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.

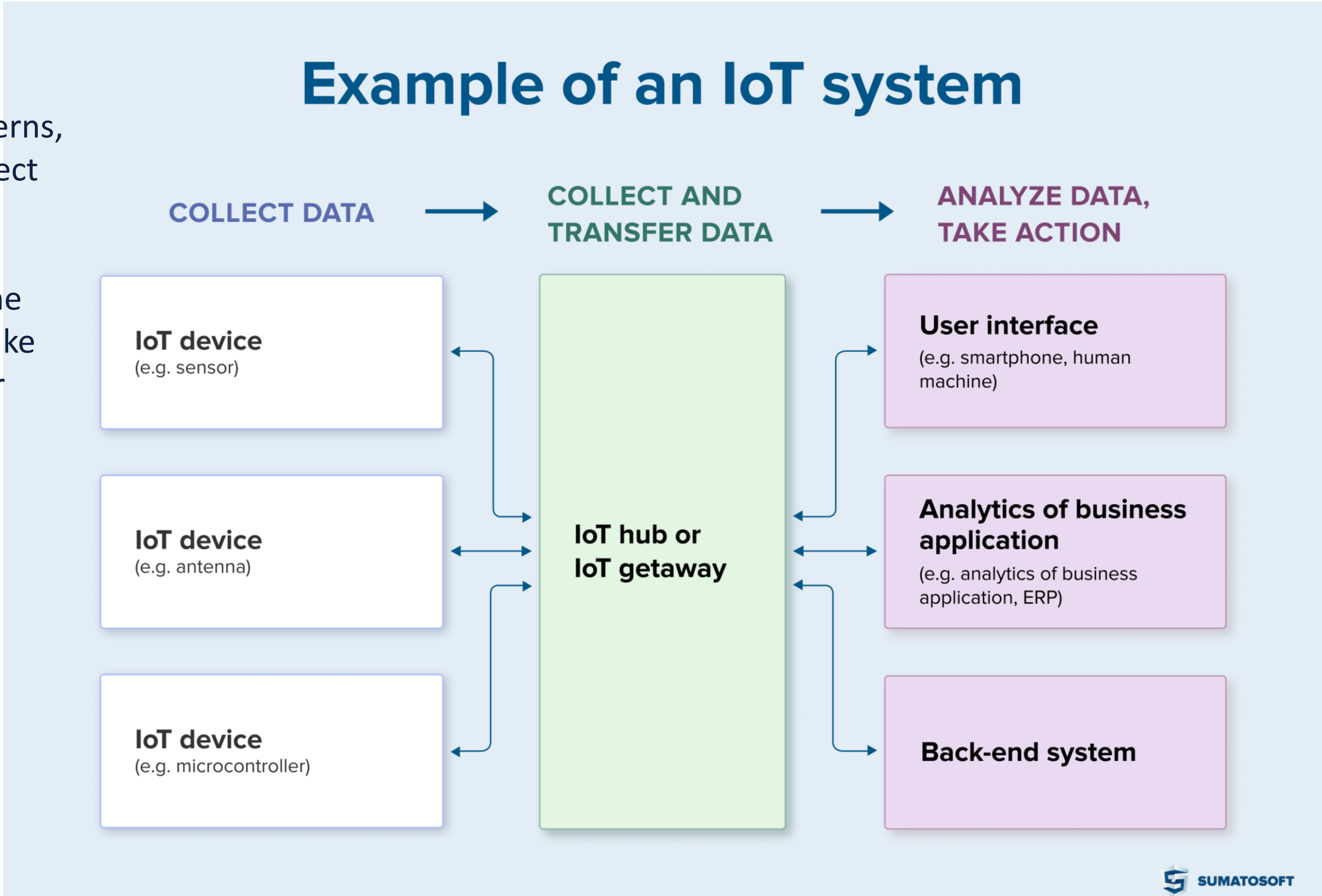
We can split above definition:

PHYSICAL OBJECTS + EMBEDDED TECHNOLOGY + COMMUNICATION

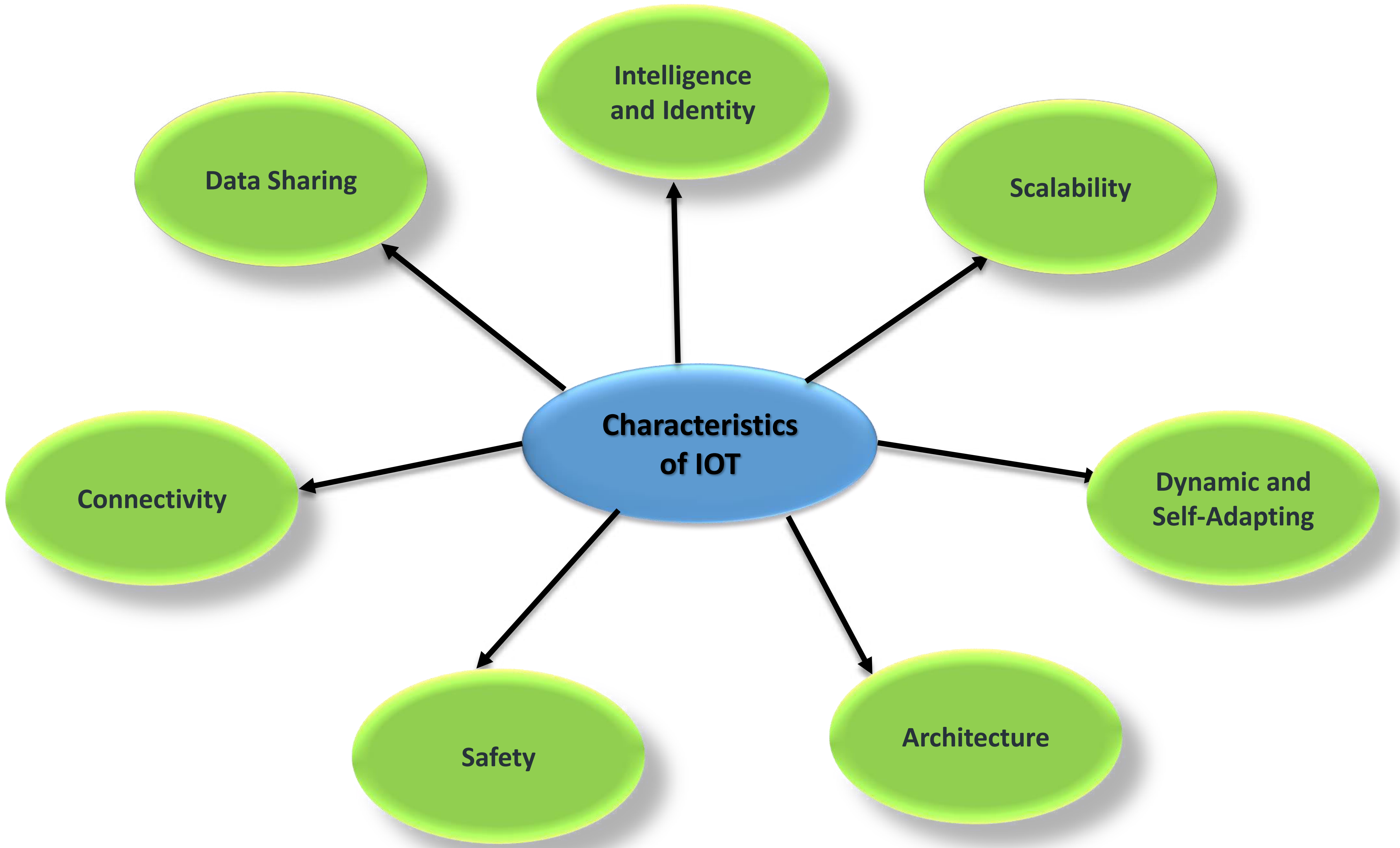
- The IoT architecture consists of smart "things" – devices that use embedded systems (processors, sensors, and hardware).
- These devices collect, send, and manage data and systems based on data acquired from the environment. IoT devices connect to IoT gateways, IoT platforms, or other devices for analysis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


Example of an IoT system
COLLECT DATA
COLLECT AND TRANSFER DATA
ANALYZE DATA, TAKE ACTION
IoT device (e.g. sensor)
IoT device (e.g. antenna)
IoT device (e.g. microcontroller)
IoT hub or IoT getaway
User interface (e.g. smartphone, human machine)
Analytics of business application (e.g. analytics of business application, ERP)
Back-end system
SUMATOSOFT

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 connectivity should be guaranteed at all times.
- With everything going on in IoT devices and hardware, with sensors and other electronics and connected hardware and control systems there needs to be a connection between various levels.

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.

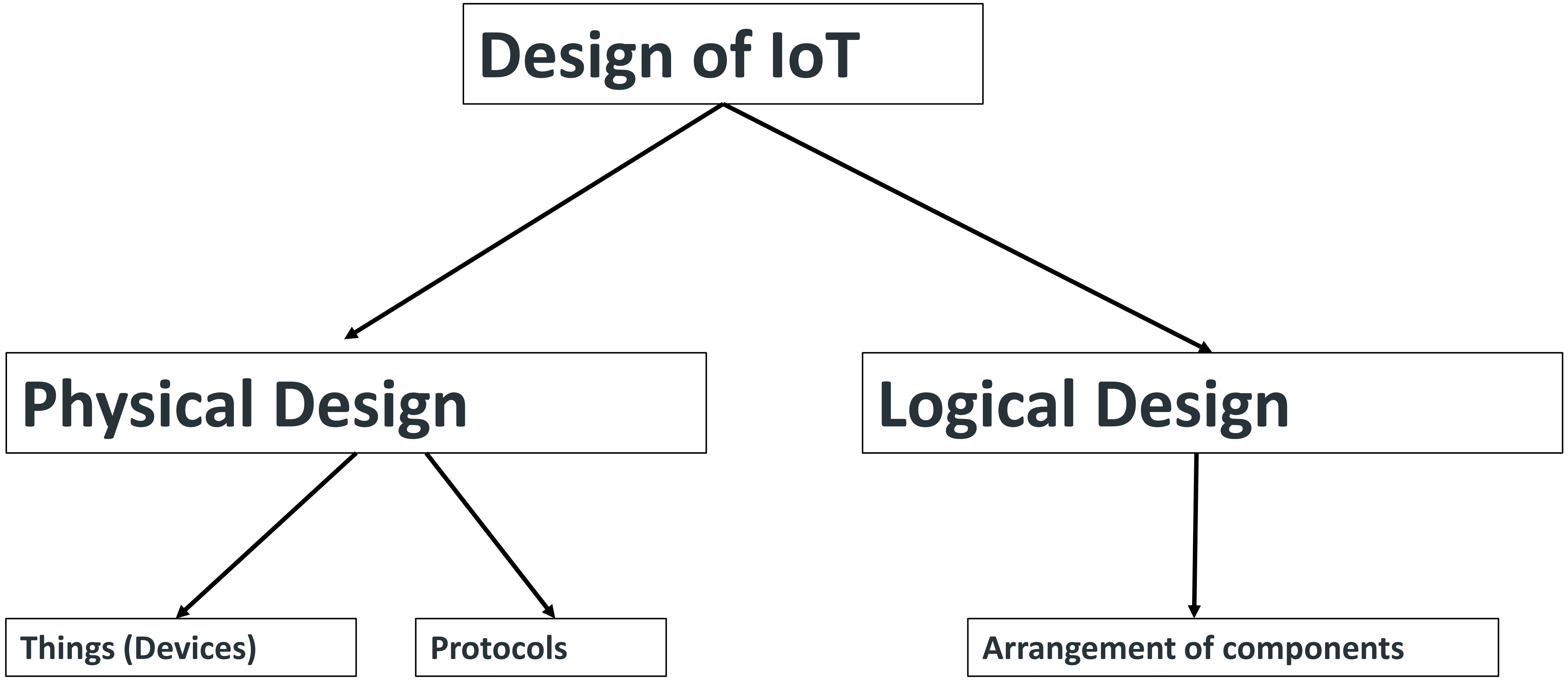
- IoT devices should dynamically adapt themselves to the changing contexts and scenarios. Assume a camera meant for the surveillance. It should be adaptable to work in different conditions and different light situations (morning, afternoon, night).

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.

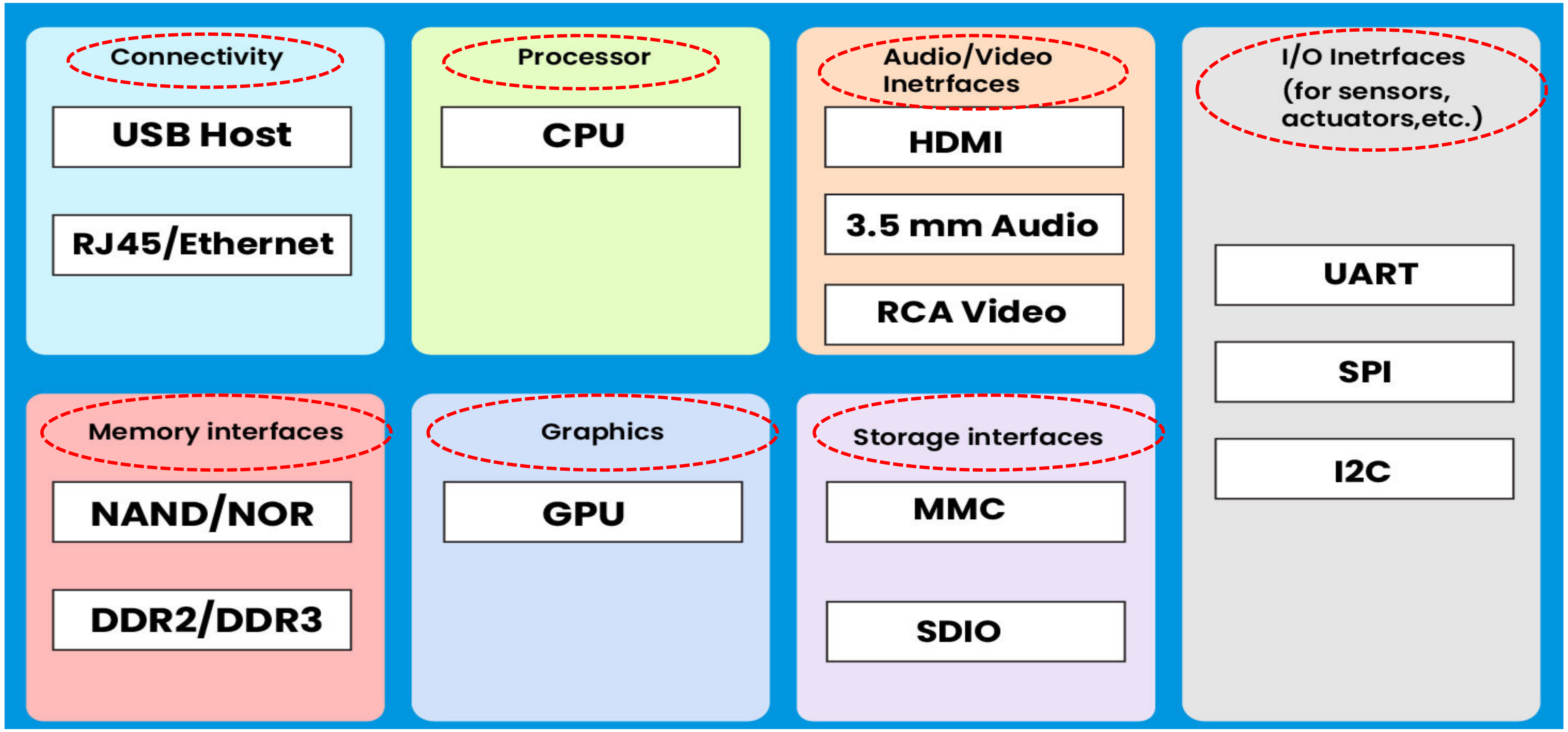
Safety –

- 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

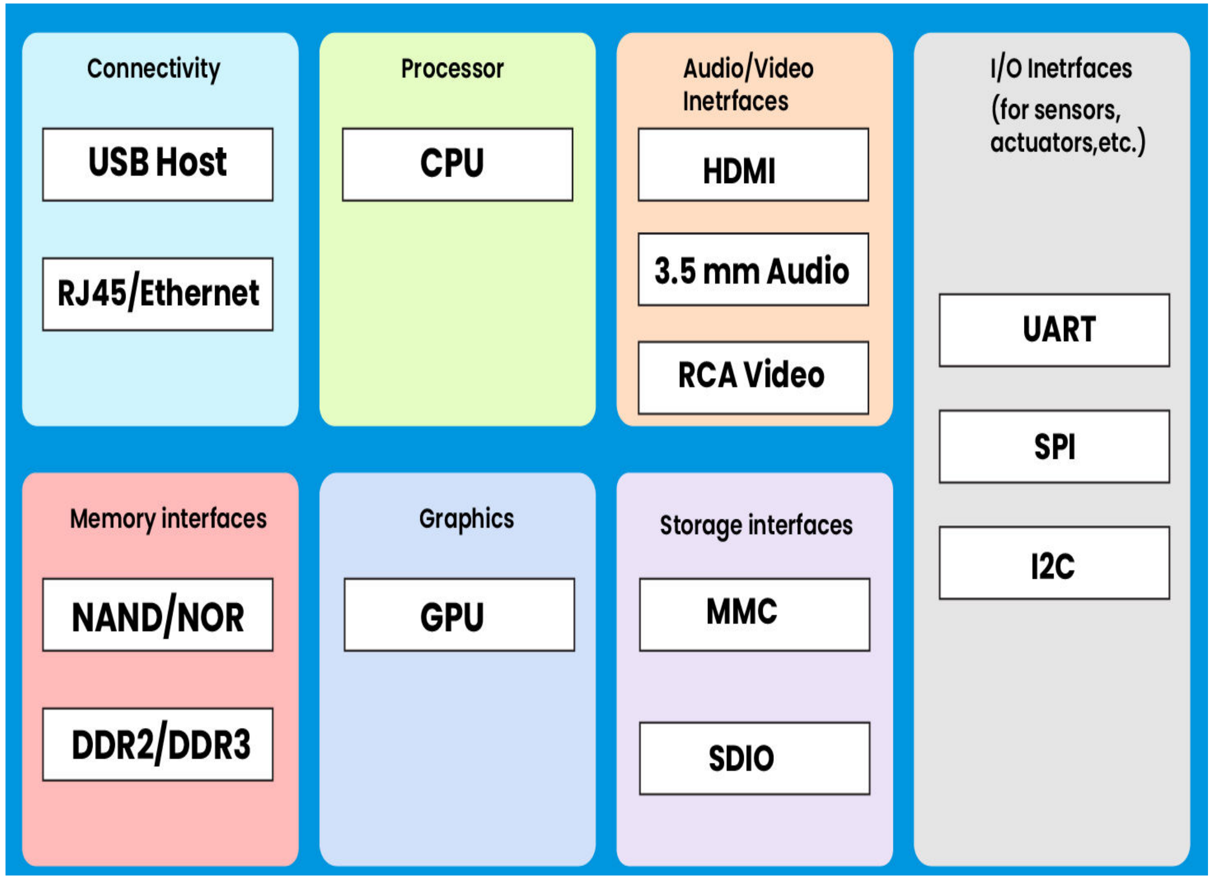
Physical Design of IoT

- 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

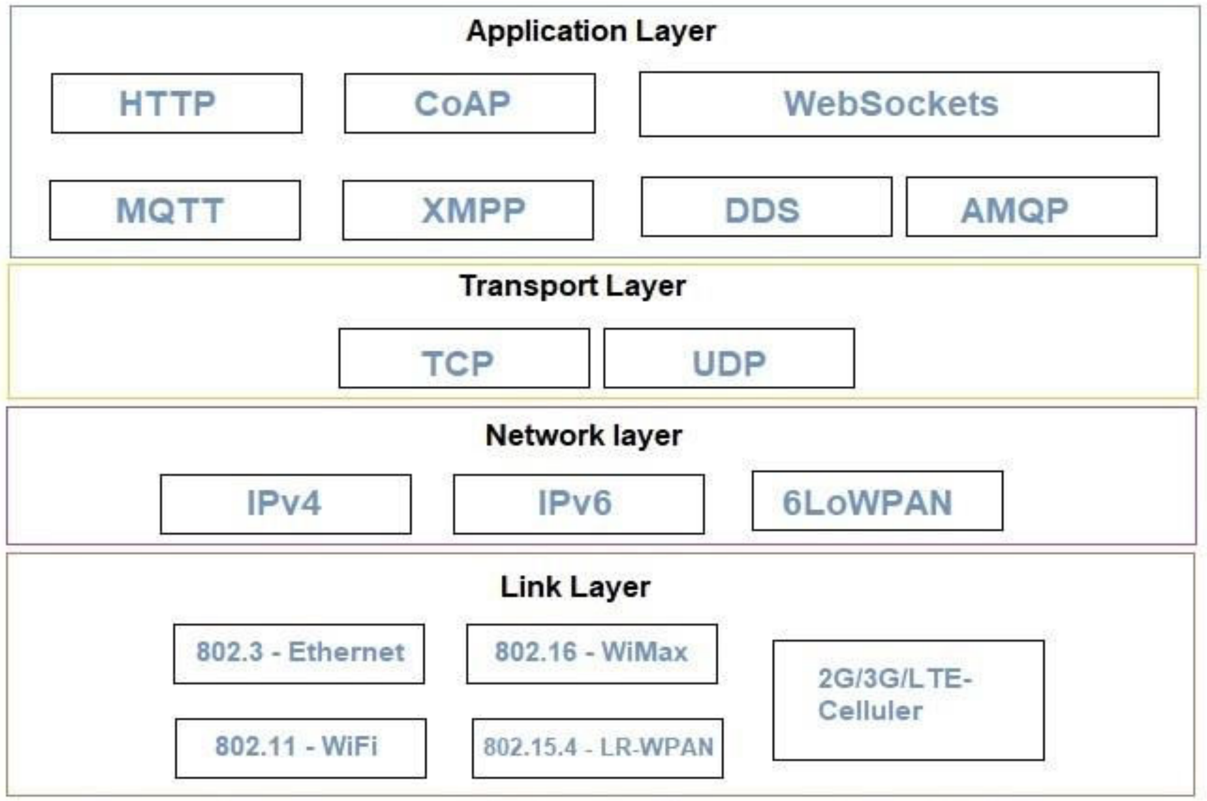
- **Connectivity:** Devices like USB hosts and ETHERNET are used for connectivity between the devices and the server.
- **Processor:** A processor like a CPU and other units are used to process the data. these data are further used to improve the decision quality of an IoT system.
- **Audio/Video Interfaces:** 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.

