IP as the IoT Network Layer

This chapter is composed of the following sections:

The Business Case for IP: This section discusses the advantages of IP from an IoT perspective and introduces the concepts of adoption and adaptation.

The Need for Optimization: This section dives into the challenges of constrained nodes and devices when deploying IP. This section also looks at the migration from IPv4 to IPv6 and how it affects IoT networks.

Optimizing IP for IoT: This section explores the common protocols and technologies in IoT networks utilizing IP, including 6LoWPAN, 6TiSCH, and RPL.

Profiles and Compliances: This section provides a summary of some of the most significant organizations and standards bodies involved with IP connectivity and IoT.

The Business Case for IP

Data flowing from or to "things" is consumed, controlled, or monitored by data centre servers either in the cloud or in locations that may be distributed or centralized.

IP was not only preferred in the IT markets but also for the OT environment.

The Key Advantages of Internet Protocol

IP has largely demonstrated its ability to integrate small and large evolutions. At the same time, it is able to maintain its operations for large numbers of devices and users.

key advantages of the IP suite for the Internet of Things:

Open and standards-based:

- The Internet of Things creates a new paradigm in which devices, applications, and users can leverage a large set of devices and functionalities while guaranteeing interchangeability and interoperability, security, and management. This calls for implementation, validation, and deployment of open, standards-based solutions.

- standards development organizations (SDOs) are working on Internet of Things definitions, frameworks, applications, and technologies, none are questioning the role of the Internet Engineering Task Force (IETF) as the foundation for specifying and optimizing the network and transport layers.

Versatile:

- A large spectrum of access technologies is available to offer connectivity of "things" in the last mile. Additional protocols and technologies are also used to transport IoT data through backhaul links and in the data center.
- solution layered IP architecture is well equipped to cope with any type of physical and data link layers. This makes IP ideal as a long-term investment because various protocols at these layers can be used in a deployment now and over time, without requiring changes to the whole architecture and data flow.

Ubiquitous:

- All recent operating system releases have an integrated dual (IPv4 and IPv6) IP stack that gets enhanced over time.
- IoT application protocols in many industrial OT solutions have been updated in recent years to run over IP.

Scalable:

- adding huge numbers of "things" to private and public infrastructures may require optimizations and design rules specific to the new devices. However, you should realize that this is not very different from the recent evolution of voice and video endpoints integrated over IP. IP has provenbefore that scalability is one of its strengths..

Manageable and highly secure:

- Communications infrastructure requires appropriate management and security capabilities for proper operations.

- Adopting IP network management also brings an operational business application to OT. Well known network and security management tools are easily leveraged with an IP network layer.
- the industry ischallenged in securing constrained nodes, handling legacy OT protocols,and scaling operations.

Stable and resilient:

- IP has a large and well-established knowledge, been deployed for critical services, such as voice and video, and transitioned from closed environments to open IP standards.
- its stability and resiliency benefit from the large ecosystem of IT professionals who can help design, deploy, and operate IP-based solutions.

Consumers' market adoption

- When developing IoT solutions and products targeting the consumer market, vendors know that consumer's access to applications and devices will occur predominantly over broadband and mobile wireless infrastructure.

The innovation factor:

- The past two decades have largely established the adoption of IP as a factor for increased innovation. IP is the underlying protocol for applications ranging from file transfer and e-mail to the World Wide Web, e-commerce, social networking, mobility, and more. Even the recent computing evolution from PC to mobile and mainframes to cloud services are perfect demonstrations of the innovative ground enabled by IP Innovations in IoT can also leverage an IP underpinning.

Adoption or Adaptation of the Internet Protocol

The use of numerous network layer protocols in addition to IP is often a point of contention between computer networking experts. Typically, one of two models, adaptation or adoption, is proposed:

- *Adaptation* means application layered gateways (ALGs) must be implemented to ensure the translation between non-IP and IP layers.
- *Adoption* involves replacing all non-IP layers with their IP layer counterparts, simplifying the deployment model and operations.

In various industries to see how IP adaptation and adoption are currently applied to IoT last-mile connectivity.

Supervisory control and data acquisition (SCADA) applications:

- Implementations that make use of IP adaptation have SCADA devices attached through serial interfaces to a gateway tunnelling or translating the traffic.
- With the IP adoption model, SCADA devices are attached via Ethernet to switches and routers forwarding their IPv4 traffic.

The following factors when trying to determine which model is best suited for last-mile connectivity:

Bidirectional versus unidirectional data flow:

- While bidirectional communications are generally expected, some last-mile technologies offer optimization for unidirectional communication.
- RFC 7228only infrequently need to report a few bytes of data to an application. These sorts of devices, particularly ones that communicate through LPWA technologies
- Implementing a full IP stack. Is not required if there is only one-way communication to upload data to an application, then it is not possible to download new software or firmware to the devices. This makes integrating new features and bug and security fixes more difficult.

Overhead for last-mile communications paths:

- IP adoption varies depending on the IP version. IPv4 has 20 bytes of header at a minimum, and IPv6 has 40 bytes at the IP network layer. For the IP transport layer, UDP has 8 bytes of header overhead, while TCP has a minimum of 20 bytes.

Data flow model:

- One benefit of the IP adoption model is the end-to-end nature of communications. the adaptation model can work because translation of traffic needs to occur only between the end device and one or two application servers. Depending on the network topology and the data flow needed, both IP adaptation and adoption models have roles to play in last-mile connectivity.

Network diversity:

- One of the drawbacks of the adaptation model is a general dependency on single PHY and MAC layers. For example, ZigBee devices must only be deployed in ZigBee network islands.

The Need for Optimization

IOT BULIT ON IP but integration with non-IP devices, need to deal with the limits at the device and network levels that IoT often imposes.

optimizations are needed at various layers of the IP stack to handle the restrictions that are present in IoT networks. Both the nodes and the network

Constrained Nodes.

- Another limit is that this network protocol stack on an IoT node may be required to communicate through an unreliable path. Even if a full IP stack is available on the node, this causes problems such as limited or unpredictable throughput and low convergence when a topology change occurs.

- power consumption is a key characteristic of constrained nodes IoT devices are battery powered, with lifetime battery requirements varying from a few months to 10+ years.
- You should also be aware that power consumption requirements on battery powered nodes impact communication intervals. To help extend battery life, you could enable a "low-power" mode instead of one that is "always on." Another option is "always off," which means communications are enabled only when needed to send data.
- IoT constrained nodes can be classified as follows:

- **Devices that are very constrained in resources, may communicateinfrequently to transmit a few bytes, and may have limited security and management capabilities:** This drives the need for the IP adaptation model, where nodes communicate through gateways and proxies.
- **Devices with enough power and capacities to implement a stripped down IP stack or non-IP stack:** In this case, you may implement either an optimized IP stack and directly communicate with application servers (adoption model) or go for an IP or non-IP stack and communicate through gateways and proxies (adaptation model).
- **Devices that are similar to generic PCs in terms of computing and power resources but have constrained networking capacities, such as bandwidth:** These nodes usually implement a full IP stack (adoption model), but network design and application behaviors must cope with the bandwidth constraints.

Constrained Networks

- network bandwidth capacity was restrained due to technical limitations. Connections often depended on low-speed modems for transferring data. However, these low-speed connections demonstrated that IP could run over low-bandwidth networks.

- high-speed connections are not usable by some IoT devices in the last mile. The reasons include the implementation of technologies with low bandwidth, limited distance and bandwidth due to regulated transmit power, and lack of or limited network services.
- A constrained network can have high latency and a high potential for packet loss and have unique characteristics and requirements.
- Limited bandwidth, it is not unusual for the packet delivery rate (PDR) to oscillate between low and high percentages..
- Large bursts of unpredictable errors and even loss of connectivity at times may occur. These behaviours can be observed on both wireless and narrowband power-line communication links, where packet delivery variation may fluctuate greatly during the course of a day.
- Control plane traffic must also be kept at a minimum; otherwise, it consumes the bandwidth that is needed by the data traffic.

IP Versions

The IETF has been working on transitioning the Internet from IP version 4 to IP version 6.

The following are some of the main factors applicable to IPv4 and IPv6 support in an IoT solution:

Application Protocol:

- IoT devices implementing Ethernet or Wi-Fi interfaces can communicate over both IPv4 and IPv6, but the application protocol may dictate the choice of the IP version. For example, SCADA protocols such as DNP3/IP (IEEE 1815), Modbus TCP, or the IEC 60870- 5-104 standards are specified only for IPv4 . For IoT devices with application protocols defined by the IETF, such as HTTP/HTTPS, CoAP, MQTT, and XMPP, both IP versions are supported. The selection of the IP version is only dependent on the implementation.

Cellular Provider and Technology:

- IoT devices with cellular modems are dependent on the generation of the cellular technology as well as the data services offered by the provider. For the first three generations of data services—GPRS, Edge, and 3G—IPv4 is the base protocol version. Consequently, if IPv6 is used with these generations, it must be tunnelled over IPv4. On 4G/LTE networks, data services can use IPv4 or IPv6 as a base protocol, depending on the provider.

Serial Communications:

- communicate through serial lines. Data is transferred using either proprietary or standards-based protocols, such as DNP3, Modbus, or IEC 60870-5-101. In the past, communicating this serial data over any sort of distance could be handled by an analog modem connection.
- However, as service provider support for analog line services has declined, the solution for communicating with these legacy devices has been to use local connections. To make this work, you connect the serial port of the legacy device to a nearby serial port on a piece of communications equipment, typically a router. This local router then forwards the serial traffic over IP to the central server for processing.

IPv6 Adaptation Layer:

IPv6-only adaptation layers for some physical and data link layers for recently standardized IoT protocols support only IPv6. While the most common physical and data link layers (Ethernet, Wi- Fi, and so on) stipulate adaptation layers for both versions, newer technologies, such as IEEE 802.15.4 (Wireless Personal Area Network), IEEE 1901.2, and ITU G.9903 (Narrowband Power Line Communications) only have an IPv6 adaptation layer specified This means that any device implementing a technology that requires an IPv6 adaptation layer must communicate over an IPv6-only sub network. This is reinforced by the IETF routing protocol for LLNs, RPL, which is IPv6 only.

Optimizing IP for IoT

- constrained nodes and constrained networks mandate optimization at various layers and on multiple protocols of the IP architecture.

