Module 3

IP as the IoT Network Layer, The Business Case for IP, The need for Optimization, Optimizing IP for IoT, Profiles and Compliances, Application Protocols for IoT, The Transport Layer, IoT Application Transport Methods.

Chapter 5

IP as the IoT Network Layer

5.1 The Business Case for IP

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

5.1.1 The Key Advantages of Internet Protocol

One of the main differences between traditional information technology (IT) and operational technology (OT) is the lifetime of the underlying technologies and products.

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

1. **Open and standards-based:** TheInternet of Things creates a new paradigm in which devices, applications,and users can leverage a large set of devices and functionalities whileguaranteeing interchangeability and interoperability, security, andmanagement. This calls for implementation, validation, and deployment ofopen, standards-based solutions. While many standards developmentorganizations (SDOs) are working on Internet of Things definitions,frameworks, applications, and technologies, none are questioning the roleof the Internet Engineering Task Force (IETF) as the foundation forspecifying and optimizing the network and transport layers.
2. **Versatile:** Even if physical and data link layers such as Ethernet, Wi-Fi, and cellular are widely adopted, the history of data communications demonstrates that no given wired or wireless technology fits all deployment criteria. Furthermore, communication technologies evolve at a pace faster than the expected 10- to 20-year lifetime of OT solutions. So, the layered IP architecture is well equipped to cope with any type of physical and data link layers.
3. **Ubiquitous:** All recent operating system releases, from general-purpose computers and servers to lightweight embedded systems (TinyOS, Contiki, and so on), have an integrated dual (IPv4 and IPv6) IP stack that gets enhanced over time. In addition, IoT

application protocols in many industrial OT solutions have been updated in recent years to run over IP. While these updates have mostly consisted of IPv4 to this point, recent standardization efforts in several areas are adding IPv6. In fact, IP is the most pervasive protocol when you look at what is supported across the various IoT solutions and industry verticals.

4. **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 proven before that scalability is one of its strengths.
5. **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.
6. **Stable and resilient:**IP has a large and well-established knowledge base and, more importantly, it has been used for years in critical infrastructures, such as financial and defense networks. In addition, IP has been deployed for critical services, such as voice and video, which havealready transitioned from closed environments to open IP standards.
7. **Consumers' market adoption:** When developing IoT solutions and products targeting the consumer market, vendors know that consumers' access to applications and devices will occur predominantly over broadband and mobile wireless infrastructure. The main consumer devices range from smart phones to tablets and PCs. The common protocol that links IoT in the consumer space to these devices is IP.
8. **The innovation factor:** 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.

5.1.2 Adoption or Adaptation of the Internet Protocol

How to implement IP in data center, cloud services, and operation centers hosting IoT applications may seem obvious, but the adoption of IP in the last mile is more complicated and often makes running IP end-to-end more difficult.

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.

The IP adaptation versus adoption model still requires investigation for particular last-mile technologies used by IoT. You should consider the following factors when trying to determine which model is best suited for last-mile connectivity:

1. **Bidirectional versus unidirectional data flow:**As defined in RFC 7228, may only infrequently need to report a few bytes of data to an application. These sorts of devices, particularly ones that communicate through LPWA technologies, include fire alarms sending alerts or daily test reports, electrical switches being pushed on or off, and water or gas meters sending weekly indexes. 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.

2. **Overhead for last-mile communications paths:** 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. If the data to be forwarded by a device is infrequent and only a few bytes, you can potentially have more header overhead than device data—again, particularly in the case of LPWA technologies.

3. **Data flow model:**Any node can easily exchange data with any other node in a network, although security, privacy, and other factors may put controls and limits on the "end-to-end" concept. However, in many IoT solutions, a device's data flow is limited to one or two applications. In this case, 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.
4. **Network diversity:** One of the drawbacks of the adaptation model is a general dependency on single PHY and MAC layers. Integration and coexistence of new physical and MAC layers or new applications impact how deployment and operations have to be planned. This is not a relevant consideration for the adoption model.

5.2 The Need for Optimization

Optimizations are needed at various layers of the IP stack to handle the restrictions that are present in IoT networks. The followings area require optimization

1. Constrained Nodes

Different classes ofdevices coexist. Depending on its functions in a network, a "thing" architecturemay or may not offer similar characteristics compared to a generic PC or serverin an IT environment.

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.

Finally, power consumption is a key characteristic of constrained nodes. Many IoT devices are battery powered, with lifetime battery requirements varying from a few months to 10+ years. This drives the selection of networking technologies since high-speed ones, such as Ethernet, Wi-Fi, and cellular, are not (yet) capable of multi-year battery life.

2. Constrained Networks

In the early years of the Internet, network bandwidth capacity was restrained dueto technical limitations. Connections often depended on low-speed modems fortransferring data. However, these low-speed connections demonstrated that IPcould run over low-bandwidth networks. A constrained network can have high latency and a high potential for packet loss.

Constrained networks are limited by low-power, low-bandwidth links (wireless and wired). They operate between a few kbps and a few hundred kbps and may utilize a star, mesh, or combined network topologies, ensuring proper operations.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.

3. IP Versions

The main driving force has been the lack of address space in IPv4 as the Internet has grown. IPv6 has a much larger range of addresses that should not be exhausted for the foreseeable future. Today, both traffic is still IPv4 based.

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

1. **Application Protocol:**IoT devices versions of IP run over the Internet, but most 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.
2. **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.
3. **Serial Communications:** 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.

4. **IPv6 Adaptation Layer:** 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.

5.3 Optimizing IP for IoT

The following sections introduce some of these optimizations already available from the market or under development by the IETF. Figure 5-1 highlights the TCP/IP layers where optimization is applied.

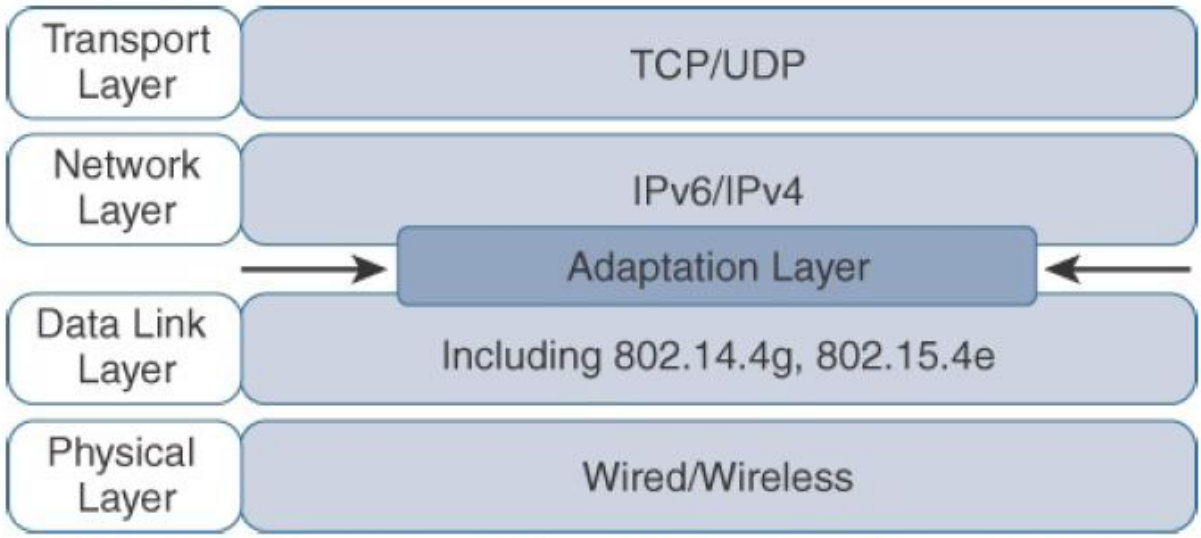

Figure 5-1 *Optimizing IP for IoT Using an Adaptation Layer*

5.3.1 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.