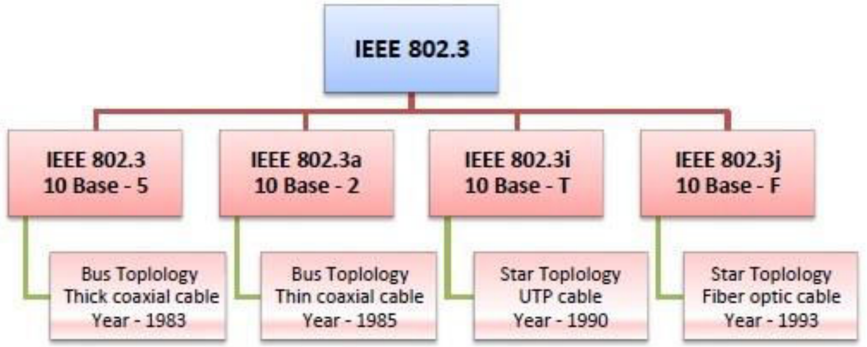


Link Layer:

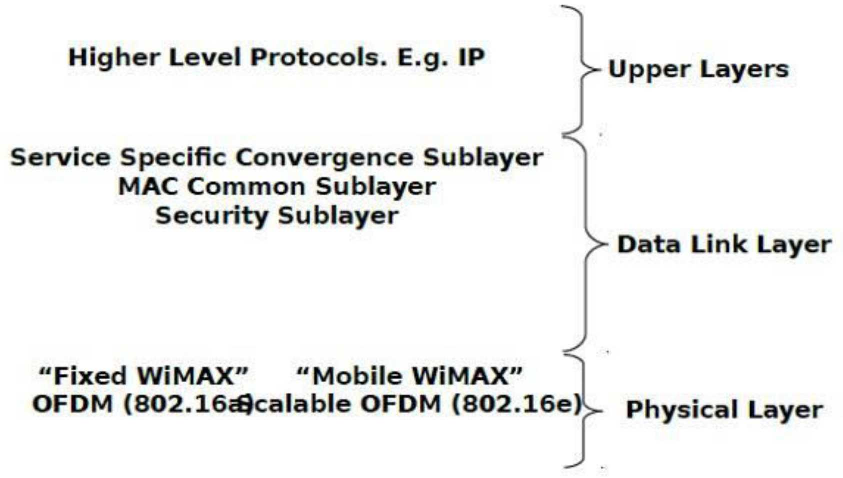
Link layer protocols determine how data is physically sent over the network's physical layer or medium (Coxial cable or other or radio wave).

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.


IEEE 802.3
IEEE 802.3 10 Base - 5
IEEE 802.3a 10 Base - 2
IEEE 802.3i 10 Base - T
IEEE 802.3j 10 Base - F
Bus Toplology Thick coaxial cable Year - 1983
Bus Toplology Thin coaxial cable Year - 1985
Star Toplology UTP cable Year - 1990
Star Toplology Fiber optic cable Year - 1993

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.**


Higher Level Protocols. E.g. IP
Upper Layers
Service Specific Convergence Sublayer
MAC Common Sublayer
Security Sublayer
Data Link Layer
"Fixed WiMAX" OFDM (802.16a)
"Mobile WiMAX" Scalable OFDM (802.16e)
Physical Layer

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 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.

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 protocols are:

•Web protocol used in M2M with constrained requirements

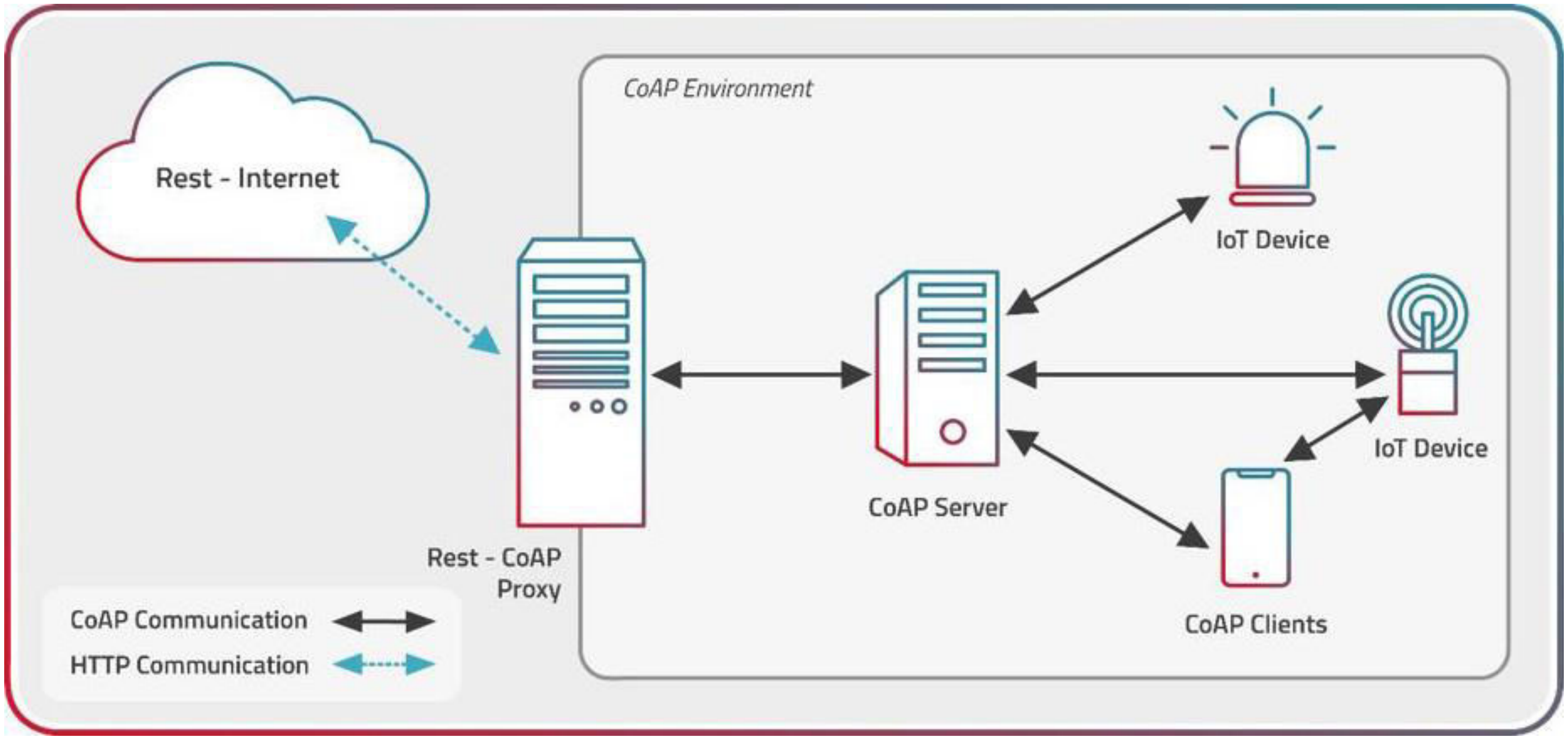
•Asynchronous message exchange

•Low overhead and very simple to parse

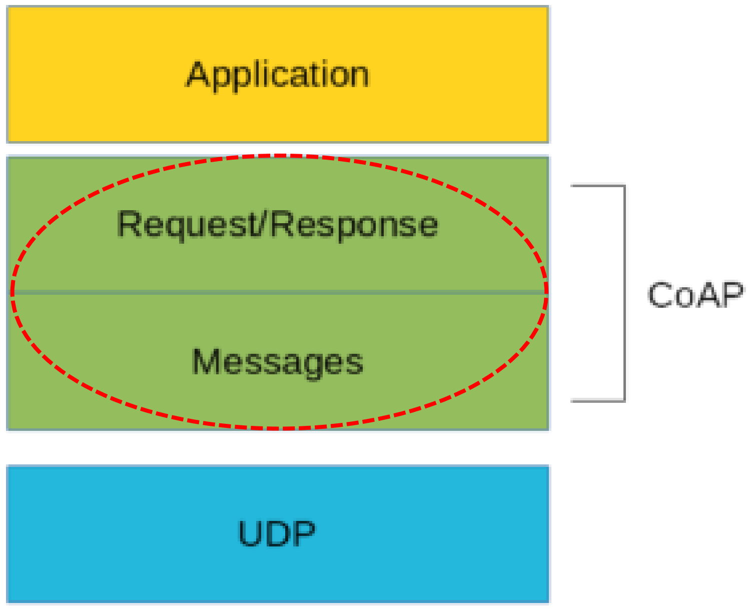
•URI and content-type support

•Proxy and caching capabilities

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 :

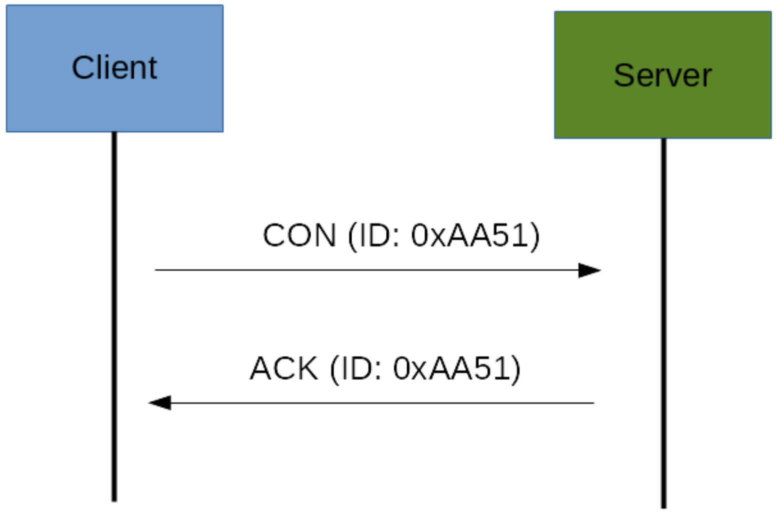
CoAP supports four different message types:

- Confirmable
- Non-confirmable
- Acknowledgment
- Reset

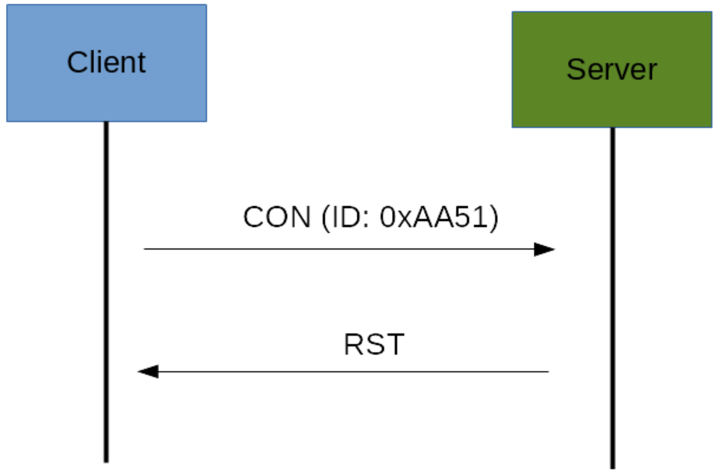
CON (Confirmable) Message:

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 Reset message (RST) instead of the Acknowledge message (ACK)



NON (Non-Confirmable) Message:

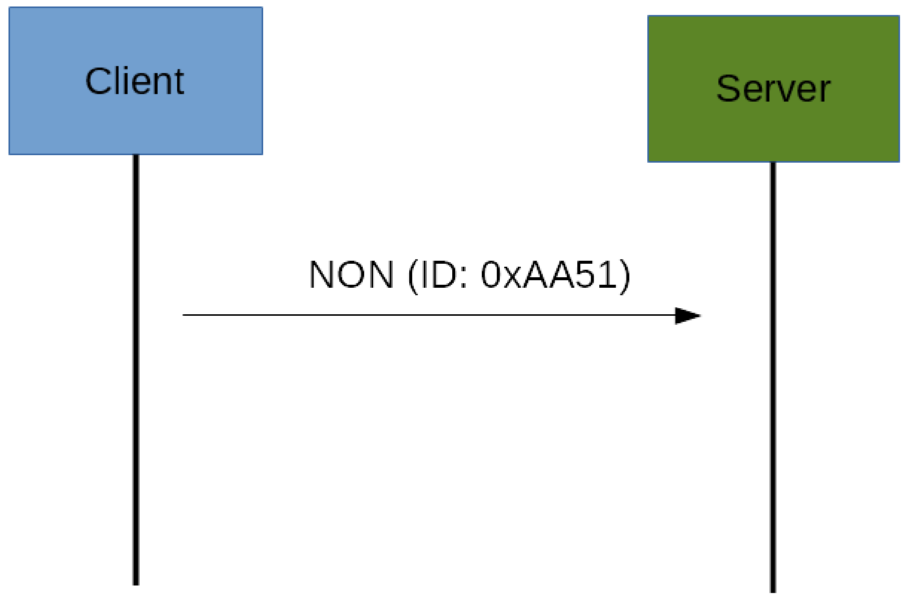
NON messages are used for faster communication without requiring acknowledgment.

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.

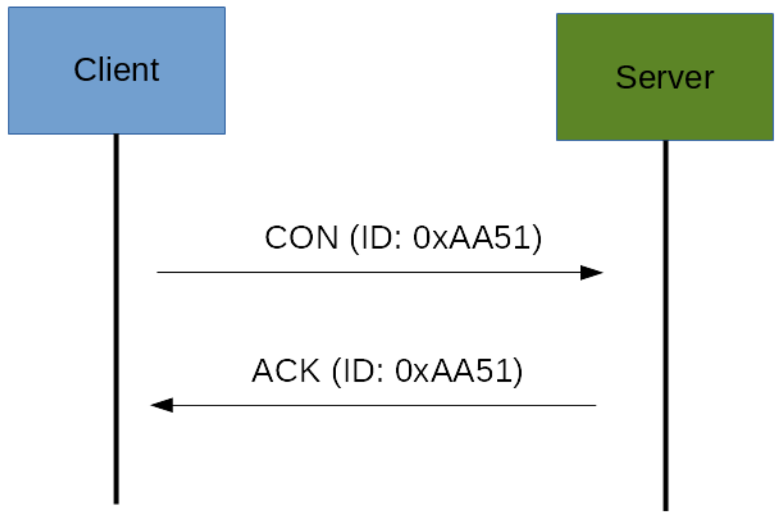


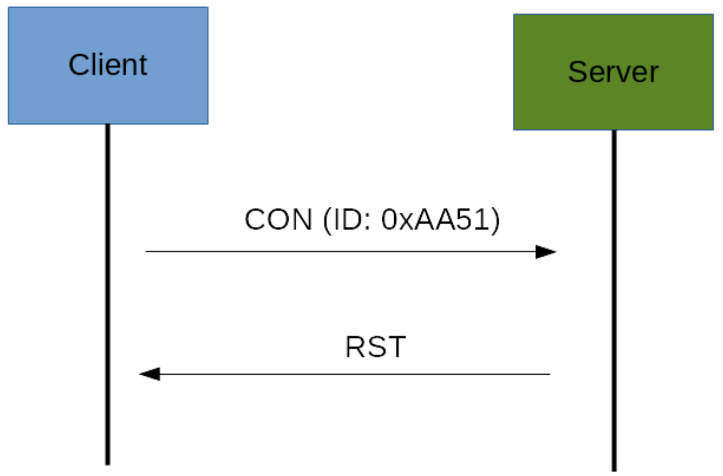
ACK (Acknowledgment) Message:

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:

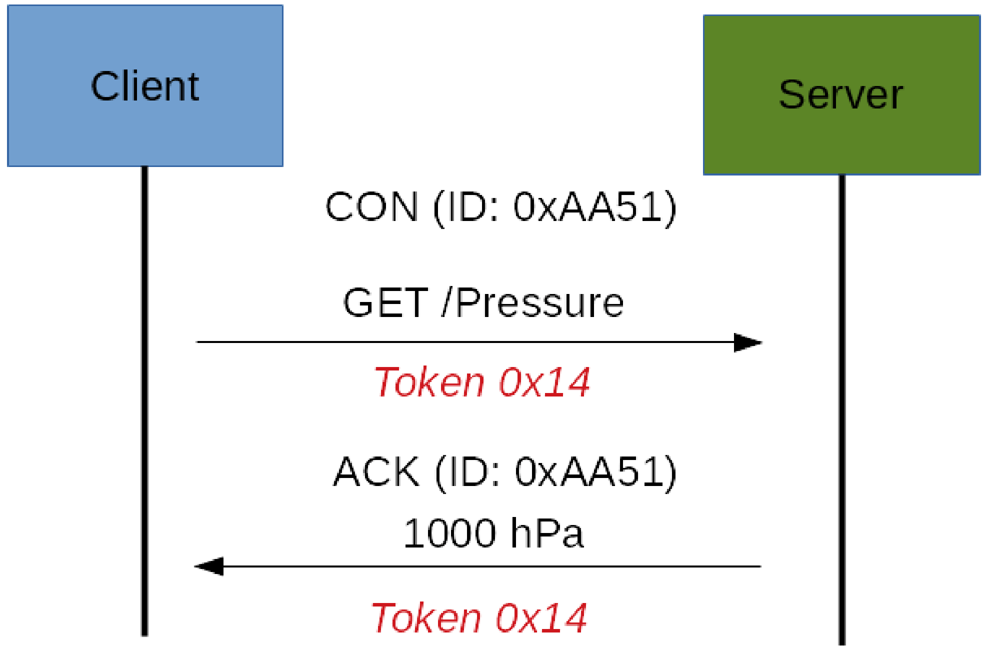
RST messages are sent to cancel a pending CON message that hasn't yet been acknowledged.
They are typically used when a receiver cannot or chooses ‘not to’ process a pending message.





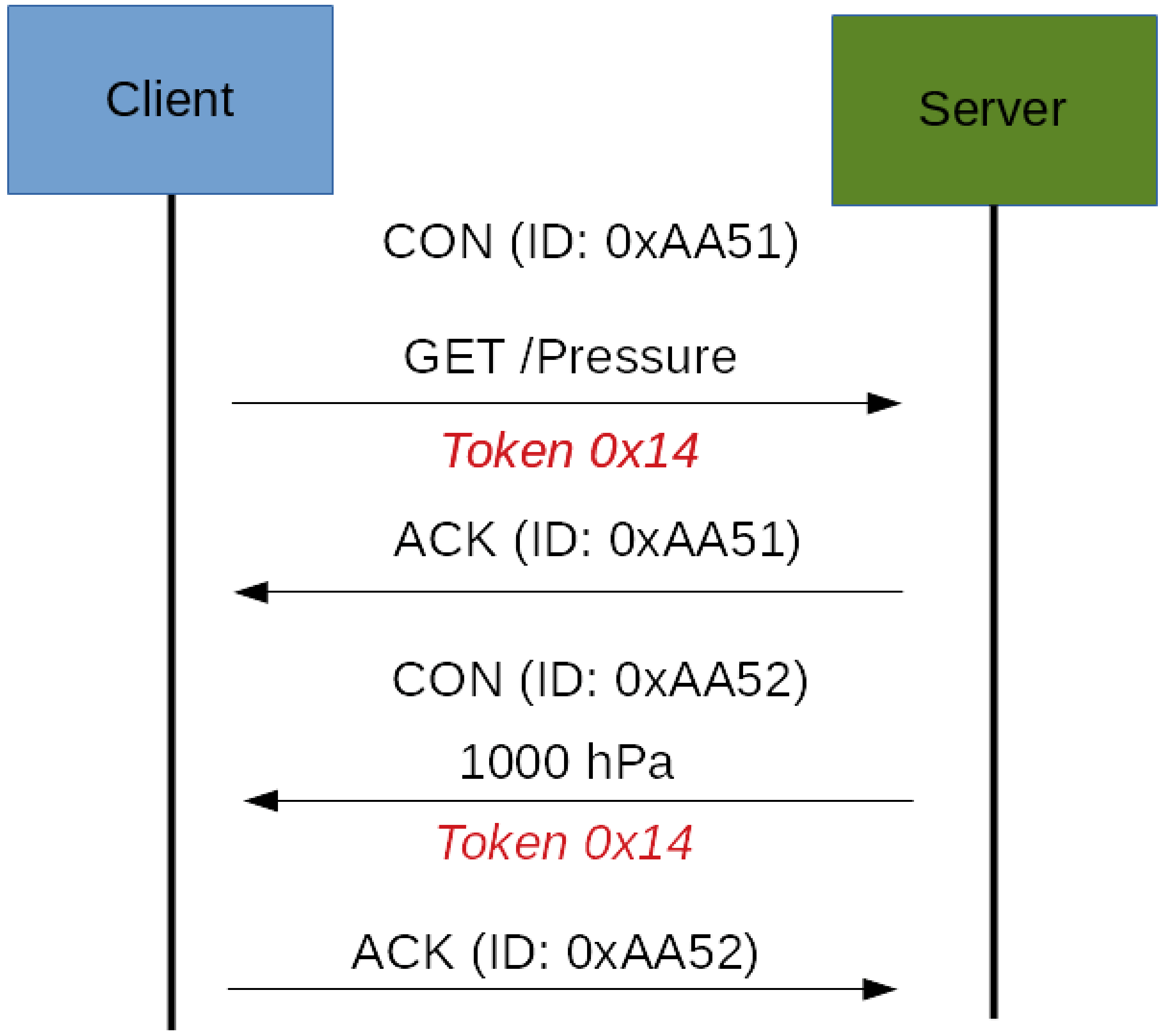
2. Request/Response :

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:

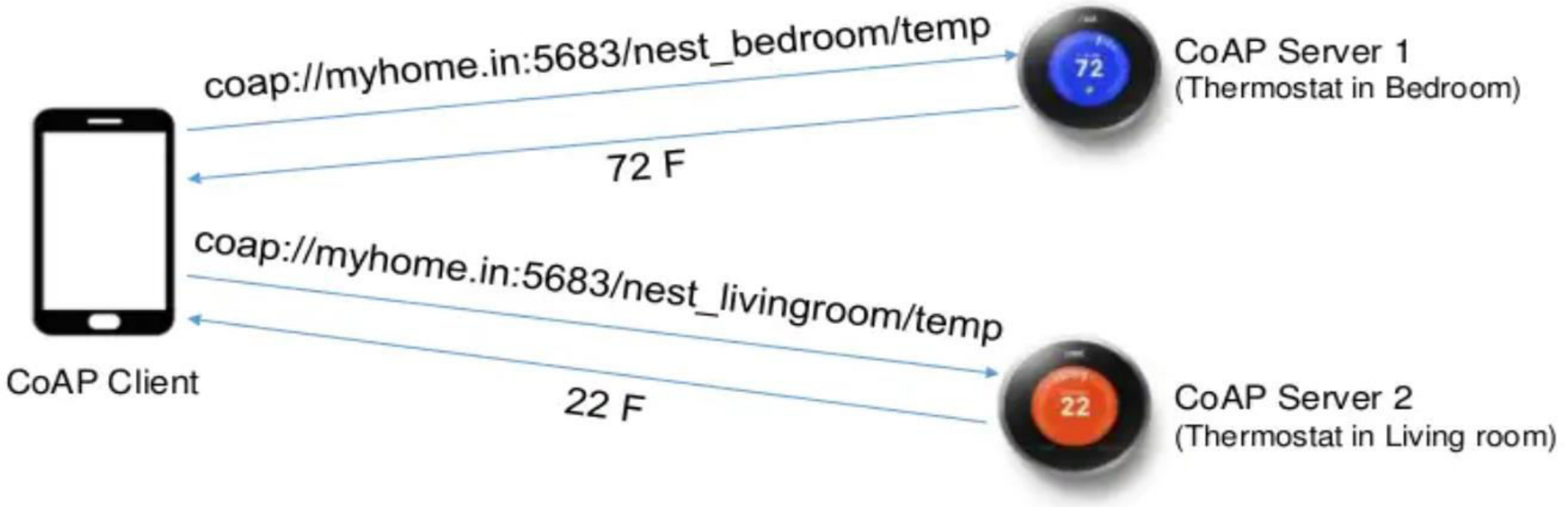


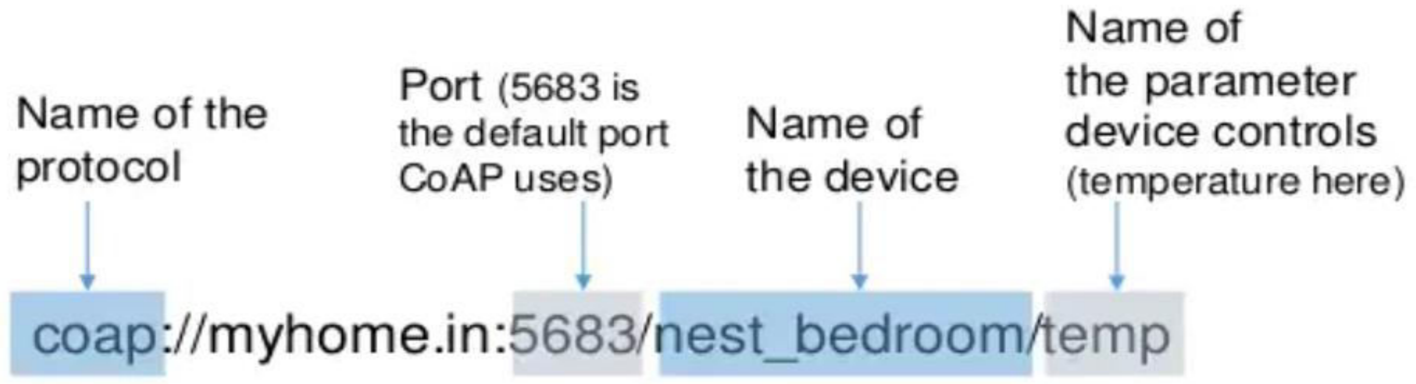
As you can notice in the CoAP message, there is a **Token**. The Token is different from the Message-ID and it is used to match the **request and the response.**

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:

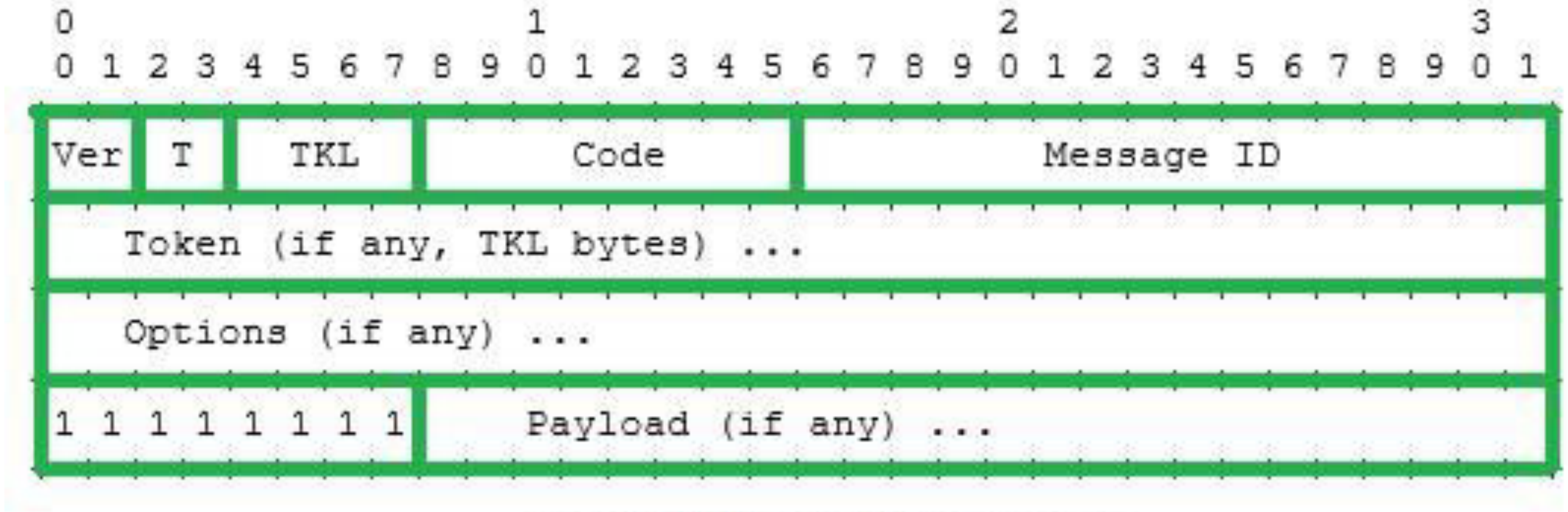


CoAP – Request Response


coap://myhome.in:5683/nest_bedroom/temp
72
CoAP Server 1
(Thermostat in Bedroom)
72 F
coap://myhome.in:5683/nest_livingroom/temp
CoAP Client
22
22 F
CoAP Server 2
(Thermostat in Living room)


Name of the protocol
Port (5683 is the default port CoAP uses)
Name of the device
Name of the parameter device controls (temperature here)
coap://myhome.in:5683/nest_bedroom/temp

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.

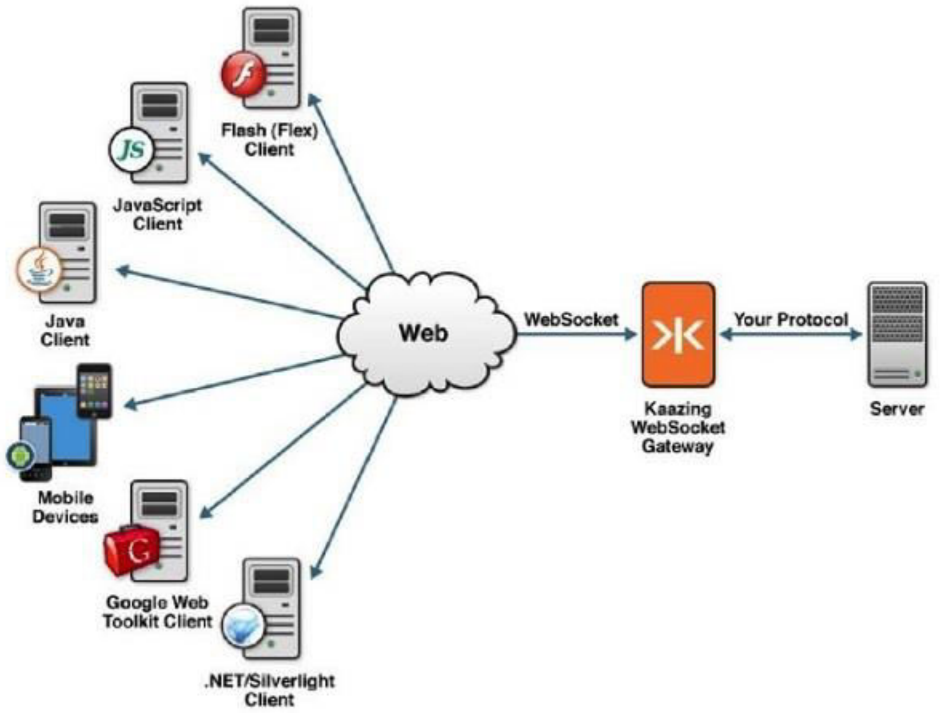
WebSocket :

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.

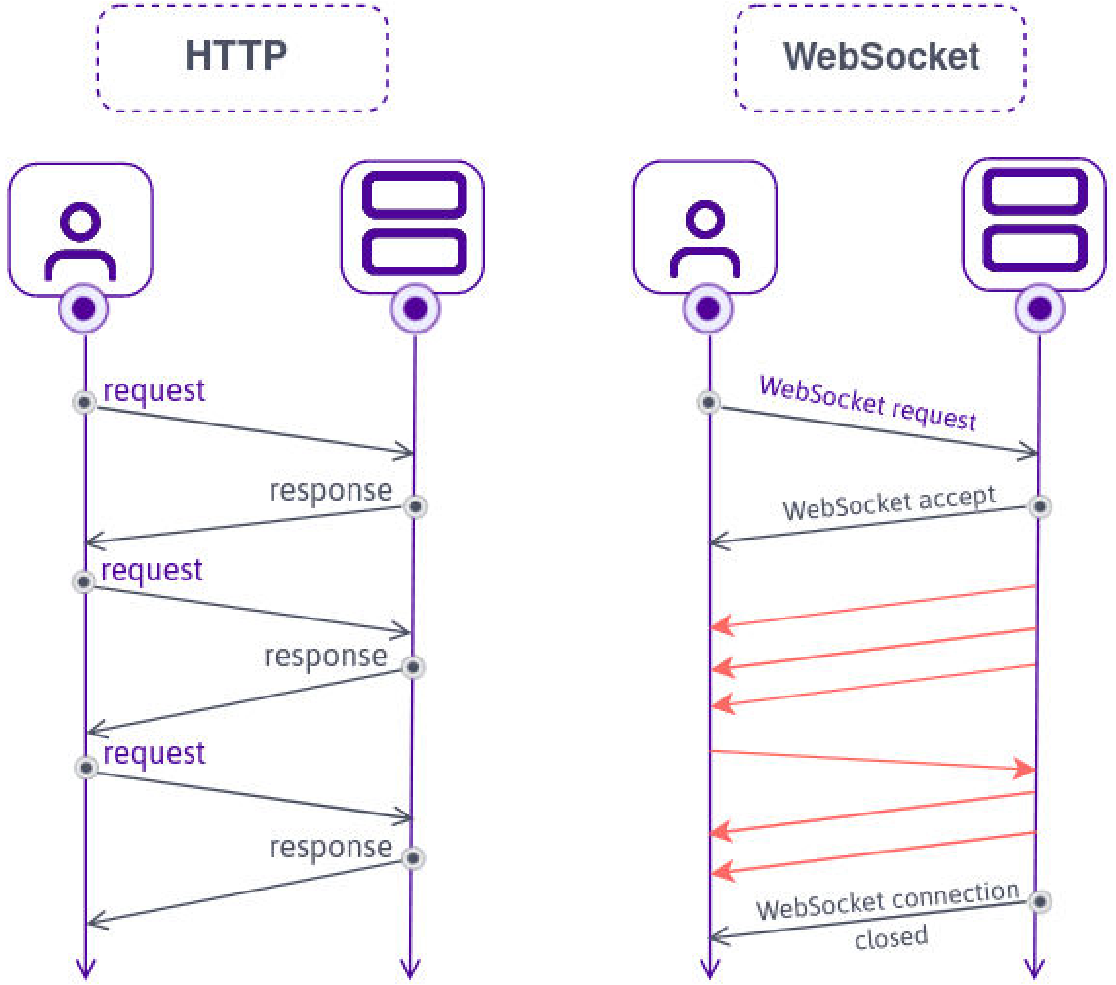
WebSocket communication presents a suitable protocol for the IoT environment where bundles of data are transmitted continuously within multiple devices. A WebSocket makes server and device communication easy. A server needs a WebSocket library to be installed and we need to have the WebSocket client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

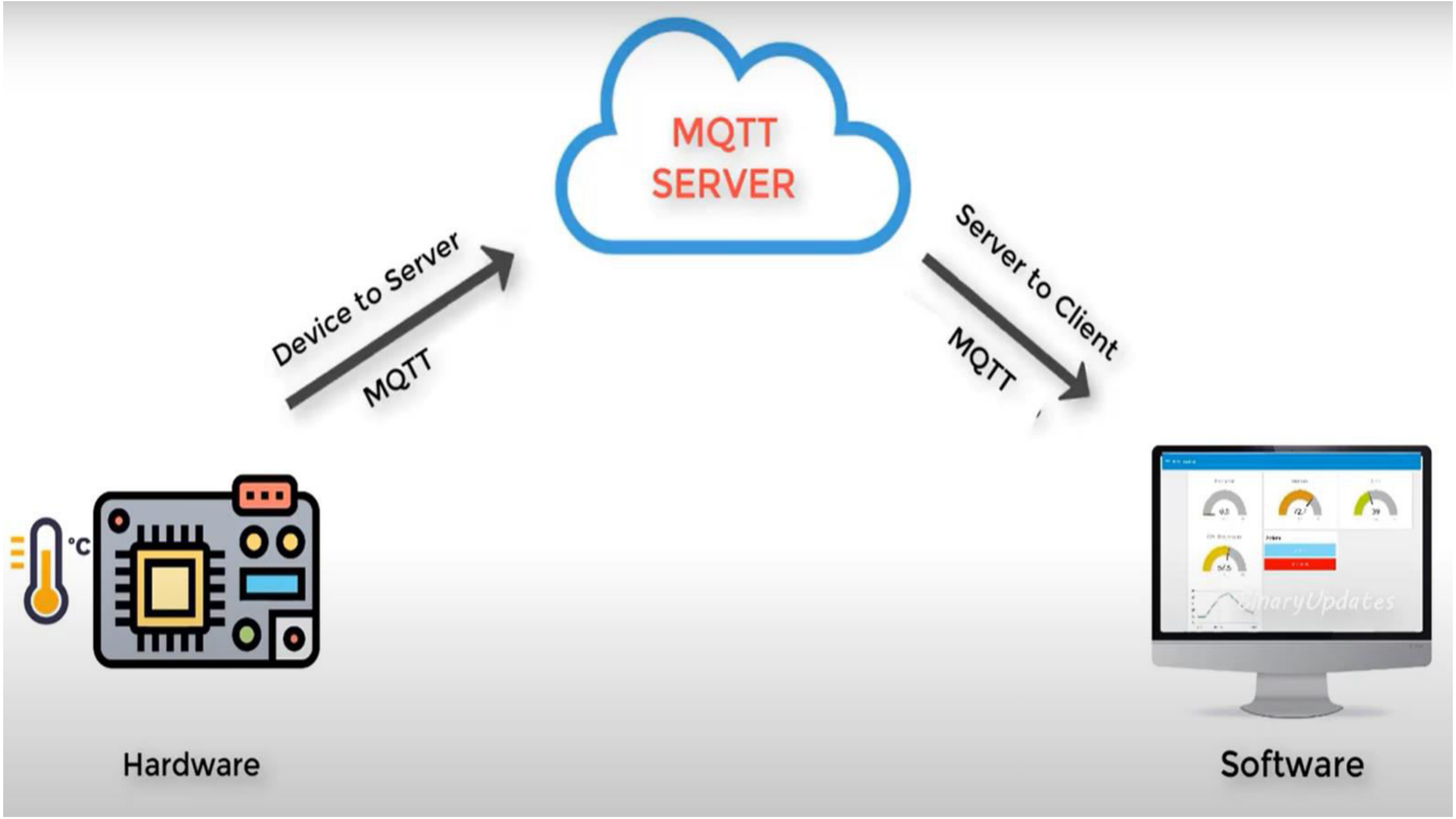


MQTT

Message Queue Telemetry Transport Protocol

Message queuing telemetry support (MQTT) is defined as a low bandwidth consumption machine-to-machine protocol 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.

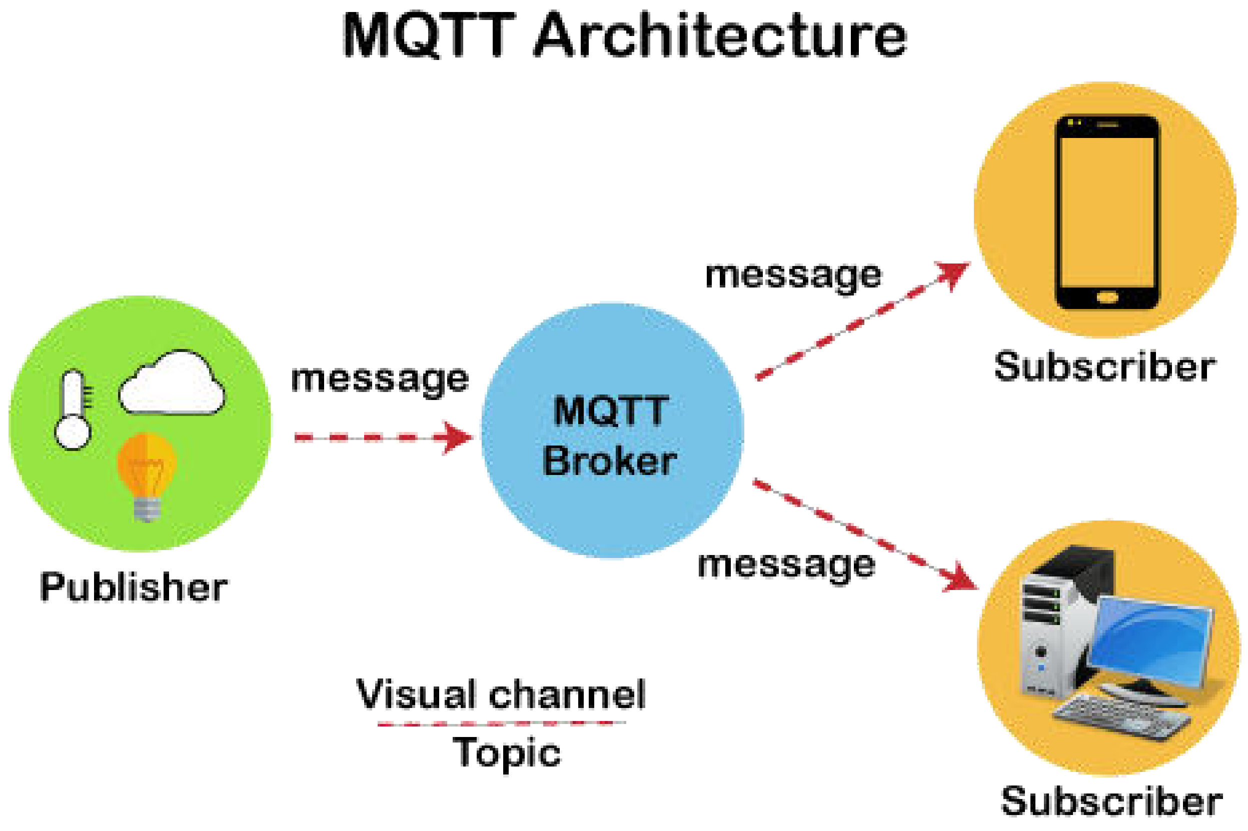

MQTT SERVER
Device to Server
MQTT
Server to Client
MQTT
°C
Hardware
Software
BinaryUpdates

Architecture of MQTT

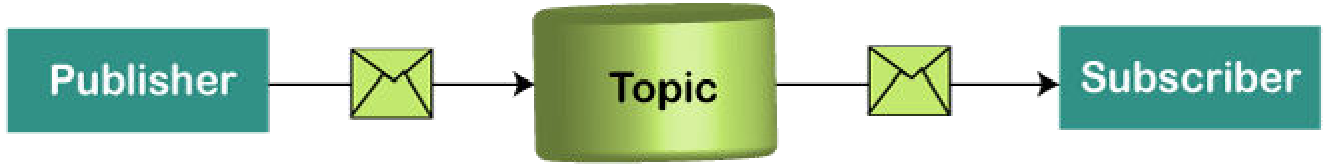
MQTT architecture is also called Publish / Subscribe Architecture

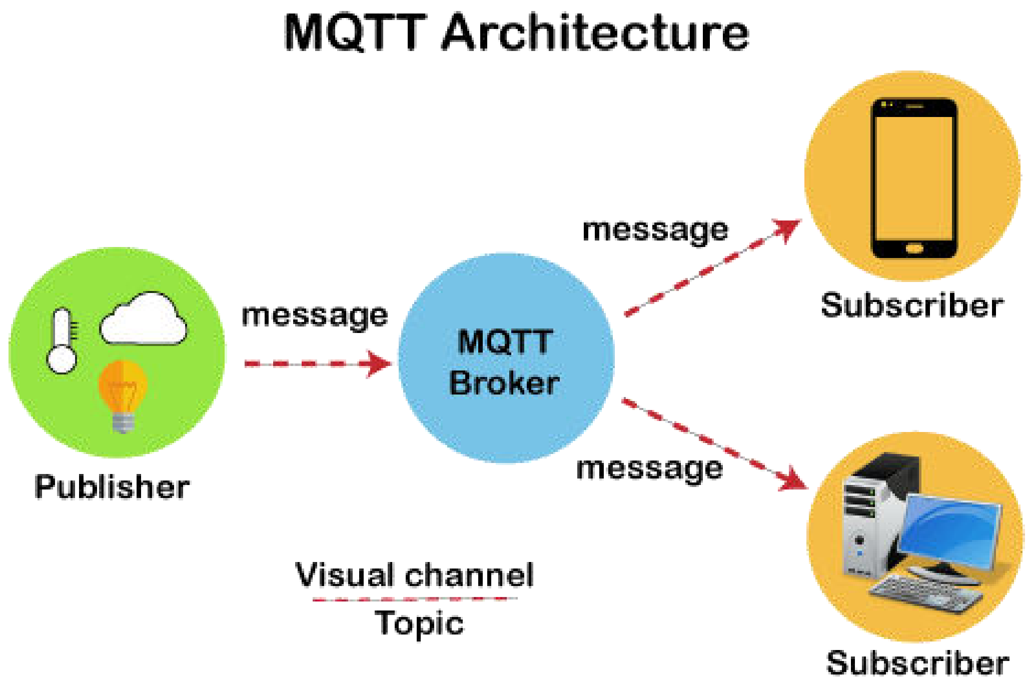
- MQTT has a client/server model, where every sensor is a client and connects to a server, known as a broker, over TCP.
- MQTT is message oriented. Every message is a discrete chunk of data, opaque to the broker.
- Every message is published to an address, known as a topic. Clients may subscribe to multiple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

