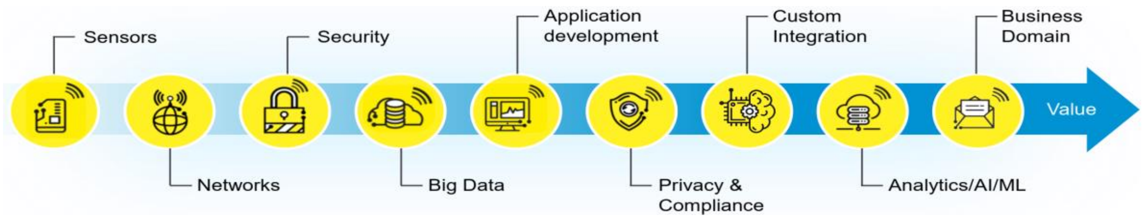
Chapter 3

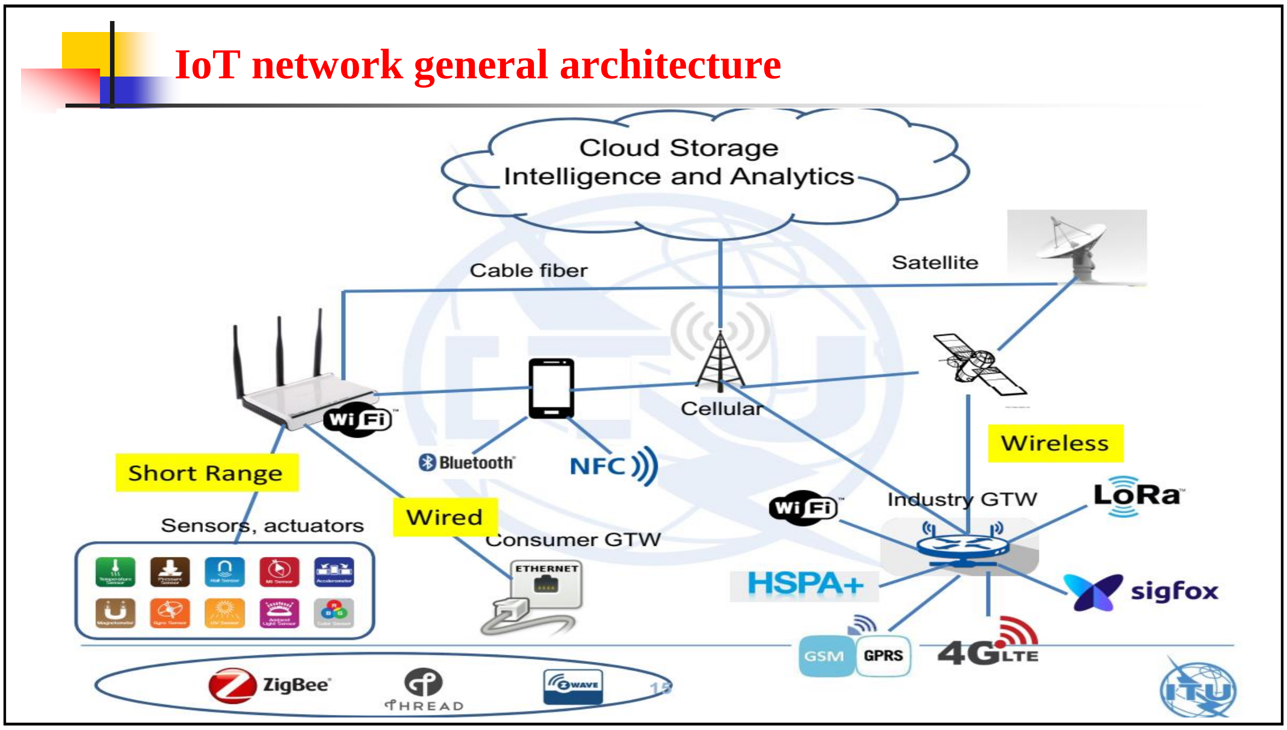
# IoT NETWORKING

## Technologies used in IoT

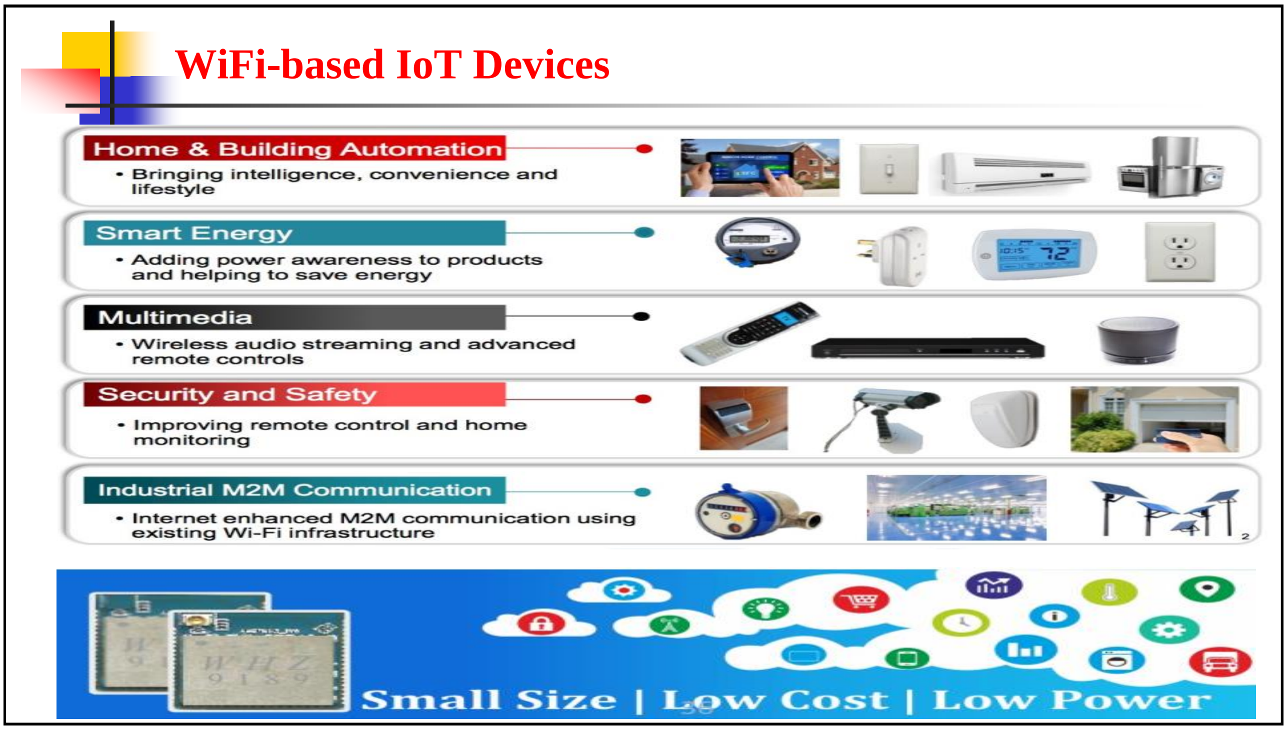
- Success in IoT Projects needs expertise in



  - Networking: connect componets in systems
  - Connecting systems can be: wired or wireless
  - Short range
  - Long range



## IoT network general architecture



## WiFi-based IoT Devices

**Home & Building Automation**
- Bringing intelligence, convenience and lifestyle

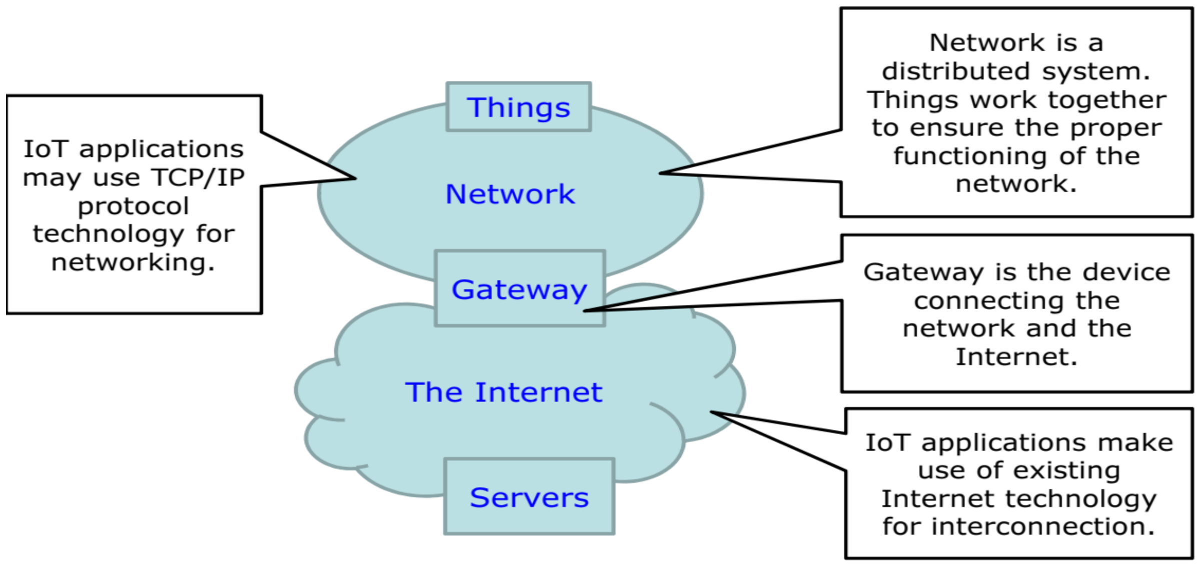
**Smart Energy**
- Adding power awareness to products and helping to save energy

**Multimedia**
- Wireless audio streaming and advanced remote controls

**Security and Safety**
- Improving remote control and home monitoring

**Industrial M2M Communication**
- Internet enhanced M2M communication using existing Wi-Fi infrastructure

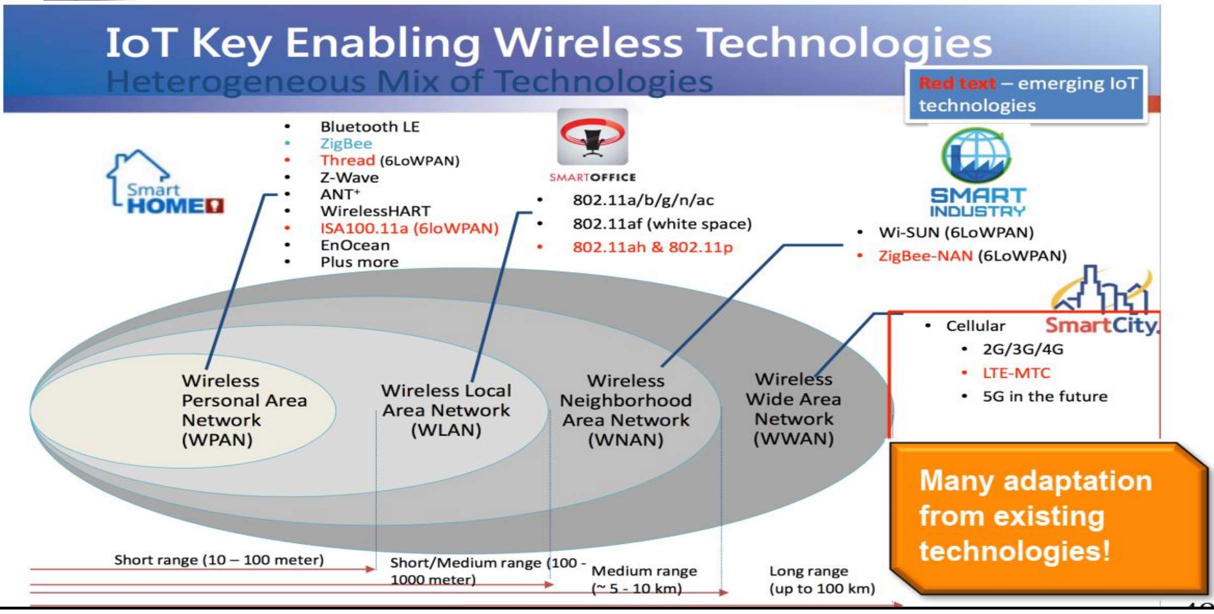
## IoT Networking


IoT applications may use TCP/IP protocol technology for networking.
Things
Network
Gateway
The Internet
Servers
Network is a distributed system. Things work together to ensure the proper functioning of the network.
Gateway is the device connecting the network and the Internet.
IoT applications make use of existing Internet technology for interconnection.

