaging protocol designed for the constrained devices and with low bandwidth, so it's a perfect solution for the internet of things applications.

MQTT protocol

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.

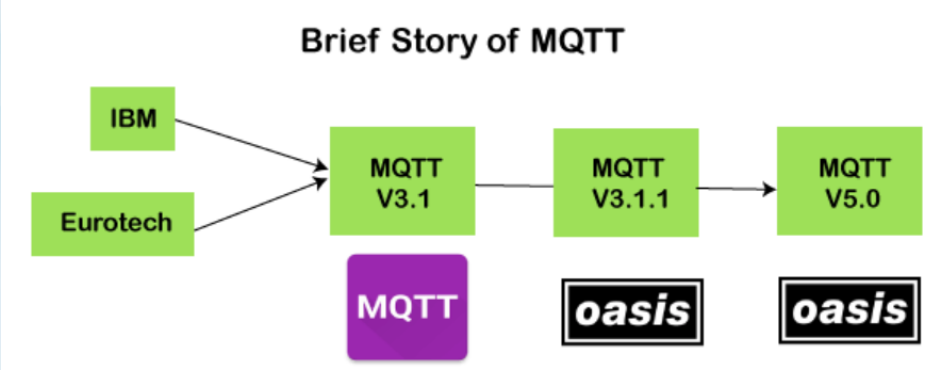
The major functional objectives in version 5.0 are:

- Enhancement in the scalability and the large-scale system in order to set up with the thousands or the millions of devices.
- Improvement in the error reporting

MQTT protocol

History of MQTT

The MQTT was developed by Dr. Andy Stanford-Clark, IBM, and Arlen Nipper. The previous versions of protocol 3.1 and 3.1.1 were made available under MQTT ORG. In 2014, the MQTT was officially published by OASIS. The OASIS becomes a new home for the development of the MQTT. Then, the OASIS started the further development of the MQTT. Version 3.1.1 is backward comfortable with a 3.1 and brought only minor changes such as changes to the connect message and clarification of the 3.1 version. The recent version of MQTT is 5.0, which is a successor of the 3.1.1 version. Version 5.0 is not backward, comfortable like version 3.1.1. According to the specifications, version 5.0 has a significant number of features that make the code in place.



MQTT protocol

MQTT Architecture

To understand the MQTT architecture, we first look at the components of the MQTT.

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

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

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:

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.

MQTT protocol

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.

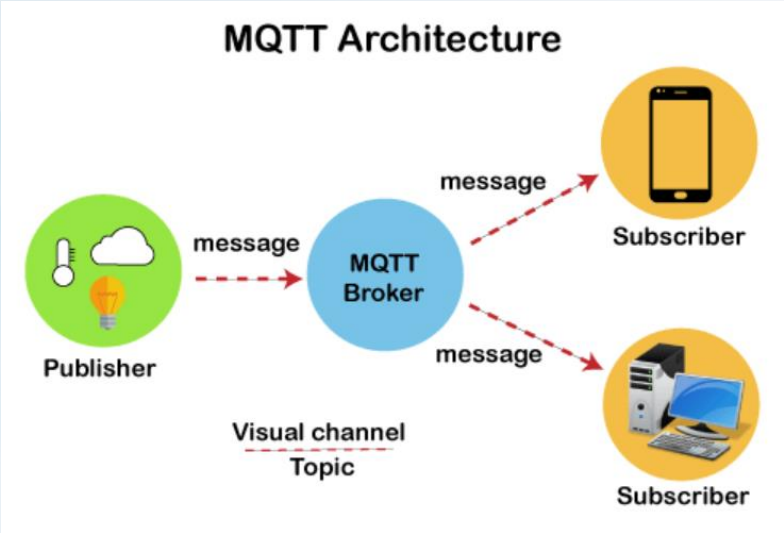
TOPIC

The label provided to the message is checked against the subscription known by the server is known as TOPIC.

MQTT protocol

Architecture of MQTT

Example. Suppose a device has a temperature sensor and wants to send the rating to the server or the broker. If the phone or desktop application wishes to receive this temperature value on the other side, then there will be two things that happened. The publisher first defines the topic; for example, the temperature then publishes the message, i.e., the temperature's value. After publishing the message, the phone or the desktop application on the other side will subscribe to the topic, i.e., temperature and then receive the published message, i.e., the value of the temperature. The server or the broker's role is to deliver the published message to the phone or the desktop application.