MQTT Components:
1) Publisher
2) Subscriber
3) Broker
4) Topic



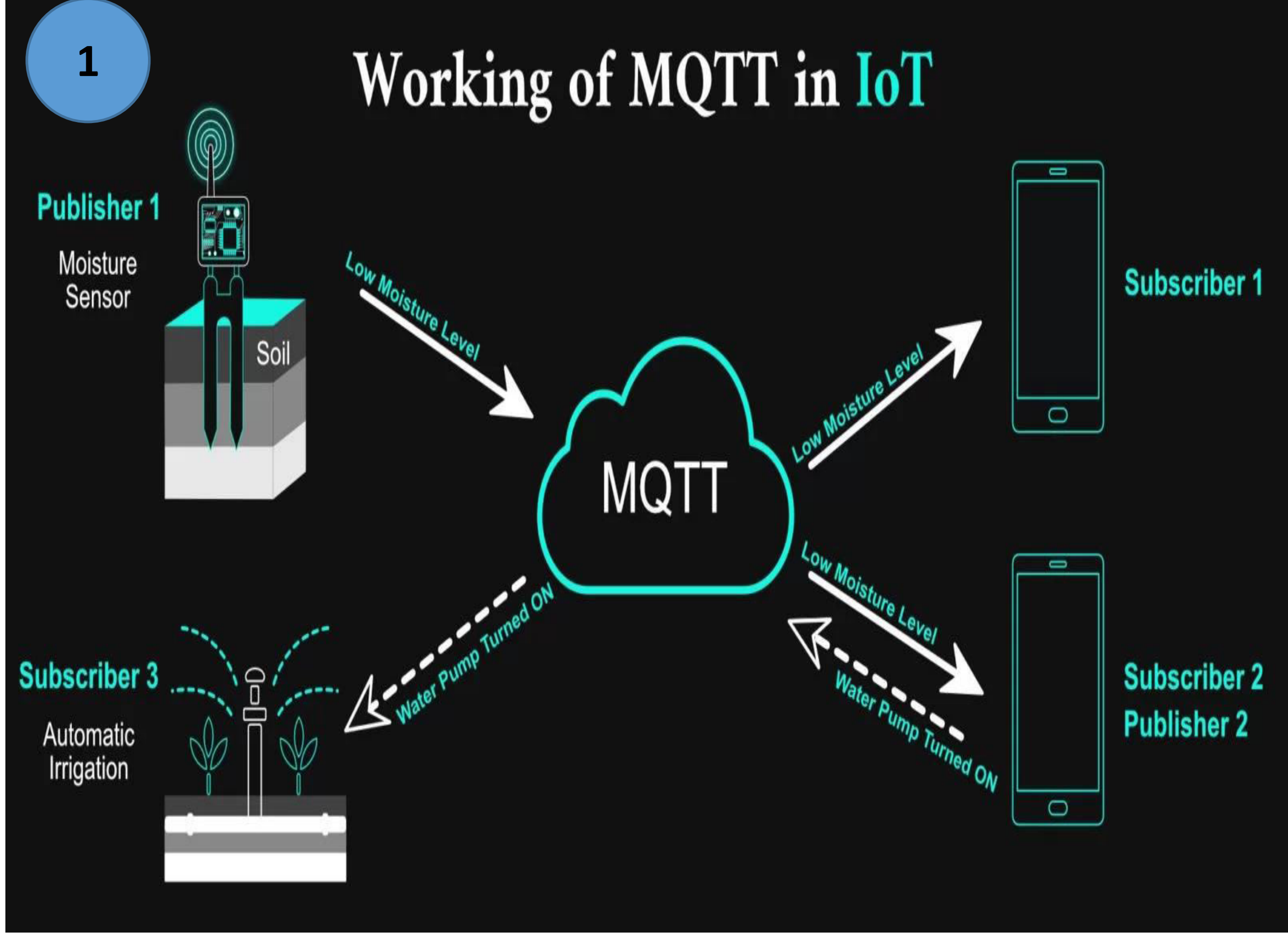
Publish: When the client sends the data to the server, then we call this operation as a publish.
Subscribe: When the client receives the data from the server, then we call this operation a subscription.
Server: The device or a program that allows the client to publish the messages and subscribe to the 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.



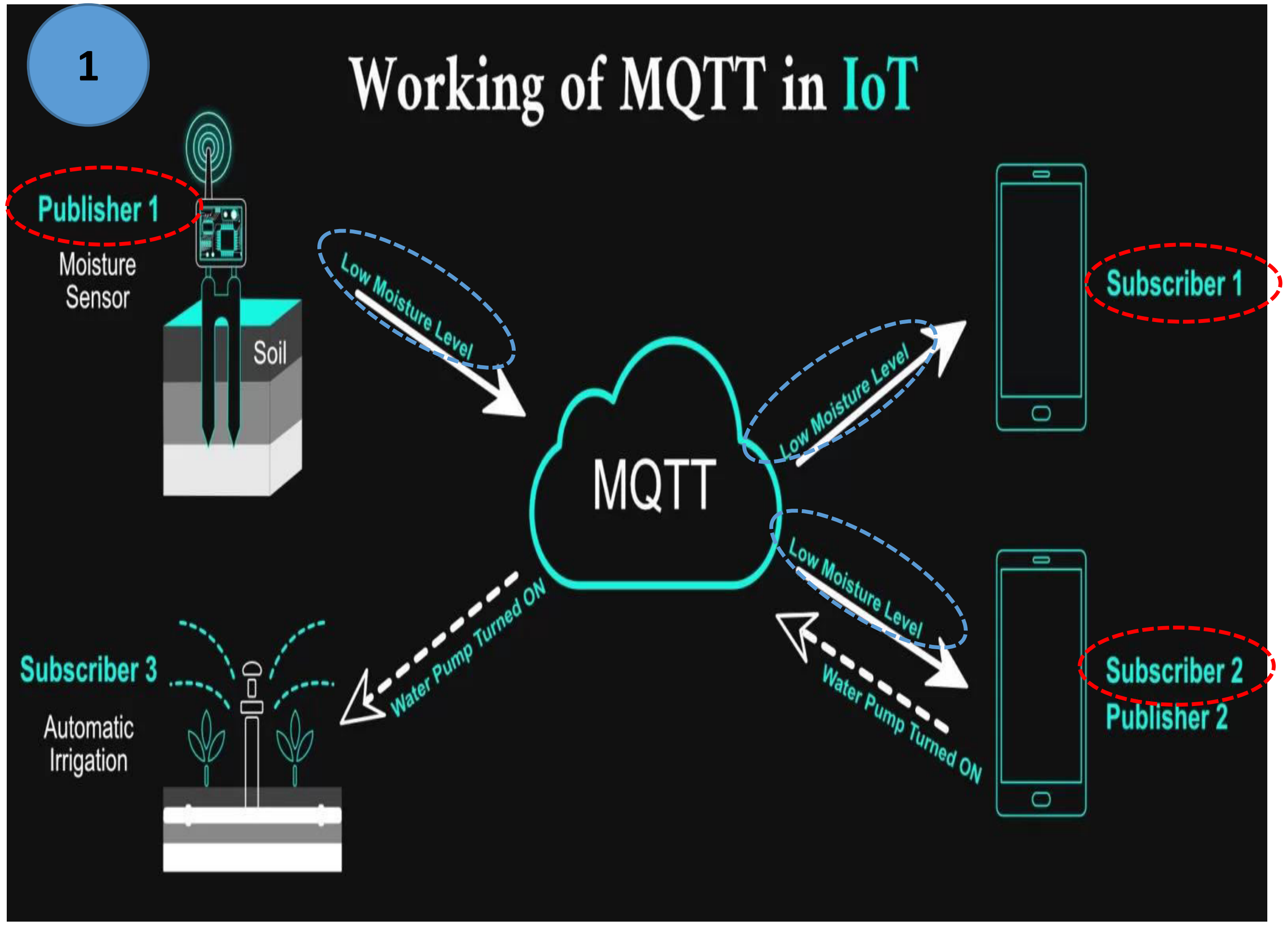


sensors based Smart Agriculture.

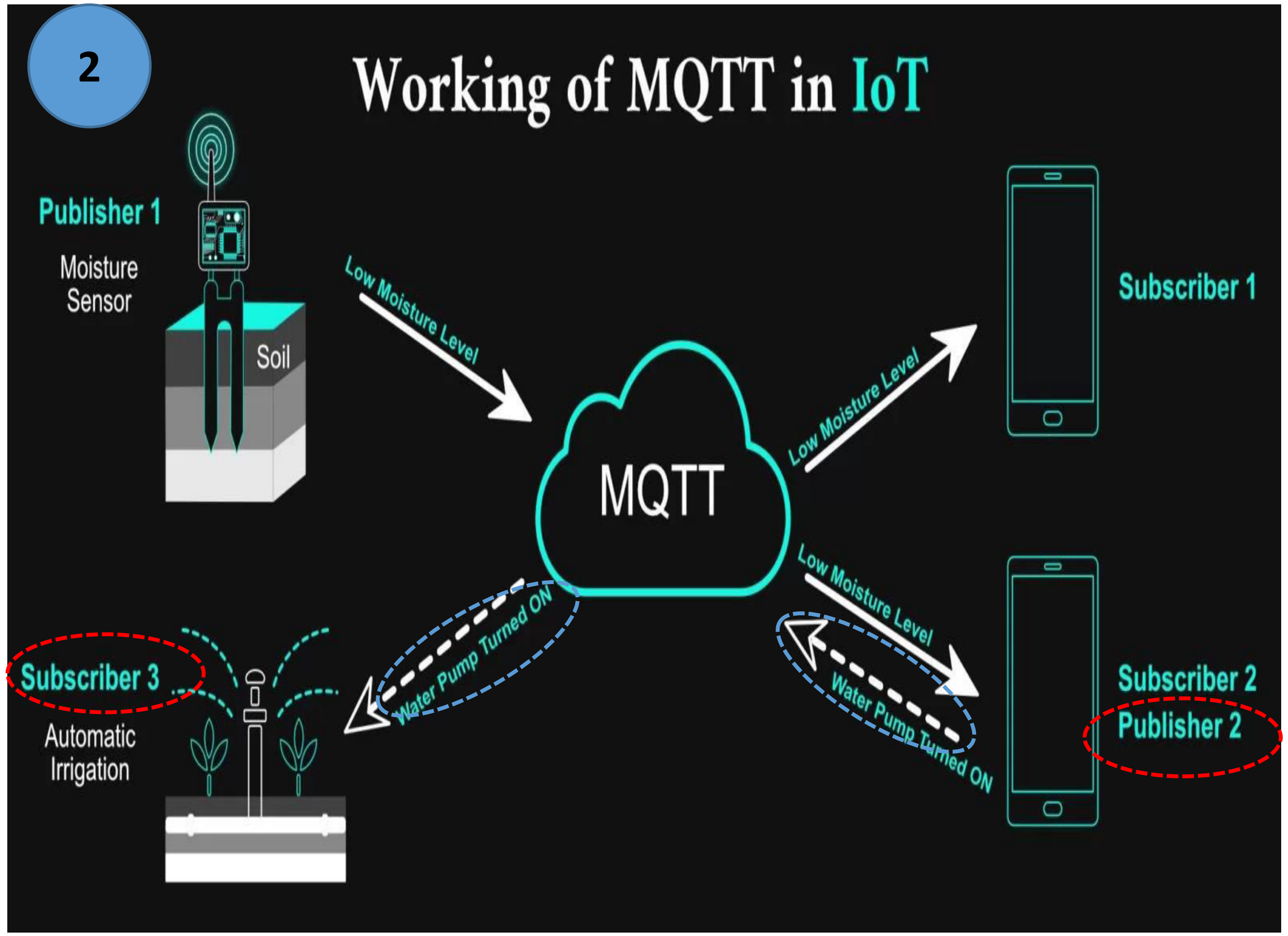
In the subscriber-publisher model, the one which sends data/ command is termed a publisher, and the one which receives data/ command is termed a subscriber.
The moisture sensor publishes moisture data and sends it to the broker, which when received by the subscriber can generate an alert based on the 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.



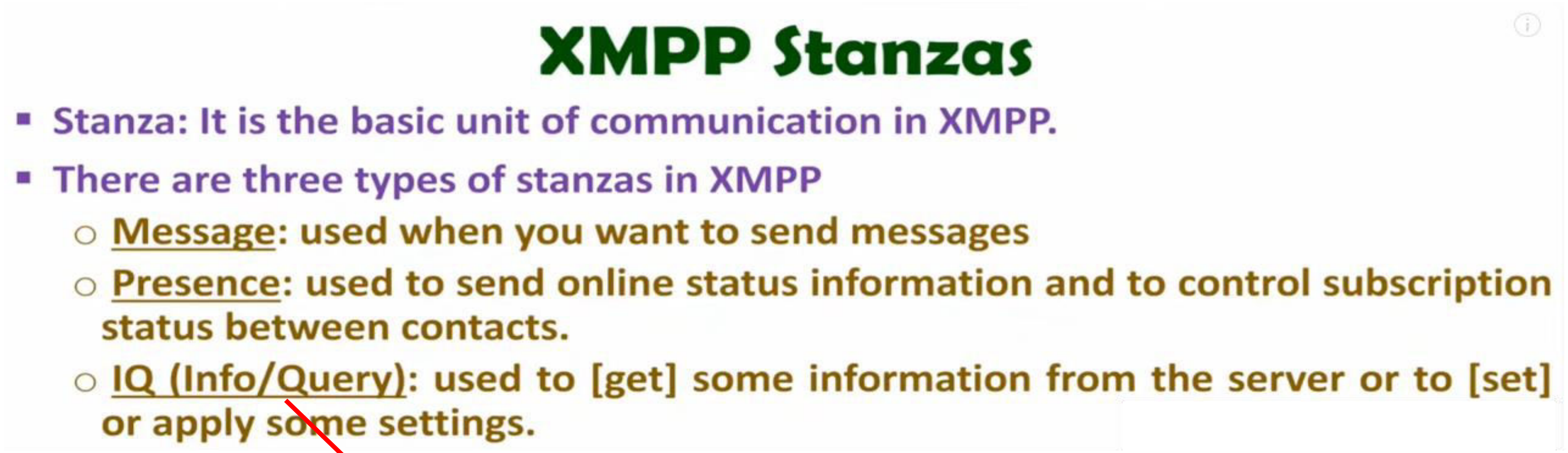
- XMPP – eXtensible Messaging and Presence Protocol

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.

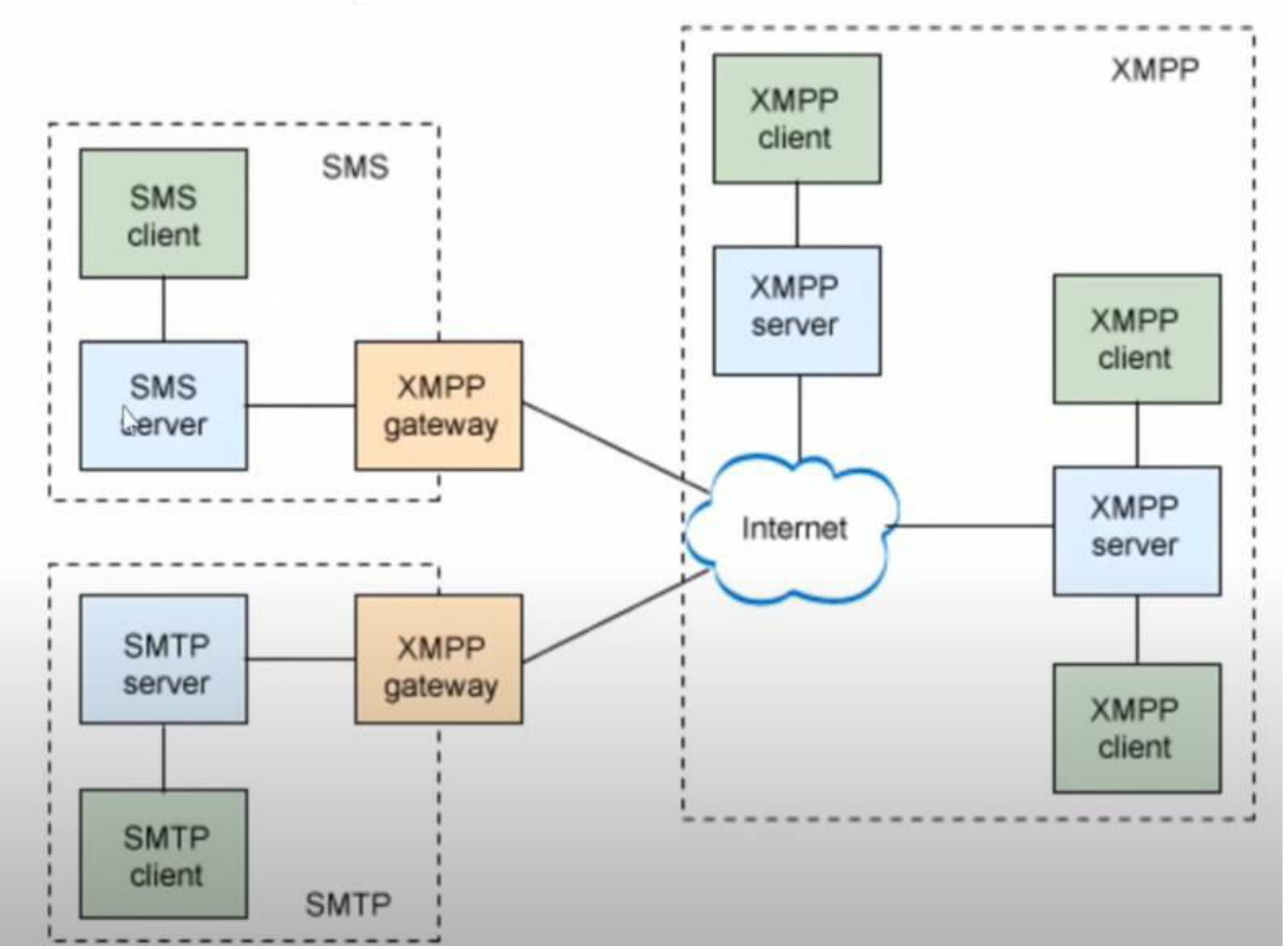


XMPP Stanzas

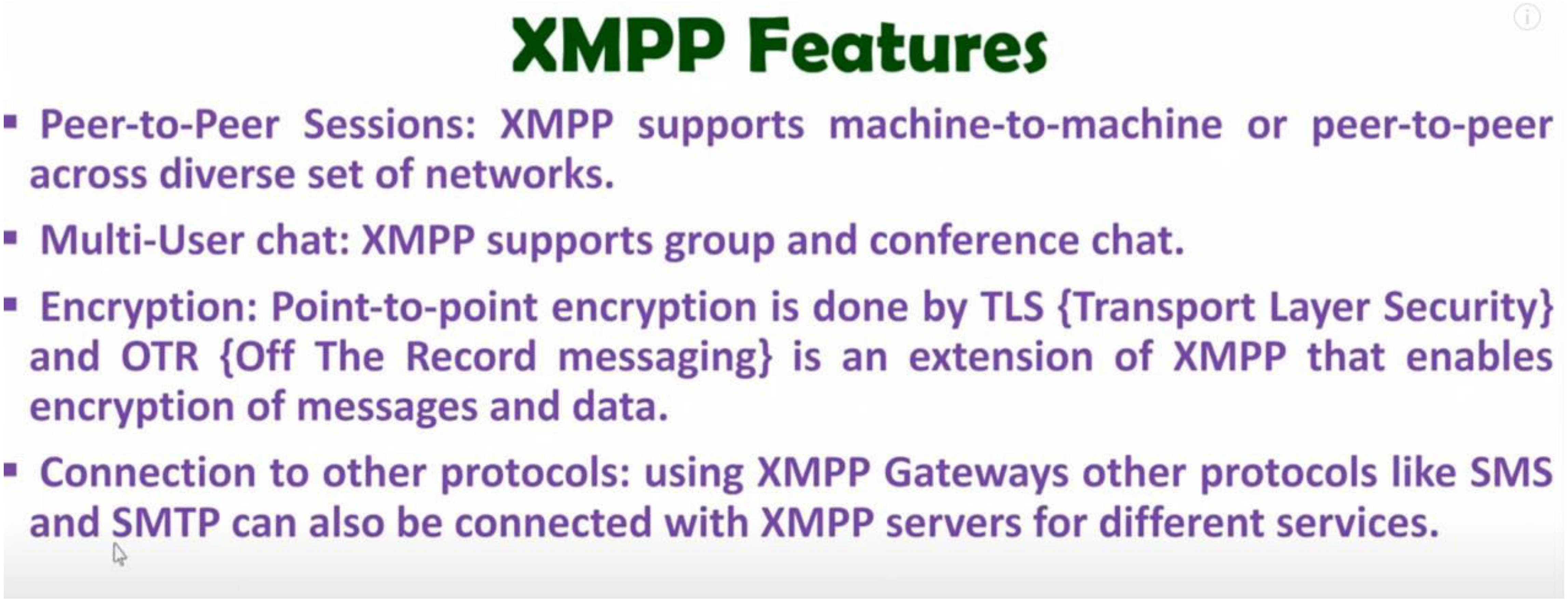
- Stanza: It is the basic unit of communication in XMPP.
- There are three types of stanzas in XMPP
  - Message: used when you want to send messages
  - Presence: used to send online status information and to control subscription status between contacts.
  - IQ (Info/Query): used to [get] some information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

Data Distribution Service (DDS)

➢ Data Distribution Service (DDS) is one of the protocol being used in the Internet of Things (IoT) applications.

