# Atria Institute of Technology

## Department of Information Science and Engineering

**Bengaluru-560024**

## ACADEMIC YEAR: 2021-2022

## EVEN SEMESTER NOTES

**Semester : 8th Semester**

**Subject Name : Internet of Things Technology**

**Subject Code : 18CS81**

**Faculty Name : Mrs. Vijayalakshmi V**

# MODULE 1

## What Is IoT?

IoT is a technology transition in which devices will allow us to sense and control the physical world by making objects smarter and connecting them through an intelligent network.

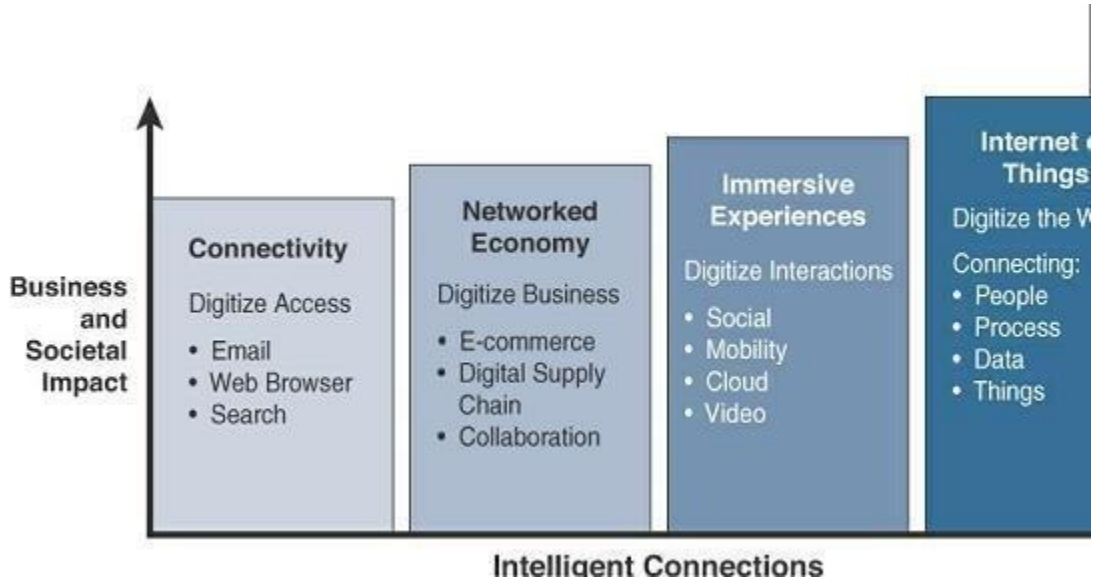
**GOAL:** The basic premise and goal of IoT is to "connect the unconnected." This means that objects that are not currently joined to a computer network, namely the Internet, will be conn ected so that they can communicate and interact with people and other objects.

When objects and machines can be sensed and controlled remotely across a network, a tighter integration between the physical world and computers is enabled.

This allows for improvements in the areas of efficiency, accuracy, automation, and the enable ment of advanced applications.

**GENESIS OF IOT**

The person credited with the creation of the term "Internet of Things" is Kevin Ashton. While working for Procter & Gamble in 1999, Kevin used this phrase to explain a new idea related to linking the company's supply chain to the Internet.




the evolution of the Internet can be categorized into four phases. Each of these phases has had a profound impact on our society and our lives. These four phases are further defined in Table below.

| Internet Phase | Definition |
|---|---|
| Connectivity (Digitize access) | This phase connected people to email, web services, and search so that information is easily accessed. |
| Networked Economy (Digitize business) | This phase enabled e-commerce and supply chain enhancements along with collaborative engagement to drive increased efficiency in business processes. |
| Immersive Experiences (Digitize interactions) | This phase extended the Internet experience to encompass widespread video and social media while always being connected through mobility. More and more applications are moved into the cloud. |
| Internet of Things (Digitize the world) | This phase is adding connectivity to objects and machines in the world around us to enable new services and experiences. It is connecting the unconnected. |

## IOT AND DIGITIZATION

*IoT* and *digitization* are terms that are often used interchangeably. In most contexts, this duality is fine, but there are key differences to be aware of.

At a high level, IoT focuses on connecting "things," such as objects and machines, to a computer network, such as the Internet. IoT is a well-understood term used across the industry as a whole. On the other hand, digitization can mean different things to different people but generally encompasses the connection of "things" with the data they generate and the business insights that result.

Digitization, as defined in its simplest form, is the conversion of information into a digital format. Digitization has been happening in one form or another for several decades. For example, the whole photography industry has been digitized. Pretty much everyone has digital cameras these days, either standalone devices or built into their mobile phones. Almost no one buys film and takes it to a retailer to get it developed. The digitization of photography has completely changed our experience when it comes to capturing images.

# CONVERGENCE OF IT AND OT

Until recently, information technology (IT) and operational technology (OT) have for the most part lived in separate worlds. IT supports connections to the Internet along with related data and technology systems and is focused on the secure flow of data across an organization. OT monitors and controls devices and processes on physical operational systems. These systems include assembly lines, utility distribution networks, production facilities, roadway systems, and many more. Typically, IT did not get involved with the production and logistics of OT environments.

Management of OT is tied to the lifeblood of a company. For example, if the network connecting the machines in a factory fails, the machines cannot function, and production may come to a standstill, negatively impacting business on the order of millions of dollars. On the other hand, if the email server (run by the IT department) fails for a few hours, it may irritate people, but it is unlikely to impact business at anywhere near the same level. **Table below highlights some of the differences between IT and OT networks and their various challenges**.

| Criterion | Industrial OT Network | Enterprise IT Network |
|---|---|---|
| Operational focus | Keep the business operating 24x7 | Manage the computers, data, and employee communication system in a secure way |
| Priorities | 1. Availability<br>2. Integrity<br>3. Security | 1. Security<br>2. Integrity<br>3. Availability |
| Types of data | Monitoring, control, and supervisory data | Voice, video, transactional, and bulk data |
| Security | Controlled physical access to devices | Devices and users authenticated to the network |
| Implication of failure | OT network disruption directly impacts business | Can be business impacting, depending on industry, but workarounds may be possible |
| Network upgrades (software or hardware) | Only during operational maintenance windows | Often requires an outage window when workers are not onsite; impact can be mitigated |
| Security vulnerability | Low: OT networks are isolated and often use proprietary protocols | High: continual patching of hosts is required, and the network is connected to Internet and requires vigilant protection |

*Source*: Maciej Kranz, *IT Is from Venus, OT Is from Mars*, blogs.cisco.com/digital/it-is-from-venus-ot-is-from-mars, July 14, 2015.

## IOT CHALLENGES

**The most significant challenges and problems that IoT is currently facing are**

| Challenge | Description |
|---|---|
| Scale | While the scale of IT networks can be large, the scale of OT can be several orders of magnitude larger. For example, one large electrical utility in Asia recently began deploying IPv6-based smart meters on its electrical grid. While this utility company has tens of thousands of employees (which can be considered IP nodes in the network), the number of meters in the service area is tens of millions. This means the scale of the network the utility is managing has increased by more than 1,000-fold! Chapter 5, "IP as the IoT Network Layer," explores how new design approaches are being developed to scale IPv6 networks into the millions of devices. |
| Security | With more "things" becoming connected with other "things" and people, security is an increasingly complex issue for IoT. Your threat surface is now greatly expanded, and if a device gets hacked, its connectivity is a major concern. A compromised device can serve as a launching point to attack other devices and systems. IoT security is also pervasive across just about every facet of IoT. For more information on IoT security, see Chapter 8, "Securing IoT." |

| | |
|---|---|
| Privacy | As sensors become more prolific in our everyday lives, much of the data they gather will be specific to individuals and their activities. This data can range from health information to shopping patterns and transactions at a retail establishment. For businesses, this data has monetary value. Organizations are now discussing who owns this data and how individuals can control whether it is shared and with whom. |
| Big data and data analytics | IoT and its large number of sensors is going to trigger a deluge of data that must be handled. This data will provide critical information and insights if it can be processed in an efficient manner. The challenge, however, is evaluating massive amounts of data arriving from different sources in various forms and doing so in a timely manner. See Chapter 7 for more information on IoT and the challenges it faces from a big data perspective. |
| Interoperability | As with any other nascent technology, various protocols and architectures are jockeying for market share and standardization within IoT. Some of these protocols and architectures are based on proprietary elements, and others are open. Recent IoT standards are helping minimize this problem, but there are often various protocols and implementations available for IoT networks. The prominent protocols and architectures—especially open, standards-based implementations—are the subject of this book. For more information on IoT architectures, see Chapter 2, "IoT Network Architecture and Design." Chapter 4, "Connecting Smart Objects," Chapter 5, "IP as the IoT Network Layer," and Chapter 6, "Application Protocols for IoT," take a more in-depth look at the protocols that make up IoT. |

## IoT Network Architecture and Design

The unique challenges posed by IoT networks and how these challenges have driven new architectural models.

- Drivers Behind New Network Architectures
- Comparing IoT Architectures.
- A Simplified IoT Architecture
- The Core IoT Functional Stack
- IoT Data Management and Compute Stack

## DRIVERS BEHIND NEW NETWORK ARCHITECTURES

This begins by comparing how using an architectural blueprint to construct a house is similar to the approach we take when designing a network. Take a closer look at some of the differences between IT and IoT networks, with a focus on the IoT requirements that are driving new network architectures, and considers what adjustments are needed.

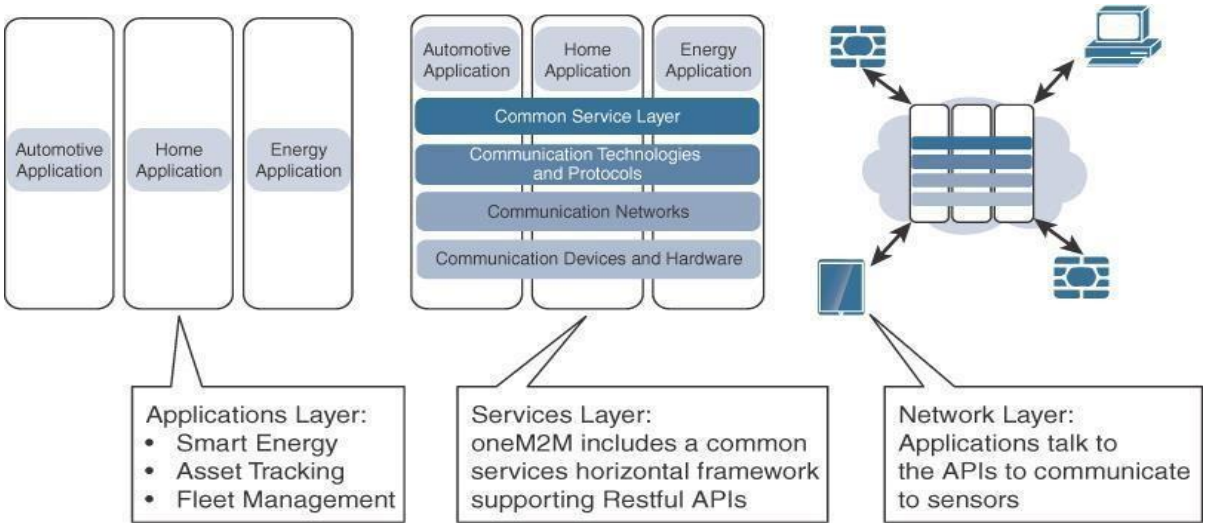
| Challenge | Description | IoT Architectural Change Required |
|---|---|---|
| Scale | The massive scale of IoT end-points (sensors) is far beyond that of typical IT networks. | The IPv4 address space has reached exhaustion and is unable to meet IoT's scalability requirements. Scale can be met only by using IPv6. IT networks continue to use IPv4 through features like Network Address Translation (NAT). |
| Security | IoT devices, especially those on wireless sensor networks (WSNs), are often physically exposed to the world. | Security is required at every level of the IoT network. Every IoT endpoint node on the network must be part of the overall security strategy and must support device-level authentication and link encryption. It must also be easy to deploy with some type of a zero-touch deployment model. |
| Devices and networks constrained by power, CPU, memory, and link speed | Due to the massive scale and longer distances, the networks are often constrained, lossy, and capable of supporting only minimal data rates (tens of bps to hundreds of Kbps). | New last-mile wireless technologies are needed to support constrained IoT devices over long distances. The network is also constrained, meaning modifications need to be made to traditional network-layer transport mechanisms. |
| The massive volume of data generated | The sensors generate a massive amount of data on a daily basis, causing network bottlenecks and slow analytics in the cloud. | Data analytics capabilities need to be distributed throughout the IoT network, from the edge to the cloud. In traditional IT networks, analytics and applications typically run only in the cloud. |
| Support for legacy devices | An IoT network often comprises a collection of modern, IP-capable endpoints as well as legacy, non-IP devices that rely on serial or proprietary protocols. | Digital transformation is a long process that may take many years, and IoT networks need to support protocol translation and/or tunneling mechanisms to support legacy protocols over standards-based protocols, such as Ethernet and IP. |
| The need for data to be analyzed in real time | Whereas traditional IT networks perform scheduled batch processing of data, IoT data needs to be analyzed and responded to in real-time. | Analytics software needs to be positioned closer to the edge and should support real-time streaming analytics. Traditional IT analytics software (such as relational databases or even Hadoop), are better suited to batch-level analytics that occur after the fact. |

# COMPARING IOT ARCHITECTURES

### The oneM2M IoT Standardized Architecture

In an effort to standardize the rapidly growing field of machine-to-machine (M2M) communications, the European Telecommunications Standards Institute (ETSI) created the M2M Technical Committee in 2008. The goal of this committee was to create a common architecture that would help accelerate the adoption of M2M applications and devices. Over time, the scope has expanded to include the Internet of Things.

One of the greatest challenges in designing an IoT architecture is dealing with the heterogeneity of devices, software, and access methods. By developing a horizontal platform architecture, oneM2M is developing standards that allow interoperability at all levels of the IoT stack




*The Main Elements of the oneM2M IoT Architecture*

The oneM2M architecture divides IoT functions into three major domains: the application layer, the services layer, and the network layer

- **Applications layer:** The oneM2M architecture gives major attention to connectivity between devices and their applications. This domain includes the application-layer protocols and attempts to standardize northbound API definitions for interaction with business intelligence (BI) systems. Applications tend to be industry-specific and have their own sets of data models, and thus they are shown as vertical entities.

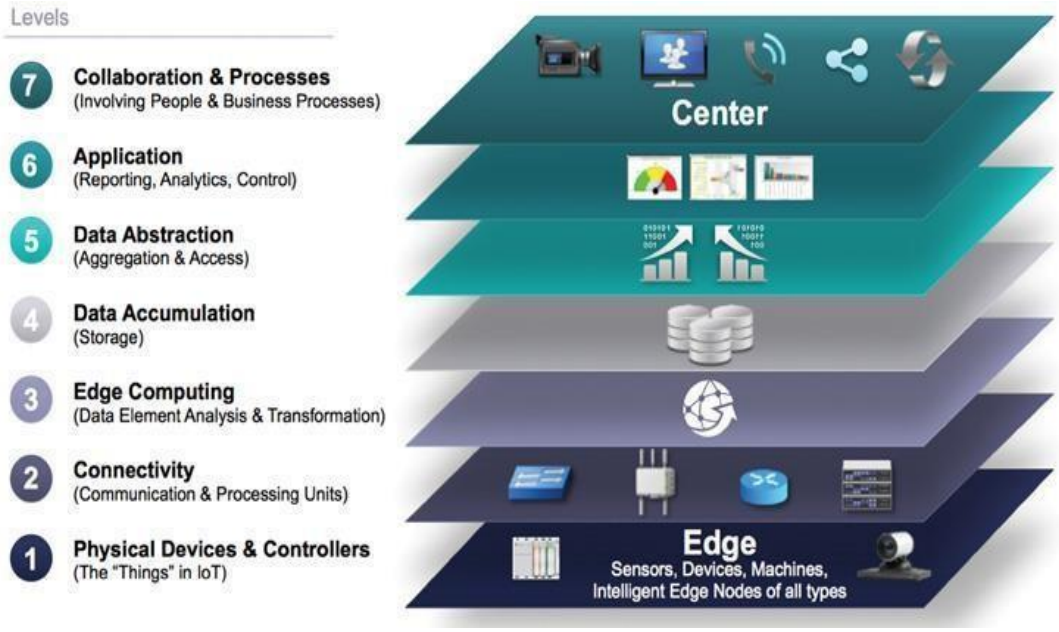
- **Services layer:** This layer is shown as a horizontal framework across the vertical industry applications. At this layer, horizontal modules include the physical network that the IoT applications run on, the underlying management protocols, and the hardware. Examples include backhaul communications via cellular, MPLS networks, VPNs, and so on. Riding on top is the common services layer.

- **Network layer:** This is the communication domain for the IoT devices and endpoints. It includes the devices themselves and the communications network that links them. Embodiments of this communications infrastructure include wireless mesh technologies, such as IEEE 802.15.4, and wireless point-to-multipoint systems, such as IEEE 801.11ah.

## The IoT World Forum (IoTWF) Standardized Architecture

This publish a seven-layer IoT architectural reference model.

- While various IoT reference models exist, the one put forth by the IoT World Forum offers a clean, simplified perspective on IoT and includes edge computing, data storage, and access. It provides a succinct way of visualizing IoT from a technical perspective. Each of the seven layers is broken down into specific functions, and security encompasses the entire model.




Using this reference model, we are able to achieve the following:

1. Decompose the IoT problem into smaller parts
2. Identify different technologies at each layer and how they relate to one another
3. Define a system in which different parts can be provided by different vendors
4. Have a process of defining interfaces that leads to interoperability
5. Define a tiered security model that is enforced at the transition points between levels

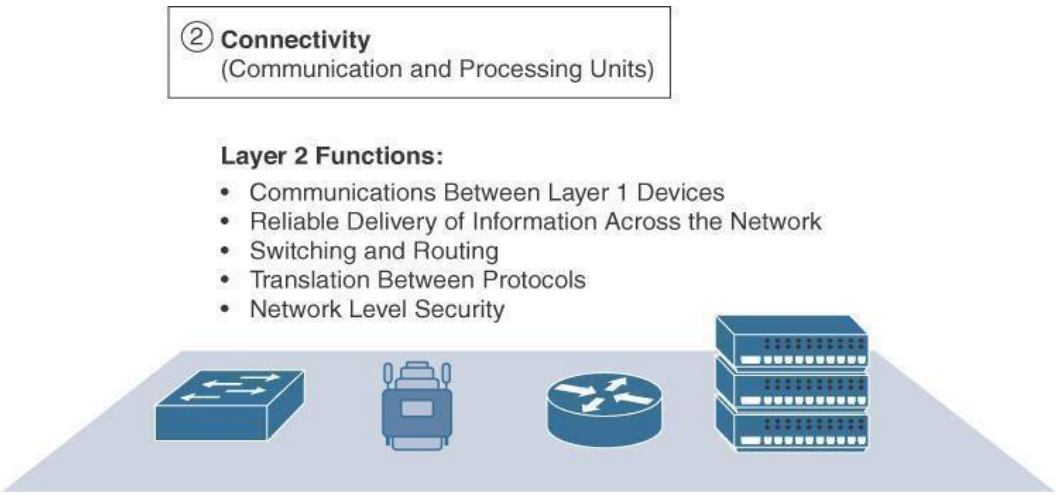
### Layer 1: Physical Devices and Controllers Layer

The first layer of the IoT Reference Model is the physical devices and controllers layer. This layer is home to the "things" in the Internet of Things, including the various endpoint devices and sensors that send and receive information. The size of these "things" can range from almost microscopic sensors to giant machines in a factory. Their primary function is generating data and being capable of being queried and/or controlled over a network.

**Layer 2: Connectivity Layer**

In the second layer of the IoT Reference Model, the focus is on connectivity. The most important function of this IoT layer is the reliable and timely transmission of data. More specifically, this includes transmissions between Layer 1 devices and the network and between the network and information processing that occurs at Layer 3 (the edge computing layer).

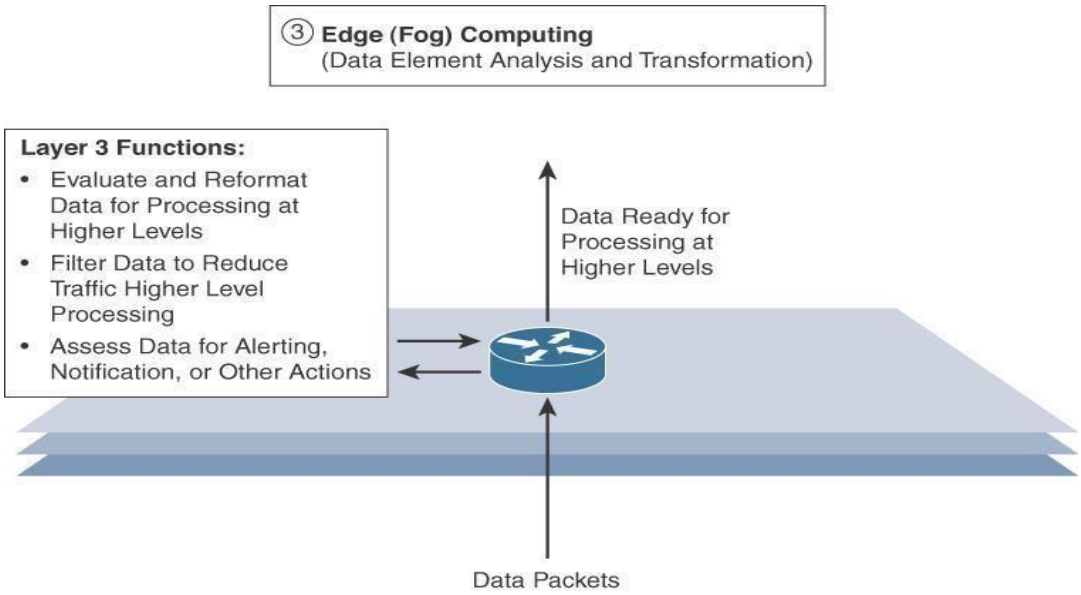
.




*IoT Reference Model Connectivity Layer Functions*

**Layer 3: Edge Computing Layer**

Edge computing is the role of Layer 3. Edge computing is often referred to as the "fog" layer and is discussed in the section "Fog Computing," later in this chapter. At this layer, the emphasis is on data reduction and converting network data flows into information that is ready for storage and processing by higher layers. One of the basic principles of this reference model is that information processing is initiated as early and as close to the edge of the network as possible




*IoT Reference Model Layer 3 Functions*

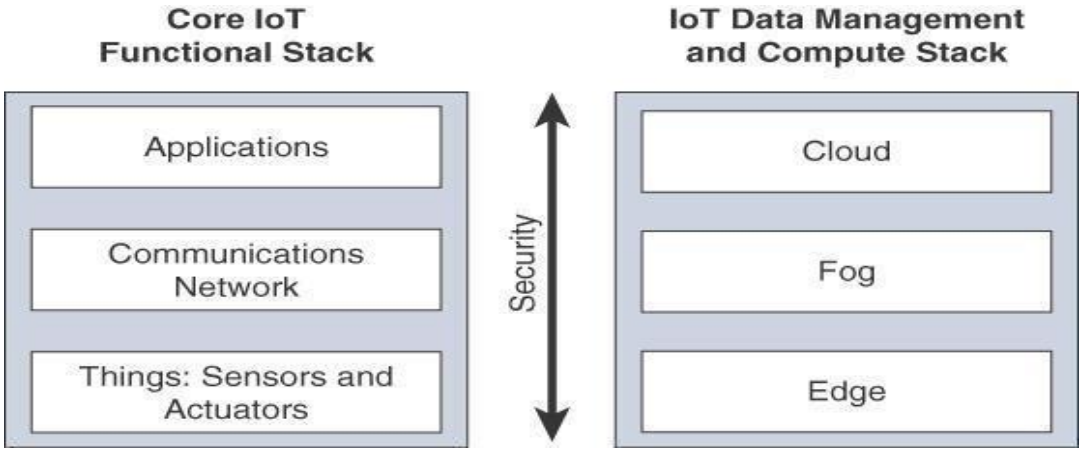
Another important function that occurs at Layer 3 is the evaluation of data to see if it can be filtered or aggregated before being sent to a higher layer. This also allows for data to be reformatted or decoded, making additional processing by other systems easier. Thus, a critical function is assessing the data to see if predefined thresholds are crossed and any action or alerts need to be sent.

**Upper Layers: Layers 4–7**

The upper layers deal with handling and processing the IoT data generated by the bottom layer. For the sake of completeness, Layers 4–7 of the IoT Reference Model are summarized in Table .

| IoT Reference Model Layer | Functions |
|---|---|
| Layer 4: Data accumulation layer | Captures data and stores it so it is usable by applications when necessary. Converts event-based data to query-based processing. |
| Layer 5: Data abstraction layer | Reconciles multiple data formats and ensures consistent semantics from various sources. Confirms that the data set is complete and consolidates data into one place or multiple data stores using virtualization. |
| Layer 6: Applications layer | Interprets data using software applications. Applications may monitor, control, and provide reports based on the analysis of the data. |
| Layer 7: Collaboration and processes layer | Consumes and shares the application information. Collaborating on and communicating IoT information often requires multiple steps, and it is what makes IoT useful. This layer can change business processes and delivers the benefits of IoT. |

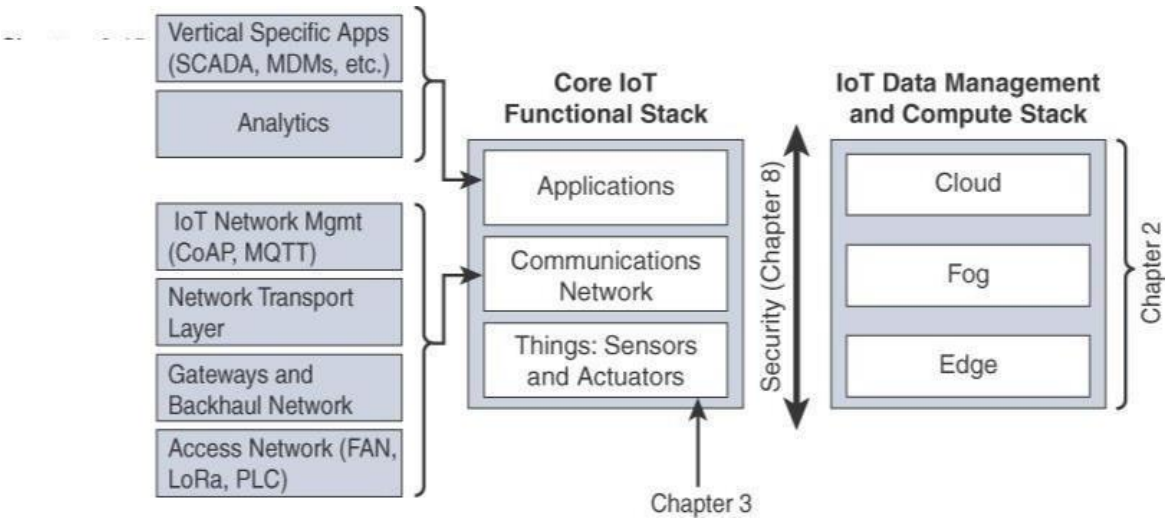
## A SIMPLIFIED IOT ARCHITECTURE




*Simplified IoT Architecture*

The presentation of the Core IoT Functional Stack in three layers is meant to simplify your understanding of the IoT architecture into its most foundational building blocks. The network communications layer of the IoT stack itself involves a significant amount of detail and incorporates a vast array of technologies.

Data management is aligned with each of the three layers of the Core IoT Functional Stack. The three data management layers are the edge layer (data management within the sensors themselves), the fog layer (data management in the gateways and transit network), and the cloud layer (data management in the cloud or central data center). An expanded view of the IoT architecture presented as below:




*Expanded View of the Simplified IoT Architecture*

The Core IoT Functional Stack can be expanded into sublayers containing greater detail and specific network functions. For example, the communications layer is broken down into four separate sublayers: the access network, gateways and backhaul, IP transport, and operations and management sublayers.

The applications layer of IoT networks is quite different from the application layer of a typical enterprise network. Instead of simply using business applications, IoT often involves a strong big data analytics component. One message that is stressed throughout this book is that IoT is not just about the control of IoT devices but, rather, the useful insights gained from the data generated by those devices. Thus, the applications layer typically has both analytics and industry-specific IoT control system components.

presented in Part II, and it gives you the tools you need to understand how these technologies are applied in key industries in Part III.

## THE CORE IOT FUNCTIONAL STACK

IoT networks are built around the concept of "things," or smart objects performing functions and delivering new connected services. These objects are "smart" because they use a combination of contextual information and configured goals to perform actions.

From an architectural standpoint, several components have to work together for an IoT network to be operational:

- "Things" layer:
- Communications network layer
- Access network sublayer
- Gateways and backhaul network sublayer
- Network transport sublayer
- IoT network management sublayer
- Application and analytics layer

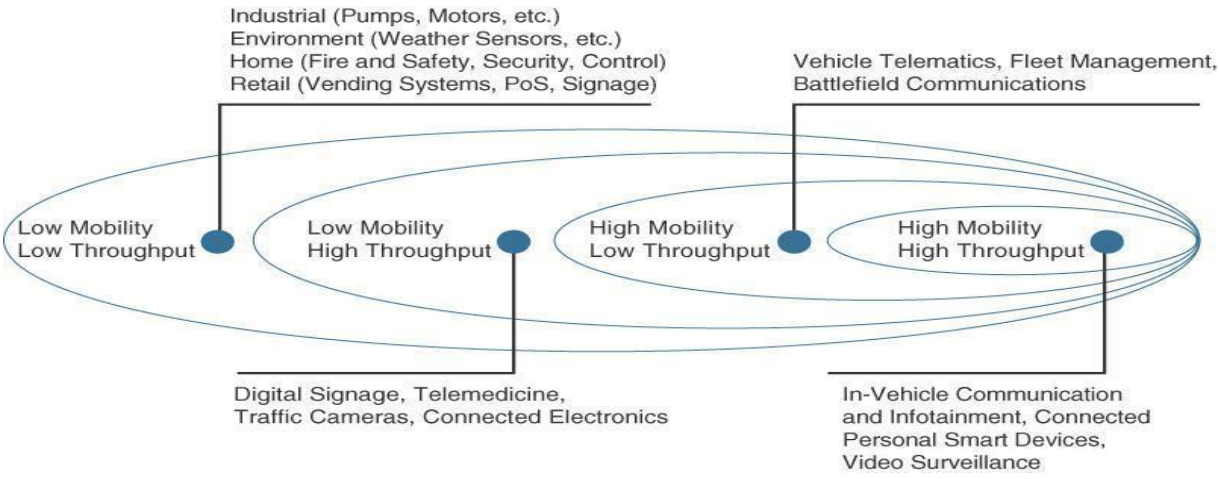
The following sections examine these elements and help you architect your IoT communication network.

**Layer 1: Things: Sensors and Actuators Layer**

"Smart Objects: The 'Things' in IoT," provides more in-depth information about smart objects. From an architectural standpoint, the variety of smart object types, shapes, and needs drive the variety of IoT protocols and architectures. One architectural classification could be:

- **Battery-powered or power-connected:** This classification is based on whether the object carries its own energy supply or receives continuous power from an external power source.
- **Mobile or static:** This classification is based on whether the "thing" should move or always stay at the same location. A sensor may be mobile because it is moved from one object to another or because it is attached to a movin
- **Low or high reporting frequency:** This classification is based on how often the object should report monitored parameters. A rust sensor may report values once a month. A motion sensor may report acceleration several hundred times per second.
- **Simple or rich data:** This classification is based on the quantity of data exchanged at each report cycle
- **Report range:** This classification is based on the distance at which the gateway is located. For example, for your fitness band to communicate with your phone, it needs to be located a few meters away at most.
- **Object density per cell:** This classification is based on the number of smart objects (with a similar need to communicate) over a given area, connected to the same gateway.

Below figure provides some examples of applications matching the combination of mobility and throughput requirements.




*Example of Sensor Applications Based on Mobility and Throughput*

**Layer 2: Communications Network Layer**

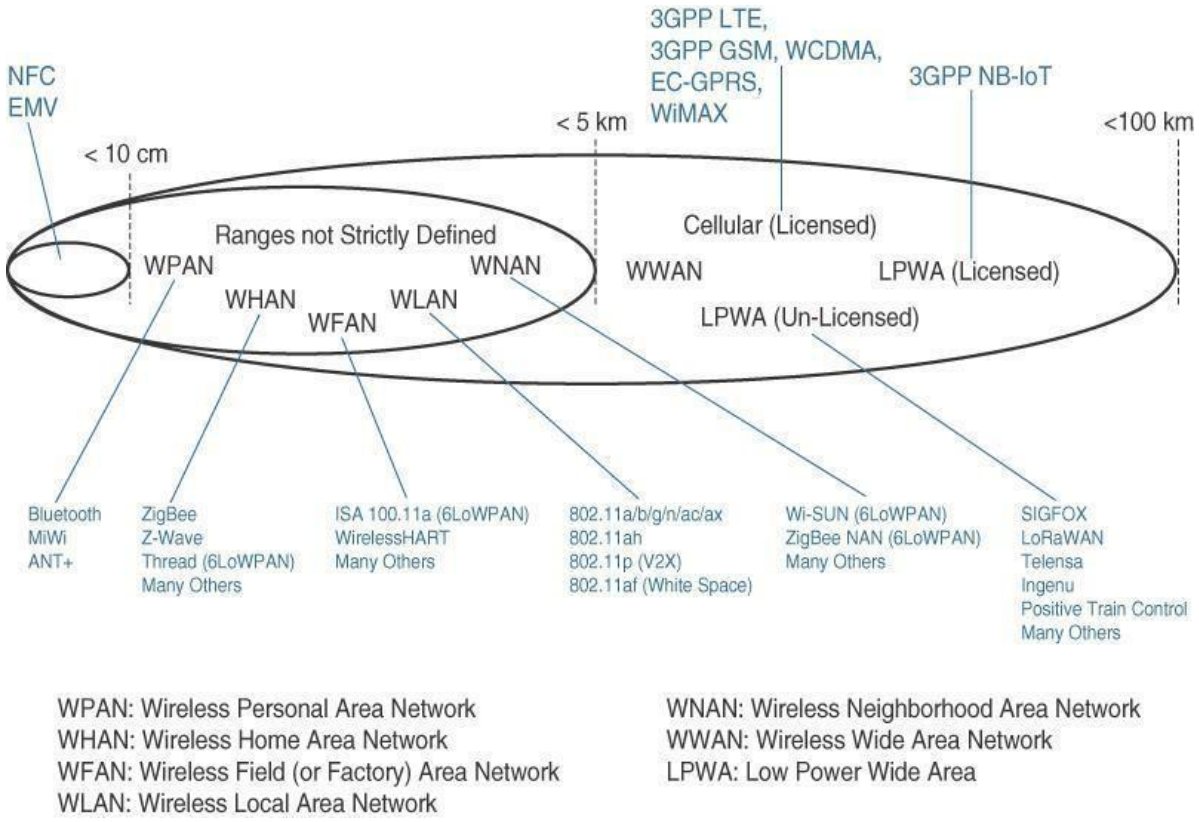
Once you have determined the influence of the smart object form factor over its transmission capabilities (transmission range, data volume and frequency, sensor density and mobility), you are ready to connect the object and communicate.

Compute and network assets used in IoT can be very different from those in IT environments. The difference in the physical form factors between devices used by IT and OT is obvious even to the most casual of observers. What typically drives this is the physical environment in which the devices are deployed. What may not be as inherently obvious, however, is their operational differences. The operational differences must be understood in order to apply the correct handling to secure the target assets.

**Access Network Sublayer**

There is a direct relationship between the IoT network technology you choose and the type of connectivity topology this technology allows. Each technology was designed with a certain number of use cases in mind (what to connect, where to connect, how much data to transport at what interval and over what distance). These use cases determined the frequency band that was expected to be most suitable, the frame structure matching the expected data pattern (packet size and communication intervals), and the possible topologies that these use cases illustrate.

One key parameter determining the choice of access technology is the range between the smart object and the information collector. Figure 2-9 lists some access technologies you may encounter in the IoT world and the expected transmission distances.



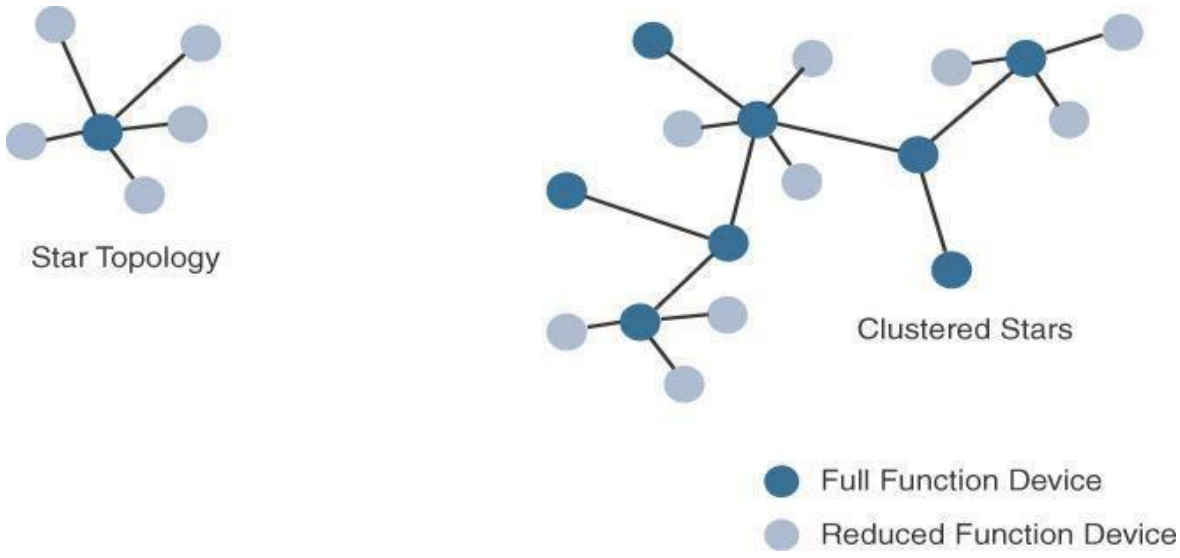

*Access Technologies and Distances*

✓ Range estimates are grouped by category names that illustrate the environment or the vertical where data collection over that range is expected. Common groups are as follows:

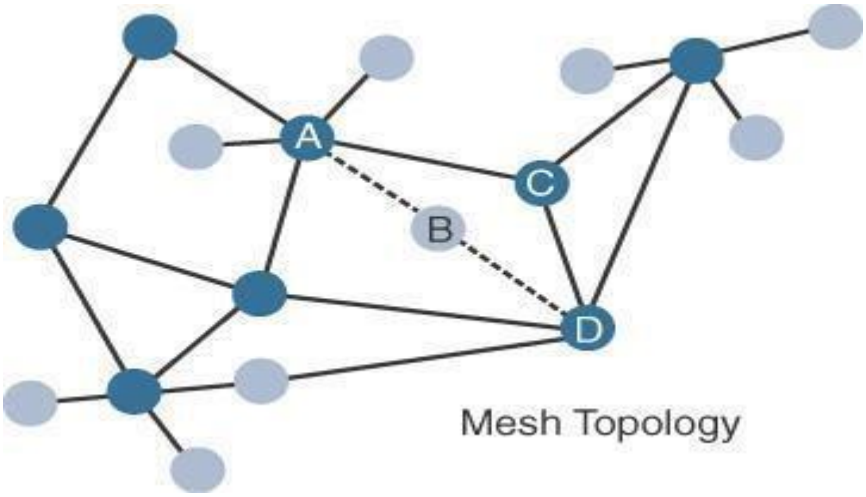
- **PAN (personal area network):** Scale of a few meters. This is the personal space around a person. A common wireless technology for this scale is Bluetooth.
- **HAN (home area network):** Scale of a few tens of meters. At this scale, common wireless technologies for IoT include ZigBee and Bluetooth Low Energy (BLE).
- **NAN (neighborhood area network):** Scale of a few hundreds of meters. The term NAN is often used to refer to a group of house units from which data is collected.
- **FAN (field area network):** Scale of several tens of meters to several hundred meters. FAN typically refers to an outdoor area larger than a single group of house units. The FAN is often seen as "open space" (and therefore not secured and not controlled).
- **LAN (local area network):** Scale of up to 100 m. This term is very common in networking, and it is therefore also commonly used in the IoT space when standard networking technologies (such as Ethernet or IEEE 802.11) are used.

✓ Similar ranges also do not mean similar topologies. Some technologies offer flexible connectivity structure to extend communication possibilities:

- **Point-to-point topologies**
- **Point-to-multipoint**




*Star and Clustered Star Topologies*




Comparison of the main solutions from an architectural angle.

| Technology | Type and Range | Architectural Characteristics |
|---|---|---|
| Ethernet | Wired, 100 m max | Requires a cable per sensor/sensor group; adapted to static sensor position in a stable environment; range is limited; link is very reliable |
| Wi-Fi (2.4 GHz, 5 GHz) | Wireless, 100 m (multipoint) to a few kilometers (P2P) | Can connect multiple clients (typically fewer than 200) to a single AP; range is limited; adapted to cases where client power is not an issue (continuous power or client battery recharged easily); large bandwidth available, but interference from other systems likely; AP needs a cable |
| 802.11ah (HaloW, Wi-Fi in sub-1 GHz) | Wireless, 1.5 km (multipoint), 10 km (P2P) | Can connect a large number of clients (up to 6000 per AP); longer range than traditional Wi-Fi; power efficient; limited bandwidth; low adoption; and cost may be an issue |
| WiMAX (802.16) | Wireless, several kilometers (last mile), up to 50 km (backhaul) | Can connect a large number of clients; large bandwidth available in licensed spectrum (fee-based); reduced bandwidth in license-free spectrum (interferences from other systems likely); adoption varies on location |
| Cellular (for example, LTE) | Wireless, several kilometers | Can connect a large number of clients; large bandwidth available; licensed spectrum (interference-free; license-based) |

*Architectural Considerations for WiMAX and Cellular Technologies*

**Layer 3: Applications and Analytics Layer**

Once connected to a network, your smart objects exchange information with other systems. As soon as your IoT network spans more than a few sensors, the power of the Internet of Things appears in the applications that make use of the information exchanged with the smart objects.

### Analytics Versus Control Applications

Multiple applications can help increase the efficiency of an IoT network. Each application collects data and provides a range of functions based on analyzing the collected data. It can be difficult to compare the features offered. From an architectural standpoint, one basic classification can be as follows:

- **Analytics application:** This type of application collects data from multiple smart objects, processes the collected data, and displays information resulting from the data that was processed. The display can be about any aspect of the IoT network, from historical reports, statistics, or trends to individual system states. The important aspect is that the application processes the data to convey a view of the network that cannot be obtained from solely looking at the information displayed by a single smart object.

- **Control application:** This type of application controls the behavior of the smart object or the behavior of an object related to the smart object. For example, a pressure sensor may be connected to a pump. A control application increases the pump speed when the connected sensor detects a drop in pressure. Control applications are very useful for controlling complex aspects of an IoT network with a logic that cannot be programmed inside a single IoT object, either because the configured changes are too complex to fit into the local system or because the configured changes rely on parameters that include elements outside the IoT object.

## Data Versus Network Analytics

*Analytics* is a general term that describes processing information to make sense of collected data. In the world of IoT, a possible classification of the analytics function is as follows:

- **Data analytics:** This type of analytics processes the data collected by smart objects and combines it to provide an intelligent view related to the IoT system. At a very basic level, a dashboard can display an alarm when a weight sensor detects that a shelf is empty in a store. In a more complex case, temperature, pressure, wind, humidity, and light levels collected from thousands of sensors may be combined and then processed to determine the likelihood of a storm and its possible path .

- **Network analytics:** Most IoT systems are built around smart objects connected to the network. A loss or degradation in connectivity is likely to affect the efficiency of the system. Such a loss can have dramatic effects. For example, open mines use wireless networks to automatically pilot dump trucks. A lasting loss of connectivity may result in an accident or degradation of operations efficiency (automated dump trucks typically stop upon connectivity loss). On a more minor scale, loss of connectivity means that data stops being fed to your data analytics platform, and the system stops making intelligent analyses of the IoT system.

### Data Analytics Versus Business Benefits

Data analytics is undoubtedly a field where the value of IoT is booming. Almost any object can be connected, and multiple types of sensors can be installed on a given object. Collecting and interpreting the data generated by these devices is where the value of IoT is realized.

## Smart Services

- The ability to use IoT to improve operations is often termed “smart services.” This term is generic, and in many cases the term is used but its meaning is often stretched to include one form of service or another where an additional level of intelligence is provided.

- Smart services can also be used to measure the efficiency of machines by detecting machine output, speed, or other forms of usage evaluation.

- Smart services can be integrated into an IoT system. For example, sensors can be integrated in a light bulb. A sensor can turn a light on or off based on the presence of a human in the room.

## IOT DATA MANAGEMENT AND COMPUTE STACK

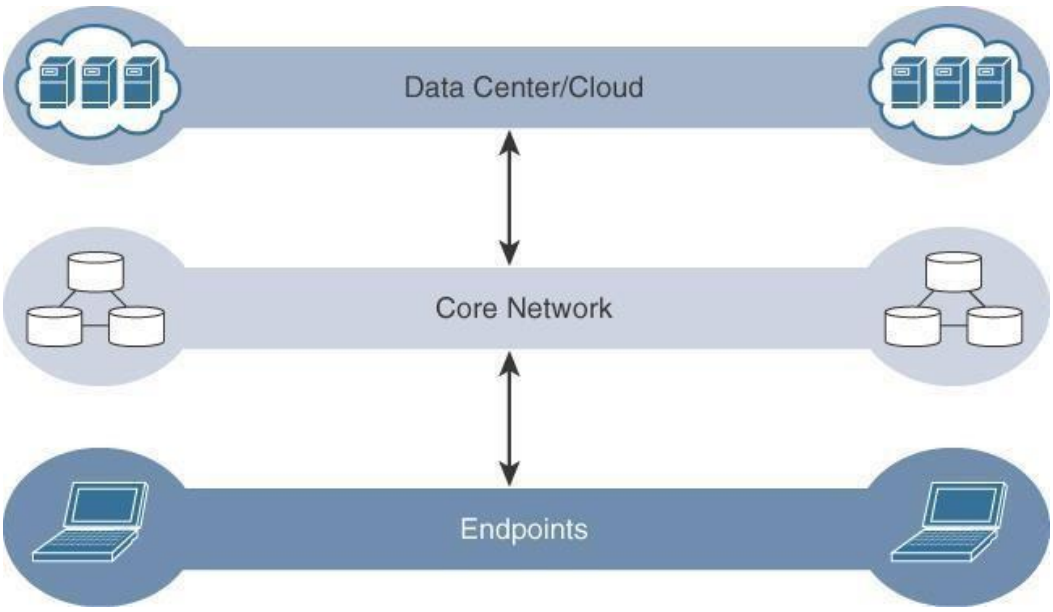
This model also has limitations. As data volume, the variety of objects connecting to the network, and the need for more efficiency increase, new requirements appear, and those requirements tend to bring the need for data analysis closer to the IoT system. These new requirements include the following:

- **Minimizing latency:** Milliseconds matter for many types of industrial systems, such as when you are trying to prevent manufacturing line shutdowns or restore electrical service. Analyzing data close to the device that collected the data can make a difference between averting disaster and a cascading system failure.

- **Conserving network bandwidth:** Offshore oil rigs generate 500 GB of data weekly. Commercial jets generate 10 TB for every 30 minutes of flight. It is not practical to transport vast amounts of data from thousands or hundreds of thousands of edge devices to the cloud. Nor is it necessary because many critical analyses do not require cloud-scale processing and storage.

- **Increasing local efficiency:** Collecting and securing data across a wide geographic area with different environmental conditions may not be useful. The environmental conditions in one area will trigger a local response independent from the conditions of another site hundreds of miles away. Analyzing both areas in the same cloud system may not be necessary for immediate efficiency.

.



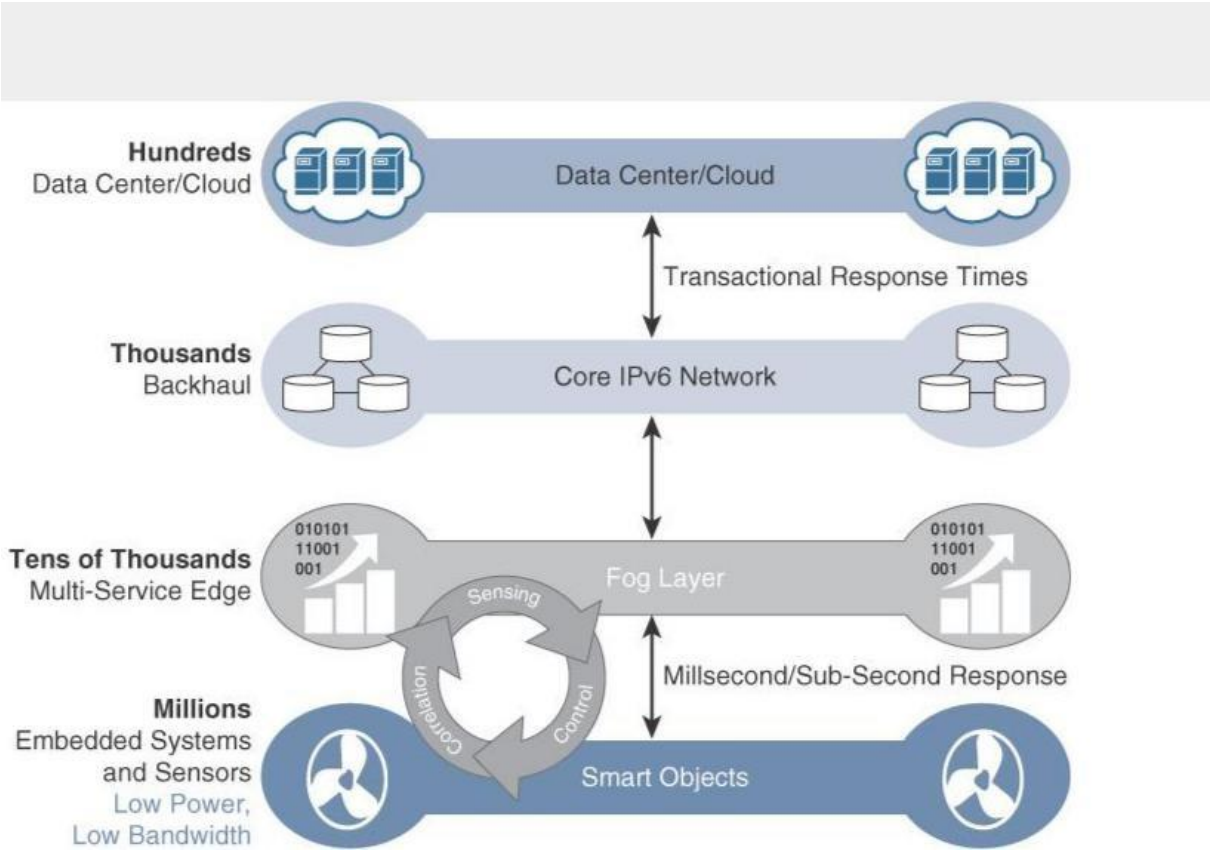

*The Traditional IT Cloud Computing Model*

IoT systems function differently. Several data-related problems need to be addressed:

- Bandwidth in last-mile IoT networks is very limited. When dealing with thousands/millions of devices, available bandwidth may be on order of tens of Kbps per device or even less.
- Latency can be very high. Instead of dealing with latency in the milliseconds range, large IoT networks often introduce latency of hundreds to thousands of milliseconds.
- Network backhaul from the gateway can be unreliable and often depends on 3G/LTE or even satellite links. Backhaul links can also be expensive if a per-byte data usage model is necessary.
- The volume of data transmitted over the backhaul can be high, and much of the data may not really be that interesting (such as simple polling messages).
- Big data is getting bigger. The concept of storing and analyzing all sensor data in the cloud is impractical. The sheer volume of data generated makes real-time analysis and response to the data almost impossible.

**Fog Computing**

The solution to the challenges mentioned in the previous section is to distribute data management throughout the IoT system, as close to the edge of the IP network as possible. The best-known embodiment of edge services in IoT is fog computing. Any device with computing, storage, and network connectivity can be a fog node. Examples include industrial controllers, switches, routers, embedded servers, and IoT gateways. Analyzing IoT data close to where it is collected minimizes latency, offloads gigabytes of network traffic from the core network, and keeps sensitive data inside the local network.




*The IoT Data Management and Compute Stack with Fog Computing*

Fog services are typically accomplished very close to the edge device, sitting as close to the IoT endpoints as possible. One significant advantage of this is that the fog node has contextual awareness of the sensors it is managing because of its geographic proximity to those sensors. For example, there might be a fog router on an oil derrick that is monitoring all the sensor activity at that location. Because the fog node is able to analyze information from all the sensors on that derrick, it can provide contextual analysis of the messages it is receiving and may decide to send back only the relevant information over the backhaul network to the cloud. In this way, it is performing distributed analytics such that the volume of data sent upstream is greatly reduced and is much more useful to application and analytics servers residing in the cloud.

Fog applications are as diverse as the Internet of Things itself. What they have in common is data reduction—monitoring or analyzing real-time data from network-connected things and then initiating an action, such as locking a door, changing equipment settings, applying the brakes on a train, zooming a video camera, opening a valve in response to a pressure reading, creating a bar chart, or sending an alert to a technician to make a preventive repair.

The defining characteristic of fog computing are as follows:

- **Contextual location awareness and low latency:** The fog node sits as close to the IoT endpoint as possible to deliver distributed computing.
- **Geographic distribution:** In sharp contrast to the more centralized cloud, the services and applications targeted by the fog nodes demand widely distributed deployments.
- **Deployment near IoT endpoints:** Fog nodes are typically deployed in the presence of a large number of IoT endpoints. For example, typical metering deployments often see 3000 to 4000 nodes per gateway router, which also functions as the fog computing node.
- **Wireless communication between the fog and the IoT endpoint:** Although it is possible to connect wired nodes, the advantages of fog are greatest when dealing with a large number of endpoints, and wireless access is the easiest way to achieve such scale.
- **Use for real-time interactions:** Important fog applications involve real-time interactions rather than batch processing. Preprocessing of data in the fog nodes allows upper-layer applications to perform batch processing on a subset of the data.

### Edge Computing

Fog computing solutions are being adopted by many industries, and efforts to develop distributed applications and analytics tools are being introduced at an accelerating pace. The natural place for a fog node is in the network device that sits closest to the IoT endpoints, and these nodes are typically spread throughout an IoT network

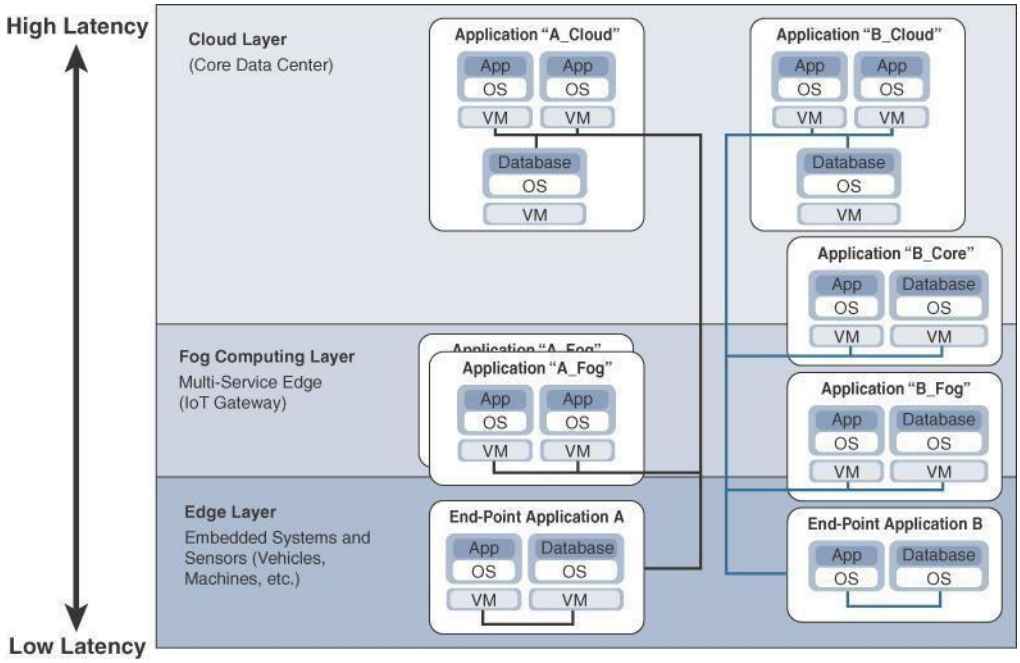
**Note**

Edge computing is also sometimes called “mist” computing. If clouds exist in the sky, and fog sits near the ground, then mist is what actually sits on the ground. Thus, the concept of mist is to extend fog to the furthest point possible, right into the IoT endpoint device itself.

### The Hierarchy of Edge, Fog, and Cloud

It is important to stress that edge or fog computing in no way replaces the cloud. Rather, they complement each other, and many use cases actually require strong cooperation between layers. In the same way that lower courts do not replace the supreme court of a

country, edge and fog computing layers simply act as a first line of defense for filtering, analyzing, and otherwise managing data endpoints. This saves the cloud from being queried by each and every node for each event.




*Distributed Compute and Data Management Across an IoT System*

From an architectural standpoint, fog nodes closest to the network edge receive the data from IoT devices. The fog IoT application then directs different types of data to the optimal place for analysis:

- The most time-sensitive data is analyzed on the edge or fog node closest to the things generating the data.
- Data that can wait seconds or minutes for action is passed along to an aggregation node for analysis and action.
- Data that is less time sensitive is sent to the cloud for historical analysis, big data analytics, and long-term storage. For example, each of thousands or hundreds of thousands of fog nodes might send periodic summaries of data to the cloud for historical analysis and storage.

In summary, when architecting an IoT network, you should consider the amount of data to be analyzed and the time sensitivity of this data. Understanding these factors will help you decide whether cloud computing is enough or whether edge or fog computing would improve your system efficiency. Fog computing accelerates awareness and response to events by eliminating a round trip to the cloud for analysis. It avoids the need for costly bandwidth additions by offloading gigabytes of network traffic from the core network. It also protects sensitive IoT data by analyzing it inside company walls.

# Module-2
# Smart Objects: The "Things" in IoT

Imagine the IoT-enabled connected vehicle and roadway highlighted in Chapter 1, ―What Is IoT?‖ That car has an impressive ecosystem of sensors that provides an immense amount of data that can be intelligently consumed by a variety of systems and services on the car itself as well as shared externally with other vehicles, the connected roadway infrastructure, or even a whole host of other cloud-based diagnostic and consumer services. From behind the steering wheel, almost everything in the car can be checked (sensed) and controlled. The car is filled with sensors of all types (for example, temperature, location [GPS], pressure, velocity) that are meant to provide a wealth of rich and relevant data to, among many other things, improve safety, simplify vehicle maintenance, and enhance the driver experience.

Such sensors are fundamental building blocks of IoT networks. In fact, they are the foundational elements found in smart objects—the ―things‖ in the Internet of Things. Smart objects are any physical objects that contain embedded technology to sense and/or interact with their environment in a meaningful way by being interconnected and enabling communication among themselves or an external agent.

This chapter provides a detailed analysis of smart objects and their architecture. It also provides an understanding of their design limitations and role within IoT networks. Specifically, the following sections are included:

- **Sensors, Actuators, and Smart Objects:** This section defines sensors, actuators, and smart objects and describes how they are the fundamental building blocks of IoT networks.
- **Sensor Networks:** This section covers the design, drivers for adoption, and deployment challenges of sensor networks.

## Sensors, Actuators, and Smart Objects

The following sections describe the capabilities, characteristics, and functionality of sensors and actuators. They also detail how the economic and technical conditions are finally right for IoT to flourish. Finally, you will see how to bring these foundational elements together to form smart objects, which are connected to form the sensor and actuator networks that make most IoT use cases possible.

### Sensors

A sensor does exactly as its name indicates: It senses. More specifically, a sensor measures some physical quantity and converts that measurement reading into a digital representation. That digital representation is typically passed to another device for transformation into useful data that can be consumed by intelligent devices or humans.

Naturally, a parallel can be drawn with humans and the use of their five senses to learn about their surroundings. Human senses do not operate independently in silos. Instead, they complement each other and compute together, empowering the human brain to make intelligent decisions. The brain is the ultimate decision maker, and it often uses several sources of sensory input to validate an event and compensate for ―incomplete‖ information.

Sensors are not limited to human-like sensory data. They can measure anything worth measuring. In fact, they are able to provide an extremely wide spectrum of rich and diverse measurement data with far greater precision than human senses; sensors provide superhuman sensory capabilities. This additional dimension of data makes the physical world an incredibly valuable source of information. Sensors can be readily embedded in any physical objects that are easily connected to the Internet by wired or wireless networks. Because these connected host physical objects with multidimensional sensing capabilities communicate with each other and external systems, they can interpret their environment and make intelligent decisions. Connecting sensing devices in this way has ushered in the world of IoT and a whole new paradigm of business intelligence.

There are myriad different sensors available to measure virtually everything in the physical world. There are a number of ways to group and cluster sensors into different categories, including the following:

- **Active or passive:** Sensors can be categorized based on whether they produce an energy output and typically require an external power supply (active) or whether they simply receive energy and typically require no external power supply (passive).
- **Invasive or non-invasive:** Sensors can be categorized based on whether a sensor is part of the environment it is measuring (invasive) or external to it (non-invasive).
- **Contact or no-contact:** Sensors can be categorized based on whether they require physical contact with what they are measuring (contact) or not (no-contact).
- **Absolute or relative:** Sensors can be categorized based on whether they measure on an absolute scale (absolute) or based on a difference.with a fixed or variable reference value (relative).
- **Area of application:** Sensors can be categorized based on the specific industry or vertical where they are being used.
- **How sensors measure:** Sensors can be categorized based on the physical mechanism used to measure sensory input (for example, thermoelectric, electrochemical, piezoresistive, optic, electric, fluid mechanic, photoelastic).
- **What sensors measure:** Sensors can be categorized based on their applications or what physical variables they measure.

| Sensor Types | Description | Examples |
|---|---|---|
| Position | A position sensor measures the position of an object; the position measurement can be either in absolute terms (absolute position sensor) or in relative terms (displacement sensor). Position sensors can be linear, angular, or multi-axis. | Potentiometer, inclinometer, proximity sensor |
| Occupancy and motion | Occupancy sensors detect the presence of people and animals in a surveillance area, while motion sensors detect movement of people and objects. The difference between the two is that occupancy sensors generate a signal even when a person is stationary, whereas motion sensors do not. | Electric eye, radar |
| Velocity and acceleration | Velocity (speed of motion) sensors may be linear or angular, indicating how fast an object moves along a straight line or how fast it rotates. Acceleration sensors measure changes in velocity. | Accelerometer, gyroscope |
| Force | Force sensors detect whether a physical force is applied and whether the magnitude of force is beyond a threshold. | Force gauge, viscometer, tactile sensor (touch sensor) |
| Pressure | Pressure sensors are related to force sensors, measuring force applied by liquids or gases. Pressure is measured in terms of force per unit area. | Barometer, Bourdon gauge, piezometer |
| Flow | Flow sensors detect the rate of fluid flow. They measure the volume (mass flow) or rate (flow velocity) of fluid that has passed through a system in a given period of time. | Anemometer, mass flow sensor, water meter |
| Acoustic | Acoustic sensors measure sound levels and convert that information into digital or analog data signals. | Microphone, geophone, hydrophone |
| Humidity | Humidity sensors detect humidity (amount of water vapor) in the air or a mass. Humidity levels can be measured in various ways: absolute humidity, relative humidity, mass ratio, and so on. | Hygrometer, humistor, soil moisture sensor |
| Light | Light sensors detect the presence of light (visible or invisible). | Infrared sensor, photodetector, flame detector |
| Radiation | Radiation sensors detect radiation in the environment. Radiation can be sensed by scintillating or ionization detection. | Geiger-Müller counter, scintillator, neutron detector |
| Temperature | Temperature sensors measure the amount of heat or cold that is present in a system. They can be broadly of two types: contact and non-contact. Contact temperature sensors need to be in physical contact with the object being sensed. Non-contact sensors do not need physical contact, as they measure temperature through convection and radiation. | Thermometer, calorimeter, temperature gauge |
| Chemical | Chemical sensors measure the concentration of chemicals in a system. When subjected to a mix of chemicals, chemical sensors are typically selective for a target type of chemical (for example, a $CO_2$ sensor senses only carbon dioxide). | Breathalyzer, olfactometer, smoke detector |
| Biosensors | Biosensors detect various biological elements, such as organisms, tissues, cells, enzymes, antibodies, and nucleic acid. | Blood glucose biosensor, pulse oximetry, electrocardiograph |

*Source*: J. Holdowsky et al., *Inside the Internet of Things: A Primer on the Technologies Building the IoT*, August 21, 2015, http://dupress.deloitte.com/dup-us-en/focus/internet-of-things/iot-primer-iot-technologies-applications.html.

**Table 3-1** *Sensor Types*

Sensors come in all shapes and sizes and, as shown in Table 3-1, can measure all types of physical conditions. A fascinating use case to highlight the power of sensors and IoT is in the area of precision agriculture (sometimes referred to as smart farming), which uses a variety of technical advances to improve the efficiency, sustainability, and profitability of traditional farming practices. This includes the use of GPS and satellite aerial imagery for determining field viability; robots for high-precision planting, harvesting, irrigation, and so on; and real-time analytics and artificial intelligence to predict optimal crop yield,weather impacts, and soil quality.

Among the most significant impacts of precision agriculture are those dealing with sensor measurement of a variety of soil characteristics. These include real-time measurement of soil quality, pH levels, salinity, toxicity levels, moisture levels for irrigation planning, nutrient levels for fertilization planning, and so on. All this detailed sensor data can be analyzed to provide highly valuable and actionable insight to boost productivity and crop yield.

Figure 3-1 shows biodegradable, passive microsensors to measure soil and crop and conditions. These sensors, developed at North Dakota State University (NDSU), can be planted directly in the soil and left in the ground to biodegrade without any harm to soil quality.

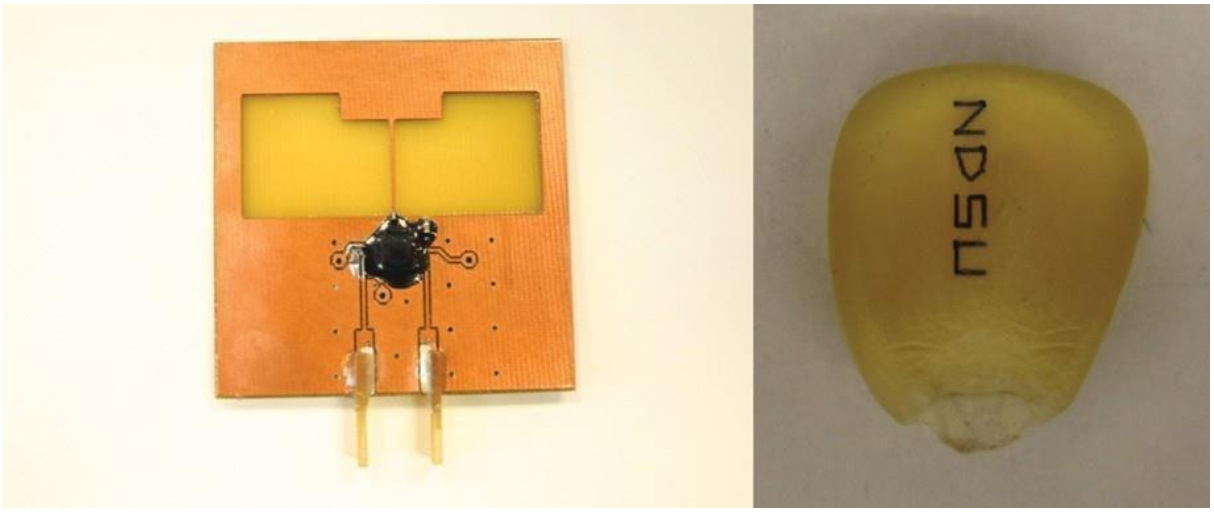



**Figure 3-1** ***Biodegradable Sensors Developed by NDSU for Smart Farming (Reprinted with permission from NDSU.)***

IoT and, by extension, networked sensors have been repeatedly named among a small number of emerging revolutionary technologies that will change the global economy and shape the future. The staggering proliferation of sensors is the principal driver of this phenomenon. The astounding volume of sensors is in large part due to their smaller size, their form factor, and their decreasing cost.