## IoT Short Range and Long Range Systems

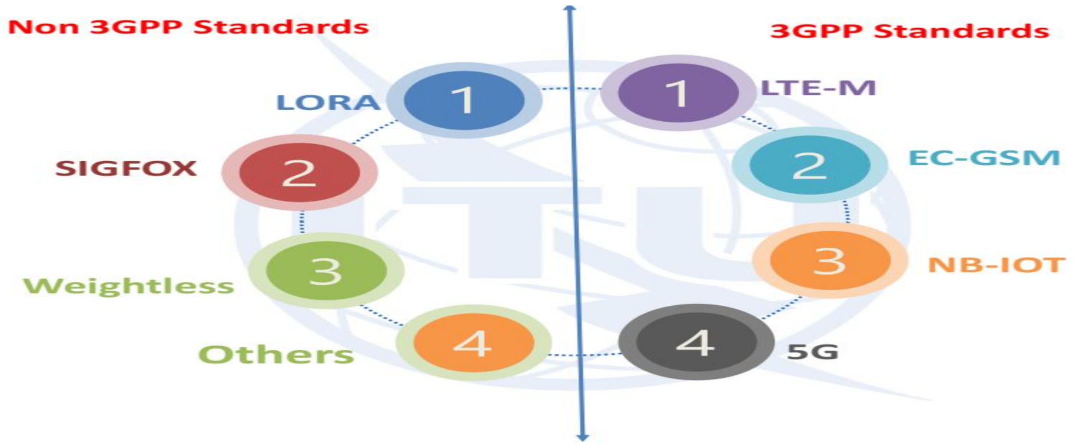
- **A. Fixed & Short Range**
  - **1. RFID**
  - **2. Bluetooth**
  - **3. Zigbee**
  - **4. WiFi**
- **B. Long Range technologies**
  - **1. Non 3GPP Standards (LPWAN)**
  - **2. 3GPP Standards**
-

## Fixed & Short Range and Long Range


IoT Key Enabling Wireless Technologies
Heterogeneous Mix of Technologies
Red text – emerging IoT technologies
Smart HOME
Bluetooth LE
ZigBee
Thread (6LoWPAN)
Z-Wave
ANT+
WirelessHART
ISA100.11a (6loWPAN)
EnOcean
Plus more
SMARTOFFICE
802.11a/b/g/n/ac
802.11af (white space)
802.11ah & 802.11p
SMART INDUSTRY
Wi-SUN (6LoWPAN)
ZigBee-NAN (6LoWPAN)
SmartCity
Cellular
2G/3G/4G
LTE-MTC
5G in the future
Wireless Personal Area Network (WPAN)
Wireless Local Area Network (WLAN)
Wireless Neighborhood Area Network (WNAN)
Wireless Wide Area Network (WWAN)
Many adaptation from existing technologies!
Short range (10 – 100 meter)
Short/Medium range (100 - 1000 meter)
Medium range (~ 5 - 10 km)
Long range (up to 100 km)

## Long Range technologies

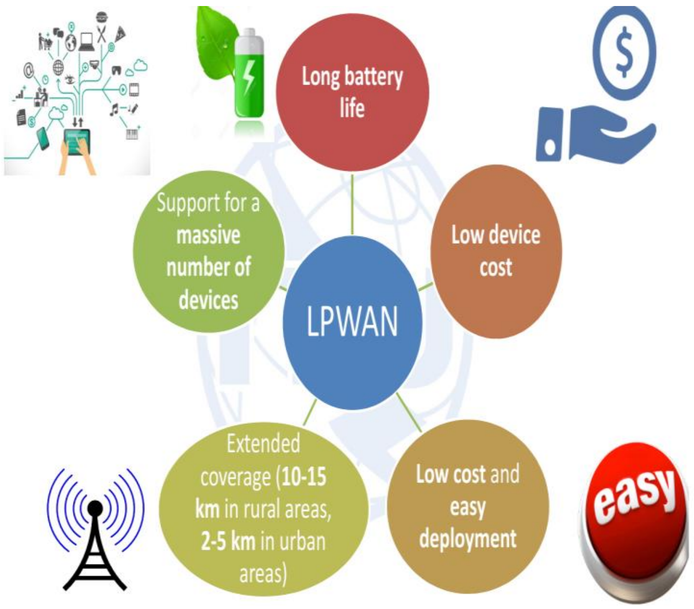
- Non 3GPP Standards (LPWAN)
- 3GPP Standards



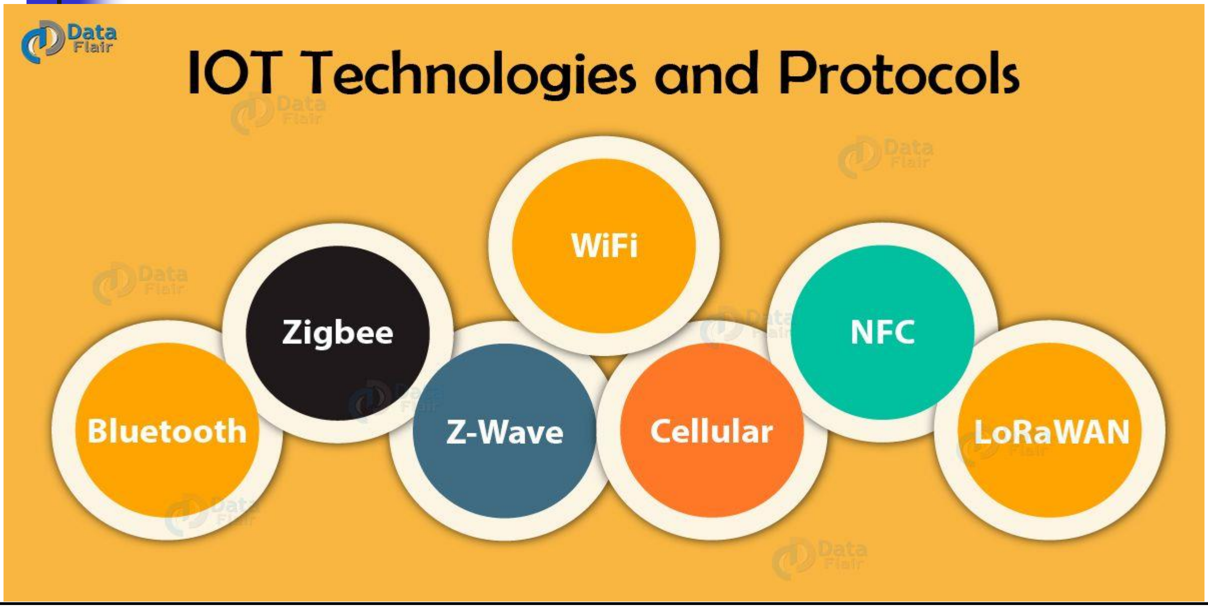
## Non 3GPP Standards (LPWAN)

- Consist of :
  - LoRaWAN
  - Sigfox
  - Weightless
  - RPMA
  - Others
- LPWAN

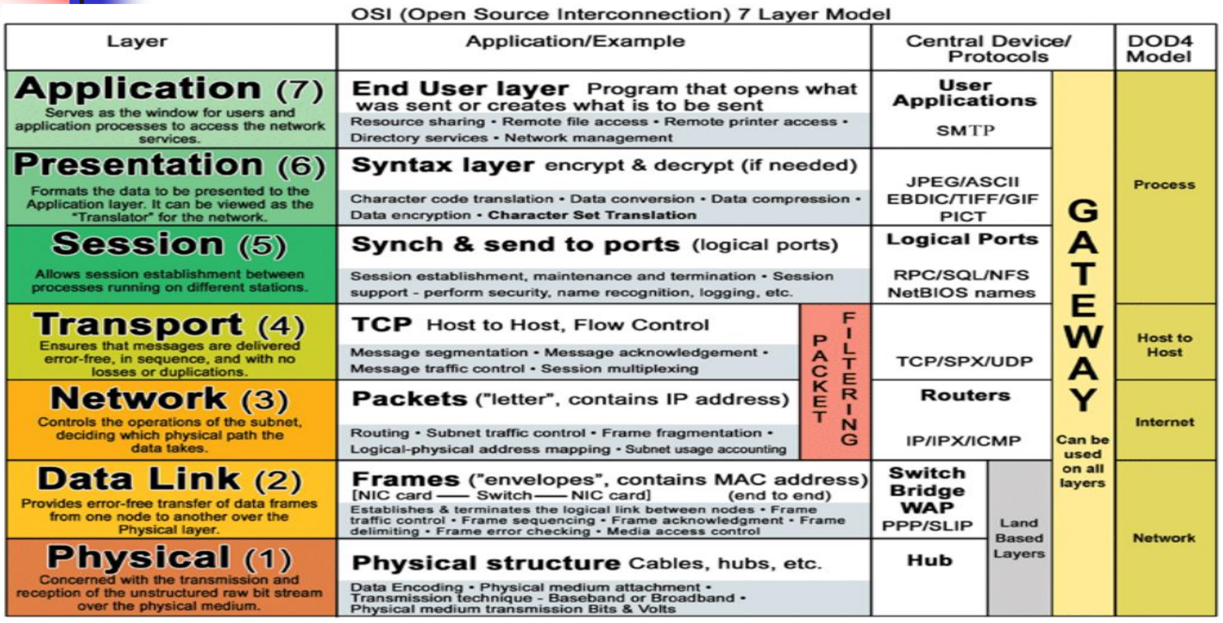
REQUIREMENTS


Long battery life
Support for a massive number of devices
LPWAN
Low device cost
Extended coverage (10-15 km in rural areas, 2-5 km in urban areas)
Low cost and easy deployment
easy