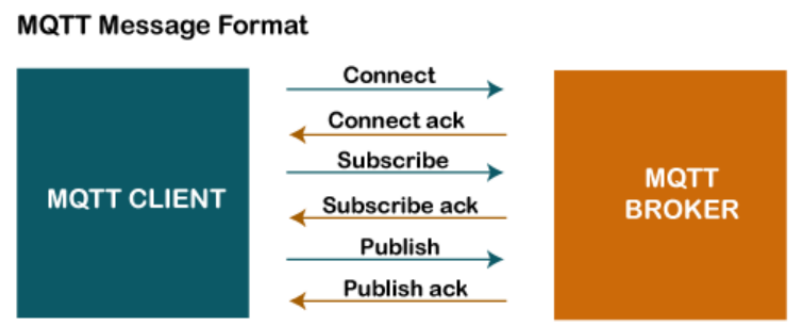


MQTT protocol

MQTT Message Format

The MQTT uses the command and the command acknowledgment format, which means that each command has an associated acknowledgment. As shown in the below figure that the connect command has connect acknowledgment, subscribe command has subscribe acknowledgment, and publish command has publish acknowledgment.

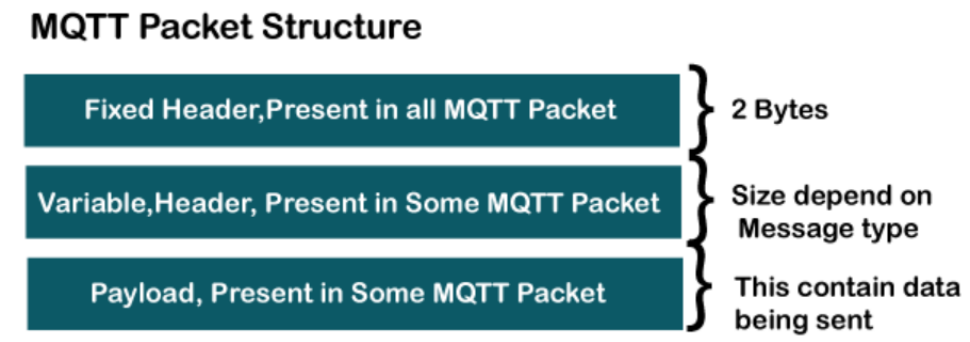
This mechanism is similar to the handshaking mechanism as in TCP protocol.



MQTT protocol

MQTT Packet structure:

The MQTT message format consists of 2 bytes fixed header, which is present in all the MQTT packets. The second field is a variable header, which is not always present. The third field is a payload, which is also not always present. The payload field basically contains the data which is being sent. We might think that the payload is a compulsory field, but it does not happen. Some commands do not use the payload field, for example, disconnect message.



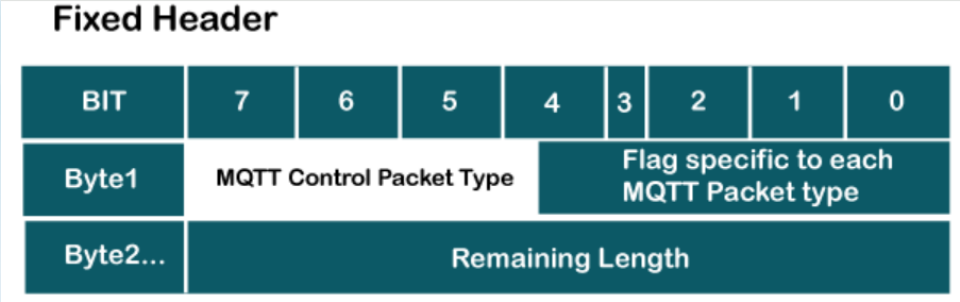
MQTT protocol

Fixed Header

the fixed header contains two bytes. The first byte contains the following fields:

- **MQTT Control Packet Type:** It occupies 4 bits, i.e., 7 to 4-bit positions. This 4-bit is an assigned value, and each bit represents the MQTT control packet type.
- **Flag specific to each MQTT packet type:** The remaining 4-bits represent flag specific to each MQTT packet type.