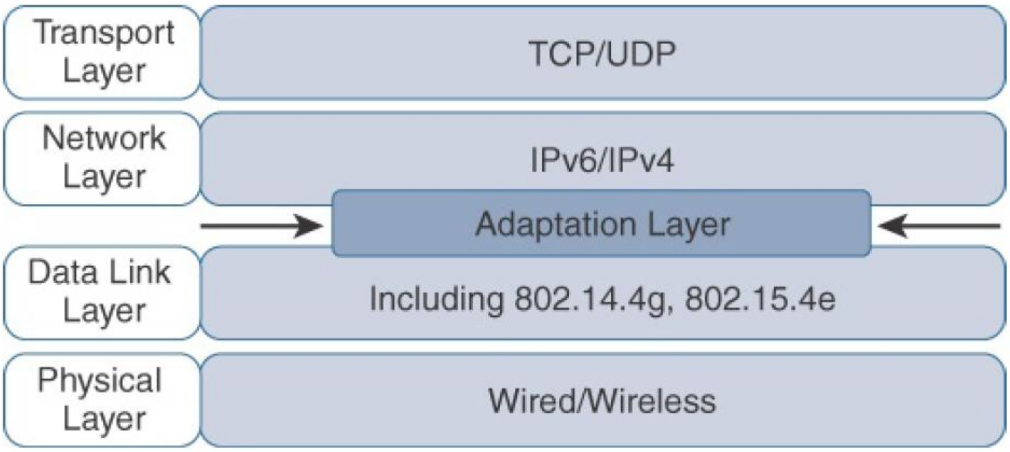


Figure 5-1 *Optimizing IP for IoT Using an Adaptation Layer*

From 6LoWPAN to 6Lo

- In the IP architecture, the transport of IP packets over any given Layer 1 (PHY) and Layer 2 (MAC) protocol must be defined and documented. The model for packaging IP into lower-layer protocols is often referred to as an *adaptation layer*.
- RFC(request for comments) is a publication from the IETF that officially documents Internet standards, specifications, protocols, procedures, and events.
- RFC 864 describes how an IPv4 packet gets encapsulated over an Ethernet frame
- adaptation layers optimized for constrained nodes or "things" are the ones under the 6LoWPAN working group and its successor, the 6Lo working group. The initial focus of the 6LoWPAN working group was to optimize the transmission of IPv6 packets over constrained networks such as IEEE 802.15.4.

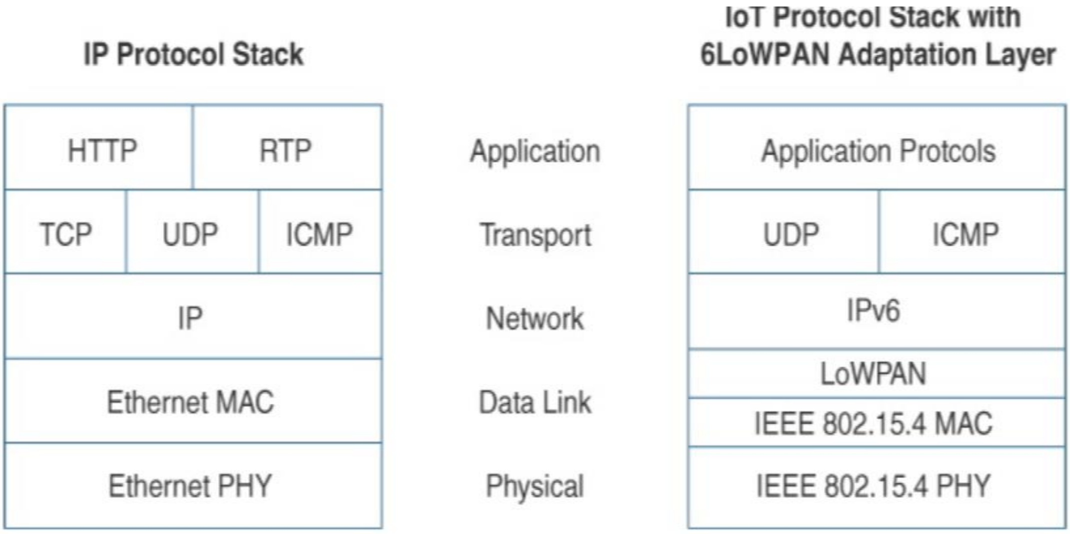


Figure 5-2 *Comparison of an IoT Protocol Stack Utilizing 6LoWPAN and an IP Protocol Stack*

- RFC 4994:it defines frame headers for the capabilities of header compression, fragmentation, and mesh addressing headers can be stacked in the adaptation layer

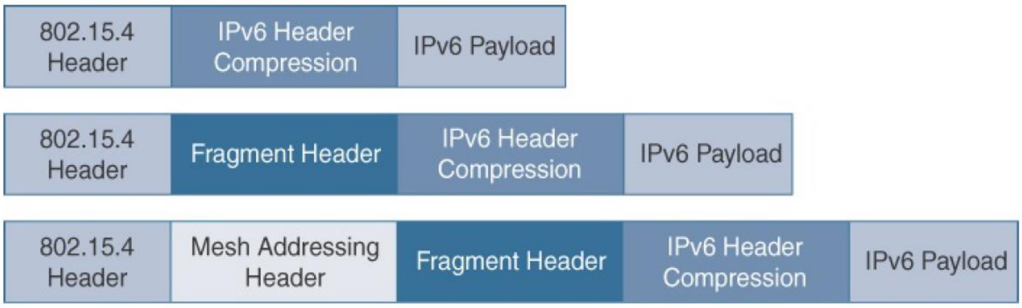


Figure 5-3 *6LoWPAN Header Stacks*

Header Compression

- IPv6 header compression for 6LoWPAN was defined initially in RFC 4944shrinks the size of IPv6's 40-byte headers and User Datagram Protocol's (UDP's) 8-byte headers down as low as 6 bytes combined in some cases

- 6LoWPAN header compression is stateless, and conceptually it is not too complicated.
- factors affect the amount of compression, such as implementation of RFC 4944 versus RFC 6922, whether UDP is included, and various IPv6 addressing scenarios.

- 6LoWPAN takes advantage of shared information known by all nodes from their participation in the local network.
- It omits some standard header fields by assuming commonly used values

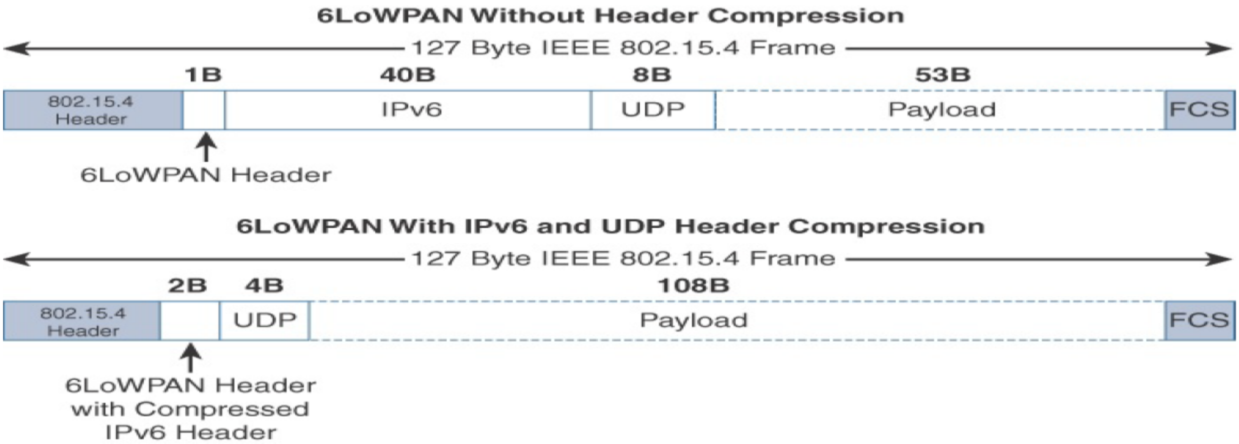


Figure 5-4 *6LoWPAN Header Compression*

- Notice that uncompressed IPv6 and UDP headers leave only 53 bytes of data payload out of the 127-byte maximum frame size in the case of IEEE 802.15.4.
- The 6LoWPAN header increases to 2 bytes to accommodate the compressed IPv6 header, and UDP has been reduced in half, to 4 bytes from 8. Most importantly, the header compression has allowed the payload to more than double, from 53 bytes to 108 bytes,
- which is obviously much more efficient.
- Note that the 2-byte header compression applies to intra cell communications, while communications external to the cell may require some field of the header to not be compressed.

Fragmentation

- The maximum transmission unit (MTU) for an IPv6 network must be at least 1280 bytes.
- The term *MTU* defines the size of the largest protocol data unit that can be passed.
- For IEEE 802.15.4, 127 bytes is the MTU.

- a problem of IPv6, with a much larger MTU, is carried inside the 802.15.4 frame with a much smaller one.
- To remedy this situation, large IPv6 packets must be fragmented across multiple 802.15.4 frames at Layer 2.

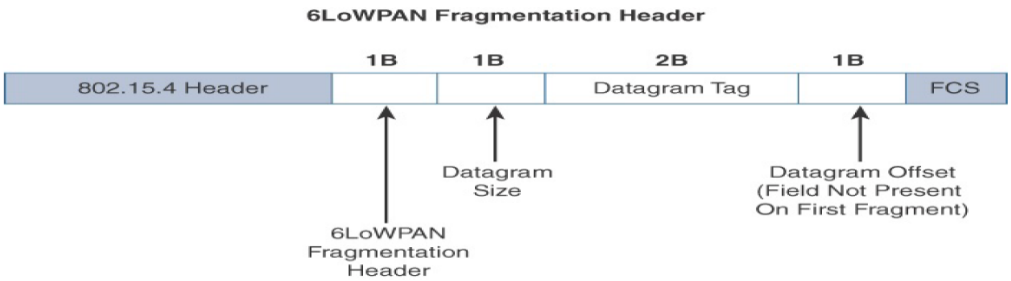


Figure 5-5 *6LoWPAN Fragmentation Header*

- Three Field
- The 1-byte Datagram Size field specifies the total size of the unfragmented payload.
- Datagram Tag identifies the set of fragments for a payload.
- the Datagram Offset field defines how far into a payload a particular fragment occurs.
- uses a unique bit value to identify that the subsequent fields behind it are fragment fields as opposed to another capability,

Mesh Addressing

- mesh addressing function is to forward packets over multiple hops.
- Three fields

1. Hop Limit:provides an upper limit on how many times the frame can be forwarded. Each hop decrements this value by 1 as it is forwarded. Once the value hits 0, it is dropped and no longer forwarded.