➢ DDS is **brokerless service** and the protocol was developed by the **Object Management Group (OMG).**

➢ The purpose behind development of DDS is **to get better machine to machine communication**.

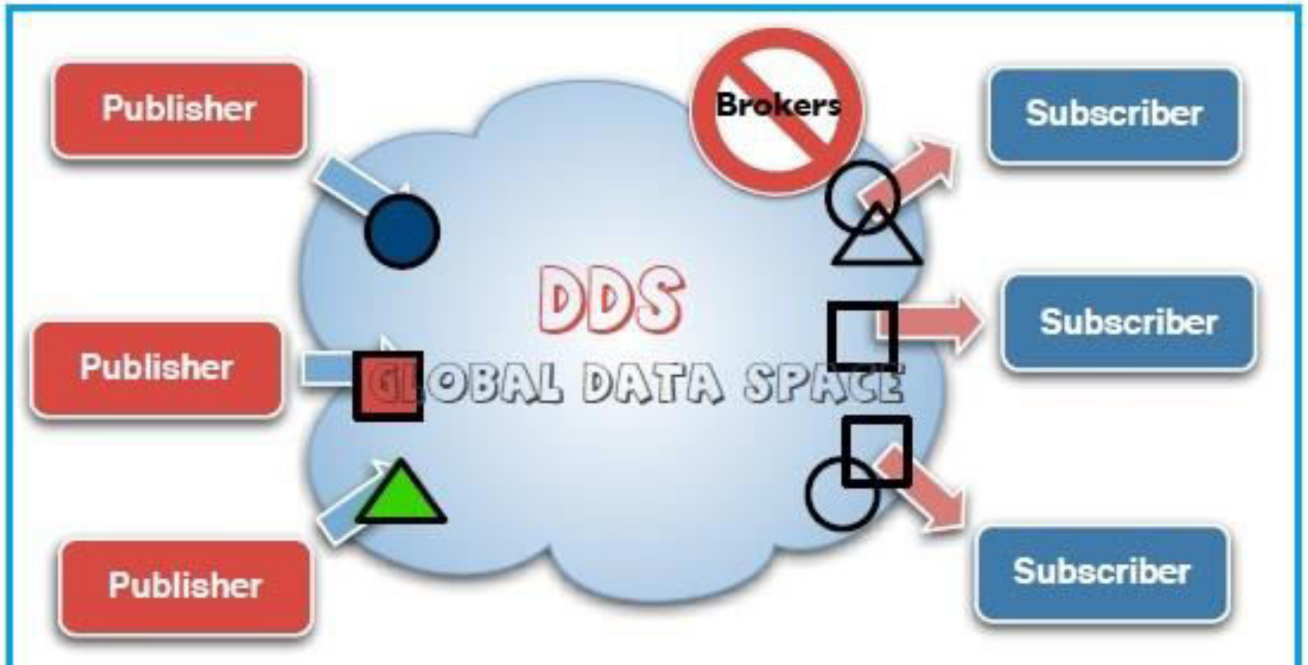
➢ Like MQTT, DDS is also using **publish-subscribe** approach.

DDS

OBJECT MANAGEMENT GROUP®

DDS Architecture

- DDS is basically a publish – subscribe model based architecture.
- Everything including sending and receiving data, event updates, command within and among the nodes are all accomplished through the publish – subscribe approach.
- The publisher sends the data to DSS cloud with particular topic.
- The interested nodes (Subscribers) subscribes the particular topic and receives data from DDS cloud.



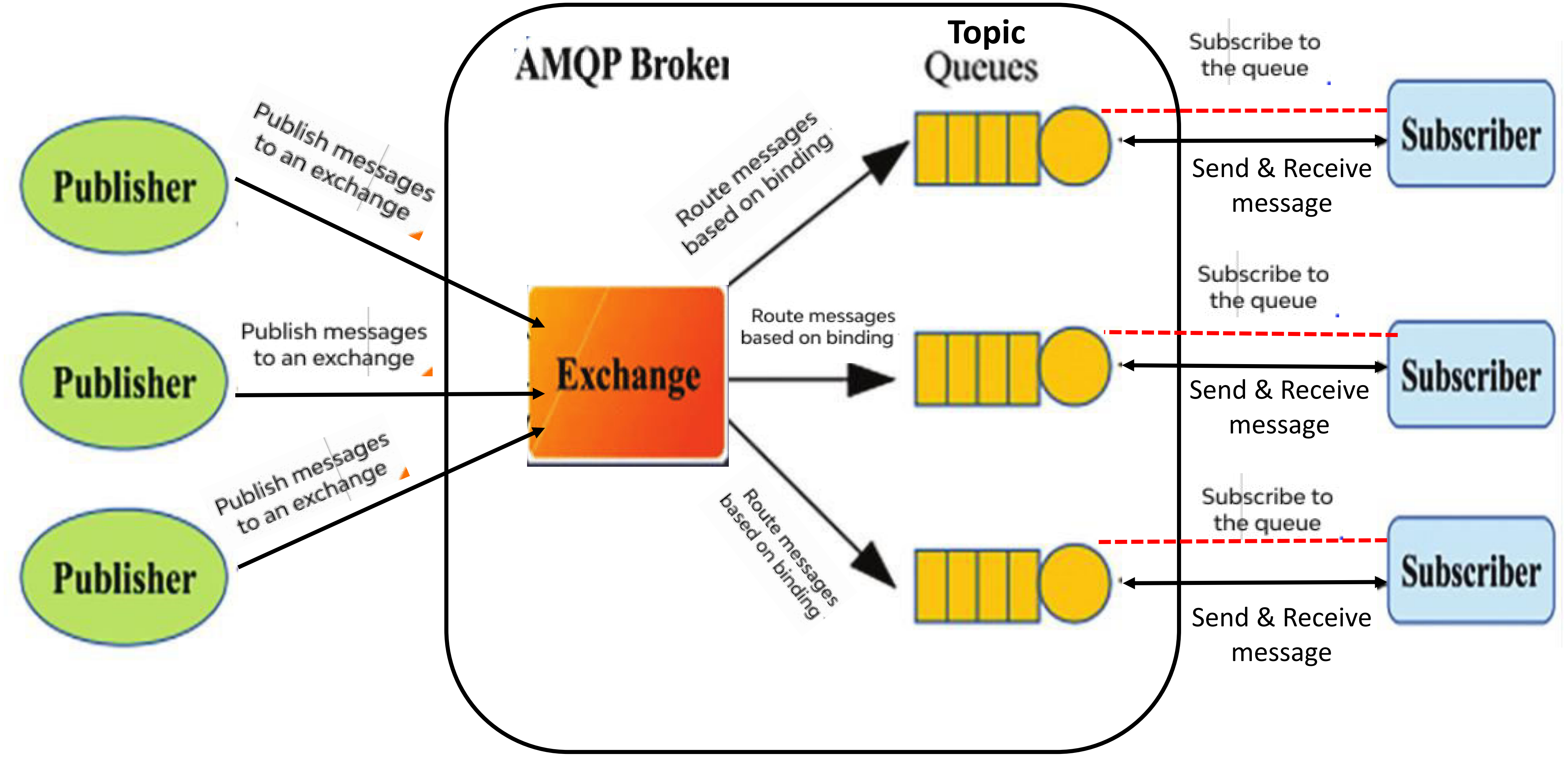
DDS Architecture

AMQP

Advanced Message Queuing Protocol

- Advanced Message Queuing Protocol is one which is used in **banking System**.
- To perform **banking transaction** like **NEFT, UPI transfer** whatever **messages** are **getting forwarded** for transaction, that can be done with the use of AMPQ protocol.
- AMQP is a peer-to-peer protocol.
- It is an open standard application layer protocol for message-oriented middleware.
- AMPQ is mainly used in transaction messages between servers and users.
- It follows Public – subscriber model for communication.

Architecture of AMPQ



Components of AMPQ Architecture

Publisher: Publisher is responsible to Publish or produce messages and that send to the AMPQ broker

AMPQ Broker : Broker (or server) plays a crucial role in AMQP protocol enablement. It is responsible for connection building that ensure better data routing and queuing at the client-side. The main job of AMPQ broker is to manage exchange and topic queues.

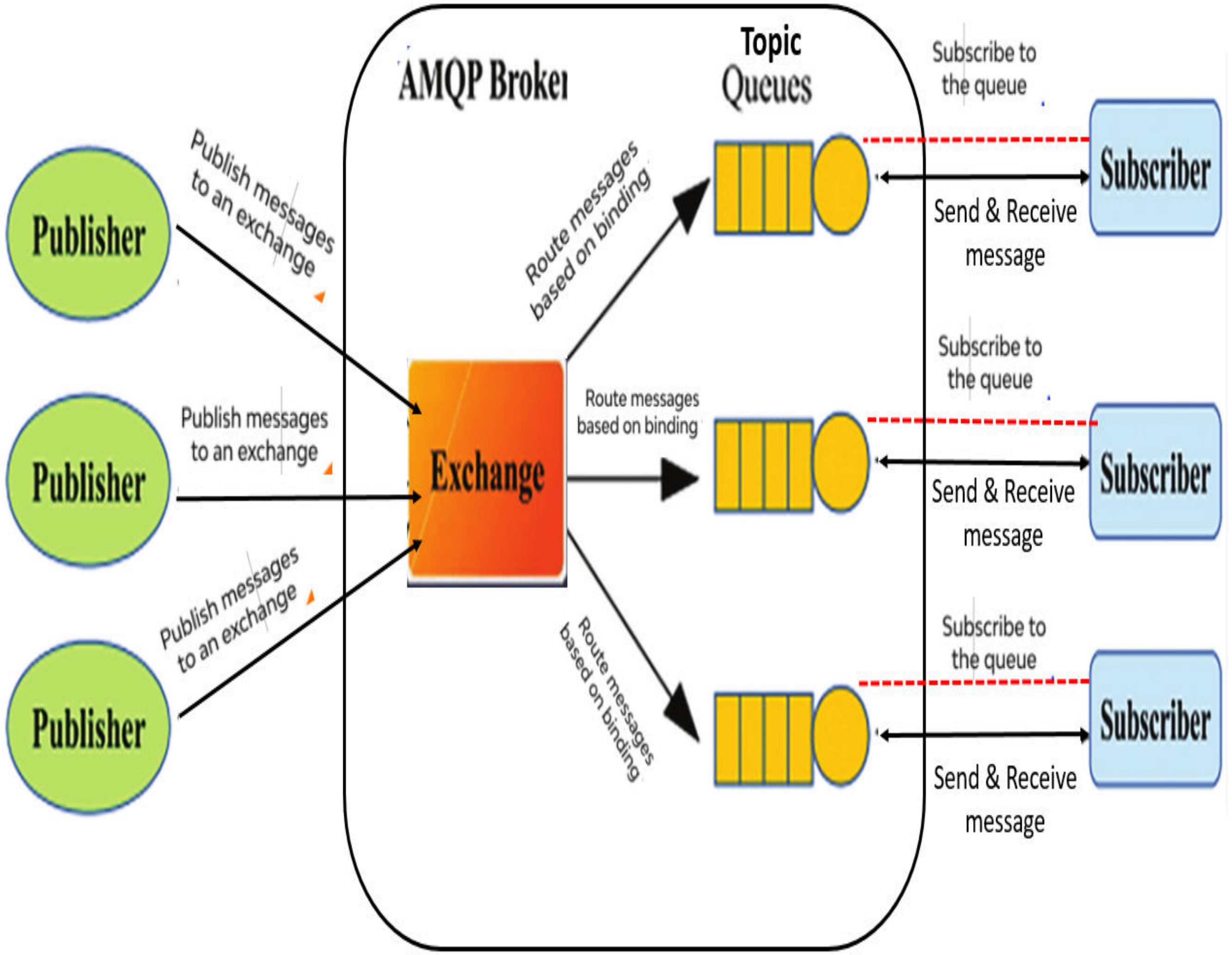
Exchange: Exchange handles the responsibility of fetching messages and placing them carefully in the right queue.
It will define how routing will happen.
There are four types of exchange :

1. Direct
2. Topic
3. Fanout
4. Headers

Topic Queue :It is an identified entity that helps link messages with their resources or point of origin.
Queue receives the message and keeps it until the subscriber use it.

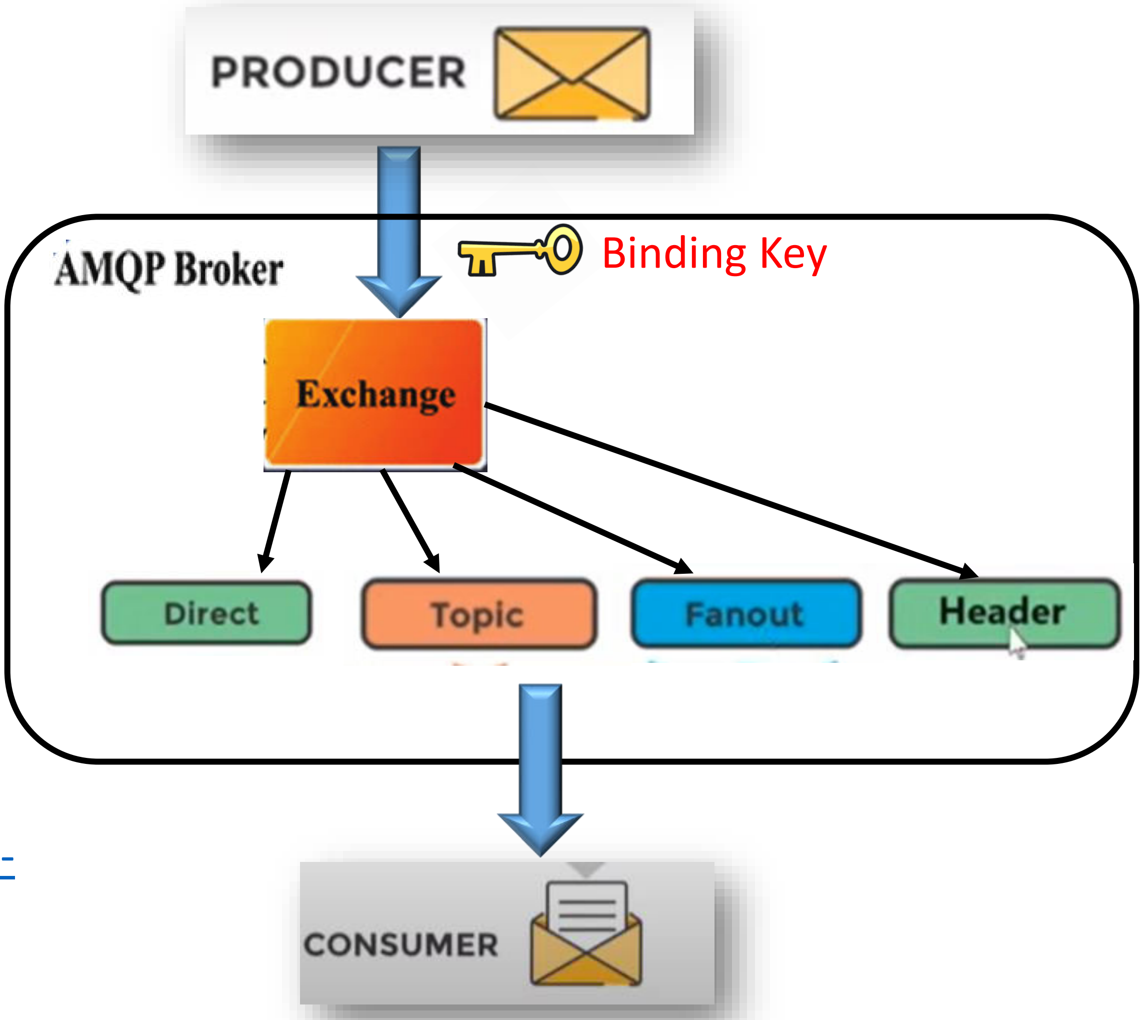
Subscriber : The job of queues generation and message acknowledgement is taken care of by consumer.

Architecture of AMPQ



Types of Exchange in AMPQ:

- AMPQ follows publish – subscribe model in which publisher is publishing messages or topics and subscribers are subscribing topic based on the subscription they will be receiving messages.
- Producers publishing the topics/messages and that goes to the AMPQ broker. And broker will be forwarding topic wise messages to subscriber.
- AMPQ broker is having exchange which will define how routing will be happen.
- The message or topic which is forwarded by producer is having **binding key**.
- Now based on binding key broker will define to which exchange that message should be forwarded.



There are four types of exchange :

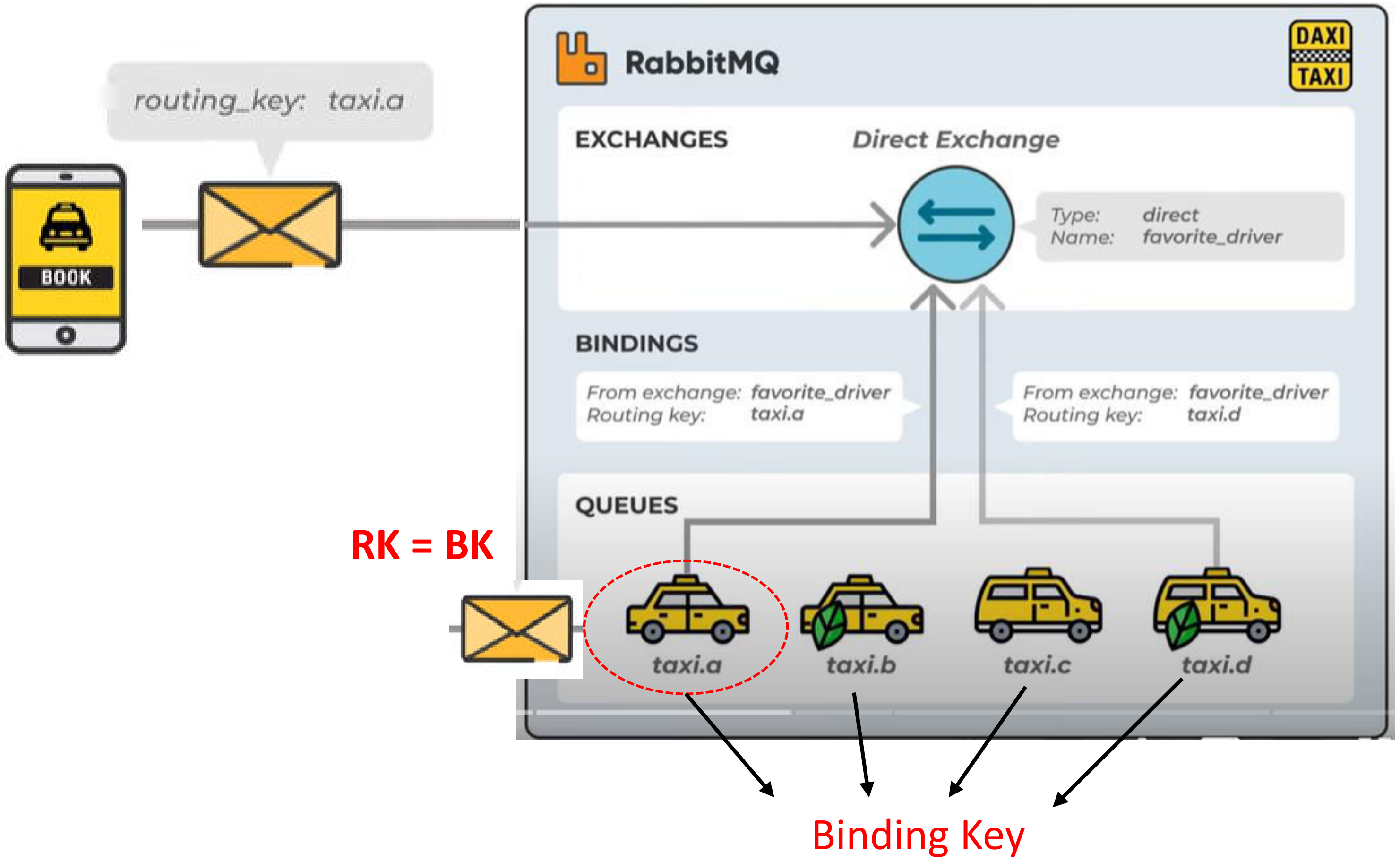
1. Direct
2. Topic
3. Fanout
4. Headers

Types of exchange :

- https://www.cloudamqp.com/blog/part4-rabbitmq-for-beginners-exchanges-routing-keys-bindings.html
- https://www.youtube.com/watch?v=o8eU5WiO8fw&t=176s

Direct Exchange

- The routing in direct exchange is simple as a message goes to the queues whose binding key exactly matches the routing key of the message.
- A direct exchange delivers messages to queues based on a message routing key. The routing key is a message attribute added to the message header by the producer. Think of the routing key as an "address" that the exchange is using to decide how to route the message. **A message goes to the queue(s) with the binding key that exactly matches the routing key of the message.** (**Message Routing Key (RK) = Binding key of Queue (BK)**)
- The direct exchange type is useful to distinguish messages published to the same exchange using a simple string identifier.
- The default exchange AMQP brokers must provide for the direct exchange is "amq.direct".



