SHRI VILE PARLE KELAVANI MANDAL'S

SHRI BHAGUBHAI MAFATLAL POLYTECHNIC

COMPUTER ENGINEERING DEPARTMENT

1. COURSE DETAILS

Programme: CSE/IT	**Semester: IV/IV**
Course: # IoT & Applications	**Course Category: AEC**
Course Code: IOT238917	**Duration:16 Weeks**

2. LEARNING AND ASSESSMENT SCHEME

Learning Scheme				Credits	Assessment Scheme									
Actual Contact Hrs./Week			Self-Learning (SL^) (Term Work + Assignment) (Hrs)		Paper Duration (Hrs.)	Theory (Marks)			Based on LL & TL Practical (Marks)			Based on Self Learning	Total Marks
CL	TL	LL				FA-TH	SA-TH	Total	FA-PR	SA-PR	SA-OR	SLA (Marks)	
04	-	02	-	06	03	30	70	100	50	-	25	-	175

3. COURSE OBJECTIVE

IoT (Internet of Things) is an advanced automation and analytics system which exploits networking, sensing, big data, and artificial intelligence technology to deliver complete systems for a product or service. These systems allow greater transparency, control, and performance when applied to any industry or system. IoT systems have applications across industries through their unique flexibility and ability to be suitable in any environment.

4. SKILL COMPETENCY/INDUSTRY/EMPLOYER EXPECTED OUTCOME

- **Develop an IoT Application.**

5. COURSE OUTCOMES (COs): At the end of the semester student will be able to: -

CO No.	COURSE OUTCOME
CO1	Conceptualize the working of IoT system.
CO2	Discuss the architecture, operation, and business benefits of an IoT solution
CO3	Describe the working of sensors and actuators and their applications in real-world IoT scenarios.
CO4	Execute the setup of IoT boards and connect essential peripherals using sensors.
CO5	Summarize the challenges associated with IoT and cloud integration.
CO6	Demonstrate the utilization of IoT technology in a specific domain application

7. COURSE CONTENTS

UNIT NO.	Topic/ Sub-Topics
I	**Introduction to Internet of Things** 1.1 Introduction to Internet of Things 1.2 Definition of IoT 1.3 IoT Characteristics 1.4 Requirements of IoT 1.5 Physical design of IoT – IoT protocol 1.6 Logical Design of IoT 1.7 Functional blocks of IoT 1.8 Communication models and APIs platforms for IoT
II	**Architectural Overview of IoT** 2.1 IoT architecture – state of the art 2.2 IoT reference model 2.3 Architecture reference model 2.4 Introduction to M2M 2.5 Applications of M2M 2.6 M2M value Chains 2.7 IoT value chains 2.8 Emerging industrial structure for IoT.
III	**IoT Sensors and Actuators** 3.1 Need for sensors and actuators 3.2 Types of sensors and actuators 3.3 Introduction to Wireless Sensor Networks 3.4 SDN and NFV for IoT- Software-defined networking, network function virtualization.
IV	**Interfacing and Programming IoT Boards** 4.1 Introduction to IoT boards 4.2 Raspberry Pi specifications and features 4.3 Architecture of Raspberry Pi 4.4 Introduction to GPIO pins 4.5 Basic Programming with Python 4.6 Overview of NodeMCU Architecture 4.7 Setting Up NodeMCU Development Environment 4.8 Programming NodeMCU with Arduino IDE
V	**Cloud computing and data management** 5.1 Introduction to cloud computing 5.2 Characteristics of cloud 5.3 Cloud Deployment Models 5.4 Cloud service models 5.5 Fog computing and Edge computing 5.6 Data management in IoT 5.7 IoT analytics
VI	**Applications of IoT** 6.1 IoT applications for industry: - Future Factory Concepts - Brownfield IoT - IoT for Retailing Industry - IoT for Oil and Gas Industry - IoT for e-Health 6.2 Domain specific application – Home automation, Surveillance applications, Agriculture, smart cities.

COMPUTER ENGINEERING DEPARTMENT

8. LIST OF PRACTICALS/ASSIGNMENTS/ TUTORIALS/DRAWINGS

Term Work consists of Journal containing minimum 10 experiments/assignments with approx. no of hours required and corresponding CO attained are specified here:

Sr. No.	Title of Experiment/Assignment	Approx. Hrs required	COs
1.	To prepare a survey on various types of sensors & its application	02	CO1
2.	Familiarization with raspberry pi components and perform necessary software installation	02	CO2
3.	To interface LED/buzzer with raspberry pi and write program to turn on LED for specific time interval	02	CO3
4.	To interface push button with raspberry pi and write program to turn on LED.	02	CO3
5.	To interface camera with raspberry pi and write a program to capture an image.	02	CO3
6.	To interface raspberrypi with RFID reader.	02	CO4
7.	To Install and setup Arduino IDE and necessary libraries	02	CO3
8.	To interface LED/buzzer with NodeMCU and write program to turn on LED for specific time interval.	02	CO3
9.	To detect occupancy of an area using PIR sensors	02	CO4
10.	To interface DHT11 sensor with NodeMCU and write a program to print temperature readings.		CO4
11.	Create a web interface to control connected LEDs remotely using Raspberry-Pi/NodeMCU	02	CO3
12.	To send messages to whatsApp account with the ESP8266 NodeMCU board.	02	CO4
13	Introduction to MQTT and sending sensor data to cloud.	02	CO5
14	Case study on IoT applications for Industry	02	CO6
15	Case study on domain specific application of IoT	02	CO6
	TOTAL	30	

Chapter 1

Introduction to Internet of Things

Introduction to Internet of Things

- The internet of things, or IoT, is a network of interrelated devices that connect and exchange data with other IoT devices.
- IoT devices are typically embedded with technology such as sensors and software and can include mechanical and digital machines and consumer objects.
- These devices encompass everything from everyday household items to complex industrial tools.
- Increasingly, organizations in a variety of industries are using IoT to operate more efficiently, deliver enhanced customer service, improve decision-making and increase the value of the business.
- With IoT, data is transferable over a network without requiring human-to-human or human-to-computer interactions.
- A thing in the internet of things can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low, or any other natural or man-made object that can be assigned an Internet Protocol address and can transfer data over a network.

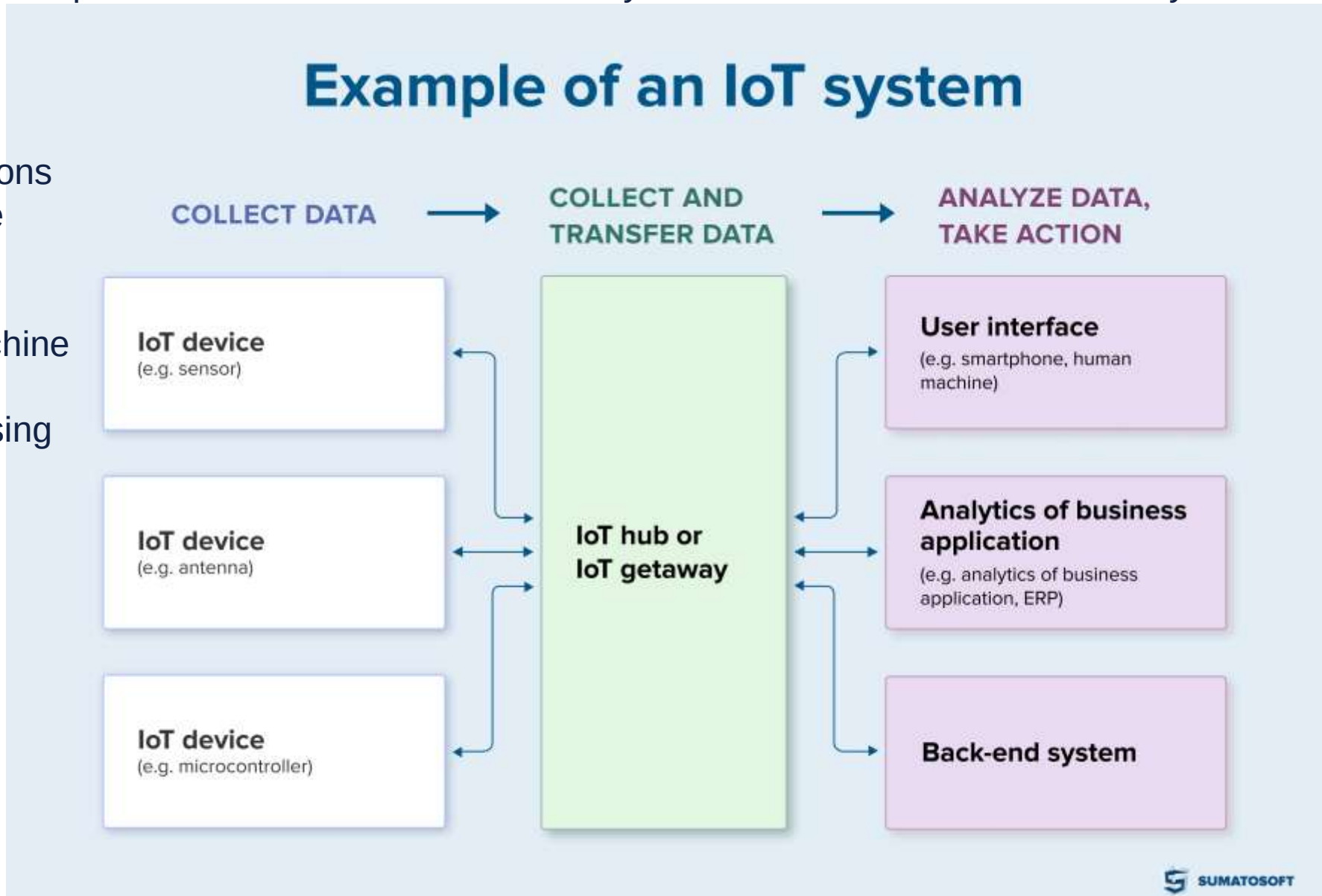
We can split above definition:

PHYSICAL OBJECTS + EMBEDDED TECHNOLOGY + COMMUNICATION

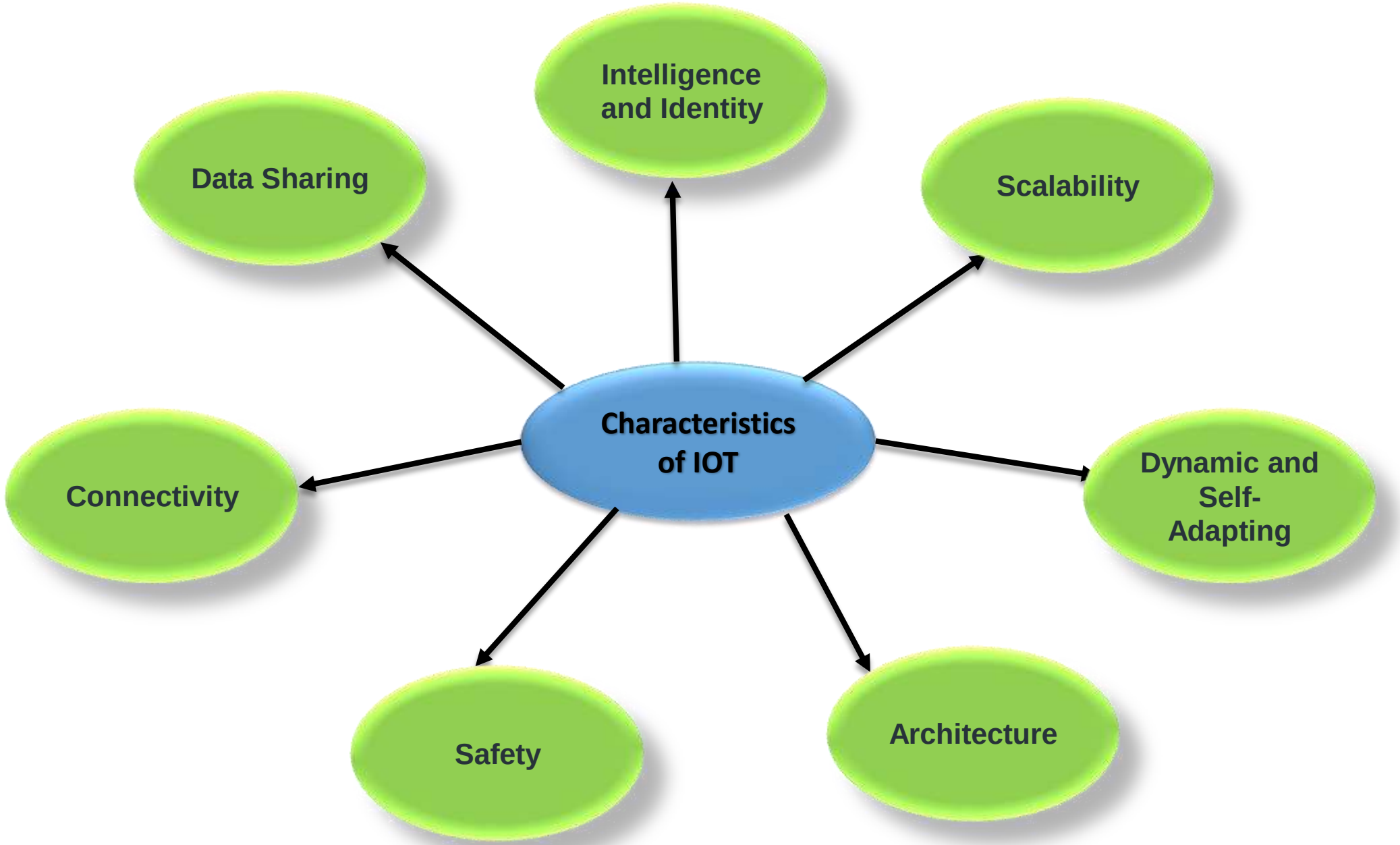
- The IoT architecture consists of smart “things” – devices that use embedded systems (processors, sensors, and hardware).
- These devices collect, send, and manage data and systems based on data acquired from the environment. IoT devices connect to IoT gateways, IoT platforms, or other devices for analysis and share the sensor data they collect.

