UNIT II IOT Protocol

Standardization for IoT – Efforts – M2M and WSN Protocols – SCADA and RFIDProtocols – Issues with IoT Standardization – Unified Data Standards – Protocols – IEEE802.15.4–BACNet Protocol– Modbus – KNX – Zigbee– Network layer – APS layer – Security.

M2M and WSN Protocols

Machine-to-Machine (M2M) communication and Wireless Sensor Networks (WSN) are two fundamental aspects of the Internet of Things (IoT) ecosystem, each with its own set of protocols and standards. Let's explore M2M and WSN protocols in the context of IoT:

M2M-Machine to Machine

M2M communication refers to the exchange of data between machines or devices without human intervention. It is a crucial component of IoT as it enables devices to collect, transmit, and act upon data autonomously.

(OR)

M2M (Machine-to-Machine) protocols in IoT (Internet of Things) are communication protocols designed to enable devices and machines to exchange data and information without human intervention. These protocols play a crucial role in facilitating the automated flow of data between IoT devices, sensors, and systems. M2M protocols are essential for various IoT applications, including industrial automation, smart cities, healthcare, agriculture, and more.

Here are some key aspects of M2M protocols in IoT:

Purpose: M2M protocols serve the purpose of connecting and enabling communication between IoT devices and systems, allowing them to collect data, monitor conditions, and trigger actions based on predefined rules or algorithms.

Efficiency: Many M2M protocols are designed to be lightweight and efficient, making them suitable for devices with limited processing power, memory, and energy resources. This is essential for IoT devices that need to conserve energy and operate on battery power for extended periods.

Real-time and Asynchronous Communication: M2M protocols support both real-time and asynchronous communication. Real-time communication is crucial for applications that require immediate responses, while asynchronous communication allows devices to exchange data when it's convenient, without the need for constant connectivity.

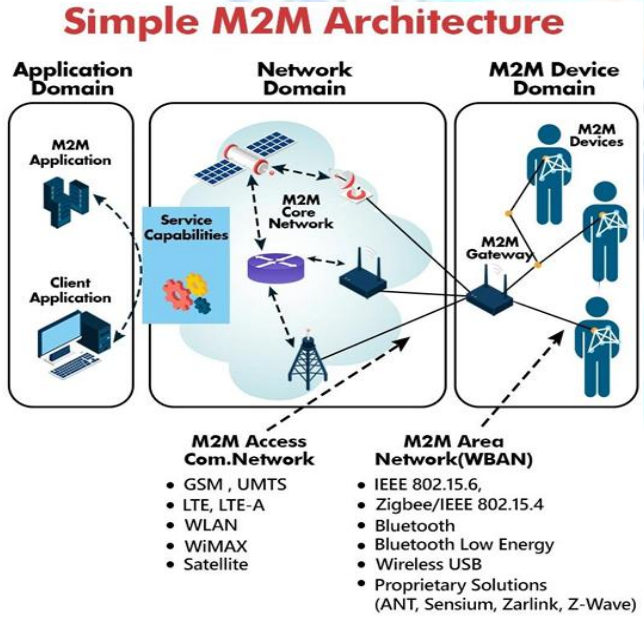
Message Formats: M2M protocols define message formats and structures that devices use to convey information. These formats are typically optimized for efficient data transfer, and they can include various data types, such as text, binary, or multimedia.

Security: Security is a paramount concern in M2M communication, as IoT devices often handle sensitive data. M2M protocols may incorporate security features like encryption, authentication, and access control to protect data in transit and at rest.

Connectivity: M2M protocols are adaptable to various communication networks, including wired, wireless, and cellular. They can operate over short-range and long-range networks, depending on the specific requirements of the IoT application.

Scalability: M2M protocols are designed to accommodate the growing number of IoT devices and the increasing volume of data they generate. They should be scalable to handle large deployments with ease.

- An M2M area network comprises of machines which have embedded hardware module for sensing actuation and communication. Various Communication protocols can be used for M2M local area network such as Zigbee, Bluetooth, Modbus M–bus, wireless, power LINE Communication ,6LoWPAN.
- These Communications protocols provide connectivity between M2M nodes within and M2M area network. The Communications Network provides connectivity to remote m2m area network. communication network can use wired or wireless network. The M2M area network use either proprietary or non-IP based protocol.



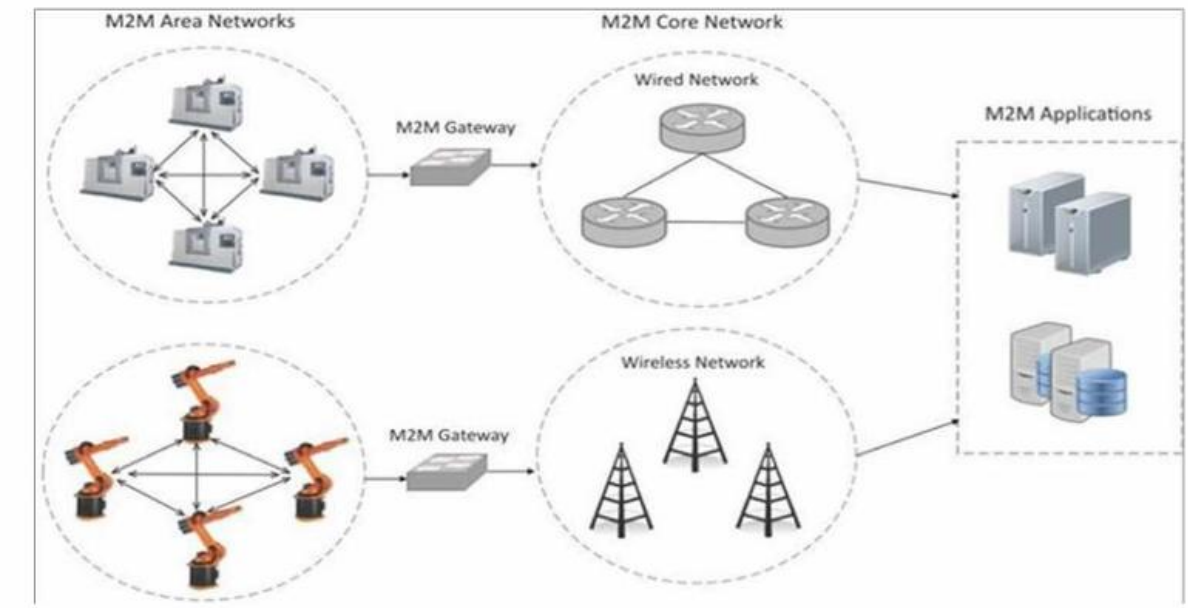
- The communication between the M2M nodes and the M2M Gateway is based on the communication protocol. M2M Gateway protocol translation to enable IP connectivity for M2M. M2M Gateway act as a proxy performing translation from / to native protocol to M2M area network.

