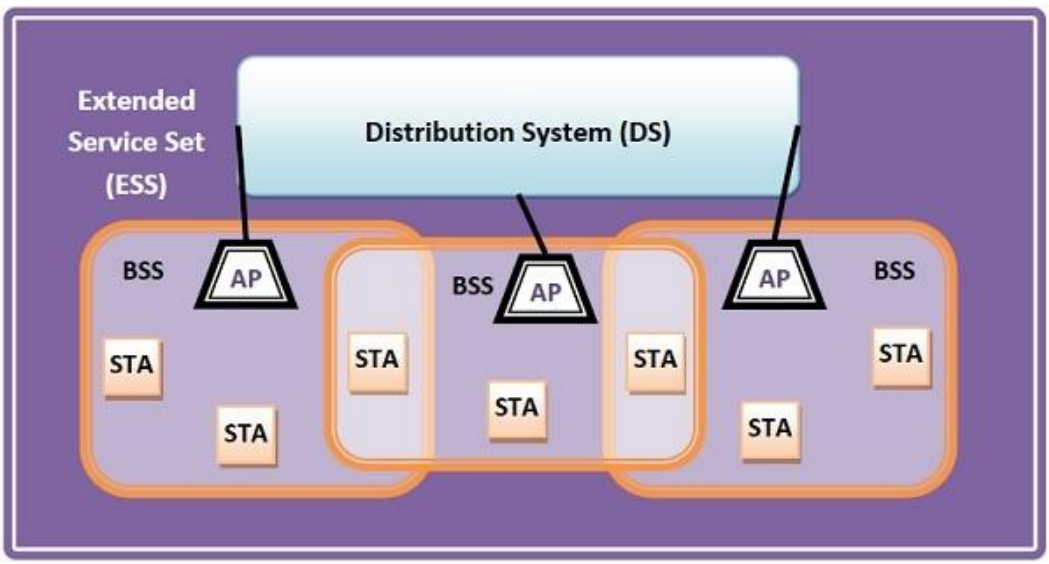
IEEE 802.11 Architecture

The components of an IEEE 802.11 architecture are as follows −

- **Stations (STA)** − Stations comprises of all devices and equipment that are connected to the wireless LAN. A station can be of two types−
  - Wireless Access Point (WAP) − WAPs or simply access points (AP) are generally wireless routers that form the base stations or access.
  - Client. Clients are workstations, computers, laptops, printers, smartphones, etc.
- Each station has a wireless network interface controller.
- **Basic Service Set (BSS)** − A basic service set is a group of stations communicating at the physical layer level. BSS can be of two categories depending upon the mode of operation−
  - Infrastructure BSS − Here, the devices communicate with other devices through access points.
  - Independent BSS − Here, the devices communicate in a peer-to-peer basis in an ad hoc manner.
- **Extended Service Set (ESS)** − It is a set of all connected BSS.
- **Distribution System (DS)** − It connects access points in ESS.



LoRaWAN Architecture

LoRaWAN networks are deployed in a **star-of-stars** topology.

A typical LoRaWAN network consists of the following elements.