## Iot Technology & Protocols – 7 Important IoT Communication Protocols



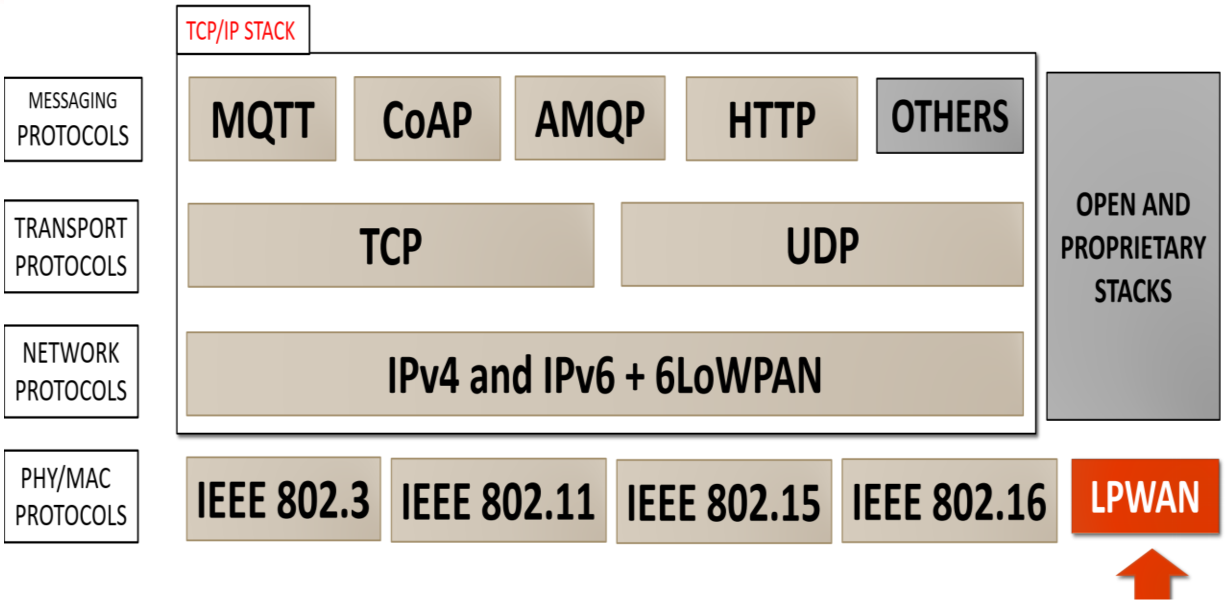
## OSI model



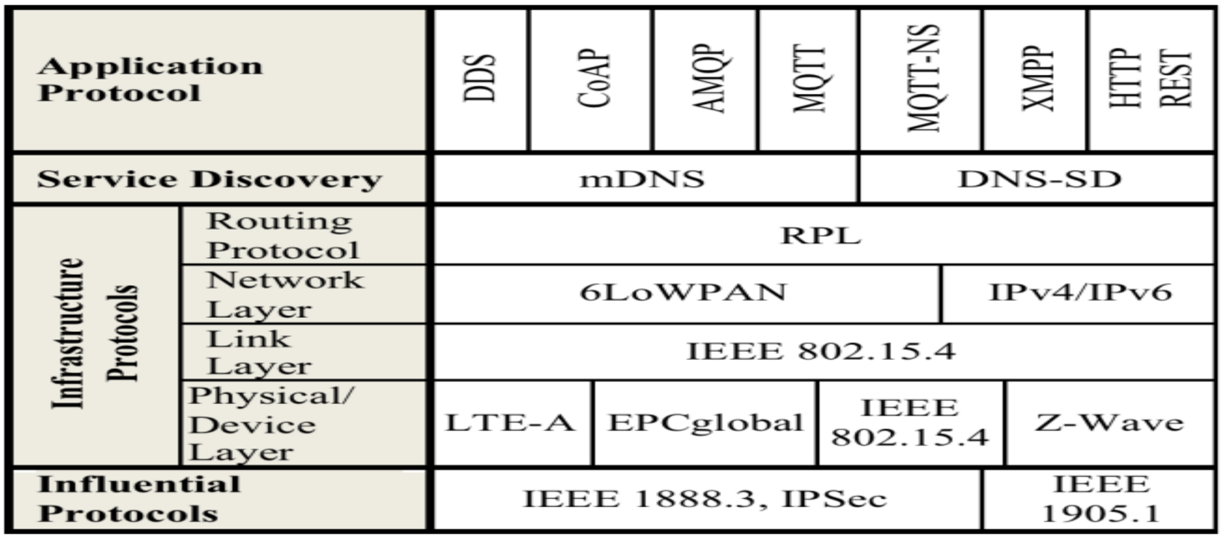
OSI (Open Source Interconnection) 7 Layer Model

| Layer | Application/Example | Central Device/ Protocols | | DOD4 Model |
|---|---|---|---|---|
| **Application (7)** Serves as the window for users and application processes to access the network services. | **End User layer** Program that opens what was sent or creates what is to be sent<br>Resource sharing • Remote file access • Remote printer access • Directory services • Network management | **User Applications**<br>SMTP | **GATEWAY**<br>Can be used on all layers | Process |
| **Presentation (6)** Formats the data to be presented to the Application layer. It can be viewed as the "Translator" for the network. | **Syntax layer** encrypt & decrypt (if needed)<br>Character code translation • Data conversion • Data compression • Data encryption • **Character Set Translation** | JPEG/ASCII<br>EBDIC/TIFF/GIF<br>PICT | | |
| **Session (5)** Allows session establishment between processes running on different stations. | **Synch & send to ports** (logical ports)<br>Session establishment, maintenance and termination • Session support - perform security, name recognition, logging, etc. | **Logical Ports**<br>RPC/SQL/NFS<br>NetBIOS names | | |
| **Transport (4)** Ensures that messages are delivered error-free, in sequence, and with no losses or duplications. | **TCP** Host to Host, Flow Control<br>Message segmentation • Message acknowledgement • Message traffic control • Session multiplexing<br>PACKET FILTERING | TCP/SPX/UDP | | Host to Host |
| **Network (3)** Controls the operations of the subnet, deciding which physical path the data takes. | **Packets** ("letter", contains IP address)<br>Routing • Subnet traffic control • Frame fragmentation • Logical-physical address mapping • Subnet usage accounting | **Routers**<br>IP/IPX/ICMP | | Internet |
| **Data Link (2)** Provides error-free transfer of data frames from one node to another over the Physical layer. | **Frames** ("envelopes", contains MAC address) [NIC card —— Switch —— NIC card] (end to end)<br>Establishes & terminates the logical link between nodes • Frame traffic control • Frame sequencing • Frame acknowledgment • Frame delimiting • Frame error checking • Media access control | **Switch Bridge WAP** PPP/SLIP<br>Land Based Layers | | Network |
| **Physical (1)** Concerned with the transmission and reception of the unstructured raw bit stream over the physical medium. | **Physical structure** Cables, hubs, etc.<br>Data Encoding • Physical medium attachment • Transmission technique - Baseband or Broadband • Physical medium transmission Bits & Volts | **Hub** | | |

## IoT Protocol Stack

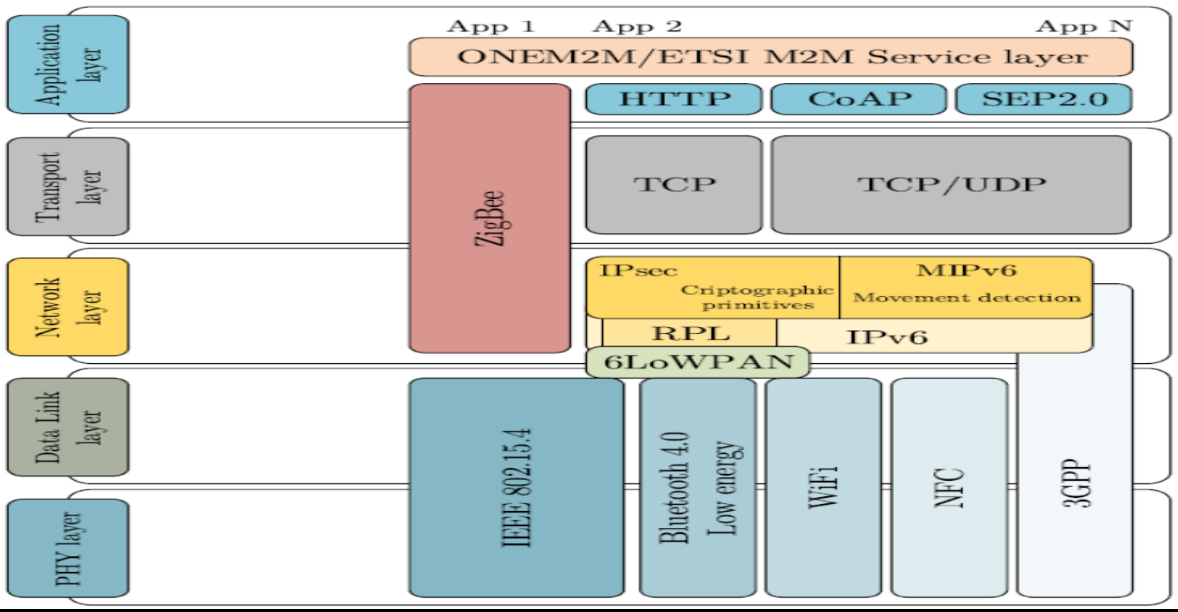


## IoT Protocols & Standards



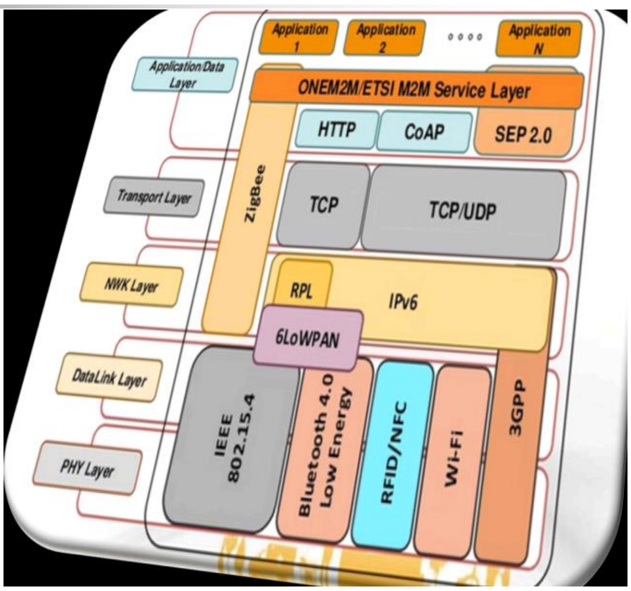
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Application Protocol | | DDS | CoAP | AMQP | MQTT | MQTT-SN | XMPP | HTTP REST |
| Service Discovery | | mDNS | | | | DNS-SD | | |
| Infrastructure Protocols | Routing Protocol | RPL | | | | | | |
| | Network Layer | 6LoWPAN | | | | | IPv4/IPv6 | |
| | Link Layer | IEEE 802.15.4 | | | | | | |
| | Physical/ Device Layer | LTE-A | EPCglobal | | IEEE 802.15.4 | | Z-Wave | |
| Influential Protocols | | IEEE 1888.3, IPSec | | | | | IEEE 1905.1 | |