2 The Source Address and

3 Destination Address fields for mesh addressing are IEEE 802.15.4 addresses indicating the endpoints of an IP hop

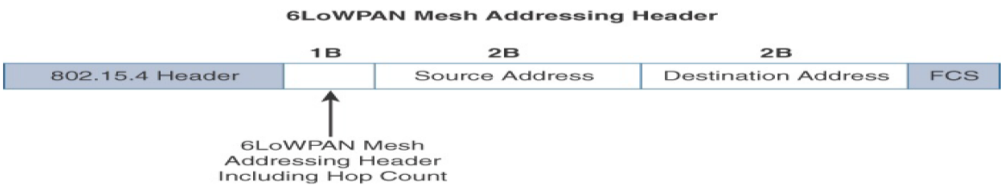


Figure 5-6 *6LoWPAN Mesh Addressing Header*

mesh addressing header is used in a single IP subnet and is a Layer 2 type of routing known as mesh-under.

Mesh-Under Versus Mesh-Over Routing

two main options exist for establishing reachability and forwarding packets.

mesh-under, the routing of packets is handled at the 6LoWPAN adaptation layer.

“mesh-over” or “route-over,” utilizes IP routing for getting packets to their destination. With mesh-under routing, the routing of IP packets leverages the 6LoWPAN

mesh addressing header

The term *mesh-under* is used because multiple link layer hops can be used to complete a single IP hop. Nodes have a Layer 2 forwarding table that they consult to route the packets to their final destination within the mesh. An edge gateway terminates the mesh-under domain. The edge gateway must also implement a mechanism to translate between the configured Layer 2 protocol and any IP routing mechanism implemented on other Layer 3 IP interfaces.

In mesh-over or route-over scenarios, IP Layer 33 routing is utilized for computing reachability and then getting packets forwarded to their destination, either inside or outside the mesh domain. Each full-functioning node acts as an IP router, so each link layer hop is an IP hop. When a LoWPAN has been implemented using different link layer technologies, a mesh-over routing setup is useful. While traditional IP routing protocols can be used, a specialized routing protocol for smart objects, such as RPL, is recommended.

6Lo Working Group

focus on IPv6 connectivity over constrained-node networks

IPv6-over-foo adaptation layer specifications using 6LoWPAN technologies (RFC4944, RFC6282, RFC6775) for link layer

technologies: For example, this includes:

IPv6 over Bluetooth Low Energy

Transmission of IPv6 packets over near-field communication

IPv6 over 802.11ah

Transmission of IPv6 packets over DECT Ultra Low Energy

Transmission of IPv6 packets on WIA-PA (Wireless Networks for

Industrial Automation–Process Automation)

Transmission of IPv6 over Master Slave/Token Passing (MS/TP)

Information and data models such as MIB modules: One example is RFC 7388, "Definition of Managed Objects for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)."

Optimizations that are applicable to more than one adaptation layer

specification: For example, this includes RFC 7400, "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal

Area Networks (6LoWPANs)."

Informational and maintenance publications needed for the IETF specifications in this area

6TiSCH

- To standardize IPv6 over the TSCH(Time-Slotted Channel Hopping) mode of IEEE 802.15.4e (known as 6TiSCH), the IETF formed the 6TiSCH working group provide architecture, information model, and minimal 6TiSCH configuration, leveraging and enhancing work done by the 6LoWPAN
- An important element specified by the 6TiSCH working group is 6top, a sublayer that glues together the MAC layer and 6LoWPAN adaptation layer.

- This sublayer provides commands to the upper network layers, such as RPL. In return, these commands enable functionalities including network layer routing decisions, configuration, and control procedures for 6TiSCH schedule management.

- The IEEE 802.15.4e standard defines a time slot structure not provide scheduling Algorithm
- Schedules in 6TiSCH are broken down into cells. A cell is simply a single element in the TSCH schedule that can be allocated for unidirectional or bidirectional communications between specific nodes

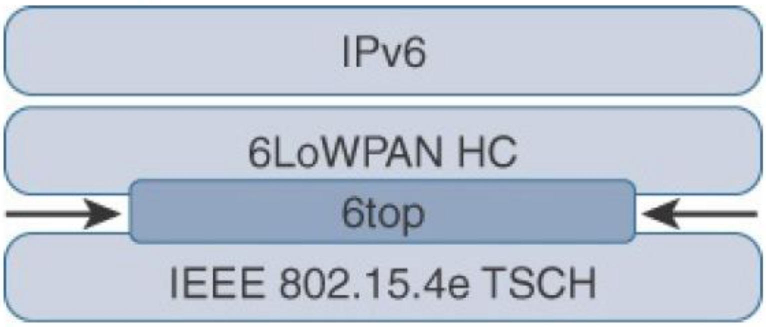


Figure 5-7 *Location of 6TiSCH's 6top Sublayer*

The 6TiSCH architecture defines four schedule management mechanisms:

Static scheduling: All nodes in the constrained network share a fixed schedule. Cells are shared, and nodes contend for slot access in a slotted aloha manner. Slotted aloha is a basic protocol for sending data using time slot boundaries when communicating over a shared medium

Neighbor-to-neighbor scheduling: A schedule is established that correlates with the observed number of transmissions between nodes. Cells in this schedule can be added or deleted as traffic requirements and bandwidth needs change.

Remote monitoring and scheduling management: Time slots and other resource allocation are handled by a management entity that can be

multiple hops away. The scheduling mechanism leverages 6top

Hop-by-hop scheduling: A node reserves a path to a destination nodemultiple hops away by requesting the allocation of cells in a schedule ateach intermediate node hop in the path. The protocol not used There are three 6TiSCH forwarding models:

Track Forwarding (TF): This is the simplest and fastest forwarding model. A “track” in this model is a unidirectional path between a source and a destination. This track is constructed by pairing bundles of receive cells in a schedule with a bundle of receive cells set to transmit.

Fragment forwarding (FF): This model takes advantage of 6LoWPAN fragmentation to build a Layer 2 forwarding table

IPv6 Forwarding (6F): This model forwards traffic based on its IPv6 routing table. Flows of packets should be prioritized by traditional QoS (quality of service) and RED (random early detection) operations.

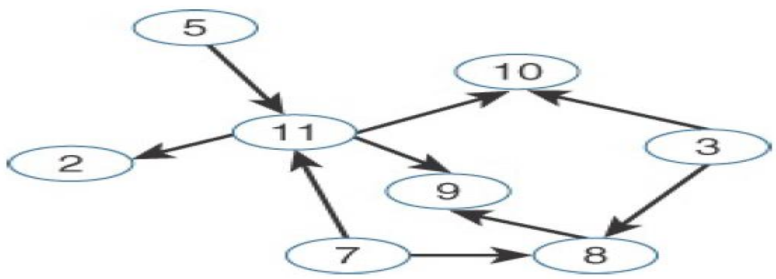
RPL

- Layer 3 IP routing protocols and determine the needs and requirements for developing a routing solution for IP smart objects.
- new distance-vector routing protocol was named the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)(RFC 6550)

- In an RPL network, each node acts as a router and becomes part of a mesh network. Routing is performed at the IP layer.
- Each node examines every received IPv6 packet and determines the next-hop destination based on the information contained in the IPv6 header.
- characteristics of constrained nodes, the protocol defines two modes:
- **Storing mode:** All nodes contain the full routing table of the RPL domain. Every node knows how to directly reach every other node.

Non-storing mode: Only the border router(s) of the RPL domain contain(s) the full routing table. All other nodes in the domain only maintain their list of parents and use this as a list of default routes toward the border router.

RPL is based on the concept of a directed acyclic graph (DAG). A DAG is a directed graph where no cycles exist. A DESTINATION ORIENTED DODAG is a DAG rooted to one destination



e 5-8 *Example of a Directed Acyclic Graph (DAG)*

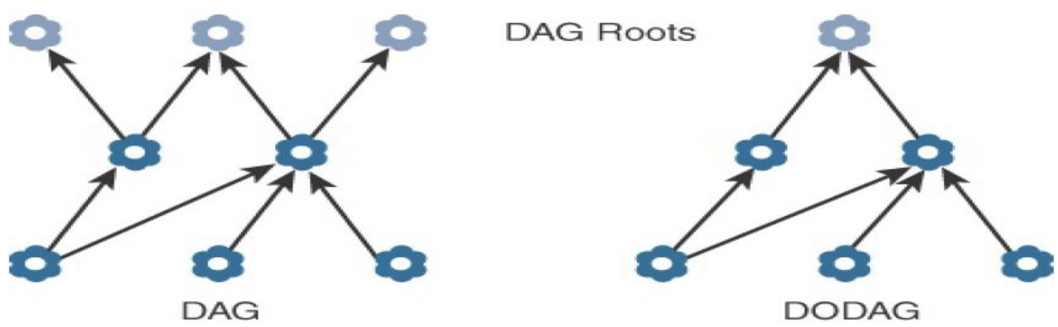


Figure 5-9 *DAG and DODAG Comparison*

In a DODAG

1 each node maintains up to three parents, preferred parent PROVIDE next hop for upward routes toward the root.

2 The routing graph created by the set of DODAG parents across all nodes defines the full set of upward routes. routes are loop free by disallowing nodes

3. Upward routes in RPL are discovered and configured using DAG Information Object (DIO) messages ,Nodes listen to DIOs to handle changes in the topology

4 Nodes establish downward routes by advertising their parent set toward the DODAG root using a Destination Advertisement Object (DAO) message.

5. In the case of the non-storing mode of RPL, nodes sending DAO messages report their parent sets directly to the DODAG root (border router), and only theroot stores the routing information.

6. For storing mode, each node keeps track of the routing information that is advertised in the DAO messages. Nodes can make their own routing decisions

RPL messages, such as DIO and DAO, run on top of IPv6. These messages exchange and advertise downstream and upstream routing information between a border router and the nodes under it

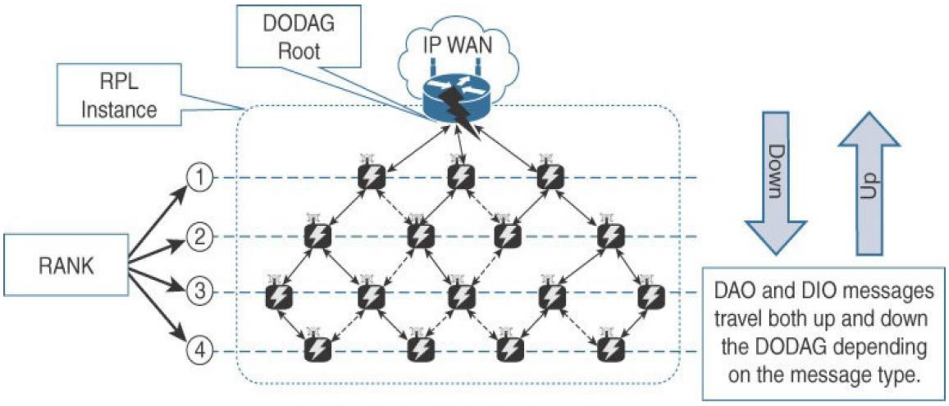


Figure 5-10 *RPL Overview*

Objective Function (OF) defines how metrics are used to select routes and establish a node's rank. Standards such as RFC 6552 and 6719 have been published to document OFs specific to certain use cases and node types.