The main examples of 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.

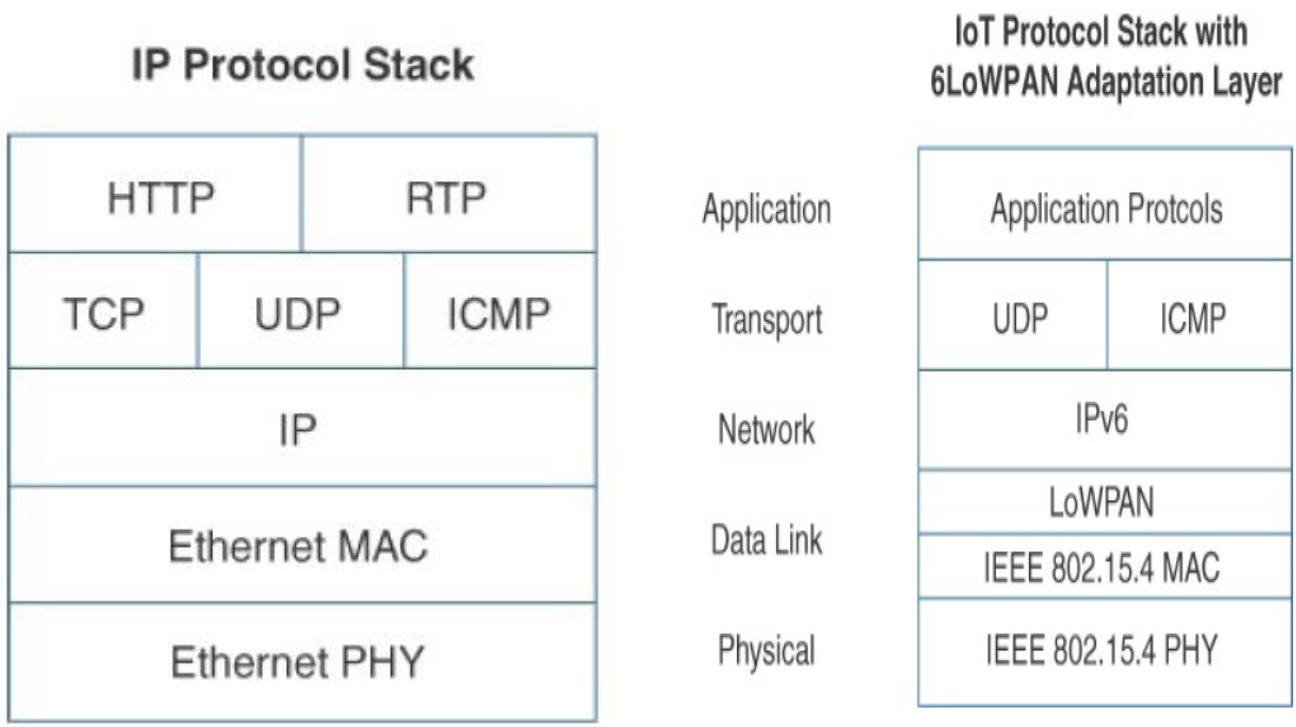

Figure 5-2 Comparison of an IoT Protocol Stack Utilizing 6LoWPAN and an IP Protocol Stack

The 6LoWPAN working group published several RFCs, but RFC 4994 is Foundational because it defines frame headers for the capabilities of header compression, fragmentation, and mesh addressing. These headers can be stacked in the adaptation layer to keep these concepts separate while enforcing a structured method for expressing each capability. Depending on the implementation, all, none, or any combination of these capabilities and their corresponding headers can be enabled. Figure 5-3 shows some examples of typical 6LoWPAN header stacks.

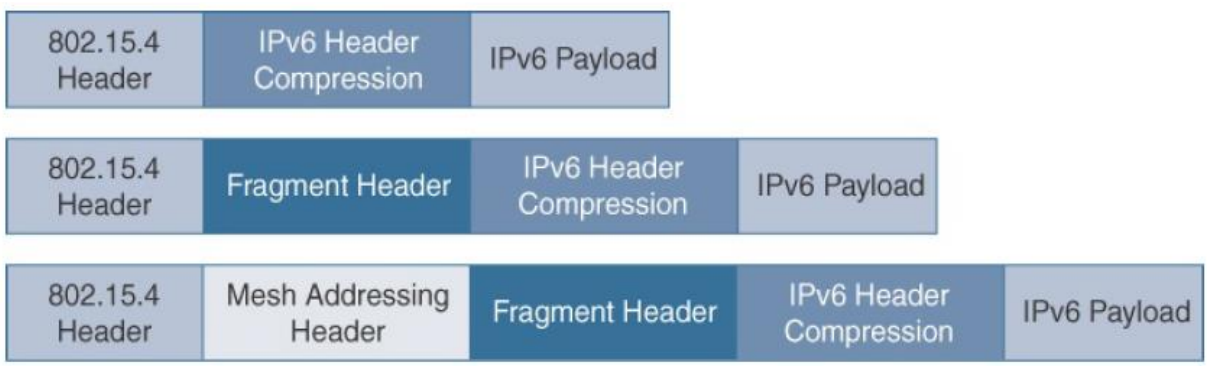

Figure 5-3 *6LoWPAN Header Stacks*

Header Compression

IPv6 header compression for 6LoWPAN was defined initially in RFC 4944 and subsequently updated by RFC 6282. This capability shrinks 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.

At a high level, 6LoWPAN works by taking advantage of shared information known by all nodes from their participation in the local network. In addition, it omits some standard header fields by assuming commonly used values. Figure 5-4 highlights an example that shows the amount of reduction that is possible with 6LoWPAN header compression.