## Some IoT protocol stacks



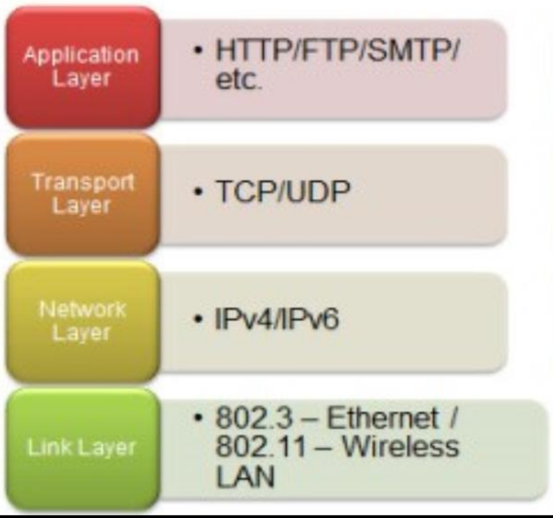
## IoT Protocols

- IoT covers a wide range of industries and use cases
  - From single constrained device to massive cross-platform deployments of embedded technologies and cloud systems connecting in real-time
- Integrating numerous legacy and emerging communication protocols into a coherent ecosystem of interconnected devices, serves is a challenge!



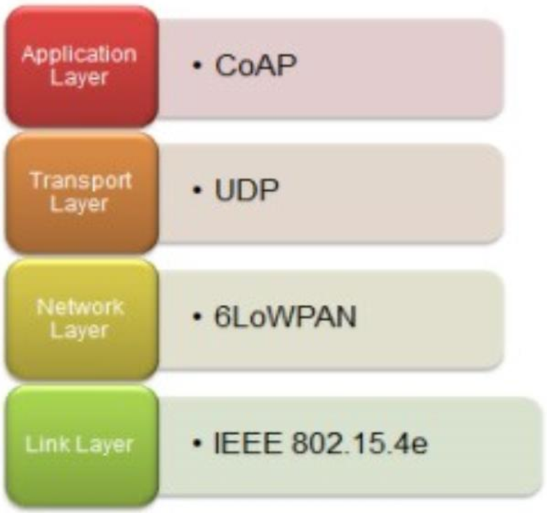
## Protocol Stacks

**Internet Protocol Suite**



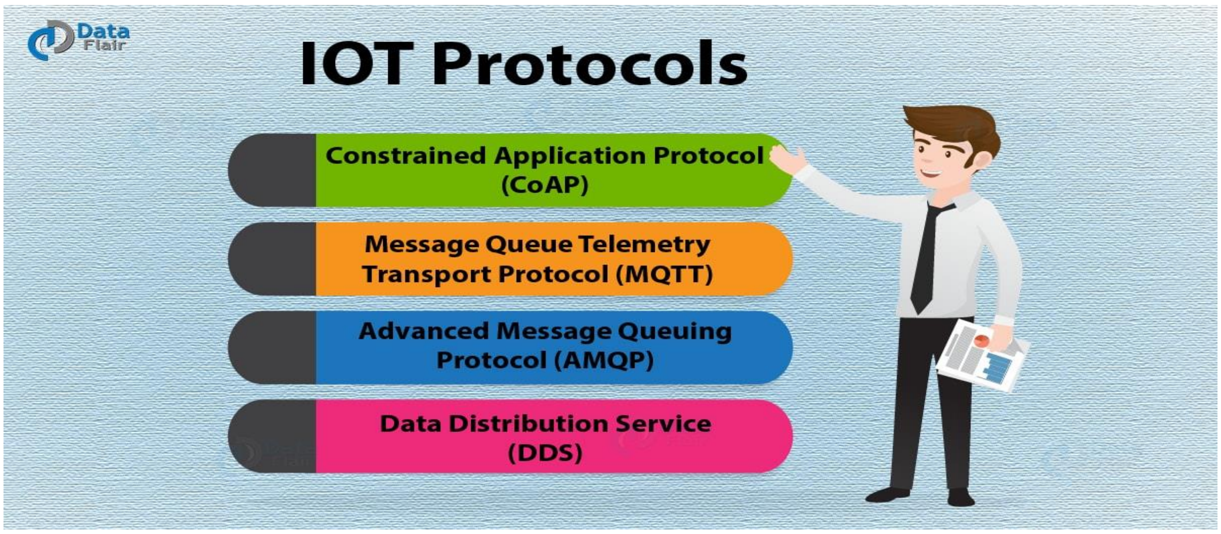
| | |
|---|---|
| Application Layer | • HTTP/FTP/SMTP/ etc. |
| Transport Layer | • TCP/UDP |
| Network Layer | • IPv4/IPv6 |
| Link Layer | • 802.3 – Ethernet / 802.11 – Wireless LAN |

**IP Smart Object Protocol Suite**

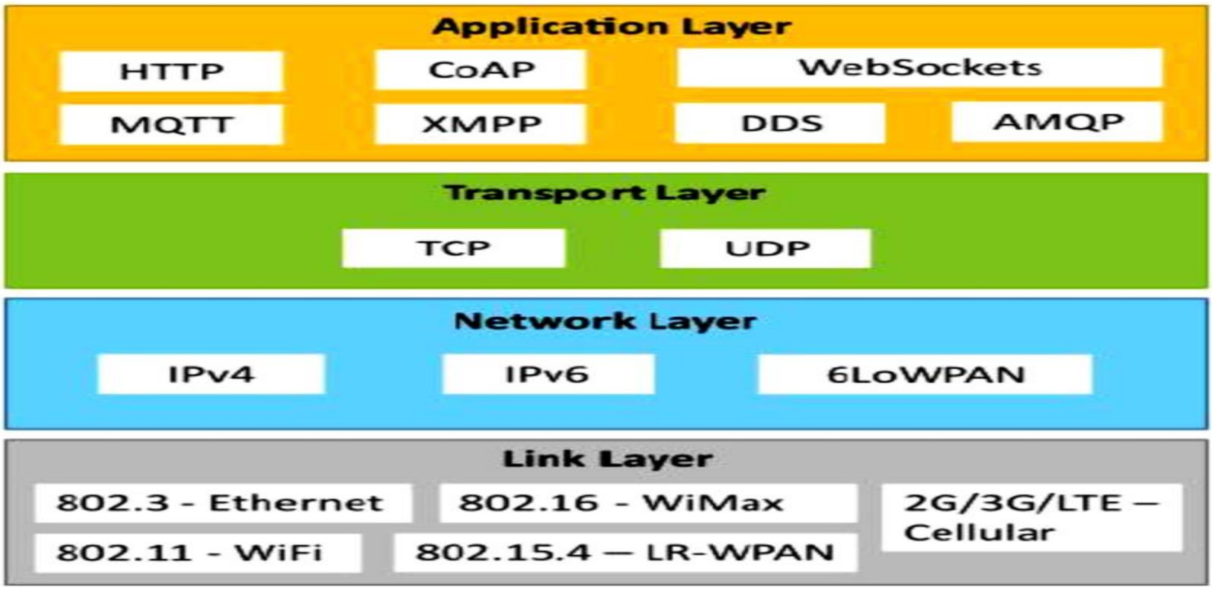


| | |
|---|---|
| Application Layer | • CoAP |
| Transport Layer | • UDP |
| Network Layer | • 6LoWPAN |
| Link Layer | • IEEE 802.15.4e |

## 4 Key IoT Protocols


Data Flair
IOT Protocols
Constrained Application Protocol (CoAP)
Message Queue Telemetry Transport Protocol (MQTT)
Advanced Message Queuing Protocol (AMQP)
Data Distribution Service (DDS)

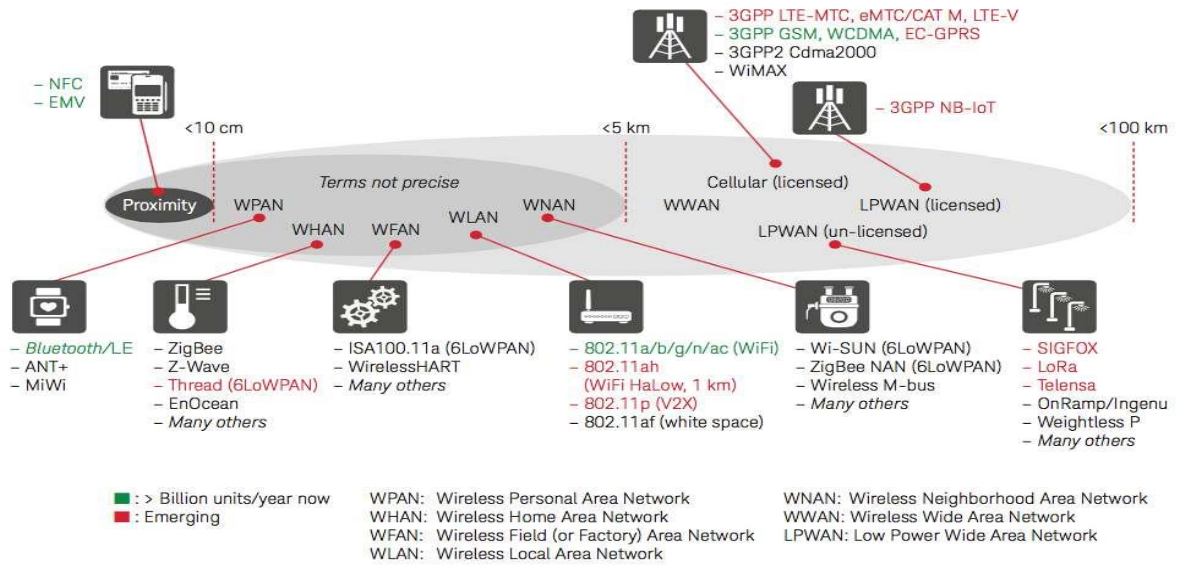
## IoT Protocols



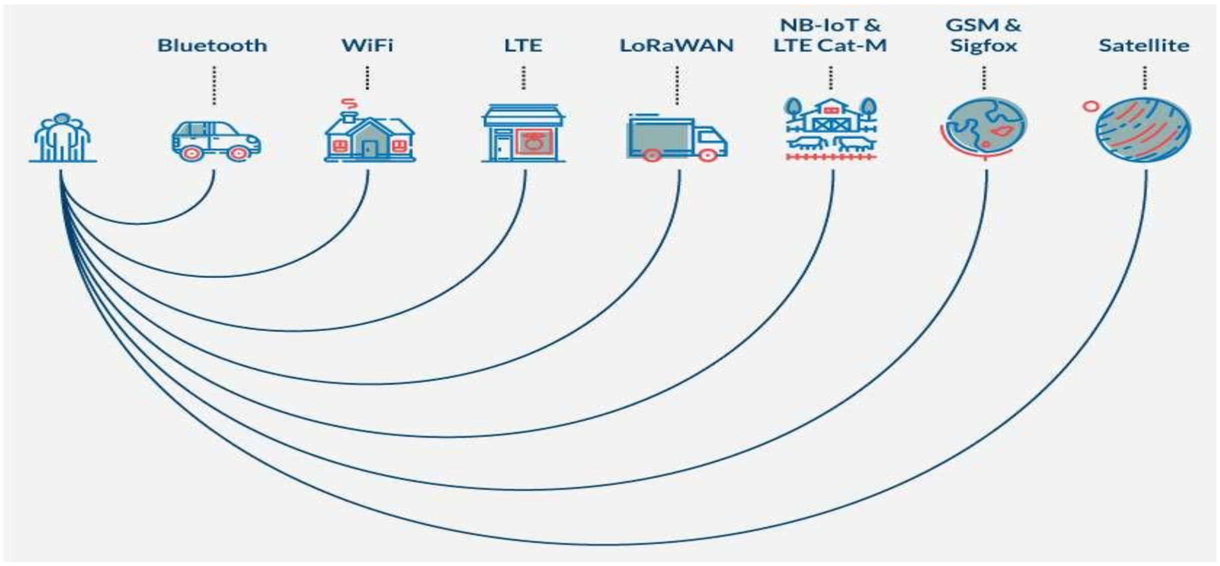
## Wireless Communication Protocols for the IoT