- M2M data is gathered into point solution such as enterprise applications, service management application for remote monitoring applications. M2M has various application domain such as smart metering, Home Automation, industrial Automation, smart grid.

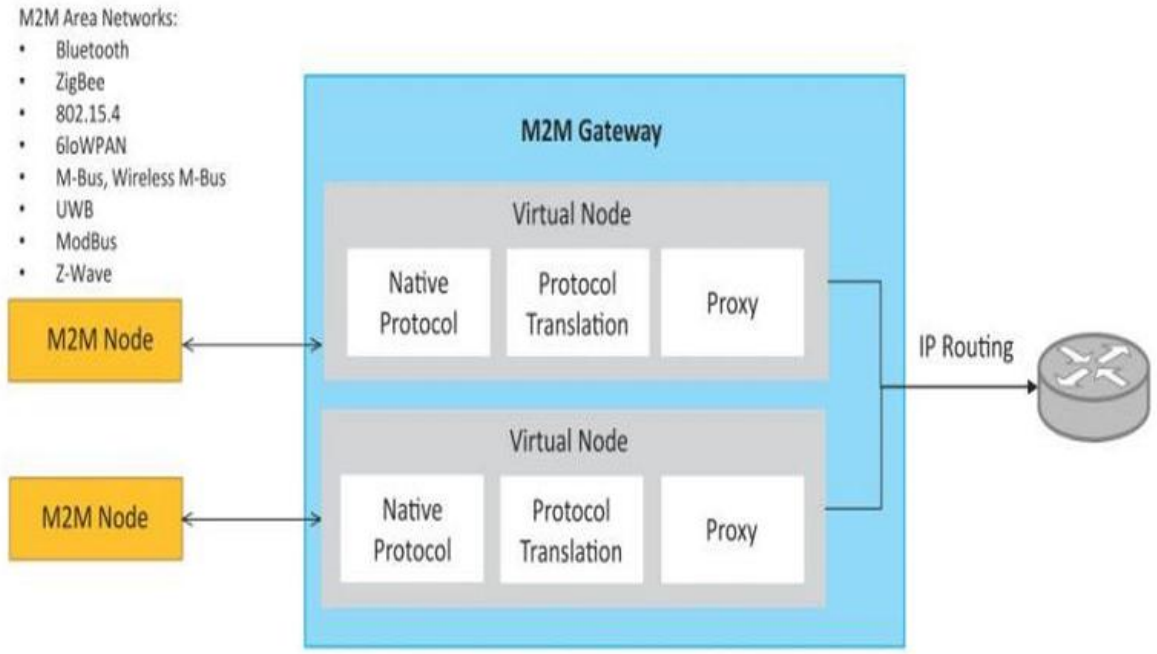
Machine-to-Machine (M2M) refers to networking of machines (or devices) for the purpose of remote monitoring and control and data exchange.

- Term which is often synonymous with IoT is Machine-to-Machine (M2M).
- IoT and M2M are often used interchangeably.

Fig. Shows the end-to-end architecture of M2M systems comprises of M2M area networks, communication networks and application domain.

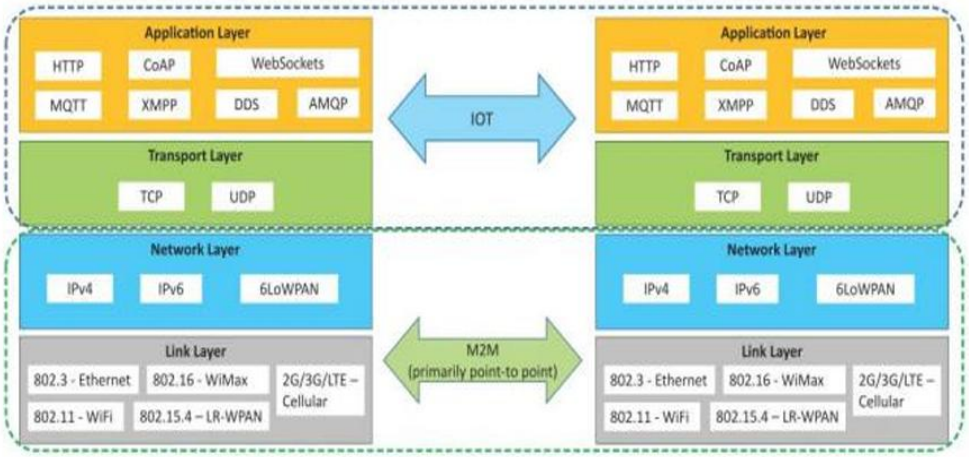


- An M2M area network comprises of machines (or M2M nodes) which have embedded network modules for sensing, actuation and communicating various communication protocols can be used for M2M LAN such as ZigBee, Bluetooth, M-bus, Wireless M-Bus etc., These protocols provide connectivity between M2M nodes within an M2M area network.
- The communication network provides connectivity to remote M2M area networks. The communication network provides connectivity to remote M2M area network. The communication network can use either wired or wireless network (IP based). While the M2M are networks use either proprietary or non-IP based communication protocols, the communication network uses IP-based network. Since non-IP based protocols are used within M2M area network, the M2M nodes within one network cannot communicate with nodes in an external network.
- To enable the communication between remote M2M are network, M2M gateways are used.



- Fig. Shows a block diagram of an M2M gateway. The communication between M2M nodes and the M2M gateway is based on the communication protocols which are naive to the M2M are network. M2M gateway performs protocol translations to enable Ip-connectivity for M2M are networks. M2M gateway acts as a proxy performing translations from/to native protocols to/from Internet Protocol (IP). With an M2M gateway, each mode in an M2M area network appears as a virtualized node for external M2M area networks.

Communication in IoT vs M2M



Differences between IoT and M2M:

1) Communication Protocols:

- Commonly uses M2M protocols include ZigBee, Bluetooth, ModBus, M-Bus, Wireless M-Bus tec.,
- In IoT uses HTTP, CoAP, WebSocket, MQTT, XMPP, DDS, AMQP etc.,

2) Machines in M2M Vs Things in IoT:

- Machines in M2M will be homogenous whereas Things in IoT will be heterogeneous.

3) Hardware Vs Software Emphasis:

- the emphasis of M2M is more on hardware with embedded modules, the emphasis of IoT is more on software.