- The IoT platform is used to find patterns, work out recommendations and detect possible issues before they occur.

- Artificial intelligence (AI) and machine learning also help IoT developers make data collection and processing easier and more dynamic.



Characteristics of the Internet of Things :



Connectivity--

- Connectivity is an important requirement of the IoT infrastructure.
- Things of IoT should be connected to the IoT infrastructure.
- Anyone, anywhere, anytime can connect , this should be guaranteed at all times.
- With everything going on IoT devices and hardware, with sensors and other electronics and connected hardware and control systems there needs to be a connection between various levels.
- For example, the connection between people through Internet devices like mobile phones, and other gadgets, also a connection between Internet devices such as routers, gateways, sensors, etc.

Data Sharing –

- Since IoT is made up of interconnected devices, they share data with each other, making each other more efficient and improving their performance.
- Due to data sharing capabilities, IoT can understand your patterns, schedule and requirements more accurately.
- This characteristic of the Internet of Things is evident in smart refrigerators, thermostats, and smart cars.
- Based on the collected data, they can even predict future events.

Intelligence and Identity –

- The extraction of knowledge from the generated data is very important.
- **For example**, a sensor generates data, but that data will only be useful if it is interpreted properly.
- Each IoT device has a unique identity. This identification is helpful in tracking the equipment and at times for querying its status.
- IoT comes with the combination of algorithms and computation, software & hardware that makes it smart.

Scalability –

- The number of elements connected to the IoT zone is increasing day by day. Hence, an IoT setup should be capable of handling the massive expansion. The data generated as an outcome is enormous, and it should be handled appropriately.(**It supports for newly incoming devices**)

- The management of data generated from devices and their interpretation for application purposes becomes more critical.

Dynamic and Self-Adapting (Complexity) –

- The primary activity of Internet of Things is to collect data from its environment, this is achieved with the dynamic changes that take place around the devices.

- The state of these devices change dynamically, example sleeping and waking up, connected and/or disconnected as well as the context of devices including temperature, location and speed.

- In addition to the state of the device, the number of devices also changes dynamically with a person, place and time.

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

Architecture –

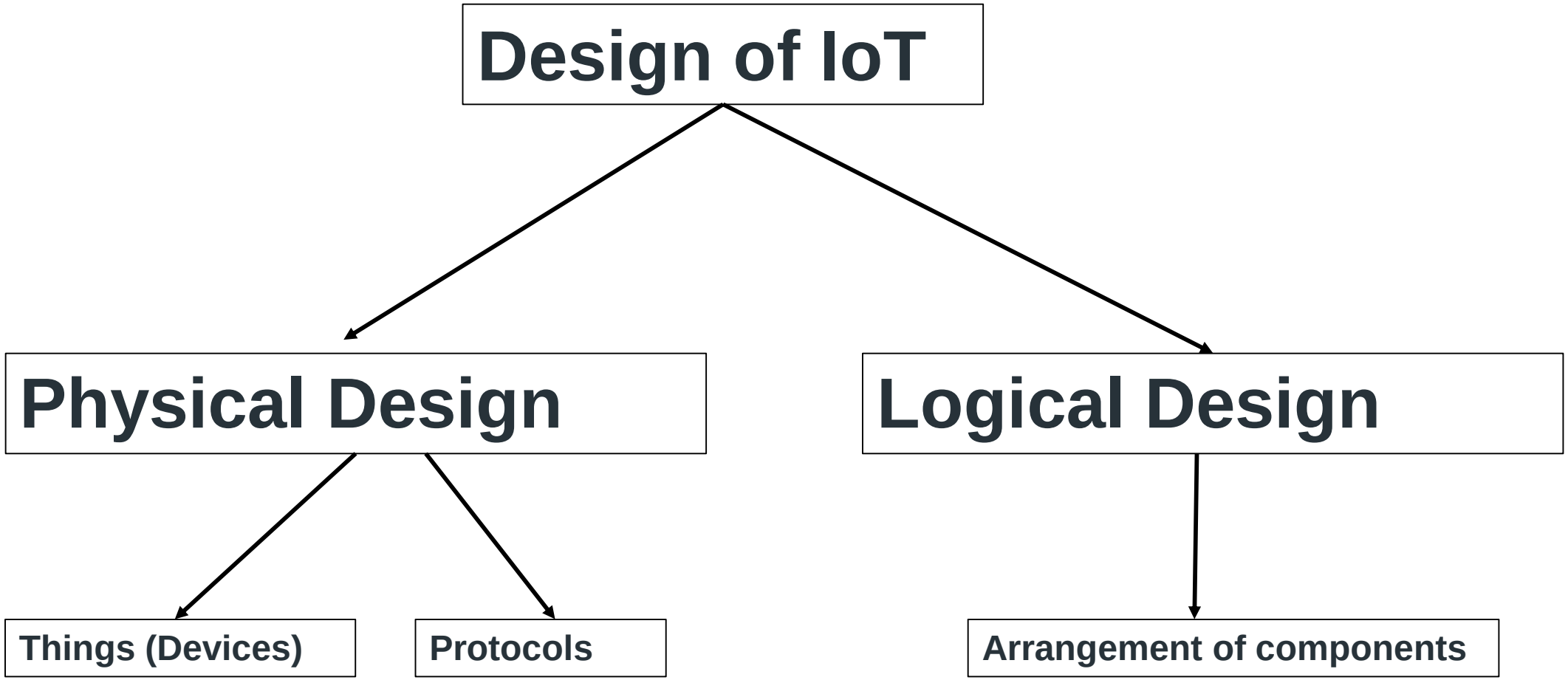
- IoT architecture cannot be **homogeneous** (uniform) in nature.

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

Safety –

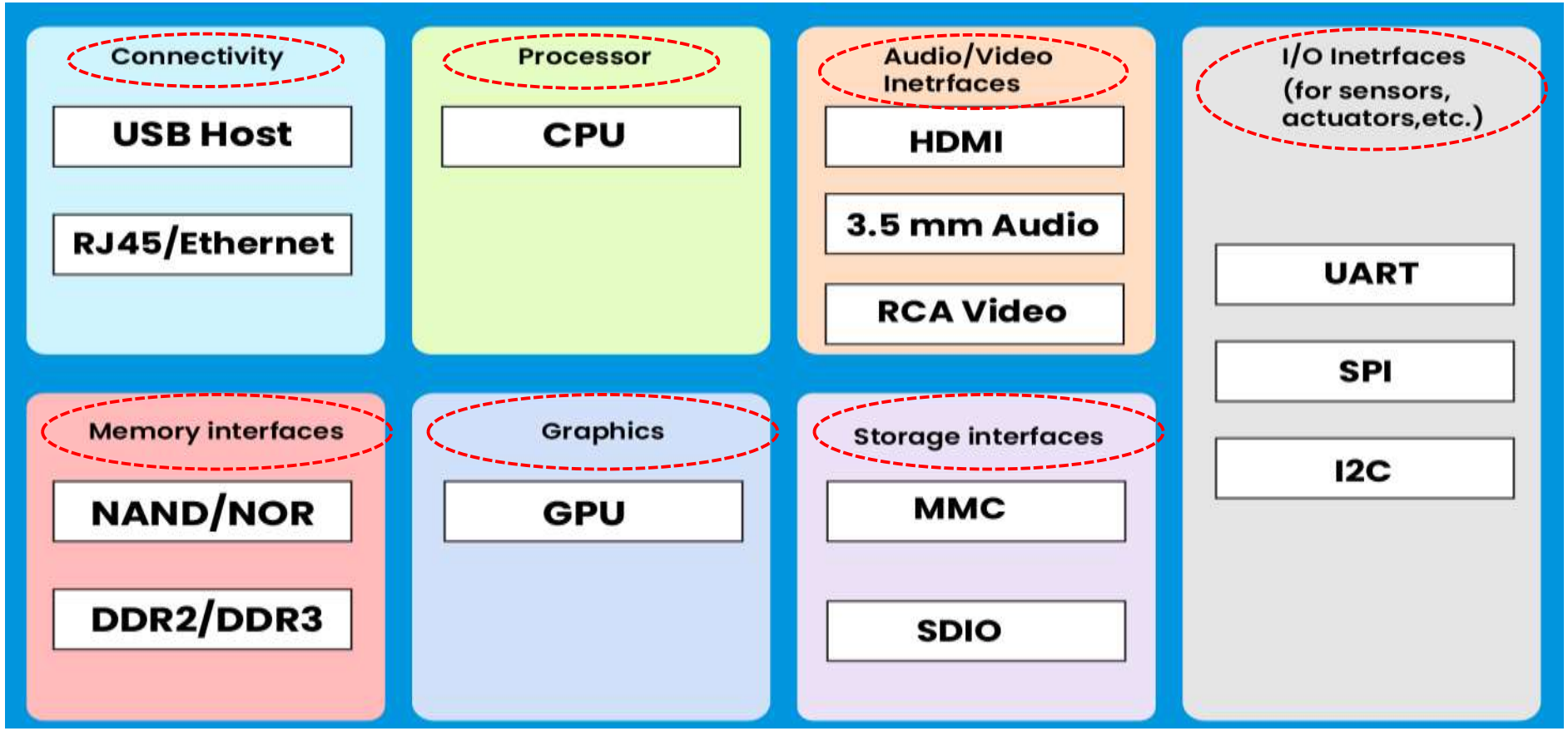
- There is a high level of transparency and privacy issues with IoT. It is important to secure the endpoints, the networks, and the data that is transferred across all of it means creating a security paradigm.

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


Design of IoT
Physical Design
Logical Design
Things (Devices)
Protocols
Arrangement of components

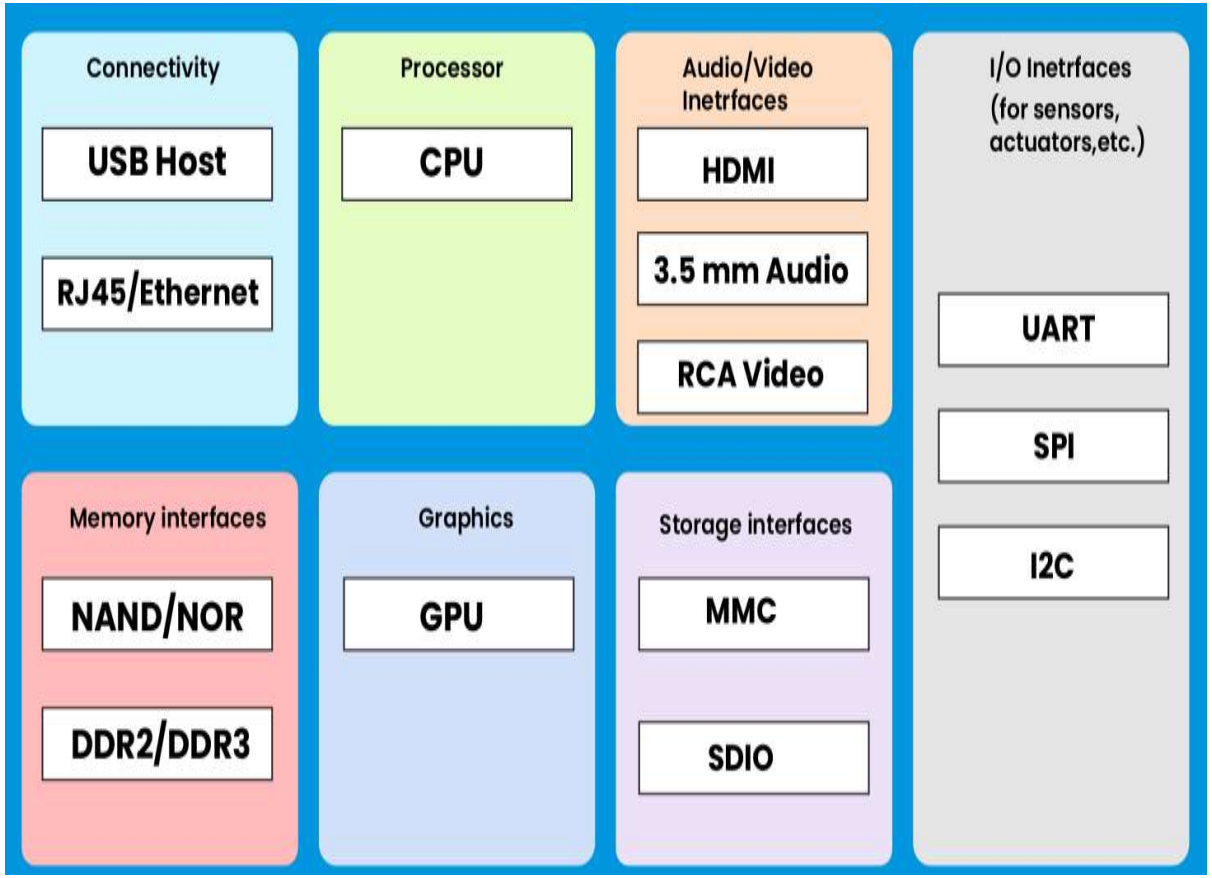
Physical Design of IoT

- The Physical design of IoT deals with the individual devices connected to the IoT network and the protocols used to create a functional IoT environment.
- Each IoT device can perform tasks of remote sensing, actuating, monitoring, etc due to the IoT network they are connected to.
- These can also transmit information through different types of wireless or wired connections.
- They can generate data, which is used to perform analysis and perform operations for improving the system.