The byte 2 contains the remaining length, which is a variable-length byte integer. It represents the number of bytes remaining in a current control packet, including data in the variable header and payload. Therefore, we can say that the remaining length is equal to the sum of the data in the variable header and the payload.



MQTT protocol

MQTT Control Packet Types

table shows the control packet types with 4-bit value and direction flow. As we can observe that every command is followed by acknowledgment like CONNECT has CONNACK, PUBLISH has PUBACK, PUBREC, PUBREL, and PUBCOMP, SUBSCRIBE has SUBACK, UNSUBSCRIBE has UNSUBACK.

MQTT Control Packet Types

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Connection request
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowlegment(QoS1)
PUBREC	5	Client to Server or Server to Client	Publish received(QoS2 delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release(QoS 2 delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (QoS 2 delivery part 3)
SUBSCRIBE	8	Client to Server	Subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server or Server to Client	Disconnect notification
AUTH	15	Client to Server or Server to Client	Authentication exchange

MQTT protocol

Flag Bit:

table shows the flag value associated with each command. Here, reserved refers to future use, which means that it is not being used right now. In the case of PUBLISH command, flag bits are further divided into DUP, QoS, and RETAIN, where DUP is a duplicate delivery of a PUBLISH packet, QoS is Quality of Service, and RETAIN is retained message flag

Flags Bit

MQTT Control Packet	Fixed Header Flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reservd	0	0	0	0
CONNACK	Reservd	0	0	0	0
PUBLISH	Used in MQTTv5.0	DUP	QoS		RETAIN
PUBACK	Reservd	0	0	0	0
PUBREC	Reservd	0	0	0	0
PUBREL	Reservd	0	0	0	0
PUBCOMP	Reservd	0	0	0	0
SUBSCRIBE	Reservd	0	0	0	0
SUBACK	Reservd	0	0	0	0
UNSUBSCRIBE	Reservd	0	0	0	0
UNSUBACK	Reservd	0	0	0	0
PINGREQ	Reservd	0	0	0	0
PINGRESP	Reservd	0	0	0	0
DISCONNECT	Reservd	0	0	0	0
AUTH	Reservd	0	0	0	0

MQTT protocol

Remaining length

The remaining length is a variable-length integer that denotes the number of bytes remaining within the current control packet, including data in the variable header and the payload. Therefore, the remaining length is equal to the data in the variable header plus payload.

Remaining length = length of variable header + length of payload

For example, if the length of the variable header is 20 and the length of the payload is 30, then the remaining length is 50.

The remaining length can be used upto 4 bytes, and it starts from 2 bytes and can be used upto 4 bytes.

This field uses 7-bit for the lengths, and MSB bit can be used to continue a flag. If the continuation flag is 1, the next byte is also a part of the remaining length. If the continuation flag is 0, a byte is the last one of the remaining length.

Variable header

Some types of MQTT control packet types contain an optional field also, i.e., variable header component. This field resides between the fixed header and the payload. The content of the variable header depends upon the packet type. The variable header contains the packet identifier field, which is common in several packet types. The variable header component of many MQTT control packet types includes 2-byte integer, i.e., the packet identifier field.

MQTT protocol

The given list below contains the packet identifier field:

- PUBLISH
- PUBACK
- PUBREC
- PUBREL
- PUBCOMP
- SUBSCRIBE
- SUBACK
- UNSUBSCRIBE
- UNSUBACK

Key points related to the packet identifier field:

- A PUBLISH packet should not contain the packet identifier field if the value of QoS (Quality of Service) is set to zero. It implies that if the value of QoS is greater than zero, only the PUBLISH packet will contain the packet identifier field.
- When a client sends a new SUBSCRIBE, UNSUBSCRIBE, or PUBLISH MQTT control packet, it should assign a non-zero packet identifier that is currently unused.
- When a server sends a new PUBLISH MQTT control packet, it should assign a non-zero packet identifier that is currently unused.
- A PUBACK, PUBREC, PUBUREL, PUBREC are the acknowledgment packets of PUBLISH command that contain the same packet identifier as the PUBLISH packet.
- A SUBACK and UNSUBACK are the acknowledgment packets of SUBSCRIBE and UNSUBSCRIBE, respectively. Both the packets, i.e., SUBACK and UNSUBACK, use the same packet identifier as the SUBSCRIBE and UNSUBSCRIBE packets.
- The packet identifier can be reusable after processing the corresponding acknowledgment packet. It can be defined as follows:

If the value of QoS is 1 then the acknowledgment packet of PUBLISH would be PUBACK. If it processes the PUBACK, then the packet identifier of PUBACK can be reused.