Rank

The rank is a rough approximation of how "close" a node is to the root and helps avoid routing loops and the count-to-infinity problem. Nodes can only increase their rank when receiving a DIO message with a larger version number.

RPL Headers

RPL Headers

Specific network layer headers are defined for datagrams being forwarded within an RPL domain An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)." The purpose of this header is to leverage data-plane packets for loop detection in a RPL instance.

Metrics

RPL defines a large and flexible set of new metrics and constraints for routing in RFC 6551. RPLrouting metrics and constraints defined in RFC 6551 include the following:

Expected Transmission Count (ETX): Assigns a discrete value to the number of transmissions a node expects to make to deliver a packet.

Hop Count: Tracks the number of nodes traversed in a path. Typically, a path with a lower hop count is chosen over a path with a higher hop count.

Latency: Varies depending on power conservation. Paths with a lower latency are preferred.

Link Quality Level: Measures the reliability of a link by taking into account packet error rates caused by factors such as signal attenuation and interference.

Link Color: Allows manual influence of routing by administratively setting values to make a link more or less desirable. These values can be either statically or dynamically adjusted for specific traffic types.

Node State and Attribute: Identifies nodes that function as traffic aggregators and nodes that are being impacted by high workloads. High workloads could be indicative of nodes that have incurred high CPU or low memory states. Naturally, nodes that are aggregators are preferred over nodes experiencing high workloads.

Node Energy: Avoids nodes with low power, so a battery-powered node that is running out of energy can be avoided and the life of that node and the network can be prolonged.

Throughput: Provides the amount of throughput for a node link. Often, nodes conserving power use lower throughput. This metric allows the prioritization of paths with higher throughput

Authentication and Encryption on Constrained Nodes

IETF working groups that are focused on their security: ACE and DICE.

ACE

Much like the RoLL working group, the Authentication and Authorization for Constrained Environments (ACE) working group is tasked with evaluating the applicability of existing authentication and authorization protocols and documenting their suitability for certain constrained-environment use cases. Once the candidate solutions are validated, the ACE working group will focus its work on CoAP with the Datagram Transport Layer Security (DTLS) protocol.

DICE

New generations of constrained nodes implementing an IP stack over constrained access networks are expected to run an optimized IP protocol stack. For example, when implementing UDP at the transport layer, the IETF Constrained Application Protocol (CoAP) should be used at the application layer.

In constrained environments secured by DTLS CoAP can be used to control resources on a device

working group is to define an optimized DTLS profile for constrained nodes.

Application Protocols for IoT

Application protocols that are sufficient for generic nodes and traditional networks often are not well suited for constrained nodes and networks.

The Transport Layer: IP-based networks use either TCP or UDP. However, the constrained nature of IoT networks requires a closer look at the use of these traditional transport mechanisms.

IoT Application Transport Methods: This section explores the various types of IoT application data and the ways this data can be carried across a network. Selection of a protocol for the transport layer as supported by the TCP/IP architecture in the context of IoT networks

Transmission Control Protocol (TCP): This connection-oriented protocol requires a session to get established between the source and destination before exchanging data.

User Datagram Protocol (UDP): With this connectionless protocol, data can be quickly sent between source and destination—but with no guarantee of delivery. This is analogous to the traditional mail delivery system, in which a letter is mailed to a destination. Confirmation of the reception of this letter does not happen until another letter is sent in response.

TCP is the main protocol used at the transport layer.

- its ability to transport large volumes of data into smaller sets of packets. In addition, it ensures reassembly in a correct sequence, flow control and window adjustment, and retransmission of lost packets.
- These benefits occur with the cost of overhead per packet and per session, potentially impacting overall packet per second performances and latency.

UDP

- Domain Name System (DNS), Network Time Protocol (NTP), Simple Network Management Protocol (SNMP), and Dynamic Host Control Protocol (DHCP), or for real-time data traffic, including voice and video over IP.

- IoT nodes may also be limited by the intrinsic characteristics of the data link layers. For example, low-power and lossy networks (LLNs). Constrained Application Protocol (CoAP), almost always uses UDP implementations of industrial application layer protocols may call for the optimization and adoption of the UDP transport layer if run over LLNs

- example, the Device Language Message Specification/Companion Specification for Energy Metering (DLMS/COSEM) application layer protocol

If you compare the transport of DLMS/COSEM over a cellular network versus an LLN deployment, you should consider the following:

- Select TCP for cellular networks because these networks are typically more robust and can handle the overhead. For LLNs, where both the devices and network itself are usually constrained, UDP is a better choice and often mandatory.
- DLMS/COSEM can reduce the overhead associated with session establishment by offering a "long association" over LLNs. *Long association* means that sessions stay up once in place because the communications overhead necessary to keep a session established is much less than is involved in opening and closing many separate sessions over the same time period. Conversely, for cellular networks, a short association better controls the costs by tearing down the open associations after transmitting.
- When transferring large amounts of DLMS/COSEM data, cellular links are preferred to optimize each open association. Smaller amounts of data can be handled efficiently over LLNs. Because packet loss ratios are generally higher on LLNs than on cellular networks, keeping the data transmission amounts small over LLNs limits the retransmission of large numbers of bytes.

IoT Application Transport Methods

IoT application protocols, there are various means for transporting these protocols across a network. Dealing with legacy utility and industrial IoT protocols that have certain requirements

The following categories of IoT application protocols and their transport methods are explored in the following sections:

Application layer protocol not present: In this case, the data payload is directly transported on top of the lower layers. No application layer protocol is used.

Supervisory control and data acquisition (SCADA): SCADA is one of the most common industrial protocols in the world, but it was developed long before the days of IP, and it has been adapted for IP networks.

Generic web-based protocols: Generic protocols, such as Ethernet, Wi- Fi, and 4G/LTE, are found on many consumer- and enterprise-class IoT devices that communicate over non-constrained networks.

IoT application layer protocols: IoT application layer protocols are devised to run on constrained nodes with a small compute footprint and are well adapted to the network bandwidth constraints on cellular or satellite links or constrained 6LoWPAN networks. Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP), covered later in this chapter, are two well-known examples of IoT application layer protocols.

Application Layer Protocol Not Present

- IETF RFC 7228 devices defined as class 0 send or receive only a few bytes of data.
- Class 0 devices are usually simple smart objects thatare severely constrained. Implementing a robust protocol stack is usually not useful and sometimes not even possible with the limited available resources.
- low-cost temperature and relative humidity (RH) sensors sending data over an LPWA LoRaWAN infrastructure Temperature is represented as 2 bytes and RH as another 2 bytes of data.

Example 6-1 *Decoding Temperature and Relative Humidity Sensor Data*

Click here to view code image

```
Temperature data payload over the network: Tx = 0x090c
Temperature conversion required by the application
T = Tx/32 - 50  to  T = 0x090c/32 - 50  to  T = 2316/32 - 50 = 22.4°
RH data payload over the network: RHx = 0x062e
RH conversion required by the application:
100RH = RHx/16-24  to 100RH = 0x062e/16-24 = 74.9  to RH = 74.9%
```

Imagine expanding Example 6-1 to different kinds of temperature sensors from different manufacturers. These sensors will report temperature data in varying formats. A temperature value will always be present in the data transmitted by each sensor, but decoding this data will be vendor specific. If you scale this scenario out across hundreds or thousands of sensors, the problem of allowing various applications to receive and interpret temperature values delivered in different formats becomes increasingly complex.

An IoT data broker is a piece of middleware that standardizes sensor output into a common format that can then be retrieved by authorized applications.

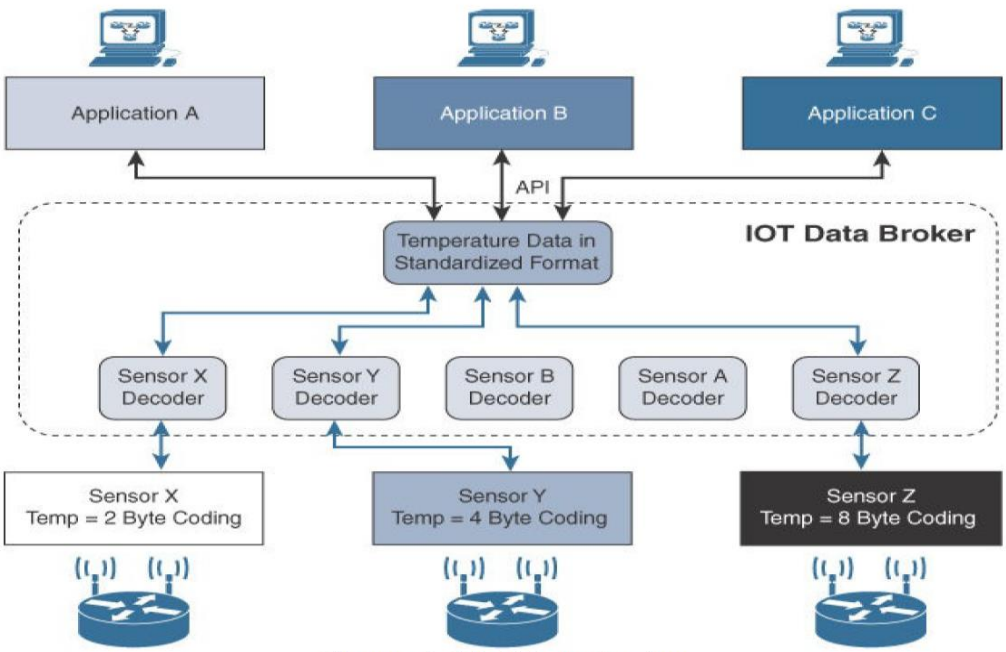


Figure 6-1 *IoT Data Broker*

In Figure 6-1, Sensors X, Y, and Z are all temperature sensors, but their output is encoded differently. The IoT data broker understands the different formats in which the temperature is encoded and is therefore able to decode this data into a common, standardized format. Applications A, B, and C in Figure 6-1 can access this temperature data without having to deal with decoding multiple temperature data formats.

IoT data brokers are also utilized from a commercial perspective to distribute and sell IoT data to third parties.

SCADA(supervisory control and data acquisition)

- Vertical industries have developed communication protocols that fit their specific requirements. ex: RS-232 and RS-485
- Well structured compared to the protocols, running directly over serial physical and data link layers.
- SCADA systems collect sensor data and telemetry from remote devices, while also providing the ability to control them.
- SCADA commonly uses certain protocols for communications between devices and applications. For example, Modbus and its variants are industrial protocols used to monitor and program remote devices via a master/slave relationship.
- The DNP3 and International Electrotechnical Commission (IEC) 60870-5-101 protocols are found mainly in the utilities industry

Adapting SCADA for IP

the IEC adopted the Open System Interconnection (OSI) layer model to define its protocol framework. protocol user groups also slightly modified their protocols to run over an IP infrastructure.