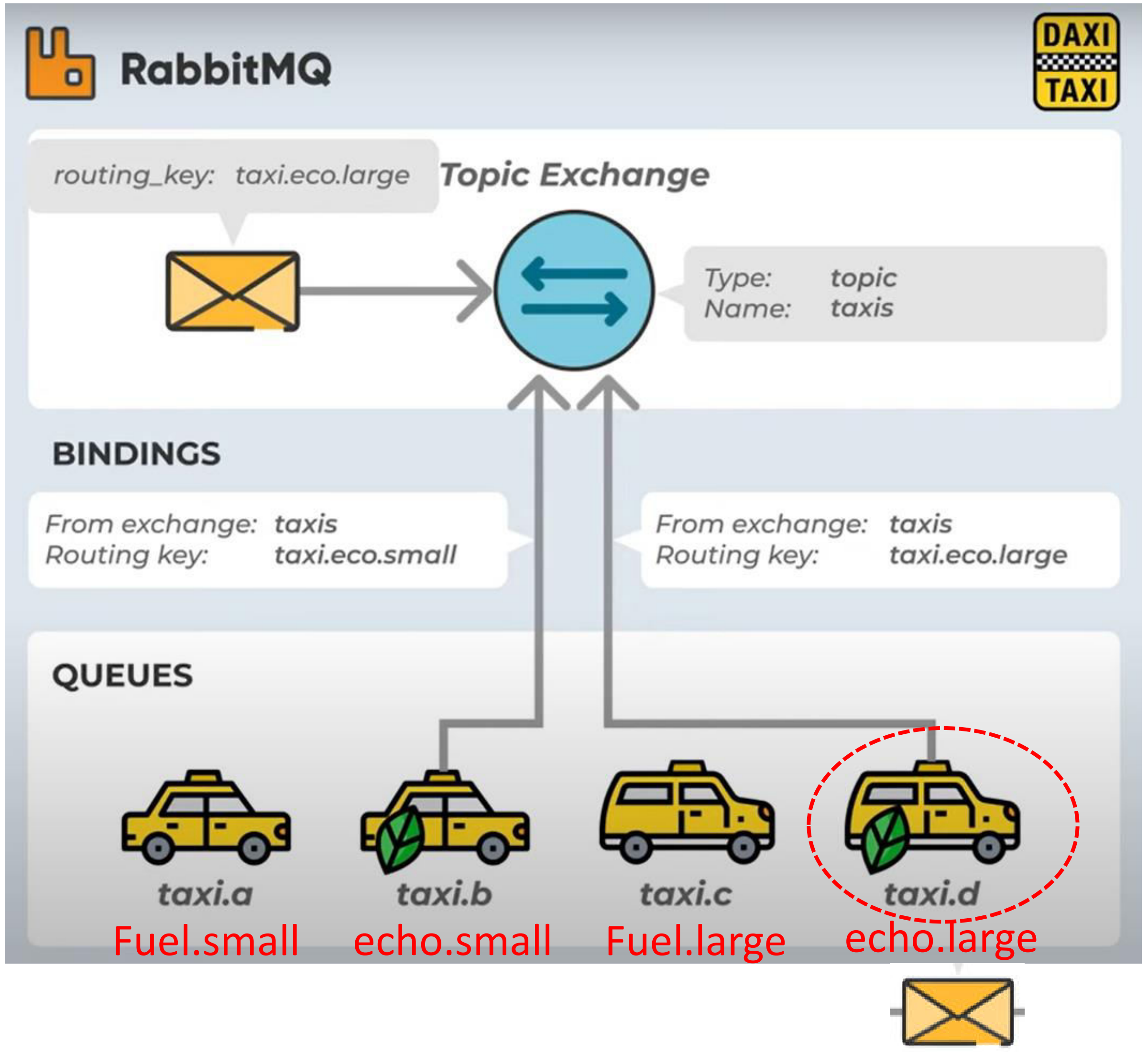
Topic Exchange

Example 1 : Message is routed to only one queue based on match

- Topic exchanges route messages to queues based on wildcard matches between the routing key and the routing pattern, which is specified by the queue binding.

- Messages are routed to one or many queues based on a matching between a message routing key and this pattern.

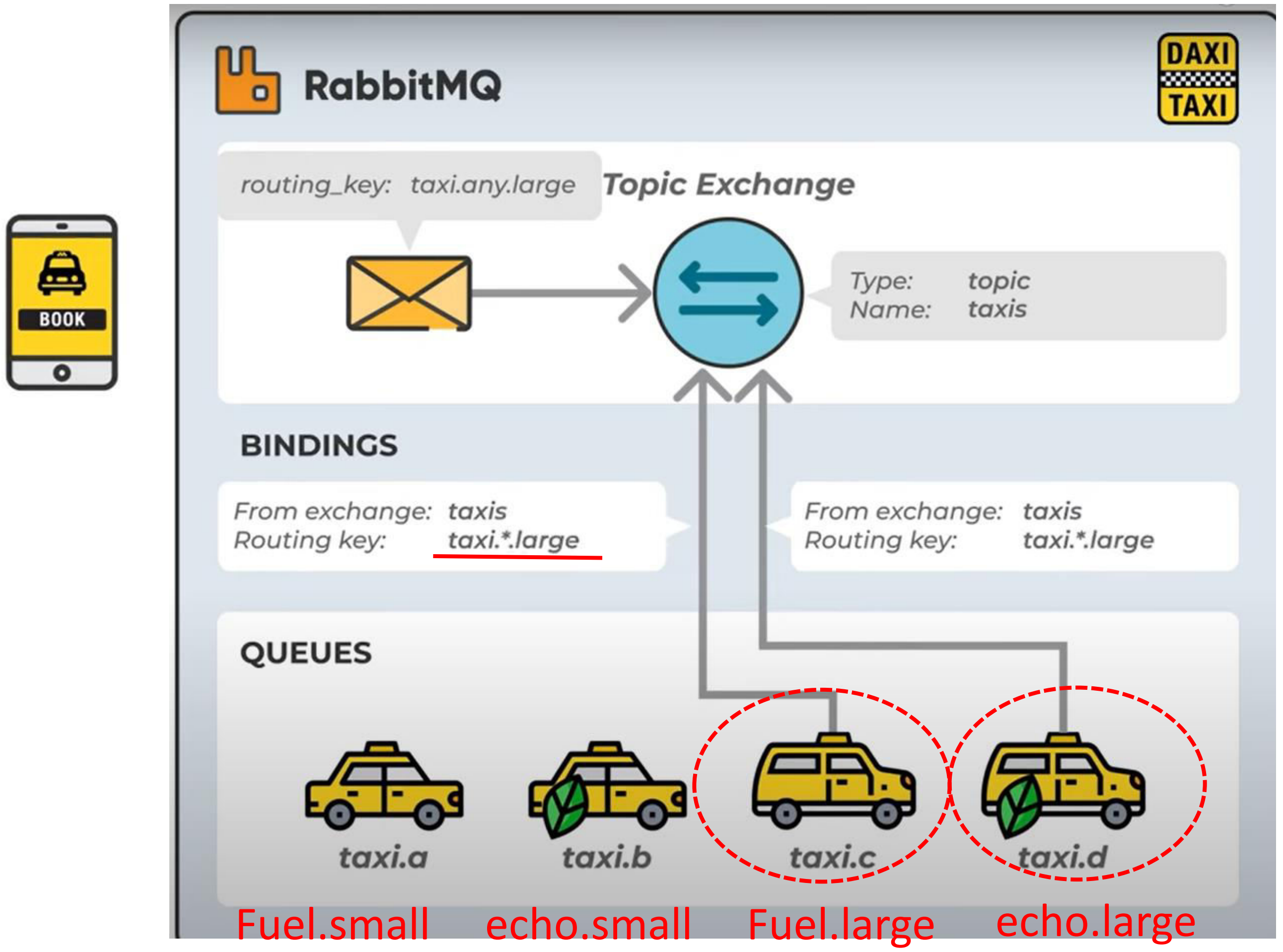
- The routing key must be a list of words, delimited by a period (.)



Topic Exchange

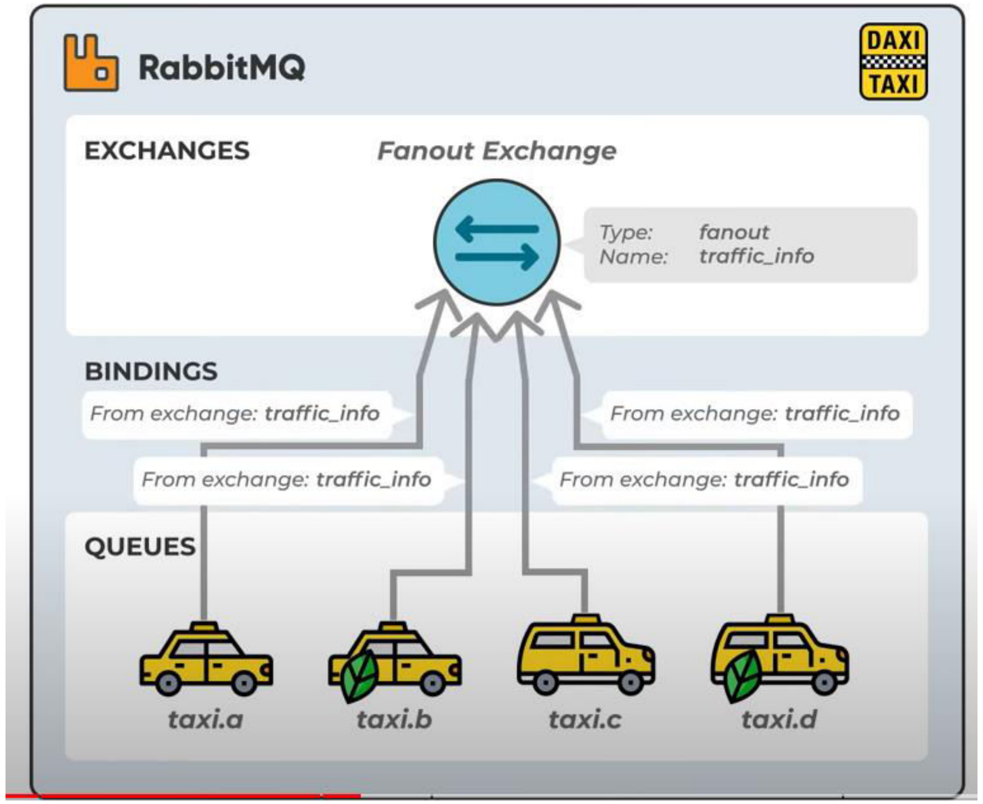
Example 2 : Message is routed to more than one queue based on match

The topic exchange supports "strict routing key matching" like a direct exchange but it will also perform wildcard matching using star (*) and hash (#) as placeholders.



Fanout Exchange

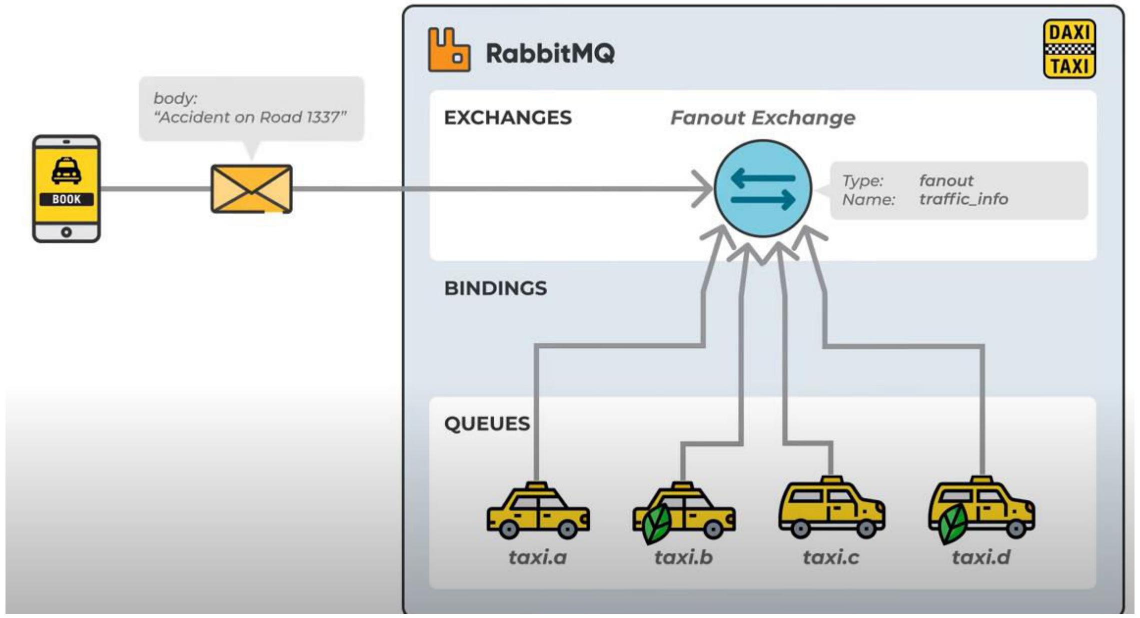
- Fanout exchange copies and routes received messages to all queues that are bound to it regardless of Routing keys.
- A provided routing key is simply ignored.
- Fanout exchanges can be useful when the same message needs to be sent to one or more queues with consumers who may process the same message in different ways.
- The default exchange AMQP brokers must provide for the topic exchange is "amq.fanout".



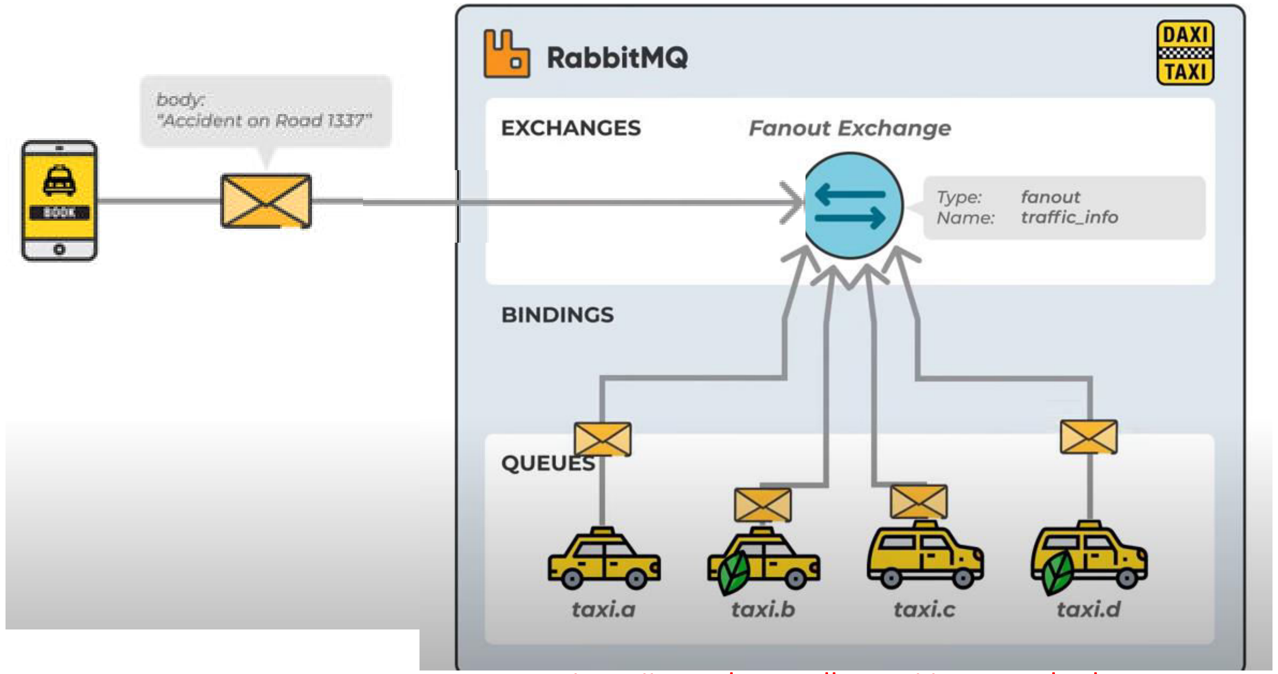
Fuel.small echo.small Fuel.large echo.large

Fanout Exchange

Example



Fuel.small echo.small Fuel.large echo.large



Fuel.small echo.small Fuel.large echo.large

Headers Exchange

A headers exchange routes messages based on arguments containing headers and optional values. Headers exchanges are very similar to topic exchanges, but route messages based on header values instead of routing keys. A message matches if the value of the header equals the value specified upon binding.

Scenario 1

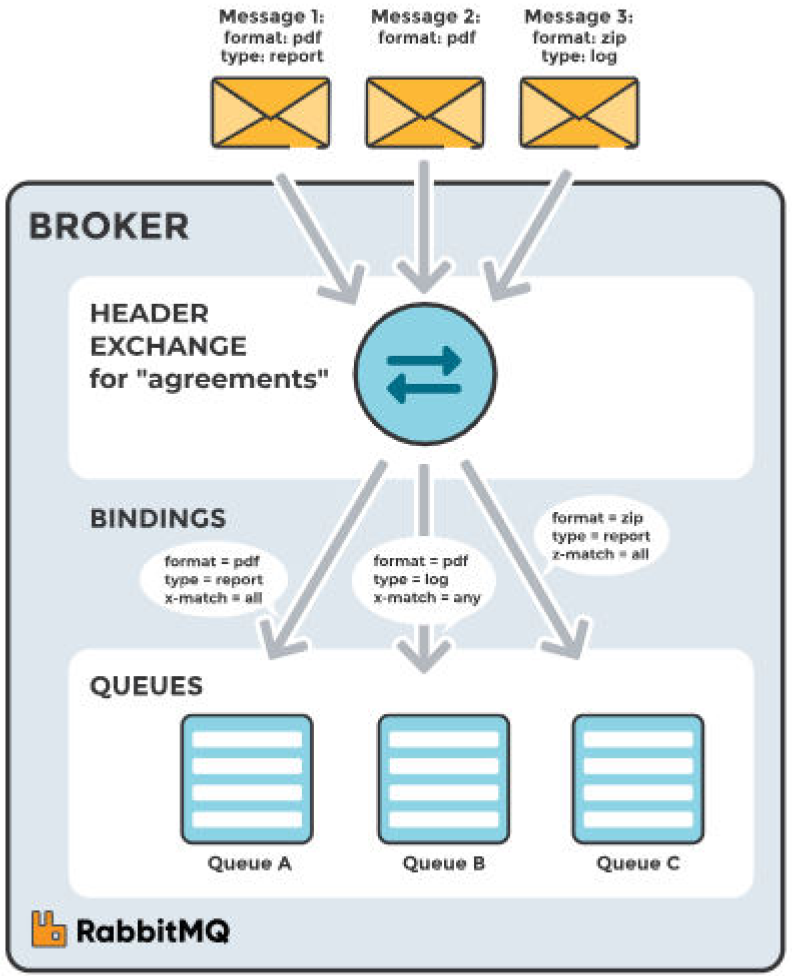
Message 1 is published to the exchange with header arguments (key = value): "format = pdf", "type = report".
Message 1 is delivered to Queue A because all key/value pairs match, and Queue B since "format = pdf" is a match (binding rule set to "x-match =any").

SCENARIO 2

Message 2 is published to the exchange with header arguments of (key = value): "format = pdf".
Message 2 is only delivered to Queue B. Because the binding of Queue A requires both "format = pdf" and "type = report" while Queue B is configured to match any key-value pair (x-match = any) as long as either "format = pdf" or "type = log" is present.

SCENARIO 3

Message 3 is published to the exchange with header arguments of (key = value): "format = zip", "type = log".
Message 3 is delivered to Queue B since its binding indicates that it accepts messages with the key-value pair "type = log", it doesn't mind that "format = zip" since "x-match = any".

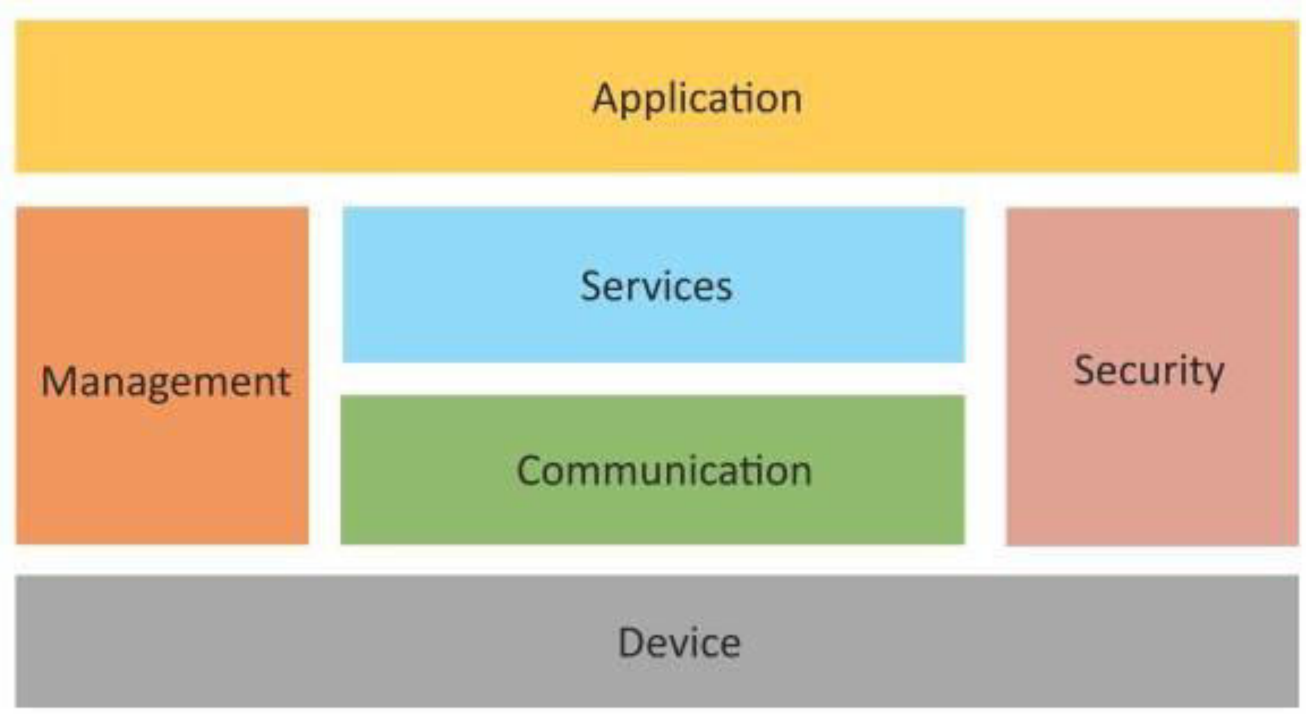


Logical Design of IoT

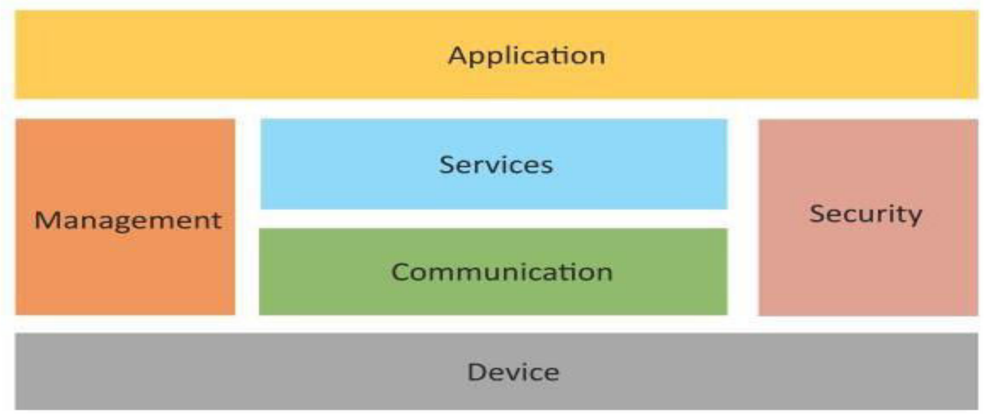
For understanding Logical Design of IoT, we describes given below terms.