4) Data Collection &Analysis

- M2M data is collected in point solutions and often in on-premises storage infrastructure.
- The data in IoT is collected in the cloud (can be public, private or hybrid cloud).

5) Applications

- M2M data is collected in point solutions and can be accessed by on-premises applications such as diagnosis applications, service management applications, and on- premises enterprise applications.
- IoT data is collected in the cloud and can be accessed by cloud applications such as analytics applications, enterprise applications, remote diagnosis and management applications, etc.

Protocols used in M2M communication in IoT:

MQTT (Message Queuing Telemetry Transport) 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 a constant environment such as for communication machine to machine and internet of things contexts. I
- t 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.

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

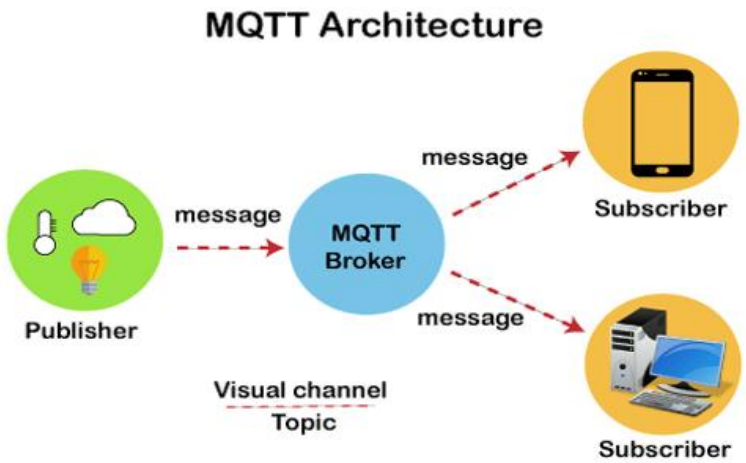
- It is a machine-to-machine protocol, i.e., it provides communication between the devices.

- 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.

MQTT Architecture

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

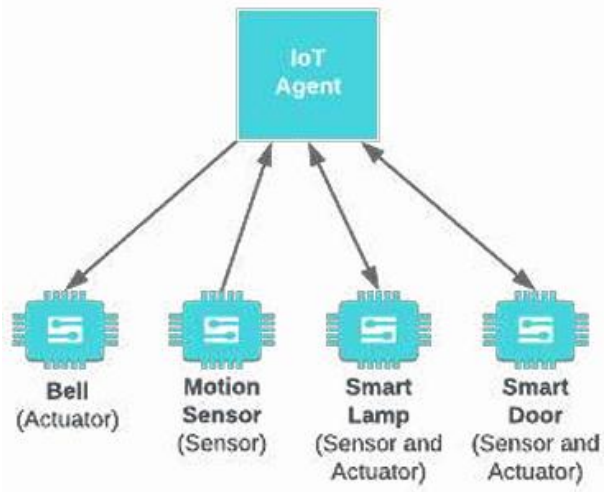
- **Message**: The message is the data that is carried out by the protocol across the network for the application. When the message is transmitted over the network, then the message contains the following parameters:
  - Pay load data
  - Quality of Service (QoS)
  - Collection of Properties
  - Topic 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
- **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 as TOPIC.

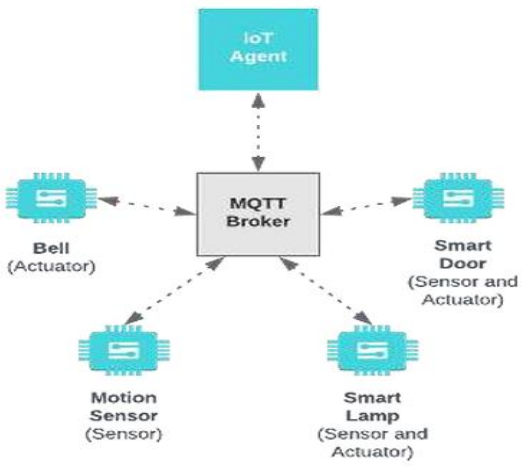


Suppose a device has a temperature sensor and wants to send the rating to the server or the broker. If the phone or desktop application wishes to receive this temperature value on the other side, then there will be two things that happened. The publisher first defines the topic; for example, the temperature 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. The differences between the two transport protocols can be seen below:

IoT Agent communicates with IoT devices directly	IoT Agent communicates with IoT devices indirectly via an MQTT Broker
Request Response Paradigm	Publish-Subscribe Paradigm
IoT Devices must always be ready to receive communication	IoT Devices choose when to receive communication
Higher Power Requirement	Low Power Requirement



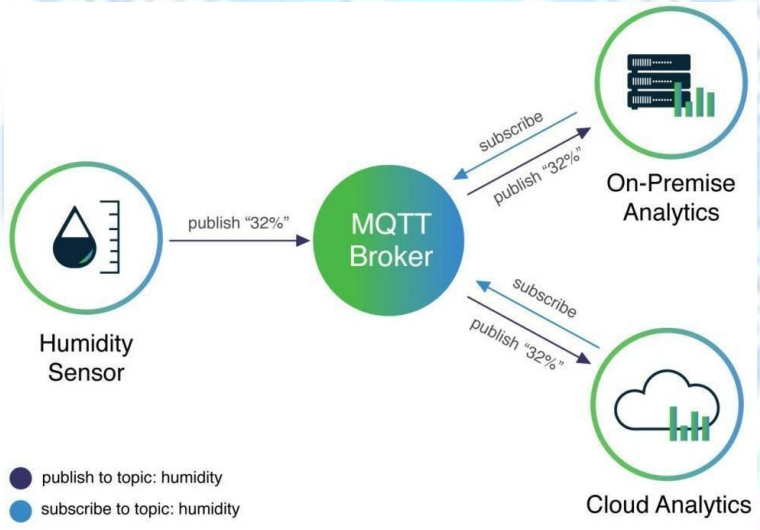


MOSQUITTO MQTT BROKER

Mosquitto is a readily available, open source MQTT broker which will be used during this tutorial. It is available licensed under EPL/EDL. More information can be found at https://mosquitto.org/

Publish/Subscribe Model

Unlike the traditional client-server model, in which a client communicates directly with an endpoint, MQTT clients are split into two groups: A sender (referred to as a publisher in MQTT) and a consumer that receives the data (an MQTT subscriber). The publisher and the subscriber do not know anything about each other, and, in fact, are never in direct contact with each other. A third component (an MQTT broker), acts like a 'traffic cop', directing messages from the publisher to any end points acting as subscribers.