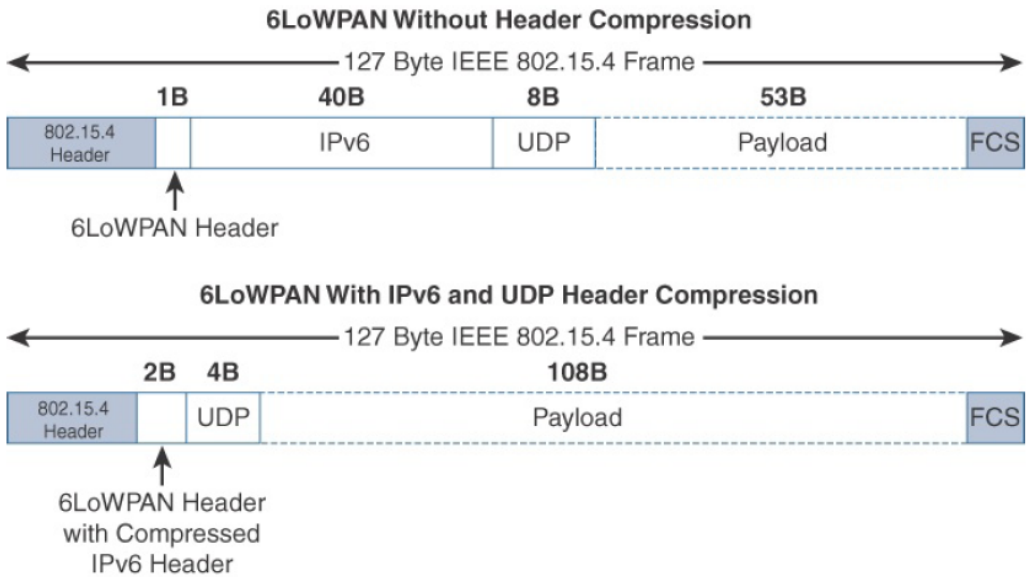

Figure 5-4 *6LoWPAN Header Compression*

At the top of Figure 5-4, you see a 6LoWPAN frame without any header compression enabled: The full 40-byte IPv6 header and 8-byte UDP header are visible. The 6LoWPAN header is only a single byte in this case. 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.

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. You can see that this is a problem because 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.

The fragment header utilized by 6LoWPAN is composed of three primary fields: Datagram Size, Datagram Tag, and Datagram Offset. The 1-byte Datagram Size field specifies the total size of the unfragmented payload. Datagram Tag identifies the set of fragments for a payload. Finally, the Datagram Offset field delineates how far into a payload a particular fragment occurs.
Figure 5-5 provides an overview of a 6LoWPAN fragmentation header.

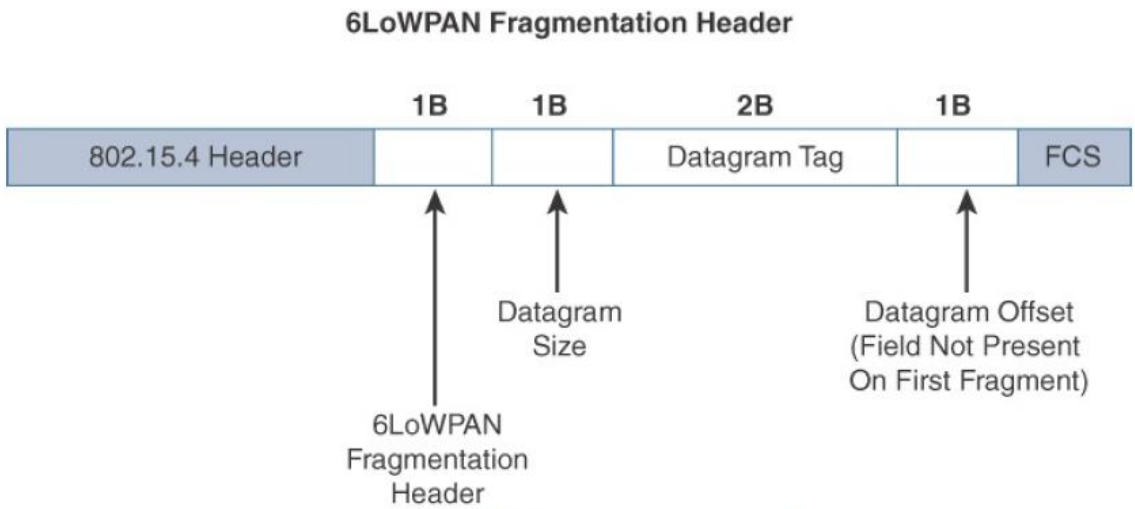

Figure 5-5 *6LoWPAN Fragmentation Header*

In Figure 5-5, the 6LoWPAN fragmentation header field itself uses a unique bit value to identify that the subsequent fields behind it are fragment fields as opposed to another capability, such as header compression. Also, in the first fragment, the Datagram Offset field is not present because it would simply be set to 0. This results in the first fragmentation header for an IPv6 payload being only 4 bytes long. The remainder of the fragments have a 5-byte header field so that the appropriate offset can be specified.

Mesh Addressing

The purpose of the 6LoWPAN mesh addressing function is to forward packetsover multiple hops. Three fields are defined for this header: Hop Limit, SourceAddress, and Destination Address. Analogous to the IPv6 hop limit field, thehop limit for mesh addressing also provides an upper limit on how many timesthe frame can be forwarded. Each hop decrements this value by 1 as it isforwarded. Once the value hits 0, it is dropped and no longer forwarded.

The Source Address and Destination Address fields for mesh addressing areIEEE 802.15.4 addresses indicating the endpoints of an IP hop. Figure 5-6details the 6LoWPAN mesh addressing header fields.

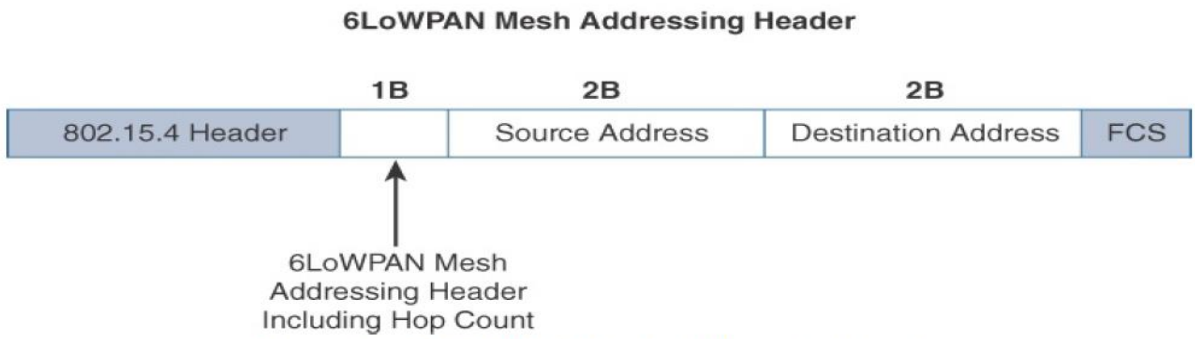

Figure 5-6 *6LoWPAN Mesh Addressing Header*

Note that the mesh addressing header is used in a single IP subnet and is a Layer2 type of routing known as mesh-under. The concept of mesh-under is discussedin the next section. Keep in mind that RFC 4944 only provisions the function inthis case as the definition of Layer 2 mesh routing specifications was outside thescope of the 6LoWPAN working group, and the IETF doesn't define "Layer 2routing." An implementation performing Layer 3 IP routing does not need toimplement a mesh addressing header unless required by a given technologyprofile.