IoT Functional Blocks

- Logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation.
- An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication, and management.



IoT Functional Blocks



Device: An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions.

Communication: Handles the communication for the IoT system.

Services: services for device monitoring, device control service, data publishing services and services for device discovery.

Management: this blocks provides various functions to govern the IoT system.

Security: this block secures the IoT system and by providing functions such as authentication , authorization, message and content integrity, and data security.

Application: This is an interface that the users can use to control and monitor various aspects of the IoT system. Application also allow users to view the system status and view or analyze the processed data.

What is Stateless and Stateful Protocols?

Stateless Protocol

- It is a network model in which the client sends a request to the server and the server in return sends a response back according to the current state just like the Request-Response model.
- The server is not obliged to keep the session information or the status of each communication partner for multiple requests.
- They are very easy to implement on the Internet.
- Stateless protocols work better when the crash occurs because no state needs to be restored, a failed server can simply reboot after a crash.
- Examples:- HTTP (Hypertext Transfer Protocol), UDP (User Datagram Protocol), DNS (Domain Name System).

Stateful Protocol

- In this protocol, suppose a client sends a request to the server and the server doesn't respond, then the client resends a request to the server.
- Stateful protocols are logically heavy to implement on the Internet.
- Stateful Protocol does not work better at the time of the crash because stateful servers must retain information about the state and session details of internal states.
- Examples:- FTP (File Transfer Protocol), Telnet.

IoT Communication APIs are:

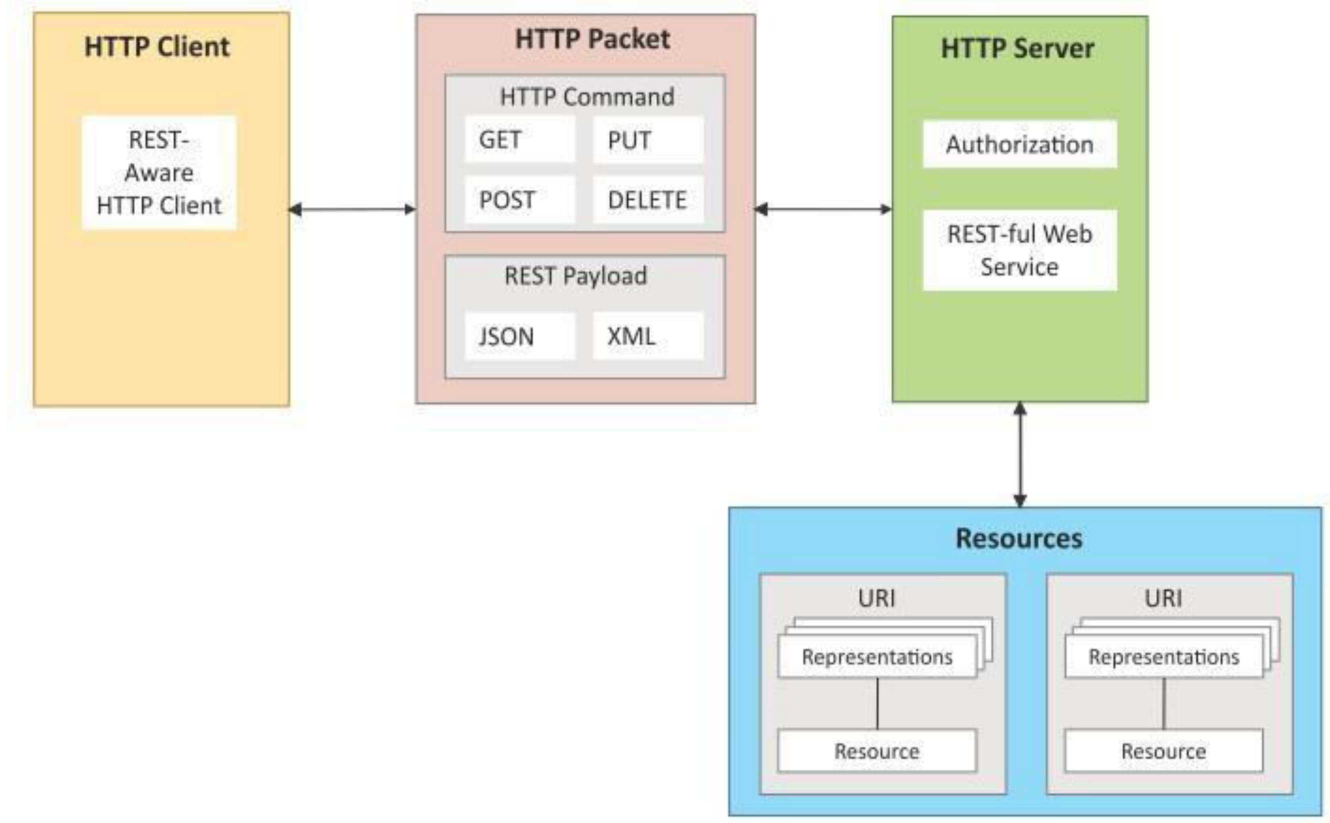
IoT APIs are the points of interaction between an IoT device and the internet and/or other elements within the network.

1) REST-based Communication APIs

2) WebSocket-based Communication APIs

IoT Communication APIs are:

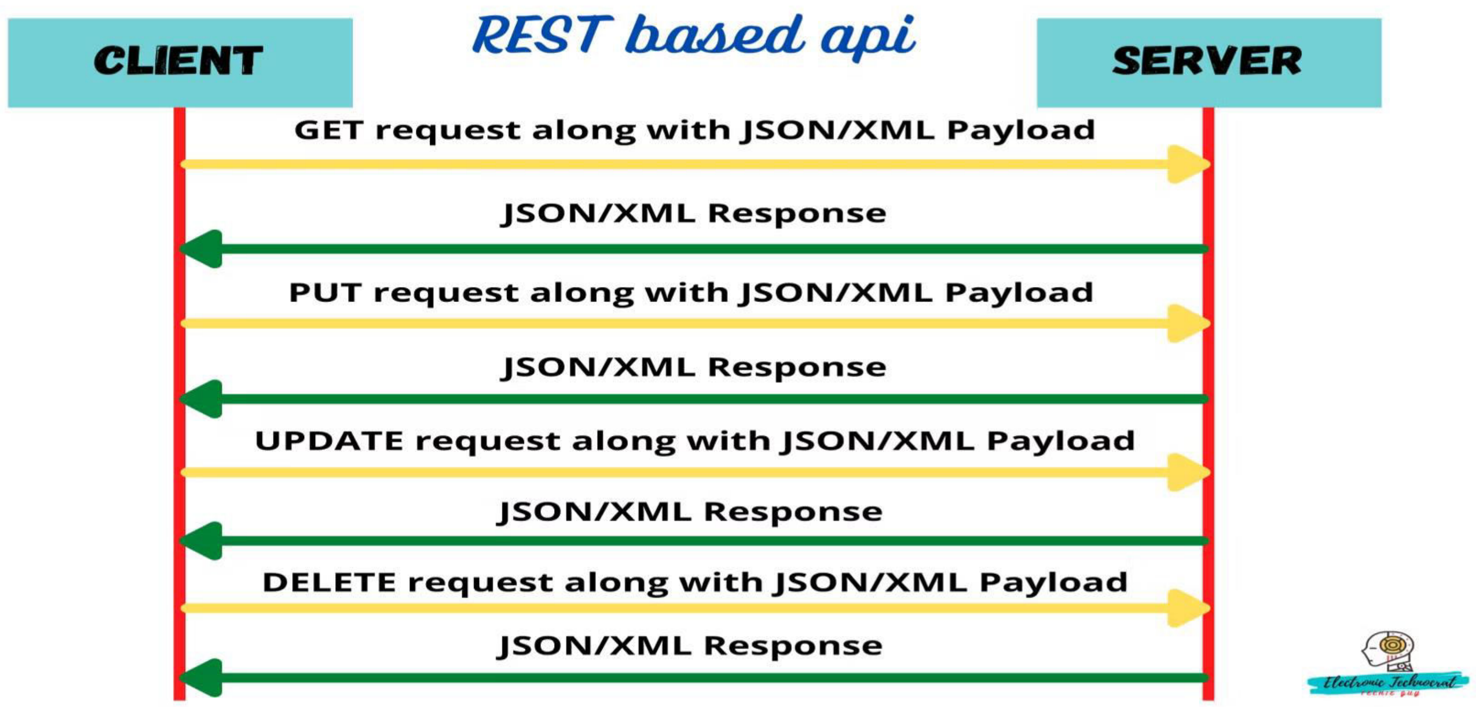
1) REST-based Communication APIs 2) WebSocket-based Communication APIs



- Representational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred.

- REST APIs follow the request- response communication model.

- The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system.


REST based api
CLIENT
SERVER
GET request along with JSON/XML Payload
JSON/XML Response
PUT request along with JSON/XML Payload
JSON/XML Response
UPDATE request along with JSON/XML Payload
JSON/XML Response
DELETE request along with JSON/XML Payload
JSON/XML Response
Electronic Technocrat

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

The rest architectural constraints are as follows:

- Client-server

Let me explain it to you by giving a suitable example. The client should not be concerned with the storage of data which is a concern of the server, similarly, the server should not be concerned about the user interface, which is the concern of the client. Separation makes it possible for the client and server to be developed and updated independently.

- Stateless

The status of the session remains entirely on the client.

- Cache-able

This property defines whether the response to any request can be cached or not. If a response can be cached, then a client cache is granted the right to reuse that response data for subsequent matching requests.

- Layered system

A layered system defines the boundaries of the components within each specific layer. For example, A client is unable to tell whether it is connected to the end server or an intermediate node. As simple as that!

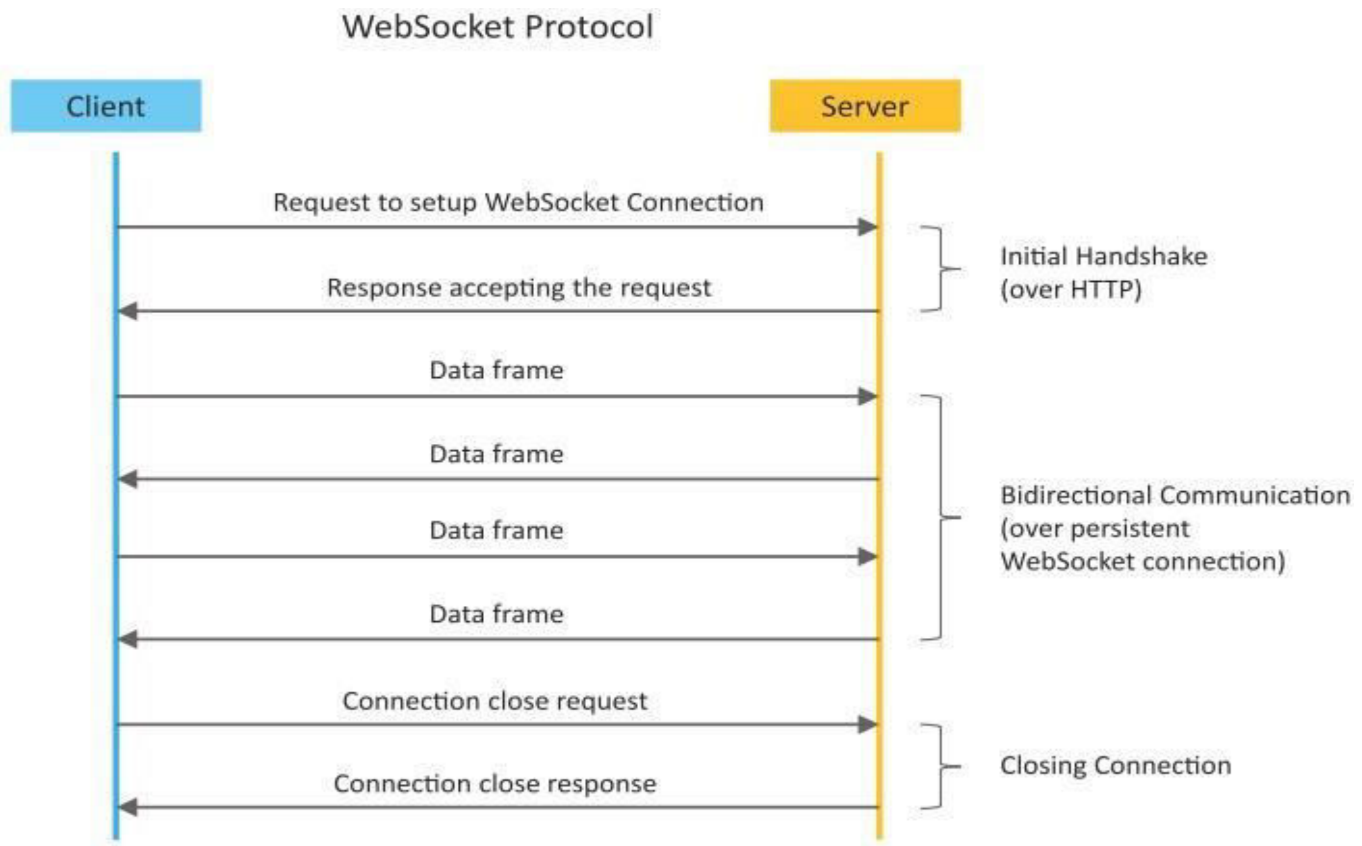
- Uniform interface

This specifies that the technique of communication between a client and a server must be uniform throughout the communication period.

- Code on Demand(Optional Constraint)

Servers may provide executable code or scripts for execution by clients in their context.

WebSocket-based Communication APIs


WebSocket Protocol
Client
Server
Request to setup WebSocket Connection
Response accepting the request
Initial Handshake (over HTTP)
Data frame
Data frame
Data frame
Data frame
Bidirectional Communication (over persistent WebSocket connection)
Connection close request
Connection close response
Closing Connection

- Websocket APIs allow bi-directional, full duplex communication between clients and servers. Websocket APIs follow the exclusive pair communication model.

- Unlike request-response model such as REST, the WebSocket APIs allow full duplex communication and do not require new conection to be setup for each message to be sent.

- Websocket APIs enable bi-directional and duplex communication between customers and servers.
- Unlike REST, There is no need to set up a connection every now and then to send messages between a client and a server.
- It works on the principle of the exclusive pair model. Can you recall it? Yes. Once a connection is set up, there is a constant exchange of messages between the client and the server. All we need is to establish a dedicated connection to start the process. the communication goes on unless the connection is terminated.
- It is a stateful type.
- Due to one time dedicated connection setup, there is less overhead, lower traffic and less latency and high throughput.

IoT Hardware

- IoT Hardware includes a wide range of devices such as **devices for routing, bridges, sensors etc**. These IoT devices manage key tasks and functions such as system **activation, security, action specifications, communication, and detection of support-specific goals and actions.**

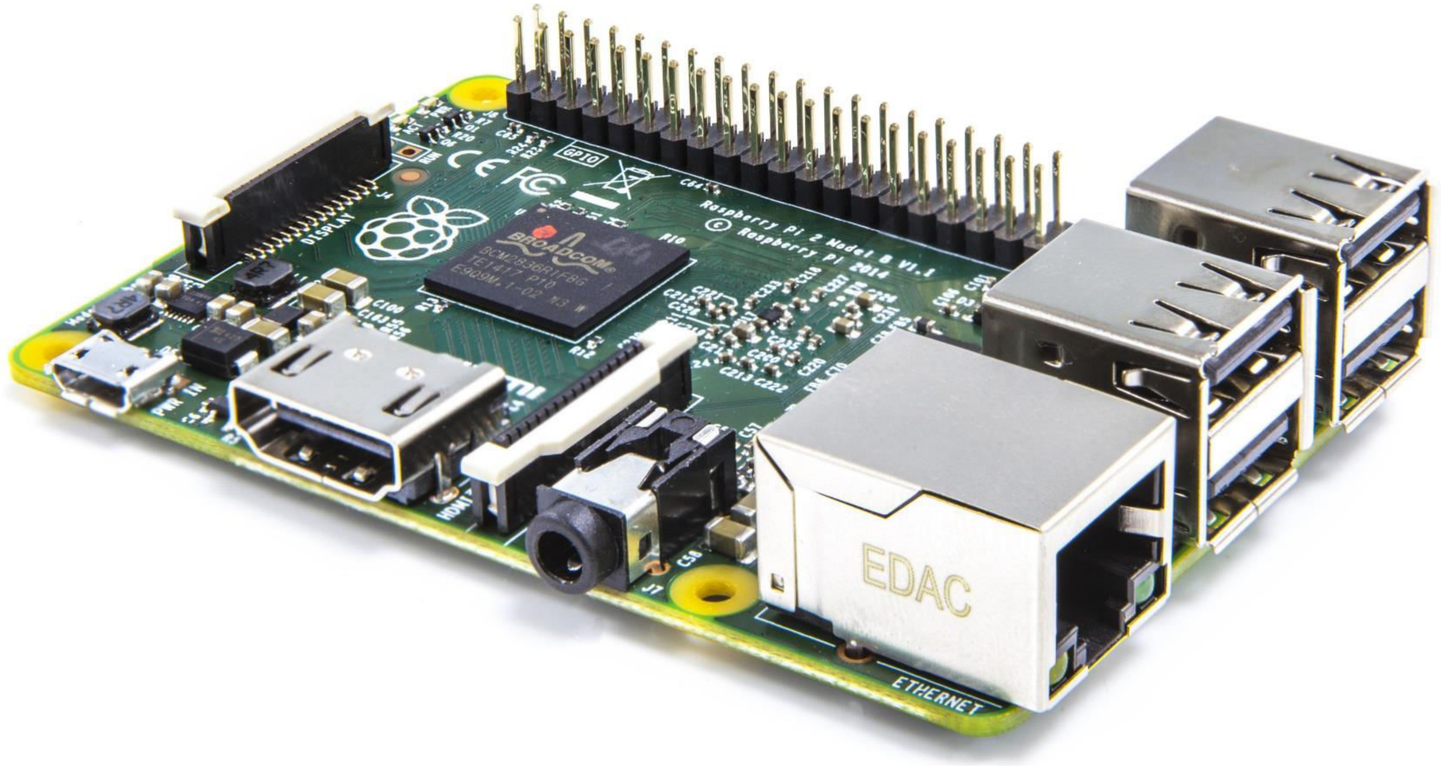
- IoT Hardware components can vary from low-power boards; single-board processors like the Arduino Uno which are basically smaller boards that are plugged into mainboards to improve and increase its functionality by bringing out specific functions or features (such as GPS, light and heat sensors, or interactive displays).

- A programmer specifies a board’s input and output, then creates a circuit design to illustrate the interaction of these inputs and outputs

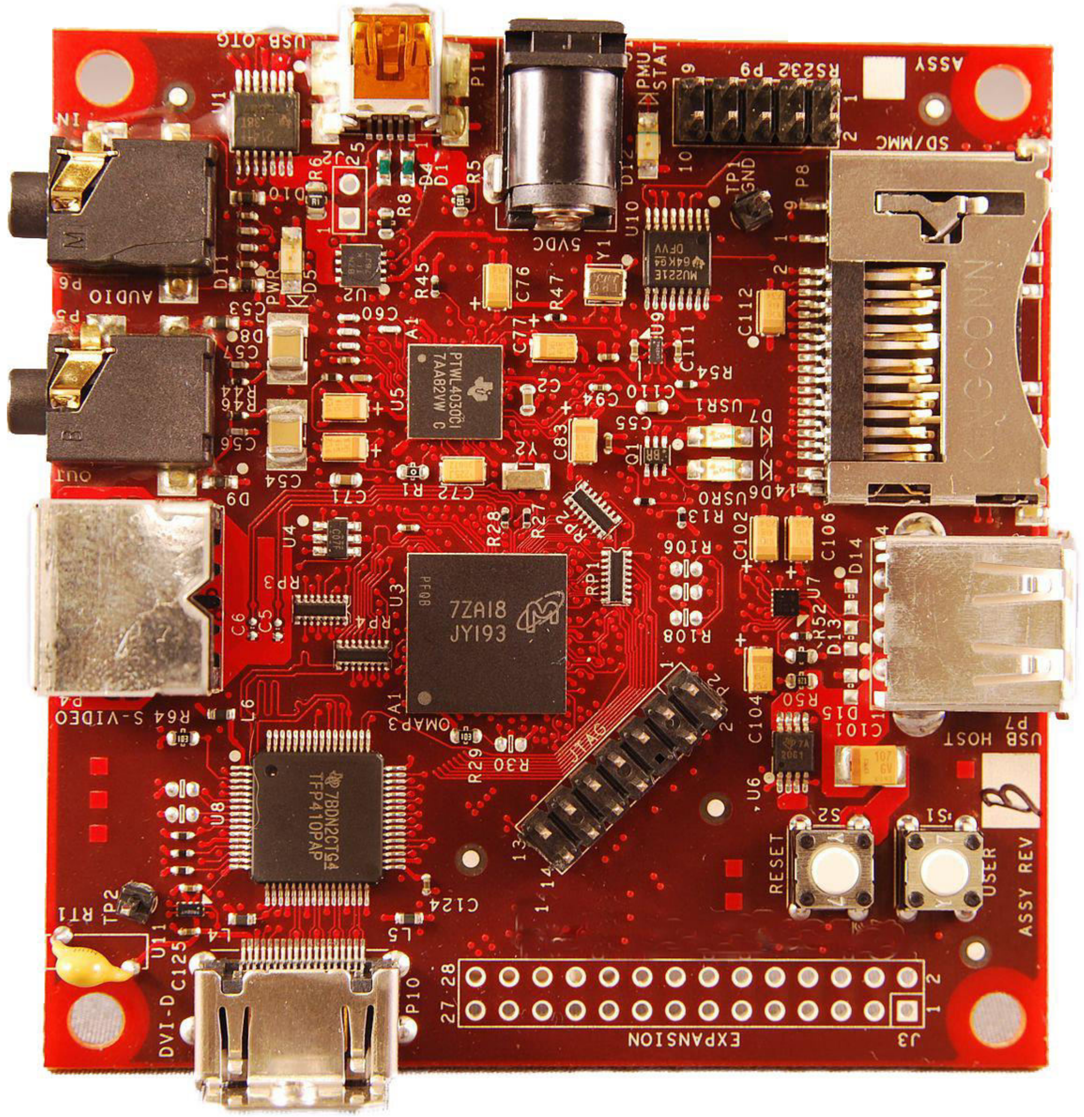
IoT Hardware – Arduino Uno

- Another well-known IoT platform is **Raspberry Pi** , which is a very affordable and tiny computer that can incorporate an entire web server. Often called “RasPi,” it has enough processing power and memory to run Windows 10 on it as well as IoT Core.