The key benefits of MQTT are:

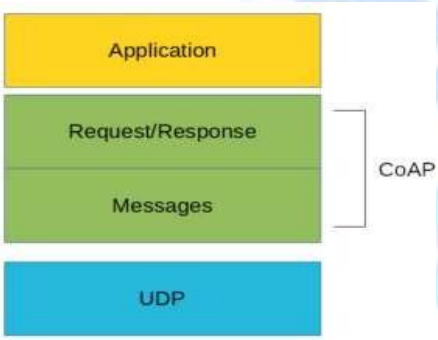
1. Lightweight and efficient to minimize resources required for the client and network bandwidth.
2. Enables bidirectional communication between devices and servers. Also, enabling broadcasting messages to groups of things.
3. Scales to millions of things.
4. MQTT specifies Quality of Service (QoS) levels to support message reliability.
5. MQTT supports persistent sessions between device and server that reduces reconnection time required over unreliable networks.
6. MQTT messages can be encrypted with TLS and support client authentication protocols.

CoAP:

- Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. CoAP is designed to enable simple, constrained devices to join the IoT even through constrained networks with low bandwidth and low availability.
- It is generally used for machine-to-machine (M2M) applications such as smart energy and building automation. The protocol was designed by the Internet Engineering Task Force (IETF).

- CoAP is a customary client-server IoT protocol. It enables clients to make requests for web transfers as per the need of the hour. On the other hand, it also lets supporting servers to respond to arriving requests. In summary, devices' nodes in the IoT ecosystem are enabled to interact over through CoAP only.
- CoAP is a simple protocol with low overhead specifically designed for constrained devices (such as microcontrollers) and constrained networks. This protocol is used in M2M data exchange and is very similar to HTTP
- CoAP and HTTP follow the same working procedure. However, CoAP attains its functionality via asynchronous transactions (using UDP). It utilizes the POST, GET, PUT, and DELETE calls.

From the abstraction protocol layer, CoAP can be represented as:



As you can see there are two different layers that make CoAp protocol: Messages and Request/Response. The Messages layer deals with UDP and with asynchronous messages. The Request/Response layer manages request/response interaction based on request/response messages.

CoAP is compatible with 4 types of information exchange:

1. **Acknowledgments** confirm the completion or failure of an event.

2. **Confirmable** are the messages that are resent on time out until the confirmation of successful sending doesn't arrive.
3. **Reset** messages are empty, with confirmable as their nature.
4. **Non-confirmable** information is just sent and has no guarantee of successful delivery. There is no acknowledgment of success either.

Endpoint: An entity that participates in the CoAP protocol. Usually, an Endpoint is identified with a host

Sender: The entity that sends a message

Recipient: The destination of a message

Client: The entity that sends a request and the destination of the response

Server: The entity that receives a request from a client and sends back a response to the client

CoAP Messages Model

This is the lowest layer of CoAP. This layer deals with UDP exchanging messages between endpoints. Each CoAP message has a unique ID; this is useful to detect message duplicates. A CoAP message is built by these parts:

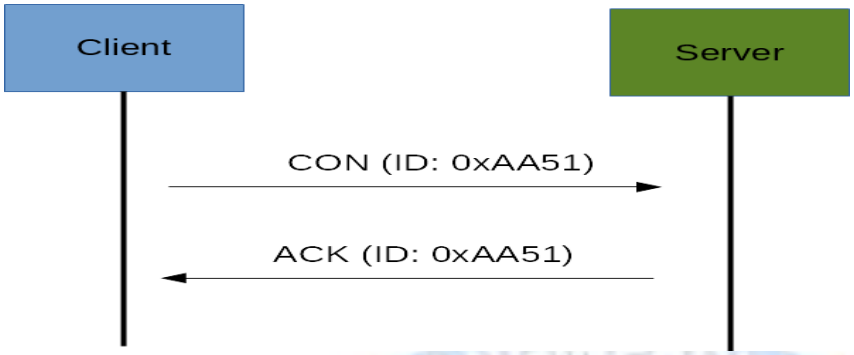
1. **A binary header**
2. **A compact option**
3. **Payload**

As said before, the CoAP protocol uses two kinds of messages:

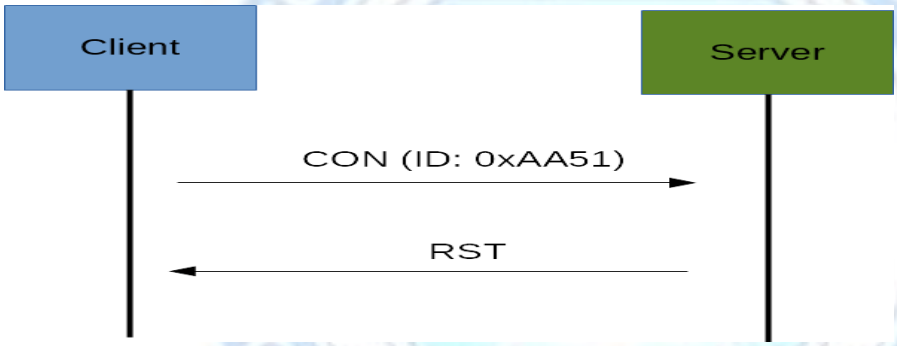
1. **Confirmable message**
2. **Non-confirmable message**

A confirmable message is a reliable message. When exchanging messages between two endpoints, these messages can be reliable. In CoAP, a reliable message is obtained using a Confirmable message (CON). Using this kind of message, the client can be sure that the message will arrive at the server. A Confirmable message is sent again and again until the other party sends an acknowledge message (ACK). The ACK message contains the same ID of the confirmable message (CON).

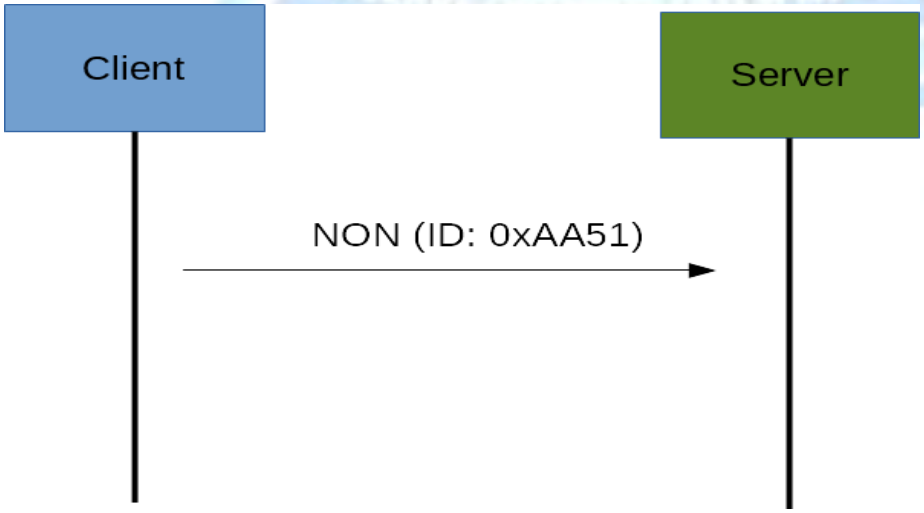
The picture below shows the message exchange process:



If the server has troubles managing the incoming request, it can send back a Rest message (RST) instead of the Acknowledge message (ACK):



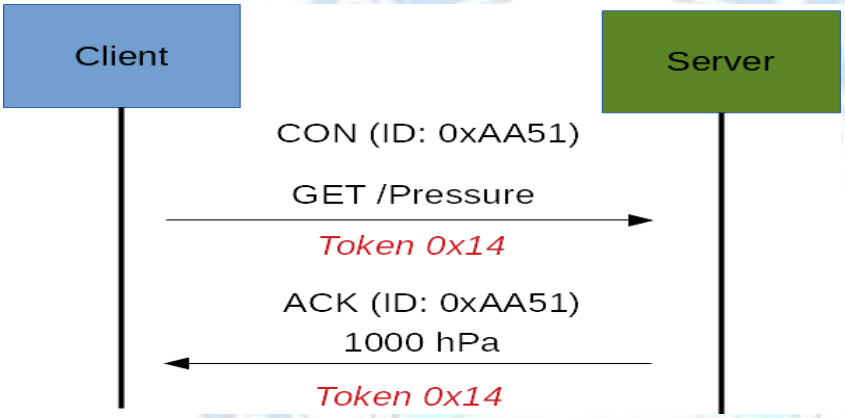
The other message category is the Non-confirmable (NON) messages. **These are messages that don't require an Acknowledge by the server.** They are unreliable messages or in other words messages that do not contain critical information that must be delivered to the server. To this category belongs messages that contain values read from sensors.



CoAp Request/Response Model

The CoAP Request/Response is the second layer in the CoAP abstraction layer. The request is sent using a Confirmable (CON) or Non-Confirmable (NON) message. There are several scenarios depending on if the server can answer immediately to the client request or the answer if not available.

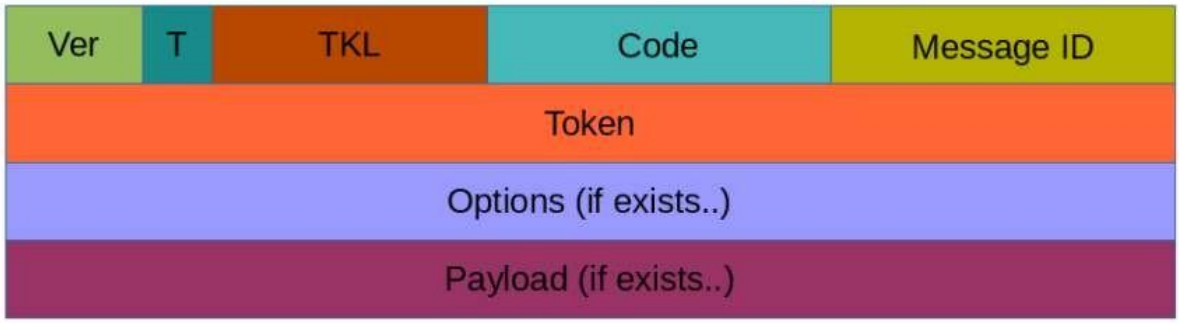
If the server can answer immediately to the client request, then if the request is carried using a Confirmable message (CON), the server sends back to the client an Acknowledge message containing the response or the error code:



Even if these messages are unreliable, they have a unique ID.

CoAp Message Format

This paragraph covers the CoAP Message format. By now, we have discussed different kinds of messages exchanged between the client and the server. Now it is time to analyze the message format. The constrained application protocol is the meat for constrained environments, and for this reason, it uses compact messages. To avoid fragmentation, a message occupies the data section of a UDP datagram. A message is made by several parts:



Where:

Ver: It is a 2 bit unsigned integer indicating the version

T: it is a 2bit unsigned integer indicating the message type: 0 confirmable, 1 non-confirmable

TKL: Token Length is the token 4bit length

Code: It is the code response (8bit length)

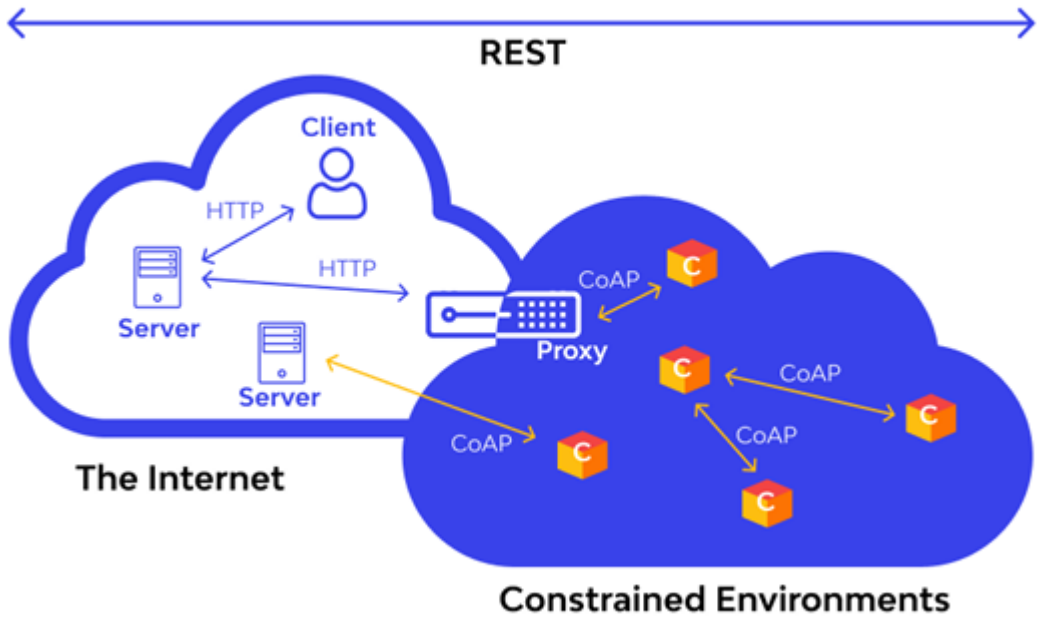
Message ID: It is the message ID expressed with 16bit