- The capability of connecting to things in a **seamless**, **ubiquitous**, and **cheap** manner have pushed the spread of IoT solutions
- IoT wireless communication protocols primarily differs in relation to
  - **coverage range**: from few cm to several km
  - **power consumption**: from few mW to several W
  - **bandwidth**: from few bytes per day to hundreds of MB/s
  - **security**: from plain data to strong encryption
  - **cost**: greatly varying, both for equipment and data transmission

## Several Wireless Communication Protocols



## Wireless Protocols per Coverage Range



## Connectivity Terminologies

- **IoT Node:** These are machines, things or computers Connected to other nodes inside a LAN via the IoT LAN, May be sometimes connected to the internet through a WAN directly
- **IoT LAN:** It is Local, Short range Comm, May or may not connect to Internet, Building or Organization wide
- **IoT WAN:** Connection of various network segments, Organizationally and geographically wide, Connects to the internet
- **IoT Gateway:** A router connecting the IoT LAN to a WAN to the Internet, can implement several LAN and WAN, Forwards packets between LAN and WAN on the IP layer IoT Proxy: Performs active application layer functions between IoT nodes and other entities
- **Gateway Prefix Allotment:**

- **Gateway Prefix Allotment:**
  - One of the strategies of address conservation in IoT is to use local addresses which exist uniquely within the domain of the gateway. These are represented by the circles in this slide.
  - The network connected to the internet has routers with their set of addresses and ranges.
  - These routers have multiple gateways connected to them which can forward packets from the nodes, to the Internet, only via these routers. These routers assign prefixes to gateways under them, so that the gateways can be identified with them

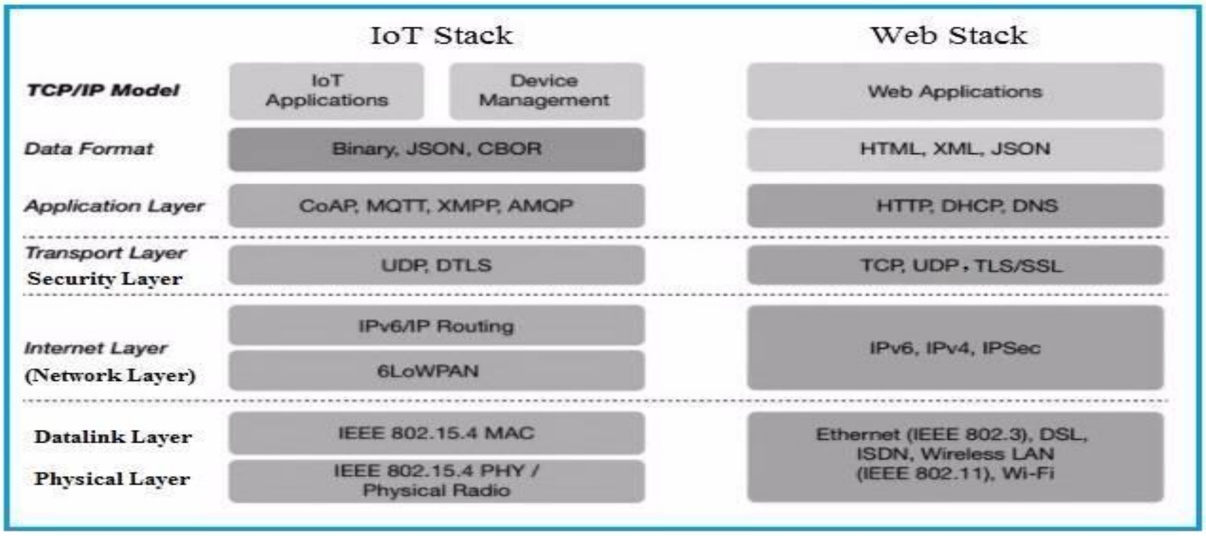
- **Impact of Mobility on Addressing**
  - The network prefix changes from 1 to 2 due to movement, making the IoT LAN safe from changes due to movements.
  - IoT gateway WAN address changes without change in LAN address. This is achieved using ULA.
  - The gateways assigned with prefixes, which are attached to a remote anchor point by using various protocols such as Mobile IPv6, and are immune to changes of network prefixes.
  - This is achieved using LU. The address of the nodes within the gateways remains unchanged as the gateways provide them with locally unique address and the change in gateway's network prefix doesn't affect them.
  - Sometimes, there is a need for the nodes to communicate directly to the internet. This is achieved by tunnelling, where the nodes communicate to a remote anchor point instead of channelling their packets through the router which is achieved by using tunnelling protocols such as IKEv2:internet key exchange version 2

## Multihoming

- Multihoming is the practice of connecting a host or a computer network to more than one network. This can be done in order to increase reliability or performance or to reduce cost. There are several different ways to perform multihoming.
- **Host multihoming**
  A single host may be connected to multiple networks. For example, a mobile phone might be simultaneously connected to a WiFi network and a 3G network, and a desktop computer might be connected to both a home network and a VPN. A multihomed host usually is assigned multiple addresses, one per connected network.
- **Classical multihoming**
  In classical multihoming a network is connected to multiple providers, and uses its own range of addresses (typically from a Provider Independent (PI) range). The network's edge routers communicate with the providers using a dynamic routing protocol, typically BGP, which announces the network's address range to all providers. If one of the links fails, the dynamic routing protocol recognizes the failure within seconds or minutes, and reconfigures its routing tables to use the remaining links, transparently to the hosts.

- **Multihoming with multiple addresses**
  In this approach, the network is connected to multiple providers, and assigned multiple address ranges, one for each provider. Hosts are assigned multiple addresses, one for each provider

## Deviation from regular Web



| Features | IoT Stack | WEB STACK |
|---|---|---|
| Function or application | It is used in constrained network having low power, low bandwidth and low memory requirements | It is used in non-constrained network having no limits on power/BW/memory. |
| Size of data to be transported | tens of bytes | hundreds or thousands of bytes |
| Data format | It uses CBOR (Concise Binary Object Representation) format as IoT is used for tiny messages. CBOR is based on JSON though CBOR uses binary encoding while JSON uses text encoding | It uses HTML, XML and JSON formats. |
| Application Layer | It uses CoAP protocol at application layer | It uses HTTP protocol at application layer. |

| | | |
|---|---|---|
| Transport layer | It uses UDP which is faster due to smaller header size compare to TCP. It is lighter protocol compare to TCP | It uses TCP which is connection oriented and slower compare to UDP. |
| Security layer | It uses DTLS (Datagram Transport Layer Security) protocol for security. | It uses TLS/SSL protocols for the same. |
| Internet layer | It uses 6LoWPAN to convert large IPv6 packets into small size packets to be carried on wireless medium as per bluetooth, zigbee etc. standards. It does fragmentation and reassembly. It also does header compression to reduce packet size | It does not require protocols like 6LoWPAN. Fragmentation and reassembly is taken care by transport layer (i.e. TCP) itself |
| Datalink or MAC layer | It will have MAC layer as per IoT wireless technology used viz. bluetooth, zigbee, zwave etc. It takes care of medium access control and resource allocation and management | It will have MAC layer as per LAN or WLAN or DSL or ISDN technologies |
| Physical layer and Radio Frequency (RF) layer | It will have physical layer (baseband) as per IoT wireless technologies viz. bluetooth, zigbee, zwave etc. It uses frequencies as per cellular or indoor wireless technologies and country wide allocations for the same | It will have PHY layer as per LAN or WLAN or DSL or ISDN technologies |

## IoT identification and Data protocols

- **IPv4:**
  - IP version four addresses are 32-bit integers which will be expressed in dotted decimal notation. Example- 192.0.2.126 could be an IPv4 address.
  - **Characteristics of IPv4**
    - IPv4 could be a 32-Bit IP Address.
    - IPv4 could be a numeric address, and its bits are separated by a dot.
    - The number of header fields is twelve and the length of the header field is twenty.
    - It has Unicast, broadcast, and multicast style of addresses.
    - IPv4 supports VLSM (Virtual Length Subnet Mask).
    - IPv4 uses the Post Address Resolution Protocol to map to the MAC address.
    - RIP may be a routing protocol supported by the routed daemon.
    - Networks ought to be designed either manually or with DHCP.
    - Packet fragmentation permits from routers and causing hos

- **IPv4 Datagram Header**

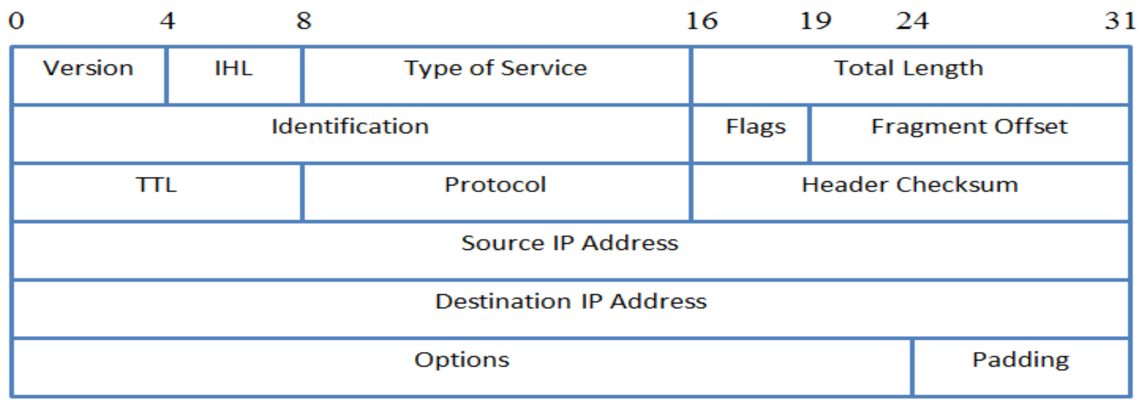


Fig: IPv4 Frame Format

- **IPv6**
  **Internet Protocol version 6** (IPv6) is also known as **Internet Protocol next generation** (**IPng**). It also accommodates more feature to meet the global requirement of growing Internet.
- To allocate a sufficient number of network address, IPv6 allows 128 bits of IP address separated into 8 sections of 2 bytes each. Unlike IPv4 where the address is represented as dotted-decimal notation, IPv6 uses hexadecimal numbers and colon (**“:”**) is used as a delimiter between the sections.
  **Example:** IPv6 address may be like this:

  FA20:B120: 6230:0000:0000: CE12:0006: ABDF

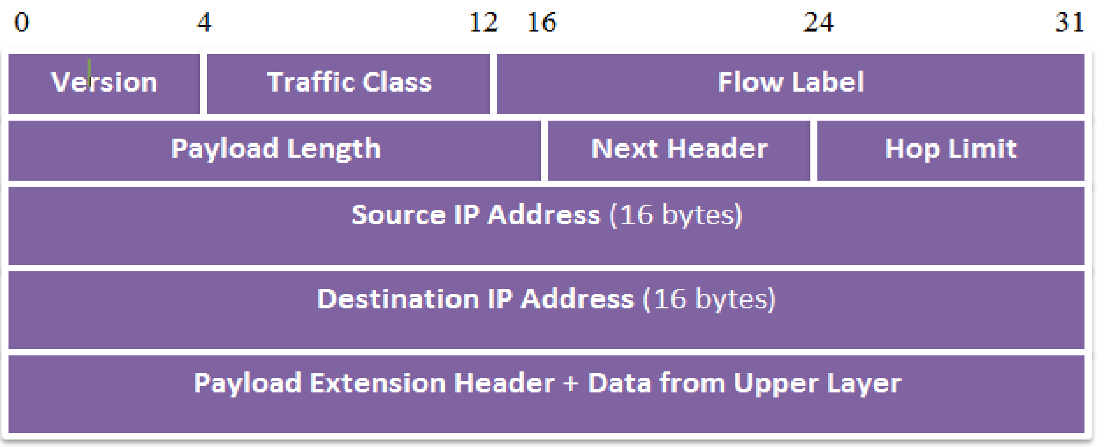

0
4
12
16
24
31
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source IP Address (16 bytes)
Destination IP Address (16 bytes)
Payload Extension Header + Data from Upper Layer

**Fig: IPv6 Packet Format**

## Data Exchange Protocols

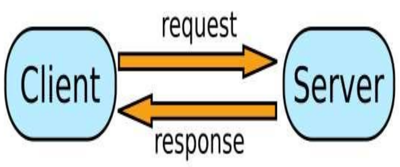
- How to encapsulate information and commands in messages?
- Data from gateways to servers: who should initiate the communication?
  - **push**: gateways autonomously decide to send messages to servers, e.g., when new sensed values are available
  - **pull**: servers ask to gateways to send messages, e.g., only when servers actually require sensed values
  - Two primary models
    - **request/response**, can be push or pull
    - **publish/subscribe**, usually only push
  - Several protocols:
    - request/response: REST/HTTP, CoAP, …
    - publish/subscribe: MQTT, AMQP, DDS, …

## Multiple Protocols supporting Different Features

- Several protocols thriving to become a de facto standard
- It could be difficult to pick up the **best solution**, since it **depends on requirements** of a given application domain
- Some request/response protocols:
  - REST architectural principles (REpresentational State Transfer), 2000
  - CoAP (Constrained Application Protocol), 2014 (also pub/sub)
- Some publish/subscribe protocols:
  - MQTT (Message Queue Telemetry Transport), 1999
  - AMQP (Advanced Message Queuing Protocol), 2003
  - DDS (Data Distribution Service), v1.0 2003; v1.2 2007
  - At different level of abstractions, LonWorks (1999), Modbus (1979)…

## Request/Response model

- A **client application** requests services (e.g., send data, require data, perform an addition) and a server application responds to the service request (e.g., by providing the data or the addition results)
- **Direct communication** between client and server
- Data exchange only if **the client starts the communication**
  - the server is not able to contact the client autonomously
- Typical interactions in the IoT scenario:
  - **push**: sensors send data to servers
  - **pull**: actuators request to servers new configurations



## REST - REpresentational State Transfer

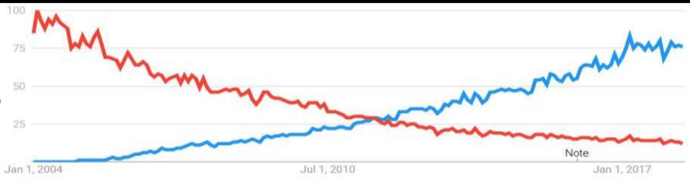
- REST is not an actual protocol, but substantially a **solution architectural style**
- Promote **client/server** and **stateless** interaction, oriented to the usage of **caching** opportunities, also with possibility of code-on-demand to clients
- Usually based on HTTP, the protocol used to surf the Web
- Very simple, but each time the client has to start the communication from the beginning

- REST: Identifying and Interacting with Resources

- **URI (Uniform Resource Identifier) to identify** the remote **resource**
  - any resource has a **persistent identifier**
  - do not transfer resources but their representations via HTTP
  - **endpoint** for a service managing user information: www.examples.com/resources/users
- **HTTP method to interact** with remote resources in a standard manner
  - **GET, retrieve** a specific resource (by id) or a collection of resources
  - **PUT, create** a new resource
  - **POST, update** a specific resource (by id)
  - **DELETE, remove** a specific resource by id

- REST: Endpoint Examples
  Remote resource: a software component managing users

| HTTP Method | URI | Operation |
|---|---|---|
| GET | www.examples.com/resources/**users** | Get list of users |
| GET | www.examples.com/resources/**users/1** | Get the user with id 1 |
| DELETE | www.examples.com/resources/**users/1** | Delete the user with id 1 |
| POST | www.examples.com/resources/**users/2** | Update the user with id 2 |
| PUT | www.examples.com/resources/**users/1** | Insert the user with id 1 |



- REST: Formatting data
  JavaScript Object Notation (JSON) most spread syntax for **formatting/ serializing data**, e.g., objects, arrays, numbers, strings, booleans, and null
- Resource: a group of users, each one with a first name and a last na
- {“users":
- [
- { "firstName":"John", "lastName":"Doe" },
- { "firstName":"Anna", "lastName":"Smith" },
- { "firstName":"Peter", "lastName":"Jones" }
- ]
- }

- REST: AWS IoT example
- Adds a thing to a thing group
  PUT **/thing groups/ addThingToThingGroup**
  HTTP/1.1
  Content-type: application/json
  {
  "thingArn": "string",
  "thingGroupArn": "string",
  "thingGroupName": "string",
  "thingName": "string"
  }

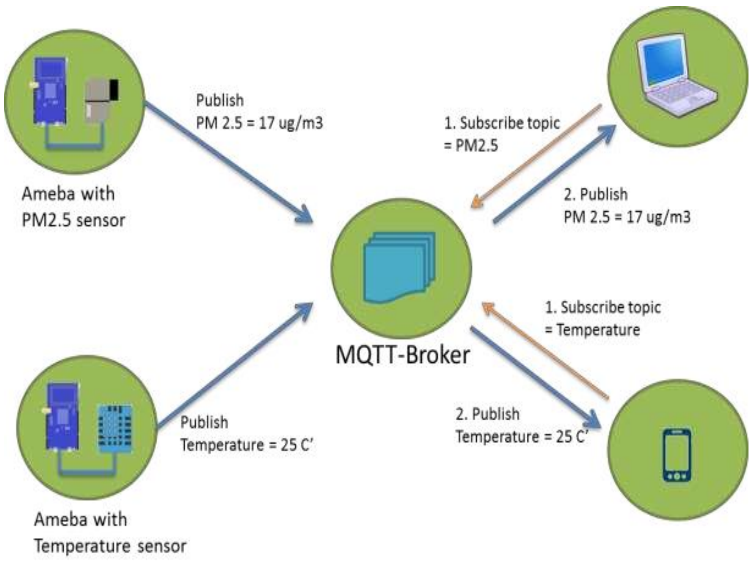
- Manage "shadows" of things

POST **endpoint/things/thingName/shadow**
HTTP/1.1
Content-type: application/json
{
"state":
{
"desired" :
{
"color" : { "r" : 10 },
"engine" : "ON"
}
}
}

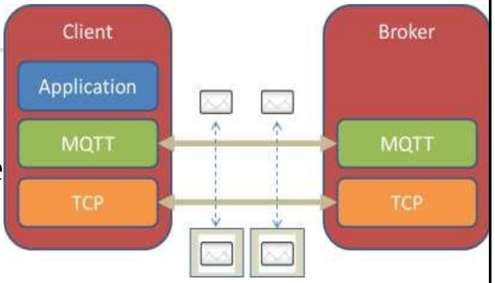
## MQTT - Message Queue Telemetry Transport

- IBM developed WebSphere MQ as a message-based backbone mainly for Enterprise Application Integration (EAI), MQTT is its evolution:
  - an open standard but it is strongly supported by IBM
  - designed to permit WebSphere MQ to talk with constrained (smart) devices at the edge of the network
- **MQTT: simple, lightweight, broker-based, publish/subscribe, open messaging protocol**
- Ideal for use in **constrained nodes and networks**
  - on embedded devices with limited processor or memory resources
  - where the network is expensive, has low bandwidth or is unreliable
- MQTT-SN for wireless sensor networks, aimed at embedded devices on non-TCP/IP networks (such as Zigbee).