These factors make possible the economic and technical feasibility of having an increased density of sensors in objects of all types. Perhaps the most significant accelerator for sensor deployments is mobile phones. More than a billion smart phones

are sold each year, and each one has well over a dozen sensors inside it and that number continues to grow each year. Imagine the exponential effect of extending sensors to practically every technology, industry, and vertical. For example, there are smart homes with potentially hundreds of sensors, intelligent vehicles with 100+ sensors each, connected cities with thousands upon thousands of connected sensors, and the list goes on and on.


**Figure 3-2 Sensors in a Smart Phone**

It's fascinating to think that that a trillion-sensor economy is around the corner. Figure 3-3 shows the explosive year-over-year increase over the past several years and some bold predictions for sensor numbers in the upcoming years. There is a strong belief in the sensor industry that this number will eclipse a trillion in the next few years. In fact, many large players in the sensor industry have come together to form industry consortia, such as the TSensors Summits (www.tsensorssummit.org), to create a strategy and roadmap for a trillion-sensor economy. The trillion-sensor economy will be of such an unprecedented and unimaginable scale that it will change the world forever. This is the power of IoT.

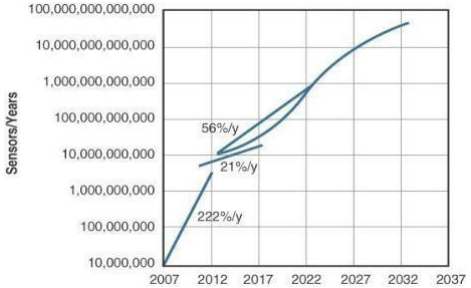



**Figure 3-3** *Growth and Predictions in the Number of Sensors*

**Actuators**

Actuators are natural complements to sensors. Figure 3-4 demonstrates the symmetry and complementary nature of these two types of devices. As discussed in the previous section, sensors are designed to sense and measure practically any measurable variable in the physical world. They convert their measurements (typically analog) into electric signals or digital representations that can be consumed by an intelligent agent (a device or a human).

Actuators, on the others hand, receive some type of control signal (commonly an electric signal or digital command) that triggers a physical effect, usually some type of motion, force, and so on.

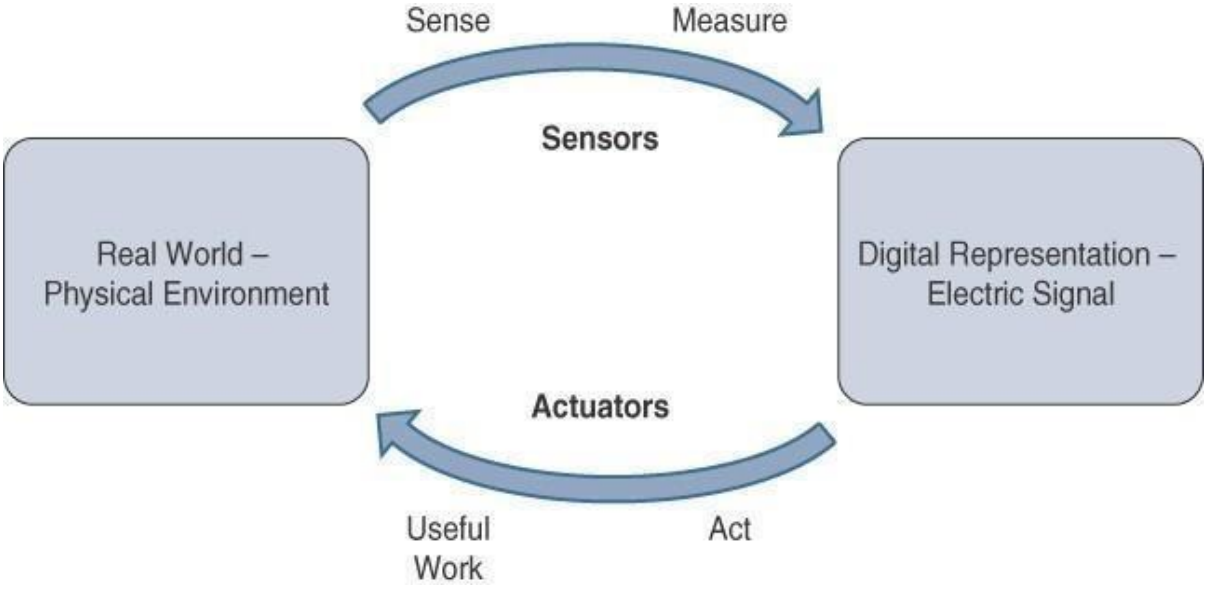



**Figure 3-4 How Sensors and Actuators Interact with the Physical World**

The previous section draws a parallel between sensors and the human senses. This parallel can be extended to include actuators, as shown in Figure 3-5.

Humans use their five senses to sense and measure their environment. The sensory organs convert this sensory information into electrical impulses that the nervous system sends to the brain for processing. Likewise, IoT sensors are devices that sense and measure the physical world and (typically) signal their measurements as electric signals sent to some type of microprocessor or microcontroller for additional processing.

The human brain signals motor function and movement, and the nervous system carries that information to the appropriate part of the muscular system. Correspondingly, a processor can send an electric signal to an actuator that translates the signal into some type of movement (linear, rotational, and so on) or useful work that changes or has a measurable impact on the physical world. This interaction between sensors, actuators, and processors and the similar functionality in biological systems is the basis for various technical fields, including robotics and biometrics.

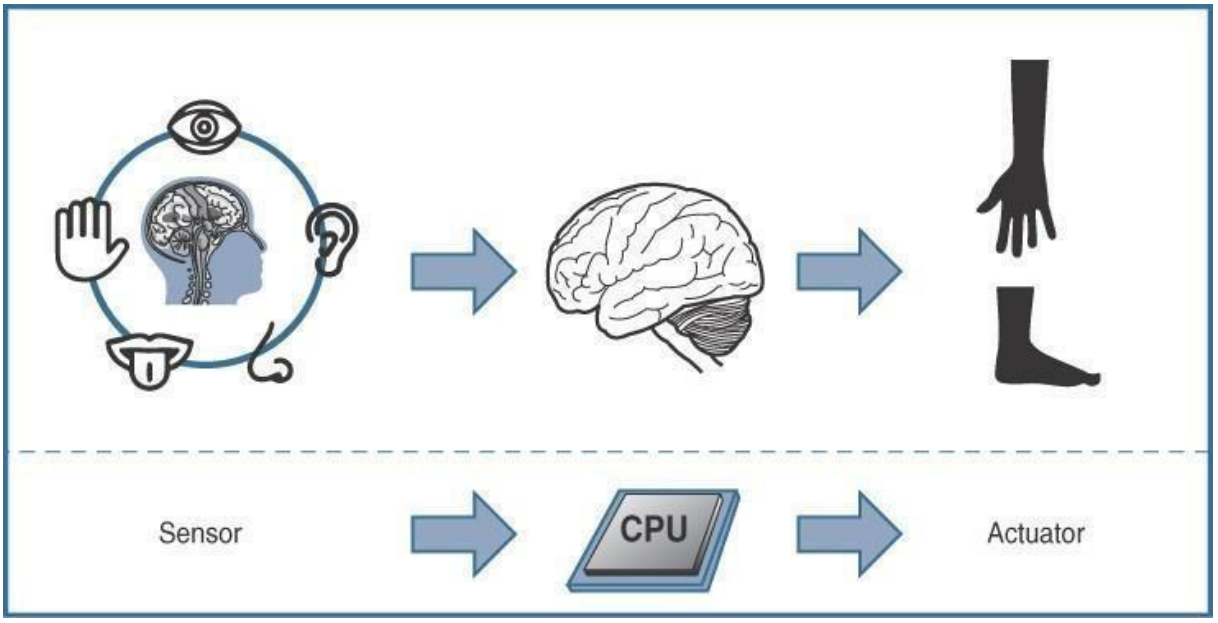



**Figure 3-5** *Comparison of Sensor and Actuator Functionality with Humans*

Much like sensors, actuators also vary greatly in function, size, design, and so on. Some common ways that they can be classified include the following:

- **Type of motion:** Actuators can be classified based on the type of motion they produce (for example, linear, rotary, one/two/three-axes).
- **Power:** Actuators can be classified based on their power output (for example, high power, low power, micro power)
- **Binary or continuous:** Actuators can be classified based on the number of stable-state outputs.
- **Area of application:** Actuators can be classified based on the specific industry or vertical where they are used.
- **Type of energy:** Actuators can be classified based on their energy type.

Categorizing actuators is quite complex, given their variety, so this is by no means an exhaustive list of classification schemes. The most commonly used classification is based on energy type. Table 3-2 shows actuators classified by energy type and some examples for each type. Again, this is not a complete list, but it does provide a reasonably comprehensive overview that highlights the diversity of function and design of actuators.

| Type | Examples |
|---|---|
| Mechanical actuators | Lever, screw jack, hand crank |
| Electrical actuators | Thyristor, biopolar transistor, diode |
| Electromechanical actuators | AC motor, DC motor, step motor |
| Electromagnetic actuators | Electromagnet, linear solenoid |
| Hydraulic and pneumatic actuators | Hydraulic cylinder, pneumatic cylinder, piston, pressure control valves, air motors |
| Smart material actuators (includes thermal and magnetic actuators) | Shape memory alloy (SMA), ion exchange fluid, magnetorestrictive material, bimetallic strip, piezoelectric bimorph |
| Micro- and nanoactuators | Electrostatic motor, microvalve, comb drive |

**Table 3-2** *Actuator Classification by Energy Type*

Whereas sensors provide the information, actuators provide the action. The most interesting use cases for IoT are those where sensors and actuators work together in an intelligent, strategic, and complementary fashion. This powerful combination can be used to solve everyday problems by simply elevating the data that sensors provide to actionable insight that can be acted on by work-producing actuators.

We can build on the precision agriculture example from the previous section to demonstrate how actuators can complement and enhance a sensor-only solution. For example, the smart sensors used to evaluate soil quality (by measuring a variety of soil, temperature, and plant characteristics) can be connected with electrically or pneumatically controlled valve actuators that control water, pesticides, fertilizers, herbicides, and so on. Intelligently triggering a high-precision actuator based on well-defined sensor readings of temperature, pH, soil/air humidity, nutrient levels, and so on to deliver a highly optimized and custom environment-specific solution is truly smart farming.

**Micro-Electro-Mechanical Systems (MEMS)**

One of the most interesting advances in sensor and actuator technologies is in how they are packaged and deployed. Micro-electro-mechanical systems (MEMS), sometimes simply referred to as micro-machines, can integrate and combine electric and mechanical elements, such as sensors and actuators, on a very small (millimeter or less) scale. One of the keys to this technology is a microfabrication technique that is similar to what is used for microelectronic integrated circuits. This approach allows mass production at very low costs.

The combination of tiny size, low cost, and the ability to mass produce makes MEMS an attractive option for a huge number of IoT applications.

MEMS devices have already been widely used in a variety of different applications and can be found in very familiar everyday devices. For example, inkjet printers use micropump MEMS. Smart phones also use MEMS technologies for things like accelerometers and gyroscopes. In fact, automobiles were among the first to commercially introduce MEMS into the mass market, with airbag accelerometers.

Figure 3-6 shows a torsional ratcheting actuator (TRA) that was developed by Sandia National Laboratory as a low-voltage alternative to a micro-engine.

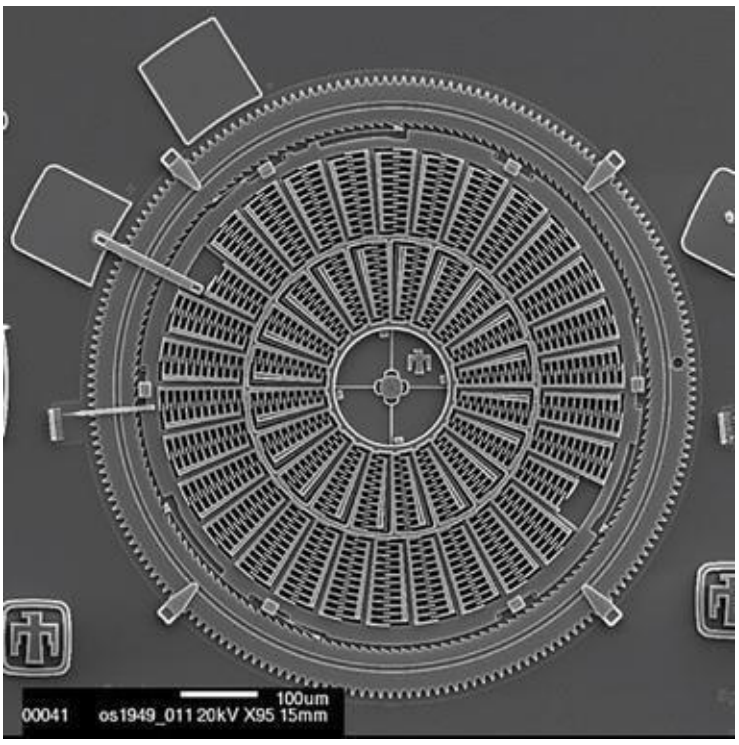



Figure 3-6 *Torsional Ratcheting Actuator (TRA) MEMS (Courtesy Sandia National Laboratories, SUMMiT™ Technologies, www.sandia.gov/mstc.)*

As Figure 3-6 shows, this MEMS is only a few hundred micrometers across; a scanning electron microscope is needed to show the level of detail visible in the figure. Micro-scale sensors and actuators are immensely embeddable in everyday objects, which is a defining characteristic of IoT. For this reason, it is expected that IoT will trigger significant advances in MEMS technology, and manufacturing and will make them pervasive across all industries and verticals as they become broadly commercialized.

## Smart Objects

Smart objects are, quite simply, the building blocks of IoT. They are what transform everyday objects into a network of intelligent objects that are able to learn from and interact with their environment in a meaningful way. It can't be stressed enough that the real power of smart objects in IoT comes from being networked together rather than being isolated as standalone objects.

This ability to communicate over a network has a multiplicative effect and allows for very sophisticated correlation and interaction between disparate smart objects. For instance, recall the smart farming sensors described previously. If a sensor is a standalone device that simply measures the humidity of the soil, it is interesting and useful, but it isn't revolutionary. If that same sensor is connected as part of an intelligent network that is able to coordinate intelligently with actuators to trigger irrigation systems as needed based on those sensor readings, we have something far more powerful.

Extending that even further, imagine that the coordinated sensor/actuator set is

intelligently interconnected with other sensor/actuator sets to further coordinate fertilization, pest control, and so on—and even communicate with an intelligent backend to calculate crop yield potential. This now starts to look like a complete system that begins to unlock the power of IoT and provides the intelligent automation we have come to expect from such a revolutionary technology.

**Smart Objects: A Definition**

Historically, the definition of a smart object has been a bit nebulous because of the different interpretations of the term by varying sources. To add to the overall confusion, the term *smart object*, despite some semantic differences, is often used interchangeably with terms such as *smart sensor*, *smart device*, *IoT device*, *intelligent device*, *thing*, *smart thing*, *intelligent node*, *intelligent thing*, *ubiquitous thing*, and *intelligent product*. In order to clarify some of this confusion, we provide here the definition of *smart object* as we use it in this book. A *smart object*, as described throughout this book, is a device that has, at a minimum, the following four defining characteristics (see Figure 3- 7):

- **Processing unit:** A smart object has some type of processing unit for acquiring data, processing and analyzing sensing information received by the sensor(s), coordinating control signals to any actuators, and controlling a variety of functions on the smart object, including the communication and power systems. The specific type of processing unit that is used can vary greatly, depending on the specific processing needs of different applications. The most common is a microcontroller because of its small form factor, flexibility, programming simplicity, ubiquity, low power consumption, and low cost.
- **Sensor(s) and/or actuator(s):** A smart object is capable of interacting with the physical world through sensors and actuators. As described in the previous sections, a sensor learns and measures its environment, whereas an actuator is able to produce some change in the physical world. A smart object does not need to contain both sensors and actuators. In fact, a smart object can contain one or multiple sensors and/or actuators, depending upon the application.
- **Communication device:** The communication unit is responsible for connecting a smart object with other smart objects and the outside world (via the network). Communication devices for smart objects can be either wired or wireless. Overwhelmingly, in IoT networks smart objects are wirelessly interconnected for a number of reasons, including cost, limited infrastructure availability, and ease of deployment. There are myriad different communication protocols for smart objects. In fact, much of this book is dedicated to how smart objects communicate within an IoT network, especially Chapter 4, ―Connecting Smart Objects,‖ Chapter 5, ―IP as the IoT Network Layer,‖ and Chapter 6, ―Application Protocols for IoT.‖ Thus, this chapter provides only a high-level overview and refers to those other chapters for a more detailed treatment of the subject matter.
- **Power source:** Smart objects have components that need to be powered. Interestingly, the most significant power consumption usually comes from the communication unit of a smart object. As with the other three smart object

building blocks, the power requirements also vary greatly from application to application. Typically, smart objects are limited in power, are deployed for a very long time, and are not easily accessible. This combination, especially when the smart object relies on battery power, implies that power efficiency, judicious power management, sleep modes, ultra-low power consumption hardware, and so on are critical design elements. For long-term deployments where smart objects are, for all practical purposes, inaccessible, power is commonly obtained from scavenger sources (solar, piezoelectric, and so on) or is obtained in a hybridized manner, also tapping into infrastructure power.

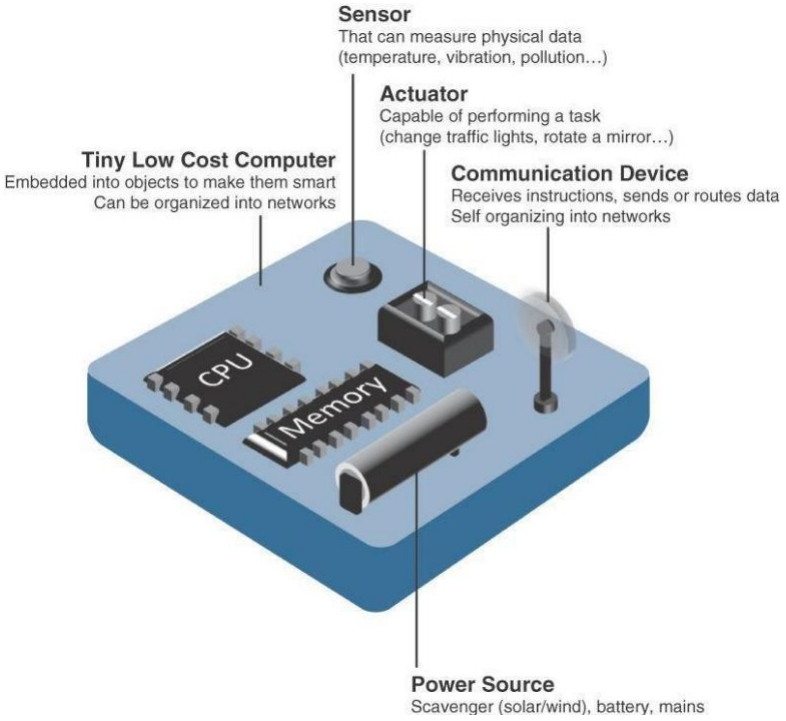



**Figure 3-7** *Characteristics of a Smart Object*

## Trends in Smart Objects

As this definition reveals, it is perhaps variability that is the key characteristic of smart objects. They vary wildly in function, technical requirements, form factor, deployment conditions, and so on. Nevertheless, there are certain important macro trends that we can infer from recent and planned future smart object deployments. Of course, these do not apply to all smart objects because there will always be application-dependent variability, but these are broad generalizations and trends impacting IoT:

- **Size is decreasing:** As discussed earlier, in reference to MEMS, there is a clear trend of ever-decreasing size. Some smart objects are so small they are not even visible to the naked eye. This reduced size makes smart objects easier to embed in everyday objects.

- **Power consumption is decreasing:** The different hardware components of a smart object continually consume less power. This is especially true for sensors, many of which are completely passive.
  Some battery-powered sensors last 10 or more years without battery replacement.

- **Processing power is increasing:** Processors are continually getting more powerful and smaller. This is a key advancement for smart objects, as they become increasingly complex and connected.

- **Communication capabilities are improving:** It's no big surprise that wireless speeds are continually increasing, but they are also increasing in range. IoT is driving the development of more and more specialized communication protocols covering a greater diversity of use cases and environments.

- **Communication is being increasingly standardized:** There is a strong push in the industry to develop open standards for IoT communication protocols. In addition, there are more and more open source efforts to advance IoT.

These trends in smart objects begin to paint a picture of increasingly sophisticated devices that are able to perform increasingly complex tasks with greater efficiency. A key enabler of this paradigm is improved communication between interconnected smart objects within a system and between that system and external entities (for example, edge compute, cloud). The power of IoT is truly unlocked when smart objects are networked together in sensor/actuator networks.

**Sensor Networks**

A sensor/actuator network (SANET), as the name suggests, is a network of sensors that sense and measure their environment and/or actuators that act on their environment. The sensors and/or actuators in a SANET are capable of communicating and cooperating in a productive manner. Effective and well-coordinated communication and cooperation is a prominent challenge, primarily because the sensors and actuators in SANETs are diverse, heterogeneous, and resource-constrained.

SANETs offer highly coordinated sensing and actuation capabilities. Smart homes are a type of SANET that display this coordination between distributed sensors and actuators. For example, smart homes can have temperature sensors that are strategically networked with heating, ventilation, and air- conditioning (HVAC) actuators. When a sensor detects a specified temperature, this can trigger an actuator to take action and heat or cool the home as needed. While such networks can theoretically be connected in a wired or wireless fashion, the fact that SANETs are typically found in the ―real world‖ means that they need an extreme level of deployment flexibility. For example, smart home temperature sensors need to be expertly located in strategic locations throughout the home, including at HVAC entry and exit points.

The following are some advantages and disadvantages that a wireless-based solution offers:

- Advantages:
  - Greater deployment flexibility (especially in extreme environments or hard-to-reach places)
  - Simpler scaling to a large number of nodes Lower
  - implementation costs
  - Easier long-term maintenance
  - Effortless introduction of new sensor/actuator nodes
  - Better equipped to handle dynamic/rapid topology changes
- Disadvantages:
  - Potentially less secure (for example, hijacked access points)
  - Typically lower transmission speeds

- Greater level of impact/influence by environment

Not only does wireless allow much greater flexibility, but it is also an increasingly inexpensive and reliable technology across a very wide spectrum of conditions—even extremely harsh ones. These characteristics are the key reason that wireless SANETs are the ubiquitous networking technology for IoT.

**Wireless Sensor Networks (WSNs)**

Wireless sensor networks are made up of wirelessly connected smart objects, which are sometimes referred to as *motes*. The fact that there is no infrastructure to consider with WSNs is surely a powerful advantage for flexible deployments, but there are a variety of design constraints to consider with these wirelessly connected smart objects. Figure 3-8 illustrates some of these assumptions and constraints usually involved in WSNs.

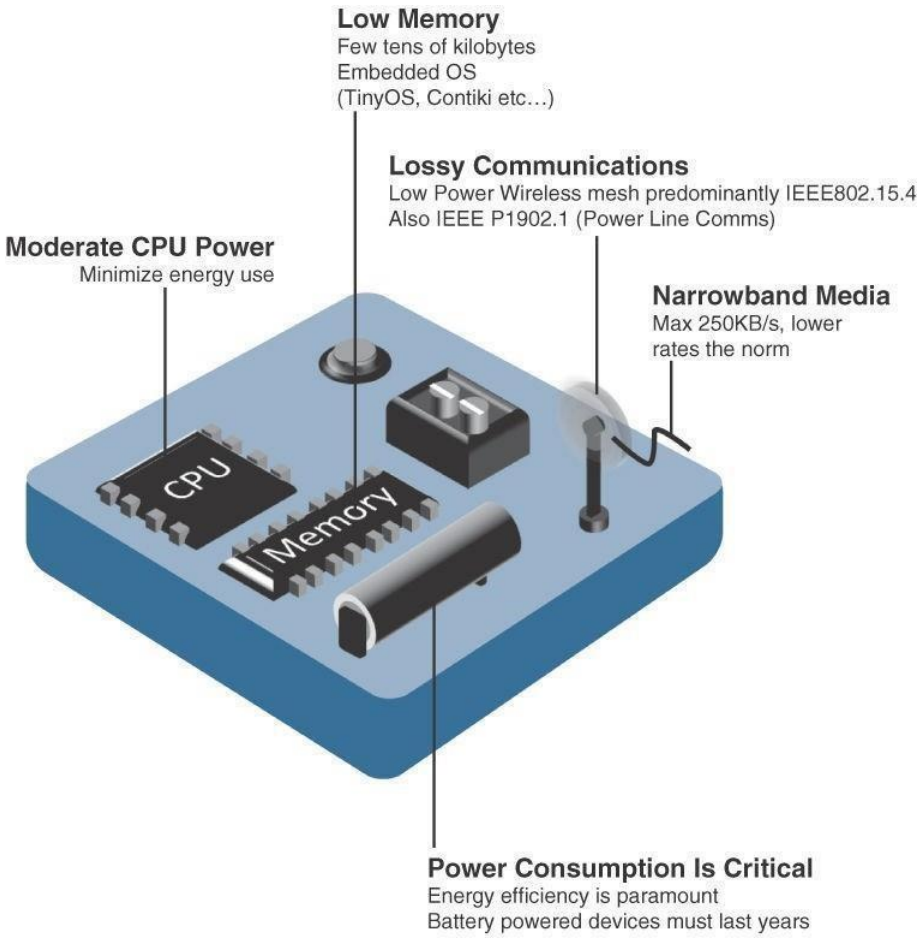



**Figure 3-8** *Design Constraints for Wireless Smart Objects*

The following are some of the most significant limitations of the smart objects in WSNs:

- Limited processing power

- Limited memory Lossy
- communication
- Limited transmission speeds
- Limited power

These limitations greatly influence how WSNs are designed, deployed, and utilized. The fact that individual sensor nodes are typically so limited is a reason that they are often deployed in very large numbers. As the cost of sensor nodes continues to decline, the ability to deploy highly redundant sensors becomes increasingly feasible. Because many sensors are very inexpensive and correspondingly inaccurate, the ability to deploy smart objects redundantly allows for increased accuracy.

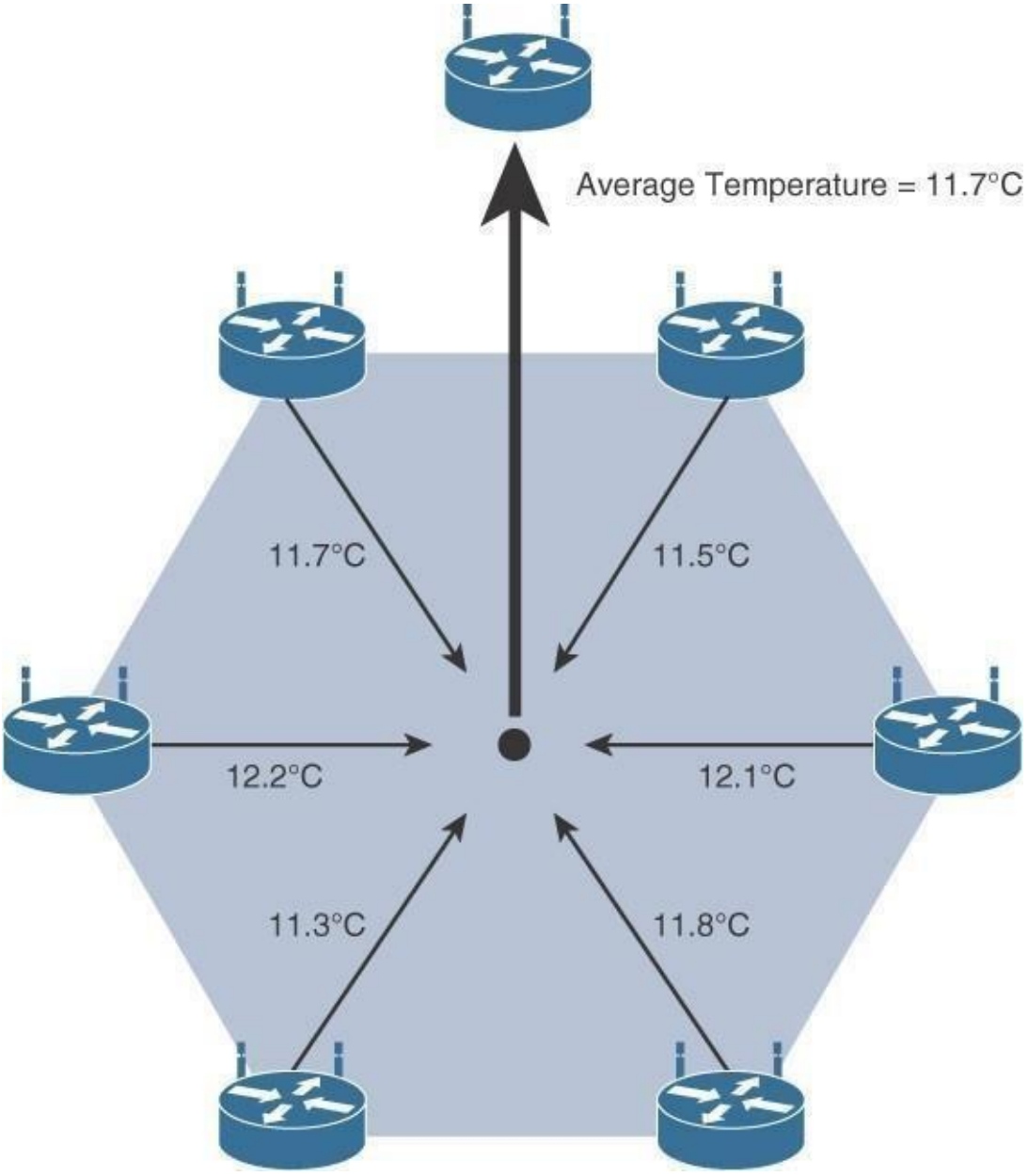



**Figure 3-9** *Data Aggregation in Wireless Sensor Networks*

These data aggregation techniques are helpful in reducing the amount of overall traffic (and energy) in WSNs with very large numbers of deployed smart objects. This data aggregation at the network edges is where fog and mist computing, discussed in Chapter 2, ―IoT Network Architecture and Design,‖ are critical IoT architectural elements needed to deliver the scale and performance required by so many IoT use cases. While there are certain instances in which sensors continuously stream their measurement data, this is typically not the case. Wirelessly connected smart objects generally have one of the following two communication patterns:

- **Event-driven:** Transmission of sensory information is triggered only when a smart object detects a particular event or predetermined threshold.
- **Periodic:** Transmission of sensory information occurs only at periodic intervals.

The decision of which of these communication schemes is used depends greatly on the specific application. For example, in some medical use cases, sensors periodically send postoperative vitals, such as temperature or blood pressure readings. In other medical use cases, the same blood pressure or temperature readings are triggered to be sent only when certain critically low or high readings are measured.

### Communication Protocols for Wireless Sensor Networks

There are literally thousands of different types of sensors and actuators. To further complicate matters, WSNs are becoming increasingly heterogeneous, with more sophisticated interactions. This heterogeneity is manifested in a variety of ways. For instance, WSNs are seeing transitions from homogenous wireless networks made up of mostly a single type of sensor to networks made up of multiple types of sensors that can even be a hybridized mix of many cheap sensors with a few expensive ones used for very specific high- precision functions. WSNs are also evolving from single-purpose networks to more flexible multipurpose networks that can use specific sensor types for multiple different applications at any given time. Imagine a WSN that has multiple types of sensors, and one of those types is a temperature sensor that can be flexibly used concurrently for environmental applications, weather applications, and smart farming applications.

Coordinated communication with sophisticated interactions by constrained devices within such a heterogeneous environment is quite a challenge. The protocols governing the communication for WSNs must deal with the inherent defining characteristics of WSNs and the constrained devices within them. For instance, any communication protocol must be able to scale to a large number of nodes. Likewise, when selecting a communication protocol, you must carefully take into account the requirements of the specific application and consider any trade-offs the communication protocol offers between power consumption, maximum transmission speed, range, tolerance for packet loss, topology optimization, security, and so on. The fact that WSNs are often deployed outdoors in harsh and unpredictable environments adds yet another variable to consider because obviously not all communication protocols are designed to be equally rugged. In addition to the aforementioned

technical capabilities, they must also enable, as needed, the overlay of autonomous techniques (for example, self-organization, self- healing, self-configuration) mentioned in the previous section.

Wireless sensor networks interact with their environment. Sensors often produce large amounts of sensing and measurement data that needs to be processed. This data can be processed locally by the nodes of a WSN or across zero or more hierarchical levels in IoT networks. (These hierarchical levels are discussed in detail in Chapter 2.) Communication protocols need to facilitate routing and message handling for this data flow between sensor nodes as well as from sensor nodes to optional gateways, edge compute, or centralized cloud compute. IoT communication protocols for WSNs thus straddle the entire protocol stack. Ultimately, they are used to provide a platform for a variety of IoT smart services.

As with any other networking application, in order to interoperate in multivendor environments, these communication protocols must be standardized. This is a critical dependency for IoT and one of the most significant success factors. IoT is one of those rare technologies that impacts all verticals and industries, which means standardization of communication protocols is a complicated task, requiring protocol definition across multiple layers of the stack, as well as a great deal of coordination across multiple standards development organizations.

## Connecting Smart Objects

IoT devices and sensors must be connected to the network for their data to be utilized. In addition to the wide range of sensors, actuators, and smart objects that make up IoT, there are also a number of different protocols used to connect them. This chapter takes a look at the characteristics and communications criteria that are important for the **technologies** that smart objects employ for their connectivity, along with a deeper dive into some of the major technologies being deployed today.

Two main sections divide this chapter. The first main section, ―Communications Criteria,‖ describes the characteristics and attributes you should consider when selecting and dealing with connecting smart objects. The various technologies used for connecting sensors can differ greatly depending on the criteria used to analyze them. The following subsections look closely at these criteria:

- **Range:** This section examines the importance of signal propagation and distance.
- **Frequency Bands:** This section describes licensed and unlicensed spectrum, including sub-GHz frequencies.
- **Power Consumption:** This section discusses the considerations required for devices connected to a stable power source compared to those that are battery powered.
- **Topology:** This section highlights the various layouts that may be supported for connecting multiple smart objects.
- **Constrained Devices:** This section details the limitations of certain smart

objects from a connectivity perspective.

- **Constrained-Node Networks:** This section highlights the challenges that are often encountered with networks connecting smart objects.

The following subsections cover technologies for connecting smart objects:

- **IEEE 802.15.4:** This section highlights IEEE 802.15.4, an older but foundational wireless protocol for connecting smart objects.
- **IEEE 802.15.4g and IEEE 802.15.4e:** This section discusses improvements to 802.15.4 that are targeted to utilities and smart cities deployments.
- **IEEE 1901.2a:** This section discusses IEEE 1901.2a, which is a technology for connecting smart objects over power lines.
- **IEEE 802.11ah:** This section discusses IEEE 802.11ah, a technology built on the well-known 802.11 Wi-Fi standards that is specifically for smart objects.
- **LoRaWAN:** This section discusses LoRaWAN, a scalable technology designed for longer distances with low power requirements in the unlicensed spectrum.
- **NB-IoT and Other LTE Variations:** This section discusses NB-IoT and other LTE variations, which are often the choice of mobile service providers looking to connect smart objects over longer distances in the licensed spectrum.

This chapter covers quite a few fundamental IoT technologies and is critical for truly understanding how smart objects handle data transport to and from the network. We encourage you to pay special attention to the protocols and technologies discussed here because they are applied and referenced in many of the other chapters of this book.

### Communications Criteria

In the world of connecting ―things,‖ a large number of wired and wireless access technologies are available or under development. Before reviewing some of these access technologies, it is important to talk about the criteria to use in evaluating them for various use cases and system solutions.

Wireless communication is prevalent in the world of smart object connectivity, mainly because it eases deployment and allows smart objects to be mobile, changing location without losing connectivity. The following sections take this into account as they discuss various criteria. In addition, wired connectivity considerations are mentioned when applicable.

### Range

How far does the signal need to be propagated? That is, what will be the area of coverage for a selected wireless technology? Should indoor versus outdoor deployments be differentiated? Very often, these are the first questions asked when discussing wired and wireless access technologies. The simplest approach to

answering these types of questions is to categorize these technologies as shown in Figure 4-1, breaking them down into the following ranges:

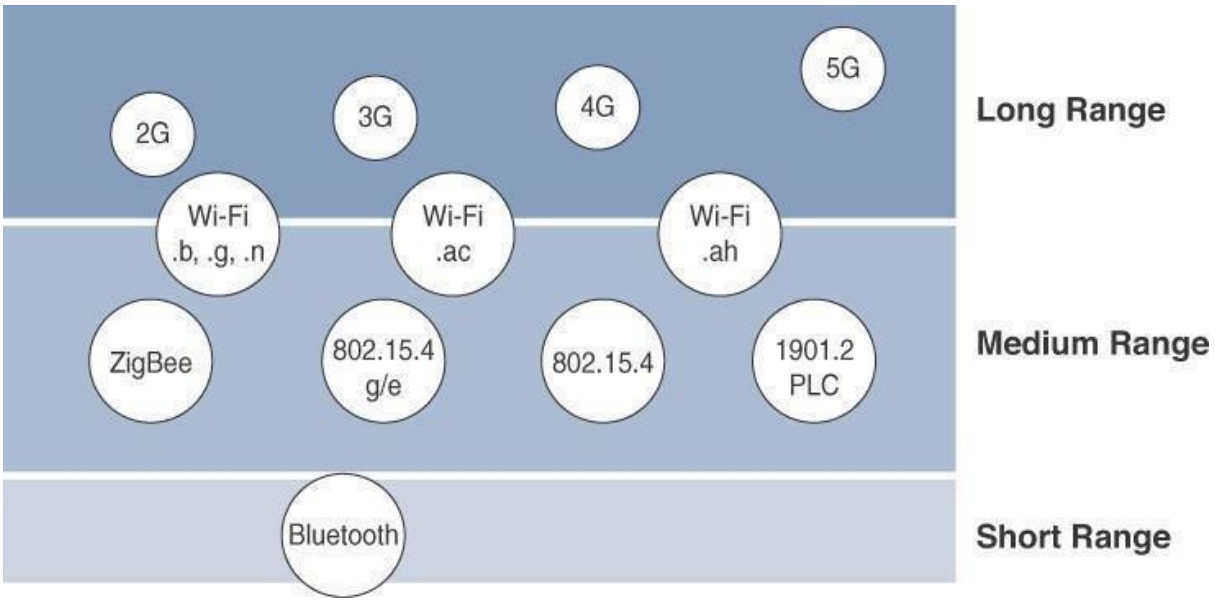



**Figure 4-1** *Wireless Access Landscape*

- **Short range:** The classical wired example is a serial cable. Wireless short-range technologies are often considered as an alternative to a serial cable, supporting tens of meters of maximum distance between two devices. Examples of short-range wireless technologies are IEEE 802.15.1 Bluetooth and IEEE 802.15.7 Visible Light Communications (VLC). These short-range communication methods are found in only a minority of IoT installations. In some cases, they are not mature enough for production deployment. For more information on these IEEE examples, see http://standards.ieee.org/about/get/802/802.15.html.

- **Medium range:** This range is the main category of IoT access technologies. In the range of tens to hundreds of meters, many specifications and implementations are available. The maximum distance is generally less than 1 mile between two devices, although RF technologies do not have real maximum distances defined, as long as the radio signal is transmitted and received in the scope of the applicable specification. Examples of medium-range wireless technologies include IEEE 802.11 Wi-Fi, IEEE 802.15.4, and 802.15.4g WPAN. Wired technologies such as IEEE 802.3 Ethernet and IEEE 1901.2 Narrowband Power Line Communications (PLC) may also be classified as medium range, depending on their physical media characteristics. (All the medium-range protocols just mentioned are covered in more detail later in this chapter.)

- **Long range:** Distances greater than 1 mile between two devices require long-range technologies. Wireless examples are cellular (2G, 3G, 4G) and some applications of outdoor IEEE 802.11 Wi-Fi and Low-Power Wide-Area

(LPWA) technologies. LPWA communications have the ability to communicate over a large area without consuming much power. These technologies are therefore ideal for battery-powered IoT sensors. (LPWA and the other examples just mentioned are discussed in more detail later in this chapter.) Found mainly in industrial networks, IEEE 802.3 over optical fiber and IEEE 1901 Broadband Power Line Communications are classified as long range but are not really considered IoT access technologies. For more information on these standards, see http://standards.ieee.org/about/get/802/802.3.htmlandhttps://standards.ie 2010.html.

**Frequency Bands**

Radio spectrum is regulated by countries and/or organizations, such as the International Telecommunication Union (ITU) and the Federal Communications Commission (FCC). These groups define the regulations and transmission requirements for various frequency bands. For example, portions of the spectrum are allocated to types of telecommunications such as radio, television, military, and so on.

Around the world, the spectrum for various communications uses is often viewed as a critical resource. For example, you can see the value of these frequencies by examining the cost that mobile operators pay for licenses in the cellular spectrum.

Focusing on IoT access technologies, the frequency bands leveraged by wireless communications are split between licensed and unlicensed bands. Licensed spectrum is generally applicable to IoT long-range access technologies and allocated to communications infrastructures deployed by services providers, public services (for example, first responders, military), broadcasters, and utilities.

An important consideration for IoT access infrastructures that wish to utilize licensed spectrum is that users must subscribe to services when connecting their IoT devices. This adds more complexity to a deployment involving large numbers of sensors and other IoT devices, but in exchange for the subscription fee, the network operator can guarantee the exclusivity of the frequency usage over the target area and can therefore sell a better guarantee of service.

bands for device communications. For IoT access, these are the most well- known ISM bands:

- 2.4 GHz band as used by IEEE 802.11b/g/n Wi-Fi
- IEEE 802.15.1 Bluetooth
- IEEE 802.15.4 WPAN

An unlicensed band, such as those in the ISM range of frequencies, is not *unregulated*. National and regional regulations exist for each of the allocated frequency bands (much as with the licensed bands). These regulations mandate device compliance on parameters such as transmit power, duty cycle and dwell time, channel bandwidth, and channel hopping.

Unlicensed spectrum is usually simpler to deploy than licensed because it does not require a service provider. However, it can suffer from more interference because other devices may be competing for the same frequency in a specific area. This becomes a key element in decisions for IoT deployments. Should an IoT infrastructure utilize unlicensed spectrum available for private networks or licensed frequencies that are dependent on a service provider? Various LPWA technologies are taking on a greater importance when it comes to answering this question. In addition to meeting low power requirements, LPWA communications are able to cover long distances that in the past required the licensed bands offered by service providers for cellular devices.

Some communications within the ISM bands operate in the sub-GHz range. Sub-GHz bands are used by protocols such as IEEE 802.15.4, 802.15.4g, and 802.11ah, and LPWA technologies such as LoRa and Sigfox. (All these technologies are discussed in more detail later in this chapter.)

The frequency of transmission directly impacts how a signal propagates and its practical maximum range. (Range and its importance to IoT access are discussed earlier in this chapter.) Either for indoor or outdoor deployments, the sub-GHz frequency bands allow greater distances between devices. These bands have a better ability than the 2.4 GHz ISM band to penetrate building infrastructures or go around obstacles, while keeping the transmit power within regulation.

The disadvantage of sub-GHz frequency bands is their lower rate of data delivery compared to higher frequencies. However, most IoT sensors do not need to send data at high rates. Therefore, the lower transmission speeds of sub-GHz technologies are usually not a concern for IoT sensor deployments.

For example, in most European countries, the 169 MHz band is often considered best suited for wireless water and gas metering applications. This is due to its good deep building basement signal penetration. In addition, the low data rate of this frequency matches the low volume of data that needs to be transmitted.

Several sub-GHz ranges have been defined in the ISM band. The most well- known ranges are centered on 169 MHz, 433 MHz, 868 MHz, and 915 MHz. However, most IoT access technologies tend to focus on the two sub-GHz frequency regions around 868 MHz and 915 MHz. These main bands are commonly found throughout the world and are applicable to nearly all countries.

The European Conference of Postal and Telecommunications Administrations (CEPT), in the European Radiocommunications Committee (ERC) Recommendation 70-03, defines the 868 MHz frequency band. CEPT was established in 1959 as a coordinating body for European state telecommunications and postal organizations. European countries generally apply Recommendation 70-03 to their national telecommunications regulations, but the 868 MHz definition is also applicable to regions and countries outside Europe. For example, India, the Middle East, Africa, and Russia have adopted the CEPT definitions, some of them making minor revisions. Recommendation 70-03 mostly characterizes the use of the 863– 870 MHz band, the allowed transmit power, or EIRP (effective isotropic radiated power), and duty cycle

(that is, the percentage of time a device can be active in transmission). EIRP is the amount of power that an antenna would emit to produce the peak power density observed in the direction of maximum antenna gain. The 868 MHz band is applicable to IoT access technologies such as IEEE 802.15.4 and 802.15.4g, 802.11ah, and LoRaWAN.

Centered on 915 MHz, the 902–928 MHz frequency band is the main unlicensed sub-GHz band available in North America, and it conforms to FCC regulations (FCC-Part-15.247). Countries around the world that do not align on the CEPT ERC 70-03 recommendation generally endorse the use of the 902–928 MHz range or a subset of it in their national regulations. For example, Brazilian regulator ANATEL defines the use of 902–907.5 and 915– 928 MHz ranges (ANATEL506), the Japanese regulator ARIB provisions the 920–928 MHz range (ARIB-T108), and in Australia, ACMA provides recommendations for the 915–928 MHz range. As mentioned previously, even though these bands are unlicensed, they are regulated. The regulators document parameters, such as channel bandwidth, channel hopping, transmit power or EIRP, and dwell time.

In summary, you should take into account the frequencies and corresponding regulations of a country when implementing or deploying IoT smart objects. Smart objects running over unlicensed bands can be easily optimized in terms of hardware supporting the two main worldwide sub-GHz frequencies, 868 MHz and 915 MHz. However, parameters such as transmit power, antennas, and EIRP must be properly designed to follow the settings required by each country's regulations.

**Power Consumption**

While the definition of *IoT device* is very broad, there is a clear delineation between powered nodes and battery-powered nodes. A powered node has a direct connection to a power source, and communications are usually not limited by power consumption criteria. However, ease of deployment of powered nodes is limited by the availability of a power source, which makes mobility more complex.

Battery-powered nodes bring much more flexibility to IoT devices. These nodes are often classified by the required lifetimes of their batteries. Does a node need 10 to 15 years of battery life, such as on water or gas meters? Or is a 5- to 7-year battery life sufficient for devices such as smart parking sensors? Their batteries can be changed or the devices replaced when a street gets resurfaced. For devices under regular maintenance, a battery life of 2 to 3 years is an option.

IoT wireless access technologies must address the needs of low power consumption and connectivity for battery-powered nodes. This has led to the evolution of a new wireless environment known as Low-Power Wide-Area (LPWA). Obviously, it is possible to run just about any wireless technology on batteries. However, in reality, no operational deployment will be acceptable if hundreds of batteries must be changed every month.

Wired IoT access technologies consisting of powered nodes are not exempt from power optimization. In the case of deployment of smart meters over PLC, the radio interface on meters can't consume 5 to 10 watts of power, or this will add up to a 20-million-meter deployment consuming 100 to 200 megawatts of energy for communications.

**Topology**

Among the access technologies available for connecting IoT devices, three main topology schemes are dominant: star, mesh, and peer-to-peer. For long- range and short-range technologies, a star topology is prevalent, as seen with cellular, LPWA, and Bluetooth networks. Star topologies utilize a single central base station or controller to allow communications with endpoints.

For medium-range technologies, a star, peer-to-peer, or mesh topology is common, as shown in Figure 4-2. Peer-to-peer topologies allow any device to communicate with any other device as long as they are in range of each other. Obviously, peer-to-peer topologies rely on multiple full-function devices.
Peer-to-peer topologies enable more complex formations, such as a mesh networking topology.

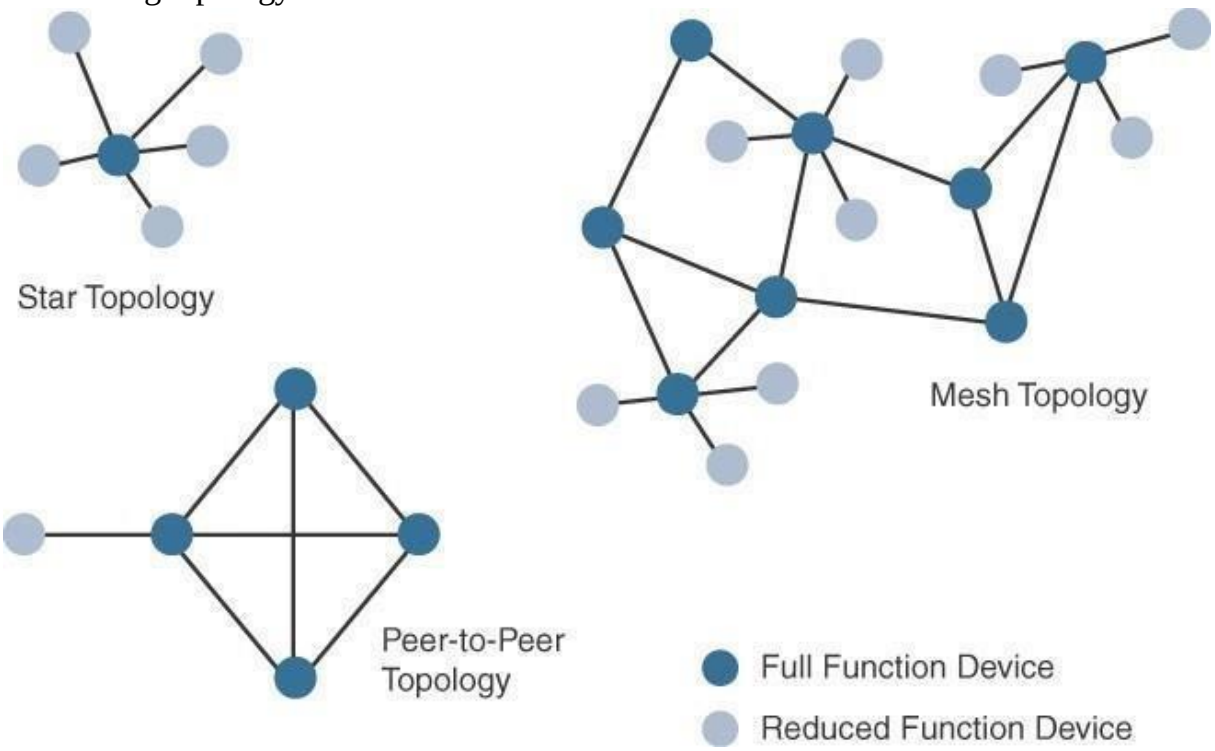



**Figure 4-2** *Star, Peer-to-Peer, and Mesh Topologies*

For example, indoor Wi-Fi deployments are mostly a set of nodes forming a star topology around their access points (APs). Meanwhile, outdoor Wi-Fi may consist of a mesh topology for the backbone of APs, with nodes connecting to the APs in a star topology. Similarly, IEEE 802.15.4 and 802.15.4g and even wired IEEE 1901.2a PLC are generally deployed as a mesh topology. A mesh topology helps cope with low transmit power, searching to reach a greater overall distance, and coverage by having intermediate nodes relaying traffic for other nodes.

Mesh topology requires the implementation of a Layer 2 forwarding protocol known as *mesh-under* or a Layer 3 forwarding protocol referred to as *mesh- over* on each intermediate node. ―IoT Network Architecture and Design,‖ an intermediate node or full-function device (FFD) is simply a node that interconnects other nodes. A node that doesn't interconnect or relay the traffic of other nodes is known as a leaf node, or

reduced-function device (RFD). (More information on full-function and reduced-function devices is also presented later in this chapter.)

While well adapted to powered nodes, mesh topology requires a properly optimized implementation for battery-powered nodes. Battery-powered nodes are often placed in a ―sleep mode‖ to preserve battery life when not transmitting. In the case of mesh topology, either the battery-powered nodes act as leaf nodes or as a ―last resource path‖ to relay traffic when used as intermediate nodes. Otherwise, battery lifetime is greatly shortened. For battery-powered nodes, the topology type and the role of the node in the topology (for example, being an intermediate or leaf node) are significant factors for a successful implementation.

### Constrained Devices

The Internet Engineering Task Force (IETF) acknowledges in RFC 7228 that different categories of IoT devices are deployed. While categorizing the class of IoT nodes is a perilous exercise, with computing, memory, storage, power, and networking continuously evolving and improving, RFC 7228 gives some definitions of constrained nodes. These definitions help differentiate constrained nodes from unconstrained nodes, such as servers, desktop or laptop computers, and powerful mobile devices such as smart phones.

| Class | Definition |
|---|---|
| Class 0 | This class of nodes is severely constrained, with less than 10 KB of memory and less than 100 KB of Flash processing and storage capability. These nodes are typically battery powered. They do not have the resources required to directly implement an IP stack and associated security mechanisms. An example of a Class 0 node is a push button that sends 1 byte of information when changing its status. This class is particularly well suited to leveraging new unlicensed LPWA wireless technology. |
| Class 1 | While greater than Class 0, the processing and code space characteristics (approximately 10 KB RAM and approximately 100 KB Flash) of Class 1 are still lower than expected for a complete IP stack implementation. They cannot easily communicate with nodes employing a full IP stack. However, these nodes can implement an optimized stack specifically designed for constrained nodes, such as Constrained Application Protocol (CoAP). This allows Class 1 nodes to engage in meaningful conversations with the network without the help of a gateway, and provides support for the necessary security functions. Environmental sensors are an example of Class 1 nodes. |
| Class 2 | Class 2 nodes are characterized by running full implementations of an IP stack on embedded devices. They contain more than 50 KB of memory and 250 KB of Flash, so they can be fully integrated in IP networks. A smart power meter is an example of a Class 2 node. |

Constrained nodes have limited resources that impact their networking feature set and capabilities. Therefore, some classes of IoT nodes do not implement an IP stack. According to RFC 7228, constrained nodes can be broken down into the classes defined in Table 4-1.

**Table 4-1** *Classes of Constrained Nodes, as Defined by RFC 7228*

## Constrained-Node Networks

While several of the IoT access technologies, such as Wi-Fi and cellular, are applicable to laptops, smart phones, and some IoT devices, some IoT access technologies are more suited to specifically connect constrained nodes.
Typical examples are IEEE 802.15.4 and 802.15.4g RF, IEEE 1901.2a PLC, LPWA, and IEEE 802.11ah access technologies.

Constrained-node networks are often referred to as low-power and lossy networks (LLNs). (See Chapter 5 for more details on LLNs.) *Low-power* in the context of LLNs refers to the fact that nodes must cope with the requirements from powered and battery-powered constrained nodes. *Lossy networks* indicates that network performance may suffer from interference and variability due to harsh radio environments. Layer 1 and Layer 2 protocols that can be used for constrained-node networks must be evaluated in the context of the following characteristics for use-case applicability: data rate and throughput, latency and determinism, and overhead and payload.

## Data Rate and Throughput

The data rates available from IoT access technologies range from 100 bps with protocols such as Sigfox to tens of megabits per second with technologies such as LTE and IEEE 802.11ac. (Sigfox, LTE, and IEEE 802.11ac are discussed later in this chapter.) However, the actual throughput is less—sometimes much less—than the data rate. Therefore, understanding the bandwidth requirements of a particular technology, its applicability to given use cases, the capacity planning rules, and the expected real throughput are important for proper network design and successful production deployment.

Technologies not particularly designed for IoT, such as cellular and Wi-Fi, match up well to IoT applications with high bandwidth requirements. For example, nodes involved with video analytics have a need for high data rates. These nodes are found in retail, airport, and smart cities environments for detecting events and driving actions. Because these types of IoT endpoints are not constrained in terms of computing or network bandwidth, the design guidelines tend to focus on application requirements, such as latency and determinism.

Short-range technologies can also provide medium to high data rates that have enough throughput to connect a few endpoints. For example, Bluetooth sensors that are now appearing on connected wearable fall into this category. In this case, the solutions focus more on footprint and battery lifetime than on data rate.

The IoT access technologies developed for constrained nodes are optimized for low power consumption, but they are also limited in terms of data rate, which depends on the selected frequency band, and throughput.

With the data rate ranging from 100 bps to less than 1 Mbps, you may think back to the years when bandwidth was a scarce resource. You often needed some expertise to understand how to design such networks. Today this sort of expertise is helpful for

LPWA networks, which are designed with a certain number of messages per day or per endpoint rather than just having a pure bandwidth usage limit in place. In addition, in an access mesh topology, an application's behavior, such as frequency polling, impacts the design because all devices share the constrained bandwidth capacity.

A discussion of data rate and bandwidth in LLNs must include a look at real throughput, or ―goodput,‖ as seen by the application. While it may not be important for constrained nodes that send only one message a day, real throughput is often very important for constrained devices implementing an IP stack. In this case, throughput is a lower percentage of the data rate, even if the node gets the full constrained network at a given time.

For example, let's consider an IEEE 802.15.4g subnetwork implementing 2FSK modulation at 150 kbps for the 915 MHz frequency band. (The IEEE 802.15.4g protocol is covered in more detail later in this chapter.) To cover the border case of distance and radio signal quality, Forward Error Correction (FEC) will be turned on, which lowers the data rate from 150 kbps to 75 kbps. If you now add in the protocol stack overhead, the two-way communication handling, and the variable data payload size, you end up with a maximum throughput of 30 to 40 kbps. This must be considered as the best value because the number of devices simultaneously communicating along with the topology and control plane overhead will also impact the throughput.

Another characteristic of IoT devices is that a majority of them initiate the communication. Upstream traffic toward an application server is usually more common than downstream traffic from the application server. Understanding this behavior also helps when deploying an IoT access technology, such as cellular, that is asymmetrical because the upstream bandwidth must be considered a key parameter for profiling the network capacity.

### Latency and Determinism

Much like throughput requirements, latency expectations of IoT applications should be known when selecting an access technology. This is particularly true for wireless networks, where packet loss and retransmissions due to interference, collisions, and noise are normal behaviors.

On constrained networks, latency may range from a few milliseconds to seconds, and applications and protocol stacks must cope with these wide- ranging values. For example, UDP at the transport layer is strongly recommended for IP endpoints communicating over LLNs. In the case of mesh topologies, if communications are needed between two devices inside the mesh, the forwarding path may call for some routing optimization, which is available using the IPv6 RPL protocol. (For more information on RPL, see Chapter 5.)

### Overhead and Payload

When considering constrained access network technologies, it is important to review the MAC payload size characteristics required by applications. In addition, you should

be aware of any requirements for IP. The minimum IPv6 MTU size is expected to be 1280 bytes. Therefore, the fragmentation of the IPv6 payload has to be taken into account by link layer access protocols with smaller MTUs.

For technologies that fall under the LLN definition but are able to transport IP, such as IEEE 802.15.4 and 802.15.4g, IEEE 1901.2, and IEEE 802.11ah,
Layer 1 or Layer 2 fragmentation capabilities and/or IP optimization is important. (The protocols IEEE 802.14 and 802.15.4g, IEEE 1901.2, and IEEE 802.11ah are covered later in this chapter.) For example, the payload size for IEEE 802.15.4 is 127 bytes and requires an IPv6 payload with a minimum MTU of 1280 bytes to be fragmented. (For more information on the fragmentation of IPv6, see Chapter 5.) On the other hand, IEEE 802.15.4g enables payloads up to 2048 bytes, easing the support of the IPv6 minimum MTU of 1280 bytes.

Most LPWA technologies offer small payload sizes. These small payload sizes are defined to cope with the low data rate and time over the air or duty cycle requirements of IoT nodes and sensors. For example, payloads may be as little as 19 bytes using LoRaWAN technology or up to 250 bytes, depending on the adaptive data rate (ADR). While this doesn't preclude the use of an IPv6/6LoWPAN payload, as seen on some endpoint implementations, these types of protocols are better suited to Class 0 and 1 nodes, as defined in RFC 7228.

In conclusion, the communication criteria just covered are fundamental to understanding IoT access technologies, their characteristics, and when they are most applicable. These criteria include range, frequency bands, power consumption, network topology, the presence of constrained devices and/or networks, and data throughput.

From a network engineer perspective, you must make sure an architecture is developed with the proper abstraction for a particular access technology. This is especially true for constrained network nodes, where quite often your choices of protocols and solutions can be limited. The next section reviews the main IoT access technologies dedicated to constrained networks.

### IoT Access Technologies

The previous section describes criteria that help you in evaluating IoT constrained network technologies for proper design and operations. This section provides an overview of the main IoT access technologies. The technologies highlighted here are the ones that are seen as having market and/or mind share. Therefore, you should have a basic familiarity with them as they are fundamental to many IoT conversations.

For each of the IoT access technologies discussed in this chapter, a common information set is being provided. Particularly, the following topics are addressed for each IoT access technology:

- **Standardization and alliances:** The standards bodies that maintain the protocols for a technology
- **Physical layer:** The wired or wireless methods and relevant

frequencies

- **MAC layer:** Considerations at the Media Access Control (MAC) layer, which bridges the physical layer with data link control
- **Topology:** The topologies supported by the technology
- **Security:** Security aspects of the technology
- **Competitive technologies:** Other technologies that are similar and may be suitable alternatives to the given technology

While having a familiarity with these protocols and their capabilities is recommended, you may find that much of the information about these technologies is better used as reference material. When you encounter these protocols, you can use this chapter as a handy overview and quick summary of the important details.

### IEEE 802.15.4

IEEE 802.15.4 is a wireless access technology for low-cost and low-data-rate devices that are powered or run on batteries. In addition to being low cost and offering a reasonable battery life, this access technology enables easy installation using a compact protocol stack while remaining both simple and flexible. Several network communication stacks, including deterministic ones, and profiles leverage this technology to address a wide range of IoT use cases in both the consumer and business markets. IEEE 802.15.4 is commonly found in the following types of deployments:

- Home and building automation automotive
- networks
- Industrial wireless sensor networks
- Interactive toys and remote controls

Criticisms of IEEE 802.15.4 often focus on its MAC reliability, unbounded latency, and susceptibility to interference and multipath fading. The negatives around reliability and latency often have to do with the Collision Sense Multiple Access/Collision Avoidance (CSMA/CA) algorithm. CSMA/CA is an access method in which a device ―listens‖ to make sure no other devices are transmitting before starting its own transmission. If another device is transmitting, a wait time (which is usually random) occurs before ―listening‖ occurs again. Interference and multipath fading occur with IEEE 802.15.4 because it lacks a frequency-hopping technique. Later variants of 802.15.4 from the IEEE start to address these issues. (See the section ―IEEE 802.15.4e and 802.15.4g,‖ later in this chapter, for more information.)

### Standardization and Alliances

IEEE 802.15.4 or IEEE 802.15 Task Group 4 defines low-data-rate PHY and MAC layer specifications for wireless personal area networks (WPAN). This standard has evolved over the years and is a well-known solution for low- complexity wireless devices with low data rates that need many months or even years of battery life. For more detailed information on IEEE 802.15.4, visit Since 2003, the IEEE has

published several iterations of the IEEE 802.15.4 specification, each labeled with the publication's year. For example, IEEE 802.15.4-2003 was published in 2003, 802.15.4-2006 was released in 2006, and 802.15.4-2011 and 802.15.4-2015 were issued in 2011 and 2015, respectively. Newer releases typically supersede older ones, integrate addendums, and add features or clarifications to previous versions.

While there is no alliance or promotion body for IEEE 802.15.4 per se, the IEEE 802.15.4 PHY and MAC layers are the foundations for several networking protocol stacks. These protocol stacks make use of 802.15.4 at the physical and link layer levels, but the upper layers are different. These protocol stacks are promoted separately through various organizations and often commercialized. Some of the most well-known protocol stacks based on 802.15.4 are highlighted in Table 4-2.

| Protocol | Description |
|---|---|
| ZigBee | Promoted through the ZigBee Alliance, ZigBee defines upper-layer components (network through application) as well as application profiles. Common profiles include building automation, home automation, and healthcare. ZigBee also defines device object functions, such as device role, device discovery, network join, and security. For more information on ZigBee, see the ZigBee Alliance webpage, at www.zigbee.org. ZigBee is also discussed in more detail later in the next Section. |
| 6LoWPAN | 6LoWPAN is an IPv6 adaptation layer defined by the IETF 6LoWPAN working group that describes how to transport IPv6 packets over IEEE 802.15.4 layers. RFCs document header compression and IPv6 enhancements to cope with the specific details of IEEE 802.15.4. (For more information on 6LoWPAN, see Chapter 5.) |
| ZigBee IP | An evolution of the ZigBee protocol stack, ZigBee IP adopts the 6LoWPAN adaptation layer, IPv6 network layer, and RPL routing protocol. In addition, it offers improvements to IP security. ZigBee IP is discussed in more detail later in this chapter. |
| ISA100.11a | ISA100.11a is developed by the International Society of Automation (ISA) as "Wireless Systems for Industrial Automation: Process Control and Related Applications." It is based on IEEE 802.15.4-2006, and specifications were published in 2010 and then as IEC 62734. The network and transport layers are based on IETF 6LoWPAN, IPv6, and UDP standards. |
| WirelessHART | WirelessHART, promoted by the HART Communication Foundation, is a protocol stack that offers a time-synchronized, self-organizing, and self-healing mesh architecture, leveraging IEEE 802.15.4-2006 over the 2.4 GHz frequency band. A good white paper on WirelessHART can be found at http://www.emerson.com/resource/blob/system-engineering-guidelines-iec-62591-wirelesshart--data-79900.pdf |
| Thread | Constructed on top of IETF 6LoWPAN/IPv6, Thread is a protocol stack for a secure and reliable mesh network to connect and control products in the home. Specifications are defined and published by the Thread Group at www.threadgroup.org. |

**Table 4-2** *Protocol Stacks Utilizing IEEE 802.15.4*

Because of its relatively long history compared to the others, ZigBee is one of the most well-known protocols listed in Table 4-2. In addition, ZigBee has continued to evolve over time as evidenced by the release of Zigbee IP and is representative of how IEEE 802.15.4 can be leveraged at the PHY and MAC layers, independent of the protocol

layers above. For these reasons, both Zigbee and Zigbee IP are discussed in more detail in the following sections.

**ZigBee**

Based on the idea of ZigBee-style networks in the late 1990s, the first ZigBee specification was ratified in 2004, shortly after the release of the IEEE 802.15.4 specification the previous year. While not released as a typical standard, like an RFC, ZigBee still had industry support from more than 100 companies upon its initial publication. This industry support has grown to more than 400 companies that are members of the ZigBee Alliance. Similar to the Wi-Fi Alliance, the Zigbee Alliance is an industry group formed to certify interoperability between vendors and it is committed to driving and evolving ZigBee as an IoT solution for interconnecting smart objects.

ZigBee solutions are aimed at smart objects and sensors that have low bandwidth and low power needs. Furthermore, products that are ZigBee compliant and certified by the ZigBee Alliance should interoperate even though different vendors may manufacture them.

The Zigbee specification has undergone several revisions. In the 2006 revision, sets of commands and message types were introduced, and increased in number in the 2007 (called Zigbee pro) iteration, to achieve different functions for a device, such as metering, temperature, or lighting control.
These sets of commands and message types are called clusters. Ultimately, these clusters from different functional domains or libraries form the building blocks of Zigbee application profiles. Vendors implementing pre-defined Zigbee application profiles like Home Automation or Smart Energy can ensure interoperability between their products.

The main areas where ZigBee is the most well-known include automation for commercial, retail, and home applications and smart energy. In the industrial and commercial automation space, ZigBee-based devices can handle various functions, from measuring temperature and humidity to tracking assets. For home automation, ZigBee can control lighting, thermostats, and security functions. ZigBee Smart Energy brings together a variety of interoperable products, such as smart meters, that can monitor and control the use and delivery of utilities, such as electricity and water. These ZigBee products are controlled by the utility provider and can help coordinate usage between homes and businesses and the utility provider itself to provide more efficient operations.

The traditional ZigBee stack is illustrated in Figure 4-3. As mentioned previously, ZigBee utilizes the IEEE 802.15.4 standard at the lower PHY and MAC layers. (The 802.15.4 PHY and MAC layers are covered in detail later in this chapter.) ZigBee specifies the network and security layer and application support layer that sit on top of the lower layers.