This included assigning TCP/UDP port numbers to the protocols, such as the following:

- DNP3 (adopted by IEEE 1815-2012) specifies the use of TCP or UDP on port 20000 for transporting DNP3 messages over IP.
- The Modbus messaging service utilizes TCP port 502.
- IEC 60870-5-104 is the evolution of IEC 60870-5-101 serial for running over Ethernet and IPv4 using port 2404.
- DLMS User Association specified a communication profile based on TCP/IP in the DLMS/COSEM Green Book (Edition 5 or higher), or in the
- IEC 62056-53 and IEC 62056-47 standards, allowing data exchange via IP and port 4059.

DNP3 is based on a master/slave relationship. The term *master* in this case refers to what is typically a powerful computer located in the control center of a utility, and a *slave* is a remote device with computing resources found in a location such as a substation. DNP3 refers to slaves specifically as *outstations*.

Outstations monitor and collect data from devices that indicate their state, such as whether a circuit breaker is on or off, and take measurements, including voltage, current, temperature, and so on. This data is then transmitted to the master when it is requested, or events and alarms can be sent in an asynchronous manner. The master also issues control commands, such as to start a motor or reset a circuit breaker, and logs the incoming data.

The IEEE 1815-2012 specification describes how the DNP3 protocol implementation must be adapted to run either over TCP (recommended) or UDP.

Connection management links the DNP3 layers with the IP layers in addition to the configuration parameters and methods necessary for implementing the network connection. The IP layers appear transparent to the DNP3 layers as each piece of the protocol stack in one station logically communicates with the respective part in the other.

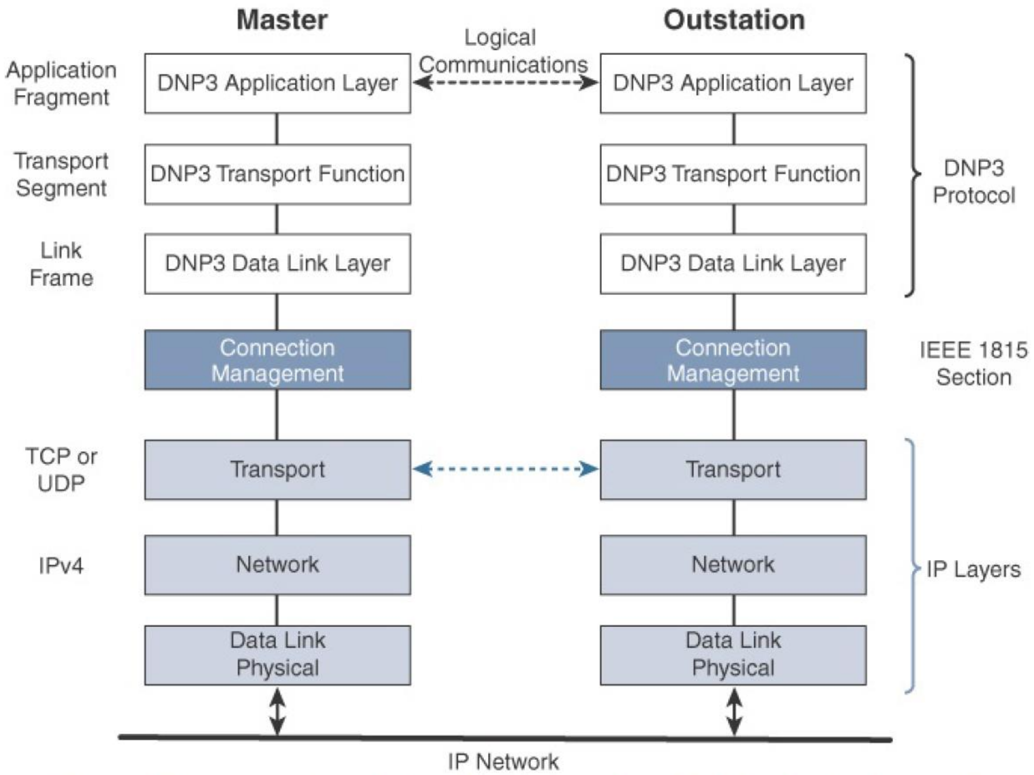


Figure 6-2 *Protocol Stack for Transporting Serial DNP3 SCADA over IP*

the master side initiates connections by performing a TCP active open. The outstation listens for a connection request by performing a TCP passive open. *Dual endpoint* is defined as a process that can both listen for connection requests and perform an active open on the channel if required.

Master stations may parse multiple DNP3 data link layer frames from a single UDP datagram, while DNP3 data link layer frames cannot span multiple UDP datagrams. Single or multiple connections to the master may get established while a TCP keep alive timer monitors the status of the connection. Keep alive messages are implemented as DNP3 data link layer status requests. If a response is not received to a keep alive message, the connection is deemed broken, and the appropriate action is taken.

Tunneling Legacy SCADA over IP Networks

- transport of the original serial protocol over IP can be achieved either by tunneling using raw sockets over TCP or UDP or by installing an intermediate device that performs protocol translation between the serial protocol version and its IP implementation.
- A raw socket connection simply denotes that the serial data is being packaged directly into a TCP or UDP transport. A socket in this instance is a standard application programming interface (API) composed of an IP address and a TCP or UDP port

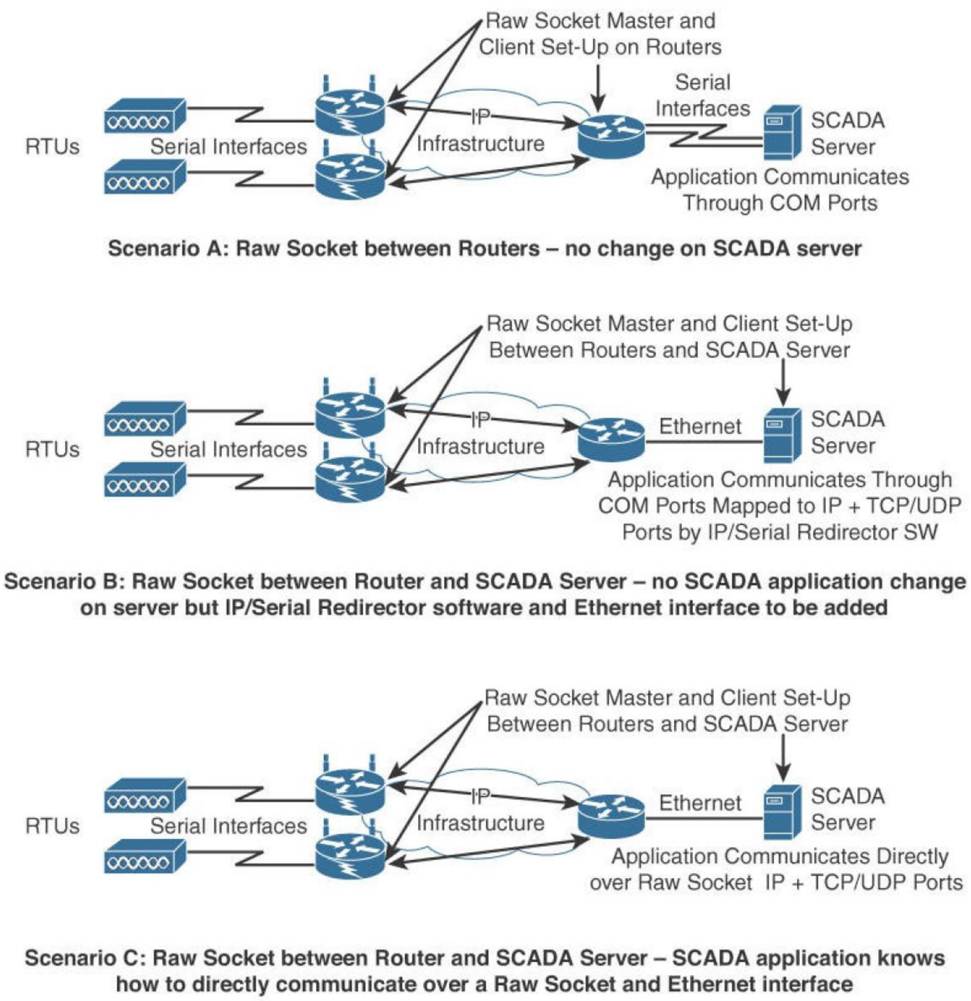


Figure 6-3 *Raw Socket TCP or UDP Scenarios for Legacy Industrial Serial Protocols*

In all the scenarios in Figure 6-3, notice that routers connect via serial interfaces to the remote terminal units (RTUs), which are often associated with SCADA networks. An RTU is a multipurpose device used to monitor and control various systems, applications, and devices managing automation. From the master/slave perspective, the RTUs are the slaves. Opposite the RTUs in each Figure 6-3 scenario is a SCADA server, or master, that varies its connection type.

- In Scenario A in Figure 6-3, both the SCADA server and the RTUs have a direct serial connection to their respective routers. The routers terminate the serial connections at both ends of the link and use raw socket encapsulation totransport the serial payload over the IP network.
- Scenario B has a small change on the SCADA server side. A piece of software isinstalled on the SCADA server that maps the serial COM ports to IP ports. This software is commonly referred to as an IP/serial redirector. The IP/serial redirector in essence terminates the serial connection of the SCADA server and converts it to a TCP/IP port using a raw socket connection.
- In Scenario C in Figure 6-3, the SCADA server supports native raw socket capability. the SCADA server has full IP support for raw socket connections.

SCADA Protocol Translation

- an alternative to a raw socket connection for transporting legacy serial data across an IP network is protocol translation
- For example, Figure 6-4 shows two serially connected DNP3 RTUs and two master applications supporting DNP3 over IP that control and pull data from the RTUs.
- The IoT gateway in this figure performs a protocol translation function that enables communication between the RTUs and servers

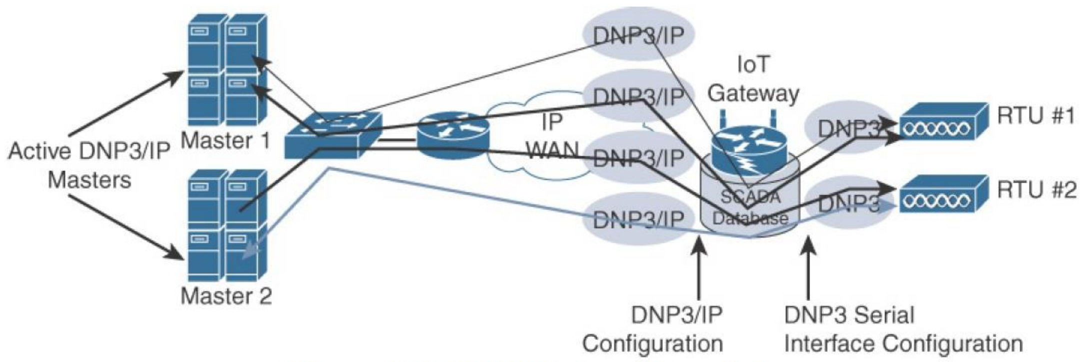


Figure 6-4 *DNP3 Protocol Translation*

- Adding computing functions close to the edge helps scale distributed intelligence in IoT networks. This can be accomplished by offering computing resources on IoT gateways or routers
- This can also be performed directly on a node connecting multiple sensors.