Physical design of IOT / General block diagram of IOT Device

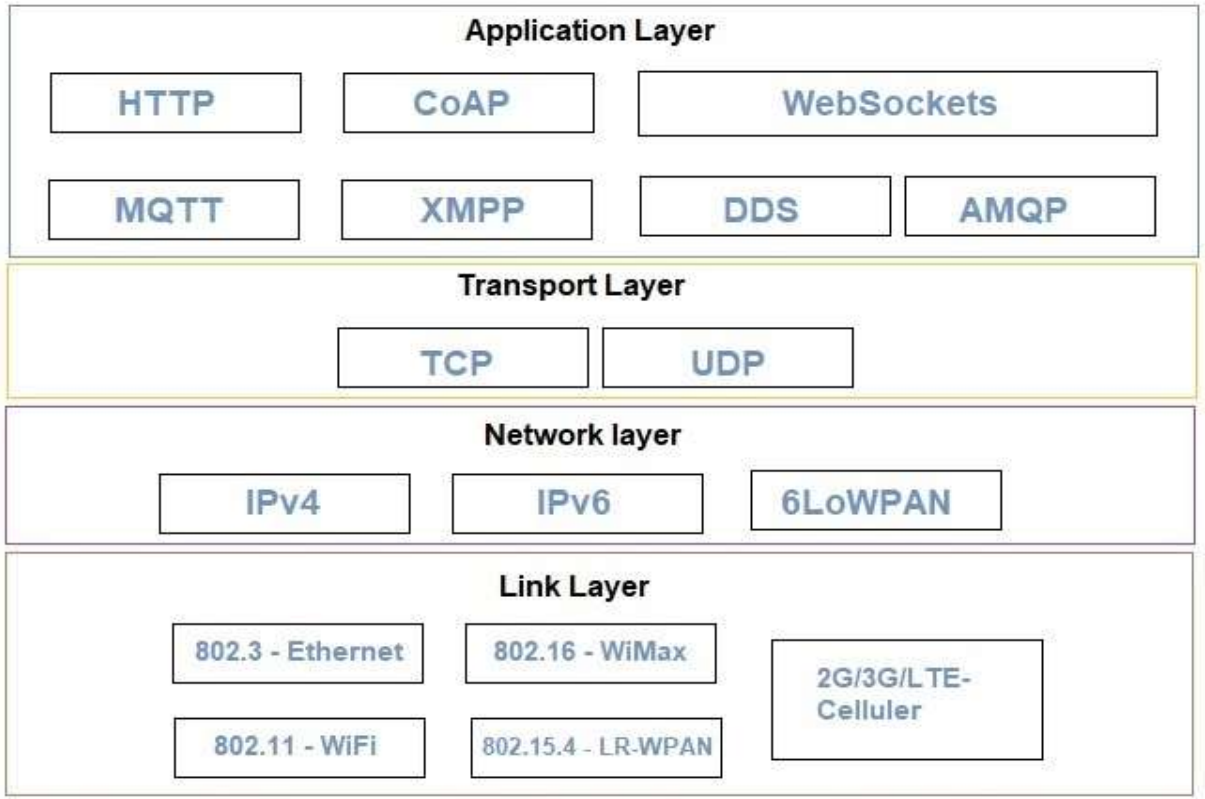
- **Connectivity:** Devices like USB hosts and ETHERNET are used for connectivity between the devices and the server.
- **Processor:** A processor like a CPU and other units are used to process the data. these data are further used to improve the decision quality of an IoT system.
- **Audio/Video Interfaces:** An interface like HDMI and RCA devices is used to record audio and videos in a system.
- **Input/Output interface:** To give input and output signals to sensors, and actuators we use things like UART, SPI, CAN, etc.
- **Storage Interfaces:** Things like SD, MMC, and SDIO are used to store the data generated from an IoT device.
- **Controlling of activity:** Devices like **DDR** and **GPU** control the activity of an IoT system.



Physical design of IOT

IoT Protocols

- IoT protocols establish communication between a node device and a server over the internet by sending commands to an IoT device and receiving data from an IoT device.
- Both the server and client-side use different types of protocols.
- By network layers, they are managed. Some of the network layers are the application, transport, network, and link layers.
- It works as a building block for logical and physical design of IoT.

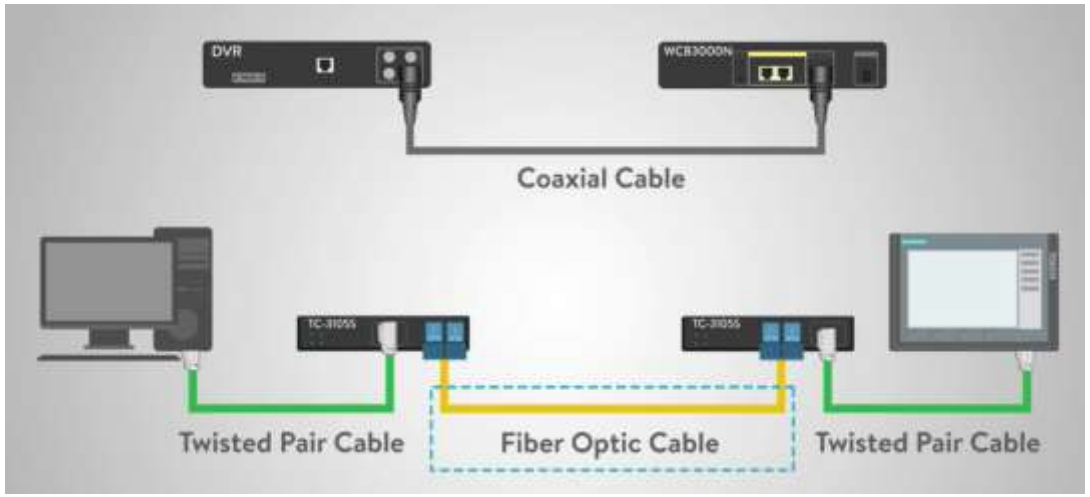


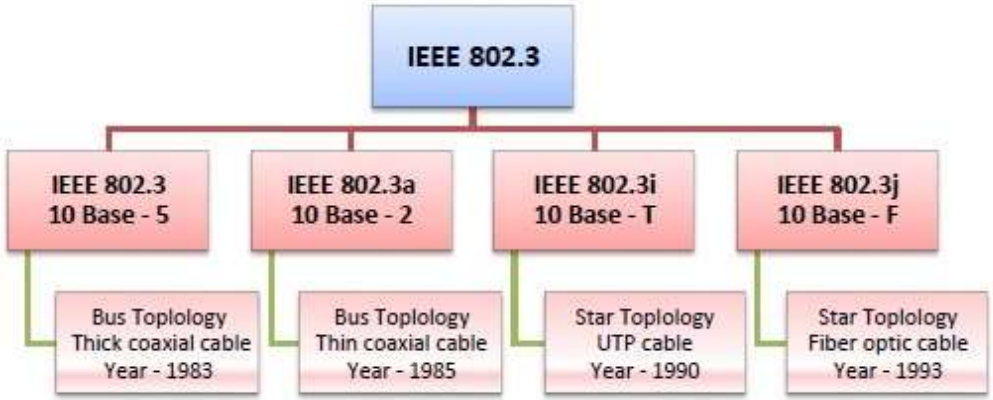
Link Layer:

Link layer protocols determine how data is physically sent over the network's physical layer or medium (Coxial cable or other or radio wave).

This Layer determines how the packets are coded and signaled by the hardware device over the medium to which the host is attached (eg. coxial cable).

1. **802.3 – Ethernet :** IEEE 802.3 is a set of standards and protocols that define **Ethernet-based networks**. Ethernet technologies are primarily used in **LANs**, though they can also be used in **MANs** and even **WANs**. IEEE 802.3 defines the **physical layer** and the **medium access control (MAC) sub-layer** of the **data link layer** for wired Ethernet networks.





2. **802.16 – Wi-Max :** The 802.16 is a set of standards defined by IEEE (Institute of Electrical and Electronics Engineers) that lays down the specifications for **wireless broadband technology**. It has been commercialized as Worldwide Interoperability for Microwave Access (WiMAX) that is responsible for delivery of last mile **wireless broadband access.**

IEEE 802.15.4

IEEE 802.15.4 is a wireless communication standard that was specifically designed for low-rate, short-range wireless communication among devices. It is part of the IEEE 802 family of standards, which encompasses various wireless and wired networking standards. IEEE 802.15.4 focuses on providing a cost-effective and energy-efficient solution for connecting devices in applications where low data rates and low power consumption are essential. Here is a more detailed definition:

IEEE 802.15.4:

Wireless Communication Standard: IEEE 802.15.4 is an industry-standard protocol for wireless communication.

Low-Rate: It is optimized for applications with low data rate requirements, typically in the range of a few kilobits per second (Kbps) to hundreds of Kbps. This makes it suitable for transmitting small amounts of data efficiently.

Short-Range: IEEE 802.15.4 is designed for relatively short-range communication, typically within a range of 10 to 100 meters. This short range is ideal for applications where devices are in close proximity to each other.

Low Power Consumption: One of its standout features is its ability to operate on minimal power. Devices using IEEE 802.15.4 can remain in low-power sleep modes and wake up only when necessary to conserve energy. This makes it well-suited for battery-powered and energy-harvesting devices.

Network Layer

Responsible for sending of IP datagrams from the source network to the destination network. Network layer performs the host addressing and packet routing. We used IPv4 and IPv6 for Host identification. IPv4 and IPv6 are hierarchical IP addressing schemes.

1. IPv4 :

An **Internet Protocol address** (**IP address**) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two main functions: host or network interface identification and location addressing. Internet Protocol version 4 (IPv4) defines an IP address as a **32-bit number**. However, because of the growth of the Internet and the depletion of available IPv4 addresses, a new version of IP (IPv6), using 128 bits for the IP address, was standardized in 1998. IPv6 deployment has been ongoing since the mid-2000s.

2. IPv6 :

Internet Protocol version 6 (**IPv6**) is successor of IPv4. IPv6 was developed by the Internet Engineering Task Force (IETF) to deal with the long-anticipated problem of IPv4 address exhaustion. In December 1998, IPv6 became a Draft Standard for the IETF, who subsequently ratified it as an Internet Standard on 14 July 2017. IPv6 uses a **128-bit** address, theoretically allowing 2^{128}, or approximately

3. 6LoWPAN : It allows **tiny, low-power devices (like sensors)** to communicate using **IPv6 over IEEE 802.15.4 networks**.

It is an acronym of *IPv6 over **Low-Power Wireless Personal Area Networks***. 6LoWPAN is the name of a concluded working group in the Internet area of the IETF. This protocol allows for the **smallest devices** with **limited processing ability** to transmit information wirelessly using an internet protocol. 6LoWPAN can communicate with 802.15.4 devices as well as other types of devices on an IP network link like WiFi.

Real-World Examples of 6LoWPAN

1. Smart Street Lighting System

- Each street light has a low-power sensor node.
- Nodes form a 6LoWPAN network.
- They send data (light status, power usage, fault info) to a central server using IPv6.

Benefit:

Remote monitoring, energy saving, automatic fault detection.

2. Smart Agriculture (Precision Farming)

- Soil moisture, temperature, and humidity sensors are placed in the field.
- Sensors use 6LoWPAN to send data to a gateway.
- The gateway forwards data to the cloud.

Benefit:

Efficient irrigation, reduced water usage, better crop yield.

3. Smart Home Automation

- Devices like smart bulbs, thermostats, and door sensors use 6LoWPAN.
- They communicate with each other and with a home gateway using IPv6.

Benefit:

Low power consumption and seamless internet connectivity.

4. Smart Health Monitoring

- Wearable sensors monitor heart rate, temperature, or blood pressure.
- Data is transmitted via 6LoWPAN to a nearby gateway and then to a hospital system.

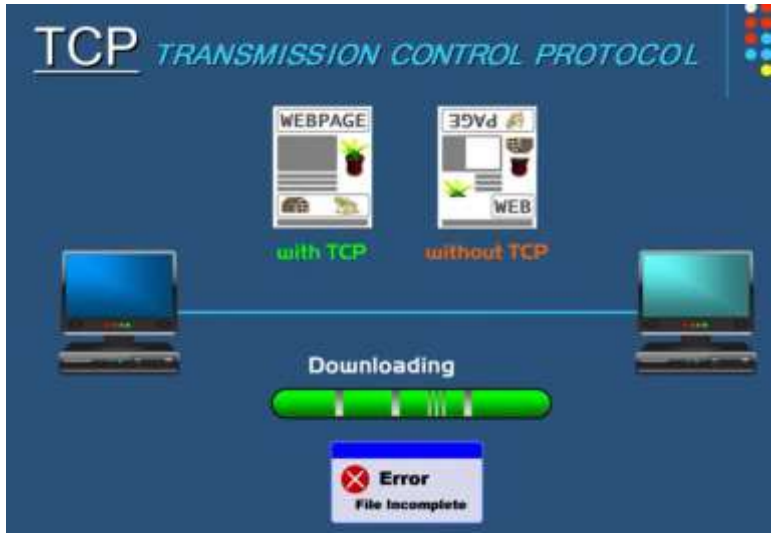
Benefit:

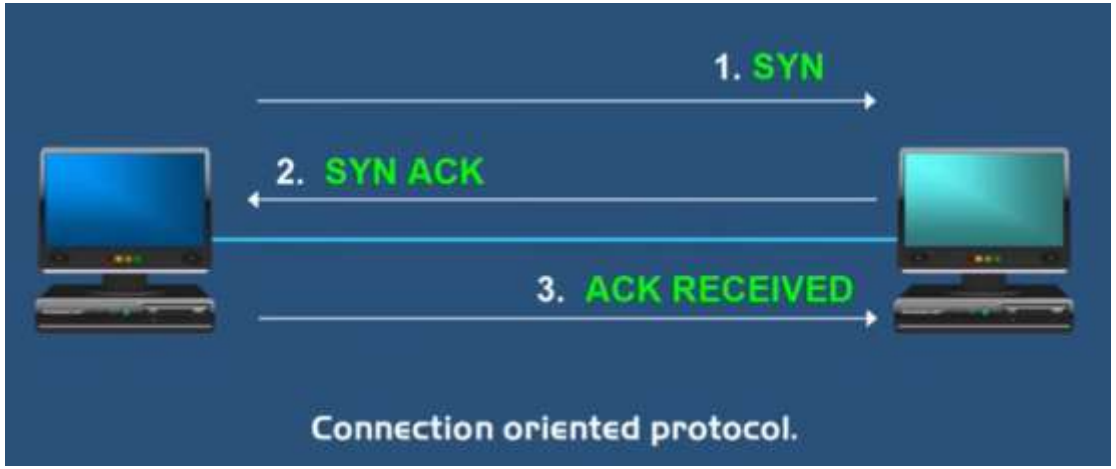
Real-time remote patient monitoring with very low battery usage.