Key traits of CoAP are:

- Works for devices in the same network types.
- Enables data transmission, to and fro, for the general internet-enabled nodes and network-connected devices.
- Works really fine for SMSs shared over mobile network connectivity.
- Suitable for internet-operative applications that use connected devices/sensors and have resource limitations.
- Capable of translating HTTP, supports multicast, and exerts the bare minimum cost burden.
- Only helps machines to communicate (in the network)

CoAP Architecture

- The WWW and the constraints ecosystem are the 2 foundational elements of the CoAP protocol architecture. Here, the server monitors and helps in communication happening using CoAP and HTTP while proxy devices bridge the existing gap for these 2 ecosystems, making the communication smoother.
- CoAP allows HTTP clients (also called CoAP clients here) to talk or exchange data/information with each other within resource constraints.
- While one tries to understand this architecture, gaining acquaintances with some key terms is crucial:
  - Endpoints are the nodes that host have knowledge of;
  - Client sends requests and replies to incoming requests;
  - Server gets and forwards requests. It also gets and forwards the messages received in response to the requests it had processed.
  - Sender creates and sends the original message.
  - Recipient gets the information sent by the client or forwarded by the server



6LoWPAN

What is 6LoWPAN - the basics

6LoWPAN provides the upper layer system for use with low power wireless communications for IoT and M2M, originally intended for 802.15.4, it is now used with many other wireless standards.

The 6LoWPAN system is used for a variety of applications including wireless sensor networks. This form of wireless sensor network sends data as packets and using IPv6 - providing the basis for the name - IPv6 over Low power Wireless Personal Area Networks.

6LoWPAN provides a means of carrying packet data in the form of IPv6 over IEEE 802.15.4 and other networks. It provides end-to-end IPv6 and as such it is able to provide direct connectivity to a huge variety of networks including direct connectivity to the Internet.

In this way, 6LoWPAN adopts a different approach to the other low power wireless sensor network solutions.

6LoWPAN interoperability

One key issue of any standard is that of interoperability. It is vital that equipment from different manufacturers operates together.

When testing for interoperability, it is necessary to ensure that all layers of the OSI stack are compatible. To ensure that this can be achieved there several different specifications that are applicable. Any item can be checked to conform it meets the standard, and also directly tested for interoperability.

6LoWPAN is a wireless / IoT style standard that has quietly gained significant ground. Although initially aimed at usage with IEEE 802.15.4, it is equally able to operate with other wireless standards making it an ideal choice for many applications.

6LoWPAN uses IPv6 and this alone has to set it aside from the others with a distinct advantage. With the world migrating towards IPv6 packet data, a system such 6LoWPAN offers many advantages for low power wireless sensor networks and other forms of low power wireless networks.

6LoWPAN wireless modules offer:

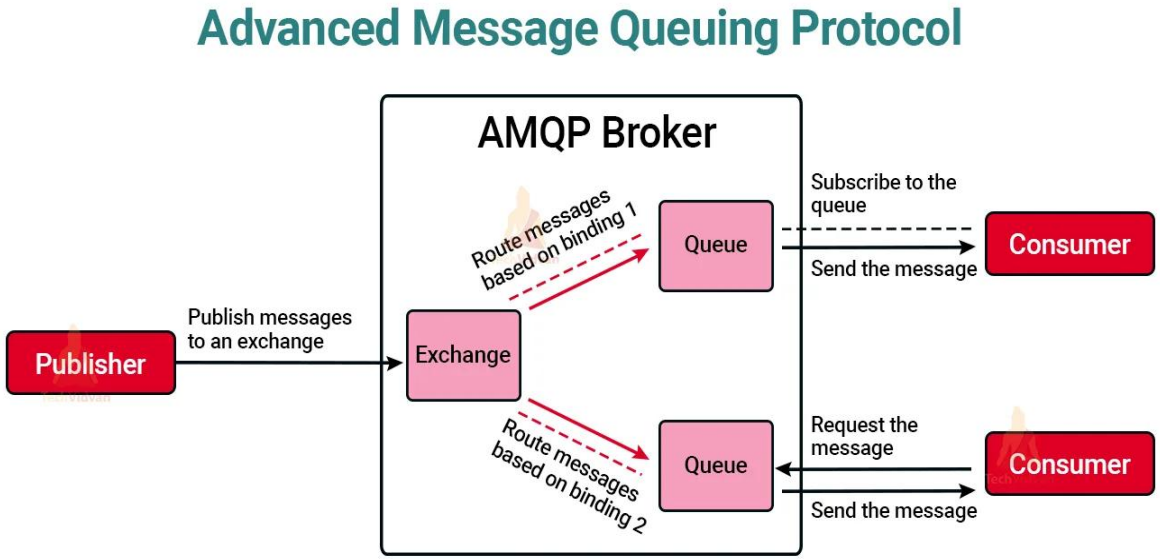
- Compatibility with all IPv6 standards – Native communication with computers of mobile devices through IEEE 802.15.4
- Lower power – Maximizes battery life
- Small footprint – Compact modules ideal for solutions with size constraints
- FCC and CE certification – Sub-GHz transceiver module is certified for use in North America and Europe
- Standard firmware – Features IDT SensorShare™ firmware, a 6LoWPAN protocol open standard stack, with no associated license fees and no royalties
- Mesh routing features – Self-healing ad-hoc mesh network to cover large areas and long ranges
- End-to-end security – Secure communication based on open standard protocols

BENEFITS OF USING 6LOWPAN IN YOUR APPLICATIONS:

- **Efficient** use of IPv6 over low-power wireless networks on simple embedded devices.
- Ideal to create mesh networks, it carries IPv6 or v4 data packets over the IEEE 802.15.4 standard. It provides end-to-end IP, while able to provide seamless connectivity to a huge variety of networks using the same standard including direct connectivity to the Internet.
- Low data rates: In this way, 6LoWPAN adopts a different approach to the other low power wireless sensor network solutions. The overall system aims at providing wireless internet connectivity at low data rates.
- Low Power: IPv6 provides a basic transport mechanism to produce complex control systems and to communicate with devices in a cost-effective manner via a low-power wireless network.

Advanced Message Queuing Protocol (AMQP)

AMQP transfers business messages between various applications and companies. This protocol is not specifically built for IOT applications but has a wide range of uses in the internet of things. But, it works effectively in transferring message communication. It connects the system, serves the system with the required information and forwards the information to achieve the necessary goals.