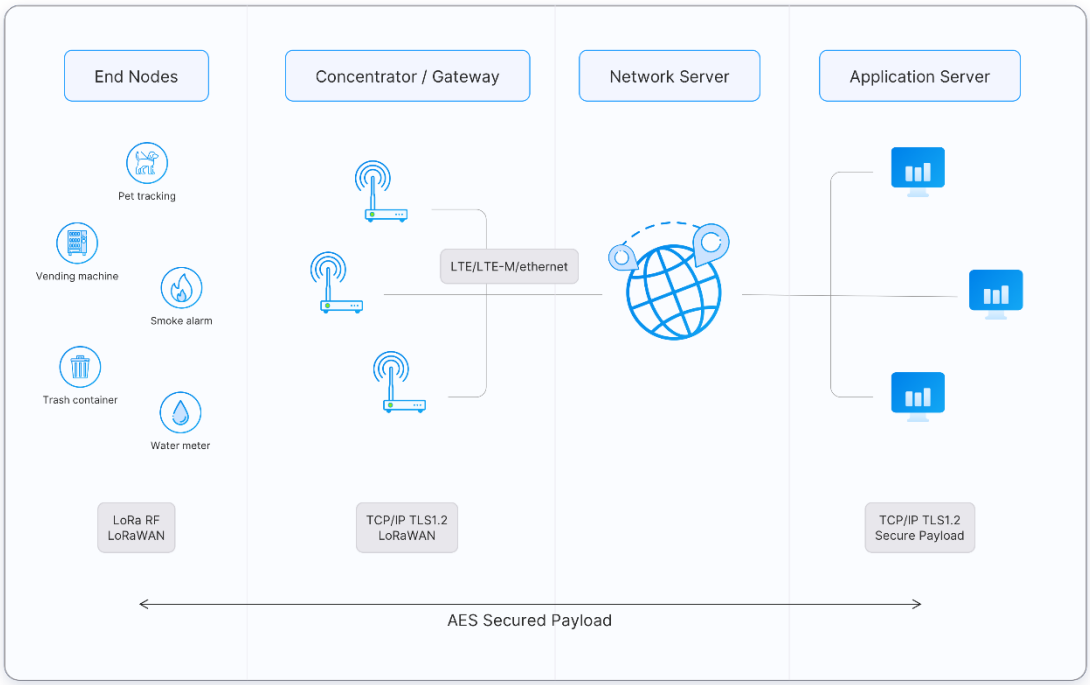


Figure: A typical LoRaWAN network architecture

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

End devices communicate with nearby gateways and each gateway is connected to the network server. LoRaWAN networks use an ALOHA based protocol, so end devices don't need to peer with specific gateways. Messages sent from end devices travel through all gateways within range. These messages are received by the Network Server. If the Network Server has received multiple copies of the same message, it keeps a single copy of the message and discards others. This is known as message deduplication.

Activation

LoRaWAN devices have a 64 bit unique identifier (DevEUI) that is assigned to the device by the chip manufacturer. However, all communication is done with a dynamic 32 bit device address (DevAddr) of which 7 bits are fixed for The Things Network, leaving 25 bits that can be assigned to individual devices, a procedure called **Activation**.

Over-the-Air Activation (OTAA)

Over-the-Air Activation (OTAA) is the preferred and most secure way to connect with The Things Network. Devices perform a join-procedure with the network, during which a dynamic DevAddr is assigned and security keys are negotiated with the device.

Activation by Personalization (ABP)

In some cases you might need to hardcode the DevAddr as well as the security keys in the device. This means activating a device by personalization (ABP). This strategy might seem simpler, because you skip the join procedure, but it has some downsides related to security.

Adaptive Data Rate

Adaptive Data Rate (ADR) is a mechanism for optimizing data rates, airtime and energy consumption in the network.

The ADR mechanism controls the following transmission parameters of an end device.

- Spreading factor
- Bandwidth
- Transmission power

ADR can optimize device power consumption while ensuring that messages are still received at gateways. When ADR is in use, the network server will indicate to the end device that it should reduce transmission power or increase data rate. End devices which are close to gateways should use a lower spreading factor and higher data rate, while devices further away should use a high spreading factor

6LoWPAN

6LoWPAN stands for IPv6 over Low-power Wireless Personal Area Networks. It is a standard protocol for realizing IPv6 communication on wireless networks composed of low-power wireless modules. 6LoWPAN specification contains packet compression and other optimization mechanisms to enable the efficient transmission of IPv6 packets on a network with limited power resources and reliability, which makes efficient IPv6 communication over low-power wireless networks possible.

The 6LoWPAN architecture is made up of low-power wireless area networks (LoWPANs), which are IPv6 subnetwork. It means a LoWPAN is the collection of 6LoWPAN nodes, which share a common IPv6 address prefix (the first 64-bits of an IPv6 address). LoWPAN nodes may play the role of host or router, along with one or more edge routers, as seen in Fig. 1. There are three types of LoWPANs which are Simple LoWPANs, Extended LoWPANs, and Ad hoc LoWPANs [10]. A Simple LoWPAN is connected through one LoWPAN Edge Router to another IP network. An Extended LoWPAN consists of multiple edge routers along with a backbone link to interconnect them. An Ad hoc LoWPAN is not connected to the Internet and operates without an infrastructure.