Transport Layer:

This layer provides functions such as error control, segmentation, flow control and congestion control. So this layer protocols provide end-to-end message transfer capability independent of the underlying network.

TCP : TCP (Transmission Control Protocol) is a standard that defines how to establish and maintain a network conversation through which application programs can exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other. Together, TCP and IP are the basic rules defining the Internet.





UDP : User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of Internet Protocol suite, referred as UDP/IP suite. Unlike TCP, it is unreliable and connectionless protocol. So, there is no need to establish connection prior to data transfer.

Application Layer

Application layer protocols define how the applications interface with the lower layer protocols to send over the network.

HTTP : Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes. HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a response. HTTP is a stateless protocol, meaning that the server does not keep any data (state) between two requests.

CoAP : Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained and constrained networks in the Internet of Things.

CoAP : Constrained Application Protocol (CoAP)

- The Constrained Application Protocol (CoAP) is a special web transfer protocol that operates with **constrained nodes** and **networks**.Constrained nodes are (microcontrollers, smart devices, sensors and actuators, and other small computers) Constrained networks are commonly used for applications such as the Internet of Things (IoT), where devices may have limited processor, memory, and power resources.
- CoAP is designed to enable simple, constrained devices to join the IoT even through constrained networks with low bandwidth and low availability.
- CoAP or Constrained Application Protocol, as the name suggests, is an application layer protocol that was introduced by the Internet Engineering Task Force in the year 2014.
- It is generally used for machine-to-machine (M2M) applications such as smart energy (powerful, sustainable renewable energy sources that promote greater eco-friendliness while driving down costs.)and building automation.
- It is a web-based protocol that resembles HTTP
- It is also based on the request-response model. Based on the REST(Representational State transfer)architecture, this protocol considers the various objects in the network as resources.
- These resources are uniquely assigned a URI or Uniform Resource Identifier. The data from one resource to another resource is transferred in the form of CoAP message packets.

The main features of CoAP protocol are:

- Web protocol used in IoT/M2M with constrained requirements

•Asynchronous message exchange

•Low overhead and very simple to parse.

•Proxy and caching capabilities

1. Web protocol used in M2M with constrained requirements

Meaning:

CoAP works like HTTP (GET, POST, PUT, DELETE) but is designed for tiny devices with limited power, memory, and bandwidth.

Real-world example:

In a **smart agriculture field**, small battery-powered soil sensors use CoAP to send moisture data to a gateway instead of HTTP because HTTP is too heavy for them.

2. Asynchronous message exchange

Meaning:

Devices do not need to stay connected continuously. They can send data and go back to sleep.

Real-world example:

A **temperature sensor in a cold storage warehouse** sends data every 10 minutes and then sleeps to save battery. It does not keep a constant connection like a web browser does.

3. Low overhead and very simple to parse

Meaning:

CoAP messages are very small, so they use less bandwidth and processing power.

Real-world example:

A **wearable health sensor** sends heart-rate data over CoAP so that battery life lasts longer and the microcontroller is not overloaded.

Proxy and caching capabilities

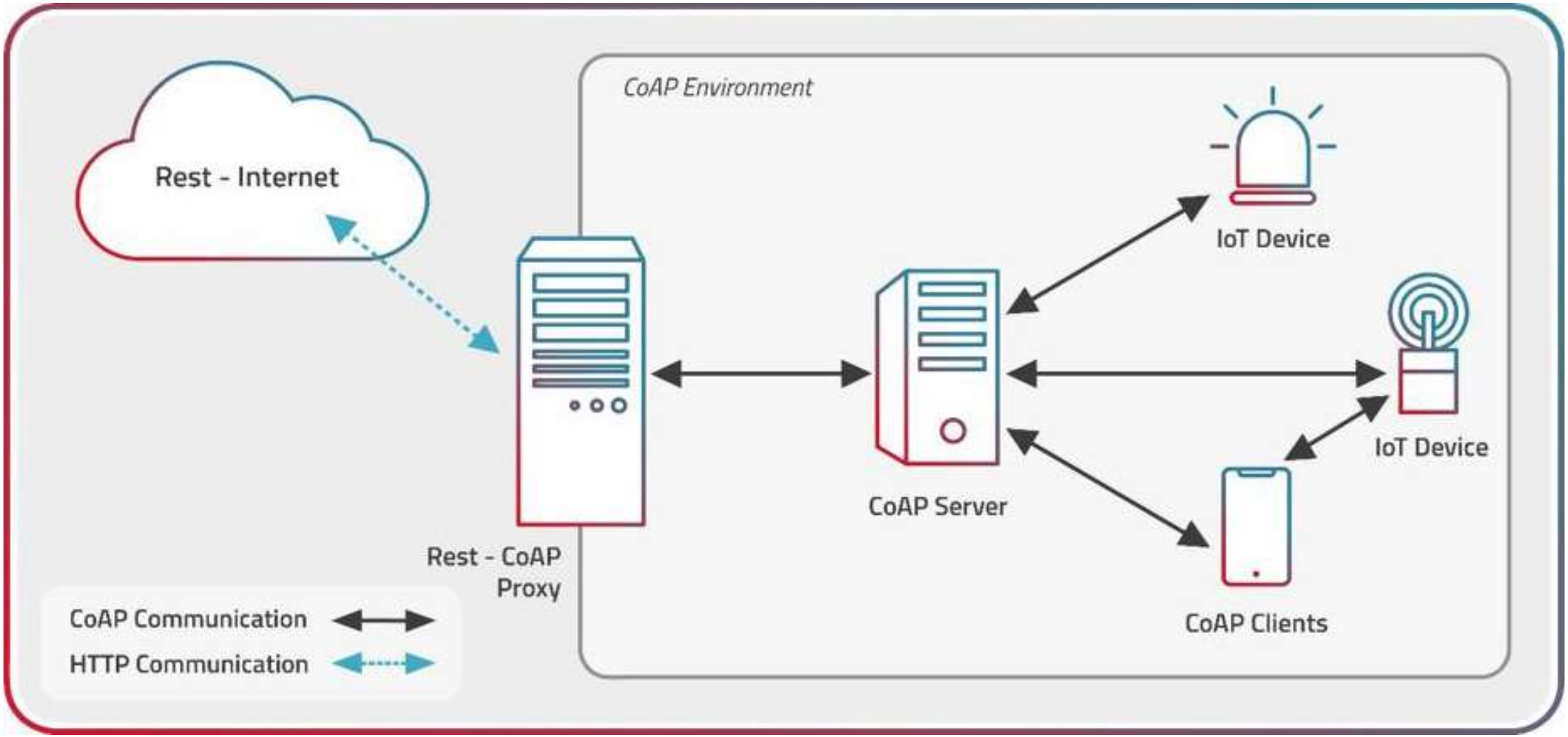
Meaning:

Gateways can act as proxies between CoAP devices and HTTP servers, and they can cache responses to reduce traffic.

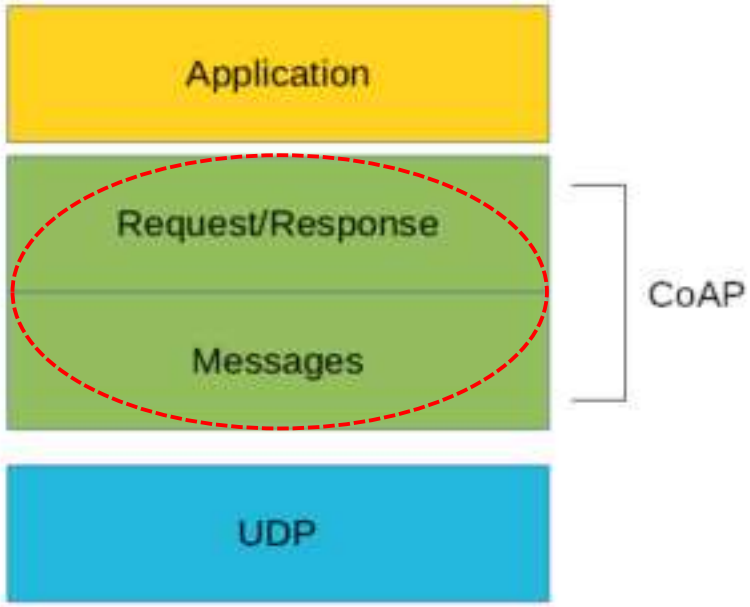
Real-world example:

In a **smart city pollution monitoring system**, roadside sensors send data to a local gateway (CoAP). The gateway converts it to HTTP and forwards it to a cloud server. It may cache values so the cloud doesn't query the sensor every time.

How Does CoAP Work?



layers that make CoAp protocol:



1. **Messages :** The Messages layer deals with UDP and with asynchronous messages.

2. **Request/Response :** The Request/Response layer manages request/response interaction based on request/response messages.

1. Messages :

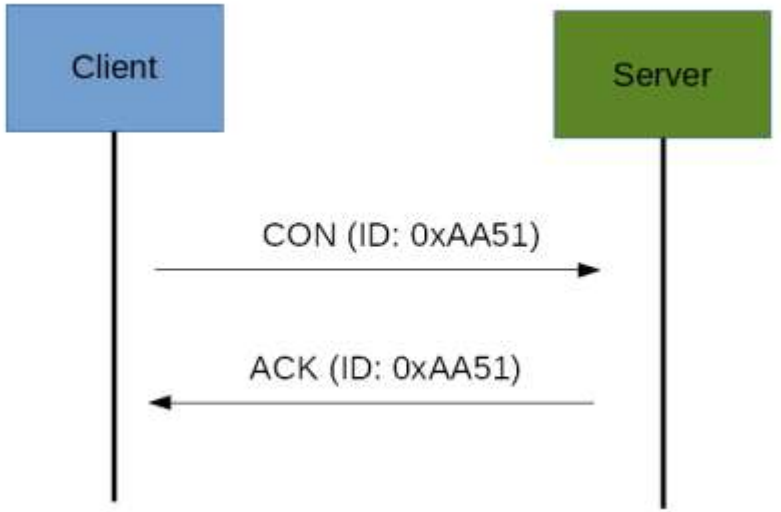
CoAP supports four different message types:

- Confirmable
- Non-confirmable
- Acknowledgment
- Reset

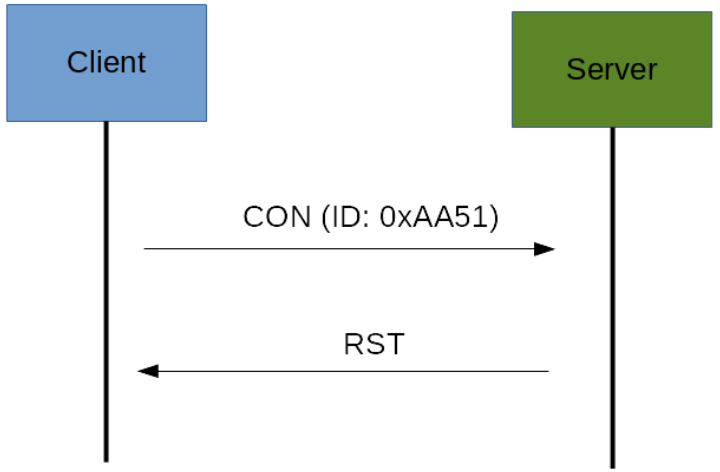
CON (Confirmable) Message:

CON messages are used for reliable communication, ensuring that the recipient sends an acknowledgment. They contain a CoAP request or response and are sent by a client or server, respectively. The sender expects an acknowledgment (ACK) from the recipient and retransmits the message until the ACK is received.

- 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)



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

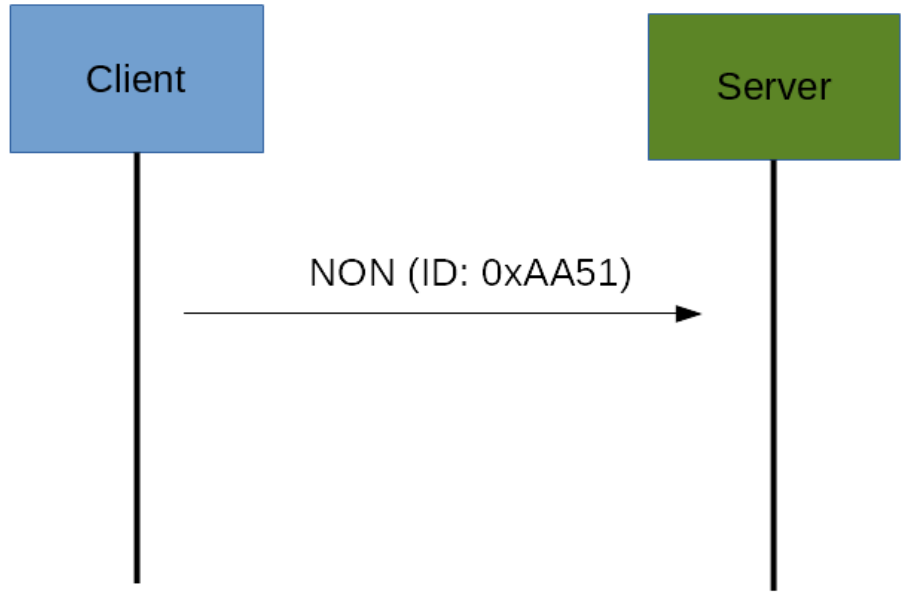


NON (Non-Confirmable) Message:

NON messages are used for faster communication without requiring acknowledgment.
They are similar to CON messages but don't demand an ACK.
NON messages are suitable for scenarios where real-time communication is prioritized over reliability.