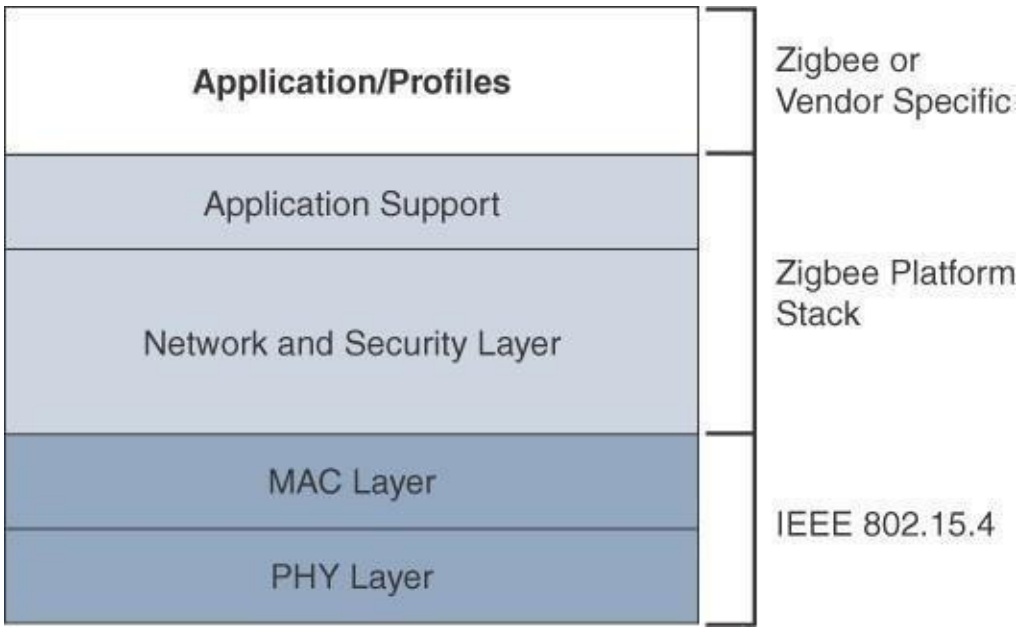



**Figure 4-3** *High-Level ZigBee Protocol Stack*

The ZigBee network and security layer provides mechanisms for network startup, configuration, routing, and securing communications. This includes calculating routing paths in what is often a changing topology, discovering neighbors, and managing the routing tables as devices join for the first time. The network layer is also responsible for forming the appropriate topology, which is often a mesh but could be a star or tree as well. From a security perspective, ZigBee utilizes 802.15.4 for security at the MAC layer, using the Advanced Encryption Standard (AES) with a 128-bit key and also provides security at the network and application layers.

The application support layer in Figure 4-3 interfaces the lower portion of the stack dealing with the networking of ZigBee devices with the higher-layer applications. ZigBee predefines many application profiles for certain industries, and vendors can optionally create their own custom ones at this layer. As mentioned previously, Home Automation and Smart Energy are two examples of popular application profiles.

ZigBee is one of the most well-known protocols built on an IEEE 802.15.4 foundation. On top of the 802.15.4 PHY and MAC layers, ZigBee specifies its own network and security layer and application profiles. While this structure has provided a fair degree of interoperability for vendors with membership in the ZigBee Alliance, it has not provided interoperability with other IoT solutions. However, this has started to change with the release of ZigBee IP, which is discussed next.

**ZigBee IP**

With the introduction of ZigBee IP, the support of IEEE 802.15.4 continues, but the IP and TCP/UDP protocols and various other open standards are now supported at the network and transport layers. The ZigBee-specific layers are now found only at the top of the protocol stack for the applications.

ZigBee IP was created to embrace the open standards coming from the IETF's work on LLNs, such as IPv6, 6LoWPAN, and RPL. (These IETF standards are discussed in

Chapter 5.) They provide for low-bandwidth, low-power, and cost-effective communications when connecting smart objects.

ZigBee IP is a critical part of the Smart Energy (SE) Profile 2.0 specification from the ZigBee Alliance. SE 2.0 is aimed at smart metering and residential energy management systems. In fact, ZigBee IP was designed specifically for SE 2.0 but it is not limited to this use case. Any other applications that need a standards-based IoT stack can utilize Zigbee IP. The ZigBee IP stack is shown in Figure 4-4.

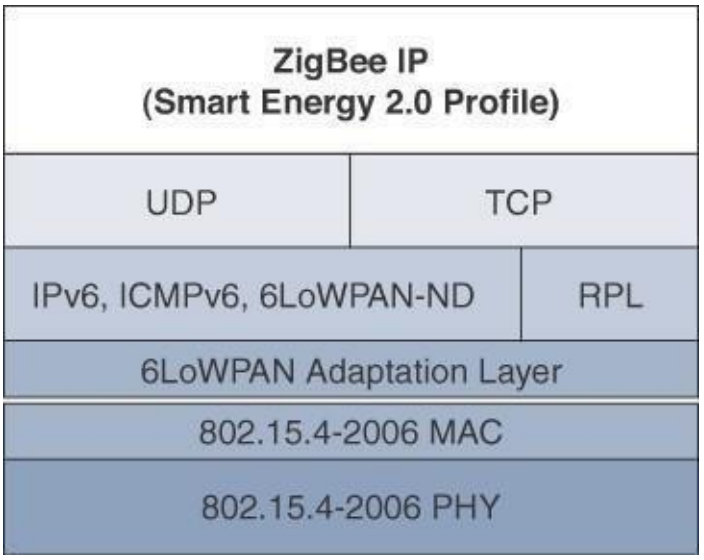



**Figure 4-4** *ZigBee IP Protocol Stack*

Unlike traditional ZigBee, discussed in the previous section, ZigBee IP supports 6LoWPAN as an adaptation layer. (The 6LoWPAN protocol is covered in Chapter 5.) The 6LoWPAN mesh addressing header is not required as ZigBee IP utilizes the mesh-over or route-over method for forwarding packets. ZigBee IP requires the support of 6LoWPAN's fragmentation and header compression schemes.

At the network layer, all ZigBee IP nodes support IPv6, ICMPv6, and 6LoWPAN Neighbor Discovery (ND), and utilize RPL for the routing of packets across the mesh network. IPv6 and RPL are discussed in more detail in Chapter 5. Both TCP and UDP are also supported, to provide both connection-oriented and connectionless service.

### Physical Layer

The 802.15.4 standard supports an extensive number of PHY options that range from 2.4 GHz to sub-GHz frequencies in ISM bands. (ISM bands are discussed earlier in this chapter.) The original IEEE 802.15.4-2003 standard specified only three PHY options based on direct sequence spread spectrum (DSSS) modulation. DSSS is a modulation technique in which a signal is intentionally spread in the frequency domain, resulting in greater bandwidth. The original physical layer transmission options were as follows:

- 2.4 GHz, 16 channels, with a data rate of 250 kbps 915
- MHz, 10 channels, with a data rate of 40 kbps 868

MHz, 1 channel, with a data rate of 20 kbps

You should note that only the 2.4 GHz band operates worldwide. The 915 MHz band operates mainly in North and South America, and the 868 MHz frequencies are used in Europe, the Middle East, and Africa. IEEE 802.15.4- 2006, 802.15.4-2011, and IEEE 802.15.4-2015 introduced additional PHY communication options, including the following:

- **OQPSK PHY:** This is DSSS PHY, employing offset quadrature phase- shift keying (OQPSK) modulation. OQPSK is a modulation technique that uses four unique bit values that are signaled by phase changes. An offset function that is present during phase shifts allows data to be transmitted more reliably.
- **BPSK PHY:** This is DSSS PHY, employing binary phase-shift keying (BPSK) modulation. BPSK specifies two unique phase shifts as its data encoding scheme.
- **ASK PHY:** This is parallel sequence spread spectrum (PSSS) PHY, employing amplitude shift keying (ASK) and BPSK modulation. PSSS is an advanced encoding scheme that offers increased range, throughput, data rates, and signal integrity compared to DSSS. ASK uses amplitude shifts instead of phase shifts to signal different bit values.

These improvements increase the maximum data rate for both 868 MHz and 915 MHz to 100 kbps and 250 kbps, respectively. The 868 MHz support was enhanced to 3 channels, while other IEEE 802.15.4 study groups produced addendums for new frequency bands. For example, the IEEE 802.15.4c study group created the bands 314–316 MHz, 430–434 MHz, and 779–787 MHz for use in China.

Figure 4-5 shows the frame for the 802.15.4 physical layer. The synchronization header for this frame is composed of the Preamble and the Start of Frame Delimiter fields. The Preamble field is a 32-bit 4-byte (for parallel construction) pattern that identifies the start of the frame and is used to synchronize the data transmission. The Start of Frame Delimiter field informs the receiver that frame contents start immediately after this byte.

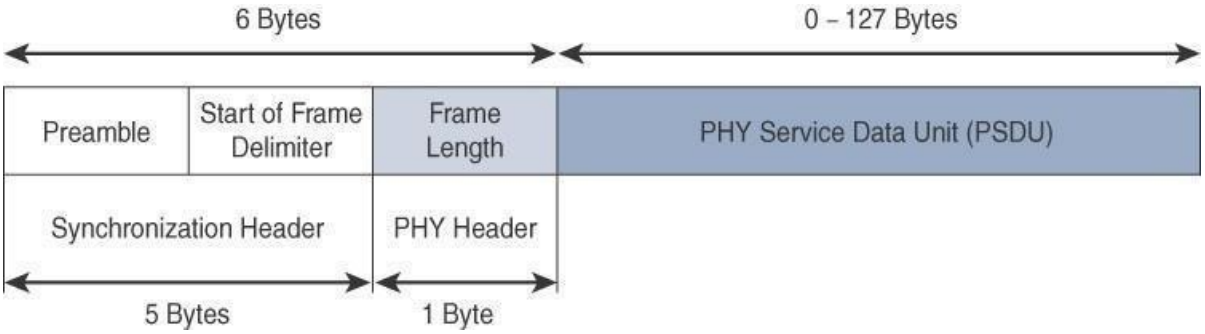



**Figure 4-5** *IEEE 802.15.4 PHY Format*

The PHY Header portion of the PHY frame shown in Figure 4-5 is simply a frame length value. It lets the receiver know how much total data to expect in the PHY service data unit (PSDU) portion of the 802.4.15 PHY. The PSDU is the data field or payload.

The various versions and addendums to 802.15.4 over the years through various working groups can make it somewhat difficult to follow. Therefore, you should pay attention to which versions of 802.15.4 particular devices support. Products and solutions must refer to the proper IEEE 802.15.4 specification, frequency band, modulation, and data rate when providing details on their physical layer implementation.

**MAC Layer**

The IEEE 802.15.4 MAC layer manages access to the PHY channel by defining how devices in the same area will share the frequencies allocated. At this layer, the scheduling and routing of data frames are also coordinated. The 802.15.4 MAC layer performs the following tasks:

- Network beaconing for devices acting as coordinators (New devices use beacons to join an 802.15.4 network)
- PAN association and disassociation by a device
- Device security
- Reliable link communications between two peer MAC entities

The MAC layer achieves these tasks by using various predefined frame types. In fact, four types of MAC frames are specified in 802.15.4:

- **Data frame:** Handles all transfers of data
- **Beacon frame:** Used in the transmission of beacons from a PAN coordinator
- **Acknowledgement frame:** Confirms the successful reception of a frame
- **MAC command frame:** Responsible for control communication between devices

Each of these four 802.15.4 MAC frame types follows the frame format shown in Figure 4-6. In Figure 4-6, notice that the MAC frame is carried as the PHY payload. The 802.15.4 MAC frame can be broken down into the MAC Header, MAC Payload, and MAC Footer fields.

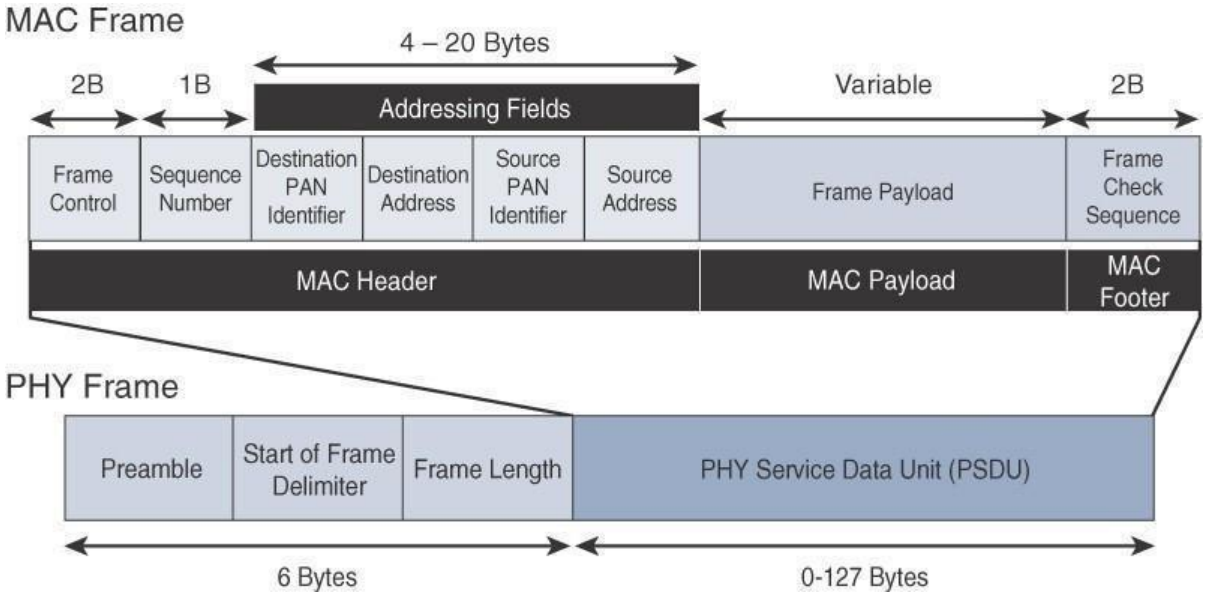



**Figure 4-6** *IEEE 802.15.4 MAC Format*

The MAC Header field is composed of the Frame Control, Sequence Number and the Addressing fields. The Frame Control field defines attributes such as frame type, addressing modes, and other control flags. The Sequence Number field indicates the sequence identifier for the frame. The Addressing field specifies the Source and Destination PAN Identifier fields as well as the Source and Destination Address fields.

The MAC Payload field varies by individual frame type. For example, beacon frames have specific fields and payloads related to beacons, while MAC command frames have different fields present. The MAC Footer field is nothing more than a frame check sequence (FCS). An FCS is a calculation based on the data in the frame that is used by the receiving side to confirm the integrity of the data in the frame.

IEEE 802.15.4 requires all devices to support a unique 64-bit extended MAC address, based on EUI-64. However, because the maximum payload is 127 bytes, 802.15.4 also defines how a 16-bit ―short address‖ is assigned to devices. This short address is local to the PAN and substantially reduces the frame overhead compared to a 64-bit extended MAC address. However, you should be aware that the use of this short address might be limited to specific upper-layer protocol stacks.

### Topology

IEEE 802.15.4–based networks can be built as star, peer-to-peer, or mesh topologies. Mesh networks tie together many nodes. This allows nodes that would be out of range if trying to communicate directly to leverage intermediary nodes to transfer communications.

Please note that every 802.15.4 PAN should be set up with a unique ID. All the nodes in the same 802.15.4 network should use the same PAN ID. Figure 4-7 shows an example of an 802.15.4 mesh network with a PAN ID of 1.

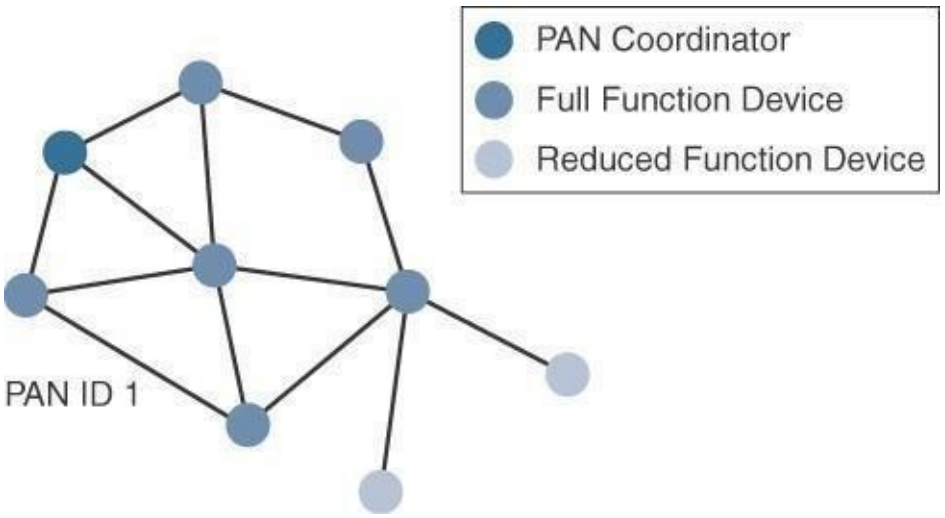



**Figure 4-7** *802.15.4 Sample Mesh Network Topology*

As mentioned earlier in this chapter, full-function devices (FFDs) and reduced-function devices (RFDs) are defined in IEEE 802.15.4. A minimum of one FFD acting as a PAN coordinator is required to deliver services that allow other devices to associate and form a cell or PAN. Notice in Figure 4-7 that a single PAN coordinator is identified for PAN ID 1. FFD devices can communicate with any other devices, whereas RFD devices can communicate only with FFD devices.

**Security**

The IEEE 802.15.4 specification uses Advanced Encryption Standard (AES) with a 128-bit key length as the base encryption algorithm for securing its data. Established by the US National Institute of Standards and Technology in 2001, AES is a block cipher, which means it operates on fixed-size blocks of data. The use of AES by the US government and its widespread adoption in the private sector has helped it become one of the most popular algorithms used in symmetric key cryptography. (A *symmetric key* means that the same key is used for both the encryption and decryption of the data.)

In addition to encrypting the data, AES in 802.15.4 also validates the data that is sent. This is accomplished by a message integrity code (MIC), which is calculated for the entire frame using the same AES key that is used for encryption.

Enabling these security features for 802.15.4 changes the frame format slightly and consumes some of the payload. Using the Security Enabled field in the Frame Control portion of the 802.15.4 header is the first step to enabling AES encryption. This field is a single bit that is set to 1 for security. Once this bit is set, a field called the Auxiliary Security Header is created after the Source Address field, by stealing some bytes from the Payload field. Figure 4-8 shows the IEEE 802.15.4 frame format at a high level, with the Security Enabled bit set and the Auxiliary Security Header field present.

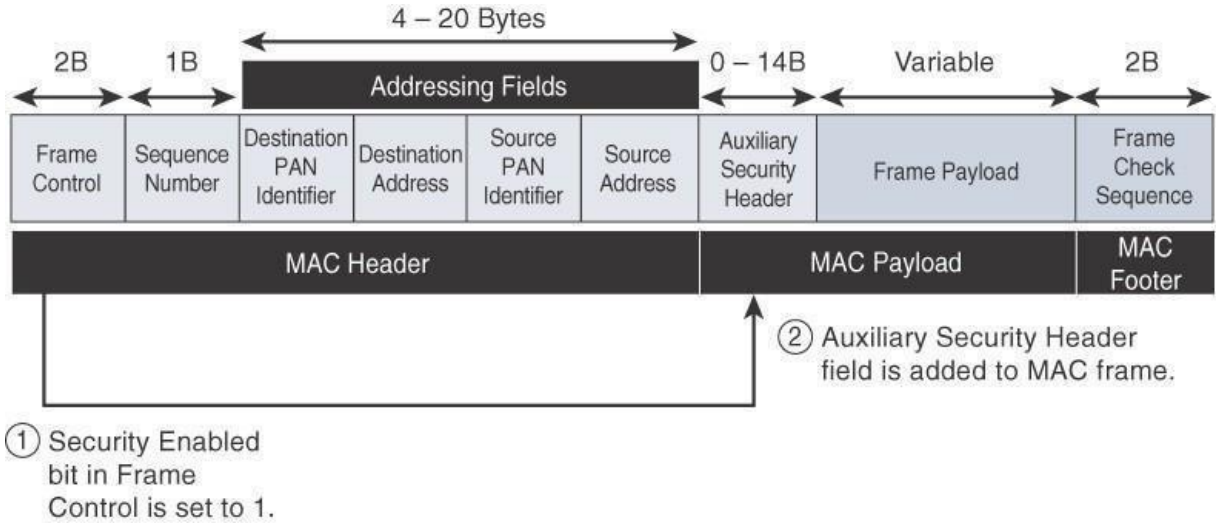



**Figure 4-8** *Frame Format with the Auxiliary Security Header Field for 802.15.4-2006 and Later Versions*

### Competitive Technologies

As detailed in Table 4-2, the IEEE 802.15.4 PHY and MAC layers are the foundations for several networking profiles that compete against each other in various IoT access environments. These various vendors and organizations build upper-layer protocol stacks on top of an 802.15.4 core. They compete and distinguish themselves based on features and capabilities in these upper layers.

A competitive radio technology that is different in its PHY and MAC layers is DASH7. DASH7 was originally based on the ISO18000-7 standard and positioned for industrial communications, whereas IEEE 802.15.4 is more generic. Commonly employed in active radio frequency identification (RFID) implementations, DASH7 was used by US military forces for many years, mainly for logistics purposes. Active RFID utilizes radio waves generated by a battery-powered tag on an object to enable continuous tracking.

The current DASH7 technology offers low power consumption, a compact protocol stack, range up to 1 mile, and AES encryption. Frequencies of 433 MHz, 868 MHz, and 915 MHz have been defined, enabling data rates up to 166.667 kbps and a maximum payload of 256 bytes.

DASH7 is promoted by the DASH7 Alliance, which has evolved the protocol from its active RFID niche into a wireless sensor network technology that is aimed at the commercial market. For more information on DASH7, see the Dash7 Alliance webpage, at www.dash7-alliance.org.

### IEEE 802.15.4 Conclusions

The IEEE 802.15.4 wireless PHY and MAC layers are mature specifications that are the foundation for various industry standards and products (refer to Table 4-2). The PHY layer offers a maximum speed of up to 250 kbps, but this varies based on modulation and frequency. The MAC layer for 802.15.4 is robust and handles how

data is transmitted and received over the PHY layer. Specifically, the MAC layer handles the association and disassociation of devices to/from a PAN, reliable communications between devices, security, and the formation of various topologies.

The topologies used in 802.15.4 include star, peer-to-peer, and cluster trees that allow for the formation of mesh networks. From a security perspective, 802.15.4 utilizes AES encryption to allow secure communications and also provide data integrity.

**IEEE 802.15.4g and 802.15.4e**

The IEEE frequently makes amendments to the core 802.15.4 specification, before integrating them into the next revision of the core specification. When these amendments are made, a lowercase letter is appended. Two such examples of this are 802.15.4e-2012 and 802.15.4g-2012, both of which are especially relevant to the subject of IoT. Both of these amendments were integrated in IEEE 802.15.4-2015 but are often still referred to by their amendment names.

The IEEE 802.15.4e amendment of 802.15.4-2011 expands the MAC layer feature set to remedy the disadvantages associated with 802.15.4, including MAC reliability, unbounded latency, and multipath fading. In addition to making general enhancements to the MAC layer, IEEE 802.15.4e also made improvements to better cope with certain application domains, such as factory and process automation and smart grid. Smart grid is associated with the modernization of the power grid and utilities infrastructure by connecting intelligent devices and communications. IEEE 802.15.4e-2012 enhanced the IEEE 802.15.4 MAC layer capabilities in the areas of frame format, security, determinism mechanism, and frequency hopping. (The specific MAC layer enhancements introduced in IEEE 802.15.4e are covered in more detail later in this chapter.)

IEEE 802.15.4g-2012 is also an amendment to the IEEE 802.15.4-2011 standard, and just like 802.15.4e-2012, it has been fully integrated into the core IEEE 802.15.4-2015 specification. The focus of this specification is the smart grid or, more specifically, smart utility network communication.
802.15.4g seeks to optimize large outdoor wireless mesh networks for field area networks (FANs). New PHY definitions are introduced, as well as some MAC modifications needed to support their implementation. This technology applies to IoT use cases such as the following:

- Distribution automation and industrial supervisory control and data acquisition (SCADA) environments for remote monitoring and control (SCADA is covered in more detail in Chapter 6, ―Application Protocols for IoT.‖)
- Public lighting
- Environmental wireless sensors in smart cities Electrical
- vehicle charging stations
- Smart parking meters Microgrids
- Renewable energy

### Standardization and Alliances

Because 802.15.4g-2012 and 802.15.4e-2012 are simply amendments to IEEE 802.15.4-2011, the same IEEE 802.15 Task Group 4 standards body authors, maintains, and integrates them into the next release of the core specification.
However, the additional capabilities and options provided by 802.15.4g-2012 and 802.15.4e-2012 led to additional difficulty in achieving the interoperability between devices and mixed vendors that users requested.

To guarantee interoperability, the Wi-SUN Alliance was formed. (SUN stands for *smart utility network*.) This organization is not a standards body but is instead an industry alliance that defines communication profiles for smart utility and related networks. These profiles are based on open standards, such as 802.15.4g-2012, 802.15.4e-2012, IPv6, 6LoWPAN, and UDP for the FAN
profile. (For more information on 6LoWPAN, see Chapter 5.) In addition, Wi- SUN offers a testing and certification program to further ensure interoperability.

The Wi-SUN Alliance performs the same function as the Wi-Fi Alliance and WiMAX Forum. Each of these organizations has an associated standards body as well as a commercial name, as shown in Table 4-3. For more information on Wi-SUN, visit www.wi-sun.org.

| Commercial Name/Trademark | Industry Organization | Standards Body |
|---|---|---|
| Wi-Fi | Wi-Fi Alliance | IEEE 802.11 Wireless LAN |
| WiMAX | WiMAX Forum | IEEE 802.16 Wireless MAN |
| Wi-SUN | Wi-SUN Alliance | IEEE 802.15.4g Wireless SUN |

**Table 4-3** *Industry Alliances for Some Common IEEE Standards*

### Physical Layer

In IEEE 802.15.4g-2012, the original IEEE 802.15.4 maximum PSDU or payload size of 127 bytes was increased for the SUN PHY to 2047 bytes. This provides a better match for the greater packet sizes found in many upper-layer protocols. For example, the default IPv6 MTU setting is 1280 bytes.
Fragmentation is no longer necessary at Layer 2 when IPv6 packets are transmitted over IEEE 802.15.4g MAC frames. Also, the error protection was improved in IEEE 802.15.4g by evolving the CRC from 16 to 32 bits.

The SUN PHY, as described in IEEE 802.15.4g-2012, supports multiple data rates in bands ranging from 169 MHz to 2.4 GHz. These bands are covered in the unlicensed ISM frequency spectrum specified by various countries and regions. Within these bands, data must be modulated onto the frequency using at least one of the following PHY mechanisms to be IEEE 802.15.4g compliant:

- **Multi-Rate and Multi-Regional Frequency Shift Keying (MR- FSK):** Offers good transmit power efficiency due to the constant envelope of the

transmit signal

- **Multi-Rate and Multi-Regional Orthogonal Frequency Division Multiplexing (MR-OFDM):** Provides higher data rates but may be too complex for low-cost and low-power devices
- **Multi-Rate and Multi-Regional Offset Quadrature Phase-Shift Keying (MR-O-QPSK):** Shares the same characteristics of the IEEE 802.15.4-2006 O-QPSK PHY, making multi-mode systems more cost- effective and easier to design

Enhanced data rates and a greater number of channels for channel hopping are available, depending on the frequency bands and modulation. For example, for the 902–928 MHz ISM band that is used in the United States, MR-FSK provides 50, 150, or 200 kbps. MR-OFDM at this same frequency allows up to 800 kbps. Other frequencies provide their own settings.

Therefore, products and solutions must refer to the proper IEEE 802.15.4 specification, frequency band, modulation, and data rate when providing details about their PHY implementation. This is important because the availability of chipsets supporting new PHY mechanisms, such as MR- OFDM, may limit the implementation of enhanced data rates. You should look to the Wi-SUN Alliance to mitigate these problems and provide some consistency in terms of implementation, interoperability, and certifications. For example, the Wi-SUN PHY working group publishes a Regional Frequency Bands specification describing the details for various regions and countries.

### MAC Layer

While the IEEE 802.15.4e-2012 amendment is not applicable to the PHY layer, it is pertinent to the MAC layer. This amendment enhances the MAC layer through various functions, which may be selectively enabled based on various implementations of the standard. In fact, if interoperability is a ―must have,‖ then using profiles defined by organizations such as Wi-SUN is necessary. The following are some of the main enhancements to the MAC layer proposed by IEEE 802.15.4e-2012:

- **Time-Slotted Channel Hopping (TSCH):** TSCH is an IEEE 802.15.4e-2012 MAC operation mode that works to guarantee media access and channel diversity. Channel hopping, also known as frequency hopping, utilizes different channels for transmission at different times. TSCH divides time into fixed time periods, or ―time slots,‖ which offer guaranteed bandwidth and predictable latency. In a time slot, one packet and its acknowledgement can be transmitted, increasing network capacity because multiple nodes can communicate in the same time slot, using different channels. A number of time slots are defined as a ―slot frame,‖ which is regularly repeated to provide ―guaranteed access.‖ The transmitter and receiver agree on the channels and the timing for switching between channels through the combination of a global time slot counter and a global channel hopping sequence list, as computed on each node to determine the channel of each time slot.

TSCH adds robustness in noisy environments and smoother coexistence with other wireless technologies, especially for industrial use cases.

- **Information elements:** Information elements (IEs) allow for the exchange of information at the MAC layer in an extensible manner, either as header IEs (standardized) and/or payload IEs (private).
  Specified in a tag, length, value (TLV) format, the IE field allows frames to carry additional metadata to support MAC layer services. These services may include IEEE 802.15.9 key management, Wi-SUN
  1.0 IEs to broadcast and unicast schedule timing information, and frequency hopping synchronization information for the 6TiSCH architecture.
- **Enhanced beacons (EBs):** EBs extend the flexibility of IEEE 802.15.4 beacons to allow the construction of application-specific beacon content. This is accomplished by including relevant IEs in EB frames.
  Some IEs that may be found in EBs include network metrics, frequency hopping broadcast schedule, and PAN information version.
- **Enhanced beacon requests (EBRs):** Like enhanced beacons, an enhanced beacon request (EBRs) also leverages IEs. The IEs in EBRs allow the sender to selectively specify the request of information.
  Beacon responses are then limited to what was requested in the EBR. For example, a device can query for a PAN that is allowing new devices to join or a PAN that supports a certain set of MAC/PHY capabilities.
- **Enhanced Acknowledgement:** The Enhanced Acknowledgement frame allows for the integration of a frame counter for the frame being acknowledged. This feature helps protect against certain attacks that occur when Acknowledgement frames are spoofed.

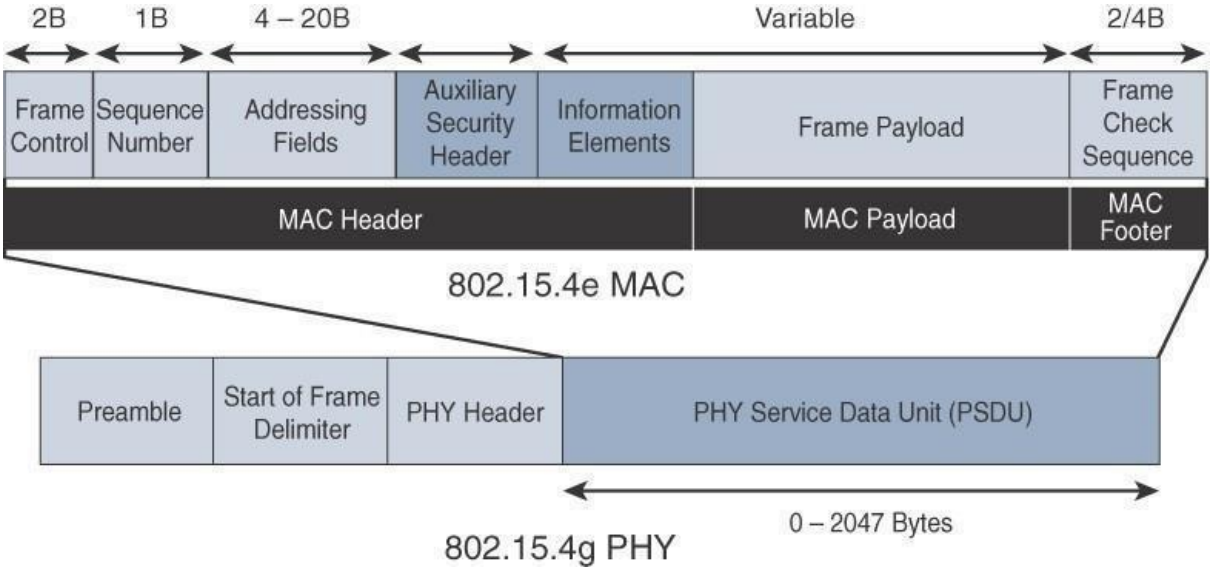



**Figure 4-9** *IEEE 802.15.4g/e MAC Frame Format*

The 802.15.4e MAC is similar to the 802.15.4 MAC in Figure 4-6. The main changes shown in the IEEE 802.15.4e header in Figure 4-9 are the presence of the Auxiliary

Security Header and Information Elements field. The Auxiliary Security header provides for the encryption of the data frame. This field is optionally supported in both 802.15.4e-2012 and 802.15.4, starting with the 802.15.4-2006 specification, as shown in Figure 4-8. As discussed earlier in this section, the IE field contains one or more information elements that allow for additional information to be exchanged at the MAC layer.

**Topology**

Deployments of IEEE 802.15.4g-2012 are mostly based on a mesh topology. This is because a mesh topology is typically the best choice for use cases in the industrial and smart cities areas where 802.15.4g-2012 is applied. A mesh topology allows deployments to be done in urban or rural areas, expanding the distance between nodes that can relay the traffic of other nodes.

Considering the use cases addressed by this technology, powered nodes have been the primary targets of implementations. Support for battery-powered nodes with a long lifecycle requires optimized Layer 2 forwarding or Layer 3 routing protocol implementations. This provides an extra level of complexity but is necessary in order to cope with sleeping battery-powered nodes.

**Security**

Both IEEE 802.15.4g and 802.15.4e inherit their security attributes from the IEEE 802.15.4-2006 specification. Therefore, encryption is provided by AES, with a 128-bit key. In addition to the Auxiliary Security Header field initially defined in 802.15.4-2006, a secure acknowledgement and a secure Enhanced Beacon field complete the MAC layer security. Figure 4-10 shows a high- level overview of the security associated with an IEEE 802.15.4e MAC frame.

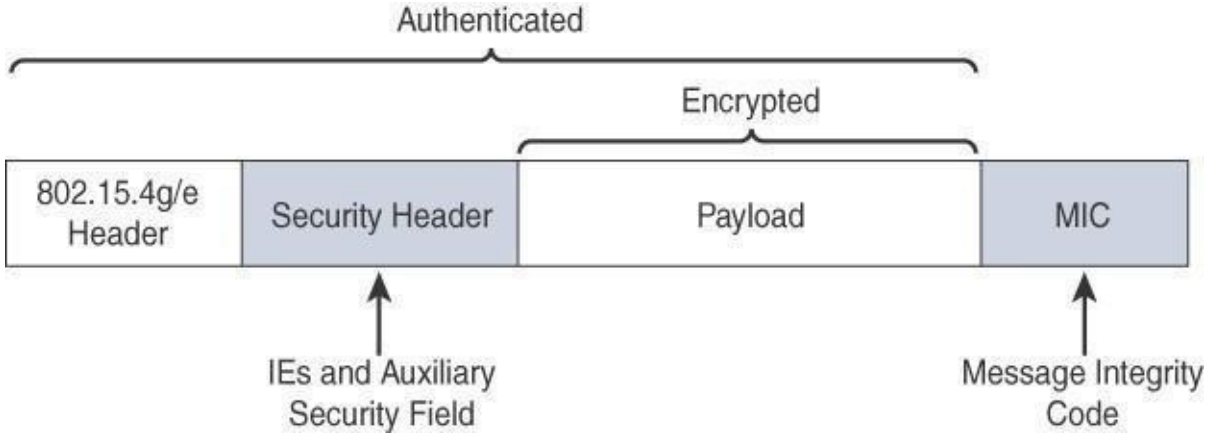



**Figure 4-10** *IEEE 802.15.4g/e MAC Layer Security*

The full frame in Figure 4-10 gets authenticated through the MIC at the end of frame. The MIC is a unique value that is calculated based on the frame contents. (The MIC is discussed in more detail earlier in this chapter.) The

Security Header field denoted in Figure 4-10 is composed of the Auxiliary Security field and one or more Information Elements fields. Integration of the Information Elements fields allows for the adoption of additional security capabilities, such as the IEEE 802.15.9 Key Management Protocol (KMP) specification. KMP provides a means for establishing keys for robust datagram security. Without key management support, weak keys are often the result, leaving the security system open to attack.

### Competitive Technologies

Competitive technologies to IEEE 802.15.4g and 802.15.4e parallel the technologies that also compete with IEEE 802.15.4, such as DASH7. (DASH7 is discussed earlier in this chapter.) In many ways, 802.15.4 and its various flavors of upper-layer protocols, as shown in Table 4-2, can be seen as competitors as well. IEEE 802.15.4 is well established and already deployed in many scenarios, mostly indoors.

### IEEE 802.15.4g and 802.15.4e Conclusions

It is important to remember that IEEE 802.15.4g and 802.15.4e are simply amendments to the IEEE 802.15.4 standard. They are mature specifications that are integrated into IEEE 802.15.4-2015. They have been successfully deployed in real-world scenarios, and already support millions of endpoints. IEEE 802.15.4g focuses mainly on improvements to the PHY layer, while IEEE 802.15.4e targets the MAC layer. These improvements overcome many of the disadvantages of IEEE 802.15.4, such as latency and vulnerability to multipath fading. In addition, provisions in these amendments make them better suited to handle the unique deployment models in the areas of smart grid/utilities and smart cities.

The Wi-SUN Alliance is an important industry alliance that provides interoperability and certification for industry implementations. Utilizing 802.15.4g as a foundation, the alliance releases profiles, such as the FAN profile, to help promote the adoption of the technology while guaranteeing interoperability between vendors. You should expect to see increasing use of both 802.15.4g and 802.15.4e, especially in the smart grid and smart cities verticals of IoT, where they have already seen strong adoption.

### IEEE 1901.2a

While most of the constrained network technologies relate to wireless, IEEE 1901.2a-2013 is a wired technology that is an update to the original IEEE 1901.2 specification. This is a standard for Narrowband Power Line Communication (NB-PLC). NB-PLC leverages a narrowband spectrum for low power, long range, and resistance to interference over the same wires that carry electric power. NB-PLC is often found in use cases such as the following:

- **Smart metering:** NB-PLC can be used to automate the reading of utility meters, such as electric, gas, and water meters. This is true particularly in Europe, where PLC is the preferred technology for utilities deploying smart meter solutions.
- **Distribution automation:** NB-PLC can be used for distribution automation, which involves monitoring and controlling all the devices in the power grid.

- **Public lighting:** A common use for NB-PLC is with public lighting— the lights found in cities and along streets, highways, and public areas such as parks.
- **Electric vehicle charging stations:** NB-PLC can be used for electric vehicle charging stations, where the batteries of electric vehicles can be recharged.
- **Microgrids:** NB-PLC can be used for microgrids, local energy grids that can disconnect from the traditional grid and operate independently.
- **Renewable energy:** NB-PLC can be used in renewable energy applications, such as solar, wind power, hydroelectric, and geothermal heat.

All these use cases require a direct connection to the power grid. So it makes sense to transport IoT data across power grid connections that are already in place.

Multiple PLC standards exist, but the formation of IEEE 1901.2a was driven by the absence of a low-frequency PLC solution below 500 kHz. IEEE 1901.2a specifies the use of both alternating and direct current electric power lines. Low- and medium-voltage lines in both indoor and outdoor environments are supported, along with multiple-mile distances. Data rates can scale up to 500 kbps. The IEEE 1901.2a PHY and MAC layers can be mixed with IEEE 802.15.4g/e on endpoints, offering a dual-PHY solution for some use cases.

**Standardization and Alliances**

The first generations of NB-PLC implementations have generated a lot of interest from utilities in Europe but have often suffered from poor reliability, low throughput (in the range of a few hundred bits per second to a maximum of 2 kbps), lack of manageability, and poor interoperability. This has led several organizations (including standards bodies and alliance consortiums) to develop their own specifications for new generations of NB-PLC technologies. Most recent NB-PLC standards are based on orthogonal frequency-division multiplexing (OFDM). However, different standards from various vendors competing with one another have created a fragmented market. OFDM encodes digital data on multiple carrier frequencies. This provides several parallel streams that suffer less from high frequency attenuation in copper wire and narrowband interference.

The IEEE 1901.2 working group published the IEEE 1901.2a specification in November 2013. Originally leveraging the work done by the G3-PLC (now ITU G.9903) and PRIME (now ITU G.9904) working groups, the IEEE 1901.2 working group only looked at standardizing the NB-PLC PHY and MAC layers (as defined by the IEEE charter and done in other IEEE standards) independently of the upper layers. This differs from G.9903 and G.9904, which were developed for a single use case, smart metering, and focused on running specific application protocols for smart meters.

The HomePlug Alliance was one of the main industry organizations that drove the promotion and certification of PLC technologies, with IEEE 1901.2a being part of its HomePlug Netricity program. In 2016, the HomePlug Alliance made the decision to offer the alliance's broadband power line networking technology to a broader audience

by making its technical specifications publicly available. It has also partnered with other alliances on continuing ongoing work. The HomePlug Alliance has struck a liaison agreement with the Wi-SUN Alliance with the goal of enabling hybrid smart grid networks that support both wireless and power line–wired connectivity.
For more information on the HomePlug Alliance and Netricity, see www.homeplug.org.

**Physical Layer**

NB-PLC is defined for frequency bands from 3 to 500 kHz. Much as with wireless sub-GHz frequency bands, regional regulations and definitions apply to NB-PLC. The IEEE 1901.2 working group has integrated support for all world regions in order to develop a worldwide standard. Specifications include support for CENELEC A and B bands, US FCC-Low and FCC- above-CENELEC, and Japan ARIB bands. CENELEC is the French Comité Européen de Normalisation Électrotechnique, which in English translates to European Committee for Electrotechnical Standardization. This organization is responsible for standardization in the area of electrical engineering for Europe. The CENELEC A and B bands refer to 9–95 kHz and 95–125 kHz, respectively. The FCC is the Federal Communications Commission, a US government organization that regulates interstate and international communications by radio, television, wire, satellite, and cable. The FCC-Low band encompasses 37.5–117.1875 kHz, and the FCC-above-CENELEC band is 154.6875–487.5 kHz. The FCC-above-CENELEC band may become the most useful frequency due to its higher throughput and reduced interference.

Figure 4-11 shows the various frequency bands for NB-PLC. Notice that the most well-known bands are regulated by CENELEC and the FCC, but the Japan Association of Radio Industries and Businesses (ARIB) band is also present. The two ARIB frequency bands are ARIB 1, 37.5–117.1875 kHz, and ARIB 2, 154.6875–403.125 kHz.

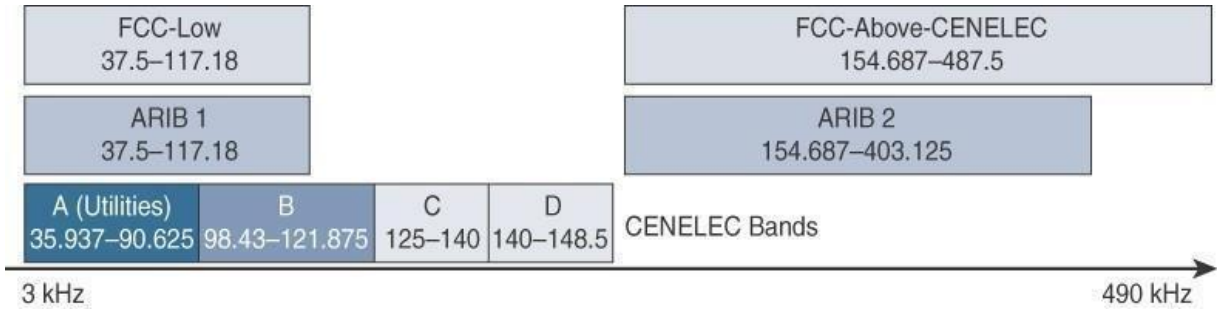



**Figure 4-11** *NB-PLC Frequency Bands*

Based on OFDM, the IEEE 1901.2 specification leverages the best from other NB-PLC OFDM technologies that were developed previously. Therefore, IEEE 1901.2a supports the largest set of coding and enables both robustness and throughput. The standard includes tone maps and modulations, such as robust modulation (ROBO), differential binary phase shift keying (DBPSK), differential quadrature phase shift keying (DQPSK), differential 8-point phase shift keying (D8PSK) for all bands, and optionally 16 quadrature amplitude modulation (16QAM) for some bands. ROBO mode transmits redundant information on multiple carriers, and DBPSK, DQPSK, and

D8PSK are all variations of phase shift keying, where the phase of a signal is changed to signal a binary data transmission. ROBO utilizes QPSK modulation, and its throughput depends on the degree to which coding is repeated across streams. For example, standard ROBO uses a repetition of 4, and Super-ROBO utilizes a repetition of 6. With IEEE 1901.2a, the data throughput rate has the ability to dynamically change, depending on the modulation type and tone map. For CENELEC A band, the data rate ranges from 4.5 kbps in ROBO mode to 46 kbps with D8PSK modulation. For the FCC-above-CENELEC frequencies, throughput varies from 21 kbps in ROBO mode to a maximum of 234 kbps using D8PSK.

## MAC Layer

The MAC frame format of IEEE 1901.2a is based on the IEEE 802.15.4 MAC frame but integrates the latest IEEE 802.15.4e-2012 amendment, which enables key features to be supported. (For more information on the 802.15.4 MAC frame format, refer to Figure 4-6. For the 802.15.4e MAC frame format, see Figure 4-9.) One of the key components brought from 802.15.4e to IEEE 1901.2a is information elements. With IE support, additional capabilities, such as IEEE 802.15.9 Key Management Protocol and SSID, are supported. Figure 4-12 provides an overview of the general MAC frame format for IEEE 1901.2. Note that the numeric value above each field in the frame shows the size of the field, in bytes.

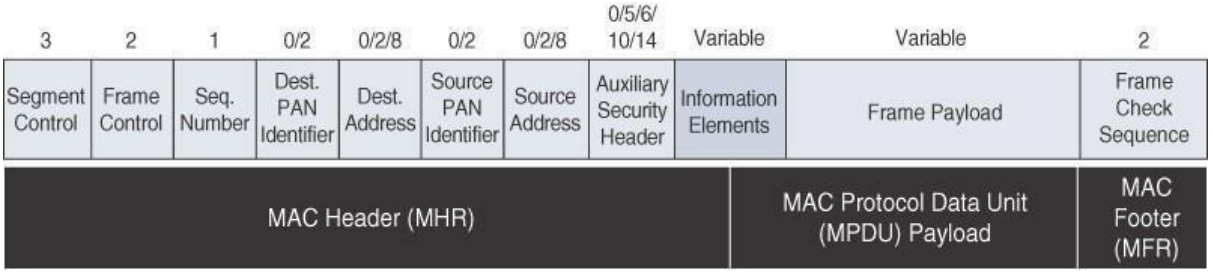



**Figure 4-12** *General MAC Frame Format for IEEE 1901.2*

As shown in Figure 4-12, IEEE 1901.2 has a Segment Control field. This is a new field that was not present in our previous discussions of the MAC frame for 802.15.4 and 802.15.4e. This field handles the segmentation or fragmentation of upper-layer packets with sizes larger than what can be carried in the MAC protocol data unit (MPDU). The rest of the fields are discussed earlier in this chapter and shown in Figures 4-6, 4-8, and 4-9.

## Topology

Use cases and deployment topologies for IEEE 1901.2a are tied to the physical power lines. As with wireless technologies, signal propagation is limited by factors such as noise, interference, distortion, and attenuation. These factors become more prevalent with distance, so most NB-PLC deployments use some sort of mesh topology. Mesh networks offer the advantage of devices relaying the traffic of other devices so longer distances can be segmented. Figure 4-13 highlights a network scenario in which a PLC mesh network is applied to a neighborhood.

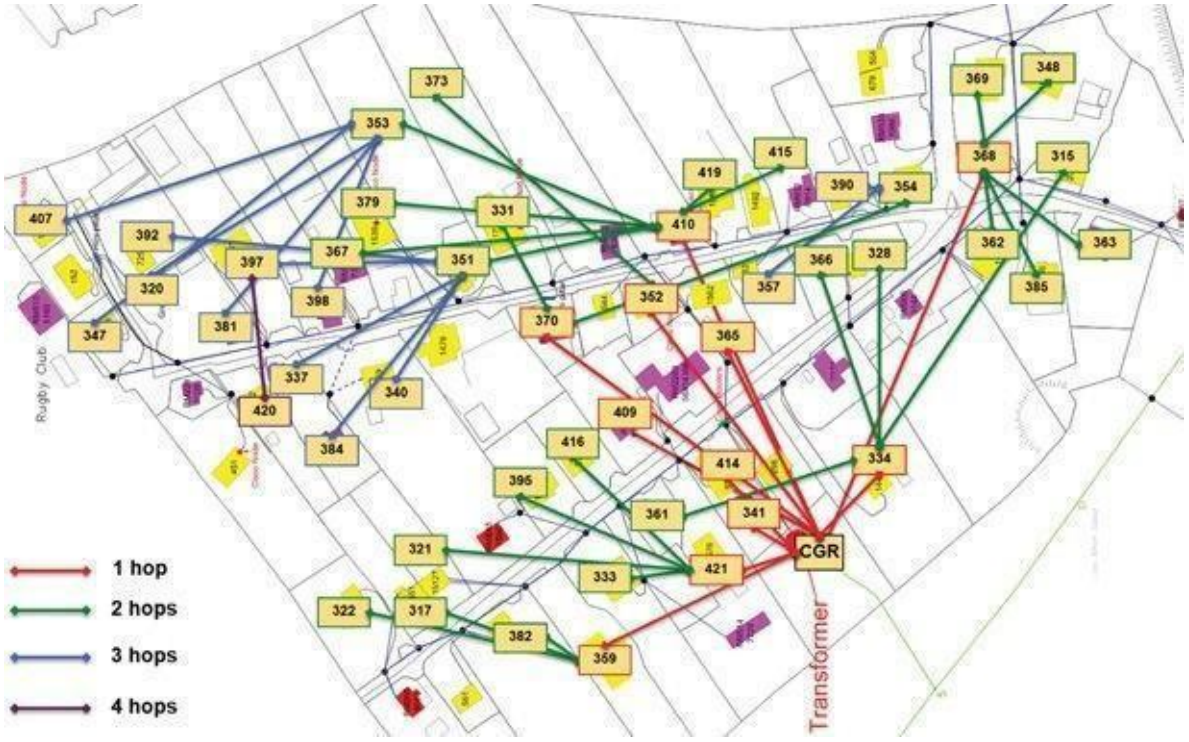



**Figure 4-13** *IPv6 Mesh in NB-PLC*

The IEEE 1901.2a standard offers the flexibility to run any upper-layer protocol. So, implementations of IPv6 6LoWPAN and RPL IPv6 protocols are supported. These protocols enable the use of network layer routing to create mesh networks over PLC. (For more information on 6LoWPAN and RPL, see Chapter 5.)

**Security**

IEEE 1901.2a security offers similar features to IEEE 802.15.4g. Encryption and authentication are performed using AES. In addition, IEEE 1901.2a aligns with 802.15.4g in its ability to support the IEEE 802.15.9 Key Management Protocol. However, some differences exist. These differences are mostly tied to the PHY layer fragmentation capabilities of IEEE 1901.2a and include the following:

- The Security Enabled bit in the Frame Control field should be set in all MAC frames carrying segments of an encrypted frame. (The Security Enabled bit is shown in Figure 4-8.)
- If data encryption is required, it should be done before packet segmentation. During packet encryption, the Segment Control field should not be included in the input to the encryption algorithm.
- On the receiver side, the data decryption is done after packet reassembly.
    - When security is enabled, the MAC payload is composed of the ciphered payload and the message integrity code (MIC) authentication tag for non-segmented payloads. If the payload is segmented, the MIC is part of the last packet (segment) only. The MIC authentication is computed using only

information from the MHR of the frame carrying the first segment.

**Competitive Technologies**

In the domain of NB-PLC, two technologies compete against IEEE 1901.2a: G3-PLC (now ITU G.9903) and PRIME (now ITU G.9904). Both of these technologies were initially developed to address a single use case: smart metering deployment in Europe over the CENELEC A band.

As mentioned previously, IEEE 1901.2a leverages portions of G3-PLC and PRIME, and it also competes with them. More specifically, G3-PLC is really close to IEEE 1901.2. The main differences include the fact that G3-PLC mandates data link layer protocol options for bootstrapping and allocating device addresses, and it is incompatible with IEEE 802.15.4g/e and an end-to- end IPv6 model. This means there is no information element support and no global IPv6 address support. PRIME is more like an ATM approach, with a Layer 7 protocol (that is, DLMS/COSEM) that runs directly on top of Layer
2. Adding IP support requires that Layer 3 protocols be added.

Following the IEEE 1901.2 working group efforts, new versions of G3-PLC and PRIME were published. These newer versions add a similar feature set, such as FCC and ARIB band support, ROBO for PRIME, and Super-ROBO and 16QAM for G3-PLC. As these competitive technologies continue to evolve and borrow from one another, it seems there might be a convergence toward compatibility at some point in the future.

**IEEE 1901.2a Conclusions**

IEEE 1901.2a is an open PHY and MAC standard approach to enable the use of Narrowband Power Line Communication. The set of use cases for this standard depends on and also benefits from the physical power lines that interconnect the devices.

The IEEE 1901.2a standard leverages the earlier standards G3-PLC (now ITU G.9903) and PRIME (now ITU G.9904). Supporting a wide range of frequencies at the PHY layer, IEEE 1901.2a also has a feature-rich MAC layer, based on 802.15.4. This flexibility in the MAC layer lends readily to the support of mesh topologies.

The HomePlug Alliance's Netricity program and the liaison agreement with the Wi-SUN Alliance provide industry support for IEEE 1901.2a by means of a profile definition and a certification program. However, IEEE 1901.2a faces competition from G3-PLC and PRIME as they are more established standards that continue to evolve.

**IEEE 802.11ah**

In unconstrained networks, IEEE 802.11 Wi-Fi is certainly the most successfully deployed wireless technology. This standard is a key IoT wireless access technology, either for connecting endpoints such as fog computing nodes, high-data-rate sensors,

and audio or video analytics devices or for deploying Wi-Fi backhaul infrastructures, such as outdoor Wi-Fi mesh in smart cities, oil and mining, or other environments. However, Wi-Fi lacks sub-GHz support for better signal penetration, low power for battery-powered nodes, and the ability to support a large number of devices. For these reasons, the IEEE 802.11 working group launched a task group named IEEE 802.11ah to specify a sub-GHz version of Wi-Fi. Three main use cases are identified for IEEE 802.11ah:

- **Sensors and meters covering a smart grid:** Meter to pole, environmental/agricultural monitoring, industrial process sensors, indoor healthcare system and fitness sensors, home and building automation sensors
- **Backhaul aggregation of industrial sensors and meter data:** Potentially connecting IEEE 802.15.4g sub networks
- **Extended range Wi-Fi:** For outdoor extended-range hotspot or cellular traffic offloading when distances already covered by IEEE 802.11a/b/g/n/ac are not good enough

**Standardization and Alliances**

In July 2010, the IEEE 802.11 working group decided to work on an ―industrial Wi-Fi‖ and created the IEEE 802.11ah group. The 802.11ah specification would operate in unlicensed sub-GHz frequency bands, similar to IEEE 802.15.4 and other LPWA technologies.

The industry organization that promotes Wi-Fi certifications and interoperability for 2.4 GHz and 5 GHz products is the Wi-Fi Alliance. The Wi-Fi Alliance is a similar body to the Wi-SUN Alliance. For more information on the Wi-Fi Alliance, see its webpage, at www.wi-fi.org

For the 802.11ah standard, the Wi-Fi Alliance defined a new brand called Wi- Fi HaLow. This marketing name is based on a play on words between ―11ah‖ in reverse and ―low power.‖ It is similar to the word ―hello‖ but it is pronounced ―hay-low.‖ The HaLow brand exclusively covers IEEE 802.11ah for sub-GHz device certification. You can think of Wi-Fi HaLow as a commercial designation for products incorporating IEEE 802.11ah technology. For more information on W-Fi HaLow, visit www.wi-fi.org/discover-wi-fi/wi-fi-halow.

**Physical Layer**

IEEE 802.11ah essentially provides an additional 802.11 physical layer operating in unlicensed sub-GHz bands. For example, various countries and regions use the following bands for IEEE 802.11ah: 868–868.6 MHz for EMEAR, 902–928 MHz and associated subsets for North America and Asia- Pacific regions, and 314–316 MHz, 430–434 MHz, 470–510 MHz, and 779– 787 MHz for China.

Based on OFDM modulation, IEEE 802.11ah uses channels of 2, 4, 8, or 16 MHz (and also 1 MHz for low-bandwidth transmission). This is one-tenth of the IEEE 802.11ac channels, resulting in one-tenth of the corresponding data rates of IEEE 802.11ac. The

IEEE 802.11ac standard is a high-speed wireless LAN protocol at the 5 GHz band that is capable of speeds up to 1 Gbps.

**MAC Layer**

The IEEE 802.11ah MAC layer is optimized to support the new sub-GHz Wi- Fi PHY while providing low power consumption and the ability to support a larger number of endpoints. Enhancements and features specified by IEEE 802.11ah for the MAC layer include the following:

- **Number of devices:** Has been scaled up to 8192 per access point.
- **MAC header:** Has been shortened to allow more efficient communication.
- **Null data packet (NDP) support:** Is extended to cover several control and management frames. Relevant information is concentrated in the PHY header and the additional overhead associated with decoding the MAC header and data payload is avoided. This change makes the control frame exchanges efficient and less power-consuming for the receiving stations.
- **Grouping and sectorization:** Enables an AP to use sector antennas and also group stations (distributing a group ID). In combination with RAW and TWT, this mechanism reduces contention in large cells with many clients by restricting which group, in which sector, can contend during which time window. (Sectors are described in more detail in the following section.)
- **Restricted access window (RAW):** Is a control algorithm that avoids simultaneous transmissions when many devices are present and provides fair access to the wireless network. By providing more efficient access to the medium, additional power savings for battery- powered devices can be achieved, and collisions are reduced.
- **Target wake time (TWT):** Reduces energy consumption by permitting an access point to define times when a device can access the network.
  This allows devices to enter a low-power state until their TWT time arrives.
  It also reduces the probability of collisions in large cells with many clients.
- **Speed frame exchange:** Enables an AP and endpoint to exchange frames during a reserved transmit opportunity (TXOP). This reduces contention on the medium, minimizes the number of frame exchanges to improve channel efficiency, and extends battery life by keeping awake times short.

**Topology**

While IEEE 802.11ah is deployed as a star topology, it includes a simple hops relay operation to extend its range. This relay option is not capped, but the IEEE 802.11ah task group worked on the assumption of two hops. It allows one 802.11ah device to act as an intermediary and relay data to another. In some ways, this is similar to a mesh, and it is important to note that the clients and not the access point handle the relay function.

This relay operation can be combined with a higher transmission rate or modulation and coding scheme (MCS). This means that a higher transmit rate is used by relay devices talking directly to the access point. The transmit rate reduces as you move further from the access point via relay clients. This ensures an efficient system that limits transmission speeds at the edge of the relays so that communications close to the AP are not negatively affected.

Sectorization is a technique that involves partitioning the coverage area into several sectors to get reduced contention within a certain sector. This technique is useful for limiting collisions in cells that have many clients. This technique is also often necessary when the coverage area of 802.11ah access points is large, and interference from neighboring access points is problematic. Sectorization uses an antenna array and beam-forming techniques to partition the cell-coverage area. Figure 4-14 shows an example of 802.11ah sectorization.

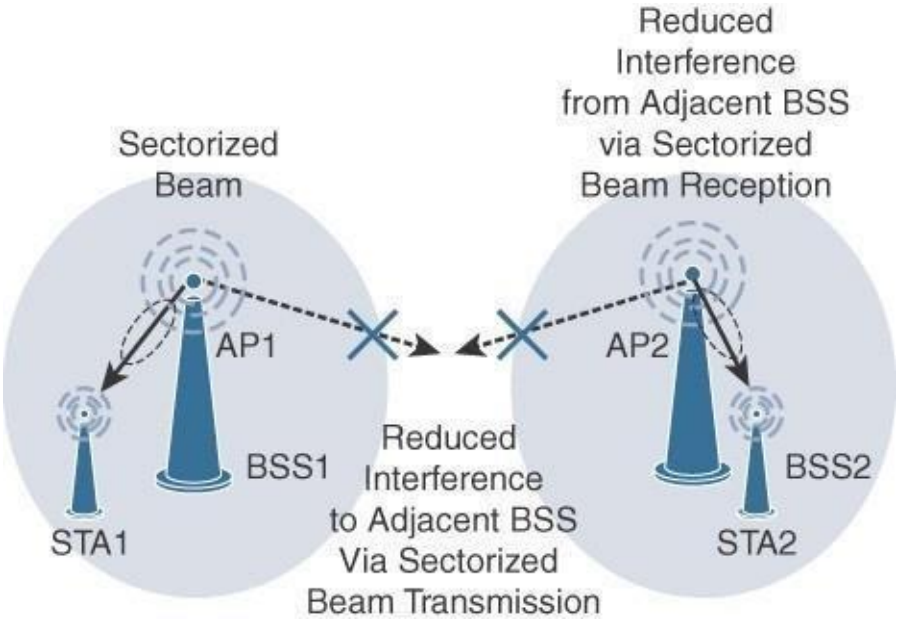



**Figure 4-14** *IEEE 802.**1** ah Sectorization*

**Security**

No additional security has been identified for IEEE 802.11ah compared to other IEEE

802.11 specifications. (The other IEEE protocols are discussed earlier in this chapter.) These protocols include IEEE 802.15.4, IEEE 802.15.4e, and IEEE 1901.2a, and the security information for them is also applicable to IEEE 802.11ah.

### Competitive Technologies

Competitive technologies to IEEE 802.11ah are IEEE 802.15.4 and IEEE 802.15.4e, along with the competitive technologies highlighted in each of their sections. (For more information on these competing technologies, see the sections ―IEEE 802.15.4‖ and ―IEEE 802.15.4g and IEEE 802.15.4e,‖ earlier in this chapter.)

### IEEE 802.11ah Conclusions

The IEEE 802.11ah access technology is an ongoing effort of the IEEE 802.11 working group to define an ―industrial Wi-Fi.‖ Currently, this standard is just at the beginning of its evolution, and it is not clear how the market will react to this new Wi-Fi standard.

This specification offers a longer range than traditional Wi-Fi technologies and provides good support for low-power devices that need to send smaller bursts of data at lower speeds. At the same time, it has the ability to scale to higher speeds as well.

IEEE 802.11ah is quite different in terms of current products and the existing Wi-Fi technologies in the 2.4 GHz and 5 GHz frequency bands. To gain broad adoption and compete against similar technologies in this space, it will need an ecosystem of products and solutions that can be configured and deployed at a low cost.

### LoRaWAN

In recent years, a new set of wireless technologies known as Low-Power Wide-Area (LPWA) has received a lot of attention from the industry and press. Particularly well adapted for long-range and battery-powered endpoints, LPWA technologies open new business opportunities to both services providers and enterprises considering IoT solutions. This section discusses an example of an unlicensed-band LPWA technology, known as LoRaWAN, and the next section, ―NB-IoT and Other LTE Variations,‖ reviews licensed-band alternatives from the 3rd Generation Partnership Project (3GPP).

### Standardization and Alliances

Initially, LoRa was a physical layer, or Layer 1, modulation that was developed by a French company named Cycleo. Later, Cycleo was acquired by Semtech. Optimized for long-range, two-way communications and low power consumption, the technology evolved from Layer 1 to a broader scope through the creation of the LoRa Alliance. For more information on the LoRa Alliance, visit www.lora-alliance.org.

The LoRa Alliance quickly achieved industry support and currently has hundreds of members. Published LoRaWAN specifications are open and can be accessed from the LoRa Alliance website.

Semtech LoRa as a Layer 1 PHY modulation technology is available through multiple chipset vendors. To differentiate from the physical layer modulation known as LoRa, the LoRa Alliance uses the term LoRaWAN to refer to its architecture and its specifications that describe end-to-end LoRaWAN communications and protocols.

Figure 4-15 provides a high-level overview of the LoRaWAN layers. In this figure, notice that Semtech is responsible for the PHY layer, while the LoRa Alliance handles the MAC layer and regional frequency bands.

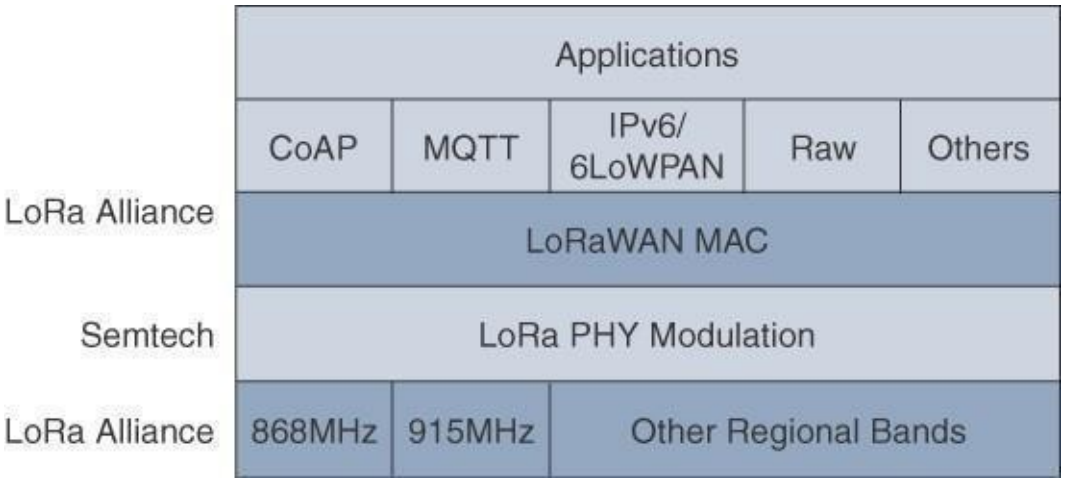



**Figure 4-15** *LoRaWAN Layers*

Overall, the LoRa Alliance owns and manages the roadmap and technical development of the LoRaWAN architecture and protocol. This alliance also handles the LoRaWAN endpoint certification program and technology promotion through its certification and marketing committees.

### Physical Layer

Semtech LoRa modulation is based on chirp spread spectrum modulation, which trades a lower data rate for receiver sensitivity to significantly increase the communication distance. In addition, it allows demodulation below the noise floor, offers robustness to noise and interference, and manages a single channel occupation by different spreading factors. This enables LoRa devices to receive on multiple channels in parallel.

LoRaWAN 1.0.2 regional specifications describe the use of the main unlicensed sub-GHz frequency bands of 433 MHz, 779–787 MHz, 863–870 MHz, and 902–928 MHz, as well as regional profiles for a subset of the 902– 928 MHz bandwidth. For example, Australia utilizes 915–928 MHz frequency bands, while South Korea uses 920–923 MHz and Japan uses 920– 928 MHz.

Understanding LoRa gateways is critical to understanding a LoRaWAN system. A LoRa gateway is deployed as the center hub of a star network architecture. It uses multiple transceivers and channels and can demodulate multiple channels at once or even demodulate multiple signals on the same channel simultaneously. LoRa gateways serve as a transparent bridge relaying data between endpoints, and the endpoints use a single-hop wireless connection to communicate with one or many gateways.

The data rate in LoRaWAN varies depending on the frequency bands and adaptive data rate (ADR). ADR is an algorithm that manages the data rate and radio signal for each endpoint. The ADR algorithm ensures that packets are delivered at the best data rate possible and that network performance is both optimal and scalable. Endpoints close to the gateways with good signal values transmit with the highest data rate, which enables a shorter transmission time over the wireless network, and the lowest transmit power. Meanwhile, endpoints at the edge of the link budget communicate at the lowest data rate and highest transmit power.

| Configuration | 863–870 MHz bps | 902–928 MHz bps |
|---|---|---|
| LoRa: SF12/125 kHz | 250 | N/A |
| LoRa: SF11/125 kHz | 440 | N/A |
| LoRa: SF10/125 kHz | 980 | 980 |
| LoRa: SF9/125 kHz | 1760 | 1760 |
| LoRa: SF8/125 kHz | 3125 | 3125 |
| LoRa: SF7/125 kHz | 5470 | 5470 |
| LoRa: SF7/250 kHz | 11,000 | N/A |
| FSK: 50 kbps | 50,000 | N/A |
| LoRa: SF12/500 kHz | N/A | 980 |
| LoRa: SF11/500 kHz | N/A | 1760 |
| LoRa: SF10/500 kHz | N/A | 3900 |
| LoRa: SF9/500 kHz | N/A | 7000 |
| LoRa: SF8/500 kHz | N/A | 12,500 |
| LoRa: SF7/500 kHz | N/A | 21,900 |

**Table 4-4** *LoRaWAN Data Rate Example*

In Table 4-4, notice the relationship between SF and data rate. For example, at an SF value of 12 for 125 kHz of channel bandwidth, the data rate is 250 bps. However, when

the SF is decreased to a value of 7, the data rate increases to 5470 bps.