- RasPi exhibits great processing capabilities, especially when using the Python programming language.

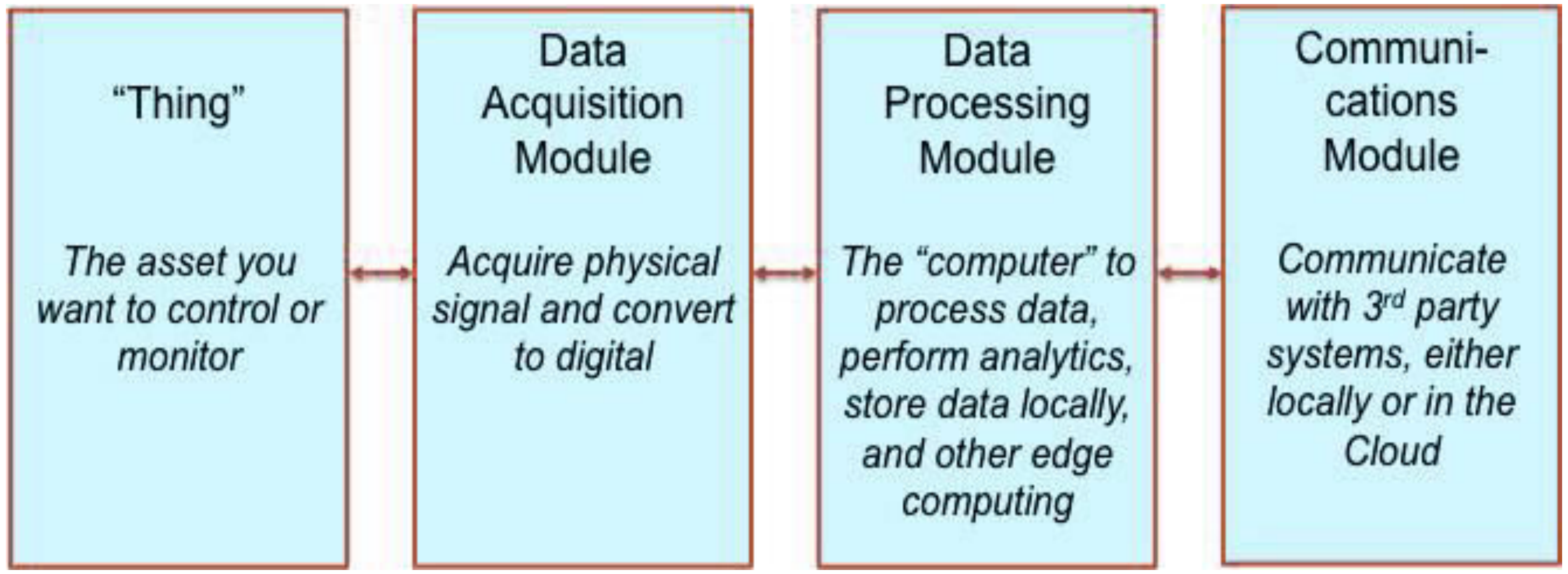

IoT Hardware – Raspberry Pi 2

- **BeagleBoard** is a single-board computer with a Linux-based OS that uses an ARM processor, capable of more powerful processing than RasPi.



IoT Hardware – BeagleBoard

a) Building Blocks of IoT Hardware


"Thing"
The asset you want to control or monitor
Data Acquisition Module
Acquire physical signal and convert to digital
Data Processing Module
The "computer" to process data, perform analytics, store data locally, and other edge computing
Communi-cations Module
Communicate with 3rd party systems, either locally or in the Cloud

1) Thing

"Thing" in IOT is the asset that you want to control or monitor or measure, that is, observe closely. In many IoT products, the "thing" gets fully incorporated into a smart device. For example, think of products like a smart refrigerator or an automatic vehicle. These products control and monitor themselves.

2. Data Acquisition Module

- The data acquisition module focuses on acquiring physical signals from the thing which is being observed or monitored and converting them into digital signals that can be manipulated or interpreted by a computer.

- This is the hardware component of an IOT system that contains all the sensors that help in acquiring real-world signals such as temperature, pressure, density, motion, light, vibration, etc. The type and number of sensors you need depend on your application.

- This module also includes the necessary hardware to convert the incoming sensor signal into digital information for the computer to use it. This includes conditioning of incoming signal, removing noise, analog-to-digital conversion, interpretation, and scaling.

3. Data Processing Module

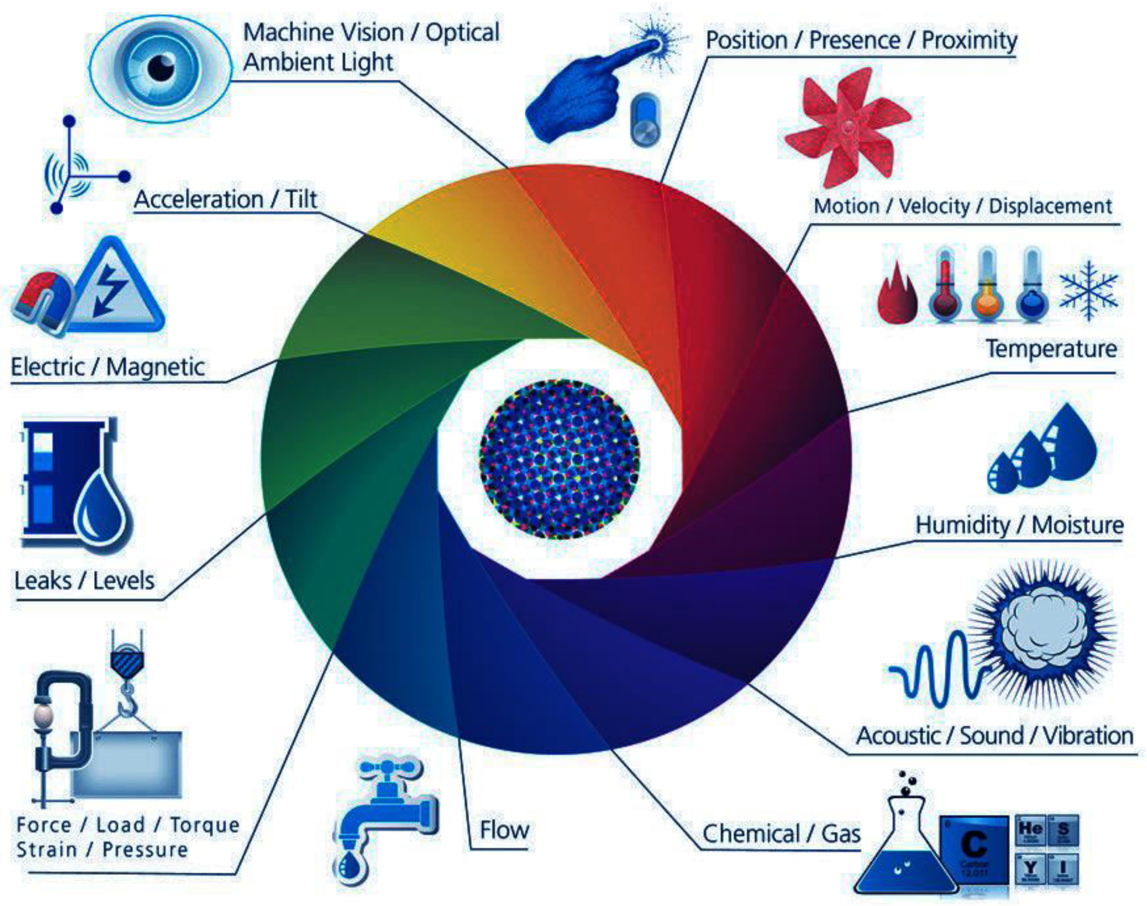
- The third building block of the IoT device is the data processing module. This is the actual "computer" and the main unit that processes the data performs operations such as local analytics, stores data locally, and performs some other computing operations.

4. Communication Module

- The last building block of IOT hardware is the communications module. This is the part that enables communications with your Cloud Platform, and with 3rd party systems either locally or in the Cloud.

b. IoT Sensors:

The most important IoT hardware might be its sensors. These devices consist of a variety of modules such as energy modules, RF modules, power management modules, and sensing modules.



c. Wearable Electronic Devices:

Wearable electronic devices are small devices that can be worn on the head, neck, arms, torso, and feet.

Current smart wearable devices include −

•**Head** − Helmets, glasses,

•**Neck** − Jewelry, collars

•**Arm** − Wristwatches, wristbands, rings

•**Torso** – Clothing pieces, backpacks

•**Feet** − Shoes, Socks

d. Basic Devices

The day to day devices that we use such as desktop, cellphones, and tablets remain integral parts of IoT system.
•The desktop provides the user with a very high level of control over the system and its settings.
•The tablet acts as a remote and provides access to the key features of the system.
•Cellphone allows remote functionality and some essential settings modification
Other key connected devices include standard network devices like routers and switches.

System:

A system is an arrangement in which all its unit assemble work together according to a set of rules. It can also be defined as a way of working, organizing or doing one or many tasks according to a fixed plan. For example, a watch is a time displaying system. Its components follow a set of rules to show time. If one of its parts fails, the watch will stop working. So we can say, in a system, all its subcomponents depend on each other.

Embedded System:

Embedded means something that is attached to another thing. An embedded system can be thought of as a computer hardware system having software embedded in it. An embedded system can be an independent system or it can be a part of a large system. An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task. For example, a fire alarm is an embedded system; it will sense only smoke.

We design an embedded system to perform a specific task. They are the most attracted device nowadays because of certain factors like low cost, low power and space consumption. In addition, they are easily portable so you can carry them around easily.

An embedded system has three components −
•It has hardware.
•It has application software.
•It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.

An embedded system hardware has 5 modules: the processor, memory, input devices, output devices and bus controllers.

•**Processors:** Embedded processors can either be a microprocessor or a microcontroller. Microprocessors needs separate integrated circuits for memory and peripherals whereas microcontrollers have on-chip peripherals which reduces power consumption, size and its cost. They include the following:

1.Microcontroller (CPU) — It is an intelligent device that computes the tasks assigned by the user. It builds small applications with precise calculations.

2.System on Chip (SoC) — It is an integrated circuit that integrates all components of a computer. It comprises of CPU, peripheral devices (timers, counters, etc), Communication interfaces (I^2C, SPI, UART), and power management circuits on a single integrated circuit.

3.ASIC processor (Application Specific Integrated Circuit) — It is mainly designed for a specific application rather than for general purposes.

4.DSP processor — removes the noise and improves signal quality in audio and video applications.

- **Memory**: Its the area used for storage of the data and the information. There are different types of memory for the embedded system including RAM(Random Access Memory), ROM(Read Only Memory), DRAM(Dynamic RAM) , SRAM(Static RAM).

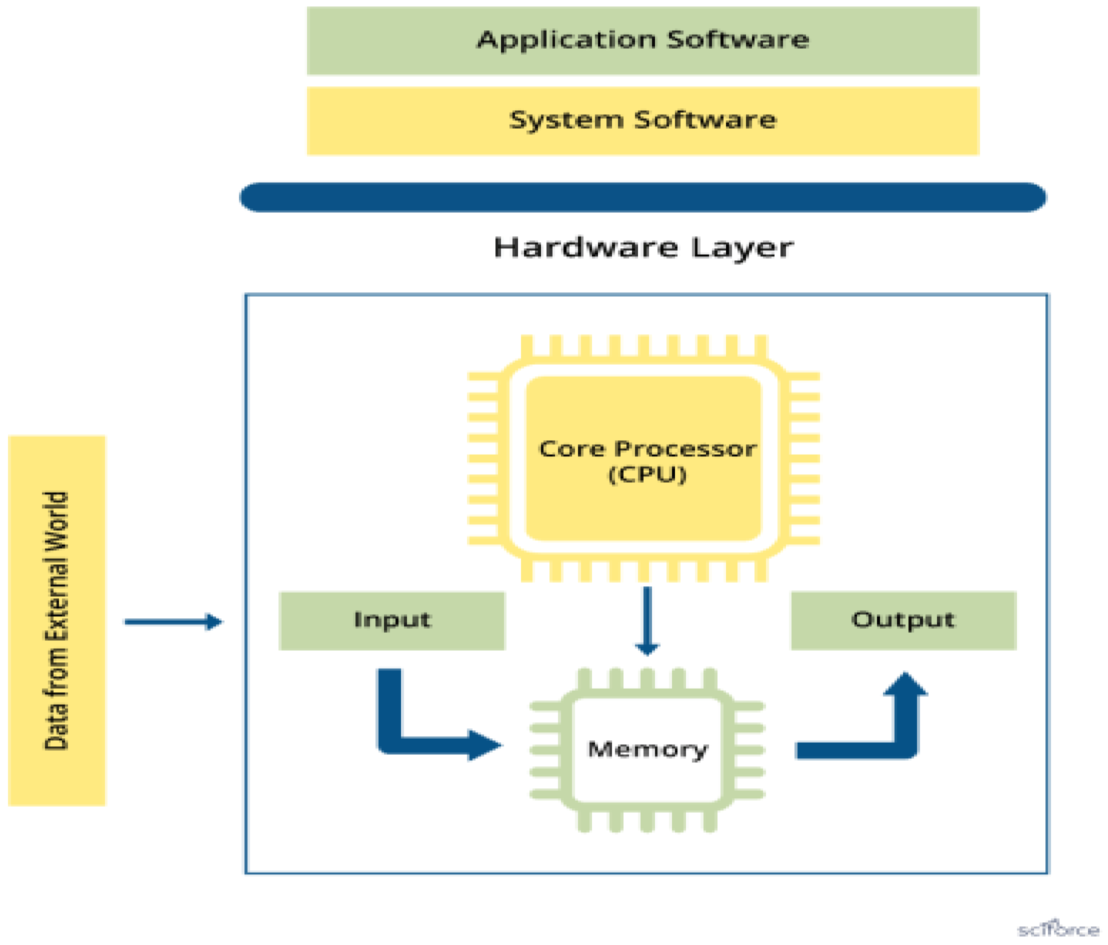
- **Input devices**: Input devices capture data from the outside. They acquire data to perform the tasks in order to provide the output. Input devices include sensors, switches, photodiode, optocouplers, etc.

- **Output devices**: Output devices respond to input events from outside the microcontroller and display it using output device. For example, LCD (Liquid Crystal Display), seven-segment displays to display output, buzzers and LEDs for notifying purposes, and controllers such as actuators, relays etc.

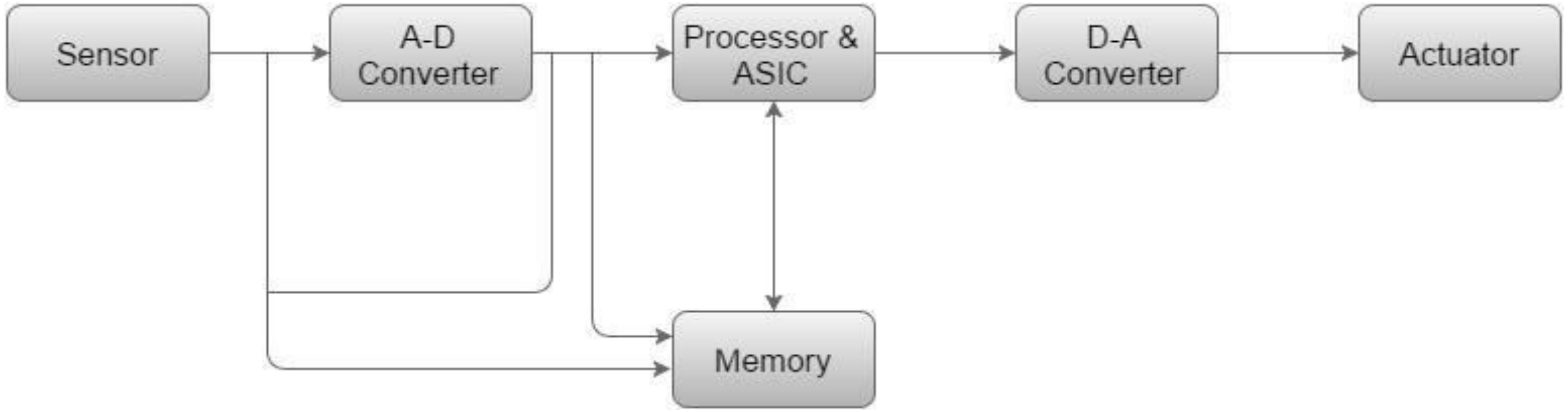
- **Bus controllers:** It acts as a communication device. The bus controller transfers data between the components inside an embedded system. For instance, commonly used bus controllers are serial buses (I2C, SPI, SMBus, etc.), RS232, RS485 and Universal Serial Bus (USB).

Embedded software

•**Device Driver**: A device driver is an embedded code written for a specific piece of hardware.
•**Operating System (OS) or MicroOS**: Embedded systems have a gamut of operating systems, including RTOS (Real-time Operating Systems), mobile embedded, stand-alone and network embedded systems.
Most of the embedded software uses any of the two languages: C and C++. C++ has features, like enhanced security and closeness to real-world applications, whereas C is more reliable and has better performance by directly interacting with the hardware.


Application Software
System Software
Hardware Layer
Data from External World
Core Processor (CPU)
Input
Output
Memory
sciforce

Basic Structure of an Embedded System



•**Sensor** − It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.

•**A-D Converter** − An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal.

•**Processor & ASICs** − Processors process the data to measure the output and store it to the memory.

•**D-A Converter** − A digital-to-analog converter converts the digital data fed by the processor to analog data

•**Actuator** − An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

Role of Embedded systems in the Internet of Things:

- Internet of Things is the concept of connecting devices via the internet to exchange data. It is the most trending technology in this modern world as we can control the embedded devices from any location using the Internet of things.

- In sum, the Internet of Things (IoT) is a process in which objects are equipped with sensors, actuators, and processors that involve hardware board design and development, software systems, web APIs, and protocols, which together create a connected environment of embedded systems.

A good example is a Smart Refrigerator that's connected to the Internet. It comes with several smart embedded features, one of which is an Embedded Camera that keeps an eye on what items does the Refrigerator contains, whether they contain the item you like and if so, how much of the item is left-over or whether they are spoilt and so on.

Either a remote App in the Smartphone or a touchscreen tablet built into the Refrigerator's right door with connectivity to the Embedded Camera with Onboard Operating System having connectivity to the IOT, allows the person to get the confirmation of the same in the form of announcement, mentioning 'D-Day Alert'.

Real World Applications of IoT

Wearables:

Wearables have experienced a explosive demand in markets all over the world. Companies like Google, Samsung have invested heavily in building such devices. But, how do they work?
Wearable devices are installed with sensors and softwares which collect data and information about the users. This data is later pre-processed to extract essential insights about user.
These devices broadly cover fitness, health and entertainment requirements. The pre-requisite from internet of things technology for wearable applications is to be highly energy efficient or ultra-low power and small sized.

Connected Cars

The automotive digital technology has focused on optimizing vehicles internal functions. But now, this attention is growing towards enhancing the in-car experience.
A connected car is a vehicle which is able to optimize it’s own operation, maintenance as well as comfort of passengers using onboard sensors and internet connectivity.
Most large auto makers as well as some brave startups are working on connected car solutions. Major brands like Tesla, BMW, Apple, Google are working on bringing the next revolution in automobiles.

Smart Cities:

Smart city is another powerful application of IoT generating curiosity among world’s population. Smart surveillance, automated transportation, smarter energy management systems, water distribution, urban security and environmental monitoring all are examples of internet of things applications for smart cities.
IoT will solve major problems faced by the people living in cities like pollution, traffic congestion and shortage of energy supplies etc. Products like cellular communication enabled Smart Belly trash will send alerts to municipal services when a bin needs to be emptied.
By installing sensors and using web applications, citizens can find free available parking slots across the city. Also, the sensors can detect meter tampering issues, general malfunctions and any installation issues in the electricity system.

IoT in agriculture:

With the continous increase in world’s population, demand for food supply is extremely raised. Governments are helping farmers to use advanced techniques and research to increase food production. Smart farming is one of the fastest growing field in IoT.
Farmers are using meaningful insights from the data to yield better return on investment. Sensing for soil moisture and nutrients, controlling water usage for plant growth and determining custom fertilizer are some simple uses of IoT.

Smart Retail:

The potential of IoT in the retail sector is enormous. IoT provides an opportunity to retailers to connect with the customers to enhance the in-store experience.
Smartphones will be the way for retailers to remain connected with their consumers even out of store. Interacting through Smartphones and using Beacon technology can help retailers serve their consumers better. They can also track consumers path through a store and improve store layout and place premium products in high traffic areas.

Energy Engagement:

Power grids of the future will not only be smart enough but also highly reliable. Smart grid concept is becoming very popular all over world.
The basic idea behind the smart grids is to collect data in an automated fashion and analyze the behavior or electricity consumers and suppliers for improving efficiency as well as economics of electricity use.
Smart Grids will also be able to detect sources of power outages more quickly and at individual household levels like near by solar panel, making possible distributed energy system.

Smart Home:

With IoT creating the buzz, ‘Smart Home’ is the most searched IoT associated feature on Google. But, what is a Smart Home?

Wouldn’t you love if you could switch on air conditioning before reaching home or switch off lights even after you have left home? Or unlock the doors to friends for temporary access even when you are not at home. Don’t be surprised with IoT taking shape companies are building products to make your life simpler and convenient.
Smart Home has become the revolutionary ladder of success in the residential spaces and it is predicted Smart homes will become as common as smartphones.