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.

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



ACK (Acknowledgment) Message:

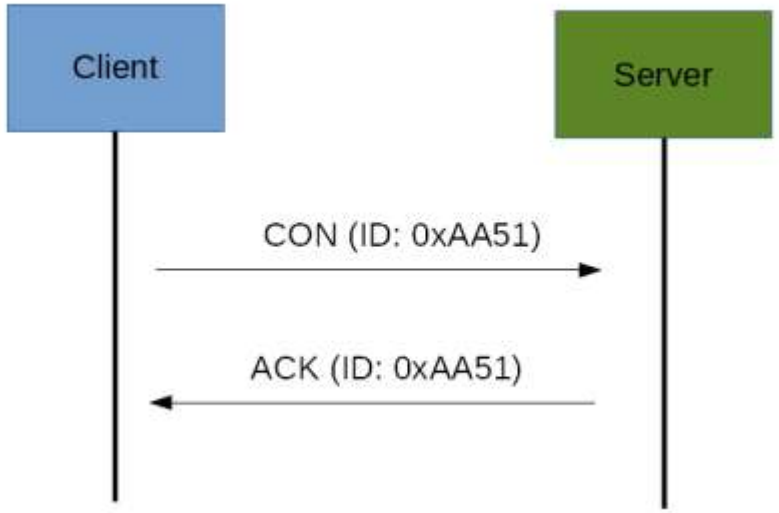
ACK messages are sent in response to CON messages to acknowledge their receipt.

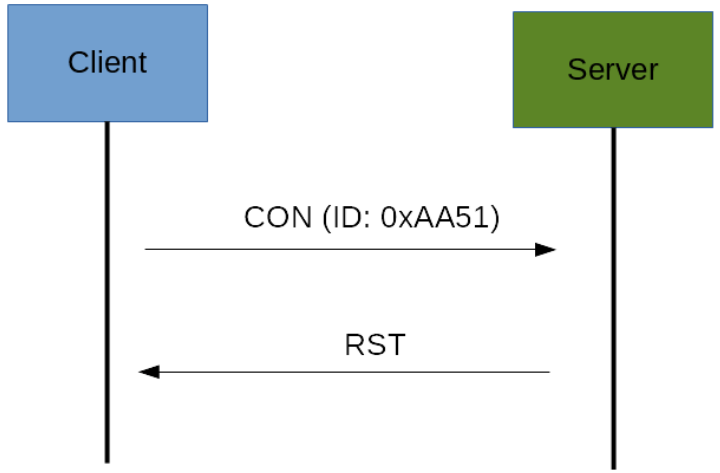
They indicate that the recipient has successfully received the message and is processing it.

RST (Reset) Message:

RST messages are sent to cancel a pending CON message that hasn't yet been acknowledged.

They are typically used when a receiver cannot or chooses not to process a pending message.

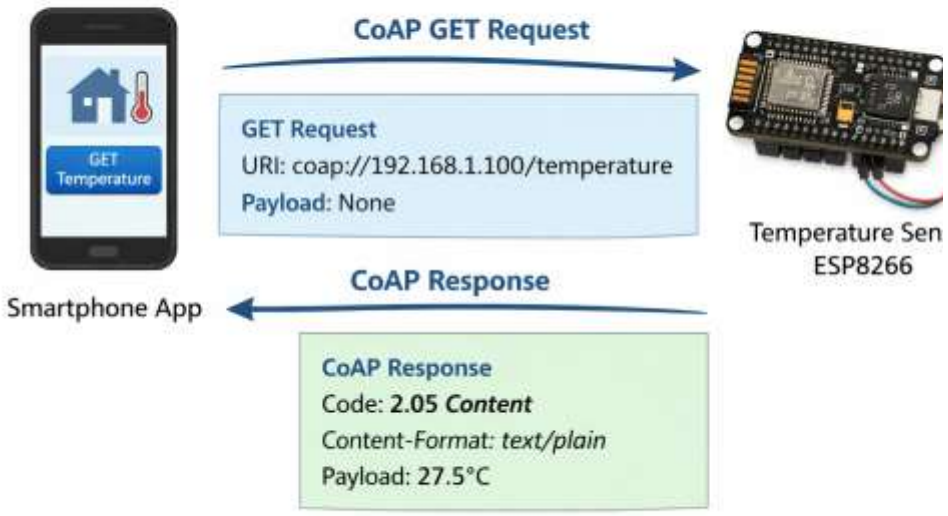


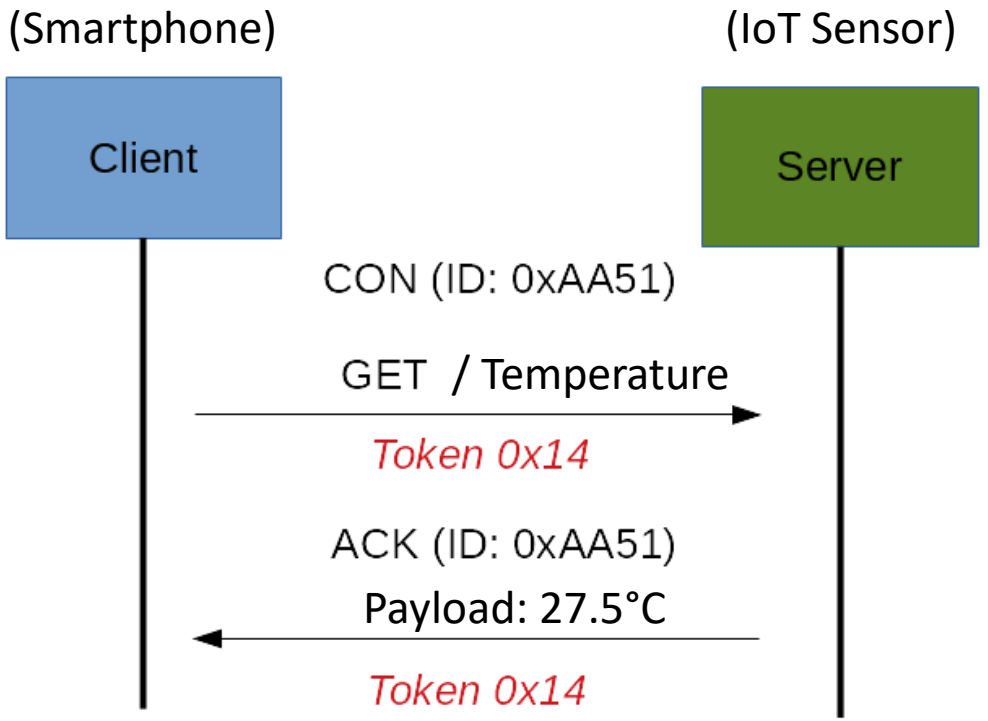


The main operations are **GET, POST, PUT, DELETE**.

2. Request/Response :

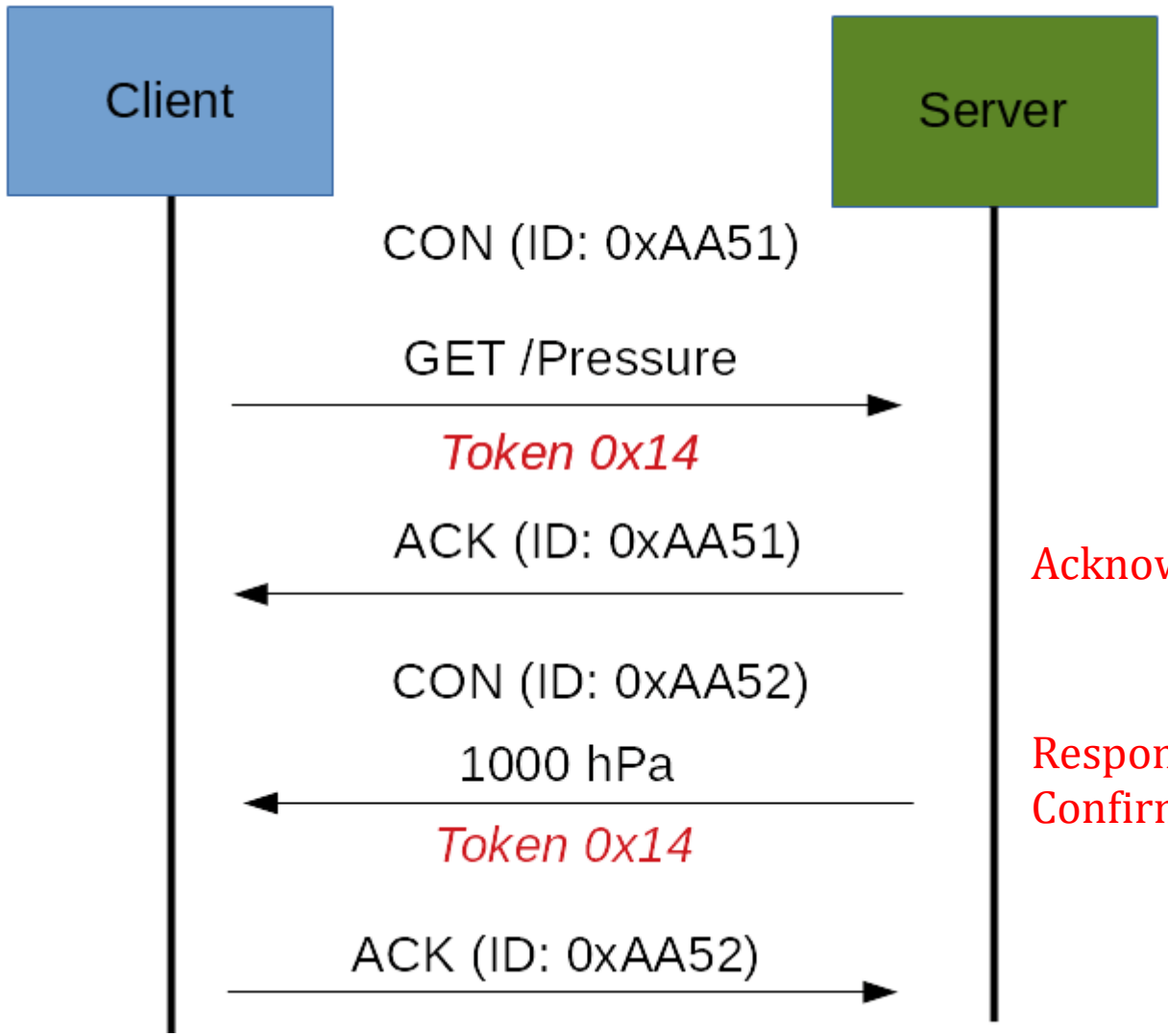
The CoAP Request/Response is the second layer in the CoAP abstraction layer.
The request is sent using a Confirmable (CON) or Non-Confirmable (NON) message.
There are several scenarios depending on if the server can answer immediately to the client request or the answer if not available.
If the server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





As you can notice in the CoAP message, there is a **Token**. The Token is different from the Message-ID and it is used to match the **request and the response.**

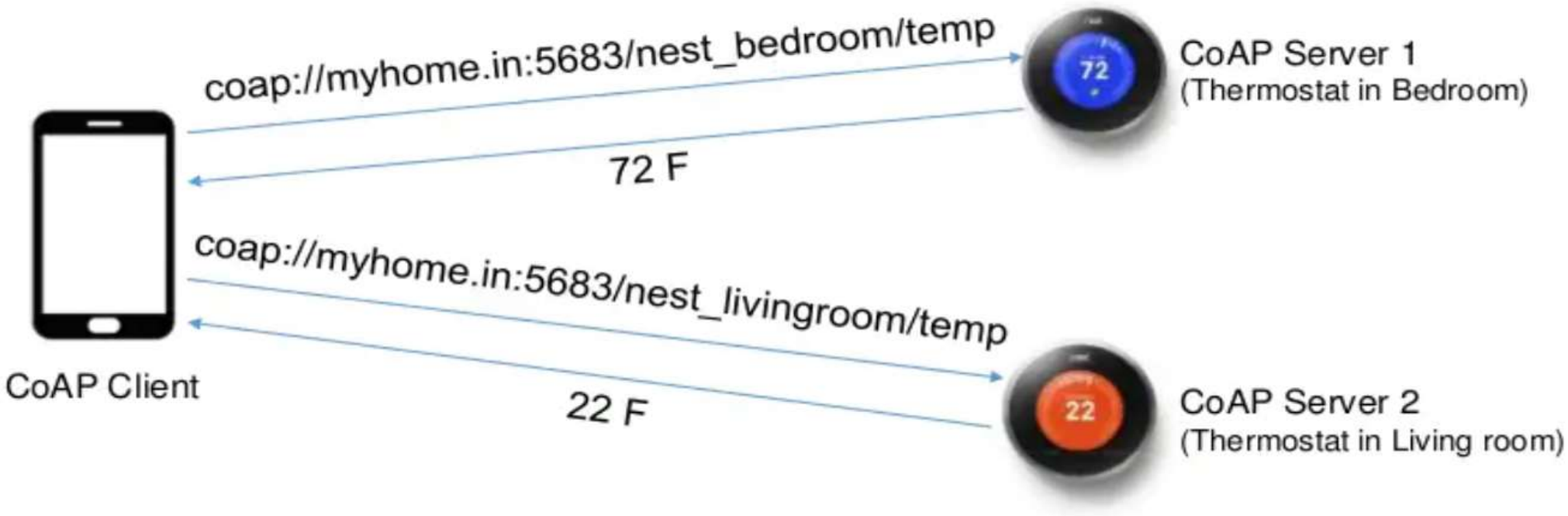
If the server can't answer to the request coming from the client immediately, then it sends an Acknowledge message with an empty response. As soon as the response is available, then the server sends a new Confirmable message to the client containing the response. At this point, the client sends back an Acknowledge message:

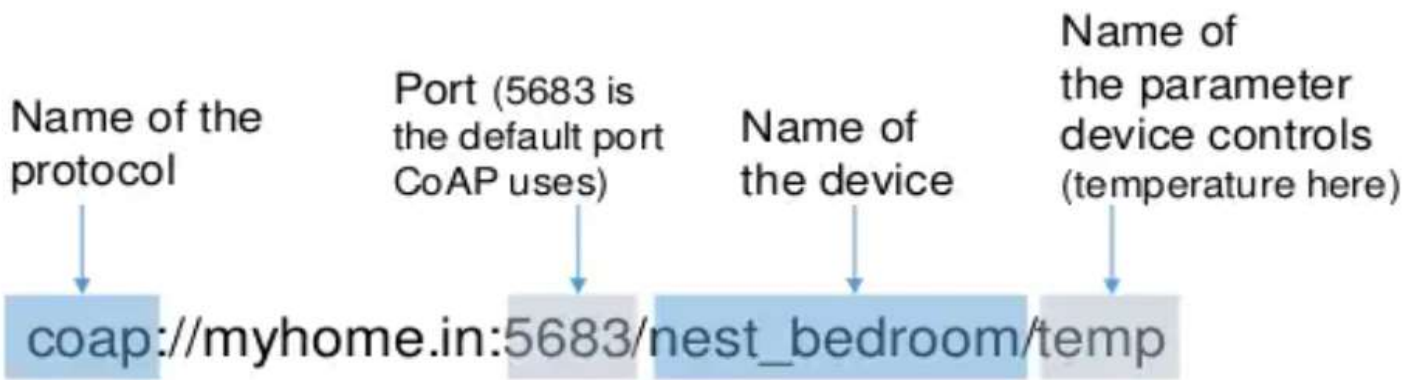


Acknowledge message with an empty response

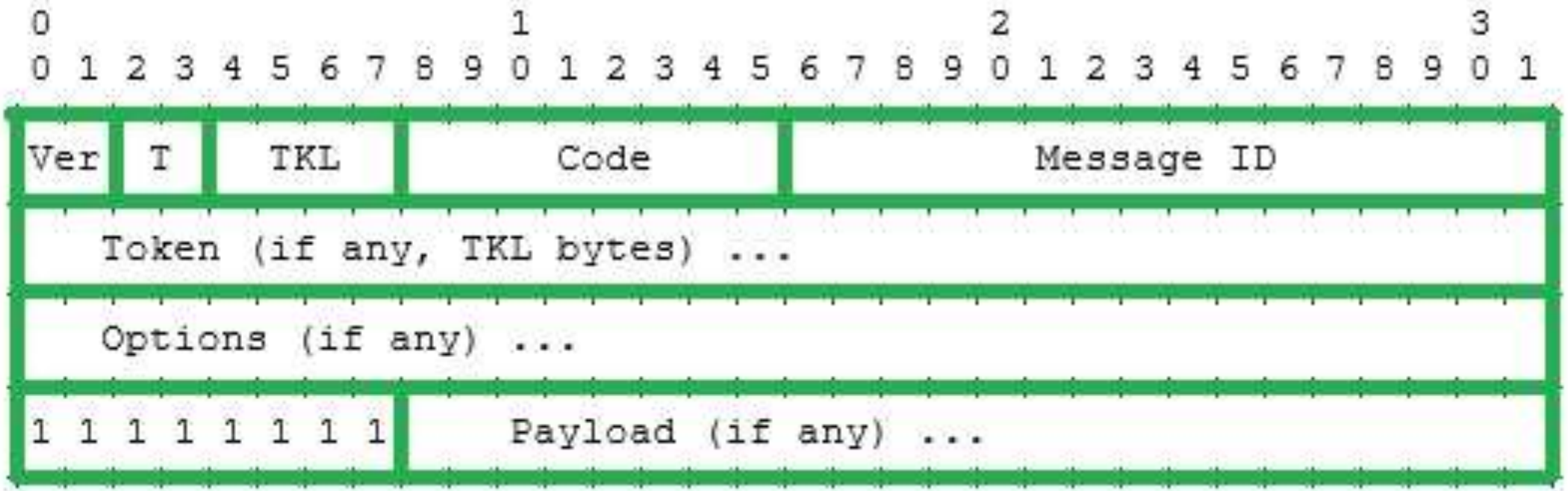
Response is available, then the server sends a new Confirmable message with response.

CoAP – Request Response





CoAP Message Format | CoAP Header



CoAP Message Format

CoAP message header	Description
Ver	It is 2 bit unsigned integer. It mentions CoAP version number. Set to one.
T	It is 2 bit unsigned integer. Indicates message type viz. confirmable (0), non-confirmable (1), ACK (2) or RESET(3).
TKL	It is 4 bit unsigned integer, Indicates length of token (0 to 8 bytes).
Code	It is 8 bit unsigned integer, It is split into two parts viz. 3 bit class (MSBs) and 5 bit detail (LSBs).
Message ID	16 bit unsigned integer. Used for matching responses. Used to detect message duplication.

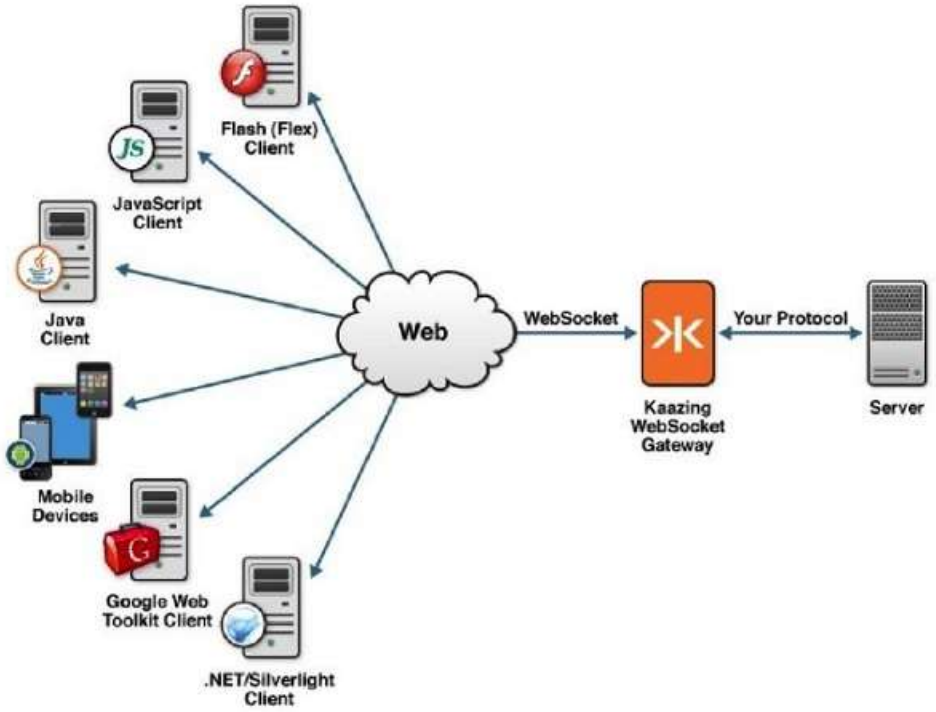
WebSocket :

WebSocket is bidirectional, a full-duplex protocol that is used in the same scenario of client-server communication, unlike HTTP it starts from **ws://** or **wss://**.
It is a stateful protocol, which means the connection between client and server will keep alive until it is terminated by either party (client or server).
After closing the connection by either of the client and server, the connection is terminated from both ends.

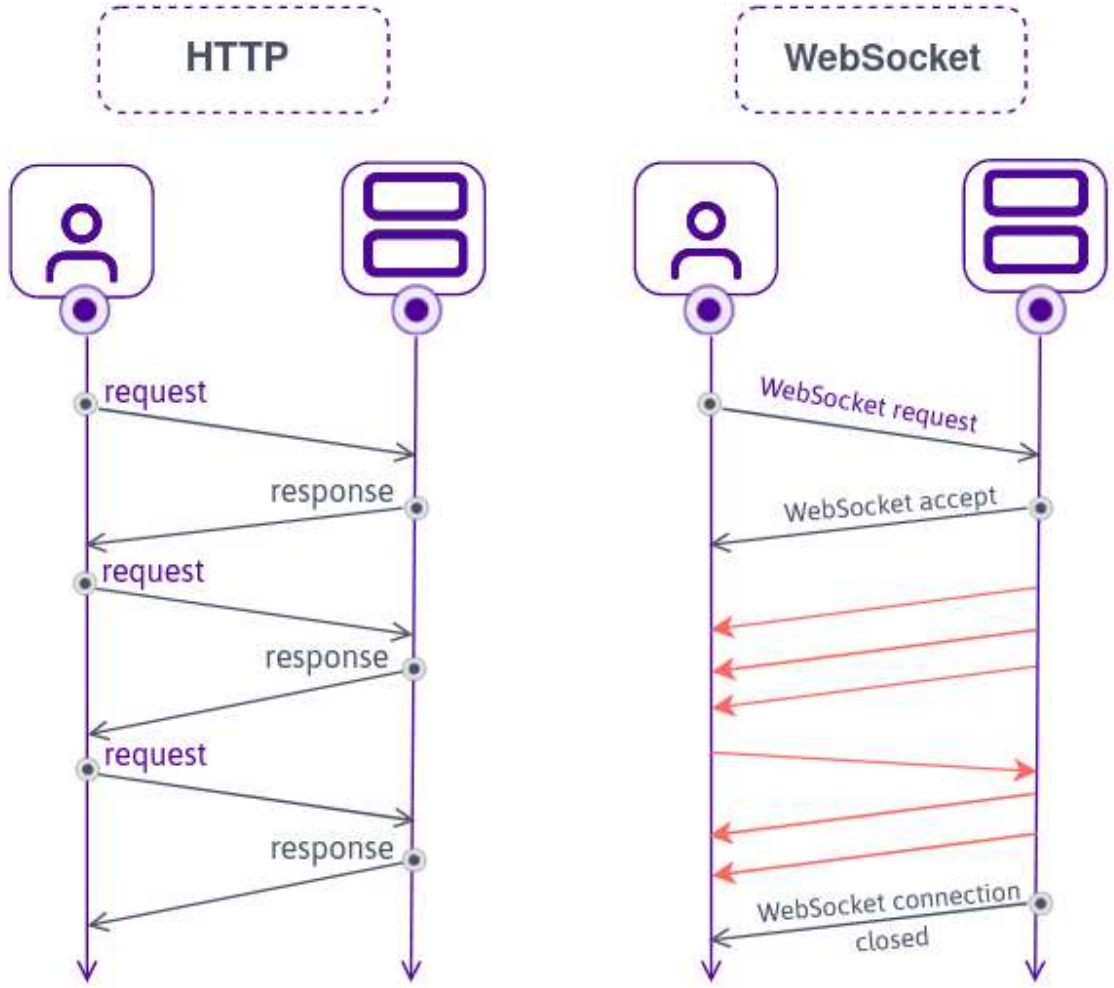
WebSocket communication presents a suitable protocol for the IoT environment where bundles of data are transmitted continuously within multiple devices. A WebSocket makes server and device communication easy. A server needs a WebSocket library to be installed and we need to have the WebSocket client and web browser installed on the client or device that supports WebSocket.

The IoT system requires real time monitoring and this is one of the problems that exists today Transferring data from the sensor via the internet network to a monitoring device must be less than 300 ms.

Using WebSocket, we won't have to pool communication data like a conventional API call does. Rather, we can have a real-time push communication setup between the server and the device. We can therefore send and receive high amounts of data in micro seconds.



- Let’s take an example of client-server communication, there is the client which is a web browser and a server, whenever we initiate the connection between client and server, the client-server made the **handshaking** and decide to create a new connection and this connection will keep alive until terminated by any of them.
- When the connection is established and alive the communication takes place using the same connection channel until it is terminated.