SCADA Transport over LLNs with MAP-T

The industrial devices supporting IP today support IPv4 only. When deployed over LLN subnet works that are IPv6 only, a transition mechanism, such as MAP-T (Mapping of Address and Port using Translation, RFC 7599), needs to be implemented.

- Figure 6.5 depicts a scenario in which a legacy endpoint is connected across an LLN running 6LoWPAN to an IP-capable SCADA server. The legacy endpoint could be running various industrial and SCADA protocols, including DNP3/IP, Modbus/TCP, or IEC 60870-5-104. In this scenario, the legacy devices and the SCADA server support only IPv4 (typical in the industry today).
- the end devices, the endpoints, and the SCADA server support only IPv4, but the network in the middle supports only IPv6.

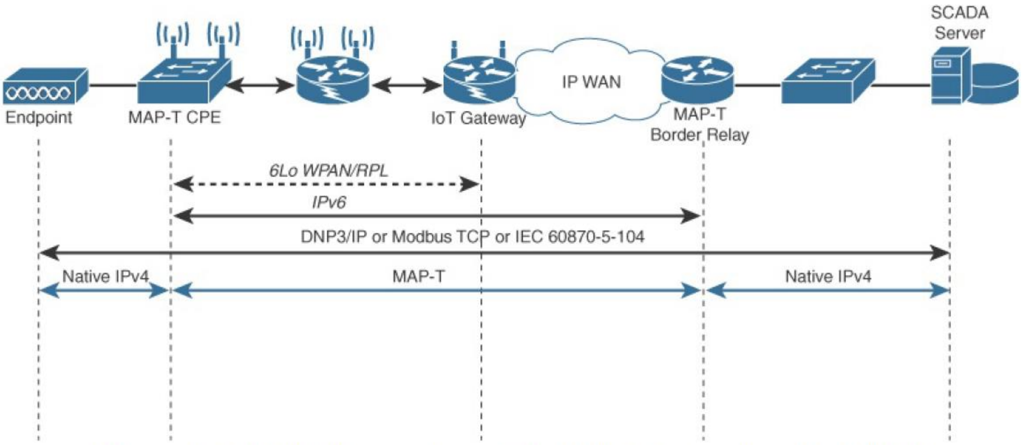


Figure 6-5 *DNP3 Protocol over 6LoWPAN Networks with MAP-T*

- MAP-T makes the appropriate mappings between IPv4 and the IPv6 protocols. This allows legacy IPv4 traffic to be forwarded across IPv6 networks.
- In Figure 6-5 the IPv4 endpoint on the left side is connected to a Customer Premise Equipment (CPE) device. The MAP-T CPE device has an IPv6 connection to the RPL mesh. On the right side, a SCADA server with native IPv4 support connects to a MAP-T border gateway. The MAP-T CPE device and MAP-T border gateway are thus responsible for the MAP-T conversion from IPv4 to IPv6.

Generic Web-Based Protocols

- web-based protocols have become common in consumer and enterprise applications and services.
- The level of familiarity with generic web-based protocols is high can work on IoT applications, and this may lead to innovative ways to deliver and handle real-time IoT data.
- an IoT device generating an event can have the result of launching a video capture, while at the same time a notification is sent to a collaboration tool, such as a Cisco Spark room.
- the definition of constrained nodes and networks must be analysed to select the most appropriate protocol.
- On non-constrained networks, such as Ethernet, Wi-Fi, or 3G/4G cellular, where bandwidth is not perceived as a potential issue, data payloads based on a verbose

data model representation, including XML or JavaScript Object Notation (JSON), can be transported over HTTP/HTTPS or WebSocket.

- This allows implementers to develop their IoT applications in contexts similar to web applications. Implementers to develop their IoT applications in contexts similar to web applications.
- When considering web services implementation on an IoT device, the choice between supporting the client or server side of the connection must be carefully weighed. The HTTP client side only initiates connections and does not accept incoming ones.
- Interactions between real-time communication tools powering collaborative applications, such as voice and video, instant messaging, chat rooms, and IoT devices, are also emerging. This is driving the need for simpler communication systems between people and IoT devices. One protocol that addresses this need is Extensible Messaging and Presence Protocol (XMPP).

IoT Application Layer Protocols

The IoT industry is working on new lightweight protocols that are better suited to large numbers of constrained nodes and networks

Two of the most popular protocols are CoAP and MQTT.

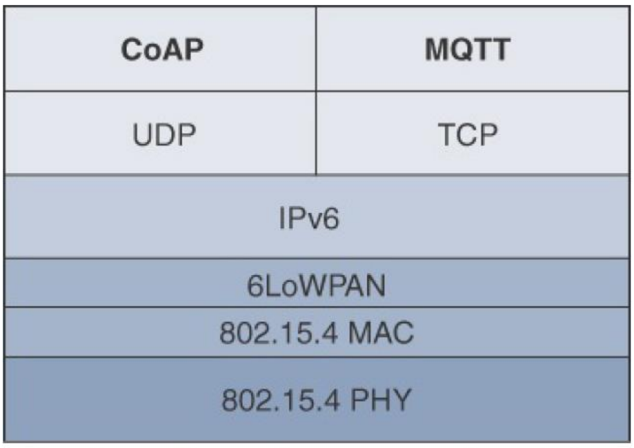


Figure 6-6 *Example of a High-Level IoT Protocol Stack for CoAP and MQTT*

CoAP(over udp) and MQTT (over tcp)are naturally at the top of this sample IoT stack, based on an IEEE 802.15.4 mesh network

CoAP

- Constrained Application Protocol (CoAP) resulted from the IETF Constrained Restful Environments (CoRE) working group’s efforts to develop a generic framework for resource-oriented applications targeting constrained nodes and networks.
- The CoAP framework defines simple and flexible ways to manipulate sensors and actuators for data or device management..

CoAP, including the following:

RFC 6690: Constrained RESTful Environments (CoRE) Link Format

RFC 7252: The Constrained Application Protocol (CoAP)

RFC 7641: Observing Resources in the Constrained Application Protocol (CoAP)

RFC 7959: Block-Wise Transfers in the Constrained Application Protocol (CoAP)

RFC 8075: Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)

- Facilitate the exchange of messages over UDP between endpoints, including the secure transport protocol Datagram Transport Layer Security (DTLS).
- Alternate transport including TCP, secure TLS, and WebSocket
- Short Message Service (SMS) as defined in Open Mobile Alliance for Lightweight Machine-to-Machine (LWM2M) for IoT device management is also being considered.

- RFC 7252 provides security CoAP with DTLS. CoAP endpoint is provisioned with keys and a filtering list. Four security modes are defined: NoSec, PreSharedKey, RawPublicKey, and Certificate. The NoSec and RawPublicKey implementations are mandatory.

- a CoAP message is composed of a short fixed length Header field (4 bytes), a variable-length but mandatory Token field (0–8 bytes), Options fields if necessary, and the Payload field.

- the CoAP message format is relatively simple and flexible. It allows CoAP to deliver low overhead, which is critical for constrained networks, while also being easy t parse and process for constrained devices.

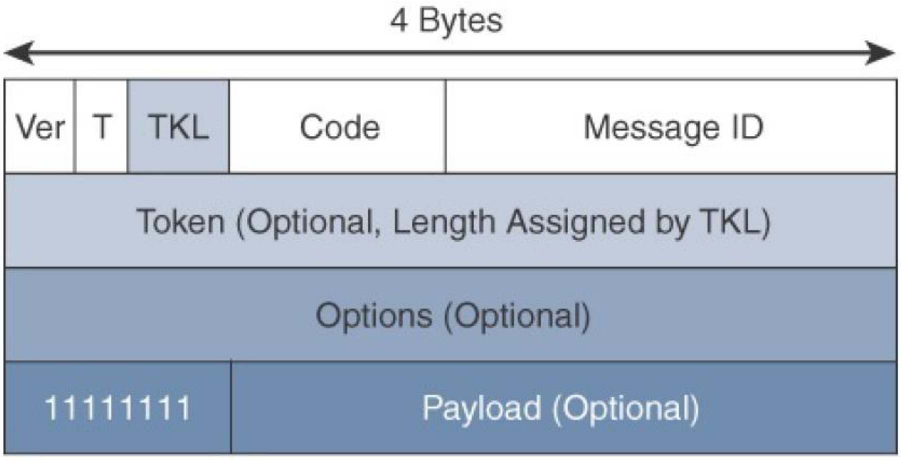


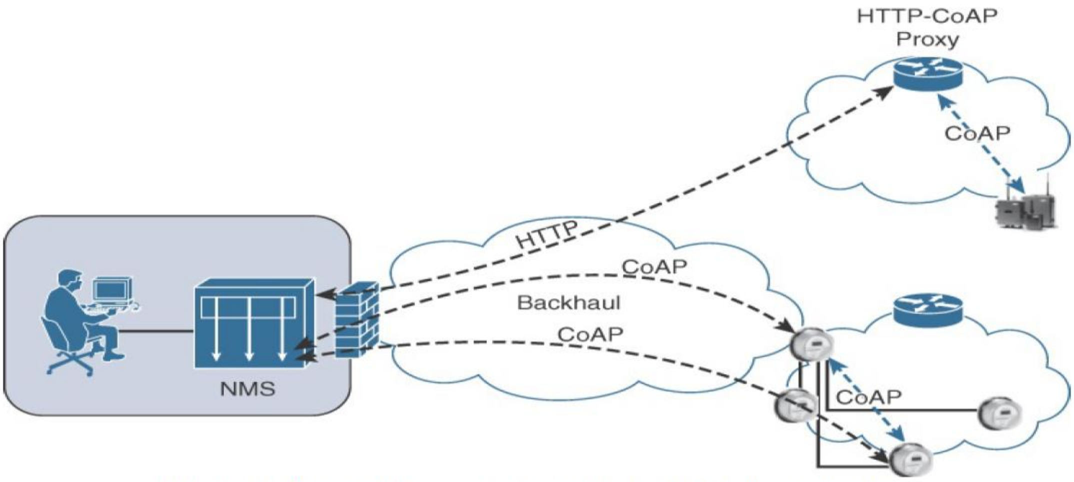
Figure 6-7 *CoAP Message Format*

CoAP Message Field	Description
Ver (Version)	Identifies the CoAP version.
T (Type)	Defines one of the following four message types: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), or Reset (RST). CON and ACK are highlighted in more detail in Figure 6-9.
TKL (Token Length)	Specifies the size (0–8 Bytes) of the Token field.
Code	Indicates the request method for a request message and a response code for a response message. For example, in Figure 6-9, GET is the request method, and 2.05 is the response code. For a complete list of values for this field, refer to RFC 7252.
Message ID	Detects message duplication and used to match ACK and RST message types to Con and NON message types.
Token	With a length specified by TKL, correlates requests and responses.
Options	Specifies option number, length, and option value. Capabilities provided by the Options field include specifying the target resource of a request and proxy functions.
Payload	Carries the CoAP application data. This field is optional, but when it is present, a single byte of all 1s (0xFF) precedes the payload. The purpose of this byte is to delineate the end of the Options field and the beginning of Payload.