### MAC Layer

As mentioned previously, the MAC layer is defined in the LoRaWAN specification. This layer takes advantage of the LoRa physical layer and classifies LoRaWAN endpoints to optimize their battery life and ensure downstream communications to the LoRaWAN endpoints. The LoRaWAN specification documents three classes of LoRaWAN devices:

- **Class A:** This class is the default implementation. Optimized for battery-powered nodes, it allows bidirectional communications, where a given node is able to receive downstream traffic after transmitting. Two receive windows are available after each transmission.
- **Class B:** This class was designated ―experimental‖ in LoRaWAN 1.0.1 until it can be better defined. A Class B node or endpoint should get additional receive windows compared to Class A, but gateways must be synchronized through a beaconing process.
- **Class C:** This class is particularly adapted for powered nodes. This classification enables a node to be continuously listening by keeping its receive window open when not transmitting.

LoRaWAN messages, either uplink or downlink, have a PHY payload composed of a 1-byte MAC header, a variable-byte MAC payload, and a MIC that is 4 bytes in length. The MAC payload size depends on the frequency band and the data rate, ranging from 59 to 230 bytes for the 863–870 MHz band and 19 to 250 bytes for the 902–928 MHz band. Figure 4-16 shows a high-level LoRaWAN MAC frame format.

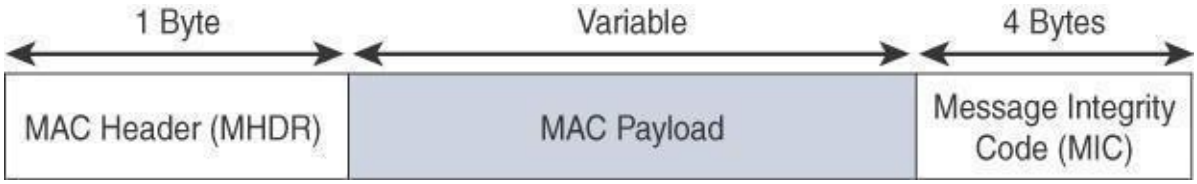



**Figure 4-16** *High-Level LoRaWAN MAC Frame Format*

In version 1.0.x, LoRaWAN utilizes six MAC message types. LoRaWAN devices use join request and join accept messages for over-the-air (OTA) activation and joining the network. The other message types are unconfirmed data up/down and confirmed data up/down. A ―confirmed‖ message is one that must be acknowledged, and ―unconfirmed‖ signifies that the end device does not need to acknowledge. ―up/down‖ is simply a directional notation identifying whether the message flows in the uplink or downlink path. Uplink messages are sent from endpoints to the network server and are relayed by one or more LoRaWAN gateways. Downlink messages flow from the network server to a single endpoint and are relayed by only a single gateway. Multicast

over LoRaWAN is being considered for future versions.

LoRaWAN endpoints are uniquely addressable through a variety of methods, including the following:

- An endpoint can have a global end device ID or DevEUI represented as an IEEE EUI-64 address.
- An endpoint can have a global application ID or AppEUI represented as an IEEE EUI-64 address that uniquely identifies the application provider, such as the owner, of the end device.
- In a LoRaWAN network, endpoints are also known by their end device address, known as a DevAddr, a 32-bit address. The 7 most significant bits are the network identifier (NwkID), which identifies the LoRaWAN network. The 25 least significant bits are used as the network address (NwkAddr) to identify the endpoint in the network.

**Topology**

LoRaWAN topology is often described as a ―star of stars‖ topology. As shown in Figure 4-17, the infrastructure consists of endpoints exchanging packets through gateways acting as bridges, with a central LoRaWAN network server. Gateways connect to the backend network using standard IP connections, and endpoints communicate directly with one or more gateways.

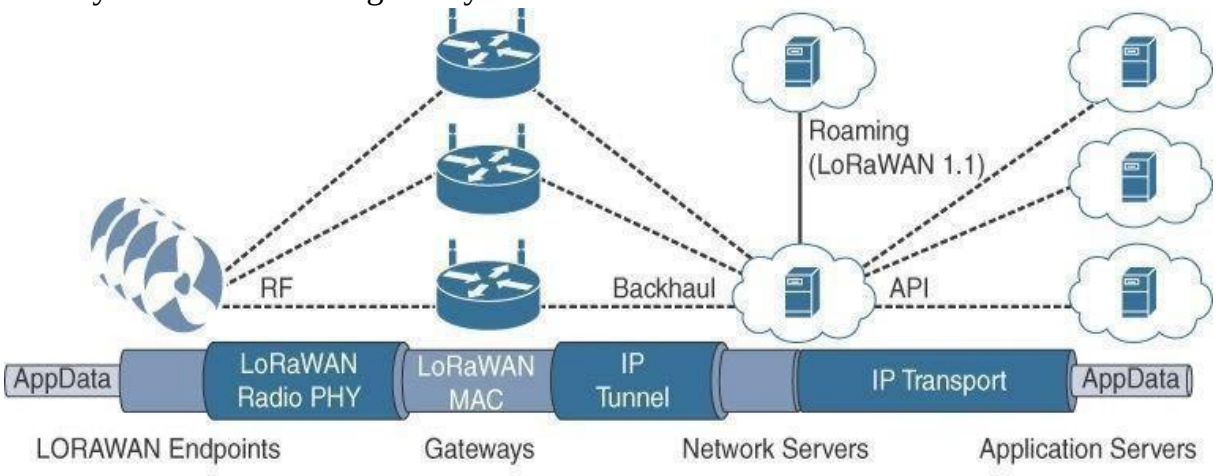



**Figure 4-17** *LoRaWAN Architecture*

In Figure 4-17, LoRaWAN endpoints transport their selected application data over the LoRaWAN MAC layer on top of one of the supported PHY layer frequency bands. The application data is contained in upper protocol layers. These upper layers are not the responsibility of the LoRa Alliance, but best practices may be developed and recommended. These upper layers could just be raw data on top of the LoRaWAN MAC layer, or the data could be stacked in multiple protocols. For example, you could

have upper-layer protocols, such as ZigBee Control Layer (ZCL), Constrained Application Protocol (CoAP), or Message Queuing Telemetry Transport (MQTT), with or without an IPv6/6LoWPAN layer. (The CoAP and MQTT protocols are covered in Chapter 6.)

Figure 4-17 also shows how LoRaWAN gateways act as bridges that relay between endpoints and the network servers. Multiple gateways can receive and transport the same packets. When duplicate packets are received, de- duplication is a function of the network server.The LoRaWAN network server manages the data rate and radio frequency (RF) of each endpoint through the adaptive data rate (ADR) algorithm.

ADR is a key component of the network scalability, performance, and battery life of the endpoints. The LoRaWAN network server forwards application data to the application servers, as depicted in Figure 4-17.

In future versions of the LoRaWAN specification, roaming capabilities between LoRaWAN network servers will be added. These capabilities will enable mobile endpoints to connect and roam between different LoRaWAN network infrastructures.

**Security**

Security in a LoRaWAN deployment applies to different components of the architecture, as detailed in Figure 4-18. LoRaWAN endpoints must implement two layers of security, protecting communications and data privacy across the network.

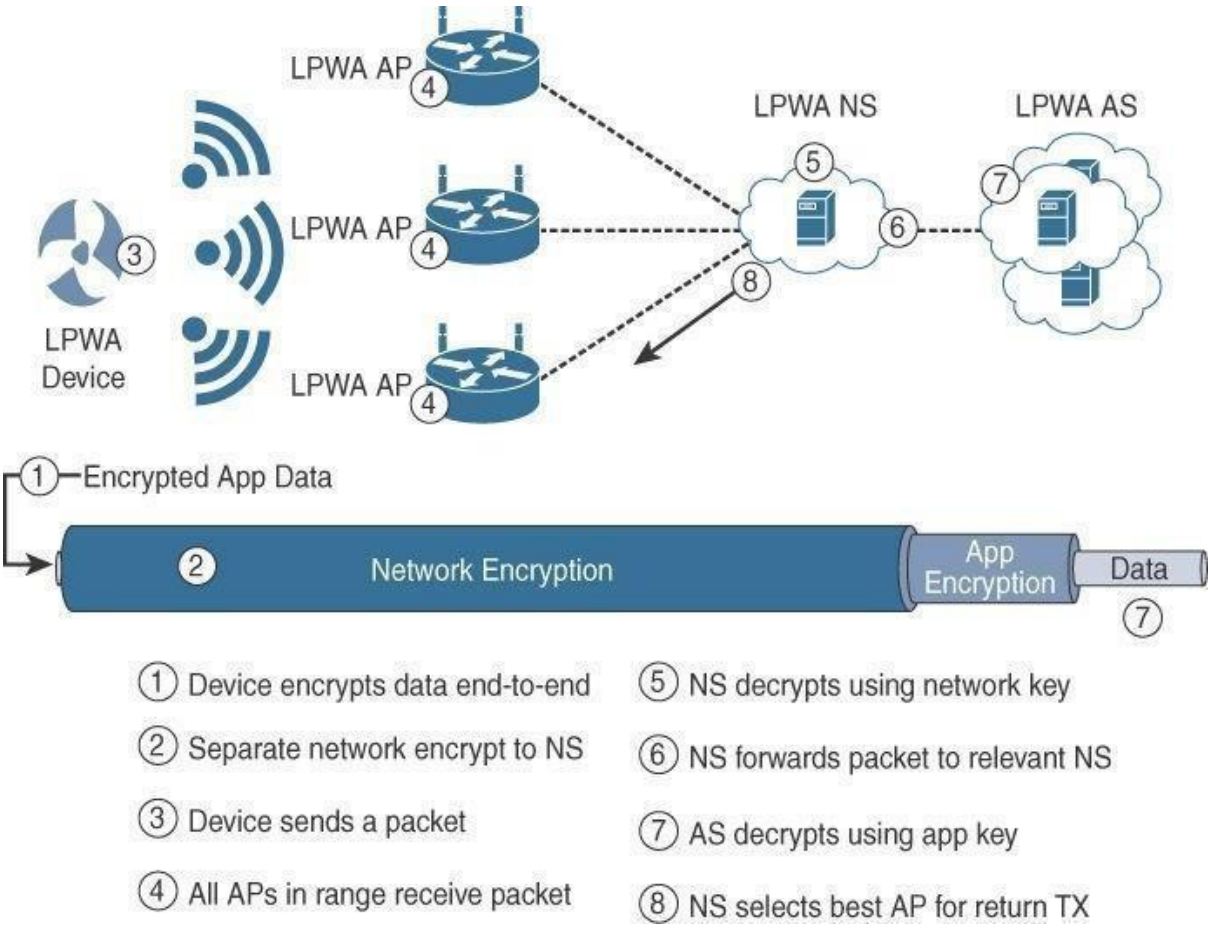



**Figure 4-18** *LoRaWAN Security*

The first layer, called ―network security‖ but applied at the MAC layer, guarantees the authentication of the endpoints by the LoRaWAN network server. Also, it protects LoRaWAN packets by performing encryption based on AES.

Each endpoint implements a network session key (NwkSKey), used by both itself and the LoRaWAN network server. The NwkSKey ensures data integrity through computing and checking the MIC of every data message as well as encrypting and decrypting MAC-only data message payloads.

The second layer is an application session key (AppSKey), which performs encryption and decryption functions between the endpoint and its application server. Furthermore, it computes and checks the application-level MIC, if included. This ensures that the LoRaWAN service provider does not have access to the application payload if it is not allowed that access.

Endpoints receive their AES-128 application key (AppKey) from the application owner. This key is most likely derived from an application- specific root key exclusively known to and under the control of the application provider.

For production deployments, it is expected that the LoRaWAN gateways are protected as well, for both the LoRaWAN traffic and the network management and operations over their backhaul link(s). This can be done using traditional VPN and IPsec technologies that demonstrate scaling in traditional IT deployments. Additional security add-ons are under evaluation by the LoRaWAN Alliance for future revisions of the specification.

LoRaWAN endpoints attached to a LoRaWAN network must get registered and authenticated. This can be achieved through one of the two join mechanisms:

- **Activation by personalization (ABP):** Endpoints don't need to run a join procedure as their individual details, including DevAddr and the NwkSKey and AppSKey session keys, are preconfigured and stored in the end device. This same information is registered in the LoRaWAN network server.
- **Over-the-air activation (OTAA):** Endpoints are allowed to dynamically join a particular LoRaWAN network after successfully going through a join procedure. The join procedure must be done every time a session context is renewed. During the join process, which involves the sending and receiving of MAC layer join request and join accept messages, the node establishes its credentials with a LoRaWAN network server, exchanging its globally unique DevEUI, AppEUI, and AppKey. The AppKey is then used to derive the session NwkSKey and AppSKey keys.

**Competitive Technologies**

| Characteristic | LoRaWAN | Sigfox | Ingenu Onramp |
|---|---|---|---|
| Frequency bands | 433 MHz, 868 MHz, 902–928 MHz | 433 MHz, 868 MHz, 902–928 MHz | 2.4 GHz |
| Modulation | Chirp spread spectrum | Ultra-narrowband | DSSS |
| Topology | Star of stars | Star | Star; tree supported with an RPMA extender |
| Data rate | 250 bps–50 kbps (868 MHz)<br>980 bps–21.9 kbps (915 MHz) | 100 bps (868 MHz)<br>600 bps (915 MHz) | 6 kbps |
| Adaptive data rate | Yes | No | No |
| Payload | 59–230 bytes (868 MHz)<br>19–250 bytes (915 MHz) | 12 bytes | 6 bytes–10 KB |
| Two-way communications | Yes | Partial | Yes |
| Geolocation | Yes (LoRa GW version 2 reference design) | No | No |
| Roaming | Yes (LoRaWAN 1.1) | No | Yes |
| Specifications | LoRA Alliance | Proprietary | Proprietary |

LPWA solutions and technologies are split between unlicensed and licensed bands. The licensed-band technologies are dedicated to mobile service providers that have acquired spectrum licenses; they are discussed in the next section. In addition, several technologies are targeting the unlicensed-band LPWA market to compete against LoRaWAN. The LPWA market is quickly evolving. Table 4-5 evaluates two of the best-established vendors known to provide LPWA options.

**Table 4-5** *Unlicensed LPWA Technology Comparison*

Table 4-5 gives you a good overview of two of the most established LoRaWAN competitors. This is a good starting point, but you should perform additional research to further differentiate these technologies if you are interested in deploying an LPWAN.

### LoRaWAN Conclusions

The LoRaWAN wireless technology was developed for LPWANs that are critical for

implementing many new devices on IoT networks. The term LoRa refers to the PHY layer, and LoRaWAN focuses on the architecture, the MAC layer, and a unified, single standard for seamless interoperability. LoRaWAN is managed by the LoRa Alliance, an industry organization.

The PHY and MAC layers allow LoRaWAN to cover longer distances with a data rate that can change depending on various factors. The LoRaWAN architecture depends on gateways to bridge endpoints to network servers. From a security perspective, LoRaWAN offers AES authentication and encryption at two separate layers.

**NB-IoT and Other LTE Variations**

Existing cellular technologies, such as GPRS, Edge, 3G, and 4G/LTE, are not particularly well adapted to battery-powered devices and small objects specifically developed for the Internet of Things. While industry players have been developing unlicensed-band LPWA technologies, 3GPP and associated vendors have been working on evolving cellular technologies to better address IoT requirements. The effort started with the definition of new LTE device categories. The aim was to both align with specific IoT requirements, such as low throughput and low power consumption, and decrease the complexity and cost of the LTE devices. This resulted in the definition of the LTE-M work item.

Because the new LTE-M device category was not sufficiently close to LPWA capabilities, in 2015 3GPP approved a proposal to standardize a new narrowband radio access technology called Narrowband IoT (NB-IoT). NB- IoT specifically addresses the requirements of a massive number of low- throughput devices, low device power consumption, improved indoor coverage, and optimized network architecture. The following sections review the proposed evolution of cellular technologies to better support the IoT opportunities by mobile service providers.

**Standardization and Alliances**

The 3GPP organization includes multiple working groups focused on many different aspects of telecommunications (for example, radio, core, terminal, and so on). Many service providers and vendors make up 3GPP, and the results of their collaborative work in these areas are the 3GPP specifications and studies. The workflow within 3GPP involves receiving contributions related to licensed LPWA work from the involved vendors. Then, depending on the access technology that is most closely aligned, such as 3G, LTE, or GSM, the IoT-related contribution is handled by either 3GPP or the GSM EDGE Radio Access Networks (GERAN) group.

Mobile vendors and service providers are not willing to lose leadership in this market of connecting IoT devices. Therefore, a couple intermediate steps have been pushed forward, leading to the final objectives set for NB-IoT and documented by 3GPP. At the same time, another industry group, the GSM Association (GSMA), has proposed the Mobile IoT Initiative, which ―is designed to accelerate the commercial availability of

LPWA solutions in licensed spectrum.‖ For more information on the Mobile IoT Initiative, go to www.gsma.com/connectedliving/mobile-iot-initiative/.

### LTE Cat 0

The first enhancements to better support IoT devices in 3GPP occurred in LTE Release 12. A new user equipment (UE) category, Category 0, was added, with devices running at a maximum data rate of 1 Mbps. Generally, LTE enhancements target higher bandwidth improvements. Category 0 includes important characteristics to be supported by both the network and end devices. Meanwhile, the UE still can operate in existing LTE systems with bandwidths up to 20 MHz. These Cat 0 characteristics include the following:

- **Power saving mode (PSM):** This new device status minimizes energy consumption. Energy consumption is expected to be lower with PSM than with existing idle mode. PSM is defined as being similar to ―powered off‖ mode, but the device stays registered with the network. By staying registered, the device avoids having to reattach or reestablish its network connection. The device negotiates with the network the idle time after which it will wake up. When it wakes up, it initiates a tracking area update (TAU), after which it stays available for a configured time and then switches back to sleep mode or PSM. A TAU is a procedure that an LTE device uses to let the network know its current tracking area, or the group of towers in the network from which it can be reached. Basically, with PSM, a device can be practically powered off but not lose its place in the network.
- **Half-duplex mode:** This mode reduces the cost and complexity of a device's implementation because a duplex filter is not needed. Most IoT endpoints are sensors that send low amounts of data that do not have a full-duplex communication requirement.

### LTE-M

Following LTE Cat 0, the next step in making the licensed spectrum more supportive of IoT devices was the introduction of the LTE-M category for 3GPP LTE Release 13. These are the main characteristics of the LTE-M category in Release 13:

- **Lower receiver bandwidth:** Bandwidth has been lowered to 1.4 MHz versus the usual 20 MHz. This further simplifies the LTE endpoint.
- **Lower data rate:** Data is around 200 kbps for LTE-M, compared to 1 Mbps for Cat 0.
- **Half-duplex mode:** Just as with Cat 0, LTE-M offers a half-duplex mode

that decreases node complexity and cost.

- **Enhanced discontinuous reception (eDRX):** This capability increases from seconds to minutes the amount of time an endpoint can ―sleep‖ between paging cycles. A paging cycle is a periodic check-in with the network. This extended ―sleep‖ time between paging cycles extends the battery lifetime for an endpoint significantly.

LTE-M requires new chipsets and additional software development. Commercial deployment is expected in 2017. Mobile carriers expect that only new LTE-M software will be required on the base stations, which will prevent re-investment in hardware.

**NB-IoT**

Recognizing that the definition of new LTE device categories was not sufficient to support LPWA IoT requirement, 3GPP specified Narrowband IoT (NB-IoT). The work on NB-IoT started with multiple proposals pushed by the involved vendors, including the following:

- Extended Coverage GSM (EC-GSM), Ericsson proposal
- Narrowband GSM (N-GSM), Nokia proposal Narrowband
- M2M (NB-M2M), Huawei/Neul proposal
- Narrowband OFDMA (orthogonal frequency-division multiple access), Qualcomm proposal
- Narrowband Cellular IoT (NB-CIoT), combined proposal of NB-M2M and NB-OFDMA
- Narrowband LTE (NB-LTE), Alcatel-Lucent, Ericsson, and Nokia
- proposal Cooperative Ultra Narrowband (C-UNB), Sigfox proposal

Consolidation occurred with the agreement to specify a single NB-IoT version based on orthogonal frequency-division multiple access (OFDMA) in the downlink and a couple options for the uplink. OFDMA is a modulation scheme in which individual users are assigned subsets of subcarrier frequencies. This enables multiple users to transmit low-speed data simultaneously. For more information on the uplink options, refer to the 3GPP specification TR 36.802.

Three modes of operation are applicable to NB-IoT:

- **Standalone:** A GSM carrier is used as an NB-IoT carrier, enabling reuse of 900 MHz or 1800 MHz.
- **In-band:** Part of an LTE carrier frequency band is allocated for use as an NB-IoT frequency. The service provider typically makes this allocation, and IoT devices are configured accordingly. You should be aware that if these devices must be deployed across different countries or regions using a different service

provider, problems may occur unless there is some coordination between the service providers, and the NB- IoT frequency band allocations are the same.

- **Guard band:** An NB-IoT carrier is between the LTE or WCDMA bands. This requires coexistence between LTE and NB-IoT bands.

Mobile service providers consider NB-IoT the target technology as it allows them to leverage their licensed spectrum to support LPWA use cases. For instance, NB-IoT is defined for a 200-kHz-wide channel in both uplink and downlink, allowing mobile service providers to optimize their spectrum, with a number of deployment options for GSM, WCDMA, and LTE spectrum, as shown in Figure 4-19.

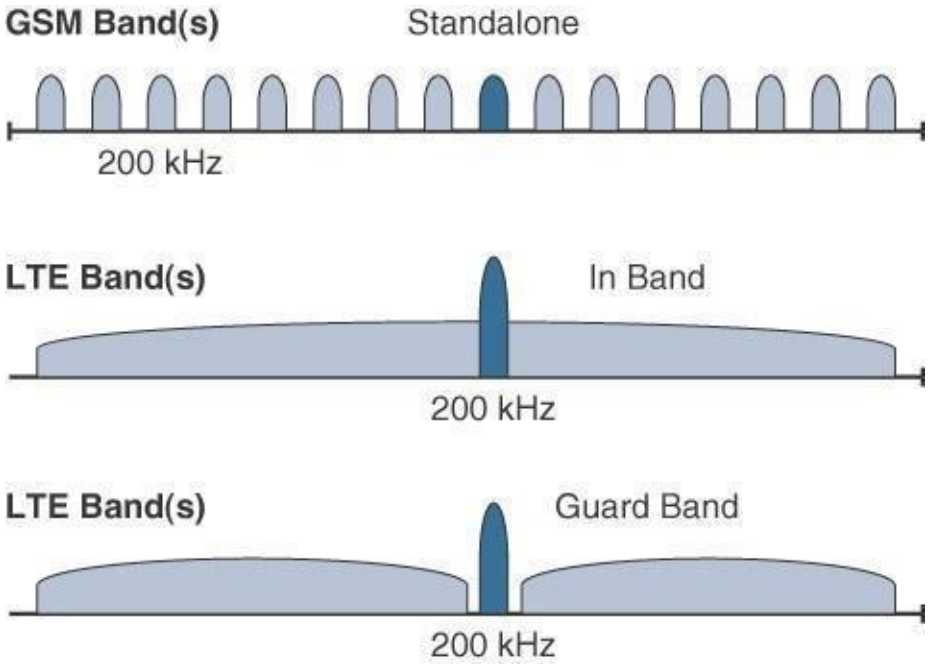



**Figure 4-19** *NB-IoT Deployment Options*

In an LTE network, resource blocks are defined with an effective bandwidth of 180 kHz, while on NB-IoT, tone or subcarriers replace the LTE resource blocks. The uplink channel can be 15 kHz or 3.75 kHz or multi-tone (n*15 kHz, n up to 12). At Layer 1, the maximum transport block size (TBS) for downlink is 680 bits, while uplink is 1000 bits. At Layer 2, the maximum Packet Data Convergence Protocol (PDCP) service data unit (SDU) size is 1600 bytes.

NB-IoT operates in half-duplex frequency-division duplexing (FDD) mode with a maximum data rate uplink of 60 kbps and downlink of 30 kbps.

### Topology

NB-IoT is defined with a link budget of 164 dB; compare this with the GPRS link budget of 144 dB, used by many machine-to-machine services. The additional 20 dB

link budget increase should guarantee better signal penetration in buildings and basements while achieving battery life requirements.

### Competitive Technologies

In licensed bands, it is expected that 3GPP NB-IoT will be the adopted LPWA technology when it is fully available. Competitive technologies are mostly the unlicensed-band LPWA technologies such as LoRaWAN. The main challenge faced by providers of the licensed bands is the opportunity for non-mobile service providers to grab market share by offering IoT infrastructure without buying expensive spectrum.

### NB-IoT and Other LTE Variations Conclusions

NB-IoT represents the future of LPWA technology for the mobile service providers who own licensed-band spectrum. IoT-related specifications must be completed and published by 3GPP to enable vendors, mobile service providers, and applications to successfully and widely endorse the technology. Evolution to eSIMs, which are still not widely supported, should be tied to NB-IoT as managing millions of SIM cards may not be an acceptable path for the market. An eSIM card is compliant across multiple operators and also reconfigurable. This means that it is a permanent part of the device and is easily rewritten if the device is switched to a different provider.

Technologies for connecting sensors. While various technologies are available for this purpose, many of them are in their infancy and will evolve over the years. This chapter provides a comprehensive look at the technologies that are the most promising going forward, based on current market trends, industry support, and market share. The technologies covered in the second part of this chapter included IEEE 802.15.4, IEEE 802.15.4g and IEEE 802.15.4e, IEEE 1901.2a, IEEE 802.11ah, LoRaWAN, and NB-IoT. You should have an awareness and base knowledge of these technologies, as they are fundamental to connecting IoT smart objects; in addition, understanding these technologies will provide a foundation for you to understand new technologies. Table 4-6 summarizes and compares some of the main characteristics of the access technologies discussed in this chapter.

| Characteristic | IEEE 802.15.4 | IEEE 802.15.4g and IEEE 802.15.4e | IEEE 1901.2a | IEEE 802.11ah | LoRaWAN | NB-IoT |
|---|---|---|---|---|---|---|
| Wired or wireless | Wireless | Wireless | Wired | Wireless | Wireless | Wireless |
| Frequency bands | Unlicensed 2.4 GHz and sub-GHz | Unlicensed 2.4 GHz and sub-GHz | Unlicensed CENELEC A and B, FCC, ARIB | Unlicensed sub-GHz | Unlicensed sub-GHz | Licensed |
| Topology | Star, mesh | Star, mesh | Mesh | Star | Star | Star |
| Range | Medium | Medium | Medium | Medium | Long | Long |
| Data rate | Low | Low | Low | Low–high | Low | Low |

**Table 4-6** *Main Characteristics of Access Technologies Discussed in This Chapter*

# MODULE - 3

# Chapter 5. 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 center servers either in the cloud or in locations that may be distributed or centralized. Dedicated applications are then run over virtualized or traditional operating systems or on network edge platforms (for example, fog computing). These lightweight applications communicate with the data center servers. Therefore, the system solutions combining various physical and data link layers call for an architectural approach with a common layer(s) independent from the lower (connectivity) and/or upper (application) layers. This is how and why the Internet Protocol (IP) suite started playing a key architectural role in the early 1990s. IP was not only preferred in the IT markets but also for the OT environment.

## 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. (For more information on IT and OT, refer to Chapter 1, ―What Is IoT?‖) An entire industrial workflow generally mandates smooth, incremental steps that evolve, with operations itself being the most time- and mission-critical factor for an organization.

One way to guarantee multi-year lifetimes is to define a layered architecture

such as the 30-year-old IP architecture. 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, such as the 3 billion Internet users.

Before evaluating the pros and cons of IP adoption versus adaptation, this section provides a quick review of the key advantages of the IP suite for the Internet of Things:

- **Open and standards-based:** Operational technologies have often been delivered as turnkey features by vendors who may have optimized the communications through closed and proprietary networking solutions. 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. While many 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. The IETF is an open standards body that focuses on the development of the Internet Protocol suite and related Internet technologies and protocols.

- **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. 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. 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 solution architecture and data flow.

- **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.

- **Scalable:** As the common protocol of the Internet, IP has been massively deployed and tested for robust scalability. Millions of private and public IP infrastructure nodes have been operational for years, offering strong foundations for those not familiar with IP network management. Of course, 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.

- **Manageable and highly secure:** Communications infrastructure requires appropriate management and security capabilities for proper operations. One of the benefits that comes from 30 years of operational IP networks is the well-understood network management and security protocols, mechanisms, and toolsets that are widely available. 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. However, you should be aware that despite the secure nature of IP, real challenges exist in this area. Specifically, the industry is challenged in securing constrained nodes, handling legacy OT protocols, and scaling operations.
- **Stable and resilient:** IP has been around for 30 years, and it is clear that IP is a workable solution. 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 have already transitioned from closed environments to open IP standards. Finally, 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 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.
- **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

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.

A similar transition is now occurring with IoT and its use of IP connectivity in the last mile. While IP is slowly becoming more prevalent, alternative protocol stacks are still often used. Let‘s look at a few examples in various industries to see how IP adaptation and adoption are currently applied to IoT last-mile connectivity.

In the industrial and manufacturing sector, there has been a move toward IP adoption. Solutions and product lifecycles in this space are spread over 10+ years, and many protocols have been developed for serial communications. While IP and Ethernet support were not specified in the initial versions, more

recent specifications for these serial communications protocols integrate Ethernet and IPv4.

Another example is a ZigBee solution that runs a non-IP stack between devices and a ZigBee gateway that forwards traffic to an application server. (For more information on ZigBee, see Chapter 4.) A ZigBee gateway often acts as a translator between the ZigBee and IP protocol stacks.

As highlighted by these examples, 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:

- **Bidirectional versus unidirectional data flow:** While bidirectional communications are generally expected, some last-mile technologies offer optimization for unidirectional communication. For example, as introduced in Chapter 4, different classes of IoT devices, as defined in RFC 7228, may only infrequently need to report a few bytes of data to an application. For these cases, it is not necessarily worth implementing a full IP stack. However, it requires the overall end-to-end architecture to solve potential drawbacks; for example, 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 implies a layered architecture with a per-packet overhead that 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. 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. Consequently, you need to decide whether the IP adoption model is necessary and, if it is, how it can be optimized. This same consideration applies to control plane traffic that is run over IP for low-bandwidth, last-mile links. Routing protocol and other verbose network services may either not be required or call for optimization.

- **Data flow model:** One benefit of the IP adoption model is the end-to-end nature of communications. 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.

- **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. This same restriction holds for ITU G.9903 G3-PLC nodes. Therefore, a deployment must consider which applications have to run on the gateway connecting these islands and the rest of the world. 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.

## The Need for Optimization

As discussed in the previous section, the Internet of Things will largely be built on the Internet Protocol suite. However, challenges still exist for IP in IoT solutions. In addition to coping with the integration of non-IP devices, you may need to deal with the limits at the device and network levels that IoT often imposes. Therefore, optimizations are needed at various layers of the IP stack to handle the restrictions that are present in IoT networks.

The following sections take a detailed look at why optimization is necessary for IP. Both the nodes and the network itself can often be constrained in IoT solutions. Also, IP is transitioning from version 4 to version 6, which can add further confinements in the IoT space.

## Constrained Nodes

As documented in Table 4-1 in Chapter 4, in IoT solutions, different classes of devices coexist. Depending on its functions in a network, a ―thing‖ architecture may or may not offer similar characteristics compared to a generic PC or server in 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. Current capabilities practically allow less than a year for these technologies on battery-powered nodes. Of course, power consumption is much less of a concern on nodes that do not require batteries as an energy source.

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.

While it has been largely demonstrated that production IP stacks perform well in constrained nodes, classification of these nodes helps when evaluating the IP adoption versus adaptation model selection. IoT constrained nodes can be classified as follows:

- **Devices that are very constrained in resources, may communicate infrequently 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

In the early years of the Internet, 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.

Constrained networks have unique characteristics and requirements. In contrast with typical IP networks, where highly stable and fast links are available, 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.

With a constrained network, in addition to 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 behaviors 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.

Unstable link layer environments create other challenges in terms of latency and control plane reactivity. One of the golden rules in a constrained network is to ―underreact to failure.‖ Due to the low bandwidth, a constrained network that overreacts can lead to a network collapse—which makes the existing problem worse.

Control plane traffic must also be kept at a minimum; otherwise, it consumes the bandwidth that is needed by the data traffic. Finally, you have to consider the power consumption in battery-powered nodes. Any failure or verbose control plane protocol may reduce the lifetime of the batteries.

## IP Versions

For 20+ years, the IETF has been working on transitioning the Internet from IP version 4 to IP version 6. 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 versions of IP run over the Internet, but most traffic is still IPv4 based.

While it may seem natural to base all IoT deployments on IPv6, you must take into account current infrastructures and their associated lifecycle of solutions, protocols, and products. IPv4 is entrenched in these current infrastructures, and so support for it is required in most cases. Therefore, the Internet of Things has to follow a similar path as the Internet itself and support both IPv4 and IPv6 versions concurrently. Techniques such as tunneling and translation need to be employed in IoT solutions to ensure interoperability between IPv4 and IPv6.

A variety of factors dictate whether IPv4, IPv6, or both can be used in an IoT solution. Most often these factors include a legacy protocol or technology that supports only IPv4. Newer technologies and protocols almost always support both IP versions. 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, as discussed in Chapter 6. So, there are no known production implementations by vendors of these protocols over IPv6 today. For IoT devices with application protocols defined by the IETF, such as HTTP/HTTPS, CoAP, MQTT, and XMPP, both IP versions are supported. (For more information on these IoT application layer protocols, see Chapter 6.) 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 tunneled over IPv4. On 4G/LTE networks, data services can use IPv4 or IPv6 as a base protocol, depending on the provider.

- **Serial Communications:** Many legacy devices in certain industries, such as manufacturing and utilities, 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. Encapsulation of serial protocols over IP leverages mechanisms such as raw socket TCP or UDP. While raw socket sessions can run over both IPv4 and IPv6, current implementations are mostly available for IPv4 only.

- **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.

(For more information on these physical and data link layers, see Chapter 4.) This means that any device implementing a technology that requires an IPv6 adaptation layer must communicate over an IPv6-only subnetwork.

### Optimizing IP for IoT

While the Internet Protocol is key for a successful Internet of Things, constrained nodes and constrained networks mandate optimization at various layers and on multiple protocols of the IP architecture. 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.

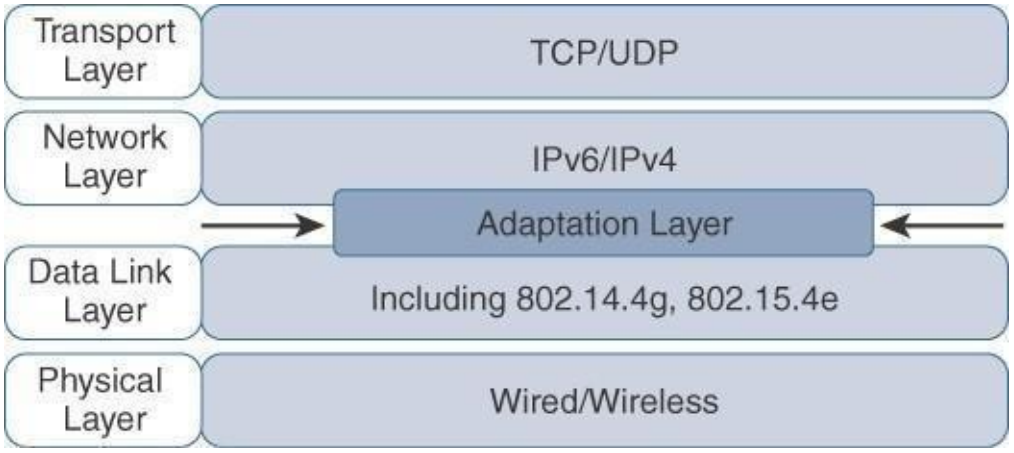



**Figure 5-1** *Optimizing IP for IoT Using an Adaptation Layer*

## 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*.

Unless the technology is proprietary, IP adaptation layers are typically defined by an IETF working group and released as a Request for Comments (RFC). An RFC is a publication from the IETF that officially documents Internet standards, specifications, protocols, procedures, and events. For example, RFC 864 describes how an IPv4 packet gets encapsulated over an Ethernet frame, and RFC 2464 describes how the same function is performed for an IPv6 packet.

IoT-related protocols follow a similar process. The main difference is that an adaptation layer designed for IoT may include some optimizations to deal with constrained nodes and networks. (See the sections ―Constrained Nodes‖

and ―Constrained Networks,‖ earlier in this chapter.)

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. (For more information on IEEE 802.15.4, see Chapter 4.) Figure 5-2 shows an example of an IoT protocol stack using the 6LoWPAN adaptation layer beside the well-known IP protocol stack for reference.

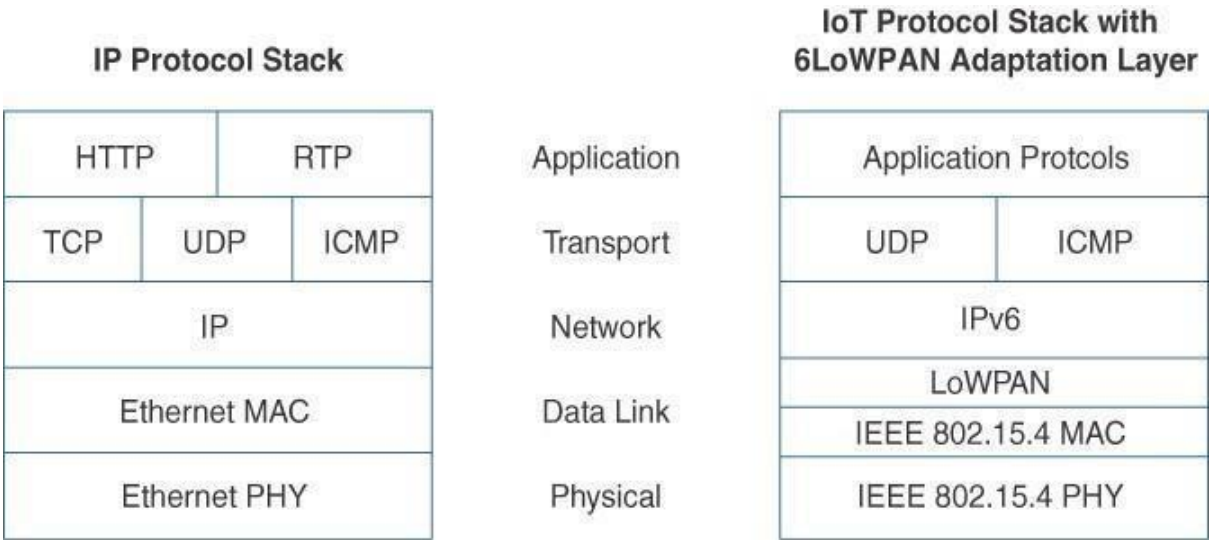



**Figure 5-2** *Comparison of an IoT Protocol Stack Utilizing 6LoWPAN and an IP Protocol Stack*

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.

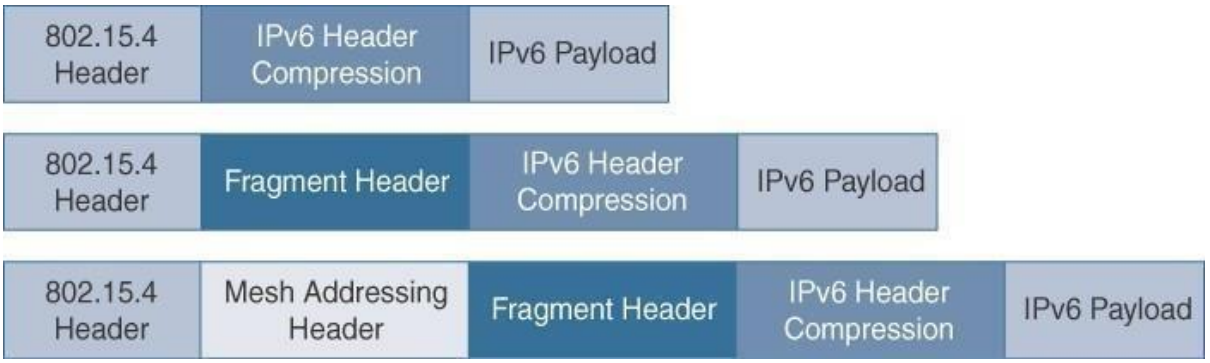



**Figure 5-3** *6LoWPAN Header Stacks*

Figure 5-3 shows the sub headers related to compression, fragmentation, and mesh addressing. You'll learn more about these capabilities in the following subsections.

## 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.

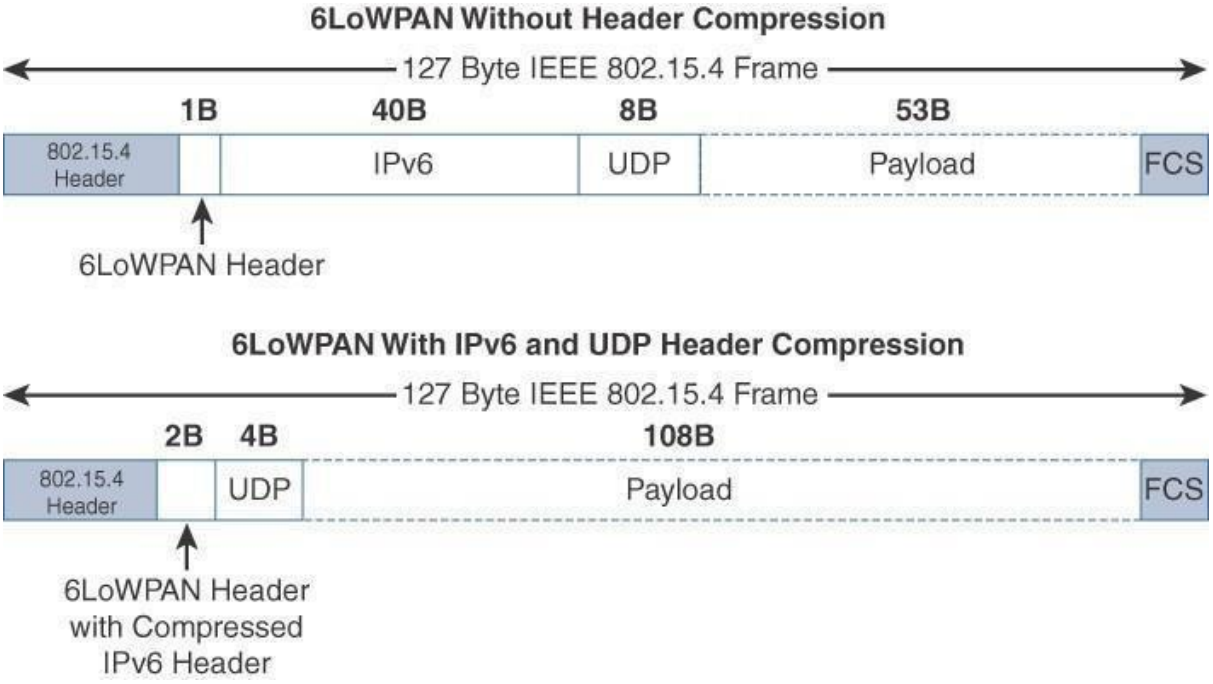



**Figure 5-4** *6LoWPAN Header Compression*

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.

The bottom half of Figure 5-4 shows a frame where header compression has been enabled for a best-case scenario. 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. 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.

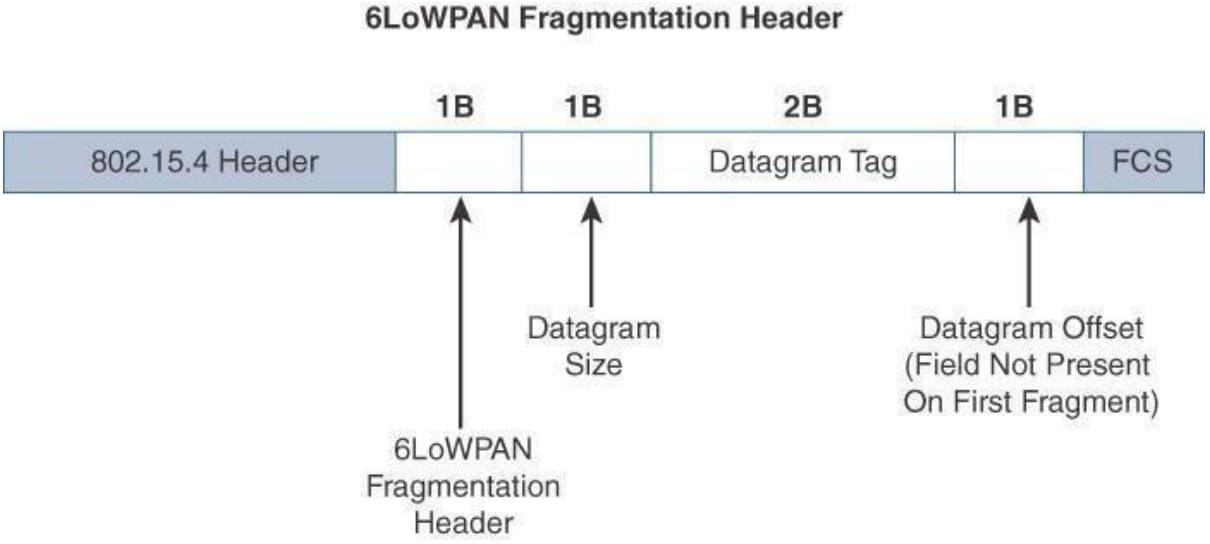



**Figure 5-5** *6LoWPAN Fragmentation Header*

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 packets over multiple hops. Three fields are defined for this header: Hop Limit, Source Address, and Destination Address. Analogous to the IPv6 hop limit field, the hop limit for mesh addressing also 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.

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

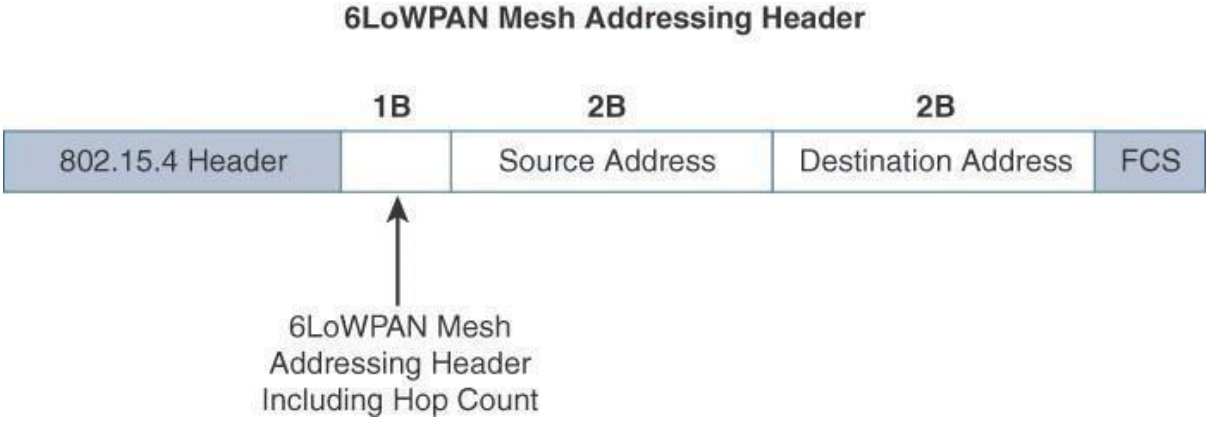



**Figure 5-6** *6LoWPAN Mesh Addressing Header*

Note that the mesh addressing header is used in a single IP subnet and is a Layer 2 type of routing known as mesh-under. The concept of mesh-under is discussed in the next section. Keep in mind that RFC 4944 only provisions the function in this case as the definition of Layer 2 mesh routing specifications was outside the scope of the 6LoWPAN working group, and the IETF doesn‘t define ―Layer 2 routing.‖ An implementation performing Layer 3 IP routing does not need to implement a mesh addressing header unless required by a given technology profile.

### Mesh-Under Versus Mesh-Over Routing

For network technologies such as IEEE 802.15.4, IEEE 802.15.4g, and IEEE 1901.2a that support mesh topologies and operate at the physical and data link layers, two main options exist for establishing reachability and forwarding packets. With the first option, mesh-under, the routing of packets is handled at the 6LoWPAN adaptation layer. The other option, known as ―mesh-over‖ or

―route-over,‖ utilizes IP routing for getting packets to their destination.

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. RPL is discussed in more detail later in this chapter.

## 6Lo Working Group

With the work of the 6LoWPAN working group completed, the 6Lo working group seeks to expand on this completed work with a focus on IPv6 connectivity over constrained-node networks. While the 6LoWPAN working group initially focused its optimizations on IEEE 802.15.4 LLNs, standardizing IPv6 over other link layer technologies is still needed.

Therefore, the charter of the 6Lo working group, now called the IPv6 over Networks of Resource-Constrained Nodes, is to facilitate the IPv6 connectivity over constrained-node networks. In particular, this working group is focused on the following:

- 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

Many proprietary wireless technologies have been developed and deployed in various industry verticals over the years. However, the publication of the IEEE 802.15.4 physical and data link layer specifications, followed by IEEE 802.15.4e amendments, has opened the path to standardized, deterministic communications over wireless networks.

IEEE 802.15.4e, Time-Slotted Channel Hopping (TSCH), is an add-on to the Media Access Control (MAC) portion of the IEEE 802.15.4 standard, with direct inheritance from other standards, such as WirelessHART and ISA100.11a.

Devices implementing IEEE 802.15.4e TSCH communicate by following a Time Division Multiple Access (TDMA) schedule. An allocation of a unit of bandwidth or time slot is scheduled between neighbor nodes. This allows the programming of predictable transmissions and enables deterministic, industrial-type applications. In comparison, other 802.15.4 implementations do not allocate slices of bandwidth, so communication, especially during times of contention, may be delayed or lost because it is always best effort.

To standardize IPv6 over the TSCH mode of IEEE 802.15.4e (known as 6TiSCH), the IETF formed the 6TiSCH working group. This working group works on the architecture, information model, and minimal 6TiSCH configuration, leveraging and enhancing work done by the 6LoWPAN working group, RoLL working group, and CoRE working group. The RoLL working group focuses on Layer 3 routing for constrained networks. The work of the RoLL working group is discussed in more detail in the upcoming section ―RPL.‖

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, but it does not mandate a scheduling algorithm for how the time slots are utilized. This is left to higher-level protocols like 6TiSCH. Scheduling is critical because it can affect throughput, latency, and power consumption. Figure 5-7 shows where 6top resides in relation to IEEE 802.15.4e, 6LoWPAN HC, and IPv6. 6LoWPAN HC is covered earlier in this chapter, in the section ―Header Compression.‖

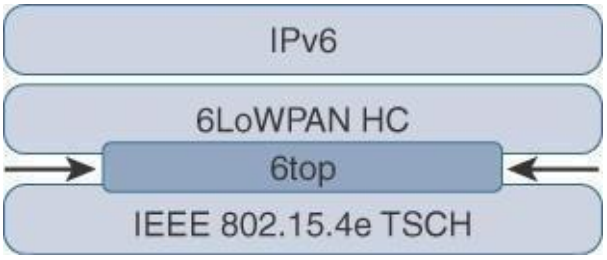



**Figure 5-7** *Location of 6TiSCH's 6top Sublayer*

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 when the schedule dictates that their cell is open for communication. 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. Static scheduling is a simple scheduling mechanism that can be used upon initial implementation or as a fallback in the case of network malfunction. The drawback with static scheduling is that nodes may expect a packet at any cell in the schedule. Therefore, energy is wasted idly listening across all cells.

- **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 and even CoAP in some scenarios. For more information on the application layer protocol CoAP, see Chapter 6. This scheduling mechanism provides quite a bit of flexibility and control in allocating cells for

communication between nodes.

- **Hop-by-hop scheduling:** A node reserves a path to a destination node multiple hops away by requesting the allocation of cells in a schedule at each intermediate node hop in the path. The protocol that is used by a node to trigger this scheduling mechanism is not defined at this point.

In addition to schedule management functions, the 6TiSCH architecture also defines three different forwarding models. Forwarding is the operation performed on each packet by a node that allows it to be delivered to a next hop or an upper-layer protocol. The forwarding decision is based on a preexisting state that was learned from a routing computation. 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. So, a frame received within a particular cell or cell bundle is switched to another cell or cell bundle. This forwarding occurs regardless of the network layer protocol.

- **Fragment forwarding (FF):** This model takes advantage of 6LoWPAN fragmentation to build a Layer 2 forwarding table. Fragmentation within the 6LoWPAN protocol is covered earlier in this chapter, in the section ―Fragmentation.‖ As you may recall, IPv6 packets can get fragmented at the 6LoWPAN sublayer to handle the differences between IEEE 802.15.4 payload size and IPv6 MTU. Additional headers for RPL source route information can further contribute to the need for fragmentation. However, with FF, a mechanism is defined where the first fragment is routed based on the IPv6 header present. The 6LoWPAN sublayer learns the next-hop selection of this first fragment, which is then applied to all subsequent fragments of that packet. Otherwise, IPv6 packets undergo hop-by-hop reassembly. This increases latency and can be power- and CPU-intensive for 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 a classification scheme for flows based on their priority, and RED is a common congestion avoidance mechanism.

For many IoT wireless networks, it is not necessary to be able to control the latency and throughput for sensor data. However, when some sort of

determinism is needed, 6TiSCH provides an open, IPv6-based standard solution for ensuring predictable communications over wireless sensor networks. However, its adoption by the industry is still an ongoing effort.

## RPL

The IETF chartered the RoLL (Routing over Low-Power and Lossy Networks) working group to evaluate all Layer 3 IP routing protocols and determine the needs and requirements for developing a routing solution for IP smart objects. After study of various use cases and a survey of existing protocols, the consensus was that a new routing protocol should be developed for use by IP smart objects, given the characteristics and requirements of constrained networks. This new distance-vector routing protocol was named the IPv6 Routing 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 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. No information from the MAC-layer header is needed to perform next-hop determination. Remember from earlier in this chapter that this is referred to as mesh-over routing.

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 toward the border router. This abbreviated routing table saves memory space and CPU. When communicating in non-storing mode, a node always forwards its packets to the border router, which knows how to ultimately reach the final destination.

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

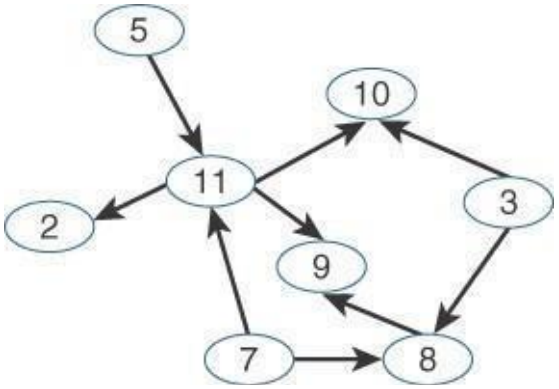



**Figure 5-8** *Example of a Directed Acyclic Graph (DAG)*

A basic RPL process involves building a destination-oriented directed acyclic graph (DODAG). A DODAG is a DAG rooted to one destination. In RPL, this destination occurs at a border router known as the DODAG root. Figure 5-9 compares a DAG and a DODAG. You can see that that a DAG has multiple roots, whereas the DODAG has just one.

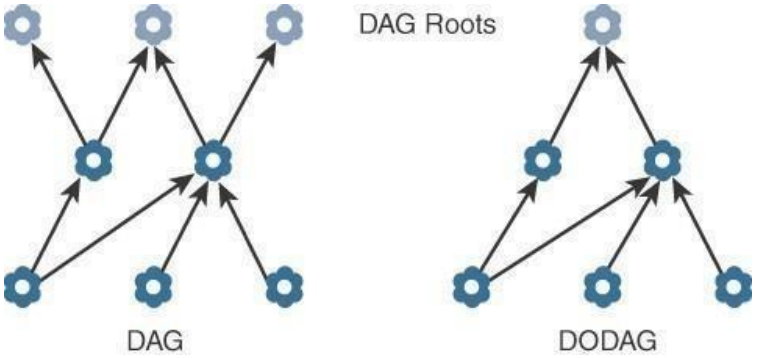



**Figure 5-9** *DAG and DODAG Comparison*

In a DODAG, each node maintains up to three parents that provide a path to the root. Typically, one of these parents is the preferred parent, which means it is the preferred next hop for upward routes toward the root.

The routing graph created by the set of DODAG parents across all nodes defines the full set of upward routes. RPL protocol implementation should ensure that routes are loop free by disallowing nodes from selected DODAG parents that are positioned further away from the border router.

Upward routes in RPL are discovered and configured using DAG Information Object (DIO) messages. Nodes listen to DIOs to handle changes in the topology that can affect routing. The information in DIO messages determines parents and the best path to the DODAG root.

Nodes establish downward routes by advertising their parent set toward the DODAG root using a Destination Advertisement Object (DAO) message. DAO messages allow nodes to inform their parents of their presence and

reachability to descendants.

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 the root stores the routing information. The root uses the information to then determine source routes needed for delivering IPv6 datagrams to individual nodes downstream in the mesh.

For storing mode, each node keeps track of the routing information that is advertised in the DAO messages. While this is more power- and CPU-intensive for each node, the benefit is that packets can take shorter paths between destinations in the mesh. The nodes can make their own routing decisions; in non-storing mode, on the other hand, all packets must go up to the root to get a route for moving downstream.

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

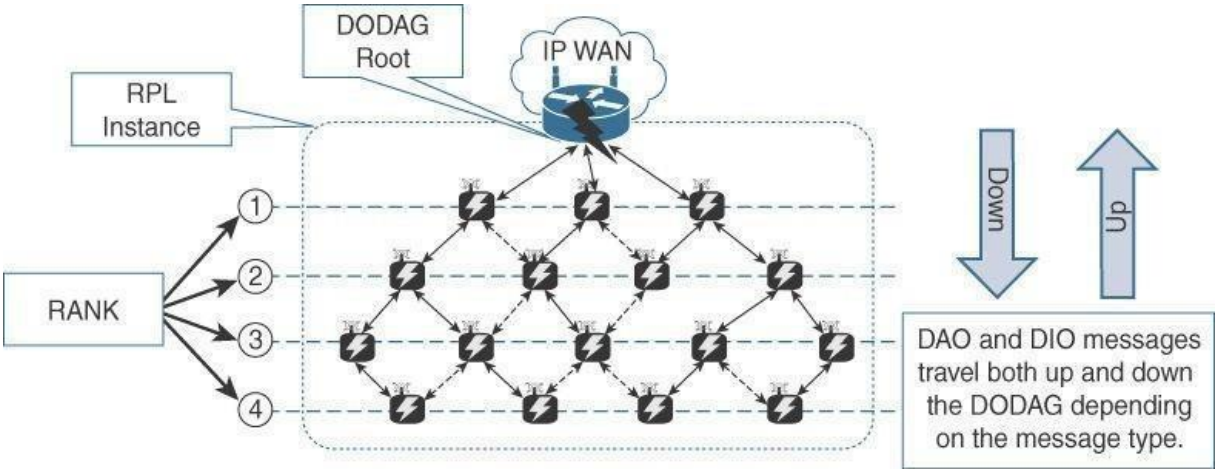



**Figure 5-10** *RPL Overview*

## Objective Function (OF)

An 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.

For example, nodes implementing an OF based on RFC 6719‘s Minimum Expected Number of Transmissions (METX) advertise the METX among their parents in DIO messages. Whenever a node establishes its rank, it simply sets the rank to the current minimum METX among its parents.

### 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. However, nodes may decrease their rank whenever they have established lower-cost routes. While the rank and routing metrics are closely related, the rank differs from routing metrics in that it is used as a constraint to prevent routing loops.

### RPL Headers

Specific network layer headers are defined for datagrams being forwarded within an RPL domain. One of the headers is standardized in RFC 6553, ―The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams,‖ and the other is discussed in RFC 6554, ―An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL).‖

RFC 6553 defines a new IPv6 option, known as the RPL option. The RPL option is carried in the IPv6 Hop-by-Hop header. The purpose of this header is to leverage data-plane packets for loop detection in a RPL instance. As discussed earlier, DODAGs only have single paths and should be loop free.

RFC 6554 specifies the Source Routing Header (SRH) for use between RPL routers. A border router or DODAG root inserts the SRH when specifying a source route to deliver datagrams to nodes downstream in the mesh network.

### Metrics

RPL defines a large and flexible set of new metrics and constraints for routing in RFC 6551. Developed to support powered and battery-powered nodes, RPL offers a far more complete set than any other routing protocol. Some of the RPL routing 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.

One of the constraints is ETX. ETX, which is described in RFC 6551, is defined earlier in this chapter. The other constraint, Relative Signal Strength Indicator (RSSI), specifies the power present in a received radio signal. Signals with low strength are generally less reliable and more susceptible to interference, resulting in packet loss.

In this scenario, a DODAG root and nodes form an IEEE 802.15.4 mesh. When a node finds a potential parent, it enters the neighbor into its routing table. However, it does not yet use the new neighbor for routing. Instead, the node must first establish that the link quality to its neighbor is sufficient for forwarding datagrams.

The node determines whether the link quality to a potential parent is sufficient by looking at its programmed constraints. In this example, the configured constraints are ETX and RSSI. If the RSSI in both directions exceeds a threshold and the ETX falls below a threshold, then the node confirms that the link quality to the potential parent is sufficient.

Once a node has determined that the link quality to a potential parent is sufficient, it adds the appropriate default route entry to its forwarding table. Maintaining RSSI and ETX for neighboring nodes is done at the link layer and stored in the link layer neighbor table.

The results from all link layer unicast traffic are fed into the RSSI and ETX computation for neighboring devices. If the link quality is not sufficient, then

the link is not added to the forwarding table and is therefore not used for routing packets.

To illustrate, Example 5-1 displays a simple RPL routing tree on a Cisco CGR-1000 router connecting an IEEE 802.15.4g mesh 6LoWPAN-based subnetwork. The first IPv6 address in this example, which ends in **1CC5**, identifies the DODAG root for the RPL tree. This DODAG root has branches to two nodes, indicated by the two IPv6 addresses ending in **924D** and **6C35**.

**Example 5-1 show wpan <interface> rpl tree** *Command from a Cisco CGR-1000*

```
pat1# show wpan 3/1 rpl tree

----------     WPAN     RPL     TREE     FIGURE     [3]     ---------
[2013:DB8:9999:8888:207:8108:B8:1CC5] (2)
--  2013:DB8:9999:8888:89C6:F7C9:D551:924D
-- 2013:DB8:9999:8888:95DF:2AD4:C1B1:6C35
RPL TREE: Num.DataEntries 2, Num.GraphNodes 3
```

RPL integration in a routing domain follows the same rules as more traditional IP routing protocols. Route redistribution, filtering, load balancing, and dynamic rerouting can be implemented the same way as other well-known protocols. For example, in IoT routers, you could see routes learned via RPL being redistributed into more well-known routing protocols, such as BGP and EIGRP.

In summary, RPL is a new routing protocol that enables an IPv6 standards-based solution to be deployed on a large scale while being operated in a similar way to today‘s IP infrastructures. RPL was designed to meet the requirements of constrained nodes and networks, and this has led to it becoming one of the main network layer IPv6-based routing protocols in IoT sensor networks.

## Authentication and Encryption on Constrained Nodes

IoT security is a complex topic that often spawns discussions and debates across the industry. While IoT security is the focus of Chapter 8, ―Securing IoT,‖ we have discussed constrained nodes and networks extensively in this chapter. So it is worth mentioning here the 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. (The CoAP protocol is covered in Chapter 6, and RFC 6437 defines the DTLS security protocol.) The ACE working group may investigate other security protocols later, with a particular focus on adapting whatever solution is chosen to HTTP and TLS.

The ACE working group expects to produce a standardized solution for authentication and authorization that enables authorized access (Get, Put, Post, Delete) to resources identified by a URI and hosted on a resource server in constrained environments. An unconstrained authorization server performs mediation of the access. Aligned with the initial focus, access to resources at a resource server by a client device occurs using CoAP and is protected by DTLS.

### 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. (See Chapter 6 for more details on CoAP.)

In constrained environments secured by DTLS, CoAP can be used to control resources on a device. (Constrained environments are network situations where constrained nodes and/or constrained networks are present. Constrained networks and constrained nodes are discussed earlier in this chapter, in the sections ―Constrained Nodes‖ and ―Constrained Networks.‖)

The DTLS in Constrained Environments (DICE) working group focuses on implementing the DTLS transport layer security protocol in these environments. The first task of the DICE working group is to define an optimized DTLS profile for constrained nodes. In addition, the DICE working group is considering the applicability of the DTLS record layer to secure multicast messages and investigating how the DTLS handshake in constrained environments can get optimized.

## Profiles and Compliances

As discussed throughout this chapter, leveraging the Internet Protocol suite for smart objects involves a collection of protocols and options that must work in coordination with lower and upper layers. Therefore, profile definitions, certifications, and promotion by alliances can help implementers

develop solutions that guarantee interoperability and/or interchangeability of devices.

This section introduces some of the main industry organizations working on profile definitions and certifications for IoT constrained nodes and networks. You can find various documents and promotions from these organizations in the IoT space, so it is worth being familiar with them and their goals.

## Internet Protocol for Smart Objects (IPSO) Alliance

Established in 2008, the Internet Protocol for Smart Objects (IPSO) Alliance has had its objective evolve over years. The alliance initially focused on promoting IP as the premier solution for smart objects communications.
Today, it is more focused on how to use IP, with the IPSO Alliance organizing interoperability tests between alliance members to validate that IP for smart objects can work together 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 IoT use cases and participates in educating the industry. As the IPSO Alliance declares in its value and mission statement, it wants to ensure that ―engineers and product builders will have access to the necessary tools for ‗how to build the IoT RIGHT.'‖ For more information on the IPSO Alliance, visit www.ipso-alliance.org.

## Wi-SUN Alliance

The Wi-SUN Alliance is an example of efforts from the industry to define a communication profile that applies to specific physical and data link layer protocols. Currently, Wi-SUN's main focus is on the IEEE 802.15.4g protocol and its support for multiservice and secure IPv6 communications with applications 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 provide resilient, secure, and cost-effective connectivity with extremely good coverage in a range of topographic environments, from dense urban neighborhoods to rural areas. (FANs are described in more detail in Chapter 11, ―Utilities.‖). You can read more about the Wi-SUN Alliance and its certification programs at the Wi-SUN Alliance website, www.wi-sun.org.

## Thread

A group of companies involved with smart object solutions for consumers created the Thread Group. This group has defined an IPv6-based wireless profile that provides the best way to connect more than 250 devices into a low-power, wireless mesh network. The wireless technology used by Thread is IEEE 802.15.4, which is different from Wi-SUN's IEEE 802.15.4g. Please see Chapter 4 for more information on 802.15.4 and 802.15.4g and their differences. For additional information on Thread and its specifications, visit http://threadgroup.org.

## IPv6 Ready Logo

Initially, the IPv6 Forum ensured the promotion of IPv6 around the world. Once IPv6 implementations became widely available, the need for interoperability and certification led to the creation of the IPv6 Ready Logo program.

The IPv6 Ready Logo program has established conformance and interoperability testing programs with the intent of increasing user confidence when implementing IPv6. The IPv6 Core and specific IPv6 components, such

as DHCP, IPsec, and customer edge router certifications, are in place. These certifications have industry-wide recognition, and many products are already certified. An IPv6 certification effort specific to IoT is currently under definition for the program.

# Chapter 6. Application Protocols for IoT

As with the wired and wireless access technologies discussed in Chapter 5, ―IP as the IoT Network Layer,‖ the IoT application protocols you select should be contingent on the use cases and vertical industries they apply to. In addition, IoT application protocols are dependent on the characteristics of the lower layers themselves. For example, application protocols that are sufficient for generic nodes and traditional networks often are not well suited for constrained nodes and networks.

This focuses on how higher-layer IoT protocols are transported. Specifically, this chapter includes the following sections:

- **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.