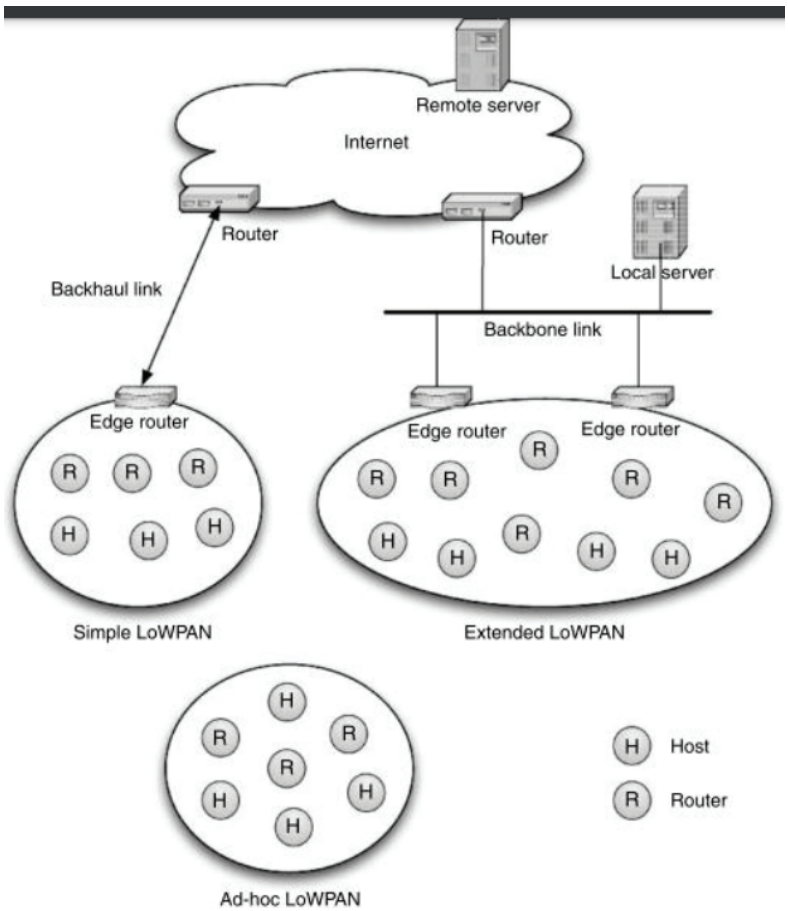


FIGURE 1. 6LoWPAN Architecture

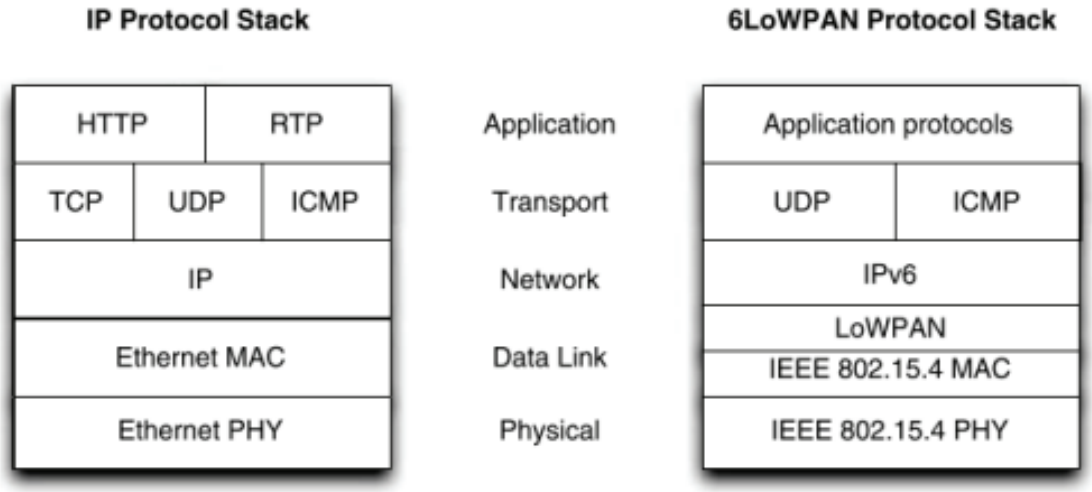


FIGURE 2. 6LoWPAN Protocol Stack

Figure 2 shows the 6LoWPAN protocol stack compared to the IP protocol stack. It is almost identical to a normal IPv6 implementation with the following two differences: x 6LoWPAN only supports IPv6, for which a small adaptation layer (LoWPAN) has been defined to optimize IPv6 over link layers. x Although 6LoWPAN is not bound to the IEEE 802.15.4 standard, it is designed to utilize it.

The MAC layer will add its header (MHR), which will contain the following fields:

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="6">MHR</td></tr>
</table>

Frame Control (2 bytes)	Contains the information about the frame being transmitted
Sequence No. (1 byte)	Sequence number of the outgoing packet. If the Frame control field indicates that the ACK is required for this packet, the receiver will transmit an ACK packet with the same sequence number as the received frame.
Destination PANId (2 bytes)	The PANId of the device to which the packet is destined to.
Destination Address (2/8 bytes)	This field contains the address of MAC destination, the packet is destined for. AD6LoWPAN supports only extended addressing. But if the frame is a broadcast frame, this value is set to 0xFFFF.
Source PANId (2 bytes)	The PANId of the device where the packet originated.
Source Address (8 bytes)	This field contains the MAC address of the source device. AD6LoWPAN supports only extended addressing.