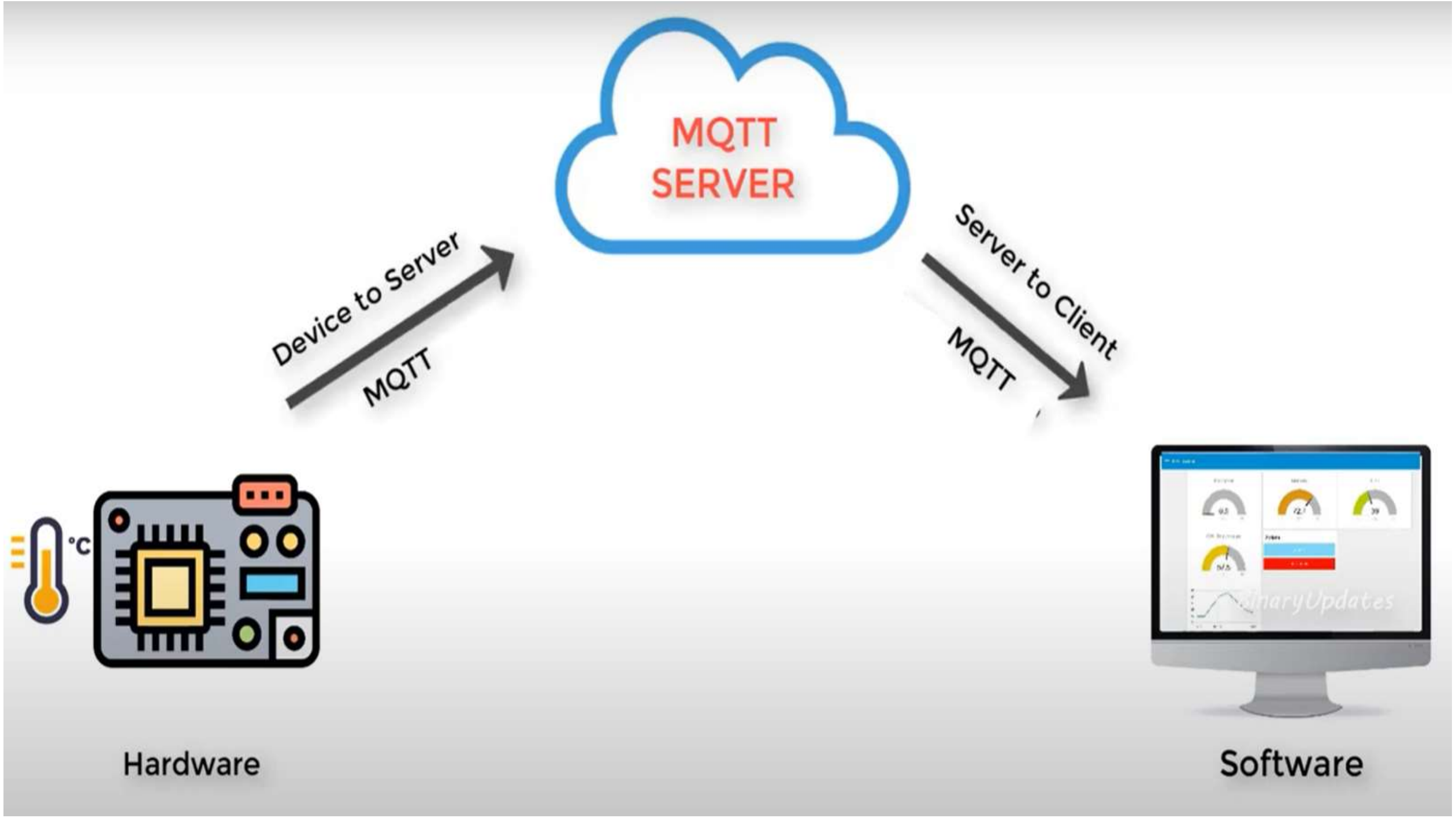


MQTT

Message Queue Telemetry Transport Protocol

Message queuing telemetry support (MQTT) is defined as a low bandwidth consumption that helps IoT devices communicate with each other, with minimal code requirements and network footprint.

- It was created by Dr. Andy Stanford-Clark and Arlen Nipper in 1999.
- The initial purpose of the communication program was to enable oil and gas sector monitoring equipment to transmit data to a distant server.
- These surveillance devices were regularly used in remote places where establishing a landline, wired link, or radio transmission connection would be difficult, if not impossible.
- MQTT is a smart solution for wireless connections facing various latency levels owing to periodic broadband limits or unreliable connections.
- It is appropriate for connecting devices with a tiny code footprint.
- The standard is used in multiple industries, including automobiles, power, and communications.


MQTT
SERVER
Device to Server
MQTT
Server to Client
MQTT
°C
Hardware
Software

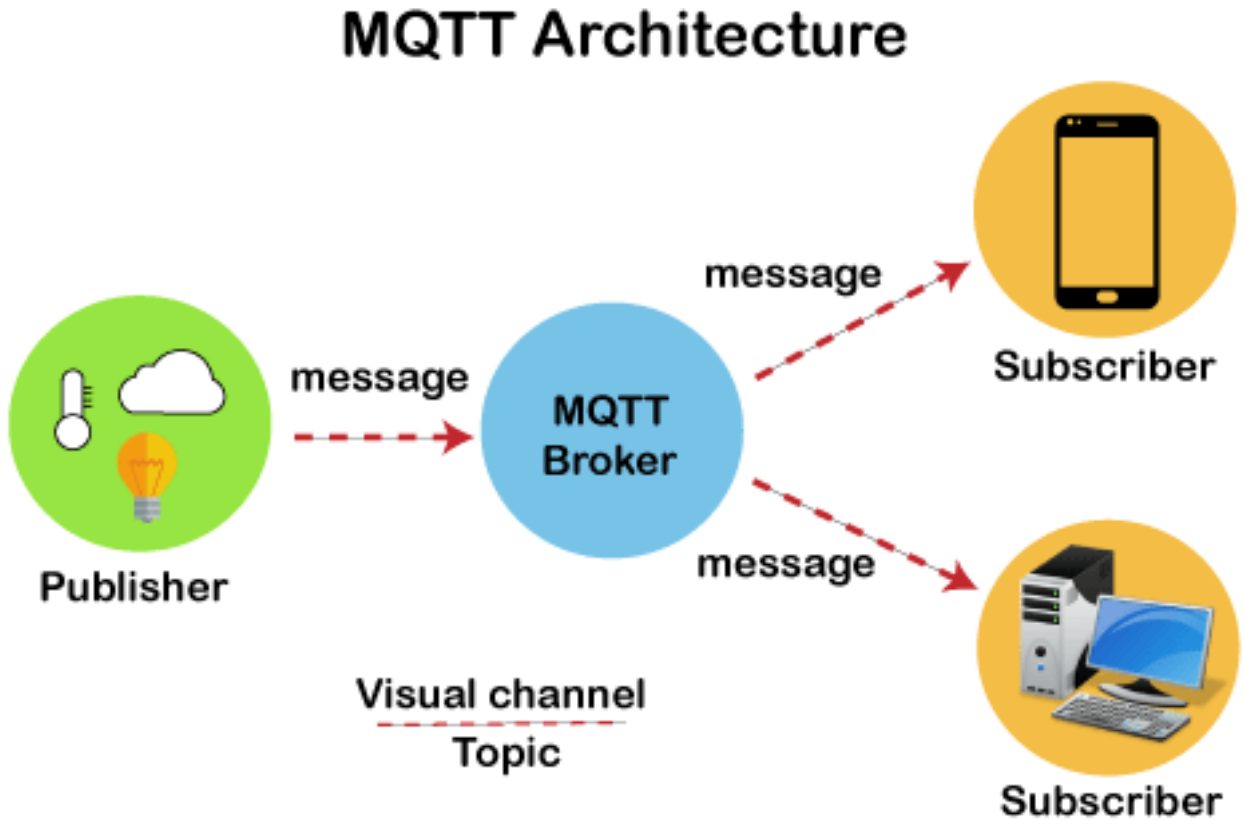
Architecture of MQTT

MQTT architecture is also called Publish / Subscribe Architecture

- MQTT has a client/server model, where every sensor is a client and connects to a server, known as a broker, over TCP.
- MQTT is message oriented. Every message is a discrete chunk of data, opaque to the broker.
- Every message is published to an address, known as a topic. Clients may subscribe to multiple topics. Every client subscribed to a topic receives every message published to the topic.

MQTT Components:

1) Publisher
2) Subscriber
3) Broker
4) Topic

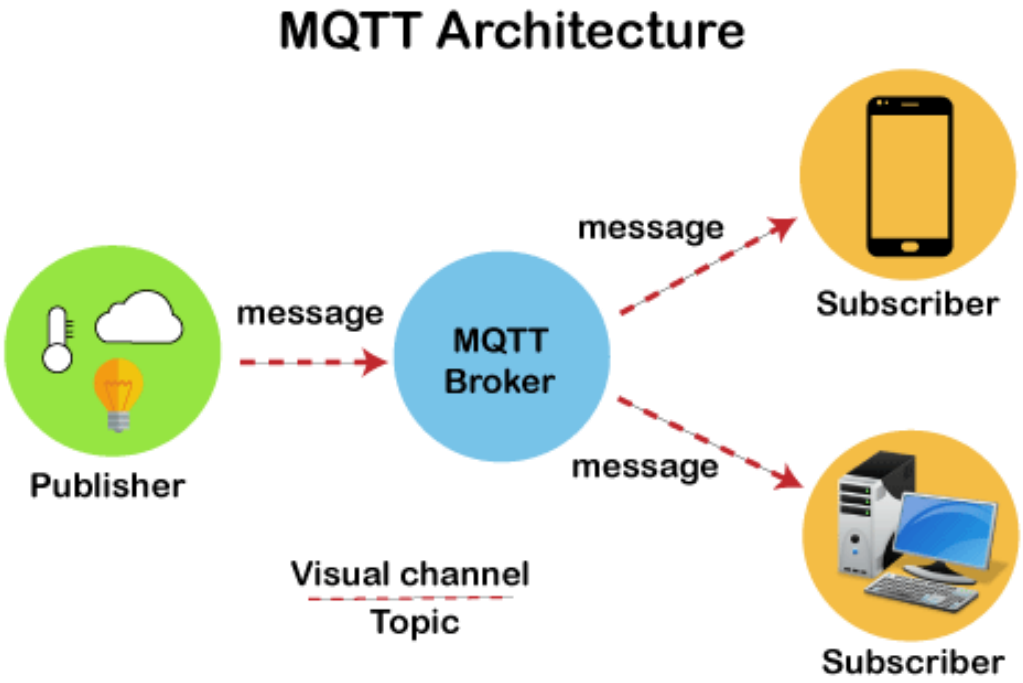


Publish: When the client sends the data to the server, then we call this operation as a publish.

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

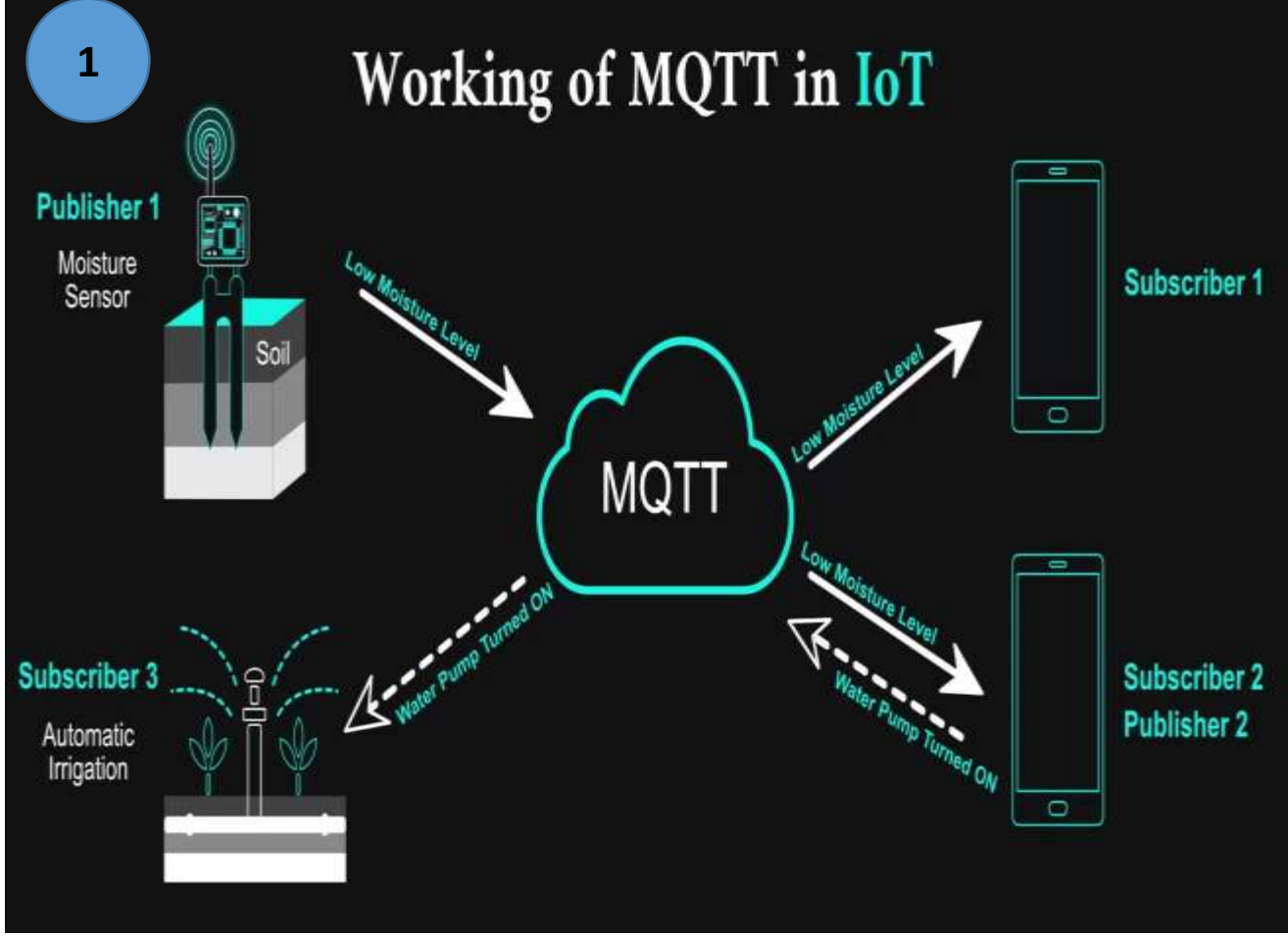
Server: The device or a program that allows the client to publish the messages and subscribe to the messages. A server accepts the network connection from the client, accepts the messages from the client, processes the subscribe and unsubscribe requests, forwards the application messages to the client,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