As in traditional networks, TCP or UDP are utilized in most cases when transporting IoT application data. The transport methods are covered in depth and form the bulk of the material in this chapter. You will notice that, as with the lower-layer IoT protocols, there are typically multiple options and solutions presented for transporting IoT application data. This is because IoT is still developing and maturing and has to account for the transport of not only new application protocols and technologies but legacy ones as well.

## The Transport Layer

This section reviews the selection of a protocol for the transport layer as supported by the TCP/IP architecture in the context of IoT networks. With the TCP/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 and destination before exchanging data. You can view it as an equivalent to a traditional telephone conversation, in which two phones must be connected and the communication link established before the parties can talk.

- **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.

With the predominance of human interactions over the Internet, TCP is the main protocol used at the transport layer. This is largely due to its inherent characteristics, such as 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.

In contrast, UDP is most often used in the context of network services, such as 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. In these cases, performance and latency are more important than packet retransmissions because re-sending a lost voice or video packet does not add value. When the reception of packets must be guaranteed error free, the application layer protocol takes care of that function.

IoT nodes may also be limited by the intrinsic characteristics of the data link layers. For example, low-power and lossy networks (LLNs), as discussed in Chapter 5, may not cope well with supporting large numbers of TCP sessions.

This may explain why a new IoT application protocol, such as Constrained Application Protocol (CoAP), almost always uses UDP and why implementations of industrial application layer protocols may call for the optimization and adoption of the UDP transport layer if run over LLNs. For example, the Device Language Message Specification/Companion Specification for Energy Metering (DLMS/COSEM) application layer protocol, a popular protocol for reading smart meters in the utilities space, is the de facto standard in Europe. Adjustments or optimizations to this protocol should be made depending on the IoT transport protocols that are present in the lower layers. For example, 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.

To guarantee interoperability, certification and compliance profiles, such as Wi-SUN, need to specify the stack from Layer 1 to Layer 4. This enables the chosen technology to be compatible with the different options of the stack while also being compatible with IP. (Chapter 4, ―Connecting Smart Objects,‖ provides more information on Wi-SUN.

## IoT Application Transport Methods

Because of the diverse types of IoT application protocols, there are various means for transporting these protocols across a network. Sometimes you may be dealing with legacy utility and industrial IoT protocols that have certain requirements, while other times you might need to consider the transport requirements of more modern application layer protocols. To make these decisions easier, it makes sense to categorize the common IoT application protocols and then focus on the transport methods available for each category. 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

As introduced in Chapter 4, IETF RFC 7228 devices defined as class 0 send or receive only a few bytes of data. For myriad reasons, such as processing capability, power constraints, and cost, these devices do not implement a fully structured network protocol stack, such as IP, TCP, or UDP, or even an application layer protocol. Class 0 devices are usually simple smart objects that are severely constrained. Implementing a robust protocol stack is usually not useful and sometimes not even possible with the limited available resources.

For example, consider low-cost temperature and relative humidity (RH) sensors sending data over an LPWA LoRaWAN infrastructure. (LPWA and LoRaWAN are discussed in Chapter 4.) Temperature is represented as 2 bytes and RH as another 2 bytes of data. Therefore, this small data payload is directly transported on top of the LoRaWAN MAC layer, without the use of TCP/IP. Example 6-1 shows the raw data for temperature and relative humidity and how it can be decoded by the application.

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

```
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/16-24to 100RH = 0x062e/16-24 = 74.9     to RH = 74.9%
```

While many constrained devices, such as sensors and actuators, have adopted deployments that have no application layer, this transportation method has not been standardized. This lack of standardization makes it difficult for generic implementations of this transport method to be successful from an interoperability perspective.

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. The solution to this problem is to use an IoT data broker, as detailed in Figure 6-1. 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. (The concept of the IoT data broker is introduced in Chapter 1, ―What Is IoT?‖)

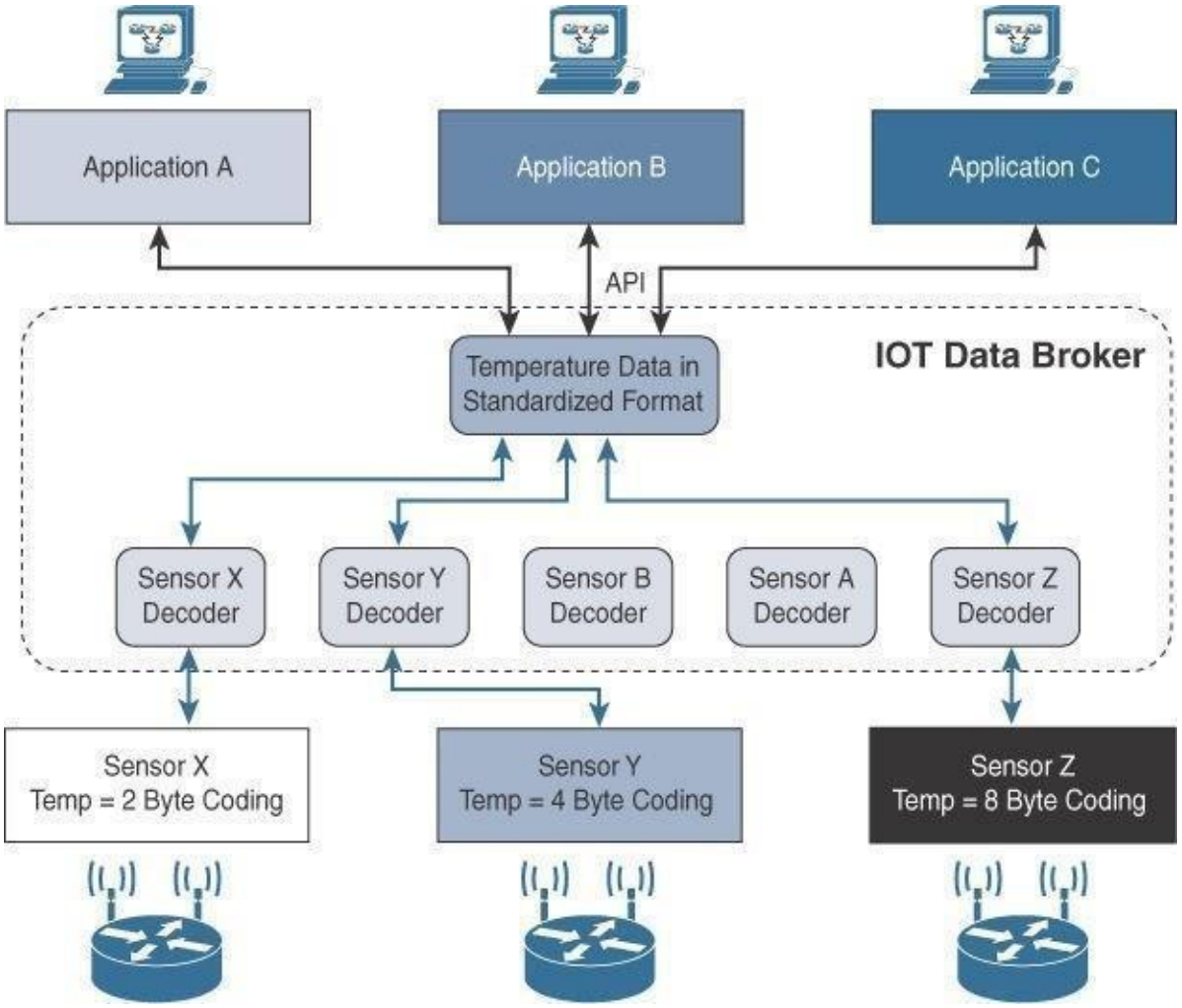



**Figure 6-1** *IoT Data Broker*

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.

You should note that IoT data brokers are also utilized from a commercial perspective to distribute and sell IoT data to third parties. Companies can provide access to their data broker from another company‘s application for a fee. This makes an IoT data broker a possible revenue stream, depending on the value of the data it contains.

## 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.

As mentioned previously, these protocols go back decades and are serial based. So, transporting them over current IoT and traditional networks requires that certain accommodations be made from both protocol and implementation perspectives. These accommodations and other adjustments form various SCADA transport methods that are the focus of upcoming sections.

## Adapting SCADA for IP

In 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. Other protocol user groups also slightly modified their protocols to run over an IP infrastructure. Benefits of this move to Ethernet and IP include the ability to leverage existing equipment and standards while integrating seamlessly the SCADA subnetworks to the corporate WAN infrastructures.

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 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, These legacy serial protocols have adapted and evolved to utilize IP and TCP/UDP as both networking and transport mechanisms.

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.

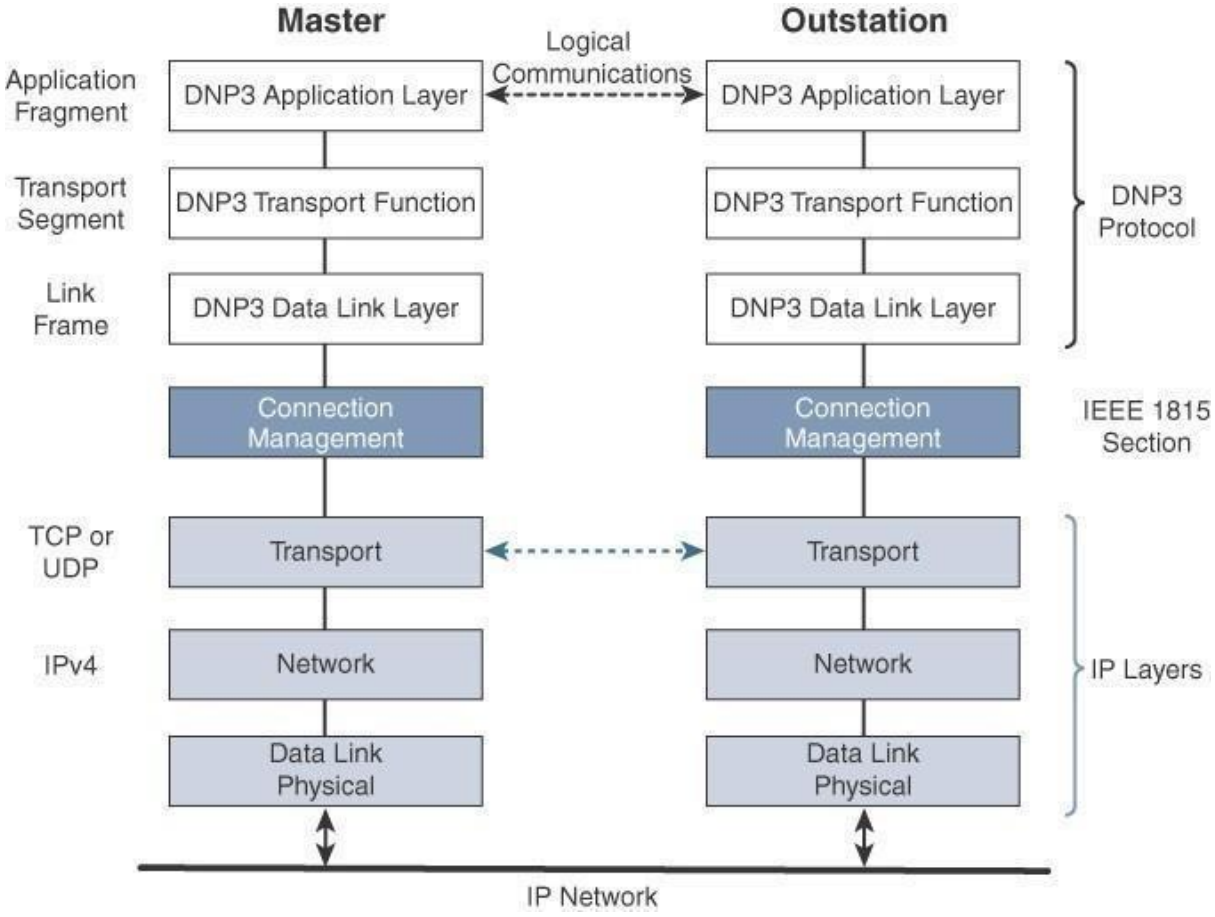



**Figure 6-2** *Protocol Stack for Transporting Serial DNP3 SCADA over IP*

In Figure 6-2, 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 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 keepalive timer monitors the status of the connection. Keepalive messages are implemented as DNP3 data link layer status requests. If a response is not received to a keepalive message, the connection is deemed broken, and the appropriate action is taken.

## Tunneling Legacy SCADA over IP Networks

Deployments of legacy industrial protocols, such as DNP3 and other SCADA protocols, in modern IP networks call for flexibility when integrating several generations of devices or operations that are tied to various releases and versions of application servers. Native support for IP can vary and may require different solutions. Ideally, end-to-end native IP support is preferred,

using a solution like IEEE 1815-2012 in the case of DNP3. Otherwise, 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 that is used to access network devices over an IP network. More modern industrial application servers may support this capability, while older versions typically require another device or piece of software to handle the transition from pure serial data to serial over IP using a raw socket. Figure 6-3 details raw socket scenarios for a legacy SCADA server trying to communicate with remote serial devices.

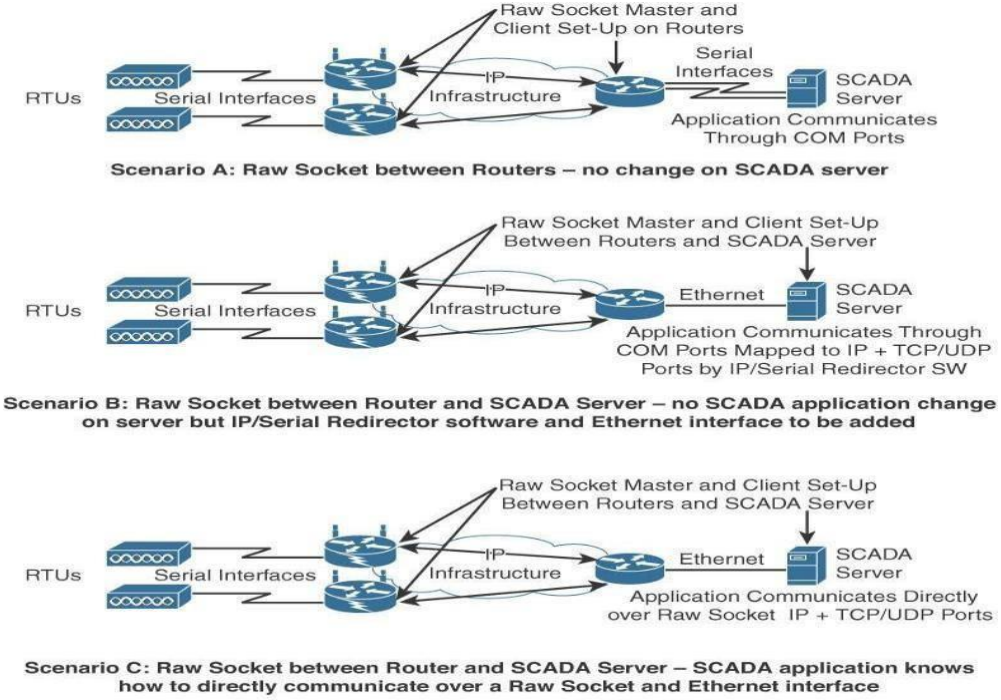



**Figure 6-3** *Raw Socket TCP or UDP Scenarios for Legacy Industrial Serial Protocols*

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 reality, other legacy industrial application servers could be shown here as well.

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 to transport the serial payload over the IP network.

Scenario B has a small change on the SCADA server side. A piece of software is installed 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. Unlike in Scenarios A and B, where a router or IP/serial redirector software has to map the SCADA server‘s serial ports to IP ports, in Scenario C the SCADA server has full IP support for raw socket connections.

## SCADA Protocol Translation

As mentioned earlier, an alternative to a raw socket connection for transporting legacy serial data across an IP network is protocol translation. With protocol translation, the legacy serial protocol is translated to a corresponding IP version. 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, despite the fact that a serial connection is present on one side and an IP connection is used on the other.

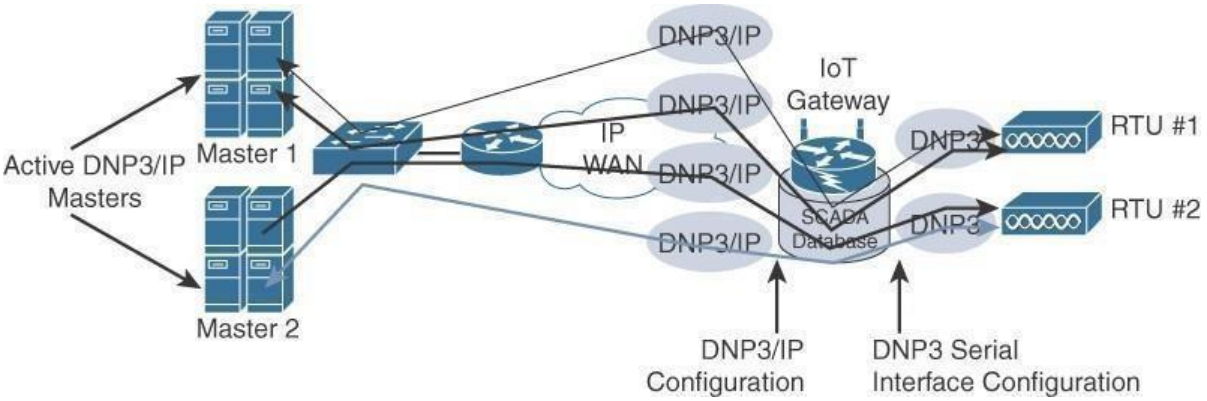



**Figure 6-4** *DNP3 Protocol Translation*

By running protocol translation, the IoT gateway connected to the RTUs in Figure 6-4 is implementing a computing function close to the edge of the network. 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, as shown in this protocol translation example. Alternatively, this can also be performed directly on a node connecting multiple sensors. In either case, this is referred to as fog computing. (For more information on fog computing, see Chapter 2, ―IoT Network Architecture and Design.‖)

## SCADA Transport over LLNs with MAP-T

Due to the constrained nature of LLNs, the implementation of industrial protocols should at a minimum be done over UDP. This in turn requires that both the application servers and devices support and implement UDP. While the long-term evolution of SCADA and other legacy industrial protocols is to natively support IPv6, it must be highlighted that most, if not all, of the industrial devices supporting IP today support IPv4 only. When deployed over LLN sub networks that are IPv6 only, a transition mechanism, such as MAP-T (Mapping of Address and Port using Translation, RFC 7599), needs to be implemented. This allows the deployment to take advantage of native IPv6 transport transparently to the application and devices.

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.

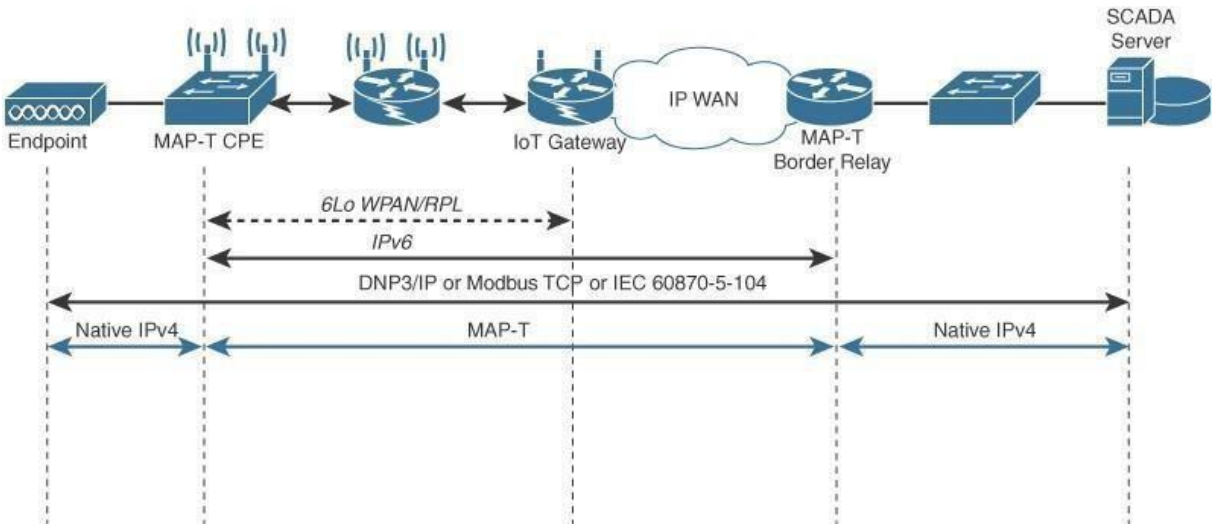



**Figure 6-5** *DNP3 Protocol over 6LoWPAN Networks with MAP-T*

The solution to this problem is to use the protocol known as MAP-T,

introduced in Chapter 5. MAP-T makes the appropriate mappings between IPv4 and the IPv6 protocols. This allows legacy IPv4 traffic to be forwarded across IPv6 networks. In other words, older devices and protocols can continue running IPv4 even though the network is requiring IPv6.

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.

Legacy implementations of SCADA and other industrial protocols are still widely deployed across many industries. While legacy SCADA has evolved from older serial connections to support IP, we can still expect to see mixed deployments for many years. To address this challenge, OT networks require mechanisms such as raw sockets and protocol translation to transport legacy versions over modern IP networks. Even when the legacy devices have IPv4 capability, the constrained portions of the network often require IPv6, not IPv4. In these cases, a MAP-T solution can be put in place to enable IPv4 data to be carried across an IPv6 network.

## Generic Web-Based Protocols

Over the years, 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.

Once again, the definition of constrained nodes and networks must be analyzed to select the most appropriate protocol. (Constrained nodes and networks are discussed in Chapter 5.) 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.

The HTTP/HTTPS client/server model serves as the foundation for the World Wide Web. Recent evolutions of embedded web server software with advanced features are now implemented with very little memory (in the range of tens of kilobytes in some cases). This enables the use of embedded web services software on some constrained devices.

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. IoT devices that only push data to an application (for example, an Ethernet- or Wi-Fi-based weather station reporting data to a weather map application or a Wi-Fi–enabled body weight scale that sends data to a health application) may need to implement web services on the client side. The HTTP client side only initiates connections and does not accept incoming ones.

On the other hand, some IoT devices, such as a video surveillance camera, may have web services implemented on the server side. However, because these devices often have limited resources, the number of incoming connections must be kept low. In addition, advanced development in data modeling should be considered as a way to shift the workload from devices to clients, including web browsers on PCs, mobile phones, tablets, and cloud applications.

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). (For more information on XMPP-IoT, see www.xmpp-iot.org.)

## 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 6-6 highlights their position in a common IoT protocol stack.

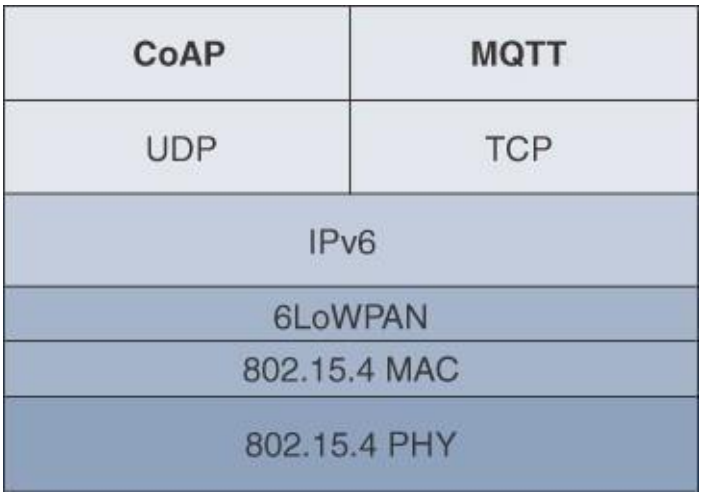



**Figure 6-6** *Example of a High-Level IoT Protocol Stack for CoAP and MQTT*

In Figure 6-6, 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, see https://datatracker.ietf.org/wg/core/charter/.) Constrained nodes and networks are discussed in Chapter 5.

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)

The CoAP messaging model is primarily designed to facilitate the exchange of messages over UDP between endpoints, including the secure transport protocol Datagram Transport Layer Security (DTLS). (UDP is discussed earlier in this chapter.) The IETF CoRE working group is studying alternate transport mechanisms, including TCP, secure TLS, and WebSocket. CoAP over Short Message Service (SMS) as defined in Open Mobile Alliance for Lightweight Machine-to-Machine (LWM2M) for IoT device management is also being considered. (For more information on the Open Mobile Alliance, see http://openmobilealliance.org.)

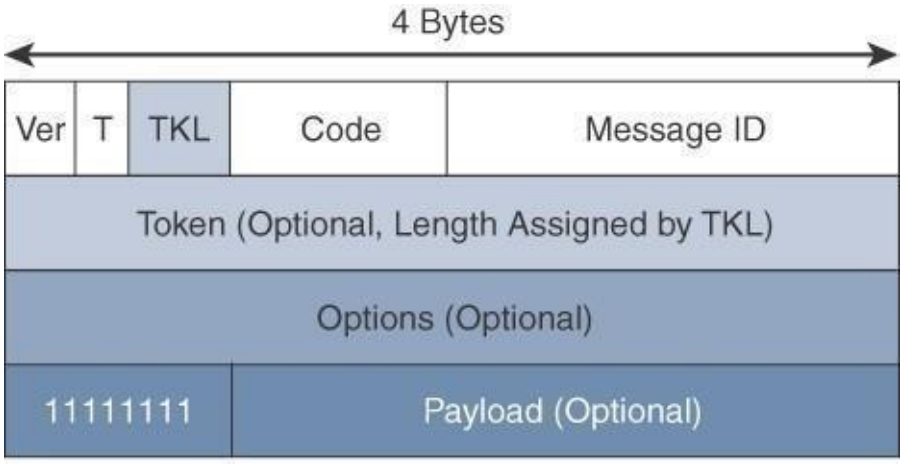



**Figure 6-7** *CoAP Message Format*

As you can see in Figure 6-7, 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 to parse and process for constrained devices. Table 6-1 provides an overview of the various fields of a CoAP message.

| 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. However, it is recommended that the message fit within a single IP packet and UDP payload to avoid fragmentation. For IPv6, with the default MTU size being 1280 bytes and allowing for no fragmentation across nodes, the maximum CoAP message size could be up to 1152 bytes, including 1024 bytes for the payload. In the case of IPv4, as IP fragmentation may exist across the network, implementations should limit themselves to more conservative values and set the IPv4 Don‘t Fragment (DF) bit.

As illustrated in Figure 6-8, CoAP communications across an IoT infrastructure can take various paths. 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. Proxy mechanisms are also defined, and RFC 7252 details a basic HTTP mapping for CoAP. 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 non-constrained networks.

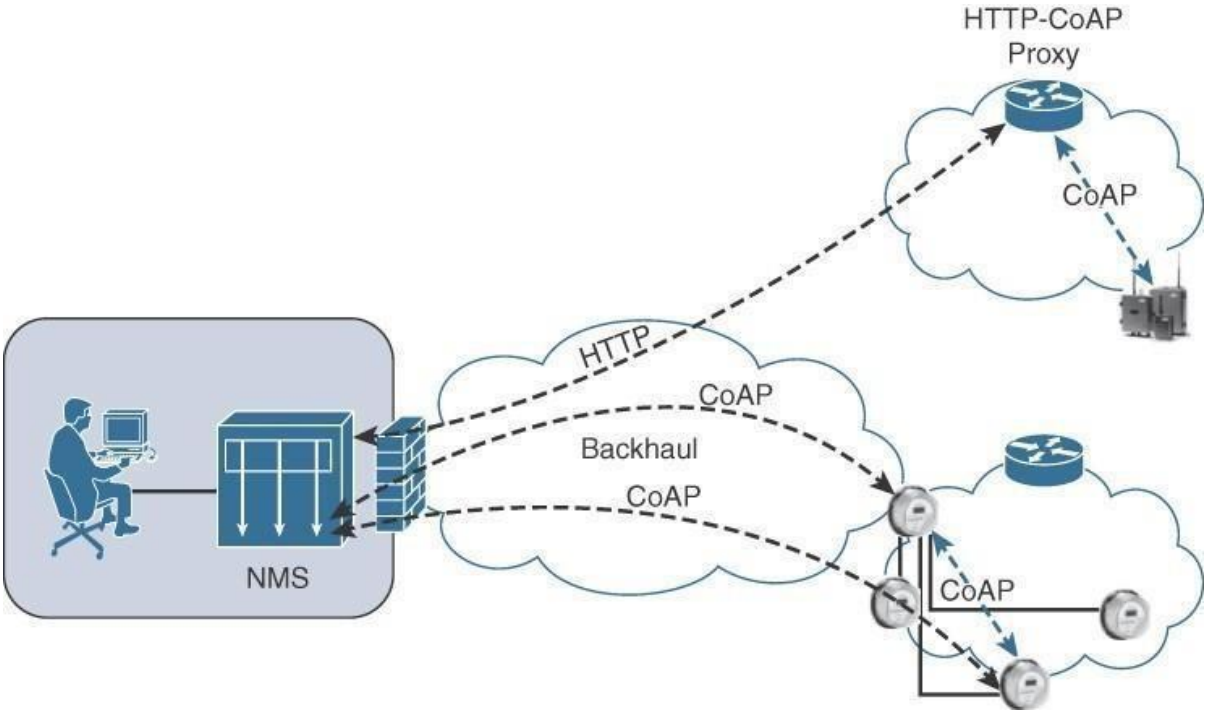



**Figure 6-8** *CoAP Communications in IoT Infrastructures*

Just like HTTP, 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.

Example 6-2 shows the CoAP URI format. You may notice that the CoAP URI format is similar to HTTP/HTTPS. The **coap**/**coaps** URI scheme identifies a resource, including host information and optional UDP port, as indicated by the **host** and **port** parameters in the URI.

**Example 6-2** *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. CoAP code, method and response codes, option numbers, and content format have been assigned by IANA as Constrained RESTful Environments (CoRE)

parameters. (For more information on these parameters, see www.iana.org/assignments/core-parameters/core-parameters.xhtml.)

While running over UDP, CoAP offers a reliable transmission of messages when a CoAP header is marked as ―confirmable.‖ In addition, 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, as shown in Figure 6-9. If a recipient can‘t process a non-confirmable message, a reset message is sent.

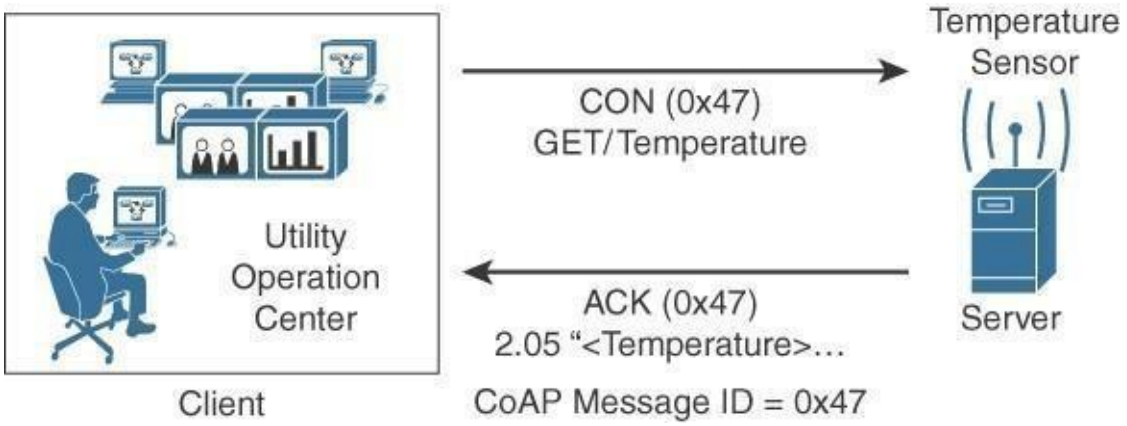



**Figure 6-9** *CoAP Reliable Transmission Example*

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.

With often no affordable manual configuration on the IoT endpoints, a CoAP server offering services and resources needs to be discovered by the CoAP clients. Services from a CoAP server can either be discovered by learning a URI in a namespace or through the ―All CoAP nodes‖ multicast address. When utilizing the URI scheme for discovering services, the default port 5683 is used for non-secured CoAP, or **coap**, while port 5684 is utilized for DTLS-secured CoAP, or **coaps**. The CoAP server must be in listening state on these ports, unless a different port number is associated with the URI in a namespace.

A wide range of CoAP implementations are available. Some are published with open source licenses, and others are part of vendor solutions.

## 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). (For more information on OASIS, see www.oasis-open.org.)

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 6-10.

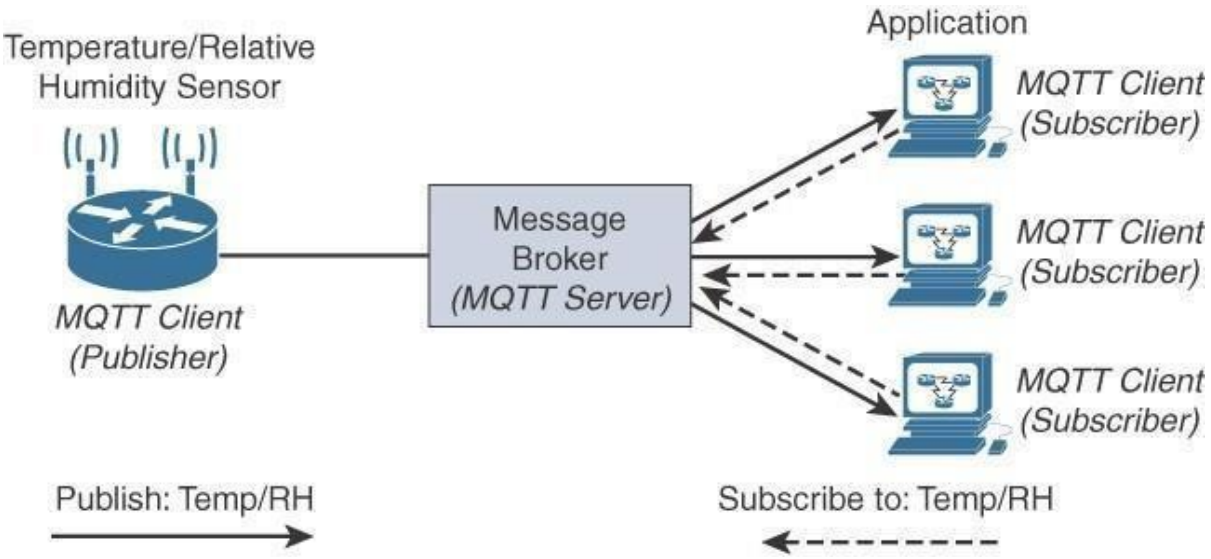



**Figure 6-10** *MQTT Publish/Subscribe Framework*

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

With MQTT, clients can subscribe to all data (using a wildcard character) or specific data from the information tree of a publisher. In addition, the presence of a message broker in MQTT decouples the data transmission between clients acting as publishers and subscribers. In fact, publishers and subscribers do not even know (or need to know) about each other. A benefit of having this decoupling is that the MQTT message broker ensures that information can be buffered and cached in case of network failures. This also means that publishers and subscribers do not have to be online at the same time.

MQTT control packets run over a TCP transport using port 1883. TCP

ensures an ordered, lossless stream of bytes between the MQTT client and the MQTT server. Optionally, MQTT can be secured using TLS on port 8883, and WebSocket (defined in RFC 6455) can also be used.

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

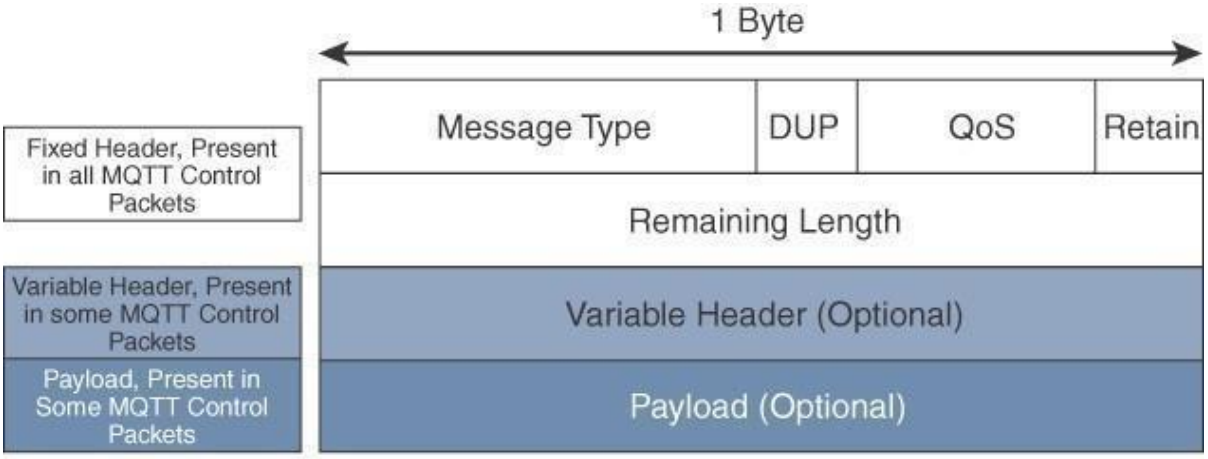



**Figure 6-11** *MQTT Message Format*

Compared to the CoAP message format in Figure 6-7, you can see that MQTT contains a smaller header of 2 bytes compared to 4 bytes for CoAP. The first MQTT field in the header is Message Type, which identifies the kind of MQTT packet within a message. 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<br>Server to client | Publish message |
| PUBACK | 4 | Client to server<br>Server to client | Publish acknowledgement |
| PUBREC | 5 | Client to server<br>Server to client | Publish received |
| PUBREL | 6 | Client to server<br>Server to client | Publish release |
| PUBCOMP | 7 | Client to server<br>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*

The next field in the MQTT header is DUP (Duplication Flag). This flag, when set, allows the client to notate that the packet has been sent previously, but an acknowledgement was not received.

The QoS header field allows for the selection of three different QoS levels. These are discussed in more detail later in this chapter.

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 string or 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 to structure the topic name. The forward slash (/) in an MQTT topic name is used to separate each level within the topic tree and provide a hierarchical structure to the topic names. Figure 6-12 illustrates these concepts with **adt/lora.adeunis** being a topic level and **adt/lora/adeunis/0018B2000000023A** being an example of a topic name.

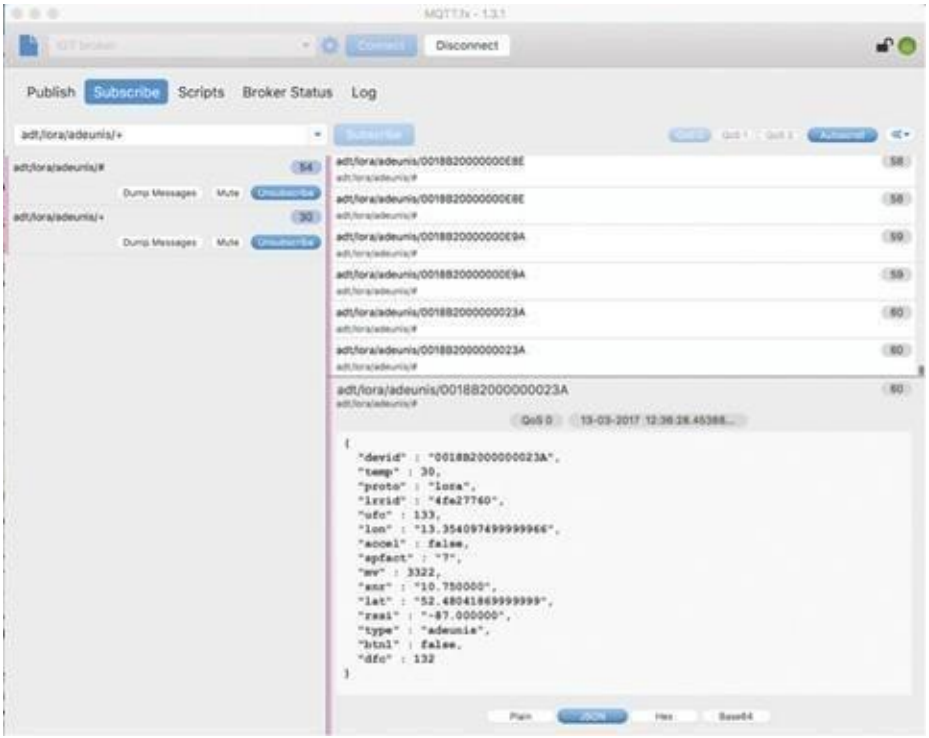



**Figure 6-12** *MQTT Subscription Example*

Wide flexibility is available to clients subscribing to a topic name. An exact topic can be subscribed to, or multiple topics can be subscribed to at once, through the use of wildcard characters. A subscription can contain one of the wildcard characters to allow subscription to multiple topics at once.

The pound sign (#) is a wildcard character that matches any number of levels within a topic. The multilevel wildcard represents the parent and any number

of child levels. For example, subscribing to **adt/lora/adeunis/#** enables the reception of the whole subtree, which could include topic names such as the following:

- **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 ($) must be excluded by the server when subscriptions start with wildcard characters (# or +). Often, these types of topic names are utilized for message broker internal statistics. So messages cannot be published to these topics by clients. For example, a subscription to **+/monitor/Temp** does not receive any messages published to **$SYS/monitor/Temp**. This topic could be the control channel for this temperature sensor.

PINGREQ/PINGRESP control packets are used to validate the connections between the client and server. Similar to ICMP pings that are part of IP, they are a sort of keepalive that helps to maintain and check the TCP session.

Securing MQTT connections through TLS is considered optional because it calls for more resources on constrained nodes. When TLS is not used, the client sends a clear-text username and password during the connection initiation. MQTT server implementations may also accept anonymous client connections (with the username/password being ―blank‖). When TLS is implemented, a client must validate the server certificate for proper authentication. Client authentication can also be performed through certificate exchanges with the server, depending on the server configuration.

The MQTT protocol offers three levels of quality of service (QoS). QoS for MQTT 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.

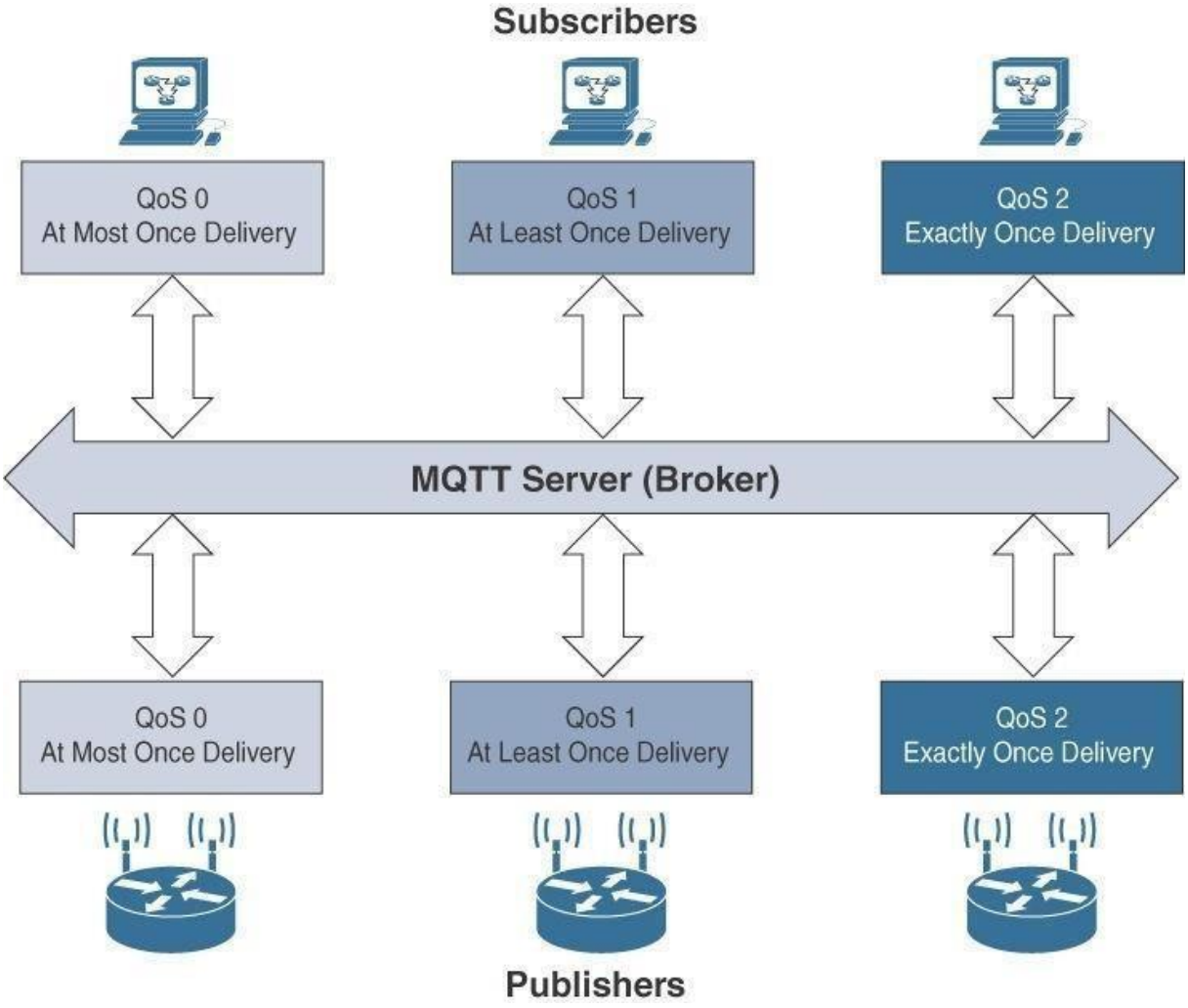



**Figure 6-13** *MQTT QoS Flows*

As with CoAP, a wide range of MQTT implementations are now available. They are either published as open source licenses or integrated into vendors' solutions, such as Facebook Messenger.

Now that both CoAP and MQTT have been discussed in detail, you can face questions like ―Which protocol is better for a given use case?‖ and ―Which one should I used in my IoT network?‖ Unfortunately, the answer is not always clear, and both MQTT and CoAP have their place. Table 6-3 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 6-3** *Comparison Between CoAP and MQTT*

# MODULE-4

# Chapter 7: Data and Analytics for IoT

In one of the famous episodes of the classic American science fiction TV series Star Trek, a harmless furry alien creature known as a ―tribble‖ is brought aboard the starship Enterprise. At first, the cute little tribble is treated like a pet, but then its unusual property shows up: It is able to multiply itself at an alarming rate, to the point that the ship soon becomes so filled with tribbles that they consume all supplies on board and begin interfering with the ship's systems.

The problems of data generated by IoT networks might well resemble ―The Trouble with Tribbles.‖ At first, IoT data is just a curiosity, and it's even useful if handled correctly. However, given time, as more and more devices are added to IoT networks, the data generated by these systems becomes overwhelming. Not only does this data begin to consume precious network bandwidth but server resources are increasingly taxed in their attempt to process, sort, and analyze the data.

Traditional data management systems are simply unprepared for the demands of what has come to be known as ―big data.‖ As discussed throughout this book, the real value of IoT is not just in connecting things but rather in the data produced by those things, the new services you can enable via those connected things, and the business insights that the data can reveal. However, to be useful, the data needs to be handled in a way that is organized and controlled. Thus, a new approach to data analytics is needed for the Internet of Things.

This chapter provides an overview of the field of data analytics from an IoT perspective, including the following sections:

- **An Introduction to Data Analytics for IoT:** This section introduces the subject of analytics for IoT and discusses the differences between structured and unstructured data. It also discusses how analytics relates to IoT data.
- **Machine Learning:** Once you have the data, what do you do with it, and how can you gain business insights from it? This section delves into the major types of machine learning that are used to gain business insights from IoT data.
- **Big Data Analytics Tools and Technology:** Big data is one of the most commonly used terms in the world of IoT. This section examines some of the most common technologies used in big data today, including Hadoop, NoSQL, MapReduce, and MPP.

- **Edge Streaming Analytics:** IoT requires that data be processed and analyzed as close to the endpoint as possible, in real-time. This section explores how streaming analytics can be used for such processing and analysis.
- **Network Analytics:** The final section of this chapter investigates the concept of network flow analytics using Flexible NetFlow in IoT systems. NetFlow can help you better understand the function of the overall system and heighten security in an IoT network.

## An Introduction to Data Analytics for IoT

In the world of IoT, the creation of massive amounts of data from sensors is common and one of the biggest challenges—not only from a transport perspective but also from a data management standpoint. A great example of the deluge of data that can be generated by IoT is found in the commercial aviation industry and the sensors that are deployed throughout an aircraft.

Modern jet engines are fitted with thousands of sensors that generate a whopping 10GB of data per second.[1] For example, modern jet engines, similar to the one shown in Figure 7-1, may be equipped with around 5000 sensors. Therefore, a twin engine commercial aircraft with these engines operating on average 8 hours a day will generate over 500 TB of data daily, and this is just the data from the engines! Aircraft today have thousands of other sensors connected to the airframe and other systems. In fact, a single wing of a modern jumbo jet is equipped with 10,000 sensors.

**Figure 7-1** Commercial Jet Engine

The potential for a petabyte (PB) of data per day per commercial airplane is not farfetched—and this is just for one airplane. Across the world, there are

approximately 100,000 commercial flights per day. The amount of IoT data coming just from the commercial airline business is overwhelming.

This example is but one of many that highlight the big data problem that is being exacerbated by IoT. Analyzing this amount of data in the most efficient manner possible falls under the umbrella of data analytics. Data analytics must be able to offer actionable insights and knowledge from data, no matter the amount or style, in a timely manner, or the full benefits of IoT cannot be realized.

Before diving deeper into data analytics, it is important to define a few key concepts related to data. For one thing, not all data is the same; it can be categorized and thus analyzed in different ways. Depending on how data is categorized, various data analytics tools and processing methods can be applied. Two important categorizations from an IoT perspective are whether the data is structured or unstructured and whether it is in motion or at rest.

## Structured Versus Unstructured Data

Structured data and unstructured data are important classifications as they typically require different toolsets from a data analytics perspective. Figure 7-2 provides a high-level comparison of structured data and unstructured data.

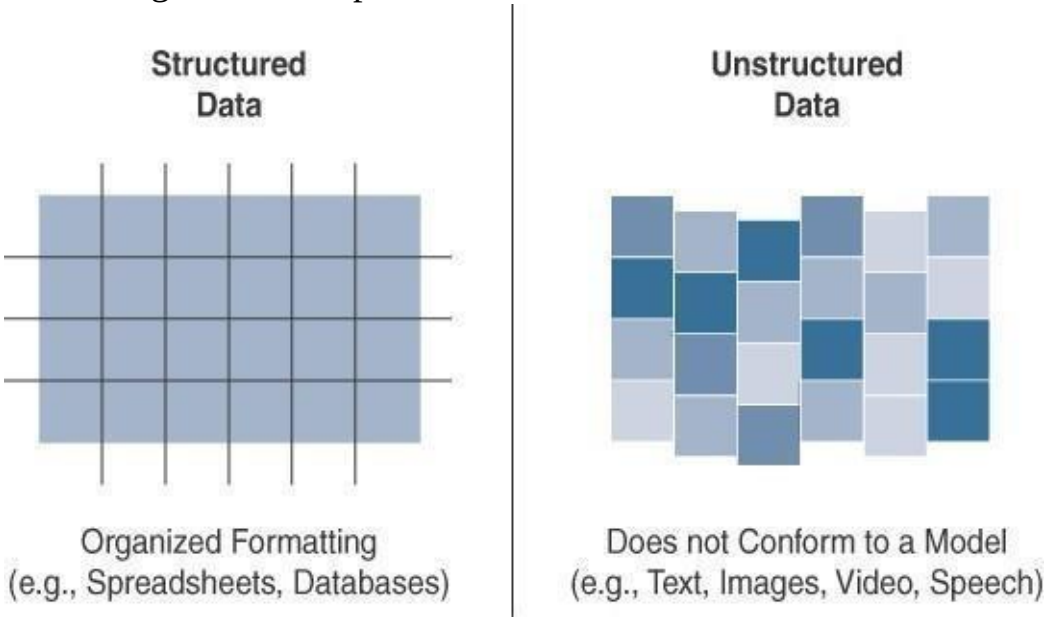



**Figure 7-2** Comparison Between Structured and Unstructured Data

Structured data means that the data follows a model or schema that defines how the data is represented or organized, meaning it fits well with a traditional relational database management system (RDBMS). In many cases you will find structured data in a simple tabular form—for example, a spreadsheet where data occupies a specific cell and can be explicitly defined and referenced.

Structured data can be found in most computing systems and includes everything from banking transaction and invoices to computer log files and router configurations. IoT sensor data often uses structured values, such as temperature, pressure, humidity, and so on, which are all sent in a known format. Structured data is easily formatted, stored, queried, and processed; for these reasons, it has been the core type of data used for making business decisions.

Because of the highly organizational format of structured data, a wide array of data analytics tools are readily available for processing this type of data. From custom scripts to commercial software like Microsoft Excel and Tableau, most people are familiar and comfortable with working with structured data.

Unstructured data lacks a logical schema for understanding and decoding the data through traditional programming means. Examples of this data type include text, speech, images, and video. As a general rule, any data that does not fit neatly into a predefined data model is classified as unstructured data.

Smart objects in IoT networks generate both structured and unstructured data. Structured data is more easily managed and processed due to its well-defined organization. On the other hand, unstructured data can be harder to deal with and typically requires very different analytics tools for processing the data. Being familiar with both of these data classifications is important because knowing which data classification you are working with makes integrating with the appropriate data analytics solution much easier.

## Data in Motion Versus Data at Rest

As in most networks, data in IoT networks is either in transit (―data in motion‖) or being held or stored (―data at rest‖). Examples of data in motion include traditional client/server exchanges, such as web browsing and file transfers, and email. Data saved to a hard drive, storage array, or USB drive is data at rest.

From an IoT perspective, the data from smart objects is considered data in motion as it passes through the network en route to its final destination. This is often processed at the edge, using fog computing. When data is processed at the edge, it may be filtered and deleted or forwarded on for further processing and possible storage at a fog node or in the data center. Data does not come to rest at the edge.

When data arrives at the data center, it is possible to process it in real-time, just like at the edge, while it is still in motion. Tools with this sort of capability, such as Spark, Storm, and Flink, are relatively nascent compared to the tools for analyzing stored data. Later sections of this chapter provide more information on these real-time streaming analysis tools that are part of

the Hadoop ecosystem.

Data at rest in IoT networks can be typically found in IoT brokers or in some sort of storage array at the data center. Myriad tools, especially tools for structured data in relational databases, are available from a data analytics perspective. The best known of these tools is Hadoop. Hadoop not only helps with data processing but also data storage. It is discussed in more detail later in this chapter.

## IoT Data Analytics Overview

The true importance of IoT data from smart objects is realized only when the analysis of the data leads to actionable business intelligence and insights. Data analysis is typically broken down by the types of results that are produced. As shown in Figure 7-3, there are four types of data analysis results:

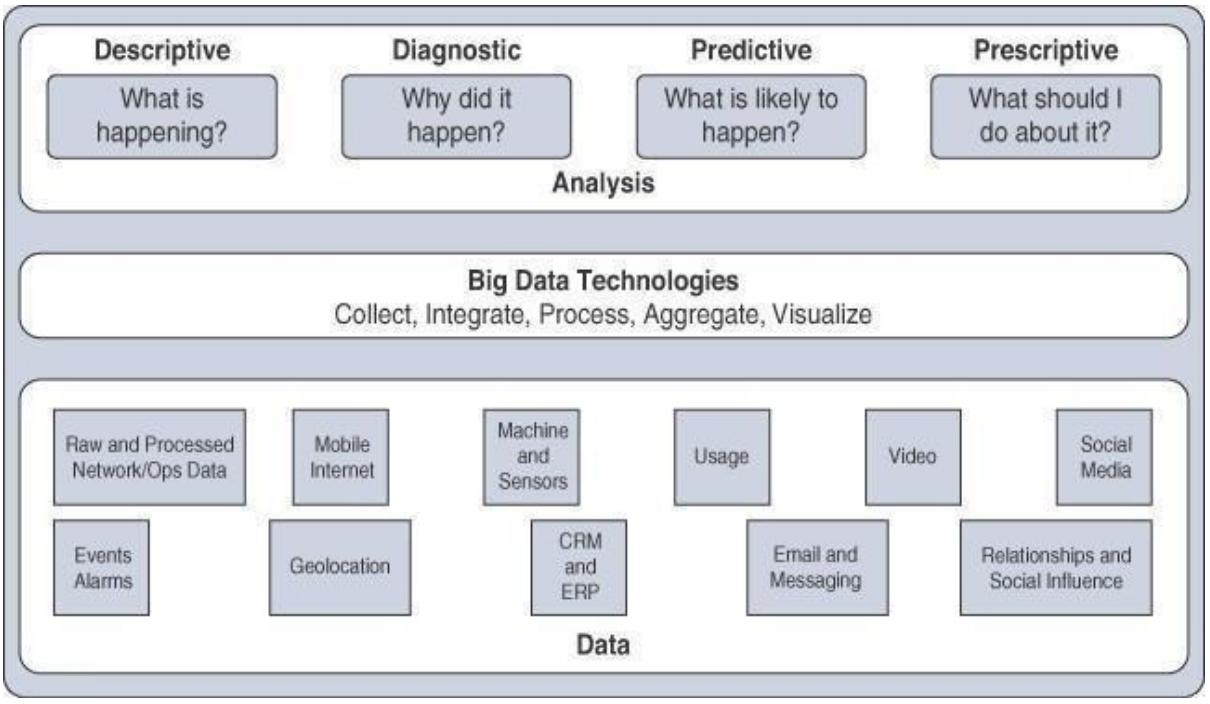



**Figure 7-3** Types of Data Analysis Results

- **Descriptive:** Descriptive data analysis tells you what is happening, either now or in the past. For example, a thermometer in a truck engine reports temperature values every second. From a descriptive analysis perspective, you can pull this data at any moment to gain insight into

the current operating condition of the truck engine. If the temperature value is too high, then there may be a cooling problem or the engine may be experiencing too much load.

- **Diagnostic:** When you are interested in the ―why,‖ diagnostic data analysis can provide the answer. Continuing with the example of the temperature sensor in the truck engine, you might wonder why the truck engine failed. Diagnostic analysis might show that the temperature of the engine was too high, and the engine overheated. Applying diagnostic analysis across the data generated by a wide range of smart objects can provide a clear picture of why a problem or an event occurred.
- **Predictive:** Predictive analysis aims to foretell problems or issues before they occur. For example, with historical values of temperatures for the truck engine, predictive analysis could provide an estimate on the remaining life of certain components in the engine. These components could then be proactively replaced before failure occurs. Or perhaps if temperature values of the truck engine start to rise slowly over time, this could indicate the need for an oil change or some other sort of engine cooling maintenance.
- **Prescriptive:** Prescriptive analysis goes a step beyond predictive and recommends solutions for upcoming problems. A prescriptive analysis of the temperature data from a truck engine might calculate various alternatives to cost-effectively maintain our truck. These calculations could range from the cost necessary for more frequent oil changes and cooling maintenance to installing new cooling equipment on the engine or upgrading to a lease on a model with a more powerful engine. Prescriptive analysis looks at a variety of factors and makes the appropriate recommendation.

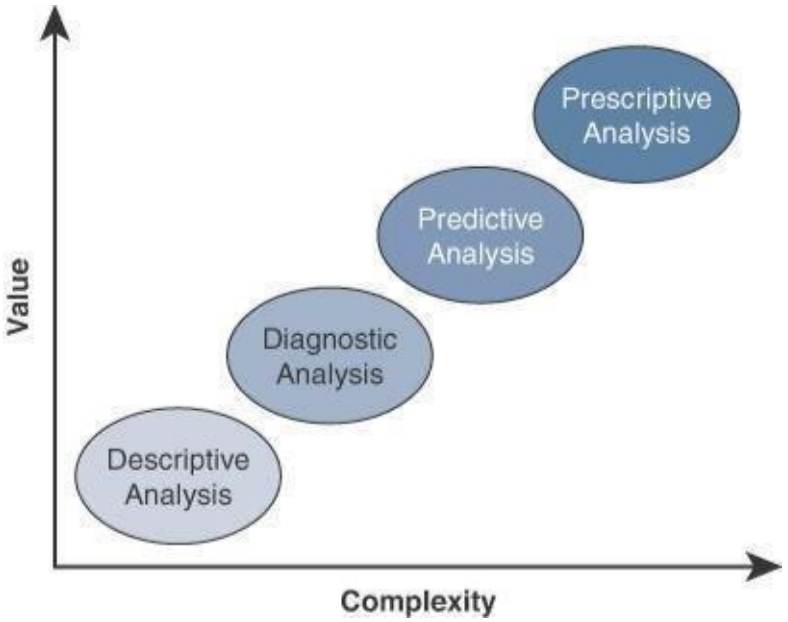



**Figure 7-4** Application of Value and Complexity Factors to the Types of Data Analysis

## IoT Data Analytics Challenges

As IoT has grown and evolved, it has become clear that traditional data analytics solutions were not always adequate. For example, traditional data analytics typically employs a standard RDBMS and corresponding tools, but the world of IoT is much more demanding. While relational databases are still used for certain data types and applications, they often struggle with the nature of IoT data. IoT data places two specific challenges on a relational database:

- **Scaling problems:** Due to the large number of smart objects in most IoT networks that continually send data, relational databases can grow incredibly large very quickly. This can result in performance issues that can be costly to resolve, often requiring more hardware and architecture changes.
- **Volatility of data:** With relational databases, it is critical that the schema be designed correctly from the beginning. Changing it later can slow or stop the database from operating. Due to the lack of flexibility, revisions to the schema must be kept at a minimum. IoT data, however, is volatile in the sense that the data model is likely to change and evolve over time. A dynamic schema is often required so that data model changes can be made daily or even hourly.

To deal with challenges like scaling and data volatility, a different type of database, known as NoSQL, is being used. Structured Query Language (SQL) is the computer language used to communicate with an RDBMS. As the name

implies, a NoSQL database is a database that does not use SQL. It is not set up in the traditional tabular form of a relational database. NoSQL databases do not enforce a strict schema, and they support a complex, evolving data model. These databases are also inherently much more scalable. (For more information on NoSQL, see the section ―NoSQL Databases‖ later in the chapter.)

In addition to the relational database challenges that IoT imposes, with its high volume of smart object data that frequently changes, IoT also brings challenges with the live streaming nature of its data and with managing data at the network level. Streaming data, which is generated as smart objects transmit data, is challenging because it is usually of a very high volume, and it is valuable only if it is possible to analyze and respond to it in real-time.

Real-time analysis of streaming data allows you to detect patterns or anomalies that could indicate a problem or a situation that needs some kind of immediate response. To have a chance of affecting the outcome of this problem, you naturally must be able to filter and analyze the data while it is occurring, as close to the edge as possible.

The market for analyzing streaming data in real-time is growing fast. Major cloud analytics providers, such as Google, Microsoft, and IBM, have streaming analytics offerings, and various other applications can be used in house. (Edge streaming analytics is discussed in depth later in this chapter.)

Another challenge that IoT brings to analytics is in the area of network data, which is referred to as network analytics. With the large numbers of smart objects in IoT networks that are communicating and streaming data, it can be challenging to ensure that these data flows are effectively managed, monitored, and secure. Network analytics tools such as Flexible NetFlow and IPFIX provide the capability to detect irregular patterns or other problems in the flow of IoT data through a network. Network analytics, including both Flexible NetFlow and IPFIX, is covered in more detail later in this chapter.

## Machine Learning

One of the core subjects in IoT is how to makes sense of the data that is generated. Because much of this data can appear incomprehensible to the naked eye, specialized tools and algorithms are needed to find the data relationships that will lead to new business insights. This brings us to the subject of machine learning (ML).

Performing this kind of operation manually is almost impossible (or very, very slow and inefficient). Machines are needed to process information fast and react instantly when thresholds are met. For example, every time a new advance is made in the field of self-driving vehicles, abnormal pattern

recognition in a crowd, or any other automated intelligent and machine-assisted decision system, ML is named as the tool that made the advance possible. But ML is not new. It was invented in the middle of the twentieth century and actually fell out of fashion in the 1980s. So what has happened in ML that makes it the new tool of choice for IoT and data analytics?

## Machine Learning Overview

Machine learning is, in fact, part of a larger set of technologies commonly grouped under the term artificial intelligence (AI). This term used to make science fiction amateurs dream of biped robots and conscious machines, or of a Matrix-like world where machines would enslave humankind. In fact, AI includes any technology that allows a computing system to mimic human intelligence using any technique, from very advanced logic to basic ―if-then-else‖ decision loops. Any computer that uses rules to make decisions belongs to this realm. A simple example is an app that can help you find your parked car. A GPS reading of your position at regular intervals calculates your speed. A basic threshold system determines whether you are driving (for example, ―if speed > 20 mph or 30 kmh, then start calculating speed‖). When you park and disconnect from the car Bluetooth system, the app simply records the location when the disconnection happens. This is where your car is parked.

Beyond the appearance of artificial intelligence (the computer knows that you are parked and where this happened), the ruleset is very simple.

### Supervised Learning

In supervised learning, the machine is trained with input for which there is a known correct answer. For example, suppose that you are training a system to recognize when there is a human in a mine tunnel. A sensor equipped with a basic camera can capture shapes and return them to a computing system that is responsible for determining whether the shape is a human or something else (such as a vehicle, a pile of ore, a rock, a piece of wood, and so on.). With supervised learning techniques, hundreds or thousands of images are fed into the machine, and each image is labeled (human or nonhuman in this case).

This is called the training set. An algorithm is used to determine common parameters and common differences between the images. The comparison is usually done at the scale of the entire image, or pixel by pixel. Images are resized to have the same characteristics (resolution, color depth, position of the central figure, and so on), and each point is analyzed. Human images have certain types of shapes and pixels in certain locations (which correspond to the position of the face, legs, mouth, and so on). Each new image is compared to the set of known ―good images,‖ and a deviation is calculated to determine how different the new image is from the average human image and, therefore,

the probability that what is shown is a human figure. This process is called classification.

After training, the machine should be able to recognize human shapes. Before real field deployments, the machine is usually tested with unlabeled pictures —this is called the validation or the test set, depending on the ML system used—to verify that the recognition level is at acceptable thresholds. If the machine does not reach the level of success expected, more training is needed.

In other cases, the learning process is not about classifying in two or more categories but about finding a correct value. For example, the speed of the flow of oil in a pipe is a function of the size of the pipe, the viscosity of the oil, pressure, and a few other factors. When you train the machine with measured values, the machine can predict the speed of the flow for a new, and unmeasured, viscosity. This process is called regression; regression predicts numeric values, whereas classification predicts categories.

## Unsupervised Learning

In some cases, supervised learning is not the best method for a machine to help with a human decision. Suppose that you are processing IoT data from a factory manufacturing small engines. You know that about 0.1% of the produced engines on average need adjustments to prevent later defects, and your task is to identify them before they get mounted into machines and shipped away from the factory. With hundreds of parts, it may be very difficult to detect the potential defects, and it is almost impossible to train a machine to recognize issues that may not be visible. However, you can test

each engine and record multiple parameters, such as sound, pressure, temperature of key parts, and so on. Once data is recorded, you can graph these elements in relation to one another (for example, temperature as a function of pressure, sound versus rotating speed over time). You can then input this data into a computer and use mathematical functions to find groups. For example, you may decide to group the engines by the sound they make at a given temperature. A standard function to operate this grouping, K-means clustering, finds the mean values for a group of engines (for example, mean value for temperature, mean frequency for sound). Grouping the engines this way can quickly reveal several types of engines that all belong to the same category (for example, small engine of chainsaw type, medium engine of lawnmower type). All engines of the same type produce sounds and temperatures in the same range as the other members of the same group.

There will occasionally be an engine in the group that displays unusual characteristics (slightly out of expected temperature or sound range). This is the engine that you send for manual evaluation. The computing process associated with this determination is called unsupervised learning. This type of learning is unsupervised because there is not a ―good‖ or ―bad‖ answer known in advance. It is the variation from a group behavior that allows the computer to learn that something is different. The example of engines is, of course, very simple. In most cases, parameters are multidimensional.