6TiSCH

Many proprietary wireless technologies have been developed and deployed invarious industry verticals over the years. However, the publication of the IEEE802.15.4 physical and data link layer specifications, followed by IEEE 802.15.4eamendments, has opened the path to standardized, deterministic communicationsover wireless networks.

IEEE 802.15.4e, Time-Slotted Channel Hopping (TSCH), is an add-on to theMedia Access Control (MAC) portion of the IEEE 802.15.4 standard, with directinheritance from other standards, such as WirelessHART and ISA100.11a.Devices implementing IEEE 802.15.4e TSCH communicate by following a TimeDivision Multiple Access (TDMA) schedule. An allocation of a unit ofbandwidth or time slot is scheduled between neighbor nodes. This allows theprogramming of predictable transmissions and enables deterministic, industrialtypeapplications. In comparison, other 802.15.4 implementations do not allocateslices of bandwidth, so communication, especially during times of contention,
May be delayed or lost because it is always best effort.

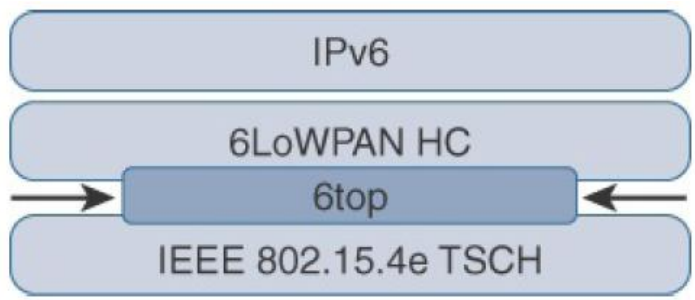

Figure 5-7 *Location of 6TiSCH's 6top Sublayer*

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. Nodes only transmit whenthe schedule dictates that their cell is open for communication. The 6TiSCHarchitecture defines four schedule management mechanisms:

1. **Static scheduling:**All nodes in the constrained network share a fixed Schedule. Cells are shared, and nodes contend for slot access in a slottedaloha manner. Slotted aloha is a basic protocol for sending data using timeslot boundaries when communicating over a shared medium. Staticscheduling is a simple scheduling mechanism that can be used upon initialimplementation or as a fallback in the case of network malfunction. Thedrawback with static scheduling is that nodes may expect a packet at anycell in the schedule. Therefore, energy is wasted idly listening across allcells.
2. **Neighbor-to-neighbor scheduling:** A schedule is established thatcorrelates with the observed number of transmissions between nodes. Cellsin this schedule can be added or deleted as traffic requirements andbandwidth needs change.
3. **Remote monitoring and scheduling management:** Time slots and otherresource allocation are handled by a management entity that can bemultiple hops away. The scheduling mechanism leverages 6top and evenCoAP in some scenarios.
4. **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 that is used by a nodeto trigger this scheduling mechanism is not defined at this point.

The forwarding decision is based on a preexistingstate that was learned from a routing computation. There are three 6TiSCHforwarding models:

- **Track Forwarding (TF):** This is the simplest and fastest forwardingmodel. A “track” in this model is a unidirectional path between a sourceand a destination. This track is constructed by pairing bundles of receivecells in a schedule with a bundle of receive cells set to transmit. So, aframe received within a particular cell or cell bundle is switched to anothercell or cell bundle. This forwarding occurs regardless of the network layerprotocol.
- **Fragment forwarding (FF):** This model takes advantage of 6LoWPAN Fragmentation to build a Layer 2 forwarding table.The 6LoWPAN sublayer learns the next-hopselection of this first fragment, which is then applied to all subsequentfragments of that packet. Otherwise, IPv6 packets undergo hop-by-hopreassembly. This increases latency and can be power- and CPU-intensivefor a constrained node.
- **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. QoS is aclassification scheme for flows based on their priority, and RED is a Common congestion avoidance mechanism.

RPL (Routing Protocol for Low Power and Lossy Networks)

New distance-vector routing protocol was named the IPv6Routing Protocol for Low Power and Lossy Networks (RPL). The RPL specification was published as RFC 6550 by the RoLL working group.

In an RPL network, each node acts as a router and becomes part of a meshnetwork. Routing is performed at the IP layer. Each node examines everyreceived IPv6 packet and determines the next-hop destination based on theinformation contained in the IPv6 header. No information from the MAC-layerheader is needed to perform next-hop determination.

To cope with the constraints of computing and memory that are common 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 towardthe border router.

RPL is based on the concept of a directed acyclic graph (DAG). A DAG is adirected graph where no cycles exist. This means that from any vertex or point inthe graph, you cannot follow an edge or a line back to this same point. All of theedges are arranged in paths oriented toward and terminating at one or more rootnodes. Figure 5-8 shows a basic DAG.

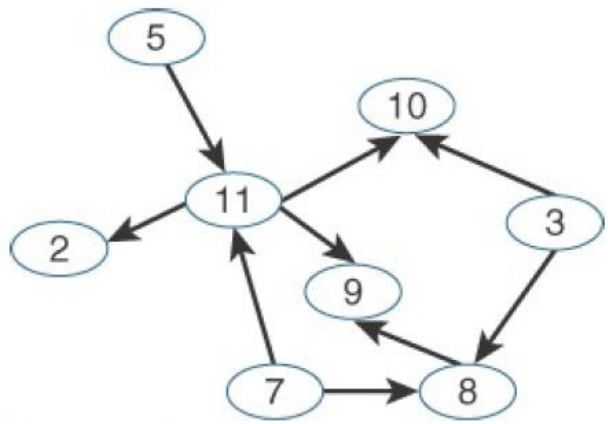

Figure 5-8 *Example of a Directed Acyclic Graph (DAG)*

A basic RPL process involves building a destination-oriented directed acyclicgraph (DODAG). A DODAG is a DAG rooted to one destination. In RPL, thisdestination occurs at a border router known as the DODAG root.