- MQTT: Typical Use Case
- 1. Subscribe to one ore more topic
  2. Publish to a topic
  3. Receive messages related to subscribed topics


Ameba with PM2.5 sensor
Publish PM 2.5 = 17 ug/m3
1. Subscribe topic = PM2.5
2. Publish PM 2.5 = 17 ug/m3
MQTT-Broker
1. Subscribe topic = Temperature
2. Publish Temperature = 25 C'
Publish Temperature = 25 C'
Ameba with Temperature sensor

- MQTT: Features (1)
- **Publish/subscribe** message pattern to provide
  - one-to-many message distribution
  - decoupling of applications
- Use of **TCP/IP** to provide basic network connectivity
- **Small transport overhead** (minimal header length just 2 bytes) and protocol exchanges minimised to reduce network traffic
- **Easy to use** with few commands: connect, subscribe, publish, disconnect
- **Retained messages**: an MQTT broker can retain a message that can be sent to newly subscribing clients

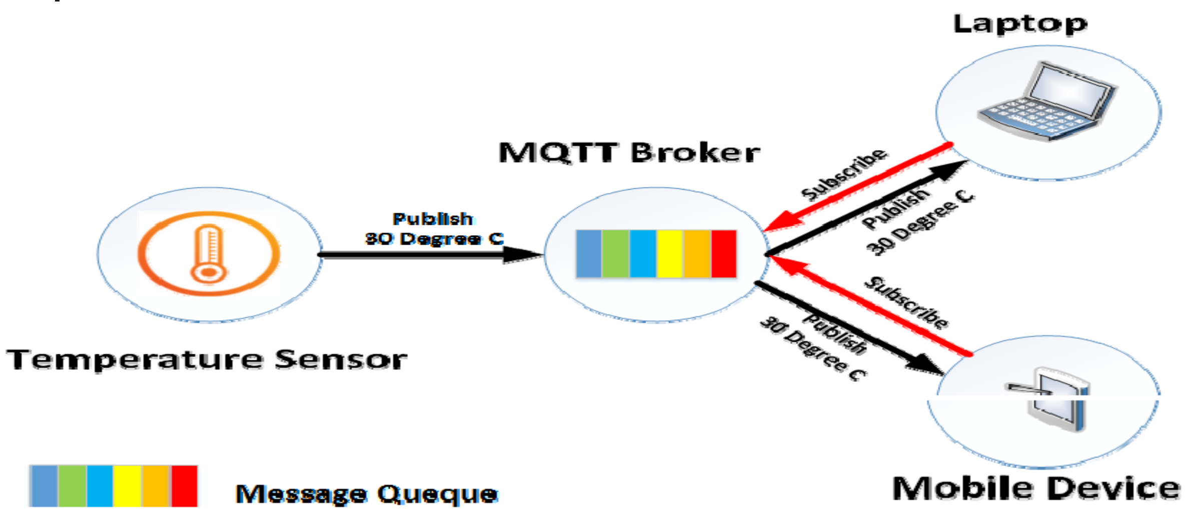


- MQTT: Features (2)
- **Three message delivery semantics** with increasing reliability and cost:
  - Quality of Service (QoS): at least once, at most once, exactly once
- **Durable connection**: client subscriptions remain in effect even in case of disconnection
  - subsequent messages with high QoS are stored for delivery after connection reestablishment
- **Wills**: a client can setup a will, a message to be published in case of unexpected disconnection, e.g., an alarm
- MQTT is a protocol adopted in several platforms: there are MQTT brokers in WebSphere, Mosquitto, RabbitMQ, etc. and clients in several languages

- **MQTT**
  - It is a publish-subscribe-based lightweight messaging protocol for use in conjunction with the TCP/IP protocol.
  - Designed to provide connectivity (mostly embedded) between applications and middle-wares on one side and networks and communications on the other side.
  - A message broker controls the publish-subscribe messaging pattern.
  - A topic to which a client is subscribed is updated in the form of messages and distributed by the message broker.
  - Designed for: Remote connections, Limited bandwidth, Small-code footprint.

- **MQTT Components**
  - **Publishers:** Lightweight sensors
  - **Subscribers:** Applications interested in sensor data
  - **Brokers:** Connect publishers and subscribers and Classify sensor data into topics

  **Communication:**



- The protocol uses a publish/subscribe architecture (HTTP uses a request/response paradigm).
- Publish/subscribe is event-driven and enables messages to be pushed to clients.
- The central communication point is the MQTT broker, which is in charge of dispatching all messages between the senders and the rightful receivers
- Each client that publishes a message to the broker, includes a topic into the message.The topic is the routing information for the broker.
- Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client.
- Therefore, the clients don't have to know each other. They only communicate over the topic.
- This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.

- **Applications**
  - Facebook Messenger uses MQTT for online chat.
  - Amazon Web Services use Amazon IoT with MQTT.
  - Microsoft Azure IoT Hub uses MQTT as its main protocol for telemetry messages.
  - The EVRYTHNG IoT platform uses MQTT as an M2M protocol for millions of connected products.
  - Adafruit launched a free MQTT cloud service for IoT experimenters called Adafruit IO.

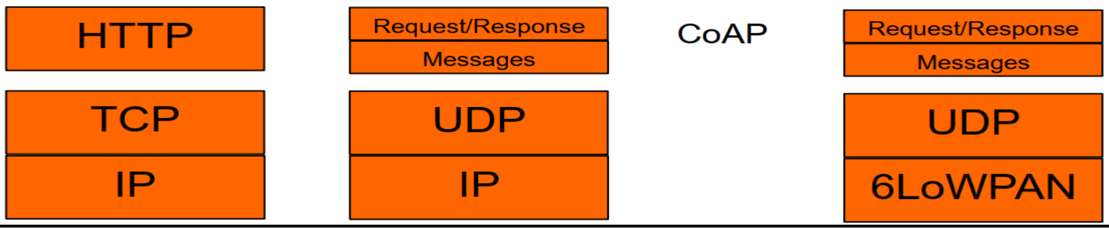
- **SMQTT**
  - Secure MQTT is an extension of MQTT which uses encryption based on lightweight attribute-based encryption.
  - The main advantage of using such encryption is the broadcast encryption feature, in which one message is encrypted and delivered to multiple other nodes, which is quite common in IoT applications.
  - In general, the algorithm consists of four main stages: setup, encryption, publish and decryption.

## CoAP - Constrained Application Protocol

- Designed for M2M/IoT applications such as smart-metering, ehealth, building and home automation with:
  - **constrained nodes**, e.g., 8-bit microcontrollers with 16KB RAM and battery-operated
  - **constrained networks**, e.g., Wireless Sensor Networks
- Low-overhead **request/response** protocol that also supports discovery of services/resources
- Based on **UDP** communications (with multicast)
- Inspired by (and compatible with) the **HTTP protocol** and REST architectures: CoAP is a specialized Web transfer protocol

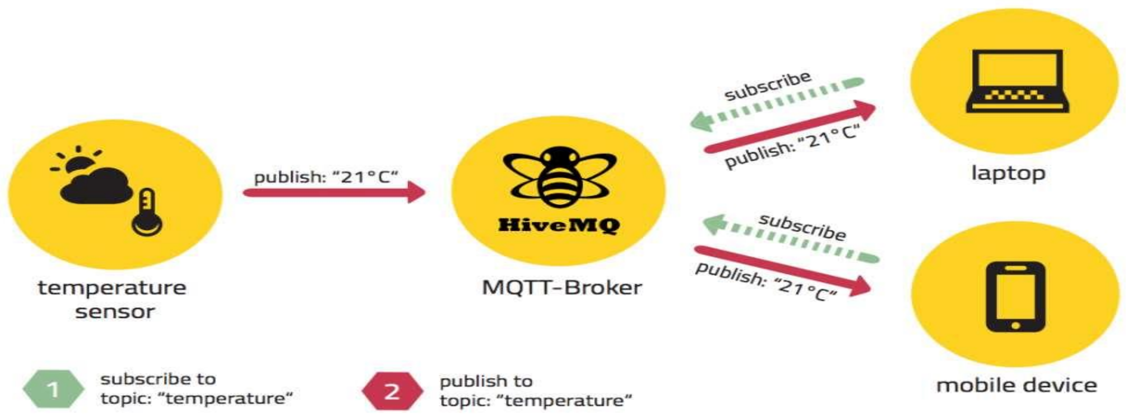
- CoAP Features
- **Web RESTful protocol** fulfilling M2M requirements in constrained environments
- **Simple request/response HTTP mapping**, to access CoAP resources via HTTP
- **URI** and Content-type support (a sensor is identified by an URI)
- **Low header overhead** and parsing complexity
- **Security** binding to Datagram Transport Layer Security (DTLS)
- UDP binding with **optional reliability**, supporting unicast and multicast
- **Asynchronous** message exchanges
- Services and resources **discovery**
- Also **publish/subscribe** (and push notifications)
- Simple **caching** (max-age parameter)
- CoAP implementations in many programming language, e.g., C, C++, and Java

- CoAP Messaging
- Duplicate detection and optional reliability
- Possible messages:
  - confirmable message (CON), wait for ACK
  - acknowledgement message (ACK), in response to CON
  - non-confirmable message (NON), no need to wait for ACK
- CoAP on top of UDP allows for multicast requests



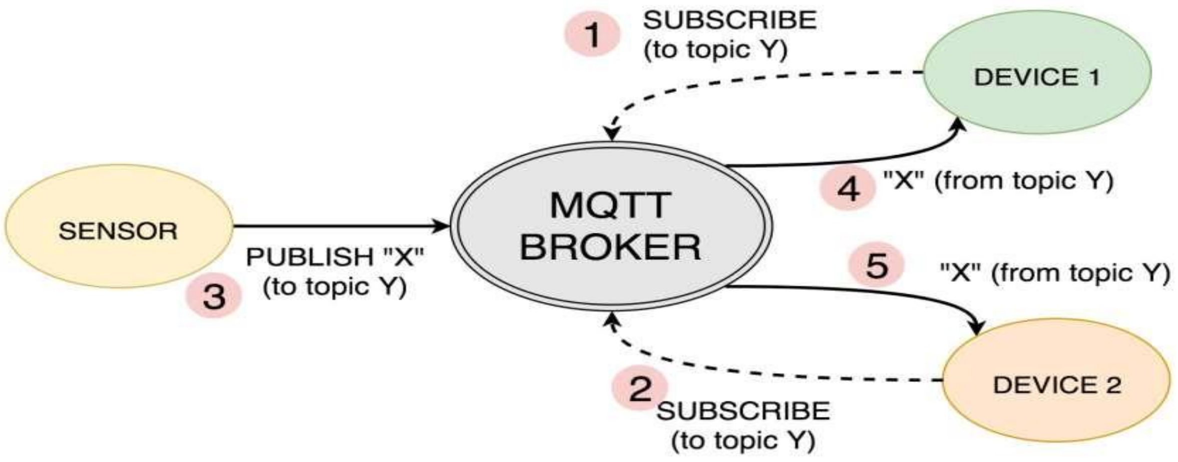
- Publish/Subscribe model
- Publish/subscribe (pub/sub) pattern alternative to traditional clientserver model, where a client communicates directly with an endpoint
- Roles:
  - **publisher**: a client sending a message
  - **subscriber**: one or more receivers waiting for the message
  - **broker**: a central component receiving and distributing messages to interested receivers

- Publish/Subscribe: Message Routing



- The broker **routes messages** (i.e., selects receivers of a message) based on:
  - **Message Topic**: a subject, part of each message. Receiving clients subscribe on the topics they are interested in with the broker and from there on they get all message based on the subscribed topics
  - **Message Type**: depending on the type of the message
  - **Message Header**: depending on a set of fields of the message
  - **Message Content**: possibly depending on the whole message content (expressive but expensive)