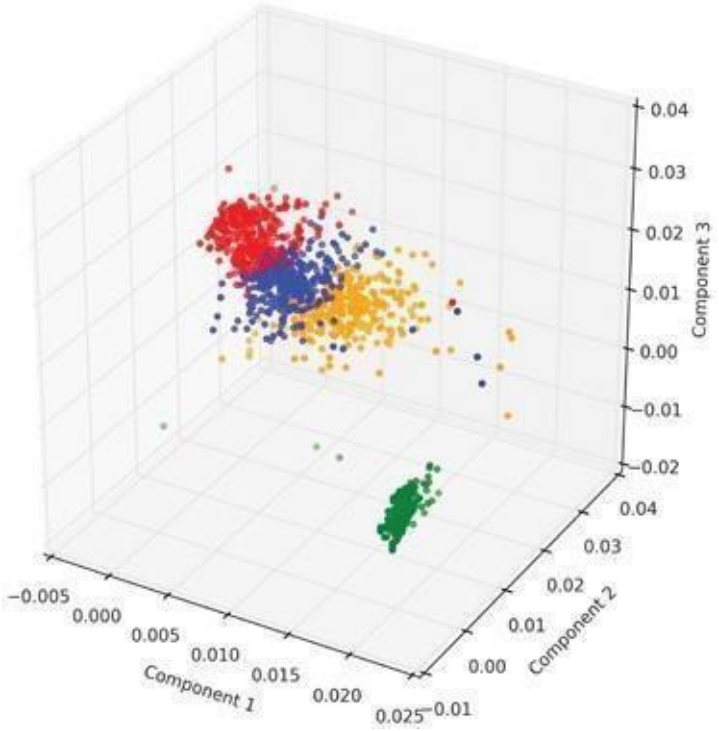



**Figure 7-5** Clustering and Deviation Detection Example

## Neural Networks

Processing multiple dimensions requires a lot of computing power. It is also difficult to determine what parameters to input and what combined variations should raise red flags. Similarly, supervised learning is efficient only with a large training set; larger training sets usually lead to higher accuracy in the prediction. This requirement is partly what made ML fade away somewhat in the 1980s and 1990s. Training the machines was often deemed too expensive and complicated.

This is where neural networks come into the picture. Neural networks are ML methods that mimic the way the human brain works. When you look at a human figure, multiple zones of your brain are activated to recognize colors, movements, facial expressions, and so on. Your brain combines these elements to conclude that the shape you are seeing is human. Neural networks mimic the same logic. The information goes through different algorithms (called units), each of which is in charge of processing an aspect of the information. The resulting value of one unit computation can be used directly or fed into another unit for further processing to occur. In this case, the neural network is said to have several layers. For example, a neural network processing human image recognition may have two units in a first layer that determines whether the image has straight lines and sharp angles—because vehicles commonly have straight lines and sharp angles, and human figures do not. If the image passes the first layer successfully (because there are no or only a small percentage of sharp angles and straight lines), a second layer may look for different features (presence of face, arms, and so on), and then a third layer might compare the image to images of various animals and conclude that the shape is a human (or not). The great efficiency of neural networks is that each unit processes a simple test, and therefore computation is quite fast. This model is demonstrated in Figure 7-6.

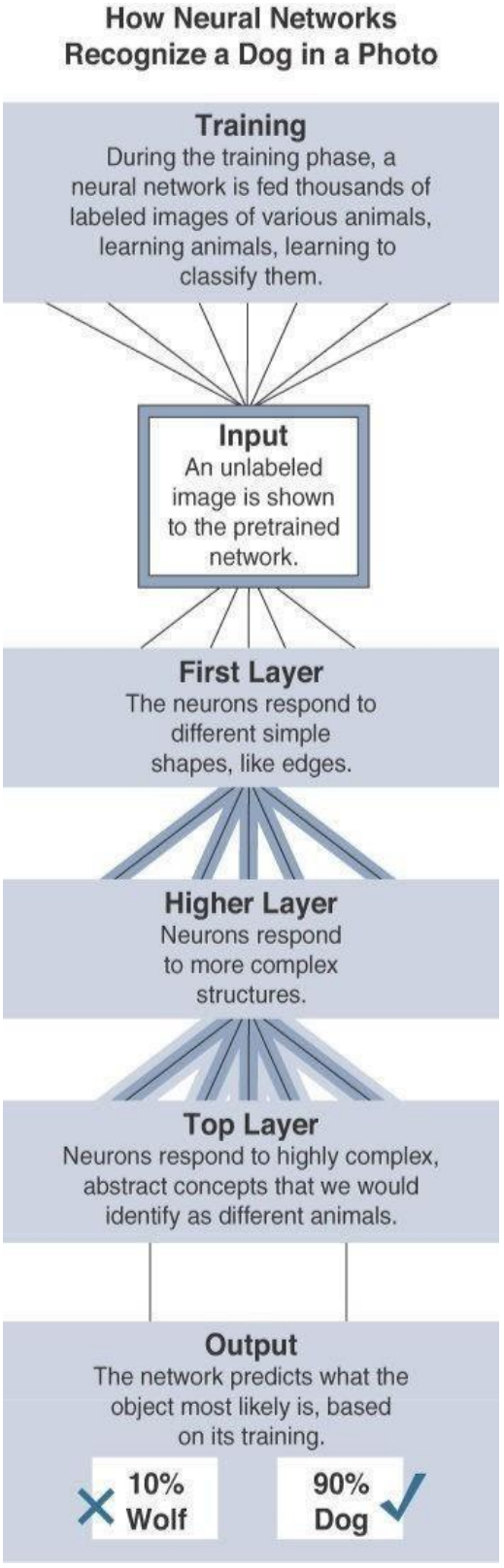



**Figure 7-6** Neural Network Example

By contrast, old supervised ML techniques would compare the human figure to potentially hundreds of thousands of images during the training phase, pixel by pixel, making them difficult and expensive to implement (with a lot of training needed) and slow to operate. Neural networks have been the

subject of much research work. Multiple research and optimization efforts have examined the number of units and layers, the type of data processed at each layer, and the type and combination of algorithms used to process the data to make processing more efficient for specific applications. Image processing can be optimized with certain types of algorithms that may not be optimal for crowd movement classification. Another algorithm may be found in this case that would revolutionize the way these movements are processed and analyzed. Possibilities are as numerous as the applications where they can be used.

In a sense, neural networks rely on the idea that information is divided into key components, and each component is assigned a weight. The weights compared together decide the classification of this information (no straight lines + face + smile = human).

When the result of a layer is fed into another layer, the process is called deep learning (―deep‖ because the learning process has more than a single layer). One advantage of deep learning is that having more layers allows for richer intermediate processing and representation of the data. At each layer, the data can be formatted to be better utilized by the next layer. This process increases the efficiency of the overall result.

## Machine Learning and Getting Intelligence from Big Data

When the principles of machine learning are clear, the application to IoT becomes obvious. The difficulty resides in determining the right algorithm and the right learning model for each use case. Such an analysis goes beyond the scope of this chapter, but it can be useful to organize ML operations into two broad subgroups:

- **Local learning:** In this group, data is collected and processed locally, either in the sensor itself (the edge node) or in the gateway (the fog node).
- **Remote learning:** In this group, data is collected and sent to a central computing unit (typically the data center in a specific location or in the cloud), where it is processed.

Regardless of the location where (and, therefore, the scale at which) data is processed, common applications of ML for IoT revolve around four major domains:

- **Monitoring:** Smart objects monitor the environment where they operate. Data is processed to better understand the conditions of operations. These conditions can refer to external factors, such as air temperature, humidity, or presence of carbon dioxide in a mine, or to

operational internal factors, such as the pressure of a pump, the viscosity of oil flowing in a pipe, and so on. ML can be used with monitoring to detect early failure conditions (for example, K-means deviations showing out-of-range behavior) or to better evaluate the environment (such as shape recognition for a robot automatically sorting material or picking goods in a warehouse or a supply chain).

- **Behavior control:** Monitoring commonly works in conjunction with behavior control. When a given set of parameters reach a target threshold—defined in advance (that is, supervised) or learned dynamically through deviation from mean values (that is, unsupervised)—monitoring functions generate an alarm. This alarm can be relayed to a human, but a more efficient and more advanced system would trigger a corrective action, such as increasing the flow of fresh air in the mine tunnel, turning the robot arm, or reducing the oil pressure in the pipe.

- **Operations optimization:** Behavior control typically aims at taking corrective actions based on thresholds. However, analyzing data can also lead to changes that improve the overall process. For example, a water purification plant in a smart city can implement a system to monitor the efficiency of the purification process based on which chemical (from company A or company B) is used, at what temperature, and associated to what stirring mechanism (stirring speed and depth).
  Neural networks can combine multiples of such units, in one or several layers, to estimate the best chemical and stirring mix for a target air temperature. This intelligence can help the plant reduce its consumption of chemicals while still operating at the same purification efficiency level. As a result of the learning, behavior control results in different machine actions. The objective is not merely to pilot the operations but to improve the efficiency and the result of these operations.

- **Self-healing, self-optimizing:** A fast-developing aspect of deep learning is the closed loop. ML-based monitoring triggers changes in machine behavior (the change is monitored by humans), and operations optimizations. In turn, the ML engine can be programmed to dynamically monitor and combine new parameters (randomly or semi-randomly) and automatically deduce and implement new optimizations when the results demonstrate a possible gain. The system becomes self-learning and self-optimizing.

For all these operations, a specific aspect of ML for IoT is the scale. A weather sensor mounted on a light pole in a street can provide information about the local pollution level. At the scale of the entire city, the authorities can monitor moving pollution clouds, and the global and local effects of mist

or humidity, pressure, and terrain. All this information can be combined with traffic data to globally regulate traffic light patterns, reduce emissions from industrial pollution sources, or increase the density of mass transit vehicles along the more affected axes. Meanwhile, at the local level, the LED on the light pole can increase or reduce its luminosity and change its color to adapt to local conditions. This change can be driven by either local condition processing (local learning) or inherited learning.

## Predictive Analytics

Machine learning and big data processing for IoT fit very well into the

digitization described in Chapter 1, ―What Is IoT?‖ The advanced stages of this model see the network self-diagnose and self-optimize. In the IoT world, this behavior is what the previous section describes. All this data can be returned to a data processing center in the cloud that can re-create a virtual twin of each locomotive. Modeling the state of each locomotive and combining this knowledge with anticipated travel and with the states (and detected failures) of all other locomotives of the same type circulating on the tracks of the entire city, province, state, or country allows the analytics platform to make very accurate predictions on what issue is likely to affect each train and each locomotive. Such predictive analysis allows preemptive maintenance and increases the safety and efficiency of operations.

Similarly, sensors combined with big data can anticipate defects or issues in vehicles operating in mines, in manufacturing machines, or any system that can be monitored, along with other similar systems.

## Big Data Analytics Tools and Technology

It is a common mistake for individuals new to the world of data management to use the terms big data and Hadoop interchangeably. Though it's true that Hadoop is at the core of many of today's big data implementations, it's not the only piece of the puzzle. Big data analytics can consist of many different software pieces that together collect, store, manipulate, and analyze all different data types. It helps to better understand the landscape by defining what big data is and what it is not. Generally, the industry looks to the ―three Vs‖ to categorize big data:

- **Velocity:** Velocity refers to how quickly data is being collected and analyzed. Hadoop Distributed File System is designed to ingest and process data very quickly. Smart objects can generate machine and sensor data at a very fast rate and require database or file systems capable of equally fast ingest functions.
- **Variety:** Variety refers to different types of data. Often you see data categorized as structured, semi-structured, or unstructured. Different database technologies may only be capable of accepting one of these

types. Hadoop is able to collect and store all three types. This can be beneficial when combining machine data from IoT devices that is very structured in nature with data from other sources, such as social media or multimedia, that is unstructured.

- **Volume:** Volume refers to the scale of the data. Typically, this is measured from gigabytes on the very low end to petabytes or even exabytes of data on the other extreme. Generally, big data implementations scale beyond what is available on locally attached storage disks on a single node. It is common to see clusters of servers that consist of dozens, hundreds, or even thousands of nodes for some large deployments.

The characteristics of big data can be defined by the sources and types of data. First is machine data, which is generated by IoT devices and is typically unstructured data. Second is transactional data, which is from sources that produce data from transactions on these systems, and, have high volume and structured. Third is social data sources, which are typically high volume and structured. Fourth is enterprise data, which is data that is lower in volume and very structured. Hence big data consists of data from all these separate sources.

Data collection and analysis are not new concepts in the industries that helped define IoT. Industrial verticals have long depended on the ability to get, collect, and record data from various processes in order to record trends and track performance and quality.

For example, many industrial automation and control systems feed data into two distinct database types, relational databases and historians. Relational databases, such as Oracle and Microsoft SQL, are good for transactional, or process, data. Their benefit is being able to analyze complex data relationships on data that arrives over a period of time. On the other hand, historians are optimized for time-series data from systems and processes. They are built with speed of storage and retrieval of data at their core, recording each data point in a series with the pertinent information about the system being logged. This data may consist of a sensor reading, the quantity of a material, a temperature reading, or flow data.

Relational databases and historians are mature technologies that have been with us for many years, but new technologies and techniques in the data management market have opened up new possibilities for sensor and machine data. These database technologies broadly fit into a few categories that each have strengths and potential drawbacks when used in an IoT context. The three most popular of these categories are massively parallel processing systems, NoSQL, and Hadoop.

## Massively Parallel Processing Databases

Enterprises have used relational databases for storing structured, row and column style data types for decades. Relational databases are often grouped into a broad data storage category called data warehouses. Though they are the centerpiece of most data architectures, they are often used for longer-term archiving and data queries that can often take minutes or hours. An example of this would be asking for all the items produced in the past year that had a particular specification. Depending on the number of items in the database and the complexity of the question being asked, the response could be slow to return.

Massively parallel processing (MPP) databases were built on the concept of the relational data warehouses but are designed to be much faster, to be efficient, and to support reduced query times. To accomplish this, MPP databases take advantage of multiple nodes (computers) designed in a scale-out architecture such that both data and processing are distributed across multiple systems.

optimized across the nodes in a structured SQL-like format that allows data analysts to work with the data using common SQL tools and applications. The earlier example of a complex SQL query could be distributed and optimized, resulting in a significantly faster response. Because data stored on MPPs must still conform to this relational structure, it may not be the only database type used in an IoT implementation. The sources and types of data may vary, requiring a database that is more flexible than relational databases allow.

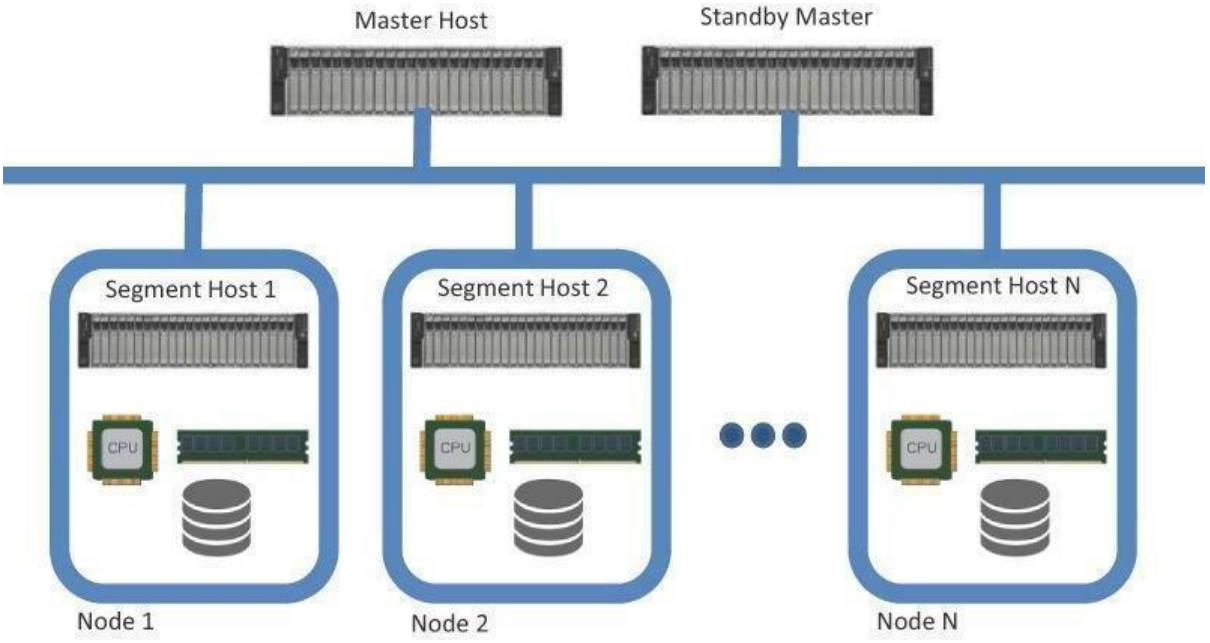



**Figure 7-7** MPP Shared-Nothing Architecture

## NoSQL Databases

NoSQL (―not only SQL‖) is a class of databases that support semi-structured and unstructured data, in addition to the structured data handled by data warehouses and MPPs. NoSQL is not a specific database technology; rather, it is an umbrella term that encompasses several different types of databases, including the following:

- **Document stores:** This type of database stores semi-structured data, such as XML or JSON. Document stores generally have query engines and indexing features that allow for many optimized queries.
- **Key-value stores:** This type of database stores associative arrays where a key is paired with an associated value. These databases are easy to build and easy to scale.
- **Wide-column stores:** This type of database stores similar to a key-value store, but the formatting of the values can vary from row to row, even in the same table.
- **Graph stores:** This type of database is organized based on the relationships between elements. Graph stores are commonly used for social media or natural language processing, where the connections between data are very relevant.

NoSQL was developed to support the high-velocity, urgent data requirements of modern web applications that typically do not require much repeated use. The original intent was to quickly ingest rapidly changing server logs and clickstream data generated by web-scale applications that did not neatly fit into the rows and columns required by relational databases. Similar to other data stores, like MPPs and Hadoop (discussed later), NoSQL is built to scale horizontally, allowing the database to span multiple hosts, and can even be distributed geographically.

Expanding NoSQL databases to other nodes is similar to expansion in other distributed data systems, where additional hosts are managed by a master node or process. This expansion can be automated by some NoSQL implementations or can be provisioned manually. This level of flexibility makes NoSQL a good candidate for holding machine and sensor data associated with smart objects.

Of the database types that fit under the NoSQL category, key-value stores and document stores tend to be the best fit for what is considered ―IoT data.‖ Key-value store is the technology that provides the foundation for many of today's RDBMSs, such as MS SQL, Oracle, and DB2.[3] However, unlike traditional RDBMSs, key-value stores on NoSQL are not limited to a single monolithic system. NoSQL key-value stores are capable of handling indexing and

persistence simultaneously at a high rate. This makes it a great choice for time-series data sets, which record a value at a given interval of time, such as a temperature or pressure reading from a sensor.

Many NoSQL databases provide additional capabilities, such as being able to query and analyze data within the database itself, eliminating the need to move and process it elsewhere. They also provide a variety of ways to query the database through an API, making it easy to integrate them with other data management applications.

## Hadoop

Hadoop is the most recent entrant into the data management market, but it is arguably the most popular choice as a data repository and processing engine. Hadoop was originally developed as a result of projects at Google and Yahoo!, and the original intent for Hadoop was to index millions of websites and quickly return search results for open source search engines. Initially, the project had two key elements:

- **Hadoop Distributed File System (HDFS):** A system for storing data across multiple nodes
- **MapReduce:** A distributed processing engine that splits a large task into smaller ones that can be run in parallel

Both of these elements are still present in current Hadoop distributions and provide the foundation for other projects that are discussed later in this chapter.

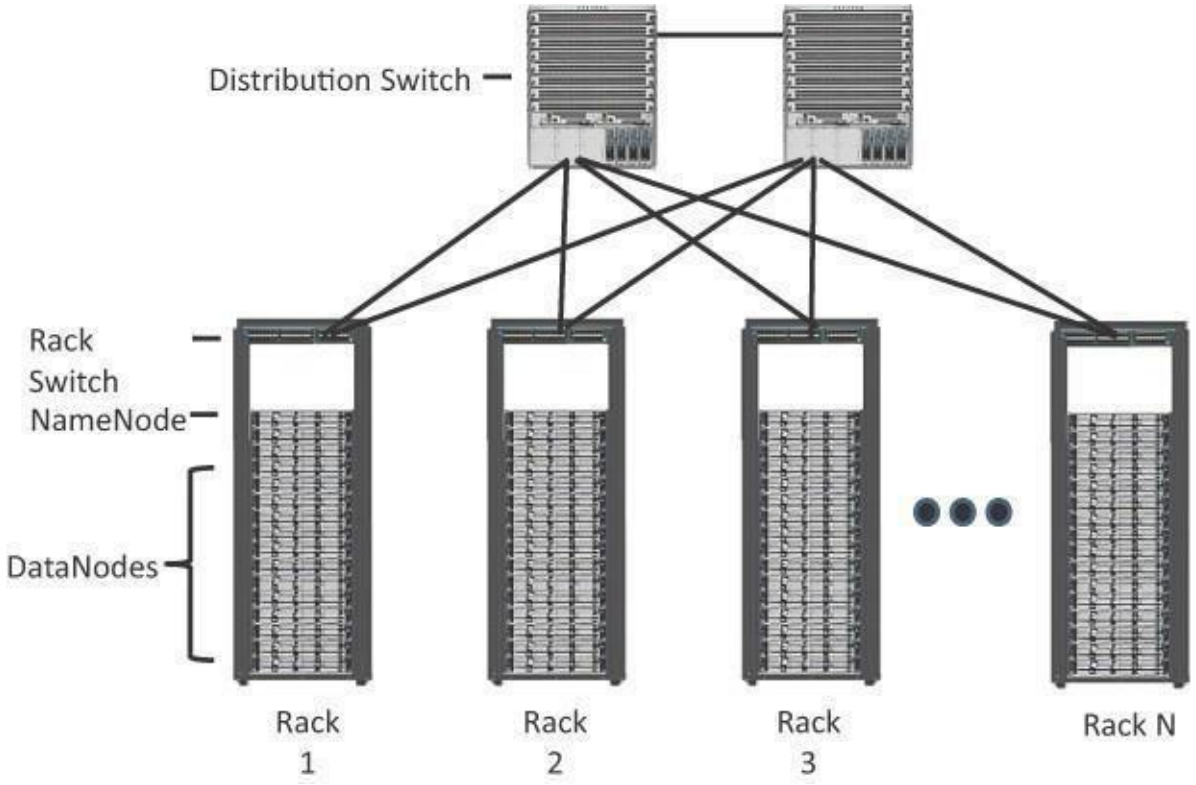



**Figure 7-8** Distributed Hadoop Cluster

Much like the MPP and NoSQL systems discussed earlier, Hadoop relies on a scale-out architecture that leverages local processing, memory, and storage to distribute tasks and provide a scalable storage system for data. Both MapReduce and HDFS take advantage of this distributed architecture to store and process massive amounts of data and are thus able to leverage resources from all nodes in the cluster. For HDFS, this capability is handled by

specialized nodes in the cluster, including NameNodes and DataNodes (see Figure 7-8):

- **NameNodes:** These are a critical piece in data adds, moves, deletes, and reads on HDFS. They coordinate where the data is stored, and maintain a map of where each block of data is stored and where it is replicated. All interaction with HDFS is coordinated through the primary (active) NameNode, with a secondary (standby) NameNode notified of the changes in the event of a failure of the primary. The NameNode takes write requests from clients and distributes those files across the available nodes in configurable block sizes, usually 64 MB or 128 MB blocks. The NameNode is also responsible for instructing the DataNodes where replication should occur.
- **DataNodes**: These are the servers where the data is stored at the direction of the NameNode. It is common to have many DataNodes in a Hadoop cluster to store the data. Data blocks are distributed across several nodes and often are replicated three, four, or more times across nodes for redundancy. Once data is written to one of the DataNodes, the DataNode selects two (or more) additional nodes, based on replication policies, to ensure data redundancy across the cluster. Disk redundancy techniques such as Redundant Array of Independent Disks (RAID) are generally not used for HDFS because the NameNodes and DataNodes coordinate block-level redundancy with this replication technique.

Figure 7-9 shows the relationship between NameNodes and DataNodes and how data blocks are distributed across the cluster.

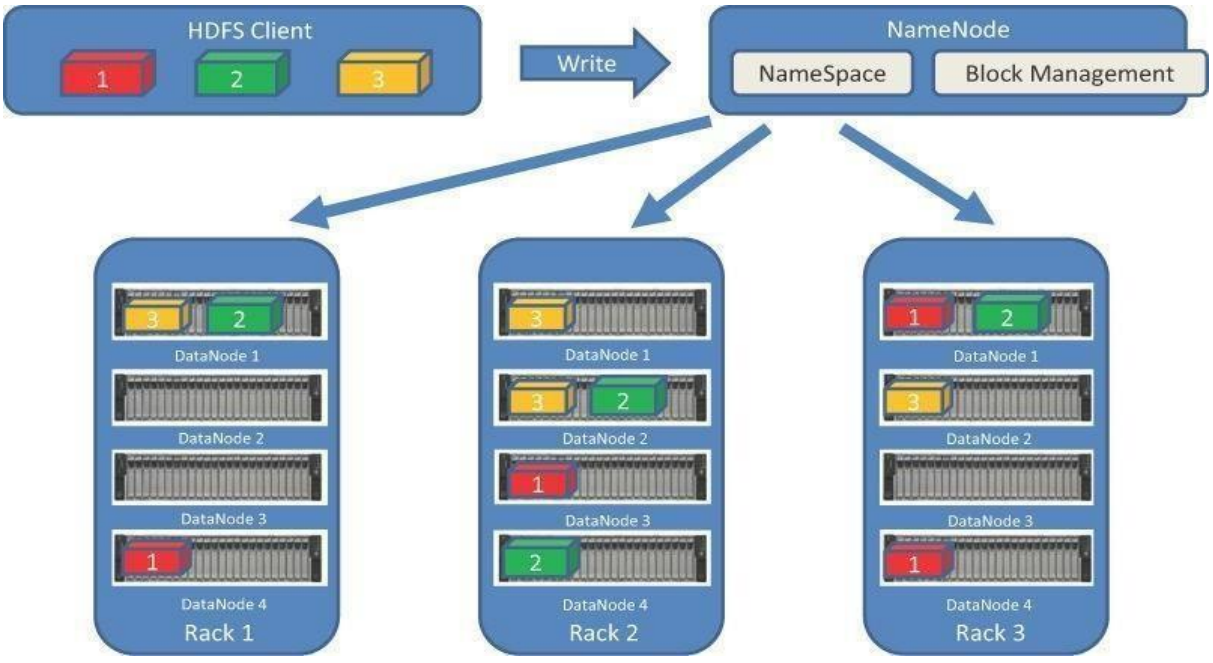



**Figure 7-9** Writing a File to HDFS

MapReduce leverages a similar model to batch process the data stored on the cluster nodes. Batch processing is the process of running a scheduled or ad hoc query across historical data stored in the HDFS. A query is broken down into smaller tasks and distributed across all the nodes running MapReduce in a cluster. While this is useful for understanding patterns and trending in historical sensor or machine data, it has one significant drawback: time.

Depending on how much data is being queried and the complexity of the query, the result could take seconds or minutes to return. If you have a real-time process running where you need a result at a moment's notice, MapReduce is not the right data processing engine for that. (Real-time streaming analytics is discussed later in this chapter.)

### YARN

Introduced with version 2.0 of Hadoop, YARN (Yet Another Resource Negotiator) was designed to enhance the functionality of MapReduce. With the initial release, MapReduce was responsible for batch data processing and job tracking and resource management across the cluster. YARN was developed to take over the resource negotiation and job/task tracking, allowing MapReduce to be responsible only for data processing.

With the development of a dedicated cluster resource scheduler, Hadoop was able to add additional data processing modules to its core feature set, including interactive SQL and real-time processing, in addition to batch processing using MapReduce.

## The Hadoop Ecosystem

As mentioned earlier, Hadoop plays an increasingly big role in the collection, storage, and processing of IoT data due to its highly scalable nature and its ability to work with large volumes of data. Many organizations have adopted Hadoop clusters for storage and processing of data and have looked for complimentary software packages to add additional functionality to their distributed Hadoop clusters. Since the initial release of Hadoop in 2011, many projects have been developed to add incremental functionality to Hadoop and have collectively become known as the Hadoop ecosystem.

Hadoop may have had meager beginnings as a system for distributed storage and processing, but it has since grown into a robust collection of projects that, combined, create a very complete data management and analytics framework. Hadoop now comprises more than 100 software projects under the Hadoop umbrella, capable of nearly every element in the data lifecycle, from collection, to storage, to processing, to analysis and visualization. Each of these individual projects is a unique piece of the overall data management solution. The following sections describe several of these packages and

discuss how they are used to collect or process data.

## Apache Kafka

Part of processing real-time events, such as those commonly generated by smart objects, is having them ingested into a processing engine. The process of collecting data from a sensor or log file and preparing it to be processed and analyzed is typically handled by messaging systems. Messaging systems are designed to accept data, or messages, from where the data is generated and deliver the data to stream-processing engines such as Spark Streaming or Storm. Apache Kafka is a distributed publisher-subscriber messaging system that is built to be scalable and fast. It is composed of topics, or message brokers, where producers write data and consumers read data from these topics. Figure 7-10 shows the data flow from the smart objects (producers), through a topic in Kafka, to the real-time processing engine. Due to the distributed nature of Kafka, it can run in a clustered configuration that can handle many producers and consumers simultaneously and exchanges information between nodes, allowing topics to be distributed over multiple nodes. The goal of Kafka is to provide a simple way to connect to data sources and allow consumers to connect to that data in the way they would like. The following sections describe several of these packages and discusses how they are used to collect or process data.

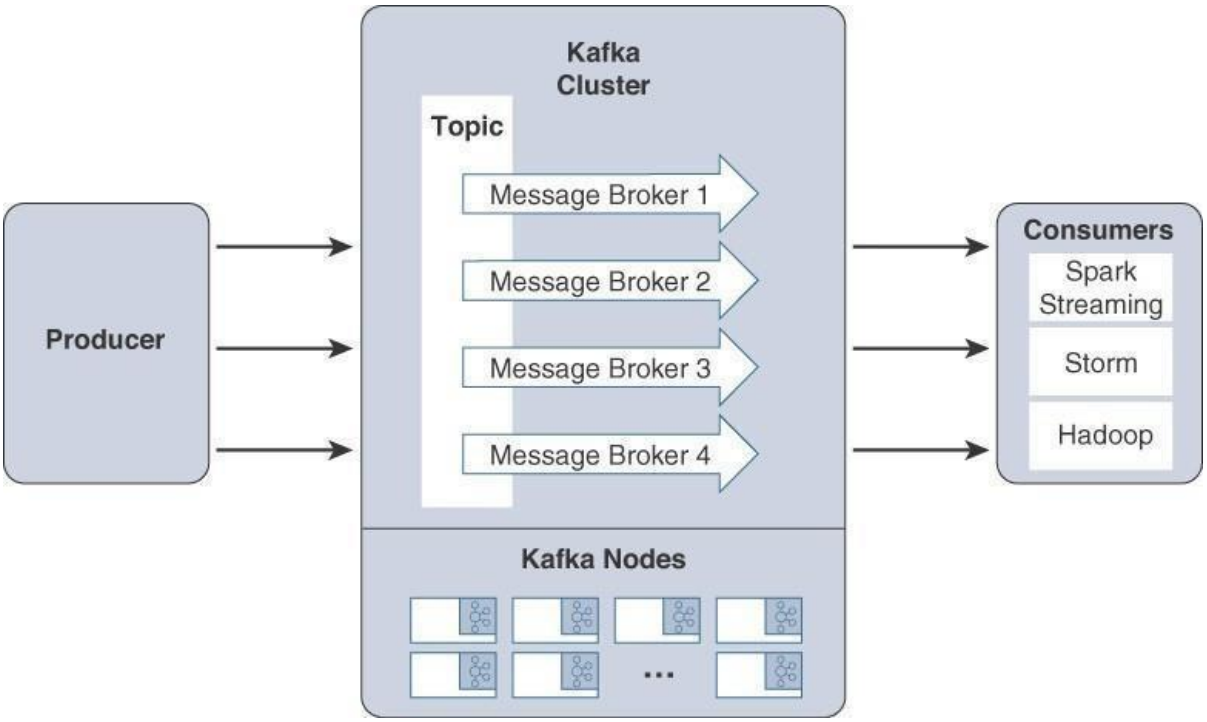



**Figure 7-10** Apache Kafka Data Flow

## Apache Spark

Apache Spark is an in-memory distributed data analytics platform designed to accelerate processes in the Hadoop ecosystem. The ―in-memory‖ characteristic of Spark is what enables it to run jobs very quickly. At each stage of a MapReduce operation, the data is read and written back to the disk, which means latency is introduced through each disk operation. However, with Spark, the processing of this data is moved into high-speed memory, which has significantly lower latency. This speeds the batch processing jobs and also allows for near-real-time processing of events.

Real-time processing is done by a component of the Apache Spark project called Spark Streaming. Spark Streaming is an extension of Spark Core that is responsible for taking live streamed data from a messaging system, like Kafka, and dividing it into smaller microbatches. These microbatches are called discretized streams, or DStreams. The Spark processing engine is able to operate on these smaller pieces of data, allowing rapid insights into the data and subsequent actions. Due to this ―instant feedback‖ capability, Spark is becoming an important component in many IoT deployments. Systems that control safety and security of personnel, time-sensitive processes in the manufacturing space, and infrastructure control in traffic management all benefit from these real-time streaming capabilities.

## Apache Storm and Apache Flink

As you work with the Hadoop ecosystem, you will inevitably notice that different projects are very similar and often have significant overlap with other projects. This is the case with data streaming capabilities. For example, Apache Spark is often used for both distributed streaming analytics and batch processing. Apache Storm and Apache Flink are other Hadoop ecosystem projects designed for distributed stream processing and are commonly deployed for IoT use cases. Storm can pull data from Kafka and process it in a near-real-time fashion, and so can Apache Flink. This space is rapidly evolving, and projects will continue to gain and lose popularity as they evolve.

## Lambda Architecture

Ultimately the key elements of a data infrastructure to support many IoT use cases involves the collection, processing, and storage of data using multiple technologies. Querying both data in motion (streaming) and data at rest (batch processing) requires a combination of the Hadoop ecosystem projects discussed. One architecture that is currently being leveraged for this functionality is the Lambda Architecture. Lambda is a data management

system that consists of two layers for ingesting data (Batch and Stream) and one layer for providing the combined data (Serving). These layers allow for the packages discussed previously, like Spark and MapReduce, to operate on the data independently, focusing on the key attributes for which they are designed and optimized. Data is taken from a message broker, commonly Kafka, and processed by each layer in parallel, and the resulting data is delivered to a data store where additional processing or queries can be run. Figure 7-11 shows this parallel data flow through the Lambda Architecture.

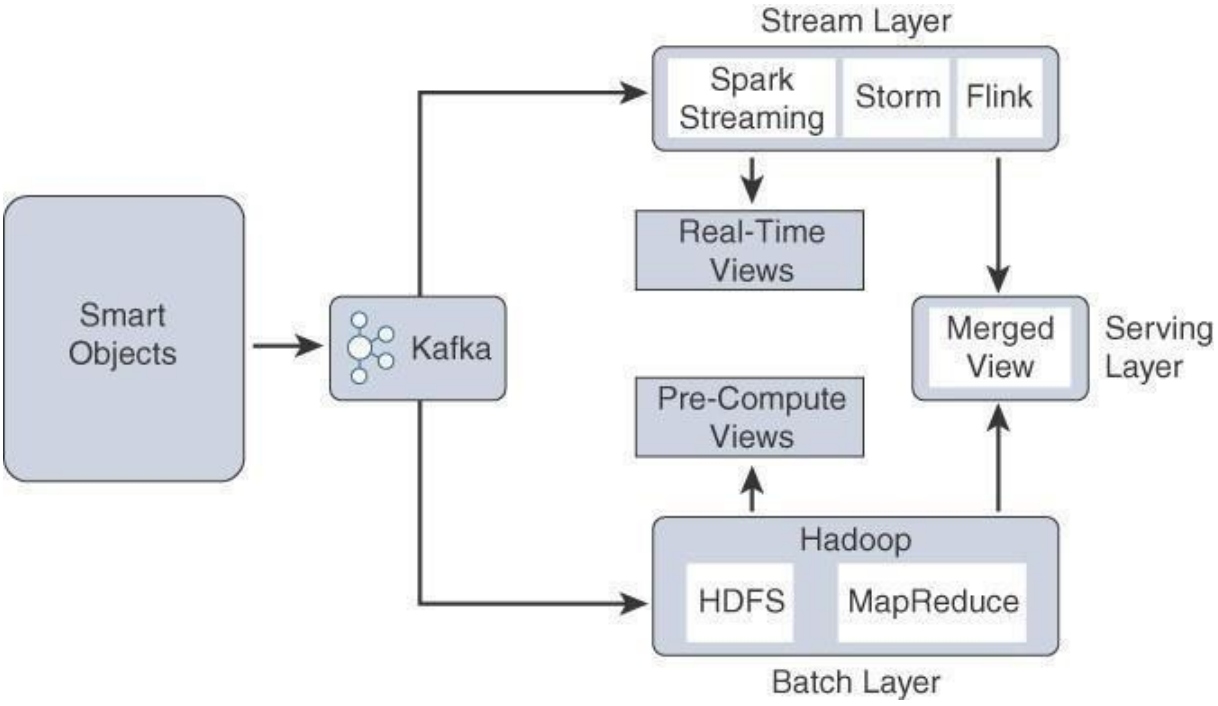



**Figure 7-11** Lambda Architecture

The Lambda Architecture is not limited to the packages in the Hadoop ecosystem, but due to its breadth and flexibility, many of the packages in the ecosystem fill the requirements of each layer nicely:

- **Stream layer:** This layer is responsible for near-real-time processing of events. Technologies such as Spark Streaming, Storm, or Flink are used to quickly ingest, process, and analyze data on this layer. Alerting and automated actions can be triggered on events that require rapid response or could result in catastrophic outcomes if not handled immediately.
- **Batch layer:** The Batch layer consists of a batch-processing engine and data store. If an organization is using other parts of the Hadoop ecosystem for the other layers, MapReduce and HDFS can easily fit the bill. Other database technologies, such as MPPs, NoSQL, or data warehouses, can also provide what is needed by this layer.
- **Serving layer:** The Serving layer is a data store and mediator that

decides which of the ingest layers to query based on the expected result or view into the data. If an aggregate or historical view is requested, it may invoke the Batch layer. If real-time analytics is needed, it may invoke the Stream layer. The Serving layer is often used by the data consumers to access both layers simultaneously.

## Edge Streaming Analytics

A major area of evolution for IT in the past few years has been the transition to cloud services. Nearly every large technology company is now selling software and services from the cloud, and this includes data analytics systems, whether they are offered as a service from a public cloud operator or are built in massive private data center clouds. However, analyzing a massive volume of time-sensitive IoT data in a centralized cloud is often not ideal.

In the world of IoT, vast quantities of data are generated on the fly and often need to be analyzed and responded to immediately. Not only is the volume of data generated at the edge immense—meaning the bandwidth requirements to the cloud or data center need to be engineered to match—but the data may be so time sensitive that it needs immediate attention, and waiting for deep analysis in the cloud simply isn't possible.

Most teams use sophisticated data analytics systems to enhance racing strategy, but in many cases, this equipment resides back in the team's data center, far away from the track. For a team that has its analytics software in a data center in the UK, the latency to Australia (the most remote race) is several hundred milliseconds away. The time it takes to collect and analyze this data as a batch process in a distant part of the world is not only inefficient but can mean the difference between a successful race strategy that adapts to changing conditions and one that lacks the flexibility and agility to send meaningful instructions to the drivers. In short, it can mean the difference between winning and losing a race.

## Comparing Big Data and Edge Analytics

When you hear the term big data, it is usually in reference to unstructured data that has been collected and stored in the cloud. The data is collected over time so that it can be analyzed through batch-processing tools, such as an RDBMS, Hadoop, or some other tool, at which point business insights are gained, and value is drawn from the data. Tools like Hadoop and MapReduce are great at tackling problems that require deep analytics on a large and complex quantity of unstructured data; however, due to their distance from the IoT endpoints and the bandwidth required to bring all the data back to the cloud, they are generally not well suited to real-time analysis of data as it is

generated.

In applying data analytics to the car racing example discussed earlier, big data analytics is used to examine all the statistics of the racing team and players based on their performance in the data center or cloud. While big data can apply analytics in real-time (as discussed earlier), it is mainly focused on batch-job analytics on large volumes of data. Streaming analytics involves analyzing a race while it is happening and trying to figure out who is going to win based on the actual performance in real-time—and this analysis is typically performed as close to the edge as possible. Streaming analytics allows you to continually monitor and assess data in real-time so that you can adjust or fine-tune your predictions as the race progresses.

In the context of IoT, with streaming analytics performed at the edge (either at the sensors themselves or very close to them, in a fog node that is, for example, integrated into the gateway), it is possible to process and act on the data in real-time without waiting for the results from a future batch-processing job in the cloud. Does this mean that streaming analytics replaces big data analytics in the cloud? Not at all. They both have roles to play anD both contribute to improved business insights and processes.

In one sense, if raw data is generated in the data center, it makes sense to analyze it there. But what if the majority of data is being generated in remote locations by sensors that are spread all over a wide area? To be truly effective at the moment it is created, the data needs to be analyzed and responded to as close to the edge as possible. Once it has been analyzed and reduced at the edge, the resultant data can be sent to the cloud and used to gain deeper insights over time. It is also important to remember that the edge isn't in just one place. The edge is highly distributed, which means analytics at the edge needs to be highly coordinated and structured. This also implies a communications system where edge/fog nodes are able to communicate with each other when necessary and report results to a big data system in the cloud.

From a business perspective, streaming analytics involves acting on data that is generated while it is still valuable, before it becomes stale. For example, roadway sensors combined with GPS wayfinding apps may tell a driver to avoid a certain highway due to traffic. This data is valuable for only a small window of time. Historically, it may be interesting to see how many traffic accidents or blockages have occurred on a certain segment of highway or to predict congestion based on past traffic data. However, for the driver in traffic receiving this information, if the data is not acted upon immediately, the data has little value.

From a security perspective, having instantaneous access to analyzed and preprocessed data at the edge also allows an organization to realize anomalies

in its network so those anomalies can be quickly contained before spreading to the rest of the network.

To summarize, the key values of edge streaming analytics include the following:

- **Reducing data at the edge:** The aggregate data generated by IoT devices is generally in proportion to the number of devices. The scale of these devices is likely to be huge, and so is the quantity of data they generate. Passing all this data to the cloud is inefficient and is unnecessarily expensive in terms of bandwidth and network infrastructure.
- **Analysis and response at the edge:** Some data is useful only at the edge (such as a factory control feedback system). In cases such as this, the data is best analyzed and acted upon where it is generated.
- **Time sensitivity:** When timely response to data is required, passing data to the cloud for future processing results in unacceptable latency. Edge analytics allows immediate responses to changing conditions.

## Edge Analytics Core Functions

To perform analytics at the edge, data needs to be viewed as real-time flows. Whereas big data analytics is focused on large quantities of data at rest, edge analytics continually processes streaming flows of data in motion. Streaming analytics at the edge can be broken down into three simple stages:

- **Raw input data:** This is the raw data coming from the sensors into the analytics processing unit.
- **Analytics processing unit (APU):** The APU filters and combines data streams (or separates the streams, as necessary), organizes them by time windows, and performs various analytical functions. It is at this point that the results may be acted on by micro services running in the APU.
- **Output streams:** The data that is output is organized into insightful streams and is used to influence the behavior of smart objects, and passed on for storage and further processing in the cloud. Communication with the cloud often happens through a standard publisher/subscriber messaging protocol, such as MQTT.

Figure 7-12 illustrates the stages of data processing in an edge APU.

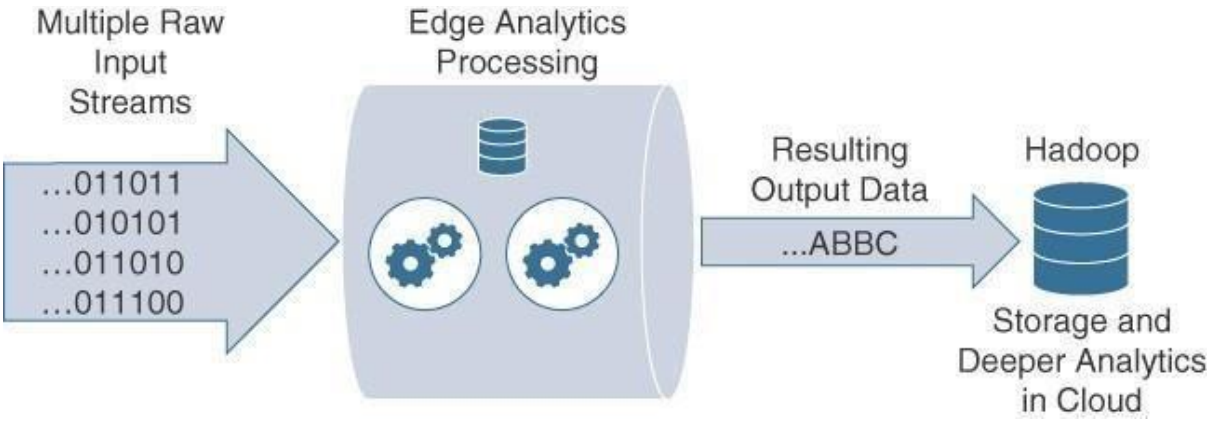



**Figure 7-12** Edge Analytics Processing Unit

In order to perform analysis in real-time, the APU needs to perform the following functions:

- **Filter:** The streaming data generated by IoT endpoints is likely to be very large, and most of it is irrelevant. For example, a sensor may simply poll on a regular basis to confirm that it is still reachable. This information is not really relevant and can be mostly ignored. The filtering function identifies the information that is considered important.

- **Transform:** In the data warehousing world, Extract, Transform, and Load (ETL) operations are used to manipulate the data structure into a form that can be used for other purposes. Analogous to data warehouse ETL operations, in streaming analytics, once the data is filtered, it needs to be formatted for processing.

- **Time:** As the real-time streaming data flows, a timing context needs to be established. This could be to correlated average temperature readings from sensors on a minute-by-minute basis. For example, Figure 7-13 shows an APU that takes input data from multiple sensors reporting temperature fluctuations. In this case, the APU is programmed to report the average temperature every minute from the sensors, based on an average of the past two minutes. (An example where this may be used is in real-time monitoring of food in a grocery store, where rolling averages of the temperature in open-air refrigeration units needs to be monitored to ensure the safety of the food.) Note that on the left side is the cleaned stream data. This data is presented as streams to the analytics engine (note the syntax at the bottom right of the figure) that establishes the time window and calculates the average temperature over the past two minutes. The results are reported on a per-minute basis (on the right side of the figure).

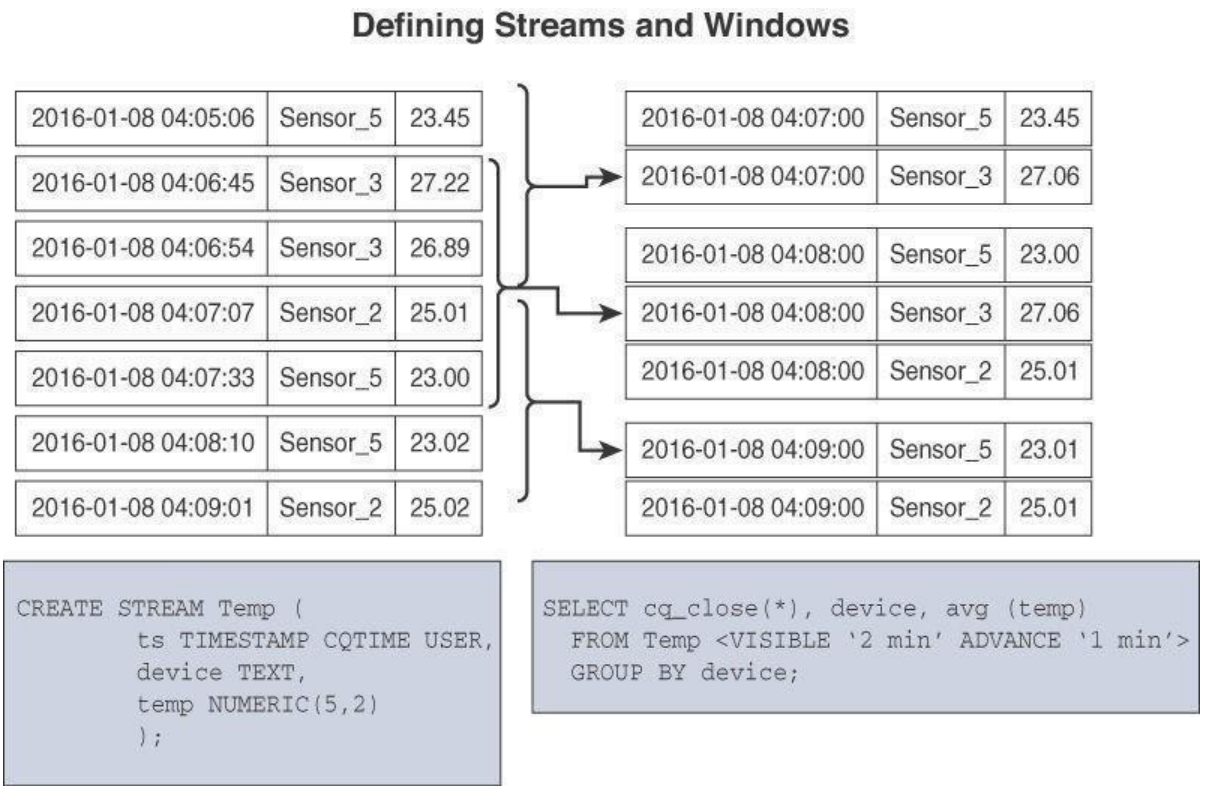



**Figure 7-13** Example: Establishing a Time Window for Analytics

- **Correlate:** Streaming data analytics becomes most useful when multiple data streams are combined from different types of sensors. For example, in a hospital, several vital signs are measured for patients, including body temperature, blood pressure, heart rate, and respiratory rate. These different types of data come from different instruments, but when this data is combined and analyzed, it provides an invaluable picture of the health of the patient at any given time.[4] However, correlation goes beyond just combining real-time data streams. Another key aspect is combining and correlating real-time measurements with preexisting, or historical, data. For example, historical data may include the patient's past medical history, such as blood test results. Combining historical data gives the live streaming data a powerful context and promotes more insights into the current condition of the patient (see Figure 7-14).

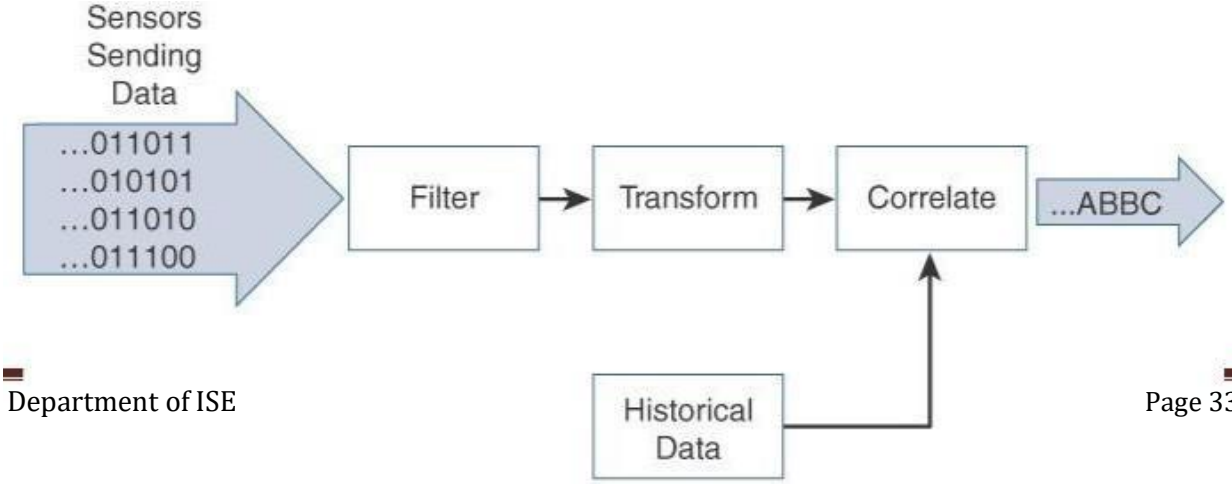


**Figure 7-14** Correlating Data Streams with Historical Data

- **Match patterns:** Once the data streams are properly cleaned, transformed, and correlated with other live streams as well as historical data sets, pattern matching operations are used to gain deeper insights to the data. For example, say that the APU has been collecting the patient's vitals for some time and has gained an understanding of the expected patterns for each variable being monitored. If an unexpected event arises, such as a sudden change in heart rate or respiration, the pattern matching operator recognizes this as out of the ordinary and can take certain actions, such as generating an alarm to the nursing staff. The patterns can be simple relationships, or they may be complex, based on the criteria defined by the application. Machine learning may be leveraged to identify these patterns.

- **Improve business intelligence:** Ultimately, the value of edge analytics is in the improvements to business intelligence that were not previously available. For example, conducting edge analytics on patients in a hospital allows staff to respond more quickly to the patient's changing needs and also reduces the volume of unstructured (and not always

useful) data sent to the cloud. Over time, the resulting changes in business logic can produce improvements in basic operations, bringing in higher levels of care as well as better efficiencies for the hospital.

## Distributed Analytics Systems

Depending on the application and network architecture, analytics can happen at any point throughout the IoT system. Streaming analytics may be performed directly at the edge, in the fog, or in the cloud data center. There are no hard-and-fast rules dictating where analytics should be done, but there are a few guiding principles. We have already discussed the value of reducing the data at the edge, as well as the value of analyzing information so it can be responded to before it gets stale. There is also value in stepping back from the edge to gain a wider view with more data. It's hard to see the forest when you are standing in the middle of it staring at a tree. In other words, sometimes better insights can be gained and data responded to more intelligently when we step back from the edge and look at a wider data set.

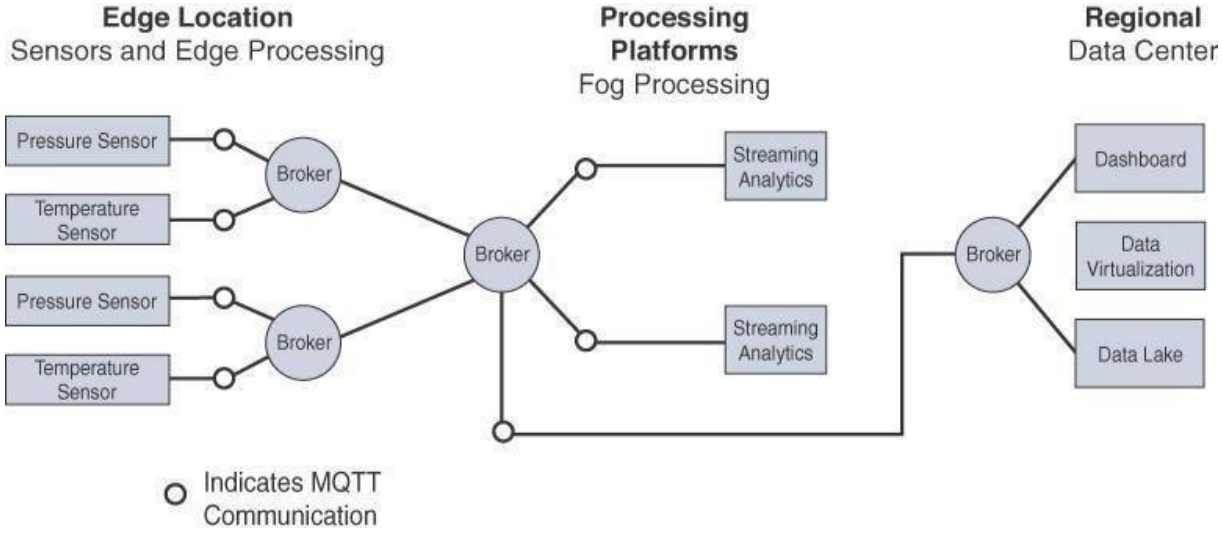



**Figure 7-15** Distributed Analytics Throughout the IoT System

## Network Analytics

Another form of analytics that is extremely important in managing IoT systems is network-based analytics. Unlike the data analytics systems previously discussed that are concerned with finding patterns in the data generated by endpoints, network analytics is concerned with discovering patterns in the communication flows from a network traffic perspective. Network analytics has the power to analyze details of communications patterns made by protocols and correlate this across the network. It allows you to understand what should be considered normal behavior in a network and to quickly identify anomalies that suggest network problems due to suboptimal paths, intrusive malware, or excessive congestion. Analysis of

traffic patterns is one of the most powerful tools in an IoT network engineer's troubleshooting arsenal.

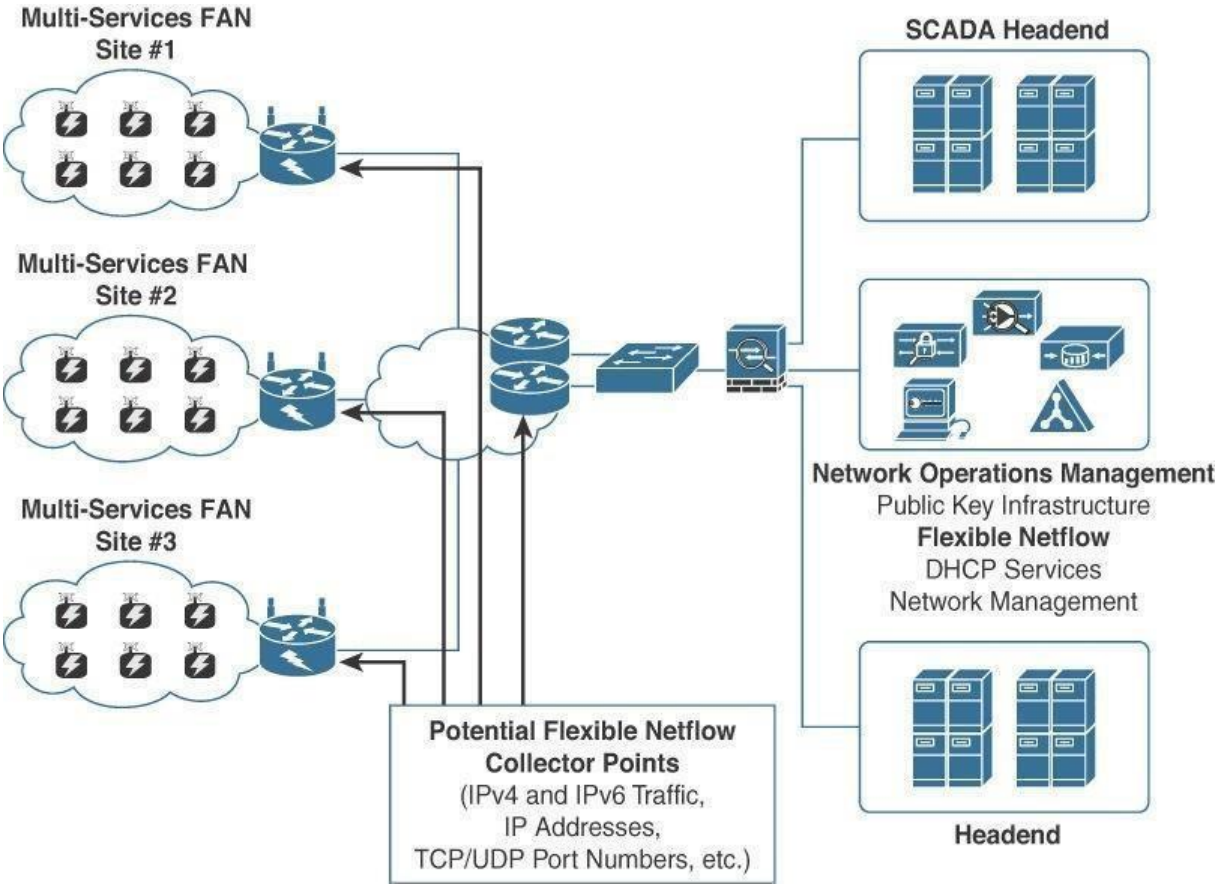



**Figure 7-16** Smart Grid FAN Analytics with NetFlow Example

This behavior represents a key aspect that can be leveraged when performing network analytics: Network analytics offer capabilities to cope with capacity planning for scalable IoT deployment as well as security monitoring in order to detect abnormal traffic volume and patterns (such as an unusual traffic spike for a normally quiet protocol) for both centralized or distributed architectures, such as fog computing.

One of the drivers of the adoption of an IP architectural framework for IoT is to leverage tools and processes largely known and deployed by Internet service providers (ISPs) as well as private corporate enterprise networks. To monitor network infrastructure, de facto industry standards and protocols allow pervasive characterization of IP traffic flows, including identification of source and/or destination addresses, data timing and volume, and application types within a network infrastructure. Flow statistics can be collected at different locations in the network. For example, centralized routers or

switches that aggregate subnetworks as well as nodes that are highly distributed and connect the last mile of the infrastructure can be used to collect flow information. After data is collected in a known format, it can be sent to an external network analytics tools that delivers unique services to network managers, like security and performance monitoring and capacity planning.

Other network management services, are as follows:

- **Network traffic monitoring and profiling:** Flow collection from the network layer provides global and distributed near-real-time monitoring capabilities. IPv4 and IPv6 networkwide traffic volume and pattern analysis helps administrators proactively detect problems and quickly troubleshoot and resolve problems when they occur.
- **Application traffic monitoring and profiling:** Monitoring and profiling can be used to gain a detailed time-based view of IoT access services, such as the application-layer protocols, including MQTT, CoAP, and DNP3, as well as the associated applications that are being used over the network.
- **Capacity planning:** Flow analytics can be used to track and anticipate IoT traffic growth and help in the planning of upgrades when deploying new locations or services by analyzing captured data over a long period of time. This analysis affords the opportunity to track and anticipate IoT network growth on a continual basis.
- **Security analysis:** Because most IoT devices typically generate a low volume of traffic and always send their data to the same server(s), any change in network traffic behavior may indicate a cyber security event, such as a denial of service (DoS) attack. Security can be enforced by ensuring that no traffic is sent outside the scope of the IoT domain. For example, with a LoRaWAN gateway, there should be no reason to see traffic sent or received outside the LoRaWAN network server and network management system. Such traffic could indicate an attack of some sort.
- **Accounting:** In field area networks, routers or gateways are often physically isolated and leverage public cellular services and VPNs for backhaul. Deployments may have thousands of gateways connecting the last-mile IoT infrastructure over a cellular network. Flow monitoring can thus be leveraged to analyze and optimize the billing, in

complement with other dedicated applications, such as Cisco Jasper, with a broader scope than just monitoring data flow.

- **Data warehousing and data mining:** Flow data (or derived information) can be warehoused for later retrieval and analysis in support of proactive analysis of multiservice IoT infrastructures and applications.

## Flexible NetFlow Architecture

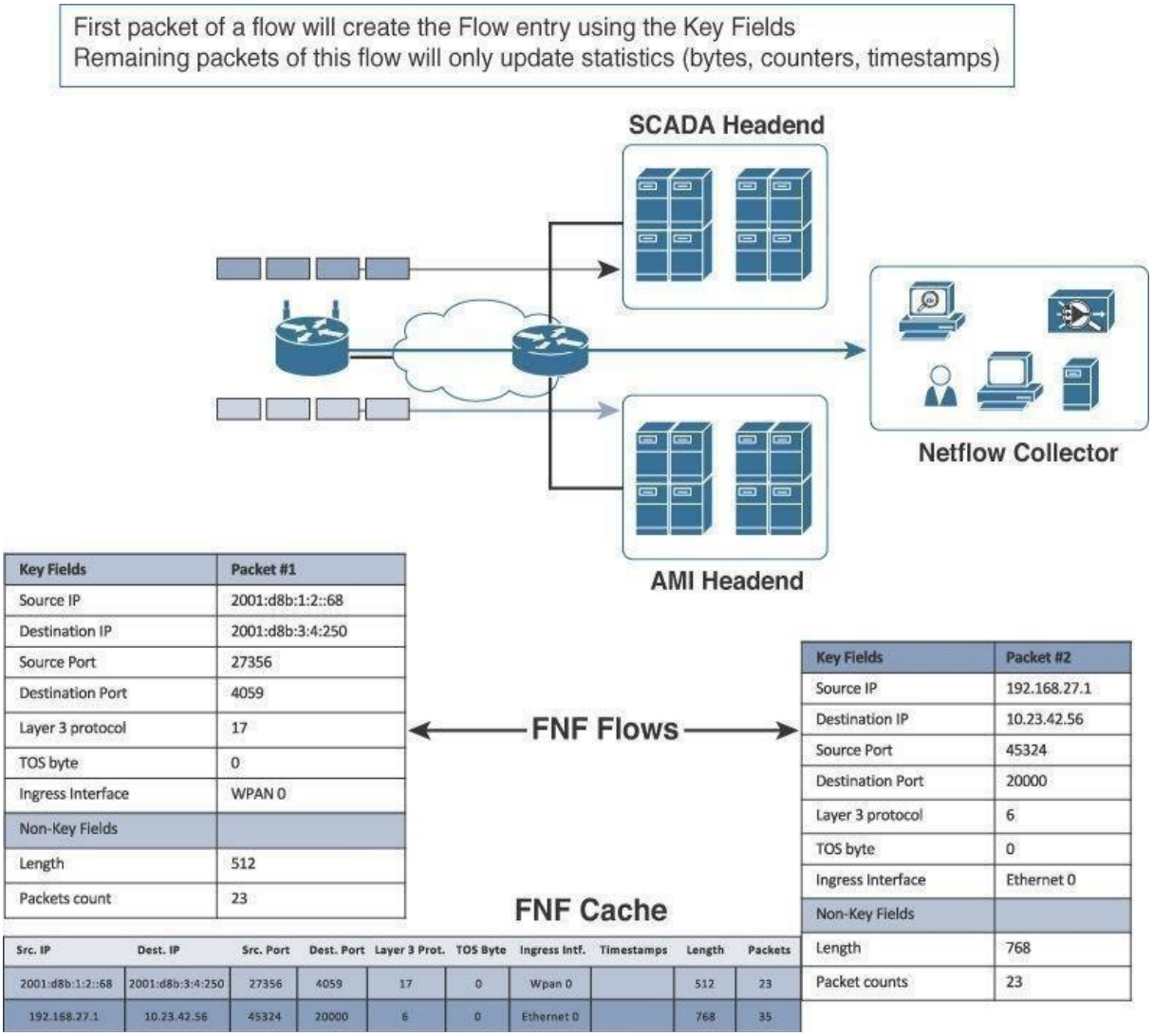
Flexible NetFlow (FNF) and IETF IPFIX (RFC 5101, RFC 5102) are examples of protocols that are widely used for networks. This section examines the fundamentals of FNF and how it may be used in an IoT deployment.

FNF is a flow technology developed by Cisco Systems that is widely deployed all over the world. Key advantages of FNF are as follows:

- Flexibility, scalability, and aggregation of flow data
- Ability to monitor a wide range of packet information and produce new information about network behavior
- Enhanced network anomaly and security detection
- User-configurable flow information for performing customized traffic identification and ability to focus and monitor specific network behavior
- Convergence of multiple accounting technologies into one accounting mechanism

### FNF Components

FNF has the following main components, as shown in Figure 7-17:




| Key Fields | Packet #1 |
|---|---|
| Source IP | 2001:d8b:1:2::68 |
| Destination IP | 2001:d8b:3:4:250 |
| Source Port | 27356 |
| Destination Port | 4059 |
| Layer 3 protocol | 17 |
| TOS byte | 0 |
| Ingress Interface | WPAN 0 |
| Non-Key Fields | |
| Length | 512 |
| Packets count | 23 |

| Key Fields | Packet #2 |
|---|---|
| Source IP | 192.168.27.1 |
| Destination IP | 10.23.42.56 |
| Source Port | 45324 |
| Destination Port | 20000 |
| Layer 3 protocol | 6 |
| TOS byte | 0 |
| Ingress Interface | Ethernet 0 |
| Non-Key Fields | |
| Length | 768 |
| Packet counts | 23 |

| Src. IP | Dest. IP | Src. Port | Dest. Port | Layer 3 Prot. | TOS Byte | Ingress Intf. | Timestamps | Length | Packets |
|---|---|---|---|---|---|---|---|---|---|
| 2001:d8b:1:2::68 | 2001:d8b:3:4:250 | 27356 | 4059 | 17 | 0 | Wpan 0 | | 512 | 23 |
| 192.168.27.1 | 10.23.42.56 | 45324 | 20000 | 6 | 0 | Ethernet 0 | | 768 | 35 |

**Figure 7-17** Flexible NetFlow overview

- **FNF Flow Monitor (NetFlow cache):** The FNF Flow Monitor describes the NetFlow cache or information stored in the cache. The Flow Monitor contains the flow record definitions with key fields (used to create a flow, unique per flow record: match statement) and non-key fields (collected with the flow as attributes or characteristics of a flow) within the cache. Also, part of the Flow Monitor is the Flow Exporter, which contains information about the export of NetFlow information, including the destination address of the NetFlow collector. The Flow Monitor includes various cache characteristics, including timers for exporting, the size of the cache, and, if required, the packet sampling rate.