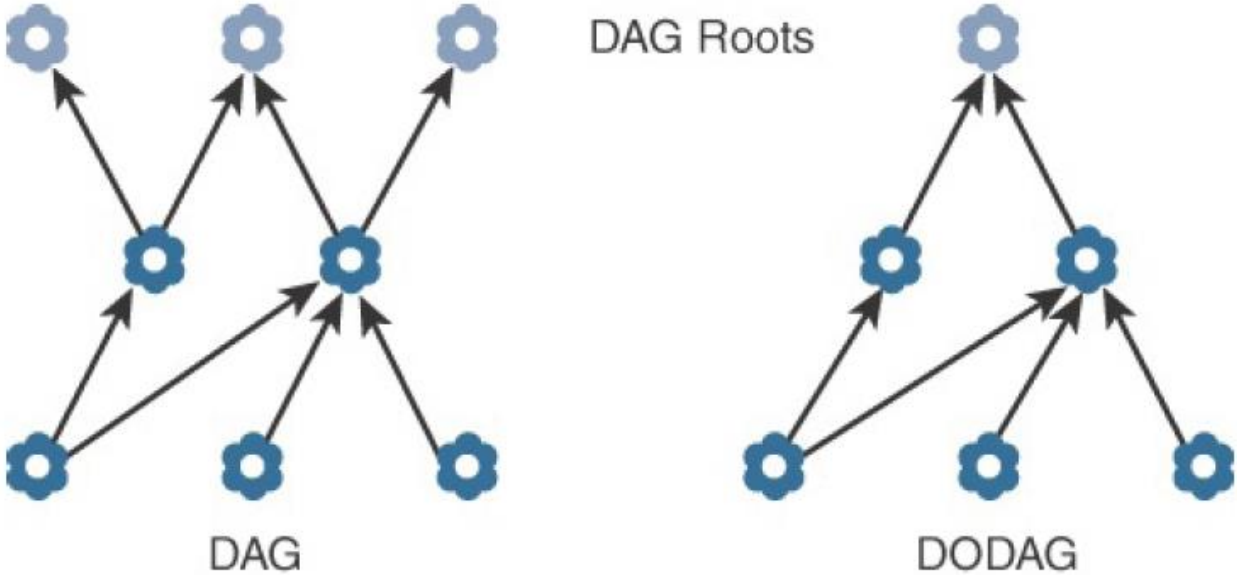

Figure 5-9 *DAG and DODAG Comparison*

In a DODAG, each node maintains up to three parents that provide a path to theroot. Typically, one of these parents is the preferred parent, which means it is thepreferred next hop for upward routes toward the root.The routing graph created by the set of DODAG parents across all nodes definesthe full set of upward routes. RPL protocol implementation should ensure thatroutes are loop free by disallowing nodes from selected DODAG parents that arepositioned further away from the border router.

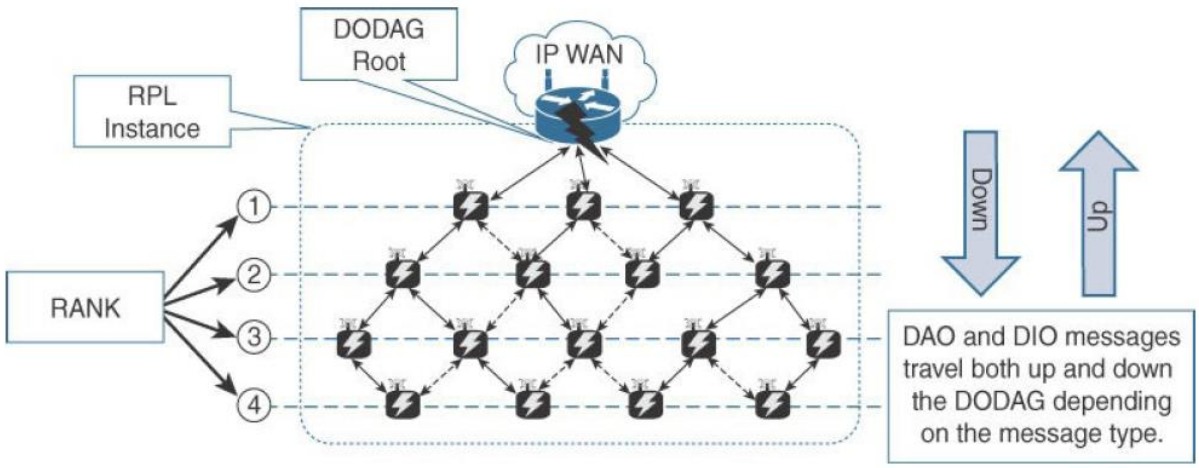

Figure 5-10 *RPL Overview*

As illustrated in Figure 5-10, DAO andDIO messages move both up and down the DODAG, depending on the exactmessage type.

5.4 Profiles and Compliances

Leveraging the Internet Protocol suite forsmart objects involves a collection of protocols and options that must work incoordination with lower and upper layers. Therefore, profile definitions,certifications, and promotion by alliances can help implementers developsolutions that guarantee interoperability and/or interchangeability of devices.

5.4.1 Internet Protocol for Smart Objects (IPSO) Alliance

Established in 2008, the Internet Protocol for Smart Objects (IPSO) Alliance hashad its objective evolve over years. The alliance initially focused on promotingIP as the premier solution for smart objects communications. Today, it is morefocused on how to use IP, with the IPSO Alliance organizing interoperabilitytests between alliance members to validate that IP for smart objects can worktogether and properly implement industry standards. The IPSO Alliance does not define technologies, as that is the role of the IETF and other standard

Organizations, but it documents the use of IP-based technologies for various IoTuse cases and participates in educating the industry.

5.4.2 Wi-SUN Alliance

The Wi-SUN Alliance is an example of efforts from the industry to define acommunication profile that applies to specific physical and data link layerprotocols. Currently, Wi-SUN's main focus is on the IEEE 802.15.4g protocoland its support for multiservice and secure IPv6 communications withapplications running over the UDP transport layer.
The utilities industry is the main area of focus for the Wi-SUN Alliance. The Wi-SUN field area network (FAN) profile enables smart utility networks to provideresilient, secure, and cost-effective connectivity with extremely good coverage ina range of topographic environments, from dense urban neighborhoods to ruralareas.

5.4.3 Thread

A group of companies involved with smart object solutions for consumerscreated the Thread Group. This group has defined an IPv6-based wireless profilethat provides the best way to connect more than 250 devices into a low-power,wireless mesh network. The wireless technology used by Thread is IEEE802.15.4, which is different from Wi-SUN's IEEE 802.15.4g.

5.4.4 IPv6 Ready Logo

Initially, the IPv6 Forum ensured the promotion of IPv6 around the world. OnceIPv6 implementations became widely available, the need for interoperability andcertification led to the creation of the IPv6 Ready Logo program.The IPv6 Ready Logo program has established conformance and interoperabilitytesting programs with the intent of increasing user confidence whenimplementing IPv6. The IPv6 Core and specific IPv6 components, such asDHCP, IPsec, and customer edge router certifications, are in place. Thesecertifications have industry-wide recognition, and many products are alreadycertified. An IPv6 certification effort specific to IoT is currently under definitionfor the program.

Chapter 6

Application Protocols for IoT

The IoT application protocols you select should becontingent on the use cases and vertical industries they apply to. In addition, IoTapplication protocols are dependent on the characteristics of the lower layers themselves. For example, application protocols that are sufficient for genericnodes and traditional networks often are not well suited for constrained nodesand networks.

6.1 The Transport Layer

Selection of a protocol for the transport layer as supported by the TCP/IP architecture in the context of IoT networks. With theTCP/IP protocol, two main protocols are specified for the transport layer.

Transmission Control Protocol (TCP): This connection-oriented
protocol requires a session to get established between the source anddestination before exchanging data. You can view it as an equivalent to atraditional telephone conversation, in which two phones must be connectedand the communication link established before the parties can talk.

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

6.2 IoT Application Transport Methods

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.