AMQP connects across various systems, technologies, time and space. There are three parts which manage the entire AMQP protocol:

- **Exchange:** receives messages from publishers and directs these messages to the messaging queues depending on the availability.
- **Message Queue:** Saves these messages in databases until they are put in use by an application.
- **Binding:** Distinguishes the relation between an Exchange and Message Queue and gives the message steering criteria.

Features of AMQP

- Application layer protocol is binary
- Cabe used as point to point or publish/subscribe messaging
- Highly suits messaging scenarios
- Aids end to end encryption in communication

Pros of AMQP

- Transfer messages through TCP or UDP
- End to end encryption

Cons of AMQP

- Uses large mounts of power and memory

Use Cases of AMQP

This protocol is mostly used in the business sector. It uses back-office data centres to specify devices like mobile phones, handsets etc.

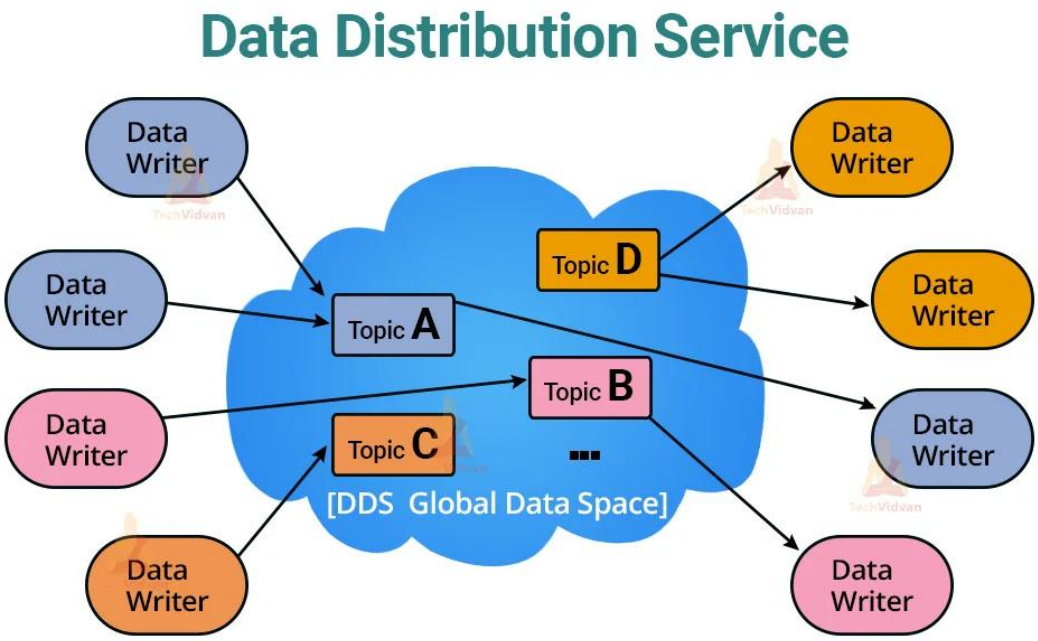
Data Distribution Service (DDS)

DDS protocol acts as a bridge between **databases** and **user applications** in a network and hence it is a **middleware** protocol. The protocol combines the parts of a system together. The protocol uses **low computing data**, it is highly efficient and reliable and its architecture is extremely extensible.

Since, DDS is a middleware software, its job is to provide effective communication and easy sharing of data. It handles the tiring and confusing job of managing the communication paths and allows developers to focus on building the applications.

DDS is a vital protocol whose main aim is **M2M(machine to machine) communication**. Data exchange happens through the well known **publish-subscribe methodology**. This protocol differs from the above two protocols as it is **brokerless.** It serves high quality QoS to the applications with the help of **multicasting.** DDS protocols are developed from low footprint devices to cloud.

DDS is **data centric** and is thus widely used in the internet of things technology, since IoT mainly involves the flow of data from one point to another. Data Centricity ensures that all messages include conceptual information and all applications require understanding the data that it receives.



Features of DDS

- Design's structure supports real time systems
- Publish/subscribe messaging
- Direct connection of devices
- Low overhead

Pros of DDS

- Easy architecture that aids "auto-discovery" of new applications
- Scalable and effective
- Effectively uses transport bandwidth
- Committed Data delivery

Cons of DDS

- DDS has a heavyweight protocol and it becomes difficult to use in embedded systems
- It consumes double the bandwidth as compared to MQTT
- It is not possible to interface with webpages

Use Cases of DDS

Hospitals and healthcare, military and borders, wind farms, tracking systems all use the DDS protocol.

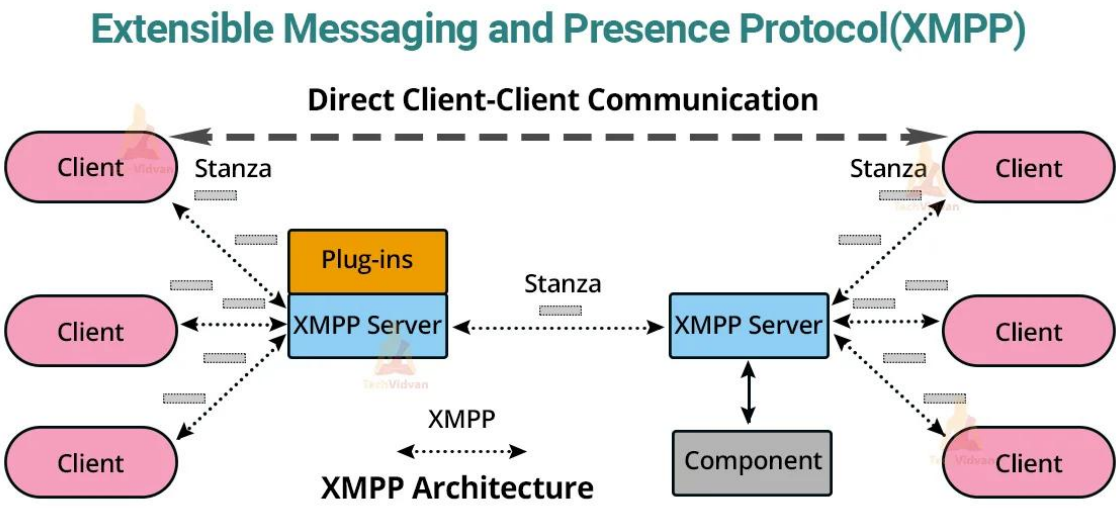
Extensible Messaging and presence protocol (XMPP)/Jabber

As the name suggests, XMPP protocol applies to **long distance messaging** due its scalability and it involves human presence or **human intervention**. XMPP originates from

XML, extensible markup language and XML originates from HTML, a protocol used to create web pages. Both are markup languages.