Table 6-1 *CoAP Message Fields*

CoAP can run over IPv4 or IPv6 CoAP message size could be up to 1152 bytes, including 1024 bytes for the payload.

- Connections can be between devices located on the same or different constrained networks or between devices and generic Internet or cloud servers, all operating over IP
- As both HTTP and CoAP are IP-based protocols, the proxy function can be located practically anywhere in the network, not necessarily at the border between constrained and nonconstrained networks.



- Figure 6-8 *CoAP Communications in IoT Infrastructures*
- CoAP is based on the REST architecture, but with a "thing" acting as both the client and the server. Through the exchange of asynchronous messages,
- a client requests an action via a method code on a server resource. A uniform resource identifier (URI) localized on the server identifies this resource.
- The server responds with a response code that may include a resource representation.
- The CoAP request/response semantics include the methods GET,POST, PUT, and DELETE.
- CoAP URI format

```
coap-URI = "coap:" "//" host [":" port] path-abempty ["?" query]
coaps-URI = "coaps:" "//" host [":" port] path-abempty  ["?" query]
```

- CoAP defines four types of messages: confirmable, non confirmable, acknowledgement, and reset.
- Method codes and response codes included in some of these messages make them carry requests or responses

- While running over UDP, CoAP offers a reliable transmission of messages when a CoAP header is marked as “confirmable.”

- CoAP supports basic congestion control with a default time-out, simple stop and wait retransmission with exponential back-off mechanism, and detection of duplicate messages through a message ID.

- If a request or response is tagged as confirmable, the recipient must explicitly either acknowledge or reject the message, using the same message ID

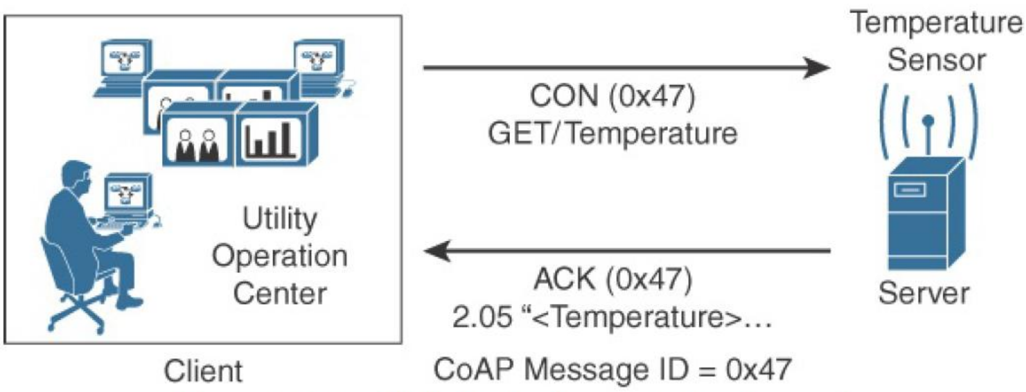
- 

Figure 6-9 *CoAP Reliable Transmission Example*

Figure 6-9 shows a utility operations center on the left, acting as the CoAP client, with the CoAP server being a temperature sensor on the right of the figure. The communication between the client and server uses a CoAP message ID of 0x47. The CoAP Message ID ensures reliability and is used to detect duplicate messages.

The client in Figure 6-9 sends a GET message to get the temperature from the sensor. Notice that the 0x47 message ID is present for this GET message and that the message is also marked with CON. A CON, or confirmable, marking in a CoAP message means the message will be retransmitted until the recipient sends an acknowledgement (or ACK) with the same message ID.

In Figure 6-9, the temperature sensor does reply with an ACK message referencing the correct message ID of 0x47. In addition, this ACK message piggybacks a successful response to the GET request itself. This is indicated by the 2.05 response code followed by the requested data.

Message Queuing Telemetry Transport (MQTT)

Considering the harsh environments in the oil and gas industries, an extremely simple protocol with only a few options was designed, with considerations for constrained nodes, unreliable WAN backhaul communications, and bandwidth constraints with variable latencies. These selection of a client/server and publish/subscribe framework based on the TCP/IP architecture, as shown in Figure 6-10.

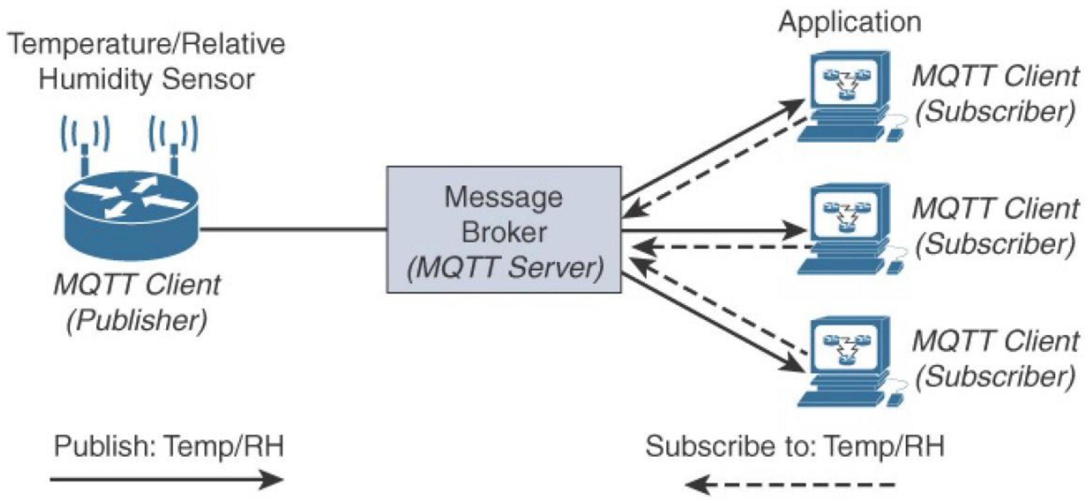


Figure 6-10 *MQTT Publish/Subscribe Framework*

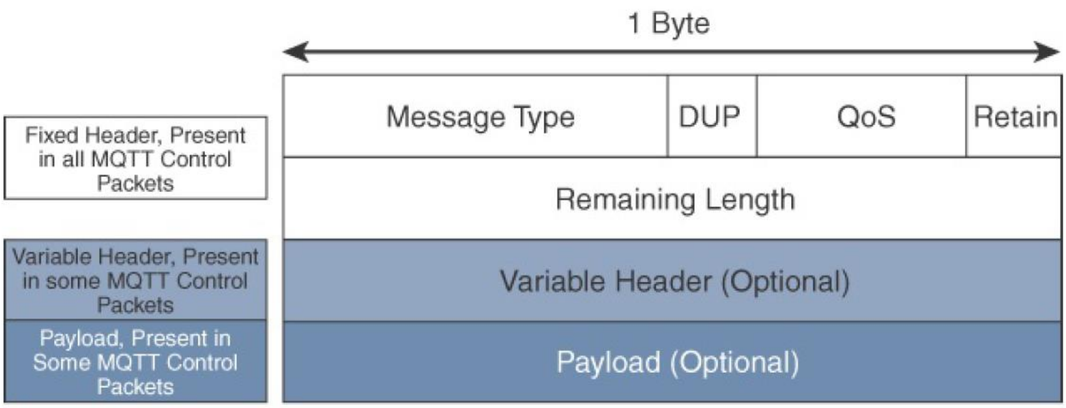
An MQTT client can act as a publisher to send data (or resource information) to an MQTT server acting as an MQTT message broker

- in Figure 6-10, the MQTT client on the left side is a temperature (Temp) and relative humidity (RH) sensor that publishes its Temp/RH data. The MQTT server (or message broker) accepts the network connection along with application messages, such as Temp/RH data, from the publishers. It also handles the subscription and unsubscription process and pushes the application data to MQTT clients acting as subscribers.

- The application on the right side of Figure 6-10 is an MQTT client that is a subscriber to the Temp/RH data being generated by the publisher or sensor on the left. This model, where subscribers express a desire to receive information from publishers,
- MQTT message broker ensures that information can be buffered and cached in case of network failures.

- MQTT is a lightweight protocol because each control packet consists of a 2-byte fixed header with optional variable header fields and optional payload. control packet can contain a payload up to 256 MB. Figure 6- 11 provides an overview of the MQTT message format.

Figure 6-11 *MQTT Message Format*



Fourteen different types of control packets are specified in MQTT version 3.1.1. Each of them has a unique value that is coded into the Message Type field. Note that values 0 and 15 are reserved. MQTT message types are summarized in Table 6-2.

Message Type	Value	Flow	Description
CONNECT	1	Client to server	Request to connect
CONNACK	2	Server to client	Connect acknowledgement
PUBLISH	3	Client to server Server to client	Publish message
PUBACK	4	Client to server Server to client	Publish acknowledgement
PUBREC	5	Client to server Server to client	Publish received
PUBREL	6	Client to server Server to client	Publish release
PUBCOMP	7	Client to server Server to client	Publish complete
SUBSCRIBE	8	Client to server	Subscribe request
SUBACK	9	Server to client	Subscribe acknowledgement
UNSUBSCRIBE	10	Client to server	Unsubscribe request
UNSUBACK	11	Server to client	Unsubscribe acknowledgement
PINGREQ	12	Client to server	Ping request
PINGRESP	13	Server to client	Ping response
DISCONNECT	14	Client to server	Client disconnecting

Table 6-2 *MQTT Message Types*

- DUP (Duplication Flag) when set, allows the client to notate that the packet has been sent previously, but an acknowledgement was not received.
- QoS: header field allows for the selection of three different QoS levels.
- Retain flag: Only found in a PUBLISH message notifies the server to hold onto the message data. This allows new subscribers to instantly receive the last known value without having to wait for the next update from the publisher.
- Remaining Length.: This field specifies the number of bytes in the MQTT packet following this field.