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

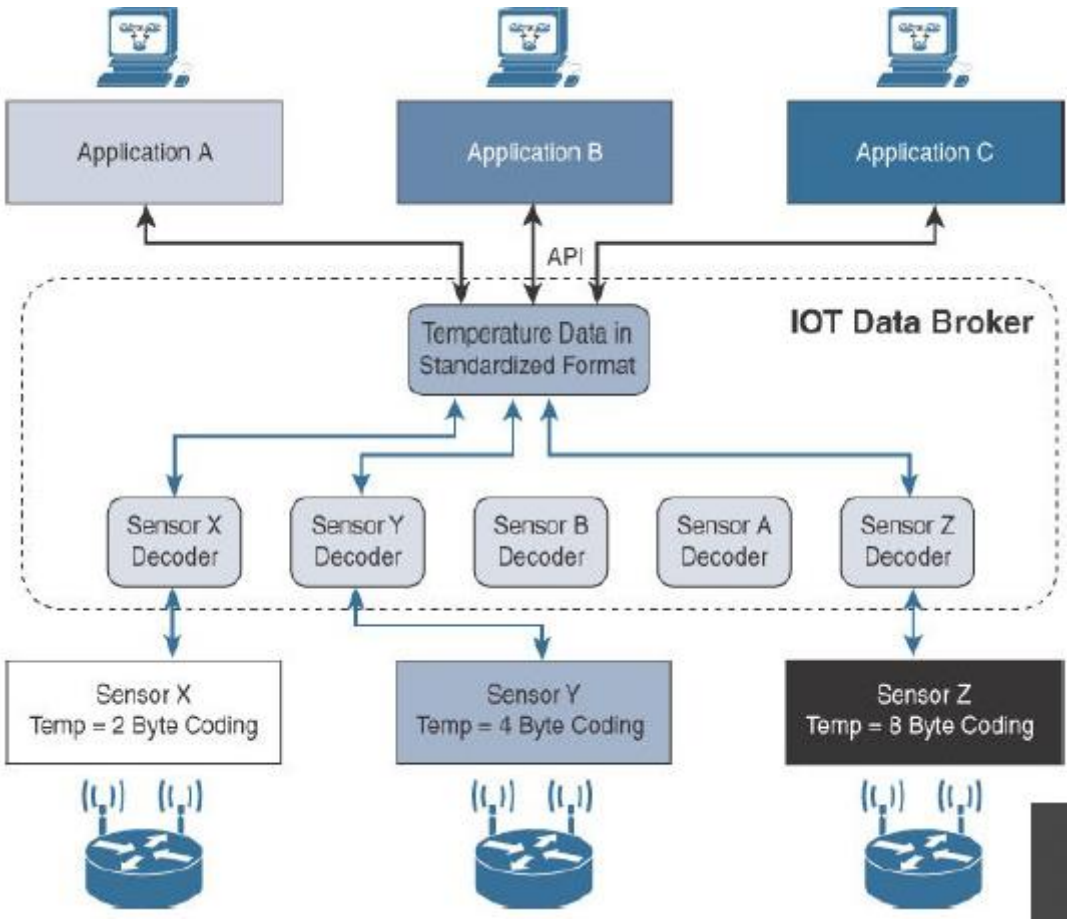

Figure 1: IOT data broker

Imagine expanding Example 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. The solution to this problem is to use an IoT data broker, as detailed in Figure 1. 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. In the above fig 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 1 can access this temperature data without having to deal with decoding multiple temperature data formats.

SCADA

In the world of networking technologies and protocols, IoT is relatively new. Combined with the fact that IP is the de facto standard for computer networking in general, older protocols that connected sensors and actuators have evolved and adapted themselves to utilize IP.
A prime example of this evolution is supervisory control and data acquisition (SCADA). Designed decades ago, SCADA is an automation control system that was initially implemented without IP over serial links, before being adapted to Ethernet and IPv4.

A Little Background on SCADA

For many years, vertical industries have developed communication protocols that fit their specific requirements. Many of them were defined and implemented when the most common networking technologies were serial link-based, such as RS-232 and RS-485. This led to SCADA networking protocols, which were well structured compared to the protocols described in the previous section, running directly over serial physical and data link layers.

At a high level, SCADA systems collect sensor data and telemetry from remote devices, while also providing the ability to control them. Used in today's networks, SCADA systems allow global, real-time, data-driven decisions to be made about how to improve business processes.

Adapting SCADA for IP

the 1990s, the rapid adoption of Ethernet networks in the industrial world
drove the evolution of SCADA application layer protocols. For example, the IEC adopted the Open System Interconnection (OSI) layer model to define its protocol framework.
To further facilitate the support of legacy industrial protocols over IP networks, protocol specifications were updated and published, documenting the use of IP for each protocol.
This included assigning TCP/UDP port numbers to theprotocols, 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.

Like many of the other SCADA protocols, 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.

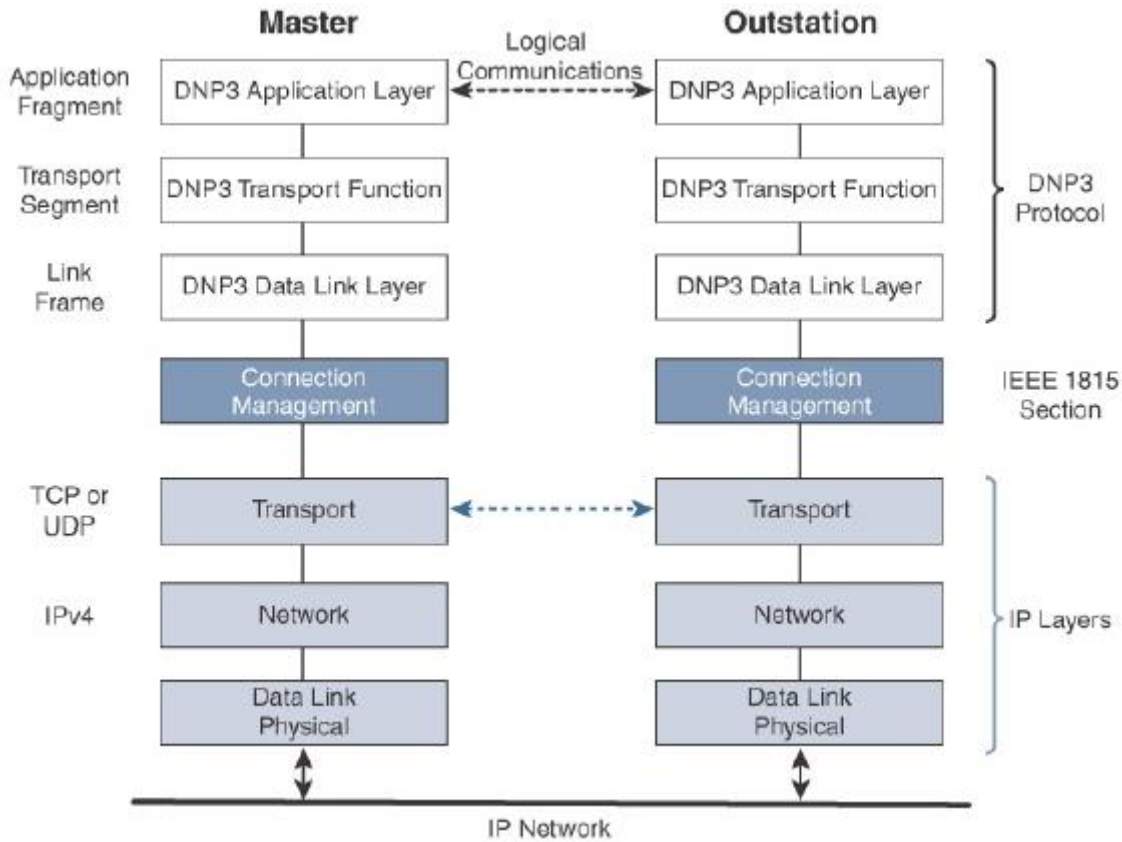

Fig 2: Protocol Stack for Transporting Serial DNP3 SCADA over IP

In Figure 2, 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

Generic Web-Based Protocols

web-based protocols have become common in consumer and enterprise applications and services. Therefore, it makes sense to try to leverage these protocols when developing IoT applications, services, and devices in order to ease the integration of data and devices from prototyping to production. The level of familiarity with generic web-based protocols is high.