If the QoS value is 2 then the acknowledgment packet of PUBLISH would be either PUBCOMP or PUBREC.

MQTT protocol

Payload

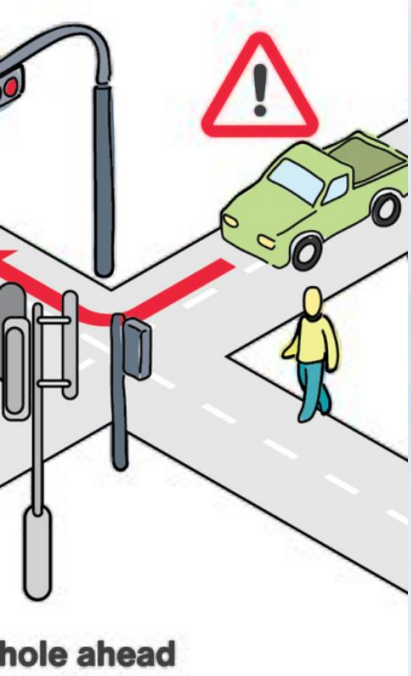
In the ICMP message format, the last MQTT control packet is the payload. This field contains the data which is to be sent. For example, in the case of the CONNECT packet, the payload is a client ID, the username and password, and the PUBLISH packet, the payload is just an application message.

MQTT control Packet that contain Payload

MQTT Control Packet	Payload
CONNECT	Required
CONNACK	None
PUBLISH	Optional
PUBACK	None
PUBREC	None
PUBREL	None
PUBCOMP	None
SUBSCRIBE	Required
SUBACK	Required
UNSUBSCRIBE	Required
UNSUBACK	Required
PINGREQ	None
PINGRESP	None
DISCONNECT	None
AUTH	None

Comparison of different IoT tools and platforms

Criteria	oneM2M	LoRaWAN	MQTT
Interoperability	High	Low	Medium
Range	Medium	High	Low



Factors to consider when choosing an IoT tool for your project

1 **Scalability**

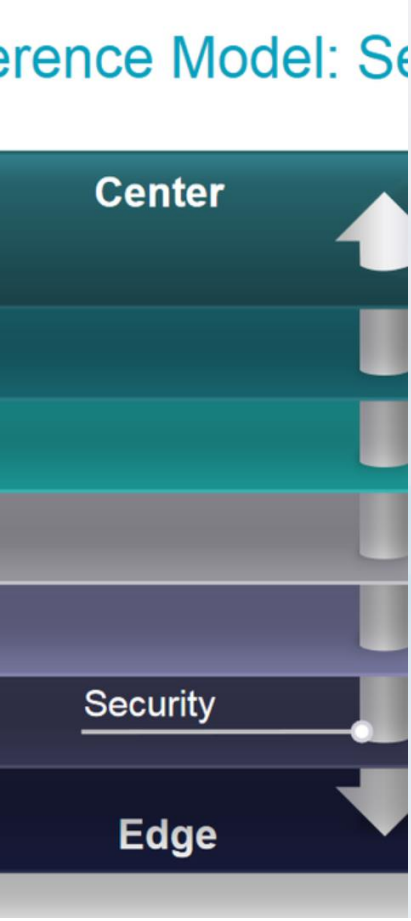
Determine if the platform can scale with the growth of your IoT deployment and accommodate future requirements.

2 **Security Features**

Evaluate the robustness of security measures to protect sensitive data and ensure the integrity of IoT systems.

3 **Integration Capabilities**

Assess the tool's compatibility and integration options with existing systems and devices in devices in your environment.



Simplified IoT Architecture: Core Functional Stack

Data Acquisition

The foundational layer involves collecting collecting data from sensors and devices, devices, which serve as the primary source source of information in the IoT ecosystem.