- **FNF flow record:** A flow record is a set of key and non-key NetFlow field values used to characterize flows in the NetFlow cache. Flow records may be predefined for ease of use or customized and user

defined. A typical predefined record aggregates flow data and allows users to target common applications for NetFlow. User-defined records allow selections of specific key or non-key fields in the flow record.

The user-defined field is the key to Flexible NetFlow, allowing a wide range of information to be characterized and exported by NetFlow. It is expected that different network management applications will support specific user-defined and predefined flow records based on what they are monitoring (for example, security detection, traffic analysis, capacity planning).

- **FNF Exporter:** There are two primary methods for accessing NetFlow data: Using the **show** commands at the command-line interface (CLI), and using an application reporting tool. NetFlow Export, unlike SNMP polling, pushes information periodically to the NetFlow reporting collector. The Flexible NetFlow Exporter allows the user to define where the export can be sent, the type of transport for the export, and properties for the export. Multiple exporters can be configured per Flow Monitor.
- **Flow export timers:** Timers indicate how often flows should be exported to the collection and reporting server.
- **NetFlow export format:** This simply indicates the type of flow reporting format.
- **NetFlow server for collection and reporting:** This is the destination of the flow export. It is often done with an analytics tool that looks for anomalies in the traffic patterns.

Figure 7-18 illustrates the analysis reported from the FNF records on a smart grid FAN. In this example, the FNF collector is able to see the patterns of traffic for various applications as well as management traffic on the FAN.

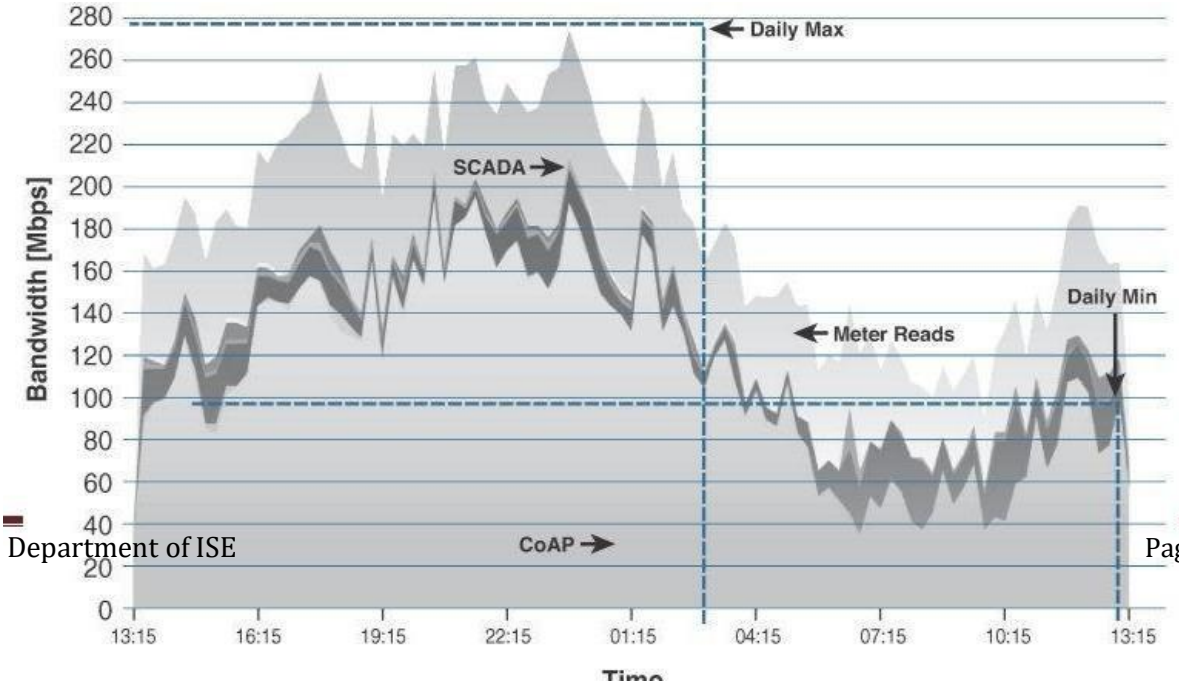


**Figure 7-18** FNF Report of Traffic on a Smart Grid FAN

### Flexible NetFlow in Multiservice IoT Networks

In the context of multiservice IoT networks, it is recommended that FNF be configured on the routers that aggregate connections from the last mile's routers. This gives a global view of all services flowing between the core network in the cloud and the IoT last-mile network (although not between IoT devices). FNF can also be configured on the last-mile gateway or fog nodes to provide more granular visibility. However, care must be taken in terms of how much northbound data is consumed through reporting.

A similar problem is encountered when using an MQTT server that sends data through an IoT broker. Some other challenges with deploying flow analytics tools in an IoT network include the following:

- The distributed nature of fog and edge computing may mean that traffic flows are processed in places that might not support flow analytics, and visibility is thus lost.
- IPv4 and IPv6 native interfaces sometimes need to inspect inside VPN tunnels, which may impact the router's performance.
- Additional network management traffic is generated by FNF reporting devices. The added cost of increasing bandwidth thus needs to be reviewed, especially if the backhaul network uses cellular or satellite communications.

# Chapter 8. Securing IoT

It is often said that if World War III breaks out, it will be fought in cyberspace. As IoT brings more and more systems together under the umbrella of network connectivity, security has never been more important. From the electrical grid system that powers our world, to the lights that control the flow of traffic in a city, to the systems that keep airplanes flying in an organized and efficient way, security of the networks, devices, and the applications that use them is foundational and essential for all modern communications systems. Providing security in such a world is not easy.

Security is among the very few, if not the only, technology disciplines that must operate with external forces continually working against desired outcomes. To further complicate matters, these external forces are able to leverage traditional technology as well as nontechnical methods (for example, physical security, operational processes, and so on) to meet their goals. With so many potential attack vectors, information and cyber security is a challenging, but engaging, topic that is of critical importance to technology vendors, enterprises, and service providers alike.

Information technology (IT) environments have faced active attacks and information security threats for many decades, and the incidents and lessons learned are well-known and documented. By contrast, operational technology (OT) environments were traditionally kept in silos and had only limited connection to other networks. Thus, the history of cyber-attacks on OT systems is much shorter and has far fewer incidents documented. Therefore, the learning opportunities and the body of cataloged incidents with their corresponding mitigations are not as rich as in the IT world. Security in the OT world also addresses a wider scope than in the IT world. For example, in OT, the word security is almost synonymous with safety. In fact, many of the industrial security standards that form the foundation for industrial IoT security also incorporate equipment and personnel safety recommendations.

It is for these reasons that this chapter focuses on the core principles of securing OT environments. IT security is a vast domain with many books dedicated to its various aspects. An exhaustive treatment of the subject is simply not possible in one chapter, so we instead focus on OT security and the

elements of IT security that are fundamental to OT security. In addition, the industry-specific chapters in Part III, ―IoT in Industry,‖ discuss the application of security to specific industry verticals.

This chapter provides a historical perspective of OT security, how it has evolved, and some of the common challenges it faces. It also details some of

The key differences between securing IT and OT environments. Finally, this chapter explores a number of practical steps for creating a more secure industrial environment, including best practices in introducing modern IT network security into legacy industrial environments. It includes the following sections:

- **A Brief History of OT Security:** This section provides an overview of how OT environments have evolved and the impact that the evolution has had on securing operational networks.
- **Common Challenges in OT Security:** This section provides a synopsis of different security challenges in operational environments, including legacy systems and insecure protocols and assets.
- **How IT and OT Security Practices and Systems Vary:** This section provides a comparison between the security practices in enterprise IT environments and operational industrial environments.
- **Formal Risk Analysis Structures: OCTAVE and FAIR:** This section provides a holistic view of securing an operational environment and a risk assessment framework that includes the people, processes, and vendor ecosystem components that make up a control system.
- **The Phased Application of Security in an Operational Environment:** This section provides a description of a phased approach to introducing modern network security into largely preexisting legacy industrial networks.

## A Brief History of OT Security

To better understand the current situation in industrial environments, it is important to differentiate between assumptions and realities. Few topics in information technology inspire more fear, uncertainty, or doubt than cybersecurity. This chapter is therefore limited to incidents and data sources from official sources rather than public media reports or uncorroborated third-party accounts.

More than in most other sectors, cybersecurity incidents in industrial environments can result in physical consequences that can cause threats to human lives as well as damage to equipment, infrastructure, and the environment. While there are certainly traditional IT-related security threats in industrial environments, it is the physical manifestations and impacts of the OT security incidents that capture media attention and elicit broad-based public concern.

One example of a reported incident where physical damage was caused by a

cybersecurity attack is the Stuxnet malware that damaged uranium enrichment systems in Iran. Another example is an event that damaged a furnace in a German smelter. In both incidents, multiple steps led to the undesirable outcomes. Many of the security policies and mitigation procedures that were in place went unheeded; however, if properly implemented, they could have impeded or possibly stopped the attacks entirely. For example, Stuxnet is thought to have been deployed on USB memory sticks up to two years before it was finally identified and discovered.

In addition to physical damage, operational interruptions have occurred in OT environments due to cybersecurity incidents. For example, in 2000, the sewage control system of Maroochy Shire in Queensland, Australia, was accessed remotely, and it released 800,000 liters of sewage into the surrounding waterways. In 2015, the control systems of the Ukrainian power distribution operator Kyiv Oblenergo were remotely accessed by attackers, causing an outage that lasted several hours and resulted in days of degraded service for thousands of customers. In both cases, known mitigation techniques could have been applied to detect the attacks earlier or block the ability to hijack production systems and affect service.

Historically, attackers were skilled individuals with deep knowledge of technology and the systems they were attacking. However, as technology has advanced, tools have been created to make attacks much easier to carry out. To further complicate matters, these tools have become more broadly available and more easily obtainable. Compounding this problem, many of the legacy protocols used in IoT environments are many decades old, and there was no thought of security when they were first developed. This means that attackers with limited or no technical capabilities now have the potential to launch cyber attacks, greatly increasing the frequency of attacks and the overall threat to end operators. It is, however, a common misconception that attackers always have the advantage and that end operators lack effective defensive capabilities. An important advantage for operators is the fact that they are far more familiar with their environment and have a better understanding of their processes, and can thus leverage multiple technologies and capabilities to defend their networks against attack. This is critical as networks will continue to face ever-evolving and changing methods of attack that will be increasingly difficult to defend against and respond to.

Communication networks, both local and geographically dispersed, have been used in industrial environments for decades. For example, remote monitoring of substations in utilities and communications between semi-autonomous systems in manufacturing are long-standing examples of such OT networks. These OT-specific communication systems have typically been standalone

and physically isolated from the traditional IT enterprise networks in the same companies. While it follows the traditional logic of ―security through obscurity,‖ this form of network compartmentalization has led to the independent evolution of IT and OT networks, with interconnections between the environments strictly segregated and monitored.

The isolation between industrial networks and the traditional IT business networks has been referred to as an ―air gap,‖ suggesting that there are no links between the two. While there are clearly examples of such extreme isolation in some industries, it is actually not an accurate description of most IoT networks today. Broadly speaking, there is a varying amount of interconnection between OT and IT network environments, and many interdependencies between the two influence the level of interconnection.

In addition to the policies, regulations, and governance imposed by the different industrial environments, there is also a certain amount of end-user preference and deployment-specific design that determines the degree of isolation between IT and OT environments. While some organizations continue to maintain strict separation, others are starting to allow certain elements of interconnection. One common example of this is the use of Ethernet and IP to transport control systems in industrial environments. As much as IT and OT networks are still operated and managed separately in a good portion of the world, the prevailing trend is to consolidate networks based on IT-centric technologies such as TCP/IP, Ethernet, and common APIs.

This evolution of ever-increasing IT technologies in the OT space comes with the benefits of increased accessibility and a larger base of skilled operators than with the nonstandard and proprietary communication methods in traditional industrial environments. The challenges associated with these well-known IT standards is that security vulnerabilities are more widely known, and abuse of those systems is often easier and occurs on a much larger scale. This accessibility and scale makes security a major concern, particularly because many systems and devices in the operational domain were never envisioned to run on a shared, open standards–based infrastructure, and they were not designed and developed with high levels of built-in security capabilities.

Projects in industrial environments are often capital intensive, with an expected life span that can be measured in decades. Unlike in IT-based enterprises, OT-deployed solutions commonly have no reason to change as they are designed to meet specific (and often single-use) functions, and have no requirements or incentives to be upgraded. A huge focus and priority in OT is system uptime and high availability, so changes are typically only made

to fix faults or introduce new system capabilities in support of that goal. As a result, deployed OT systems often have slower development and upgrade cycles and can quickly become out of sync with traditional IT network environments. The outcome is that both OT technologies and the knowledge of those looking after those operational systems have progressed at a slower pace than their IT counterparts.

Most of the industrial control systems deployed today, their components, and the limited associated security elements were designed when adherence to published and open standards were rare. The proprietary nature of these systems meant that threats from the outside world were unlikely to occur and were rarely addressed. There has, however, been a growing trend whereby OT system vulnerabilities have been exposed and reported. This increase is depicted in Figure 8-1, which shows the history of vulnerability disclosures in industrial control systems (ICSs) since 2010. While the number of reports has been increasing over the past years, it is likely that there are still many others that are not reported or discovered.

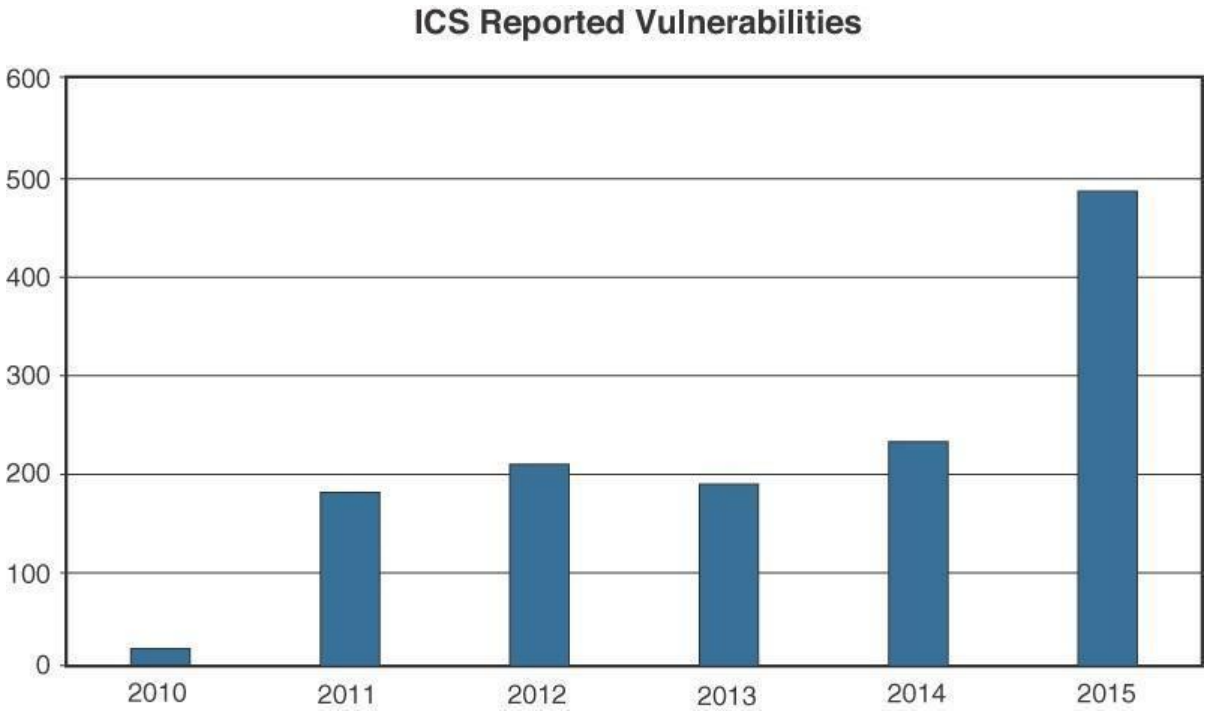



**Figure 8-1** History of Vulnerability Disclosures in Industrial Control Systems Since 2010 (US Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) https://ics-cert.us-cert.gov).

## Erosion of Network Architecture

Two of the major challenges in securing industrial environments have been initial design and ongoing maintenance. The initial design challenges arose

from the concept that networks were safe due to physical separation from the enterprise with minimal or no connectivity to the outside world, and the assumption that attackers lacked sufficient knowledge to carry out security attacks. In many cases, the initial network design is sound and even follows well-defined industrial best practices and standards, such as the Purdue Model for Control Hierarchy that was introduced in Chapter 2, ―IoT Network Architecture and Design.‖ The challenge, and the biggest threat to network security, is standards and best practices either being misunderstood or the network being poorly maintained. In fact, from a security design perspective, it is better to know that communication paths are insecure than to not know the actual communication paths. It is more common that, over time, what may have been a solid design to begin with is eroded through ad hoc updates and individual changes to hardware and machinery without consideration for the broader network impact. This kind of organic growth has led to miscalculations of expanding networks and the introduction of wireless communication in a standalone fashion, without consideration of the impact to the original security design. These uncontrolled or poorly controlled OT network evolutions have, in many cases, over time led to weak or inadequate network and systems security.

There is a wide variety in secured network designs within and across different industries. For example, power utilities have a strong history of leveraging modern technologies for operational activities, and in North America there are regulatory requirements in place from regulatory authorities, such as North American Electric Reliability Corporation's (NERC's) Critical Infrastructure Protection (CIP), discussed in greater detail in Chapter 11, ―Utilities‖), to implement secure network connectivity and control with reasonably prescriptive actions. By contrast, in other industries, there are often no legislative requirements or compliance policies, which has resulted in widespread differences in security capabilities.

## Pervasive Legacy Systems

Due to the static nature and long lifecycles of equipment in industrial environments, many operational systems may be deemed legacy systems. For example, in a power utility environment, it is not uncommon to have racks of old mechanical equipment still operating alongside modern intelligent electronic devices (IEDs). In many cases, legacy components are not restricted to isolated network segments but have now been consolidated into the IT operational environment. From a security perspective, this is potentially dangerous as many devices may have historical vulnerabilities or weaknesses that have not been patched and updated, or it may be that patches are not even available due to the age of the equipment.

Beyond the endpoints, the communication infrastructure and shared centralized compute resources are often not built to comply with modern standards. In fact, their communication methods and protocols may be generations old and must be interoperable with the oldest operating entity in the communications path. This includes switches, routers, firewalls, wireless access points, servers, remote access systems, patch management, and network management tools.

## Insecure Operational Protocols

Many industrial control protocols, particularly those that are serial based, were designed without inherent strong security requirements. Furthermore, their operation was often within an assumed secure network. In addition to any inherent weaknesses or vulnerabilities, their operational environment may not have been designed with secured access control in mind.

Industrial protocols, such as supervisory control and data acquisition (SCADA) (refer to Chapter 6, ―Application Protocols for IoT‖), particularly the older variants, suffer from common security issues. Three examples of this are a frequent lack of authentication between communication endpoints, no means of securing and protecting data at rest or in motion, and insufficient granularity of control to properly specify recipients or avoid default broadcast approaches. These may not be as critical in self-contained systems, but between zones or on longer network segments, such as a WAN (particularly a public WAN), they may be significant considerations.

The structure and operation of most of these protocols is often publicly available. While they may have been originated by a private firm, for the sake of interoperability, they are typically published for others to implement. Thus, it becomes a relatively simple matter to compromise the protocols themselves and introduce malicious actors that may use them to compromise control systems for either reconnaissance or attack purposes that could lead to undesirable impacts in normal system operation.

### Modbus

Modbus is commonly found in many industries, such as utilities and manufacturing environments, and has multiple variants (for example, serial, TCP/IP). It was created by the first programmable logic controller (PLC) vendor, Modicon, and has been in use since the 1970s. It is one of the most widely used protocols in industrial deployments, and its development is governed by the Modbus Organization.

The security challenges that have existed with Modbus are not unusual. Authentication of communicating endpoints was not a default operation

because it would allow an inappropriate source to send improper commands to the recipient. For example, for a message to reach its destination, nothing more than the proper Modbus address and function call (code) is necessary.

Some older and serial-based versions of Modbus communicate via broadcast. The ability to curb the broadcast function does not exist in some versions. There is potential for a recipient to act on a command that was not specifically targeting it. Furthermore, an attack could potentially impact unintended recipient devices, thus reducing the need to understand the details of the network topology.

Validation of the Modbus message content is also not performed by the initiating application. Instead, Modbus depends on the network stack to perform this function. This could open up the potential for protocol abuse in the system.

**DNP3 (Distributed Network Protocol)**

DNP3 is found in multiple deployment scenarios and industries. It is common in utilities and is also found in discrete and continuous process systems. Like many other ICS/SCADA protocols, it was intended for serial communication between controllers and simple IEDs.

There is an explicit ―secure‖ version of DNP3, but there also remain many insecure implementations of DNP3 as well. DNP3 has placed great emphasis on the reliable delivery of messages. That emphasis, while normally highly desirable, has a specific weakness from a security perspective. In the case of DNP3, participants allow for unsolicited responses, which could trigger an undesired response. The missing security element here is the ability to establish trust in the system's state and thus the ability to trust the veracity of the information being presented. This is akin to the security flaws presented by Gratuitous ARP messages in Ethernet networks, which has been addressed by Dynamic ARP Inspection (DAI) in modern Ethernet switches.

**ICCP (Inter-Control Center Communications Protocol)**

ICCP is a common control protocol in utilities across North America that is frequently used to communicate between utilities. Given that it must traverse the boundaries between different networks, it holds an extra level of exposure and risk that could expose a utility to cyber attack.

Unlike other control protocols, ICCP was designed from inception to work across a WAN. Despite this role, initial versions of ICCP had several significant gaps in the area of security. One key vulnerability is that the system did not require authentication for communication. Second, encryption across the protocol was not enabled as a default condition, thus exposing

connections to man-in-the-middle (MITM) and replay attacks.

### OPC (OLE for Process Control)

OPC is based on the Microsoft interoperability methodology Object Linking and Embedding (OLE). This is an example where an IT standard used within the IT domain and personal computers has been leveraged for use as a control protocol across an industrial network.

In industrial control networks, OPC is limited to operation at the higher levels of the control space, with a dependence on Windows-based platforms. Concerns around OPC begin with the operating system on which it operates. Many of the Windows devices in the operational space are old, not fully patched, and at risk due to a plethora of well-known vulnerabilities. The dependence on OPC may reinforce that dependence. While newer versions of OPC have enhanced security capabilities, they have also opened up new communications modes, which have both positive and negative security potential Of particular concern with OPC is the dependence on the Remote Procedure Call (RPC) protocol, which creates two classes of exposure. The first requires you to clearly understand the many vulnerabilities associated with RPC, and the second requires you to identify the level of risk these vulnerabilities bring to a specific network.

### International Electrotechnical Commission (IEC) Protocols

The IEC 61850 standard was created to allow vendor-agnostic engineering of power utility systems, which would, in turn, allow interoperability between vendors and standardized communication protocols. Three message types were initially defined: MMS (Manufacturing Message Specification), GOOSE (Generic Object Oriented Substation Event), and SV (Sampled Values). Web services was a fourth protocol that was added later. Here we provide a short summary of each, but for more information on IEC protocols, see Chapter 11:

- **MMS (61850-8.1):** MMS is a client/server protocol that leverages TCP/IP and operates at Layer 3. It provides the same functionality as other SCADA protocols, such as IEC 60870 and Modbus.
- **GOOSE (61850-8.1):** GOOSE is a Layer 2 protocol that operates via multicast over Ethernet. It allows IEDs to exchange data ―horizontally,‖ between bays and between substations, especially for interlocking, measurement, and tripping signals.
- **SV (61850-9-2):** SV is a Layer 2 protocol that operates via multicast over Ethernet. It carries voltage and current samples, typically on the process bus, but it can also flow over the station bus.

Both GOOSE and SV operate via a publisher/subscriber model, with no reliability mechanism to ensure that data has been received.

When the standard was first released, there was minimal security capability in these protocols, but this is being addressed by IEC 62351 with the introduction of well-known IT-based security measures, such as certificate exchange.

IEC 60870 is widely used for SCADA telecontrol in Europe, particularly in the power utility industry, and for widely geographically dispersed control systems. Part 5 of the standard outlines the communication profiles used between endpoints to exchange telecontrol messages. 60870-5-101 is the serial implementation profile, 60870-5-104 is the IP implementation profile, and 60870-5-103 is used for protection equipment. Again, in the early iterations of IEC 60870-5, security was lacking. This is now being addressed by IEC 62351, with the 60870-5-7 security extensions work, applicable to 60870-101 and 60870-104.

## Other Protocols

At times, discussions about the security of industrial systems are decidedly focused on industrial control protocols as if they were the sum total of what would be observed or considered. This assumption is narrow-minded and problematic on many levels. In fact, it is highly recommended that a security practitioner passively identify all aspects of the traffic traversing the network prior to implementing any kind of controls or security measures therein. Of particular importance are proper accounting, handling, and understanding of the most basic protocols, transport mechanisms, and foundational elements of any network, including ARP, UDP, TCP, IP, and SNMP.

Some specialized environments may also have other background control protocols. For example, many IoT networks reach all the way to the individual sensors, so protocols such as Constrained Application Protocol (CoAP) (see Chapter 6) and Datagram Transport Layer Security (DTLS) are used, and have to be considered separately from a security perspective.

## Device Insecurity

Beyond the communications protocols that are used and the installation base of legacy systems, control and communication elements themselves have a history of vulnerabilities. As mentioned earlier in this chapter (see Figure 8-1), prior to 2010, the security community paid little attention to industrial compute, and as a result, OT systems have not gone through the same ―trial by fire‖ as IT systems. Figure 8-2 shows this graphically by simply overlaying the count of industrial security topics presented at the Black Hat

security conference with the number of vulnerabilities reported for industrial control systems. The correlation between presentations on the subject of OT security at Black Hat and the number of vulnerabilities discovered is obvious, including the associated slowing of discoveries.

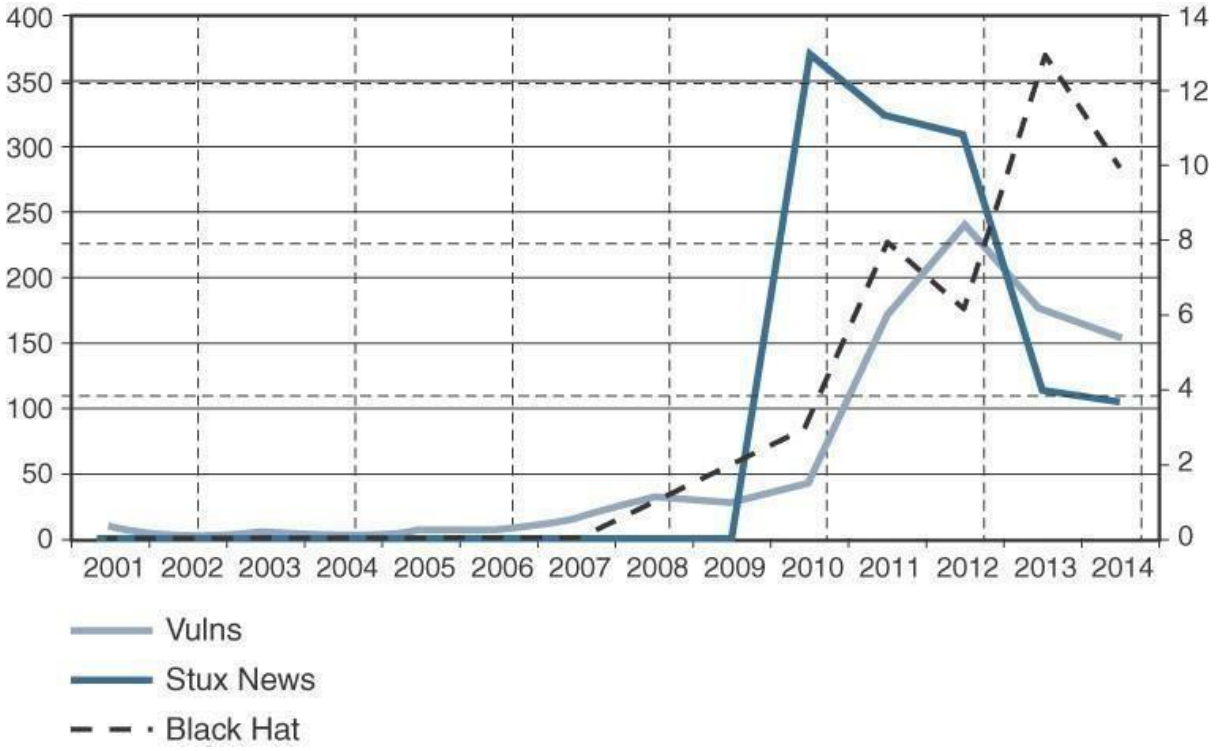



**Figure 8-2** Correlation of Industrial Black Hat Presentations with Discovered Industrial Vulnerabilities (US Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) https://ics-cert.us-cert.gov).

To understand the nature of the device insecurity, it is important to review the history of what vulnerabilities were discovered and what types of devices were affected. A review of the time period 2000 to 2010 reveals that the bulk of discoveries were at the higher levels of the operational network, including control systems trusted to operate plants, transmission systems, oil pipelines, or whatever critical function is in use.

It is not difficult to understand why such systems are frequently found vulnerable. First, many of the systems utilize software packages that can be easily downloaded and worked against. Second, they operate on common hardware and standard operating systems, such as Microsoft Windows. Third, Windows and the components used within those applications are well known to traditionally IT-focused security researchers. There is little need to develop new tools or techniques when those that have long been in place are sufficiently adequate to breach the target's defenses. For example, Stuxnet, the most famous of the industrial compute-based attacks, was initially successful because it was able to exploit a previously unknown vulnerability in Windows.

The ICS vendor community is also lagging behind IT counterparts with regard to security capabilities and practices, as well as cooperation with third-party security researchers. That said, this situation is beginning to get

significant industry focus and is improving through a number of recent initiatives designed to formally address security vulnerability and system testing in the industrial environment. While there are some formal standards, such as ISO/IEC 15408 (Common Criteria), ISO/IEC 19790, and a few others, there remain few formal security testing entities. Beyond formal testing, there is little regulatory enforcement of common criteria that address device security testing.

It was not too long ago that the security research community was viewed as a threat, rather than as a valued and often free service to expose potential dangers. While the situation has improved, operational efforts still significantly lag behind IT-based initiatives, such as bug bounty reward programs and advanced vulnerability preparation programs, along the lines of something like the Microsoft Active Protections Program (MAPP). To go a step further, in the industrial realm, there aren't even parallels to the laws that protect individuals' private data. While many states and countries require notification if an individual's personal and financial data is possibly exposed, outside the electrical utility industry, very few laws require the reporting of incidents that may have put lives at risk.

## Dependence on External Vendors

While modern IT environments may be outsourcing business operations or relegating certain processing or storage functions to the cloud, it is less common for the original equipment manufacturers of the IT hardware assets to be required to operate the equipment. However, that level of vendor dependence is not uncommon in some industrial spaces.

Direct and on-demand access to critical systems on the plant floor or in the field are sometimes written directly into contracts or are required for valid product warranties. This has clear benefits in many industries as it allows vendors to remotely manage and monitor equipment and to proactively alert the customer if problems are beginning to creep in. While contracts may be written to describe equipment monitoring and management requirements with explicit statements of what type of access is required and under what conditions, they generally fail to address questions of shared liability for security breaches or processes to ensure communication security.

## Security Knowledge

In the industrial operations space, the technical investment is primarily in connectivity and compute. It has seen far less investment in security relative to its IT counterpart. According to the research firm Infonetics, the industrial

firewall market in 2015 was only approximately 4% the size of the overall firewall market.

Another relevant challenge in terms of OT security expertise is the comparatively higher age of the industrial workforce. According to a study by the US Bureau of Labor, in North America the average age gap between manufacturing workers and other non-farm workers doubled between 2000 and 2012, and the trend shows no sign of reversing. Simultaneously, new connectivity technologies are being introduced in OT industrial environments that require up-to-date skills, such as TCP/IP, Ethernet, and wireless that are quickly replacing serial-based legacy technologies. The rapid expansion of extended communications networks and the need for an industrial controls-aware workforce creates an equally serious gap in security awareness.

This gap in OT security knowledge is actively being addressed. Education for industrial security environments has grown steadily, particularly in the electrical utility space, where regulations such as NERC CIP (CIP 004) and IEC 62351 (01) require ongoing training.

## How IT and OT Security Practices and Systems Vary

The differences between an enterprise IT environment and an industrial-focused OT deployment are important to understand because they have a direct impact on the security practice applied to them. Some of these areas are touched on briefly earlier in this chapter, and they are more explicitly discussed in the following sections.

## The Purdue Model for Control Hierarchy

Regardless of where a security threat arises, it must be consistently and unequivocally treated. IT information is typically used to make business decisions, such as those in process optimization, whereas OT information is instead characteristically leveraged to make physical decisions, such as closing a valve, increasing pressure, and so on. Thus, the operational domain must also address physical safety and environmental factors as part of its security strategy—and this is not normally associated with the IT domain.

Organizationally, IT and OT teams and tools have been historically separate, but this has begun to change, and they have started to converge, leading to more traditionally IT-centric solutions being introduced to support operational activities. For example, systems such as firewalls and intrusion prevention systems (IPS) are being used in IoT networks.

As the borders between traditionally separate OT and IT domains blur, they must align strategies and work more closely together to ensure end-to-end

security. The types of devices that are found in industrial OT environments are typically much more highly optimized for tasks and industrial protocol-specific operation than their IT counterparts. Furthermore, their operational profile differs as well.

Industrial environments consist of both operational and enterprise domains. To understand the security and networking requirements for a control system, the use of a logical framework to describe the basic composition and function is needed. The Purdue Model for Control Hierarchy, introduced in Chapter 2, is the most widely used framework across industrial environments globally and is used in manufacturing, oil and gas, and many other industries. It segments devices and equipment by hierarchical function levels and areas and has been incorporated into the ISA99/IEC 62443 security standard, as shown in Figure 8-3. For additional detail on how the Purdue Model for Control Hierarchy is applied to the manufacturing and oil and gas industries, see Chapter 9, ―Manufacturing,‖ and Chapter 10, ―Oil and Gas.‖


**Figure 8-3** The Logical Framework Based on the Purdue Model for Control Hierarchy

This model identifies levels of operations and defines each level. The enterprise and operational domains are separated into different zones and kept in strict isolation via an industrial demilitarized zone (DMZ):

- Enterprise zone
  - **Level 5: Enterprise network:** Corporate-level applications such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), document management, and services such as Internet access and VPN entry from the outside world exist at this level.

**Level 4: Business planning and logistics network:** The IT services exist at this level and may include scheduling systems, material flow applications, optimization and planning systems, and local IT services such as phone, email, printing, and security monitoring.

- Industrial demilitarized zone
  - **DMZ:** The DMZ provides a buffer zone where services and data can be shared between the operational and enterprise zones. It also allows for easy segmentation of organizational control. By default, no traffic should traverse the DMZ; everything should originate from or terminate on this area.
- Operational zone
  - **Level 3: Operations and control:** This level includes the functions involved in managing the workflows to produce the desired end products and for monitoring and controlling the entire operational system. This could include production scheduling, reliability assurance, system wide control optimization, security management, network management, and potentially other required IT services, such as DHCP, DNS, and timing.
  - **Level 2: Supervisory control:** This level includes zone control rooms, controller status, control system network/application administration, and other control-related applications, such as human-machine interface (HMI) and historian.
  - **Level 1: Basic control:** At this level, controllers and IEDs, dedicated HMIs, and other applications may talk to each other to run part or all of the control function.
  - **Level 0: Process:** This is where devices such as sensors and actuators and machines such as drives, motors, and robots communicate with controllers or IEDs.
- **Safety zone**
  - **Safety-critical:** This level includes devices, sensors, and other equipment used to manage the safety functions of the control system.

One of the key advantages of designing an industrial network in structured levels, as with the Purdue model, is that it allows security to be correctly applied at each level and between levels. For example, IT networks typically reside at Levels 4 and 5 and use security principles common to IT networks. The lower levels are where the industrial systems and IoT networks reside. As shown in Figure 8-3, a DMZ resides between the IT and OT levels. Clearly, to protect the lower industrial layers, security technologies such as firewalls,

proxy servers, and IPSs should be used to ensure that only authorized connections from trusted sources on expected ports are being used. At the DMZ, and, in fact, even between the lower levels, industrial firewalls that are capable of understanding the control protocols should be used to ensure the continuous operation of the OT network.

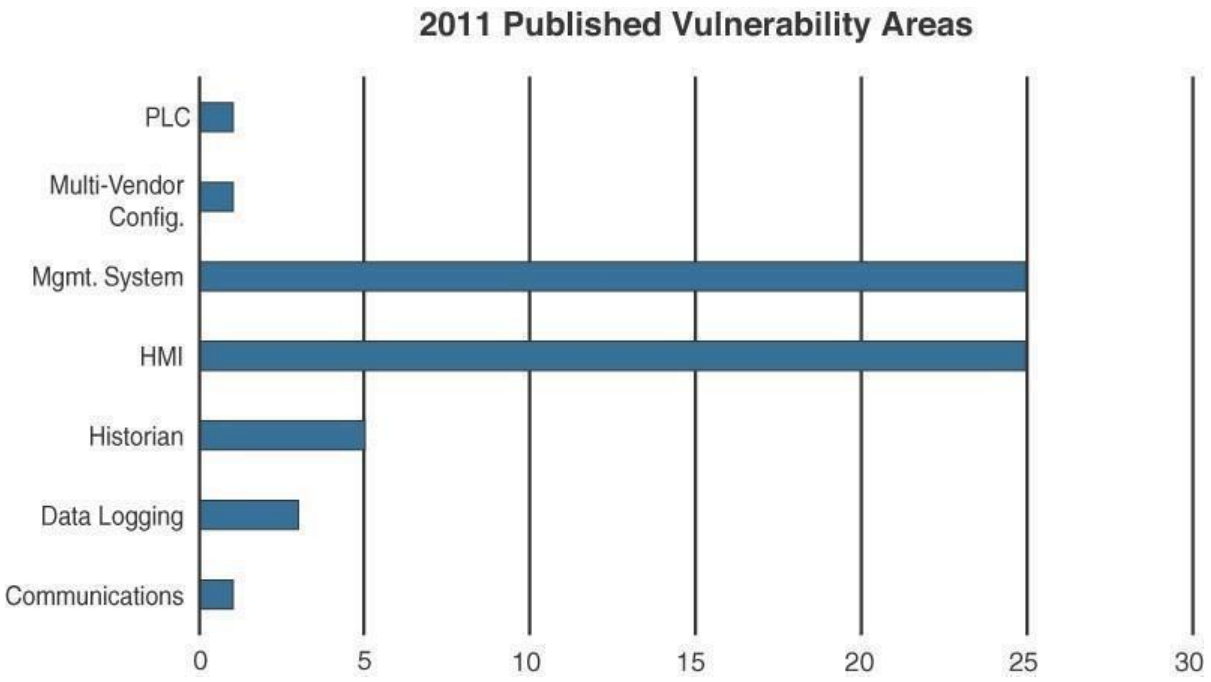



**Figure 8-4** 201 Industrial Security Report of Published Vulnerability Areas (US Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) https://ics-cert.us-cert.gov).

## OT Network Characteristics Impacting Security

While IT and OT networks are beginning to converge, they still maintain many divergent characteristics in terms of how they operate and the traffic they handle. These differences influence how they are treated in the context of a security strategy. For example, compare the nature of how traffic flows across IT and OT networks:

- **IT networks:** In an IT environment, there are many diverse data flows. The communication data flows that emanate from a typical IT endpoint travel relatively far. They frequently traverse the network through layers of switches and eventually make their way to a set of local or remote servers, which they may connect to directly. Data in the form of email, file transfers, or print services will likely all make its way to the central data center, where it is responded to, or triggers actions in more local services, such as a printer. In the case of email or web browsing, the endpoint initiates actions that leave the confines of the enterprise network and potentially travel around the earth.

- **OT networks:** By comparison, in an OT environment (Levels 0–3), there are typically two types of operational traffic. The first is local traffic that may be contained within a specific package or area to provide local monitoring and closed-loop control. This is the traffic that is used for real-time (or near-real-time) processes and does not need to

leave the process control levels. The second type of traffic is used for monitoring and control of areas or zones or the overall system. SCADA traffic is a good example of this, where information about remote devices or summary information from a function is shared at a system level so that operators can understand how the overall system, or parts of it, are operating. They can then implement appropriate control commands based on this information.

When IT endpoints communicate, it is typically short and frequent conversations with many connections. The nature of the communications is open, and almost anybody can speak with anybody else, such as with email or browsing. Although there are clearly access controls, most of those controls are at the application level rather than the network level.

In an OT environment, endpoint communication is typically point-to-point, such as a SCADA master to SCADA slave, or uses multicast or broadcast, leveraging a publisher/subscriber type of model. Communication could be TCP or UDP or neither (as in the case of PROFINET, discussed in Chapter 9, ―Manufacturing‖).

IT networks are typically more mature and use up-to-date technologies. These mature modern networking practices are critical to meet the high degree of flexibility required in the IT environment. Virtual networking, virtual workspaces, and virtual servers are commonplace. It is likely that there are a wide variety of device types actively participating in any given network at any one time. Flexible interoperability is thus critical. To achieve interoperability, there is usually minimal proprietary communication activity, and the emphasis is typically on open standards. The movement to IPv6 continues to progress, and higher-order network services, such as quality of service (QoS), are normal as well. Endpoints are not just promiscuous in their communications, but they operate a wide number of applications from a large number of diverse vendors. The open nature of these compute systems means a wide range of protocols are traversing the OT network.

Industrial networks often still rely on serial communication technologies or have mixed serial and Ethernet. This means that not only do many devices lack IP capabilities, but it is not even possible to monitor and secure the serial traffic in the same way you do for IP or Ethernet. In some environments, the network remains very static, meaning a baseline of traffic patterns can be built up and monitored for changes. In static environments, the visibility of devices, protocols, and traffic flows can be managed and secured more easily. However, there is a continued growth of mobile devices and ad hoc connectivity, especially in industries such as transportation and smart cities, as well as a rise in mobile fleet assets across a plethora of other industries.

These dynamic and variable networks are much more difficult to baseline, monitor, and secure.

## Security Priorities: Integrity, Availability, and Confidentiality

Security priorities are driven by the nature of the assets in each environment. In an IT realm, the most critical element and the target of attacks has been information. In an OT realm, the critical assets are the process participants: workers and equipment. Security priorities diverge based on those differences.

In the IT business world, there are legal, regulatory, and commercial obligations to protect data, especially data of individuals who may or may not be employed by the organization. This emphasis on privacy focuses on the confidentiality, integrity, and availability of the data—not necessarily on a system or a physical asset. The impact of losing a compute device is considered minimal compared to the information that it could hold or provide access to. By way of comparison, in the OT world, losing a device due to a security vulnerability means production stops, and the company cannot perform its basic operation. Loss of information stored on these devices is a lower concern, but there are certainly confidential data sets in the operating environment that may have economic impacts, such as formulations and processes.

## Security Focus

Security focus is frequently driven by the history of security impacts that an organization has experienced. In an IT environment, the most painful experiences have typically been intrusion campaigns in which critical data is extracted or corrupted. The result has been a significant investment in capital goods and human power to reduce these external threats and minimize potential internal malevolent actors.

In the OT space, the history of loss due to external actors has not been as long, even though the potential for harm on a human scale is clearly significantly higher. The result is that the security events that have been experienced have come more from human error than external attacks. Interest and investment in industrial security have primarily been in the standard access control layers. Where OT has diverged, to some degree, is to emphasize the application layer control between the higher-level controller layer and the receiving operating layer. Later in this chapter you will learn more about the value and risks associated with this approach.

## Formal Risk Analysis Structures: OCTAVE and FAIR

Within the industrial environment, there are a number of standards, guidelines, and best practices available to help understand risk and how to mitigate it. IEC 62443 is the most commonly used standard globally across industrial verticals. It consists of a number of parts, including 62443-3-2 for risk assessments, and 62443-3-3 for foundational requirements used to secure the industrial environment from a networking and communications perspective. Also, ISO 27001 is widely used for organizational people, process, and information security management. In addition, the National Institute of Standards and Technology (NIST) provide a series of documents for critical infrastructure, such as the NIST Cyber security Framework (CSF). In the utilities domain, the North American Electric Reliability Corporation's (NERC's) Critical Infrastructure Protection (CIP) has legally binding guidelines for North American utilities, and IEC 62351 is the cyber security standard for power utilities.

The key for any industrial environment is that it needs to address security holistically and not just focus on technology. It must include people and processes, and it should include all the vendor ecosystem components that make up a control system.

In this section, we present a brief review of two such risk assessment frameworks:

- OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation) from the Software Engineering Institute at Carnegie Mellon University
- FAIR (Factor Analysis of Information Risk) from The Open Group

These two systems work toward establishing a more secure environment but with two different approaches and sets of priorities. Knowledge of the environment is key to determining security risks and plays a key role in driving priorities.

## OCTAVE

OCTAVE has undergone multiple iterations. The version this section focuses on is OCTAVE Allegro, which is intended to be a lightweight and less burdensome process to implement. Allegro assumes that a robust security team is not on standby or immediately at the ready to initiate a comprehensive security review. This approach and the assumptions it makes are quite appropriate, given that many operational technology areas are similarly lacking in security-focused human assets. Figure 8-5 illustrates the OCTAVE

Allegro steps and phases.

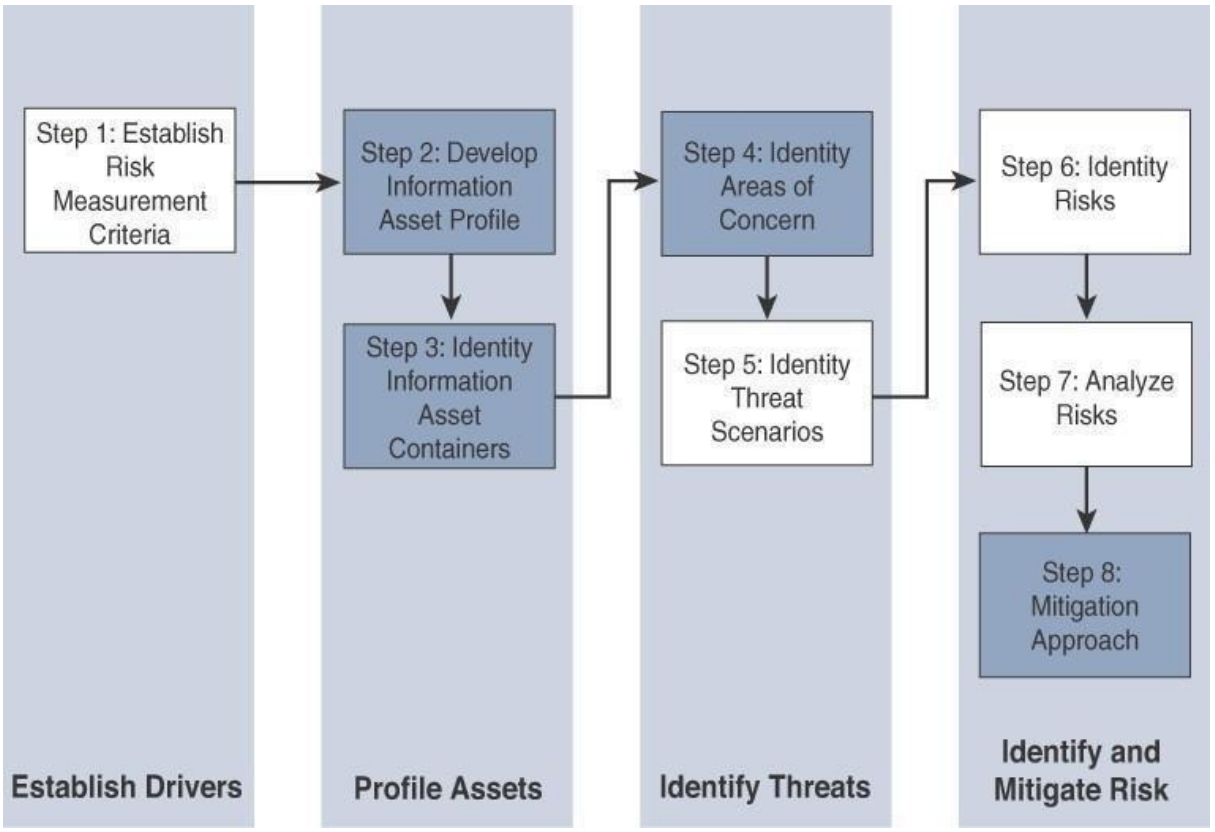



**Figure 8-5** OCTAVE Allegro Steps and Phases

The first step of the OCTAVE Allegro methodology is to establish a risk measurement criterion. OCTAVE provides a fairly simple means of doing this with an emphasis on impact, value, and measurement. The point of having a risk measurement criterion is that at any point in the later stages, prioritization can take place against the reference model. (While OCTAVE has more details to contribute, we suggest using the FAIR model, described next, for risk assessment.)

The second step is to develop an information asset profile. This profile is populated with assets, a prioritization of assets, attributes associated with each asset, including owners, custodians, people, explicit security requirements, and technology assets. It is important to stress the importance of process.
Certainly, the need to protect information does not disappear, but operational safety and continuity are more critical.

Within this asset profile, process are multiple sub stages that complete the definition of the assets. Some of these are simply survey and reporting activities, such as identifying the asset and attributes associated with it, such as its owners, custodians, human actors with which it interacts, and the composition of its technology assets. There are, however, judgment-based attributes such as prioritization. Rather than simply assigning an arbitrary ranking, the system calls for a justification of the prioritization. With an

understanding of the asset attributes, particularly the technical components, appropriate threat mitigation methods can be applied. With the application of risk assessment, the level of security investment can be aligned with that individual asset.

The third step is to identify information asset containers. Roughly speaking, this is the range of transports and possible locations where the information might reside. This references the compute elements and the networks by which they communicate. However, it can also mean physical manifestations such as hard copy documents or even the people who know the information. Note that the operable target here is information, which includes data from which the information is derived.

In OCTAVE, the emphasis is on the container level rather than the asset level. The value is to reduce potential inhibitors within the container for information operation. In the OT world, the emphasis is on reducing potential inhibitors in the containerized operational space. If there is some attribute of the information that is endemic to it, then the entire container operates with that attribute because the information is the defining element. In some cases this may not be true, even in IT environments. Discrete atomic-level data may become actionable information only if it is seen in the context of the rest of the data. Similarly, operational data taken without knowledge of the rest of the elements may not be of particular value either.

The fourth step is to identify areas of concern. At this point, we depart from a data flow, touch, and attribute focus to one where judgments are made through a mapping of security-related attributes to more business-focused use cases. At this stage, the analyst looks to risk profiles and delves into the previously mentioned risk analysis. It is no longer just facts, but there is also an element of creativity that can factor into the evaluation. History both within and outside the organization can contribute. References to similar operational use cases and incidents of security failures are reasonable associations.

Closely related is the fifth step, where threat scenarios are identified. Threats are broadly (and properly) identified as potential undesirable events. This definition means that results from both malevolent and accidental causes are viable threats. In the context of operational focus, this is a valuable consideration. It is at this point that an explicit identification of actors, motives, and outcomes occurs. These scenarios are described in threat trees to trace the path to undesired outcomes, which, in turn, can be associated with risk metrics.

At the sixth step risks are identified. Within OCTAVE, risk is the possibility of an undesired outcome. This is extended to focus on how the organization is

impacted. For more focused analysis, this can be localized, but the potential impact to the organization could extend outside the boundaries of the operation.

The seventh step is risk analysis, with the effort placed on qualitative evaluation of the impacts of the risk. Here the risk measurement criteria defined in the first step are explicitly brought into the process.

Finally, mitigation is applied at the eighth step. There are three outputs or decisions to be taken at this stage. One may be to accept a risk and do nothing, other than document the situation, potential outcomes, and reasons for accepting the risk. The second is to mitigate the risk with whatever control effort is required. By walking back through the threat scenarios to asset profiles, a pairing of compensating controls to mitigate those threat/risk pairings should be discoverable and then implemented. The final possible action is to defer a decision, meaning risk is neither accepted nor mitigated. This may imply further research or activity, but it is not required by the process.

OCTAVE is a balanced information-focused process. What it offers in terms of discipline and largely unconstrained breadth, however, is offset by its lack of security specificity. There is an assumption that beyond these steps are seemingly means of identifying specific mitigations that can be mapped to the threats and risks exposed during the analysis process.

## FAIR

FAIR (Factor Analysis of Information Risk) is a technical standard for risk definition from The Open Group. While information security is the focus, much as it is for OCTAVE, FAIR has clear applications within operational technology. Like OCTAVE, it also allows for non-malicious actors as a potential cause for harm, but it goes to greater lengths to emphasize the point. For many operational groups, it is a welcome acknowledgement of existing contingency planning. Unlike with OCTAVE, there is a significant emphasis on naming, with risk taxonomy definition as a very specific target.

FAIR places emphasis on both unambiguous definitions and the idea that risk and associated attributes are measurable. Measurable, quantifiable metrics are a key area of emphasis, which should lend itself well to an operational world with a richness of operational data.

At its base, FAIR has a definition of risk as the probable frequency and probable magnitude of loss. With this definition, a clear hierarchy of sub-elements emerges, with one side of the taxonomy focused on frequency and the other on magnitude.

Loss even frequency is the result of a threat agent acting on an asset with a resulting loss to the organization. This happens with a given frequency called the threat event frequency (TEF), in which a specified time window becomes a probability. There are multiple sub-attributes that define frequency of events, all of which can be understood with some form of measurable metric. Threat event frequencies are applied to vulnerability. Vulnerability here is not necessarily some compute asset weakness, but is more broadly defined as the probability that the targeted asset will fail as a result of the actions applied. There are further sub-attributes here as well.

The other side of the risk taxonomy is the probable loss magnitude (PLM), which begins to quantify the impacts, with the emphasis again being on measurable metrics. The FAIR specification makes it a point to emphasize how ephemeral some of these cost estimates can be, and this may indeed be the case when information security is the target of the discussion. Fortunately for the OT operator, a significant emphasis on operational efficiency and analysis makes understanding and quantifying costs much easier.

FAIR defines six forms of loss, four of them externally focused and two internally focused. Of particular value for operational teams are productivity and replacement loss. Response loss is also reasonably measured, with fines and judgments easy to measure but difficult to predict. Finally, competitive advantage and reputation are the least measurable.

## The Phased Application of Security in an Operational Environment

It is a security practitioner's goal to safely secure the environment for which he or she is responsible. For an operational technologist, this process is different because the priorities and assets to be protected are highly differentiated from the better-known IT environment. The differences have been discussed at length in this chapter, but many of the processes used by IT security practitioners still have validity and can be used in an OT environment. If there is one key concept to grasp, it is that security for an IoT environment is an ongoing process in which steps forward can be taken, but there is no true finish line.

The following sections present a phased approach to introduce modern network security into largely preexisting legacy industrial networks.

## Secured Network Infrastructure and Assets

Given that networks, compute, or operational elements in a typical IoT or industrial system have likely been in place for many years and given that the

physical layout largely defines the operational process, this phased approach to introducing modern network security begins with very modest, non-intrusive steps.

As a first step, you need to analyze and secure the basic network design. Most automated process systems or even hierarchical energy distribution systems have a high degree of correlation between the network design and the operational design. It is a basic tenet of ISA99 and IEC 62443 that functions should be segmented into zones (cells) and that communication crossing the boundaries of those zones should be secured and controlled through the concept of conduits. In response to this, it is suggested that a security professional discover the state of his or her network and all communication channels.

Figure 8-6 illustrates inter-level security models and inter-zone conduits in the process control hierarchy.

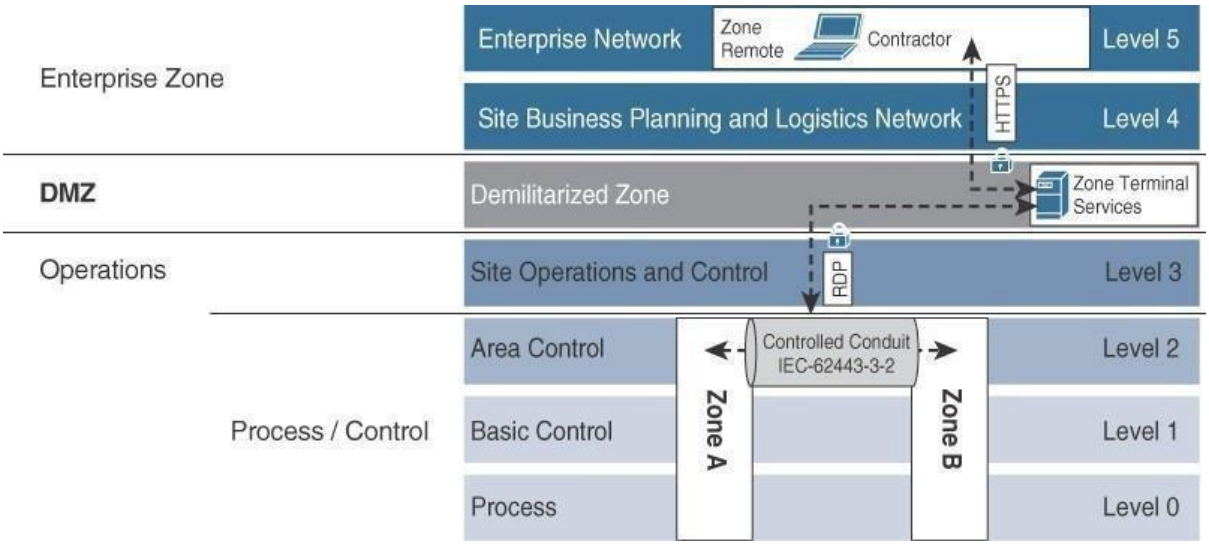



**Figure 8-6** Security Between Levels and Zones in the Process Control Hierarchy Model

Normal network discovery processes can be highly problematic for older networking equipment. In fact, the discovery process in pursuit of improved safety, security, and operational state can result in degradation of all three. Given that condition, the network discovery process may require manual inspection of physical connections, starting from the highest accessible aggregation point and working all the way down to the last access layer. This discovery activity must include a search for wireless access points. For the sake of risk reduction, any on-wire network mapping should be done passively as much as possible.

It is fair to note that this prescribed process is much more likely to succeed in a smaller confined environment such as a plant floor. In geographically distributed environments, it may not be possible to trace the network, and in such cases, the long-haul connections may not be physical or may be carried by an outside communication provider. For those sections of the operational network, explicit partnering with other entities is required.

A side activity of this network tracing process is to note the connectivity state of the physical connections. This is not just an exercise to see what fiber or cables are in what ports but to observe the use or operational state of other physical connections, such as USB, SD card, alarm channel, serial, or other connections, at each network appliance. For more modern environments

where updated networking devices and protocols are used, tools like NetFlow and IPFIX can also be used to discover the network communication paths.

Normally, in an IT environment, the very first stage of discovery is focused on assets connected to the network. Assets remain critical, but from an efficiency and criticality perspective, it is generally recommended to find data paths into and between zones (cells) rather than the serial links between devices within a zone. One thing to continually be on the lookout for is the ever-dangerous, unsecured, and often undocumented convenience port. Any physical port that is not physically locked down or doesn't have an enforceable protection policy is an uncontrolled threat vector.

Once the network is physically mapped, the next step is to perform a connectivity analysis through the switch and router ARP tables and DHCP requests within the network infrastructure. This should help further illuminate connectivity, good or bad, that has occurred. Firewall and network infrastructure data can contribute to understanding what devices are talking to other devices and the traffic paths over which this is done.

At this stage, the network should be reasonably well understood and prepared for secure connectivity.

Modern networking equipment offers a rich set of access control and secured communications capabilities. Starting at the cell/zone level, it is important to ensure that there is a clear ingress/egress aggregation point for each zone. If your communications patterns are well identified, you can apply access control policies to manage who and what can enter those physical portions of the process. If you are not comfortable explicitly controlling the traffic, then begin with alert-only actions. With time, you should be confident enough in your knowledge to apply controls.

At upstream levels, consider traffic controls such as denial of service (DoS) protection, traffic normalization activities, and quality of service (QoS) controls (such as marking and black-holing or rate-limiting scavenger-class traffic). The goal here is to ensure that these aggregated traffic segments are carrying high-priority traffic without impediment.

Network infrastructure should also provide the ability to secure communications between zones via secured conduits (see Figure 8-6). The primary method is encrypted communications in the form of virtual private networks (VPNs). VPNs can come in multiple forms, such as site-to-site, which would be appropriate between a utility substation and a control center,

or perhaps in cell-to-cell communications. Remote access controls can be established in more ad hoc situations and utilize the convenience of browser-based VPNs with Secure Sockets Layer (SSL)–based VPNs. If latency concerns are not particularly high, you can use Media Access Control Security (MACSec) hop-by-hop encryption to allow for potential controls and visibility at key junctions.

The next discovery phase should align with the software and configurations of the assets on the network. At this point, the rights and roles of the network administrator may be insufficient to access the required information.
Certainly, the network infrastructure and its status are within the network admin's view, but the individual assets likely are not. At this point, organizational cooperation is required for success. For an experienced IT-based network practitioner, this is not an unusual situation. It is very common, especially in larger enterprises, to see a separation of responsibilities and controls between the communications transport and the assets to which they are connected. At the operations level, similar cooperation is required with those responsible for the maintenance of the OT assets.

There are reasonable sources of information describing the configuration state of OT assets. The control systems associated with the processes hold historical data describing what is connected and what those assets are doing. A review of historical data should provide an idea of what assets are present and what operations are being performed on them, and it should identify such things as firmware updates and health status. The volume of data to analyze may be challenging, but if it is organized correctly, it would be valuable for understanding asset operation.

With an initial asset inventory completed, you can initiate a risk analysis based on the network and assets, and determine an initial scope of security needs.

## Deploying Dedicated Security Appliances

The next stage is to expand the security footprint with focused security functionality. The goal is to provide visibility, safety, and security for traffic within the network. Visibility provides an understanding of application and communication behavior. With visibility, you can set policy actions that reflect the desired behaviors for inter-zone and conduit security.

While network elements can provide simplified views with connection histories or some kind of flow data, you get a true understanding when you

look within the packets on the network. This level of visibility is typically achieved with deep packet inspection (DPI) technologies such as intrusion detection/prevention systems (IDS/IPS). These technologies can be used to detect many kinds of traffic of interest, from simply identifying what applications are speaking, to whether communications are being obfuscated, to whether exploits are targeting vulnerabilities, to passively identifying assets on the network.

With the goal of identifying assets, an IDS/IPS can detect what kind of assets are present on the network. Passive OS identification programs can capture patterns that expose the base operating systems and other applications communicating on the network. The organizationally unique identifier (OUI) in a captured MAC address, which could have come from ARP table exploration, is yet another means of exposure. Coupled with the physical and historical data mentioned before, this is a valuable tool to expand on the asset inventory without having to dangerously or intrusively prod critical systems.

Application-specific protocols are also detectable by IDS/IPS systems. For more IT-like applications, user agents are of value, but traditionally, combinations of port numbers and other protocol differentiators can contribute to identification. Some applications have behaviors that are found only in certain software releases. Knowledge of those differences can help to determine the software version being run on a particular asset.

Within applications and industrial protocols are well-defined commands and, often, associated parameter values. Again, an IDS/IPS can be configured to identify those commands and values to learn what actions are being taken and what associated settings are being changed.

All these actions can be done from a non-intrusive deployment scenario. Modern DPI implementations can work out-of-band from a span or tap. Viewing copies of packets has no impact on traffic performance or latency. It is easily the safest means of getting deep insight into the activities happening on a network.

Visibility and an understanding of network connectivity uncover the information necessary to initiate access control activity. Access control is typically achieved with access control lists (ACLs), which are available on practically all modern network equipment. For improved scalability, however, a dedicated firewall would be preferred. Providing strong segmentation and zone access control is the first step. Access control, however, is not just limited to the typical address and protocol identifiers. Modern firewalls have

the ability to discern attributes associated with the user accessing the network, allowing controls to be placed on the ―who‖ element also. In addition, access control can be aligned with applications and application behaviors. Equipped with the right toolset, a modern OT practitioner can ensure that only those operators in a certain user class can initiate any external commands to that particular asset.

Safety is a particular benefit as application controls can be managed at the cell/zone edge through an IDS/IPS. The same technologies that observe the who and what can also manage the values being passed to the target asset.

For example, in a manufacturing scenario where a robot operates, there may be an area frequented by workers who are within the potential range of the robot's operation. The range is unique to the physical layout of the cell, and parameter changes could cause physical harm to a plant worker. With an IDS/IPS, the system can detect that a parameter value exceeds the safety range and act accordingly to ensure worker safety.

Safety and security are closely related linguistically (for example, in German, the same word, Sicherheit, can be used for both), but for a security practitioner, security is more commonly associated with threats. Threat identification and protection is a key attribute of IPSs using DPI.

Mature IPSs have thousands of threat identifiers, which address the complete range of asset types where remotely exploitable vulnerabilities are known. In some cases, the nature of the threat identifier is generic enough that it addresses a common technique without having to be associated with a particular application instance of the vulnerability type.

Placement priorities for dedicated security devices vary according to the security practitioner's perception of risk. If visibility is incomplete and concern dictates that further knowledge is necessary prior to creating a proactive defense, the security device should be placed where that gap is perceived. It is important to note that the process of gaining visibility or addressing risk is dynamic. Networks change, and as knowledge is gained, new priorities (either in the form of visible threats or a reduction of gaps) creates new points of emphasis. Given this dynamism, consider the idea that placement of a dedicated security device can change as well. In other words, just because you start with a device in one location does not mean you can't move it later to address security gaps.

A particularly valuable function is enabled if a security device can terminate

VPNs in addition to performing deep packet inspection. Secured communication, potentially from a vendor representative outside the organization, can be terminated at the ingress to the device and then inspected. The time cost of the termination would be similar to what would be done on the switch, and then inspection of what that remote user accessing the network is doing is viable. Naturally, any potential threat traffic can be halted as well.

If the zone/cell houses critical infrastructure and remote operation is requisite, a redundant high-availability configuration for both the network and security infrastructure is advised.

For the purposes of pure visibility, hanging off a mirror or span port from the switch would be optimal. For control capabilities, one must be in-line to truly act on undesired traffic. In most cases, the preferred location is upstream of the zone/cell access switch between the aggregation layer and the zone switch. It may be viable to have the security device between the zone assets and the zone access switch as well.

For broader, less detailed levels of control, placement of dedicated security devices upstream of the aggregation switches is the preferred approach. If the network has multiple zones going through the aggregation switch with mostly redundant functionality but with no communication between them, this may be a more efficient point of deployment.

At some point, a functional layer above the lowest zone layer becomes connected to the network, and there should be a device located between those functions and their OT charges in the zones/cells. At that next layer up, there may be HMIs or other lower-level operational tools. For safety considerations, a control point between that layer and the cell is valuable.

## Higher-Order Policy Convergence and Network Monitoring

So far we have focused on very basic concepts that are common and easily implemented by network engineering groups. Finding network professionals with experience performing such functions or even training those without prior experience is not difficult. Another security practice that adds value to a networked industrial space is convergence, which is the adoption and integration of security across operational boundaries.

This means coordinating security on both the IT and OT sides of the organization. Convergence of the IT and OT spaces is merging, or at least

there is active coordination across formerly distinct IT and OT boundaries. From a security perspective, the value follows the argument that most new networking and compute technologies coming to the operations space were previously found and established in the IT space. It is expected to also be true that the practices and tools associated with those new technologies are likely to be more mature in the IT space.

There are advanced enterprise-wide practices related to access control, threat detection, and many other security mechanisms that could benefit OT security. As stated earlier, the key is to adjust the approach to fit the target environment.

Network security monitoring (NSM) is a process of finding intruders in a network. It is achieved by collecting and analyzing indicators and warnings to prioritize and investigate incidents with the assumption that there is, in fact, an undesired presence.

The practice of NSM is not new, yet it is not implemented often or thoroughly enough even within reasonably mature and large organizations. There are many reasons for this underutilization, but lack of education and organizational patience are common reasons.

It is important to note that NSM is inherently a process in which discovery occurs through the review of evidence and actions that have already happened. This is not meant to imply that it is a purely postmortem type of activity. If you recognize that intrusion activities are, much like security, an ongoing process, then you sees that there is a similar set of stages that an attacker must go through. The tools deployed will slow that process and introduce opportunities to detect and thwart the attacker, but there is rarely a single event that represents an attack in its entirety. NSM is the discipline that will most likely discover the extent of the attack process and, in turn, define the scope for its remediation.