MQTT sessions between each client and server consist of four phases: session establishment, authentication, data exchange, and session termination.

- Each client connecting to a server has a unique client ID, which allows the identification of the MQTT session between both parties. When the server is delivering an application message to more than one client, each client is treated independently.

- Subscriptions to resources generate SUBSCRIBE/SUBACK control packets, while unsubscription is performed through the exchange of UNSUBSCRIBE/UNSUBACK control packets. Graceful termination of a connection is done through a DISCONNECT control packet, which also offers the capability for a client to reconnect by re-sending its client ID to resume the operations.

A message broker uses a topic name to filter messages for its subscribers. When subscribing to a resource, the subscriber indicates the one or more topic levels that are used forward slash (/) to structure the topic name. provide a hierarchical structure to the topic names. Figure 6-12 illustrates these

concepts with Example **adt/lora.adeunis** being a topic level and

- **adt/lora/adeunis/0018B2000000023A**

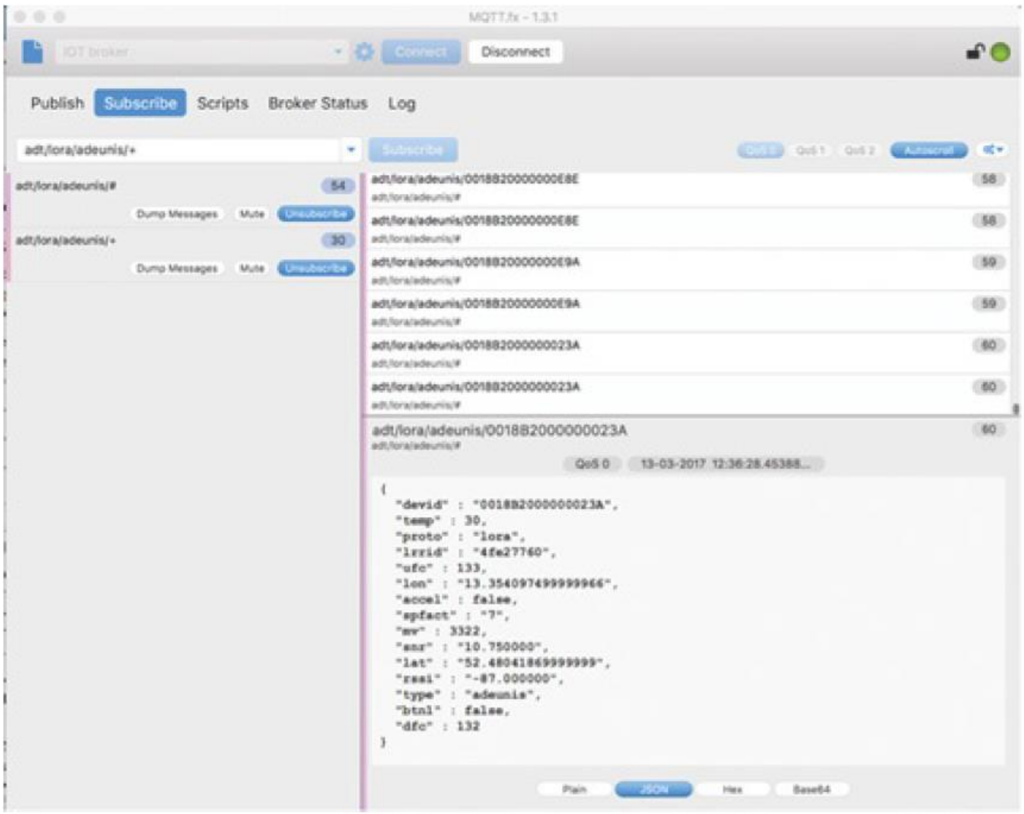


Figure 6-12 *MQTT Subscription Example*

A subscription can contain one of the wildcard characters to allow subscription to multiple topics at once

The pound sign (#): multilevel wildcard represents the parent and any number of child levels.
adt/lora/adeunis/# enables the
reception of the whole subtree,
adt/lora/adeunis/0018B20000000E9E
adt/lora/adeunis/0018B20000000E8E
adt/lora/adeunis/0018B20000000E9A
The plus sign (+) is a wildcard character that matches only one topic level. For example, **adt/lora/+** allows access to **adt/lora/adeunis/** and **adt/lora/abeeway** but not to **adt/lora/adeunis/0018B20000000E9E**.

- Topic names beginning with the dollar sign ($) must be excluded by the server
- PINGREQ/PINGRESP control packets are used to validate the connections between the client and server.

These are the three levels of MQTT QoS:[FIGURE 6.13]

- **QoS 0:** This is a best-effort and unacknowledged data service referred to as “at most once” delivery. The publisher sends its message one time to a server, which transmits it once to the subscribers. No response is sent by the receiver, and no retry is performed by the sender. The message arrives at the receiver either once or not at all.

- **QoS 1:** This QoS level ensures that the message delivery between the publisher and server and then between the server and subscribers occurs at least once. In PUBLISH and PUBACK packets, a packet identifier is included in the variable header. If the message is not acknowledged by a PUBACK packet, it is sent again. This level guarantees “at least once” delivery.

- **QoS 2:** This is the highest QoS level, used when neither loss nor duplication of messages is acceptable. There is an increased overhead associated with this QoS level because each packet contains an optional variable header with a packet identifier. Confirming the receipt of a PUBLISH message requires a two-step acknowledgement process. The first step is done through the PUBLISH/PUBREC packet pair, and the second is achieved with the PUBREL/PUBCOMP packet pair. This level provides a "guaranteed service" known as "exactly once" delivery, with no consideration for the number of retries as long as the message is delivered once.

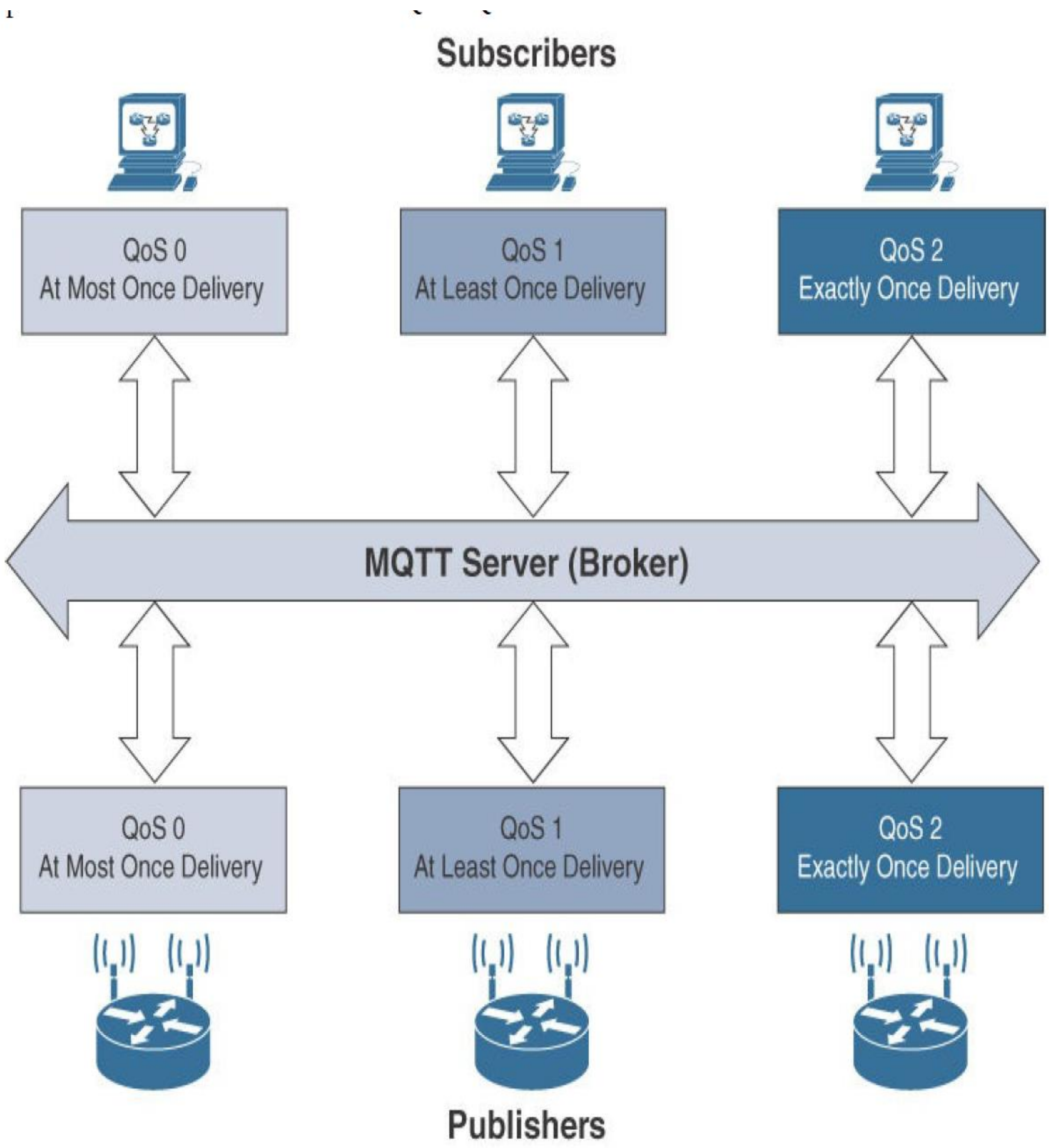


Figure 6-13 *MQTT QoS Flows*

Factor	CoAP	MQTT
Main transport protocol	UDP	TCP
Typical messaging	Request/response	Publish/subscribe
Effectiveness in LLNs	Excellent	Low/fair (Implementations pairing UDP with MQTT are better for LLNs.)
Security	DTLS	SSL/TLS
Communication model	One-to-one	many-to-many
Strengths	Lightweight and fast, with low overhead, and suitable for constrained networks; uses a RESTful model that is easy to code to; easy to parse and process for constrained devices; support for multicasting; asynchronous and synchronous messages	TCP and multiple QoS options provide robust communications; simple management and scalability using a broker architecture
Weaknesses	Not as reliable as TCP-based MQTT, so the application must ensure reliability.	Higher overhead for constrained devices and networks; TCP connections can drain low-power devices; no multicasting support

Table 6-3 *Comparison Between CoAP and MQTT*