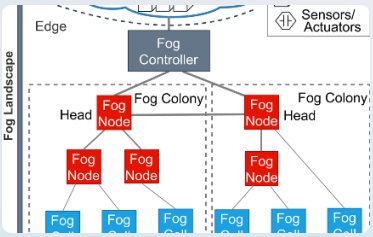
Data Processing

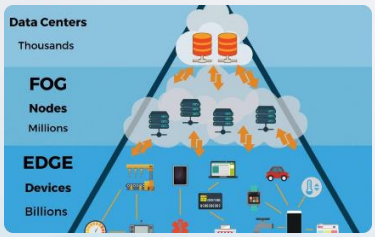
Once collected, data undergoes processing to extract meaningful insights, enabling informed decision-making and automated actions.

Data Application

Finally, the processed data is used to derive applications, services, and experiences that benefit that benefit individuals and organizations.

FOG, EDGE AND CLOUD IN IoT





Fog Computing

Fog computing extends cloud capabilities to the edge of the network, reducing latency and and enabling real-time data processing and analysis.

Edge Computing

Edge computing brings computation and data storage storage closer to the source of of data generation, optimizing optimizing network traffic and and enhancing efficiency.

Cloud Computing

Cloud computing forms the foundational infrastructure for for IoT, providing scalable and secure storage for vast amounts amounts of data generated by by connected devices.

FOG Computing

Fog Computing, also known as Edge Computing, is a decentralized computing architecture that brings computing resources closer to the edge of the network, closer to where data is generated, processed, and consumed. In fog computing, instead of relying solely on centralized cloud servers, some computing tasks are performed on local devices or "fog nodes" situated at the network's edge. This approach is particularly relevant in scenarios where low latency, real-time processing, and efficient use of bandwidth are crucial. Here are key aspects of fog computing:

1.Proximity to Edge Devices:

1. Fog computing places computing resources closer to the edge devices (sensors, IoT devices, etc.) that generate and consume data. This reduces the latency associated with sending data to a distant cloud server for processing.

2.Distributed Architecture:

1. Fog computing distributes computing tasks across a network of devices, which can include routers, switches, gateways, and other edge devices. Each device in the network becomes a potential fog node.

3.Real-Time Processing:

1. By processing data locally at the edge, fog computing enables real-time or near-real-time analysis of information. This is particularly important for applications that require instant responses, such as in industrial automation, healthcare monitoring, or autonomous vehicles.

4.Bandwidth Efficiency:

1. Fog computing helps reduce the amount of data that needs to be transmitted to the centralized cloud. Only essential data or aggregated results may be sent to the cloud, optimizing bandwidth usage and lowering the load on the network.

5.Scalability:

1. Fog computing allows for easy scalability by adding or removing fog nodes based on the requirements of the specific edge environment. This flexibility is beneficial for dynamic and evolving IoT ecosystems.

Cloud Computing

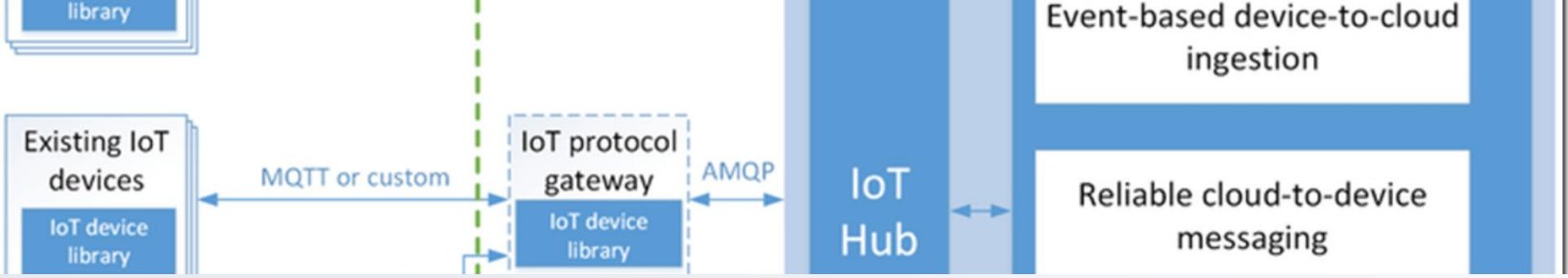
Cloud computing is a technology paradigm that involves delivering computing services, including storage, processing power, and software applications, over the internet. Instead of owning and maintaining physical servers and data centers, users can access and use computing resources on a pay-as-you-go basis through a network connection. This model offers several advantages, including scalability, flexibility, cost-efficiency, and accessibility. Here are key aspects of cloud computing:

1.Service Models:

1. **Infrastructure as a Service (IaaS):** Provides virtualized computing resources over the internet. Users can rent virtual machines, storage, and networks on a pay-as-you-go basis.
2. **Platform as a Service (PaaS):** Offers a platform that includes tools and services for application development, reducing the need for users to manage underlying infrastructure.
3. **Software as a Service (SaaS):** Delivers software applications over the internet, eliminating the need for users to install, manage, and maintain the software locally.

2.Deployment Models:

1. **Public Cloud:** Services are provided by third-party cloud service providers and are accessible to the general public. Resources are shared among multiple users.
2. **Private Cloud:** Cloud infrastructure is used exclusively by a single organization. It can be managed by the organization itself or by a third party.
3. **Hybrid Cloud:** Combines elements of both public and private clouds, allowing data and applications to be shared between them. This model provides greater flexibility and optimization.



Functional Blocks of an IoT - Sensors, Actuators, Smart Objects

1 Sensors

These devices detect and measure physical properties, converting them into electrical signals for use in monitoring and control applications.

2 Actuators

Actuators are components components that respond respond to signals from from the IoT system, controlling physical devices and processes based on input received. received.

3 Smart Objects

Smart objects integrate sensors, actuators, and communication capabilities, playing a pivotal role in IoT deployments and applications.

Functional Blocks of IoT

In an IoT (Internet of Things) system, the functional blocks include various components that work together to enable the communication, data processing, and functionality of the interconnected devices. Let's break down the functional blocks you mentioned:

1.Sensors:

1. **Function:** Sensors are devices that collect data from the physical world. They measure various parameters such as temperature, humidity, light, motion, or other environmental conditions.
2. **Examples:** Temperature sensors, humidity sensors, motion sensors, and more.

2.Actuators:

1. **Function:** Actuators are devices that perform physical actions based on received data. They can execute commands, control physical processes, or manipulate the environment.
2. **Examples:** Motors, servos, relays, and valves.

3.Smart Objects:

1. **Function:** Smart objects refer to physical devices or things that are embedded with sensors, actuators, and processing capabilities. These objects can collect data, make decisions, and interact with their environment or other devices.
2. **Examples:** Smart thermostats, wearable devices, connected appliances, and industrial machinery.

Connecting Smart Objects

•**Communication Protocols:** To enable communication between smart objects, various communication protocols and technologies are used. These protocols facilitate the exchange of data and commands between devices.

•**Examples:** MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), HTTP/HTTPS, Zigbee, Bluetooth, and LoRaWAN.

•**Middleware:** Middleware components play a role in managing communication between smart objects. They may include message brokering, device discovery, and other services to ensure efficient and reliable communication.

•**Device Management:** This involves functionalities related to registering, configuring, monitoring, and maintaining smart objects within the IoT system.

•**Security Measures:** Security protocols are implemented to secure the communication between smart objects. This includes authentication, encryption, and measures to protect against unauthorized access.

•**Edge Computing:** In some cases, data processing may occur at the edge of the network, closer to the smart objects. This can reduce latency and improve real-time responsiveness.

IoT Real World Applications

1K

Industrial Automation

IoT drives automation in manufacturing, improving productivity, efficiency, and predictive maintenance of machinery and equipment.

25M

Connected Vehicles

IoT-enabled vehicles enhance safety, navigation, and connectivity, offering features such as remote diagnostics and and autonomous driving capabilities.

5B

Smart Home Devices

The proliferation of IoT has resulted in smart home devices devices that offer convenience, convenience, energy savings, and and security features for homeowners.

Future Trends in IoT

Innovative Applications	IoT will witness innovation in various sectors, including healthcare, agriculture, and environmental monitoring.
AI Integration	Integration of artificial intelligence will enable IoT devices to make autonomous decisions and deliver more personalized experiences.
Security Enhancement	There will be a greater focus on enhancing IoT IoT security, leading to robust measures to protect protect data and devices from cyber threats.