Therefore, programmers with basic web programming skills can work on IoT applications, and this may lead to innovative ways to deliver and handle real-time IoT data.For example, 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. This notification allows technicians and engineers to immediately start working on this alert. In addition to a generally high level of familiarity with web-based protocols, scaling methods for web environments are also well understood—and this is crucial when developing consumer applications for potentially large numbers of IoT devices.

IoT Application Layer Protocols

When considering constrained networks and/or a large-scale deployment of constrained nodes, verbose web-based and data model protocols, as discussed in the previous section, may be too heavy for IoT applications. To address this problem, 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 3 highlights their position in a common IoT protocol stack.

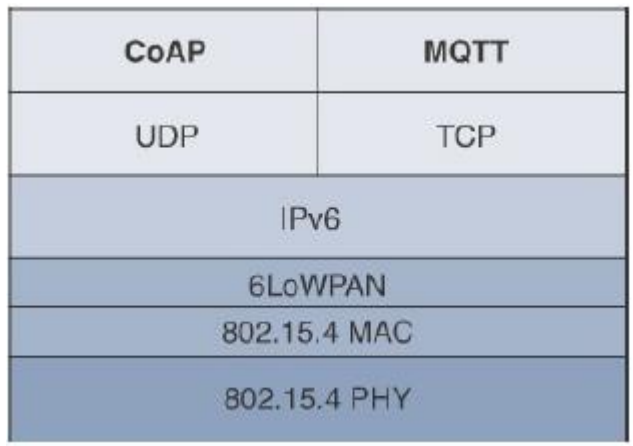

Fig 3 Example of a High-Level IoT Protocol Stack for CoAP and MQTT

CoAP and MQTT are naturally at the top of this sample IoT stack, based on an IEEE 802.15.4 mesh network. While there are a few exceptions, you will almost always find CoAP deployed over UDP and MQTT running over TCP. The following sections take a deeper look at CoAP and MQTT.

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. (For more information on the IETF CoRE working group,

The CoAP framework defines simple and flexible ways to manipulate sensors and actuators for data or device management. The IETF CoRE working group has published multiple standards-track specifications for 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)

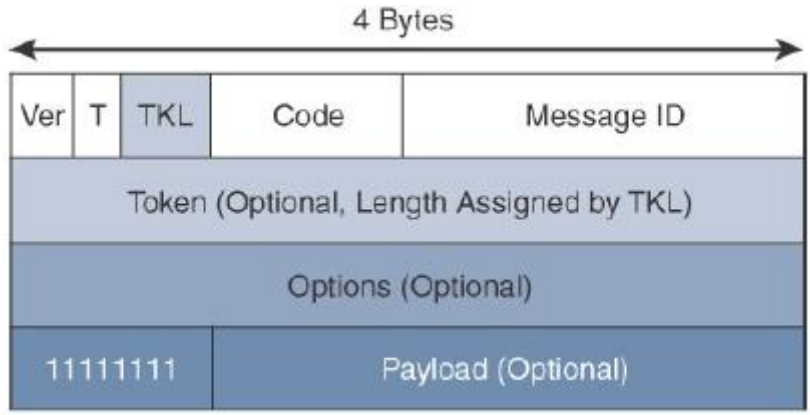

Fig 4 CoAP Message Format

From a formatting perspective, a CoAP message is composed of a short fixedlength Header field (4 bytes), a variable-length but mandatory Token field (0–8 bytes), Options fields if necessary, and the Payload field. Figure 4 details the CoAP message format, which delivers low overhead while decreasing parsing complexity.

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 1:CoAP Message Fields

While running over UDP, CoAP offers a reliable transmission of messages when a CoAP header is marked as "confirmable." In addition, CoAP supports basic cogestion 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, as shown in Figure 5. If a recipient can't process a nonconfirmable message, a reset message is sent.

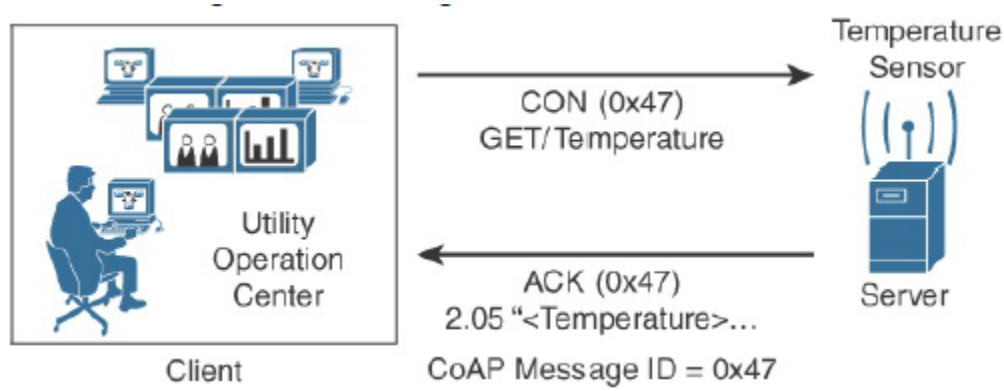

Fig 5. CoAP reliable Transmision Example

Figure 5 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 5 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.

Message Queuing Telemetry Transport (MQTT)

At the end of the 1990s, engineers from IBM and Arcom (acquired in 2006 by Eurotech) were looking for a reliable, lightweight, and cost-effective protocol to monitor and control a large number of sensors and their data from a central server location, as typically used by the oil and gas industries. Their research resulted in the development and implementation of the Message Queuing Telemetry Transport (MQTT) protocol that is now standardized by the Organization for the Advancement of Structured Information Standards (OASIS).
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 were some of the rationales for the selection of a client/server and publish/subscribe framework based on the TCP/IP architecture, as shown in Figure 6.