# MODULE-5

## IoT Physical Devices and Endpoints – Arduino UNO and Raspberry Pi

## and

## Smart and Connected Cities

# Topics Covered


- IoT Physical Devices and Endpoints – Arduino UNO:
  - Introduction to Arduino,
    - Why Arduino?
    - Which Arduino?
  - Exploring Arduino UNO learning Board
    - Things that Arduino do
  - Installing the Software (Arduino IDE),
    - Connecting Arduino UNO learning Board
  - Fundamentals of Arduino Programming.
    - Difference between Analog, Digital and PWM Pins

# Arduino UNO
# Introduction to Arduino

- Arduino is a basic single board microcontroller designed to make applications, interactive controls, or environments easily adaptive.
  - The hardware consists of a board designed around an 8-bit microcontroller, or a 32-bit ARM.
  - Current models feature things like a USB interface, analog inputs, and GPIO pins which allows the user to attach additional boards.
- Introduced in 2005, the Arduino platform was designed to provide a cheaper way for students and professionals to create applications that play in the human interface world using sensors, actuators, motors, and other rudimentary products.
- It offers a simple integrated **IDE (integrated development environment)** that runs on regular personal computers and allows users to write programs for Arduino using C or C++.

# Arduino UNO
# Introduction to Arduino

- **Why Arduino?**
  - **Inexpensive:**
    - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand.
  - **Cross-platform:**
    - The Arduino software runs on Windows, Macintosh OS and Linux operating systems.
  - **Simple, clear programming environment:**
    - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well.
  - **Open source and extensible software:**
    - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries.
  - **Open source and extensible hardware:**
    - The Arduino is based on Atmel's ATMEGA microcontrollers. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

# Arduino UNO
# Introduction to Arduino

- **Which Arduino?**
  - Entry Level
    - Easy to use and ready to first creative projects. These boards and modules are the best to start learning and tinkering with electronics and coding.
  - Enhanced Features
    - Experience the excitement of more complex projects, with advanced functionalities, or faster performances.
  - Internet of Things
    - Make connected devices easily with IoT and the world wide web.
  - Wearable
    - Add smartness to projects and sewing the power of electronics directly to textiles.

# Arduino UNO
# Introduction to Arduino

- ARDUINO UNO
  - A microcontroller board based on the ATmega328P.
  - It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.
  - Connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
- ARDUINO MEGA 2560
  - A microcontroller board based on the ATmega2560.
  - It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
  - It is the recommended board for 3D printers and robotics projects.
- ARDUINO MICRO
  - A microcontroller board based on the ATmega32U4, featuring a built-in USB which makes the Micro recognisable as a mouse or keyboard.
  - It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and a reset button.

# Arduino UNO
# Introduction to Arduino

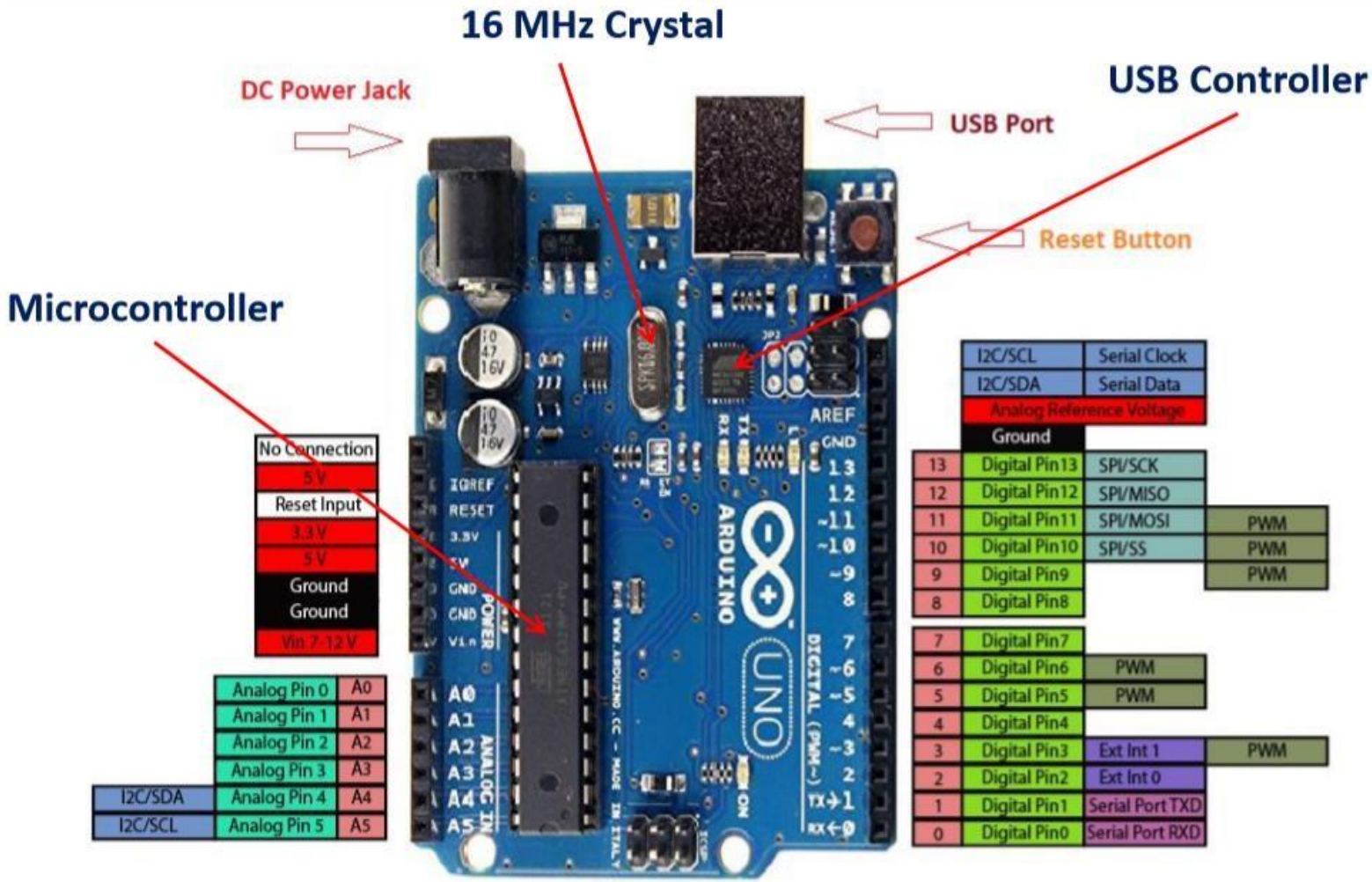
- ARDUINO MKR1000
  - It is based on the Atmel ATSAMW25 ARM SoC (System on Chip), that is part of the Smart Connect family of Atmel Wireless devices, specifically designed for IoT projects and devices.
  - The ATSAMW25 is composed of three main blocks:
    - SAMD21 Cortex-M0+ 32bit low power ARM MCU
    - WINC1500 low power 2.4GHz IEEE® 802.11 b/g/n Wi-Fi
    - ECC508 Crypto Authentication
    - PCB Antenna.

# Arduino UNO

## Exploring Arduino UNO learning Board

# Arduino UNO

## Exploring Arduino UNO learning Board



# Arduino UNO

## Exploring Arduino UNO learning Board

- 14 digital pins on the Uno can be used as an input or output,
- 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values).
- Serial:
  - Pin 0 (RX) and Pin 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts:
  - Pin 2 and Pin 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM:
  - Pin 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output.
- SPI (Serial Peripheral Interface):
  - Pin 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED:
  - 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- TWI (Two Wire Interface):
  - A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.
- Reset:
  - Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- AREF (Analog REFerence):
  - Reference voltage for the analog inputs.

# Arduino UNO

## Exploring Arduino UNO learning Board

- Things that Arduino Can Do

# Arduino UNO

## Installing the Software (Arduino IDE)

- The Arduino Software (IDE) allows you to write programs and upload them to board. In the Arduino Software page you will find two options:
  - 1. Online IDE (Arduino Web Editor). It will allow to save sketches in the cloud, having them available from any device and backed up.
  - 2. Offline, should use the latest version of the desktop IDE.

- Install the Arduino Desktop IDE accordingly to operating system.
  - Windows
  - Mac OS X
  - Linux
  - Portable IDE (Windows and Linux)
  - Choose board in the list here on the right to learn how to get started with it and how to use it on the Desktop IDE.

# Arduino UNO

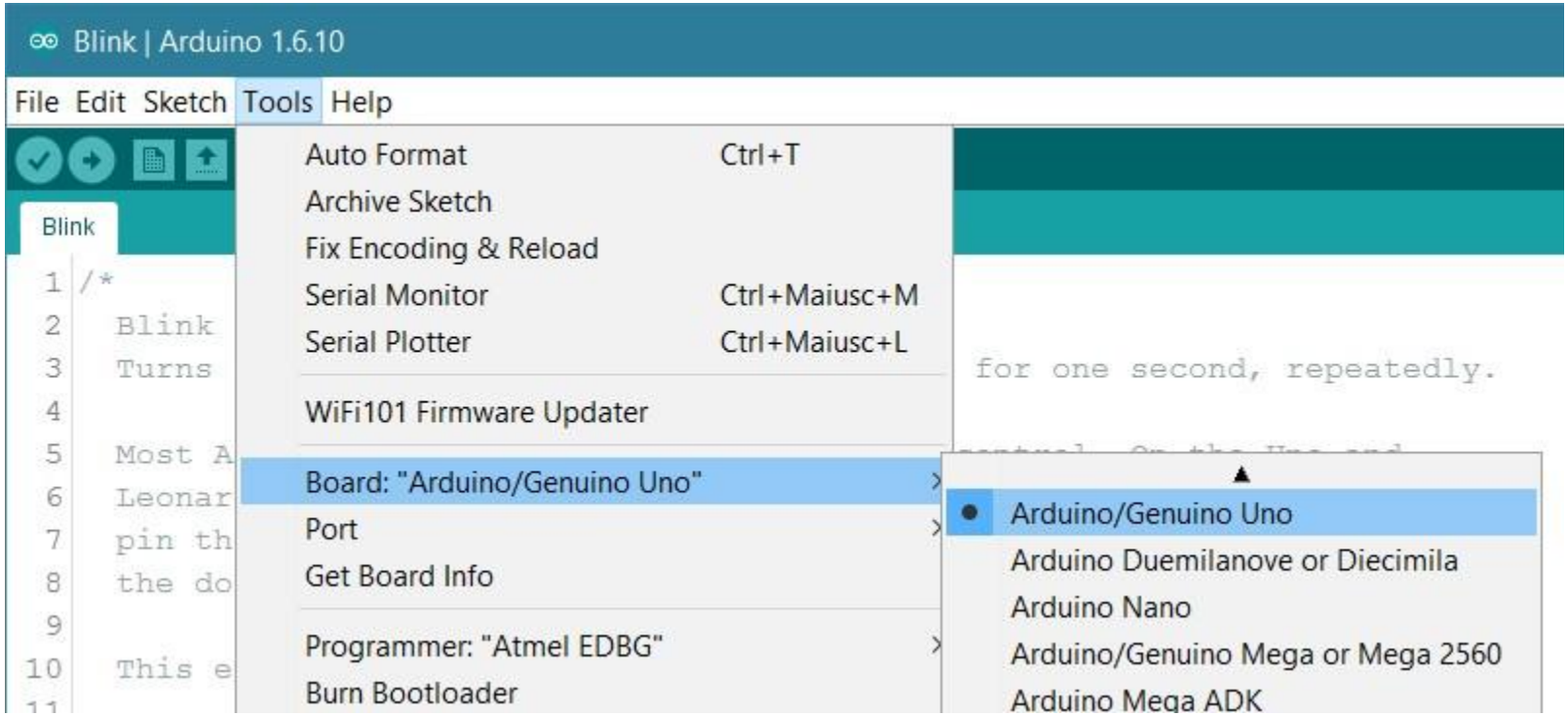
## Installing the Software (Arduino IDE)

- **Connecting Arduino UNO Learning Board:**
  - If you want to program your Arduino Uno while offline you need to install the Arduino Desktop IDE.
  - Connect your Uno board with an A B USB cable; sometimes this cable is called a USB printer cable.
  - If you used the Installer, Windows – from XP up to 10 – will install drivers automatically as soon as you connect your board.

# Arduino UNO

## Installing the Software (Arduino IDE)

- You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino or Genuino board.



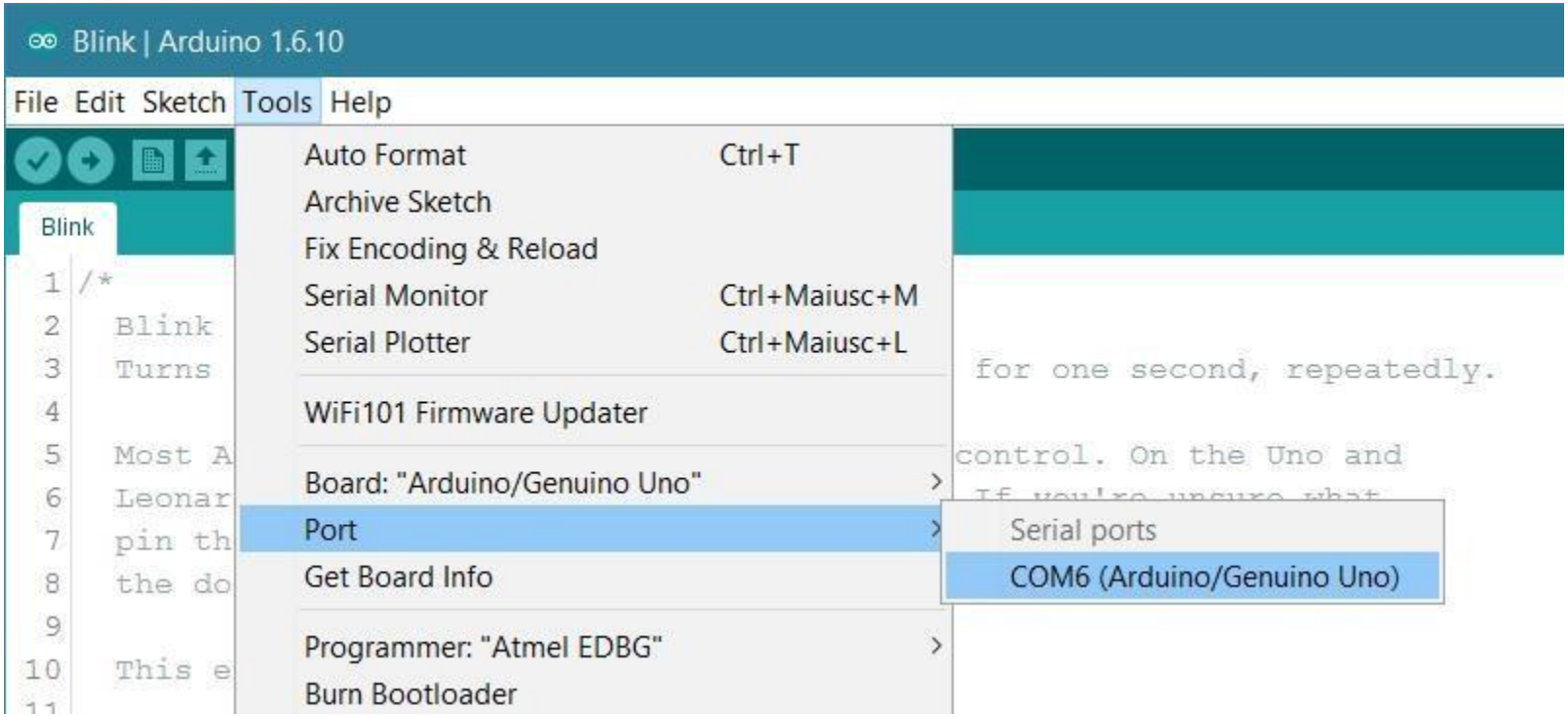
# Arduino UNO

## Installing the Software (Arduino IDE)

- Select the serial device of the board from the Tools | Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your board and re-open the menu; the entry that disappears should be the Arduino or Genuino board. Reconnect the board and select that serial port.

# Arduino UNO

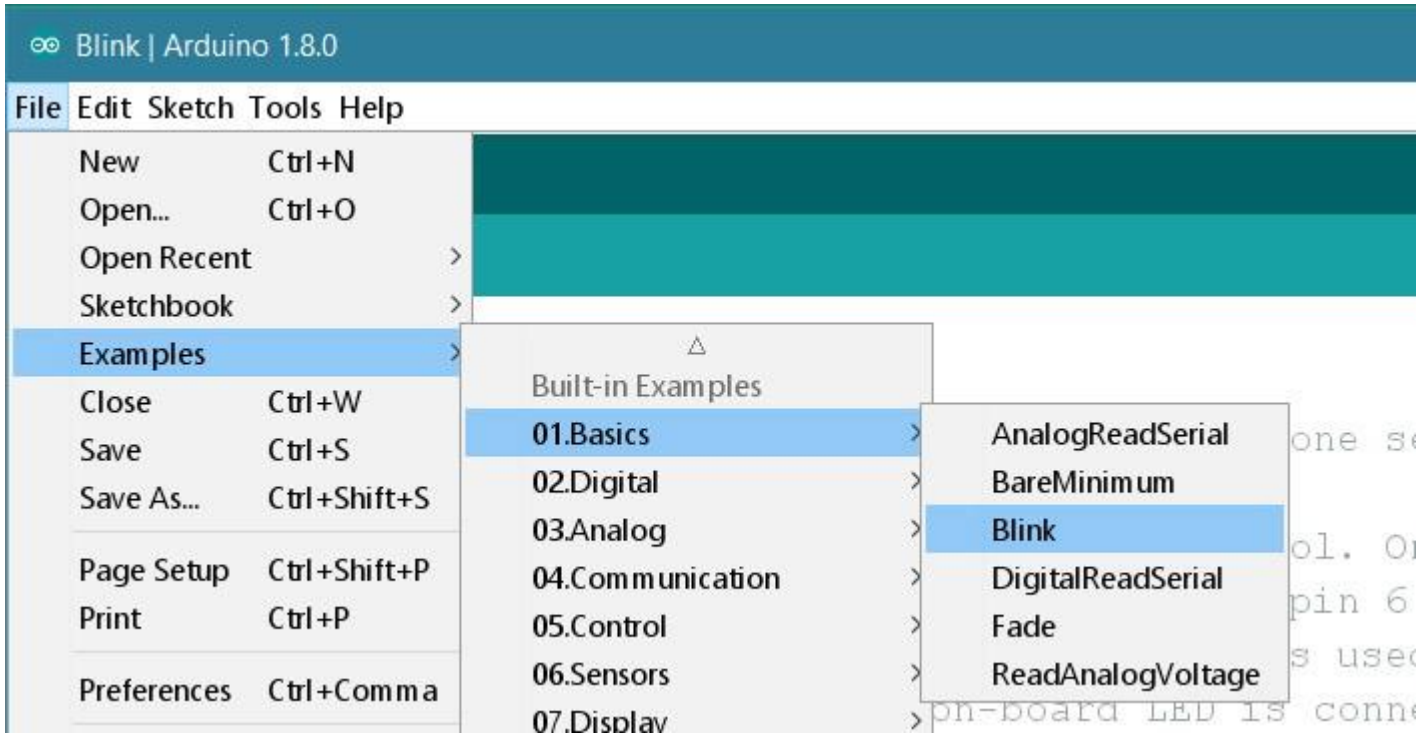
## Installing the Software (Arduino IDE)



# Arduino UNO

## Installing the Software (Arduino IDE)

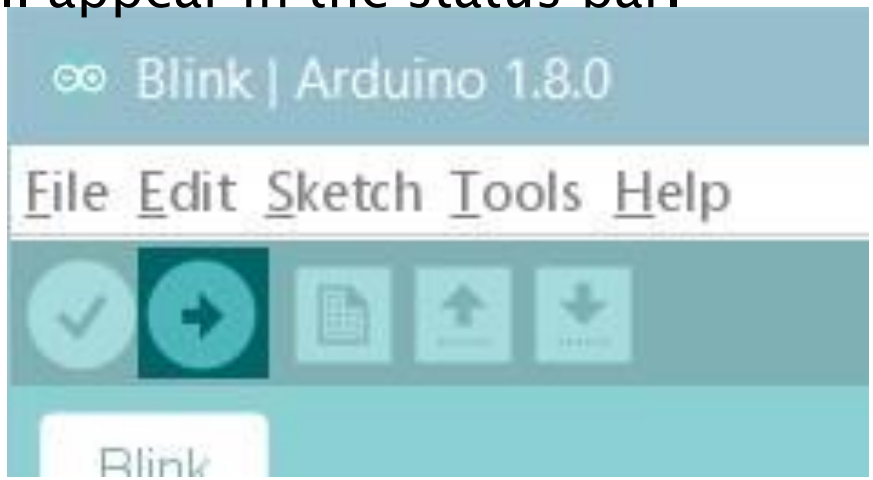
- Open your first sketch
- Open the LED blink example sketch: File > Examples >01.Basics > Blink.



# Arduino UNO

## Installing the Software (Arduino IDE)

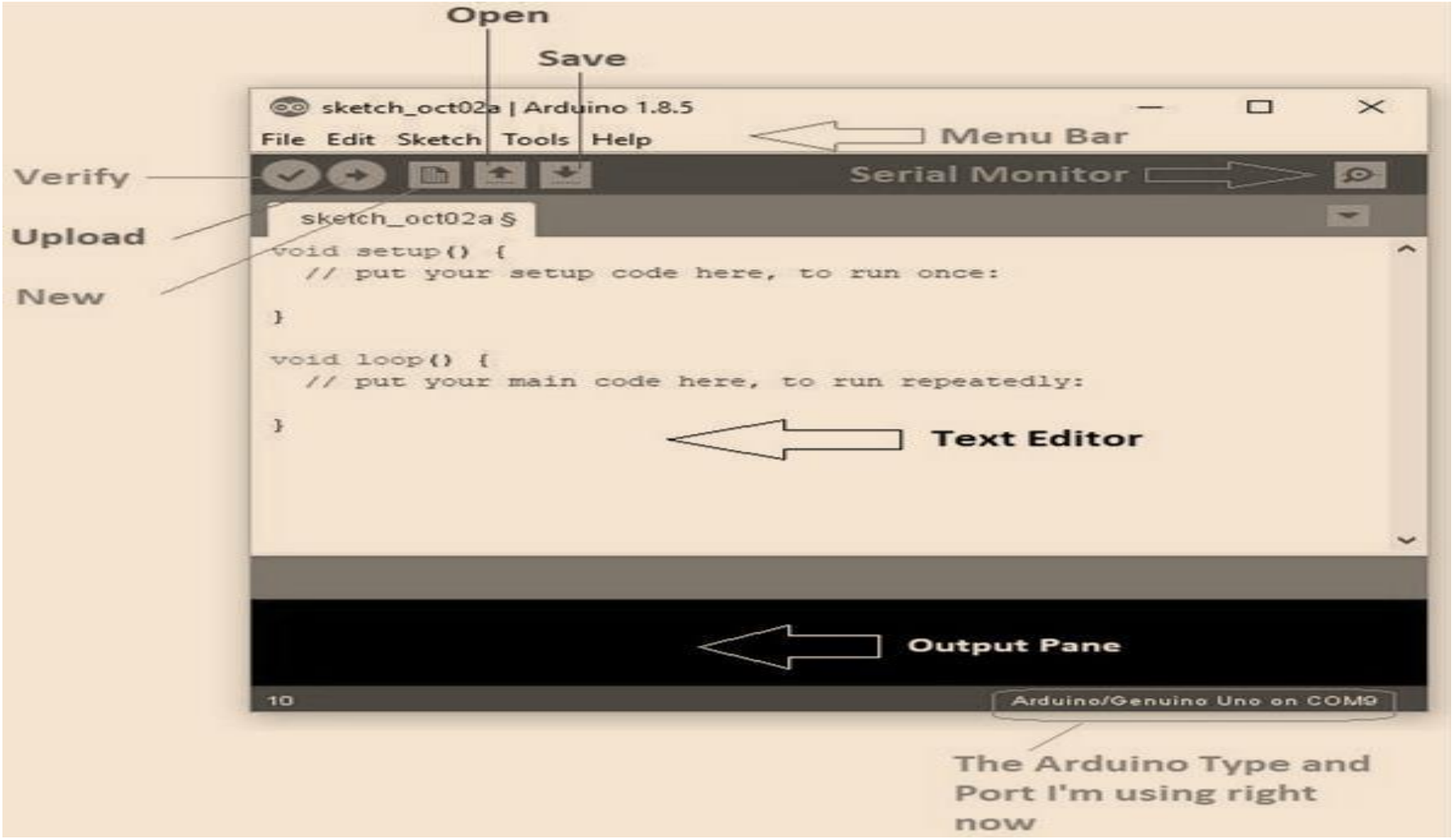
- Upload the program
- Now, simply click the "Upload" button in the environment. Wait a few seconds – you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.




- A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange). If it

# Arduino UNO

## Installing the Software (Arduino IDE)



# Arduino UNO

## Installing the Software (Arduino IDE)

| File | |
|---|---|
| **New** | This is used to open new text editor window to write your code |
| **Open** | Used for opening the existing written code |
| **Open Recent** | The option reserved for opening recently closed program |
| **Sketchbook** | It stores the list of codes you have written for your project |
| **Examples** | Default examples already stored in the IDE software |
| **Close** | Used for closing the main screen window of recent tab. If two tabs are open, it will ask you again as you aim to close the second tab |
| **Save** | It is used for saving the recent program |
| **Save as** | It will allow you to save the recent program in your desired folder |
| **Page setup** | Page setup is used for modifying the page with portrait and landscape options. Some default page options are already given from which you can select the page you intend to work on |
| **Print** | It is used for printing purpose and will send the command to the printer |
| **Preferences** | It is page with number of preferences you aim to setup for your text editor page |
| **Quit** | It will quit the whole software all at once |

# Arduino UNO

## Installing the Software (Arduino IDE)

- Technical Specification

| | |
|---|---|
| Microcontroller | ATmega328P |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328P) of which 0.5 KB used by bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| LED_BUILTIN | 13 |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 g |

# Arduino UNO

## Fundamentals of Arduino Programming

- The Arduino IDE supports the languages C and C++ using special rules of code structuring.
- The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures.
- User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution.

# Arduino UNO

## Fundamentals of Arduino Programming

- Sketch
  - A sketch is a program written with the Arduino IDE.[57] Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde.
  - A minimal Arduino C/C++ program consist of only two functions:
    - **setup()**: This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.
    - **loop()**: After setup() function exits (ends), the loop() function is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

# Arduino UNO

## Fundamentals of Arduino Programming

- Blink example:
  - Most Arduino boards contain a light-emitting diode (LED) and a current limiting resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions.
  - A typical program used by beginners, akin to Hello, World!, is "blink", which repeatedly blinks the on-board LED integrated into the Arduino board.
  - This program uses the functions pinMode(), digitalWrite(), and delay(), which are provided by the internal libraries included in the IDE environment.

# Arduino UNO

## Fundamentals of Arduino Programming

```
#define LED_PIN 13                  // Pin number attached to LED.

void setup() {
    pinMode(LED_PIN, OUTPUT);       // Configure pin 13 to be a digital output.
}

void loop() {
    digitalWrite(LED_PIN, HIGH);    // Turn on the LED.
    delay(1000);                    // Wait 1 second (1000 milliseconds).
    digitalWrite(LED_PIN, LOW);     // Turn off the LED.
    delay(1000);                    // Wait 1 second.
}
```

# Arduino UNO

## Fundamentals of Arduino Programming

- **Variables and Data Types**

| DATA TYPE | CONTENTS |
|---|---|
| `void` | No data type |
| `boolean` | True or false |
| `char` | One character, stored as an ASCII number ('A', 'B', 'C'...) |
| `unsigned char` | Decimal numbers, from 0 to 255 |

# Arduino UNO

## Fundamentals of Arduino Programming

- **Variables and Data Types**

| DATA TYPE | CONTENTS |
|---|---|
| `byte` | Decimal numbers, from 0 to 255 |
| `int` | Decimal numbers, from –32,768 to 32,767<br>(Arduino Due, from –2,147,483,648 to 2,147,483,647) |
| `unsigned int` | Decimal numbers, from 0 to 65,535<br>(Arduino Due, from 0 to 4,294,967,295) |
| `word` | Decimal numbers, from 0 to 65,535 |
| `long` | Decimal numbers, from –2,147,483,648 to 2,147,483,647 |
| `unsigned long` | Decimal numbers, from 0 to 4,294,967,295 |
| `short` | Decimal numbers, –32,768 to 32,767 |
| `float` | Floating point numbers, from $-3.4028235 \times 10^{38}$ $3.4028235 \times 10^{38}$ |
| `double` | Floating point numbers |
| `string` | An array of char |
| `String` | Advanced arrays of char |
| `array` | A collection of variables |

# Arduino UNO

## Fundamentals of Arduino Programming

- **if and if ..else Statement**

```
if (expression)
 {
statement;
}

if (expression)
{
do_this;
}
else
{
do_that;
}
```

# Arduino UNO

## Fundamentals of Arduino Programming

- **while Loop**

```
while (button == false)
{
button = check_status(pin4);
}
```

# Arduino UNO

## Fundamentals of Arduino Programming

- **Digital I/O**
  - pinMode()
  - digitalRead()
  - digitalWrite()

- **pinMode()**
- Before using a pin as a digital input or output, must first configure the pin, which is done with pinMode().
- pinMode() uses two parameters: pin and mode.
  - pinMode(pin, mode)
  - The pin parameter is simply the digital pin number want to set.
  - The mode parameter is one of three constants: INPUT or OUTPUT,

# Arduino UNO

## Fundamentals of Arduino Programming

- **digitalRead()**
- In order to read the state of a digital pin, you must use digitalRead():
  - result = digitalRead(pin);
- The pin parameter is the pin number you want to read from.
- This function returns either HIGH or LOW, depending on the input

# Arduino UNO

## Fundamentals of Arduino Programming

- **digitalWrite()**
- To write the state of a pin that was declared as an OUTPUT, use the digitalWrite() function:
  - digitalWrite(pin, value);
- The pin parameter is the pin number you want to write to, and the value is the logical level you want to write; HIGH or LOW.

# Arduino UNO

## Fundamentals of Arduino Programming

- **Analog I/O**
- analogRead()
- To read a value from an analog pin, you call analogRead().
  - int analogRead(pin)
  - analogRead() reads the voltage value on a pin and returns the value as an int.
  - The pin argument denotes the analog pin you want to read from. When referring to an analog pin, call them as A0, A1, A2,…A6.
  - This function takes approximately 100 microseconds to perform.

# Arduino UNO

## Fundamentals of Arduino Programming

- analogWrite()
  - analogWrite() is used to write an analog output on a digital pin.
  - Arduinos use Pulse-width modulation (PWM).
  - PWM is digital but can be used for some analog devices.
  - It uses a simple technique to “emulate” an analog output.
  - It relies on two things:
    - a pulse width and a duty cycle.
  - It is a way of simulating any value within a range by rapidly switching between 0 volts and 5 volts.

# Arduino UNO

## Fundamentals of Arduino Programming

- **Time Functions**
- **delay()**
  - tells the microcontroller to wait for a specified number of milliseconds before resuming the sketch.
- **millis()**
  - millis() returns the number of milliseconds that the sketch has been running, returning the number as an unsigned long.

# Arduino UNO

## Fundamentals of Arduino Programming

- **Mathematical Functions**
- **min()**
  - min() returns the smaller of two numbers.
  - E.g. result = min(x, y)
- **max()**
  - max() returns the higher of two values.
  - result = max(x, y)

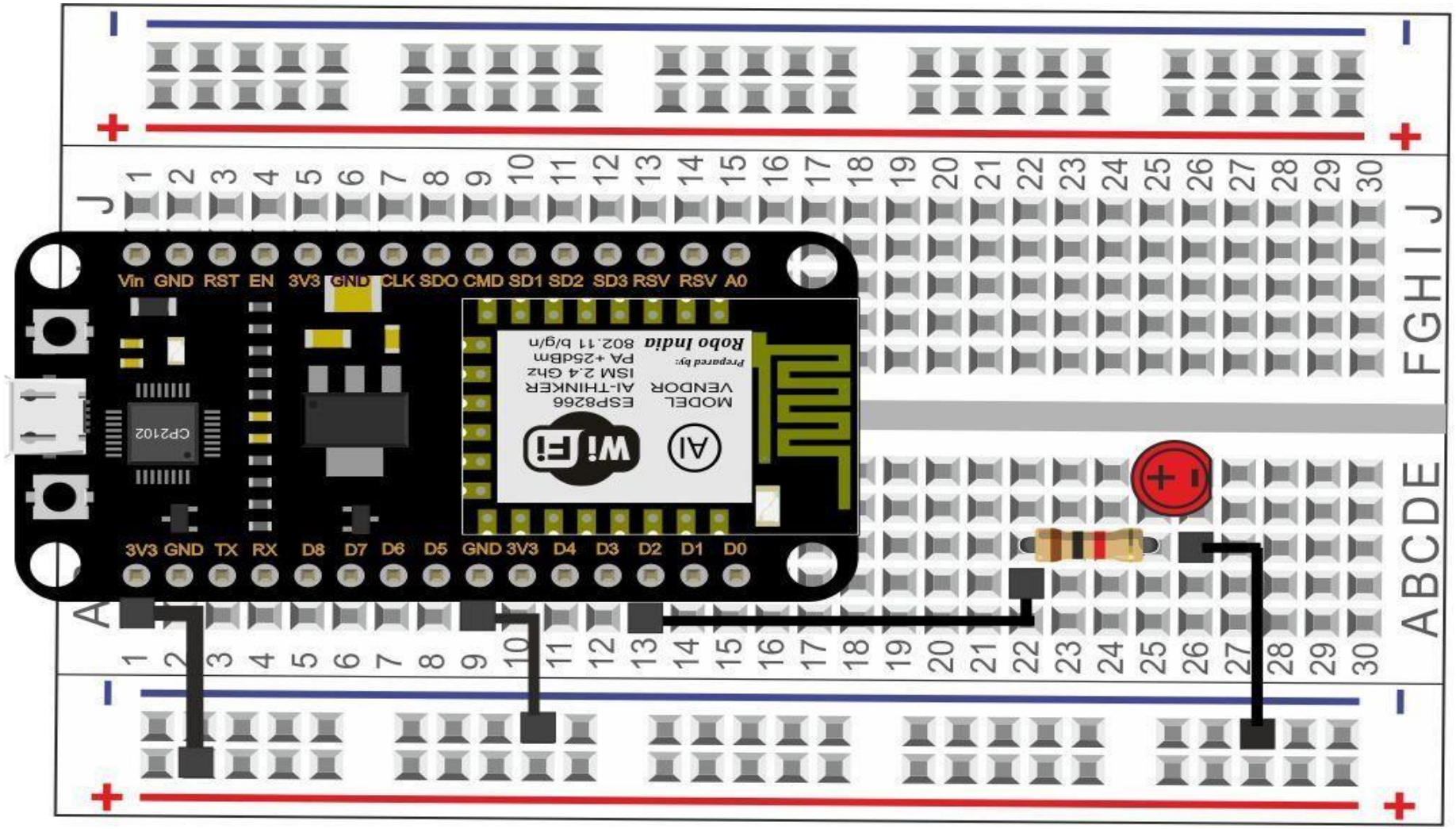
# Arduino UNO

## Fundamentals of Arduino Programming

- **random()**
  - Arduinos are capable of generating pseudo-random numbers using the random() function:
  - result = random(max);
  - result = random(min, max);
  - This function takes one or two parameters specifying the range for the random number to be chosen.
  - If the min parameter is omitted, the result will be a number between zero and max, otherwise the number will be between min and max.
    - The result is returned as a long.

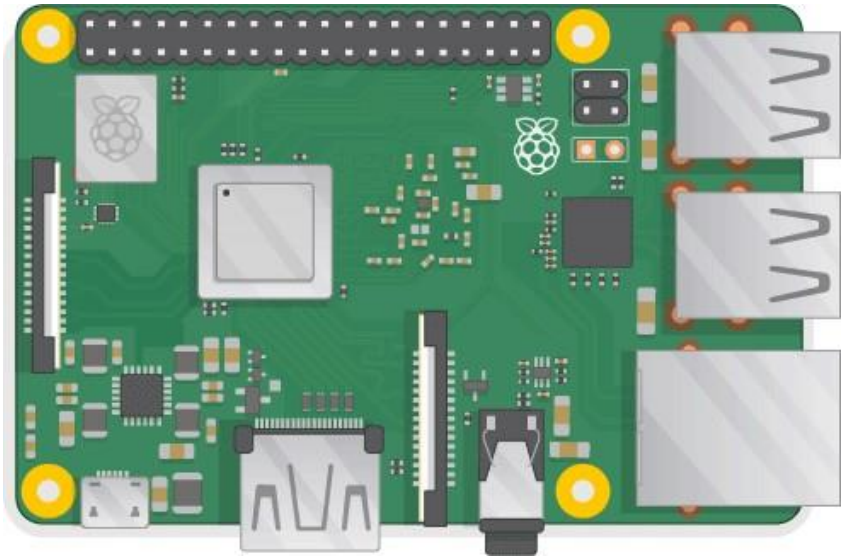
# Arduino UNO

## Fundamentals of Arduino Programming



# Topics Covered

- IoT  Physical Devices and Endpoints – RaspberryPi:
  - Introduction to Raspberry Pi,
  - Exploring the Raspberry Pi Learning Board,
    - Description of System on Chip (SoC)
    - Raspberry Pi interface
  - Raspberry Pi Operating Systems
    - Operating System (Not Linux Based)
    - Operating System (Linux Based)
    - Media centre Operating System
    - Audio Operating System
    - Recalbox
  - Operating Systems Setup on Raspberry Pi
    - Formatting SD Card
    - OS Instllation
    - First Boot
    - Login Information
  - Raspberry Pi commands
  - Configuring  RaspberryPi,
  - Programming RaspberryPi with Python,



# Topics Covered

- Smart  and Connected Cities,
  - An IoT Strategy for Smarter Cities
    - Vertical IoT Needs for Smarter Cities
    - Global vs. Siloed Strategies
  - Smart City IoT Architecture
    - Street Layer
    - City Layer
    - Data Center Layer
    - Services Layer
    - On-Premises vs. Cloud
  - Smart City Security Architecture
  - Smart City Use-Case Examples
    - Connected Street Lighting
      - Connected Street Lighting Solution
      - Street Lighting Architecture
    - Smart Parking
      - Smart Parking Use Cases
      - Smart Parking Architecture
    - Smart Traffic Control
      - Smart Traffic Control Architecture
      - Smart Traffic Applications
    - Connected Environment
      - The Need for a Connected Environment
      - Connected Environment Architecture

# DS18B20 Temperature Sensor

**The DS18B20 is a 1-wire programmable Temperature sensor from maxim integrated. It is widely used to measure temperature in hard environments like in chemical solutions, mines or soil etc. The constriction of the sensor is rugged and also can be purchased with a waterproof option making the mounting process easy. It can measure a wide range of temperature from -55°C to +125° with a decent accuracy of ±5°C. Each sensor has a unique address and requires only one pin of the MCU to transfer data so it a very good choice for measuring temperature at multiple points without compromising much of your digital pins on the microcontroller.**

## DS18B20 Temperature Sensor

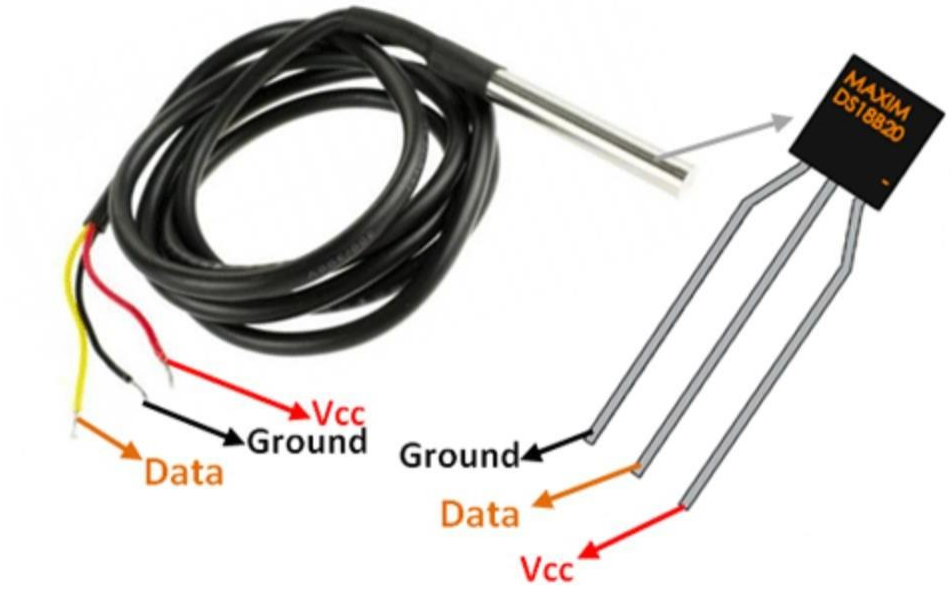
**Applications:**

- **Measuring temperature at hard environments**
- **Liquid temperature measurement**
- **Applications where temperature has to be measured at multiple points**

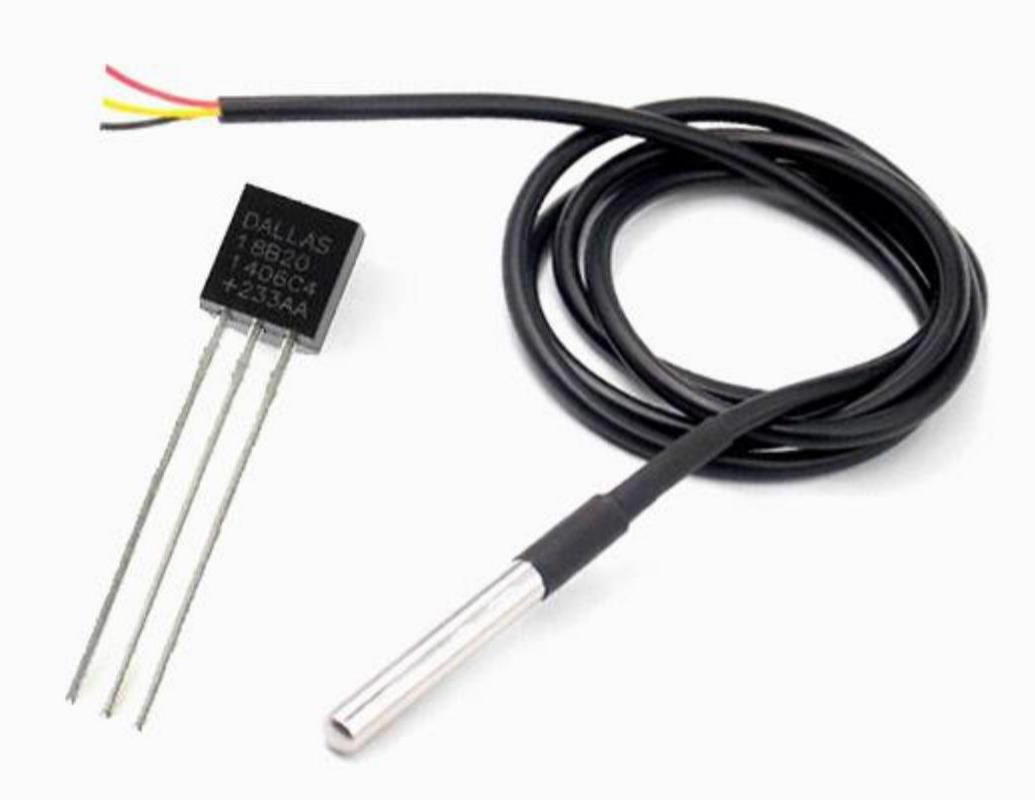
**Pin Configuration:**

| No | Pin Name | Description |
|---|---|---|
| 1 | Ground | Connect to the ground of the circuit |
| 2 | Vcc | Powers the Sensor, can be 3.3V or 5V |
| 3 | Data | This pin gives output the temperature value which can be read using 1-wire method |

## DS18B20 Temperature Sensor Pinout




## DS18B20 Temperature Sensor



## Connecting Raspberry Pi via SSH:

**You can access the command line of a Raspberry Pi remotely from another computer or device on the same network using SSH.The Raspberry Pi will act as a remote device: you can connect to it using a client on another machine.**

**1. Set up your local network and wireless connectivity**

**2. Enable SSH**

**3. Enable SSH on a headless Raspberry Pi (add file to SD card on another machine)**

**4. Set up your client**

## Accessing Temperature from DS18B20 sensors:

**The DS18B20 is a digital thermometer that allows to get 9-bit to 12-bit Celsius temperature measurements (programmable resolution). The temperature conversion time depends on the resolution used. For a 9-bit resolution it takes at most 93.75 ms and for a 12-bit resolution it takes at most 750 ms.The device is able to measure temperatures from -55°C to +125°C and has a ±0.5°C accuracy in the range from -10°C to +85°C.**

**Additionally, it has an alarm functionality with programmable upper and lower temperature trigger points. These thresholds are stored internally in non-volatile memory, which means they are kept even if the device is powered off .**

**The sensor communicates using the OneWire protocol, which means it only requires a pin from a microcontroller to be connected to it. Furthermore, each sensor has a unique 64-bit serial code, allowing multiple DS18B20 devices to function on the same OneWire bus.In terms of power supply, the device can operate with a voltage between 3.0 V and 5.5 V, which means it can operate with the same voltage of the ESP32 without the need for level conversion.**

## Remote access to RaspberryPi:

**To access a Raspberry Pi (or any home computer for that matter) from outside your home network, you'd usually need to jump through a lot of hoops, get an IP address, and tweak a few settings on your home router. If you just need to control a few simple things on your Raspberry Pi, that's overkill. We're going to outline two methods that skip all of that.**
**The first thing you need to do is get your Raspberry Pi set up and connected to your home network. Since you're exposing your Raspberry Pi to the internet, be sure you change your default password during the set up process. Once that's done, come back here to set up everything else.**

## Remote Log Into Your Raspberry Pi's Full Operating System Using VNC Connect:

**VNC has long been the best way to access any computer remotely on the same network. Recently, VNC Connect came out to make it easy to access your Raspberry Pi from anywhere using a cloud connection. Once it's set up, you can access your Raspberry Pi's graphic interface from any other computer or smartphone using the VNC Viewer app.**

**RaspberryPi Interface:**

**Serial:** The serial interface on Raspberry Pi has receive(rx) and transmit(Tx) pins for communication with serial peripherals.

**SPI:** Serial Peripheral Interface(SPI) is a synchronous serial data protocol used for communicating with one or more peripheral devices. In an SPI connection, there is one master device and one or more peripheral devices. There are five pins on Raspberry Pi for SPI interface:

- ✓ MISO(Master In Slave Out): Master line for sending data to the peripherals.
- ✓ MOSI(Master out Slave In): Slave line for sending data to the master.
- ✓ SCK(Serial Clock): Clock generated by master to synchronize data transmissions.
- ✓ CE0(Chip Enable 0): To enable or disable devices.
- ✓ CE0(Chip Enable 1): To enable or disable devices

**I2C:** The I2C interface pins on Raspberry Pi allow you to connect hardware modules. I2C interface allows synchronous data transfer with just two pins-SDA(data line) and SCL(clock line).

**RaspberryPi OS: (Not linux)**

1. RISC OS Pi
2. Free BSD
3. NetBSD
4. Plan 9
5. Haiku

**RaspberryPi OS: (Linux based)**

1. Xbean
2. Open SUSE
3. Arc OS
4. Kano OS
5. Nard SDX

**RaspberryPi OS: (Media center based)**

1. OSMC
2. OpenELEC
3. LitreELEC
4. Xbian
5. Rasplex

**RaspberryPi OS: (Audio based)**

1. Volumio
2. Pimusixbox
3. Runeaudio

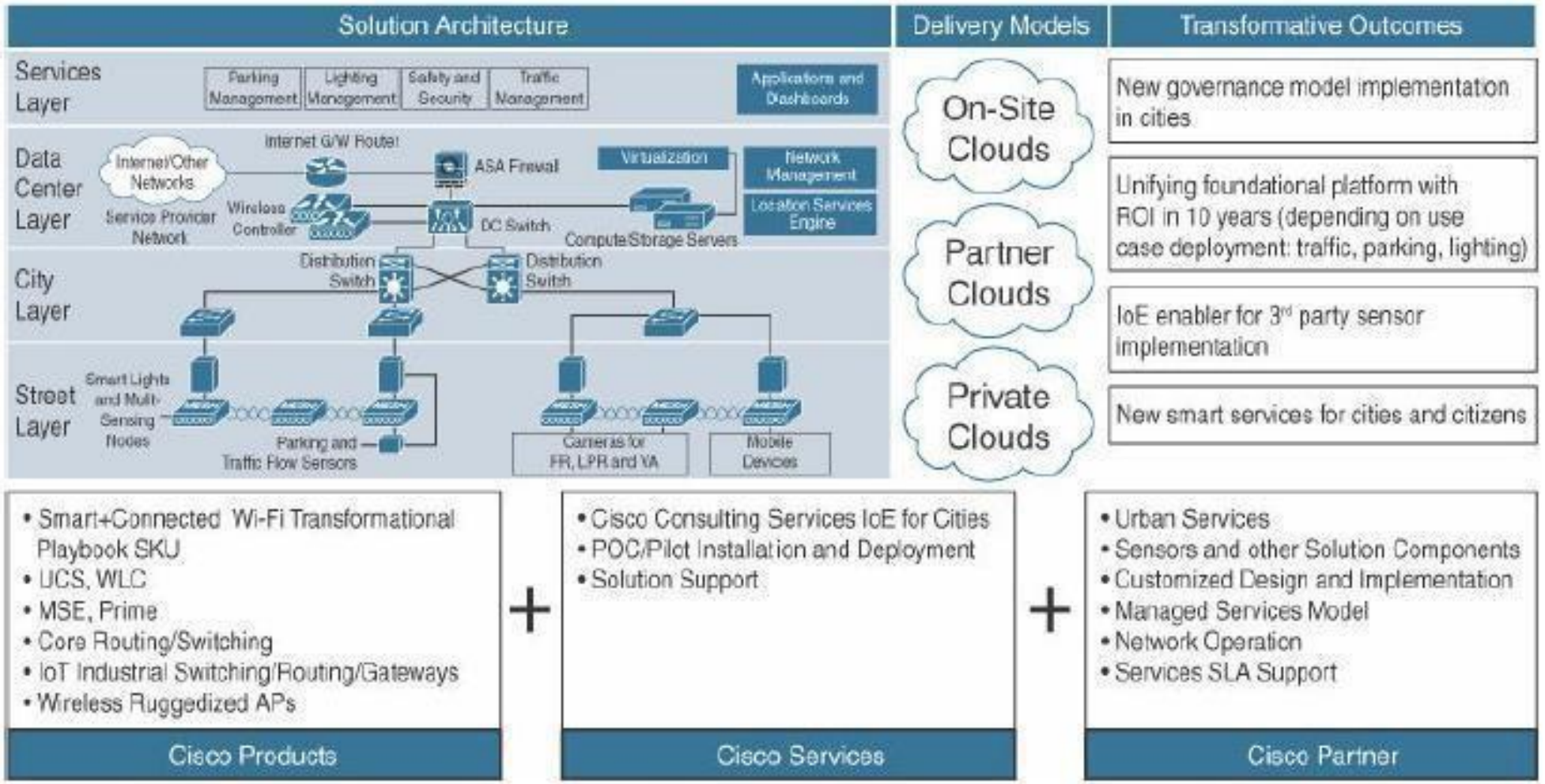
# Smart City IOT Architecture

**A smart city IoT infrastructure is a four-layered architecture, as shown in Figure**

**Data flows from devices at the street layer to the city network layer and connect to the data center layer, where the data is aggregated, normalized, and virtualized.**

**The data center layer provides information to the services layer, which consists of the applications that provide services to the city.**

**In smart cities, multiple services may use IoT solutions for many different purposes. These services may use different IoT solutions, with different protocols and different application languages**




Smart Cities Layered Architecture

## Street Layer:

**The street layer is composed of devices and sensors that collect data and take action based on instructions from the overall solution, as well as the networking components needed to aggregate and collect data.**

**A sensor is a data source that generates data required to understand the physical world. Sensor devices are able to detect and measure events in the physical world.**

**ICT connectivity solutions rely on sensors to collect the data from the world around them so that it can be analyzed and used to operationalize use cases for cities.**

## Street Layer:

A variety of sensors are used at the street layer for a variety of smart city use cases. Here is a short representative list:

- A magnetic sensor can detect a parking event by analyzing changes in the surrounding magnetic field when a heavy metal object, such as a car or a truck, comes close to it (or on top of it).
- A lighting controller can dim and brighten a light based on a combination of time-based and ambient conditions.
- Video cameras combined with video analytics can detect vehicles, faces, and traffic conditions for various traffic and security use cases.
- An air quality sensor can detect and measure gas and particulate matter concentrations to give a hyper-localized perspective on pollution in a given area.
- Device counters give an estimate of the number of devices in the area, which provides a rough idea of the number of vehicles moving or parked in a street or a public parking area, of pedestrians on a sidewalk, or even of birds in public parks or on public monuments—for cities where bird control has become an issue.

## City Layer:

**At the city layer, which is above the street layer, network routers and switches must be deployed to match the size of city data that needs to be transported.**

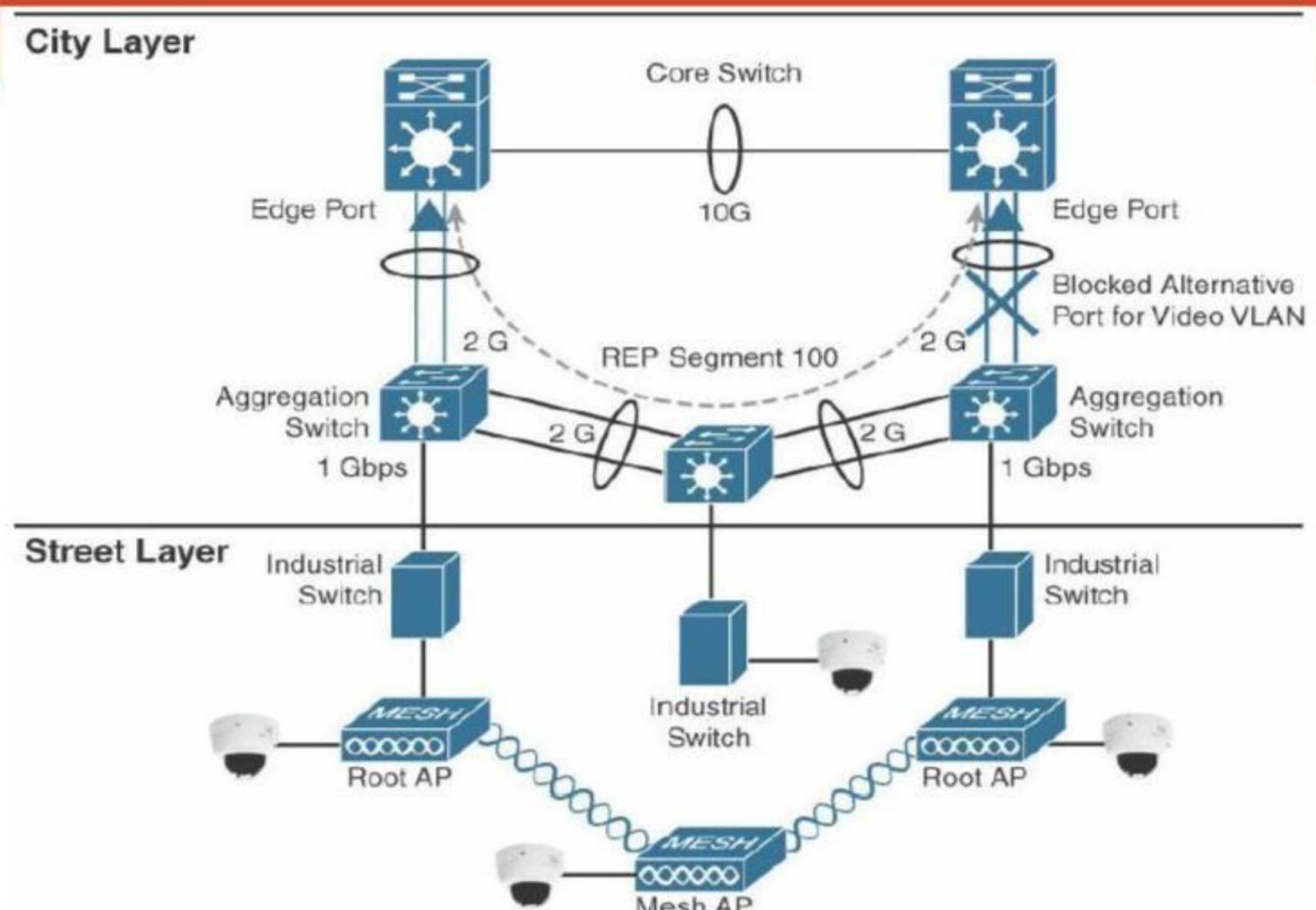
**This layer aggregates all data collected by sensors and the end-node network into a single transport network.**

**The city layer may appear to be a simple transport layer between the edge devices and the data center or the Internet.**

## City Layer:

**However, one key consideration of the city layer is that it needs to transport multiple types of protocols, for multiple types of IoT applications. Some applications are delay- and jitter sensitive, and some other applications require a deterministic approach to frame delivery.**

**A missed packet may generate an alarm or result in an invalid status report. As a result, the city layer must be built around resiliency, to ensure that a packet coming from a sensor or a gateway will always be forwarded successfully to the headend station.**




Street Layer Resiliency

## City Layer:

**In this model, at least two paths exist from any aggregation switch to the data center layer. A common protocol used to ensure this resiliency is Resilient Ethernet Protocol (REP).**

## Data Center Layer:

**Ultimately, data collected from the sensors is sent to a data center, where it can be processed and correlated.**

**Based on this processing of data, meaningful information and trends can be derived, and information can be provided back.**

**For example, an application in a data center can provide a global view of the city traffic and help authorities decide on the need for more or less common transport vehicles. At the same time, an automated response can be generated**

## Data Center Layer:

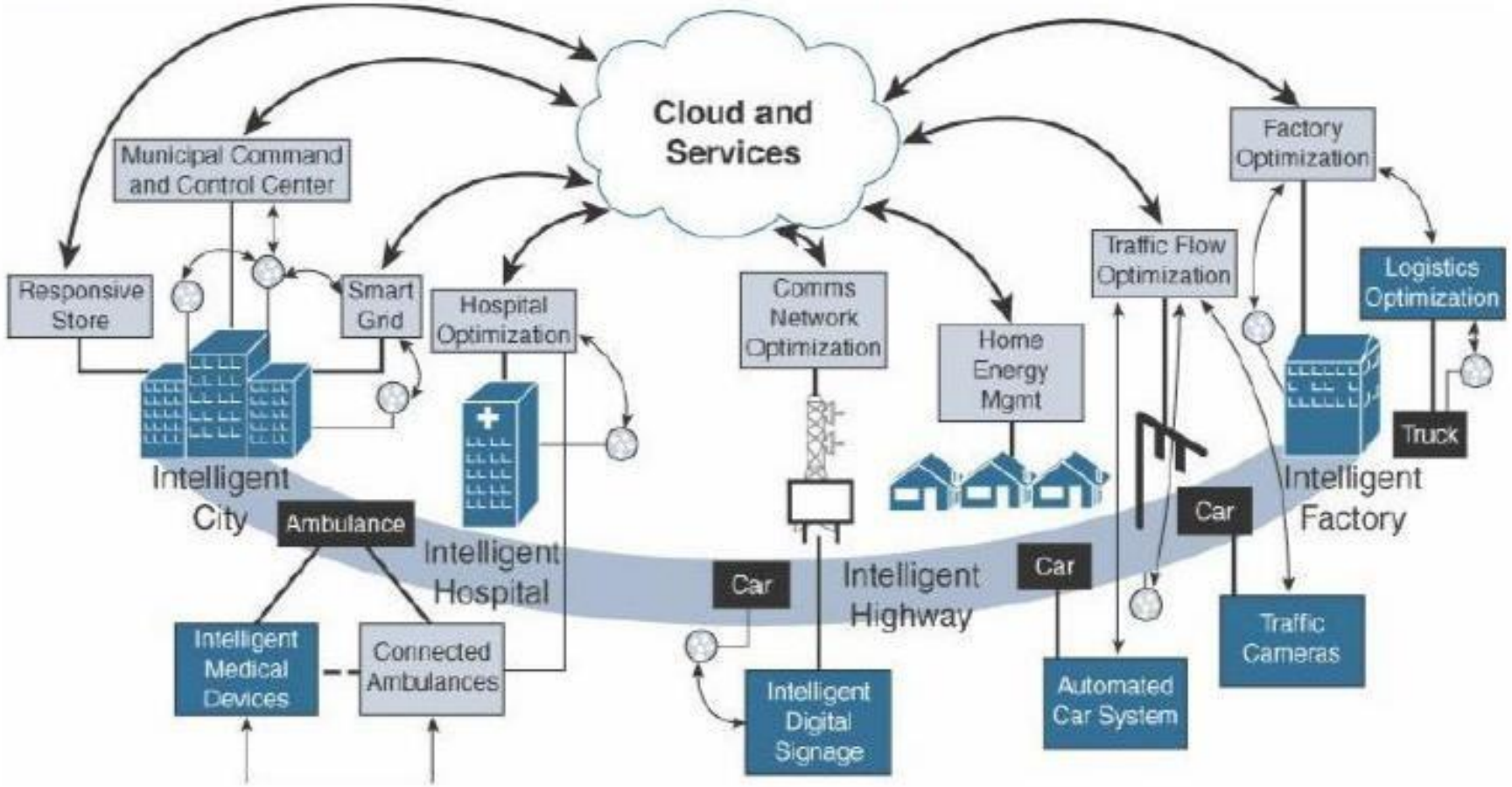
**The cloud model is the chief means of delivering storage, virtualization, adaptability, and the analytics know-how that city governments require for the technological mashup and synergy of information embodied in a smart city.**

**Traditional city networks simply cannot keep up with the real-time data needs of smart cities; they are encumbered by their physical limitations.**

**The cloud enables data analytics to be taken to server farms with large and extensible processing capabilities.**

## Data Center Layer:

Figure shows the vision of utilizing the cloud in smart solutions for cities. The cloud provides a scalable, secure, and reliable data processing engine that can handle the immense amount of data passing through it.



## Service Layer:

**Ultimately, the true value of ICT connectivity comes from the services that the measured data can provide to different users operating within a city.**

**Smart city applications can provide value to and visibility for a variety of user types, including city operators, citizens, and law enforcement.**

**The collected data should be visualized according to the specific needs of each consumer of that data and the particular user experience requirements and individual use cases.**

## Smart City Security Architecture:

**A serious concern of most smart cities and their citizens is data security.**

**Vast quantities of sensitive information are being shared at all times in a layered, real-time architecture, and cities have a duty to protect their citizens' data from unauthorized access, collection, and tampering.**

**In general, citizens feel better about data security when the city itself, and not a private entity, owns public or city-relevant data.**

**It is up to the city and the officials who run it to determine how to utilize this data.**

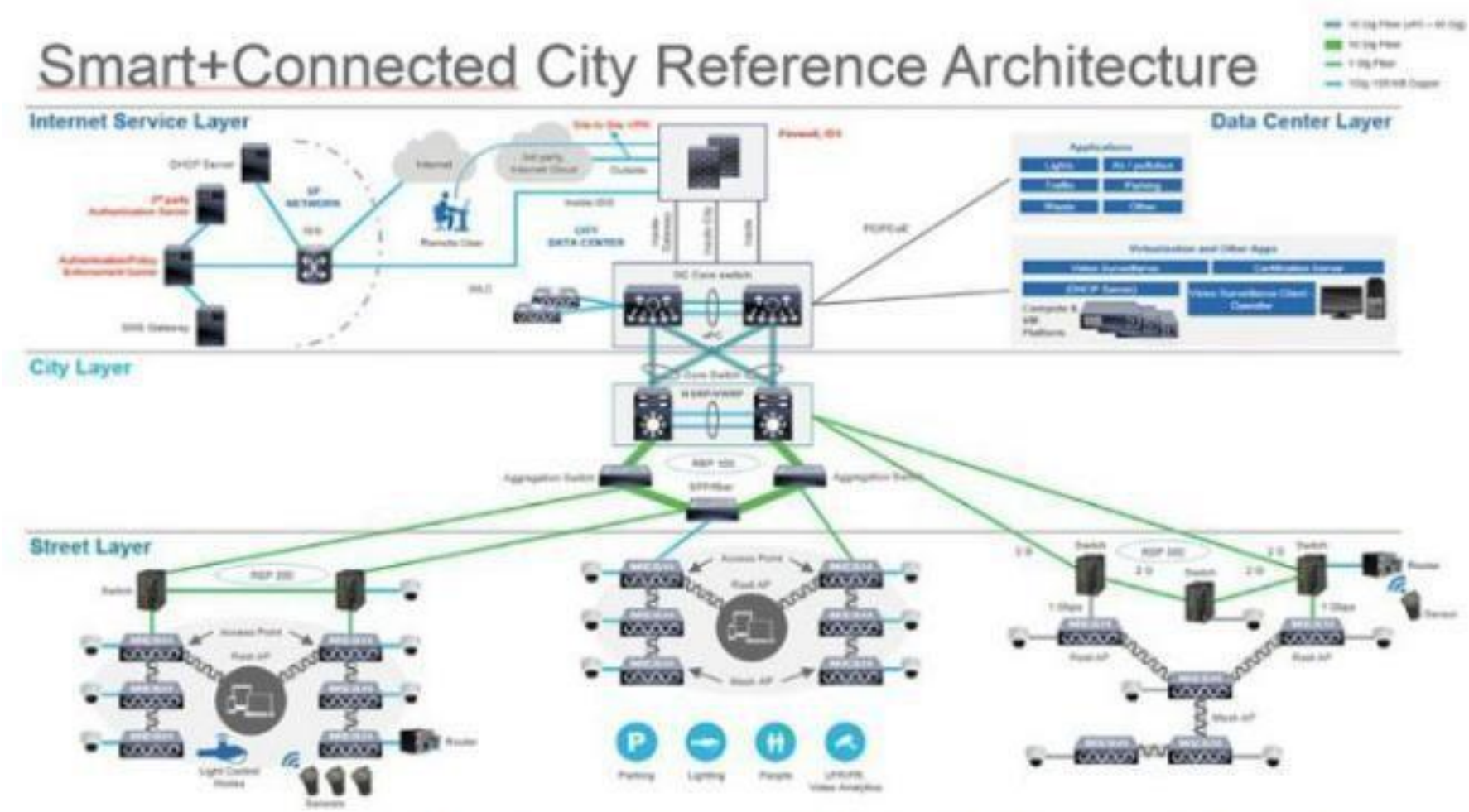
## Smart City Security Architecture:

**A security architecture for smart cities must utilize security protocols to fortify each layer of the architecture and protect city data.**

**Figure shows a reference architecture, with specific security elements highlighted.**

**Security protocols should authenticate the various components and protect data transport throughout.**

## Smart City Security Architecture:




Key Smart and Connected Cities Reference Architecture

## Smart City Security Architecture:

**Starting from the street level, sensors should have their own security protocols.**

**Some industry-standard security features include device/sensor identification and authorization; device/sensor data encryption; Trusted Platform Module, which enables self-destruction when the sensor is physically handled; and user ID authentication and authorization.**

## Smart City Security Architecture:

**Sensor identification and authorization typically requires a pre-installed factory X.509 certificate and public key infrastructure (PKI) at the organization level, where a new certificate is installed through a zero-touch deployment process.**

**This additional processing may slow the deployment but ensures the security of the exchanges.**

**Another consideration may be the type of data that the sensor is able to collect and process. For example, a roadside car counter may include a Bluetooth sensor that uniquely identifies each driver or pedestrian**

## Smart City Security Architecture:

The city layer transports data between the street layer and the data center layer. It acts as the network layer. The following are common industry elements for security on the network layer:

- **Firewall:** A firewall is located at the edge, and it should be IPsec- and VPN-ready, and include user- and role-based access control. It should also be integrated with the architecture to give city operators remote access to the city data center.
- **VLAN:** A VLAN provides end-to-end segmentation of data transmission, further protecting data from rogue intervention. Each service/domain has a dedicated VLAN for data transmission.
- **Encryption:** Protecting the traffic from the sensor to the application is a common requirement to avoid data tampering and eavesdropping. In most cases, encryption starts at the sensor level. In some cases, the sensor-to-gateway link uses one type of encryption, and the gateway-to-application connection uses another encryption (for example, a VPN).