XMPP has a broad range of uses that can inter communicate with each other due to its **extensibility.** It uses the standard internet communication protocol (ICP) and this makes it universally communicable and it also communicates through HTTP. XMPP was created a long time ago and further extensions in its development make it more accessible.

XMPP networks contain gateways to connect to other protocols. It is designed in such a way that it supports instant message transfer through TCP link. This is highly useful in the internet of things technology.



Features of XMPP

- Extensible design
- Open standards
- Client/server architecture

Pros of XMPP

- Labelling scheme to locate devices on the entire network easily

Cons of XMPP

- End to end encryption not possible
- Quality of service not available

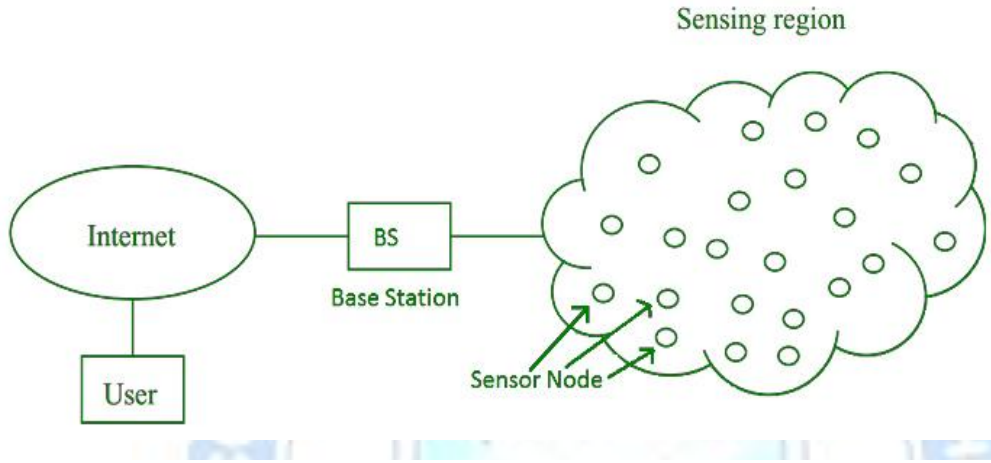
XMPP Use Cases

- Smartphones can access a smart thermostat via the internet
- A gaming console that allows gamers to message other gamers instantly

WSN (WIRELESS SENSOR NETWORK) PROTOCOLS:

Wireless Sensor Networks are specialized networks of spatially distributed sensors that monitor physical or environmental conditions and send collected data to a central system. In IoT, WSNs are frequently used for data acquisition from sensors.

Base Station in a WSN System is connected through the Internet to share data.



WSN can be used for processing, analysis, storage, and mining of the data.

Applications of WSN:

1. Internet of Things (IoT)
2. Surveillance and Monitoring for security, threat detection
3. Environmental temperature, humidity, and air pressure
4. Noise Level of the surrounding
5. Medical applications like patient monitoring
6. Agriculture
7. Landslide Detection

Challenges of WSN:

1. Quality of Service
2. Security Issue
3. Energy Efficiency
4. Network Throughput
5. Performance
6. Ability to cope with node failure
7. Cross layer optimisation
8. Scalability to large scale of deployment

A modern Wireless Sensor Network (WSN) faces several challenges, including:

- **Limited power and energy:**

WSNs are typically composed of battery-powered sensors that have limited energy resources. This makes it challenging to ensure that the network can function for long periods of time without the need for frequent battery replacements.

- **Limited processing and storage capabilities:** Sensor nodes in a WSN are typically small and have limited processing and storage capabilities. This makes it difficult to perform complex tasks or store large amounts of data.
- **Heterogeneity:** WSNs often consist of a variety of different sensor types and nodes with different capabilities. This makes it challenging to ensure that the network can function effectively and efficiently.
- **Security:** WSNs are vulnerable to various types of attacks, such as eavesdropping, jamming, and spoofing. Ensuring the security of the network and the data it collects is a major challenge.
- **Scalability:** WSNs often need to be able to support a large number of sensor nodes and handle large amounts of data. Ensuring that the network can scale to meet these demands is a significant challenge.
- **Interference:** WSNs are often deployed in environments where there is a lot of interference from other wireless devices. This can make it difficult to ensure reliable communication between sensor nodes.
- **Reliability:** WSNs are often used in critical applications, such as monitoring the environment or controlling industrial processes. Ensuring that the network is reliable and able to function correctly in all conditions is a major challenge.

Components of WSN:

1. **Sensors:**
   Sensors in WSN are used to capture the environmental variables and which is used for data acquisition. Sensor signals are converted into electrical signals.
2. **Radio Nodes:**
   It is used to receive the data produced by the Sensors and sends it to the WLAN access point. It consists of a microcontroller, transceiver, external memory, and power source.
3. **WLAN Access Point:**
   It receives the data which is sent by the Radio nodes wirelessly, generally through the internet.
4. **Evaluation Software:**
   The data received by the WLAN Access Point is processed by a software called as

Evaluation Software for presenting the report to the users for further processing of the data which can be used for processing, analysis, storage, and mining of the data.

Advantages of Wireless Sensor Networks (WSN):

Low cost: WSNs consist of small, low-cost sensors that are easy to deploy, making them a cost-effective solution for many applications.

Wireless communication: WSNs eliminate the need for wired connections, which can be costly and difficult to install. Wireless communication also enables flexible deployment and reconfiguration of the network.

Energy efficiency: WSNs use low-power devices and protocols to conserve energy, enabling long-term operation without the need for frequent battery replacements.

Scalability: WSNs can be scaled up or down easily by adding or removing sensors, making them suitable for a range of applications and **environments.**

Real-time monitoring: WSNs enable real-time monitoring of physical phenomena in the environment, providing timely information for decision making and control.

Disadvantages of Wireless Sensor Networks (WSN):

Limited range: The range of wireless communication in WSNs is limited, which can be a challenge for large-scale deployments or in environments with obstacles that obstruct radio signals.

Limited processing power: WSNs use low-power devices, which may have limited processing power and memory, making it difficult to perform complex computations or support advanced applications.

Data security: WSNs are vulnerable to security threats, such as eavesdropping, tampering, and denial of service attacks, which can compromise the confidentiality, integrity, and availability of data.

Interference: Wireless communication in WSNs can be susceptible to interference from other wireless devices or radio signals, which can degrade the quality of data transmission.

Deployment challenges: Deploying WSNs can be challenging due to the need for proper sensor placement, power management, and network configuration, which can require significant time and resources.