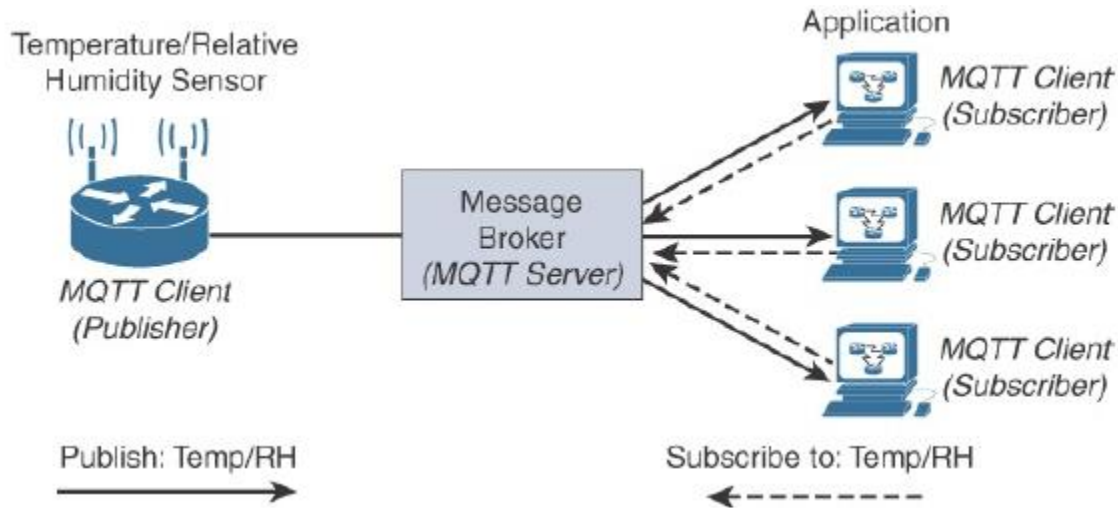

Fig 6. MQTT publish / subscribe framework

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 the example illustrated in Figure 6, 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, 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, is well known. A great example is the collaboration and social networking application Twitter.

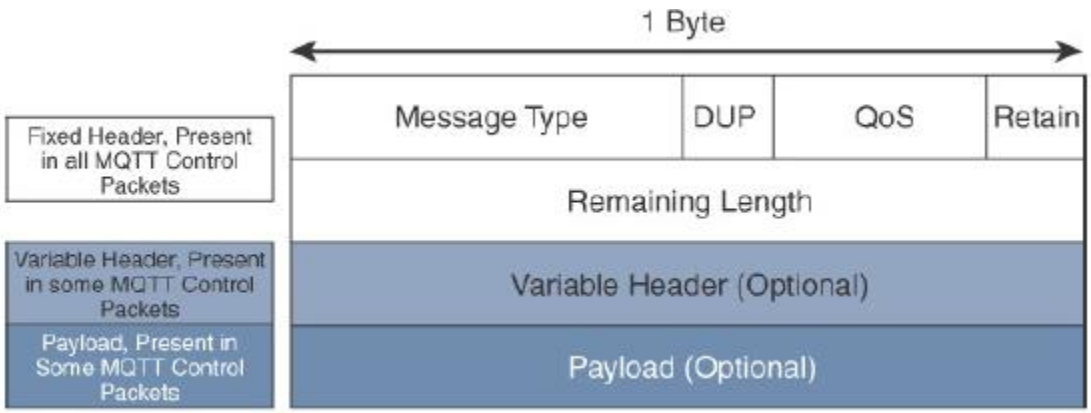

Fig 7. *MQTT Message Format*

MQTT message types are summarized in the below table 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 2: MQTT message format

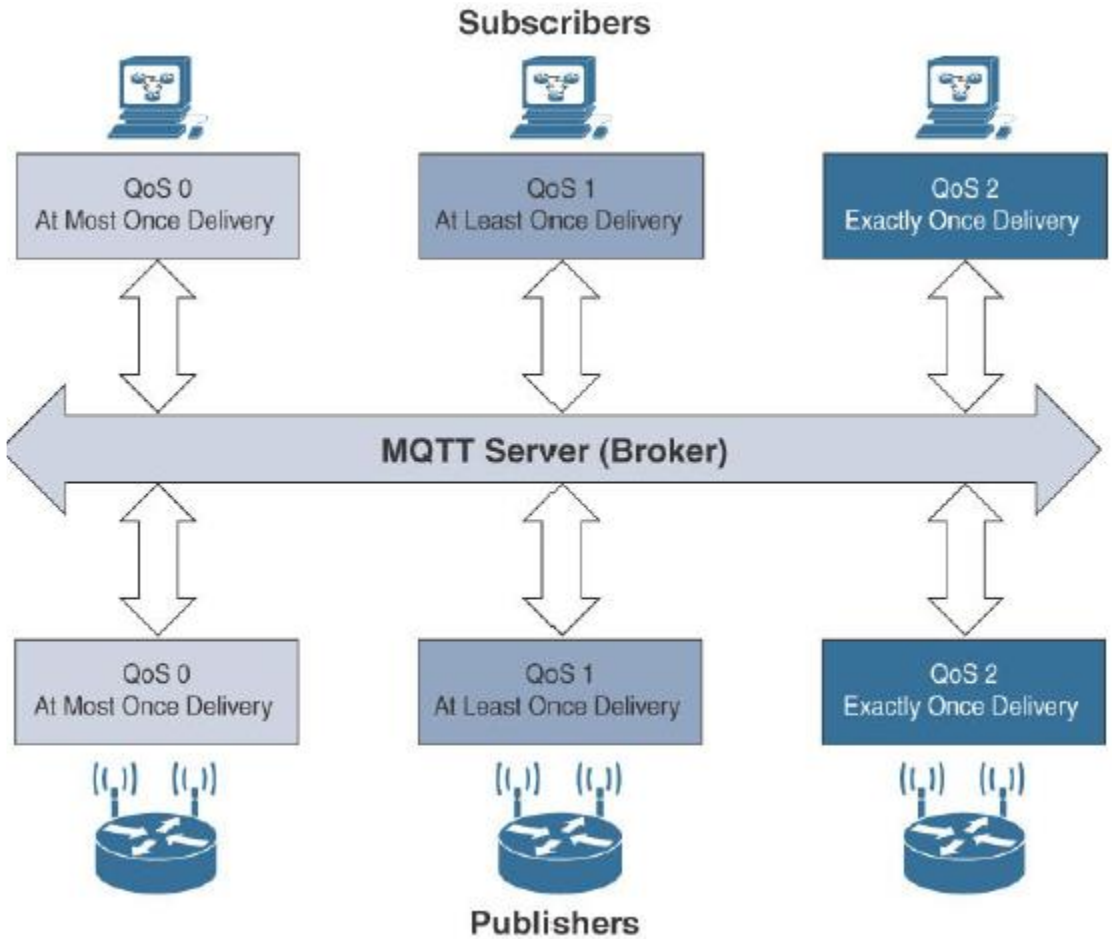

Fig 8 MQTT QoS Flows

The MQTT protocol offers three levels of quality of service (QoS). QoS for MQTT as in fig 8 is implemented when exchanging application messages with publishers or subscribers, and it is different from the IP QoS that most people are familiar with. The delivery protocol is symmetric. This means the client and server can each take the role of either sender or receiver. The delivery protocol is concerned solely with the delivery of an application message from a single sender to a single receiver. These are the three levels of MQTT QoS:

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.

Table 3 provides an overview of the differences between MQTT and CoAP, along with their strengths and weaknesses from an IoT perspective.

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 3 Comparison Between CoAP and MQTT