Before handing the packet down to PHY to initiate the transmission, the MAC will add its MFR (MAC footer) which will have the checksum (CRC) of the packet.

MQTT

MQTT (originally an initialism of **MQ Telemetry Transport**) is a lightweight, publish-subscribe, machine to machine network protocol. It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth. It must run over a transport protocol that provides ordered, lossless, bi-directional connections—typically, TCP/IP.[1] It is an open OASIS standard and an ISO recommendation (ISO/IEC 20922).

Before we delve into what is MQTT in IoT is, we need to look at the history of MQTT itself. In 1999, IBM employees invented Message Queuing Telemetry Transport (MQTT) as a messaging protocol. In its initial version, MQTT helped oil pipeline sensors communicate with satellites.

In the following two decades since its inception, MQTT became the go-to messaging protocol used by IoT devices.

In 2014, Oasis accepted MQTT as an ISO standard, thus marking its universal rise as an IoT messaging protocol.

Today, MQTT in IoT helps to connect millions of devices across numerous industries.

How Does MQTT Work – The Publish/Subscribe Architecture

Let us now understand in detail how the MQTT protocol architecture in IoT or the Pub/Sub model is set up. As we know, any IoT network consists of thousands or millions of tiny devices called sensors. In MQTT parlance, these MQTT IoT devices are known as 'clients.'

The clients can be of two types, depending on whether they are transmitting information or receiving information. Clients who transmit information to other devices are called 'publishers.' Clients who receive information transmitted by publishers are called 'subscribers.'

Subscriber client devices can subscribe to any topic of information or even sub-topics. In complex networks, hierarchy structures are present for various topics and sub-topics.

In MQTT, servers called IoT MQTT brokers to transmit information between clients. In practical terms, MQTT brokers are cloud-based servers from platforms such as Amazon Web Services, Microsoft Azure, or Google Cloud. These brokers transmit information seamlessly between publishers and subscribers, thus transmitting information in the entire IoT network.

How Does MQTT work—Message Structure and QoS Levels

MQTT protocols also use a concept called Quality of Service or QoS levels to ensure transmission of messages even during unstable connections between devices. In the MQTT protocol, there are three QoS levels:

1. **QoS 0**: Also called “at most once,” in this level, the message is sent at most once, without any guarantee of delivery. This level is used when connections between devices are stable and the message is not critical.
2. **QoS 1**: Also called “at least once,” in this level, the message is sent repeatedly until the delivery is confirmed by the receiver. This level is used to transmit critical messages whose delivery needs to be guaranteed.
3. **QoS 2**: Also called “exactly once,” in this level, the message is sent only once. Guaranteed delivery requires technical coordination between sender and receiver, and thus this level is more energy-consuming than the other two levels.

Advantages of MQTT

MQTT protocol is essentially a lightweight IoT protocol with several advantages:

1. **Lightweight and efficient**: MQTT protocol requires a minimal amount of code and consumes very less power. The MQTT protocol is thus energy-efficient and easy to deploy for millions of devices.
2. **Connecting devices during unreliable networks**: MQTT in IoT uses QoS levels to ensure guaranteed delivery of messages to receivers, even when connections between devices are unreliable.
3. **Enabling communication between the cloud and devices**: MQTT protocol for IoT ensures speedy communication between cloud servers and IoT devices in remote areas.
4. **Last will feature**: If any IoT device disconnects unexpectedly, MQTT protocol uses the last will feature to broadcast a relevant message to other IoT devices in the network.
5. **Extensive programming support**: MQTT has extensive support in programming languages such as Python, making it very easy for developers to use it.

How is MQTT Used in IoT?

Let us now look at MQTT IoT examples.

1. **Automotive**: MQTT IoT projects in the automotive sector enable vehicle theft prevention, vehicle monitoring, and remote maintenance of vehicles.

2. **Logistics**: One of the best MQTT protocols in the IoT example is in the logistics sector. MQTT IoT hubs such as Airtel IoT help track freight vehicles and provide real-time alerts for freight safety and movement.
3. **Energy**: IoT MQTT panels in the energy sector help to build a smarter energy grid and optimise power consumption by consumers.
4. **Home automation**: IoT dashboards use MQTT to manage home devices instantly with your mobile phones.