- Publish/Subscribe: Typical Sequence



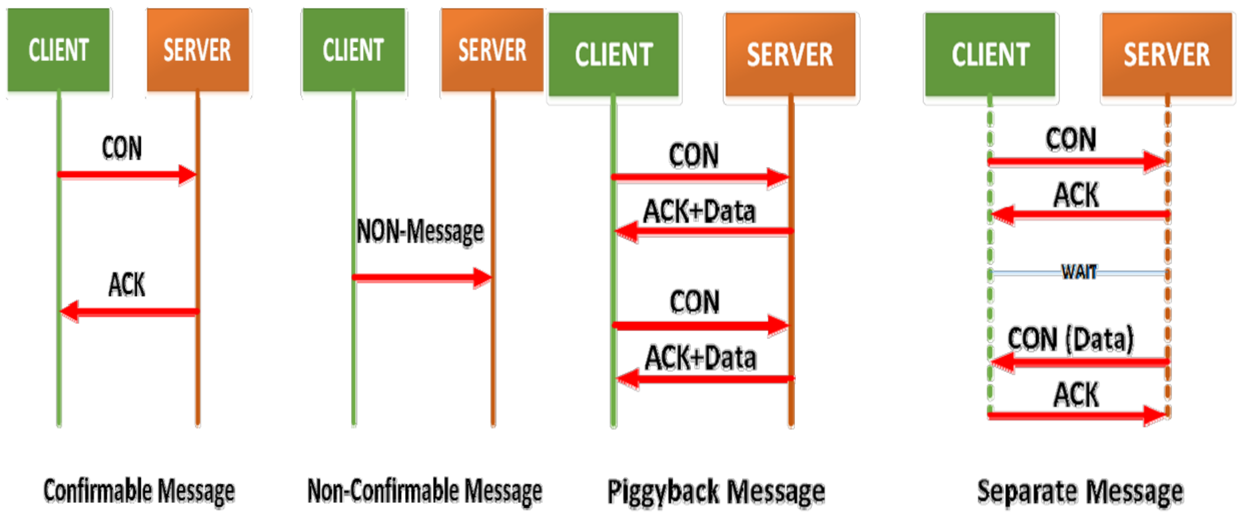
- Publish/Subscribe: Decoupling
- **Space decoupling**: publisher and subscriber do not need to know each other (by ip address and port for example)
- **Time decoupling**: publisher and subscriber do not need to run at the same time
- **Synchronization decoupling**: operations on both components are not halted during publish or receiving
- Event system as **logically centralized** system
  - anonymous communication
  - possibility to use filters (on headers or entire messages)
  - basic primitives: subscribe, unsubscribe, publish

- **CoAP**
  - CoAP – Constrained Application Protocol.
  - Web transfer protocol for use with constrained nodes and networks.
  - Designed for Machine to Machine (M2M) applications such as smart energy and building automation and Based on Request-Response model between end-points
  - Client-Server interaction is asynchronous over a datagram-oriented transport protocol such as UDP
  - The Constrained Application Protocol (CoAP) is a session layer protocol designed by IETF Constrained RESTful Environment (CoRE) working group to provide lightweight RESTful (HTTP) interface

- Representational State Transfer (REST) is the standard interface between HTTP client and servers.
- Lightweight applications such as those in IoT, could result in significant overhead and power consumption by REST.
- CoAP is designed to enable low-power sensors to use RESTful services while meeting their power constraints
- Built over UDP, instead of TCP (which is commonly used with HTTP) and has a light mechanism to provide reliability.
- CoAP architecture is divided into two main sub-layers:
  - Messaging
  - Request/response

- The messaging sub-layer is responsible for reliability and duplication of messages, while the request/response sub-layer is responsible for communication.
- CoAP has four messaging modes:
  - Confirmable
  - Non-confirmable
  - Piggyback
  - Separate

## CoAP Request-Response Model



- Confirmable and non-confirmable modes represent the reliable and unreliable transmissions, respectively, while the other modes are used for request/response.
- Piggyback is used for client/server direct communication where the server sends its response directly after receiving the message, i.e., within the acknowledgment message.
- On the other hand, the separate mode is used when the server response comes in a message separate from the acknowledgment, and may take some time to be sent by the server
- Similar to HTTP, CoAP utilizes GET, PUT, PUSH, DELETE messages

  requests to retrieve, create, update, and delete, respectively.

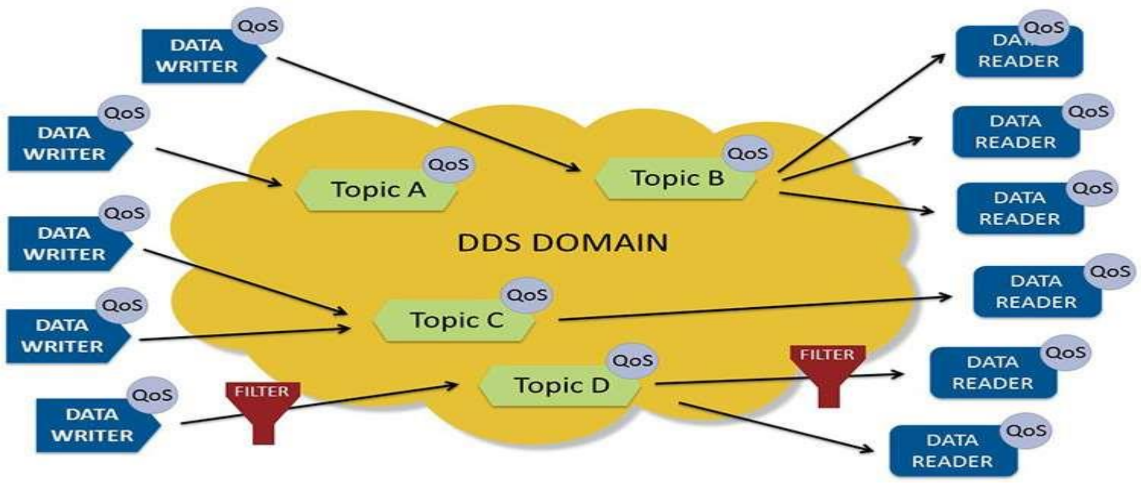
- **XMPP**
  - XMPP – Extensible Messaging and Presence Protocol.
  - A communication protocol for message-oriented middleware based on XML (Extensible Markup Language).
  - Real-time exchange of structured data.
  - It is an open standard protocol
  - XMPP uses a client-server architecture.
  - As the model is decentralized, no central server is required.
  - XMPP provides for the discovery of services residing locally or across a network, and the availability information of these services
  - Well-suited for cloud computing where virtual machines, networks, and firewalls would otherwise present obstacles to alternative service discovery and presence- based solutions.
  - Open means to support machine-to-machine or peer-to-peer communications across a diverse set of networks.

- **Applications:**
  - Publish-subscribe systems.
  - Signaling for VoIP.
  - Video.
  - File transfer.
  - Gaming.
  - Internet of Things applications: Smart grid and Social networking

## AMQP – Advanced Message Queuing Protocol

- **Richer semantic** than MQTT, e.g., supports topics and queues, but also **heavier**, e.g., the broker is much more complex
- AMQP implementations, e.g., Apache Qpid, focus on providing **several features**: queuing, message distribution, security, management, clustering, federation, heterogeneous multi-platform support
  - most of them (possibly) not essential in the IoT scenario
- Originally developed at JPMorgan Chase in London, AMQP was designed as a message-oriented protocol for the integration of enterprise IT components (Enterprise Message Bus)

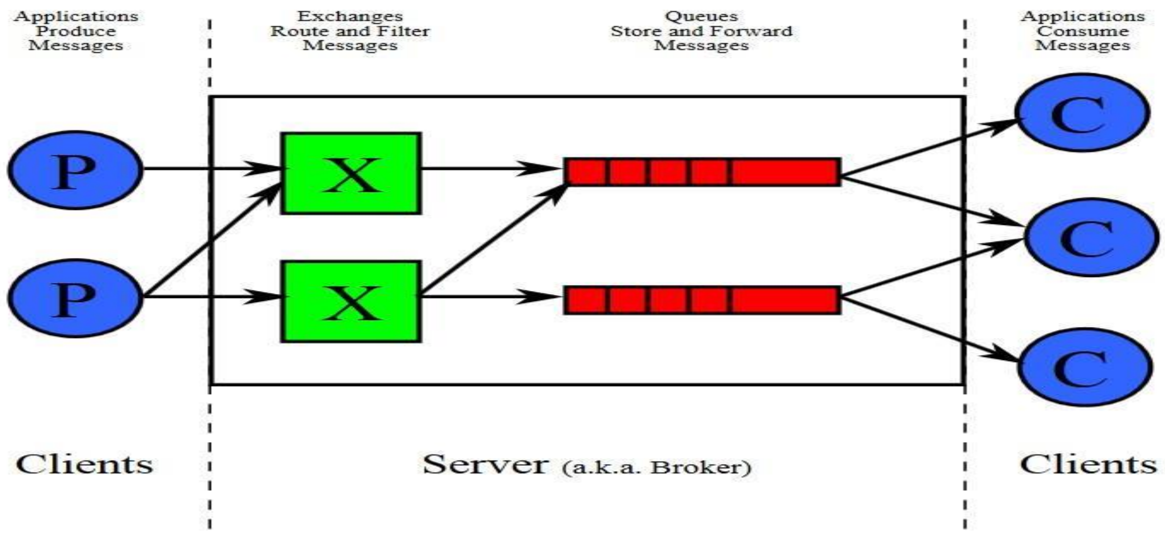
- DDS -Data Distribution Service



- Publish/subscribe, but **broker-less** (based on multicast)
- **scalable**, **real-time**, **dependable**, **high performance** and **interoperable**
- Proposed in several **mission-critical environments** where performance and reliability are essential, e.g., industrial automation, financial applications, air traffic control and management, and even military environments
  - http://twinoakscomputing.com/datasheets/DDS-Brochure.pdf
- Standard developed by the Object Management Group

- **AMQP**
  - Advanced Message Queuing Protocol.
  - Open standard for passing business messages between applications or organizations.
  - Connects between systems and business processes.
  - It is a binary application layer protocol.
  - Basic unit of data is a frame

- **Components**
- **Exchange:**
  - Part of Broker.
  - Receives messages and routes them to Queues.
- **Queue:**
  - Separate queues for separate business processes.
  - Consumers receive messages from queues.
- **Bindings:**
  - Rules for distributing messages (who can access what message, destination of the message)


Applications Produce Messages
Exchanges Route and Filter Messages
Queues Store and Forward Messages
Applications Consume Messages
P
P
X
X
C
C
C
Clients
Server (a.k.a. Broker)
Clients

- **AMQP Features**
  - Targeted QoS (Selectively offering QoS to links).
  - Persistence (Message delivery guarantees).
  - Delivery of messages to multiple consumers.
  - Possibility of ensuring multiple consumption.
  - Possibility of preventing multiple consumption.
  - High speed protocol.

- **Applications**
  - Monitoring and global update sharing.
  - Connecting different systems and processes to talk to each other.
  - Allowing servers to respond to immediate requests quickly and delegate time consuming tasks for later processing.
  - Distributing a message to multiple recipients for consumption.
  - Enabling offline clients to fetch data at a later time.
  - Introducing fully asynchronous functionality for systems.
  - Increasing reliability and uptime of application deployments.