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

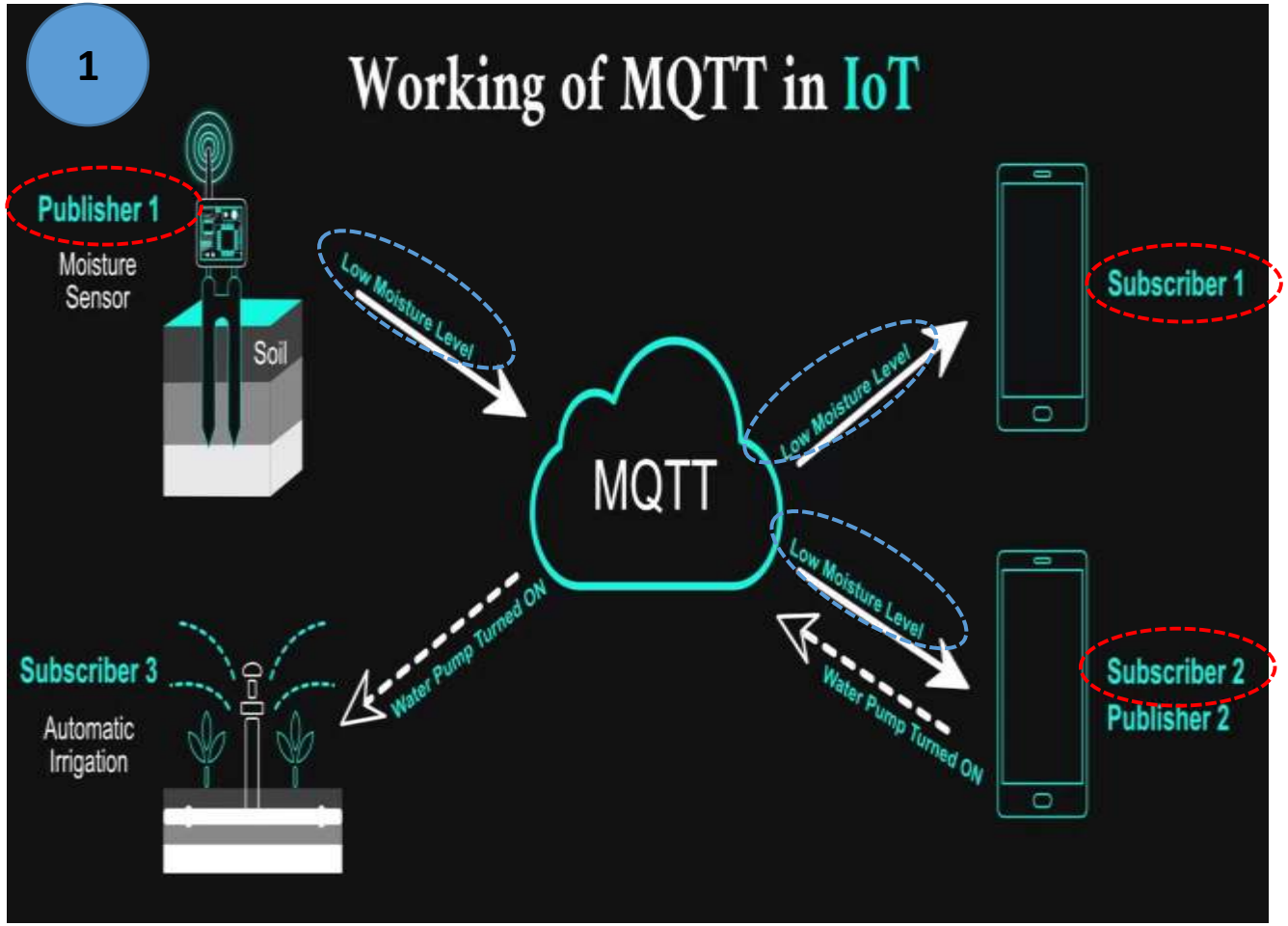


sensors based Smart Agriculture.

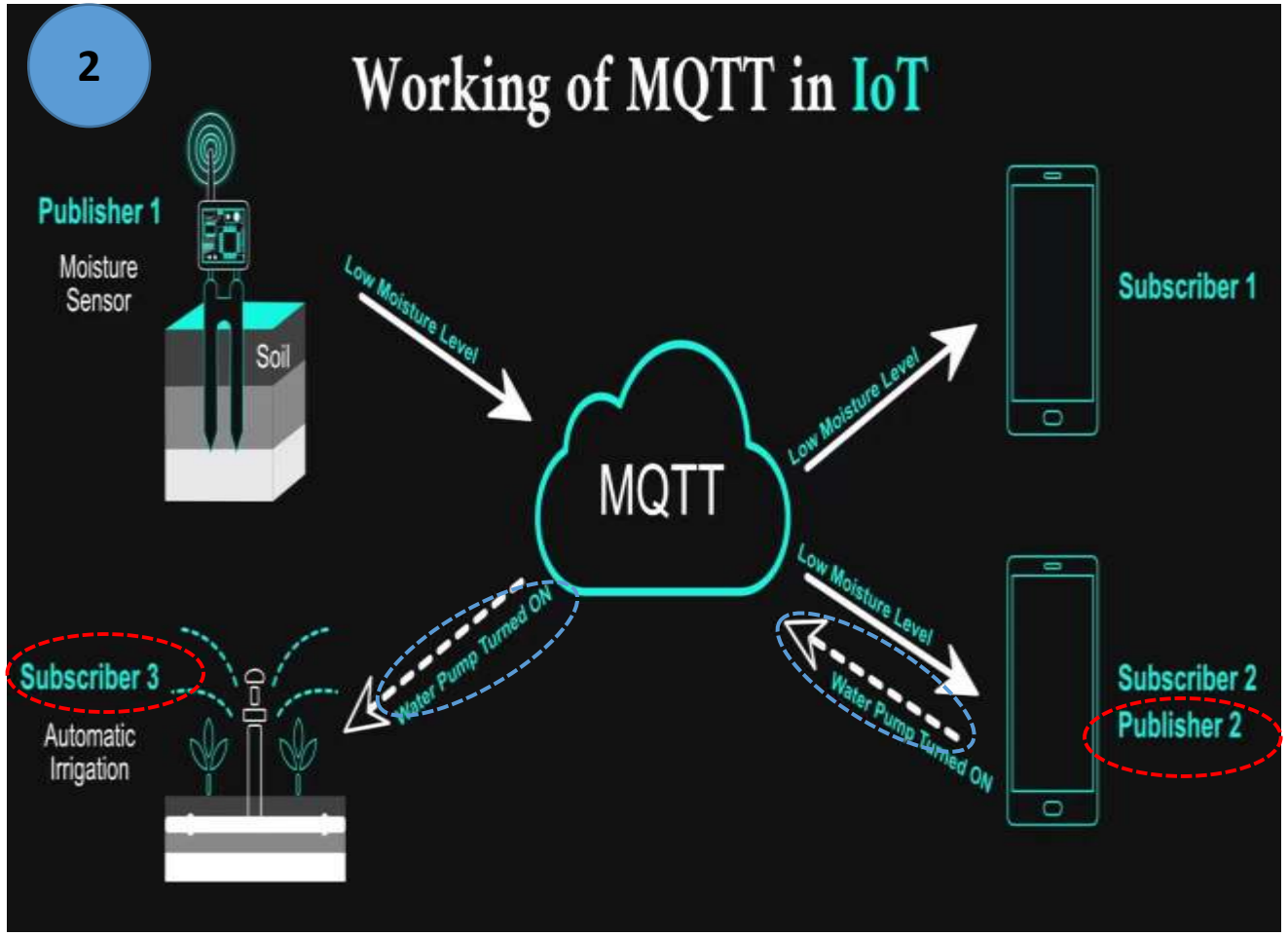
In the subscriber-publisher model, the one which sends data/ command is termed a publisher, and the one which receives data/ command is termed a subscriber.
The moisture sensor publishes moisture data and sends it to the broker, which when received by the subscriber can generate an alert based on the threshold value. If the moisture level is low, and the user wants to turn ON the sprinkler, then the user device publishes data/command to turn the sprinkler ON and the sprinkler receives (working as a subscriber here) command and actuates the water valve.



sensors based Smart Agriculture.



sensors based Smart Agriculture.



XMPP – eXtensible Messaging and Presence Protocol

Extensible means here with this protocol many features can be added in future

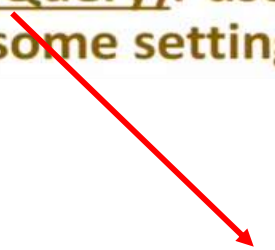
Messaging means this protocol is designed for messaging system like that you have chat system for that.

Presence means this protocol is designed presence when somebody message at that time one can see online status of their context.

- It is a protocol (standard) that is used to build chat systems.
- It uses XML to exchange data between client and server.
- eXtensible: The protocol can be {has been} extended with new features.
- Messaging: Send one-to-one and group messages.
- Presence: See your contact's online status.

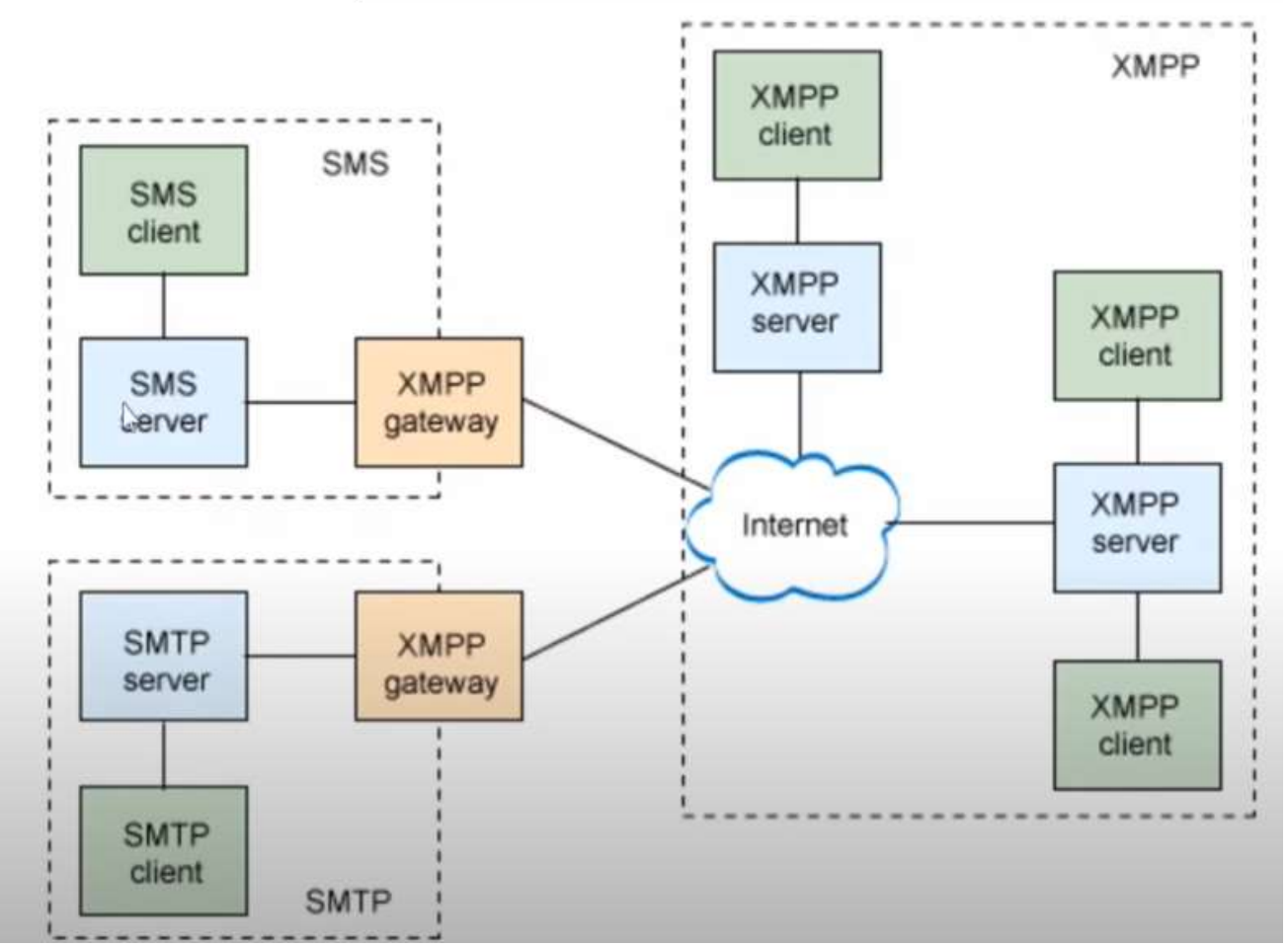
XMPP Stanzas

- Stanza: It is the basic unit of communication in XMPP.
- There are three types of stanzas in XMPP
  - Message: used when you want to send messages
  - Presence: used to send online status information and to control subscription status between contacts.
  - IQ (Info/Query): used to [get] some information from the server or to [set] or apply some settings.



For example if you want to update the profile photo

XMPP Architecture



- XMPP can be interconnected with verities of other protocols like SMS(Short message service) protocol and SMTP(Simple mail transfer) protocol.
- SMS client is connected with the SMS server and for interconnection of SMS and XMPP, XMPP gateway is used.
- Similarly SMTP client is connected with SMTP server and interconnected with XMPP through XMPP gateway.
- In XMPP, XMPP clients are connected with XMPP server and communicate with all other types of protocols like SMS and SMTP.

XMPP Features

- Peer-to-Peer Sessions: XMPP supports machine-to-machine or peer-to-peer across diverse set of networks.
- Multi-User chat: XMPP supports group and conference chat.
- Encryption: Point-to-point encryption is done by TLS {Transport Layer Security} and OTR {Off The Record messaging} is an extension of XMPP that enables encryption of messages and data.
- Connection to other protocols: using XMPP Gateways other protocols like SMS and SMTP can also be connected with XMPP servers for different services.

XMPP Use Cases

1. Real Time Web Chat

Live chat

2. Instant Messaging

Instant message
Like to get OTP

3. Real time group chat

Whats App group chat

4. IOT device control

Turn ON AC by sending message